

PROFESSIONAL TRAINING REPORT

at

Sathyabama Institute of Science and Technology

(Deemed to be University)

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering Degree in Computer Science and Engineering

By

NAME : SHOUVIN A

Reg.No. 40731093



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING SCHOOL OF
COMPUTING**

**SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY JEPPIAAR NAGAR,
RAJIV GANDHI SALAI,**

CHENNAI – 600119, TAMILNADU

APRIL 2023



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC

(Established under Section 3 of UGC Act, 1956)

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI- 600119

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **Shouvin A** who carried out the project entitled "**Sentiment Analysis**" under my supervision from Aug 2022 to Oct 2022.

Internal Guide

Dr . A Annie Micheal M.E.Ph.D

Head of the Department

Dr.S.Vigneshwari M.E.,Ph.D

Submitted for Viva voce Examination held on 12-09-2022

Internal Examiner

External Examiner

DECLARATION

I **Shouvin A** hereby declare that the Project Report entitled **Sentiment Analysis** by using Flutter was Done by me under the guidance of **Dr.A Annie Micheal M.E.Ph.D** at (Company name and address) is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE: **12-09-2022**

PLACE: **Chennai**



SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management** of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph.D, Dean**, School of Computing, **Dr. S. Vigneshwari, M.E., Ph.D., Head of the Department of Computer Science and Engineering** for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr . Annie Micheal** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

Sentiment analysis (also known as **opinion mining** or **emotion AI**) is the use of natural language processing, text analysis, computational linguistics, and biometrics to systematically identify, extract, quantify, and study affective states and subjective information. Sentiment analysis is widely applied to voice of the customer materials such as reviews and survey responses, online and social media, and healthcare materials for applications that range from marketing to customer service to clinical medicine. With the rise of deep language models, such as RoBERTa, also more difficult data domains can be analyzed, e.g., news texts where authors typically express their opinion/sentiment less explicitly.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
1.	ABSTRACT	7
	LIST OF FIGURES	10
	INTRODUCTION	11
	1.1 Natural Language Processing	11
	1.2 Sentiment Analysis	13
	1.3 Text Classification	16
	1.4 Machine Learning	18
2.	AIM AND SCOPE OF THE PRESENT INVESTIGATION	20
	2.1 Aim	20
	2.2 Objective	20
	2.3 Goal	20
	2.4 Need Of The Application	21
3.	FUNCTIONS & TOOLS	22
	3.1 Applications & Frameworks	22

	3.1.1 Python	22
	3.1.2 Flask	24
	3.1.3 HTML	27
	3.1.4 CSS	30
	3.1.5 Javascript	34
	3.2 Hardware And Software Requirements	39
	3.2.1 Hardware Requirements	39
	3.2.2 Software Requirements	39
	3.3 Types of Packages	40
4.	RESULTS & SCREENSHOTS	41
5.	SUMMARY & CONCLUSION	42
	REFERENCES	43
	SOURCE CODE	44

LIST OF FIGURES

Fig no	Fig title	Page no
1	Machine Learning	17
2	Working of ML	17
3	Python	21
4	Flask	25
5	HTML	27
6	Tags	27
7	CSS	31
8	JavaScript	36
9	System Requirements	37

CHAPTER 1 :_INTRODUCTION

1.1 Natural Language Processing

Natural language processing (NLP) refers to the branch of computer science and more specifically, the branch of artificial intelligence or AI concerned with giving computers the ability to understand text and spoken words in much the same way human beings can.

NLP combines computational linguistics rule-based modeling of human language with statistical, machine learning, and deep learning models. Together, these technologies enable computers to process human language in the form of text or voice data and to ‘understand’ its full meaning, complete with the speaker or writer’s intent and sentiment.

NLP drives computer programs that translate text from one language to another, respond to spoken commands, and summarize large volumes of text rapidly even in real time. There’s a good chance you’ve interacted with NLP in the form of voice-operated GPS systems, digital assistants, speech-to-text dictation software, customer service chatbots, and other consumer conveniences. But NLP also plays a growing role in enterprise

solutions that help streamline business operations, increase employee productivity, and simplify mission-critical business processes.

Several NLP tasks break down human text and voice data in ways that help the computer make sense of what it's ingesting. Some of these tasks include the following:

- **Speech recognition**, also called speech-to-text, is the task of reliably converting voice data into text data. Speech recognition is required for any application that follows voice commands or answers spoken questions. What makes speech recognition especially challenging is the way people talk—quickly, slurring words together, with varying emphasis and intonation, in different accents, and often using incorrect grammar.
- **Part of speech tagging**, also called grammatical tagging, is the process of determining the part of speech of a particular word or piece of text based on its use and context. Part of speech identifies 'make' as a verb in 'I can make a paper plane,' and as a noun in 'What make of car do you own?'
- **Word sense disambiguation** is the selection of the meaning of a word with multiple meanings through a process of semantic analysis that determine the word that makes the most sense in the given context. For example, word sense disambiguation helps distinguish the meaning of the verb 'make' in 'make the grade' (achieve) vs. 'make a bet' (place).
- **Named entity recognition**, or NEM, identifies words or phrases as useful entities. NEM identifies 'Kentucky' as a location or 'Fred' as a man's name.

- **Co-reference resolution** is the task of identifying if and when two words refer to the same entity. The most common example is determining the person or object to which a certain pronoun refers (e.g., 'she' = 'Mary'), but it can also involve identifying a metaphor or an idiom in the text (e.g., an instance in which 'bear' isn't an animal but a large hairy person).
- **Sentiment analysis** attempts to extract subjective qualities—attitudes, emotions, sarcasm, confusion, suspicion—from text.
- **Natural language generation** is sometimes described as the opposite of speech recognition or speech-to-text; it's the task of putting structured information into human language.

1.2 Sentiment Analysis

Vendors that offer sentiment analysis platforms include Brandwatch, Critical Mention, Hootsuite, Lexalytics, Meltwater, MonkeyLearn, NetBase Quid, Sprout Social, Talkwalker and Sentiment analysis, also referred to as opinion mining, is an approach to natural language processing ([NLP](#)) that identifies the emotional tone behind a body of text. This is a popular way for organizations to determine and categorize opinions about a product, service or idea. Sentiment analysis involves the use of [data mining](#), machine learning ([ML](#)), artificial intelligence and computational linguistics to mine text for sentiment and subjective information such as whether it is expressing positive, negative or neutral feelings.

Sentiment analysis systems help organizations gather insights into real-time customer sentiment, [customer experience](#) and brand reputation.

Generally, these tools use text analytics to analyze online sources such as emails, blog posts, online reviews, customer support tickets, news articles, survey responses, case studies, web chats, tweets, forums and comments. [Algorithms](#) are used to implement rule-based, automatic or hybrid methods of scoring whether the customer is expressing positive words, negative words or neutral ones.

In addition to identifying sentiment, sentiment analysis can extract the polarity or the amount of positivity and negativity, subject and opinion holder within the text. This approach is used to analyze various parts of text, such as a full document or a paragraph, sentence or subsentence.

Zoho. Businesses that use these tools to analyze sentiment can review customer feedback more regularly and proactively respond to changes of opinion within the market.

Working of Sentiment Analysis :

Sentiment analysis generally follows these steps:

1. **Collect data.** The text being analyzed is identified and collected. This involves using a web scraping [bot](#) or a scraping application programming interface.
2. **Clean the data.** The data is processed and cleaned to remove noise and parts of speech that don't have meaning relevant to the sentiment of the text. This includes contractions, such as *I'm*, and words that have little information such as *is*, articles such as *the*,

punctuation, [URLs](#), special characters and capital letters. This is referred to as standardizing.

3. **Extract features.** A machine learning algorithm automatically extracts text features to identify negative or positive sentiment. ML approaches used include the bag-of-words technique that tracks the occurrence of words in a text and the more nuanced word-embedding technique that uses [neural networks](#) to analyze words with similar meanings.
4. **Pick an ML model.** A sentiment analysis tool scores the text using a rule-based, automatic or hybrid ML model. Rule-based systems perform sentiment analysis based on predefined, lexicon-based rules and are often used in domains such as law and medicine where a high degree of precision and human control is needed. Automatic systems use ML and deep learning techniques to learn from data sets. A hybrid model combines both approaches and is generally thought to be the most accurate model. These models offer different approaches to assigning sentiment scores to pieces of text.
5. **Sentiment classification.** Once a model is picked and used to analyze a piece of text, it assigns a sentiment score to the text including positive, negative or neutral. Organizations can also decide to view the results of their analysis at different levels, including document level, which pertains mostly to professional reviews and coverage; sentence level for comments and customer reviews; and sub-sentence level, which identifies phrases or clauses within sentences.

Types of Sentiment Analysis :

Sentiment analysis systems fall into several different categories:

Fine-grained sentiment analysis breaks down sentiment indicators into more precise categories, such as very positive and very negative. This approach is similar to opinion ratings on a one to five star scale. This approach is therefore effective at grading [customer satisfaction surveys](#).

Emotion detection analysis identifies emotions rather than positivity and negativity. Examples include happiness, frustration, shock, anger and sadness.

Intent-based analysis recognizes motivations behind a text in addition to opinion. For example, an online comment expressing frustration about changing a battery may carry the intent of getting [customer service](#) to reach out to resolve the issue.

Aspect-based analysis examines the specific component being positively or negatively mentioned. For example, a customer might review a product saying the battery life was too short. The sentiment analysis system will note that the negative sentiment isn't about the product as a whole but about the battery life.

1.3 Text Classification

Text classification is a machine learning technique that assigns a set of predefined categories to [open-ended text](#). Text classifiers can be used to organize, structure, and categorize pretty much any kind of text – from documents, medical studies and files, and all over the web.

For example, new articles can be organized by topics; support tickets can be organized by urgency; chat conversations can be organized by language; brand mentions can be organized by sentiment; and so on.

Text classification is one of the fundamental tasks in natural language processing with broad applications such as [sentiment analysis](#), topic labeling, spam detection, and intent detection.

Reasons :

- **Scalability**

Manually analyzing and organizing is slow and much less accurate.. Machine learning can automatically analyze millions of [surveys](#), comments, emails, etc., at a fraction of the cost, often in just a few minutes. Text classification tools are scalable to any business needs, large or small.

- **Real-time analysis**

There are critical situations that companies need to identify as soon as possible and take immediate action (e.g., PR crises on social media). Machine learning text classification can follow your brand mentions constantly and in real time, so you'll identify critical information and be able to take action right away.

- **Consistent criteria**

Human annotators make mistakes when classifying text data due to distractions, fatigue, and boredom, and human subjectivity creates inconsistent criteria. Machine learning, on the other hand, applies the

same lens and criteria to all data and results. Once a text classification model is properly trained it performs with unsurpassed accuracy.

1.4 Machine Learning

Machine Learning is said as a subset of **artificial intelligence** that is mainly concerned with the development of algorithms which allow a computer to learn from the data and past experiences on their own. The term machine learning was first introduced by **Arthur Samuel** in **1959**.

With the help of sample historical data, which is known as **training data**, machine learning algorithms build a **mathematical model** that helps in making predictions or decisions without being explicitly programmed. Machine learning brings computer science and statistics together for creating predictive models. Machine learning constructs or uses the algorithms that learn from historical data. The more we will provide the information, the higher will be the performance.

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

Suppose we have a complex problem, where we need to perform some predictions, so instead of writing a code for it, we just need to feed the data to generic algorithms, and with the help of these algorithms, machine builds the logic as per the data and predict the output. Machine learning

has changed our way of thinking about the problem. The below block diagram explains the working of Machine Learning algorithm:

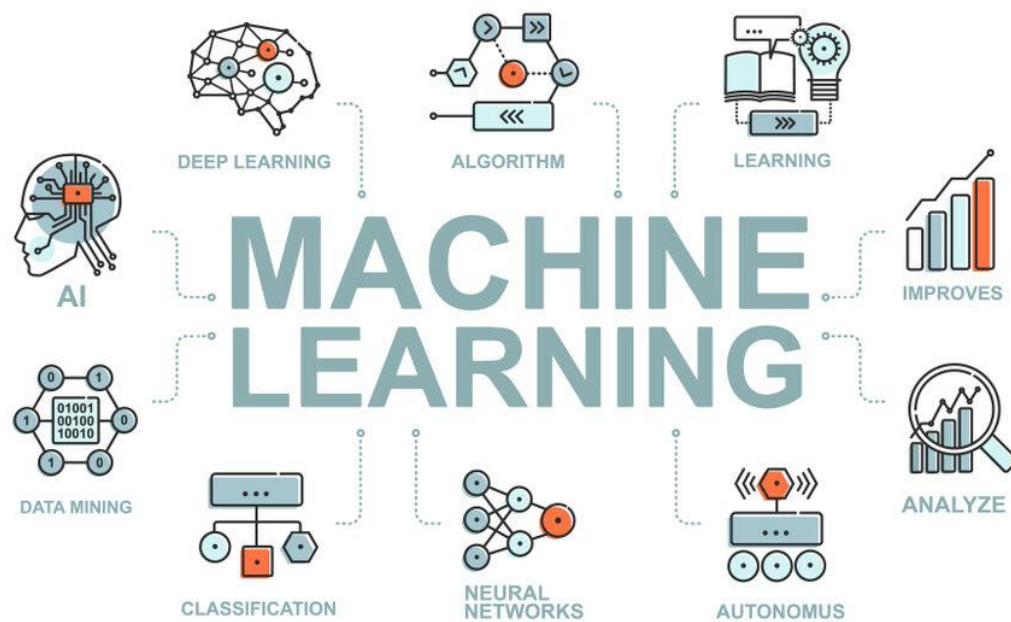


Fig 1 : Machine Learning

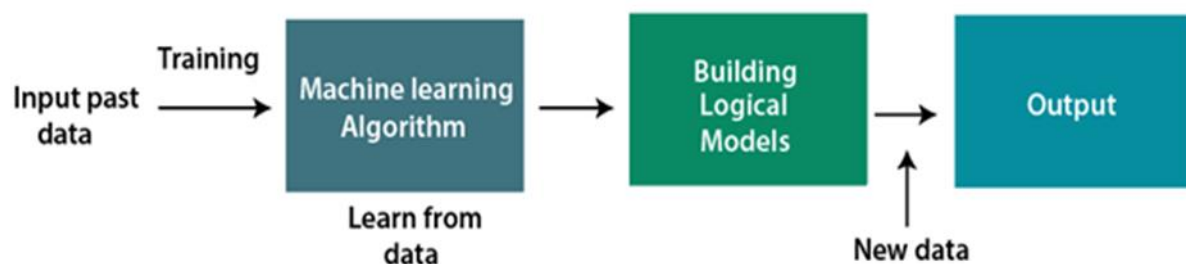


Fig 2 : Working of ML

CHAPTER 2

2.1 AIM :

To build an NLP model for text classification for analyzing the positive and negative words and giving it an accuracy score on a web application.

2.2 Objective:

Develop a web application and NLP model to analyze the words for text classification and give it an accuracy score to predict the emotion of the word/sentence to help identify the sentiment of it.

2.3 Goal:

Sentiment analysis, also referred to as opinion mining, is an approach to as opinion mining, is an approach to natural language processing (NLP) that identifies the emotional tone behind a body of text . This is a popular way for organizations to determine and categorize opinions about a product, service or idea. Sentiment analysis involves the use of data mining, machine learning (ML), artificial intelligence and computational linguistics to mine text for sentiment and subjective information such as whether it is expressing positive, negative or neutral feelings.

2.3 Need of Application:

- To help companies study the customer sentiment around a particular product and to visualize the collected data clearly and effectively.
- To help system users analyse a huge amount of data, quickly and efficiently.
- To segregate customer sentiment on a scale of -1 to 1, where -1 represents a strong negative sentimentality towards the keyword(s), and 1 represents a strongly positive reaction.

CHAPTER 3 : FUNCTIONS AND TOOLS

3.1 Applications and Frameworks

3.1.1 Python :

Python is a computer programming language often used to build websites and software, automate tasks, and conduct data analysis. Python is a general-purpose language, meaning it can be used to create a variety of different programs and isn't specialized for any specific problems. This versatility, along with its beginner-friendliness, has made it one of the most-used programming languages today.

Stack Overflow's 2022 Developer Survey revealed that Python is the fourth most popular programming language, with respondents saying that they use Python almost 50 percent of the time in their development work. Survey results also showed that Python is tied with Rust as the most-wanted technology, with 18% percent of developers who aren't using it already saying that they are interested in learning Python .

Python is commonly used for developing websites and software, task automation, data analysis, and data visualization. Since it's relatively easy to learn, Python has been adopted by many non-programmers such as accountants and scientists, for a variety of everyday tasks, like organizing finances.

“Writing programs is a very creative and rewarding activity,” says University of Michigan and Coursera instructor Charles R Severance in

his book *Python for Everybody*. “You can write programs for many reasons, ranging from making your living to solving a difficult data analysis problem to having fun to helping someone else solve a problem.”

What can you do with python? Some things include:

- Data analysis and machine learning
- Web development
- Automation or scripting
- Software testing and prototyping
- Everyday tasks



Fig 3 : Python

Python is currently the most widely used multi-purpose, high-level programming language, which allows programming in Object-Oriented and Procedural paradigms. Python programs generally are smaller than other programming languages like Java. Programmers have to type relatively less and the indentation requirement of the language, makes them readable all the time.

Python is popular for a number of reasons. Here’s a deeper look at what makes it so versatile and easy to use for coders.

- It has a **simple syntax** that mimics natural language, so it's easier to read and understand. This makes it quicker to build projects, and faster to improve on them.
- It's **versatile**. Python can be used for many different tasks, from web development to machine learning.
- It's **beginner friendly**, making it popular for entry-level coders.
- It's **open source**, which means it's free to use and distribute, even for commercial purposes.
- Python's archive of **modules and libraries**—bundles of code that third-party users have created to expand Python's capabilities—is vast and growing.
- Python has a **large and active community** that contributes to Python's pool of modules and libraries, and acts as a helpful resource for other programmers. The vast support community means that if coders run into a stumbling block, finding a solution is relatively easy; somebody is bound to have encountered the same problem before.

3.1.2 Flask :

Flask is a micro [web framework](#) written in [Python](#). It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common

functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools.

Applications that use the Flask framework include [Pinterest](#) and [LinkedIn](#).

Components

The microframework Flask is part of the *Pallets Projects* (formerly *Pocoo*), and based on several others of them, all under a [BSD license](#).

Werkzeug

Werkzeug (German for "tool") is a utility library for the Python programming language for [Web Server Gateway Interface](#) (WSGI) applications. Werkzeug can instantiate objects for request, response, and utility functions. It can be used as the basis for a custom software framework and supports Python 2.7 and 3.5 and later.

Jinja

Main article: [Jinja \(template engine\)](#)

Jinja, also by Ronacher, is a [template engine](#) for the Python programming language. Similar to the Django web framework, it handles templates in a sandbox.

MarkupSafe

MarkupSafe is a string handling library for the Python programming language. The eponymous MarkupSafe type extends the Python string type and marks its contents as "safe"; combining MarkupSafe with regular

strings automatically escapes the unmarked strings, while avoiding double escaping of already marked strings.

ItsDangerous

ItsDangerous is a safe [data serialization](#) library for the Python programming language. It is used to store the session of a Flask application in a cookie without allowing users to tamper with the session contents.

Features

- Development server and [debugger](#)
- Integrated support for [unit testing](#)
- [RESTful](#) request dispatching
- Uses Jinja templating
- Support for secure cookies (client side sessions)
- 100% [WSGI](#) 1.0 compliant
- [Unicode](#)-based
- Complete documentation
- [Google App Engine](#) compatibility
- Extensions available to extend functionality



Fig 4 : Flask

3.1.3 HTML (HyperText Markup Language)

HTML (HyperText Markup Language) is the most basic building block of the Web. It defines the meaning and structure of web content. Other technologies besides HTML are generally used to describe a web page's appearance/presentation ([CSS](#)) or functionality/behavior ([JavaScript](#)).

"Hypertext" refers to links that connect web pages to one another, either within a single website or between websites. Links are a fundamental aspect of the Web. By uploading content to the Internet and linking it to pages created by other people, you become an active participant in the World Wide Web.

HTML uses "markup" to annotate text, images, and other content for display in a Web browser. HTML markup includes special "elements" such as [<head>](#), [<title>](#), [<body>](#), [<header>](#), [<footer>](#), [<article>](#), [<section>](#), [<p>](#), [<div>](#), [](#), [](#), [<aside>](#), [<audio>](#), [<canvas>](#), [<datalist>](#),

[<details>](#), [<embed>](#), [<nav>](#), [<output>](#), [<progress>](#), [<video>](#), [](#), [](#), [](#) and many others.

An HTML element is set off from other text in a document by "tags", which consist of the element name surrounded by "<" and ">". The name of an element inside a tag is case insensitive. That is, it can be written in uppercase, lowercase, or a mixture. For example, the <title> tag can be written as <Title>, <TITLE>, or in any other way. However, the convention and recommended practice is to write tags in lowercase.

- **<DOCTYPE! html>** – A doctype or document type declaration is an instruction that tells the web browser about the markup language in which the current page is written. It is not an element or tag. The doctype declaration is not case-sensitive.
- [<html>](#) – This tag is used to define the root element of HTML document. This tag tells the browser that it is an HTML document. It is the second outer container element that contains all other elements within it.
- [<head>](#) – This tag is used to define the head portion of the HTML document that contains information related to the document. Elements within the head tag are not visible on the front-end of a webpage.
- [<body>](#) – The body tag is used to enclose all the visible content of a webpage. In other words, the body content is what the browser will show on the front end.



Fig 5 : HTML

HTML Page Structure

`<!DOCTYPE html>` ← Tells version of HTML

`<html>` ← HTML Root Element

`<head>` ← Used to contain page HTML metadata

`<title>Page Title</title>` ← Title of HTML page

`</head>`

`<body>` ← Hold content of HTML

`<h2>Heading Content</h2>` ← HTML heading tag

`<p>Paragraph Content</p>` ← HTML paragraph tag

`</body>`

`</html>`

Fig 6 : Tags

3.1.4 CSS (Cascading Style Sheet)

Cascading Style Sheets (CSS) is a [stylesheet](#) language used to describe the presentation of a document written in [HTML](#) or [XML](#) (including XML dialects such as [SVG](#), [MathML](#) or [XHTML](#)). CSS describes how elements should be rendered on screen, on paper, in speech, or on other media.

CSS is among the core languages of the **open web** and is standardized across Web browsers according to [W3C specifications](#). Previously, the development of various parts of CSS specification was done synchronously, which allowed the versioning of the latest recommendations. You might have heard about CSS1, CSS2.1, or even CSS3. There will never be a CSS3 or a CSS4; rather, everything is now CSS without a version number.

After CSS 2.1, the scope of the specification increased significantly and the progress on different CSS modules started to differ so much, that it became more effective to develop and release recommendations separately per module. Instead of versioning the CSS specification, W3C now periodically takes a snapshot of the latest stable state of the CSS specification and individual modules progress. CSS modules now have version numbers, or levels, such as CSS Color Module Level 5.

CSS First Steps

CSS (Cascading Style Sheets) is used to style and layout web pages — for example, to alter the font, color, size, and spacing of your content, split it into multiple columns, or add animations and other decorative features. This module provides a gentle beginning

to your path towards CSS mastery with the basics of how it works, what the syntax looks like, and how you can start using it to add styling to HTML.

CSS Building Blocks

This module carries on where [CSS first steps](#) left off — now you've gained familiarity with the language and its syntax, and got some basic experience with using it, it's time to dive a bit deeper. This module looks at the cascade and inheritance, all the selector types we have available, units, sizing, styling backgrounds and borders, debugging, and lots more.

The aim here is to provide you with a toolkit for writing competent CSS and help you understand all the essential theory, before moving on to more specific disciplines like [text styling](#) and [CSS layout](#).

CSS Styling Sheet

With the basics of the CSS language covered, the next CSS topic for you to concentrate on is styling text — one of the most common things you'll do with CSS. Here we look at text styling fundamentals, including setting font, boldness, italics, line and letter spacing, drop shadows, and other text features. We round off the module by looking at applying custom fonts to your page, and styling lists and links.

CSS Style Layout

At this point we've already looked at CSS fundamentals, how to style text, and how to style and manipulate the boxes that your content sits inside. Now it's time to look at how to place your boxes in the right place in relation to the viewport, and to each other. We have covered the necessary prerequisites so we can now dive deep into CSS layout, looking at different display settings, modern layout tools like flexbox, CSS grid, and positioning, and some of the legacy techniques you might still want to know about.

```
p.important { border-style: solid; border-width: 5px; border-color: purple;
}
```

```
border-width: 5px 8px 3px 9px;
```

```
34  /* A reference to a type */
35  span.ts span.type-ref {
36      color:  rgb(175, 0, 219) !important;
37  }
38
39  /* Signature details */
40  div.signature > table {
41      border-collapse: collapse;
42      border: thin  darkgray solid;
43      width: 60%;
44  }
```



Fig 7 : CSS

3.1.5 JavaScript

JavaScript (JS) is a lightweight, interpreted, or [just-in-time](#) compiled programming language with first-class functions. While it is most well-known as the scripting language for Web pages, many non-browser environments also use it, such as [Node.js](#), Apache CouchDB and [Adobe Acrobat](#). JavaScript is a prototype-based, multi-paradigm, single-threaded, dynamic language, supporting object-oriented, imperative, and declarative (e.g. functional programming) styles.

JavaScript's dynamic capabilities include runtime object construction, variable parameter lists, function variables, dynamic script creation (via [eval](#)), object introspection (via [for...in](#) and [Object utilities](#)), and source-code recovery (JavaScript functions store their source text and can be retrieved through [toString\(\)](#)).

This section is dedicated to the JavaScript language itself, and not the parts that are specific to Web pages or other host environments. For information about [APIs](#) that are specific to Web pages, please see [Web APIs](#) and [DOM](#).

The standards for JavaScript are the ECMAScript Language Specification (ECMA-262) and the [ECMAScript Internationalization API specification](#) (ECMA-402). As soon as one browser implements a feature, we try to document it. This means that cases where some proposals for new ECMAScript features have already been implemented in browsers, documentation and examples in MDN articles may use some of those new features. Most of the time, this happens between the stages 3 and 4, and is usually before the spec is officially published.

Do not confuse JavaScript with the Java programming language — **JavaScript is *not* "Interpreted Java"**. Both "Java" and "JavaScript" are trademarks or registered trademarks of Oracle in the U.S. and other countries. However, the two programming languages have very different syntax, semantics, and use.

Understanding client-side JavaScript frameworks :

JavaScript frameworks are an essential part of modern front-end web development, providing developers with proven tools for building scalable, interactive web applications. This module gives you some fundamental background knowledge about how client-side frameworks work and how they fit into your toolset, before moving on to a series of tutorials covering some of today's most popular ones.

JavaScript language overview :

An overview of the basic syntax and semantics of JavaScript for those coming from other programming languages to get up to speed.

JavaScript Data Structures :

Overview of available data structures in JavaScript.

Equality Sameness and Comparisons

JavaScript provides three different value comparison operations: strict equality using `===`, loose equality using `==`, and the [Object.is\(\)](#) method.

Enumerability and ownership of properties :

How different methods that visit a group of object properties one-by-one handle the enumerability and ownership of properties.

Closures

A closure is the combination of a function and the lexical environment within which that function was declared.

Reason to Learn JavaScript

JavaScript is used by many developers (65% of the total development community), and the number is increasing day by day. JavaScript is one such programming language that has more than 1444231 libraries and increasing rapidly. It is preferred over any other programming language by most developers. Also, major tech companies like Microsoft, Uber, Google, Netflix, and Meta use JavaScript in their projects.

JavaScript can be added to your HTML file in two ways:

- Internal JavaScript
- External JavaScript

Internal JavaScript: We can add JS code directly to our HTML file by writing the code inside the `<script>` & `</script>`. The `<script>` tag can either be placed inside the `<head>` or the `<body>` tag according to the requirement.

JavaScript Used for : It is mainly used to develop websites and web-based applications. JavaScript is a language that can be used as a front-end as well as a backend.

- **Creating Interactive Websites:** JavaScript is used to make web pages dynamic and interactive. It means using JavaScript, we can change the web page content and styles dynamically.
- **Building Applications:** JavaScript is used to make web and mobile applications. To build web and mobile apps, we can use the most popular JavaScript frameworks like – ReactJS, React Native, Node.js etc.
- **Web Servers:** We can make robust server applications using JavaScript. To be precise we use JavaScript frameworks like Node.js and Express.js to build these servers.
- **Game Development:** JavaScript can be used to design Browser games. In JavaScript, lots of game engines are available that provide frameworks for building games.

Things that makes JavaScript demanding

JavaScript is the most popular and hence the most loved language around the globe. Apart from this, there are abundant reasons to become the most demanding. Below are a listing of a few important points:

- **No need for compilers:** Since JavaScript is an interpreted language, therefore it does not need any compiler for compilation.
- **Used both Client and Server-side:** Earlier JavaScript was used to build client-side applications only, but with the evolution of its frameworks namely Node.js and Express.js, it is now widely used for building server-side applications too.
- **Helps to build a complete solution:** As we saw, JavaScript is widely used in both client and server-side applications, therefore it helps us to build an end-to-end solution to a given problem.

- **Used everywhere:** JavaScript is so loved because it can be used anywhere. It can be used to develop websites, games or mobile apps, etc.
- **Huge community support:** JavaScript has a huge community of users and mentors who love this language and take it's legacy forward.



Fig 8 : JavaScript

3.1.6 Hardware and Software requirements

Hardware Requirements:

- I3 Processor or above, 8th gen or above recommended
- 4 GB RAM minimum, 8 GB RAM recommended
- 2 GB of available disk space minimum
- 4 GB Recommended (500 MB for IDE + 1.5 GB for Android SDK and emulator system image)
- 1280 x 800 minimum screen resolution

Software Requirements:

- Operating Systems: Windows 10 or later (64-bit), x86-64 based
- Disk Space: 1.64 GB (does not include disk space for IDE/tools)



Fig 9 : System Requirements

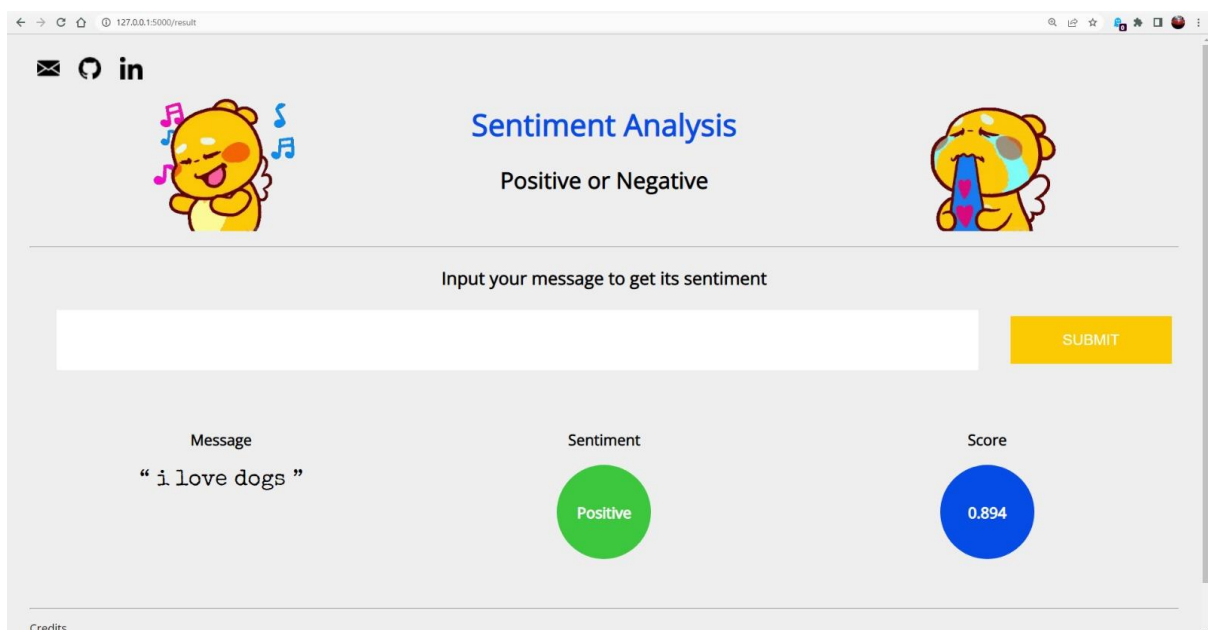
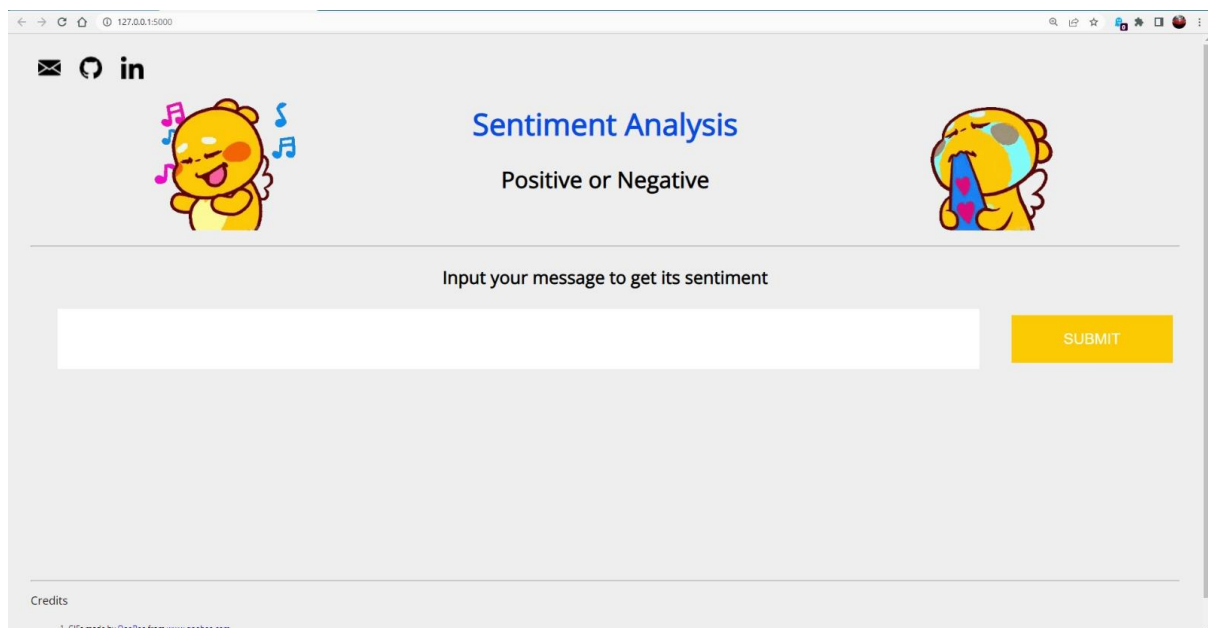
3.1.7 Packages used in Sentiment Analysis :

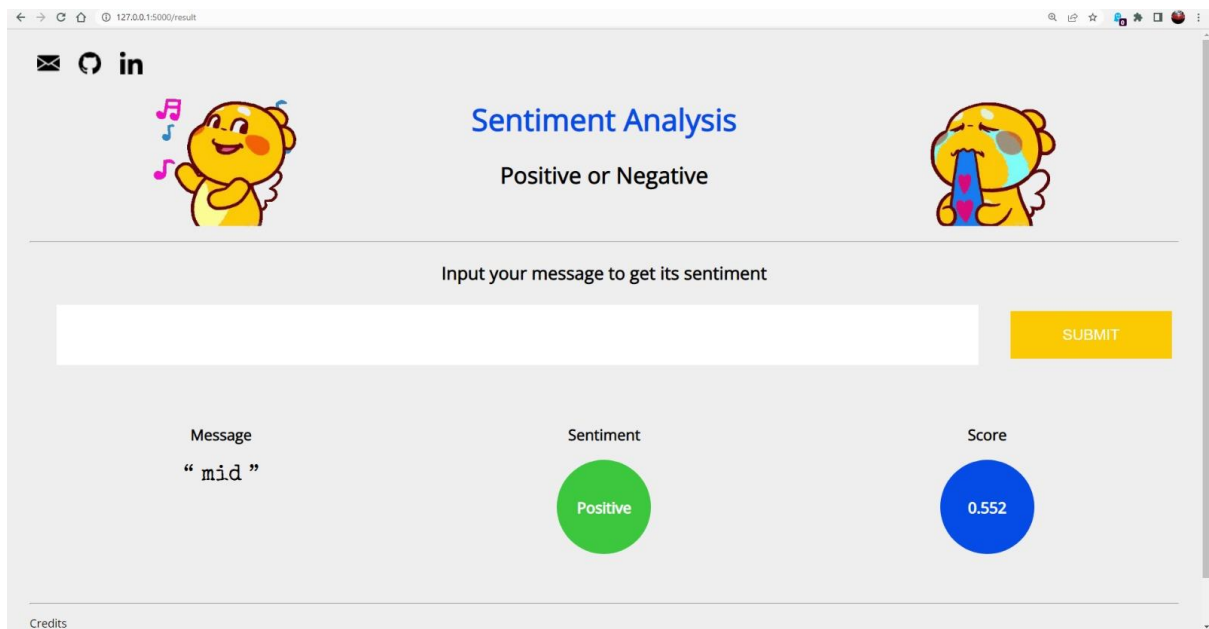
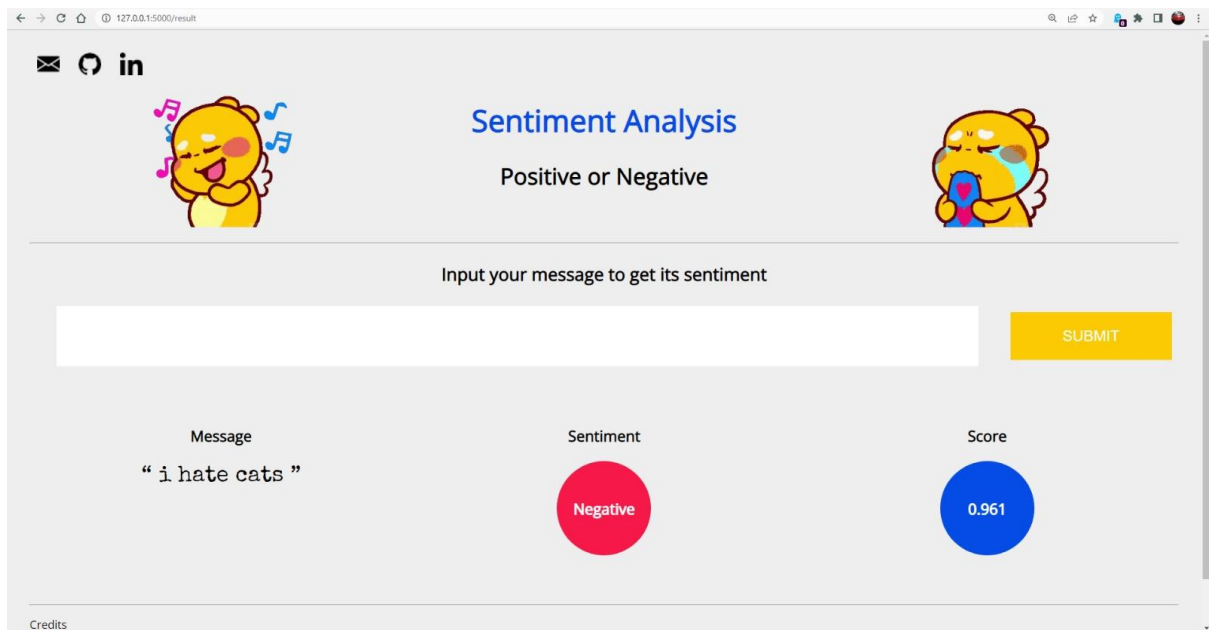
NLTK :

The **Natural Language Toolkit**, or more commonly **NLTK**, is a suite of [libraries](#) and programs for symbolic and statistical [natural language processing](#) (NLP) for English written in the [Python programming language](#). It was developed by Steven Bird and Edward Loper in the Department of Computer and Information Science at the University of Pennsylvania.^[4] NLTK includes graphical demonstrations and sample data. It is accompanied by a book that explains the underlying concepts behind the language processing tasks supported by the toolkit, plus a cookbook.^[6]

NLTK is intended to support research and teaching in [NLP](#) or closely related areas, including empirical [linguistics](#), [cognitive science](#), [artificial intelligence](#), [information retrieval](#), and [machine learning](#).^[7] NLTK has been used successfully as a teaching tool, as an individual study tool, and as a platform for prototyping and building research systems. There are 32 universities in the US and 25 countries using NLTK in their courses. NLTK supports classification, tokenization, stemming, tagging, parsing, and semantic reasoning functionalities.

CHAPTER 4 : RESULTS AND SCREENSHOTS





CHAPTER : 5 SUMMARY & CONCLUSION

Sentiment Analysis, as the name suggests, it means to identify the view or emotion behind a situation. It basically means to analyze and find the emotion or intent behind a piece of text or speech or any mode of communication.

We, humans, communicate with each other in a variety of languages, and any language is just a mediator or a way in which we try to express ourselves. And, whatever we say has a sentiment associated with it. It might be positive or negative or it might be neutral as well.

Sentiment analysis helps data analysts within large enterprises gauge public opinion, conduct nuanced market research, monitor brand and product reputation, and understand customer experiences. In addition, data analytics companies often integrate third-party [sentiment analysis APIs](#) into their own customer experience management, social media monitoring, or workforce analytics platform, in order to deliver useful insights to their own customers.

Broadly speaking, sentiment analysis is most effective when used as a tool for Voice of Customer and Voice of Employee. Business analysts, product managers, customer support directors, human resources and workforce analysts, and other stakeholders use sentiment analysis to understand how customers and employees feel about particular subjects, and why they feel that way.

REFERENCES

- <https://www.techtarget.com/searchbusinessanalytics/definition/opinion-mining-sentiment-mining>
- <https://monkeylearn.com/text-classification>
- <https://www.ibm.com/in-en/topics/natural-language-processing>
- <https://www.javatpoint.com/machine-learning>
- <https://developer.mozilla.org/en-US/docs/Web/CSS>
- <https://www.geeksforgeeks.org/javascript>

SOURCE CODE

```
import pickle
from flask import Flask
from .model import predict # Import predict function from
model.py

'''
Initiate a new flaskr app
1. Input some random secret key to be used by the
application
2. Input some flaskr commands that would be used by the
application
'''
app = Flask(__name__)
app.config.from_mapping(
    SECRET_KEY='\xe0\xcd\xac#\x06\xd9\xe4\x00\xa5\xf2\x88\xc
3\xef$\xa5\x05n\x97\xd8\x1269i\xd3'
)
from flask import (
    redirect, render_template, request, session, url_for
)

'''
Load the machine learning libraries
1. Logistic regression model is used to predict the
sentiment on the newly computed matrix
```

```
'''
# Load the machine learning model
with open('./flaskr/static/LogisticRegression.pickle', 'rb')
as input_file:
    model = pickle.load(input_file)

'''

Home Page
1. It will take both GET and POST requests
2. For GET request, base.html (homepage) will be rendered
without any results shown
3. For POST request, input message will be obtained from the
form in base.html.
    a) Session will then be cleared (to remove anything
belonged to previous session) and 'message' will be passed
into the session
    so that it can be reused throughout the session
    b) The page will then be redirected to /result page
'''

@app.route('/', methods=('GET', 'POST'))
def index():
    if request.method == 'POST':
        message = request.form['message']
        if message is not None:
            session.clear()
            session['message'] = message
            return redirect(url_for('result'))
    return render_template("base.html")
```

```
'''
```

Result Page

1. It will take both GET and POST requests

2. For GET request, 'message' will be obtained from the session, remember the 'message' is from the Home page!

a) Sentiment and its score(probability) will be predicted by passing in the vectorizer (optional), model and message from the session

b) The result page will then be rendered based on the message, sentiment and score computed by the predictions

3. For POST request, input message will be obtained from the form in result.html

a) Session will then be cleared (to remove anything belonged to previous session) and 'message' will be passed into the session

so that it can be reused throughout the session

c) The page will then be redirected to /result page

```
'''
```

```
@app.route('/result', methods=('GET', 'POST'))
```

```
def result():
```

```
    message = session.get('message')
```

```
    df_pred = predict(model=model, text=message)
```

```
    sentiment = df_pred.head(1)['sentiment'].values[0]
```

```
    score = df_pred.head(1)['score'].values[0]
```

```
    if request.method == 'POST':
```

```
        message = request.form['message']
```

```
        if message is not None:
```

```
            session.clear()
```

```

        session['message'] = message
        return redirect(url_for('result'))
    return render_template("result.html", message=message,
sentiment=sentiment, score=score)
import re
import numpy as np
import pandas as pd

import nltk
nltk.download('wordnet')

from nltk.stem import WordNetLemmatizer
from sklearn.feature_extraction.text import
HashingVectorizer

'''
Preprocess the message, return the preprocessed message
1. Will do a sequence of text preprocessing
2. List of stop words to be used, we can either hardcode the
stopwords in the code or use stopwords from nltk,
remember to exclude words like 'not','nor' and etc as it
will affect the meaning significantly!
3. The stop words will be removed from the preprocessed
message
4. Message will then be lemmatized (e.g. ran, run, runs,
running -> run)
'''
def preprocess(text):
    # Preprocess the message

```

```

    text = text.lower() # Convert everything to lowercase
    text =
re.sub('((www\.([^\s]+)| (https?://([^\s]+)| (http?://([^\s]+)))',
'', text) # Remove www, http://, https://
    text = re.sub('@([^\s]+)', '', text) # Remove @username
    text = re.sub('#([^\s]+)', '', text) # Remove #hashtag
    text = re.sub('[^0-9A-Za-z \t]', ' ', text) # Remove
anything that is not alphanumeric, space or tab
    text = re.sub(r"(\.)\1\1+", r"\1\1",
                text) # Replace heyyy into hey or youuuu
to youu, only keep the last 2 characters
    text = text.strip() # remove trailing characters from
both front and back

# Create a list of stop words
stop_words = ['a', 'about', 'above', 'after', 'again',
'ain', 'all', 'am', 'an',
                'and', 'any', 'are', 'as', 'at', 'be',
'because', 'been', 'before',
                'being', 'below', 'between', 'both', 'by',
'can', 'd', 'did', 'do',
                'does', 'doing', 'down', 'during', 'each',
'few', 'for', 'from',
                'further', 'had', 'has', 'have', 'having',
'he', 'her', 'here',
                'hers', 'herself', 'him', 'himself',
'his', 'how', 'i', 'if', 'in',
                'into', 'is', 'it', 'its', 'itself',
'just', 'll', 'm', 'ma',

```

```

        'me', 'more', 'most', 'my', 'myself',
'now', 'o', 'of', 'on', 'once',
        'only', 'or', 'other', 'our', 'ours',
'ourselves', 'out', 'own', 're',
        's', 'same', 'she', "shes", 'should',
"shouldve", 'so', 'some', 'such',
        't', 'than', 'that', "thatll", 'the',
'their', 'theirs', 'them',
        'themselves', 'then', 'there', 'these',
'they', 'this', 'those',
        'through', 'to', 'too', 'under', 'until',
'up', 've', 'very', 'was',
        'we', 'were', 'what', 'when', 'where',
'which', 'while', 'who', 'whom',
        'why', 'will', 'with', 'won', 'y', 'you',
"youd", "youll", "youre",
        "youve", 'your', 'yours', 'yourself',
'yourselves']

```

```

    lemmatizer = WordNetLemmatizer()
    tokens = []
    for token in text.split():
        if token not in stop_words:
            tokens.append(lemmatizer.lemmatize(token))
    return " ".join(tokens)

```

```

'''

```

Predict the sentiment of message, return a dataframe with
columns : text, sentiment and score

1. Preprocess the message using above function

2. Vectorize the data using vectorizer model
3. Predict the sentiment using model
4. Obtain the score(probability) of either positive/negative sentiment, round it to 3 decimal places
5. (Beware that not all model has predict_proba method to obtain the score!)
6. Convert the results into a dataframe and replace sentiment column with 0:Negative, 1:Positive

```
'''
def predict(model, text,
vectoriser=HashingVectorizer(ngram_range=(1,2))):
    textdata = vectoriser.transform([preprocess(text)])
    sentiment = model.predict(textdata)
    score = round(np.amax(model.predict_proba(textdata)), 3)

    data = [(text, sentiment, score)]
    df = pd.DataFrame(data, columns=['text', 'sentiment',
'score'])
    df['sentiment'] =
df['sentiment'].replace([0,1],['Negative','Positive'])
    return df
```