1. SYNOPSIS

A) Title of Project:

Sentiment Analysis and Personality Test

B) Objective:

- The main objective is to develop the web application to detect the sentiment of the text provided by the user.
- This project help to know find the opinion of the person.

C) Description:

Sentiment analysis is a kind of data mining where you measure the inclination of people's opinions by using NLP (natural language processing), text analysis, and computational linguistics. It gives precise results to what the public opinion is about the subject. It classified its results in different categories such as: Very Negative, Negative, Neutral, Positive, Very Positive.

In this application, it detect the person opinion from the text what the user have entered.

D) Modules:

Admin Module
User Module

E) System Requirement Specification(SRS):

Hardware Requirement Specification:

Processor : Standard processor with a speed of 1.6 GHz
RAM : 512 MB
Hard Disk : 10 GB
Monitor : Standard Color Monitor
Keyboard : Standard Keyboard
Mouse : Standard Mouse

Software Requirement Specification:

Operating System : Windows XP
Database : MySQL
Browser : Internet Explorer

F) Used Technologies:

Front End	: Python Django
Web Technologies	: HTML, CSS, JavaScript
Back End	: SQLite3

2. Introduction

This project is designed so as to analysis the sentence using Natural Language Processing technique. This project is help to detect the sentiment of the sentence whether it's represent the happy, sad or netural and also help to test the personality of the user. Sentiment Analysis is the most common text classification tool that analyses an incoming message and tells whether the underlying sentiment is positive, negative our neutral.

Reason for the Project

With advancements in computer technology, new attempts have been made to analyze psychological traits through computer programming and to predict them quickly, efficiently, and accurately. Especially with the rise of Machine Learning (ML), Deep Learning (DL), and Natural Language Processing (NLP), researchers in the field of psychology are widely adopting NLP to assess psychological construct or to predict human behaviors.

- There was a time when the social media
  services like Facebook used to just have two emotions associated with
  each post, i.e You can like a post or you can leave the post without any
  reaction and that basically signifies that you didn't like it. But, over time these reactions to post have
  changed and grew into more granular sentiments which we see as of now,
  such as "like", "love", "sad",
  "angry" etc.
- there are billions of people and they have their own style of communicating, i.e. a lot of tiny variations are added to the language and a lot of sentiments are attached to it which is easy for us to interpret but it becomes a challenge for the machines.
- Sentiment analysis is extremely useful in social media monitoring as it **allows us to gain an overview of the wider public opinion behind certain topics**.
- It's often used by businesses **to detect sentiment in social data, gauge brand reputation, and understand customers**.
- Personality Testing helps to understand your own character traits can be a powerful tool when deciding your career path.

Problem Statement

The problem in sentiment analysis is classifying the polarity of a given text at the document, sentence or feature/aspect level. Whether the expressed opinion in a document, a sentence or an entity feature/aspect is posivie, negative or netural.

The problem in the personality test is classifying the persoanlity of perrson on basis of MBTI model where model is train with the past data and help to predict from past train data.

Aims & Objectives

To develop the web application for the user to test the sentiment, persoanlity and language translator. This application help to detect the sentiment wheather the person is happy, sad or netural and personality test show which type of person is using this applicaion.

Scope

The web application is going to be developed is known as Sentiment and Persoanality Test.The main users of this system are administrator, and staff. This system help to provide the user to test the sentiment and the persoanlity and also it's help to make to know the user mode. The model of this project are:

I. **User Module**

This model is used to register and login to the application for the user. So, that user can utitlitze the services of this application.

II. **Admin Module**

This model is admistration module where suprme user can look after all the infromation about the user and also can control and manage the application different part and data.

III. **Personality Module**

In this model, user id, user text, user result are store in the database where the user can test their persoanlity.

IV. **Sentiment Module**

In this model, user id, user text, user result are store in the database where the user can test their sentiment where sad, happy and normal.

V. **Language Module**

In this model, user id, user text, user result are store in the database where the user can used to translate the english language to Kanada.

The system is a multi-user system since it is used by different groups of users. It is developed as a web application which can run in any platform where browser and internet is avliable. The database system that is going to be built for the application is SQLite.

Project significance

Sentiment And Personality Test is web application which developed website for the sentiment and personality test. It's developed on Python programming language which is easy to understand and update in future.

Expected Output

Sentiment And Personality Test expected to be well manage and proper aplication. There should not have any errors occur for the predicting the output from the model as it learn from the past dataset.

Conclusion

Sentiment And Personality Test is one of the the intesting project and shows the power of natural language processing where user can check their personality and sentiment test which is work on the model trained by the past data taken from different platform.

3. System Analysis

To produce a web application based system that allow user to detect the sentiment and personality first the user have to register and login into application. It's show almost high accuracy and very easy to used.

Facts and Findings

While developing this application the lots of website are using API which is made by gaint company. Sentiment and Persoanlity test is work on MBTI persoanlitly. Personality typing is a system of categorizing people according to their tendencies to think and act in particular ways. Personality typing attempts to find the broadest, most important ways in which people are different, and make sense of these differences by sorting people into meaningful groups. Sentiment Test is also of three types I.e Sad, Happy and Normal.

Domain

Sentiment and Personality Test is the machine language based on python program language. It's based on Natural Language Programming. It's one of the best personality and sentiment test web application in the domain of NLP and ML.

Existing System

Studying the current system is a method that is used to gather the requirements in the research. The purpose of studying the current system is to identify the current data and input. The user have to study the data and detect that the sentence is sad, happy or netural. It's also detect the persoanlity test by the interview and many other thing so work load and many people are engaged and time consuming. The summary of the research made and are shown in below.

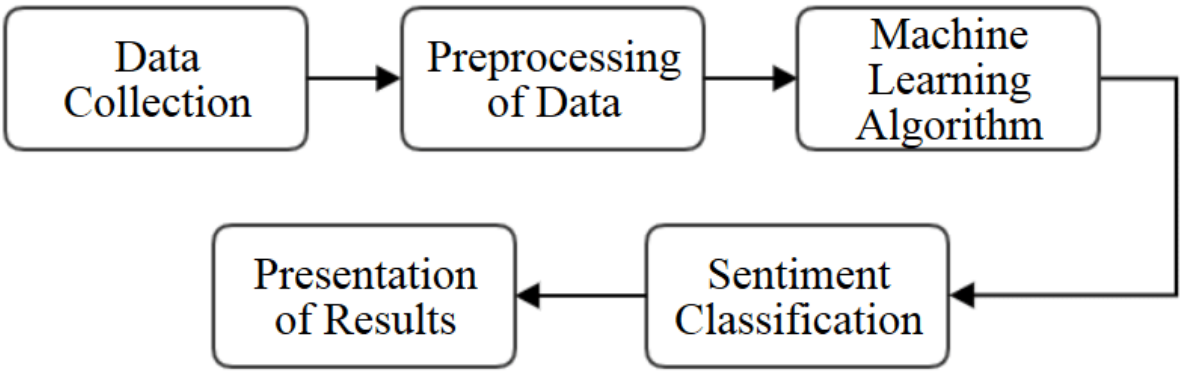
Case Study – Using Simulation to chose between Sentiment and Personality Test

This research seeks to determine the relationship between human emotional states and interaction, as well as to demonstrate that emotion has a greater impact on interaction than other factors. One of the primary goals of this research is to determine the relationship between human emotional states and interactions. As a result, it will aid the researcher in comprehending and inventing new techniques and technologies to realize the goal of affective computing as mentioned above. Our research team performed trials with real individuals who were interacting with different computer systems to evaluate this hypothesis, and they discovered the importance of the emotional state in human-computer interaction. In this experiment, the following are the goals that will be pursued. As we know that, there is a big difference in personality

type and their characteristics respectively. The personality type may be negative, positive, very negative, very positive and neutral in general, but the characteristics refer their properties. The personality is referred as the combination of characteristics which form a person's distinction. On the other side if the personality is being identified from unknown persons using a media like Facebook, twitter and other likelihood as platforms then it becomes critical to identify the personality. For that reason, we need the word and sentence data set with its description and context where these are used and when. There are some other called big-5 personality traits might be Openness, conscientiousness, extraversion, introversion, extroversion, agreeableness, sensitiveness and neuroticism, which makes a person different from others.

Technique

First, I gather the data from the internet and get with what type of persroanlity that text are carried on and from their I clean and preprocessing the data. After that I preprocesed the data and apply machine learning algorithm and detect and sentiment classification and personality test.



4. Technologies Used

Python [Front- End]:

Python is a widely-used, interpreted, object-oriented, and high-level programming language with dynamic semantics, used for general-purpose programming. It's everywhere, and people use numerous Python-powered devices on a daily basis, whether they realize it or not.

Python was created by Guido van Rossum, and first released on February 20, 1991. While you may know the python as a large snake, the name of the Python programming language comes from an old BBC television comedy sketch series called *Monty Python's Flying Circus*.

Python is omnipresent, and people use numerous Python-powered devices on a daily basis, whether they realize it or not. There are billions of lines of code written in Python, which means almost unlimited opportunities for code reuse and learning from well-crafted examples. What's more, there is a large and very active Python community, always happy to help.

Python is a great choice for career paths related to software development, engineering, DevOps, machine learning, data analytics, web development, and testing. What's more, there are also many jobs outside the IT industry that use Python. Since our lives are becoming more computerized every day, and the computer and technology areas previously associated only with technically gifted people are now opening up to non-programmers, Python has become one of the must-have tools in the toolbox of educators, managers, data scientists, data analysts, economists, psychologists, artists, and even secretaries.

There are also a couple of factors that make Python great for learning:

- It is easy to learn – the time needed to learn Python is shorter than for many other languages; this means that it's possible to start the actual programming faster;
- It is easy to use for writing new software – it's often possible to write code faster when using Python;
- It is easy to obtain, install and deploy – Python is free, open and multiplatform; not all languages can boast that.

In 1999, Guido van Rossum defined his goals for Python:

- an easy and intuitive language just as powerful as those of the major competitors;
- open source, so anyone can contribute to its development;
- code that is as understandable as plain English;
- suitable for everyday tasks, allowing for short development times.

Python is a great choice for:

- Web and Internet development (e.g., Django and Pyramid frameworks, Flask and Bottle micro-frameworks)
- Scientific and numeric computing (e.g., SciPy – a collection of packages for the purposes of mathematics, science, and engineering; Ipython – an interactive shell that features editing and recording of work sessions)
- Education (it's a brilliant language for teaching programming!)
- Desktop GUIs (e.g., wxWidgets, Kivy, Qt)
- Software Development (build control, management, and testing – Scons, Buildbot, Apache Gump, Roundup, Trac)
- Business applications (ERP and e-commerce systems – Odoo, Tryton)
- Games (e.g., Battlefield series, Sid Meier's Civilization IV…), websites and services (e.g., Dropbox, UBER, Pinterest, BuzzFeed…)

Python Django

Django is a high-level Python web framework that enables rapid development of secure and maintainable websites. Built by experienced developers, Django takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel. It is free and open source, has a thriving and active community, great documentation, and many options for free and paid-for support.

Django helps you write software that is:

Complete

Django follows the "Batteries included" philosophy and provides almost everything developers might want to do "out of the box". Because everything you need is part of the one "product", it all works seamlessly together, follows consistent design principles, and has extensive and up-to-date documentation.

Versatile

Django can be (and has been) used to build almost any type of website — from content management systems and wikis, through to social networks and news sites. It can work with any client-side framework, and can deliver content in almost any format (including HTML, RSS feeds, JSON, and XML).

Internally, while it provides choices for almost any functionality you might want (e.g. several popular databases, templating engines, etc.), it can also be extended to use other components if needed.

Secure

Django helps developers avoid many common security mistakes by providing a framework that has been engineered to "do the right things" to protect the website automatically. For example, Django provides a secure way to manage user accounts and passwords, avoiding common mistakes like putting session information in cookies where it is vulnerable (instead cookies just contain a key, and the actual data is stored in the database) or directly storing passwords rather than a password hash.

A password hash is a fixed-length value created by sending the password through a cryptographic hash function. Django can check if an entered password is correct by running it through the hash function and comparing the output to the stored hash value. However due to the "one-way" nature of the function, even if a stored hash value is compromised it is hard for an attacker to work out the original password.

Django enables protection against many vulnerabilities by default, including SQL injection, cross-site scripting, cross-site request forgery and clickjacking (see Website security for more details of such attacks).

Scalable

Django uses a component-based "shared-nothing" architecture (each part of the architecture is independent of the others, and can hence be replaced or changed if needed). Having a clear separation between the different parts means that it can scale for increased traffic by adding hardware at any level: caching servers, database servers, or application servers. Some of the busiest sites have successfully scaled Django to meet their demands (e.g. Instagram and Disqus, to name just two).

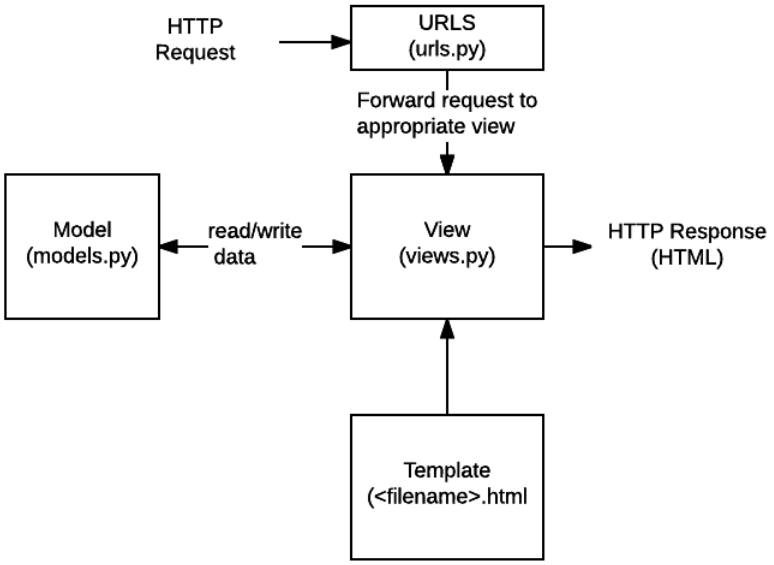
Maintainable

Django code is written using design principles and patterns that encourage the creation of maintainable and reusable code. In particular, it makes use of the Don't Repeat Yourself (DRY) principle so there is no unnecessary duplication, reducing the amount of code. Django also promotes the grouping of related functionality into reusable "applications" and, at a lower level, groups related code into modules (along the lines of the Model View Controller (MVC) pattern).

Portable

Django is written in Python, which runs on many platforms. That means that you are not tied to any particular server platform, and can run your applications on many flavors of Linux, Windows, and macOS. Furthermore, Django is well-supported by many web hosting providers, who often provide specific infrastructure and documentation for hosting Django sites.

In a traditional data-driven website, a web application waits for HTTP requests from the web browser (or other client). When a request is received the application works out what is needed based on the URL and possibly information in POST data or GET data. Depending on what is required it may then read or write information from a database or perform other tasks required to satisfy the request. The application will then return a response to the web browser, often dynamically creating an HTML page for the browser to display by inserting the retrieved data into placeholders in an HTML template.



URLs: While it is possible to process requests from every single URL via a single function, it is much more maintainable to write a separate view function to handle each resource. A URL mapper is used to redirect HTTP requests to the appropriate view based on the request URL. The URL mapper can also match particular patterns of strings or digits that appear in a URL and pass these to a view function as data.

- **View:** A view is a request handler function, which receives HTTP requests and returns HTTP responses. Views access the data needed to satisfy requests via *models*, and delegate the formatting of the response to *templates*.
- **Models:** Models are Python objects that define the structure of an application's data, and provide mechanisms to manage (add, modify, delete) and query records in the database.
- **Templates:** A template is a text file defining the structure or layout of a file (such as an HTML page), with placeholders used to represent actual content. A *view* can dynamically create an HTML page using an HTML template, populating it with data from a *model*. A template can be used to define the structure of any type of file; it doesn't have to be HTML!

Database Design

SQLite Database:SQLite is an in-process library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. It is a database, which is zero-configured, which means like other databases you do not need to configure it in your system.
SQLite engine is not a standalone process like other databases, you can link it statically or dynamically as per your requirement with your application. SQLite accesses its storage files directly.

Why SQLite?

- SQLite does not require a separate server process or system to operate (serverless).
- SQLite comes with zero-configuration, which means no setup or administration needed.
- A complete SQLite database is stored in a single cross-platform disk file.
- SQLite is very small and light weight, less than 400KiB fully configured or less than 250KiB with optional features omitted.
- SQLite is self-contained, which means no external dependencies.
- SQLite transactions are fully ACID-compliant, allowing safe access from multiple processes or threads.
- SQLite supports most of the query language features found in SQL92 (SQL2) standard.
- SQLite is written in ANSI-C and provides simple and easy-to-use API.
- SQLite is available on UNIX (Linux, Mac OS-X, Android, iOS) and Windows (Win32, WinCE, WinRT).

SQLite Limitations

There are few unsupported features of SQL92 in SQLite which are listed in the following table.

Sr.No.	Feature & Description
1	**RIGHT OUTER JOIN** Only LEFT OUTER JOIN is implemented.
2	**FULL OUTER JOIN** Only LEFT OUTER JOIN is implemented.
3	**ALTER TABLE** The RENAME TABLE and ADD COLUMN variants of the ALTER TABLE command are supported. The DROP COLUMN, ALTER COLUMN, ADD CONSTRAINT are not supported.
4	**Trigger support** FOR EACH ROW triggers are supported but not FOR EACH STATEMENT triggers.
5	**VIEWs** VIEWs in SQLite are read-only. You may not execute a DELETE, INSERT, or UPDATE statement on a view.
6	**GRANT and REVOKE** The only access permissions that can be applied are the normal file access permissions of the underlying operating system.

SQLite Commands

The standard SQLite commands to interact with relational databases are similar to SQL. They are CREATE, SELECT, INSERT, UPDATE, DELETE and DROP. These commands can be classified into groups based on their operational nature −

DDL - Data Definition Language

Sr.No.	Command & Description
1	**CREATE** Creates a new table, a view of a table, or other object in database.
2	**ALTER** Modifies an existing database object, such as a table.
3	**DROP** Deletes an entire table, a view of a table or other object in the database.

DML - Data Manipulation Language

Sr.No.	Command & Description
1	**INSERT** Creates a record
2	**UPDATE** Modifies records
3	**DELETE** Deletes records

DQL - Data Query Language

Sr.No.	Command & Description
1	**SELECT** Retrieves certain records from one or more tables

Jupyter Notebook

The Jupyter Notebook is an incredibly powerful tool for interactively developing and presenting data science projects. This article will walk you through how to use Jupyter Notebooks for data science projects and how to set it up on your local machine.

A notebook integrates code and its output into a single document that combines visualizations, narrative text, mathematical equations, and other rich media. In other words: it's a single document where you can run code, display the output, and also add explanations, formulas, charts, and make your work more transparent, understandable, repeatable, and shareable.

Using Notebooks is now a major part of the data science workflow at companies across the globe. If your goal is to work with data, using a Notebook will speed up your workflow and make it easier to communicate and share your results.

Jupyter Notebooks can also act as a flexible platform for getting to grips with pandas and even Python, as will become apparent in this tutorial.

We will:

- Cover the basics of installing Jupyter and creating your first notebook
- Delve deeper and learn all the important terminology
- Explore how easily notebooks can be shared and published online.

The short answer: each .ipynb file is one notebook, so each time you create a new notebook, a new .ipynb file will be created.

5. General Description

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.

The Meyers-Briggs Type Indicator (MBTI) is **a self-help assessment test which helps people gain insights about how they work and learn**. It is a framework for relationship-building, developing positivism, and achieving excellence.

Product Perspective:

Persoanlity and Sentiment test is a web application which will be operate by the user and control by admistration. It's support to many pheripharal device like keyboard, monitor, mobile, etc.

Product Functions:

- It is very easy to use.
- It saves a lot of time, money and labour.
- The software acts as an office that is open 24/7.
- It helps to know the characterstics of people.
- It helps to know the sentiments of the people.

User Characteristics:

This application will be operate by basic technical skills of the user where user can easily operate application properly. User will be very interest to used this application and user will also know about themselves by using this application.

General Constraints:

While designing this system, we can't able to imagine what is going to the user mind. User can also write false information that manipulate the application.

Assumptions and Dependencies:

While developing this software, we assume that the computer is having preinstall minimum web browser in the system and with python installed.

6. Functional Requirements and Non-functional Requirements

Requirement analysis is a software engineering technique that is composed of the various tasks that determine the needs or conditions that are to be met for a new or altered product, taking into consideration the possible conflicting requirements of the various users.

I. Functional Requirements:

Functional requirements are those requirements that are used to illustrate the internal working nature of the system, the description of the system, and explanation of each subsystem. It consists of what task the system should perform, the processes involved, which data should the system holds and the interfaces with the user.
The functional requirements identified are:

- **User's registration:** The system should allow new users to register and test the personality and sentiments.
- **Personality Test:** User is allow to type the text in the textbox and used to perdict what type of persoanlity user is having.
- **Automatic update to database once reservation is made or new customer registered:** Whenever there's new reservation or new registration, the system should be able update the database without any additional efforts from the admin.
- **Sentiment Test:** User is allow to type the text in the textbox and used to predict what type of sentiment test about the user.
- **Language Translate:** User is allow to type the text in the textbox in english and help to convert into Kannada Language.
- **Admin Mode:** It is used to keep the record of administration and manage the whole application. It's also called super user.

II. Non-Functional Requirements

It describes aspects of the system that are concerned with how the system provides the functional requirements.
They are:

- **Security:**

  The subsystem should provide a high level of security and integrity of the data held by the system, only authorized personnel of the company can gain access to the company's secured page on the system; and only users with valid password and username can login to view user's page.

- **Performance and Response time:**

The system should have high performance rate when executing user's input and should be able to provide feedback or response within a short time span usually 50 seconds for highly complicated task and 20 to 25 seconds for less complicated task.

- **Error handling:**

  Error should be considerably minimized and an appropriate error message that guides the user to recover from an error should be provided. Validation of user's input is highly essential. Also the standard time taken to recover from an error should be 15 to 20 seconds.

- **Availability:**

  This system should always be available for access at 24 hours, 7 days a week. Also in the occurrence of any major system malfunctioning, the system should be available in 1 to 2 working days, so that the business process is not severely affected.

- **Ease of use:**

  Considered the level of knowledge possessed by the users of this system, a simple but quality user interface should be developed to make it easy to understand and required less training.

7. System Architecture

Here, we have one Desktop application which is connected with Database and one is for User who operate the Software.

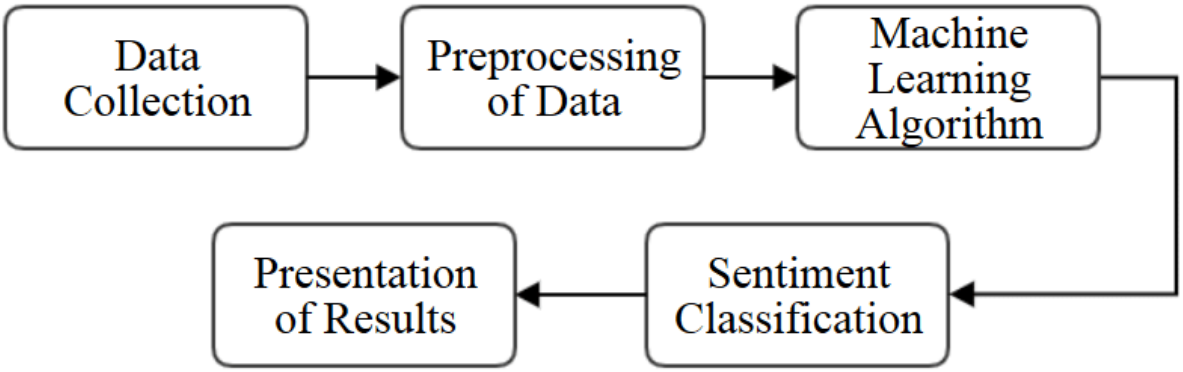


Fig. Of System Architecture

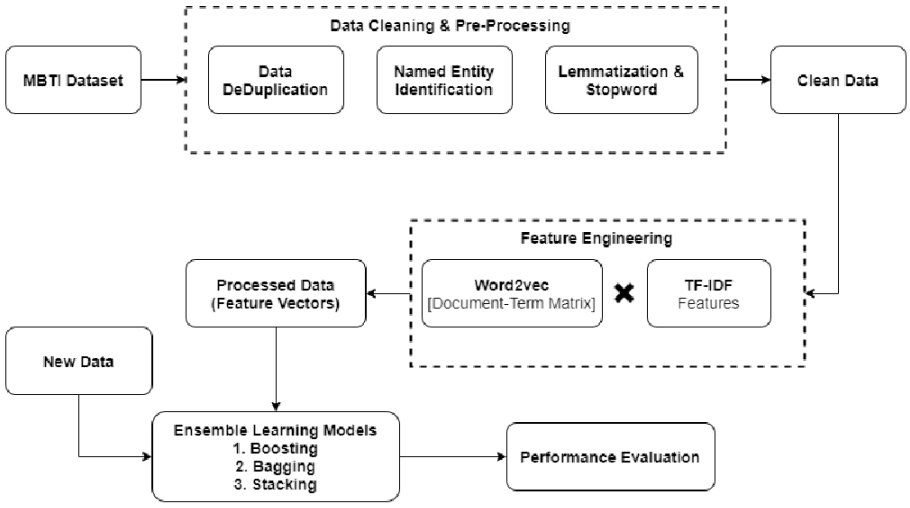


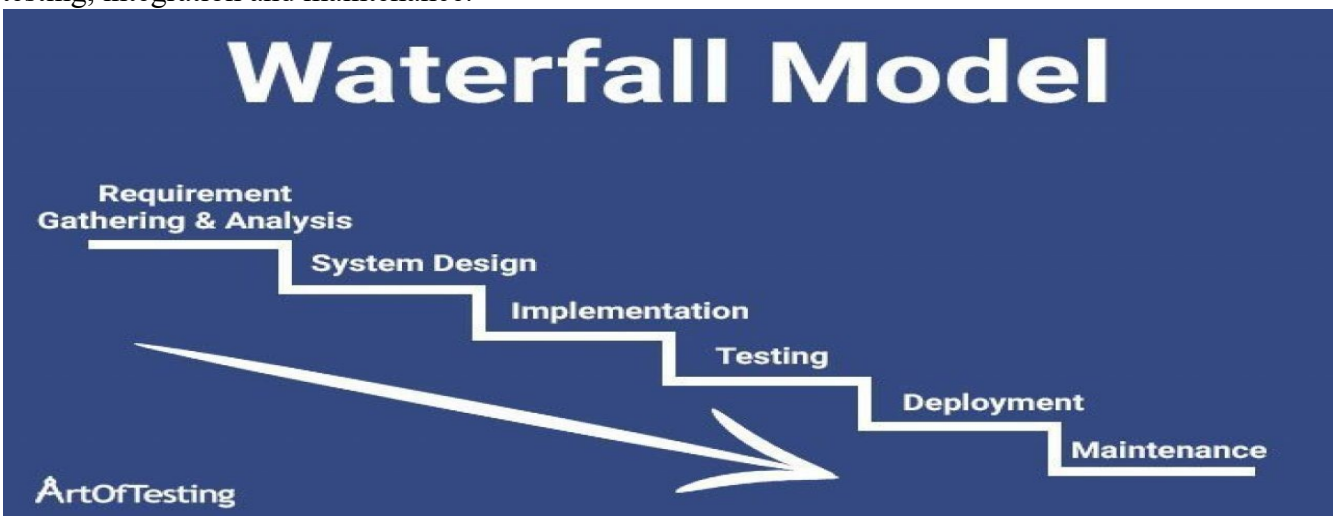
Figure 1 Personality Analysis Framework

8. System Model

The SDLC(Software Development Life Cycle) method will be used to defining tasks performed at each step in the software development process. SDLC is a structure followed by a development team within the software organization. It consists of a detailed plan describing how to develop, maintain and replace specific software.

Waterfall Model:

For SDLC, we used **Waterfall Model** which is developed by W. W. Royce and defined as the development of sequential completion of one steps after the other like analysis, design, implementation, testing, integration and maintenance.



I. Requirement Gathering and Analysis:

We gather the information of the manual Rental System and analysis the drawbacks. So, to overcome that drawback we are going to implement all the services in VRMS(Vehicle Rental Management System) . We gather the customer details and requirement about the process where we perform different feasibility study like

Feasibility Study:

A feasibility study is an analysis that considers all of a project's relevant factors—including economic, technical, legal, and scheduling considerations—to ascertain the likelihood of completing the project successfully. We perform different feasibility study whether to take project further or not. They are:

Economic Feasibility:

After studying all the requirement we came to know that software is profitable or not for us.

Technical Feasibility:

We also study, what type of technology to used in the project and to developed the project.

Operational Feasibility:

We also research on whether the staff can operate it properly or not.

II. Software Design:

Software Design is an iterative process through which requirements are translated into a "blueprint" for developing the software.

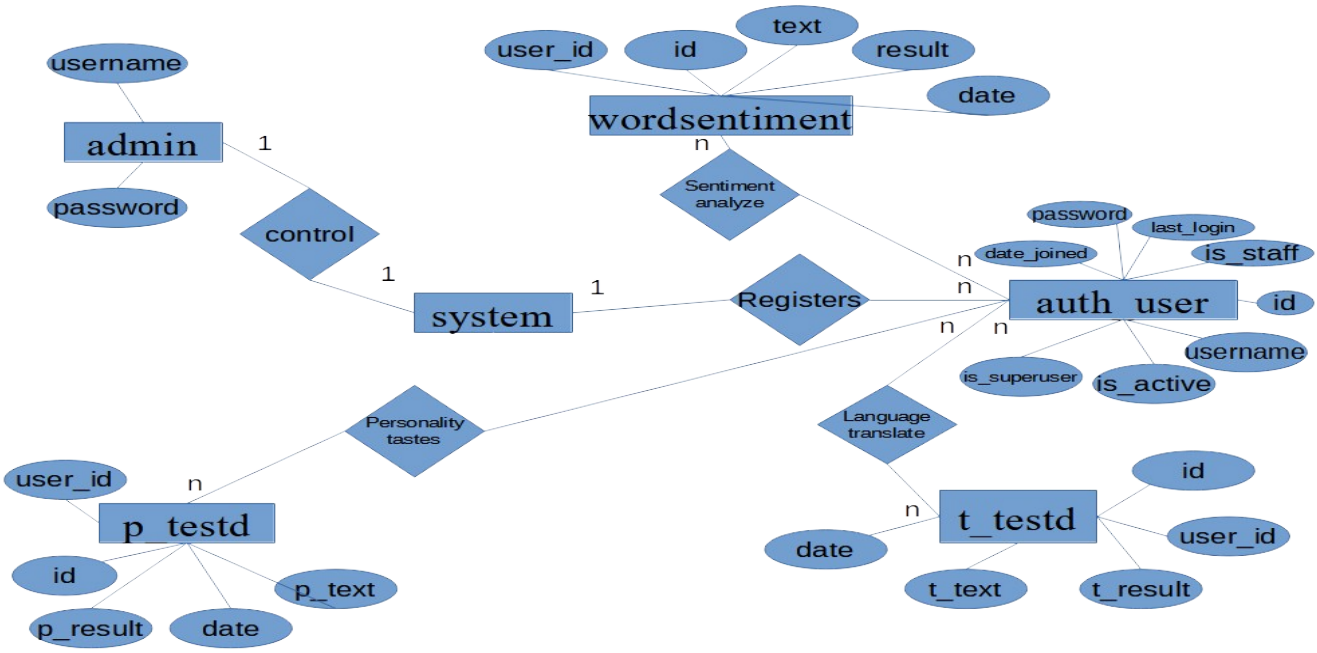


Fig. Of ER Diagram

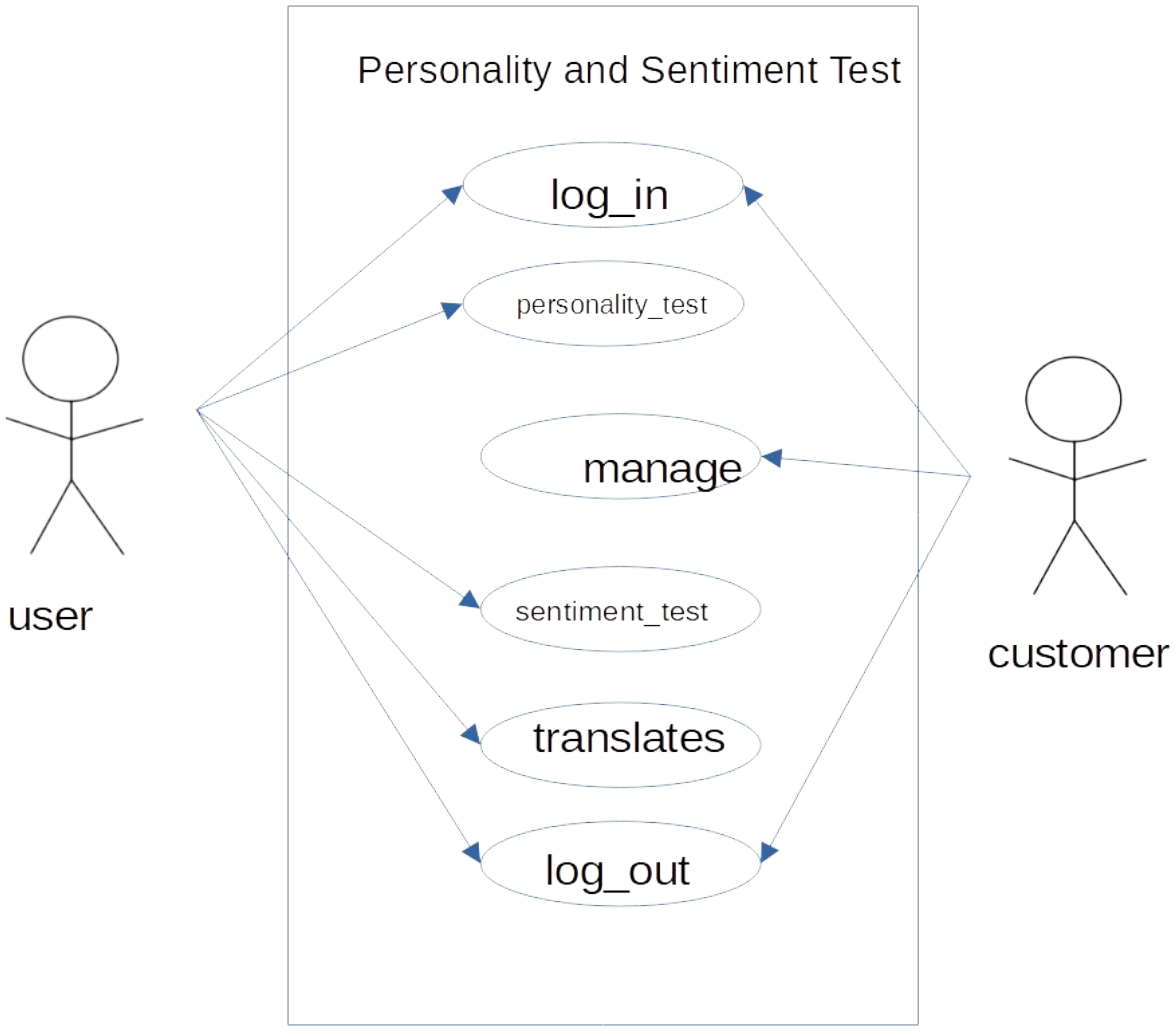


fig. Of Use Case Digram

III. Implementation

We implement the plan of the our project after designing the process by using Visual Basic 6.0 and MS Access software where we coded in Basic programming language. We used Adodc and DAO connection to communicate with database.

IV. Testing

Testing is the process of analyzing a software item to detect the differences between existing and required conditions and to evaluate the features of the software item.
We have perform following testing in the software.

- **Unit Testing:**

  Individual components are tested to ensure that they operate correctly or not. Each component is tested independently, without referring other system components. We have tested each and every component of the software like command button, form behavior, etc.

- **Module Testing:**

  A module is a collection of dependent components such as an object class, an abstract data type or some collection of procedures and function. We have tested each and every module like Staff/User module, Customer Module, etc how they are interacting with software.

- **Integration Testing:**

  This phase involves testing collection of modules which have been integrated into sub-systems. We integrated each and every component and module and tested colletively to confirm wheather the system working correctly or not.

- **Acceptance Testing:**

  This is the final stage in the testing process before the system is accepetd for operational use. In this testing, we provide the different input to see the how data is interacting with the software.

- **Validation:**

  Validation is the process of checking that what has been specified in what the user actually wanted.We perform different validation technique to get proper input from the user like phone number validation, serial number validation, full name validation, email validation to work the software effiectively.

V. Maintenance:

If any problems will occurs in future, we and our team ensure for the proper maintenance of the software and the User side also have to cooperate and interact with software properly.

Python Django FrameWork Model:

Django, a Python framework to create web applications, is based on Model-View-Template (MVT) architecture. **MVT** is a software design pattern for developing a web application. It consists of the following three entities:

1. Model
2. View
3. Template

Model

A **Model** is an object that defines the structure of the data in the Django application.

It is responsible for maintaining the entire application's data for which it provides various mechanisms to add, update, read and delete the data in the database.

View

A **View** is a handler function that accepts HTTP requests, processes them, and returns the HTTP response.

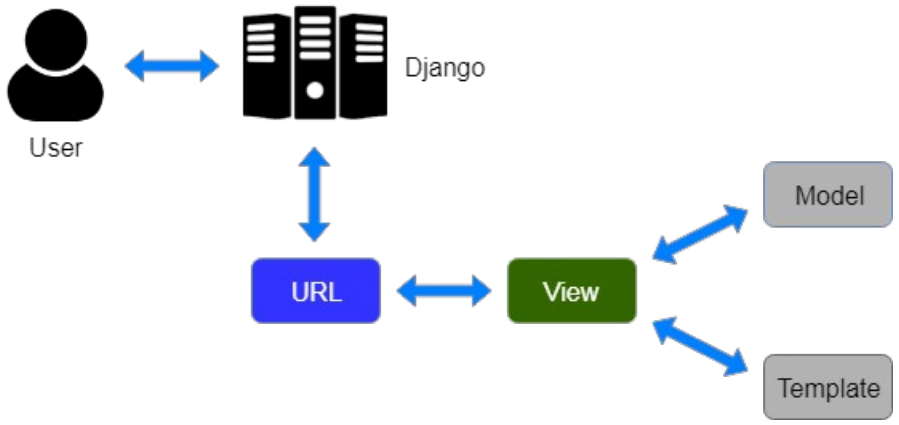
It retrieves the necessary data to fulfill the request using Models and renders them on the user interface using Templates.

It can also create an HTML page using an HTML template dynamically, and populate it with data fetched from the model.

Template

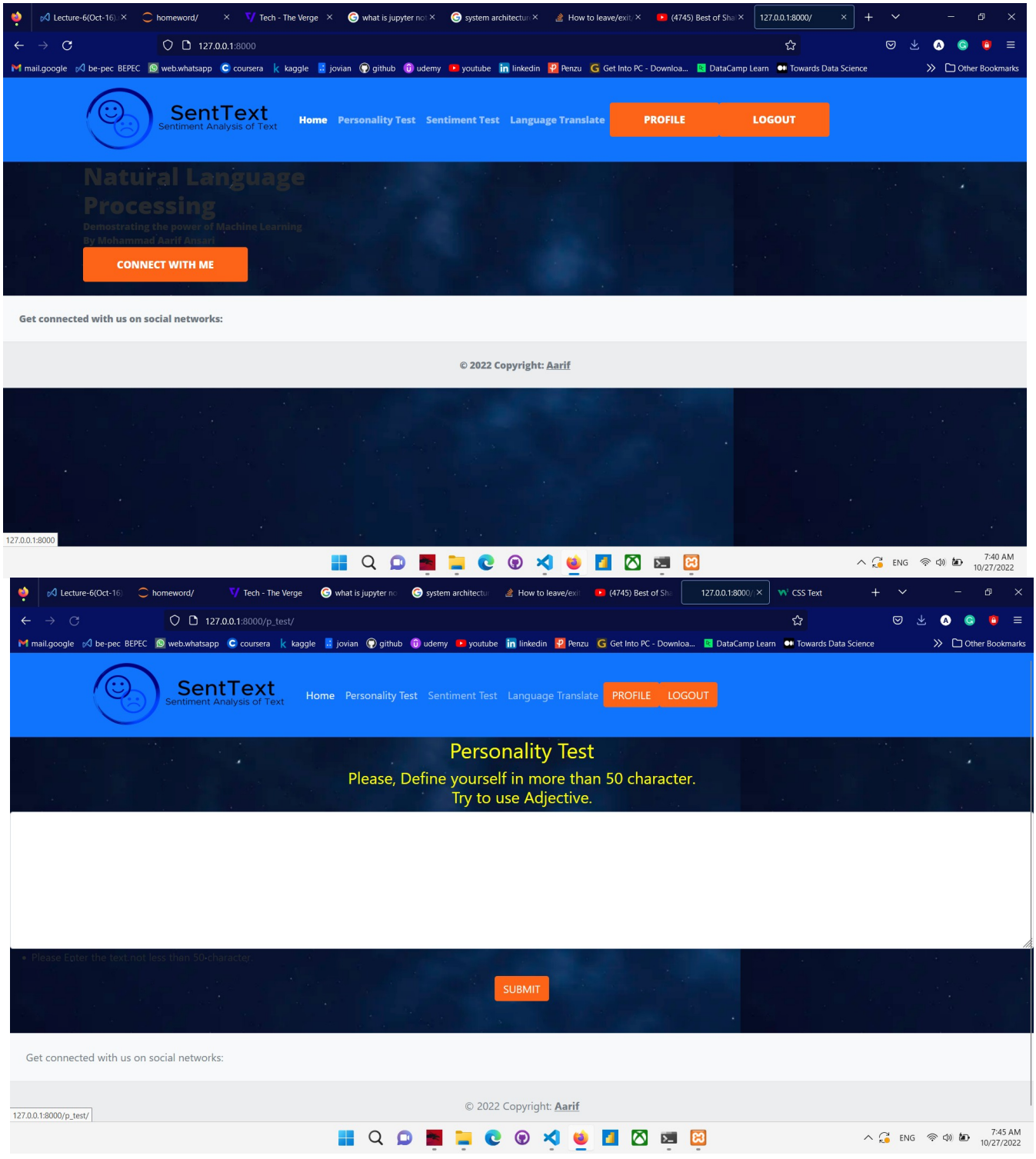
A **Template** is a text file that defines the structure or layout of the user interface. The text file can be any type of file; for example HTML, XML, etc.

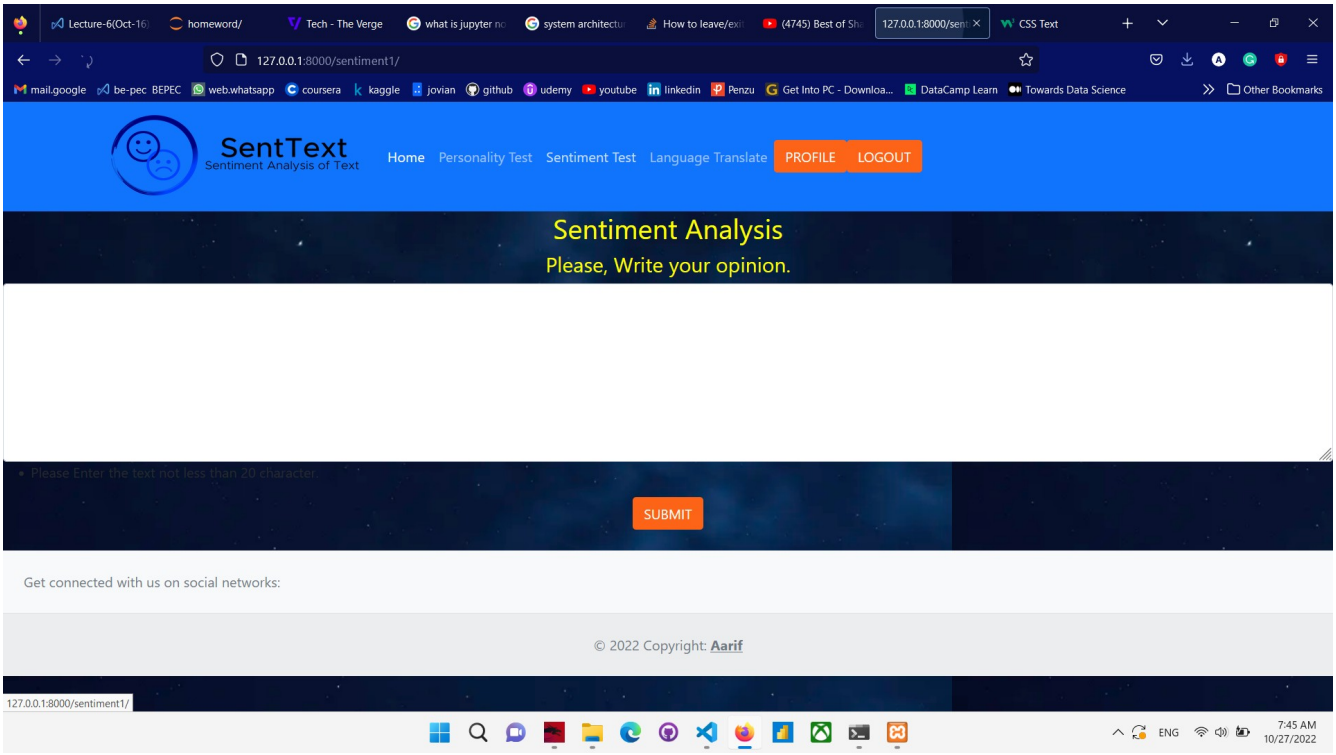
It can accept data from the view and render it using jinja syntax.

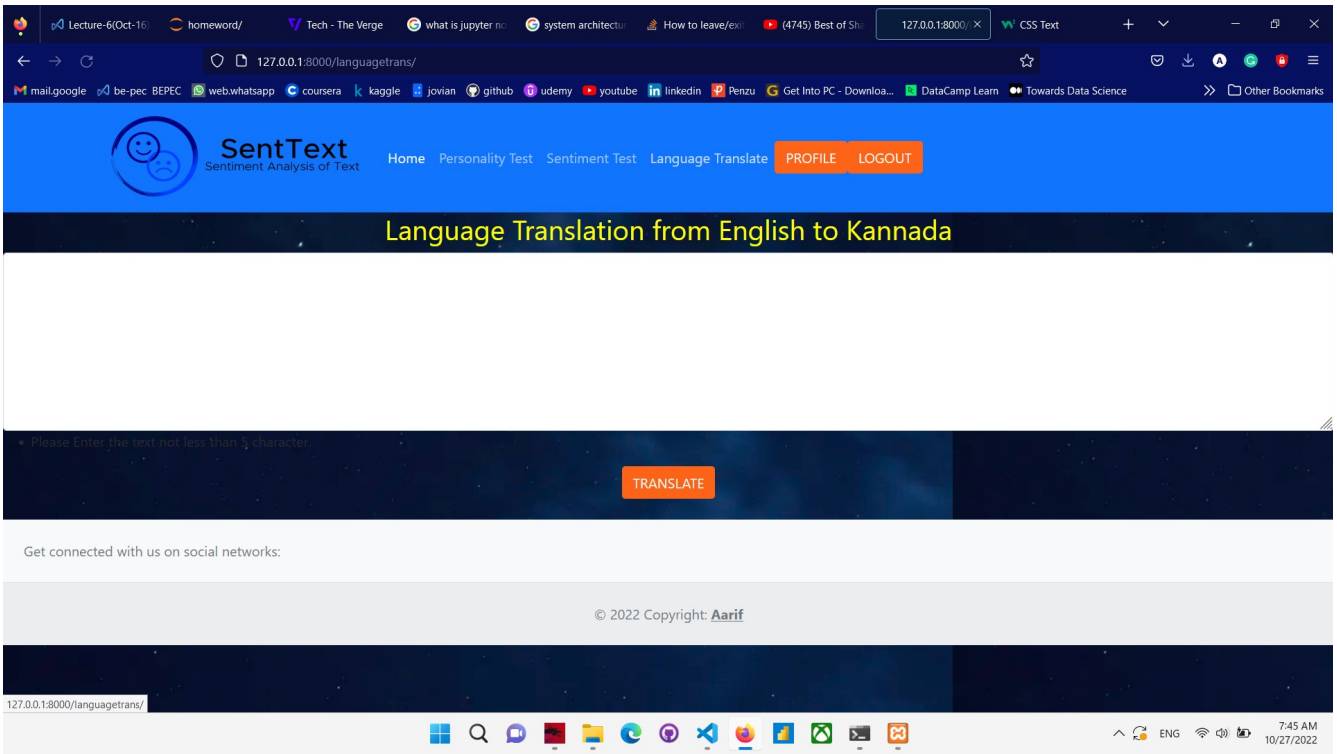


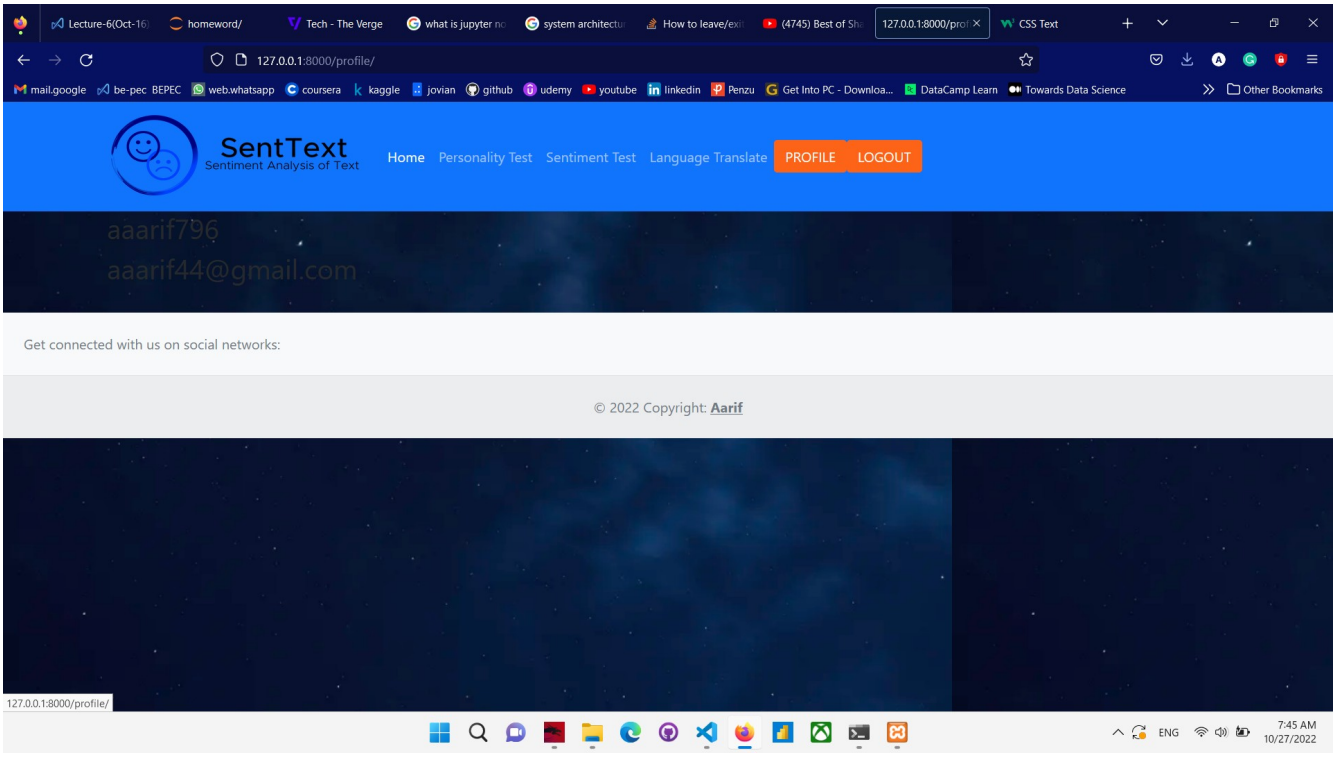
- The user interacts with a Django application using a URL that is passed to the MVT architecture. A URL mapper is used to redirect the requests to the appropriate view based on the request URL.
- If an appropriate view is found, it will be invoked.
- The View will interact with the Model and retrieve the necessary data from the database via Model.
- The View will render an appropriate template along with the retrieved data to the user.

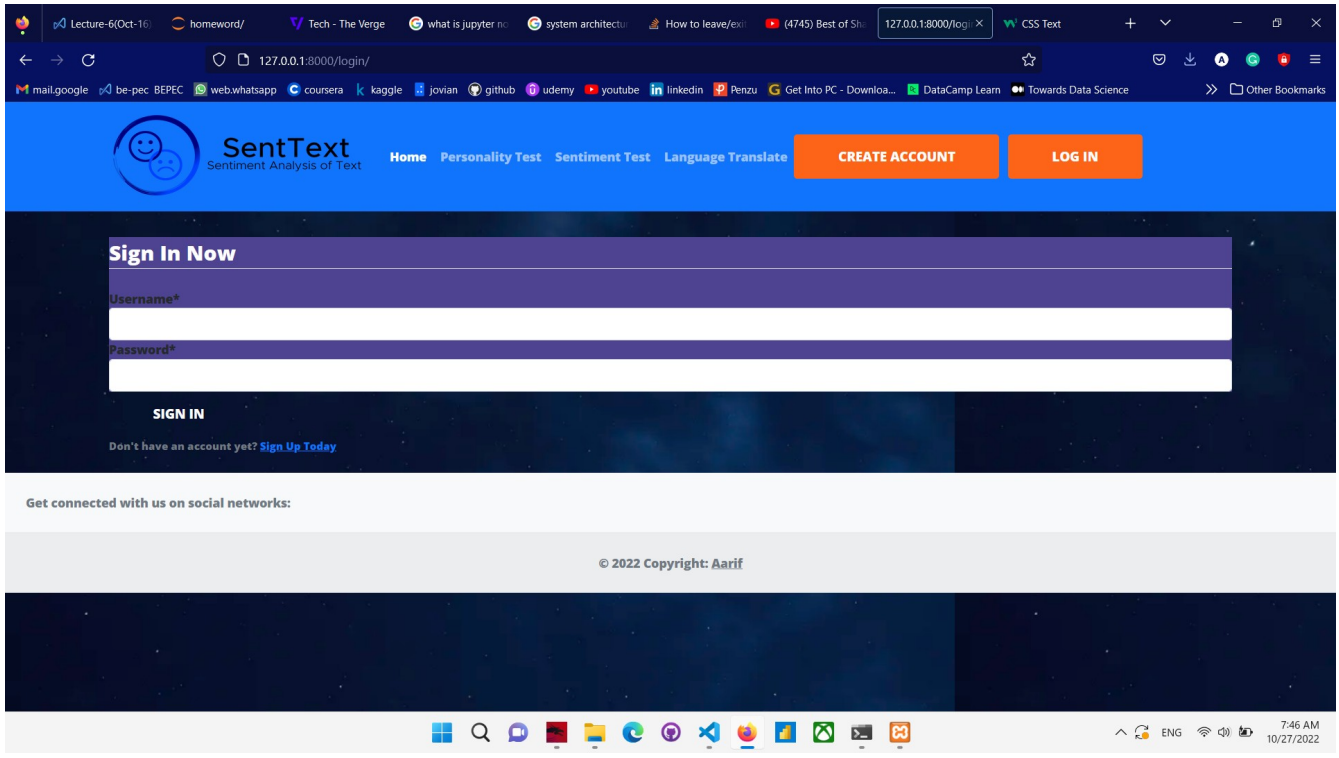
9. Screen Shots

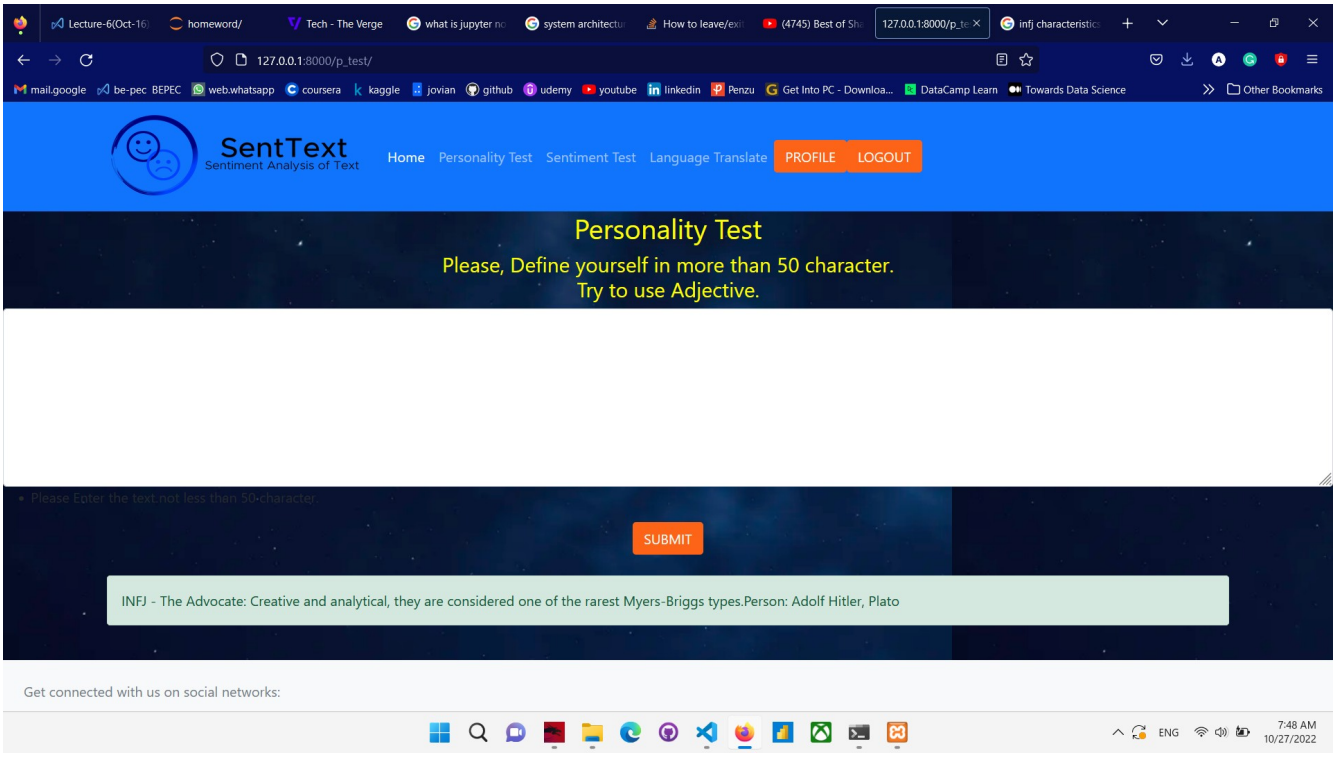


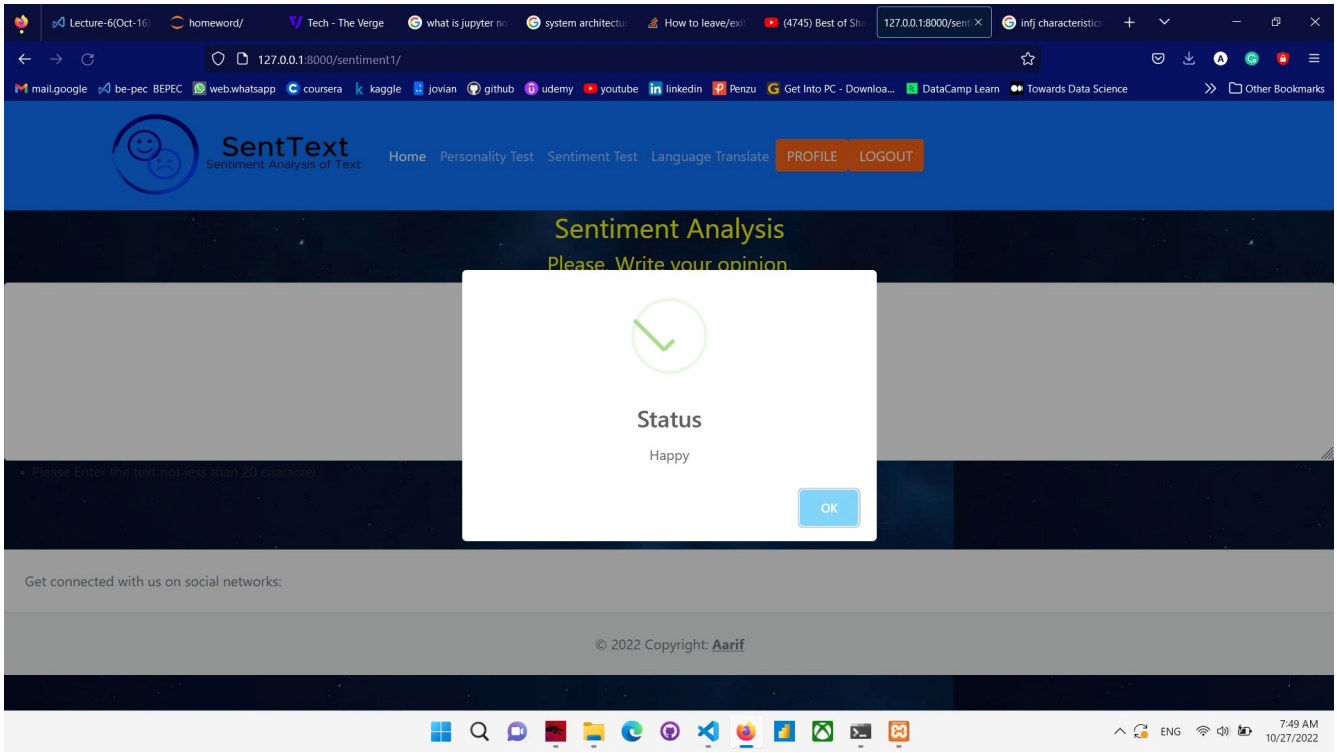

127.0.0.1:8000/sentiment1/
SentText
Sentiment Analysis of Text
Home
Personality Test
Sentiment Test
Language Translate
PROFILE
LOGOUT
Sentiment Analysis
Please, Write your opinion.
Please Enter the text not less than 20 character.
SUBMIT
Get connected with us on social networks:
© 2022 Copyright: Aarif
127.0.0.1:8000/sentiment1/
7:45 AM
10/27/2022

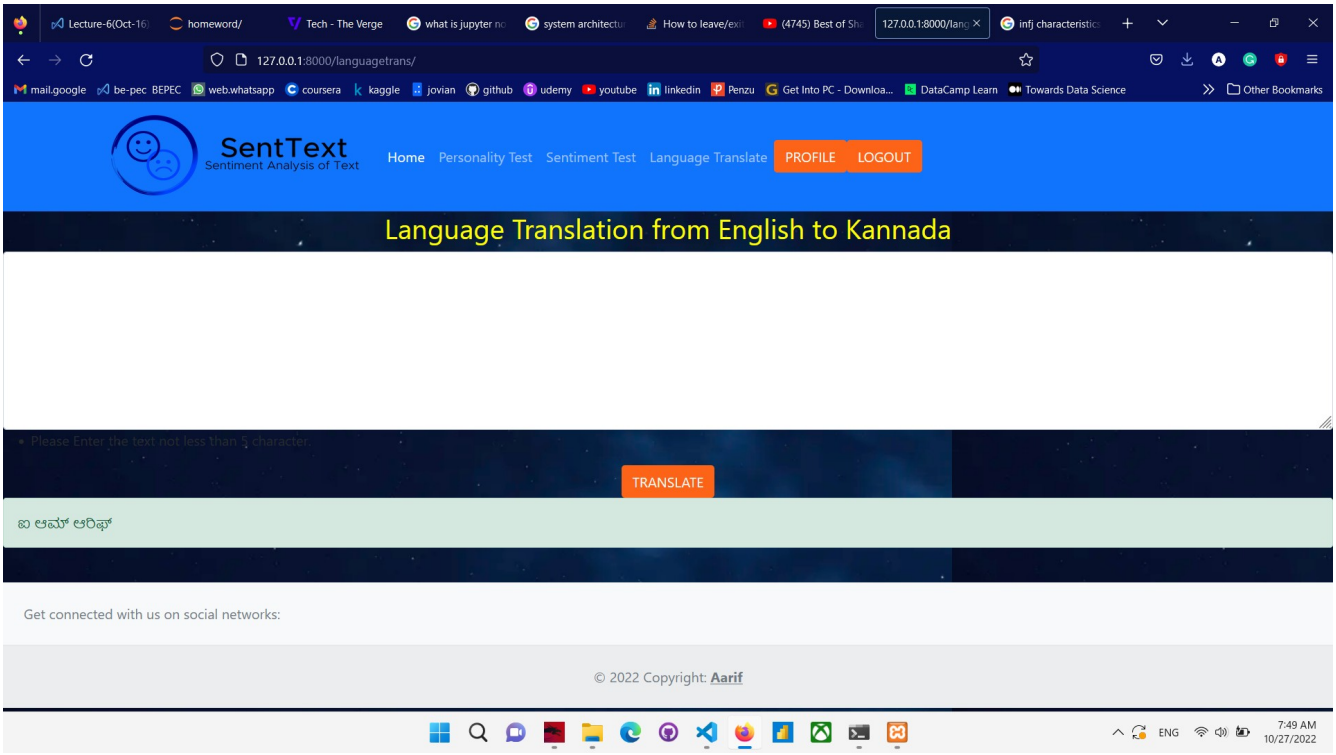

127.0.0.1:8000/languagetrans/
SentText
Sentiment Analysis of Text
Home
Personality Test
Sentiment Test
Language Translate
PROFILE
LOGOUT
Language Translation from English to Kannada
Please Enter the text not less than 5 character.
TRANSLATE
Get connected with us on social networks:
© 2022 Copyright: Aarif
127.0.0.1:8000/languagetrans/
7:45 AM
10/27/2022


127.0.0.1:8000/profile/
SentText
Sentiment Analysis of Text
Home
Personality Test
Sentiment Test
Language Translate
PROFILE
LOGOUT
aaarif796
aaarif44@gmail.com
Get connected with us on social networks:
© 2022 Copyright: Aarif
7:45 AM
10/27/2022


127.0.0.1:8000/login/
SentText
Sentiment Analysis of Text
Home
Personality Test
Sentiment Test
Language Translate
CREATE ACCOUNT
LOG IN
Sign In Now
Username*
Password*
SIGN IN
Don't have an account yet? Sign Up Today
Get connected with us on social networks:
© 2022 Copyright: Aarif
7:46 AM
10/27/2022


127.0.0.1:8000/p_test/
SentText
Sentiment Analysis of Text
Home
Personality Test
Sentiment Test
Language Translate
PROFILE
LOGOUT
Personality Test
Please, Define yourself in more than 50 character.
Try to use Adjective.
SUBMIT
INFJ - The Advocate: Creative and analytical, they are considered one of the rarest Myers-Briggs types.Person: Adolf Hitler, Plato
Get connected with us on social networks:


127.0.0.1:8000/sentiment1/
SentText
Sentiment Analysis of Text
Home
Personality Test
Sentiment Test
Language Translate
PROFILE
LOGOUT
Sentiment Analysis
Status
Happy
OK
Get connected with us on social networks:
© 2022 Copyright: Aarif


127.0.0.1:8000/languagetrans/
SentText
Sentiment Analysis of Text
Home
Personality Test
Sentiment Test
Language Translate
PROFILE
LOGOUT
Language Translation from English to Kannada
Please Enter the text not less than 5 character.
TRANSLATE
ಐ ಆಮ್ ಆರಿಫ್
Get connected with us on social networks:
© 2022 Copyright: Aarif
ENG
7:49 AM
10/27/2022

10. Database

auth_user

Id	username	email	password	date_joined	last_login	is_superuser	is_staff	is_active
Number	Varchar	Varchar	Varchar	Date	Date	Bool	Bool	bool

authentication_wordsentiment

user_id	id	Text	date	result
number	number	Varchar	date	varchar

authentication_t_testd

user_id	id	t_text	date	t_result
number	number	Varchar	date	varchar

authentication_p_testd

user_id	id	p_text	date	p_result
number	number	Varchar	date	varchar

auth_user:

db.sqlite3 ▶ auth_user

Reset Filters | Records: 4 | Search 4 records...

	id	password	last_login	is_superuser	username	last...	email	is_staff	is_active	date_joined	first_name
1	1	pbkdf2_sha256$39...	2022-10-27 02:17:5...	0	aaarif796		aaarif44@gmail.com	0	1	2022-09-03 18:42:2...	
2	3	pbkdf2_sha256$39...	2022-09-10 05:13:0...	0	Ajith303		Ajithkumar303@g...	0	1	2022-09-10 05:12:5...	
3	4	pbkdf2_sha256$39...	2022-10-21 14:59:1...	0	nawaz1		nawaz@gmail.com	0	1	2022-10-21 14:56:3...	
4	2	pbkdf2_sha256$39...	2022-10-21 15:02:2...	1	aarif		aarif@gmail.com	1	1	2022-09-04 04:53:0...	

authentication_wordsentiment:

db.sqlite3 ▶ authentication_wordsentiment

Reset Filters | Records: 2 | Search 2 records...

	id	text	date	result	user_id
1	89	I am fine what abou...	2022-10-21	Happy	2
2	90	Hi, today i am very ...	2022-10-27	Happy	1

authentication_t_testd:

db.sqlite3 ▶ authentication_t_testd

Reset Filters | Records: 1 | Search 1 records...

	id	t_text	date	t_result	user_id
	Search column...	Search column...	Search column...	Search column...	Search column...
1	1	I am Aarif	2022-10-27	ಐ ಆಮ್ ಆರಿಫ್	1

authentication_p_testd:

db.sqlite3 ▶ authentication_p_testd

Reset Filters | Records: 5 | Search 5 records...

	id	p_text	date	p_result	user_id
	Search column...	Search column...	Search column...	Search column...	Search column...
1	1	A Defender (ISFJ) is ...	2022-10-21	ISFJ	2
2	2	A Defender (ISFJ) is ...	2022-10-21	ISFJ	2
3	3	ISFJ - The Protector:...	2022-10-21	ISFJ	2
4	4	Hi, I am fine i am ve...	2022-10-21	INTP	2
5	5	INFJs are energized...	2022-10-27	INFJ	1

11. Coding

base.html

```
<!doctype html>
{% load static %}
<html lang="en">
  <head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <!--Css Style -->
    <link rel="stylesheet" href="{% static 'css/main.css' %}" type="text/css">
    <!-- Bootstrap CSS -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/css/bootstrap.min.css" rel="stylesheet"
integrity="sha384-eOJMYsd53ii+scO/bJGFsiCZc+5NDVN2yr8+0RDqr0Ql0h+rP48ckxlpbzKgwra6"
crossorigin="anonymous">
    <!--Font Link-->
    <link rel="preconnect" href="https://fonts.gstatic.com">
    <link href="https://fonts.googleapis.com/css2?
family=Montserrat:wght@500&family=Open+Sans:wght@800&display=swap" rel="stylesheet">
  </head>
  <body id="bg" style="background-image: url({% static 'images/bg.png' %});">
  {% load static %}
    <nav class="navbar navbar-expand-lg navbar-dark bg-primary" id="main-navbar">
       <div class="container">
         <a class="navbar-brand" href="{% url 'home' %}"><img src="{% static 'images/s_logo.png' %}" width="300"
height="100" bgcolor="white" alt="" ></a>
         <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNavAltMarkup"
aria-controls="navbarNavAltMarkup" aria-expanded="false" aria-label="Toggle navigation">
           <span class="navbar-toggler-icon"></span>
         </button>
         <div class="collapse navbar-collapse" id="navbarNavAltMarkup">
           <div class="navbar-nav">
             <a class="nav-link active" aria-current="page" href="{% url 'home' %}">Home</a>
             {% if user.is_authenticated %}
               <a class="nav-link" href="{% url 'p_test' %}">Personality Test</a>
               <a class="nav-link" href="{% url 'sentimentWord' %}" aria-current="page">Sentiment Test</a>
               <a class="nav-link" href="{% url 'languagetrans' %}" aria-current="page">Language Translate</a>
             {% else %}
               <a class="nav-link" href="">Personality Test</a>
               <a class="nav-link" href="" aria-current="page">Sentiment Test</a>
               <a class="nav-link" href="" aria-current="page">Language Translate</a>
             {% endif %}
           </div>
           <form class="d-flex" id="authstyle">
             {% if user.is_authenticated %}
               <a href="{% url 'profile' %}" class="btn" style="color: white; background-color: #fd5e14;"type="submit"
id="header-links"> Profile </a>
               <a href="{% url 'logout' %}" class="btn" style="color: white; background-color: #fd5e14;"type="submit"
id="header-links"> Logout </a>
             {% else %}
               <a href="{% url 'register' %}" class="btn" style="color: white; background-color: #fd5e14;"type="submit"
id="header-links"> Create Account </a>
```

```
            <a href="{% url 'login' %}" class="btn" style="color: white; background-color: #fd5e14; margin-left: 10px;
"type="submit" id="header-links">Log In</a>
          {% endif %}
          </form>
        </div>
      </div>
  </nav>
    <div>
      {% comment %} <div class="container">
       {% if messages %}
       {% for message in messages %}
         <div class="alert alert-{{message.tags}}">{{message}}</div>
       {% endfor %}
      {% endif %}
      </div> {% endcomment %}
      {% block content %}
      {% endblock content %}
    </div>
    <!-- Optional JavaScript; choose one of the two! -->
    <!-- Option 1: Bootstrap Bundle with Popper -->
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/js/bootstrap.bundle.min.js" integrity="sha384-
JEW9xMcG8R+pH31jmWH6WWP0WintQrMb4s7ZOdauHnUtxwoG2vI5DkLtS3qm9Ekf"
crossorigin="anonymous"></script>

    <!-- Option 2: Separate Popper and Bootstrap JS -->
    <!--
    <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.1/dist/umd/popper.min.js" integrity="sha384-
SR1sx49pcuLnqZUnnPwx6FCym0wLsk5JZuNx2bPPENzswTNFaQU1RDvt3wT4gWFG"
crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/js/bootstrap.min.js" integrity="sha384-
j0CNLUeiqtyaRmlzUHCPZ+Gy5fQu0dQ6eZ/xAww941Ai1SxSY+0EQqNXNE6DZiVc"
crossorigin="anonymous"></script>
    -->
    <!-- Footer -->
 <br>
<footer class="text-center text-lg-start bg-light text-muted" button="0" style="flex-direction:column;">
 <!-- Section: Social media -->
 <section class="d-flex justify-content-center justify-content-lg-between p-4 border-bottom">
  <!-- Left -->
  <div class="me-5 d-none d-lg-block">
    <span>Get connected with us on social networks:</span>
  </div>
  <!-- Left -->

  <!-- Right -->

  <!-- Right -->
 </section>
 <div class="text-center p-4" style="background-color: rgba(0, 0, 0, 0.05);">
  © 2022 Copyright:
  <a class="text-reset fw-bold" href="">Aarif</a>
 </div>
 <!-- Copyright -->
```

```
</footer>
<!-- Footer -->
  </body>
</html>
```

home.html

```
{% extends 'base.html' %}
{% load static %}
{% block content %}
<link rel="stylesheet" href="{% static 'css/main.css' %}">
   <div class="container">
         <div class="row g-0">
            <div class="col-sm-6 col-md-8" id="header-text" style="color:white;">
              <h1 id="header-h1">Natural Language <br> Processing </h1>
              <p id="header-paragraph">Demostrating the power of Machine Learning <br> By Mohammad Aarif Ansari</p>
              <a class="btn" href="https://www.linkedin.com/in/aarif-ansari-a6b869b6/" style="color: white; background-
color: #fd5e14;"type="submit" id="body-link-row">Connect with Me</a>
            </div>

            <div class="col-6 col-md-4" id="header-image"><img src="" alt=""></div>
          </div>

   </div>
{% endblock content %}
```

languagetrans.html:

```
{% extends 'base.html' %}
{% load crispy_forms_tags %}
{% block content%}
<form method="POST" action="">
  {% csrf_token %}
  <div class="form-group">
    <h2 style="color:yellow;text-align:center;">Language Translation from English to Kannada</h2>
  <textarea class="form-control" id="exampleFormControlTextarea1" name="text1" rows="8"></textarea>
  <label><ul><li>Please Enter the text not less than 5 character.</li></ul></label>
  <br><center>
  <button class="btn" style="color: white; background-color: #fd5e14;" type="submit" id="submit"> Translate </button>
  </center>

</div>
</form>
{% for message in messages%}
  {% if message.tags == 'success' %}
  <div id="a1" class="alert alert-{{message.tags}}">{{message}}</div>
  {% endif %}
{% endfor %}
{% endblock %}
```

login.html;

```
{% extends 'base.html' %}
{% load crispy_forms_tags %}
{% load static %}
```

```
{% block content %}
   <link rel="stylesheet" href="{% static 'css/main.css' %}">
   <div class="container" style="margin-top: 30px;">
   <form action="" method="POST">
     {% csrf_token %}
     <fieldset class="form-group" style="background-color:darkslateblue;">
        <legend class="border-bottom mb-4" style="color: white;">Sign In Now</legend>
        {{ form|crispy }}
     </fieldset>
     <div class="form-group" id="signup-button">
        <button class="btn ">Sign In</button>
     </div>
     <div class="form-group">
        <small class="text-muted ml-3" >Don't have an account yet? <a href="{% url 'register' %}">Sign Up
Today</a></small>
     </div>
   </form>
   </div>
{% endblock %}
```

logout.html:

```
{% extends 'base.html' %}
{% load crispy_forms_tags %}

{% block content %}
  <div class='container' style="color:white;">
    <h1>You have been logged out</h1>
    <h5><a href="{% url 'login' %}">LogIn In Again</a></h5>
  </div>
{% endblock %}
```

p_test.html:

```
{% extends 'base.html' %}
{% load crispy_forms_tags %}
{% block content%}
<script src="https://unpkg.com/sweetalert/dist/sweetalert.min.js"></script>
<form method="POST" action="">
  {% csrf_token %}
  <div class="form-group">
    <h2 style="color:yellow;text-align:center;">Personality Test</h2>
    <h4 style="color:yellow;text-align:center;">Please, Define yourself in more than 50 character.<br>Try to use
Adjective.</h4>
  <textarea class="form-control" id="exampleFormControlTextarea1" name="text1" rows="8"></textarea>
  <label><ul><li>Please Enter the text not less than 50 character.</li></ul></label>
  <br><center>
  <button class="btn" style="color: white; background-color: #fd5e14;" type="submit" id="submit"> Submit </button>
  </center>
</div>
</form>
<div>
```

```
<br>
<div class="container">
    {% if messages %}
    {% for message in messages %}
      <div id="a1" class="alert alert-{{message.tags}}">{{message}}</div>
    {% endfor %}
  {% endif %}
</div>
{% endblock %}
```

profile.html:

```
{% extends 'base.html' %}
{% load crispy_forms_tags %}
{% block content %}
    <div class='container'>
    <h2>{{user.username}}</h2>
    <h2>{{user.email}}</h2>
    </div>
{% endblock %}
```

register.html

```
{% extends 'base.html' %}
{% load crispy_forms_tags %}
{% load static %}

{% block content %}
    <link rel="stylesheet" href="{% static 'css/main.css' %}">
    <div class="container" style="margin-top: 30px;">
    <form action="" method="POST">
      {% csrf_token %}
      <fieldset class="form-group" style="background-color:darkslateblue;">
          <legend class="border-bottom mb-4" style="color: white;">Join Now</legend>
          {{ form|crispy }}
      </fieldset>
      <div class="form-group" id="signup-button">
          <button class="btn ">Sign Up</button>
      </div>
      <div class="form-group">
          <small class="text-muted ml-3" >Already have an account? <a href="{% url 'login' %}">Sign In</a></small>
      </div>
    </form>
    </div>
{% endblock %}
```

sentiment1.html

```
{% extends 'base.html' %}
{% load crispy_forms_tags %}
{% block content%}
<form method="POST" action="">
  {% csrf_token %}
  <div class="form-group">
```

```
    <h2 style="color:yellow;text-align:center;">Sentiment Analysis</h2>
    <h4 style="color:yellow;text-align:center;">Please, Write your opinion.</h4>
  <textarea class="form-control" id="exampleFormControlTextarea1" name="text1" rows="8"></textarea>
  <label><ul><li>Please Enter the text not less than 20 character.</li></ul></label>
  <br><center>
  <button class="btn" style="color: white; background-color: #fd5e14;" type="submit" id="submit"> Submit </button>
  </center>
</div>
</form>
{% for message in messages%}
  {% if message.tags == 'success' %}
    <script src="https://unpkg.com/sweetalert/dist/sweetalert.min.js"></script>
    <script>
      var m="{{message}}";
      swal("Status",m,"success")
    </script>
  {% else %}
  <script src="https://unpkg.com/sweetalert/dist/sweetalert.min.js"></script>
  <script>
    var m="{{message}}";
    swal("Perfect",m,"error")
  </script>
  {% endif %}
{% endfor %}
{% endblock %}
```

admin.py:

```
from django.contrib import admin
from authentication.models import wordSentiment
from .models import p_testd, t_testd, wordSentiment
# Register your models here.

admin.site.register(wordSentiment)
admin.site.register(p_testd)
admin.site.register(t_testd)
```

apps.py:

```
from django.apps import AppConfig
class AuthenticationConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'authentication'
```

forms.py:

```
from django.contrib.auth.models import User
from django.contrib.auth.forms import UserCreationForm
from django import forms

class UserRegisterForm(UserCreationForm):
    email=forms.EmailField()

    class Meta:
```

```
        model=User
        fields=['username','email','password1','password2']
```

models.py:

```
from tkinter import CASCADE
from django.db import models
from django.contrib.auth.models import User
from django_userforeignkey.models.fields import UserForeignKey

# Create your models here.
class wordSentiment(models.Model):
    user=models.ForeignKey(User,on_delete=models.CASCADE)
    text=models.TextField(error_messages = {
                'required':"Please Enter the Text"
                })
    date=models.DateField()
    result=models.CharField(max_length=12)
    def __str__(self):
        return self.result

class p_testd(models.Model):
    user=models.ForeignKey(User,on_delete=models.CASCADE)
    p_text=models.TextField(error_messages = {
                'required':"Please Enter the Text"
                })
    date=models.DateField()
    p_result=models.CharField(max_length=12)

    def __str__(self):
        return self.p_result

class t_testd(models.Model):
    user=models.ForeignKey(User,on_delete=models.CASCADE)
    t_text=models.TextField(error_messages = {
                'required':"Please Enter the Text"
                })
    date=models.DateField()
    t_result=models.CharField(max_length=12)
    def __str__(self):
        return self.t_result
```

urls.py:

```
from django.contrib import admin
from django.urls import path,include
from django.contrib.auth import views as auth_view
from . import views

urlpatterns = [
    path('',views.home,name='home'),
    path('register/',views.register,name='register'),
    path('profile/',views.profile,name='profile'),
    path('login/',auth_view.LoginView.as_view(template_name='login.html'),name='login'),
    path('logout',auth_view.LogoutView.as_view(template_name='logout.html'),name='logout'),
```

```
    path('sentiment1/',views.sentimentWord,name='sentimentWord'),
    path('p_test/',views.p_test,name='p_test'),
    path('languagetrans/',views.languagetrans,name='languagetrans')
]
```

view.py:

```
from datetime import datetime
from email import message
import re
from django.http import HttpResponse,HttpRequest
from django.shortcuts import redirect, render
from django.http import HttpResponse
from django.contrib.auth.forms import UserCreationForm
from django.contrib import messages
from authentication.forms import UserRegisterForm
from authentication.models import wordSentiment,p_testd,t_testd
from textblob import TextBlob
from datetime import datetime
from nltk.stem import WordNetLemmatizer
from tqdm import tqdm
import re
import joblib
import nltk
import os
import numpy
from catboost import CatBoostClassifier
from django.contrib import messages

# Create your views here.

def home(request):
    return render(request,"home.html")

def register(request):
    if request.method=="POST":
        form=UserRegisterForm(request.POST)
        if form.is_valid():
            form.save()
            username=form.cleaned_data.get('username')
            messages.success(request,f'Hi {username},your account was created sucessfully')
            return redirect('home')
    else:
        form=UserRegisterForm()
    return render(request,'register.html',{'form':form})

def profile(request):
    return render(request,'profile.html')

def sentiment_analysis(text):
    blob=TextBlob(text)
    sent_polarity=blob.sentiment.polarity
    if(sent_polarity<-0.1):
```

```
        return "Sad"
    elif(sent_polarity>0.1):
        return "Happy"
    else:
        return "Neutral"


def languagetrans(request):
    if request.method=="POST":
        text=request.POST.get('text1')
        if len(text)>=5:
            blob=TextBlob(text)
            result=blob.translate(from_lang="en",to="kn")
            p_testdbb=t_testd(user=request.user,t_text=text,t_result=result,date=datetime.today())
            p_testdbb.save()
            messages.success(request, result)
        else:
            messages.success(request,"Please Enter character more than 5")
    return render(request,'languagetrans.html')

def sentimentWord(request):
    if request.method=="POST":
        text=request.POST.get('text1')
        if len(text)>=20:
            result=sentiment_analysis(text)
            wordsentiemt=wordSentiment(user=request.user,text=text,result=result,date=datetime.today())
            wordsentiemt.save()
            messages.success(request, result)
        else:
            messages.error(request,"Please Enter character more than 20")
    return render(request,'sentiment1.html')
# Create your views here.


def p_test(request):
    if request.method=="POST":
        text=request.POST.get('text1')
        if len(text)<=50:
            messages.success(request,"Enter the text greater than 50")
            return render(request,'p_test.html')
        list_text=[text]

        def Lemmatizer():
            lemmatizer=WordNetLemmatizer()
        def clear_text1(data):
            data_length=[]
            lemmatizer=WordNetLemmatizer()
            cleaned_text=[]
            for sentence in tqdm(data):
                sentence=sentence.lower()
                sentence=re.sub('https?://[^\s<>"]+|www\.[^\s<>"]+',",sentence)
                sentence=re.sub('[^0-9a-z]',' ',sentence)
                data_length.append(len(sentence.split()))
```

```
            cleaned_text.append(sentence)
        return cleaned_text

    text_data=clear_text1(list_text)
    target=joblib.load(os.path.join('./model/target_enc1.pkl'))
    vector=joblib.load(os.path.join('./model/vectorizertext1.pkl'))
    main_model=joblib.load(os.path.join('./model/catboostmodel1.pkl'))
    text_v=vector.transform(text_data).toarray()
    a=main_model.predict(text_v)
    result=target.classes_[a[0][0]]
    p_testdbb=p_testd(user=request.user,p_text=text,p_result=result,date=datetime.today())
    p_testdbb.save()
    str1=""

    if result=='ISTJ':
        str1="ISTJ - The Inspector: Reserved and practical, they tend to be loyal, orderly, and traditional.Person: George Washington, Henry Ford"
    elif result=='ISTP':
        str1="ISTP - The Crafter: Highly independent, they enjoy new experiences that provide first-hand learning.Person: Ernest Hemingway"
    elif result=='ISFJ':
        str1="ISFJ - The Protector: Warm-hearted and dedicated, they are always ready to protect the people they care about.Person: William H. Taft"
    elif result=='ISFP':
        str1="ISFP - The Artist: Easy-going and flexible, they tend to be reserved and artistic.Person: Marie Antoineete"
    elif result=='INFJ':
        str1="INFJ - The Advocate: Creative and analytical, they are considered one of the rarest Myers-Briggs types.Person: Adolf Hitler, Plato"
    elif result=='INFP':
        str1="INFP - The Mediator: Idealistic with high values, they strive to make the world a better place. Person: Willian Shakespeare"
    elif result=="INTJ":
        str1="INTJ - The Architect: High logical, they are both very creative and analytical. Peron: Augustus Caesar"
    elif result=="INTP":
        str1="INTP - The Thinker: Quiet and introverted, they are known for having a rich inner world.Person: Albert Einstein, Abraham Lincoln"
    elif result=="ESTP":
        str1="ESTP - The Persuader: Out-going and dramatic, they enjoy spending time with others and focusing on the here-and-now.Person: Malcolm X, Steve Jobs"
    elif result=="ESTJ":
        str1="ESTJ - The Director: Assertive and rule-oriented, they have high principles and a tendency to take charge.Person: Andrew Jackson, Saddam Hussein"
    elif result=="ESFP":
        str1="ESFP - The Performer: Outgoing and spontaneous, they enjoy taking center stage.Person: Ronal Reagan"
    elif result=="ESFJ":
        str1="ESFJ - The Caregiver: Soft-hearted and outgoing, they tend to believe the best about other people.Person: William McKinley"
    elif result=="ENFP":
        str1="ENFP - The Champion: Charismatic and energetic, they enjoy situations where they can put their creativity to work.Person: Charles Dickens, Dr. Seuss"
    elif result=="ENFJ":
        str1="ENFJ - The Giver: Loyal and sensitive, they are known for being understanding and generous.Person: Martin Luther King, Jr, Nelson Mandela"
```

```
        elif result=="ENTP":
            str1="ENTP - The Debater: Highly inventive, they love being surrounded by ideas and tend to start many projects
(but may struggle to finish them).Person: Leonardo da Vinci"
        else:
            str1="ENTJ - The Commander: Outspoken and confident, they are great at making plans and organizing projects.
Person: Alexander Hamilton, Elizabeth I"
        messages.success(request, str1)
    return render(request,'p_test.html')
```

settings.py:

```
"""
Django settings for Sentiment project.

Generated by 'django-admin startproject' using Django 4.0.6.

For more information on this file, see
https://docs.djangoproject.com/en/4.0/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/4.0/ref/settings/
"""
import os
from pathlib import Path
from django.contrib import messages

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent


# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/4.0/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'django-insecure-a5g-nl&u%3bvr9rgn8z_!_+$9yr(er%@uqrsu-_vgn*@ntt7u6'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []


# Application definition

INSTALLED_APPS = [
    'jazzmin',
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'authentication.apps.AuthenticationConfig',
```

```
    'crispy_forms',
    'django_userforeignkey',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
    'django_userforeignkey.middleware.UserForeignKeyMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
]

ROOT_URLCONF = 'Sentiment.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR,"templates")],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'Sentiment.wsgi.application'


# Database
# https://docs.djangoproject.com/en/4.0/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}


# Password validation
# https://docs.djangoproject.com/en/4.0/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
```

```
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]


# Internationalization
# https://docs.djangoproject.com/en/4.0/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_TZ = True


# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/4.0/howto/static-files/

STATIC_URL = 'static/'

# Default primary key field type
# https://docs.djangoproject.com/en/4.0/ref/settings/#default-auto-field

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
STATICFILES_DIRS=[
    os.path.join(BASE_DIR,'static'),
]
CRISPY_TEMPLATE_PACK='bootstrap4'
LOGIN_REDIRECT_URL='home'
```

model_construction

```
import pandas as pd
from catboost import CatBoostClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import LabelEncoder
from sklearn.feature_extraction.text import TfidfVectorizer
from tqdm import tqdm
import nltk
from nltk.stem import WordNetLemmatizer
from nltk.corpus import stopwords
import re
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC,LinearSVC
from sklearn.ensemble import RandomForestClassifier,GradientBoostingClassifier
from xgboost import XGBClassifier
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import accuracy_score
from sklearn.naive_bayes import MultinomialNB
# from sklearn.experimental import enable_hist_gradient_boosting
from sklearn.ensemble import HistGradientBoostingClassifier
from imblearn.over_sampling import SMOTE
import plotly.express as px
import warnings
warnings.filterwarnings('ignore')

data=pd.read_csv('mbti.csv')
data.head()
data.describe(include='all')
data.describe()
train_data,test_data=train_test_split(data,test_size=0.2,random_state=42,
                                      stratify=data.type)
def clear_text(data):
    data_length=[]
    lemmatizer=WordNetLemmatizer()
    cleaned_text=[]
    for sentence in tqdm(data.posts):
```

```
        sentence=sentence.lower()

        sentence=re.sub('https?://[^\s<>"]+|www\.[^\s<>"]+',",sentence)
        sentence=re.sub('[^0-9a-z]',' ',sentence)

        data_length.append(len(sentence.split()))
        cleaned_text.append(sentence)
    return cleaned_text,data_length

train_data.posts,train_length=clear_text(train_data)
test_data.posts,test_length=clear_text(test_data)

def Lemmatizer(sentence):
    for word in sentence.split():
        if len(word)>2:
            lemmatizer.lemmatize(word)
def Lemmatizer():
    lemmatizer=WordNetLemmatizer()

vectorizer=TfidfVectorizer(max_features=5000,stop_words='english',tokenizer=Lemmatizer())
vectorizer.fit(train_data.posts)

train_post=vectorizer.transform(train_data.posts).toarray()
test_post=vectorizer.transform(test_data.posts).toarray()

target_encoder=LabelEncoder()
train_target=target_encoder.fit_transform(train_data.type)
test_target=target_encoder.fit_transform(test_data.type)

model_log=LogisticRegression(max_iter=3000,C=0.5,n_jobs=-1)
model_log.fit(train_post,train_target)

print('train classification report \n',classification_report(train_target,model_log.predict(train_post),
                                        target_names=target_encoder.inverse_transform(
                                        [i for i in range(16)])))

print('test classification report \n',
    classification_report(test_target,model_log.predict(test_post),
                    target_names=target_encoder.inverse_transform(
```

```
                    [i for i in range(16)])))

models_accuracy['logistic regression']=accuracy_score(test_target,
                                        model_log.predict(test_post))

model_linear_svc=LinearSVC(C=0.1)
model_linear_svc.fit(train_post,train_target)

print('train classification report\n',
    classification_report(train_target,model_linear_svc.predict(train_post),
                    target_names=target_encoder.inverse_transform([i for i in range(16)])))
print('test classification report\n',
    classification_report(test_target,model_linear_svc.predict(test_post),
                    target_names=target_encoder.inverse_transform(
                    [i for i in range(16)])))

models_accuracy['Linear Support Vector Classifier']=accuracy_score(
test_target,model_linear_svc.predict(test_post))

model_cat=CatBoostClassifier(loss_function='MultiClass',eval_metric='MultiClass',
                      task_type='GPU',verbose=False)
model_cat.fit(train_post,train_target)

print('Train Classification report\n',
    classification_report(train_target,model_cat.predict(train_post),
                    target_names=target_encoder.inverse_transform(
                    [i for i in range(16)])))
print('Test Classification report\n',
    classification_report(test_target,model_cat.predict(test_post),
                    target_names=target_encoder.inverse_transform(
                    [i for i in range(16)])))

models_accuracy['CatBoost Classifier']=accuracy_score(test_target,
                                        model_cat.predict(test_post))

def clear_text1(data):
   data_length=[]
   lemmatizer=WordNetLemmatizer()
   cleaned_text=[]
```

```
    for sentence in tqdm(data):
        sentence=sentence.lower()

        sentence=re.sub('https?://[^\s<>"]+|www\.[^\s<>"]+',",sentence)
        sentence=re.sub('[^0-9a-z]',' ',sentence)

        data_length.append(len(sentence.split()))
        cleaned_text.append(sentence)
    return cleaned_text
```

12. Conclusion

Hence, Sentiment and Persoanlity Test web application is application which is used to predict the Sentiment and Personality of the user and also help to translate the English language to Kannada. It's one of the fine project which is user-friendly.

13. Limitation

- ✓ **Django is Monolithic**– Django framework has a certain way to define and perform tasks. It is a logical file structure and easy to learn. But, that also makes it mandatory that you can't use your own file structure. It is because the framework has a way, popularly known as **"The Django way"** of doing things. If you don't follow those rules, you may not be able to deploy anything using Django.
- ✓ **Heavy Application** – It's is very light weight application but it's consume a lot of space as it contain lot of data.
- ✓ **Only one language** – It's only in English language. So, other language people feel difficult to operate.
- ✓ **Not Access with out internet –** As it's new web application so it can't be used without internet.

14. Future Enhancement

Sentiment analysis is simply **the process of categorizing the sentiments underlying a text**. It is such a simple task that it can also be done manually; simply read each piece of feedback and determine whether it is positive or negative.

Personality traits are defined as relatively constant patterns of thoughts, feelings, and behavior that have been linked to a variety of significant life outcomes and decisions.

This system is enhance in future with the help of data as this project is based on data. So, when the user enter the data, it’s help to relearn from that data to make the algorithm stornger and stronger and make enchancement in application.

15. Bibliography

(a) Medium: https://medium.com/@bian0628/data-science-final-project-myers-briggs-prediction-ecfa203cef8

(b) 16personalities: https://www.16personalities.com/free-personality-test

(c) Monkey Learn: https://monkeylearn.com/sentiment-analysis/

(d) Computer Learning Arena for All: https://gichaidon.co.ke/

(e) Google Clour: https://cloud.google.com/translate/docs/languages

(f) SQLite: https://www.sqlite.org/index.html

(g) w3schools : https://www.w3schools.com/django/

(h) CatBoost: https://catboost.ai/docs//