LONDON METROPOLITAN UNIVERSITY

Islington college
(इस्लिङ्टन कलेज)

Module Code & Module Title

CU6051NI - Artificial Intelligence

Assessment Weightage & Type

75% Individual Coursework

Year and Semester

2023-24 Autumn

Student Name: Aayam Maharjan

London Met ID: 21049467

College ID: np01cp4a210117

Assignment Due Date: 17th January 2024

Assignment Submission Date: 17th January 2024

I confirm that I understand my coursework needs to be submitted online via Google Classroom under the relevant module page before the deadline in order for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a marks of zero will be awarded.

Acknowledgement

I would like to show our gratitude towards our module leader Mr. Sarun Dahal and tutor Mr, Rajesh Mahara for providing us with this wonderful opportunity to work on this project. This project was fruitful for us as we were able to understand Machine learning algorithm work and their positive impact in daily life.

Abstract

This project focuses on harnessing the power AI and NLP techniques to conduct sentiment analysis on Twitter dataset using the Sentiment140 dataset containing 1.6 million tweets. With help of supervised learning under Machine Learning, this project will utilize supervised learning algorithms like BernoulliNB, Logistic Regression and Linear SVC. the goal is to categorize tweets into positive or negative sentiments. The project involves a comprehensive workflow, including data cleaning, preprocessing, and model evaluation, aiming to provide high accuracy for business insights based on real-time public sentiment with proper results from the models before and after hyperparameter tuning.

Keywords: Artificial Intelligence, Machine Learning, Sentiment Analysis, Supervised Learning, Logistic Regression, Naïve Bayes, Support Vector Classification (SVC) Dataset, BernoulliNB, Logistic Regression, Linear SVC, hyperparameter.

Table of Contents

Table of Figures

Table of Tables

Table of Equations

1 Introduction

1.1 Artificial Intelligence

Intelligence can be defined as the ability to work and perform different activities while solving the problem to achieve goals. The capability of a machine to work based on its analysis of the surroundings with the ability to learn and perform different tasks while solving the problem that may have arisen in the process is known as AI (Artificial Intelligence). But with development in the IT field definition of AI has changed over time. The use of AI, the definition has also been changed overtime.

According to Stanford Professor John McCarthy in 1955, “AI is the science and engineering that makes machine intelligent” *(Manning, 2020).*

“Whereas in 2019, Kaplan and Haenlein define AI as a system’s ability to correctly interpret external data, to learn from such data, and to use those learnings to achieve specific goals and tasks through flexible adaptation” *(Barrneck, 2021).*

Figure 1: Artificial Intelligence

(EastVentures, 2023)

As AI is becoming more and more prevalent, AI has developed from self-driving cars to assistance like Google Voice Assistance and Siri to bots like ChatGPT that can process data and give output according to the input provided. With growth in AI in this extinct, people have understood the power and potential of AI. Artificial intelligence seems to be the one providing different industries and people with a lot of benefits, but it doesn't mean, one can develop an AI without following any precautions or principles.

One must follow these principles to develop safe, ethical, and trustworthy AI, that one can use without any risk of losing one's identity and privacy. Some of the Principles are listed down below:

1.1.1 Principles of Artificial Intelligence

a. Fairness and bias

   This principle is based on how the AI should not harm people through inequitable treatment. The AI developed should dictate that AI must treat every individual equally without favouring against other on any characteristic such as race, gender, etc. For this developer must rectify biases with training models, data and output within the AI *(Bellini, 2022).*

b. Privacy and Security

   With growth of Artificial Intelligence, it's almost everywhere like snapchat, google, etc. People tend to ask different question to the AI that might be personal. An AI should be designed in a reliable, secure and with aligned standards so that the questions asked to AI, things done with help of AI will not be disclosed to anyone except the user (Bellini, 2022).

c. Accountability

   Accountability is one of the fundamental principle of the Artificial Intelligence. This principle emphasizes the responsibility of the developers, organization for ethical, legal, and social implication of the AI. This principle ensures that AI system designed and developed, is aligned with ethical and social standards in a society *(Bellini, 2022).*

d. Social Benefit

This principle refers to the positive impact of AI that can have on a society. Here the AI application should be able to contribute to overall well-being of the community. The AI should help in advancements in healthcare, education or any other sectors *(Bellini, 2022).*

e. Trust and Transparency

The trust in AI generally means the establishment of confidence among the users, that the provided information by the AI is reliable, secure and is within the ethical standards. While transparency emphasizes openness in how AI system can operate and make decisions *(Bellini, 2022).*

1.2 Machine Learning

Often many people tend to confuse Machine Learning and Artificial Intelligence. Machine learning is a branch of AI that allows machines to learn how to learn data efficiently and improve performance without being specifically programmed. Machine Learning (ML) is generally programmed with the help of numerous algorithms, that allow machines to operate a large number of data efficiently allowing them to compare data, look for patterns, and find a better and efficient way to make accurate predictions with minimal human intervention *(Mahesh, 2020).*

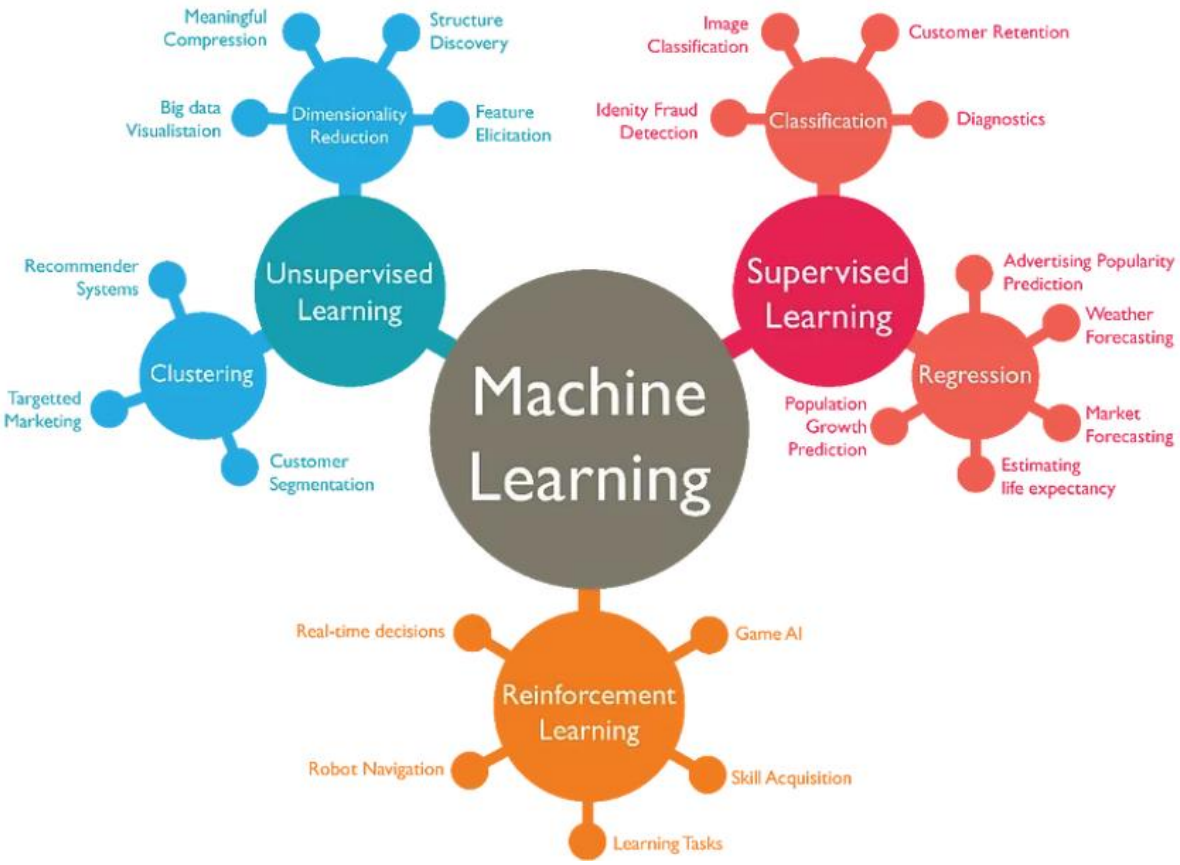


Figure 2: Machine Learning

(Lorberfeld, 2019)

1.2.1 Principles of Machine Learning

a. Accuracy and Precision

The ML models should provide the highest accuracy with utmost precision in their prediction and classifications. The ML models should minimize the errors ensuring the truth and desired outcomes *(Belle, 2021).*

b. Fairness and Bias Mitigation

ML models should give proper results with fairness to avoid any bias with help of the algorithms and trained datasets. ML models should be trained with data and algorithm to go against unfair treatments *(Belle, 2021).*

c. Interpretability and Explainability

ML models should be transparent and understandable. This allows users to interpret the decision made by the Machine. This principle helps in enhancing the trust accountability among the users *(Belle, 2021).*

d. Data Privacy and Security

Developers should prioritize the privacy of sensitive data and information that can be used in the training of the ML models. This allows the ML model to enforce secure data, with privacy regulations *(Belle, 2021).*

e. Robustness and Generalization

ML models should perform well on diverse datasets, with real world data and scenarios, demonstrating robustness against different patterns. The ML models should be generalized way beyond the training data to make them reliable and effective *(Belle, 2021).*

The machine can learn about different data with the help of different datasets which can be done through different types of machine learning i.e., supervised learning or unsupervised learning.

1.2.2 Supervised Learning

Supervised learning can be defined as the process where machines are trained under the supervision of humans with the help of training samples and labelled datasets. Generally, in supervised learning datasets are used to make predictions to learn the mapping between input and output. Datasets usually have the correct answer, but Supervised learning allows us to find different patterns to find out the classification or regression of input data *(Liu, 2015).*

a. Classification

Classification can be defined as a process of grouping data into different categories or classes based on some criteria. These classes are also known as labels and can be anything such as gender, location, true or false, positive, or negative, etc. There are two types of classification:

- Binary Classification

This is a simple type of classification. Here the data is separated into two categories for example the classification of email i.e., spam or not spam *(avcontentteam, 2023).*

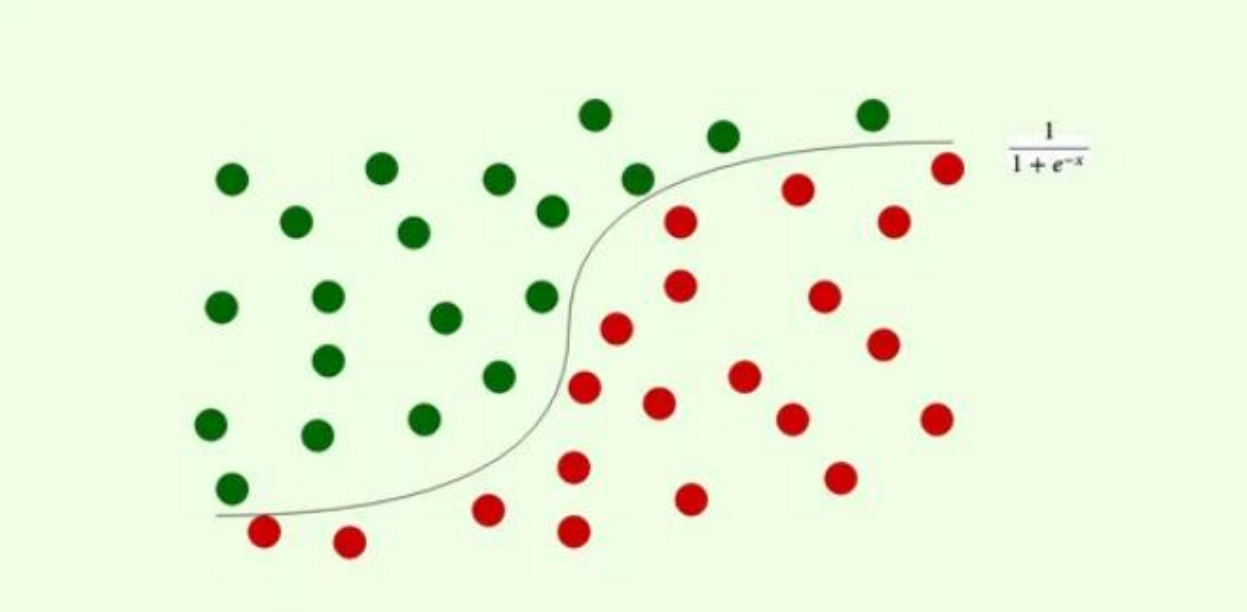


Figure 3: Binary Classification

(avcontentteam, 2023)

- Multiclass classification

  This is called multiclass classification because the data can be classified into two or more categories. For example, classifying different fruit types such as mango, banana, apple, etc.

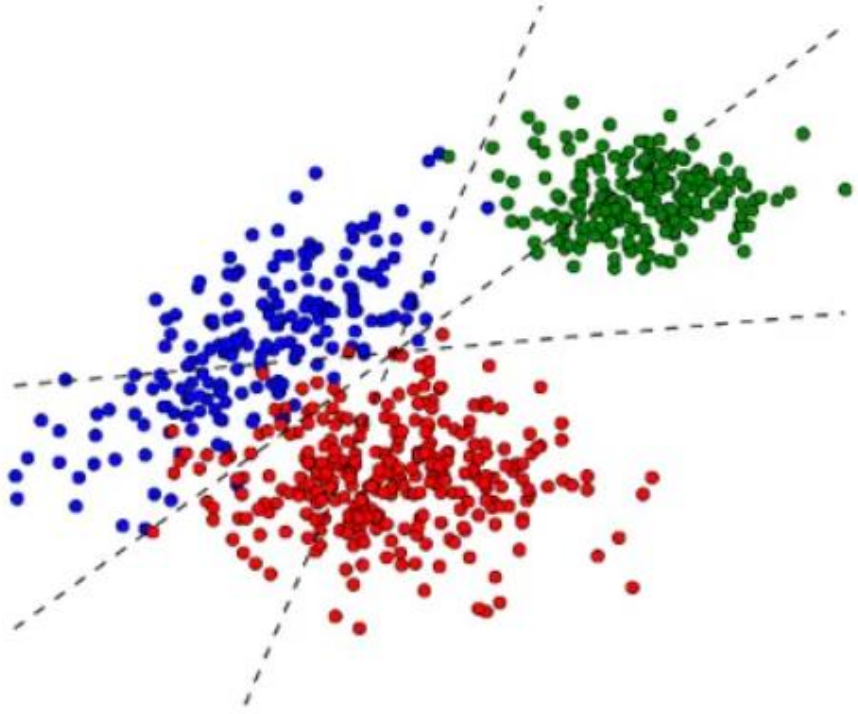


Figure 4: Multiclass classification

(avcontentteam, 2023)

Whereas classification can be done through different algorithms such as KNN, Naïve Bayes, Decision tree, Logistic Regression, and many more *(avcontentteam, 2023)*.

b. Regression

Regression analysis is a simple and statistical method of understanding two or more variables to find a certain pattern in the data. Here the variables help to determine how one variable is associated with another, which allows us to make predictions and impact of factors responsible for the outcome. The simple linear regression formula is represented by Y = a + bX *(Simplilearn, 2023).*

1.2.3 Unsupervised learning

Unsupervised learning can be defined as the process where the computer learns independently with minimal human intervention mostly using unlabelled data. Here we aim to train a model, learn features generalization, and classify data samples. Here our task is to find patterns or relationships between variables *(Wu, 2022).*

a. Clustering

Just like classification, clustering is also a technique to group objects with similar attributes into clusters. However, unlike classification, the clusters are predefined, and grouping is done with the help of mapping similarities and common attributes in real time *(avcontentteam, 2023).*

b. Association

The association can be defined as the process of determining and finding relationships among data. The association is generally done in Market Analysis where we identify different products based on an association i.e., when the items are purchased, what products are purchased together *(GeeksforGeeks, 2023).*

1.3 Natural Language Processing

Natural Language Processing (NLP) is the analysis of linguistic data such as documents, using computing methods. NLP is generally used to assist computers in understanding human language. It acts as a bridge of communication between machines and humans that allows humans to understand machine language. Because of NLP, humans can use, interact, and understand Machine learning tools like auto-correction *(Sodhar, 2022)*.

Even popular tools AI tools like ChatGPT use NLP and machine learning algorithms to carry out different tasks and interact with humans to provide meaningful information. Hence, Artificial Intelligence, Machine Learning, and Natural Language Processing are interrelated with each other *(Sodhar, 2022).*

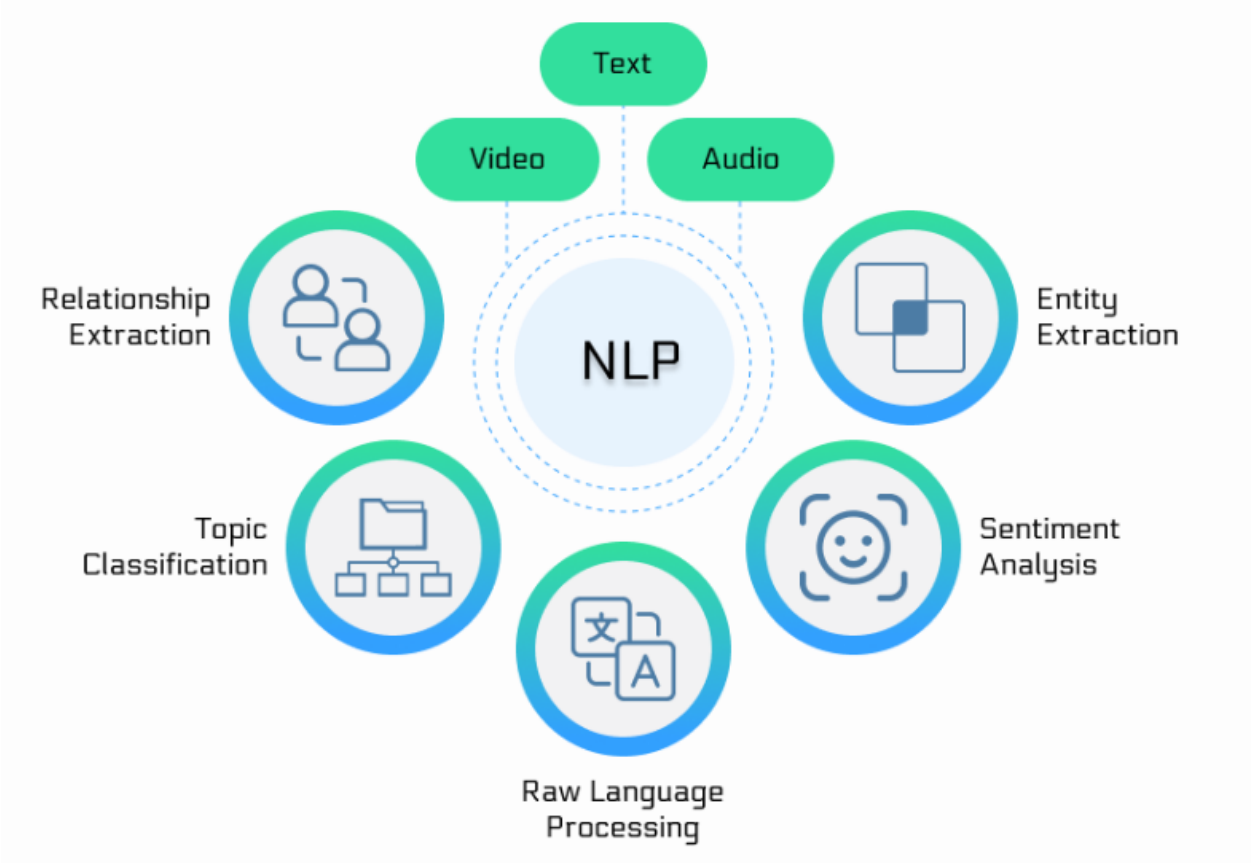


Figure 5: Natural Language Processing

(AMAZINUM, 2023)

1.4 Problem Domain

In this new generation, the world is transforming into a digital platform. The Internet serves as the main factor for this growth because almost all the people in the world are connected through social media platforms such as Facebook, Twitter, etc. These platforms have made it easy to share one's feelings, opinions, and thoughts freely. Among these platforms, Twitter is one of the most popular platforms with an average of 326 million active users who tweet almost 500 million tweets per day and around 200 billion tweets per year. These tweets can be about almost any event such as movies, politicians, governmental organizations, and many more. The tweets posted are often treated as valuable feedback for almost all the industry as positive feedback can help one to draw a large audience and vice versa *(Pfeffer, 2023).*

Figure 6: Tweets from twitter comparison in years

(InternetLiveStatus, 2014)

Based on different tweets, we can classify the tweets as positive or negative also known as sentiment analysis. Different tweets from the users can be classified as positive or negative. These tweets related to different trends, political opinions, trends in stock markets, and sales of products, when analyzed properly and concluded, we could find if the people had positive feedback of the new trends, political changes, etc, or had more negative tweets. For example, if there is more negative feedback for the product in the tweets from the public and the company doesn't know about it, they could suffer from great loss and vice versa. Whereas with the help of sentiment analysis, they can find the public demand and make changes as needed or demanded. In this competitive globe, any firm would want to know how their goods and services are performing in the market, not using Sentimental analysis would be a great downside for any firm.

With increase reliance on social media platform, communication and expression on different topics have risen a vast number of tweets and posts. Understanding and extracting meaningful insights from this enormous volume of textual data pose big challenge.

Sentiment analysis, in this context, becomes necessary for every businesses, organizations, and even individuals. By leveraging sentiment analysis, companies can not only evaluate the public's response to their products or services but also gain valuable insights into the prevailing sentiment towards broader topics such as political events, social movements, and cultural phenomena. This information is helpful in shaping marketing strategies, improving customer experiences, and staying ahead in an ever-evolving market landscape. As social media like Twitter plays a vital role in shaping public opinion, sentiment analysis emerges as a crucial tool for navigating the dynamic and competitive digital landscape *(Robinson, 2021).*

Hence, this project will help in analyzing the different tweets and finding out the sentiment of the tweet i.e., positive, or negative tweets.

1.4.1 Some facts and figures

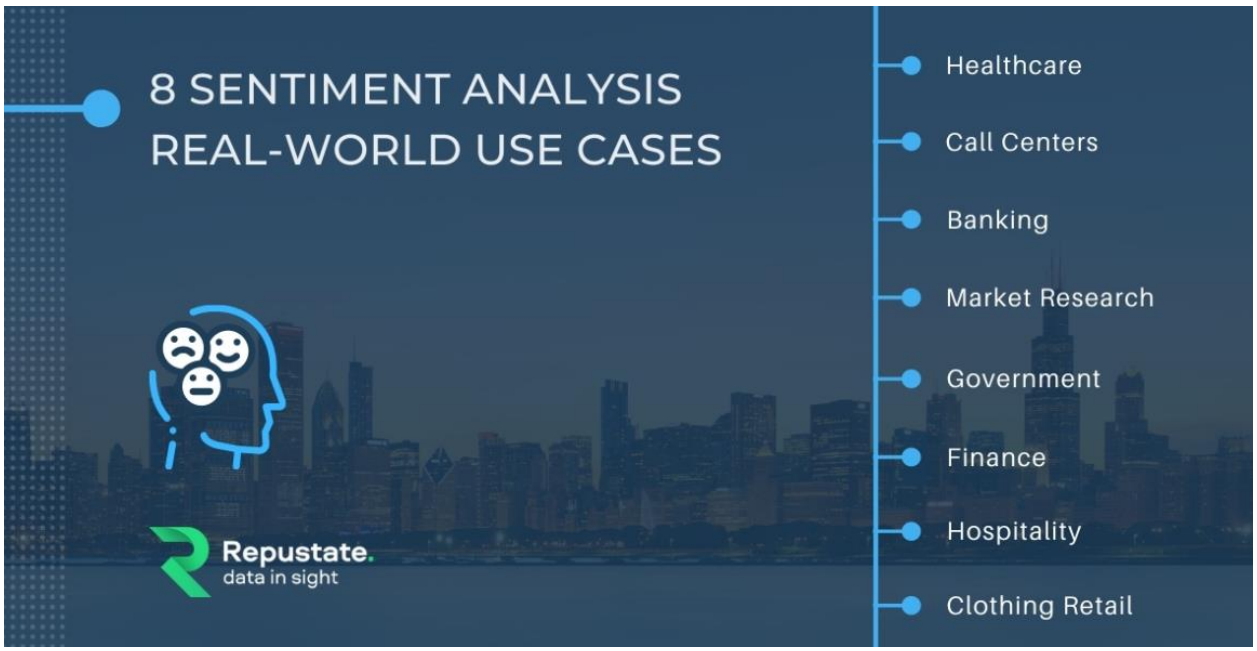


Figure 7: Sentimental Analysis applications

(Alnajem, 2021)

a. The global sentiment analytics market is estimated to reach $7.5 billion by 2028, growing at the rate of 18.4% per year.
b. As of 2022, 52% of large companies have been using sentiment analysis software, whereas small business adoption stands at 12%
c. Sentiment analysis has an error rate of up to 30% which can be further minimized with the help of further analysis.
d. Industries have faced a higher rate of success (91%) using sentiment analysis as they adjust the product customers use after analyzing the sentiments of people before launching (Brito, 2023).

2 Background

2.1 Research on chosen problem domain: Sentimental Analysis

Sentimental analysis is also known as opinion or feelings mining. Sentimental analysis is used to classify the subjective opinions in a source material (data) into mainly two outcomes or sentiments (positive or negative) which refers to the application of Natural Language Processing (NLP). However, the outcome can also be classified further such as neutral, happy, ecstatic, and many more. In the process of sentiment identification, the main task is to identify the opinion words, generally, these words are adjectives, adverbs, and verbs *(Luo, 2013).*

Sentimental analysis is an automated machine learning process done with the help of Realtime data, enabling insights into data-driven decisions. Analysis of the Realtime data is usually done with the help of various surveys, blogs, responses, comments, or even tweets from social media. The analysis of these data usually has 3 major outcomes i.e., positive, negative, and neutral *(Zulfadzli Drus, 2019).*

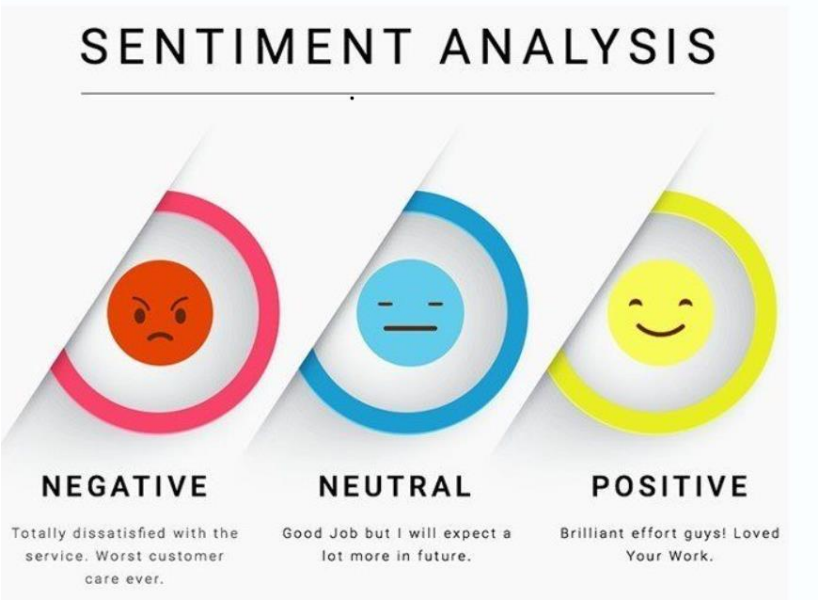


Figure 8: Sentimental Analysis

(Zulfadzli Drus, 2019)

Sentimental analysis of any data contains the overview opinion of the public, which is generally termed as a powerful opinion. The output of the sentiment especially gained from social media such as Twitter has been correlated with numerous factors in different industries. From the shift in the stock market to campaigns in political affairs, industries, and governmental bodies can plan to strategize and plan to get a lead in this competitive industry. Apart from the stock market and politics, commercial businesses have also started sentimental analysis in a more accurate way to monitor brands, customer demands, and services. The top 10% of businesses have integrated sentiment analysis as a part of a team to gain insights into customer opinions and correlate the changes to the products according to the opinion of the public *(Bannister, 2018).*

Sentiment analysis has started to become a vital part of all industries. As sentiment analysis can help them gain better insights and perspectives from the public making the product, they make it more efficient and better.

Some of the reasons Sentimental analysis is necessary in today digital age are:

a. Customer Feedback

   Businesses can track and examine client opinion on social media and review platforms with help of sentiment analysis. This aids in maintaining and improving their internet reputation by quickly resolving problems and realizing client sentiment.

b. Product Improvement

   Through sentiment analysis found in reviews and comments, businesses can learn what aspects of their products and services users find appealing and hate. This data is essential for meeting consumer expectations and making improvements.

c. Market Research and Competitor Analysis

   Sentiment analysis offers useful information for market research, assisting businesses in comprehending the attitudes, trends, and preferences of their target audience. Which in return also helps with market opportunities and competitive analysis among competitors.

d. Brand Monitoring

   To evaluate brand perception, one must keep an eye on the sentiment surrounding a brand. While negative sentiment might point to prospective problems or places for improvement, positive sentiment can be used to market a product.

e. Political and Social Analysis

   Sentiment analysis is a tool used in social science research and political campaigns to gauge public sentiment towards laws, policies, and social concerns. It also assists in forming plans and regulations according to popular opinion *(Fairlie, 2023).*

2.1.1 ML/AI in sentimental analysis

Sentimental analysis is an ML-model approach that helps to gain valuable insights into people's emotions i.e., positive, or negative. In this competitive world, any firm would want to know how their goods and services are performing in the market. By understanding the demands of the public through different opinions such as tweets from Twitter, one can know the demanded and denied products. Because of this reason, Sentiment Analysis has been an important factor to reckon with in this era.

The use of sentimental analysis is growing day by day, as 52% of large companies have been using sentiment analysis software, whereas small business adoption stands at 12%. Whereas it is estimated that the use of the model of Sentimental Analysis is growing at a rate of 18.4% each year. AI and ML have made it easy for different forms, to gather data from different social media platforms, and articles and process it with the help of different algorithms such as Naïve Bayes, Random Forest, Logistic Regression, and many more *(Brito, 2023).*

The use of Sentimental analysis has been quite a boon to the market, as more than 91% of businesses have found higher success rates after the use of Sentimental analysis. As they can get an overview of the problem and mitigate it to provide a greater product. Thanks to the development of AI and ML, understanding human emotions has been possible through machines *(Taherdoost, 2023).*

2.1.2 Advantages of Sentimental Analysis

There are several advantages of using sentiment analysis in almost every industry. Some of the advantages are:

a. Analysis of sentiment can help companies understand the customer sentiment toward the product enabling them to make data-driven changes.
b. With the help of sentiment analysis through different social media platforms such as Twitter, the company can identify problems early and resolve them quickly.
c. As with greater competition in the industry, sentiment analysis can gain better insights and be a step forward from the competitors.
d. With the help of sentiment analysis, we can get to know the complaints from the customers and take necessary steps.
e. Alongside identifying different issues early, one can allocate resources easily, helping in working efficiently in a cost-effective manner *(Kim Schouten, 2015).*

2.1.3 Disadvantages/Challenges for Sentimental Analysis

Sentimental Analysis is the process of emotion mining of human beings. Even humans have a hard time understanding each other, hence identifying the emotions of humans with the help of Machine learning is a challenging task. Some of the challenges of sentiment analysis using ML are:

a. Words such as love, and hate are easy to understand in terms of positive (-1) and negative (-1) sentiment polarity but words like “not so bad” are sometimes misunderstood giving it a score of -75 in polarity of sentiment score.
b. People often use slang and sarcasm to express their emotions, which can be difficult to understand by machine as positive or negative.
c. With the rise of Gen-Z, the use of emoji is quite popular, The “ 😭 ” emoji is often treated as a laughing emotion, whereas the machine would interpret this as a negative emotion crying.
d. Negotiation terms such as not, never, etc can confuse the ML sub-model as the phrase “I can’t go to my class reunion” means the person intends to go but the ML model would treat it as negative *(Repustate, 2021).*

2.1.4 Application for Sentimental Analysis

a. Retail and E-commerce

Retailers examine customer comments and product reviews using sentiment analysis. Negative feelings point out areas where a product needs to be improved, while positive attitudes can be leveraged for marketing.

b. Finance

Sentiment analysis is used in the financial sector to assess market sentiment in news stories, social media posts, and financial reports. It helps investors and traders make wise choices.

c. Healthcare

Sentiment analysis is used in the medical field to examine opinions shared on social media, patient feedback, and evaluations of medical services. It enhances patient experiences and points out areas that could be better.

d. Travel and Tourism

In the travel sector, sentiment analysis is used to examine opinions expressed in social media posts and reviews. It aids businesses in comprehending customer preferences and enhancing their offerings.

e. Education

Sentiment analysis is a tool used by educational institutions to examine social media conversations about education, course assessments, and student feedback. This knowledge aids in the development of instructional strategies and curricula *(Sharma, 2023).*

2.1.5 Chosen Dataset

The dataset for the coursework “Sentiment Analysis” was searched in Kaggle, Open Data, and many more. However, the data chosen was found on Kaggle. The dataset is called with the name: “Sentiment140 dataset with 1.6 million tweets” which was extracted from, Twitter API and created by “ΜΑΡΙΟΣ ΜΙΧΑΗΛΙΔΗΣ KAZANOVA”. It consists of sentiment data extracted from the Twitter API containing the field mentioned below:

Table Name	Description
Target	The polarity of the tweet (0 = negative, 2 = neutral, 4 = positive)
IDs	The id of the tweet.
Date	The date of the tweet.
Flag	The query *(lxy).* If there is no query then the value is NO_QUERY.
User	The user that tweeted the tweet.
tweet	The text of the tweet.

Table 1: Dataset description

	0	1	2	3	4	5
0	0	1467810369	Mon Apr 06 22:19:45 PDT 2009	NO_QUERY	_TheSpecialOne_	@switchfoot http://twitpic.com/2y1zl - Awww, t...
1	0	1467810672	Mon Apr 06 22:19:49 PDT 2009	NO_QUERY	scotthamilton	is upset that he can't update his Facebook by ...
2	0	1467810917	Mon Apr 06 22:19:53 PDT 2009	NO_QUERY	mattycus	@Kenichan I dived many times for the ball. Man...
3	0	1467811184	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	ElleCTF	my whole body feels itchy and like its on fire
4	0	1467811193	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	Karoli	@nationwideclass no, it's not behaving at all....

Figure 9: Dataset Overview

Link to dataset: https://www.kaggle.com/datasets/kazanova/sentiment140 (KAZANOVA, 2017)

2.2 Review and analysis of existing work in the problem domain

2.2.1 Twitter Sentiment Analysis Using Supervised Machine Learning

Authors: Nikhil Yadav, Omkar Kudale, Aditi Rao, Srishti Gupta.

Issued date: 1st December 2020

This research paper was done for the sentimental analysis of the Twitter Dataset named “140dataset with 1.6 million sentiments”. This report states it used the concepts of Stop words like “I like to write” to be termed as “I” and “to” and Steaming to convert words like “chocolatey” to “chocolate” to get better accuracy from the data. After cleaning the data, the report states that they used models like Logistic Regression, and Linear SVC to perform Sentimental Analysis. Where outcome from Linear SVC had the highest accuracy of 82.47% while logistic regression had an accuracy rate of 83.71% (Nikhil Yadav, 2020).

2.2.2 Solving the Twitter sentiment analysis problem based on a machine learning approach.

Authors: Fatemeh Zarisfi Kermani, Faramarz Sadeghi and Esfandiar Eslami

Issued date: 9th January 2019

This research paper was done following the Lexicon approach by following the polarity of tweets on a scale of 0 to 4. The sentiment scoring module based on the SWN lexicon was used by converting the tweets into a bag of words for positive and negative tweets. The data was extracted from 4 different datasets “Four datasets (Stanford Twitter Sentiment (STS), STS-Gold, original Obama-McCain Debate (OMD), and Strict Obama-McCain Debate (SOMD)” containing over 1 million tweets, which were then trained and used by using models like Vector Machines (SVM), Multinomial Naïve Bayes (MNB), and Bernoulli Naïve Bayes (BNB). After using these all models, they were able to achieve a combined accuracy rate of 85.92% with combined models (Fatemeh Zarisfi Kermani, 2019).

2.2.3 Sentiment Analysis of Twitter Data: A Survey of Techniques

Authors: Vishal A. Kharde and S.S. Sonawane

Issued Date: 11th April 2016

The authors researched and gathered data from the various datasets some of them being from “HASH containing 31,861 Pos tweets 64,850 Neg tweets”, “EMOT containing 230,811 Pos& 150,570 Neg tweets” and “Stanford 4 million tweets categorized as positive and negative”. These data were cleaned by removing stop words and removing non-English tweets. The cleaned data were processed through models like Naïve Bayes and Support Vector Machines (SVM) to gain an overview of the people’s sentiment. Where Naïve Bayes had an accuracy rate of 84.23% whereas Support Vector Machine (SVM) had an accuracy rate of 82.52% *(Vishal A. Kharde, 2016).*

2.2.4 Sentiment Analysis of Twitter Data

Authors: Apoorv Agarwal, Boyi Xie, Ilia Vovsha, Owen Rambow, and Rebecca Passonneau

Issued Date: 23rd June 2011

This research paper was done to calculate the sentimental Analysis data for the tweets of Twitter which were collected manually through surveys (11875 tweets), and tweets from the commercial source which were collected by archiving real-time data. Where each tweet was labeled by a “human annotator” as positive, negative, neutral, or junk. These data were then processed using the model “Tree kernel model”, where a new tree representation for tweets was designed. When further analyzed and concluded gave an accuracy of 73.93% for the sentiment analyzed *(Apoorv Agarwal, 2011).*

2.2.5 Sentiment Analysis in Twitter Data

Authors: K.L. Sudha and Prerana B Patil

Issued Date: 7th July 2022

The research paper was done to find the sentimental analysis to collect the views on products, trends, and politics. Where 3000 tweets were extracted from the dataset “Ola Datasets and Twitter API”. The data was then cleaned and tokenized using Python which were then classified and processed using 3 different models such as logistic regression, Multinominal Bayes, and Random forest to find positive and negative sentiments. The accuracy generated from logistic regression and Multinominal Bayes was quite similar achieving 82.2% accuracy whereas as Random forest accuracy rate was highest at 84.32% *(K.L. Sudha, 2022).*

2.2.6 Summarization of Analysis

These studies on research paper consist of the use of different datasets and ways of extracting and cleaning them such as the use of tokenization, and the use of a bag of words to human annotated cleaning methods. Alongside the machine learning approach for calculating accuracy, a different approach like Lexicon was also used.

The research paper included the use of many models to calculate sentiment analysis. Whereas older version models like the tree kernel models gave the lowest accuracy rate of 73.39%. Whereas many of them also used modernized algorithms like Random Forest, Logistic Regression, and Naïve Bayes. These modern algorithms proved to be more efficient and more accurate as even when working with millions of data it had an accuracy rate of 82-85%. Alongside these, a useful finding was the exclusion of neutral sentiment in most of the research papers.

3 Solution

3.1 Proposed Approach to solving the problem.

From the research done with the help of different research papers, the supervised learning approach for solving the problem seems to be the best. Sentimental analysis is the process of understanding the emotions of people through text datasets. In my case dataset of Twitter “Sentiment140 dataset with 1.6 million tweets” is being used for gathering tweets of people. For algorithms, 3 different models have been chosen (Naïve Bayes Classifier, Logistic Regression, and Linear Support Vector Classification (SVC)) to calculate the dataset with high accuracy. Later on, the accuracy calculated from these 3 models will be compared to know which algorithm performed the best. Also, libraries like pandas, NumPy, and matplotlib will be used to process data and visualize it.

3.2 Explanation of the Algorithm Used.

3.2.1 Naïve Bayes Classifier

With changes in the AI and ML field and the introduction of many new algorithms, Naïve Bayes has still been proven to be quick and accurate. Also, it is one of the simple models hence called Naïve.

Naïve Bayes is a classification technique that utilizes the Bayes theorem. Bayes's theorem says that the predicted probabilities of an event are by incorporating new information. Hence Naïve Bayes uses a probabilistic approach that is denoted as prior probability and posterior probability. Which says that the probability of an event is based on the initial probability of an event with the highest probability. One of the assumptions made for Naïve Bayes is the datasets are independent of each other and give the same importance to the outcome of the data. The main approaches for the Naïve Bayes algorithm include spam filtrations, Sentimental Analysis, and many more *(Webb, 2016).*

Naïve Bayes works to find probability which can be found out by the following equation:

$(Y|X) = \frac{P(X|Y)* P(Y)}{P(X)}$

Equation 1: Probability

Here,

$P(Y|X)$ = Posterior probability –> the probability of Y occurring given X,

$P(X|Y)$ = Likelihood –> the probability of X occurring given Y,

$P(Y)$ = class prior probability –> the probability of Y occurring independently,

$P(X)$ = predictor prior probability –> the probability of X occurring independently.

$P(X)$ = predictor prior probability –> the probability of X occurring independently.

the independence of the classifier's features, a1, a2,... The probability of each trait is multiplied to get ->$p(a_1, a_2, \dots, a_n|y) = p(a_1|y) * p(a_2|y) * \dots p(a_n|y)$.

Which is also expressed as:

$$p(a_1, a_2, \dots, a_n|y) = \prod_{j=1}^{n} p(a_j|y)$$

Equation 2: Probability of each trait

Finally, Naïve Bayes =

$$y_{new} = \ argmax_{y \in y}\, p(y) \prod_{j=1}^{n} p(a_j|y)$$

Equation 3: Naive Bayes

(GeeksforGeeks, 2023)

3.2.2 Logistic Regression

Logistic Regression is one of the supervised learning algorithms that is used to predict the probability of binary outcomes and hyperplane with the use of logistic functions with a probabilistic approach. In logistic regression, the outcome is generally classified in binary i.e., either 0 or 1. It is easy to implement interpret and train than other Machine Learning models. Logistic regression helps in measuring how appropriate an independent variable is (coefficient size) while revealing the direction of the relationship *(Kanade, 2022).*

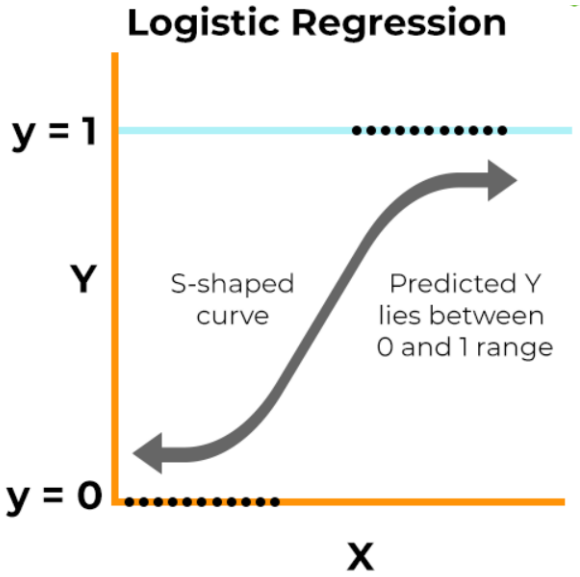


Figure 10: Logistic Regression

(Kanade, 2022)

The equation of logistic regression is written as:

$$f(x) = \frac{1}{1 + e^{-x}}$$

Equation 4: Logistic Regression

Here,

e = base of natural logarithms

values = numerical value one wishes to transform

Logistic regression uses a logistic function called the sigmoid function for map predictions and their probabilities. The sigmoid function is an S-shape curve that converts real value to 0 or 1. For example, if the output of the sigmoid function is 0.65, it implies that there are 65% chance of the event occurring *(Kanade, 2022).*

The function is referred to as the activation function for logistic regression. It is defined as:

$$y = \frac{e^{(b_0 + b_1 X)}}{1 + e^{(b_0 + b_1 X)}}$$

Equation 5: Sigmoid Function

Here,

x = input value

y = predicted output

b0 = bias or intercept term

b1 = coefficient for input (x)

3.2.3 Linear Support Vector Classification (Linear SVC)

SVM is a powerful supervised algorithm that is used for calculating complex datasets. SVM is an ML problem that tries to find the hyperplane unlike Logistic regression SVM is based on a statistical approach. Support Vector Machine (SVM) can be used for both regression and classification but generally is used for classification problems.

Under SVM, we can use a classifier called SVM Classifier, here a dataset is classified into two major classes by using a single straight line, which is also known as linearly separable data. Here the main objective is to find the optimal Hyperplane in an N-dimensional space. Where the hyperplane tries the margin between the closest points of different classes should be maximum.

In the Binary classification of Linear SVC, two classes are labeled as +1 and -1 where we have to input the dataset accordingly. The equation of linear hyperplane can be written as:

$$w * x + b = 0$$

Equation 6: Linear hyperplane

Here,

w = normal vector to hyperplane

b = distance from the hyperplane

Where the distance between datapoint x_i and the decision boundary can be calculated through the:

$$d_i = \frac{w^T x_i + b}{||w||}$$

Equation 7: Distance between datapoints

Here,

||w|| = Euclidean norm of the weight vector w

For Linear SVM classifier:

$$\hat{y} = \begin{cases} 1 : w^T x + b \geq 0 \\ 0 : w^T x + b \leq 0 \end{cases}$$

Equation 8: Linear SVM

3.3 Pseudocode

START

IMPORT libraries

IMPORT Dataset

ANALYZE, CLEAN Dataset

IF unnecessary columns exist

REMOVE unnecessary columns.

END IF

IF duplicate data exist

REMOVE duplicate data.

END IF

CONVERT Dataset labels to sentiment (0 Negative) and 1(Positive)

PRE-PROCESS Datasets using libraries like (WordNetLemmatizer, re)

GENERATE word clouds.

SPLIT dataset into train and test data (

VECTORIZE dataset

CREATE Model for naïve Bayes, logistic regression, and Linear SVC

INSERT and TRAIN the trained data into all models.

TEST the model with test data.

CALCULATE accuracy from the models.

VIEW accuracy from the models

HYPERTUNE 3 different models naïve Bayes, logistic regression, and Linear SVC

COMPARE accuracy from the models

END

3.4 Flowchart

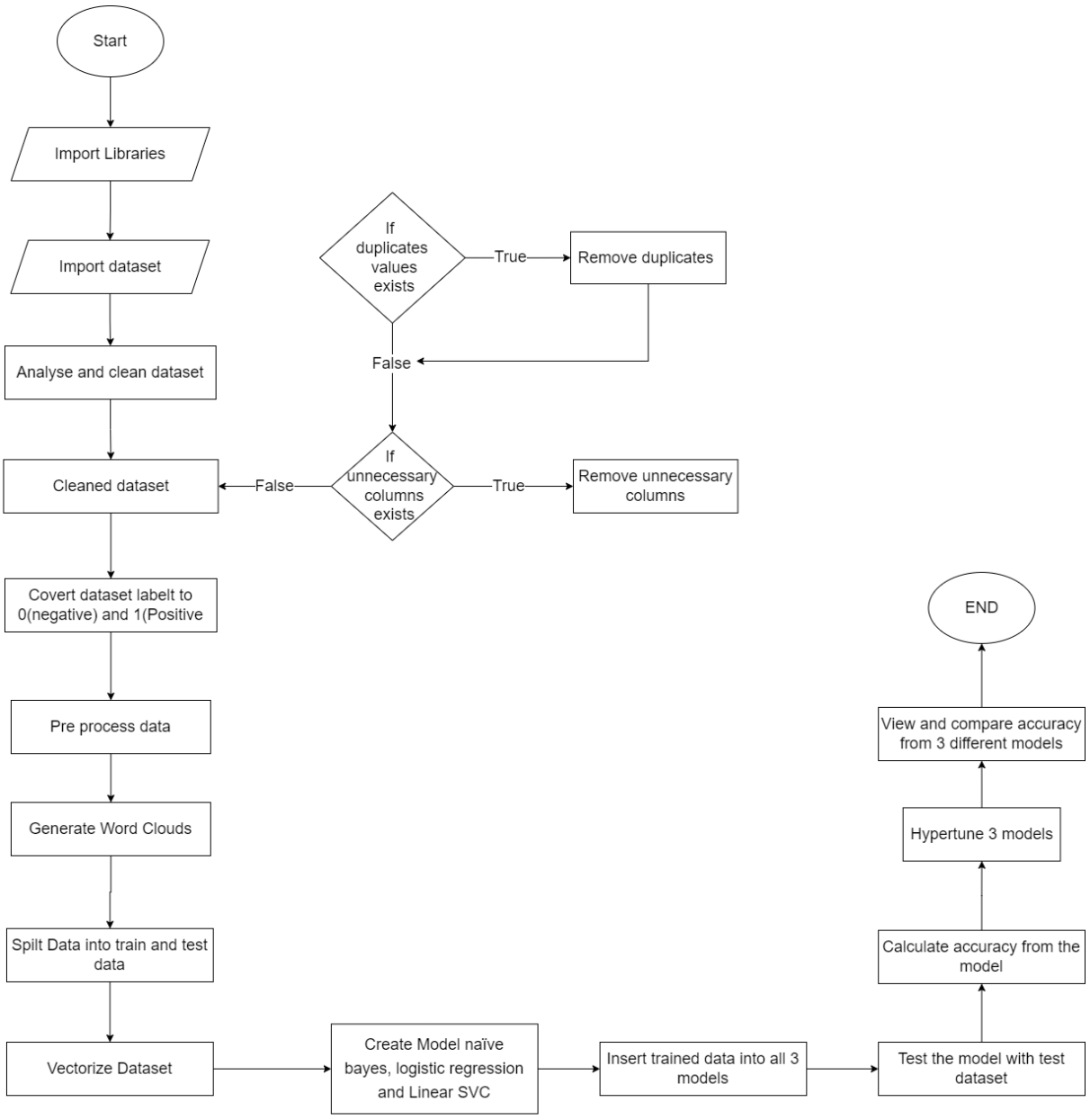


Figure 11: Flowchart

3.5 Development Process involved in Sentimental Analysis System

3.5.1 Tools used.

3.5.1.1 Python

Python is one of the most popular open-sourced programming language. Python is an interpreted, general-purpose, high-level programming language which is popularly used for website development and data analytics, Machine learning and excels in the field of data analysis, AI, and ML. *(Coursera, 2022)*

Python is called general purpose language because it is versatile and can use different function that helps user be more efficient. Python provides with libraries such as pandas, NumPy, Matplotlib and many more to deal with data analysis and visualizing them. While other libraries like Scikit learn, vectorizer allows users to use different Machine learning model *(Maheta, 2023).*

3.5.1.2 Anaconda Navigator (Jupyter Notebook)

Anaconda is one of the most famous open-source distribution of Python and R programming languages. Anaconda is mainly used for data science and machine learning purposes, containing more than 300 libraries for data science allowing suers to work on it easily. Anaconda has features to use different libraries together to work on a given dataset made to popular among users *(Waseem, 2023).* We also will be doing the project in Anaconda Navigator using Jupyter Notebook to read and analyse the dataset. Jupyter Notebook allows us to write codes and visualize them easily in an orderly manner.

Figure 12: Anaconda

(Waseem, 2023)

3.5.1.3 Kaggle

Kaggle is an online platform famous for its the availability of datasets and machine learning models. This platform allows users to share the project datasets, projects especially developed in the field of data science. Kaggle offers many dataset, notebooks in the platform to showcase diverse coding methods and view datasets, which can be looked like a reference for practicing *(Elkabir, 2023).*

Thanks to Kaggle I was able to get a dataset for my project and build a proper Machine learning project by developing proper models and understand different methods like hyperparameter.

3.5.1.4 Microsoft Excel

Microsoft Excel is a software that uses spreadsheets to organize the data. I have used Microsoft excel to store the dataset imported from the Kaggle “Sentiment140 dataset with 1.6 million tweets”.

3.5.2 Toolkit Used

3.5.2.1 NumPy

NumPy is one of the most used Python library that allows user to perform mathematical calculations. Generally used in field of ML and data science, we can almost always see the use of NumPy in every project. NumPy allows user to explore the data efficiently due to its arrays, fast calculation, and various functions it provides *(Chaudhary, 2023).*

I have used NumPy in different places for my project for calculation of complex mathematical operations.

3.5.2.2 Pandas

Just like NumPy, Pandas is another famous library used in Python. Pandas allows use to analyse and manipulate data easily with help of its feature likes creating data frames, cleaning of data and more. Mostly Pandas is also used to display the output in an orderly manner in a tabular form *(Suhani, 2020).*

I have used Pandas in many forms for my project such as removing unnecessary data, displaying output and more.

3.5.2.3 Matplotlib

Matplotlib is another library used in Python, solely for the purpose of data visualization. Matplotlib allows user to develop graphs, plots and many more diagram to visualize the data *(Remi, 2021).*

I have used Matplotlib to create graphs to visualize the dataset.

3.5.2.4 Seaborn

Seaborn is another popular library used in Python that is used for data visualization. Seaborn is built on top of Matplotlib allowing users to build colourful and attractive data visualizations diagrams *(DataScientest, 2023).*

I have used seaborn to create confusion matrix for visualizing different models.

3.5.2.5 WordCloud

WordCloud is one of the most powerful visualization method for the text data, the size of words in the WordCloud denote the importance of word and how frequently it is used. WordCloud generally only comes to play when learning or using machine learning project related to text analysis *(Wang, 2022).*

3.5.2.6 Scikit Learn

Scikit Learn is one of the most powerful library used in machine learning models. Scikit learning provides user with different ways to learn machine learning models including mathematical, statistical, and general-purpose algorithms. Alongside, importing different algorithms, scikit learn allows us to use feature like tuning hyperparameter, generating accuracy score and many more *(Rouse, 2019).*

I have used Scikit learn in my project for building and evaluating Machine Learning Models such as: Bernoulli Naïve Bayes, Linear SVC, and Logistic Regression. Alongside these, Scikit learn has allowed me to calculate accuracy, precision, f1-score and recall. Hyperparameter tuning was also done with help of Scikit learn using Parameter Grid and GridSearchCV.

3.5.2.7 TF-IDF Vectoriser

TF-IDF Vectoriser is an abbreviation for Term Frequency Inverse Document Frequency. This was used to transform the text in the dataset in the form of matrix, which allows the ML model to make better predictions using the dataset fitted in the model *(Chaudhary, 2020).* TF-IDF works by proportionally increasing the number of times words appear in a document. This ultimately helps in preserving rare words that remain in the dataset *(Dutta, 2021).*

3.5.2.8 Libraries and methods used for Text Pre-processing.

a. NLTK (Natural Language Toolkit)

   Natural Language Toolkit is a library used to pre-process human language data. It acts as a bridge of communication between machines and humans that allows machines to understand human language and makes it easy for tasks of classification.

b. Regular expression (re)

   This library is used for removing the HTML tags and special characters that can be found in the dataset.

c. nltk. stem import WordNetLemmatizer.

   Here the words are grouped into different inflected forms of a word. Just like Stemming it brings the context to words linking similar meaning words for example.

   Rocks: rock
   Better: good *(GeeksforGeeks, 2023).*

d. Defining Stop words

   Here the words that don't add much meaning to the sentence are ignored or cleared. For example: the, he, etc.

3.5.3 Explanation of the development Process

3.5.3.1 Import Necessary Libraries

This is the first step of the Sentimental Analysis System. Here, we import all the libraries that are required for the project “Sentimental Analysis”. And Installation of necessary libraries are done.

In [1]:
```
# utilities for the code changes
import re
import pickle
import numpy as np
import pandas as pd

# plots for the diagrams
import seaborn as sns
from wordcloud import WordCloud
import matplotlib.pyplot as plt


# sklearn
from sklearn.svm import LinearSVC
from sklearn.naive_bayes import BernoulliNB
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score, f1_score, classification_report

#for hyperparameter testing
from sklearn.model_selection import ParameterGrid, GridSearchCV
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix


import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
import nltk


# natural language tool kit
from nltk.stem import WordNetLemmatizer
from nltk.stem import wordnet
from nltk.corpus import stopwords

#already downloaded once
# nltk.download()
# nltk.download('wordnet')
```

Figure 13: Import necessary tools and libraries.

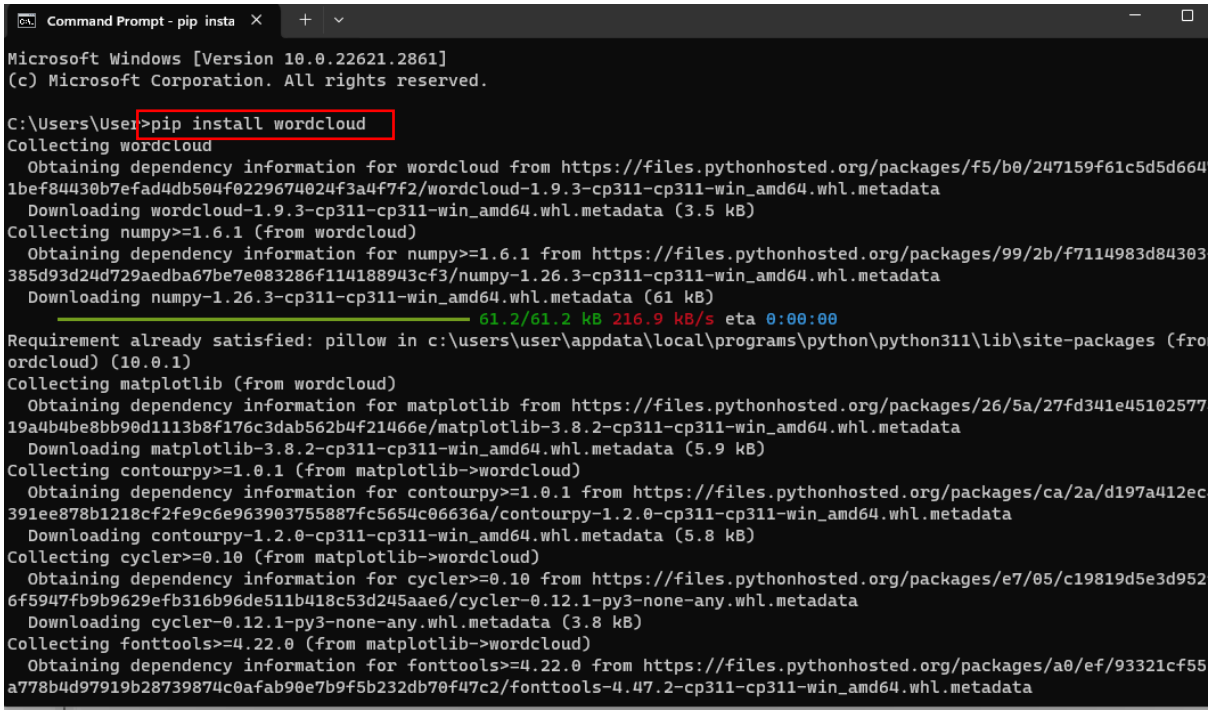


Figure 14: installation of WordCloud

3.5.3.2 Import Dataset

Here the datasets named “Sentiment140 dataset with 1.6 million tweets” includes “Sentiments” from many people that are users in the social media “Twitter”. The dataset is store in form of csv file. We have downloaded the file and will import and load the dataset in a data frame named “dataset” while defining the column header names for the data frame.

Importing dataset

In [3]:
```
#defining columns headers
DATASET_COLUMNS = ["sentiment", "ids", "date", "flag", "user", "tweet"]

#dataset encoding -. ISO-8859-1 for supporting a wide range of characters
DATASET_ENCODING = "ISO-8859-1"
#importing dataset “Sentiment140 dataset with 1.6 million tweets"
dataset = pd.read_csv('dataset/training.1600000.processed.noemoticon.csv',
                      encoding=DATASET_ENCODING , names=DATASET_COLUMNS)

dataset.head()
```

Out[3]:

	sentiment	ids	date	flag	user	tweet
0	0	1467810369	Mon Apr 06 22:19:45 PDT 2009	NO_QUERY	_TheSpecialOne_	@switchfoot http://twitpic.com/2y1zl - Awww, t...
1	0	1467810672	Mon Apr 06 22:19:49 PDT 2009	NO_QUERY	scotthamilton	is upset that he can't update his Facebook by ...
2	0	1467810917	Mon Apr 06 22:19:53 PDT 2009	NO_QUERY	mattycus	@Kenichan I dived many times for the ball. Man...
3	0	1467811184	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	ElleCTF	my whole body feels itchy and like its on fire
4	0	1467811193	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	Karoli	@nationwideclass no, it's not behaving at all....

In [4]:
```
dataset.tail()
```

Out[4]:

	sentiment	ids	date	flag	user	tweet
1599995	4	2193601966	Tue Jun 16 08:40:49 PDT 2009	NO_QUERY	AmandaMarie1028	Just woke up. Having no school is the best fee...
1599996	4	2193601969	Tue Jun 16 08:40:49 PDT 2009	NO_QUERY	TheWDBoards	TheWDB.com - Very cool to hear old Walt interv...
1599997	4	2193601991	Tue Jun 16 08:40:49 PDT 2009	NO_QUERY	bpbabe	Are you ready for your MoJo Makeover? Ask me f...
1599998	4	2193602064	Tue Jun 16 08:40:49 PDT 2009	NO_QUERY	tinydiamondz	Happy 38th Birthday to my boo of alll time!!! ...
1599999	4	2193602129	Tue Jun 16 08:40:50 PDT 2009	NO_QUERY	RyanTrevMorris	happy #charitytuesday @theNSPCC @SparksCharity...

Figure 15: Importing dataset

3.5.3.3 Removing Unnecessary Columns

Here, we now remove unnecessary columns in the dataset, that we don't require for our analysis such as id, date, flag, and user.

Removing Unnecessary columns

In [5]:
```
# Removing unnecessary columns from the dataset
dataset = dataset.drop(['ids', 'date','flag','user'], axis =1)
dataset.head()
```

Out[5]:

	sentiment	tweet
0	0	@switchfoot http://twitpic.com/2y1zl - Awww, t...
1	0	is upset that he can't update his Facebook by ...
2	0	@Kenichan I dived many times for the ball. Man...
3	0	my whole body feels itchy and like its on fire
4	0	@nationwideclass no, it's not behaving at all....

In [6]:
```
dataset.tail()
```

Out[6]:

	sentiment	tweet
1599995	4	Just woke up. Having no school is the best fee...
1599996	4	TheWDB.com - Very cool to hear old Walt interv...
1599997	4	Are you ready for your MoJo Makeover? Ask me f...
1599998	4	Happy 38th Birthday to my boo of alll time!!! ...
1599999	4	happy #charitytuesday @theNSPCC @SparksCharity...

Figure 16: Removing unnecessary columns.

3.5.3.4 Checking for NaN Values

Here, we check the dataset by using the method describe(). And check for any null values available in the dataset. Here the non-null count of the entire dataset implies that there are no missing values (NaN or null values) in either the "sentiment" or "tweet" columns.

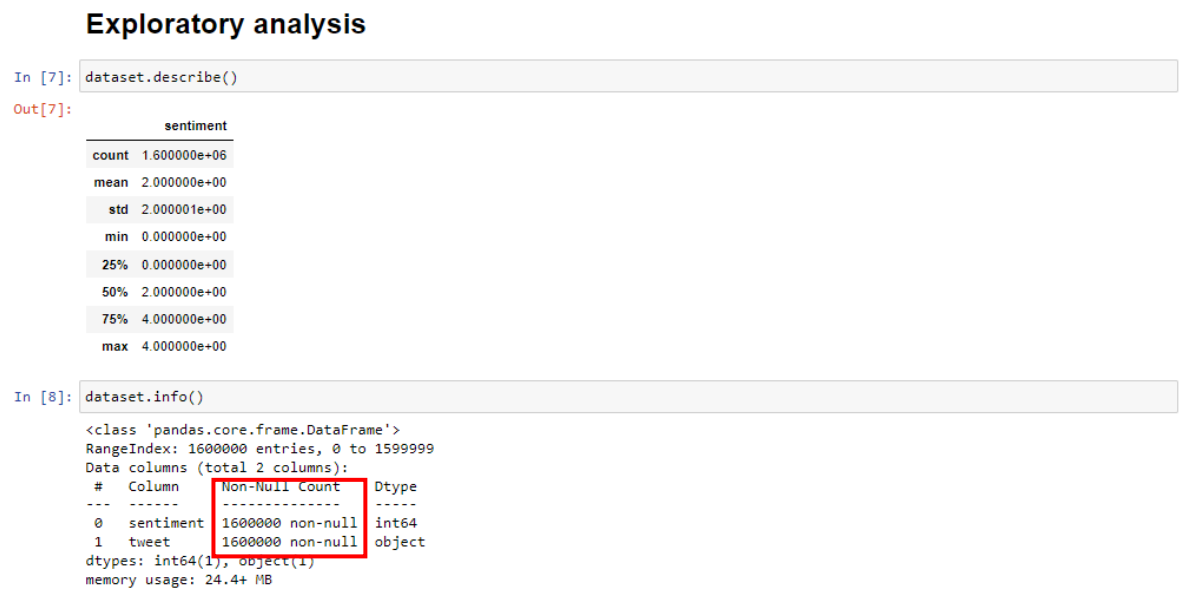


Figure 17: Checking NaN values

3.5.3.5 Changing Sentiment in 0 and 1

Here the sentiment in column is regarded as 0 and 4. This suggests that dataset is in binary. Here the sentiment, polarity of the tweet is (0 = negative, 4 = positive)

Changing Data Accoridng to the need

```
In [9]: #The unique values in the 'sentiment' column are 0 and 4.
        dataset['sentiment'].unique()

Out[9]: array([0, 4], dtype=int64)
```

Figure 18: Sentiment in default data

Here now, we change the data into binary i.e., negative=0 and positive as 1.

```
In [10]: #Replacing data for ease understanding
         dataset['sentiment'] = dataset['sentiment'].replace(4,1)

In [11]: #changed dataset value
         dataset['sentiment'].unique()

Out[11]: array([0, 1], dtype=int64)
```

Figure 19: Sentiment data changes

3.5.3.6 Visualizing the dataset through bar graph

Here, we use matplotlib to visualize the data i.e., positive, and Negative in a bar graph.

Plotting distrubution in dataset

```
In [12]: # Plotting the distribution for dataset.
         ax = dataset.groupby('sentiment').count().plot(kind='bar', title='Distribution of data',
                                                        legend=False, color='#1F77B4')
         ax.set_xticklabels(['Negative','Positive'], rotation=0)

Out[12]: [Text(0, 0, 'Negative'), Text(1, 0, 'Positive')]
```

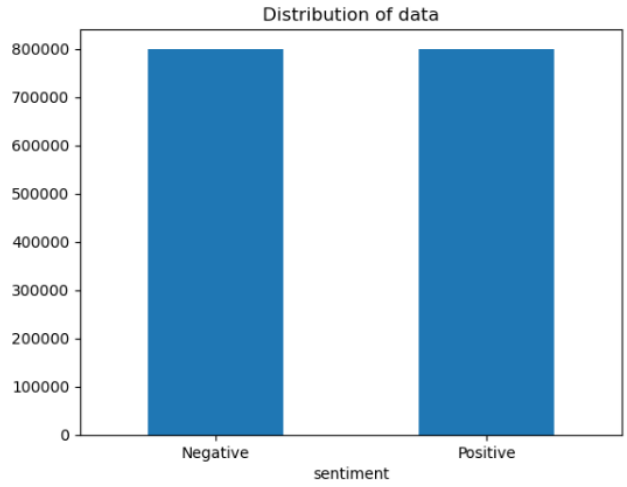


Figure 20: Visualization of sentiment

3.5.3.7 Storing data in the form of list

Here, we now store the data tweet and sentiment in form of list named text and sentiments.

```
In [13]: # Storing data in lists.
         text, sentiment = list(dataset['tweet']), list(dataset['sentiment'])

         # Print the first 5 elements of the 'text' and 'sentiment' lists
         print(dataset[['tweet', 'sentiment']].head())
         print("")
         print(dataset[['tweet', 'sentiment']].tail())
```

```
                                               tweet  sentiment
0  @switchfoot http://twitpic.com/2y1zl - Awww, t...          0
1  is upset that he can't update his Facebook by ...          0
2  @Kenichan I dived many times for the ball. Man...          0
3    my whole body feels itchy and like its on fire           0
4  @nationwideclass no, it's not behaving at all....          0

                                                    tweet  sentiment
1599995  Just woke up. Having no school is the best fee...          1
1599996  TheWDB.com - Very cool to hear old Walt interv...          1
1599997  Are you ready for your MoJo Makeover? Ask me f...          1
1599998  Happy 38th Birthday to my boo of alll time!!! ...          1
1599999  happy #charitytuesday @theNSPCC @SparksCharity...          1
```

Figure 21: Storing data in list.

3.5.3.8 Text Preprocessing

Here we first define dictionary with emojis and set of stop words in a list, to use later to preprocess the text that would be east to analyse. Here, the stopwordlist like “the” will be removed later.

```
In [14]: # Defining dictionary containing all emojis with their meanings.
emojis = {':)': 'smile', ':-)': 'smile', ';d': 'wink', ':-E': 'vampire', ':(': 'sad',
          ':-(': 'sad', ':-<': 'sad', ':P': 'raspberry', ':O': 'surprised',
          ':-@': 'shocked', ':@': 'shocked',':-$': 'confused', ':\\': 'annoyed',
          ':#': 'mute', ':X': 'mute', ':^)': 'smile', ':-&': 'confused', '$_$': 'greedy',
          '@@': 'eyeroll', ':-!': 'confused', ':-D': 'smile', ':-0': 'yell', 'O.o': 'confused',
          '<(-_-)>': 'robot', 'd[-_-]b': 'dj', ":'-)": 'sadsmile', ';)': 'wink',
          ';-)': 'wink', 'O:-)': 'angel','O*-)': 'angel','(:-D': 'gossip', '=^.^=': 'cat'}

## Defining set containing all stopwords in english.
stopwordlist = ['a', 'about', 'above', 'after', 'again', 'ain', 'all', 'am', 'an',
             'and','any','are', 'as', 'at', 'be', 'because', 'been', 'before',
             'being', 'below', 'between','both', 'by', 'can', 'd', 'did', 'do',
             'does', 'doing', 'down', 'during', 'each','few', 'for', 'from',
             'further', 'had', 'has', 'have', 'having', 'he', 'her', 'here',
             'hers', 'herself', 'him', 'himself', 'his', 'how', 'i', 'if', 'in',
             'into','is', 'it', 'its', 'itself', 'just', 'll', 'm', 'ma',
             'me', 'more', 'most','my', 'myself', 'now', 'o', 'of', 'on', 'once',
             'only', 'or', 'other', 'our', 'ours','ourselves', 'out', 'own', 're',
             's', 'same', 'she', "shes", 'should', "shouldve",'so', 'some', 'such',
             't', 'than', 'that', "thatll", 'the', 'their', 'theirs', 'them',
             'themselves', 'then', 'there', 'these', 'they', 'this', 'those',
             'through', 'to', 'too','under', 'until', 'up', 've', 'very', 'was',
             'we', 'were', 'what', 'when', 'where','which','while', 'who', 'whom',
             'why', 'will', 'with', 'won', 'y', 'you', "youd","youll", "youre",
             "youve", 'your', 'yours', 'yourself', 'yourselves']
```

Figure 22: Defining emojis and stopwords.

Here we are creating different functions with the parameter “text”(list we created) to clean the text before loading into the ML models. Here the functions are created to change the tweets into lowercase. While other functions are created for replacing URL, emojis, mentions, non-alphanumeric and consecutive letter into different text.

In [15]:
```
#from re and wordNetLemmatizer

#changing into lowercase
def to_lowercase(text):
    return text.lower()

#replacing url texts
def replace_urls(text):
    url_pattern = r"((http://)[^ ]*|(https://)[^ ]*|( www\.)[^ ]*)"
    return re.sub(url_pattern, ' URL', text)

#replacing emoji with texts
def replace_emojis(text, emojis):
    for emoji in emojis.keys():
        text = text.replace(emoji, "EMOJI" + emojis[emoji])
    return text

#replacing mentions like "@"
def replace_mentions(text):
    user_pattern = '@[^\s]+'
    return re.sub(user_pattern, ' USER', text)

#replacing non aplhanumeric words
def replace_non_alphanumeric(text):
    alpha_pattern = "[^a-zA-Z0-9]"
    return re.sub(alpha_pattern, " ", text)

#replacing consecutive letters like letterrrrr
def replace_consecutive_letters(text):
    sequence_pattern = r"(.)\1\1+"
    seq_replace_pattern = r"\1\1"
    return re.sub(sequence_pattern, seq_replace_pattern, text)
```

Figure 23: Creating functions for text preprocessing.

Finally, lemmatization is used for converting words to base or root form i.e., turning words like “running” into run. This is done so that words can be analysed easily in ML models.

Using a for loop in the function “preprocess” with parameter text data and emojis, The all the tweets will be converted into lowercase using the function. Where all special characters, nun alphanumeric text, emojis, consecutive letters and HTML tags are removed from the tweets by converting all these into texts. Also, stop words created above are removed.

In [16]:
```
# Function to lemmatize each word in the text
def lemmatize_text(text, lemmatizer):
    words = ''
    for word in text.split():
        if len(word) > 1:
            word = lemmatizer.lemmatize(word)
            words += (word + ' ')
    return words


#Function to preprocess a list of text data
def preprocess(textdata, emojis):
    processed_text = []
    lemmatizer = WordNetLemmatizer()
    stop_words = set(stopwords.words('english'))

    for tweet in textdata:
        tweet = to_lowercase(tweet)  # Convert text to lowercase
        tweet = replace_urls(tweet)  # Replace URLs with 'URL'
        tweet = replace_emojis(tweet, emojis)  # Replace emojis
        tweet = replace_mentions(tweet)  # Replace mentions with 'USER'
        tweet = replace_non_alphanumeric(tweet)  # Replace non-alphanumeric characters
        tweet = replace_consecutive_letters(tweet)  # Replace consecutive letters
        tweet = lemmatize_text(tweet, lemmatizer)  # Lemmatize the text

        # Remove stopwords
        tweet_tokens = tweet.split()
        tweet = ' '.join([word for word in tweet_tokens if word not in stop_words])

        processed_text.append(tweet)  # Append the preprocessed text to the result

    return processed_text
```

Figure 24: Implementing the preprocessing functions.

Then the testing of the function “preprocess()”is done using a sample data that we created ourselves. We can see that emoji 😍 is changed in Emojiheart eye, @username is changed into USER, lettersss is changed into letters and the words are transformed into lowercase. Whereas stop words is used to remove words like “the”

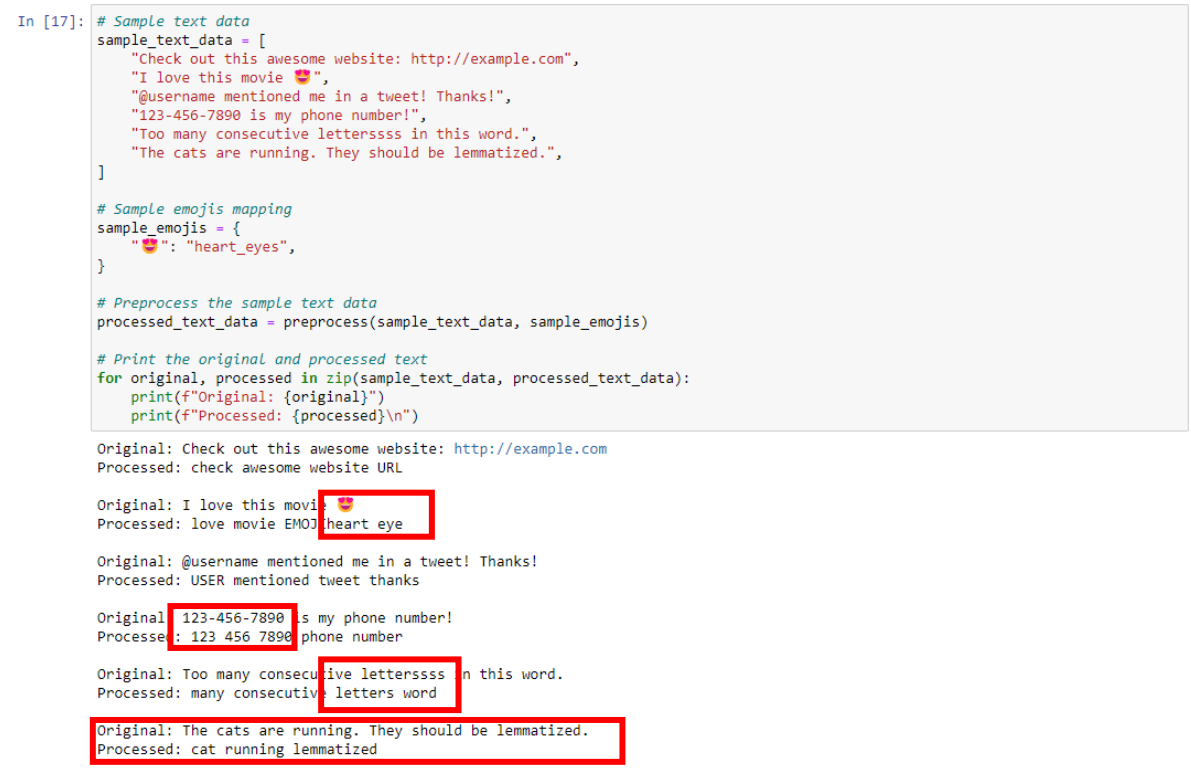


Figure 25: Testing the preprocess function.

3.5.3.9 Implementing Pre-Processing function in dataset

Here, we are pre-processing the data from the dataset using the function used above with help of function preprocess(text, emoji) to remove all the unnecessary data not required for the analysis like, HTML tags, special characters, etc. and checking time for CPU to do so.

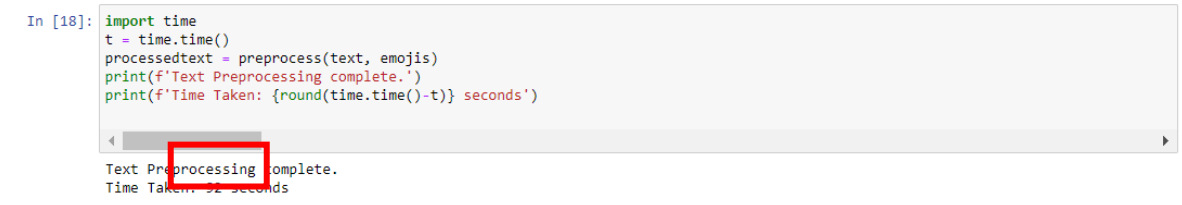


Figure 26: Implementing preprocess function in dataset.

3.5.3.10 Checking the Pre-processed data of the dataset

Here, we can see the data in the dataset has been changed, as the data before preprocess contains html tags, special characters and many more. We can see that username @switchfoot is changed in USER , HTML link is changed into URL, and the words in the dataset are transformed into lowercase.

```
                                                   tweet  sentiment
0  @switchfoot http://twitpic.com/2y1zl - Awww, t...          0
1  is upset that he can't update his Facebook by ...          0
2  @Kenichan I dived many times for the ball. Man...          0
3    my whole body feels itchy and like its on fire           0
4  @nationwideclass no, it's not behaving at all....          0

                                                   tweet  sentiment
1599995  Just woke up. Having no school is the best fee...          1
1599996  TheWDB.com - Very cool to hear old Walt interv...          1
1599997  Are you ready for your MoJo Makeover? Ask me f...          1
1599998  Happy 38th Birthday to my boo of alll time!!! ...          1
1599999  happy #charitytuesday @theNSPCC @SparksCharity...          1
```

Figure 27: Dataset before preprocessing.

```
In [19]: df = pd.DataFrame({
             'text': processedtext,
             'sentiment': sentiment
         })

         # Display the first few rows of the DataFrame
         print(df.head())
         print("")
         print(df.tail())

                                                 text  sentiment
         0  USER URL aww that bummer you shoulda got david...          0
         1  is upset that he can update his facebook by te...          0
         2  USER dived many time for the ball managed to s...          0
         3      my whole body feel itchy and like it on fire           0
         4  USER no it not behaving at all mad why am here...          0

                                                      text  sentiment
         1599995  just woke up having no school is the best feel...          1
         1599996  thewdb com very cool to hear old walt intervie...          1
         1599997  are you ready for your mojo makeover ask me fo...          1
         1599998  happy 38th birthday to my boo of all time tupa...          1
         1599999               happy charitytuesday USER USER USER           1
```

Figure 28: Dataset after preprocessing

3.5.3.11 Word Cloud

Here word cloud is developed from the dataset. Here the library WordCloud is used to plot the data in word clouds for positive and negative tweets to see which occurs the most.

For Positive Tweets

Word Cloud for Positive Tweets

```
In [20]: data_pos = processedtext[800000:]
wc = WordCloud(max_words = 1000 , width = 1600 , height = 800,
               collocations=False).generate(" ".join(data_pos))
plt.figure(figsize = (20,20))
plt.imshow(wc)
```

```
Out[20]: <matplotlib.image.AxesImage at 0x200781c7970>
```

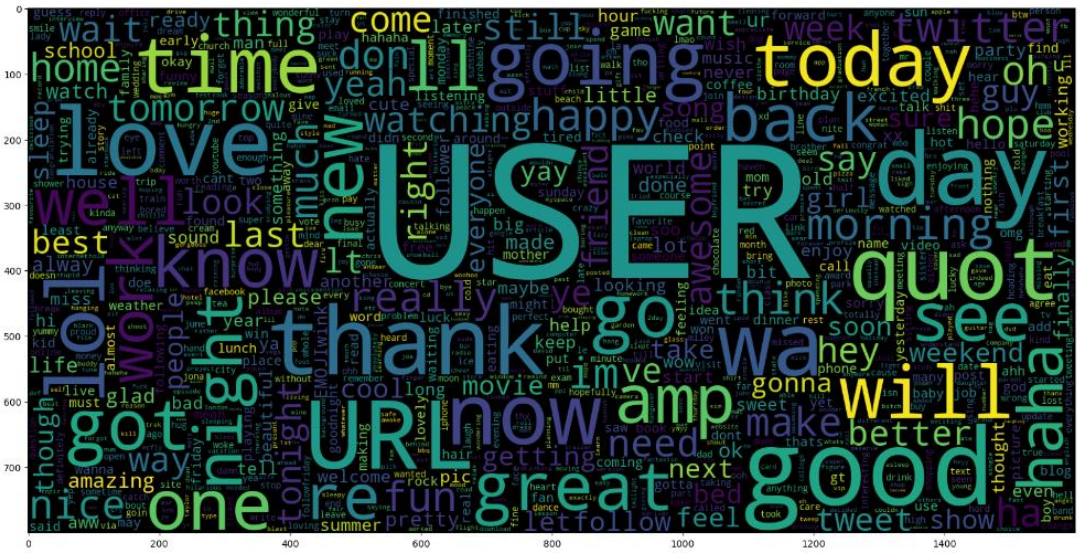


Figure 29: Positive WordCloud

Here we can see that the words generated form the WordCloud for positive tweets can be noted as nice, amazing, glad, best, good, love, haha, pretty, and many more were commonly written in positive review.

For Negative Review

Word Cloud for Negative Tweets

```
In [21]: data_neg = processedtext[:800000]
         plt.figure(figsize=(20,20))
         wc = WordCloud(max_words = 1000, width = 1600, height = 800, collocations=False).generate(" ".join(data_neg))
         plt.imshow(wc)
```

```
Out[21]: <matplotlib.image.AxesImage at 0x2000044fbb0>
```

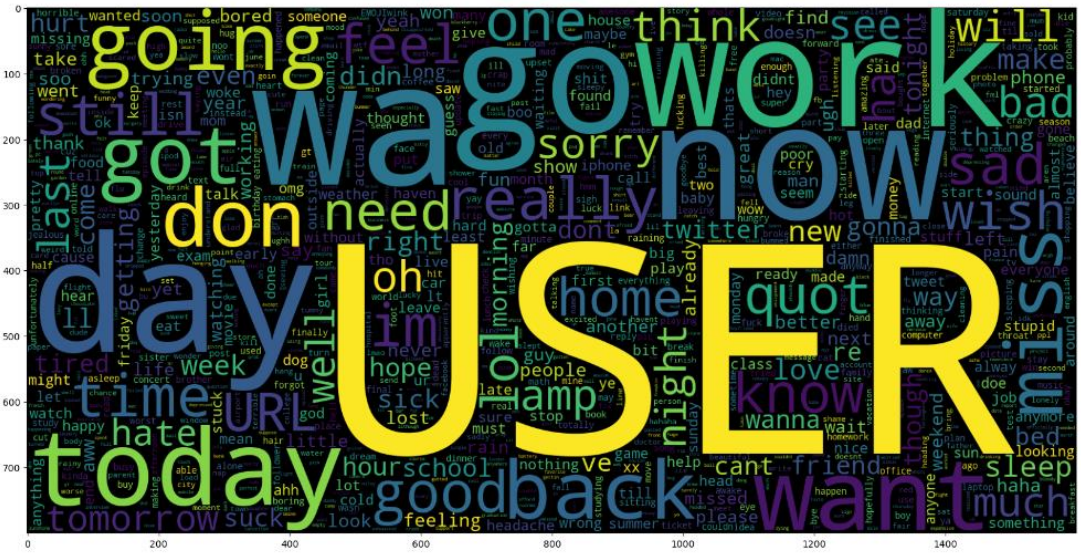


Figure 30: Negative WordCloud

Here we can see that the words generated form the WordCloud for negative tweets can be noted as hate, suck, bad, wrong, missed, stupid, problem, and many more were commonly written in negative review.

3.5.3.12 Splitting data into training and test data

The function returns four sets: X_train: The features for the training set (processed text in this case). X_test: The features for the testing set. y_train: The labels for the training set (sentiments in this case). y_test: The labels for the testing set. So, after running this code, we will have X_train, X_test, y_train, and y_test ready for use in training and evaluating machine learning models.

The training set (X_train and y_train) is used to train the model, and the testing set (X_test and y_test) is used to evaluate its performance.

As the dataset chooses in large, Here the training data consists of 95% of data while the test data contains of 5% of data.

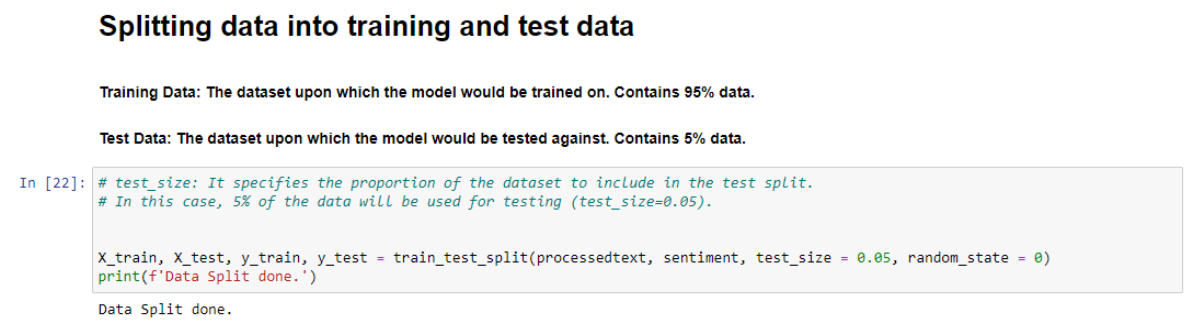


Figure 31: Splitting dataset

3.5.3.13 Data Vectorization

Here, first, the data are split into two parts Training data and Test data. Then the data is vectorized with the help of the library TfidfVectorizer. This code initializes a TF-IDF vectorizer for text data with parameters for considering unigrams and bigrams, limiting the vocabulary size to 500,000 features. It fits the vectorizer to the training data (X_train) and prints the number of unique words and word pairs in the vocabulary. The TF-IDF vectorizer then transform new text data into numerical representations.

```
In [23]: # Initialize the TfidfVectorizer with specified parameters
         vectoriser = TfidfVectorizer(ngram_range=(1,2), max_features=500000)
         vectoriser.fit(X_train)
         print(f'Vectoriser fitted.')

         # Print the number of feature words (terms) in the vocabulary
         print('No. of feature_words: ', len(vectoriser.get_feature_names()))

         Vectoriser fitted.
         No. of feature_words:  500000
```

Figure 32: Data Vectorisation

Transforming the X_train and X_test dataset into matrix of TF-IDF Features by using the TF-IDF Vectoriser. This datasets will be used to train the model and test against it.

```
In [24]: X_train = vectoriser.transform(X_train)
         X_test  = vectoriser.transform(X_test)
         print(f'Data Transformed.')

         Data Transformed.
```

Figure 33: Implementing data vectorisation on train and test data

3.5.3.14 Creation of Machine Learning Model

This function creates the models and plots the confusion matrix. While also giving precision, recall, accuracy and f1-score.

Creating models

In [25]:
```
def model_Evaluate(model):

    # Predict values for Test dataset
    y_pred = model.predict(X_test)

    # Print the evaluation metrics for the dataset.
    print(classification_report(y_test, y_pred))

    # Compute and plot the Confusion matrix
    cf_matrix = confusion_matrix(y_test, y_pred)

    # Define categories, group names and percentage value for the confusion matrix
    categories  = ['Negative','Positive']
    group_names = ['True Neg','False Pos', 'False Neg','True Pos']
    group_percentages = ['{0:.2%}'.format(value) for value in cf_matrix.flatten() / np.sum(cf_matrix)]

    #creating lable for confusion matrix
    labels = [f'{v1}\n{v2}' for v1, v2 in zip(group_names,group_percentages)]
    labels = np.asarray(labels).reshape(2,2)

    # Plot the Confusion Matrix using a heatmap
    sns.heatmap(cf_matrix, annot = labels, cmap = 'Blues',fmt = '',
                xticklabels = categories, yticklabels = categories)

    # Set labels and title for the Confusion Matrix plot
    plt.xlabel("Predicted values", fontdict = {'size':14}, labelpad = 10)
    plt.ylabel("Actual values"   , fontdict = {'size':14}, labelpad = 10)
    plt.title ("Confusion Matrix", fontdict = {'size':18}, pad = 20)
```

Figure 34: Creating models and confusion matrix.

Accuracy Calculation

Formula,

$$Accuracy = \frac{(True\ positive + True\ negative)}{(True\ positive + True\ negative + False\ positive + False\ negative)}$$

Equation 9: Accuracy forumula

Precision Calculation

Formula,

$$Precision = \frac{(True\ positive)}{(True\ positive + \ False\ positive)}$$

Equation 10: Precision formula

Recall Calculation

Formula,

$$Recall = \frac{(True\ positive)}{(True\ positive + \ False\ negative)}$$

Equation 11: Recall formula

F1-Score Calculation

Formula,

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

Equation 12: F1-Score formula

3.5.3.15 Creation and training of BernoulliNB model

The library sklearn is used to import model BernoulliNB. The trained and vectorized data will fit into the models to calculate the accuracy, f1-score, recall, and precision of the sentimental analysis. Alongside the calculation, visual representation of the accuracy is be done through a confusion matrix.

Bernoulli Naive Bayes Model

In [26]:
```
# Create a Bernoulli Naive Bayes model with a specified alpha value
BNBmodel = BernoulliNB(alpha = 2)

# Train the model on the training data
BNBmodel.fit(X_train, y_train)

# Evaluate the model using the model_Evaluate function
model_Evaluate(BNBmodel)
```

```
              precision    recall  f1-score   support

           0       0.79      0.77      0.78     39989
           1       0.78      0.80      0.79     40011

    accuracy                           0.78     80000
   macro avg       0.78      0.78      0.78     80000
weighted avg       0.78      0.78      0.78     80000
```

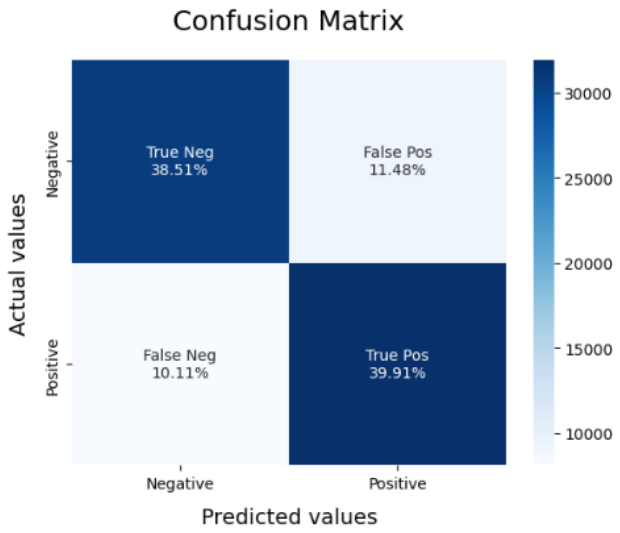


Figure 35: Creation of BernoulliNB model

The accuracy yielded by this model is 0.78 (78%)

The precision yielded by this model is class 0: 0.79(79%) and class 1: 0.78(78%).

The recall yielded by this model is class 0: 0.77(77%) and class 1: 0.80(80%).

The F1-score yielded by this model is class 0: 0.78(78%) and class 1: 0.79(79%).

3.5.3.16 Creation and training of LinearSVC model.

The library sklearn is used to import model LinearSVC. The trained and vectorized data will fit into the models to calculate the accuracy, f1-score, recall, and precision of the sentimental analysis. Alongside the calculation, visual representation of the accuracy is be done through a confusion matrix.

LinearSVC Model

```
In [27]: # Create a LinearSVC Model
SVCmodel = LinearSVC()
SVCmodel.fit(X_train, y_train)
model_Evaluate(SVCmodel)
```

	precision	recall	f1-score	support
0	0.79	0.77	0.78	39989
1	0.78	0.80	0.79	40011
accuracy			0.79	80000
macro avg	0.79	0.79	0.79	80000
weighted avg	0.79	0.79	0.79	80000

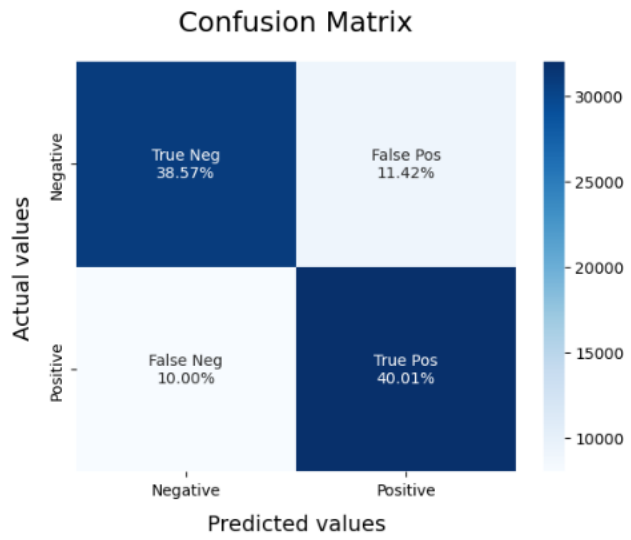


Figure 36: Creation of LinearSVC model

The accuracy yielded by this model is 0.79 (79%)

The precision yielded by this model is class 0: 0.79(79%) and class 1: 0.78(78%).

The recall yielded by this model is class 0: 0.77(77%) and class 1: 0.80(80%).

The F1-score yielded by this model is class 0: 0.78(78%) and class 1: 0.79(79%).

3.5.3.17 Creation and training of Logistic Regression Model.

The library sklearn is used to import model Logistic Regression. The trained and vectorized data will fit into the models to calculate the accuracy, f1-score, recall, and precision of the sentimental analysis. Alongside the calculation, visual representation of the accuracy is be done through a confusion matrix.

Logistic Regression Model

```
In [28]: #creating a Logistic Regression Model
LRmodel = LogisticRegression(C = 2, max_iter = 1000, n_jobs=-1)
LRmodel.fit(X_train, y_train)
model_Evaluate(LRmodel)
```

```
              precision    recall  f1-score   support

           0       0.81      0.78      0.79     39989
           1       0.79      0.81      0.80     40011

    accuracy                           0.80     80000
   macro avg       0.80      0.80      0.80     80000
weighted avg       0.80      0.80      0.80     80000
```

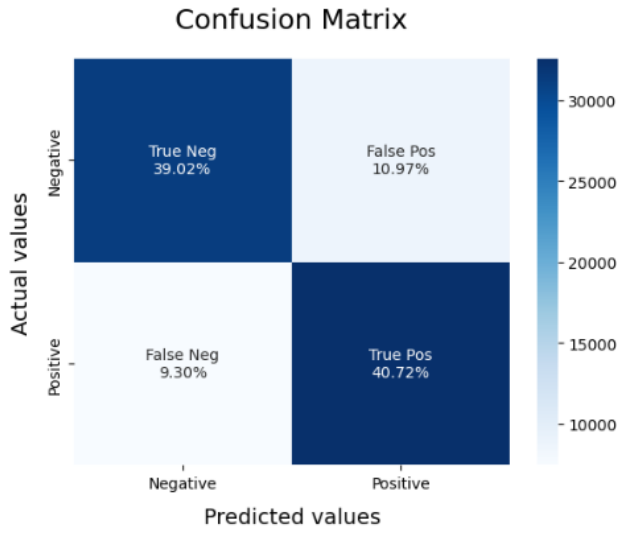


Figure 37: Creation of Logistic Regression Model

The accuracy yielded by this model is 0.80 (80%)

The precision yielded by this model is class 0: 0.81(81%) and class 1: 0.79(79%).

The recall yielded by this model is class 0: 0.78(78%) and class 1: 0.81(81%).

The F1-score yielded by this model is class 0: 0.79(79%) and class 1: 0.80(80%).

3.5.3.18 Tuning hyperparameter

Hyperparameter tuning is done with help of Parameter Grid and GridSearchCV which takes 10 parameters and trains them. The parameter gives the best results and new scores, accuracy, recall, and precision are generated with confusion matrix.

Here 10 values were created which was passed on as parameter on the model. Then the model was tuned and trained with the hyperparameter to find new scores.

3.5.3.18.1 For BernoulliNB

```
In [29]: # Define hyperparameter grid for BNB
param_grid_bnb = {
    'alpha': [0.1, 0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5]
}

# Generate all combinations of hyperparameters
param_combinations_bnb = list(ParameterGrid(param_grid_bnb))

# Initialize BNB model
bnb_model = BernoulliNB()

# Initialize lists to store results
results_bnb = []

# Iterate over hyperparameter combinations
for params in param_combinations_bnb:
    # Set hyperparameters
    bnb_model.set_params(**params)

    # Fit the model
    bnb_model.fit(X_train, y_train)

    # Predict on Test Set
    y_pred_bnb = bnb_model.predict(X_test)

    # Calculate metrics
    accuracy = accuracy_score(y_test, y_pred_bnb)
    precision = precision_score(y_test, y_pred_bnb)
    recall = recall_score(y_test, y_pred_bnb)
    f1 = f1_score(y_test, y_pred_bnb)

    # Compute confusion matrix
    confusion_matrix_bnb = confusion_matrix(y_test, y_pred_bnb)

    # Store results in a dictionary
    result_dict = {
        'Hyperparameters': str(params),
        'Accuracy': accuracy,
        'Precision': precision,
        'Recall': recall,
        'F1-Score': f1,
        'Confusion Matrix': confusion_matrix_bnb
    }

    # Append results to the list
    results_bnb.append(result_dict)

# Create a DataFrame from the results list
results_df_bnb = pd.DataFrame(results_bnb)

# Display the results DataFrame
display(results_df_bnb)
```

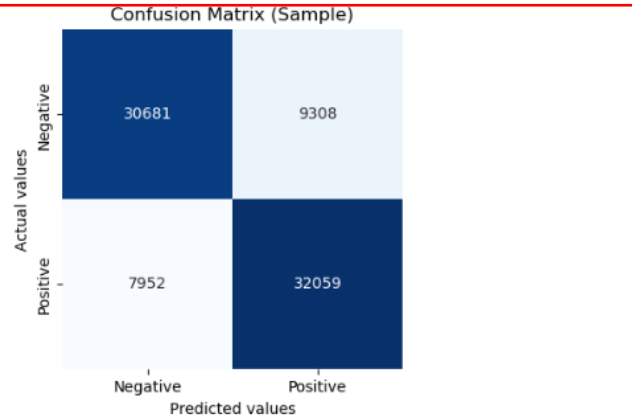
Figure 38: Hyperparameter tuning for BernoulliNB pic 1.

```
display(results_df_bnb)

# Display confusion matrix with heatmap and labels
plt.figure(figsize=(4, 4))
sns.heatmap(confusion_matrix_bnb, annot=True, fmt='d', cmap='Blues', cbar=False,
            xticklabels=['Negative', 'Positive'], yticklabels=['Negative', 'Positive'])
plt.title("Confusion Matrix (Sample)")
plt.xlabel("Predicted values")
plt.ylabel("Actual values")
plt.show()

# Find the row with the highest F1-Score
best_f1_row = results_df_bnb.iloc[results_df_bnb['F1-Score'].idxmax()]

# Find the row with the highest Precision
best_precision_row = results_df_bnb.iloc[results_df_bnb['Precision'].idxmax()]

# Find the row with the highest Recall
best_recall_row = results_df_bnb.iloc[results_df_bnb['Recall'].idxmax()]

# Find the row with the highest Accuracy
best_accuracy_row = results_df_bnb.iloc[results_df_bnb['Accuracy'].idxmax()]

# Print best hyperparameters and corresponding scores for F1-Score, Precision, Recall, and Accuracy
best_params_f1 = best_f1_row['Hyperparameters']
best_score_f1 = best_f1_row['F1-Score']

best_params_precision = best_precision_row['Hyperparameters']
best_score_precision = best_precision_row['Precision']

best_params_recall = best_recall_row['Hyperparameters']
best_score_recall = best_recall_row['Recall']

best_params_accuracy = best_accuracy_row['Hyperparameters']
best_score_accuracy = best_accuracy_row['Accuracy']

print(f"Best Hyperparameters for BNB (F1-Score): {best_params_f1}")
print(f"Best Score for BNB (F1-Score): {best_score_f1}")
print("")
print(f"Best Hyperparameters for BNB (Precision): {best_params_precision}")
print(f"Best Score for BNB (Precision): {best_score_precision}")
print("")
print(f"Best Hyperparameters for BNB (Recall): {best_params_recall}")
print(f"Best Score for BNB (Recall): {best_score_recall}")
print("")
print(f"Best Hyperparameters for BNB (Accuracy): {best_params_accuracy}")
print(f"Best Score for BNB (Accuracy): {best_score_accuracy}")
```

Figure 39: Hyperparameter tuning for BernoulliNB pic 2.

Results after Hyperparameter tuning can be seen in the picture below:

	Hyperparameters	Accuracy	Precision	Recall	F1-Score	Confusion Matrix
0	{'alpha': 0.1}	0.780937	0.775591	0.790808	0.783125	[[30834, 9155], [8370, 31641]]
1	{'alpha': 0.5}	0.783600	0.777650	0.794482	0.785976	[[30900, 9089], [8223, 31788]]
2	{'alpha': 1}	0.784175	0.777561	0.796256	0.786797	[[30875, 9114], [8152, 31859]]
3	{'alpha': 1.5}	0.784338	0.777253	0.797281	0.787139	[[30847, 9142], [8111, 31900]]
4	{'alpha': 2}	0.784125	0.776580	0.797931	0.787111	[[30804, 9185], [8085, 31926]]
5	{'alpha': 2.5}	0.784713	0.776580	0.799580	0.787912	[[30785, 9204], [8019, 31992]]
6	{'alpha': 3}	0.784462	0.776083	0.799805	0.787765	[[30756, 9233], [8010, 32001]]
7	{'alpha': 3.5}	0.784637	0.775838	0.800755	0.788099	[[30732, 9257], [7972, 32039]]
8	{'alpha': 4}	0.784550	0.775573	0.801005	0.788084	[[30715, 9274], [7962, 32049]]
9	{'alpha': 4.5}	0.784250	0.774990	0.801255	0.787903	[[30681, 9308], [7952, 32059]]




```
Best Hyperparameters for BNB (F1-Score): {'alpha': 3.5}
Best Score for BNB (F1-Score): 0.7880994256337093

Best Hyperparameters for BNB (Precision): {'alpha': 0.5}
Best Score for BNB (Precision): 0.7776500232404531

Best Hyperparameters for BNB (Recall): {'alpha': 4.5}
Best Score for BNB (Recall): 0.8012546549698832

Best Hyperparameters for BNB (Accuracy): {'alpha': 2.5}
Best Score for BNB (Accuracy): 0.7847125
```

Figure 40: Hyperparameter tuning for BernoulliNB Results

3.5.3.18.2 For Linear SVC

In [30]:
```
# Define hyperparameter grid for SVC
param_grid_svc = {
    'C': [0.1, 0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5],
}

# Generate all combinations of hyperparameters
param_combinations_svc = list(ParameterGrid(param_grid_svc))

# Initialize SVC model
svc_model = LinearSVC()

# Initialize lists to store results
results_svc = []

# Initialize best F1-Score, Precision, Recall, and Accuracy
best_f1_svc = 0.0
best_precision_svc = 0.0
best_recall_svc = 0.0
best_accuracy_svc = 0.0

# Iterate over hyperparameter combinations
for params in param_combinations_svc:
    # Set hyperparameters
    svc_model.set_params(**params)

    # Fit the model
    svc_model.fit(X_train, y_train)

    # Predict on Test Set
    y_pred_svc = svc_model.predict(X_test)

    # Calculate metrics
    accuracy = accuracy_score(y_test, y_pred_svc)
    precision = precision_score(y_test, y_pred_svc)
    recall = recall_score(y_test, y_pred_svc)
    f1 = f1_score(y_test, y_pred_svc)

    # Compute confusion matrix
    confusion_matrix_svc = confusion_matrix(y_test, y_pred_svc)

    # Store results in a dictionary
    result_dict = {
        'Hyperparameters': str(params),
        'Accuracy': accuracy,
        'Precision': precision,
        'Recall': recall,
        'F1-Score': f1,
        'Confusion Matrix': confusion_matrix_svc  # Keep the confusion matrix as is
    }

    # Append results to the list
    results_svc.append(result_dict)
```

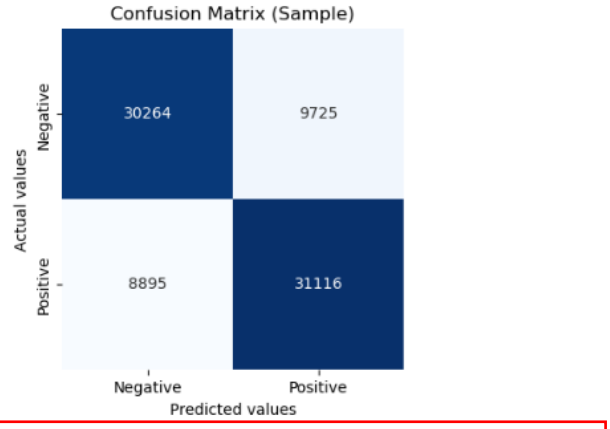
Figure 41: Hyperparameter tuning for Linear SVC pic 1.

```
        'Precision': precision,
        'Recall': recall,
        'F1-Score': f1,
        'Confusion Matrix': confusion_matrix_svc  # Keep the confusion matrix as is
    }

    # Append results to the list
    results_svc.append(result_dict)

    # Update best metrics if the current model has higher values
    if f1 > best_f1_svc:
        best_f1_svc = f1
        best_params_f1_svc = params

    if precision > best_precision_svc:
        best_precision_svc = precision
        best_params_precision_svc = params

    if recall > best_recall_svc:
        best_recall_svc = recall
        best_params_recall_svc = params

    if accuracy > best_accuracy_svc:
        best_accuracy_svc = accuracy
        best_params_accuracy_svc = params

# Create a DataFrame from the results list
results_df_svc = pd.DataFrame(results_svc)

# Display the results DataFrame
display(results_df_svc)

# Display confusion matrix with heatmap and labels
plt.figure(figsize=(4, 4))
sns.heatmap(confusion_matrix_svc, annot=True, fmt='d', cmap='Blues', cbar=False,
            xticklabels=['Negative', 'Positive'], yticklabels=['Negative', 'Positive'])
plt.title("Confusion Matrix (Sample)")
plt.xlabel("Predicted values")
plt.ylabel("Actual values")
plt.show()

# Print best hyperparameters and corresponding scores for F1-Score, Precision, Recall, and Accuracy
print(f"Best Hyperparameters for LinearSVC (F1-Score): {best_params_f1_svc}")
print(f"Best Score for LinearSVC (F1-Score): {best_f1_svc}")
print("")
print(f"Best Hyperparameters for LinearSVC (Precision): {best_params_precision_svc}")
print(f"Best Score for LinearSVC (Precision): {best_precision_svc}")
print("")
print(f"Best Hyperparameters for LinearSVC (Recall): {best_params_recall_svc}")
print(f"Best Score for LinearSVC (Recall): {best_recall_svc}")
print("")
print(f"Best Hyperparameters for LinearSVC (Accuracy): {best_params_accuracy_svc}")
print(f"Best Score for LinearSVC (Accuracy): {best_accuracy_svc}")
```

Figure 42: Hyperparameter tuning for Linear SVC pic 2.

Results after Hyperparameter tuning can be seen in the picture below:

	Hyperparameters	Accuracy	Precision	Recall	F1-Score	Confusion Matrix
0	{'C': 0.1}	0.798188	0.786342	0.819025	0.802350	[[31085, 8904], [7241, 32770]]
1	{'C': 0.5}	0.791913	0.782460	0.808803	0.795413	[[30992, 8997], [7650, 32361]]
2	{'C': 1}	0.785825	0.778005	0.800055	0.788876	[[30855, 9134], [8000, 32011]]
3	{'C': 1.5}	0.780525	0.773490	0.793557	0.783395	[[30691, 9298], [8260, 31751]]
4	{'C': 2}	0.777438	0.770884	0.789708	0.780182	[[30598, 9391], [8414, 31597]]
5	{'C': 2.5}	0.774837	0.768766	0.786309	0.777439	[[30526, 9463], [8550, 31461]]
6	{'C': 3}	0.772637	0.766812	0.783734	0.775181	[[30453, 9536], [8653, 31358]]
7	{'C': 3.5}	0.770837	0.765142	0.781760	0.773362	[[30388, 9601], [8732, 31279]]
8	{'C': 4}	0.769138	0.763633	0.779761	0.771613	[[30332, 9657], [8812, 31199]]
9	{'C': 4.5}	0.767250	0.761881	0.777686	0.769703	[[30264, 9725], [8895, 31116]]




```
Best Hyperparameters for LinearSVC (F1-Score): {'C': 0.1}
Best Score for LinearSVC (F1-Score): 0.8023504927465264

Best Hyperparameters for LinearSVC (Precision): {'C': 0.1}
Best Score for LinearSVC (Precision): 0.7863416038777176

Best Hyperparameters for LinearSVC (Recall): {'C': 0.1}
Best Score for LinearSVC (Recall): 0.8190247681887481

Best Hyperparameters for LinearSVC (Accuracy): {'C': 0.1}
Best Score for LinearSVC (Accuracy): 0.7981875
```

Figure 43: Hyperparameter tuning for Linear SVC Results

3.5.3.18.3 For Logistic Regression

In [31]:

```
# Define hyperparameter grid for LR
param_grid_lr = {
    'C': [0.1, 0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5],
    'max_iter': [100, 200, 300, 400, 500],
}

# Generate all combinations of hyperparameters
param_combinations_lr = list(ParameterGrid(param_grid_lr))

# Initialize LR model
lr_model = LogisticRegression()

# Initialize lists to store results
results_lr = []

# Initialize best F1-Score, Precision, Recall, and Accuracy
best_f1_lr = 0.0
best_precision_lr = 0.0
best_recall_lr = 0.0
best_accuracy_lr = 0.0


# Iterate over hyperparameter combinations
for params in param_combinations_lr:
    # Set hyperparameters
    lr_model.set_params(**params)

    # Fit the model
    lr_model.fit(X_train, y_train)

    # Predict on Test Set
    y_pred_lr = lr_model.predict(X_test)

    # Calculate metrics
    accuracy = accuracy_score(y_test, y_pred_lr)
    precision = precision_score(y_test, y_pred_lr)
    recall = recall_score(y_test, y_pred_lr)
    f1 = f1_score(y_test, y_pred_lr)

    # Compute confusion matrix
    confusion_matrix_lr = confusion_matrix(y_test, y_pred_lr)

    # Store results in a dictionary
    result_dict = {
        'Hyperparameters': str(params),
        'Accuracy': accuracy,
        'Precision': precision,
        'Recall': recall,
        'F1-Score': f1,
        'Confusion Matrix': confusion_matrix_lr  # Keep the confusion matrix as is
    }
```

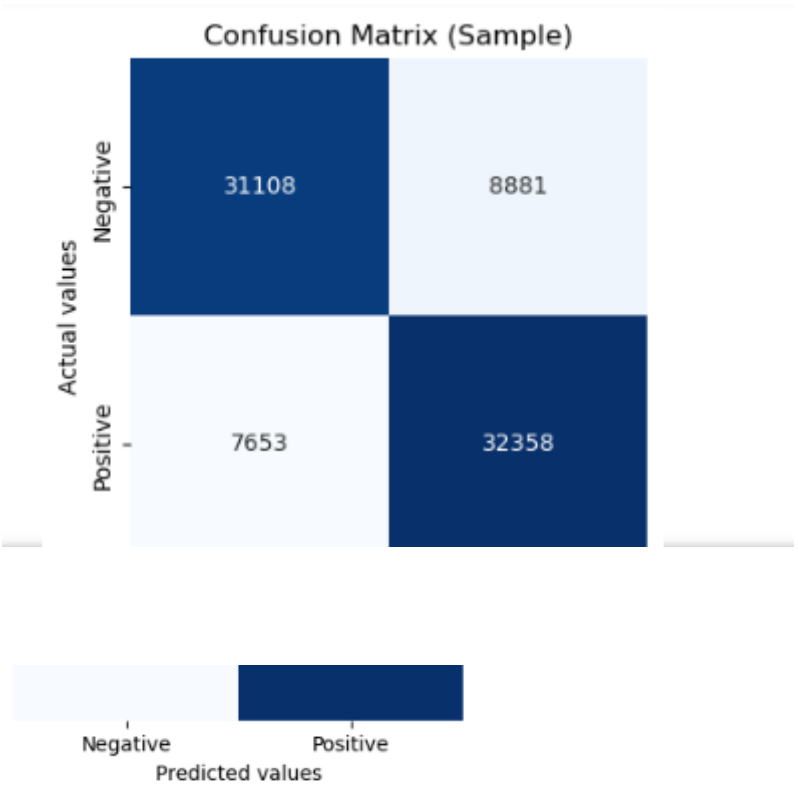
Figure 44: Hyperparameter tuning for Logistic Regression pic 1.

```
    }

    # Append results to the list
    results_lr.append(result_dict)

    # Update best metrics if the current model has higher values
    if f1 > best_f1_lr:
        best_f1_lr = f1
        best_params_f1_lr = params

    if precision > best_precision_lr:
        best_precision_lr = precision
        best_params_precision_lr = params

    if recall > best_recall_lr:
        best_recall_lr = recall
        best_params_recall_lr = params

    if accuracy > best_accuracy_lr:
        best_accuracy_lr = accuracy
        best_params_accuracy_lr = params


# Create a DataFrame from the results list
results_df_lr = pd.DataFrame(results_lr)

# Display the results DataFrame
display(results_df_lr)

# Display confusion matrix with heatmap and labels
plt.figure(figsize=(4, 4))
sns.heatmap(confusion_matrix_lr, annot=True, fmt='d', cmap='Blues', cbar=False,
            xticklabels=['Negative', 'Positive'], yticklabels=['Negative', 'Positive'])
plt.title("Confusion Matrix (Sample)")
plt.xlabel("Predicted values")
plt.ylabel("Actual values")
plt.show()

# Print best hyperparameters and corresponding scores for F1-Score, Precision, Recall, and Accuracy
print(f"Best Hyperparameters for Logistic Regression (F1-Score): {best_params_f1_lr}")
print(f"Best Score for Logistic Regression (F1-Score): {best_f1_lr}")
print("")
print(f"Best Hyperparameters for Logistic Regression (Precision): {best_params_precision_lr}")
print(f"Best Score for Logistic Regression (Precision): {best_precision_lr}")
print("")
print(f"Best Hyperparameters for Logistic Regression (Recall): {best_params_recall_lr}")
print(f"Best Score for Logistic Regression (Recall): {best_recall_lr}")
print("")
print(f"Best Hyperparameters for Logistic Regression (Accuracy): {best_params_accuracy_lr}")
print(f"Best Score for Logistic Regression (Accuracy): {best_accuracy_lr}")
```

Figure 45: Hyperparameter tuning for Logistic Regression pic 2.

Results after Hyperparameter tuning can be seen in the picture below:

	Hyperparameters	Accuracy	Precision	Recall	F1-Score	Confusion Matrix
0	{'C': 0.1, 'max_iter': 100}	0.785763	0.775752	0.804079	0.789662	[[30689, 9300], [7839, 32172]]
1	{'C': 0.1, 'max_iter': 200}	0.785763	0.775712	0.804154	0.789677	[[30686, 9303], [7836, 32175]]
2	{'C': 0.1, 'max_iter': 300}	0.785763	0.775712	0.804154	0.789677	[[30686, 9303], [7836, 32175]]
3	{'C': 0.1, 'max_iter': 400}	0.785763	0.775712	0.804154	0.789677	[[30686, 9303], [7836, 32175]]
4	{'C': 0.1, 'max_iter': 500}	0.785763	0.775712	0.804154	0.789677	[[30686, 9303], [7836, 32175]]
5	{'C': 0.5, 'max_iter': 100}	0.795150	0.784910	0.813276	0.798841	[[31072, 8917], [7471, 32540]]
6	{'C': 0.5, 'max_iter': 200}	0.796362	0.786114	0.814426	0.800020	[[31123, 8866], [7425, 32586]]
7	{'C': 0.5, 'max_iter': 300}	0.796375	0.786050	0.814576	0.800059	[[31118, 8871], [7419, 32592]]
8	{'C': 0.5, 'max_iter': 400}	0.796288	0.786014	0.814401	0.799956	[[31118, 8871], [7426, 32585]]
9	{'C': 0.5, 'max_iter': 500}	0.796288	0.786014	0.814401	0.799956	[[31118, 8871], [7426, 32585]]
10	{'C': 1, 'max_iter': 100}	0.797338	0.787249	0.815051	0.800909	[[31176, 8813], [7400, 32611]]
11	{'C': 1, 'max_iter': 200}	0.798275	0.788203	0.815901	0.801813	[[31217, 8772], [7366, 32645]]




```
Best Hyperparameters for Logistic Regression (F1-Score): {'C': 1, 'max_iter': 200}
Best Score for Logistic Regression (F1-Score): 0.8018126442992582

Best Hyperparameters for Logistic Regression (Precision): {'C': 1, 'max_iter': 300}
Best Score for Logistic Regression (Precision): 0.7884410940369189

Best Hyperparameters for Logistic Regression (Recall): {'C': 1.5, 'max_iter': 300}
Best Score for Logistic Regression (Recall): 0.8159756067081553

Best Hyperparameters for Logistic Regression (Accuracy): {'C': 1, 'max_iter': 300}
Best Score for Logistic Regression (Accuracy): 0.7983125
```

Figure 46: : Hyperparameter tuning for Logistic Regression Results.

3.5.3.19 Saving models and testing them

We are using library called Pickle to save the Vectoriser and models BernoulliNB and Logistic regression.

```
In [32]: file = open('vectoriser-ngram-(1,2).pickle','wb')
         pickle.dump(vectoriser, file)
         file.close()

         file = open('Sentiment-LR.pickle','wb')
         pickle.dump(LRmodel, file)
         file.close()

         file = open('Sentiment-BNB.pickle','wb')
         pickle.dump(BNBmodel, file)
         file.close()
```

Figure 47: Saving models.

Model saved in the working directory.

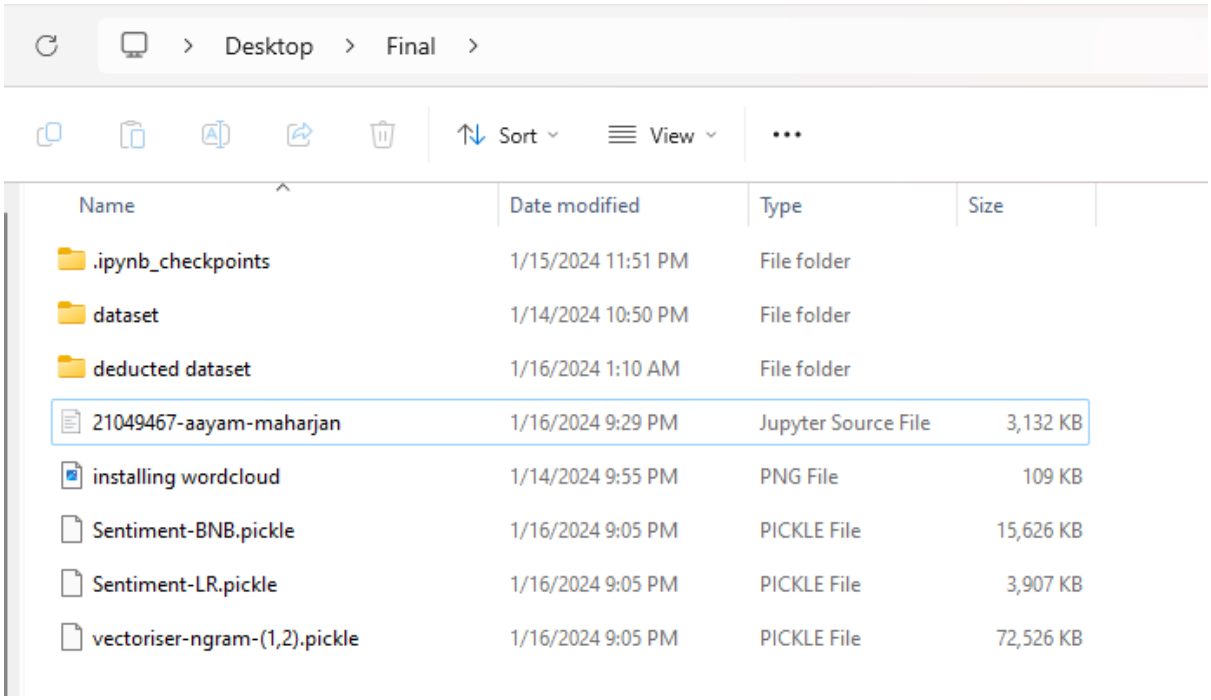


Figure 48: Saved Directory.

Then we test the model by leading them through pickle and initialize the function. After that sample data is loaded in the models.

```
In [33]: def load_models():

             # Loading the vectoriser.
             file = open('vectoriser-ngram-(1,2).pickle', 'rb')
             vectoriser = pickle.load(file)
             file.close()

             # Loading the LR Model.
             file = open('Sentiment-LR.pickle', 'rb')
             LRmodel = pickle.load(file)
             file.close()

             return vectoriser, LRmodel

         #to predict the sentiment
         def predict(vectoriser, model, text):
             # Predict the sentiment
             textdata = vectoriser.transform(preprocess(text, emojis))
             sentiment = model.predict(textdata)

             # Make a list of text with sentiment.
             data = []
             for text, pred in zip(text, sentiment):
                 data.append((text,pred))

             # Convert the list into a Pandas DataFrame.
             df = pd.DataFrame(data, columns = ['text','sentiment'])
             df = df.replace([0,1], ["Negative","Positive"])
             return df

         if __name__=="__main__":
             # Loading the models.

             # Text to classify should be in a list.
             text = ["I hate twitter",
                     "May you love yourself.",
                     "Mr. Stark, I feel so good"]

             df = predict(vectoriser, LRmodel, text)
             print(df.head())
```

```
                        text sentiment
0            I hate twitter  Negative
1     May you love yourself.  Positive
2  Mr. Stark, I feel so good  Positive
```

Figure 49: Loading and testing of the model.

We can see that; the ML model can predict the sentiment correctly and provide correct sentiment based on the text provided to it.

3.5.4 Achieved Results

From the above explanation, it is concluded that without hyperparameter tuning, the resulting scores is slightly less as to when hyperparameter tuning is done with the models. the models performed moderately well without hyperparameter tuning, with BernoulliNB having the best score, SVM coming second and Naïve Bayes performing the worst.

Models	Accuracy	Precision	Recall	F1 Score
BernoulliNB	78%	79%	80%	79%
Linear SVC	79%	79%	80%	79%
Logistic Regression	80%	81%	81%	80%

Table 2: Results of all 3 models

As the hyper tuning was done with small parameters, the results were slightly changed. However, after proper hyperparameter tuning of all 3 models, there were slight improvements in the recalls of all 3 models. The logistic regression accuracy went down by 0.1 % whereas all the precision for 3 models were gone down by few %. Similarly, the F1-score of BernoulliNB went down by 1%, the F1-score for other 2 models were slightly increased.

Models	Accuracy	Precision	Recall	F1 Score
BernoulliNB	78.4%	77%	80.12%	78%
Linear SVC	79.8%	78%	81.9%	80.2%
Logistic Regression	79.9%	78%	81.5%	80.2%

Table 3: Results of models after hyperparameter tuining

At last, I would like to conclude that the Logistic Regression is the best model to perform the sentiment analysis with as it has the best accuracy with highest precision and f1-score. But logistic regression is time consuming as it took the longest among 3 of the models to calculate the data. While BernoulliNB is also a good choice with almost similar accuracy among other 3 models, while being the fastest one to calculate the data.

4 Conclusion

4.1 Analysis of work done

In this document, we can conclude that the project is based on the concepts of AI (Artificial Intelligence), ML (Machine Learning), and NLP (Natural Language Processing). The project is about a Sentimental Analysis of the tweets of people on the application Twitter. The project will be done using the ML algorithm Naïve Bayes, Logistic Regression, and Linear Support Vector Classification (SVC) on the dataset found on Kaggle “Sentiment140 dataset with 1.6 million tweets”.

The project is mainly focused on finding the sentiment of the tweets i.e., positive, or negative from the dataset “Sentiment140 dataset with 1.6 million tweets”. These tweets contain valuable feedback from the public in the form of tweets on various topics like products, trends, stock markets, and political affairs. These sentiment analyses would help organizations take necessary actions to review the tweets. Sentiment analysis can help the growth of a firm to the popularity of a product and find necessary feedback for the organization to perform better and more efficiently.

Alongside, depth research was done by viewing different research papers and finding out how analysis of sentiments could be done with supervised learning methods and how different algorithms would perform and give different accuracy. These research paper helped us in getting new ideas and information on sentimental analysis.

This is the initial document for understanding the main problem domain i.e., getting to know how sentimental data from Twitter can affect different firms. How different firms can plan and manage their products according to the sentimental data calculated from the people tweets dataset.

Alongside these, how the different tools, toolkit was used for the project development can be seen. The whole development process of the project has been documented with proper picture where required. The project used 3 different models to test the sentimental analysis on the dataset “Sentiment140 dataset with 1.6 million tweets”. Where first the data was cleaned thoroughly with help of pandas and different libraries like wordlemmantizer was used to preprocess the data.

While after preprocessing the data was split, trained a vectorised and made ready to train under the 3 different models BernoulliNB, Logistic Regression, and Linear Support Vector Classification (SVC). The model was created using Scikit Learn. While With help of these 3 different models' accuracy, recall, f1-score and precision were found out.

Models	Accuracy	Precision	Recall	F1 Score
BernoulliNB	78%	79%	80%	79%
Linear SVC	79%	79%	80%	79%
Logistic Regression	80%	81%	81%	80%

Table 4: Results of all 3 models pic 2

Which later was further reviewed with help hyperparameter tuning. After this process the model was saved and later tested if the project gave the correct analysis.

Models	Accuracy	Precision	Recall	F1 Score
BernoulliNB	78.4%	77%	80.12%	78%
Linear SVC	79.8%	78%	81.9%	80.2%
Logistic Regression	79.9%	78%	81.5%	80.2%

Table 5: Results after hyperparameter tuning pic 2

We can also conclude that the Logistic regression is the best model for sentimental analysis as it has the highest accuracy of 79.9 % with great precision, recall and f1 score. Only downside being the model took the longest to process the data While Bernoulli NB is also a good choice as, it has almost same accuracy with 78.4 % , being the fastest model among 3 to process the data.

With this documentation, a greater understanding of sentimental analysis can be gained. The research paper gave the idea and understanding of using models and gain the knowledge of hyperparameter tuning. Alongside this, the report allowed us to gain hands on experience on BernoulliNB, Linear SVC and Logistic Regression. Ultimately gaining us experience on AI and ML.

4.2 How the solution addresses a real-world problem

Sentiment analysis as said above is the task of mining people's opinions. Here by analyzing the sentiment from the tweets, we can find opinions such as positive and negative as reviews for different aspects of industries such as the stock market, products, trends, political affairs, and many more. From the sentiment analysis, different firms can find different reviews about themselves and their competition in the market. With the help of the reviews, they can make changes in their product, way of work, etc to increase customer satisfaction and get more positive feedback.

ML / AI is the near future. The accuracy calculated with different algorithms can help the firms plan better for future and mitigate any risk that may turn up in near future. Sentimental analysis will also help them understand customer behaviour and max customer satisfaction as mentioned above.

This project calculates the data from the “Sentiment140 dataset with 1.6 million tweets”. From these sentiment, one can filter out the different sentiment according to the need of the user and carry out different activities like, product promotions, review of product, enhance product, conduct campaigns, and many more. Alongside these tweets related to different trends, political opinions, trends in stock markets, and sales of products, when analyzed properly and concluded according to the user demand, they can find the people positive feedback of the new trends, political changes, etc, and vice versa. And if there is more negative feedback for the product, certain trend, political affairs, etc in the tweets from the public and the company or the party doesn’t know about it, they could know the feedback and change according to the need. Hence, with the help of sentiment analysis, once can find the public demand and make changes as needed or demanded.

4.3 Limitation of the system

Almost every system, consists of some limitation. Although the system is flawless with no mistakes, the system can be better than now and thrive in the future. The system could use more algorithms like KNN, Forest classifier, etc. to develop models and calculate the accuracy score to compete with each other.

Alongside these, only one dataset is used for this project limiting the data scope of the project that could be used. The project could have used more dataset to gain even more data to get more sentiment analysis on more data. Similarly, only few emojis and stopwords were used in this project to preprocess the text, huge list of the stopwords could be used to remove stopwords which will ultimately make the accuracy of the model better. Alongside the WordNetLemmatizer tokenizer could also be used for preprocessing for better results.

Along side these, while tuning the hyperparameters, more iteration could be carryout further to gain more insight into the accuracy, f1-score, recall and precision of the models. More use of confusion matrix could also be done to visualize the results more clearly.

4.4 Further work

This is the project that determines that sentimental analysis for the dataset “Sentiment140 dataset with 1.6 million tweets”. Opinions of people on a certain topic from the tweet and whether it will be positive or negative is determined with the help of the prediction model “BernoulliNB”, “Logistic Regression” and “Linear SVC”. The accuracy and many other analyses have been thoroughly tested and determined.

In the future, more models like KNN, Forest classifier can be used to explore more option in form of ML algorithms to find even for accurate and efficient model. Alongside these hyperparameter tuning iteration can be done more rather than only 10 to further know the accuracy of the models. While tokenization and other preprocessing ways can be used to preprocess the text furthermore.

In this, project we have only used one dataset “Sentiment140 dataset with 1.6 million tweets”. While in the future, more datasets can be included at once to gather more data. As more data the merrier. This will also allow firms to get data from different sources rather than only one at once.

5 Bibliography

Alnajem, D. S., 2021. *8 Sentiment Analysis Real-World Use Cases.* [Online] Available at: https://www.repustate.com/blog/sentiment-analysis-real-world-examples/
[Accessed 18 December 2023].

AMAZINUM, 2023. *What is NLP and how It is Implemented in Our Lives.* [Online] Available at: https://amazinum.com/insights/what-is-nlp-and-how-it-is-implemented-in-our-lives/
[Accessed 19 December 2023].

Apoorv Agarwal, B. X. I. V. O. R. R. P., 2011. Sentiment Analysis of Twitter Data. *Proceedings of the Workshop on Language in Social Media,* p. 9.

avcontentteam, 2023. *Classification vs. Clustering- Which One is Right for Your Data?.* [Online]
Available at: https://www.analyticsvidhya.com/blog/2023/05/classification-vs-clustering/#h-types-of-classification
[Accessed 17 December 2023].

Bannister, K., 2018. *Understanding Sentiment Analysis: What It Is & Why It's Used.* [Online]
Available at: https://www.brandwatch.com/blog/understanding-sentiment-analysis/
[Accessed 17 December 2023].

Barrneck, C., 2021. *What is AI.* Calafornia: Creativecommons.

Belle, V., 2021. Principles and Practice of Explainable Machine Learning. *Frontiers in Big Data,* Volume 4.

Bellini, V., 2022. Understanding basic principles of Artificial Intelligence: a practical guide for intensivists. *Acta Biomed ,* Volume 93.

Brito, M., 2023. *Sentiment Analysis: The Bedrock Of Data-Informed Marketing.* [Online]
Available at: https://www.britopian.com/data/sentiment-analysis/#:~:text=The%20global%20sentiment%20analytics%20market,rate%20of%

20up%20to%2030%25.
[Accessed 17 December 2023].

Chaudhary, M., 2020. *TF-IDF Vectorizer scikit-learn.* [Online] Available at: https://medium.com/@cmukesh8688/tf-idf-vectorizer-scikit-learn-dbc0244a911a
[Accessed 16 January 2024].

Chaudhary, T., 2023. *What is NumPy in Python – Introduction to NumPy – NumPy Tutorial,* s.l.: s.n.

Coursera, 2022. *What Is Python Used For? A Beginner's Guide.* [Online] Available at: https://www.coursera.org/articles/what-is-python-used-for-a-beginners-guide-to-using-python
[Accessed 20 8 2022].

DataScientest, 2023. *Seaborn: Everything you need to know about the Python data visualization tool.* [Online] Available at: https://datascientest.com/en/seaborn-everything-you-need-to-know-about-the-python-data-visualization-tool
[Accessed 16 January 2024].

Dutta, M., 2021. *Bag-of-words vs TFIDF vectorization –A Hands-on Tutorial,* s.l.: Analytics Vidya.

EastVentures, 2023. *Artificial Intelligence in Indonesia: The current state and its opportunities.* [Online] Available at: https://east.vc/insights/artificial-intelligence-in-indonesia-the-current-state-and-its-opportunities/
[Accessed 18 December 2023].

Elkabir, A., 2023. *What Is Kaggle? How to Compete in Kaggle Competitions.,* s.l.: s.n.

Fairlie, M., 2023. *Sentiment analysis is a tool used in social science research and political campaigns to gauge public sentiment towards laws, policies, and social concerns. It assists in forming plans and regulations according to popular opinion..* [Online]
Available at: https://www.businessnewsdaily.com/10018-sentiment-analysis-improve-

business.html
[Accessed 16 January 2024].

Fatemeh Zarisfi Kermani, F. S. E. E., 2019. Solving the□twitter sentiment analysis problem based on□a□machine learning-based approach. *Evolutionary Intelligence.*

GeeksforGeeks, 2023. *Association Rule.* [Online] Available at: https://www.google.com/search?q=association+in+ml&sca_esv=591637205&sxsrf=AM9HkKlspKxkgu9-AOCVgkLmzYeAwCBBEw%3A1702811822114&ei=rth-ZfzJBuKUseMP5Z69kA8&ved=0ahUKEwj82JqurJaDAxViSmwGHWVPD_IQ4dUDCBA&uact=5&oq=association+in+ml&gs_lp=Egxnd3Mtd2l6LXNlcnAiEWFz
[Accessed 17 December 2023].

GeeksforGeeks, 2023. *Naive Bayes Classifiers.* [Online] Available at: https://www.geeksforgeeks.org/naive-bayes-classifiers/
[Accessed 18 December 2023].

GeeksforGeeks, 2023. *Python | Lemmatization with NLTK.* [Online] Available at: https://www.geeksforgeeks.org/python-lemmatization-with-nltk/
[Accessed 19 December 2023].

GeeksforGeeks, 2023. *Twitter Sentiment Analysis using Python.* [Online] Available at: https://www.geeksforgeeks.org/twitter-sentiment-analysis-using-python/
[Accessed 18 December 2023].

Huilgol, P., 2023. *Precision and Recall | Essential Metrics for Machine Learning (2024 Update).* [Online] Available at: https://www.analyticsvidhya.com/blog/2020/09/precision-recall-machine-learning/
[Accessed 16 January 2024].

InternetLiveStatus, 2014. *Twitter Usage Statistics.* [Online] Available at: https://www.internetlivestats.com/twitter-statistics/
[Accessed 18 December 2023].

K.L. Sudha, P. B. P., 2022. Sentiment Analysis in Twitter Data. *International Journal Of Creative Research Thoughts,* 10(7), p. 6.

Kanade, V., 2022. *What Is Logistic Regression? Equation, Assumptions, Types, and Best Practices,* s.l.: SpiceWorks.

KAZANOVA, M. M., 2017. *Sentiment140 dataset with 1.6 million tweets.* [Online] Available at: https://www.kaggle.com/datasets/kazanova/sentiment140 [Accessed 19 December 2023].

Kermani, F. Z., 2020. *Solving the twitter sentiment analysis problem based on a machine learning-based approach,* Berlin: Springer-Verlag.

Kim Schouten, F. F., 2015. *The Benefit of Concept-Based Features for Sentiment Analysis.* s.l., Semantic Web Evaluation.

Kundu, R., 2022. *F1 Score in Machine Learning: Intro & Calculation.* [Online] Available at: https://www.v7labs.com/blog/f1-score-guide#:~:text=F1%20score%20is%20a%20machine%20learning%20evaluation%20metric%20that%20measures,prediction%20across%20the%20entire%20dataset. [Accessed 16 January 2024].

Liu, Q., 2015. *Supervised Learning,* California: s.n.

Lorberfeld, A., 2019. *Machine Learning Algorithms In Layman's Terms, Part 1.* [Online] Available at: https://towardsdatascience.com/machine-learning-algorithms-in-laymans-terms-part-1-d0368d769a7b [Accessed 18 December 2023].

Luo, T., 2013. *Sentiment Analysis.* s.l.:Guandong Xu.

Mahesh, B., 2020. Machine Learning Algorithms -A Review. *International Journal of Science and Research (IJSR),* 9(1), p. 386.

Maheta, D., 2023. *Python with Machine Learning: Make User Experience Interactive.* [Online]
Available at: https://www.bacancytechnology.com/blog/python-with-machine-learning [Accessed 16 January 2024].

Manning, P. C., 2020. *Artificial Intelligence Definitions,* California: Stanford University.

Medium, 2019. *Confusion Matrix, Accuracy, Precision, Recall, F1 Score.* [Online] Available at: https://medium.com/analytics-vidhya/confusion-matrix-accuracy-

precision-recall-f1-score-ade299cf63cd
[Accessed 16 January 2024].

Nikhil Yadav, O. K. A. R. S. G., 2020. Twitter Sentiment Analysis Using Supervised Machine Learning. *Intelligent Data Communication Technologies,* p. 18.

Pfeffer, J., 2023. *Just Another Day on Twitter: A Complete 24 Hours of Twitter Data,* München: Technische Universität München.

Remi, 2021. *What Is Matplotlib In Python? How to use it for plotting?.* [Online] Available at: https://www.activestate.com/resources/quick-reads/what-is-matplotlib-in-python-how-to-use-it-for-plotting/
[Accessed 16 January 2024].

Repustate, 2021. *Sentiment analysis challenges and how to overcome them.* [Online] Available at: https://www.repustate.com/blog/sentiment-analysis-challenges-with-solutions/
[Accessed 17 December 2023].

Robinson, S., 2021. *Sentiment analysis: Why it's necessary and how it improves CX,* s.l.: s.n.

Rouse, M., 2019. *What Does Scikit-Learn Mean?.* [Online] Available at: https://www.techopedia.com/definition/33860/scikit-learn
[Accessed 16 January 2024].

Sharma, S., 2023. *Top 10 Applications of Sentiment Analysis in Business,* s.l.: s.n.

Simplilearn, 2023. *What is Regression Analysis - Types | Examples | Uses.* [Online] Available at: https://www.simplilearn.com/tutorials/excel-tutorial/regression-analysis
[Accessed 17 December 2023].

Sodhar, I. H., 2022. *Natural Language Processing: Applications, Techniques and Challenges,* Delhi: AkiNik Publications .

Suhani, 2020. *What Is Pandas in Python? Everything You Need to Know.* [Online] Available at: https://www.activestate.com/resources/quick-reads/what-is-pandas-in-python-everything-you-need-to-know/
[Accessed 16 January 2024].

Taherdoost, H., 2023. *Artificial Intelligence and Sentiment Analysis: A Review in Competitive Research,* Vancouver: Hamta Business Corporation.

Vishal A. Kharde, S. S., 2016. Sentiment Analysis of Twitter Data: A Survey of. *International Journal of Computer Applications,* 139(11), p. 15.

Wang, M., 2022. *WordClouds with Python.* [Online]
Available at: https://medium.com/mlearning-ai/wordclouds-with-python-c287887acc8b#:~:text=Word%20cloud%20is%20one%20of,learning%20projects%20or%20text%20analysis.
[Accessed 16 January 2024].

Waseem, M., 2023. *Python Anaconda Tutorial : Everything You Need To Know.* [Online]
Available at: https://www.edureka.co/blog/python-anaconda-tutorial/
[Accessed 16 January 2024].

Webb, G. I., 2016. *Naïve Bayes.* s.l.:Sprint New York.

Wilson, J., 2022. *Python.* [Online]
Available at: https://www.techopedia.com/definition/3533/python#python-frameworks
[Accessed 20 8 2022].

Wu, W., 2022. *Unsupervised Learning,* Tübingen: University of Tübingen.

Zulfadzli Drus, H. K., 2019. *Sentiment Analysis in Social Media and Its Application: Systematic.* Kuala Lumpur, Procedia Computer Science.