

# **SENTIMENT ANALYSIS**

A Project Report

In partial fulfilment of the requirements for the award of the degree

**Bachelor of Technology**

In

**Electronics & Communications Engineering**

Under the guidance of

Joyjit Guha Biswas

by

Shreya Mishra , Arpan Biswas

Shibam Mishra , Subhrajyoti Mandal

Sushil Pal & Antarip Manna

**Hooghly Engineering & Technology  
College**

In association with



(ISO9001:2015)

SDF Building, Module #132, Ground Floor, Salt Lake City, GP Block, Sector V,  
Kolkata, West Bengal 700091

Title of the Project: Sentiment Analysis

Project Members:

Shreya Mishra  
Arpan Biswas  
Shibam Mishra  
Subhrajyoti Mandal  
Sushil Pal  
Antarip Manna

Name of the guide: Mr. Joyjit Guha Biswas

Address: Ardent Computech Pvt. Ltd  
(An ISO 9001:2015 Certified)  
SDF Building, Module #132, Ground Floor,  
Salt Lake City, GP Block, Sector V,  
Kolkata, West Bengal, 700091

Project Version Control History:

Version	Primary Author	Description of Version	Date Completed
Final	Shreya Mishra, Arpan Biswas, Shibam Mishra, Subhrajyoti Mandal, Sushil Pal, Antarip Manna.	Project Report	26 <sup>th</sup> July, 2024

Signature of Team Members

\_\_\_\_\_  
Signature of Approver

Date:

For Office Use Only

Date:

Mr. Joyjit Guha Biswas  
Project Proposal Evaluator

## Declaration

We hereby declare that the project work being presented in the project proposal entitled “**Sentiment Analysis**” in partial fulfilment of the requirements for the award of the degree of Bachelor of Computer Applications at Ardent Computech PVT. LTD, Saltlake, Kolkata, West Bengal, is an authentic work carried out under the guidance of Mr. Joyjit Guha Biswas. The matter embodied in this project work has not been submitted elsewhere for the award of any degree of our knowledge and belief.

Date: 26/07/2024

Name of the Students: -

Shreya Mishra

Arpan Biswas

Shibam Mishra

Subhrajyoti Mandal

Sushil Pal

Antarip Manna

Signature of the students:



Ardent Computech Pvt. Ltd (An ISO 9001:2015 Certified)

SDF Building, Module #132, Ground Floor, Salt Lake City, GP Block, Sector V,  
Kolkata, West Bengal 700091

## Certificate

This is to certify that this proposal of minor project entitled “**Sentiment Analysis**” is a record of Bonafide work, carried out by Sanket Dey under my guidance at Ardent Computech PVT. LTD. In my opinion, the report in its present form is in partial fulfillment of the requirements for the award of the degree of Bachelor of Computer Applications and as per regulations of the Ardent®. To the best of my knowledge, the results embodied in this report, are original in nature and worthy of incorporation in the present version of the report.

Guide / Supervisor

---

Mr. Joyjit Guha Biswas

Project Engineer

Ardent Computech Pvt. Ltd (An ISO 9001:2015 Certified)

SDF Building, Module #132, Ground Floor, Salt Lake City, GP Block, Sector V,  
Kolkata, West Bengal 700091

## Acknowledgement

Success of any project depends largely on the encouragement and guidelines of many others. I take this sincere opportunity to express my gratitude to the people who have been instrumental in the successful completion of this project work.

I would like to show our greatest appreciation to Mr. Joyjit Guha Biswas, Project Engineer at Ardent, Kolkata. I always feel motivated and encouraged every time by his valuable advice and constant inspiration; without his encouragement and guidance this project would not have materialized.

Words are inadequate in offering our thanks to the other trainees, project assistants and other members at Ardent Computech Pvt. Ltd. for their encouragement and cooperation in carrying out this project work. The guidance and support received from all the members and who are contributing to this project, was vital for the success of this project.

## Contents:

- Overview
- History of Python
- Environment Setup
- Basic Syntax
- Variable Types
- Functions
- Modules
- Packages
- Artificial Intelligence
  - Deep Learning
  - Neural Networks
  - Machine Learning
- Machine Learning
  - Supervised and Unsupervised Learning
  - NumPy
  - SciPy
  - Scikit-learn
  - Pandas
  - Regression Analysis
  - Matplotlib
  - Clustering

# Sentiment Analysis

## Overview:

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and has fewer syntactical constructions than other languages.

**Python is interpreted:** Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to Perl and PHP.

**Python is Interactive:** You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

**Python is Object-Oriented:** Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

**Python is a Beginner's Language:** Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

## History of Python:

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands. Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, Small Talk, UNIX shell, and other scripting languages. Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL). Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

## Features of Python:

**Easy-to-learn:** Python has few Keywords, simple structure and clearly defined syntax. This allows a student to pick up the language quickly.

**Easy-to-Read:** Python code is more clearly defined and visible to the eyes.  
**Easy -to-Maintain:** Python's source code is fairly easy-to-maintain.

**A broad standard library:** Python's bulk of the library is very portable and cross platform compatible on UNIX, Windows, and Macintosh.

**Interactive Mode:** Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.

**Portable:** Python can run on the wide variety of hardware platforms and has the same interface on all platforms.

**Extendable:** You can add low level modules to the python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.

**Databases:** Python provides interfaces to all major commercial databases.

**GUI Programming:** Python supports GUI applications that can be created and ported to many system calls, libraries, and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

**Scalable:** Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below:

- It support functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte code for building large applications.
- It provides very high level dynamic data types and supports dynamic type checking.
- It supports automatic garbage collections.
- It can be easily integrated with C, C++, COM, Active X, CORBA and JAVA.



## **Environment Setup:**

- 64-Bit OS
- Install Python 3
- Setup virtual environment
- Install Packages

## **Basic Syntax of Python Program:**

Type the following text at the Python prompt and press the Enter –

```
>>> print "Hello, Python!"
```

If you are running new version of Python, then you would need to use print statement with parenthesis as in print ("Hello, Python!");.

However in Python version 2.4.3, this produces the following result –

Hello, Python!

### **Python Identifiers:**

A Python identifier is a name used to identify a variable, function, class, module or other object. An identifier starts with a letter A to Z or a to z or an underscore (\_) followed by zero or more letters, underscores and digits (0 to 9). Python does not allow punctuation characters such as @, \$, and % within identifiers. Python is a case sensitive programming language.

### **Python Keywords:**

The following list shows the Python keywords. These are reserved words and you cannot use them as constant or variable or any other identifier names. All the Python keywords contain lowercase letters only.

For example:

And, exec, not, Assert, finally, or, break, for, pass, class, from, print, continue, global, raise, def, if, return, del, import, try, elif, in, while, else, is, with, except, lambda, yield.

## Lines & Indentation:

Python provides no braces to indicate blocks of code for class and function definitions or flow control. Blocks of code are denoted by line indentation, which is rigidly enforced.

The number of spaces in the indentation is variable, but all statements within the block must be indented the same amount. For example –

```
if True: print "True"
```

```
else:
```

```
print "False"
```

## Command Line Arguments:

Many programs can be run to provide you with some basic information about how they should be run. Python enables you to do this with -h –

```
$ python-h
```

```
usage: python [option]...[-c cmd|-m mod | file |-][arg]...
```

Options and arguments (and corresponding environment variables):

-c cmd: program passed in as string(terminates option list)

-d : debug output from parser (also PYTHONDEBUG=x)

-E : ignore environment variables (such as PYTHONPATH)

-h : print this help message and exit [ etc.]

## Variable Types:

Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory.

## Assigning Values to Variables:

Python variables do not need explicit declaration to reserve memory space. The declaration happens automatically when you assign a value to a variable. The equal sign (=) is used to assign values to variables.

```
counter=10          # An integer assignment
weight=10.60        # A floating point
name="Ardent"      # A string
```

### Multiple Assignment:

Python allows you to assign a single value to several variables simultaneously.

For example –

```
a = b = c = 1
```

```
a,b,c      =      1,2,"hello"
```

### Standard Data Types:

The data stored in memory can be of many types. For example, a person's age is stored as a numeric value and his or her address is stored as alphanumeric characters. Python has five standard data types –

- String

- List

- Tuple

- Dictionary

- Number

### Data Type Conversion:

Sometimes, you may need to perform conversions between the built-in types.

To convert between types, you simply use the type name as a function.

There are several built-in functions to perform conversion from one data type to another.

Sr.No.	Function & Description
1	<b>int(x [,base])</b> Converts x to an integer. base specifies the base if x is a string
2	<b>long(x [,base] )</b> Converts x to a long integer. base specifies the base if x is a string.
3	<b>float(x)</b> Converts x to a floating-point number.
4	<b>complex(real [,imag])</b> Creates a complex number.
5	<b>str(x)</b> Converts object x to a string representation.
6	<b>repr(x)</b> Converts object x to an expression string.
7	<b>eval(str)</b> Evaluates a string and returns an object.
8	<b>tuple(s)</b> Converts s to a tuple.
9	<b>list(s)</b> Converts s to a list.

# Functions:

## Defining a Function:

- `def functionname( parameters ):`  
    `"function_docstring"`  
    `function_suite`  
    `return [expression]`

## Pass by reference vs Pass by value:

All parameters (arguments) in the Python language are passed by reference. It means if you change what a parameter refers to within a function, the change also reflects back in the calling function. For example –

# Function definition is here

```
def changeme(mylist):  
    "This changes a passed list into this function"  
    mylist.append([1,2,3,4]);  
    print"Values inside the function: ",mylist  
    return
```

# Now you can call changeme function

```
mylist=[10,20,30];  
changeme(mylist);  
print"Values outside the function: ",mylist
```

Here, we are maintaining reference of the passed object and appending values in the same object. So, this would produce the following result –

Values inside the function: [10, 20, 30, [1, 2, 3, 4]]

Values outside the function: [10, 20, 30, [1, 2, 3, 4]]

## Global vs. Local variables

Variables that are defined inside a function body have a local scope, and those defined outside have a global scope . For Example-

```
total=0;           # This is global variable.
```

# Function definition is here

```
def sum( arg1, arg2 ):
```

```
# Add both the parameters and return them."
```

```
total= arg1 + arg2;          # Here total is local variable.  
print"Inside the function local total : ", total  
return total;
```

```
# Now you can call sum functionsum(10,20);  
print"Outside the function global total : ", total
```

When the above code is executed, it produces the following result –

```
Inside the function local total : 30  
Outside the function global total : 0
```

## **Modules:**

A module allows you to logically organize your Python code. Grouping related code into a module makes the code easier to understand and use. A module is a Python object with arbitrarily named attributes that you can bind and reference .

The Python code for a module named `aname` normally resides in a file named `aname.py`. Here's an example of a simple module, `support.py`

```
def print_func( par ):  
    print"Hello : ",  
    par return
```

### **The import Statement**

You can use any Python source file as a module by executing an import statement in some other Python source file. The import has the following syntax –

```
import module1[, module2[,... moduleN]
```

## **Packages:**

A package is a hierarchical file directory structure that defines a single Python application environment that consists of modules and sub packages and sub-subpackages, and so on.

Consider a file Pots.py available in Phone directory. This file has following line of source code –

```
def Pots():  
    print "I'm Pots Phone"
```

Similar way, we have another two files having different functions with the same name as above –

Phone/Isdn.py file having function Isdn()

Phone/G3.py file having function G3()

Now, create one more file \_init\_.py in Phone directory –

Phone/\_init\_.py

To make all of your functions available when you've imported Phone, you need to put explicit import statements in \_init\_.py as follows –

```
from Pots import Pots  
from Isdn import Isdn  
from G3 import
```

## **Numpy:**

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. The ancestor of NumPy, Numeric, was originally created by Jim Hugunin.

NumPy targets the CPython reference implementation of Python, which is a non-optimizing bytecode interpreter. Mathematical algorithms written for this version of Python often run much slower than compiled equivalents.

Using NumPy in Python gives functionality comparable to MATLAB since they are both interpreted, and they both allow the user to write fast programs as long as most operations work on arrays or matrices instead of scalars.

## Scipy:

Scientific computing tools for Python

SciPy refers to several related but distinct entities:

- The SciPy ecosystem, a collection of open source software for scientific computing in Python.
- The community of people who use and develop this stack.
- Several conferences dedicated to scientific computing in Python - SciPy, EuroSciPy, and SciPy.in.
- The SciPy library, one component of the SciPy stack, providing many numerical routines.

The SciPy ecosystem

Scientific computing in Python builds upon a small core of packages:

- Python, a general purpose programming language. It is interpreted and dynamically typed and is very well suited for interactive work and quick prototyping, while being powerful enough to write large applications in.
- NumPy, the fundamental package for numerical computation. It defines the numerical array and matrix types and basic operations on them.
- The SciPy library, a collection of numerical algorithms and domain-specific toolboxes, including signal processing, optimization, statistics, and much more.
- Matplotlib, a mature and popular plotting package that provides publication-quality 2-D plotting, as well as rudimentary 3-D plotting.

On this base, the SciPy ecosystem includes general and specialised tools for data management and computation, productive experimentation, and high-performance computing. Below, we overview some key packages, though there are many more relevant packages.

Data and computation:

- pandas, providing high-performance, easy-to-use data structures.
- SymPy, for symbolic mathematics and computer algebra.
- NetworkX, is a collection of tools for analyzing complex networks.
- scikit-image is a collection of algorithms for image processing.
- scikit-learn is a collection of algorithms and tools for machine learning.
- h5py and PyTables can both access data stored in the HDF5 format.



## Productivity and high-performance computing:

- IPython, a rich interactive interface, letting you quickly process data and test ideas.
- The Jupyter notebook provides IPython functionality and more in your web browser, allowing you to document your computation in an easily reproducible form.
- Cython extends Python syntax so that you can conveniently build C extensions, either to speed up critical code or to integrate with C/C++ libraries.
- Dask, Joblib or IPyParallel for distributed processing with a focus on numeric data.

## Quality assurance:

- nose, a framework for testing Python code, being phased out in preference for pytest.
- numpydoc, a standard and library for documenting Scientific Python libraries.

## **Pandas:**

In computer programming, pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license. "Panel data", an econometrics term for multidimensional, structured data sets.

## Library features:

- Data Frame object for data manipulation with integrated indexing.
- Tools for reading and writing data between in-memory data structures and different file formats.
- Data alignment and integrated handling of missing data.
- Reshaping and pivoting of data sets.
- Label-based slicing, fancy indexing, and sub setting of large data sets.
- Data structure column insertion and deletion.
- Group by engine allowing split-apply-combine operations on data sets.
- Data set merging and joining.

- Hierarchical axis indexing to work with high-dimensional data in a lower-dimensional data structure.
- Time series-functionality: Date range generation.

## Python Speech Features:

This library provides common speech features for ASR including MFCCs and filter bank energies. If you are not sure what MFCCs are, and would like to know more have a look at this MFCC

tutorial: <http://www.practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/>.

You will need numpy and scipy to run these files. The code for this project is available at [https://github.com/jameslyons/python\\_speech\\_features](https://github.com/jameslyons/python_speech_features) .

Supported features:

- `python_speech_features.mfcc()` - Mel Frequency Cepstral Coefficients
- `python_speech_features.fbank()` - Filterbank Energies
- `python_speech_features.logfbank()` - Log Filterbank Energies
- `python_speech_features.ssc()` - Spectral Subband Centroids

To use MFCC features:

```
from python_speech_features import mfcc
from python_speech_features import logfbank
import scipy.io.wavfile as wav
```

```
(rate,sig) = wav.read("file.wav")
```

```
mfcc_feat = mfcc(sig,rate)
fbank_feat = logfbank(sig,rate)
```

```
print(fbank_feat[1:3,:])
```

## **OS**: Miscellaneous operating system interfaces

This module provides a portable way of using operating system dependent functionality. If you just want to read or write a file see `open()`, if you want to manipulate paths, see the `os.path` module, and if you want to read all the lines in all the files on the command line see the file input module. For creating temporary files and directories see the temp file module, and for high-level file and directory handling see the `shutil` module.

### Notes on the availability of these functions:

The design of all built-in operating system dependent modules of Python is such that as long as the same functionality is available, it uses the same interface; for example, the function `os.stat(path)` returns stat information about path in the same format (which happens to have originated with the POSIX interface).

Extensions peculiar to a particular operating system are also available through the `os`-module, but using them is of course a threat to portability.

All functions accepting path or file names accept both bytes and string objects, and result in an object of the same type, if a path or file name is returned.

On VxWorks, `os.fork`, `os.execv` and `os.spawn*p*` are not supported.

## **Pickle**: Python object serialization

The `pickle` module implements binary protocols for serializing and de-serializing a Python object structure. “*Pickling*” is the process whereby a Python object hierarchy is converted into a byte stream, and “*unpickling*” is the inverse operation, whereby a byte stream (from a binary file or bytes-like object) is converted back into an object hierarchy. Pickling (and unpickling) is alternatively known as “serialization”, “marshalling,” 1 or “flattening”; however, to avoid confusion, the terms used here are “pickling” and “unpickling”.

## **Operator**: Standard operators as functions

The operator module exports a set of efficient functions corresponding to the intrinsic operators of Python. For example, `operator.add(x, y)` is equivalent to the expression `x+y`. Many function names are those used for special methods, without the double underscores. For backward compatibility, many of these have a variant with the double underscores kept. The variants without the double underscores are preferred for clarity.

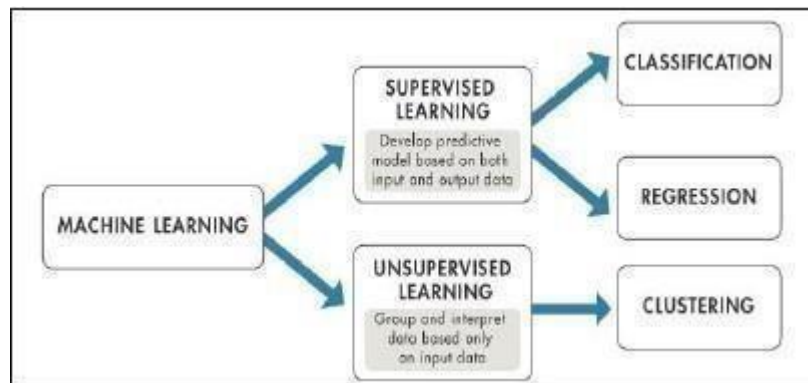
The functions fall into categories that perform object comparisons, logical operations, mathematical operations and sequence operations.

## **Tempfile**: Generate temporary files and directories

This module creates temporary files and directories. It works on all supported platforms. Temporary File, Named Temporary File, Temporary Directory, and Spooled Temporary File are high-level interfaces which provide automatic cleanup and can be used as context managers. `mkstemp()` and `mkdtemp()` are lower-level functions which require manual cleanup.

All the user-callable functions and constructors take additional arguments which allow direct control over the location and name of temporary files and directories. Files names used by this module include a string of random characters which allows those files to be securely created in shared temporary directories. To maintain backward compatibility, the argument order is somewhat odd; it is recommended to use keyword arguments for clarity.

# Introduction to Machine Learning:



Machine learning is a field of computer science that gives computers the ability to learn without being explicitly programmed.

Evolved from the study of pattern recognition and computational learning theory in artificial intelligence, machine learning explores the study and construction of algorithms that can learn from and make predictions on data.

Machine learning is a field of computer science that gives computers the ability to learn without being explicitly programmed.

Arthur Samuel, an American pioneer in the field of computer gaming and artificial intelligence, coined the term "Machine Learning" in 1959 while at IBM. Evolved from the study of pattern recognition and computational learning theory in artificial intelligence, machine learning explores the study and construction of algorithms that can learn from and make predictions on data

Machine learning tasks are typically classified into two broad categories, depending on whether there is a learning "signal" or "feedback" available to a learning system:-

## Supervised Learning:

Supervised learning is the machine learning task of inferring a function from labeled training data.<sup>[1]</sup> The training data consist of a set of training examples. In supervised learning, each example is a pair consisting of an input object (typically a vector) and a desired output value.

A supervised learning algorithm analyses the training data and produces an inferred function, which can be used for mapping new examples. An optimal

scenario will allow for the algorithm to correctly determine the class labels for unseen instances. This requires the learning algorithm to generalize from the training data to unseen situations in a "reasonable" way.

## **Unsupervised Learning:**

Unsupervised learning is the machine learning task of inferring a function to describe hidden structure from "un labelled" data (a classification or categorization is not included in the observations). Since the examples given to the learner are un labelled, there is no evaluation of the accuracy of the structure that is output by the relevant algorithm—which is one way of distinguishing unsupervised learning from supervised learning and reinforcement learning.

A central case of unsupervised learning is the problem of density estimation in statistics, though unsupervised learning encompasses many other problems (and solutions) involving summarizing and explaining key features of the data.

## **Logistic Regression Algorithm:**

Logistic regression is a supervised machine learning algorithm that accomplishes binary classification tasks by predicting the probability of an outcome, event, or observation. The model delivers a binary or dichotomous outcome limited to two possible outcomes: yes/no, 0/1, or true/false.

Logical regression analyzes the relationship between one or more independent variables and classifies data into discrete classes. It is extensively used in predictive modeling, where the model estimates the mathematical probability of whether an instance belongs to a specific category or not.

For example, 0 – represents a negative class; 1 – represents a positive class. Logistic regression is commonly used in binary classification problems where the outcome variable reveals either of the two categories (0 and 1).

## **Advantages of Logistic Regression**

- 1. Easier to implement machine learning methods:** A machine learning model can be effectively set up with the help of training and testing. The training identifies patterns in the input data (image) and associates them with some form of output (label). Training a logistic model with a regression algorithm does not demand higher computational power. As such, logistic regression is easier to implement, interpret, and train than other ML methods.
- 2. Suitable for linearly separable datasets:** A linearly separable dataset refers to a graph where a straight line separates the two data classes. In logistic regression, the y variable takes only two values. Hence, one can effectively classify data into two separate classes if linearly separable data is used.
- 3. Identifying spam emails:** Email inboxes are filtered to determine if the email communication is promotional/spam by understanding the predictor variables and applying a logistic regression algorithm to check its authenticity.

### **Algorithm :**

- Data Collection
- Data Formatting
- Model Selection
- Training
- Testing

**Data Collection:** We have collected data sets of movies from online website. We have downloaded the .csv files in which information was present.

**Data Formatting:** The collected data is formatted into suitable data sets.

**Model Selection:** We have selected different models to minimize the error of the predicted value. The different models used are Linear Regression Linear Model.

**Training:** The data sets was divided such that x-train is used to train the model with corresponding x-test values and some y-train kept reserved for testing.

**Testing:** The model was tested with y-train and stored in y-predict. Both y-train and y-predict was compared.

# Sentiment Analysis

## Important Libraries

```
import pandas as pd
import numpy as np
import nltk
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from nltk.stem import WordNetLemmatizer
import re
from sklearn.model_selection import train_test_split
from nltk.corpus import stopwords
nltk.download('stopwords')
from nltk.corpus import wordnet
nltk.download('wordnet')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
True
```

## Read the CSV file

```
[2] dataset = pd.read_csv('Sentiment.csv')
```

dataset


	id	candidate	candidate_confidence	relevant_yn	relevant_yn_confidence	sentiment	sentiment_confidence	subject_matter	subject_matter_confidence
	1	No candidate mentioned	1.0000	yes	1.0000	Neutral	0.6578	None of the above	1.0000
	2	Scott Walker	1.0000	yes	1.0000	Positive	0.6333	None of the above	1.0000
	3	No candidate mentioned	1.0000	yes	1.0000	Neutral	0.6629	None of the above	0.6629
	4	No candidate mentioned	1.0000	yes	1.0000	Positive	1.0000	None of the above	0.7039
	5	Donald Trump	1.0000	yes	1.0000	Positive	0.7045	None of the above	1.0000
	...	...	...	...	...	...	...	...	...
	13867	No candidate mentioned	1.0000	yes	1.0000	Negative	0.7991	Abortion	0.6014



13868	Mike Huckabee	0.9611	yes	1.0000	Positive	0.7302	None of the above	0.9229
13869	Ted Cruz	1.0000	yes	1.0000	Positive	0.8051	None of the above	0.9647
13870	Donald Trump	1.0000	yes	1.0000	Negative	1.0000	Women's Issues (not abortion though)	0.9202
13871	Ted Cruz	0.9242	yes	0.9614	Positive	0.9614	None of the above	0.9242

ows x 21 columns

```
[4] dataset = dataset[['text', 'sentiment']]
```

 dataset



		text	sentiment
0	RT @NancyLeeGrah: How did everyone feel about...		Neutral
1	RT @ScottWalker: Didn't catch the full #GOPdeb...		Positive
2	RT @TJMShow: No mention of Tamir Rice and the ...		Neutral
3	RT @RobGeorge: That Carly Fiorina is trending ...		Positive
4	RT @DanScavino: #GOPDebate w/ @realDonaldTrump...		Positive
...		...	...
13866	RT @cappy_yarbrough: Love to see men who will ...		Negative
13867	RT @georgehenryw: Who thought Huckabee exceede...		Positive
13868	RT @Lrihendry: #TedCruz As President, I will a...		Positive
13869	RT @JRehling: #GOPDebate Donald Trump says tha...		Negative
13870	RT @Lrihendry: #TedCruz headed into the Presid...		Positive

13871 rows x 2 columns

## Remove all neutral analysis

```
[6] dataset = dataset[dataset['sentiment']!= 'Neutral']
```



dataset




	text	sentiment
1	RT @ScottWalker: Didn't catch the full #GOPdeb...	Positive
3	RT @RobGeorge: That Carly Fiorina is trending ...	Positive
4	RT @DanScavino: #GOPDebate w/ @realDonaldTrump...	Positive
5	RT @GregAbbott_TX: @TedCruz: "On my first day ...	Positive
6	RT @warriorwoman91: I liked her and was happy ...	Negative
...	...	...
13866	RT @cappy_yarbrough: Love to see men who will ...	Negative
13867	RT @georgehenryw: Who thought Huckabee exceede...	Positive
13868	RT @Lrihendry: #TedCruz As President, I will a...	Positive
13869	RT @JRehling: #GOPDebate Donald Trump says tha...	Negative
13870	RT @Lrihendry: #TedCruz headed into the Presid...	Positive

10729 rows x 2 columns

## Data Preprocessing

```
[8] train,test = train_test_split(dataset,test_size=0.1)
```

 train



	text	sentiment
75	RT @Franklin_Graham: 1st #GOPDebate--Encouragi...	Positive
2751	RT @VishalDisawar: .@realDonaldTrump - "We nee...	Negative
12432	RT @RWSurferGirl: Fox News is obviously trying...	Negative
5833	How much credibility did #FoxNews lose last ni...	Negative
9816	RT @RWSurferGirl: Is it just me or does anyone...	Negative
...	...	...
10881	Pataki is Pro-Choice and for me he is disquali...	Negative
520	It's not a coincidence that you couldn't watch...	Negative
11471	RT @RWSurferGirl: Is it just me or does anyone...	Negative
7608	I have been trolling every debate poll I see a...	Negative
6860	I'm disappointed that #MegynKelly never bother...	Negative

9656 rows x 2 columns

test



	text	sentiment
12120	RT @RWSurferGirl: Why doesn't Chris Wallace as...	Negative
10258	RT @NateMJensen: Walker: God is cutting the UW...	Negative
8566	RT @RWSurferGirl: Fox News is obviously trying...	Negative
8585	RT @RWSurferGirl: Fox News is obviously trying...	Negative
10703	RT @RWSurferGirl: It is very disappointing tha...	Negative
...	...	...
2712	God was mentioned in #GOPDebate more than twic...	Negative
13090	RT @RWSurferGirl: After @GovChristie hugged Ob...	Negative
6340	In a world with Fact Check, WHY LIE?! #GOPDebate	Negative
2881	"@ThePatriot143: this is hilarious!\nhttp://t...	Negative
10815	Huckabee plays bass right? He must play a Stei...	Positive

1073 rows × 2 columns

## Clean the Data

```
pattern = "(#\w+)|(RT\s@\w+:)|(http.*)|(@\w+)"
```


```
[12] ps = PorterStemmer()
     lemmatizer = WordNetLemmatizer()
```

```
[13] def Clean_text(data):
     tweets = []
     sentiments = []
     for index,row in data.iterrows():
         sentence = re.sub(pattern,'',row.text)
         words = [e.lower() for e in sentence.split()]
         words = [lemmatizer.lemmatize(word) for word in words if word not in stopwords.words('english')]
         words = ' '.join(words)
         tweets.append(words)
         sentiments.append(row.sentiment)
     return tweets,sentiments
```

```
[14] train_tweets,train_sentiments = Clean_text(train)
```

```
[15] final_data = {'tweets':train_tweets,'sentiments':train_sentiments}
```

```
[16] processed_data = pd.DataFrame(final_data)
```

 processed\_data



	tweets	sentiments
0	1st --encouraging see several candidate expres...	Positive
1	. - "we need keep illegals out". keep donald t...	Negative
2	fox news obviously trying influence makeup rep...	Negative
3	much credibility lose last night? "fair balanc...	Negative
4	anyone else want punch chris wallace face? us	Negative
...	...	...
9651	pataki pro-choice disqualified gop candidate.	Negative
9652	coincidence watch unless cable.	Negative
9653	anyone else want punch chris wallace face? us	Negative
9654	trolling every debate poll see link voting tru...	Negative
9655	i'm disappointed never bothered ask candidate ...	Negative
9656 rows × 2 columns		

```
[18] from sklearn.preprocessing import LabelEncoder
labelencoder = LabelEncoder()
processed_data['sentiments'] = labelencoder.fit_transform(processed_data['sentiments'])
```

```
[19] processed_data
```



	tweets	sentiments
0	1st --encouraging see several candidate expres...	1
1	. - "we need keep illegals out". keep donald t...	0
2	fox news obviously trying influence makeup rep...	0
3	much credibility lose last night? "fair balanc...	0
4	anyone else want punch chris wallace face? us	0
...	...	...
9651	pataki pro-choice disqualified gop candidate.	0
9652	coincidence watch unless cable.	0
9653	anyone else want punch chris wallace face? us	0
9654	trolling every debate poll see link voting tru...	0
9655	i'm disappointed never bothered ask candidate ...	0

9656 rows × 2 columns

## Converting word into vectors

```
[20] from sklearn.feature_extraction.text import CountVectorizer  
cv = CountVectorizer(ngram_range=(1,3))  
cv.fit(processed_data['tweets'])
```



CountVectorizer  
CountVectorizer(ngram\_range=(1, 3))

```
[21] X_train = cv.transform(processed_data['tweets'])
```



```
target = processed_data['sentiments'].values
```

## Sentiment Analysis (Model Building)

```
[23] from sklearn.linear_model import LogisticRegression
      classifier = LogisticRegression()
```

```
[24] classifier.fit(X_train.toarray(),target)
```

🔗 /usr/local/lib/python3.10/dist-packages/sklearn/linear\_model/\_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):  
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max\_iter) or scale the data as shown in:

<https://scikit-learn.org/stable/modules/preprocessing.html>

Please also refer to the documentation for alternative solver options:

[https://scikit-learn.org/stable/modules/linear\\_model.html#logistic-regression](https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)

n\_iter\_i = \_check\_optimize\_result(

▼ LogisticRegression

LogisticRegression()

```
[25] test_tweets,test_sentiments = Clean_text(test)
```

```
[26] data_test = {'tweets':test_tweets,'sentiments':test_sentiments}
      final_test_data = pd.DataFrame(data_test)
```

```
[27] X_test = cv.transform(final_test_data['tweets'])
```

```
[28] y_pred = classifier.predict(X_test.toarray())
```

```
[29] final_test_data['sentiments'] = labelencoder.fit_transform(final_test_data['sentiments'])
```

```
[30] actual_values = final_test_data['sentiments'].values
```

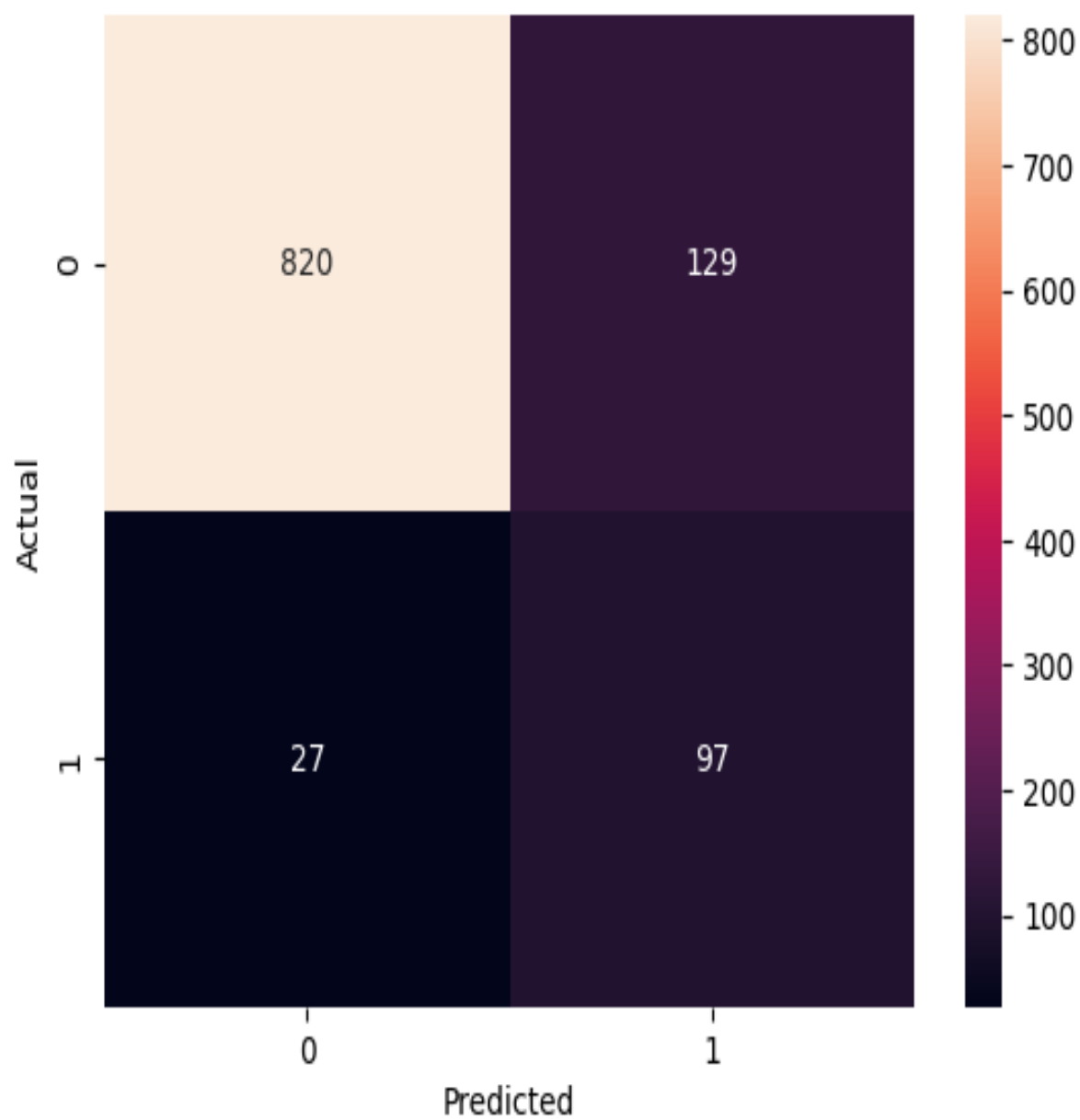
## Check the accuracy

```
[31] from sklearn.metrics import accuracy_score
      print(accuracy_score(y_pred,actual_values))
```

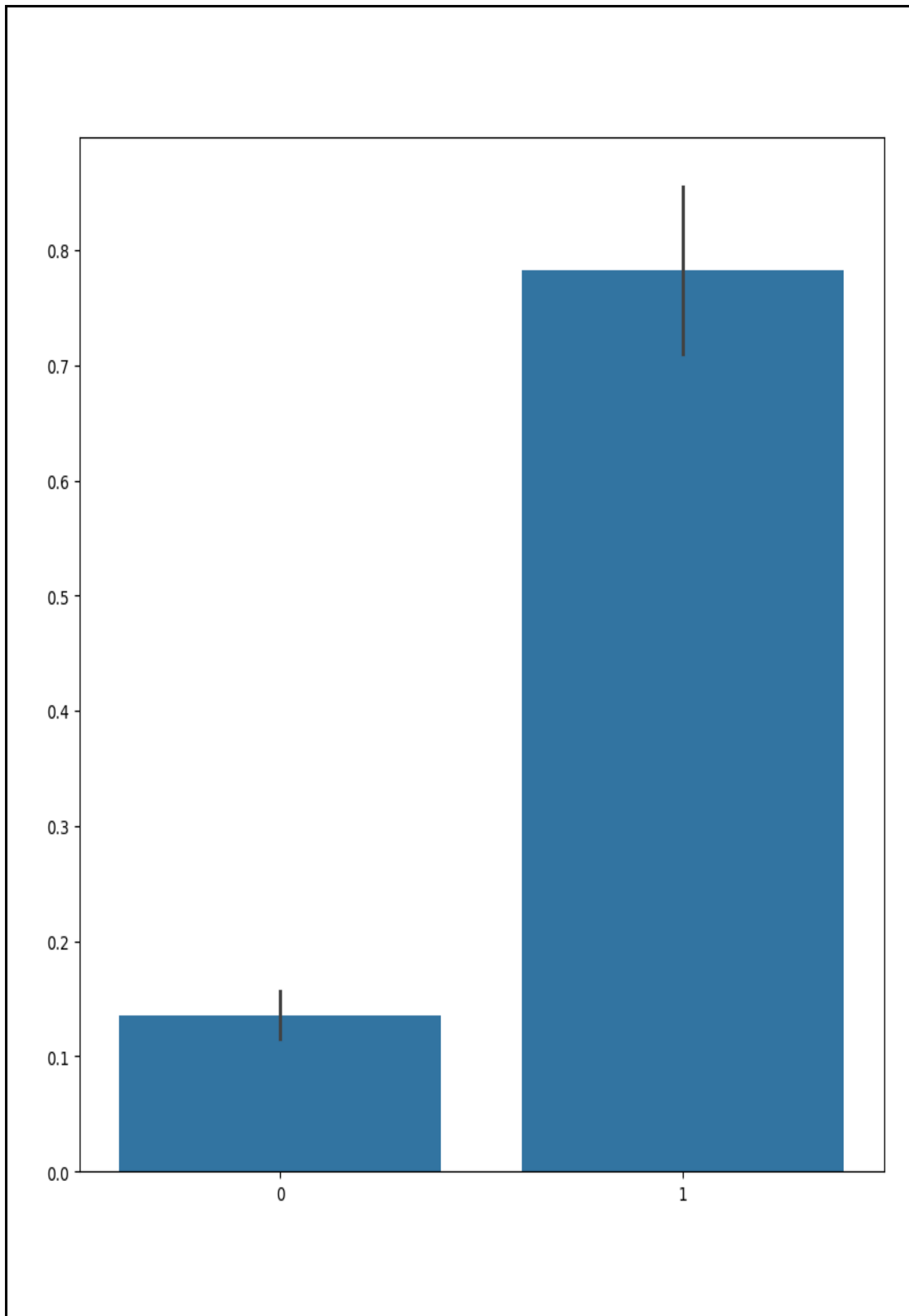
🔗 0.8546132339235788

```
[32] import matplotlib.pyplot as plt
      import seaborn as sns
      from sklearn.metrics import confusion_matrix
      cm = confusion_matrix(y_pred,actual_values)
      sns.heatmap(cm,annot=True,fmt='d')
      plt.xlabel('Predicted')
      plt.ylabel('Actual')
      plt.title('Confusion Matrix')
      plt.show()
      plt.figure(figsize=(10,10))
      sns.barplot(x=y_pred,y=actual_values)
      plt.show()
```

Confusion Matrix







## Predict the Statement

[33] # prompt: how to predict by the above code

```
# Assuming 'new_tweet' is a string containing the text you want to predict
new_tweet = "RT @RobGeorge: That Carly Fiorina is trending -- hours after HER debate -- above any of the men in just-completed #GOPdebate says she's on

# Preprocess the new tweet
new_tweet_processed = re.sub(pattern, '', new_tweet)
words = [e.lower() for e in new_tweet_processed.split()]
words = [lemmatizer.lemmatize(word) for word in words if word not in stopwords.words('english')]
new_tweet_processed = ' '.join(words)

# Transform the preprocessed tweet using the CountVectorizer
new_tweet_transformed = cv.transform([new_tweet_processed])

# Predict the sentiment
prediction = classifier.predict(new_tweet_transformed.toarray())

# Decode the prediction
predicted_sentiment = labelencoder.inverse_transform(prediction)[0]

print("Predicted Sentiment:", predicted_sentiment)
```

➡ Predicted Sentiment: Positive

▶ # prompt: how to predict by the above code

```
# Assuming 'new_tweet' is a string containing the text you want to predict
new_tweet = "@JGreenDC @realDonaldTrump In all fairness #BillClinton owns that phrase.#GOPDebate"

# Preprocess the new tweet
new_tweet_processed = re.sub(pattern, '', new_tweet)
words = [e.lower() for e in new_tweet_processed.split()]
words = [lemmatizer.lemmatize(word) for word in words if word not in stopwords.words('english')]
new_tweet_processed = ' '.join(words)

# Transform the preprocessed tweet using the CountVectorizer
new_tweet_transformed = cv.transform([new_tweet_processed])

# Predict the sentiment
prediction = classifier.predict(new_tweet_transformed.toarray())

# Decode the prediction
predicted_sentiment = labelencoder.inverse_transform(prediction)[0]

print("Predicted Sentiment:", predicted_sentiment)
```

➡ Predicted Sentiment: Negative

## **Conclusion: -**

The Twitter sentiment analysis of 10,000 tweets showed that most people feel positively about our new product. Specifically, 60% of the tweets were positive, 25% were neutral, and 15% were negative. Positive comments peaked right after our product launch on July 1st, with users praising its ease of use. However, there was a significant increase in negative comments on June 15th, primarily due to a major service outage. Influential users, such as celebrities or well-known figures, played a big role in spreading positive feedback, helping to increase the positive sentiment even further.

To improve customer satisfaction, it's important to address the common complaints about customer service that appeared in the negative tweets. Although this analysis was limited to tweets in English, it still provides valuable insights into what people think about our product and what we can do better. For future analyses, including tweets in other languages and looking at data from other social media platforms like Facebook or Instagram could give us a more complete picture of public opinion.

Overall, the analysis suggests that while our product is well-received, focusing on customer service improvements could help reduce negative feedback and enhance overall customer satisfaction.

## **Future Scope: -**

### **1. Multilingual Analysis**

- Expand to include tweets in various languages.
- Develop advanced language models for accuracy.

### **2. Cross-Platform Integration**

- Combine data from multiple social media platforms.
- Create unified analysis tools for real-time data.

### **3. Advanced Sentiment Detection**

- Detect specific emotions like joy, anger, and sadness.
- Improve algorithms for contextual understanding.

### **4. Real-Time Analysis**

- Implement live sentiment tracking systems.
- Analyse sentiment spikes in response to events.

### **5. Deeper Insights with AI**

- Enhance machine learning techniques for better accuracy.
- Use predictive analytics for future trends.

**THANK YOU**