1. INTRODUCTION

INTRODUCTION

- In recent years, the widespread use of social media platforms like Twitter has led to a notable increase in the presence of social bots. These automated accounts can perform a variety of tasks, ranging from benign activities such as content sharing and customer support to malicious actions like spreading misinformation, phishing, and spamming. The detection and mitigation of these malicious social bots are critical to maintaining the integrity of online social networks.
- This paper introduces an advanced method for detecting malicious social bots in the Twitter network. The proposed approach combines learning automata, an adaptive decision-making algorithm, with URL features extracted from tweets. URLs in tweets are commonly used to share information but can also be exploited by malicious bots to spread harmful content. By analyzing the patterns and behaviors associated with URLs, the system aims to accurately identify and flag potentially malicious bots.
- The study's key contributions include the integration of learning automata with URL feature analysis for bot detection, a comprehensive evaluation of the method's effectiveness using real-world Twitter data, and insights into the behaviors and characteristics of malicious bots based on URL usage patterns. This innovative approach aims to enhance existing social bot detection methodologies, providing a robust tool for protecting the Twitter network from malicious activities.
- The machine will execute an output following the logical statement. When the system grows complex, more rules need to be written. It can quickly become unsustainable to maintain.

2. LITERATURE SURVEY

LITERATURE SURVEY

1. Detecting malicious social bots based on clickstream sequences

AUTHORS: P. Shi, Z. Zhang, and K.-K.-R. Choo

With the significant increase in the volume, velocity, and variety of user data (e.g., user-generated data) in online social networks, there have been attempted to design new ways of collecting and analyzing such big data. For example, social bots have been used to perform automated analytical services and provide users with improved quality of service. However, malicious social bots have also been used to disseminate false information (e.g., fake news), and this can result in real-world consequences.

Therefore, detecting and removing malicious social bots in online social networks is crucial. The most existing detection methods of malicious social bots analyse the quantitative features of their behaviour. These features are easily imitated by social bots; thereby resulting in low accuracy of the analysis. A novel method of detecting malicious social bots, including both features selection based on the transition probability of clickstream sequences and semi-supervised clustering, is presented in this paper. This method not only analyses transition probability of user behaviour clickstreams but also considers the time feature of behaviour.

2. Adaptive deep Q-learning model for detecting social bots and influential users in online social networks

AUTHORS: G. Lingam, R. R. Rout, and D. V. L. N. Somayajulu

In an online social network (like Twitter), a botmaster (i.e., leader among a group of social bots) establishes a social relationship among legitimate participants to reduce the probability of social bot detection. Social bots generate fake tweets and spread malicious information by manipulating the public opinion. Therefore, the detection of social bots in an online social network is an important task. In this paper, we consider social attributes,

such as tweet-based attributes, user profile-based attributes and social graph-based attributes for detecting the social bots among legitimate participants.

We design a deep Q-network architecture by incorporating a Deep Q-Learning (DQL) model using the social attributes in the Twitter network for detection of social bots based on updating Q-value function (i.e., state-action value function). We consider each social attribute of a user as a state and the learning agent's movement from one state to another state is considered as an action.

3. Fluxing botnet command and control channels with URL shortening services

AUTHORS: S. Lee and J. Kim

URL shortening services (USSes), which provide short aliases to registered long URLs, have become popular owing to Twitter. Despite their popularity, researchers do not carefully consider their security problems. In this paper, we explore botnet models based on USSes to prepare for new security threats before they evolve. Specifically, we consider using USSes for alias flux to hide botnet command and control (C&C) channels. In alias flux, a botmaster obfuscates the IP addresses of his C&C servers, encodes them as URLs, and then registers them to USSes with custom aliases generated by an alias generation algorithm

4. A neural network-based ensemble approach for spam detection in Twitter

AUTHORS: S. Madisetty and M. S. Desarkar

As the social networking sites get more popular, spammers target these sites to spread spam posts. Twitter is one of the most popular online social networking sites where users communicate and interact on various topics. Most of the current spam filtering methods in Twitter focus on detecting the spammers and blocking them. However, spammers can create a new account and start posting new spam tweets again. So there is a

need for robust spam detection techniques to detect the spam at tweet level. These types of techniques can prevent the spam in real time. To detect the spam at tweet level, often features are defined, and appropriate machine learning algorithms are applied in the literature. Recently, deep learning methods are showing fruitful results on several natural language processing tasks. We want to use the potential benefits of these two types of methods for our problem. The feature-based model uses content-based, user-based, and n-gram features. Our approach combines both deep learning and traditional feature-based models using a multilayer neural network which acts as a meta-classifier. We evaluate our method on two data sets, one data set is balanced, and another one is imbalanced. The experimental results show that our proposed method outperforms the existing methods.

3.INPUT AND OUTPUT DESIGN

INPUT DESIGN AND OUTPUT DESIGN

INPUT DESIGN:

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

OBJECTIVES:

1. Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

3. When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be

in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow.

OUTPUT DESIGN:

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

2. Select methods for presenting information.

3. Create document, report, or other formats that contain information produced by the system.

The output form of an information system should accomplish one or more of the following objectives.

- Convey information about past activities, current status or projections of the
- Future.
- Signal important events, opportunities, problems, or warnings.
- Trigger an action.
- Confirm an action.

4.SYSTEM REQUIREMENTS

SYSTEM REQUIREMENTS

HARDWARE REQUIREMENTS:

- System : Pentium IV 2.4 GHz.
- Hard Disk : 40 GB
- Floppy Drive : 1.44 Mb.
- Monitor : 15 VGA Colour.
- Mouse : Logitech.
- Ram : 512 Mb.

SOFTWARE REQUIREMENTS:

- Operating system : Windows 7.
- Coding Language : Python
- Database : MYSQL

5.SYSTEM ANALYSIS

SYSTEM ANALYSIS

EXISTING SYSTEM:

- The existing malicious URL detection approaches are based on DNS information and lexical properties of URLs. The malicious social bots use URL redirections in order to avoid detection.
- Besel et al. analyzed social botnet attack on Twitter. The authors have presented that social bots use URL shortening services and URL redirection in order to redirect users to malicious web pages.
- Echeverria and Zhou presented methods to detect, retrieve, and analyze botnet over thousands of users to observe the social behavior of bots.
- Dorri et.al., proposed a social bot hunter model has been presented based on the user behavioral features, such as follower ratio, the number of URLs, and reputation score.

DISADVANTAGES OF EXISTING SYSTEM:

- The malicious social bots can manipulate profile features, such as hashtag ratio, follower ratio, URL ratio, and the number of retweets. The malicious social bots can also manipulate tweet-content features, such as sentimental words, emoticons, and most frequent words used in the tweets, by manipulating the content of each tweet. The social relationship-based features are highly robust because the malicious social bots cannot easily manipulate the social interactions of users in the Twitter network.
- The existing approaches rely on statistical features instead of analyzing the social behavior of users. Moreover, these approaches are not highly robust in detecting the temporal data patterns with noisy data (i.e., where the data is biased with untrustworthy or fake information) because the behavior of malicious bots changes over time in order to avoid detection.

PROPOSED SYSTEM:

- In the proposed system, the malicious behavior of participants is analyzed by considering features extracted from the posted URLs (in the tweets), such as URL redirection, frequency of shared URLs, and spam content in URL, to distinguish between legitimate and malicious tweets. To protect against the malicious social bot attacks, algorithm integrates a trust computational model with a set of URL-based features for the detection of malicious social bots.
- In the proposed system, we Analyze the malicious behavior of a participant by considering URL-based features, such as URL redirection, the relative position of URL, frequency of shared URLs, and spam content in URL.
- In the proposed system, we Evaluate the trustworthiness of tweets (posted by each participant) by using the Bayesian learning and Dempster–Shafer theory (DST).
- Also we Design the system by integrating a trust model with a set of URL-based features.

ADVANTAGES OF PROPOSED SYSTEM:

- The proposed system helps to detect malicious social bots accurately.
- The experimental results illustrate that our proposed system gives better performance compared with conventional machine learning algorithms in terms of precision.
- The precision value obtained for The Fake Project data set is better than the Social Honeypot data set because++++ the Social Honeypot data set contains noisy and untrustworthy information in its user content feature-s than The Fake Project data set.
- The proposed system achieves the highest precision level. This is due to the fact that the proposed system executes for a finite set of learning actions to update the action probability value and achieves the advantages of incremental learning. Hence, the LA model with a trust component identifies the malicious tweets that are posted by malicious social bots.

6.SYSTEM STUDY

SYSTEM STUDY

FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

- ECONOMICAL FEASIBILITY
- TECHNICAL FEASIBILITY
- SOCIAL FEASIBILITY

• ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

- ## TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

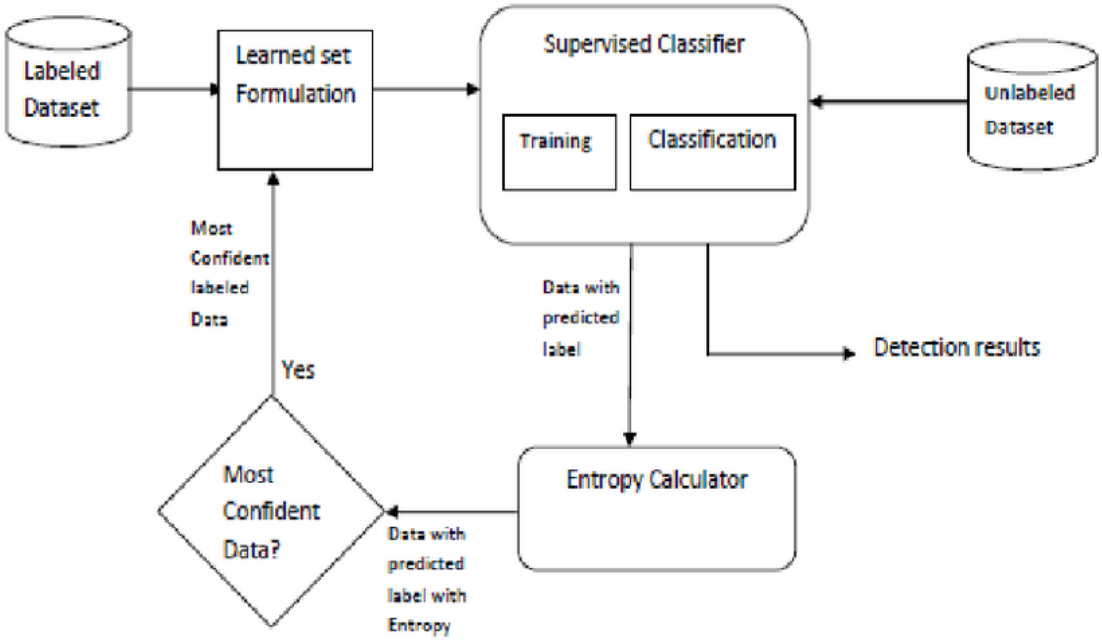
- ## SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

7.SYSTEM DESIGN

SYSTEM DESIGN

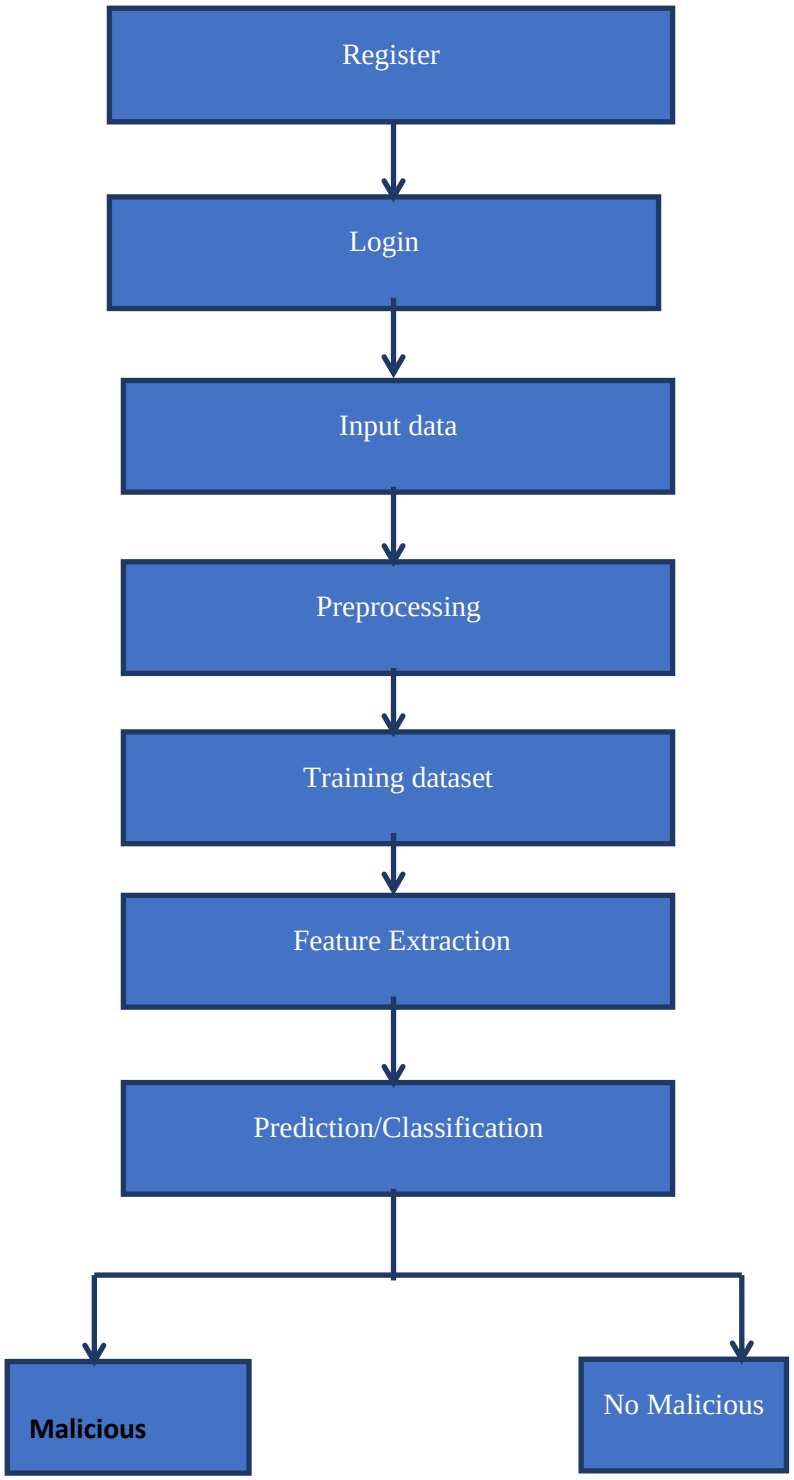
SYSTEM ARCHITECTURE:



DATA FLOW DIAGRAM:

1. The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
2. The data flow diagram (DFD) is one of the most important modelling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.
3. DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.

4. DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.



UML DIAGRAMS

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

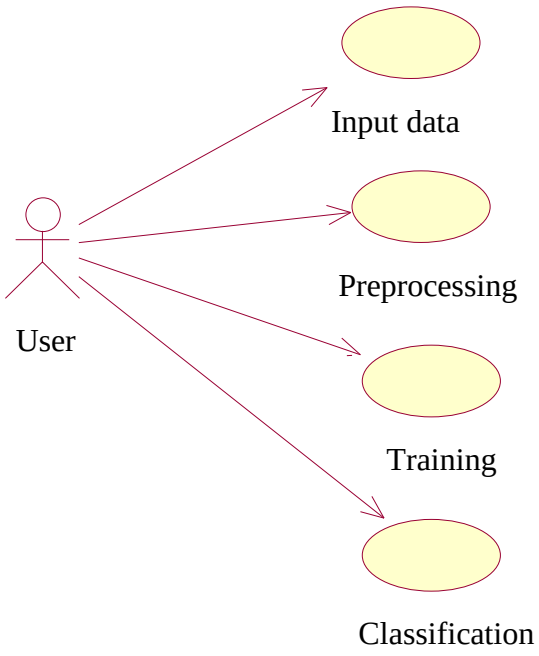
GOALS:

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

USE CASE DIAGRAM:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.



8.IMPLEMENTATION

IMPLEMENTATION

MODULES:

- Data Collection
- Dataset
- Data Preparation
- Model Selection
- Analyze and Prediction
- Accuracy on test set
- Saving the Trained Model
- Database connecting using Mysql

MODULES DESCSRIPTION:

Data Collection:

This is the first real step towards the real development of a machine learning model, collecting data. This is a critical step that will cascade in how good the model will be, the more and better data that we get, the better our model will perform.

There are several techniques to collect the data, like web scraping, manual interventions and etc. Detection of Malicious Social Bots taken from kaggle and some other source.

Dataset:

The dataset consists of 969812 individual data. There are 3 columns in the dataset, which are described below

1. Id: Unique id
2. Labels : Labels
   Malicious
   No Malicious
3. URL: url comment

Data Preparation:

We will transform the data. by getting rid of missing data and removing some columns. First we will create a list of column names that we want to keep or retain.

Next we drop or remove all columns except for the columns that we want to retain.

Finally we drop or remove the rows that have missing values from the data set.

Steps to follow:

Removing extra symbols

Removing punctuations

1. Removing the Stopwords
2. Stemming
3. Tokenization
4. Feature extractions
5. CountVectorizer
6. Counter vectorizer with TF-IDF transformer

Model Selection:

We used Logistic regression algorithm: Logistic regression is a classification algorithm, used when the value of the target variable is *categorical* in nature. Logistic regression is most commonly used when the data in question has binary output, so when it belongs to one class or another, or is either a 0 or 1.Remember that classification tasks have discrete categories, unlike regressions tasks.

Here, by the idea of using a regression model to solve the classification problem, we rationally raise a question of whether we can draw a hypothesis function to fit to the binary dataset. For simplification, we only concern the binary classification problem.

The answer is that you will have to use a type of function, different from linear functions, called a logistic function, or a **sigmoid function.**

(Note: Here's something important to remember: although the algorithm is called "Logistic Regression", it is, in fact, a classification algorithm, not a regression algorithm. This can be confusing at first, but just try to remember it.

The Sigmoid Function

The sigmoid function/logistic function is a function that resembles an "S" shaped curve when plotted on a graph. It takes values between 0 and 1 and "squishes" them towards the margins at the top and bottom, labelling them as 0 or 1.

The equation for the Sigmoid function is this:

What is the variable e in this instance? The e represents the exponential function or exponential constant, and it has a value of approximately 2.71828.

Let's see how the sigmoid function represent the given dataset.

This gives a value y that is extremely close to 0 if xis a large negative value and close to 1 if x is a large positive value. After the input value has been squeezed towards 0 or 1, the input can be run through a typical linear function, but the inputs can now be put into distinct categories.

Analyse and Prediction:

In the actual dataset, we chose only 2 features :

1. URL: url comment
2. Labels : Labels

   Malicious

   No Malicious

Accuracy on test set:

We got a accuracy of 96.02% on test set.

Saving the Trained Model:

Once you're confident enough to take your trained and tested model into the production-ready environment, the first step is to save it into a .h5 or . pkl file using a library like `pickle` .

Make sure you have `pickle` installed in your environment.

Next, let's import the module and dump the model into `.pkl` file

Import Mysqldb:

So long as that works, do a quick control+d to exit the python instance. Next, we want to make a Python file that can connect to the database. Generally you will have a separate "connect" file, outside of any main files you may have. This is usually true across languages, and here's why. Initially, you may have just a simple __init__.py, or app.py, or whatever, and that file does all of your operations. What can happen in time, however, is that your website does other things. For example, with one of my websites, Sentdex.com, I perform a lot of analysis, store that analysis to a database, and I also operate a website for users to use. Generally, for tasks, you will use what is called a "cron." A cron is a scheduled task that runs when you program it to run. Generally this runs another file, almost certain to not be your website's file. So then, to connect to a database, you'd have to write the database connecting code again in the file being run by your cron.

As time goes on, these sorts of needs stack up where you have some files modifying the database, but you still want the website to be able to access it, and maybe modify it too. Then, consider what might happen if you change your database password. You'd then need to go to every single file that connects to the database and change that too. So, usually, you will find the smartest thing to do is to just create one file, which houses the connection code.

Import the module.

Create a connection function to run our code. Here we specify where we're connecting to, the user, the user's password, and then the database that we want to connect to.

As a note, we use "localhost" as our host. This just means we'll use the same server that this code is running on. You can connect to databases remotely as well, which can be pretty neat. To do that, you would connect to a host by their IP, or their domain. To connect to a database remotely, you will need to first allow it from the remote database that will be accessed/modified.

Next, let's go ahead and edit our __init__.py file, adding a register function. For now we'll keep it simple, mostly just to test our connection functionality.

We allow for GET and POST, but aren't handling it just yet.

We're going to just try to run the imported connection function, which returns c and conn (cursor and connection objects).

If the connection is successful, we just have the page say okay, otherwise it will output the error.

9.SOURCE CODE

Login.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta content="width=device-width, initial-scale=1.0" name="viewport">
  <title>Malicious Social Bots </title>
  <meta content="" name="description">
  <meta content="" name="keywords">
                                                                                    <link
href="https://fonts.googleapis.com/css?family=Open+Sans:300,300i,400,400i,600,600i,7
00,700i|Roboto:300,300i,400,400i,500,500i,600,600i,700,700i|Poppins:300,300i,400,400
i,500,500i,600,600i,700,700i" rel="stylesheet">

  <!-- Vendor CSS Files -->
  <link href="../static/vendor/animate.css/animate.min.css" rel="stylesheet">
  <link href="../static/vendor/bootstrap/css/bootstrap.min.css" rel="stylesheet">
  <link href="../static/vendor/bootstrap-icons/bootstrap-icons.css" rel="stylesheet">
  <link href="../static/vendor/boxicons/css/boxicons.min.css" rel="stylesheet">
  <link href="../static/vendor/glightbox/css/glightbox.min.css" rel="stylesheet">
  <link href="../static/vendor/swiper/swiper-bundle.min.css" rel="stylesheet">

  <!-- Template Main CSS File -->
  <link href="../static/css/style.css" rel="stylesheet">

</head>
<body>
  <header id="header" class="fixed-top d-flex align-items-center">
    <div class="container d-flex align-items-center justify-content-between">
      <h1 class="logo"><a href="index.html">Malicious</a></h1>
      <!-- Uncomment below if you prefer to use an image logo -->
         <!-- <a href="index.html" class="logo"><img src="assets/img/logo.png" alt=""
class="img-fluid"></a>-->
      <nav id="navbar" class="navbar">
        <ul>
         <li><a class="nav-link scrollto " href="{{ url_for('login')}}">Login</a></li>
     <li><a class="nav-link scrollto " href="{{ url_for('register')}}">Register</a></li>
        </ul>
        <i class="bi bi-list mobile-nav-toggle"></i>
      </nav><!-- .navbar -->
    </div>
  </header><!-- End Header -->
```

```
 <section id="hero">
   <div class="hero-container">
     <div id="heroCarousel" data-bs-interval="4000" class="carousel slide carousel-fade"
data-bs-ride="carousel">

      <ol class="carousel-indicators" id="hero-carousel-indicators"></ol>
      <div class="carousel-inner" role="listbox">
       <div class="carousel-item active" style="background: url(../static/img/mal6.jpg);">
         <div class="carousel-container">
          <div class="carousel-content">
           <h2 class="animate__animated animate__fadeInDown">Detection of Malicious
Social Bots Using machine Learning</h2>
            <div>
            </div>
           </div>
         </div>
       </div>
      </div>
      </div>
    </div>
   </div>
 </section
 <main id="main">
   <section id="services" class="services">
    <div class="container">
      <div class="section-title">
       <h2></h2>
                    <head>
</head>
  <body id="page-top">
   <!-- Portfolio Section -->
 <section class="page-section portfolio" id="portfolio">
   <div class="container">
    <br>
   <br>
    <!-- Portfolio Section Heading -->
    <div class="login-title text-center">
   <div class="row">
               </div>
        </div>
    <!-- Icon Divider -->
    <div class="divider-custom">
```

```
      <div class="divider-custom-line"></div>
      <div class="divider-custom-icon">
        <i class="fas fa-star"></i>
      </div>
      <div class="divider-custom-line"></div>
    </div>

    <!-- Portfolio Grid Items -->
  <form action="{{ url_for('login') }}" method="post">
    <div class="row">
  <h2>Login</h2>
      <!-- Portfolio Item 1 -->
      <div class="col-md-6 col-lg-4" style="margin-left:380px">
   <div class="control-group">
          <!-- Username -->


          <div class="controls">
         <label style="color:black"><b>Username :</b></label>
            <input type="text"   name="username" placeholder="Username"  >
        </div>
        </div>
      </br>
        <div class="control-group">
          <!-- Password-->


          <div class="controls">
        <label style="color:black"><b>Password :</b></label>
            <input type="password" name="password" placeholder="Password"   required>

        </div>
        </div>
       <div class="col-md-6 col-lg-4" style="margin-left:-120px">
        <div class="control-group">
          <!-- Button -->
        <br>
          <div class="controls">
         <div class="msg">{{ msg }}</div>
            <input type="submit" class="btn btn-success" value="Login" style="margin-left:
240px" >

        </div>
```

```
        </div>
       </div>
  </div>
  </div>
  </form>
  </div>
  </div>
  </section>

 {% with messages = get_flashed_messages() %}
  {% if messages %}
    <script>
      var messages = {{ messages | safe }};
      for (var i=0; i<messages.length; i++) {
        alert(messages[i]);
      }
    </script>
  {% endif %}
{% endwith %}

    </body>
       </div>
       </div>
    </section>
  <footer id="footer">
    <div class="footer-top">
      <div class="container">
        <div class="row">
          <div class="col-lg-3 col-md-6">
            <div class="footer-info">
            </div>
          </div>
          <div class="col-lg-2 col-md-6 footer-links">
          </div>
          <div class="col-lg-3 col-md-6 footer-links">
          </div>
          <div class="col-lg-4 col-md-6 footer-newsletter"
          </div>
        </div>
      </div>
    </div>
    <div class="container">
      <div class="copyright">
```

```
    </div>
    <div class="credits">
   </div>
  </footer>

   <a href="#" class="back-to-top d-flex align-items-center justify-content-center"><i class="bi bi-arrow-up-short"></i></a>

  <!-- Vendor JS Files -->
  <script src="../static/vendor/bootstrap/js/bootstrap.bundle.min.js"></script>
  <script src="../static/vendor/glightbox/js/glightbox.min.js"></script>
  <script src="../static/vendor/isotope-layout/isotope.pkgd.min.js"></script>
  <script src="../static/vendor/php-email-form/validate.js"></script>
  <script src="../static/vendor/purecounter/purecounter.js"></script>
  <script src="../static/vendor/swiper/swiper-bundle.min.js"></script>

  <!-- Template Main JS File -->
  <script src="../static/js/main.js"></script>
</body>
</html>
```

Register.html

```
 <!DOCTYPE html>
<html lang="en">
 <head>
 <meta charset="utf-8">
<meta content="width=device-width, initial-scale=1.0" name="viewport">
 <title>Malicious Social Bots </title>
<meta content="" name="description">
 <meta content="" name="keywords">

<link
href="https://fonts.googleapis.com/css?family=Open+Sans:300,300i,400,400i,600,600i,700,700i|Roboto:300,300i,400,400i,500,500i,600,600i,700,700i|Poppins:300,300i,400,400i,500,500i,600,600i,700,700i" rel="stylesheet">
  <link href="../static/vendor/animate.css/animate.min.css" rel="stylesheet">
  <link href="../static/vendor/bootstrap/css/bootstrap.min.css" rel="stylesheet">
  <link href="../static/vendor/bootstrap-icons/bootstrap-icons.css" rel="stylesheet">
  <link href="../static/vendor/boxicons/css/boxicons.min.css" rel="stylesheet">
  <link href="../static/vendor/glightbox/css/glightbox.min.css" rel="stylesheet">
  <link href="../static/vendor/swiper/swiper-bundle.min.css" rel="stylesheet">
  <link href="../static/css/style.css" rel="stylesheet">
```

```
</head>
<body>
  <header id="header" class="fixed-top d-flex align-items-center">
    <div class="container d-flex align-items-center justify-content-between">
      <h1 class="logo"><a href="index.html">Malicious</a></h1>
        <!-- <a href="index.html" class="logo"><img src="assets/img/logo.png" alt="" class="img-fluid"></a>-->

      <nav id="navbar" class="navbar">
        <ul>
         <li><a class="nav-link scrollto " href="{{ url_for('login')}}">Login</a></li>
    <li><a class="nav-link scrollto " href="{{ url_for('register')}}">Register</a></li>
        </ul>
        <i class="bi bi-list mobile-nav-toggle"></i>
      </nav><!-- .navbar -->
    </div>
  </header>
  <section id="hero">
    <div class="hero-container">
      <div id="heroCarousel" data-bs-interval="4000" class="carousel slide carousel-fade" data-bs-ride="carousel">

        <ol class="carousel-indicators" id="hero-carousel-indicators"></ol>

        <div class="carousel-inner" role="listbox">

          <!-- Slide 1 -->
          <div class="carousel-item active" style="background: url(../static/img/mal6.jpg);">
            <div class="carousel-container">
              <div class="carousel-content">
                <h2 class="animate__animated animate__fadeInDown">Detection of Malicious Social Bots Using machine Learning</h2>
                 <div>
                 </div>
              </div>
            </div>
          </div>
        </div>
      </div>
    </div>
  </section><!-- End Hero -->
  <main id="main">
    <section id="services" class="services">
```

```
    <div class="container">

      <div class="section-title">
        <h2></h2>
              <head>
</head>
  <body id="page-top">
 <!-- Portfolio Section -->
 <section class="page-section portfolio" id="portfolio">
  <div class="container">
    <br>
  <br>
    <!-- Portfolio Section Heading -->
    <div class="login-title text-center">
   <div class="row">
              </div>
        </div>
    <!-- Icon Divider -->
    <div class="divider-custom">
      <div class="divider-custom-line"></div>
      <div class="divider-custom-icon">
        <i class="fas fa-star"></i>
      </div>
      <div class="divider-custom-line"></div>
    </div>

    <!-- Portfolio Grid Items -->
    <div class="row">
   <h2>Register</h2>
<form action="{{ url_for('register') }}" method="post" >
      <!-- Portfolio Item 1 -->
      <div class="col-md-6 col-lg-4" style="margin-left:380px">
   <div class="control-group>
         <div class="controls">
        <label style="color:black"><b>Username :</b></label>
          <input type="text" name="username" placeholder="Username"  required>
        </div>
        </div>
      </br>
       <div class="control-group">
         <!-- Password-->
         <div class="controls">
```

```
    <label style="color:black"><b>Email ID :    </b></label>
          <input type="email" name="email" placeholder="Email"  required>
        </div>
        </div>
       </br>
        <div class="form-group">
          <!-- Password-->
         <label style="color:black"><b>Password :</b></label>
           <input type="password" name="password"   placeholder="Password" required>
        </div>
       <div class="col-md-6 col-lg-4" style="margin-left:-100px">
        <div class="control-group">
          <!-- Button -->
        <br>
          <div class="controls">
         <div class="msg">{{ msg }}</div>
           <input type="submit" class="btn btn-success" value="submit" style="margin-left:
240px"  >

        </div>
        </div>
       </div>
 </div>
 </div>
 </form>
 </div>
 </div>
 </section>
  </body>
      </div>
      </div>
  </section>
 <footer id="footer">
  <div class="footer-top">
   <div class="container">
    <div class="row">
     <div class="col-lg-3 col-md-6">
      <div class="footer-info">
      </div>
     </div>
     <div class="col-lg-2 col-md-6 footer-links">
     </div>
     <div class="col-lg-3 col-md-6 footer-links">
```

```
        </div>
        <div class="col-lg-4 col-md-6 footer-newsletter">
        </div>
       </div>
     </div>
   </div>
   <div class="container">
     <div class="copyright">

     </div>
     <div class="credits">
   </div>
  </footer>
   <a href="#" class="back-to-top d-flex align-items-center justify-content-center"><i class="bi bi-arrow-up-short"></i></a>

  <!-- Vendor JS Files -->
  <script src="../static/vendor/bootstrap/js/bootstrap.bundle.min.js"></script>
  <script src="../static/vendor/glightbox/js/glightbox.min.js"></script>
  <script src="../static/vendor/isotope-layout/isotope.pkgd.min.js"></script>
  <script src="../static/vendor/php-email-form/validate.js"></script>
  <script src="../static/vendor/purecounter/purecounter.js"></script>
  <script src="../static/vendor/swiper/swiper-bundle.min.js"></script>
  <script src="../static/js/main.js"></script>
</body>
</html>
```

User.html

```
 <!DOCTYPE html>
<html lang="en">`
<head>
  <meta charset="utf-8">
  <meta content="width=device-width, initial-scale=1.0" name="viewport">
  <title>Malicious Social Bots </title>
  <meta content="" name="description">
  <meta content="" name="keywords">
                                                                    <link
href="https://fonts.googleapis.com/css?family=Open+Sans:300,300i,400,400i,600,600i,700,700i|Roboto:300,300i,400,400i,500,500i,600,600i,700,700i|Poppins:300,300i,400,400i,500,500i,600,600i,700,700i" rel="stylesheet">

  <!-- Vendor CSS Files -->
```

```
  <link href="../static/vendor/animate.css/animate.min.css" rel="stylesheet">
  <link href="../static/vendor/bootstrap/css/bootstrap.min.css" rel="stylesheet">
  <link href="../static/vendor/bootstrap-icons/bootstrap-icons.css" rel="stylesheet">
  <link href="../static/vendor/boxicons/css/boxicons.min.css" rel="stylesheet">
  <link href="../static/vendor/glightbox/css/glightbox.min.css" rel="stylesheet">
  <link href="../static/vendor/swiper/swiper-bundle.min.css" rel="stylesheet">

  <link href="../static/css/style.css" rel="stylesheet">
</head>

<body>
  <header id="header" class="fixed-top d-flex align-items-center">
    <div class="container d-flex align-items-center justify-content-between">

      <h1 class="logo"><a href="index.html">Malicious</a></h1>
      <nav id="navbar" class="navbar">
        <ul>
          <li><a class="nav-link scrollto" href="{{ url_for('index')}}">Home</a></li>
              <li><a class="nav-link scrollto" href="{{ url_for('userdetail')}}">Register
details</a></li>
      <li><a class="nav-link scrollto " href="{{ url_for('admin')}}">Full details</a></li>
      <li><a class="nav-link scrollto " href="{{ url_for('user')}}">Analysis</a></li>
       </ul>
        <i class="bi bi-list mobile-nav-toggle"></i>
      </nav><!-- .navbar -->

    </div>
  </header>
  <section id="hero">
    <div class="hero-container">
      <div id="heroCarousel" data-bs-interval="5000" class="carousel slide carousel-fade"
data-bs-ride="carousel">

        <ol class="carousel-indicators" id="hero-carousel-indicators"></ol>

        <div class="carousel-inner" role="listbox">

          <!-- Slide 1 -->
          <div class="carousel-item active" style="background: url(../static/img/mal6.jpg);">
            <div class="carousel-container">
              <div class="carousel-content">
                <h2 class="animate__animated animate__fadeInDown">Detection of Malicious
Social Bots Using machine Learning</h2>
```

```
            <div>
            </div>
          </div>
        </div>
      </div>
    </div>
  </div>
 </section><!-- End Hero -->

 <main id="main">
  <section id="services" class="services">
   <div class="container">

    <div class="section-title">
     <h2>Analysis</h2>
    <link href="../static/css/bootstrap.min.css" rel="stylesheet">
  <link href="../static/css/jumbotron-narrow.css" rel="stylesheet">
  <script src="../static/js/jquery-1.11.2.js"></script>
  <script src="../static/js/Chart.min.js"></script>
</head>
<body>
  <div class="container">
   <div class="header">


   </div>
  <canvas id="chart" width="600" height="400"></canvas>
<footer class="footer">

<script>

  // bar chart data
  var chartData = {
  labels : [{% for item in labels %}
             "{{item}}",
         {% endfor %}],
  datasets : [
    {
        label: '{{legend}}',
        fillColor: "rgba(151,187,205,0.2)",
        strokeColor: "rgba(151,187,205,1)",
```

```
        pointColor: "rgba(151,187,205,1)",
        pointStrokeColor: "#fff",
        pointHighlightFill: "#fff",
        pointHighlightStroke: "rgba(151,187,205,1)",
        bezierCurve : false,
        data : [{% for value in values %}
                 {{value}},
               {% endfor %}]
    }]
  }

   Chart.defaults.global.animationSteps = 50;
   Chart.defaults.global.tooltipYPadding = 16;
   Chart.defaults.global.tooltipCornerRadius = 0;
   Chart.defaults.global.tooltipTitleFontStyle = "normal";
   Chart.defaults.global.tooltipFillColor = "rgba(0,0,0,0.8)";
   Chart.defaults.global.animationEasing = "easeOutBounce";
   Chart.defaults.global.responsive = false;
   Chart.defaults.global.scaleLineColor = "black";
   Chart.defaults.global.scaleFontSize = 16;

  // get bar chart canvas
  var ctx = document.getElementById("chart").getContext("2d");

  steps = 10
  max = 40

  var BarChartDemo = new Chart(ctx).Bar(chartData, {
      scaleOverride: true,
      scaleSteps: steps,
      scaleStepWidth: Math.ceil(max / steps),
      scaleStartValue: 0,
      scaleShowVerticalLines: true,
      scaleShowGridLines : true,
      barShowStroke : true,
      scaleShowLabels: true,
      bezierCurve: false,

  });

</script>
</body>
</html>
```

```
  </div>
</div>
    </section>
   <a href="#" class="back-to-top d-flex align-items-center justify-content-center"><i class="bi bi-arrow-up-short"></i></a>
  <script src="../static/vendor/bootstrap/js/bootstrap.bundle.min.js"></script>
  <script src="../static/vendor/glightbox/js/glightbox.min.js"></script>
  <script src="../static/vendor/isotope-layout/isotope.pkgd.min.js"></script>
  <script src="../static/vendor/php-email-form/validate.js"></script>
  <script src="../static/vendor/purecounter/purecounter.js"></script>
  <script src="../static/vendor/swiper/swiper-bundle.min.js"></script>
  <script src="../static/js/main.js"></script>
</body>
</html>
```

Prediction.html

```
 <!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta content="width=device-width, initial-scale=1.0" name="viewport">
  <title>Malicious Social Bots </title>
  <meta content="" name="description">
  <meta content="" name="keywords">
 <link href="https://fonts.googleapis.com/css?family=Open+Sans:300,300i,400,400i,600,600i,700,700i|Roboto:300,300i,400,400i,500,500i,600,600i,700,700i|Poppins:300,300i,400,400i,500,500i,600,600i,700,700i" rel="stylesheet">
  <link href="../static/vendor/animate.css/animate.min.css" rel="stylesheet">
  <link href="../static/vendor/bootstrap/css/bootstrap.min.css" rel="stylesheet">
  <link href="../static/vendor/bootstrap-icons/bootstrap-icons.css" rel="stylesheet">
  <link href="../static/vendor/boxicons/css/boxicons.min.css" rel="stylesheet">
  <link href="../static/vendor/glightbox/css/glightbox.min.css" rel="stylesheet">
  <link href="../static/vendor/swiper/swiper-bundle.min.css" rel="stylesheet">

  <!-- Template Main CSS File -->
  <link href="../static/css/style.css" rel="stylesheet">
</head>
<body>
  <header id="header" class="fixed-top d-flex align-items-center">
    <div class="container d-flex align-items-center justify-content-between">
      <h1 class="logo"><a href="index.html">Malicious</a></h1>
```

```
    <!-- Uncomment below if you prefer to use an image logo -->
      <!-- <a href="index.html" class="logo"><img src="assets/img/logo.png" alt="" class="img-fluid"></a>-->

    <nav id="navbar" class="navbar">
      <ul>
    <li><a class="nav-link scrollto " href="{{ url_for('profile')}}">profile</a></li>
      <li><a class="nav-link scrollto " href="{{ url_for('prediction')}}">Tweets</a></li>
    <li><a class="nav-link scrollto " href="{{ url_for('users')}}">Timeline</a></li>
       </ul>
      <i class="bi bi-list mobile-nav-toggle"></i>
    </nav><!-- .navbar -->

   </div>
  </header><!-- End Header -->

  <!-- ======= Hero Section ======= -->
  <section id="hero">
    <div class="hero-container">
     <div id="heroCarousel" data-bs-interval="5000" class="carousel slide carousel-fade" data-bs-ride="carousel">

      <ol class="carousel-indicators" id="hero-carousel-indicators"></ol>

      <div class="carousel-inner" role="listbox">

       <!-- Slide 1 -->
       <div class="carousel-item active" style="background: url(../static/img/mal6.jpg);">
        <div class="carousel-container">
         <div class="carousel-content">
          <h2 class="animate__animated animate__fadeInDown">Detection of Malicious Social Bots Using machine Learning</h2>
           <div>
           </div>
         </div>
        </div>
       </div>
      </div>
    </div>
   </div>
  </section><!-- End Hero -->

  <main id="main">
```

```
   <section id="services" class="services">
     <div class="container">

       <div class="section-title">
         <h2>Tweet</h2>
         <body>
 <div class="login">

   <h1 class="cover-heading" ></h1>
    <div class="col-lg-4 ml-auto" style="margin-left:390px;"  >

    <!-- Main Input For Receiving Query to our ML -->
   <form action="{{ url_for('chart')}}"method=['POST']>
          <textarea type="text" class="form-control text-black" style="width:300px;
height:100px" name="news" placeholder="post your tweet"  ></textarea><br>
<div class="col-lg-4 ml-auto" style="margin-left:90px;"  >
                    <center><button type="submit" class="btn btn-primary btn-
large">Predict</button></a></center>
    </div>
   </form>
</div>
 </div>
</body>
       </div>
       </div>
   </section>
  <footer id="footer">
    <div class="footer-top">
     <div class="container">
       <div class="row">
         <div class="col-lg-3 col-md-6">
           <div class="footer-info">
           </div>
         </div>
         <div class="col-lg-2 col-md-6 footer-links">
         </div>
         <div class="col-lg-3 col-md-6 footer-links">
         </div>
         <div class="col-lg-4 col-md-6 footer-newsletter">
         </div>
       </div>
     </div>
    </div>
```

```
    <div class="container">
      <div class="copyright">
      </div>
      <div class="credits">
    </div>
  </footer>
   <a href="#" class="back-to-top d-flex align-items-center justify-content-center"><i class="bi bi-arrow-up-short"></i></a>
  <!-- Vendor JS Files -->
  <script src="../static/vendor/bootstrap/js/bootstrap.bundle.min.js"></script>
  <script src="../static/vendor/glightbox/js/glightbox.min.js"></script>
  <script src="../static/vendor/isotope-layout/isotope.pkgd.min.js"></script>
  <script src="../static/vendor/php-email-form/validate.js"></script>
  <script src="../static/vendor/purecounter/purecounter.js"></script>
  <script src="../static/vendor/swiper/swiper-bundle.min.js"></script>
  <!-- Template Main JS File -->
  <script src="../static/js/main.js"></script>
</body>
</html>
```

Upload.html

```
 <!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta content="width=device-width, initial-scale=1.0" name="viewport">
  <title>Malicious Social Bots </title>
  <meta content="" name="description">
  <meta content="" name="keywords">
 <link href="https://fonts.googleapis.com/css?family=Open+Sans:300,300i,400,400i,600,600i,700,700i|Roboto:300,300i,400,400i,500,500i,600,600i,700,700i|Poppins:300,300i,400,400i,500,500i,600,600i,700,700i" rel="stylesheet">
  <!-- Vendor CSS Files -->
  <link href="../static/vendor/animate.css/animate.min.css" rel="stylesheet">
  <link href="../static/vendor/bootstrap/css/bootstrap.min.css" rel="stylesheet">
  <link href="../static/vendor/bootstrap-icons/bootstrap-icons.css" rel="stylesheet">
  <link href="../static/vendor/boxicons/css/boxicons.min.css" rel="stylesheet">
  <link href="../static/vendor/glightbox/css/glightbox.min.css" rel="stylesheet">
  <link href="../static/vendor/swiper/swiper-bundle.min.css" rel="stylesheet">
  <link href="../static/css/style.css" rel="stylesheet">
</head>
<body></body>
```

```
 <header id="header" class="fixed-top d-flex align-items-center">
  <div class="container d-flex align-items-center justify-content-between">
   <h1 class="logo"><a href="index.html"> Malicious </a></h1>
   <nav id="navbar" class="navbar">
    <ul>
     <li><a class="nav-link scrollto active" href="{{ url_for('index')}}">Home</a></li>
      <li><a class="nav-link scrollto" href="#services">Abstract</a></li>
   <li><a class="nav-link scrollto " href="{{ url_for('upload')}}">upload</a></li>
    </ul>
    <i class="bi bi-list mobile-nav-toggle"></i>
   </nav>
  </div>
 </header>
 <section id="hero">
  <div class="hero-container">
    <div id="heroCarousel" data-bs-interval="5000" class="carousel slide carousel-fade"
data-bs-ride="carousel">
     <ol class="carousel-indicators" id="hero-carousel-indicators"></ol>
     <div class="carousel-inner" role="listbox">
      <!-- Slide 1 -->
      <div class="carousel-item active" style="background: url(../static/img/mal6.jpg);">
       <div class="carousel-container">
        <div class="carousel-content">
         <h2 class="animate__animated animate__fadeInDown">Detection of Malicious
Social Bots Using machine Learning</h2>
          <div>
          </div>
        </div>
       </div>
      </div>
     </div>
   </div>
  </div>
 </section><!-- End Hero -->
 <main id="main">
  <section id="services" class="services">
   <div class="container">
    <div class="section-title">
     <h2>upload</h2>
   <body id="page-top">
<!-- About Section -->
 <section class="col-md-12 col-sm-12 col-xs-12" id="about">
      <div class="col-md-12 col-sm-12 col-xs-12" style="margin-left:100px;">
```

```
          <form action="http://localhost:5000/preview" name="fs" id="fs" method="post"
enctype=multipart/form-data>
                    <br/>
                   <input type="file" name="datasetfile" id="file1" required />
                    <br/>
           <br/>
           <br/>
                <input type="submit" style="margin-right:230px" class="btn btn-success btn-
large" value="Upload">
                   </form>
     </div>

  </section>


</body>


       </div>
       </div>
    </section>
  <footer id="footer">
    <div class="footer-top">
      <div class="container">
        <div class="row">

          <div class="col-lg-3 col-md-6">
            <div class="footer-info">

            </div>
          </div>

          <div class="col-lg-2 col-md-6 footer-links">

          </div>
          <div class="col-lg-3 col-md-6 footer-links">
          </div>
          <div class="col-lg-4 col-md-6 footer-newsletter">
          </div>

        </div>
      </div>
```

```
    </div>
    <div class="container">
      <div class="copyright">
      </div>
      <div class="credits">
    </div>
  </footer>
   <a href="#" class="back-to-top d-flex align-items-center justify-content-center"><i class="bi bi-arrow-up-short"></i></a>
  <script src="../static/js/main.js"></script>
</body>
</html>
```

Python code:

```
from flask import Flask, render_template, request, url_for, Markup, jsonify
import pickle
import pandas as pd
import numpy as np
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from flask import Flask, render_template, request, url_for, session, redirect, flash
from flask_mysqldb import MySQL
import MySQLdb.cursors
import re
import pickle
import pandas as pd
import pymsgbox
app = Flask(__name__)
pickle_in = open('model.pkl','rb')
pac = pickle.load(pickle_in)
tfid = open('tfidf_vectorizer.pkl','rb')
tfidf_vectorizer = pickle.load(tfid)
app.secret_key = 'neha'
app.config['MYSQL_HOST'] = 'localhost'
app.config['MYSQL_USER'] = 'root'
app.config['MYSQL_PASSWORD'] = 'root'
app.config['MYSQL_DB'] = 'malware'
try:
mysql = MySQLdb.connect(
host='localhost',
user='root',
```

```
passwd='root',
db='malware'
)
except Exception as e:
print("---- error is ",e)

Id=3
@app.route('/')
def index():
return render_template('index.html')


@app.route('/login', methods = ['GET',"POST"])
def login():
msg = ''

if request.method == 'POST' and 'username' in request.form and 'password' in request.form:
username = request.form['username']
password = request.form['password']
cursor = mysql.cursor(MySQLdb.cursors.DictCursor)
cursor.execute('SELECT * FROM employee WHERE username = %s AND password =
%s AND status="Approved" ', (username, password))
account = cursor.fetchone()
print("----------",account)
if account:
session['loggedin'] = True
global Id
session['Id'] = account['Id']

Id = session['Id']
session['username'] = account['username']
return redirect(url_for('profile'))
else:
flash('Incorrect username/password! or Account Blocked')
return redirect(url_for('login'))
return render_template('login.html', msg=msg)

@app.route('/register',methods= ['GET',"POST"])
def register():
msg = ''
if request.method == 'POST' and 'username' in request.form and 'password' in request.form
and 'email' in request.form:
username = request.form['username']
```

```
password = request.form['password']
email = request.form['email']
reg = "^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[@$!%*#?&])[A-Za-z\d@$!#%*?&]{6,10}$"
pattern = re.compile(reg)
cursor = mysql.cursor(MySQLdb.cursors.DictCursor)
cursor.execute('SELECT * FROM employee WHERE Username = %s', (username,))
account = cursor.fetchone()
if account:
msg = 'Account already exists!'
elif not re.match(r'[^@]+@[^@]+\.[^@]+', email):
msg = 'Invalid email address!'
elif not re.match(r'[A-Za-z0-9]+', username):
msg = 'Username must contain only characters and numbers!'
elif not re.search(pattern,password):
msg = 'Password should contain atleast one number, one lower case character, one uppercase character,one special symbol and must be between 6 to 10 characters long'
elif not username or not password or not email:
msg = 'Please fill out the form!'
else:
cursor.execute('INSERT INTO employee VALUES (NULL, %s, %s, %s, "Approved")', (username, email, password))
mysql.commit()
flash('You have successfully registered! Please proceed for login!')
return redirect(url_for('login'))
elif request.method == 'POST':
# Form is empty... (no POST data)
msg = 'Please fill out the form!'
return msg
# Show registration form with message (if any)
return render_template('register.html', msg=msg)

@app.route('/loginadmin')
def loginadmin():
return render_template('loginadmin.html')


@app.route('/upload')
def upload():
return render_template('upload.html')
@app.route('/preview',methods=["POST"])
def preview():
if request.method == 'POST':
```

```
dataset = request.files['datasetfile']
df = pd.read_csv(dataset,encoding = 'unicode_escape')
df.set_index('Id', inplace=True)
return render_template("preview.html",df_view = df)

@app.route('/prediction')
def prediction():
return render_template("prediction.html")

@app.route('/chart')
def chart():
abc = request.args.get('news')
pattern =  r'(?:http://)?\w+\.\S*[^.\s]'
input_data=re.findall(pattern, abc)

if input_data:
tfidf_test = tfidf_vectorizer.transform(input_data)
y_pred = pac.predict(tfidf_test)
pred=format(y_pred[0])

login()
cur = mysql.cursor(MySQLdb.cursors.DictCursor)
cur.execute("INSERT INTO review(news_content,pred,userid,url) VALUES ( %s, %s,%s,%s) ", (input_data,pred,Id,abc))
mysql.commit()
cur.close()

else:
pymsgbox.alert('Enter the url', 'warning')
return render_template('prediction.html')
return redirect('/users')

@app.route('/users')
def users():
cur = mysql.cursor(MySQLdb.cursors.DictCursor)
resultValue = cur.execute(" SELECT * from employee INNER JOIN review ON employee.ID = review.USERID WHERE status !='Blocked';")

if resultValue > 0:
userDetails = cur.fetchall()
print("user details are",userDetails[0])
return render_template('users.html',userDetails=userDetails)
```

```
@app.route('/admin')
def admin():
cur = mysql.cursor(MySQLdb.cursors.DictCursor)
resultValue = cur.execute(" SELECT * from employee INNER JOIN review ON employee.ID = review.USERID;")

if resultValue > 0:
userDetails = cur.fetchall()
return render_template('admin.html',userDetails=userDetails)
@app.route('/profile')
def profile():

cursor = mysql.cursor(MySQLdb.cursors.DictCursor)
global Id
print("id is ",Id)
cursor.execute('SELECT * from employee WHERE Id = %s', (Id,))
account = cursor.fetchone()
print("account is",account)
return render_template('profile.html', account=account)

@app.route('/userdetail')
def userdetail():
cur = mysql.cursor(MySQLdb.cursors.DictCursor)
cur.execute("SELECT * from employee")
useradmin=cur.fetchall()
print(useradmin)
return render_template('userdetail.html',useradmin=useradmin)

@app.route('/blockUser',methods=['GET','POST'])
def blockUser():
if request.method == 'POST' and 'fid' in request.form:
f1 = request.form['fid']
con=mysql.cursor(MySQLdb.cursors.DictCursor)
con.execute("UPDATE employee SET status='Blocked' WHERE Id=%s " , (f1,))
mysql.commit()
con.close()
return redirect(url_for('admin'))

@app.route('/user')
def user():
legend = "review by pred"

cursor = mysql.cursor(MySQLdb.cursors.DictCursor)
```

```
values=[]
try:
cursor.execute("SELECT pred from review GROUP BY pred")
# data = cursor.fetchone()
rows1 = cursor.fetchall()
print("rows------",rows1)
labels = list()
i = 'pred'
for row1 in rows1:
labels.append(row1['pred'])


cursor.execute("SELECT pred from review GROUP BY pred")
# data = cursor.fetchone()
rows2 = cursor.fetchall()

label = list()
j = 0
values = list()
k = 0
for row2 in rows2:
print("row 2 is",row2)
label.append(row2['pred'])
print("after label append",row2)
cursor.execute("SELECT COUNT(id) from review WHERE pred=%s", (row2['pred'],))
rows3 = cursor.fetchall()
j=j+1
print("---------- row3333",rows3)
for row3 in rows3:
values.append(row3['COUNT(id)'])
print("done")
except Exception as e:
print('Error: unable to fetch items',e)
return render_template('user.html', values=values, labels = labels, legend=legend)

if __name__=='__main__':
app.run(debug=True)
```

10.SCREENSHOTS

SCREEN SHOTS

Fig 10.1 Abstract Page

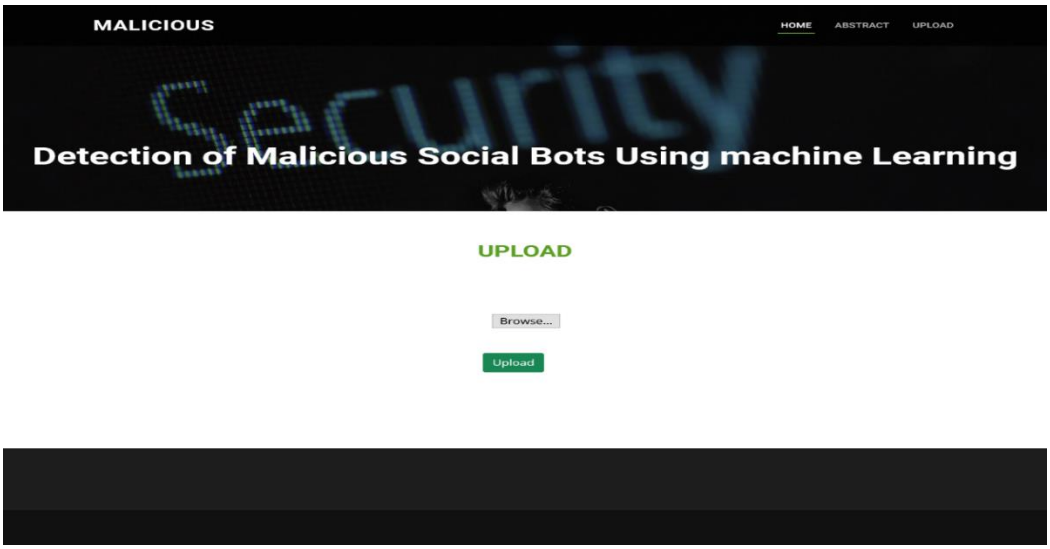


Fig 10.2 Upload Details

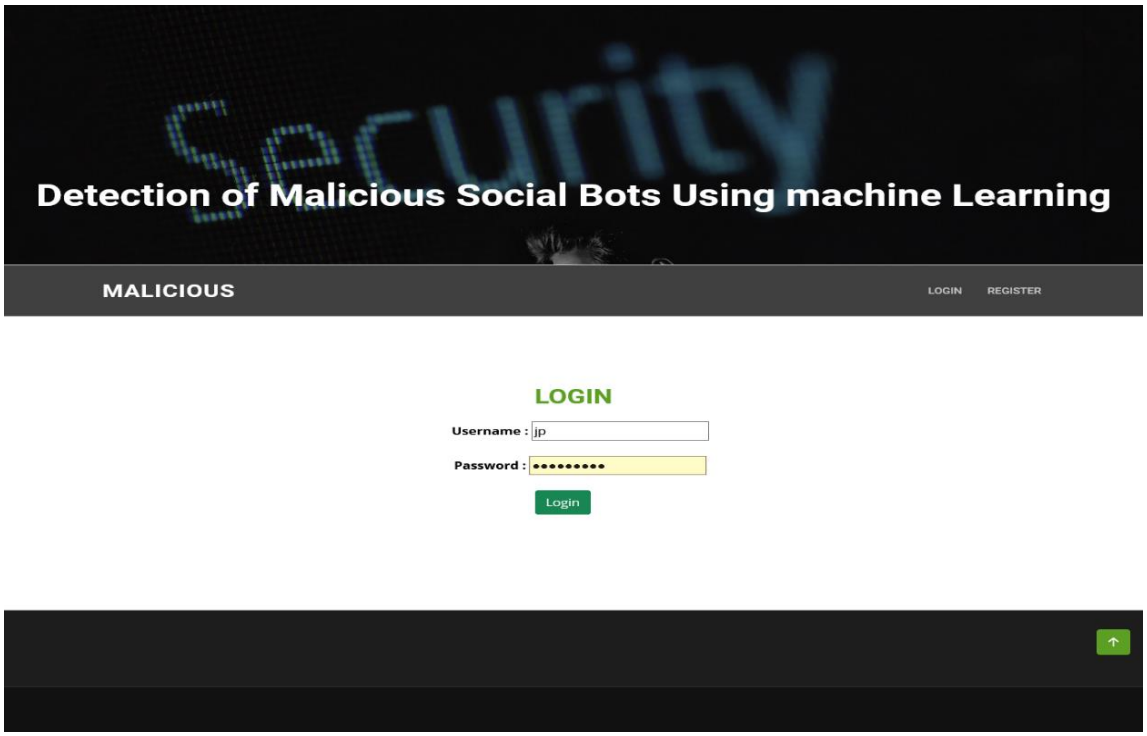


Fig 10.3 Login Page

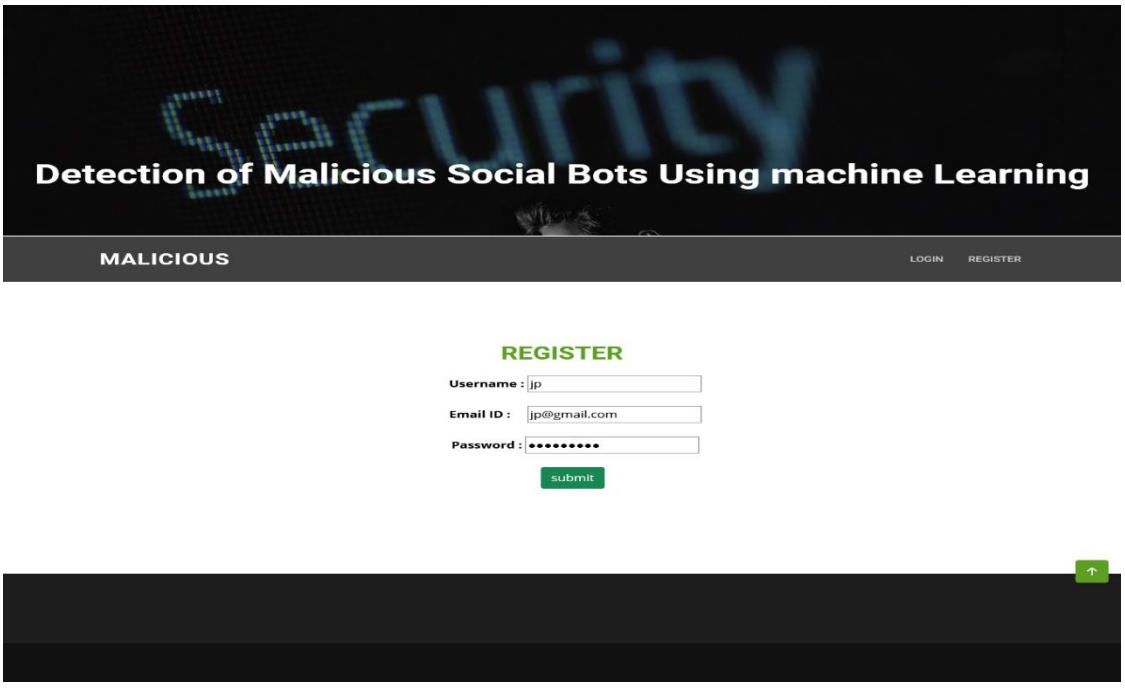


Fig 10.4 Register Page

Fig 10.5 Register Details

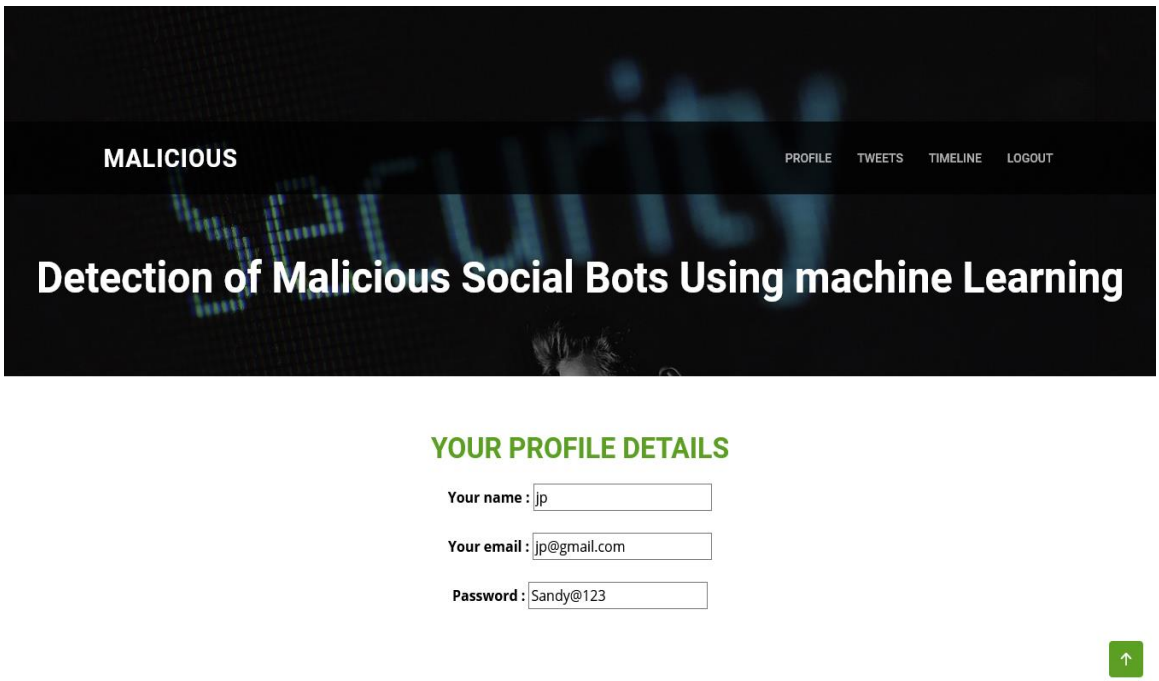


Fig 10.6 Profile details

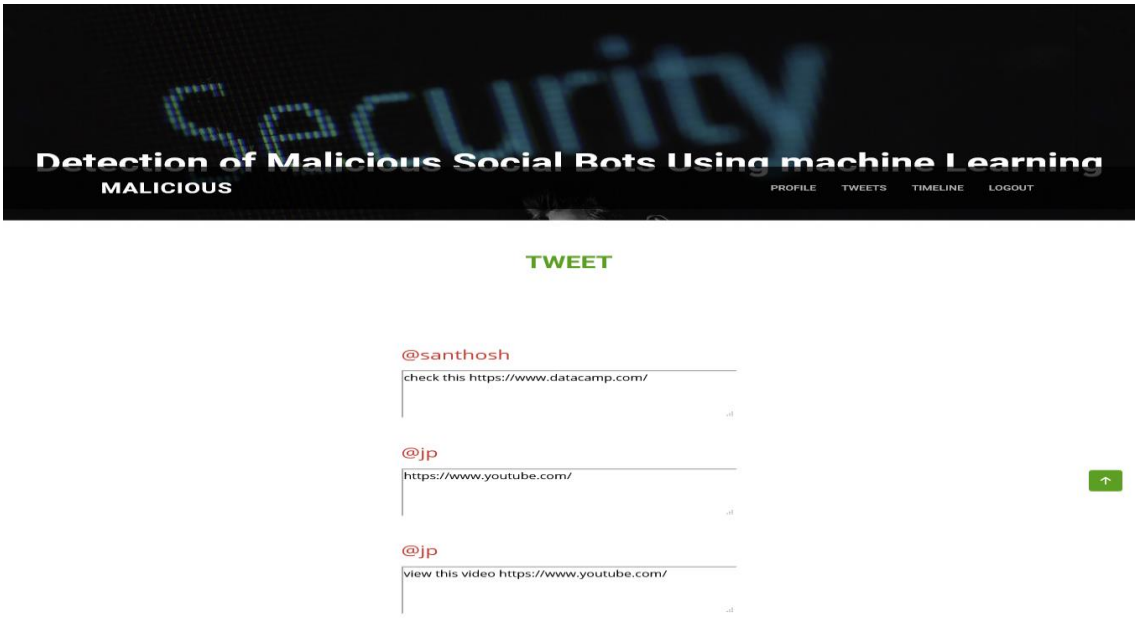


Fig 10.7 Tweet Details

Fig 10.8 Admin login Page

Fig 10.9 Full Details of User

Fig 10.10 Analysis

11. SYSTEM TESTING

SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product .

Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

Types of test:

1. Unit test

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

2. Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

3. Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

4. System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

- **White Box Testing**

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

- **Black Box Testing**

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under

test is treated, as a black box .you cannot "see" into it. The test provides inputs and responds to outputs without considering how the software works.

6.1 Unit Testing:

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

- **Test objectives**

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

- **Features to be tested**

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

6.2 Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

6.3 Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

12.CONCLUSION

CONCLUSION

This article presents an LA-MSBD algorithm by integrating a trust computational model with a set of URL-based features for MSBD. In addition, we evaluate the trustworthiness of tweets (posted by each participant) by using the Bayesian learning and DST. Moreover, the proposed LA-MSBD algorithm executes a finite set of learning actions to update action probability value (i.e., probability of a participant posting malicious URLs in the tweets). The proposed LA-MSBD algorithm achieves the advantages of incremental learning. Two Twitter data sets are used to evaluate the performance of our proposed LA-MSBD algorithm. The experimental results show that the proposed LA-MSBD algorithm achieves up to 7% improvement of accuracy compared with other existing algorithms. For The Fake Project and Social Honeypot data sets, the proposed LA-MSBD algorithm has achieved precisions of 95.37% and 91.77% for MSBD, respectively. Furthermore, as a future research challenge, we would like to investigate the dependence among the features and its impact on MSBD.

FUTURE ENHANCEMENT

1. Collect Data: Gather comprehensive Twitter data, focusing on URL features, user behavior, and content patterns.
2. Extract URL Features: Analyze URLs for characteristics such as domain names, shortening services, and redirection patterns.
3. Analyze User Behavior: Measure tweet frequency, follower/following ratios, and interaction patterns.
4. Develop Learning Automata: Create learning automata to dynamically adjust detection strategies based on evolving bot behaviors.
5. Train Machine Learning Models: Use extracted features and validated datasets to train robust machine learning models.
6. Evaluate Models: Assess model performance using precision, recall, F1-score, and AUC-ROC metrics to ensure effectiveness.
7. Ensure Ethical Compliance: Adhere to data privacy regulations and maintain transparency in the detection process.
8. Deploy for Real-time Monitoring: Implement the system for real-time detection, ensuring scalability and continuous adaptation to new bot behaviors.

13.REFERENCES

REFERENCES

[1] P. Shi, Z. Zhang, and K.-K.-R. Choo, "Detecting malicious social bots based on clickstream sequences," IEEE Access, vol. 7, pp. 28855–28862, 2019.

[2] G. Lingam, R. R. Rout, and D. V. L. N. Somayajulu, "Adaptive deep Q-learning model for detecting social bots and influential users in online social networks," Appl. Intell., vol. 49, no. 11, pp. 3947–3964, Nov. 2019.

[3] D. Choi, J. Han, S. Chun, E. Rappos, S. Robert, and T. T. Kwon, "Bit.ly/practice: Uncovering content publishing and sharing through URL shortening services," Telematics Inform., vol. 35, no. 5, pp. 1310–1323, 2018.

[4] S. Lee and J. Kim, "Fluxing botnet command and control channels with URL shortening services," Comput. Commun., vol. 36, no. 3, pp. 320–332, Feb. 2013.

[5] S. Madisetty and M. S. Desarkar, "A neural network-based ensemble approach for spam detection in Twitter," IEEE Trans. Comput. Social Syst., vol. 5, no. 4, pp. 973–984, Dec. 2018.

[6] H. B. Kazemian and S. Ahmed, "Comparisons of machine learning techniques for detecting malicious webpages," Expert Syst. Appl., vol. 42, no. 3, pp. 1166–1177, Feb. 2015.

[7] H. Gupta, M. S. Jamal, S. Madisetty, and M. S. Desarkar, "A framework for real-time spam detection in Twitter," in Proc. 10th Int. Conf. Commun. Syst. Netw. (COMSNETS), Jan. 2018, pp. 380–383.

[8] T. Wu, S. Liu, J. Zhang, and Y. Xiang, "Twitter spam detection based on deep learning," in Proc. Australas. Comput. Sci. Week Multiconf. (ACSW), 2017, p. 3.

[9] Y. Boshmaf, I. Muslukhov, K. Beznosov, and M. Ripeanu, "Key challenges in defending against malicious socialbots," Presented at the 5th USENIX Workshop Large-Scale Exploits Emergent Threats, 2012, pp. 1–4.

[10] G. Yan, “Peri-watchdog: Hunting for hidden botnets in the periphery of online social networks,” Comput. Netw., vol. 57, no. 2, pp. 540–555, Feb. 2013.

[11] F. Morstatter, J. Pfeffer, H. Liu, and K. M. Carley, “Is the sample good enough? comparing data from twitter’s streaming API with Twitter’s firehose,” in Proc. ICWSM, 2013, pp. 1–9.

[12] A. Neumann, J. Barnickel, and U. Meyer, “Security and privacy implications of url shortening services,” in Proc. Workshop Web 2.0 Secur. Privacy, 2010, pp. 1–31.

[13] Online. WHOIS Database. Accessed: Nov.24, 2018. [Online]. Available: http://whois.domaintools.com//

[14] K. Hans, L. Ahuja, and S. K. Muttoo, “Detecting redirection spam using multilayer perceptron neural network,” Soft Comput., vol. 21, no. 13, pp. 3803–3814, Jul. 2017.