**A Project Report on**

# Detection of Fake Profiles in Twitter

submitted in partial fulfillment for the award of

**Bachelor of Technology**

in

## Computer Science & Engineering

By

**S. Komala Sangeeta (Y20ACS556)** **S. B. G. S. Vamsi (Y20ACS566)**

**R. L. K. M. Lakshmi (Y20ACS548)** **P. Naga Satvik (Y20ACS525)**


Under the guidance of

**Asst. Prof. N. Srikanth.**

Department of Computer Science and Engineering

**Bapatla Engineering College**

(Autonomous)

(Affiliated to Acharya Nagarjuna University)

**BAPATLA – 522 102, Andhra Pradesh, INDIA**

**2023-2024**

# Department of
# Computer Science & Engineering


## CERTIFICATE

This is to certify that the project report entitled **Detection of Fake Profiles in twitter** that is being submitted by S. Komala Sangeeta (Y20ACS556), S. B. G. S. Vamsi (Y20ACS566), R. L. K. M. Lakshmi (Y20ACS548), P. Naga Satvik (Y20ACS525) in partial fulfillment for the award of the Degree of Bachelor of Technology in Computer Science & Engineering to the Acharya Nagarjuna University is a record of bonafide work carried out by them under our guidance and supervision.

Date:

**Signature of the Guide**
**Mr. Srikanth Nalluri**
**Asst. Prof**

**Signature of the HOD**
**Dr. M. Rajesh Babu**
**Assoc. Prof. & HOD**

# DECLARATION

We declare that this project work is composed by ourselves, that the work contained herein is our own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

**S. Komala Sangeeta (Y20ACS556)**
**S. B. G. S. Vamsi (Y20ACS566)**
**R. L. K. M. Lakshmi (Y20ACS548)**
**P. Naga Satvik (Y20ACS525)**

# Acknowledgement

We sincerely thank the following distinguished personalities who have given their advice and support for successful completion of the work.

We are deeply indebted to our most respected guide **Mr. Srikanth Nalluri,** Assistant Professor**,** Department of CSE, for his/her valuable and inspiring guidance, comments, suggestions and encouragement.

We extend our sincere thanks to **Dr. M. Rajesh Babu**, Prof. & Head of the Dept. for extending his cooperation and providing the required resources.

We would like to thank our beloved Principal **Dr. Nazeer Shaik** for providing the online resources and other facilities to carry out this work.

We would like to express our sincere thanks to our project coordinator **Dr. N. Sudhakar,** Prof. Dept. of CSE for his helpful suggestions in presenting this document.

We extend our sincere thanks to all other teaching faculty and non-teaching staff of the department, who helped directly or indirectly for their cooperation and encouragement.

**S. Komala Sangeeta (Y20ACS556)**
**S. B. G. S. Vamsi (Y20ACS566)**
**R. L. K. M. Lakshmi (Y20ACS548)**
**P. Naga Satvik (Y20ACS525)**

# Table of Contents

# List of Figures

# List of tables

# Abstract

With the exponential growth of Twitter's user base, the presence of fake profiles has escalated, posing challenges to user security and platform integrity. This study aims to develop a robust machine learning-based system specifically tailored for the detection of fake profiles on Twitter. Harnessing the power of gradient boosting classifiers, renowned for their capability to handle complex datasets and enhance prediction accuracy by sequentially combining weak learners, we delve into the analysis of various account attributes. The data undergoes thorough preprocessing using Python libraries to ensure optimal model performance.

In addition to the sophisticated machine learning algorithm employed, the system is equipped with a user-friendly GUI. This interface enhances user accessibility and engagement. The gradient boosting classifier is thoroughly tested to demonstrate its effectiveness and reliability in distinguishing between real and fake Twitter profiles. The findings underscore the potential of gradient boosting classifiers as a pivotal tool in combating the proliferation of fake profiles, thereby fortifying the trustworthiness of online social networks.



**Keywords:** Twitter, Fake Profile Detection, Gradient Boosting Classifier, Machine Learning, GUI, Online Social Networks.

# 1 Introduction

Accessing information has become effortless with the Internet, allowing people to retrieve data from various global sources. The popularity of social media platforms enables users to gather vast amounts of information and data about other users. However, this accessibility also attracts fake users to these platforms. Twitter has quickly become a leading online platform where users share real-time updates, news, opinions, and even their emotions, reaching a broad audience instantly.

With the rise of online social networks (OSNs), there is an increasing need to study and analyse user behaviours on these platforms. People with limited knowledge about OSNs can easily be deceived by fraudsters. Additionally, there is a growing concern about individuals misusing OSNs solely for advertising, resulting in spamming other users' accounts. Detecting and controlling spam on social networking sites has become essential to maintain the security and privacy of users.

Twitter fake accounts have various objectives, including spreading false information, fake news, rumours, and unsolicited messages. Accounts falsely claiming affiliation with organizations or individuals can damage their reputation, leading to a decrease in followers and likes. Many studies have attempted to spot false accounts on social networking platforms (OSNs), but these approaches often fall short against sophisticated attacks, making them vulnerable. There's currently no foolproof solution to completely address this challenge.

## 1.1 Problem Statement

Social media platforms are vulnerable to fake accounts that spread misinformation, scam users, and manipulate public opinion, highlighting the need for effective detection and removal to maintain platform integrity. And also, traditional machine learning algorithms like SVM and Naïve Bayes may struggle to accurately detect fake accounts, leading to inefficiencies, higher false positive rates, and challenges in adapting to evolving tactics used by malicious actors.

## 1.2 Objective

To develop a machine learning-based solution for detecting fake accounts by analysing various attributes of Twitter accounts through a web interface and thus to determine their legitimacy, providing insights into whether an account is genuine or fabricated.

## 1.3 Machine Learning

Machine learning is a subset of artificial intelligence in the field of computer science that often uses statistical techniques to give computers the ability to learn (i.e., progressively improve performance on a specific task) with data, without being explicitly programmed. Machine learning is the design and study of software artifacts that use past experience to make future decisions. It is the study of programs that learn from data. The fundamental goal of machine learning is to generalize, or to induce an unknown rule from examples of the rule's application.

Machine learning systems are often described as learning from experience either with or without supervision from humans. At a broad level, machine learning can be classified into three types, namely Supervised learning, Unsupervised learning, Semi

Supervised Learning and Reinforcement learning. For detecting fake profiles, we use a supervised learning algorithm.

### 1.3.1 Supervised Learning

A model is prepared through a training process in which it is required to make predictions and is corrected when those predictions are wrong. The training process continues until the model achieves a desired level of accuracy on the training data. A program predicts an output for an input by learning from pairs of labelled inputs and outputs, the program learns from examples of the right answers. Input data is called training data and has a known label or result such as spam/not-spam or a stock price at a time. Supervised learning in machine learning is generally divided into two categories: classification and regression.

**Classification:** Classification is a supervised learning technique focused on assigning predefined labels or categories to input data based on the learned patterns from labelled training examples. In this process, the algorithm analyses the input features to make predictions about the corresponding class labels. The output variable in classification is categorical, representing distinct classes or categories. The goal is to build a model that can accurately classify new, unseen instances by generalizing from the training data. Various algorithms are employed in classification tasks, including logistic regression, support vector machines, decision trees, random forests, and neural networks, each with its strengths and suitable applications. These algorithms learn from the training data to create decision boundaries or rules that best separate the different classes, optimizing for accuracy and predictive performance.

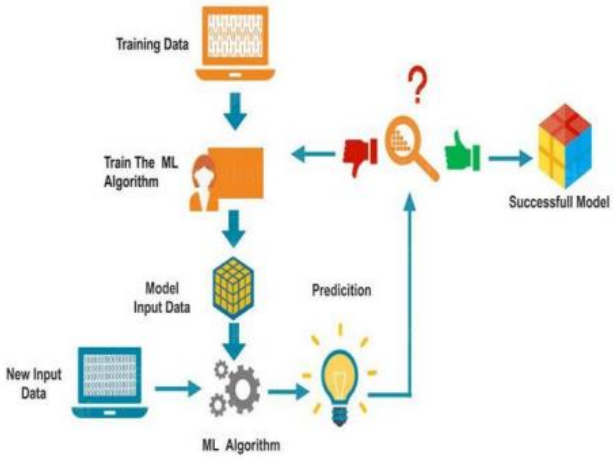


**Figure 1.1 System Architecture for ML classification model**

The classification process starts with data collection, followed by preprocessing and organizing the data into training and testing sets. Then, using algorithms like SVM the model is trained on the training data to learn the relationships between features and labels. After training, the model's performance is evaluated using the testing data, assessing metrics like accuracy and precision. Once validated, the trained model is deployed into production for making predictions on new data, automating decision-making in various applications.

**Regression:** Regression is another type of supervised learning task that focuses on predicting continuous numerical values rather than discrete class labels. In regression, the algorithm learns a mapping from input features to a continuous output variable. The goal is to predict a numeric value based on the input data and the relationships learned from the training examples. Common regression tasks include predicting house prices based on features like size, location, and number of bedrooms, forecasting stock prices based on historical data, and estimating sales revenue based on marketing spend and other factors. Popular regression algorithms include linear

regression, polynomial regression, ridge regression, lasso regression, decision trees, and neural networks.

### 1.3.2 Unsupervised Learning

Unsupervised learning differs from supervised learning in that it deals with unlabeled data. In unsupervised learning, the algorithm is tasked with finding patterns or structures in the data without explicit labels or predefined outcomes. The goal is to discover the inherent structure of the data, such as clusters, associations, or anomalies. This type of learning is particularly useful for exploratory data analysis, data visualization, and feature extraction. Clustering methods like k-means, hierarchical clustering, and DBSCAN are popular unsupervised learning techniques that group similar data points into clusters based on their characteristics.

### 1.3.3 Semi Supervised Learning

Semi-supervised learning bridges the gap between supervised and unsupervised learning by utilizing both labelled and unlabelled data for training. In many real-world scenarios, obtaining labelled data can be costly, time-consuming, or even impossible. Semi-supervised learning addresses this challenge by leveraging the abundant unlabelled data alongside a smaller set of labelled data to improve the performance of the model.

The algorithm uses the labelled data to learn from the known outcomes and the unlabelled data to infer and generalize from the learned information. This approach is particularly beneficial when only a limited amount of labelled data is available, making it more efficient and cost-effective. Popular semi-supervised learning techniques include self-training, where the model iteratively labels the unlabeled data based on its

current predictions, co-training, which uses multiple views or representations of the data to improve learning, and multi-view learning, where different subsets of features or views are used to capture complementary information.

### 1.3.4 Reinforcement Learning

Reinforcement learning is a type of machine learning where an agent learns to make decisions by interacting with an environment to achieve a specific goal or maximize cumulative reward over time. Unlike supervised and unsupervised learning, where the algorithm is trained on a static dataset, reinforcement learning involves an ongoing learning process where the agent takes actions, receives feedback in the form of rewards or penalties, and adjusts its actions accordingly. The goal of the agent is to learn an optimal policy, which is a strategy or set of rules that dictates the actions to take in different states of the environment to maximize the expected reward.

Reinforcement learning is commonly used in various applications, including robotics, gaming, and autonomous systems. Key algorithms in reinforcement learning include Q-learning, which uses a value function to estimate the expected future rewards, deep Q-networks (DQN), which combine Q-learning with deep neural networks to handle complex environments, and policy gradient methods like proximal policy optimization (PPO) and deep deterministic policy gradients (DDPG), which directly optimize the policy to maximize the expected reward.

# 2 Literature Survey

Tehlan et al. [1] have proposed work to identify spam by exhausting Machine Learning Calculation, Fluffy rationale, and Counterfeit intelligence. This approach overcomes confinement spamming by utilizing the Semi-Directed Learning calculation. Therefore 2 G Swathi. International Journal of Science, Engineering and Technology, 2023, 11:6 utilizing fuzzy logic handles the huge information set exceptionally productively and identifies the spam inside a moment, in this paper. It decreases the cost efficiency, time, and utilization of complex programs. The fake accounts on the social location are capable of spamming. The spamming may get to the individual data and start back mail them. Agreeing to a few reports, spam reports spam to contain the account URL.

The Padmavati et al. [2]approach the problem of fake accounts on social media by a method using Deterministic Finite Automata (DFA). The paper analyzes the features of the existing user and their friends by creating an accounting pattern. The pattern is made with regular expressions based on some attributes such as the working and living community and so on which are used for pattern matching with any friend requests. The drawback of this approach is that the generation of regular expression takes quite a long for a person who has friends in many communities. The authors contend that the approach could be even more efficient for the real world.

Machine learning techniques were employed by Ananya et al. [3]in 2021 ICRITO to identify phony social media profiles. They received the data from Kaggle, an open-source web site where data sets are stored for public use. The information came from Weibo, a Chinese version of Twitter and a popular networking site. Later, they

used five supervised learning models for training and to check which one gave a better test score (cross-validation).

Out of all 5 techniques they chose Gradient Boosting Classifier and Random Forest Classifier as they showed a better performance concerning others. Finally, after which they chose a random forest classifier as it gave a 1% better result than the gradient boosting classifier. They aim to make an automated system that can train and take more attributes than the limited ones in this paper for their future work.

Preethi Harris et al. described that fake Instagram profiles could be identified using machine learning [4] The profile data from social media called Instagram was taken from the Kaggle website. They employed classification algorithms such as SVM, KNN, RF, NB, and XG Boost to train the model. After computing the accuracy and confusion matrix, the RF classifier stood out as the suitable model for the data set with the best results of prediction. Later the IDs of the Fake profiles are written into a data dictionary.

Smruthi et al [5] has utilized a cross-breed show and skin location calculation to identify fake accounts on social sites. The result of this proposed work is utilizing 400 blended five Directed Machine Learning Algorithm datasets of fake and genuine accounts. On the other hand, the skin location calculation is additionally utilized. Based on the skin exposure rate, the picture is collected for the fake and real accounts calculated.

Here supervised machine learning calculation uses a choice tree classifier having the most noteworthy exactness which is 80% rest of the classifier gave 60% to 80% precision with a 20% mistake rate. The accuracy rate of the KNN calculation is 60%

and other classifiers help to extend the precision rate up to 80% such as the Bayes and Choice tree classifier.

The skin detection algorithm to begin with was embedded profound learning algorithm for the detection of a picture which picture is human or not. On the off chance that the image may be a human being, at that point apply the skin detection algorithm and calculate the skin uncovered rate. If an image contains more than 13 rate skin exposed percentage, it might be considered as a fake profile since most fake profiles is taken a tall rate of skin exposed area or not have any picture. On the off chance that the skin discovery percentage of the picture is tall, accounts are considered as fake accounts.

# 3 Existing System

In the existing system, two datasets containing 556 and 776 entries, respectively, are utilized. After collecting the data, a cleaning process is performed to prepare the datasets for analysis. Subsequently, the Random Forest Classifier is selected and implemented for the task of identifying fake profiles.

## 3.1 Random Forest

The Random Forest Classifier is a popular machine learning algorithm that belongs to the ensemble learning family. It is a versatile and powerful classification algorithm capable of handling both regression and classification tasks. The core idea behind the Random Forest algorithm is to build multiple decision trees during training and output the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Each tree in the forest is trained on a random subset of the data, and a random subset of features is considered for each split, which helps to introduce diversity and reduce overfitting. Random Forests are known for their robustness and ability to handle large datasets with high dimensionality. They are less prone to overfitting compared to individual decision trees and often provide high accuracy and generalization performance.

One of the key advantages of the Random Forest Classifier is its ability to measure the importance of features in making accurate predictions. After training, the algorithm ranks the features based on their importance, allowing users to identify which features contribute the most to the predictive accuracy of the model. This feature importance analysis can be invaluable for understanding the underlying patterns in the data and for feature selection in subsequent analyses. Additionally, Random Forests are

relatively easy to implement and require minimal hyperparameter tuning compared to other complex algorithms. They are also parallelizable, making them computationally efficient and suitable for handling large datasets. Overall, the Random Forest Classifier is a robust, versatile, and efficient algorithm that is widely used in various domains, including finance, healthcare, and marketing, for its ability to deliver high performance and reliable predictions.

## 3.2 Drawbacks

1. Model's performance can be affected when trained on limited or imbalanced datasets. Limited data can lead to overfitting, where the model performs well on the training data but poorly on unseen data, resulting in reduced generalization and accuracy.
2. Bias-Variance Trade off: Random forests can sometimes suffer from overfitting, especially when the dataset is noisy or when there are irrelevant features. This means that the model may perform exceptionally well on the training data but may not generalize well to new, unseen data.
3. Random forests can be computationally intensive, especially when dealing with large datasets. Training multiple decision trees and aggregating their predictions can require significant computational resources and time.
4. Some times in addition to data cleaning other preprocessing steps may required.

# 4 Proposed System

The proposed system aims to address the pressing issue of fake profile detection on social media platforms, specifically focusing on Twitter. And also to enhance the accuracy of predictions compared to previous algorithms such as Random Forest, SVM, Naïve Bayes and Decision Tree as they may lead to inefficiencies, higher false positive rates. To overcome these limitations, the proposed system leverages the Gradient Boosting Classifier.

## 4.1 Task Overview

The task is to detect the fake profiles on Twitter involves utilizing the Gradient Boosting Classifier, a robust supervised learning algorithm suitable for classification tasks. The process starts with gathering various attributes from Twitter profiles. The data is then pre-processed to prepare it for training. Subsequently, the Gradient Boosting Classifier is trained on labelled data, where each profile is categorized as either genuine or fake based on ground truth labels. Once the model is trained, it is integrated into a web application using the Flask framework. This web application offers a user-friendly interface where users can input the attributes which are used to detect the fake profiles, prompting the model to analyse the profile and predict its legitimacy.

## 4.2 Dataset

The dataset is gathered from Github. It is an open-source platform where datasets are kept for public usage. In the GitHub repository, the datasets for fake and genuine profiles are separated, as we require a dataset that contains both fake and genuine profiles, both the datasets were concatenated as single dataset that consists of 2922

samples. These samples will be divided into two parts as training and testing sets. Few samples are also added manually.

## 4.3 Preprocessing

Pre-processing refers to the transformations applied to our data before feeding it to the algorithm. Data preprocessing is a technique that is used to convert the raw data into a clean data set. In proposed system in addition to the data cleaning, Feature selection, Feature extraction and Feature transformation will also performed.

## 4.4 Model

In machine learning, models serve as mathematical representations of real-world processes, allowing algorithms to make predictions or decisions without being explicitly programmed. These models are trained using data, learning patterns and relationships to generalize from examples. There are various types of machine learning models, including regression models for predicting continuous outcomes, classification models for categorizing data into classes, and clustering models for grouping similar data points.

Additionally, ensemble models combine multiple individual models to improve predictive accuracy and robustness. As the foundation of any machine learning application, the choice of model significantly impacts performance and the ability to derive meaningful insights from data. Thus, selecting the right model is a critical step in developing effective machine learning solutions. For fake profile detection we use Gradient Boosting Classifier model.

### 4.4.1 Gradient Boosting

Gradient Boosting is a popular boosting algorithm in machine learning used for classification and regression tasks. Boosting is one kind of ensemble Learning method which trains the model sequentially and each new model tries to correct the previous model. Here ensemble learning is a machine learning technique that enhances accuracy and resilience in forecasting by merging predictions from multiple models. It aims to mitigate errors or biases that may exist in individual models by leveraging the collective intelligence of the ensemble. Gradient boosting algorithm that combines several weak learners into strong learners.

### 4.4.2 Architecture

The architecture of a Gradient Boosting Classifier (GBC) can be understood as a sequential ensemble of decision trees, where each tree is a weak learner trained to correct the errors of its predecessor.

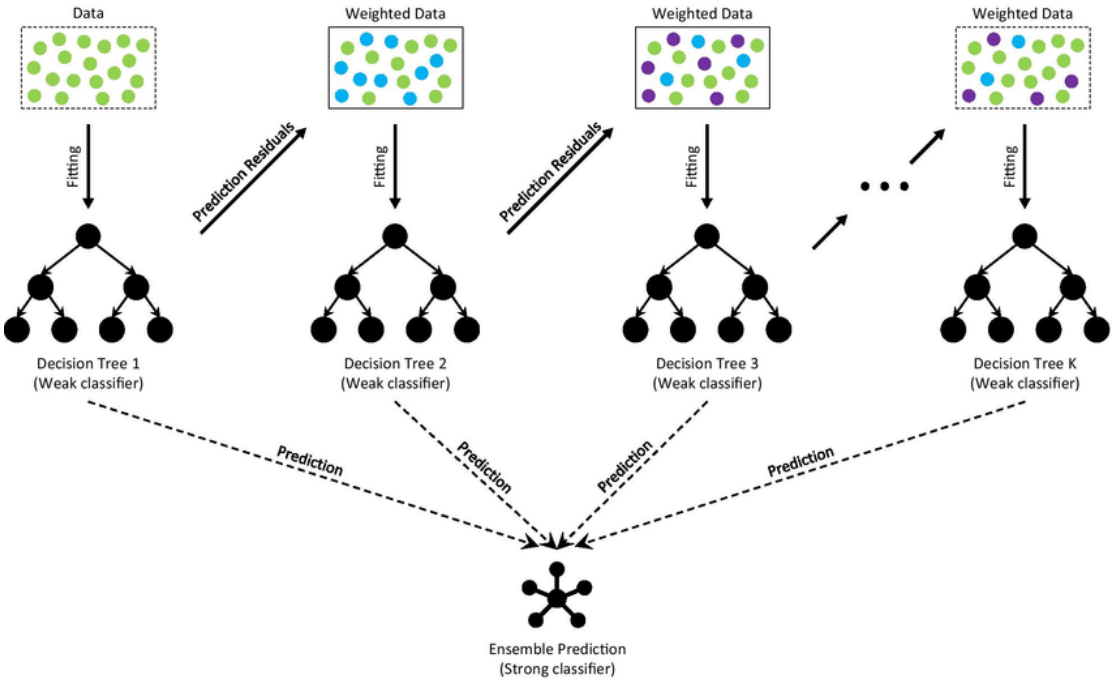


**Figure 4.1 The gradient boosting classifier architecture**

Here is an overview of the architecture

1. Initial Model: The process begins with an initial weak learner, often a simple decision tree, trained on the dataset to make predictions.

2. Residual Calculation: After the initial model makes predictions, the residuals or errors (the differences between the predicted and actual values) are computed for each data point in the dataset.

3. Sequential Training of Trees: In the subsequent iterations, new decision trees are added to the ensemble, each focusing on learning and correcting the residuals from the previous trees. The new tree is trained on the residuals rather than the original target values.

4. Gradient Descent: The term "gradient" in Gradient Boosting refers to the gradient of the loss function, which guides the model to minimize the residuals. The model uses gradient descent optimization to minimize the loss when adding new trees to the ensemble.

5. Tree Fitting: Each decision tree is fit to the residuals, and its predictions are combined with the predictions of the previous trees in the ensemble. The learning rate controls the contribution of each tree to the final prediction.

6. Final Prediction: The final prediction is obtained by aggregating the predictions of all the trees in the ensemble. The combined model, with multiple trees, collectively works to reduce the prediction errors and improve the overall accuracy of the classifier.

# 5 Overview of Flask Framework

Flask is a lightweight WSGI (Web Server Gateway Interface) web application framework. It is designed to make getting started quick and easy, with the ability to scale up to complex applications. It began as a simple wrapper around Werkzeug and Jinja and has become one of the most popular Python web application frameworks. Flask offers suggestions, but doesn't enforce any dependencies or project layout. It is up to the developer to choose the tools and libraries they want to use. There are many extensions provided by the community that make adding new functionality easy.

## 5.1 Features

Flask offers a lightweight and flexible approach to web development with features like routing, HTTP methods, templates, and extensions. It also provides built-in development servers, RESTful API capabilities, configuration options, and middleware support. Here is an overview of Flask's key features and components:

1. **Routing**: Flask uses decorators to define routes and associate them with view functions, allowing you to map URLs to specific functions in your application.

2. **HTTP Methods**: Flask supports various HTTP methods like GET, POST, PUT, DELETE, etc., enabling you to handle different types of requests.

3. **Templates**: Flask integrates with Jinja2, a popular templating engine, to render dynamic HTML content.

4. **Extensions**: Flask has a rich ecosystem of extensions that add additional functionalities like form validation, authentication, database integration, and more.

5. **Configuration**: Flask allows you to configure your application using various methods, such as environment variables, configuration files, or directly in your code.

6. **Development Server**: Flask includes a built-in development server, making it easy to test your application locally.

7. **RESTful APIs**: Flask can be used to build RESTful APIs using extensions like Flask-RESTful or by defining custom routes and methods to handle API endpoints.

8. **Middleware**: Flask allows you to use middleware to handle tasks like authentication, logging, and error handling, helping you keep your application clean and modular.

## 5.2 Flask Development Server

The Flask development server is a built-in, lightweight web server that comes bundled with the Flask framework. It is designed to simplify the development process by providing a quick and easy way to run and test Flask applications locally. When you start the Flask development server using the app.run(debug=True) command, it initializes a local server that listens for incoming HTTP requests. By default, the development server runs on http://127.0.0.1:5000/, which is localhost on port 5000. This allows you to access your Flask application through a web browser on your local machine, making it convenient for testing and debugging your web application during the development phase.

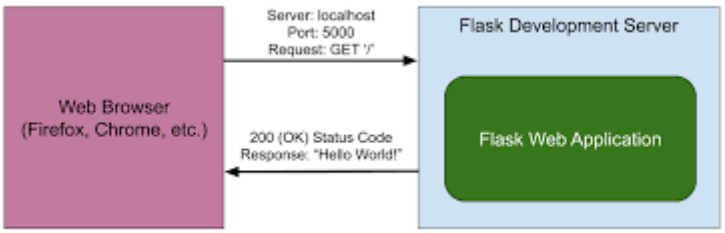


**Figure 5.1 Flask Development Server handling request and responses.**

One of the key features of the Flask development server is its built-in debugger, which can be enabled by setting debug value to True when starting the server. The debugger provides detailed error messages, stack traces, and allows you to inspect variables and step through code execution, facilitating easier debugging of your Flask application. Additionally, the development server supports automatic code reloading. This means that whenever you make changes to your Flask application's code, the server will automatically restart, reflecting the changes without requiring you to manually stop and restart the server. This feature helps streamline the development process, allowing developers to quickly iterate on their code and see the results in real-time without interruption.

## 5.3 Routing

Routing in Flask is the mechanism that maps URLs to specific functions or views within the application. It defines how incoming HTTP requests are handled and processed by the server. The routing system uses decorators like **@app.route()** to associate URL patterns with Python functions. When a request matches a defined route, the corresponding function is executed, performing the necessary operations such as data processing, computation, or rendering templates. This structured approach allows developers to organize the application's logic and handle different types of requests effectively. Routing plays a fundamental role in Flask applications, enabling developers

to create dynamic and interactive web experiences by directing users to the appropriate functionalities based on their actions and inputs.

## 5.4 Rendering templates

Rendering templates in Flask involves combining dynamic data with static HTML templates to generate HTML pages that can be sent as responses to client requests. Flask uses the **render_template()** function to render these templates. This function, accepts a template file's name and optional data to pass to the template. When a route associated with a template is accessed, Flask processes the template, substitutes the dynamic data into the placeholders defined in the template file, and generates a complete HTML page. This allows for the creation of dynamic web pages that can adapt and change based on the data or user input, providing a flexible and interactive user experience.

In our Fake Profile Detection project, we utilized several key Flask elements to build and deploy the web application. First, we initialized the Flask application instance. We set up different paths or URLs that users could visit on our web application. When a user goes to the main or root page of our application, represented by the / URL, they are shown the homepage. This homepage is designed using a html template and is displayed by a function named. Additionally, we created another path, /predict, which is specifically designed to handle predictions. When a user submits a form with data on this /predict page, our function kicks in. This function takes care of making predictions using the submitted data, calculating performance metrics, and then showing the results on a page designed with the result.html template. For rendering dynamic web pages, we employed Flask's render_template() function.

# 6 System Design

System design is the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. It involves translating user requirements into a detailed blueprint that guides the implementation phase. The goal is to create a well-organized and efficient structure that meets the intended purpose while considering factors like scalability, maintainability, and performance.

Unified Modelling Language (UML) diagrams serve as valuable tools in visualizing and documenting the system design, facilitating clear communication among stakeholders and development teams. UML offers a standardized set of graphical notations to represent different facets of the system, including its structure, behaviour, and interactions.

## 6.1 Use Case diagram

Use case diagrams model the behaviour of a system and help to capture the requirements of the system. Use-case diagrams describe the high-level functions and scope of a system. These diagrams also identify the interactions between the system and its actors. The use cases and actors in use-case diagrams describe what the system does and how the actors use it, but not how the system operates internally. Use case diagrams illustrate and define the context and requirements of either an entire system or the important parts of the system.

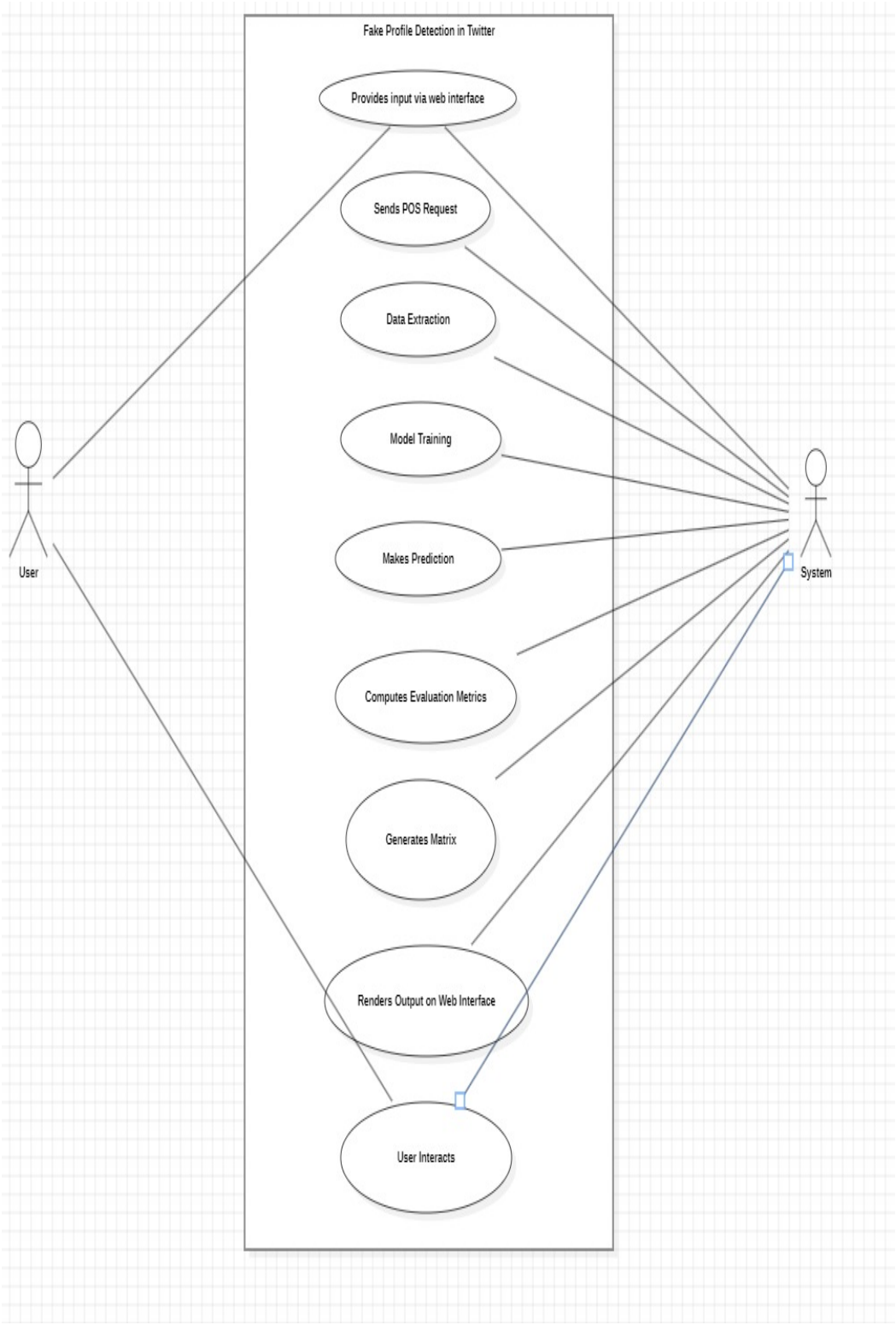


**Figure 6.1 Use case diagram**

## 6.2 Activity Diagram

An activity diagram is used to display the sequence of activities. Activity diagrams show the workflow from a start point to the finish point detailing the many decision paths that exist in the progression of events contained in the activity. They may be used to detail situations where parallel processing may occur in the execution of some activities. Activity diagrams are useful for business modelling where they are used for detailing the processes involved in business activities.

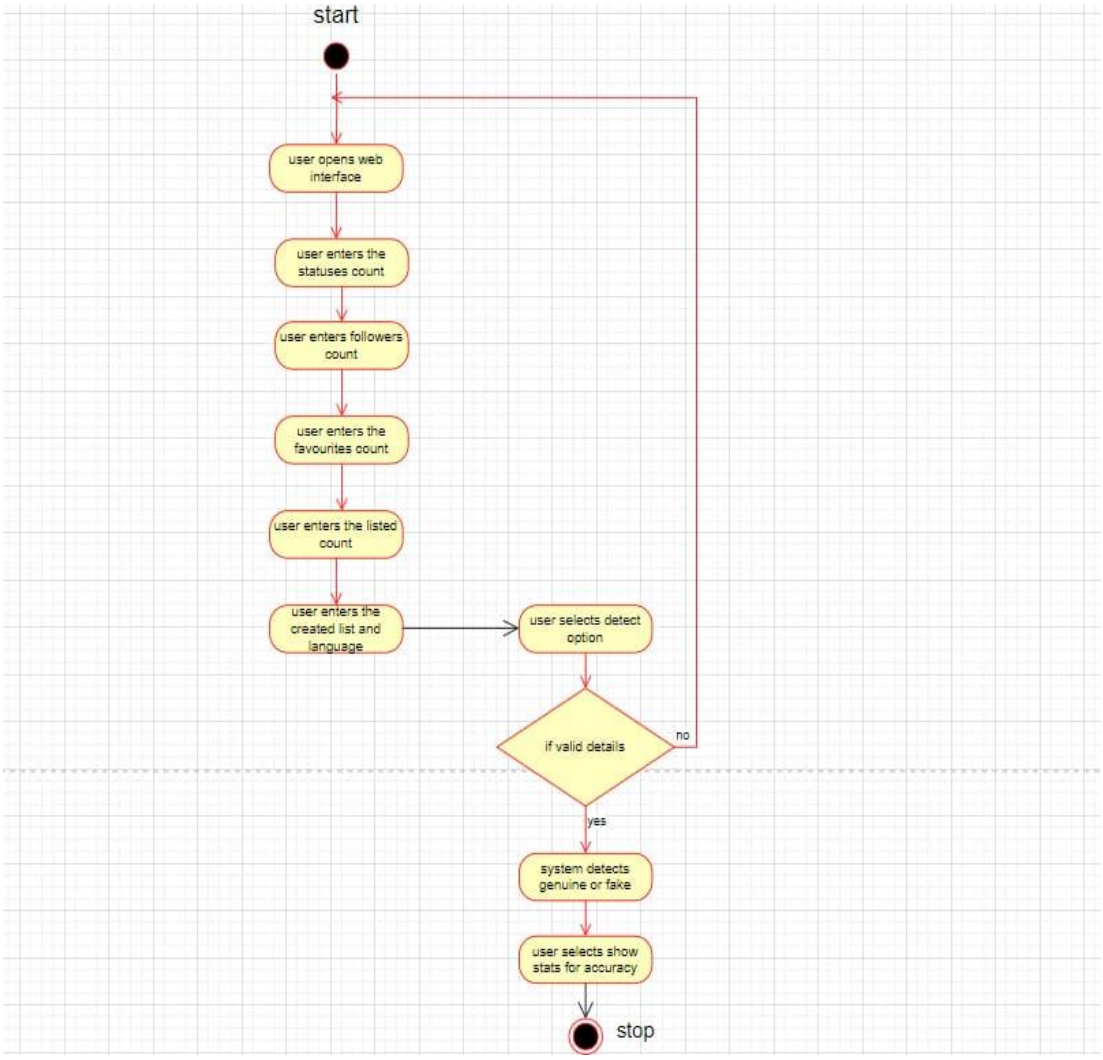


**Figure 6.2 Activity Diagram**

## 6.3 State Chart Diagram

Statechart diagram describes the flow of control from one state to another state. States are defined as a condition in which an object exists and it changes when some event is triggered. The most important purpose of Statechart diagram is to model lifetime of an object from creation to termination.

Statechart diagrams are also used for forward and reverse engineering of a system. However, the main purpose is to model the reactive system.

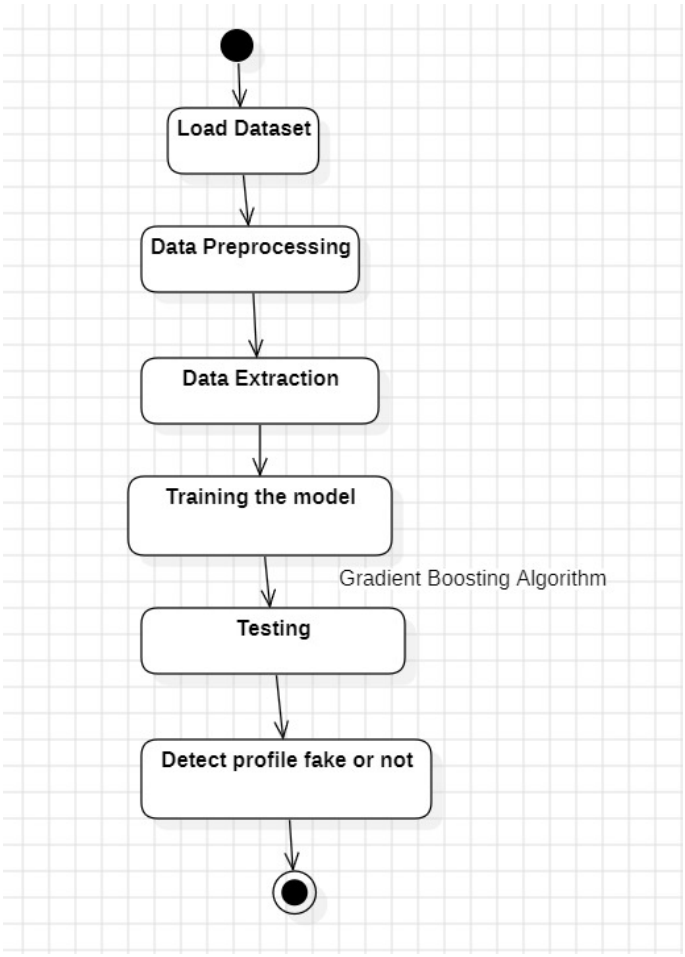


**Figure 6.3 State Chart Diagram**

## 6.4 Sequence Diagram

A sequence diagram is a Unified Modelling Language (UML) diagram that illustrates the sequence of messages between objects in an interaction. A sequence diagram consists of a group of objects that are represented by lifelines, and the messages that they exchange over time during the interaction. A sequence diagram shows the sequence of messages passed between objects. Sequence diagrams can also show the control structures between objects.

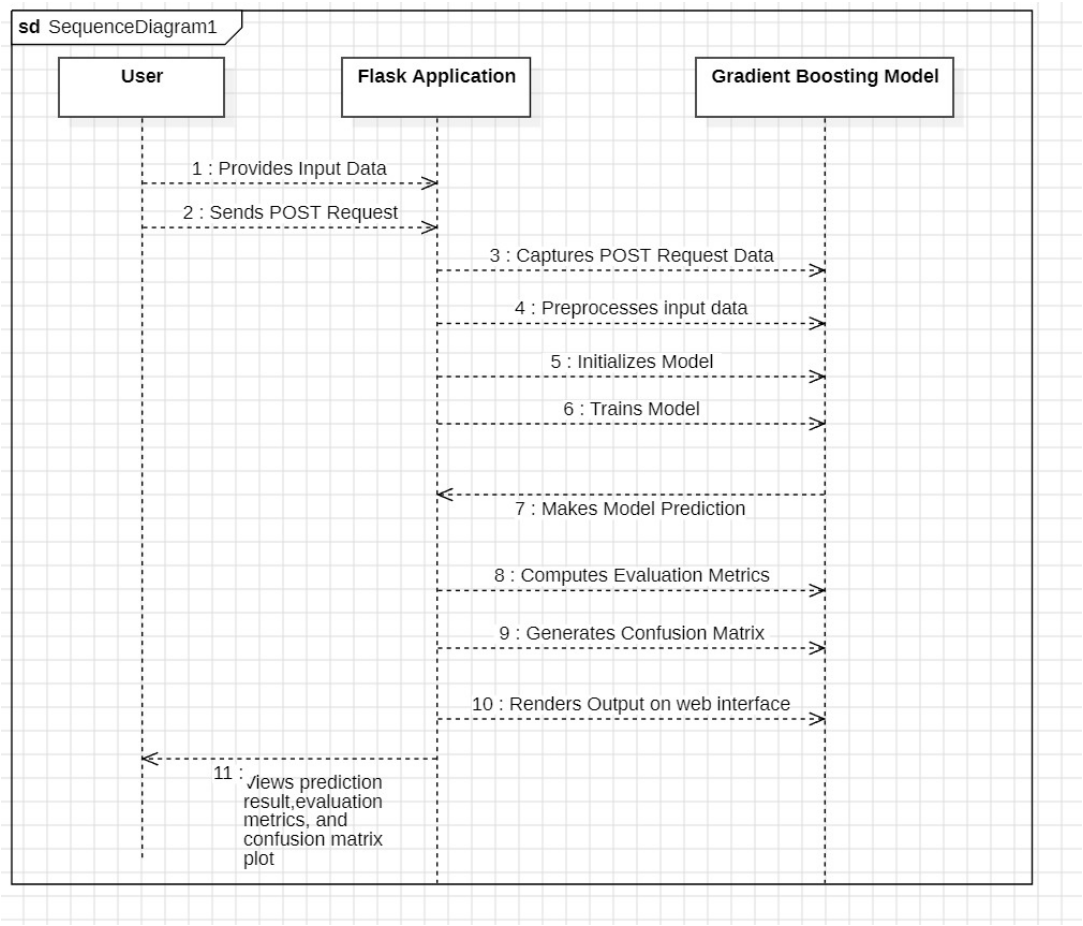


**Figure 6.4 Sequence Diagram**

# 7 System Requirements and specifications

## 7.1 Hardware Requirements

1. **Processor (CPU)**:
    - **Minimum Requirement**: Intel Core i5 or AMD Ryzen 5
    - **Recommended**: Intel Core i7 or AMD Ryzen 7 for better performance
2. **Memory (RAM)**:
    - **Minimum Requirement**: 4 GB RAM
    - **Recommended**: 8 GB or more for smoother performance
3. **Storage**:
    - **Minimum Requirement**: 50 GB HDD/SSD storage
    - **Recommended**: 128 GB SSD for faster data access
4. **Operating System**:
    - **Supported**: Windows 10, macOS 10.12+, Linux (Ubuntu 18.04 LTS or higher)
    - Ensure the chosen OS is compatible with Python and required libraries
5. **Network Connectivity**:
    - **Requirement**: Stable internet connection

## 7.2 Software Requirements

### 7.2.1 Python

Ensure you have Python version 3.7 or higher installed on your system. Python serves as the backbone for fake profile detection, providing the necessary tools and libraries. it is a high-level, interpreted programming language that was created by Guido van Rossum and first released in 1991. It's known for its simplicity and readability, making it a popular choice for beginners as well as experienced developers. Here are the features of python:

1. **Easy to Read and Write**: Python's syntax is designed to be clear and intuitive, which makes it easy to read and write code.

2. **Interpreted Language**: Unlike compiled languages like C or Java, Python is an interpreted language. This means that you can run Python code directly without the need for a separate compilation step.

3. **Dynamic Typing**: Python uses dynamic typing, which means you don't have to declare the data type of a variable when you define it. The type of a variable is determined at runtime.

4. **Rich Standard Library**: Python comes with a vast standard library that provides a wide range of modules and functions for tasks such as file I/O, networking, database interaction, and more.

5. **Object-Oriented**: Python supports object-oriented programming (OOP), allowing you to create and use classes and objects to organize and structure your code.

6. **Cross-Platform**: Python is a cross-platform language, meaning you can write code on one operating system (like Windows) and run it on another (like Linux or macOS) without any changes

**Python Versions:** There are two major versions of Python in use today: Python 2 and Python 3. Python 2 was the original version released in 2000, while Python 3 was introduced in 2008 with many improvements and new features. Python 2 reached its end of life in January 2020, and developers are encouraged to use Python 3 for all new projects.

### 7.2.2 Packages

**Pandas:**

Pandas is a powerful data manipulation and analysis library for Python. It offers data structures and functions designed to work with structured data, such as tables and time series. With Pandas, users can easily load, clean, transform, and analyze data from various sources, including CSV files, Excel spreadsheets, databases, and more. Its Data Frame and Series data structures provide a flexible and intuitive way to handle and manipulate data. Pandas also integrates seamlessly with other Python libraries, making it a fundamental tool for data scientists, analysts, and developers working on data-intensive projects.

**Scikit-learn:**

Scikit-learn is a comprehensive machine learning library for Python that provides a range of supervised and unsupervised learning algorithms. It offers tools for data mining and data analysis, including classification, regression, clustering, and dimensionality reduction. Scikit-learn is built on top of other Python libraries, such as

NumPy, SciPy, and matplotlib, leveraging their capabilities for efficient computation and data visualization. With its consistent and straightforward API, Scikit-learn simplifies the process of building and evaluating machine learning models. It also includes utilities for data preprocessing, model selection, and performance evaluation, making it a valuable resource for machine learning practitioners and researchers.

**NumPy:**

NumPy is a fundamental package for scientific computing in Python, providing support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays efficiently. NumPy's array object, ndarray, is essential for numerical operations in Python, offering advantages in terms of performance and memory efficiency compared to traditional Python lists. It also provides tools for linear algebra, Fourier analysis, and random number generation. NumPy serves as the foundation for many other scientific computing libraries in Python, including Pandas, Scikit-learn, and Matplotlib, making it an integral part of the Python data ecosystem.

**Matplotlib:**

Matplotlib is a comprehensive library for creating static, interactive, and animated visualizations in Python. It provides a MATLAB-like interface for plotting and visualization, enabling users to generate a wide range of charts, plots, histograms, and other graphical representations of data. Matplotlib works seamlessly with NumPy and Pandas, allowing users to plot data directly from arrays and DataFrames. Its extensive customization options and support for various output formats make it suitable for creating publication-quality figures and visualizations for both scientific research and data analysis projects.

**Seaborn:**

Seaborn is a statistical data visualization library based on Matplotlib. It provides a higher-level interface for creating informative and attractive statistical graphics. Seaborn simplifies the process of visualizing complex datasets by providing easy-to-use functions for plotting various types of statistical plots, including scatter plots, bar plots, box plots, and heatmaps. It offers built-in themes and color palettes to enhance the visual appeal of plots and facilitates the exploration of relationships between variables. Seaborn is designed to work seamlessly with Pandas DataFrames and integrates well with other Python libraries, making it a valuable tool for data visualization and exploratory data analysis.

### 7.2.3 Workspace

**Spyder:**

Spyder is an open-source integrated development environment (IDE) designed specifically for scientific computing, data analysis, and machine learning using Python. It provides a powerful and user-friendly interface that simplifies the process of writing, debugging, and executing Python scripts. With its comprehensive set of tools, Spyder facilitates interactive development by offering features like variable exploration, debugging capabilities, and integrated IPython consoles. These tools enable users to interactively develop and test code, making it an ideal environment for data scientists, researchers, and engineers working on data-intensive projects.

Beyond its basic code editing capabilities, Spyder offers an array of features tailored to scientific computing and data analysis tasks. It includes an advanced editor

with syntax highlighting, code completion, and real-time code analysis, which help streamline the coding process and minimize errors. Spyder's variable explorer provides a graphical interface for viewing and manipulating data variables, making it easier to understand and manage complex datasets. Additionally, Spyder's integrated IPython consoles allow users to execute code interactively, view results, and debug issues in real-time. These features combined make Spyder a powerful and versatile IDE for Python developers, particularly those focused on scientific computing and data analysis.

**Notepad:**

Notepad is a simple text editor that comes pre-installed on Windows operating systems. While it lacks the advanced features and functionalities of dedicated code editors or IDEs, Notepad serves as a basic tool for writing and editing plain text files, including HTML code. It provides a straightforward interface for manually writing HTML markup, making it accessible for beginners and users who prefer a minimalist approach. However, Notepad does not offer syntax highlighting, code completion, or other features commonly found in specialized code editors, which can make writing and managing HTML code more challenging and prone to errors for complex projects.

# 8 Implementation and Results

The execution of the Fake Profile Detection project can be divided into four major steps, each contributing to the overall process of building, training, and deploying the machine learning model.

## 8.1 Data Preprocessing

1. In the initial step, the project starts by reading the dataset containing information about both genuine and fake social media profiles.

2. The dataset is then split into feature variables (X) and target variable (y), where X contains various attributes like statuses count, followers count, friends count, etc., and y contains labels indicating whether a profile is genuine or fake.

3. **Label Encoding** is a technique that is used to convert categorical columns into numerical ones so that they can be fitted by machine learning models which only take numerical data. It is an important pre-processing step in a machine-learning project. In our project language labels like 'en', 'it', etc., which are represented as categorical variable are converted into numerical format, using map function and astype function, making it suitable for machine learning algorithms.

   **Input:** A pandas series containing categorical language labels like 'en', 'it', etc.
   **Output:** A new column in the DataFrame containing numerical codes corresponding to the language labels.

4. **Feature Extraction:** The account age in days is calculated by subtracting the 'created_at' date from the current date, and the result is stored in a new column

named 'account_age_days' and also the follower-to-friend ratio for each user profile is calculated.

**Input:** A pandas series containing datetime information representing the creation date of each user profile and Series containing the number of followers and friends for each user profile.

**Output:** Two new columns in the DataFrame x containing the account age in days for each user profile, follower-to-friend ratio.

5. **Feature Selection:** statuses count, followers count, friends count, favourites count, listed count, lang code, Follower-to-Friend Ratio, account_age_days features are selected.

   **Input:** The DataFrame containing the pre-processed and feature-engineered data, including the columns mentioned above.

   **Output:** A list of selected feature names that will be used to subset the DataFrame to only include these columns for model training.

## 8.2 Model Training

1. The prepared data is divided into training and testing sets using the and 70% of the data for training and 30% for testing is allocated.

   **Input:** The dataset containing the selected features and target variable (genuine or fake profiles).

   **Output:** Feature matrices containing 70% and 30% of the data, respectively, allocated for training and testing the model and target variables corresponding to the training and testing feature matrices, indicating genuine (0) or fake (1) profiles.

2. A Gradient Boosting Classifier model is then instantiated and trained on the training data using the fit method. This model is chosen for its ability to handle complex datasets and produce accurate predictions.

   **Input:** Gradient Boosting Classifier Parameters: learning_rate, n_estimators, max_depth.

   **Output:** A Gradient Boosting Classifier model that has been trained on the data using the fit method. This trained model has learned from the training data to make predictions on new, unseen data.

## 8.3 Evaluation

Post-training, the model's performance is evaluated using various metrics like accuracy, precision, recall, and F1 score. These metrics provide insights into the model's predictive capabilities and help assess its overall performance

1. **Accuracy**: Measures the proportion of correctly predicted instances among all instances.
2. **Precision**: Measures the proportion of true positive predictions among all positive predictions.
3. **Recall**: Measures the proportion of true positive predictions among all actual positives.
4. **F1 Score**: Harmonic mean of precision and recall, providing a balance between the two metrics.

**Confusion Matrix**: Table used to visualize the performance of an algorithm, showing true positives, true negatives, false positives, and false negatives.

**Input:** Actual target values from the testing dataset and Predicted target values generated by the trained model using the testing feature set.

**Output:** accuracy, precision, recall, and F1 score values.

**Table 8.1 Evaluation Metrics**

| S. No. | Metric | Value Obtained |
|---|---|---|
| 1 | Accuracy | 0.9965792474344356 |
| 2 | Precision | 1.0 |
| 3 | Recall | 0.9931972789115646 |
| 4 | F1 score | 0.9965870307167235 |

## 8.4 Prediction and Result Visualization in Web Interface

The trained model is utilized to make predictions on both the test dataset and user-submitted data via the Flask web application Home Page. The predictions are then used to evaluate the model's performance and provide insights into its predictive accuracy and reliability.

The Flask framework is utilized to develop a web application that serves as the user interface for the Fake Profile Detection system. Routes are defined to handle different client requests, including displaying the homepage, processing prediction requests, and presenting the model's evaluation metrics and results to the user.

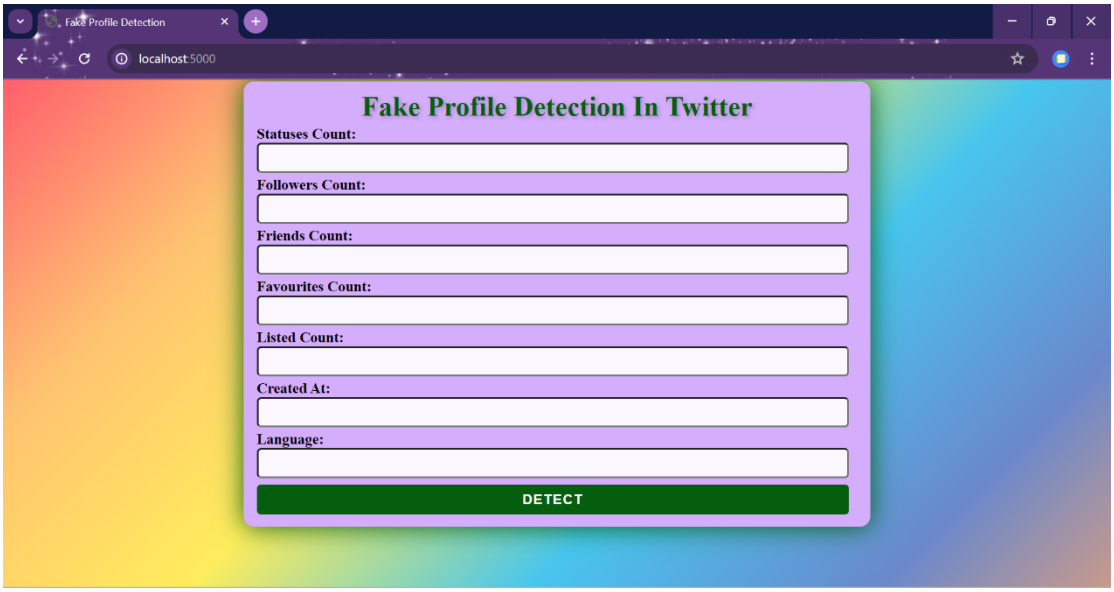


**Figure 8.1 Root Page**

The Figure 8.1 is the home page where the statuses count, followers count, friends count, favourites count, listed count, created at and language fields are present, which serves as main attributes for detecting fake profiles.

For user-submitted data, feature extraction is performed to preprocess the input, and the model predicts whether the profile is genuine or fake.

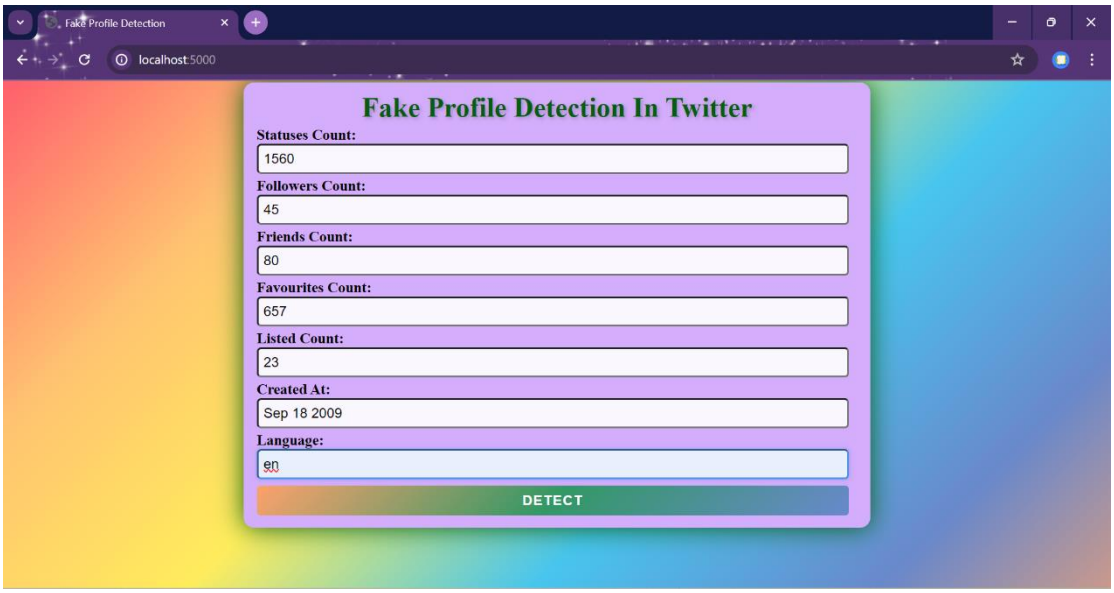


**Figure 8.2 Root Page after entering the details**

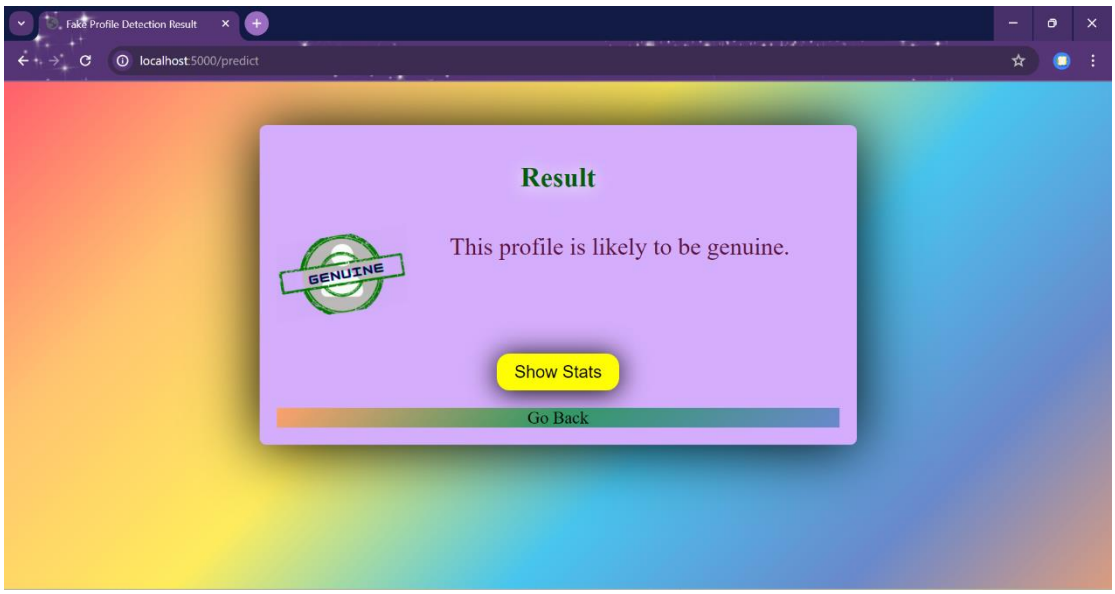


**Figure 8.3 Prediction page (output: genuine profile)**

After giving the details as in Figure 8.2 Root Page after entering the details the ML model will give appropriate prediction which in this case as Genuine Profile shown in Figure 8.3

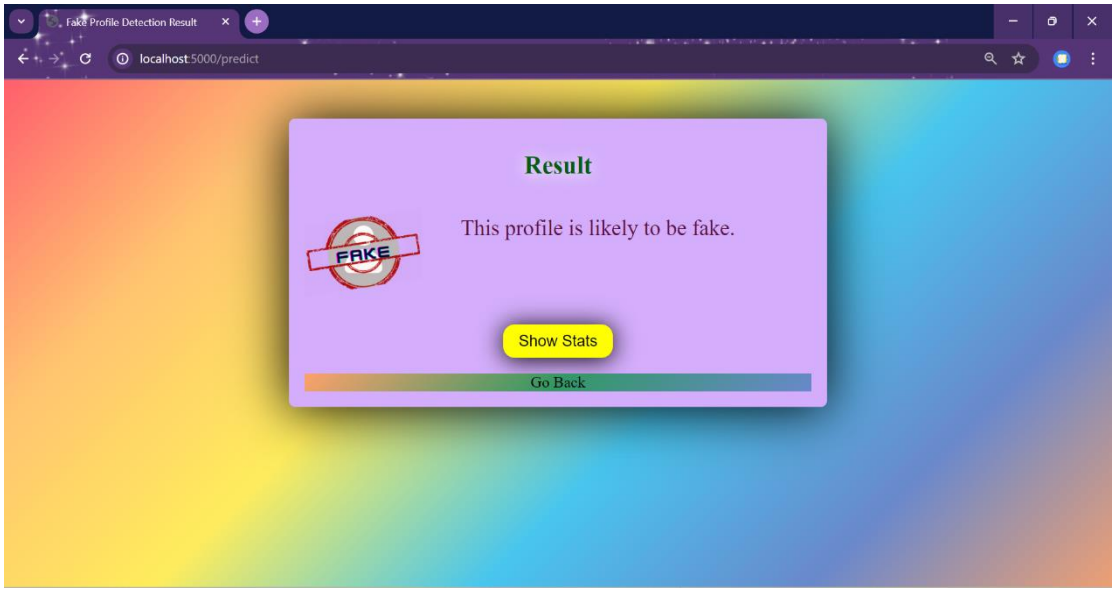


**Figure 8.4 Prediction page (output: fake profile)**

Similar to the Figure 8.3 After giving appropriate details that matches to the fake profile the prediction page will output as fake profile as shown in the Figure 8.4

Additionally, a confusion matrix is generated and visualized using seaborn and matplotlib libraries to provide a graphical representation of the model's performance on the test dataset.

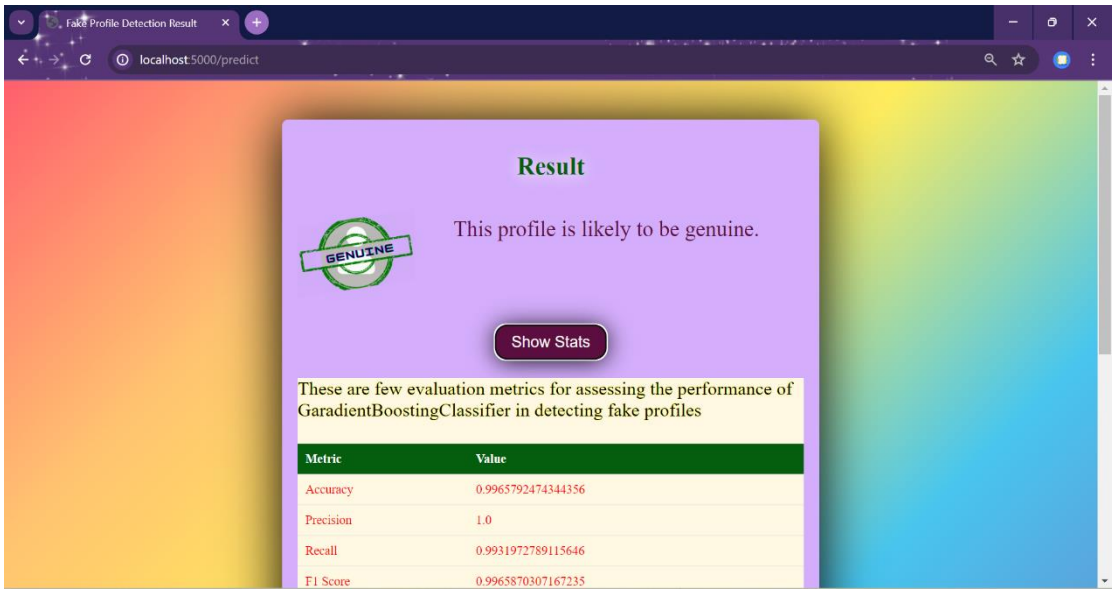


**Figure 8.5 Evaluation metrics 1**

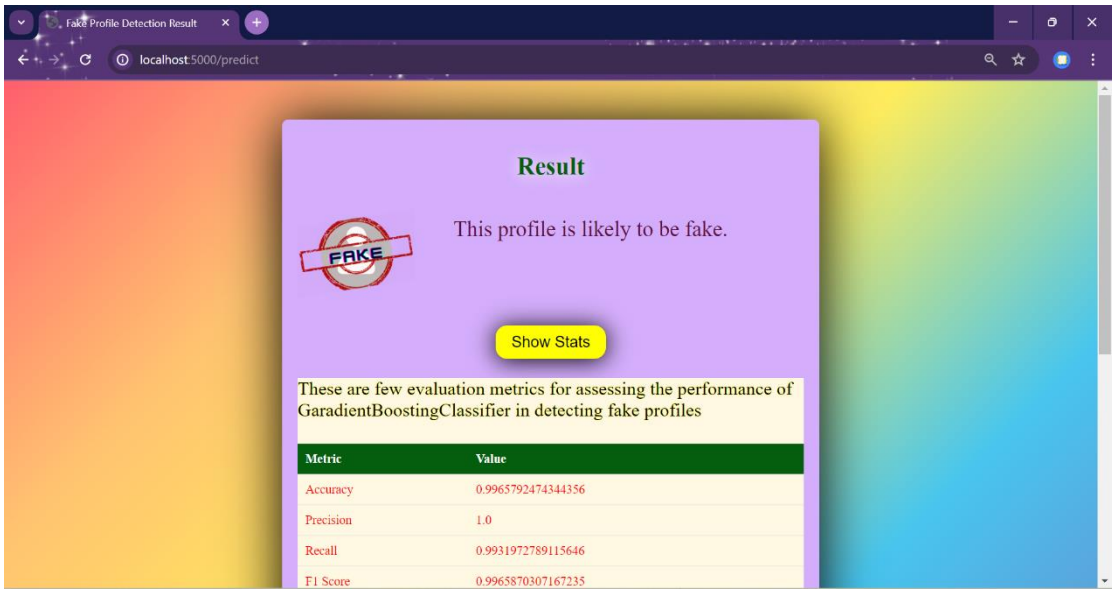


**Figure 8.6 evaluation metrics 2**

As shown in the figures Figure 8.5, Figure 8.6 when we click on the show stats button the evaluation metrics namely, accuracy, precision, recall and F1 score will be displayed.


**Figure 8.7 Confusion Matrix in interface**

As shown in the Figure 8.7 When we scroll down the evaluation metrics confusion matrix will be displayed.

# 9 Conclusion

The Fake Profile Detection project aimed to distinguish between genuine and fake social media profiles using machine learning techniques. Leveraging features like user activity, followers count, and account age, the Gradient Boosting Classifier was trained to predict the authenticity of user profiles. With an accuracy score reported at the end of the prediction process, the model demonstrated a high level of performance in classifying fake profiles correctly.

In addition to accuracy, the project also evaluated the model's precision, recall, and F1 score, providing a comprehensive understanding of its predictive capabilities. The confusion matrix visualization further illustrated the model's performance by showing the true positives, true negatives, false positives, and false negatives. Overall, the project successfully showcased the effectiveness of machine learning in detecting fake social media profiles, emphasizing the importance of data-driven approaches in ensuring online platform security and trustworthiness.

# 10 Future Scope

Integrating real-time data feeds from social media platforms is another promising avenue for future development. By continuously updating the model with new data, it could adapt to evolving trends and tactics employed by malicious actors. This adaptability would ensure that the detection system remains effective over time and can quickly respond to new challenges in the realm of fake profile creation and management.

Additionally, broadening the scope of the project to include multi-platform analysis could offer a more holistic approach to fake profile detection. Analyzing user activity across multiple social media platforms could provide cross-platform insights that help identify inconsistencies or irregularities in user behaviour. Such an approach would not only improve detection accuracy but also offer a comprehensive view of potential fake profiles across various online platforms, strengthening the overall security and trustworthiness of online communities.

# 11 References


[1] Tehlan, Pooja, M. R. and K. B. Komal, “A Spam Detection Mechanism in social Media using Soft Computing,” in *2019 6th International Conference on Computing for Sustainable Global Development (INDIACom)*, 2019.

[2] P. Krishnan, J. A. D. and P. Bhanu Prakash Reddy, “Finite Automata for Fake Profile Identification in Online Social Networks,” *Proc. Of ICICCS 2020,* 2020.

[3] B. Ananya, B. Ruchika, R. Ajay and A. Ginni, “Application of Machine Learning Techniques in Detecting Fake Profiles on Social Media,” in *Name: 9th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, Amity University, Noida, India, 2021.

[4] H. Preethi, J. Gojal, R. Chitra and S. Anithra, “2021 2nd Global Conference for Advancement in Technology (GCAT),” Bangole, India, 2021.

[5] S. M. and H. N., “A Hybrid Scheme for Detecting Fake Accounts in Facebook,” *International Journal of Recent Technology and Engineering (IJRTE),* vol. 7, no. 5S3, 2019.