**Smart Banking Chatbot**

**INTRODUCTION**

At a very basic level chatbot can be considered to be an agent which can interact with a user like a human (voice or text) and understand what the user wants from it and provide a specific set of services to the user according to the user’s demands. Chatbot also acts as an information-gathering tool using which the organization can know their customer base more clearly and understand their needs. Chatbots are used in a wide range of domains right from health care services to customer relationship management services. A chatbot broadly comprises three subsystems which are Natural Language Understanding(NLU) which understands what the user is trying to communicate or what is the intent of the provided sentence, Dialog Management which maintains the state of communication by remembering what key pieces of information have been provided by the user and how to drive the conversation ahead and Natural Language Generation(NLG) which handles the part where the chatbot figures out what the bot should respond to back to the user. RASA is an open-source Natural Language Processing (NLP) tool that is used to build virtual assistants and contextual chatbots. RASA provides two main modules Rasa Open Source and Rasa Action Server. Rasa Open Source has Rasa NLU to recognize the intent of communication and Rasa core to decide what to do next and how to proceed with the conversation. Rasa Action Server allows the hosting of Python scripts to perform certain custom tasks like processing or modifying databases in the backend. This paper presents a chatbot deployed on a banking website that offers certain services like checking account balance, making a transaction, and many more to users right in the chat window

**1.1 Objective of the project:**

The purpose of the project is to make any domain-specific website, in our case banking, more usable by integrating a chatbot that serves as an interface for customer inquiries about services. This reduces customer interaction time with websites, thereby valuing their time and improving their overall experience. As part of this project, we explored and attempted to create an intelligent chatbot that could extract relevant information, recognize various intents and execute the pre-mapped actions. To create a contextual assistant for the above purpose, we used the RASA framework. To train the model, we constructed a custom dataset that includes multiple intents and entities. Additionally, we provide some python scripts (RASA actions) that will be executed when some intents are detected. Our solution consisted of creating a pipeline having a chatbot and several actions triggered by the chatbot. These actions will connect with the database and then provide the required data or make changes according to the user’s query and display the feedback back to the user via the chat widget.

**2. LITERATURE SURVEY:**

# An Intelligent Chatbot System Based on Entity Extraction Using RASA NLU and Neural Network

Intelligent chatbot systems are popular issues in the application fields of robot system and natural language processing. As the development of natural language processing and neural network algorithms, the application of artificial intelligence is increasing in Chatbot systems, which are typically used in dialog systems for various practical purposes including customer service or information acquisition. This paper designs the functional framework and introduces the principle of RASA NLU for the Chatbot system, then it integrates RASA NLU and neural network (NN) methods and implements the system based on entity extraction after intent recognition. With the experimental comparison and validation, our developed system can realize automatic learning and answering the collected questions about finance. The system analysis of two methods also validate that RASA NLU outperforms NN in accuracy for a single experiment, but NN has better integrity to classify entities from segmented words.

# Conversation to Automation in Banking Through Chatbot Using Artificial Machine Intelligence Language

Artificial Machine Intelligence is a very complicated topic. It involves creating machines that are capable of simulating knowledge. This paper examines some of the latest AI patterns and activities and then provides alternative theory of change in some of the popular and widely accepted postulates of today. Based on basic A.I. (Artificial Intelligence) structuring and working for this, System-Chatbots are made (or chatter bots). The paper shows that A.I is ever improving. As of now there isn't enough information on A.I. however this paper provides a new concept which addresses machine intelligence and sheds light on the potential of intelligent systems. The rise of chatbots in the finance sector is the latest disruptive force that has changed the way customers interact. In the banking industry, the introduction of Artificial Intelligence has driven chatbots and changed the face of the interaction between bank and customers. The banking sector plays an important role in development into any country. It also explores the existing usability of chatbot to assess whether it can fulfill customers ever-changing needs.

# Chatbots and Virtual Assistant in Indian Banks

This paper discusses the adoption of chatbots and virtual Assistants by different category of banks (private sector banks and public sector banks) in India. The research paper presents a brief introduction of banking industry in India, history, characteristics, and architecture of chatbots and virtual assistants. The research paper also included basic features, way to connect, services offered, accuracy, technology providers of chatbots and virtual assistants implemented by Indian banks. Research paper concluded that Indian banks are aggressively investing in chatbots and virtual assistant technology but features are limited. Most of the questions answered by chatbots/ virtual assistant are routine for which data is already available on websites of the banks. Secondly, awareness of chatbots/ virtual assistants is very low among the customers and employees of the banks. It necessitates enhancement of existing capabilities of chatbots/ virtual assistant deployed by Indian banks and awareness with respect to usefulness among employees and customers.

# Smart Banking Chatbot

The purpose of the project is to make any domain-specific website, in our case banking, more usable by integrating a chatbot that serves as an interface for customer inquiries about services. This reduces customer interaction time with websites, thereby valuing their time and improving their overall experience. As part of this project, we explored and attempted to create an intelligent chatbot that could extract relevant information, recognize various intents and execute the pre-mapped actions. To create a contextual assistant for the above purpose, we used the RASA framework. To train the model, we constructed a custom dataset that includes multiple intents and entities. Additionally, we provide some python scripts (RASA actions) that will be executed when some intents are detected. Our solution consisted of creating a pipeline having a chatbot and several actions triggered by the chatbot. These actions will connect with the database and then provide the required data or make changes according to the user’s query and display the feedback back to the user via the chat widget

**3. SYSTEM ANALYSIS**

**3.1 Existing System**

In existing system, Technology use in banking sector continues to be high, pushed by the ever-increasing need for more banking services. In the competitive banking area, the customer expects quick responses increasing the load for banks to catch up with the requirements. This project aims to develop a website with a chatbot that is not only capable of having conversations with the customer but also has the ability to listen to commands and make changes to the database.

**Disadvantages:**

1. Less Accuracy
2. More time taking process

**3.2 PROPOSED SYSTEM**

In this project we have employed Artificial Intelligence technique to process text data and then employ machine learning algorithms to predict answers for user questions. Chatbot will take user queries and then employ ML algorithm to predict answer and then this answer will be given as output to the user.

To get accurate answer we have evaluate performance of multiple algorithms like Random Forest, KNN and SVM. In all algorithms Random Forest is giving best accuracy.

**Advantages:**

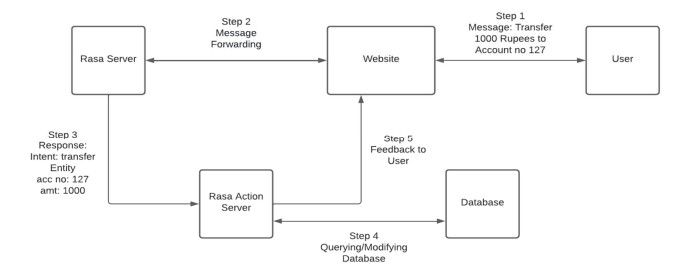
1. High Accuracy
2. Takes less time

**Modules Information:**

To implement this project we have designed following modules

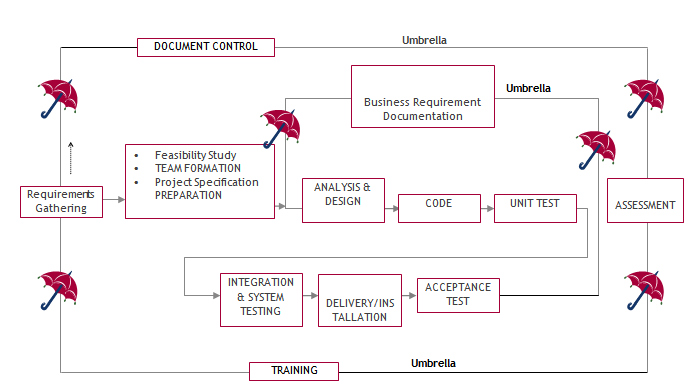
1. Admin Login: using this module admin can login to system using username and password as ‘admin’ and then can train ML algorithms. Admin can view list of sign up users and can view all interaction between all users and Chatbot.
2. New user sign up: using this module user can sign up with the application
3. User Login: registered user can login to system
4. Interact with Chatbot: using this module user can communicate with Chatbot.

**SYSTEM ARCHITECTURE:**

****

**3.3. PROCESS MODEL USED WITH JUSTIFICATION**

**SDLC (Umbrella Model):**

****

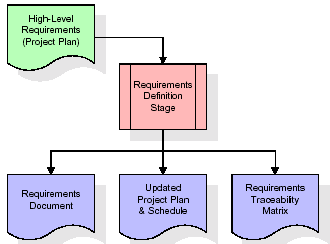
SDLC is nothing but Software Development Life Cycle. It is a standard which is used by software industry to develop good software.

**Stages in SDLC:**

* Requirement Gathering
* Analysis
* Designing
* Coding
* Testing
* Maintenance

**Requirements Gathering stage:**

The requirements gathering process takes as its input the goals identified in the high-level requirements section of the project plan. Each goal will be refined into a set of one or more requirements. These requirements define the major functions of the intended application, define operational data areas and reference data areas, and define the initial data entities. Major functions include critical processes to be managed, as well as mission critical inputs, outputs and reports. A user class hierarchy is developed and associated with these major functions, data areas, and data entities. Each of these definitions is termed a Requirement. Requirements are identified by unique requirement identifiers and, at minimum, contain a requirement title and textual description.



These requirements are fully described in the primary deliverables for this stage: the Requirements Document and the Requirements Traceability Matrix (RTM). The requirements document contains complete descriptions of each requirement, including diagrams and references to external documents as necessary. Note that detailed listings of database tables and fields are *not* included in the requirements document.

The title of each requirement is also placed into the first version of the RTM, along with the title of each goal from the project plan. The purpose of the RTM is to show that the product components developed during each stage of the software development lifecycle are formally connected to the components developed in prior stages.

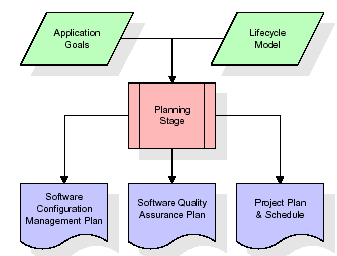
In the requirements stage, the RTM consists of a list of high-level requirements, or goals, by title, with a listing of associated requirements for each goal, listed by requirement title. In this hierarchical listing, the RTM shows that each requirement developed during this stage is formally linked to a specific product goal. In this format, each requirement can be traced to a specific product goal, hence the term requirements traceability.

The outputs of the requirements definition stage include the requirements document, the RTM, and an updated project plan.

* Feasibility study is all about identification of problems in a project.
* No. of staff required to handle a project is represented as Team Formation, in this case only modules are individual tasks will be assigned to employees who are working for that project.
* Project Specifications are all about representing of various possible inputs submitting to the server and corresponding outputs along with reports maintained by administrator.

**Analysis Stage:**

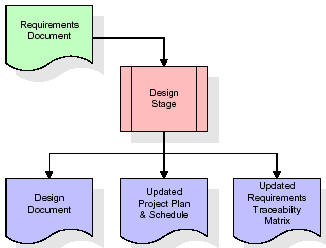
The planning stage establishes a bird's eye view of the intended software product, and uses this to establish the basic project structure, evaluate feasibility and risks associated with the project, and describe appropriate management and technical approaches.



The most critical section of the project plan is a listing of high-level product requirements, also referred to as goals. All of the software product requirements to be developed during the requirements definition stage flow from one or more of these goals. The minimum information for each goal consists of a title and textual description, although additional information and references to external documents may be included. The outputs of the project planning stage are the configuration management plan, the quality assurance plan, and the project plan and schedule, with a detailed listing of scheduled activities for the upcoming Requirements stage, and high level estimates of effort for the out stages.

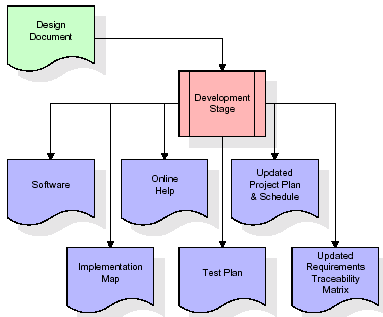
**Designing Stage:**

The design stage takes as its initial input the requirements identified in the approved requirements document. For each requirement, a set of one or more design elements will be produced as a result of interviews, workshops, and/or prototype efforts. Design elements describe the desired software features in detail, and generally include functional hierarchy diagrams, screen layout diagrams, tables of business rules, business process diagrams, pseudo code, and a complete entity-relationship diagram with a full data dictionary. These design elements are intended to describe the software in sufficient detail that skilled programmers may develop the software with minimal additional input.

  
When the design document is finalized and accepted, the RTM is updated to show that each design element is formally associated with a specific requirement. The outputs of the design stage are the design document, an updated RTM, and an updated project plan.

**Development (Coding) Stage:**

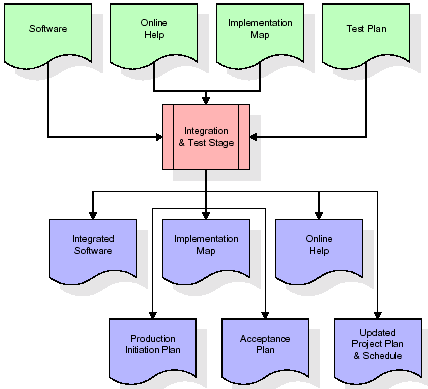
The development stage takes as its primary input the design elements described in the approved design document. For each design element, a set of one or more software artifacts will be produced. Software artifacts include but are not limited to menus, dialogs, and data management forms, data reporting formats, and specialized procedures and functions. Appropriate test cases will be developed for each set of functionally related software artifacts, and an online help system will be developed to guide users in their interactions with the software.



The RTM will be updated to show that each developed artifact is linked to a specific design element, and that each developed artifact has one or more corresponding test case items. At this point, the RTM is in its final configuration. The outputs of the development stage include a fully functional set of software that satisfies the requirements and design elements previously documented, an online help system that describes the operation of the software, an implementation map that identifies the primary code entry points for all major system functions, a test plan that describes the test cases to be used to validate the correctness and completeness of the software, an updated RTM, and an updated project plan.

**Integration & Test Stage:**

During the integration and test stage, the software artifacts, online help, and test data are migrated from the development environment to a separate test environment. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite confirms a robust and complete migration capability. During this stage, reference data is finalized for production use and production users are identified and linked to their appropriate roles. The final reference data (or links to reference data source files) and production user list are compiled into the Production Initiation Plan.

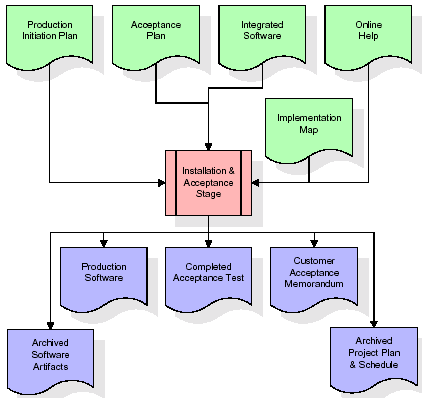


The outputs of the integration and test stage include an integrated set of software, an online help system, an implementation map, a production initiation plan that describes reference data and production users, an acceptance plan which contains the final suite of test cases, and an updated project plan.

* **Installation & Acceptance Test:**

During the installation and acceptance stage, the software artifacts, online help, and initial production data are loaded onto the production server. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite is a prerequisite to acceptance of the software by the customer.

After customer personnel have verified that the initial production data load is correct and the test suite has been executed with satisfactory results, the customer formally accepts the delivery of the software.



The primary outputs of the installation and acceptance stage include a production application, a completed acceptance test suite, and a memorandum of customer acceptance of the software. Finally, the PDR enters the last of the actual labor data into the project schedule and locks the project as a permanent project record. At this point the PDR "locks" the project by archiving all software items, the implementation map, the source code, and the documentation for future reference.

**Maintenance:**

Outer rectangle represents maintenance of a project, Maintenance team will start with requirement study, understanding of documentation later employees will be assigned work and they will undergo training on that particular assigned category. For this life cycle there is no end, it will be continued so on like an umbrella (no ending point to umbrella sticks).

**3.4. Software Requirement Specification**

**3.4.1. Overall Description**

A Software Requirements Specification (SRS) – a requirements specification for a software system is a complete description of the behaviour of a system to be developed. It includes a set of use cases that describe all the interactions the users will have with the software. In addition to use cases, the SRS also contains non-functional requirements. Non-functional requirements are requirements which impose constraints on the design or implementation (such as performance engineering requirements, quality standards, or design constraints).

System requirements specification: A structured collection of information that embodies the requirements of a system. A business analyst, sometimes titled system analyst, is responsible for analysing the business needs of their clients and stakeholders to help identify business problems and propose solutions. Within the systems development lifecycle domain, the BA typically performs a liaison function between the business side of an enterprise and the information technology department or external service providers. Projects are subject to three sorts of requirements:

* Business requirements describe in business terms what must be delivered or accomplished to provide value.
* Product requirements describe properties of a system or product (which could be one of several ways to accomplish a set of business requirements.)
* Process requirements describe activities performed by the developing organization. For instance, process requirements could specify .Preliminary investigation examine project feasibility, the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All system is feasible if they are unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:
* **ECONOMIC FEASIBILITY**

A system can be developed technically and that will be used if installed must still be a good investment for the organization. In the economical feasibility, the development cost in creating the system is evaluated against the ultimate benefit derived from the new systems. Financial benefits must equal or exceed the costs. The system is economically feasible. It does not require any addition hardware or software. Since the interface for this system is developed using the existing resources and technologies available at NIC, There is nominal expenditure and economical feasibility for certain.

* **Operational Feasibility**

Proposed projects are beneficial only if they can be turned out into information system. That will meet the organization’s operating requirements. Operational feasibility aspects of the project are to be taken as an important part of the project implementation. This system is targeted to be in accordance with the above-mentioned issues. Beforehand, the management issues and user requirements have been taken into consideration. So there is no question of resistance from the users that can undermine the possible application benefits. The well-planned design would ensure the optimal utilization of the computer resources and would help in the improvement of performance status.

* **TECHNICAL FEASIBILITY**

Earlier no system existed to cater to the needs of ‘Secure Infrastructure Implementation System’. The current system developed is technically feasible. It is a web-based user interface for audit workflow at NIC-CSD. Thus, it provides an easy access to. the users. The database’s purpose is to create, establish and maintain a workflow among various entities in order to facilitate all concerned users in their various capacities or roles. Permission to the users would be granted based on the roles specified. Therefore, it provides the technical guarantee of accuracy, reliability and security.

**3.4.2. External Interface Requirements**

**User Interface**

The user interface of this system is a user friendly python Graphical User Interface.

**Hardware Interfaces**

The interaction between the user and the console is achieved through python capabilities.

**Software Interfaces**

The required software is python.

**SYSTEM REQUIREMENT:**

**HARDWARE REQUIREMENTS:**

# Processor - Intel i3(min)

* Speed - 1.1 GHz
* RAM - 4GB(min)
* Hard Disk - 500 GB

**SOFTWARE REQUIREMENTS:**

* Operating System - Windows10(min)
* Programming Language - Python(3.7.0) with Django

**4. SYSTEM DESIGN**

**UML Diagram:**

The Unified Modelling Language allows the software engineer to express an analysis model using the modelling notation that is governed by a set of syntactic semantic and pragmatic rules.

A UML system is represented using five different views that describe the system from distinctly different perspective. Each view is defined by a set of diagram, which is as follows.

* + **User Model View**
    1. This view represents the system from the users perspective.
    2. The analysis representation describes a usage scenario from the end-users perspective.
  + **Structural Model view**
    1. In this model the data and functionality are arrived from inside the system.
    2. This model view models the static structures.
* **Behavioural Model View**

It represents the dynamic of behavioural as parts of the system, depicting the interactions of collection between various structural elements described in the user model and structural model view.

* **Implementation Model View**

In this the structural and behavioural as parts of the system are represented as they are to be built.

* **Environmental Model View**

In this the structural and behavioural aspects of the environment in which the system is to be implemented are represented.

**Class Diagram:**

The class diagram is the main building block of object oriented modeling. It is used both for general conceptual modeling of the systematic of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling. The classes in a class diagram represent both the main objects, interactions in the application and the classes to be programmed. In the diagram, classes are represented with boxes which contain three parts:

* The upper part holds the name of the class
* The middle part contains the attributes of the class
* The bottom part gives the methods or operations the class can take or undertake.



**Use case Diagram:**

A **use case diagram** at its simplest is a representation of a user's interaction with the system and depicting the specifications of a use case. A use case diagram can portray the different types of users of a system and the various ways that they interact with the system. This type of diagram is typically used in conjunction with the textual use case and will often be accompanied by other types of diagrams as well.

****

**Sequence diagram:**

A sequence diagram is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.



**Collaboration diagram:**

A collaboration diagram describes interactions among objects in terms of sequenced messages. Collaboration diagrams represent a combination of information taken from class, sequence, and use case diagrams describing both the static structure and dynamic behaviour of a system.



**Component Diagram:**

In the Unified Modelling Language, a component diagram depicts how components are wired together to form larger components and or software systems. They are used to illustrate the structure of arbitrarily complex systems.

Components are wired together by using an assembly connector to connect the required interface of one component with the provided interface of another component. This illustrates the service consumer - service provider relationship between the two components.

****

**Deployment Diagram:**

A **deployment diagram** in the Unified Modeling Language models the *physical* deployment of artifacts on nodes. To describe a web site, for example, a deployment diagram would show what hardware components ("nodes") exist (e.g., a web server, an application server, and a database server), what software components ("artifacts") run on each node (e.g., web application, database), and how the different pieces are connected (e.g. JDBC, REST, RMI).

The nodes appear as boxes, and the artifacts allocated to each node appear as rectangles within the boxes. Nodes may have sub nodes, which appear as nested boxes. A single node in a deployment diagram may conceptually represent multiple physical nodes, such as a cluster of database servers.

****

**Activity Diagram:**

Activity diagram is another important diagram in UML to describe dynamic aspects of the system. It is basically a flow chart to represent the flow form one activity to another

activity. The activity can be described as an operation of the system. So the control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent

**Admin Login**

**New User Sign UP**

**User Login**

datasetSplit

datasetSplit

datasetSplit

**Train ML**

**Interact with Chatbot**

**Data Flow Diagram:**

Data flow diagrams illustrate how data is processed by a system in terms of inputs and outputs. Data flow diagrams can be used to provide a clear representation of any business function. The technique starts with an overall picture of the business and continues by analyzing each of the functional areas of interest. This analysis can be carried out in precisely the level of detail required. The technique exploits a method called top-down expansion to conduct the analysis in a targeted way.

As the name suggests, Data Flow Diagram (DFD) is an illustration that explicates the passage of information in a process. A DFD can be easily drawn using simple symbols. Additionally, complicated processes can be easily automated by creating DFDs using easy-to-use, free downloadable diagramming tools. A DFD is a model for constructing and analyzing information processes. DFD illustrates the flow of information in a process depending upon the inputs and outputs. A DFD can also be referred to as a Process Model. A DFD demonstrates business or technical process with the support of the outside data saved, plus the data flowing from the process to another and the end results.

**User**

1. Admin Login 2.Admin Login successfully

3. New User Sign UP 4. New User Sign UP successfully

5. User Login 6. User Login Successfully

7. Train ML 8. Train ML Successfully

9. Interact with Chatbot 10. Successfully Interact with Chatbot

**5. IMPLEMETATION**

**5.1 PYTHON**

\* One of the most popular languages is Python. Guido van Rossum released this language in 1991. Python is available on the Mac, Windows, and Raspberry Pi operating systems. The syntax of Python is simple and identical to that of English. When compared to Python, it was seen that the other language requires a few extra lines.

\*It is an interpreter-based language because code may be run line by line after it has been written. This implies that rapid prototyping is possible across all platforms. Python is a big language with a free, binary-distributed interpreter standard library.

\* It is inferior to maintenance that is conducted and is straightforward to learn. It is an object-oriented, interpreted programming language. It supports several different programming paradigms in addition to object-oriented programming, including functional and procedural programming.

\* It supports several different programming paradigms in addition to object-oriented programming, including practical and procedural programming. Python is mighty while maintaining a relatively straightforward syntax. Classes, highly dynamic data types, modules, and exceptions are covered. Python can also be utilised by programmes that require programmable interfaces as an external language.

Here are some key features and characteristics of Python:

* Readability: Python emphasizes code readability with its clean and intuitive syntax. It uses indentation and whitespace to structure code blocks, making it easy to understand and maintain.
* Easy to Learn: Python's simplicity and readability make it an excellent choice for beginners. Its straightforward syntax and extensive documentation make it accessible for newcomers to programming.
* Interpreted Language: Python is an interpreted language, meaning that it doesn't need to be compiled before running. The Python interpreter reads and executes the code directly, making the development process faster and more interactive.
* Cross-platform Compatibility: Python is available for major operating systems like Windows, macOS, and Linux. This cross-platform compatibility allows developers to write code once and run it on different platforms without modifications.
* Large Standard Library: Python comes with a vast standard library that provides ready-to-use modules and functions for various tasks. It covers areas such as file I/O, networking, regular expressions, databases, and more, saving developers time and effort.
* Extensible and Modular: Python supports modular programming, enabling developers to organize code into reusable modules and packages. Additionally, Python allows integrating modules written in other languages, such as C or C++, providing flexibility and performance optimizations.
* Wide Range of Libraries and Frameworks: Python has a vibrant ecosystem with numerous third-party libraries and frameworks. These libraries, such as NumPy, pandas, TensorFlow, and Django, extend Python's capabilities for specific domains, making it a powerful tool for diverse applications.
* Object-Oriented: Python supports object-oriented programming (OOP) principles, allowing developers to create and work with classes and objects. OOP provides a structured approach to code organization, promoting code reuse and modularity.
* Dynamic Typing: Python is dynamically typed, meaning variable types are determined at runtime. Developers do not need to declare variable types explicitly, which enhances flexibility and simplifies code writing.

**5.2 Installation**

To install Python on your computer, follow these basic steps:

* Step 1: Visit the Python website Go to the official Python website at <https://www.python.org/>.
* Step 2: Select the operating system Choose the appropriate installer for your operating system. Python supports Windows, macOS, and various Linux distributions. Make sure to select the correct version that matches your operating system.
* Step 3: Check which version of Python is installed; if the 3.7.0 version is not there, uninstall it through the control panel and
* Step 4: Install Python 3.7.0 using Cmd.
* Step 5: Install the all libraries that required to run the project
* Step 6: Run

**5.3 Python Features:**

1. **Easy:** Because Python is a more accessible and straightforward language, Python programming is easier to learn.
2. **Interpreted language:** Python is an interpreted language, therefore it can be used to examine the code line by line and provide results.
3. **Open Source:** Python is a free online programming language since it is open-source.
4. **Portable:** Python is portable because the same code may be used on several computer standard
5. **libraries:** Python offers a sizable library that we may utilize to create applications quickly.
6. **GUI:** It stands for GUI (Graphical User Interface)
7. **Dynamical typed:** Python is a dynamically typed language, therefore the type of the value will be determined at runtime.

**5.4 Python GUI (Tkinter)**

* Python provides a wide range of options for GUI development (Graphical User Interfaces).
* Tkinter, the most widely used GUI technique, is used for all of them.
* The Tk GUI toolkit offered by Python is used with the conventional Python interface.
* Tkinter is the easiest and quickest way to write Python GUI programs.
* Using Tkinter, creating a GUI is simple.
* A part of Python's built-in library is Tkinter. The GUI programs were created.
* Python and Tkinter together give a straightforward and quick way. The Tk GUI toolkit's object-oriented user interface is called Tkinter.

Making a GUI application is easy using Tkinter. Following are the steps:

1) Install the Tkinter module in place.

2) The GUI applicatioMakeske the primary window

3) Include one or more of the widgets mentioned above in the GUI application.

4) Set up the main event loop such that it reacts to each user-initiated event.

Although Tkinter is the only GUI framework included in the Python standard library, Python includes a GUI framework. The default library for Python is called Tkinter. Tk is a scripting language often used in designing, testing, and developing GUIs. Tk is a free, open-source widget toolkit that may be used to build GUI applications in a wide range of computer languages.

**5.5 Python IDLE**

* Python IDLE offers a full-fledged file editor, which gives you the ability to write and execute Python programs from within this program. The built-in file editor also includes several features, like code completion and automatic indentation, that will speed up your coding workflow.
* Guido Van Rossum named Python after the British comedy group Monty Python while the name IDLE was chosen to pay tribute to Eric Idle, who was one of the Monty Python's founding members. IDLE comes bundled with the default implementation of the Python language since the 01.5. 2b1 release
* IDLE is used to execute statements similar to Python Shell. IDLE is used to create, modify, and execute Python code. IDLE provides a fully-featured text editor to write Python scripts and provides features like syntax highlighting, auto-completion, and smart indent.
* IDLE has two modes: interactive and script. We wrote our first program, “Hello, World!” in interactive mode. Interactive mode immediately returns the results of commands you enter into the shell. In script mode, you will write a script and then run it.
* The IDE Python IDLE is a good place to start as it helps you become familiar with the way Python works and understand its syntax. This IDE is good to start programming in Python due to its great debugger, but once you are fluent and start developing projects it is necessary to jump to another, more complete IDE.
* Python IDLE (Integrated Development and Learning Environment) is an interactive development environment included with the Python programming language. It provides a convenient way to write, execute, and debug Python code.

When you install Python, IDLE is typically installed along with it. To open IDLE, you can follow these steps:

* Open the command prompt (Windows) or terminal (macOS/Linux).
* Type "idle" and press Enter. Alternatively, you can specify the version with "idle3" or "idle2" for Python 3 or Python 2, respectively.
* Once IDLE is launched, you will see the Python shell, which is an interactive environment where you can type and execute Python code directly.

Here are some features and functionalities provided by Python IDLE:

* Editor: IDLE includes a text editor where you can write your Python code. It offers syntax highlighting, automatic indentation, and code completion to enhance your coding experience.
* Interactive Shell: The Python shell in IDLE allows you to execute Python code interactively. You can type commands, statements, or function calls directly in the shell, and Python will execute them immediately.
* Debugging: IDLE provides basic debugging capabilities to help you find and fix errors in your code. You can set breakpoints, step through code, inspect variables, and track the program's execution.
* Python Help: IDLE provides access to the Python documentation and built-in help. You can access the help menu to find information about Python modules, functions, classes, and more.
* Script Execution: In addition to the interactive shell, IDLE allows you to run Python scripts stored in files. You can write your code in the editor and execute it as a script to see the output or interact with the program.
* Customization: IDLE can be customized to suit your preferences. You can modify settings related to syntax highlighting, indentation, fonts, and more.
* Python IDLE serves as a beginner-friendly development environment and learning tool. It is suitable for writing small scripts, testing code snippets, experimenting with Python features, and learning the language's basics. However, for more advanced development projects, you may consider using other code editors or integrated development environments (IDEs) that provide additional features and better project management capabilities.

**5.6 Libraries**

In Python, libraries (also referred to as modules or packages) are collections of pre-written code that provide additional functionality and tools to extend the capabilities of the Python language. Libraries contain reusable code that developers can leverage to perform specific tasks without having to write everything from scratch.

Python libraries are designed to solve common problems, such as handling data, performing mathematical operations, interacting with databases, working with files, implementing networking protocols, creating graphical user interfaces (GUIs), and much more. They provide ready-to-use functions, classes, and methods that simplify complex operations and save development time.

**Libraries in Python offer various advantages:**

* Code Reusability:
* Efficiency:
* Collaboration
* Domain-Specific Functionality
* To use a Python library, you need to install it first.

There are some libraries following:

* **Pandas:**

Pandas are a Python computer language library for data analysis and manipulation. It offers a specific operation and data format for handling time series and numerical tables. It differs significantly from the release3-clause of the BSD license. It is a well-liked open-source of opinion that is utilized in machine learning and data analysis.

Pandas are a Python package providing fast, flexible, and expressive data structures designed to make working with “relational” or “labeled” data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real-world data analysis in Python. Pandas are a Python library used for working with data sets.

* It has functions for analysing, cleaning, exploring, and manipulating data.
* The name "Pandas" has a reference to both "Panel Data", and "Python Data Analysis" and was created by Wes McKinney in 2008.
* Pandas allow us to analyse big data and make conclusions based on statistical theories.
* Pandas can clean messy data sets, and make them readable and relevant.

Relevant data is very important in data science. Pandas are a Python library for data analysis. Started by Wes McKinney in 2008 out of a need for a powerful and flexible quantitative analysis tool, pandas have grown into one of the most popular Python libraries. It has an extremely active community of contributors. The name is derived from the term "panel data", an econometrics term for data sets that include observations over multiple time periods for the same individuals. Its name is a play on the phrase "Python data analysis" itself.

* **NumPy:**

The NumPy Python library for multi-dimensional, big-scale matrices adds a huge number of high-level mathematical functions. It is possible to modify NumPy by utilizing a Python library. Along with line, algebra, and the Fourier transform operations, it also contains several matrices-related functions.

NumPy can be used to perform a wide variety of mathematical operations on arrays. It adds powerful data structures to Python that guarantee efficient calculations with arrays and matrices and it supplies an enormous library of high-level mathematical functions that operate on these arrays and matrices.

* NumPy is a Python library used for working with arrays.
* It also has functions for working in domain of linear algebra, Fourier transform, and matrices.
* NumPy was created in 2005 by Travis Oliphant. It is an open source project and you can use it freely.
* NumPy stands for Numerical Python.
* In Python we have lists that serve the purpose of arrays, but they are slow to process.
* NumPy aims to provide an array object that is up to 50x faster than traditional Python lists.
* The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy.
* Arrays are very frequently used in data science, where speed and resources are very important.
* **Matplotlib:**

It is a multi-platform, array-based data visualization framework built to interact with the whole SciPy stack. MATLAB is proposed as an open-source alternative. Matplotlib is a Python extension and a cross-platform toolkit for graphical plotting and visualization.

Matplotlib is a popular Python library for creating static, animated, and interactive visualizations. It provides a flexible and comprehensive set of tools for generating plots, charts, histograms, scatter plots, and more. Matplotlib is widely used in various fields, including data analysis, scientific research, and data visualization.

Here are some key features and functionalities of the Matplotlib library:

* Plotting Functions
* Customization Options
* Multiple Interfaces
* Integration with NumPy and pandas
* Subplots and Figures:
* Saving and Exporting
* **Scikit-learn:**

The most stable and practical machine learning library for Python is scikit-learn. Regression, dimensionality reduction, classification, and clustering are just a few of the helpful tools it provides through the Python interface for statistical modeling and machine learning. It is an essential part of the Python machine learning toolbox used by JP Morgan. It is frequently used in various machine learning applications, including classification and predictive analysis.

Scikit-learn (also referred to as sklearn) is a widely used open-source machine learning library for Python. It provides a comprehensive set of tools and algorithms for various machine learning tasks, including classification, regression, clustering, dimensionality reduction, model selection, and pre-processing.

Here are some key features and functionalities of the Scikit-learn library:

* Easy-to-Use Interface:
* Broad Range of Algorithms:
* Data Pre-processing and Feature Engineering:
* Model Evaluation and Validation:
* Integration with NumPy and pandas:
* Robust Documentation and Community Support:
* **Keras:**

\* Google's Keras is a cutting-edge deep learning API for creating neural networks. It is created in Python and is designed to simplify the development of neural networks. Additionally, it enables the use of various neural networks for computation. Deep learning models are developed and tested using the free and open-source Python software known as Keras.

Keras is a high-level deep learning library for Python. It is designed to provide a user-friendly and intuitive interface for building and training deep learning models. Keras acts as a front-end API, allowing developers to define and configure neural networks while leveraging the computational backend engines, such as Tensor Flow or Theano.

Here are some key features and functionalities of the Keras library:

* User-Friendly API
* Multi-backend Support
* Wide Range of Neural Network Architectures
* Pre-trained Models and Transfer Learning:
* Easy Model Training and Evaluation:
* GPU Support:
* **h5py:**

\* The h5py Python module offers an interface for the binary HDF5 data format. Thanks to p5py, the top can quickly halt the vast amount of numerical data and alter it using the NumPy library. It employs common syntax for Python, NumPy, and dictionary arrays.

h5py is a Python library that provides a simple and efficient interface for working with datasets and files in the Hierarchical Data Format 5 (HDF5) format. HDF5 is a versatile data format commonly used for storing and managing large volumes of numerical data.

Here are some key features and functionalities of the h5py library:

* + HDF5 File Access
  + Dataset Handling:
  + Group Organization:
  + Attributes:
  + Compatibility with NumPy
  + Performance
* **Tensor flow**

TensorFlow is a Python library for fast numerical computing created and released by Google. It is a foundation library that can be used to create Deep Learning models directly or by using wrapper libraries that simplify the process built on top of TensorFlow. TensorFlow is an end-to-end open source platform for machine learning. TensorFlow is a rich system for managing all aspects of a machine learning system; however, this class focuses on using a particular TensorFlow API to develop and train machine learning models.

TensorFlow is a popular open-source library for machine learning and deep learning. It provides a comprehensive set of tools, APIs, and computational resources for building and training various types of machine learning models, especially neural networks.

Here are some key features and functionalities of TensorFlow:

* Neural Network Framework:
* Computational Graphs
* Automatic Differentiation
* GPU and TPU Support
* Distributed Computing
* Deployment Capabilities
* **Tkinter**

Tkinter is an acronym for "Tk interface". Tk was developed as a GUI extension for the Tcl scripting language by John Ousterhout. The first release was in 1991. Tkinter is the de facto way in Python to create Graphical User interfaces (GUIs) and is included in all standard Python Distributions. In fact, it's the only framework built into the Python standard library.

Tkinter is a standard Python library used for creating graphical user interfaces (GUIs). It provides a set of modules and classes that allow you to develop interactive and visually appealing desktop applications.

Here are some key features and functionalities of Tkinter:

* Cross-Platform Compatibility
* Simple and Easy-to-Use
* Widgets and Layout Management
* Event-Driven Programming
* Customization and Styling
* Integration with Other Libraries
* **NLTK**

NLTK is a toolkit build for working with NLP in Python. It provides us various text processing libraries with a lot of test datasets. A variety of tasks can be performed using NLTK such as tokenizing, parse tree visualization, etc NLTK (Natural Language Toolkit) is the go-to API for NLP (Natural Language Processing) with Python. It is a really powerful tool to pre-process text data for further analysis like with ML models for instance. It helps convert text into numbers, which the model can then easily work with.

NLTK (Natural Language Toolkit) is a Python library widely used for working with human language data and implementing natural language processing (NLP) tasks. It provides a set of tools, corpora, and resources for tasks such as tokenization, stemming, tagging, parsing, sentiment analysis, and more.

Here are some key features and functionalities of NLTK:

* Text Processing
* Part-of-Speech Tagging
* Named Entity Recognition
* Chunking and Parsing
* Sentiment Analysis:
* WordNet Integration:
* **Scipy**

SciPy is a collection of mathematical algorithms and convenience functions built on the NumPy extension of Python. It adds significant power to the interactive Python session by providing the user with high-level commands and classes for manipulating and visualizing data.

SciPy is a powerful scientific computing library for Python that provides a wide range of mathematical algorithms and functions. It builds upon NumPy, another fundamental library for numerical computing, and extends its capabilities by adding additional tools for scientific and technical computing tasks.

Here are some key features and functionalities of SciPy:

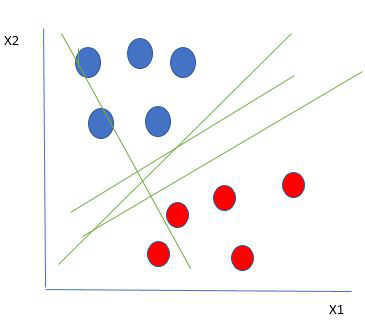
* Numerical Integration:
* Optimization and Root Finding
* Linear Algebra
* Signal and Image Processing
* Statistics

**ALGORITHMS:**

**Support Vector Machine (SVM):**

Support Vector Machine (SVM) is a powerful machine learning algorithm used for linear or nonlinear classification, regression, and even outlier detection tasks. SVMs can be used for a variety of tasks, such as text classification, image classification, spam detection, handwriting identification, gene expression analysis, face detection, and anomaly detection. SVMs are adaptable and efficient in a variety of applications because they can manage high-dimensional data and nonlinear relationships. SVM algorithms are very effective as we try to find the maximum separating hyperplane between the different classes available in the target feature. Support Vector Machine (SVM) is a [supervised machine learning](https://www.geeksforgeeks.org/supervised-unsupervised-learning/) algorithm used for both classification and regression. Though we say regression problems as well it’s best suited for classification. The main objective of the SVM algorithm is to find the optimal [hyperplane](https://www.geeksforgeeks.org/separating-hyperplanes-in-svm/) in an N-dimensional space that can separate the data points in different classes in the feature space. The hyperplane tries that the margin between the closest points of different classes should be as maximum as possible. The dimension of the hyperplane depends upon the number of features. If the number of input features is two, then the hyperplane is just a line. If the number of input features is three, then the hyperplane becomes a 2-D plane. It becomes difficult to imagine when the number of features exceeds three.

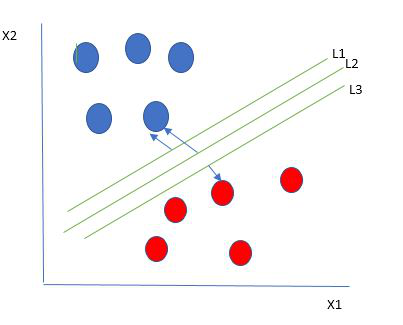
Let’s consider two independent variables x1, x2, and one dependent variable which is either a blue circle or a red circle.



Linearly Separable Data points

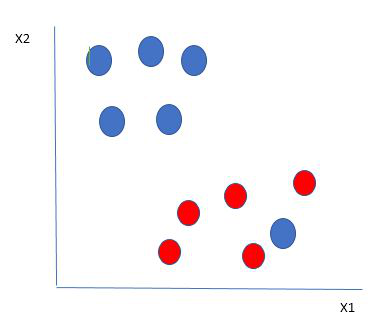
From the figure above it’s very clear that there are multiple lines (our hyperplane here is a line because we are considering only two input features x1, x2) that segregate our data points or do a classification between red and blue circles. So how do we choose the best line or in general the best hyperplane that segregates our data points?

One reasonable choice as the best hyperplane is the one that represents the largest separation or margin between the two classes.



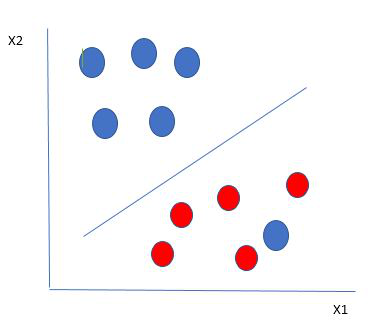
Multiple hyperplanes separate the data from two classes

So we choose the hyperplane whose distance from it to the nearest data point on each side is maximized. If such a hyperplane exists it is known as the **maximum-margin hyperplane/hard margin**. So from the above figure, we choose L2. Let’s consider a scenario like shown below



Selecting hyperplane for data with outlier

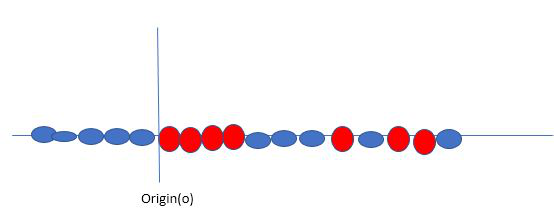
Here we have one blue ball in the boundary of the red ball. So how does SVM classify the data? It’s simple! The blue ball in the boundary of red ones is an outlier of blue balls. The SVM algorithm has the characteristics to ignore the outlier and finds the best hyperplane that maximizes the margin. SVM is robust to outliers.



Hyperplane which is the most optimized one

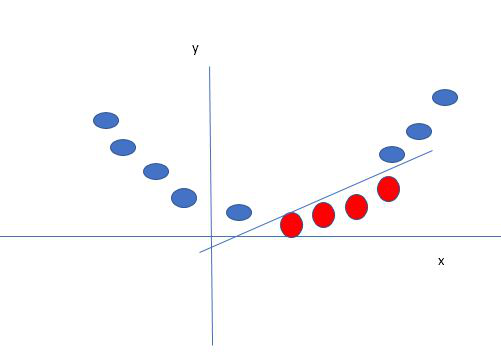
So in this type of data point what SVM does is, finds the maximum margin as done with previous data sets along with that it adds a penalty each time a point crosses the margin. So the margins in these types of cases are called **soft margins**. When there is a soft margin to the data set, the SVM tries to minimize *(1/margin+∧(∑penalty))*. Hinge loss is a commonly used penalty. If no violations no hinge loss.If violations hinge loss proportional to the distance of violation.

Till now, we were talking about linearly separable data(the group of blue balls and red balls are separable by a straight line/linear line). What to do if data are not linearly separable?



Original 1D dataset for classification

Say, our data is shown in the figure above. SVM solves this by creating a new variable using a **kernel**. We call a point xion the line and we create a new variable yi as a function of distance from origin o.so if we plot this we get something like as shown below



Mapping 1D data to 2D to become able to separate the two classes

In this case, the new variable y is created as a function of distance from the origin. A non-linear function that creates a new variable is referred to as a kernel.

**Support Vector Machine Terminology**

1. **Hyperplane:**Hyperplane is the decision boundary that is used to separate the data points of different classes in a feature space. In the case of linear classifications, it will be a linear equation i.e. wx+b = 0.
2. **Support Vectors:**Support vectors are the closest data points to the hyperplane, which makes a critical role in deciding the hyperplane and margin.
3. **Margin**: Margin is the distance between the support vector and hyperplane. The main objective of the support vector machine algorithm is to maximize the margin.  The wider margin indicates better classification performance.
4. **Kernel**: Kernel is the mathematical function, which is used in SVM to map the original input data points into high-dimensional feature spaces, so, that the hyperplane can be easily found out even if the data points are not linearly separable in the original input space. Some of the common kernel functions are linear, polynomial, radial basis function(RBF), and sigmoid.
5. **Hard Margin:** The maximum-margin hyperplane or the hard margin hyperplane is a hyperplane that properly separates the data points of different categories without any misclassifications.
6. **Soft Margin:**When the data is not perfectly separable or contains outliers, SVM permits a soft margin technique. Each data point has a slack variable introduced by the soft-margin SVM formulation, which softens the strict margin requirement and permits certain misclassifications or violations. It discovers a compromise between increasing the margin and reducing violations.
7. **C:**Margin maximisation and misclassification fines are balanced by the regularisation parameter C in SVM. The penalty for going over the margin or misclassifying data items is decided by it. A stricter penalty is imposed with a greater value of C, which results in a smaller margin and perhaps fewer misclassifications.
8. **Hinge Loss:** A typical loss function in SVMs is hinge loss. It punishes incorrect classifications or margin violations. The objective function in SVM is frequently formed by combining it with the regularisation term.
9. **Dual Problem:** A dual Problem of the optimisation problem that requires locating the Lagrange multipliers related to the support vectors can be used to solve SVM. The dual formulation enables the use of kernel tricks and more effective computing.

**SVM implementation in Python**

Predict if cancer is Benign or malignant. Using historical data about patients diagnosed with cancer enables doctors to differentiate malignant cases and benign ones are given independent attributes.

**Steps**

* Load the breast cancer dataset from sklearn.datasets
* Separate input features and target variables.
* Buil and train the SVM classifiers using RBF kernel.
* Plot the scatter plot of the input features.
* Plot the decision boundary.
* Plot the decision boundary

Example:

# Load the important packages

from sklearn.datasets import load\_breast\_cancer

import matplotlib.pyplot as plt

from sklearn.inspection import DecisionBoundaryDisplay

from sklearn.svm import SVC

# Load the datasets

cancer = load\_breast\_cancer()

X = cancer.data[:, :2]

y = cancer.target

#Build the model

svm = SVC(kernel="rbf", gamma=0.5, C=1.0)

# Trained the model

svm.fit(X, y)

# Plot Decision Boundary

DecisionBoundaryDisplay.from\_estimator(

svm,

X,

response\_method="predict",

cmap=plt.cm.Spectral,

alpha=0.8,

xlabel=cancer.feature\_names[0],

ylabel=cancer.feature\_names[1],

)

# Scatter plot

plt.scatter(X[:, 0], X[:, 1], c=y, s=20, edgecolors="k")

plt.show()

**Output**:



Breast Cancer Classifications with SVM RBF kernel

**K-Nearest Neighbors (KNN):**

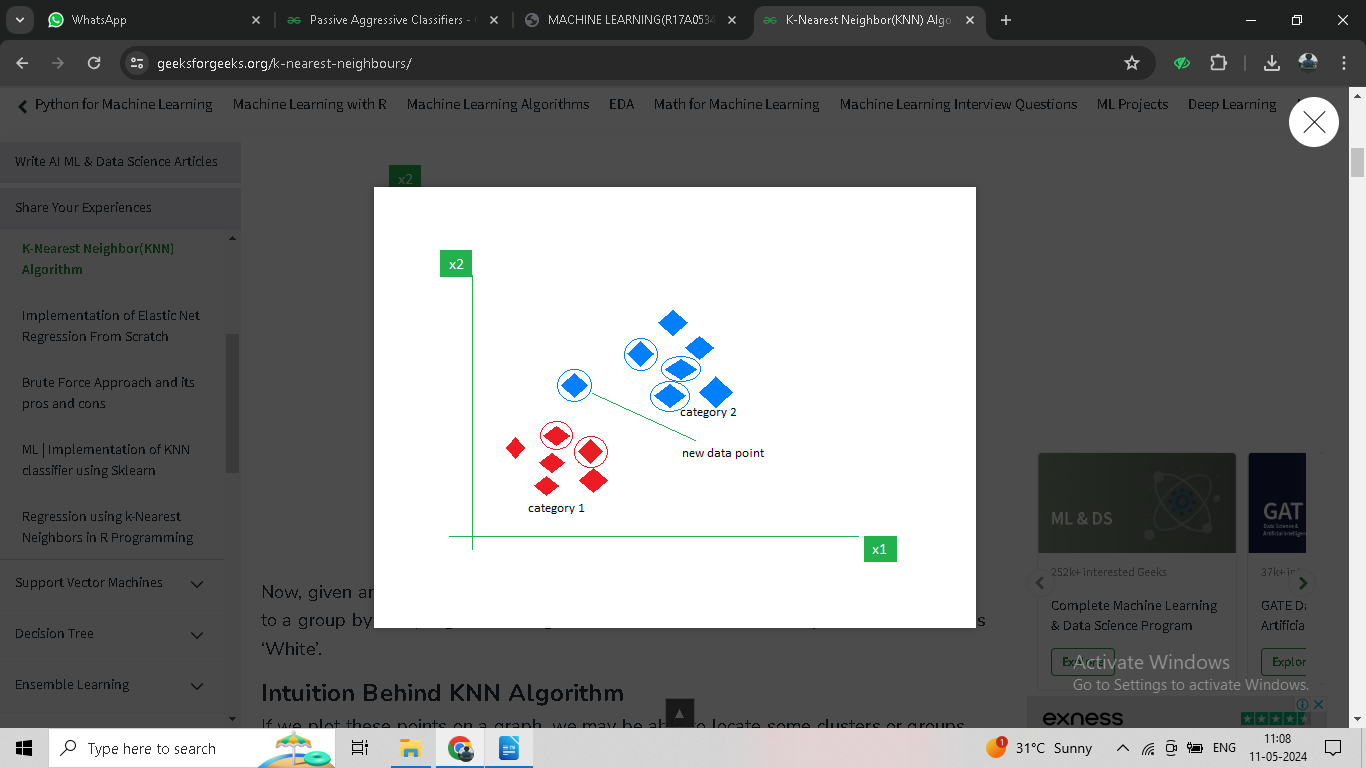
The **K-Nearest Neighbors (KNN) algorithm** is a supervised machine learning method employed to tackle classification and regression problems. Evelyn Fix and Joseph Hodges developed this algorithm in 1951, which was subsequently expanded by Thomas Cover. The article explores the fundamentals, workings, and implementation of the KNN algorithm.

## What is the K-Nearest Neighbors Algorithm?

KNN is one of the most basic yet essential classification algorithms in machine learning. It belongs to the supervised learning domain and finds intense application in pattern recognition, data mining, and intrusion detection.

It is widely disposable in real-life scenarios since it is non-parametric, meaning it does not make any underlying assumptions about the distribution of data (as opposed to other algorithms such as GMM, which assume a Gaussian distribution of the given data). We are given some prior data (also called training data), which classifies coordinates into groups identified by an attribute.

As an example, consider the following table of data points containing two features:



KNN Algorithm working visualization

Now, given another set of data points (also called testing data), allocate these points to a group by analyzing the training set. Note that the unclassified points are marked as ‘White’

## Why do we need a KNN algorithm?

(KNN) algorithm is a versatile and widely used machine learning algorithm that is primarily used for its simplicity and ease of implementation. It does not require any assumptions about the underlying data distribution. It can also handle both numerical and categorical data, making it a flexible choice for various types of datasets in classification and regression tasks. It is a non-parametric method that makes predictions based on the similarity of data points in a given dataset. KNN is less sensitive to outliers compared to other algorithms.

The KNN algorithm works by finding the K nearest neighbors to a given data point based on a distance metric, such as Euclidean distance. The class or value of the data point is then determined by the majority vote or average of the K neighbors. This approach allows the algorithm to adapt to different patterns and make predictions based on the local structure of the data.

## Distance Metrics Used in KNN Algorithm

As we know that the KNN algorithm helps us identify the nearest points or the groups for a query point. But to determine the closest groups or the nearest points for a query point we need some metric. For this purpose, we use below distance metrics:

### Euclidean Distance

This is nothing but the cartesian distance between the two points which are in the plane/hyperplane. Euclidean distance can also be visualized as the length of the straight line that joins the two points which are into consideration. This metric helps us calculate the net displacement done between the two states of an object.

 \text{distance}(x, X_i) = \sqrt{\sum_{j=1}^{d} (x_j - X_{i_j})^2} ]       

### Manhattan Distance

Manhattan Distance metric is generally used when we are interested in the total distance traveled by the object instead of the displacement. This metric is calculated by summing the absolute difference between the coordinates of the points in n-dimensions.

### Minkowski Distance

We can say that the Euclidean, as well as the Manhattan distance, are special cases of the Minkowski distance.

d\left ( x,y \right )=\left ( {\sum_{i=1}^{n}\left ( x_i-y_i \right )^p} \right )^{\frac{1}{p}}        

From the formula above we can say that when p = 2 then it is the same as the formula for the Euclidean distance and when p = 1 then we obtain the formula for the Manhattan distance.

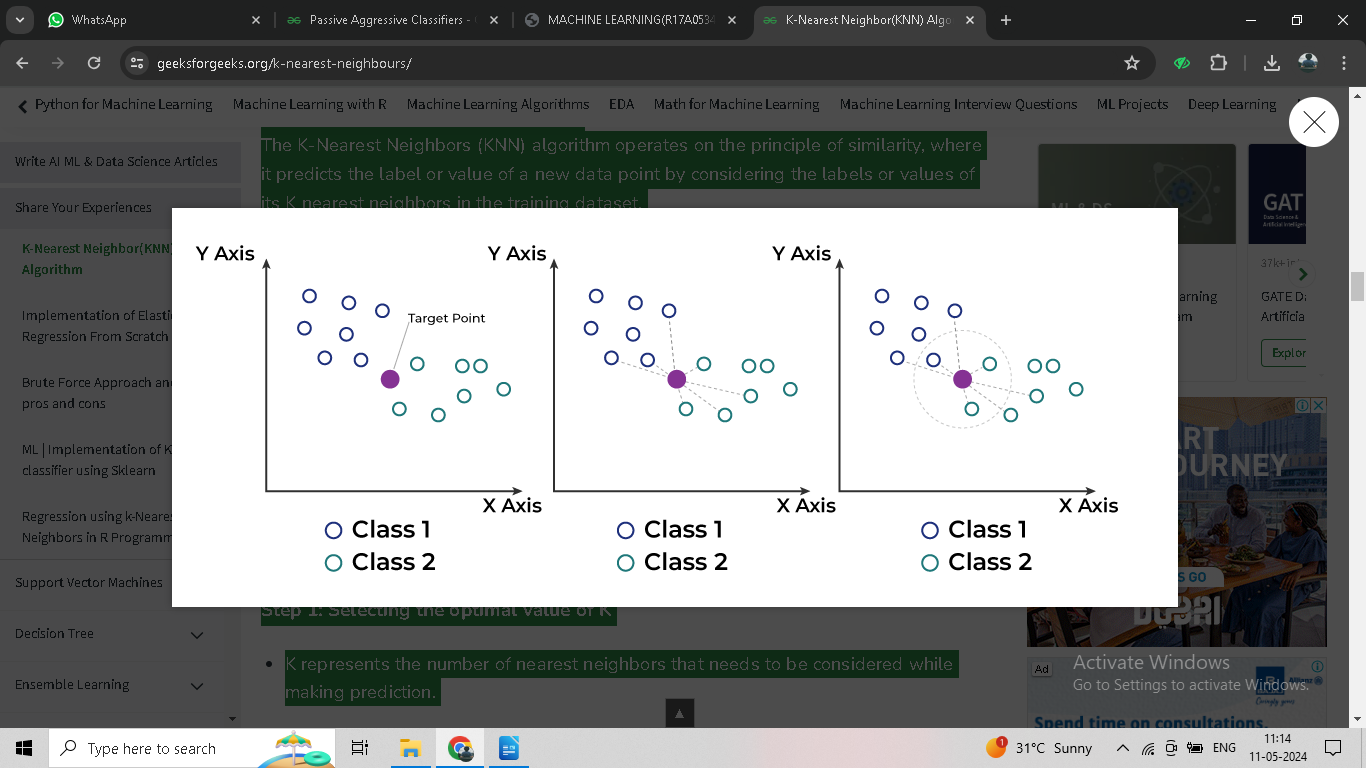
The above-discussed metrics are most common while dealing with a Machine Learning problem but there are other distance metrics as well like Hamming Distance which come in handy while dealing with problems that require overlapping comparisons between two vectors whose contents can be Boolean as well as string values.

## How to choose the value of k for KNN Algorithm?

The value of k is very crucial in the KNN algorithm to define the number of neighbors in the algorithm. The value of k in the k-nearest neighbors (k-NN) algorithm should be chosen based on the input data. If the input data has more outliers or noise, a higher value of k would be better. It is recommended to choose an odd value for k to avoid ties in classification. Cross-validation methods can help in selecting the best k value for the given dataset.

## Workings of KNN algorithm

Thе K-Nearest Neighbors (KNN) algorithm operates on the principle of similarity, where it predicts the label or value of a new data point by considering the labels or values of its K nearest neighbors in the training dataset.



Step-by-Step explanation of how KNN works is discussed below:

### Step 1: Selecting the optimal value of K

* K represents the number of nearest neighbors that needs to be considered while making prediction.

### Step 2: Calculating distance

* To measure the similarity between target and training data points, Euclidean distance is used. Distance is calculated between each of the data points in the dataset and target point.

### Step 3: Finding Nearest Neighbors

* The k data points with the smallest distances to the target point are the nearest neighbors.

### Step 4: Voting for Classification or Taking Average for Regression

* In the classification problem, the class labels of are determined by performing majority voting. The class with the most occurrences among the neighbors becomes the predicted class for the target data point.
* In the regression problem, the class label is calculated by taking average of the target values of K nearest neighbors. The calculated average value becomes the predicted output for the target data point.

Let X be the training dataset with n data points, where each data point is represented by a d-dimensional feature vector X_i        and Y be the corresponding labels or values for each data point in X. Given a new data point x, the algorithm calculates the distance between x and each data point X_i        in X using a distance metric, such as Euclidean distance: \text{distance}(x, X_i) = \sqrt{\sum_{j=1}^{d} (x_j - X_{i_j})^2} ]       

The algorithm selects the K data points from X that have the shortest distances to x. For classification tasks, the algorithm assigns the label y that is most frequent among the K nearest neighbors to x. For regression tasks, the algorithm calculates the average or weighted average of the values y of the K nearest neighbors and assigns it as the predicted value for x.

## Advantages of the KNN Algorithm

* **Easy to implement** as the complexity of the algorithm is not that high.
* **Adapts Easily** – As per the working of the KNN algorithm it stores all the data in memory storage and hence whenever a new example or data point is added then the algorithm adjusts itself as per that new example and has its contribution to the future predictions as well.
* **Few Hyper parameters** – The only parameters which are required in the training of a KNN algorithm are the value of k and the choice of the distance metric which we would like to choose from our evaluation metric.

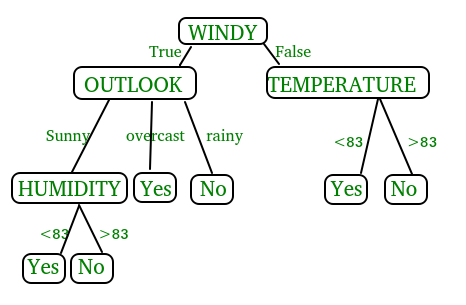
## Disadvantages of the KNN Algorithm

* **Does not scale** – As we have heard about this that the KNN algorithm is also considered a Lazy Algorithm. The main significance of this term is that this takes lots of computing power as well as data storage. This makes this algorithm both time-consuming and resource exhausting.
* **Curse of Dimensionality** – There is a term known as the peaking phenomenon according to this the KNN algorithm is affected by the curse of dimensionality which implies the algorithm faces a hard time classifying the data points properly when the dimensionality is too high.
* **Prone to Over fitting** – As the algorithm is affected due to the curse of dimensionality it is prone to the problem of overfitting as well. Hence generally feature selection as well as dimensionality reduction techniques are applied to deal with this problem.

**Decision Tree:**

Decision Tree is one of the most powerful and popular algorithms. Python Decision-tree algorithm falls under the category of supervised learning algorithms. It works for both continuous as well as categorical output variables. In this article, We are going to implement a Decision tree in Python algorithm on the [Balance Scale Weight & Distance Database](https://archive.ics.uci.edu/ml/machine-learning-databases/balance-scale/) presented on the UCI. A Decision tree is a tree-like structure that represents a set of decisions and their possible consequences. Each node in the tree represents a decision, and each branch represents an outcome of that decision. The leaves of the tree represent the final decisions or predictions.

Decision trees are created by recursively partitioning the data into smaller and smaller subsets. At each partition, the data is split based on a specific feature, and the split is made in a way that maximizes the information gain.



In the above figure, decision tree is a flowchart-like tree structure that is used to make decisions. It consists of Root Node(WINDY), Internal nodes(OUTLOOK, TEMPERATURE), which represent tests on attributes, and leaf nodes, which represent the final decisions. The branches of the tree represent the possible outcomes of the tests.

**Key Components of Decision Trees in Python**

1. **Root Node:** The decision tree’s starting node, which stands for the complete dataset.
2. **Branch Nodes:**Internal nodes that represent decision points, where the data is split based on a specific attribute.
3. **Leaf Nodes:**Final categorization or prediction-representing terminal nodes.
4. **Decision Rules:** Rules that govern the splitting of data at each branch node.
5. **Attribute Selection:** The process of choosing the most informative attribute for each split.
6. **Splitting Criteria:** Metrics like information gain, entropy, or the Gini Index are used to calculate the optimal split.

**Assumptions we make while using Decision tree**

* At the beginning, we consider the whole training set as the root.
* Attributes are assumed to be categorical for information gain and for gini index, attributes are assumed to be continuous.
* On the basis of attribute values records are distributed recursively.
* We use statistical methods for ordering attributes as root or internal node

Implementing Decision Tree Using Python:

from sklearn.datasets import load\_iris

from sklearn.model\_selection import train\_test\_split

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy\_score

# Load the Iris dataset

iris = load\_iris()

X = iris.data

y = iris.target

# Split the dataset into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Initialize the decision tree classifier

clf = DecisionTreeClassifier()

# Train the classifier on the training data

clf.fit(X\_train, y\_train)

# Make predictions on the testing data

y\_pred = clf.predict(X\_test)

# Calculate the accuracy of the classifier

accuracy = accuracy\_score(y\_test, y\_pred)

print("Accuracy:", accuracy)

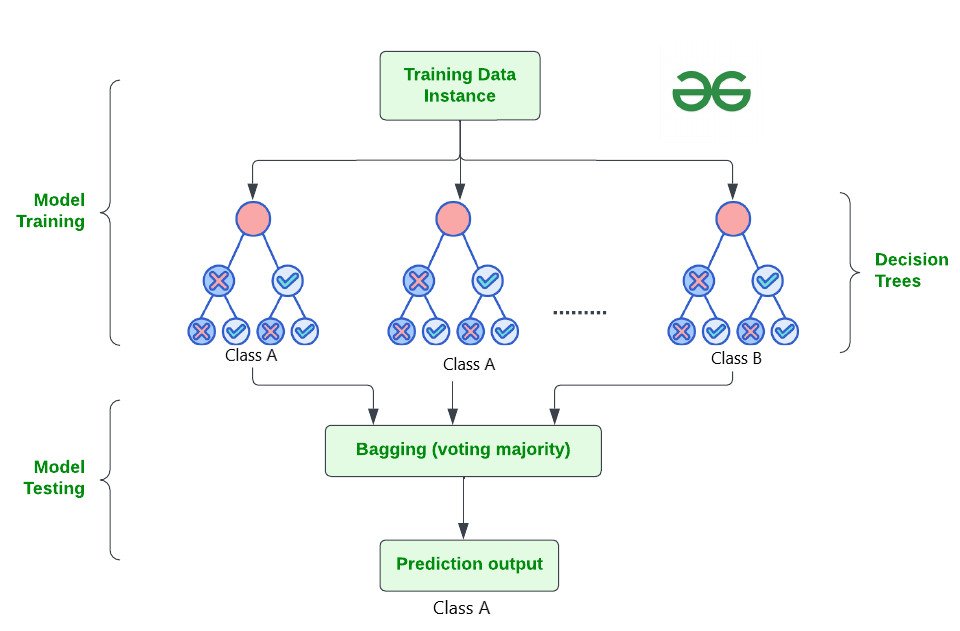
In This Example :

* We first import necessary modules from scikit-learn.
* We load the Iris dataset, a popular dataset for classification tasks.
* We split the dataset into training and testing sets using the **train test split** function.
* We initialize a Decision Tree Classifier.
* We train the classifier on the training data using the **fit** method.
* We make predictions on the testing data using the **predict** method.
* Finally, we calculate the accuracy of the classifier using the **accuracy score** function.

**.Random Forest:**

Machine learning, a fascinating blend of computer science and statistics, has witnessed incredible progress, with one standout algorithm being the **Random Forest**. **Random forests or Random Decision Trees** is a collaborative team of **decision trees** that work together to provide a single output. Originating in 2001 through Leo Breiman, Random Forest has become a cornerstone for machine learning enthusiasts. In this article, we will explore the fundamentals and implementation of **Random Forest Algorithm.**

Random Forest algorithm is a powerful tree learning technique in Machine Learning. It works by creating a number of Decision Trees during the training phase. Each tree is constructed using a random subset of the data set to measure a random subset of features in each partition. This randomness introduces variability among individual trees, reducing the risk of overfitting and improving overall prediction performance. In prediction, the algorithm aggregates the results of all trees, either by voting (for classification tasks) or by averaging (for regression tasks) This collaborative decision-making process, supported by multiple trees with their insights, provides an example stable and precise results. Random forests are widely used for classification and regression functions, which are known for their ability to handle complex data, reduce overfitting, and provide reliable forecasts in different environments.



**How Does Random Forest Work?**

The random Forest algorithm works in several steps which are discussed below–>

* **Ensemble of Decision Trees:** Random Forest leverages the power of ensemble learning by constructing an army of Decision Trees. These trees are like individual experts, each specializing in a particular aspect of the data. Importantly, they operate independently, minimizing the risk of the model being overly influenced by the nuances of a single tree.
* **Random Feature Selection:** To ensure that each decision tree in the ensemble brings a unique perspective, Random Forest employs random feature selection. During the training of each tree, a random subset of features is chosen. This randomness ensures that each tree focuses on different aspects of the data, fostering a diverse set of predictors within the ensemble.
* **Bootstrap Aggregating or Bagging:** The technique of bagging is a cornerstone of Random Forest’s training strategy which involves creating multiple bootstrap samples from the original dataset, allowing instances to be sampled with replacement. This results in different subsets of data for each decision tree, introducing variability in the training process and making the model more robust.
* **Decision Making and Voting:** When it comes to making predictions, each decision tree in the Random Forest casts its vote. For classification tasks, the final prediction is determined by the [mode](https://www.geeksforgeeks.org/mode/) (most frequent prediction) across all the trees. In regression tasks, the average of the individual tree predictions is taken. This internal voting mechanism ensures a balanced and collective decision-making process.

**Key Features of Random Forest**

Some of the Key Features of Random Forest are discussed below–>

1. **High Predictive Accuracy:** Imagine Random Forest as a team of decision-making wizards. Each wizard (decision tree) looks at a part of the problem, and together, they weave their insights into a powerful prediction tapestry. This teamwork often results in a more accurate model than what a single wizard could achieve.
2. **Resistance to Overfitting:** Random Forest is like a cool-headed mentor guiding its apprentices (decision trees). Instead of letting each apprentice memorize every detail of their training, it encourages a more well-rounded understanding. This approach helps prevent getting too caught up with the training data which makes the model less prone to overfitting.
3. **Large Datasets Handling:** Dealing with a mountain of data? Random Forest tackles it like a seasoned explorer with a team of helpers (decision trees). Each helper takes on a part of the dataset, ensuring that the expedition is not only thorough but also surprisingly quick.
4. **Variable Importance Assessment:** Think of Random Forest as a detective at a crime scene, figuring out which clues (features) matter the most. It assesses the importance of each clue in solving the case, helping you focus on the key elements that drive predictions.
5. **Built-in Cross-Validation:** Random Forest is like having a personal coach that keeps you in check. As it trains each decision tree, it also sets aside a secret group of cases (out-of-bag) for testing. This built-in validation ensures your model doesn’t just ace the training but also performs well on new challenges.
6. **Handling Missing Values:** Life is full of uncertainties, just like datasets with missing values. Random Forest is the friend who adapts to the situation, making predictions using the information available. It doesn’t get flustered by missing pieces; instead, it focuses on what it can confidently tell us.
7. **Parallelization for Speed:** Random Forest is your time-saving buddy. Picture each decision tree as a worker tackling a piece of a puzzle simultaneously. This parallel approach taps into the power of modern tech, making the whole process faster and more efficient for handling large-scale projects.

## Implement Random Forest for Classification :

# Import necessary libraries

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score, classification\_report

import warnings

warnings.filterwarnings('ignore')

# Load the Titanic dataset

url = "https://raw.githubusercontent.com/datasciencedojo/datasets/master/titanic.csv"

titanic\_data = pd.read\_csv(url)

# Drop rows with missing target values

titanic\_data = titanic\_data.dropna(subset=['Survived'])

# Select relevant features and target variable

X = titanic\_data[['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare']]

y = titanic\_data['Survived']

# Convert categorical variable 'Sex' to numerical using .loc

X.loc[:, 'Sex'] = X['Sex'].map({'female': 0, 'male': 1})

# Handle missing values in the 'Age' column using .loc

X.loc[:, 'Age'].fillna(X['Age'].median(), inplace=True)

# Split the dataset into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Create a Random Forest Classifier

rf\_classifier = RandomForestClassifier(n\_estimators=100, random\_state=42)

# Train the classifier

rf\_classifier.fit(X\_train, y\_train)

# Make predictions on the test set

y\_pred = rf\_classifier.predict(X\_test)

# Evaluate the model

accuracy = accuracy\_score(y\_test, y\_pred)

classification\_rep = classification\_report(y\_test, y\_pred)

# Print the results

print(f"Accuracy: {accuracy:.2f}")

print("\nClassification Report:\n", classification\_rep)

**Output:**

Accuracy: 0.80

Classification Report:

precision recall f1-score support

0 0.82 0.85 0.83 105

1 0.77 0.73 0.75 74

accuracy 0.80 179

macro avg 0.79 0.79 0.79 179

weighted avg 0.80 0.80 0.80 179

In the above code, we’re using a Random Forest Classifier to make sense of the Titanic dataset. First, we gather our tools – importing libraries to handle data and evaluate our model. Next, we dive into the Titanic dataset, fixing missing information and choosing important details like a detective solving a mystery. We even teach the computer to understand ‘male’ and ‘female’ by turning them into numbers. Then, we split our dataset into pieces – one part to train our model, and the other to test its newfound skills. Our Random Forest Classifier is like a student, learning from the training set. Once trained, it faces a test – making predictions on the test set. We’re like judges, using a classification report to grade how well our model did.

**5.2 Sample Code:**

**Views.py**

**from django.shortcuts import render**

**from django.template import RequestContext**

**from django.contrib import messages**

**from django.http import HttpResponse**

**import os**

**import pandas as pd**

**import numpy as np**

**import pickle**

**from sklearn.feature\_extraction.text import TfidfVectorizer**

**import pymysql**

**from numpy import dot**

**from numpy.linalg import norm**

**from django.core.files.storage import FileSystemStorage**

**from datetime import date**

**from string import punctuation**

**from nltk.corpus import stopwords**

**import nltk**

**from nltk.stem import WordNetLemmatizer**

**from nltk.stem import PorterStemmer**

**from sklearn.model\_selection import train\_test\_split**

**from sklearn.feature\_extraction.text import TfidfVectorizer #loading tfidf vector**

**from sklearn.preprocessing import LabelEncoder**

**from sklearn.metrics import accuracy\_score**

**from sklearn.metrics import precision\_score**

**from sklearn.metrics import recall\_score**

**from sklearn.metrics import f1\_score**

**from sklearn.preprocessing import LabelEncoder**

**from sklearn.preprocessing import StandardScaler**

**from sklearn.neighbors import KNeighborsClassifier**

**from numpy import dot**

**from numpy.linalg import norm**

**from sklearn.ensemble import RandomForestClassifier**

**from sklearn import svm**

**global uname, tfidf\_vectorizer, scaler**

**global X\_train, X\_test, y\_train, y\_test**

**accuracy, precision, recall, fscore = [], [], [], []**

**dataset = pd.read\_csv("Dataset/BankFAQs.csv")**

**dataset = dataset.dropna()**

**question = []**

**answer = []**

**Y = []**

**labels = np.unique(dataset['Class'])**

**le = LabelEncoder()**

**dataset['Class'] = pd.Series(le.fit\_transform(dataset['Class'].astype(str))) #encoding non-numeric labels into numeric**

**Y = dataset['Class']**

**dataset = dataset.values**

**#define object to remove stop words and other text processing**

**stop\_words = set(stopwords.words('english'))**

**lemmatizer = WordNetLemmatizer()**

**ps = PorterStemmer()**

**#define function to clean text by removing stop words and other special symbols**

**def cleanText(doc):**

**tokens = doc.split()**

**table = str.maketrans('', '', punctuation)**

**tokens = [w.translate(table) for w in tokens]**

**tokens = [word for word in tokens if word.isalpha()]**

**tokens = [w for w in tokens if not w in stop\_words]**

**tokens = [word for word in tokens if len(word) > 1]**

**tokens = [ps.stem(token) for token in tokens]**

**tokens = [lemmatizer.lemmatize(token) for token in tokens]**

**tokens = ' '.join(tokens)**

**return tokens**

**question = np.load("model/question.npy")**

**answer = np.load("model/answer.npy")**

**Y = np.load("model/Y.npy")**

**tfidf\_vectorizer = TfidfVectorizer(stop\_words=stop\_words, use\_idf=True, smooth\_idf=False, norm=None, decode\_error='replace', max\_features=600)**

**tfidf\_X = tfidf\_vectorizer.fit\_transform(question).toarray()**

**print(tfidf\_X.shape)**

**#scaler = StandardScaler()**

**#X = scaler.fit\_transform(tfidf\_X)**

**X = tfidf\_X**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, Y, test\_size=0.2) #split data into train & test**

**X\_train, X\_test1, y\_train, y\_test1 = train\_test\_split(X, Y, test\_size=0.1) #split data into train & test**

**def calculateMetrics(algorithm, predict, y\_test):**

**global accuracy, precision, recall, fscore**

**a = accuracy\_score(y\_test,predict)\*100**

**p = precision\_score(y\_test, predict,average='macro') \* 100**

**r = recall\_score(y\_test, predict,average='macro') \* 100**

**f = f1\_score(y\_test, predict,average='macro') \* 100**

**accuracy.append(a)**

**precision.append(p)**

**recall.append(r)**

**fscore.append(f)**

**knn = KNeighborsClassifier(n\_neighbors=2)**

**knn.fit(X\_train, y\_train)**

**predict = knn.predict(X\_test)**

**calculateMetrics("KNN", predict, y\_test)**

**rf = RandomForestClassifier()**

**rf.fit(X\_train, y\_train)**

**predict = rf.predict(X\_test)**

**calculateMetrics("Random Forest", predict, y\_test)**

**svm\_cls = svm.SVC()**

**svm\_cls.fit(X\_train, y\_train)**

**predict = svm\_cls.predict(X\_test)**

**calculateMetrics("SVM", predict, y\_test)**

**def TextChatbot(request):**

**if request.method == 'GET':**

**return render(request, 'TextChatbot.html', {})**

**def TrainML(request):**

**if request.method == 'GET':**

**output = ''**

**output+='<table border=1 align=center width=100%><tr><th><font size="" color="black">Algorithm Name</th><th><font size="" color="black">Accuracy</th><th><font size="" color="black">Precision</th>'**

**output+='<th><font size="" color="black">Recall</th><th><font size="" color="black">FSCORE</th></tr>'**

**global accuracy, precision, recall, fscore**

**algorithms = ['KNN', 'Random Forest', 'SVM']**

**for i in range(len(algorithms)):**

**output+='<td><font size="" color="black">'+algorithms[i]+'</td><td><font size="" color="black">'+str(accuracy[i])+'</td><td><font size="" color="black">'+str(precision[i])+'</td><td><font size="" color="black">'+str(recall[i])+'</td><td><font size="" color="black">'+str(fscore[i])+'</td></tr>'**

**output+= "</table></br></br></br></br>"**

**context= {'data':output}**

**return render(request, 'AdminScreen.html', context)**

**def UpdateStatus(request):**

**if request.method == 'GET':**

**lid = request.GET['lid']**

**status = request.GET['status']**

**db\_connection = pymysql.connect(host='127.0.0.1',port = 3306,user = 'root', password = 'root', database = 'BankChatbot',charset='utf8')**

**db\_cursor = db\_connection.cursor()**

**student\_sql\_query = "update loan set status='"+status+"' where loan\_id='"+lid+"'"**

**db\_cursor.execute(student\_sql\_query)**

**db\_connection.commit()**

**context= {'data':'Application Status successfully Updated : '+status}**

**return render(request, 'AdminScreen.html', context)**

**def ViewApplications(request):**

**if request.method == 'GET':**

**output = ''**

**output+='<table border=1 align=center width=100%><tr><th><font size="" color="black">Application ID</th><th><font size="" color="black">Applicant Name</th>'**

**output+='<th><font size="" color="black">Loan Purpose</th><th><font size="" color="black">Amount</th><th><font size="" color="black">Pan No</th>'**

**output+='<th><font size="" color="black">Aadhar No</th><th><font size="" color="black">Applied Date</th>'**

**output+='<th><font size="" color="black">Status</th><th><font size="" color="black">Accept Application</th><th><font size="" color="black">Reject Application</th></tr>'**

**con = pymysql.connect(host='127.0.0.1',port = 3306,user = 'root', password = 'root', database = 'BankChatbot',charset='utf8')**

**with con:**

**cur = con.cursor()**

**cur.execute("select \* from loan where status='Pending'")**

**rows = cur.fetchall()**

**output+='<tr>'**

**for row in rows:**

**output+='<td><font size="" color="black">'+str(row[0])+'</td><td><font size="" color="black">'+str(row[1])+'</td>'**

**output+='<td><font size="" color="black">'+row[2]+'</td><td><font size="" color="black">'+row[3]+'</td><td><font size="" color="black">'+row[4]+'</td>'**

**output += '<td><font size="" color="black">'+row[5]+'</td>'**

**output += '<td><font size="" color="black">'+row[6]+'</td><td><font size="" color="black">'+row[7]+'</td>'**

**output+='<td><a href=\'UpdateStatus?lid='+str(row[0])+'&status=Accepted\'><font size=3 color=black>Click Here to Accept</font></a></td>'**

**output+='<td><a href=\'UpdateStatus?lid='+str(row[0])+'&status=Rejected\'><font size=3 color=black>Click Here to Reject</font></a></td></tr>'**

**output+= "</table></br></br></br></br>"**

**context= {'data':output}**

**return render(request, 'AdminScreen.html', context)**

**def ApplyLoan(request):**

**if request.method == 'GET':**

**return render(request, 'ApplyLoan.html', {})**

**def ApplyLoanAction(request):**

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

**global uname**

**purpose = request.POST.get('t1', False)**

**amount = request.POST.get('t2', False)**

**pan = request.POST.get('t3', False)**

**aadhar = request.POST.get('t4', False)**

**today = str(date.today())**

**loan\_id = 0**

**con = pymysql.connect(host='127.0.0.1',port = 3306,user = 'root', password = 'root', database = 'BankChatbot',charset='utf8')**

**with con:**

**cur = con.cursor()**

**cur.execute("select max(loan\_id) FROM loan")**

**rows = cur.fetchall()**

**for row in rows:**

**loan\_id = row[0]**

**if loan\_id is not None:**

**loan\_id += 1**

**else:**

**loan\_id = 1**

**db\_connection = pymysql.connect(host='127.0.0.1',port = 3306,user = 'root', password = 'root', database = 'BankChatbot',charset='utf8')**

**db\_cursor = db\_connection.cursor()**

**student\_sql\_query = "INSERT INTO loan(loan\_id,username,loan\_purpose,amount,pan\_no,aadhar\_no,applied\_date,status) VALUES('"+str(loan\_id)+"','"+uname+"','"+purpose+"','"+amount+"','"+pan+"','"+aadhar+"','"+today+"','Pending')"**

**db\_cursor.execute(student\_sql\_query)**

**db\_connection.commit()**

**context= {'data':'Your Application submitted successfully with ID : '+str(loan\_id)+"<br/>Our Admin will reply"}**

**return render(request, 'ApplyLoan.html', context)**

**def saveInteraction(user\_question, output):**

**global uname**

**today = str(date.today())**

**interact\_id = 0**

**con = pymysql.connect(host='127.0.0.1',port = 3306,user = 'root', password = 'root', database = 'BankChatbot',charset='utf8')**

**with con:**

**cur = con.cursor()**

**cur.execute("select max(interact\_id) FROM interaction")**

**rows = cur.fetchall()**

**for row in rows:**

**interact\_id = row[0]**

**if interact\_id is not None:**

**interact\_id += 1**

**else:**

**interact\_id = 1**

**db\_connection = pymysql.connect(host='127.0.0.1',port = 3306,user = 'root', password = 'root', database = 'BankChatbot',charset='utf8')**

**db\_cursor = db\_connection.cursor()**

**student\_sql\_query = "INSERT INTO interaction(interact\_id,username,question,answer,interact\_date) VALUES('"+str(interact\_id)+"','"+uname+"','"+user\_question+"','"+output+"','"+today+"')"**

**db\_cursor.execute(student\_sql\_query)**

**db\_connection.commit()**

**def ChatData(request):**

**if request.method == 'GET':**

**global answer, tfidf\_vectorizer, X, question, scaler**

**user\_question = request.GET.get('mytext', False)**

**query = user\_question**

**print(query)**

**query = query.strip().lower()**

**query = cleanText(query)#clean description**

**testData = tfidf\_vectorizer.transform([query]).toarray()**

**#testData = scaler.transform(testData)**

**testData = testData[0]**

**print(testData.shape)**

**output = "Sorry! unable to answer"**

**index = -1**

**max\_accuracy = 0**

**for i in range(len(X)):**

**predict\_score = dot(X[i], testData)/(norm(X[i])\*norm(testData))**

**if predict\_score > max\_accuracy:**

**max\_accuracy = predict\_score**

**index = i**

**if index != -1:**

**output = answer[index]**

**saveInteraction(user\_question, output)**

**print(output)**

**return HttpResponse("Chatbot: "+output, content\_type="text/plain")**

**def AddQuestion(request):**

**if request.method == 'GET':**

**return render(request, 'AddQuestion.html', {})**

**def Signup(request):**

**if request.method == 'GET':**

**return render(request, 'Signup.html', {})**

**def index(request):**

**if request.method == 'GET':**

**return render(request, 'index.html', {})**

**def UserLogin(request):**

**if request.method == 'GET':**

**return render(request, 'UserLogin.html', {})**

**def AdminLogin(request):**

**if request.method == 'GET':**

**return render(request, 'AdminLogin.html', {})**

**def AdminLoginAction(request):**

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

**global userid**

**user = request.POST.get('t1', False)**

**password = request.POST.get('t2', False)**

**if user == "admin" and password == "admin":**

**context= {'data':'Welcome '+user}**

**return render(request, 'AdminScreen.html', context)**

**else:**

**context= {'data':'Invalid Login'}**

**return render(request, 'AdminLogin.html', context)**

**def UserLoginAction(request):**

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

**global uname**

**username = request.POST.get('t1', False)**

**password = request.POST.get('t2', False)**

**index = 0**

**con = pymysql.connect(host='127.0.0.1',port = 3306,user = 'root', password = 'root', database = 'BankChatbot',charset='utf8')**

**with con:**

**cur = con.cursor()**

**cur.execute("select \* FROM register")**

**rows = cur.fetchall()**

**for row in rows:**

**if row[0] == username and password == row[1]:**

**uname = username**

**index = 1**

**break**

**if index == 1:**

**context= {'data':'welcome '+username}**

**return render(request, 'UserScreen.html', context)**

**else:**

**context= {'data':'login failed'}**

**return render(request, 'UserLogin.html', context)**

**def SignupAction(request):**

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

**username = request.POST.get('t1', False)**

**password = request.POST.get('t2', False)**

**contact = request.POST.get('t3', False)**

**email = request.POST.get('t4', False)**

**address = request.POST.get('t5', False)**

**status = "none"**

**con = pymysql.connect(host='127.0.0.1',port = 3306,user = 'root', password = 'root', database = 'BankChatbot',charset='utf8')**

**with con:**

**cur = con.cursor()**

**cur.execute("select \* FROM register")**

**rows = cur.fetchall()**

**for row in rows:**

**if row[0] == username:**

**status = "Username already exists"**

**break**

**if status == "none":**

**db\_connection = pymysql.connect(host='127.0.0.1',port = 3306,user = 'root', password = 'root', database = 'BankChatbot',charset='utf8')**

**db\_cursor = db\_connection.cursor()**

**student\_sql\_query = "INSERT INTO register(username,password,contact,email,address) VALUES('"+username+"','"+password+"','"+contact+"','"+email+"','"+address+"')"**

**db\_cursor.execute(student\_sql\_query)**

**db\_connection.commit()**

**print(db\_cursor.rowcount, "Record Inserted")**

**if db\_cursor.rowcount == 1:**

**status = "Signup Process Completed. You can Login now"**

**context= {'data': status}**

**return render(request, 'Signup.html', context)**

**def ViewUser(request):**

**if request.method == 'GET':**

**output = ''**

**output+='<table border=1 align=center width=100%><tr><th><font size="" color="black">Username</th><th><font size="" color="black">Password</th><th><font size="" color="black">Contact No</th>'**

**output+='<th><font size="" color="black">Email ID</th><th><font size="" color="black">Address</th></tr>'**

**con = pymysql.connect(host='127.0.0.1',port = 3306,user = 'root', password = 'root', database = 'BankChatbot',charset='utf8')**

**with con:**

**cur = con.cursor()**

**cur.execute("select \* from register")**

**rows = cur.fetchall()**

**output+='<tr>'**

**for row in rows:**

**output+='<td><font size="" color="black">'+row[0]+'</td><td><font size="" color="black">'+str(row[1])+'</td><td><font size="" color="black">'+row[2]+'</td><td><font size="" color="black">'+row[3]+'</td><td><font size="" color="black">'+row[4]+'</td></tr>'**

**output+= "</table></br></br></br></br>"**

**context= {'data':output}**

**return render(request, 'AdminScreen.html', context)**

**def ViewChats(request):**

**if request.method == 'GET':**

**output = ''**

**output+='<table border=1 align=center width=100%><tr><th><font size="" color="black">Interaction ID</th><th><font size="" color="black">Username</th><th><font size="" color="black">Question</th>'**

**output+='<th><font size="" color="black">Chatbot Answer</th><th><font size="" color="black">Interaction Date</th></tr>'**

**con = pymysql.connect(host='127.0.0.1',port = 3306,user = 'root', password = 'root', database = 'BankChatbot',charset='utf8')**

**with con:**

**cur = con.cursor()**

**cur.execute("select \* from interaction")**

**rows = cur.fetchall()**

**output+='<tr>'**

**for row in rows:**

**output+='<td><font size="" color="black">'+str(row[0])+'</td><td><font size="" color="black">'+str(row[1])+'</td><td><font size="" color="black">'+row[2]+'</td><td><font size="" color="black">'+row[3]+'</td><td><font size="" color="black">'+row[4]+'</td></tr>'**

**output+= "</table></br></br></br></br>"**

**context= {'data':output}**

**return render(request, 'AdminScreen.html', context)**

**def ViewStatus(request):**

**if request.method == 'GET':**

**global uname**

**output = ''**

**output+='<table border=1 align=center width=100%><tr><th><font size="" color="black">Application ID</th><th><font size="" color="black">Applicant Name</th>'**

**output+='<th><font size="" color="black">Loan Purpose</th><th><font size="" color="black">Amount</th><th><font size="" color="black">Pan No</th>'**

**output+='<th><font size="" color="black">Aadhar No</th><th><font size="" color="black">Applied Date</th>'**

**output+='<th><font size="" color="black">Status</th></tr>'**

**con = pymysql.connect(host='127.0.0.1',port = 3306,user = 'root', password = 'root', database = 'BankChatbot',charset='utf8')**

**with con:**

**cur = con.cursor()**

**cur.execute("select \* from loan where username='"+uname+"'")**

**rows = cur.fetchall()**

**output+='<tr>'**

**for row in rows:**

**output+='<td><font size="" color="black">'+str(row[0])+'</td><td><font size="" color="black">'+str(row[1])+'</td>'**

**output+='<td><font size="" color="black">'+row[2]+'</td><td><font size="" color="black">'+row[3]+'</td><td><font size="" color="black">'+row[4]+'</td>'**

**output += '<td><font size="" color="black">'+row[5]+'</td>'**

**output += '<td><font size="" color="black">'+row[6]+'</td><td><font size="" color="black">'+row[7]+'</td></tr>'**

**output+= "</table></br></br></br></br>"**

**context= {'data':output}**

**return render(request, 'UserScreen.html', context)**

**6. TESTING**

**Implementation and Testing:**

Implementation is one of the most important tasks in project is the phase in which one has to be cautions because all the efforts undertaken during the project will be very interactive. Implementation is the most crucial stage in achieving successful system and giving the users confidence that the new system is workable and effective. Each program is tested individually at the time of development using the sample data and has verified that these programs link together in the way specified in the program specification. The computer system and its environment are tested to the satisfaction of the user.

## **Implementation**

## The implementation phase is less creative than system design. It is primarily concerned with user training, and file conversion. The system may be requiring extensive user training. The initial parameters of the system should be modifies as a result of a programming. A simple operating procedure is provided so that the user can understand the different functions clearly and quickly. The different reports can be obtained either on the inkjet or dot matrix printer, which is available at the disposal of the user. The proposed system is very easy to implement. In general implementation is used to mean the process of converting a new or revised system design into an operational one.

## **Testing**

Testing is the process where the test data is prepared and is used for testing the modules individually and later the validation given for the fields. Then the system testing takes place which makes sure that all components of the system property functions as a unit. The test data should be chosen such that it passed through all possible condition. Actually testing is the state of implementation which aimed at ensuring that the system works accurately and efficiently before the actual operation commence. The following is the description of the testing strategies, which were carried out during the testing period.

### **System Testing**

Testing has become an integral part of any system or project especially in the field of information technology. The importance of testing is a method of justifying, if one is ready to move further, be it to be check if one is capable to with stand the rigors of a particular situation cannot be underplayed and that is why testing before development is so critical. When the software is developed before it is given to user to use the software must be tested whether it is solving the purpose for which it is developed. This testing involves various types through which one can ensure the software is reliable. The program was tested logically and pattern of execution of the program for a set of data are repeated. Thus the code was exhaustively checked for all possible correct data and the outcomes were also checked.

**Module Testing**

To locate errors, each module is tested individually. This enables us to detect error and correct it without affecting any other modules. Whenever the program is not satisfying the required function, it must be corrected to get the required result. Thus all the modules are individually tested from bottom up starting with the smallest and lowest modules and proceeding to the next level. Each module in the system is tested separately. For example the job classification module is tested separately. This module is tested with different job and its approximate execution time and the result of the test is compared with the results that are prepared manually. The comparison shows that the results proposed system works efficiently than the existing system. Each module in the system is tested separately. In this system the resource classification and job scheduling modules are tested separately and their corresponding results are obtained which reduces the process waiting time.

**Integration Testing**

After the module testing, the integration testing is applied. When linking the modules there may be chance for errors to occur, these errors are corrected by using this testing. In this system all modules are connected and tested. The testing results are very correct. Thus the mapping of jobs with resources is done correctly by the system.

**Acceptance Testing**

When that user fined no major problems with its accuracy, the system passers through a final acceptance test. This test confirms that the system needs the original goals, objectives and requirements established during analysis without actual execution which elimination wastage of time and money acceptance tests on the shoulders of users and management, it is finally acceptable and ready for the operation

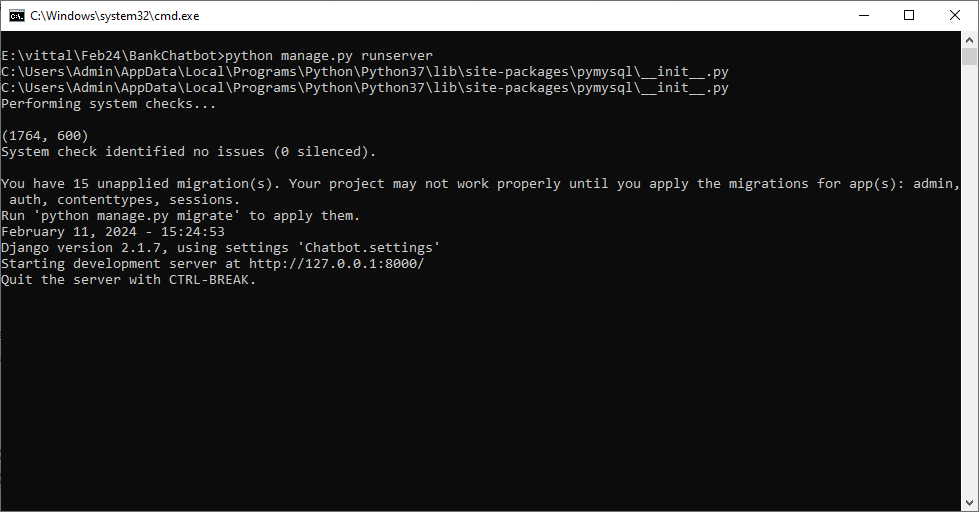
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case Id** | **Test Case Name** | **Test Case Desc.** | **Test Steps** | | | **Test Case Status** | **Test Priority** |
| **Step** | **Expected** | **Actual** |
| 01 | Admin Login | Test whether Admin Login Is Successfully done or not | If the Admin Login may not Done | We cannot do further operations | Dataset uploaded we will do further operations | High | High |
| 02 | New User Sign Up | Test whether the new user signup Successfully done or not | If the New user signup May not done | We cannot do further operations | we will do further  operations | High | High |
| 03 | User Login | Test whether User Login Successfully done or not | If the  User Login may not done | We cannot do further operations | we will do further  operations | High | High |
| 04 | Train ML Algorithm | Test whether ML Algorithm Run Successfully or not | If the ML Algorithm may not Run Successfully | We cannot do further operations | we will do further  operations | High | High |
| 05 | Interact with Chatbot | Test whether Interact with Chatbot Successfully done or not | If the Interact with Chatbot may not Run Successfully | We cannot do further operations | we will do further  operations | High | High |

**7.SCREENSHOTS**:

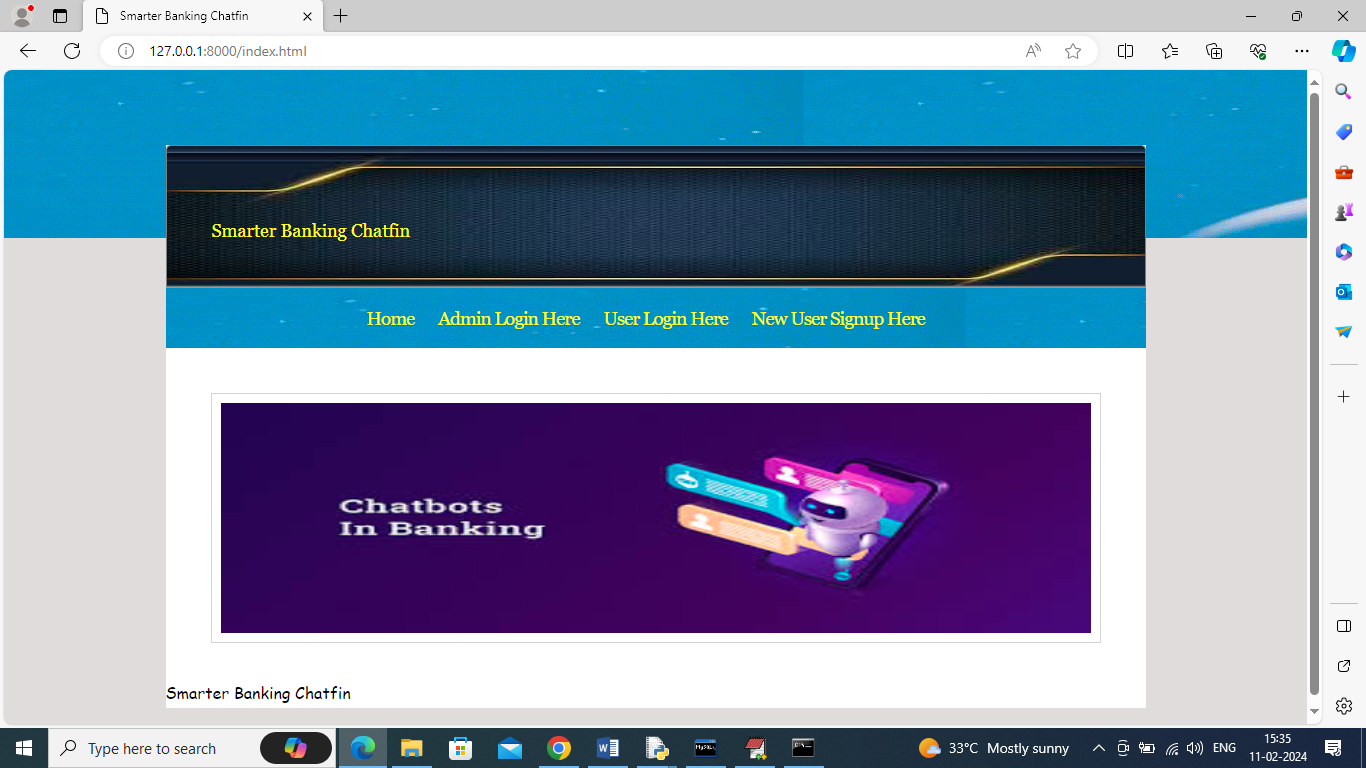
SCREEN SHOTS

To run project install MYSQL and then install python 3.7 and then copy content from DB.txt file and paste in MYSQL console to create database.

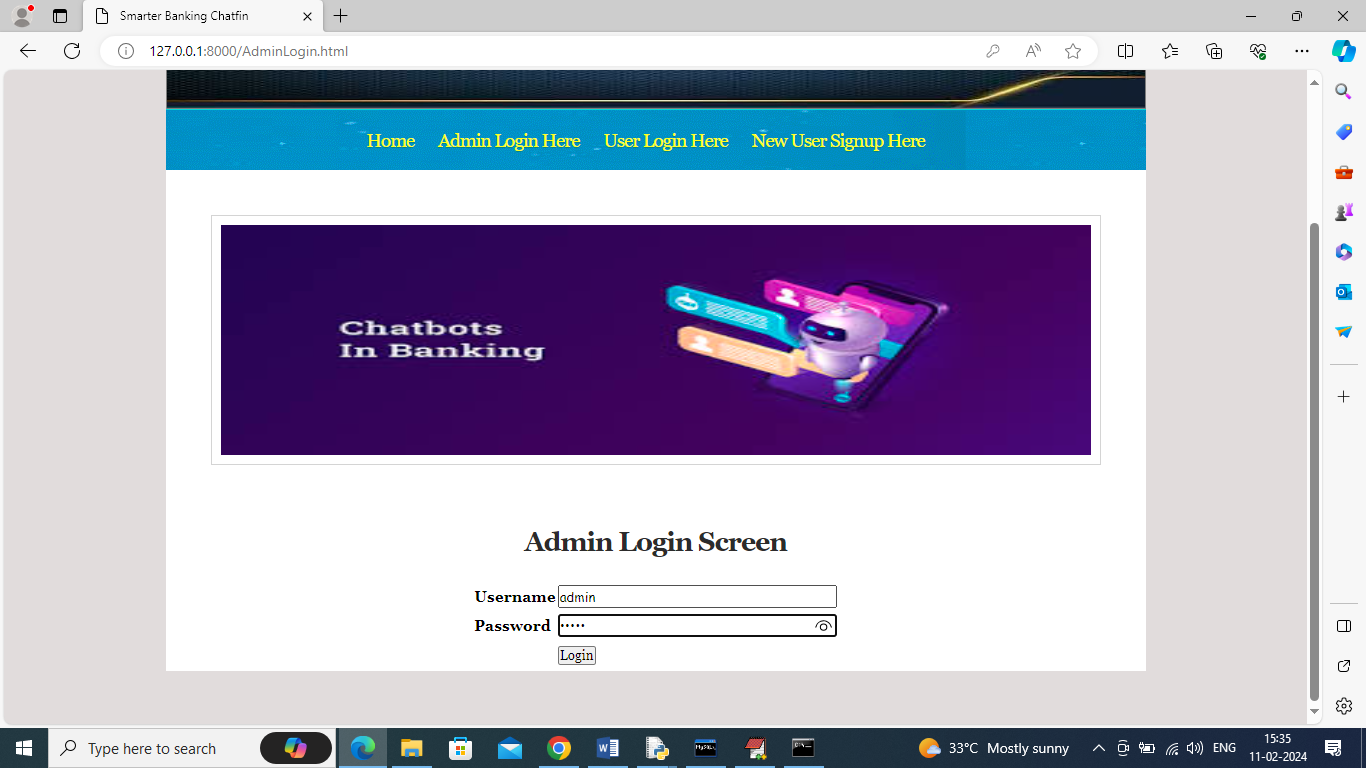
Double click on run.bat file to start python web server and then will get below page



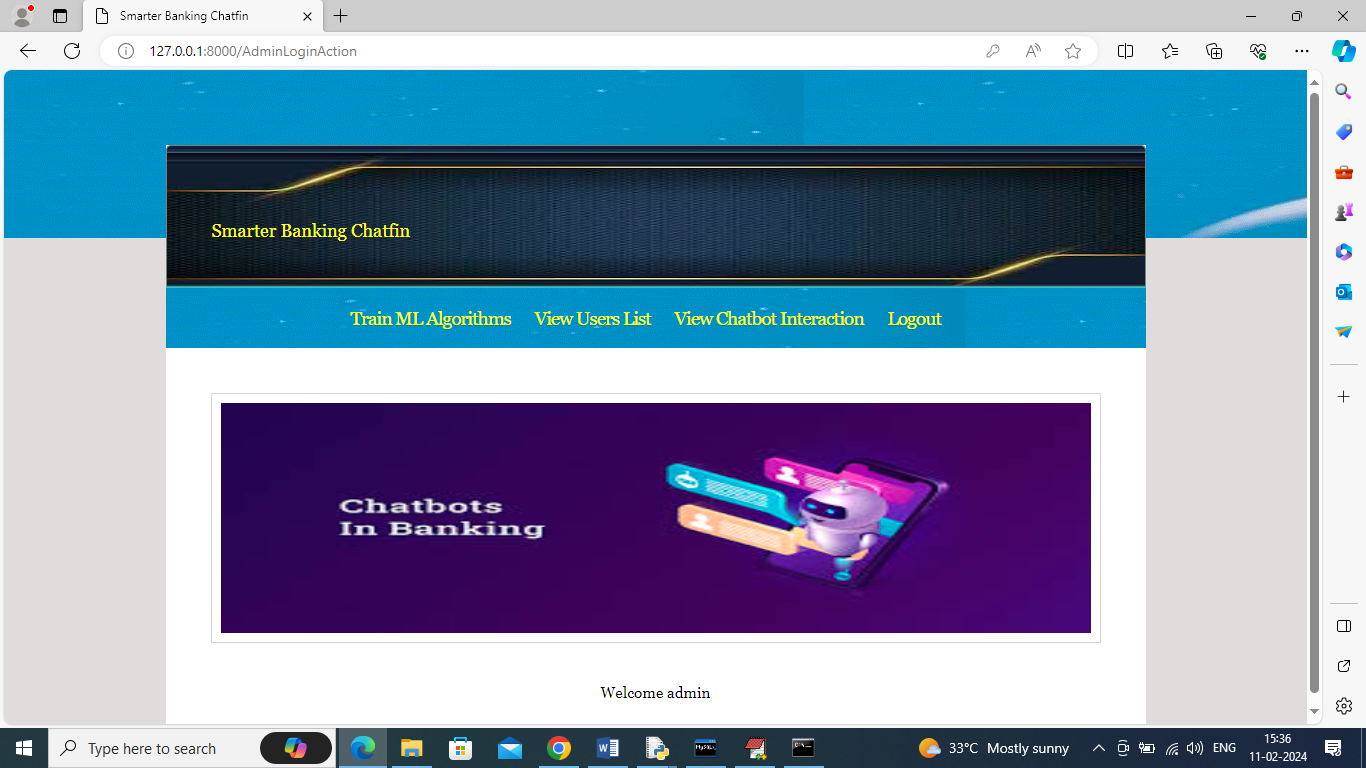
In above screen python server started and now open browser and then enter URL as <http://127.0.0.1:8000/index.html> and then press enter key to get below page



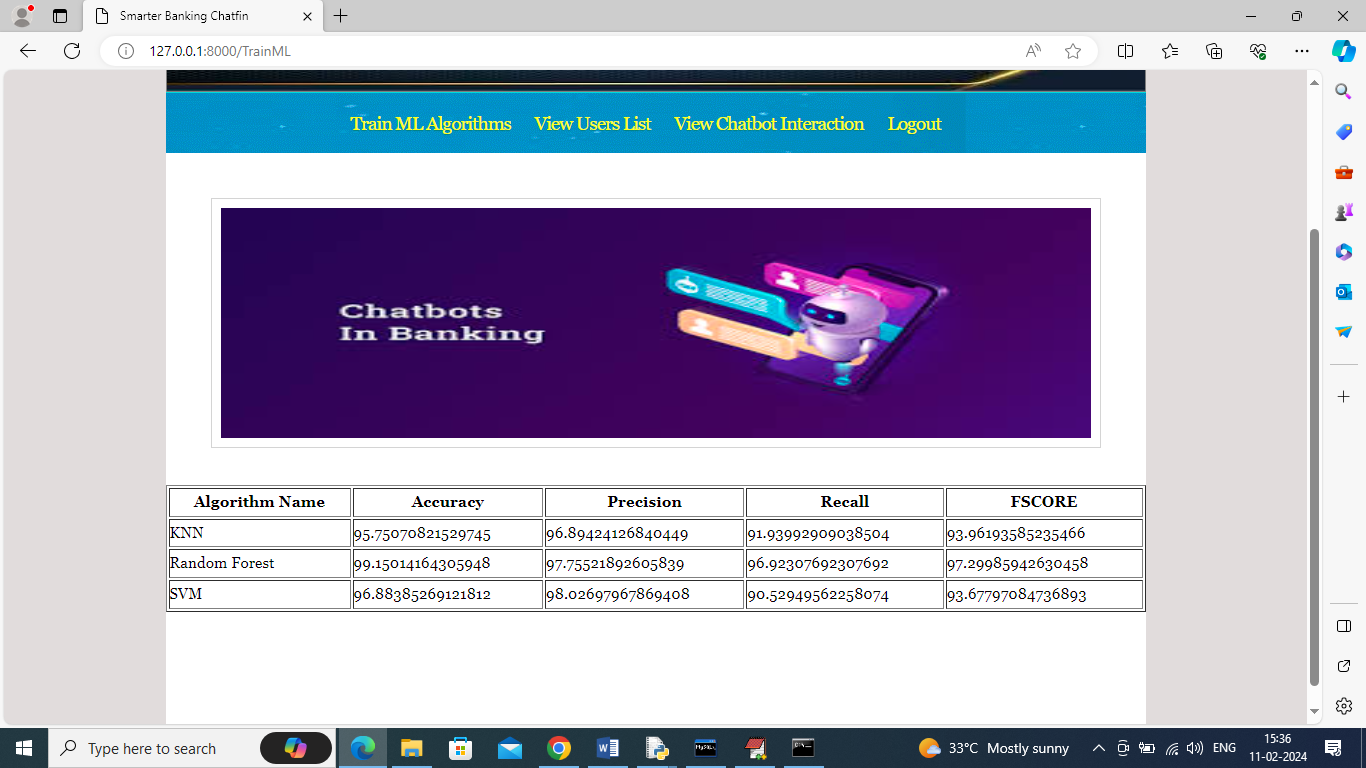
In above screen click on ‘Admin Login Here’ link to get below login page



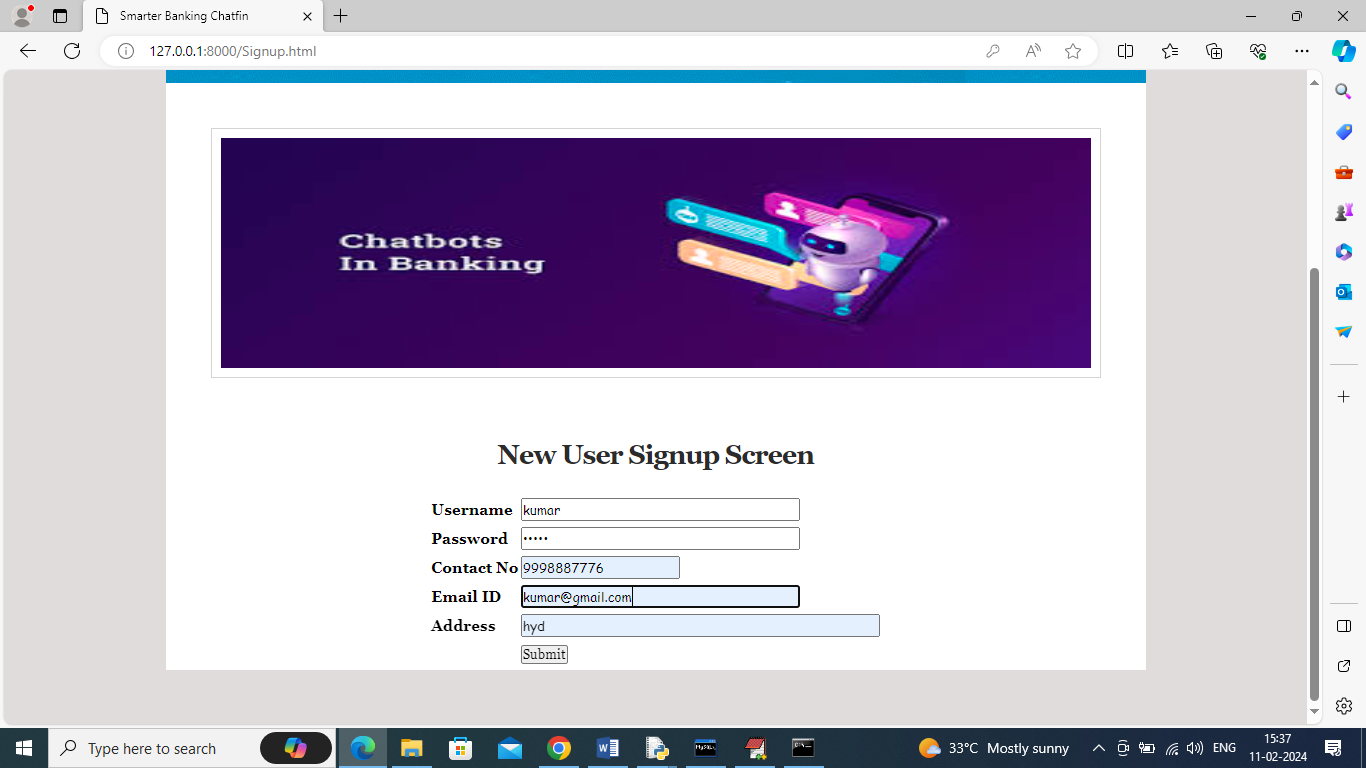
In above screen admin is login and after login will get below page



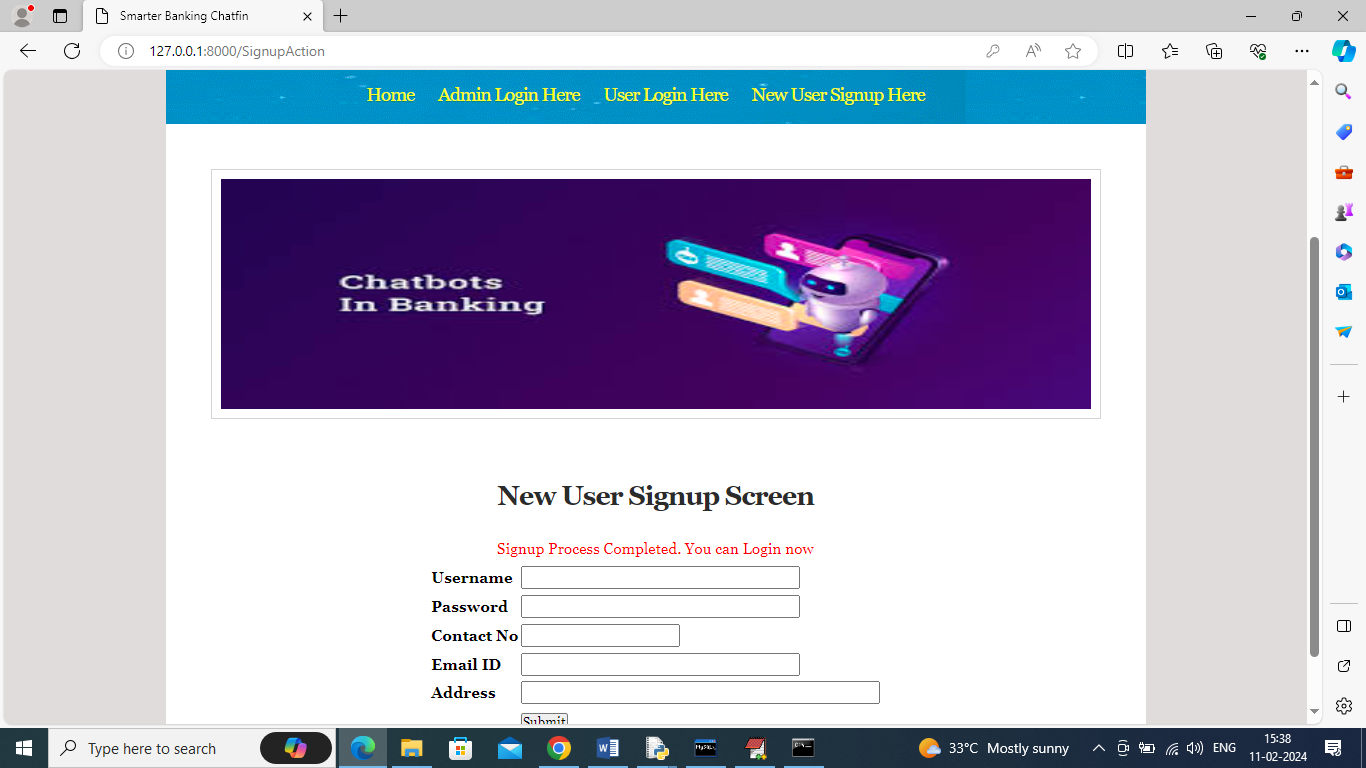
In above screen click on ‘Train ML Algorithms’ link to train algorithms and get below page



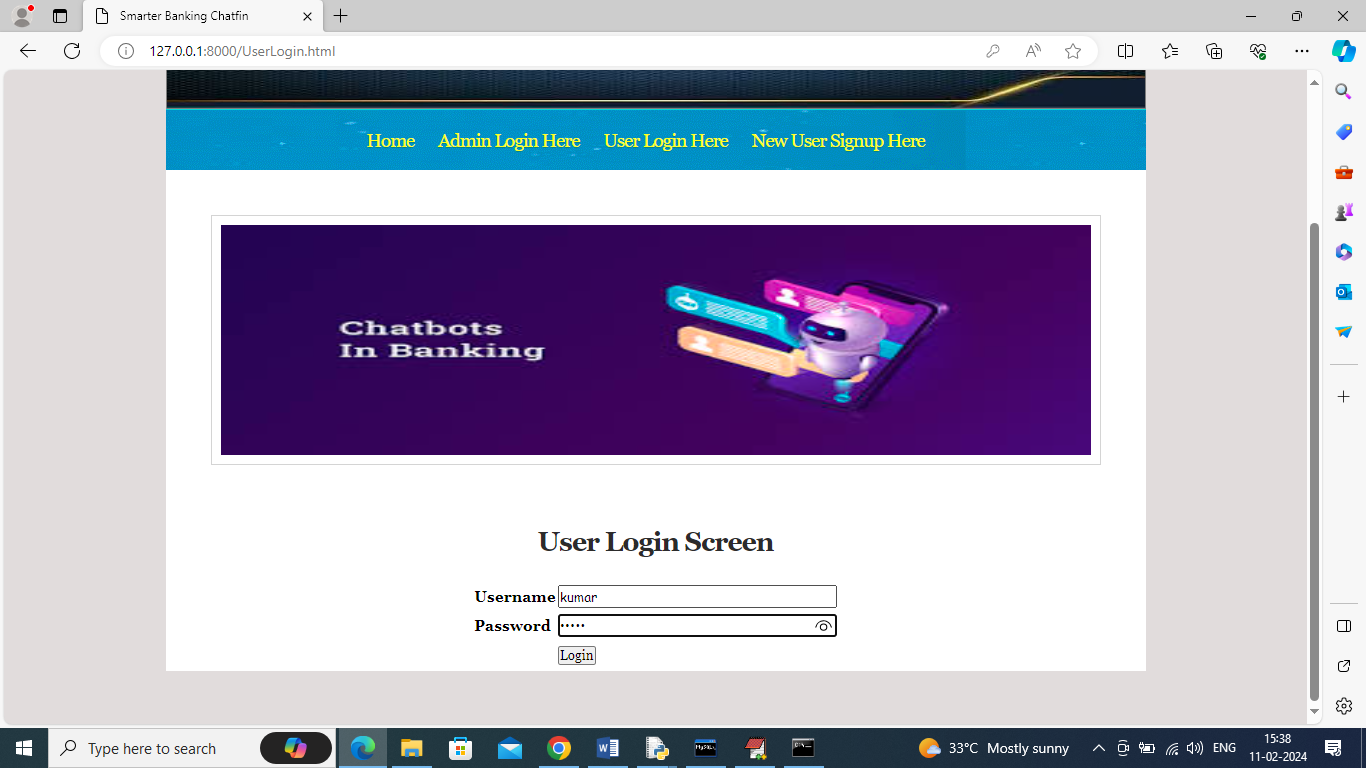
In above screen can see result of all ML algorithms and in all algorithms Random Forest got high accuracy and now ‘Logout’ and sign up new user



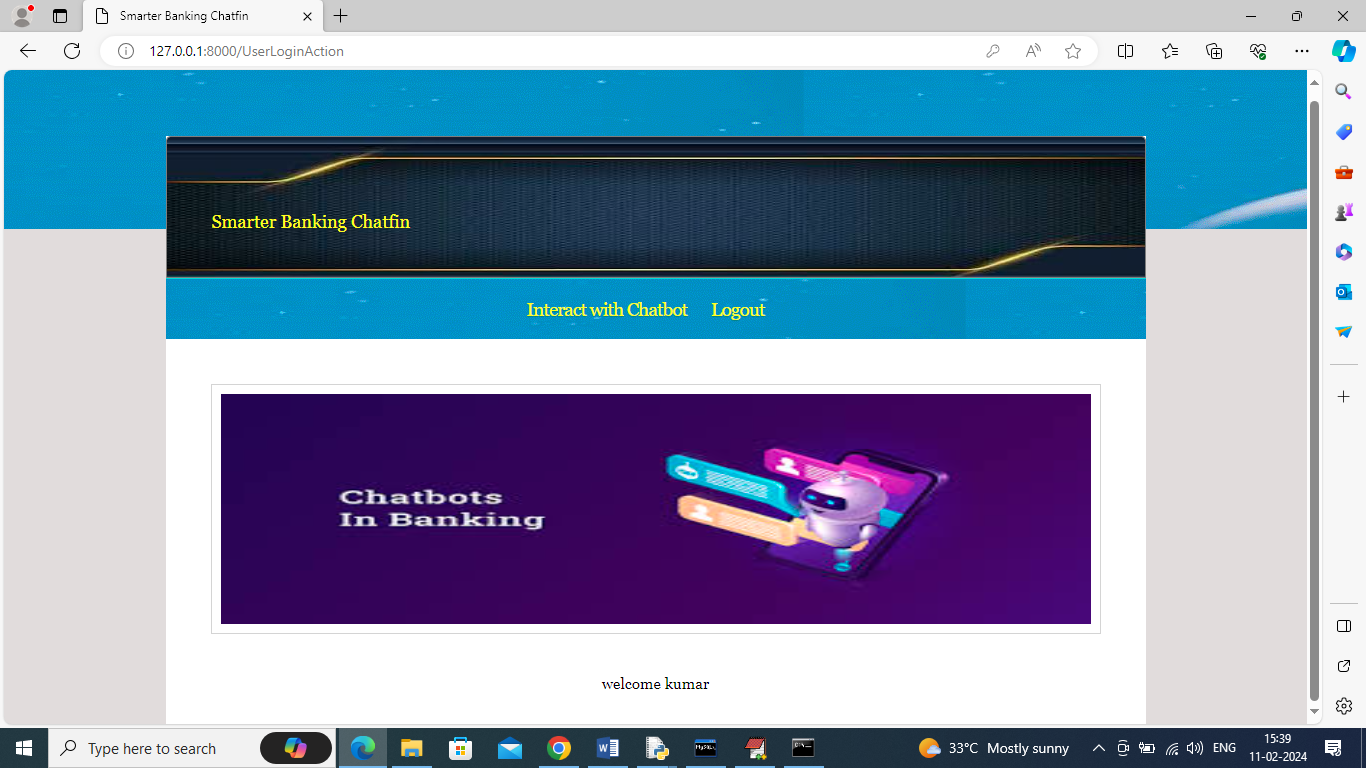
In above screen user is entering sign up details and then press button to get below page



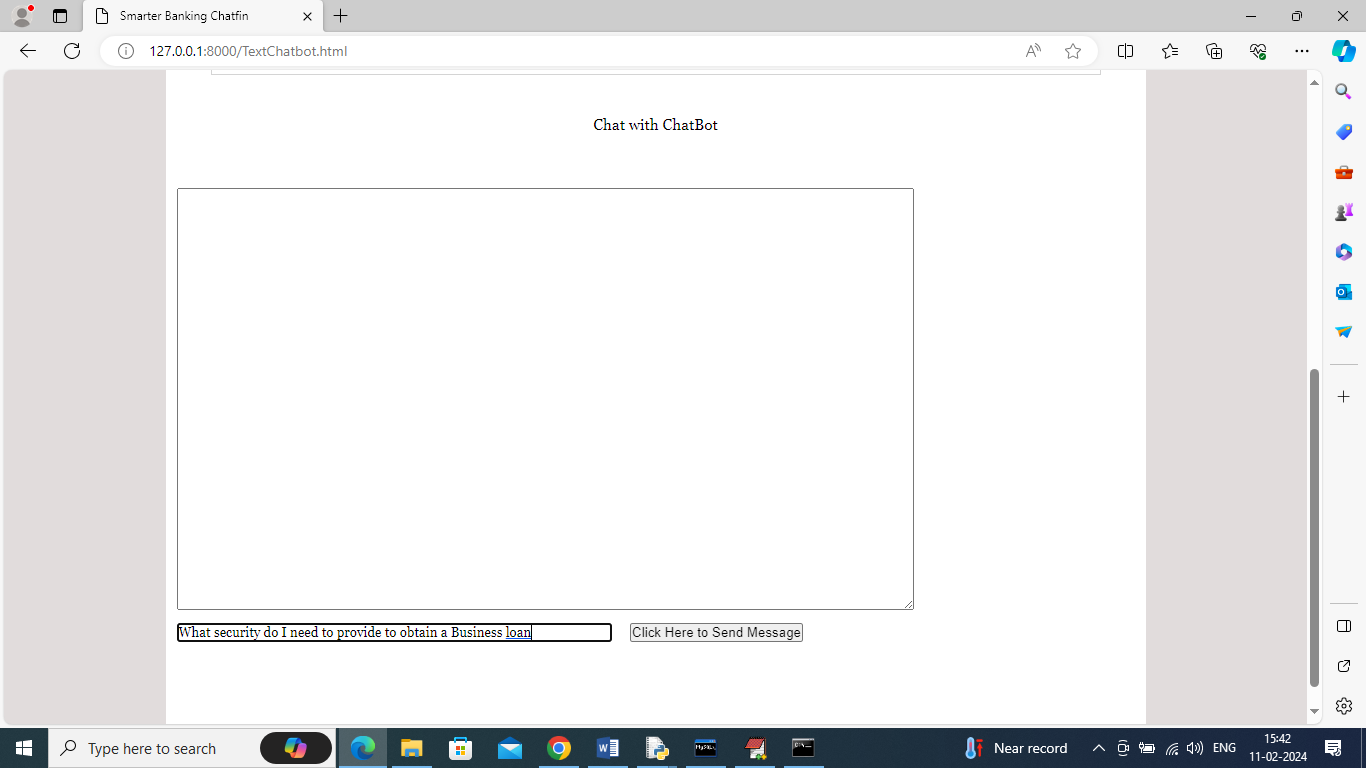
In above screen sign up task completed and now click on ‘User Login’ link to get below page



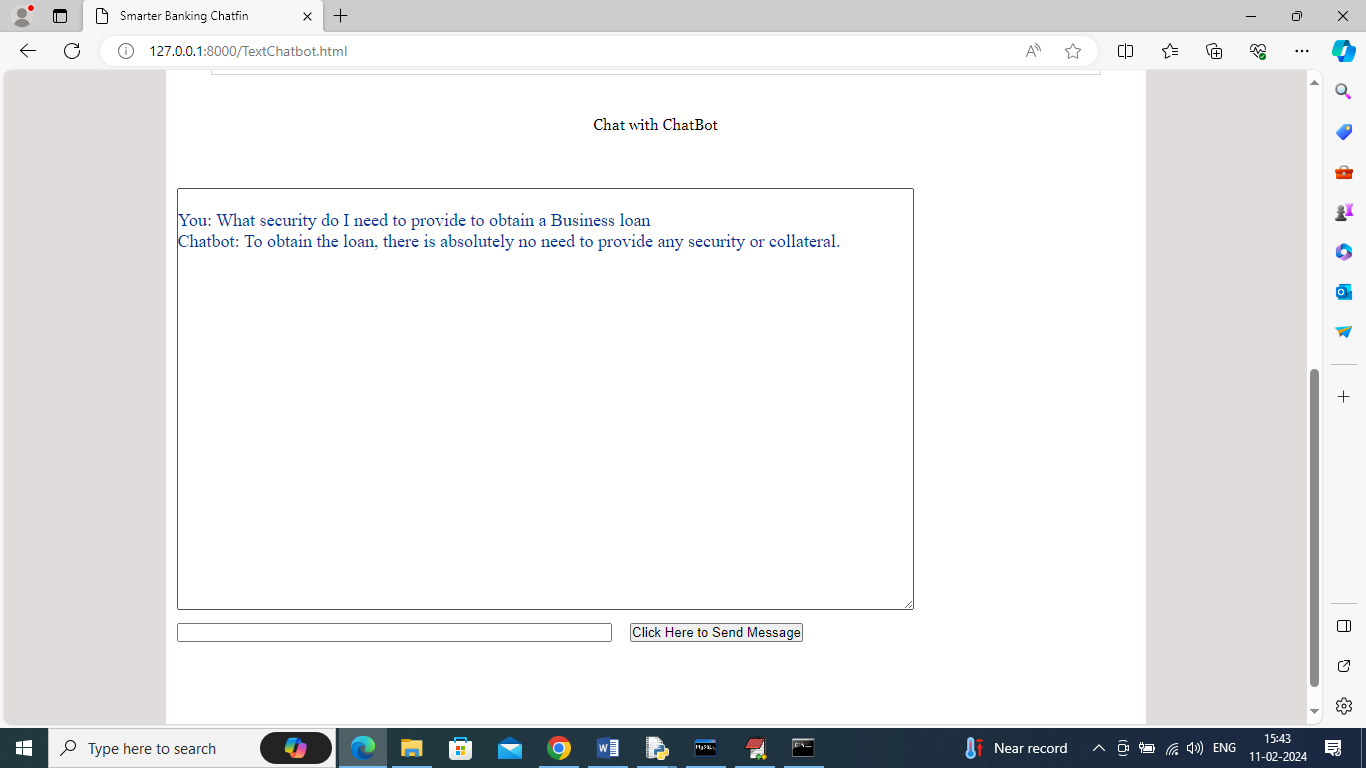
In above screen user is login and after login will get below page



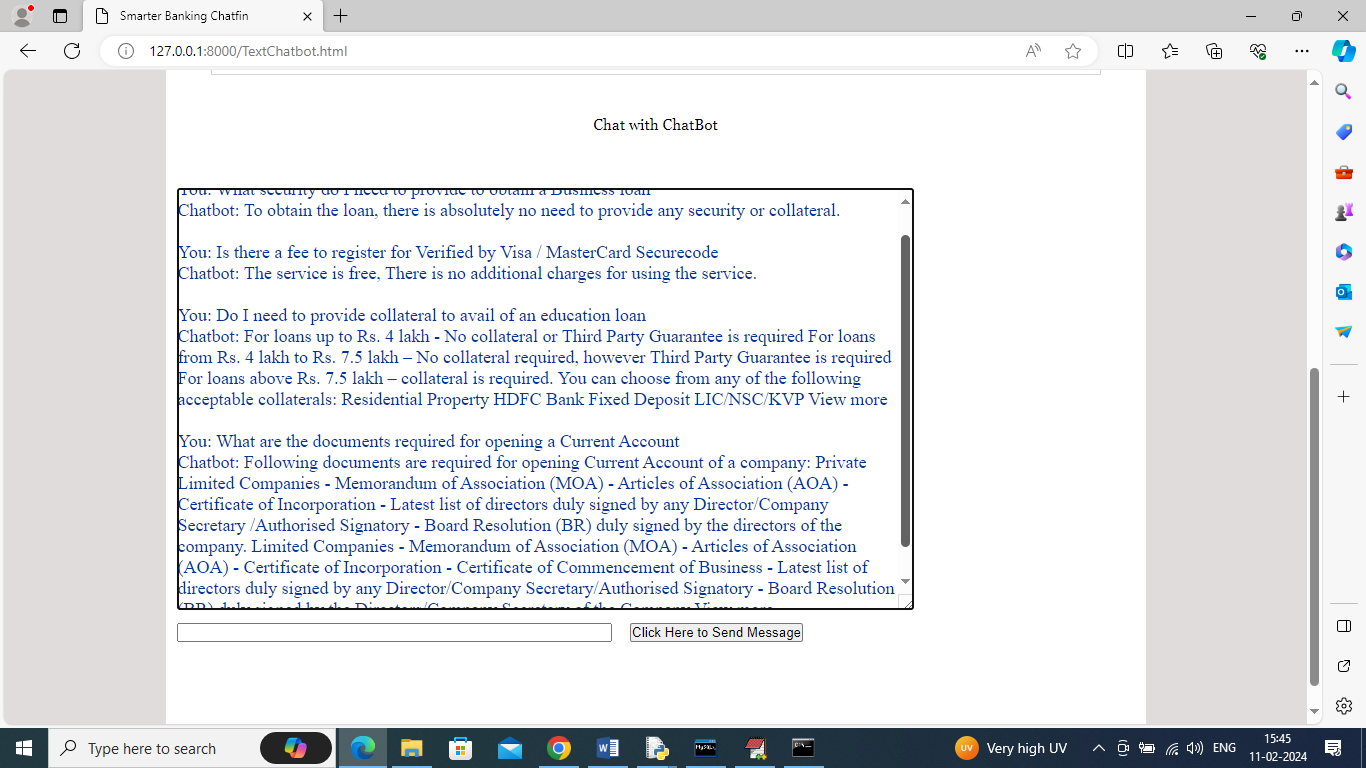
In above screen click on ‘Interact with Chatbot’ link to get below page



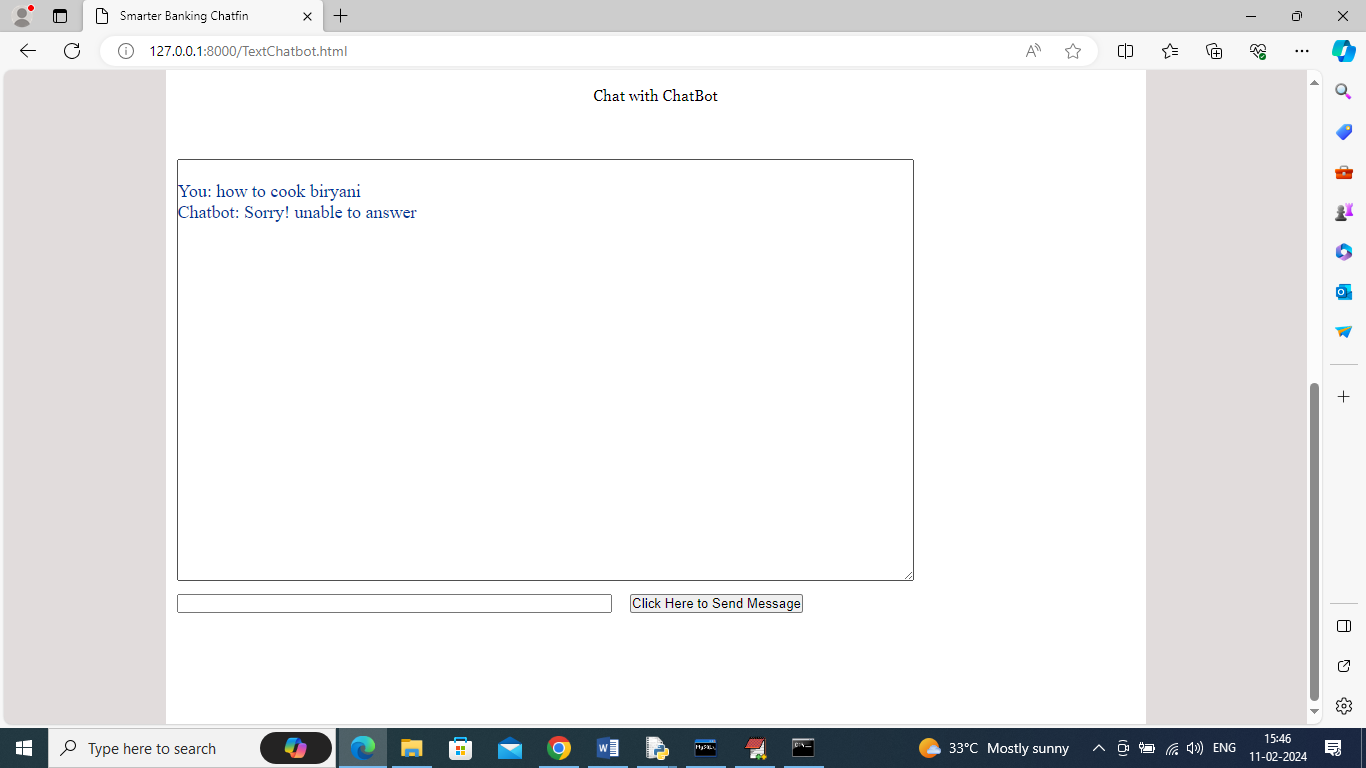
In above screen in text box I entered some text and then press button to get reply from Chatbot like below page



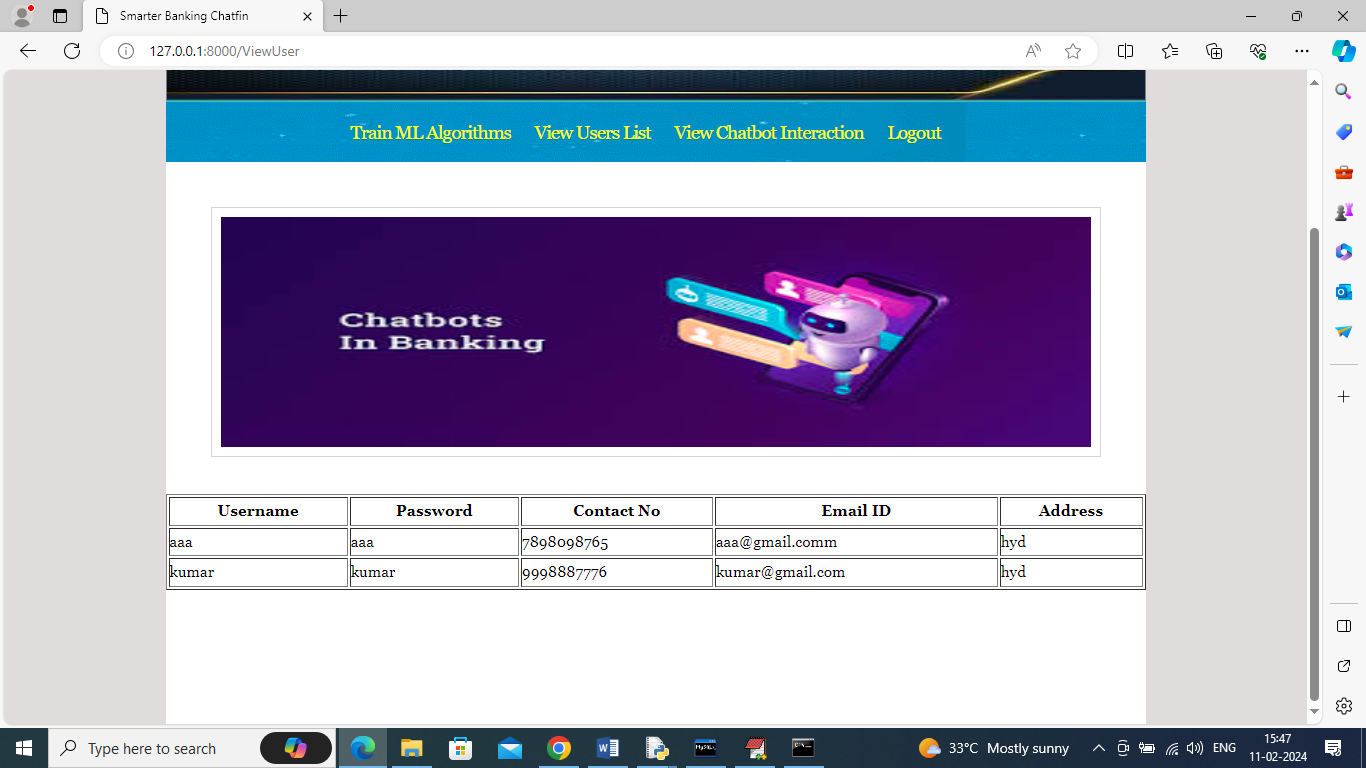
In above text area ‘You’ refers to user question and then can see Chatbot answer and similarly you can ask any question from dataset and get reply



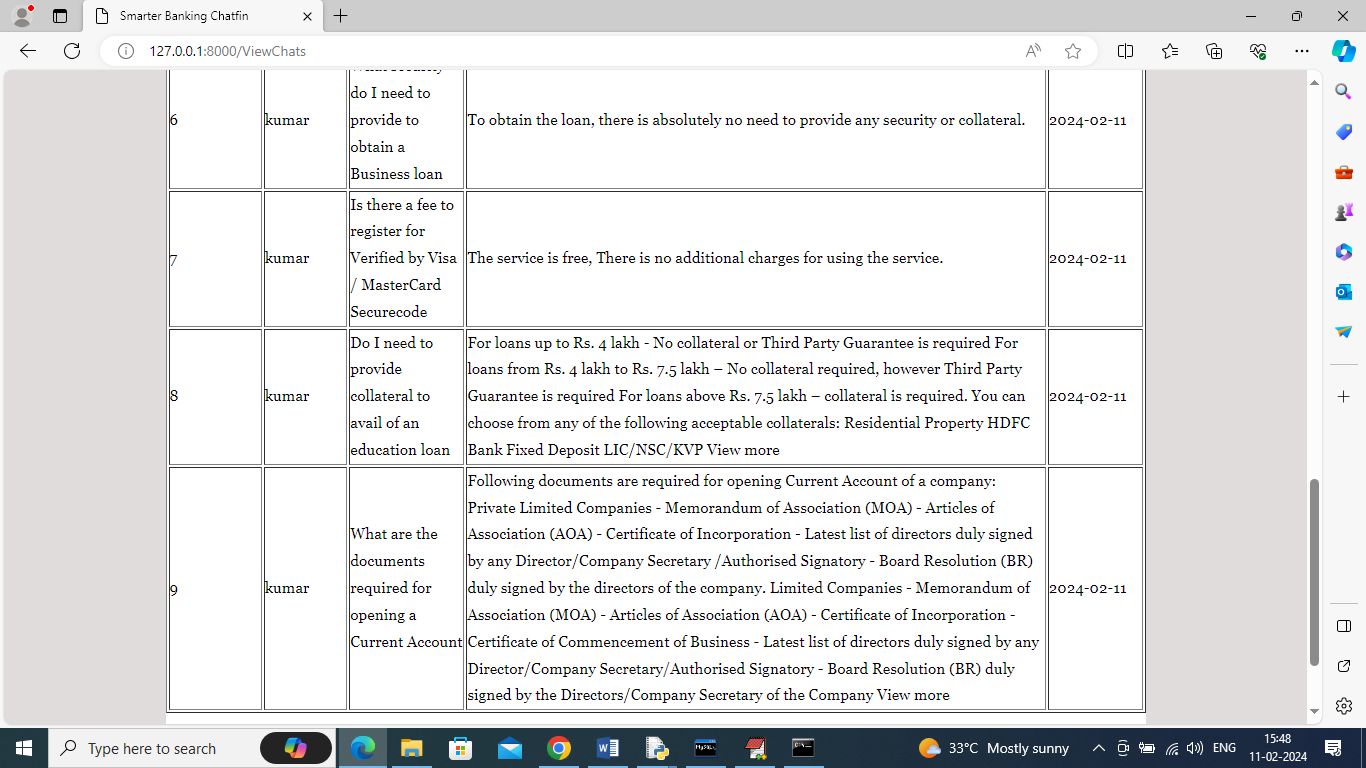
In above screen I entered some questions and then Chatbot replied for all questions



In above screen if we ask unrelated questions then Chatbot will reply with error message and now logout and login as ‘admin’ to view all interaction



In above screen admin can click on ‘View Users List’ link to get all registered users list and now click on ‘View Chatbot Interaction’ link to get all interactions list



In above screen admin can view which users ask what questions and what is the Chatbot reply along with date can be viewed by admin.

Similarly by following above screens you can run entire project.

**8. CONCLUSION:**

Implementation of this project would make the user experience of a customer on the website, more friendly and seamless. A chatbot can also act as an information-gathering tool that would help the organizations to inspect the needs of customers and then implement them.

Performance can be enhanced by using various procedures of machine learning. Chatbots can be integrated with voice commands to enhance the user experience. More banking services can be simplified by the use of a powerful chatbot. The quality of chatbots can be improved in the future by research in the field of banking. Security of chatbots will have to be improved as the security will be improved, people will be willing to share their information with the chatbot and also data collection will get easier. In the banking domain, contextual assistants need to cover more Indian languages thereby increasing the usability of chatbots.

**9. REFERENCES:**

[1] Jiao, Anran. (2020). An Intelligent Chatbot System Based on Entity Extraction Using RASA NLU and Neural Network. Journal of Physics: Conference Series. 1487. 012014. 10.1088/1742-6596/1487/1/012014.

[2] Fathima, Sasha & Student, Suhel & Shukla, Vinod & Vyas, Dr Sonali & Mishra, Ved P. (2020). Conversation to Automation in Banking Through Chatbot Using Artificial Machine Intelligence Language. 10.1109/ICRITO48877.2020.9197825.

[3] Singh, Netra & Singh, Devender. (2019). Chatbots and Virtual Assistant in Indian Banks. Industrija. 47. 75-101. 10.5937/industrija47-24578.

[4] Petr Lorenc, “Joint model for intent and entity recognition” in arXiv:2109.03221v1 [cs.CL] 7 Sep 2021

[5] The Rasa documentation. [Online]. Available: <https://rasa.com/docs/rasa/2.x>

[6] Django documentation. [Online]. Available: https://docs.djangoproject.com/en/4.0/