

TWITTER SENTIMENT ANALYSIS

A Project Report Submitted in partial fulfillment of the requirements for
the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

By

NAVADEEP REDDY (2010030313)

K. SHASHIKANTH (2010030494)

MANOJ PERAVALI (2010030503)

M.ROHIT (2010030439)



**DEPARTMENT OF
COMPUTER SCIENCE AND ENGINEERING
K L DEEMED TO BE UNIVERSITY
AZIZNAGAR, MOINABAD , HYDERABAD-500 075
MARCH 2023**

BONAFIDE CERTIFICATE

This is to certify that the project titled **TWITTER SENTIMENT ANALYSIS** is a bonafide record of the work done by

NAVADEEP REDDY (2010030313)

K. SHASHIKANTH (2010030494)

MANOJ PERAVALI (2010030503)

M.ROHIT (2010030439)

in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **COMPUTER SCIENCE AND ENGINEERING** of the **K L DEEMED TO BE UNIVERSITY, AZIZNAGAR, MOINABAD , HYDERABAD-500 075**, during the year 2022-2023.

Dr. Subhranginee Das

Project Guide

Dr.Arпита Gupta

Head of the Department

Project Viva-voce held on _____

Internal Examiner

External Examiner

ABSTRACT

The World Wide Web, including social networks, forums, review sites, and blogs, creates a lot of data with users' views, emotions, opinions, and arguments on various topics like social events, products, brands, and politics. These opinions greatly influence readers, product vendors, and politicians. Analyzing this unstructured data from social media is crucial, and sentiment analysis is a method gaining significant attention for organizing it.

Sentiment analysis categorizes expressed feelings in different ways, such as positive, negative, favorable, or unfavorable. However, one challenge is the lack of labeled data in Natural Language Processing (NLP). Sentiment analysis, also known as opinion mining, is useful for understanding the emotions and opinions in any text. It's especially valuable for expressing the opinions of groups or individuals, particularly on social media platforms like Twitter.

In a recent study, various machine learning techniques, including Naïve Bayes, Logistic Regression, Stochastic Gradient Descent (SGD), and Support Vector Machine (SVM), were tested for sentiment analysis of tweets. The study aimed to identify the most efficient machine learning model for sentiment analysis in English. Twitter's API was used to collect data, which was then preprocessed and analyzed using Natural Language Processing. Then the trained model is used for analysis of sentiment of the sentence provided.

ACKNOWLEDGEMENT

We would like to thank the following people for their support and guidance without whom the completion of this project in fruition would not be possible.

Dr. Subhranginee Das, our project guide, for helping us and guiding us in the course of this project .

Dr.Arpita Gupta, the Head of the Department, Department of Computer Science and Engineering.

Our internal reviewers, **Dr.Kakali Das** , **Dr.Figlu Mohanty** , **Mr.Shasider K** , **Mr.Chanda Raj Kumar** for their insight and advice provided during the review sessions.

We would also like to thank our individual parents and friends for their constant support.

TABLE OF CONTENTS

Title	Page No.
ABSTRACT	ii
ACKNOWLEDGEMENT	iii
TABLE OF CONTENTS	iv
LIST OF FIGURES	v
1 Introduction	1
2 Literature Review	2
2.1 Article 1	2
2.2 Article 2	2
2.3 Article 3	3
2.4 Article 4	3
2.5 Article 5	3
3 Proposed System	4
3.1 Logistic regression	4
3.2 Naive Bayes	5
3.3 Stochastic Gradient Descent	6
3.4 Support Vector Machine	7
3.5 System Requirements	7
4 Implementation	8

4.1	Libraries	8
4.2	Tools and Technologies	9
5	Results and Analysis	12
5.1	Performance Evaluation	12
5.2	Result	13
6	Conclusion and Future Scope	18
6.1	Summary of the Project	18
6.2	Future Scope	18
	References	20
	Appendices	23
A	Source Code	24
B	Dataset used in the project	32

List of Figures

3.1	Logistic regression	5
3.2	Naive Bayes	6
3.3	Stochastic Gradient Descent	6
3.4	Support Vector Machine	7
5.1	Logistic Regression	12
5.2	Naive Bayes	13
5.3	Stochastic Gradient Descent	13
5.4	Result 1	14
5.5	Result 2	14

5.6	Result 3	15
5.7	Result 4	15
5.8	Result 5	16
5.9	Result 6	16
5.10	Result 7	17
5.11	Result 8	17
5.12	Result 9	17
A.1	Implementation 1	24
A.2	Implementation 2	25
A.3	Implementation 3	25
A.4	Implementation 4	25
A.5	Implementation 5	26
A.6	Implementation 6	26
A.7	Implementation 7	26
A.8	Implementation 8	27
A.9	Implementation 9	27
A.10	Implementation 10	28
A.11	Implementation 11	28
A.12	Implementation 12	29
A.13	Implementation 13	30
A.14	Implementation 14	30
A.15	Implementation 15	31
A.16	Implementation 16	31
A.17	Implementation 17	31
B.1	An image of dataset with negative tweets	32
B.2	An image of dataset with positive tweets	32

Chapter 1

Introduction

Sentiment analysis examines people's feelings in written language. It's used in many areas because people's comments and reviews reflect their emotions toward products and services. Methods like Natural Language Processing (NLP) help computers understand human language. Machine Learning (ML) includes various techniques for learning from data quickly.

Machine learning algorithms like Naive Bayes and Logistic Regression are commonly used for classification tasks. Data collection is growing rapidly, with data doubling every two years. Data mining helps find patterns in large datasets, like those from Twitter. This study focuses on classifying sentiment in tweets using NLP and ML algorithms.

Different models will be compared based on their performance metrics. Finding the best classifier is important for accurately assessing people's emotions. Sentiment analysis plays a crucial role in various fields, including marketing and understanding people's changing needs. It requires a quality training set and semantic analysis for accurate results.

Chapter 2

Literature Review

2.1 Article 1

Authors: S Qi and Y. Shabrina

Title: Sentiment Analysis Using Twitter Data

Review: Explored sentiment analysis methods, comparing lexicon-based and machine-learning-based approaches. Highlighted the importance of analyzing social media content for understanding emotions and perceptions

Methods: Lexicon-based, machine-learning approach.

2.2 Article 2

Authors: Venkata Sasank Pagolu, Kamal Nayan Reddy Challa, Ganapati Panda and Babita Majhi

Title: Sentiment Analysis of Twitter Data for Financial Market Prediction

Review: This paper explores classifier with features as Word2vec representations of human annotated tweets trained on Random Forest algorithm with and With N-gram representations, the classifier model with same algorithm. The model is trained with word2vec representations is picked to classify the nonhuman annotated tweets because of its promising accuracy for large datasets and the sustainability in word meaning. The results obtained very good figures while predicting the sentiments in short texts, tweets, less than 140 characters in length.

Methods: N-gram and Word2vec.

2.3 Article 3

Authors: C.J. Hutto and Eric Gilbert

Title: VADER: A Parsimonious Rule-Based Model for Sentiment Analysis of Social Media Text

Review: This paper introduces VADER (Valence Aware Dictionary and sEntiment Reasoner), a lexicon and rule-based sentiment analysis tool specifically designed for social media text. VADER is known for its effectiveness in handling noisy and informal text commonly found on platforms like Twitter.

Methods: VADER

2.4 Article 4

Authors: Zulfadzli Drus and Haliyana Khalid

Title: Sentiment Analysis in Social Media and Its Application: Systematic Literature Review

Review: They conducted systematic literature review that provides information on studies on sentiment analysis in social media. It shows what is the method used in analyzing sentiment in social media, most common type of social media site to extract Information and the application of sentiment analysis in social media.

Methods: Review

2.5 Article 5

Authors: Kassinda Francisco Martins Panguila and Dr. Chandra J

Title: Sentiment Analysis on Social Media Data Using Intelligent Techniques

Review: Sentiment analysis using intelligent techniques approach was proposed to deal with social media data. According to their experiment results Neural Networks methods such as Multi-layer Perceptron (MLP) and Convolutional Neural Network (CNN) performed better than others classifier in general.

Methods: Multi-layer Perceptron and Convolutional Neural Network.

Chapter 3

Proposed System

3.1 Logistic regression

Logistic regression is used for understanding the meaning of words in sentences because it's good at predicting things. It's like a helpful friend who can tell if something is likely to happen or not. People use logistic regression because it's straightforward and works well with lots of data. It's especially handy for figuring out probabilities, like whether a sentence is positive or negative. Logistic regression helps make sense of words by guessing the chances of different meanings, making it easier to understand what's being said in a bunch of writing.

In terms of odds, logistic regression can also be stated as follows: The formula for odds($y=1|x$) is $P(y=1|x) \frac{P(y=1|x)}{1-P(y=1|x)} = e^z$.

By taking both sides' natural logarithms (log), we obtain:

The expression

$$\ln \left(\frac{P(y = 1|x)}{1 - P(y = 1|x)} \right) = \ln(e^z)$$

The formula is

$$\ln \left(\frac{P(y = 1|x)}{1 - P(y = 1|x)} \right) = z$$

where z is the previously defined linear combination of the input features and their coefficients. Since logistic regression coefficients indicate the change in the log-odds of the positive class for a one-unit change in the corresponding input feature, they are frequently interpreted using this formulation.

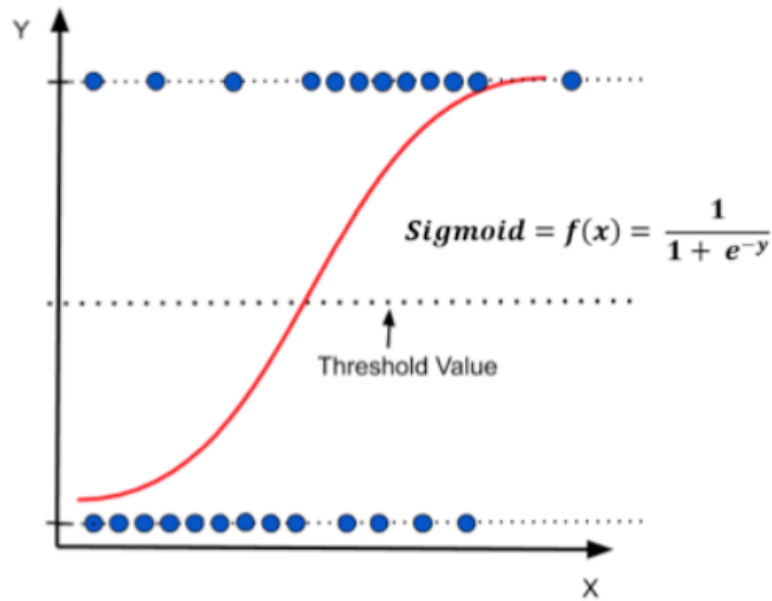


Figure 3.1: Logistic regression

3.2 Naive Bayes

Naive Bayes is widely used in semantic analysis because of its ease of use, sharpness, and efficiency while processing textual input. Understanding the meaning included in textual information, such as emails, documents, and posts on social media, is the main objective of semantic analysis. Because Naive Bayes classifiers can effectively classify text based on the likelihood that a given document belongs to a specific class or category, they are especially well-suited for this purpose.

Naive Bayes has several advantages for semantic analysis, one of which is the simplicity of handling massive amounts of textual input. Naive Bayes classifiers are also appropriate for analyzing unstructured text data, which frequently contains noise and irrelevant information because they can handle irrelevant features gracefully and are robust to noisy data.

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)}$$

Likelihood
Class Prior Probability
Posterior Probability
Predictor Prior Probability

$$P(c|X) = P(x_1|c) \times P(x_2|c) \times \dots \times P(x_n|c) \times P(c)$$

Figure 3.2: Naive Bayes

3.3 Stochastic Gradient Descent

A stochastic Gradient Descent (SGD) classifier is used for understanding the meaning of words in sentences because it's fast and efficient. It's like a speedy worker who quickly learns from examples to figure out what words mean. People like using SGD because it can handle large amounts of text data without taking too much time. It's especially good at learning from lots of examples and making decisions based on them. SGD classifier helps make sense of words by quickly adjusting its guesses to match what's being said in a bunch of writing, making it easier to understand the meaning behind the words.

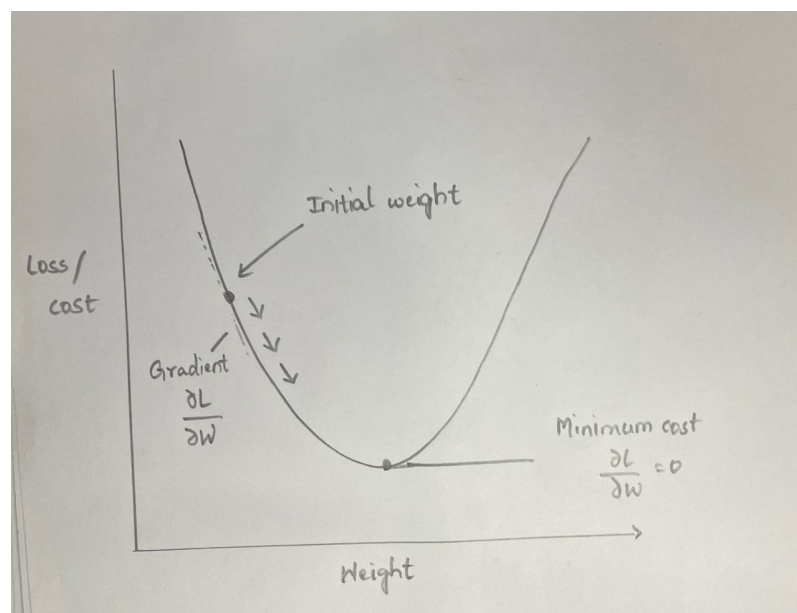


Figure 3.3: Stochastic Gradient Descent

3.4 Support Vector Machine

Support Vector Machines (SVM) are used for understanding what words mean in sentences because they're good at finding patterns in data. They're like detectives looking for clues to figure out the meaning behind text. People like using SVM because they're reliable and can handle big amounts of text easily. They're especially good at separating different types of text, like figuring out if something is positive or negative. SVM helps make sense of words by finding the best way to group them together, helping us understand the meaning in a bunch of writing.

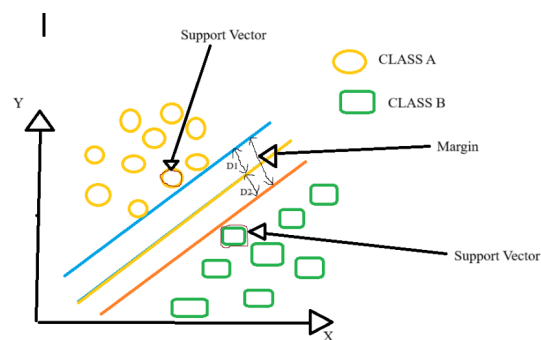


Figure 3.4: Support Vector Machine

3.5 System Requirements

Hardware Requirements:

- Processor: intel core i7, i9
- Memory Space: 16gb
- RAM: 8gb

Software Requirements:

- Operating System – Windows,linux
- IDE – Pycharm
- Programming Language - Python

Chapter 4

Implementation

4.1 Libraries

1. Kaggle API (!pip install kaggle): The Kaggle API allows users to interact with the Kaggle platform programmatically, enabling them to download datasets, participate in competitions, and more.
2. Pandas: Pandas is a powerful data manipulation and analysis library in Python. It provides data structures and functions for working with structured data, such as
3. NumPy: NumPy is a fundamental package for scientific computing with Python. It provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays.
4. NLTK (Natural Language Toolkit): NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources, such as WordNet. In this code, NLTK is used for text preprocessing tasks like tokenization and stemming.
5. Scikit-learn (sklearn): Scikit-learn is a popular machine learning library in Python, providing simple and efficient tools for data mining and analysis. It features various classification, regression, and clustering algorithms, along with tools for model selection and evaluation.
6. Matplotlib: Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. It can be used to generate plots, his-

tograms, power spectra, bar charts, error charts, scatterplots, etc. However, it is not explicitly used in the provided code snippet.

7. **ZipFile:** ZipFile is a module in Python's standard library that provides tools to create, read, write, append, and list a ZIP file. In the code snippet, it's used to extract files from a ZIP archive.
8. **pickle:** Pickle is a module in Python's standard library used for serializing and deserializing Python objects. It's commonly used for saving trained machine learning models to disk.

These libraries and tools are utilized to perform various tasks such as data manipulation, text preprocessing, machine learning model training, evaluation, and sentiment analysis.

4.2 Tools and Technologies

1. **Jupyter Notebook or Google Colab:** Jupyter Notebook is an open-source web application that allows you to create and share documents containing live code, equations, visualizations, and narrative text. Google Colab is a cloud-based version of Jupyter Notebook provided by Google.
2. **Kaggle:** Kaggle is a platform for data science and machine learning competitions, hosting datasets, notebooks, and discussions related to data science. In the provided code, Kaggle is used to download the sentiment analysis dataset.
3. **Python:** Python is a high-level, interpreted programming language known for its simplicity and readability. It is widely used in data science, machine learning, web development, automation, and more. Tools and technologies:
4. **Pandas:** Pandas is used to load the sentiment analysis dataset from a CSV file (`pd.read.csv`). It provides data structures like dataframes, which facilitate data manipulation and analysis.

5. NumPy: NumPy may be used alongside Pandas for numerical operations and array manipulation if necessary, although it's not explicitly mentioned in the provided code snippet.
6. NLTK (Natural Language Toolkit): NLTK is employed for text preprocessing tasks such as stemming (PorterStemmer) and removing stopwords. Stemming reduces words to their root form, and stopwords are common words like "the," "is," "and," etc., which are often removed to focus on meaningful content. Regular Expressions (re module): Regular expressions (re) are used for pattern matching and text manipulation. In this case, re.sub is utilized to remove non-alphabetic characters from the text.
7. TfidfVectorizer (from sklearn.feature extraction.text): TfidfVectorizer is used to convert textual data into numerical feature vectors suitable for machine learning algorithms. It transforms text data into a matrix of TF-IDF features, representing the importance of each word in the document relative to the entire corpus. Model Training and Evaluation:
8. Scikit-learn (sklearn): Scikit-learn is the primary machine learning library used for model training, evaluation, and performance metrics calculation. It provides various classifiers like Logistic Regression, SVM, Random Forest, etc., as well as metrics functions like accuracy score, precision score, recall score, and f1 score.
9. Logistic Regression, SVC, RandomForestClassifier (from sklearn.linear model, sklearn.svm, sklearn.ensemble): These are machine learning algorithms used for sentiment analysis classification tasks. Logistic Regression is a linear classification model, SVC (Support Vector Classifier) is a kernel-based classifier, and RandomForestClassifier is an ensemble learning method based on decision trees.
10. Pickle (pickle module): Pickle is employed to save the trained model (pickle.dump) to disk or load a pre-trained model (pickle.load). It serializes/deserializes Python objects, allowing the model to be reused without retraining.

11. User Input and Prediction: The code snippet demonstrates how to use the trained sentiment analysis model to make predictions on new text data. It defines a function `predict sentiment` that takes user input, preprocesses it, transforms it using the `TfidfVectorizer`, and then predicts the sentiment using the trained model. Overall, these tools and technologies form a comprehensive stack for performing sentiment analysis, encompassing data loading, preprocessing, feature extraction, model training, evaluation, and prediction.

Chapter 5

Results and Analysis

5.1 Performance Evaluation

```
from sklearn.linear_model import SGDClassifier
sgd_clf = SGDClassifier(random_state=2)
sgd_clf.fit(x_train, y_train)
x_train_pred = sgd_clf.predict(x_train)
trd_acc = accuracy_score(y_train, x_train_pred)
print('Accuracy on Training Data:', trd_acc)

x_test_pred = sgd_clf.predict(x_test)
tsd_acc = accuracy_score(y_test, x_test_pred)
print('Accuracy on Test Data:', tsd_acc)

precision = precision_score(y_test, x_test_pred)
recall = recall_score(y_test, x_test_pred)
f1 = f1_score(y_test, x_test_pred)

Accuracy on Training Data: 0.75661875
Accuracy on Test Data: 0.7561770833333333

[ ] print(f'Precision: {precision}')
    print(f'Recall: {recall}')
    print(f'F1 Score: {f1}')

Precision: 0.7271165716692094
Recall: 0.8201541666666666
F1 Score: 0.7708381876959286
```

Figure 5.1: Logistic Regression

```

from sklearn.naive_bayes import MultinomialNB
m2 = MultinomialNB()
m2.fit(x_train, y_train)
x_train_pred = m2.predict(x_train)
trd_acc = accuracy_score(y_train, x_train_pred)
print('Accuracy on Training Data:', trd_acc)
x_test_pred = m2.predict(x_test)
tsd_acc = accuracy_score(y_test, x_test_pred)
print('Accuracy on Test Data:', tsd_acc)

Accuracy on Training Data: 0.8216964285714285
Accuracy on Test Data: 0.7554416666666667

[ ] from sklearn.metrics import precision_score, recall_score, f1_score
precision = precision_score(y_test, x_test_pred)
recall = recall_score(y_test, x_test_pred)
f1 = f1_score(y_test, x_test_pred)

print(f'Precision: {precision}')
print(f'Recall: {recall}')
print(f'F1 Score: {f1}')

Precision: 0.7709082713943561
Recall: 0.7268958333333333
F1 Score: 0.7482554074861998

```

Figure 5.2: Naive Bayes

```

model = LogisticRegression(max_iter=1000)

model.fit(x_train, y_train)

y_train_pred = model.predict(x_train)

y_test_pred = model.predict(x_test)

training_accuracy = accuracy_score(y_train, y_train_pred)

testing_accuracy = accuracy_score(y_test, y_test_pred)

print(f'Training Accuracy: {training_accuracy}')
print(f'Testing Accuracy: {testing_accuracy}')

Training Accuracy: 0.8106169642857143
Testing Accuracy: 0.7776708333333333

[ ] from sklearn.metrics import precision_score, recall_score, f1_score

precision = precision_score(y_test, y_test_pred)
recall = recall_score(y_test, y_test_pred)
f1 = f1_score(y_test, y_test_pred)

print(f'Precision: {precision:.4f}')
print(f'Recall: {recall:.4f}')
print(f'F1 Score: {f1:.4f}')

Precision: 0.7671
Recall: 0.7974
F1 Score: 0.7820

```

Figure 5.3: Stochastic Gradient Descent

5.2 Result

The graphical representation of performance scores of the four machine learning models. The dataset is split into a training set of 80 percent tweets and a testing set of 20 percent tweets to analyze the precision, recall, f1 score, and accuracy of different classification models.

Based on the results, Logistic Regression performs better than the other models in the detection of potential mental health crisis tweets. shows the precision, recall, f1 score,

and accuracy of the four machine learning techniques. Naïve Bayes Classifier has the highest precision compared to the other models. Support Vector Classifier has the lowest accuracy of 69 percent only whereas Stochastic Gradient Descent, Naive Bayes, and Logistic Regression obtained an accuracy of 75.5 percent, 75 percent, and 77 percent respectively. The performance scores indicate that the sentiment analysis model performed well in classifying positive and negative tweets.

```
import numpy as np
import pandas as pd
import re
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

[ ] import nltk
nltk.download('stopwords')
print(stopwords.words('english'))

['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'h
```

Figure 5.4: Result 1

```
[ ] td=pd.read_csv("../content/training.1600000.processed.noemoticon.csv", encoding="ISO-8859-1")

[ ] td.shape

(1599999, 6)

td.head()
```

0	1467818369	Mon Apr 06 22:19:45 PDT 2009	NO_QUERY	_TheSpecialOne_	@switchfoot	http://twitpic.com/2y1z1 - Awww, that's a bummer. You shoulda got David Carr of Third Day to do it. ;D
0	1467810672	Mon Apr 06 22:19:49 PDT 2009	NO_QUERY	scotthamilton		is upset that he can't update his Facebook by ...
1	1467810917	Mon Apr 06 22:19:53 PDT 2009	NO_QUERY	mattyus	@Kenichan	I dived many times for the bail. Man...
2	1467811184	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	ElleCTF		my whole body feels itchy and like its on fire
3	1467811193	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	Karoli	@nationwideclass	no, it's not behaving at all....
4	1467811372	Mon Apr 06 22:20:00 PDT 2009	NO_QUERY	joy_wolf	@Kwesidei	not the whole crew

```
[ ] cn=['target', 'id', 'date', 'flag', 'user', 'text']
td=pd.read_csv("../content/training.1600000.processed.noemoticon.csv", names=cn, encoding="ISO-8859-1")
```

Figure 5.5: Result 2

```
[ ] td['target'].value_counts()
0      800000
4      800000
Name: target, dtype: int64

[ ] td.replace({'target': {4:1}},inplace=True)
td['target'].value_counts()
0      800000
1      800000
Name: target, dtype: int64

[ ] ps=PorterStemmer()

[ ] def stemming(sc):
    sc = re.sub('[^a-zA-Z]', ' ', sc)
    sc = sc.lower()
    sc = sc.split()
    sc = [ps.stem(word) for word in sc if word not in stopwords.words('english')]
    sc = ' '.join(sc)
    return sc

[ ] #td['stemmed_content'] = td['text'].apply(stemming)
```

Figure 5.6: Result 3

```
[ ] import pickle

[ ] td['sc']=pickle.load(open('/content/stemmed_model.sav', 'rb'))

[ ] Start coding or generate with AI.

[ ] print(td['sc'])
0      switchfoot http twitpic com zl awww bummer sho...
1      upset updat facebook text might cri result sch...
2      kenichan dive mani time ball manag save rest g...
3      whole bodi feel itchi like fire
4      nationwideclass behav mad see
...
1599995      woke school best feel ever
1599996      thewdb com cool hear old walt interview http b...
1599997      readi mojo makeov ask detail
1599998      happi th birthday boo all time tupac amaru sh...
1599999      happi charitytuesday thenspcc sparksschar speak...
Name: sc, Length: 1600000, dtype: object

[ ] print(td['target'])
0      0
1      0
2      0
3      0
4      0
```

Figure 5.7: Result 4

```
[ ] x=td['sc'].values

[ ] y=td['target'].values

[ ] print(x)

['switchfoot http twitpic com z1 awww bummer shoulda got david carr third day'
 'upset updat facebook text might cri result school today also blah'
 'kenichan dive mani time ball manag save rest go bound' ...
 'readi mojo makeov ask detail'
 'happi th birthday boo alll time tupac amaru shakur'
 'happi charitytuesday thenspcc sparksschar speakinguph h']

[ ] Start coding or generate with AI.

▶ print(y)

🔍 [0 0 0 ... 1 1 1]

[ ] x_train,x_test,y_train,y_test=train_test_split(x, y, test_size=0.3, stratify=y, random_state=2)

[ ] print(x.shape, x_train.shape, x_test.shape)

(1600000,) (1120000,) (480000,)
```

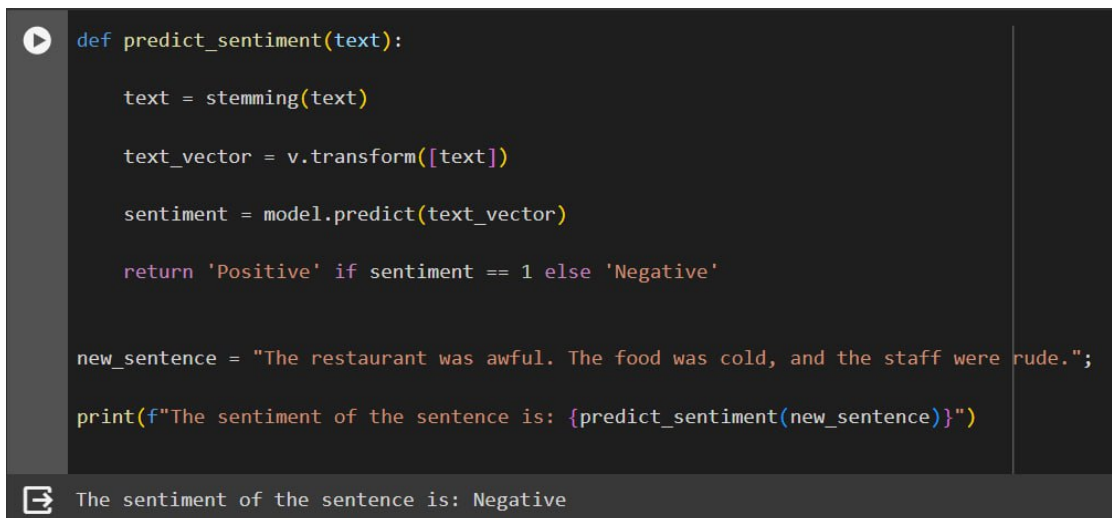
Figure 5.8: Result 5

```
▶ def predict_sentiment(text):
    text = stemming(text)
    text_vector = v.transform([text])
    sentiment = model.predict(text_vector)
    return 'Positive' if sentiment == 1 else 'Negative'

new_sentence = "Movie is Good to watch";
print(f"The sentiment of the sentence is: {predict_sentiment(new_sentence)}")

🔍 The sentiment of the sentence is: Positive
```

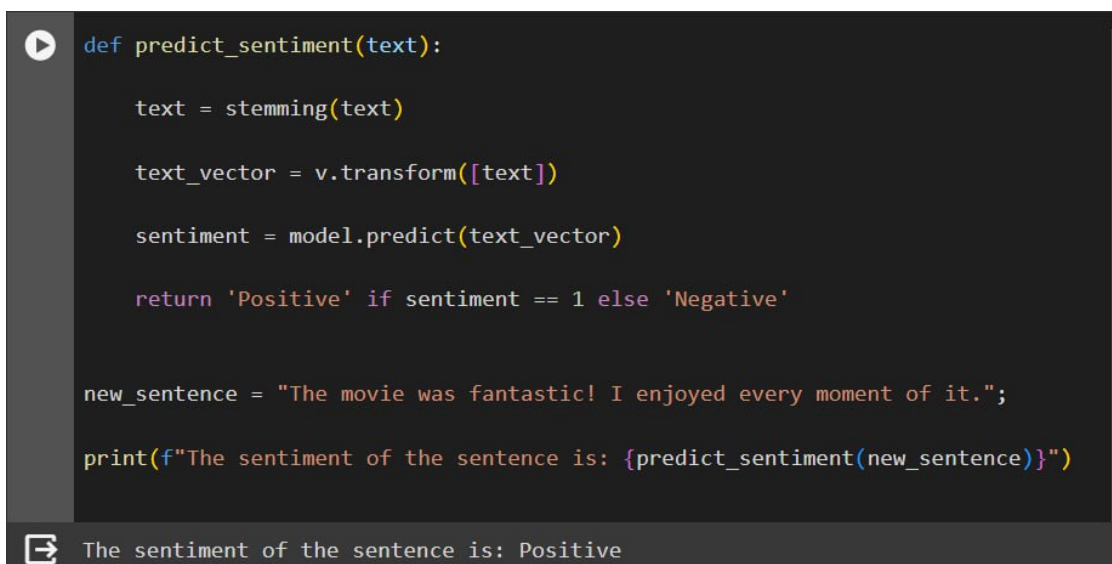
Figure 5.9: Result 6



```
def predict_sentiment(text):  
    text = stemming(text)  
    text_vector = v.transform([text])  
    sentiment = model.predict(text_vector)  
    return 'Positive' if sentiment == 1 else 'Negative'  
  
new_sentence = "The restaurant was awful. The food was cold, and the staff were rude.";  
print(f"The sentiment of the sentence is: {predict_sentiment(new_sentence)}")
```

The sentiment of the sentence is: Negative

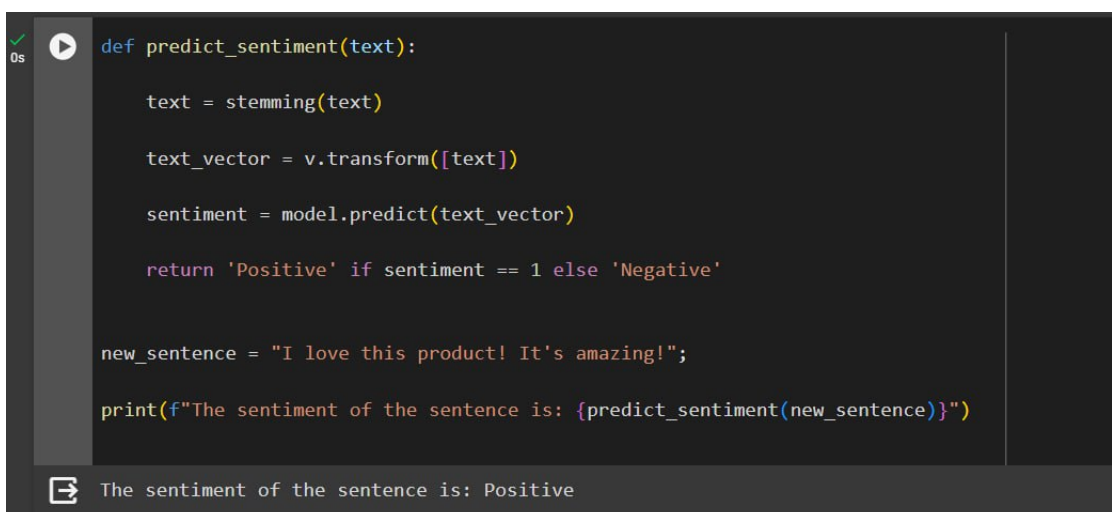
Figure 5.10: Result 7



```
def predict_sentiment(text):  
    text = stemming(text)  
    text_vector = v.transform([text])  
    sentiment = model.predict(text_vector)  
    return 'Positive' if sentiment == 1 else 'Negative'  
  
new_sentence = "The movie was fantastic! I enjoyed every moment of it.";  
print(f"The sentiment of the sentence is: {predict_sentiment(new_sentence)}")
```

The sentiment of the sentence is: Positive

Figure 5.11: Result 8



```
def predict_sentiment(text):  
    text = stemming(text)  
    text_vector = v.transform([text])  
    sentiment = model.predict(text_vector)  
    return 'Positive' if sentiment == 1 else 'Negative'  
  
new_sentence = "I love this product! It's amazing!";  
print(f"The sentiment of the sentence is: {predict_sentiment(new_sentence)}")
```

The sentiment of the sentence is: Positive

Figure 5.12: Result 9

Chapter 6

Conclusion and Future Scope

6.1 Summary of the Project

Twitter sentiment analysis and image captioning tasks presented here demonstrate the power and versatility of machine learning techniques. The sentiment analysis task, which involves preprocessing steps such as tokenization, lemmatization, and stemming, as well as the application of TF-IDF vectorization and model training, allows us to predict whether a given input line carries a positive or negative sentiment. This can be incredibly useful in understanding public opinion on a wide range of topics.

On the other hand, the image captioning task, which employs a CNN-LSTM model with a Visual Attention Mechanism and is evaluated using the CIDEr metric, showcases the ability of such models to generate diverse, contextually relevant captions that align with human judgments. This can be particularly useful in applications such as aiding visually impaired individuals or automating the tagging and categorization of images. These advancements will undoubtedly open up new possibilities and applications, further integrating machine learning into our daily lives.

6.2 Future Scope

Technological developments in natural language processing (NLP) and machine learning promise great things for Twitter sentiment analysis going forward. Sentiment analysis models are positioned to advance in accuracy and nuance as NLP techniques con-

tinue to develop, enabling them to comprehend the minute details and context that are present in tweets. It is anticipated that the use of multimodal analysis—which includes text, photos, emojis, and videos—will enhance sentiment analysis’s depth and provide a more thorough comprehension of user sentiment. Moreover, domain-specific models designed for the financial, political, and medical domains will improve the accuracy and application of sentiment analysis.

Businesses and organizations will be able to use real-time analysis capabilities to rapidly derive insights from twitter data for market estimation, crisis management, and brand monitoring. In future this project can directly integrated to applications or online web-sites to predict the trend of any particular topic by training with data from different social media platforms for better accuracy.

Bibliography

- [1] P. J. M. Loresco, I. C. Valenzuela and E. P. Dadios, “Color space analysis using KNN for lettuce crop stages identification in smart farm setup,” TENCON 2018-2018 IEEE Region 10 Conference , pp. 2040-2044, 2018
- [2] A. . PappuRajan and S. P. Victor, “Web Sentiment Analysis for Scoring Positive or Negative Words using Tweeter Data,” International Journal of Computer Applications, vol. 96, no. 6, pp. 33-37, 2014.
- [3] Carenini, G., Ng, R. and Zwart, E. Extracting Knowledge from Evaluative Text. Proceedings of the Third International Conference on Knowledge Capture (K-CAP’05), 2005.
- [4] W. Gordon, ”Understanding OAuth: What Happens When You Log Into a Site with Google, Twitter, or Facebook,” 2020.
- [5] Bordes A, Glorot X, Weston J, Bengio Y (2014) A semantic matching energy function for learning with multi-relational data. Mach Learn 94(2):233–259.
- [6] A. . PappuRajan and S. P. Victor, “Web Sentiment Analysis for Scoring Positive or Negative Words using Tweeter Data,” International Journal of Computer Applications, vol. 96, no. 6, pp. 33-37, 2014..
- [7] P. e. a. Peduzzi, “A Simulation Study of the Number of Events per Variable in Logistic Regression Analysis,” Journal of Clinical Epidemiology, vol. 49, p. 1373–1379, 1996.
- [8] H. Uysal, A Genetic Programming Approach to Classification Problems, GRIN Verlag, 2016.

- [9] A. U. Aquino, M. E. M. Fernandez, A. P. Guzman, A. A. Matias, I. C. Valenzuela and E. P. Dadios, “ An Artificial Neural Network (ANN) Model for the Cell Density Measurement of Spirulina (A. platensis),” 2018 IEEE 10th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM), pp. 1-5, 2018.
- [10] P. Xu, F. Davoine and T. Denoeux, “Evidential Logistic Regression for Binary SVM Classifier Calibration,” In International Conference on Belief Functions, pp. 49 57, 2014.
- [11] S. M. Kamruzzaman and C. M. Rahman, “Text Categorization using Association Rule and Naive Bayes Classifier,” arXiv: Information Retrieval, vol. , no. , p. , 2010.
- [12] D. e. a. Mittal, “An Effective Hybridized Classifier for Breast Cancer Diagnosis 2015,” IEEE International Conference on Advanced Intelligent Mechatronics (AIM), 2015.
- [13] F. Kabir, S. A. Siddique, M. R. A. Kotwal and M. N. Huda, “Bangla text document categorization using Stochastic Gradient Descent (SGD) classifier,” 2015 International Conference on Cognitive Computing and Information Processing (CCIP), pp. 1-4, 2015.
- [14] Nasukawa, Tetsuya, and Jeonghee Yi. ”Sentiment analysis: Capturing favorability using natural language processing.” In Proceedings of the 2nd international conference on Knowledge capture, ACM, pp. 70-77, 2003.
- [15] P. . Domingos, “A few useful things to know about machine learning,” Communications of The ACM, vol. 55, no. 10, pp. 78-87, 2012.
- [16] A. Aquino, Ma. Veronica Bautista, C. Diaz, I. Valenzuela and E. Dadios, “A Vision-Based Closed Spirulina (A. Platensis) Cultivation System with Growth Monitoring using Artificial Neural Network,” 2018 IEEE 10th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM), pp. 1-5, 2018.

- [17] I. Valenzuela, R. Baldovino, A. Bandala and E. Dadios, “Pre-Harvest Factors Optimization Using Genetic Algorithm for Lettuce,” *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, pp. 1-4, 2018
- [18] H. Saif, Y. He, M. Fernandez and H. Alani, “Semantic Patterns for Sentiment Analysis of Twitter,” *The Semantic Web – ISWC 2014*, Springer International Publishing, p. 324–340, 2014
- [19] G. M. Fung, O. L. Mangasarian and J. W. Shavlik, “Knowledge-Based Support Vector Machine Classifiers,” *Advances in neural information processing systems*, pp. 537-544, 2002.
- [20] G. P.-S. C. M. W.J. Frawley, “Knowledge discovery in databases: An overview,” *Knowledge Discovery in Databases*, pp. 1-27, 1991.

Appendices

Appendix A

Source Code

```
Content 0:
[ ] ! pip install kaggle

Requirement already satisfied: kaggle in /usr/local/lib/python3.10/dist-packages (1.5.16)
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.10/dist-packages (from kaggle) (1.16.0)
Requirement already satisfied: certifi in /usr/local/lib/python3.10/dist-packages (from kaggle) (2024.2.2)
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.10/dist-packages (from kaggle) (2.8.2)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from kaggle) (2.31.0)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from kaggle) (4.66.2)
Requirement already satisfied: python-slugify in /usr/local/lib/python3.10/dist-packages (from kaggle) (8.0.4)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.10/dist-packages (from kaggle) (2.0.7)
Requirement already satisfied: bleach in /usr/local/lib/python3.10/dist-packages (from kaggle) (6.1.0)
Requirement already satisfied: webencodings in /usr/local/lib/python3.10/dist-packages (from bleach->kaggle) (0.5.1)
Requirement already satisfied: text-unidecode>=1.3 in /usr/local/lib/python3.10/dist-packages (from python-slugify->kaggle) (1.3)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->kaggle) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->kaggle) (3.6)

[ ] !mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle/
!chmod 600 ~/.kaggle/kaggle.json

[ ] !kaggle datasets download -d kazanova/sentiment140

Downloading sentiment140.zip to /content
88% 71.0M/80.9M [00:00<00:00, 265MB/s]
100% 80.9M/80.9M [00:00<00:00, 258MB/s]
```

Figure A.1: Implementation 1

```
[ ] from zipfile import ZipFile
    ds='/content/sentiment140.zip'
    with ZipFile(ds,'r') as e:
        e.extractall()
        print("Done")

Done

import numpy as np
import pandas as pd
import re
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
```

Figure A.2: Implementation 2

```
[ ] import nltk
    nltk.download('stopwords')
    print(stopwords.words('english'))

['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'h',
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data] Unzipping corpora/stopwords.zip.

[ ] Start coding or generate with AI.

import numpy as np
import pandas as pd
import re
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

[ ] import nltk
    nltk.download('stopwords')
    print(stopwords.words('english'))

['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'h',
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data] Package stopwords is already up-to-date!
```

Figure A.3: Implementation 3

```
import numpy as np
import pandas as pd
import re
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

[ ] import nltk
    nltk.download('stopwords')
    print(stopwords.words('english'))

['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'h',
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data] Package stopwords is already up-to-date!
```

Figure A.4: Implementation 4


```
[ ] td=pd.read_csv('/content/training.1600000.processed.noemoticon.csv', encoding="ISO-8859-1")

[ ] td.shape

(1599999, 6)

td.head()
0  1467810369  Mon Apr 06 22:19:45 PDT 2009  NO_QUERY  _TheSpecialOne_  @switchfoot http://twitpic.com/2y1z1 - Awww, that's a bummer. You shoulda got David Carr of Third Day to do it. ;D
1  1467810672  Mon Apr 06 22:19:49 PDT 2009  NO_QUERY  scotthamilton  is upset that he can't update his Facebook by ...
2  1467810917  Mon Apr 06 22:19:53 PDT 2009  NO_QUERY  mattycus  @Kenichan I dived many times for the ball. Man...
3  1467811184  Mon Apr 06 22:19:57 PDT 2009  NO_QUERY  ElleCTF  my whole body feels itchy and like its on fire
4  1467811193  Mon Apr 06 22:19:57 PDT 2009  NO_QUERY  Karoli  @nationwideclass no, it's not behaving at all...
5  1467811372  Mon Apr 06 22:20:00 PDT 2009  NO_QUERY  joy_wolf  @Kwesidei not the whole crew

[ ] cn=['target', 'id', 'date', 'flag', 'user', 'text']
td=pd.read_csv('/content/training.1600000.processed.noemoticon.csv', names=cn, encoding="ISO-8859-1")
```

Figure A.5: Implementation 5

```
[ ] td.shape

(1600000, 6)

[ ] td.head()
   target  id  date  flag  user  text
0      0  1467810369  Mon Apr 06 22:19:45 PDT 2009  NO_QUERY  _TheSpecialOne_  @switchfoot http://twitpic.com/2y1z1 - 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...

td.isnull().sum()
target    0
id        0
date      0
flag      0
user      0
text      0
dtype: int64
```

Figure A.6: Implementation 6

```
[ ] td['target'].value_counts()
0    800000
4    800000
Name: target, dtype: int64

td.replace({'target': (4:1)}, inplace=True)
td['target'].value_counts()
0    800000
1    800000
Name: target, dtype: int64

ps=PorterStemmer()

[ ] def stemming(sc):
    sc = re.sub('[^a-zA-Z]', ' ', sc)
    sc = sc.lower()
    sc = sc.split()
    sc = [ps.stem(word) for word in sc if word not in stopwords.words('english')]
    sc = ' '.join(sc)
    return sc

[ ] #td['stemmed_content'] = td['text'].apply(stemming)
```

Figure A.7: Implementation 7

```
[ ] import pickle

[ ] td['sc']=pickle.load(open('/content/stemmed_model.sav', 'rb'))

[ ] Start coding or generate with AI.

[ ] print(td['sc'])

0      switchfoot http twitpic com z1 awww bummer sho...
1      upset updat facebook text might cri result sch...
2      kenichan dive mani time ball manag save rest g...
3      whole bodi feel itchi like fire
4      nationwideclass behav mad see
...
1599995      woke school best feel ever
1599996      thewdb com cool hear old walt interview http b...
1599997      readi mojo makeov ask detail
1599998      happi th birthday boo alll time tupac amaru sh...
1599999      happi charitytuesday thenspcc sparksschar speak...
Name: sc, Length: 1600000, dtype: object

▶ print(td['target'])

0      0
1      0
2      0
3      0
4      0
```

Figure A.8: Implementation 8

```
[ ] x=td['sc'].values

[ ] y=td['target'].values

[ ] print(x)

['switchfoot http twitpic com z1 awww bummer shoulda got david carr third day'
'upset updat facebook text might cri result school today also blah'
'kenichan dive mani time ball manag save rest go bound' ...
'readi mojo makeov ask detail'
'happi th birthday boo alll time tupac amaru shakur'
'happi charitytuesday thenspcc sparksschar speakinguph h']

[ ] Start coding or generate with AI.

▶ print(y)

0 0 0 ... 1 1 1

[ ] x_train,x_test,y_train,y_test=train_test_split(x, y, test_size=0.3, stratify=y, random_state=2)

[ ] print(x.shape, x_train.shape, x_test.shape)

(1600000,) (1120000,) (480000,)
```

Figure A.9: Implementation 9

```
[ ] v=TfidfVectorizer()

[ ] x_train=v.fit_transform(x_train)

[ ] x_test=v.transform(x_test)
```

Figure A.10: Implementation 10

```
▶ model = LogisticRegression(max_iter=1000)

model.fit(x_train, y_train)

y_train_pred = model.predict(x_train)

y_test_pred = model.predict(x_test)

training_accuracy = accuracy_score(y_train, y_train_pred)

testing_accuracy = accuracy_score(y_test, y_test_pred)

print(f'Training Accuracy: {training_accuracy}')
print(f'Testing Accuracy: {testing_accuracy}')
```

📄 Training Accuracy: 0.8106169642857143
Testing Accuracy: 0.7776708333333333

```
[ ] from sklearn.metrics import precision_score, recall_score, f1_score

precision = precision_score(y_test, y_test_pred)
recall = recall_score(y_test, y_test_pred)
f1 = f1_score(y_test, y_test_pred)

print(f"Precision: {precision:.4f}")
print(f"Recall: {recall:.4f}")
print(f"F1 Score: {f1:.4f}")

Precision: 0.7671
Recall: 0.7974
F1 Score: 0.7820
```

Figure A.11: Implementation 11

```
from sklearn.naive_bayes import MultinomialNB
m2 = MultinomialNB()
m2.fit(x_train, y_train)
x_train_pred = m2.predict(x_train)
trd_acc = accuracy_score(y_train, x_train_pred)
print('Accuracy on Training Data:', trd_acc)
x_test_pred = m2.predict(x_test)
tsd_acc = accuracy_score(y_test, x_test_pred)
print('Accuracy on Test Data:', tsd_acc)
```

Accuracy on Training Data: 0.8216964285714285
Accuracy on Test Data: 0.7554416666666667

```
[ ] from sklearn.metrics import precision_score, recall_score, f1_score
precision = precision_score(y_test, x_test_pred)
recall = recall_score(y_test, x_test_pred)
f1 = f1_score(y_test, x_test_pred)

print(f'Precision: {precision}')
print(f'Recall: {recall}')
print(f'F1 Score: {f1}')
```

Precision: 0.7709082713943561
Recall: 0.7268958333333333
F1 Score: 0.7482554074861998

Figure A.12: Implementation 12

```
from sklearn.linear_model import SGDClassifier
sgd_clf = SGDClassifier(random_state=2)
sgd_clf.fit(x_train, y_train)
x_train_pred = sgd_clf.predict(x_train)
trd_acc = accuracy_score(y_train, x_train_pred)
print('Accuracy on Training Data:', trd_acc)

x_test_pred = sgd_clf.predict(x_test)
tsd_acc = accuracy_score(y_test, x_test_pred)
print('Accuracy on Test Data:', tsd_acc)

precision = precision_score(y_test, x_test_pred)
recall = recall_score(y_test, x_test_pred)
f1 = f1_score(y_test, x_test_pred)

Accuracy on Training Data: 0.75661875
Accuracy on Test Data: 0.7561770833333333

[ ] print(f'Precision: {precision}')
    print(f'Recall: {recall}')
    print(f'F1 Score: {f1}')

Precision: 0.7271165716692094
Recall: 0.8201541666666666
F1 Score: 0.7708381876959286
```

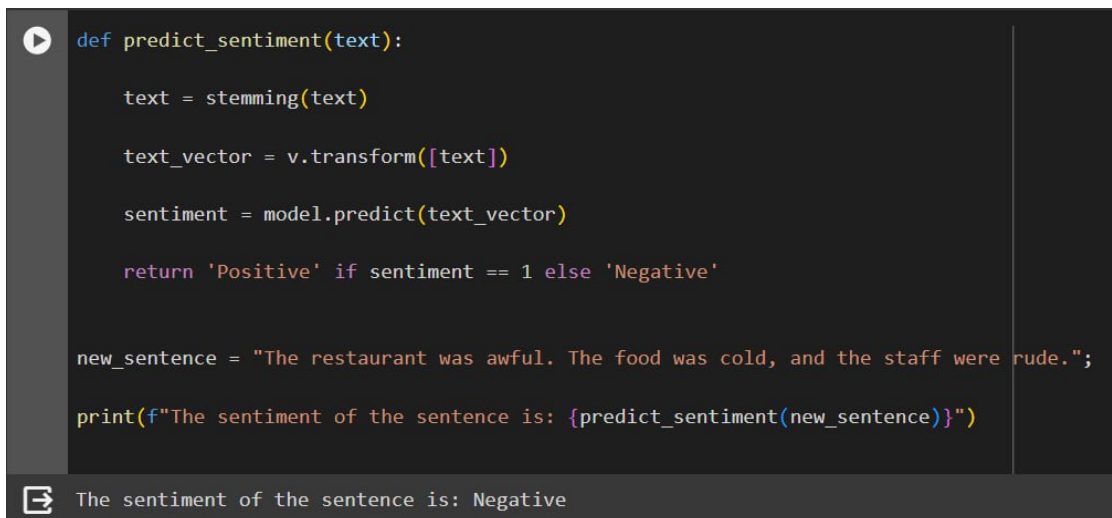
Figure A.13: Implementation 13

```
def predict_sentiment(text):
    text = stemming(text)
    text_vector = v.transform([text])
    sentiment = model.predict(text_vector)
    return 'Positive' if sentiment == 1 else 'Negative'

new_sentence = "Movie is Good to watch";
print(f"The sentiment of the sentence is: {predict_sentiment(new_sentence)}")

The sentiment of the sentence is: Positive
```

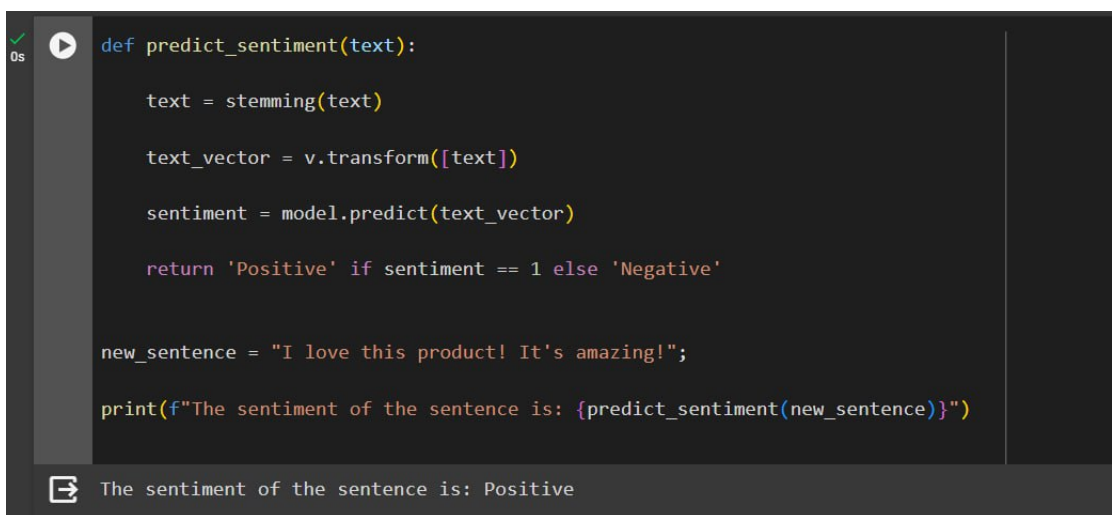
Figure A.14: Implementation 14



```
def predict_sentiment(text):  
  
    text = stemming(text)  
  
    text_vector = v.transform([text])  
  
    sentiment = model.predict(text_vector)  
  
    return 'Positive' if sentiment == 1 else 'Negative'  
  
new_sentence = "The restaurant was awful. The food was cold, and the staff were rude.";  
  
print(f"The sentiment of the sentence is: {predict_sentiment(new_sentence)}")
```

The sentiment of the sentence is: Negative

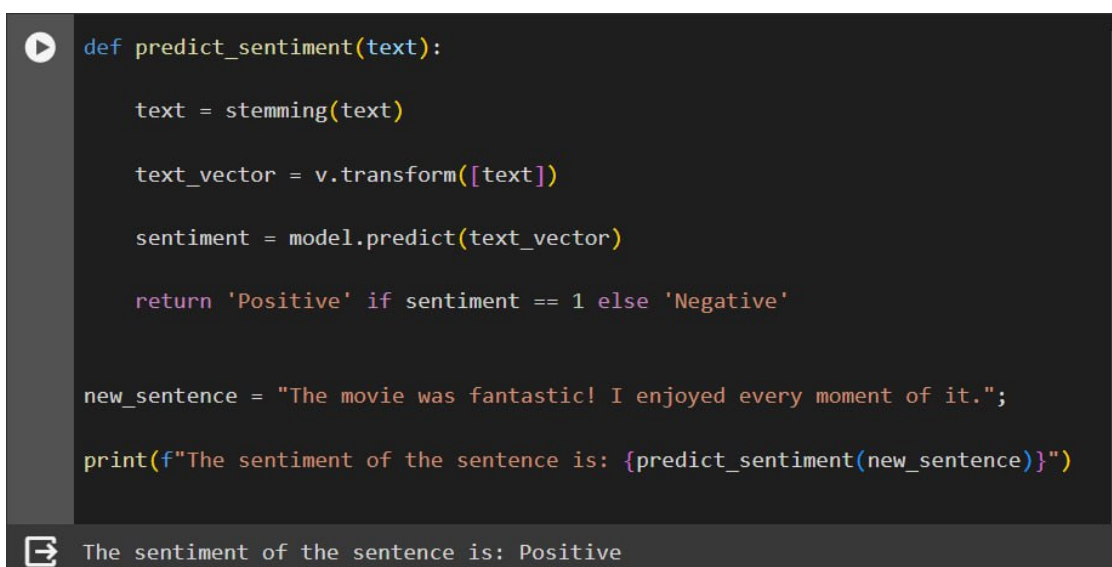
Figure A.15: Implementation 15



```
def predict_sentiment(text):  
  
    text = stemming(text)  
  
    text_vector = v.transform([text])  
  
    sentiment = model.predict(text_vector)  
  
    return 'Positive' if sentiment == 1 else 'Negative'  
  
new_sentence = "I love this product! It's amazing!";  
  
print(f"The sentiment of the sentence is: {predict_sentiment(new_sentence)}")
```

The sentiment of the sentence is: Positive

Figure A.16: Implementation 16



```
def predict_sentiment(text):  
  
    text = stemming(text)  
  
    text_vector = v.transform([text])  
  
    sentiment = model.predict(text_vector)  
  
    return 'Positive' if sentiment == 1 else 'Negative'  
  
new_sentence = "The movie was fantastic! I enjoyed every moment of it.";  
  
print(f"The sentiment of the sentence is: {predict_sentiment(new_sentence)}")
```

The sentiment of the sentence is: Positive

Figure A.17: Implementation 17

Appendix B

Dataset used in the project

1	0	1467810369	Mon Apr 06 22:19:45 F NO_QUERY	_TheSpecialOne_	@switchfoot http://tw
2	0	1467810672	Mon Apr 06 22:19:49 F NO_QUERY	scotthamilton	is upset that he can't u
3	0	1467810917	Mon Apr 06 22:19:53 F NO_QUERY	mattycus	@Kenichan I dived ma
4	0	1467811184	Mon Apr 06 22:19:57 F NO_QUERY	ElleCTF	my whole body feels it
5	0	1467811193	Mon Apr 06 22:19:57 F NO_QUERY	Karoli	@nationwideclass no,
6	0	1467811372	Mon Apr 06 22:20:00 F NO_QUERY	joy_wolf	@Kwesidei not the wh
7	0	1467811592	Mon Apr 06 22:20:03 F NO_QUERY	mybirch	Need a hug
8	0	1467811594	Mon Apr 06 22:20:03 F NO_QUERY	coZZ	@LOLTrish hey long ti
9	0	1467811795	Mon Apr 06 22:20:05 F NO_QUERY	2Hood4Hollywood	@Tatiana_K nope they
10	0	1467812025	Mon Apr 06 22:20:09 F NO_QUERY	mimismo	@twittera que me mu
11	0	1467812416	Mon Apr 06 22:20:16 F NO_QUERY	erinx3leannexo	spring break in plain ci
12	0	1467812579	Mon Apr 06 22:20:17 F NO_QUERY	pardonlauren	I just re-pierced my ea
13	0	1467812723	Mon Apr 06 22:20:19 F NO_QUERY	TLeC	@caregiving I couldn't
14	0	1467812771	Mon Apr 06 22:20:19 F NO_QUERY	robrobblerober	@octolnz16 It it count
15	0	1467812784	Mon Apr 06 22:20:20 F NO_QUERY	bayofwolves	@smarrison i would've
16	0	1467812799	Mon Apr 06 22:20:20 F NO_QUERY	HairByless	@iamjazzfizzle I wish
17	0	1467812964	Mon Apr 06 22:20:22 F NO_QUERY	lovesongwriter	Hollis' death scene wi
18	0	1467813137	Mon Apr 06 22:20:25 F NO_QUERY	armotley	about to file taxes
19	0	1467813579	Mon Apr 06 22:20:31 F NO_QUERY	starkissed	@LettYA ahh i've alway
20	0	1467813782	Mon Apr 06 22:20:34 F NO_QUERY	gl_gi_bee	@FakerPattyPattz Oh
21	0	1467813985	Mon Apr 06 22:20:37 F NO_QUERY	quamvu	@alydesigns i was out
22	0	1467813992	Mon Apr 06 22:20:38 F NO_QUERY	swinspeedx	one of my friend calle
23	0	1467814119	Mon Apr 06 22:20:40 F NO_QUERY	coolodoc	@angry_barista I bake
24	0	1467814180	Mon Apr 06 22:20:40 F NO_QUERY	vILLante	this week is not going
25	0	1467814192	Mon Apr 06 22:20:41 F NO_QUERY	Ljelli3166	blagh class at 8 tomor
26	0	1467814438	Mon Apr 06 22:20:44 F NO_QUERY	ChicagoCubbie	I hate when I have to c

Figure B.1: An image of dataset with negative tweets

1048551	4	1960185400	Fri May 29 07:33:37 P NO_QUERY	stevey88	@GillDeCosemo Than
1048552	4	1960185409	Fri May 29 07:33:37 P NO_QUERY	BeckySmithster	and this http://bit.ly/1
1048553	4	1960185431	Fri May 29 07:33:38 P NO_QUERY	mbreinolt	@ItsJustDI
1048554	4	1960185444	Fri May 29 07:33:38 P NO_QUERY	JACI_BEATON	Holy Only 12 fuckin d
1048555	4	1960185457	Fri May 29 07:33:38 P NO_QUERY	alyari	@sonnycentral Thank
1048556	4	1960185460	Fri May 29 07:33:38 P NO_QUERY	CelticsFan27	@JonathanRKnight Yo
1048557	4	1960185466	Fri May 29 07:33:38 P NO_QUERY	flamingokitty	I'm eating cheezits...w
1048558	4	1960185467	Fri May 29 07:33:38 P NO_QUERY	KIKI804	TGIF snitches!.....kids
1048559	4	1960185479	Fri May 29 07:33:38 P NO_QUERY	MoRgAnGILeY	Going to class till noor
1048560	4	1960185519	Fri May 29 07:33:38 P NO_QUERY	zombie_joe	#followfriday because
1048561	4	1960185613	Fri May 29 07:33:39 P NO_QUERY	Decembersown21	watchin Espn's First T
1048562	4	1960185622	Fri May 29 07:33:39 P NO_QUERY	barb1686	is showering. Then off
1048563	4	1960185633	Fri May 29 07:33:39 P NO_QUERY	quiz_master	Muhaha! ThankGoodr
1048564	4	1960185647	Fri May 29 07:33:39 P NO_QUERY	Nicoledyan	Good morning people
1048565	4	1960185684	Fri May 29 07:33:39 P NO_QUERY	Oliviaaemly	Just sitting in the gard
1048566	4	1960185706	Fri May 29 07:33:39 P NO_QUERY	withLoveJazzy	in tampa... going to se
1048567	4	1960185825	Fri May 29 07:33:40 P NO_QUERY	ahousley	Looking forward to a r
1048568	4	1960185836	Fri May 29 07:33:40 P NO_QUERY	WindmillBrixton	GRINGO STAR tonight.
1048569	4	1960185847	Fri May 29 07:33:40 P NO_QUERY	techcoach	@DavidBass hee hee.
1048570	4	1960185945	Fri May 29 07:33:41 P NO_QUERY	timg888	today's message in the
1048571	4	1960186091	Fri May 29 07:33:42 P NO_QUERY	debraladiva	Back home, thought I
1048572	4	1960186342	Fri May 29 07:33:44 P NO_QUERY	Madlinedugganx	My GrandMa is making
1048573	4	1960186409	Fri May 29 07:33:43 P NO_QUERY	OffRoad_Dude	Mid-morning snack tin
1048574	4	1960186429	Fri May 29 07:33:44 P NO_QUERY	Falchion	@ShaDeLa same here
1048575	4	1960186445	Fri May 29 07:33:44 P NO_QUERY	jonasobsessedx	@DestinyHope92 im g
1048576	4	1960186607	Fri May 29 07:33:45 P NO_QUERY	sugabababz	cant wait til her date t

Figure B.2: An image of dataset with positive tweets