A

Mini Project On

SOCIAL MEDIA FAKE ACCOUNT DETECTION USING MACHINE LEARNING

(Submitted in partial fulfillment of the requirements for the award of Degree)

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

By

K. ANJALI (197R1A0518)

BINDHU KOMMU(197R1A0521)

Under the Guidance of

G. POORNIMA

(Assistant Professor)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CMR TECHNICAL CAMPUS

UGC AUTONOMOUS

(Accredited by NAAC, NBA, Permanently Affiliated to JNTUH, Approved by AICTE, New Delhi)

Recognized Under Section 2(f) & 12(B) of the UGCAct.1956, Kandlakoya (V), Medchal Road, Hyderabad-501401.

2019-2023

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project entitled **"FAKE ACCOUNT DETECTION ON SOCIAL MEDIA USING MACHINE LEARNING"** being submitted by **K. ANJALI (197R1A0518) & BINDHU KOMMU(197R1A0521)** in partial fulfillment of the requirements for the award of the degree of B.Tech in Computer Science and Engineering to the Jawaharlal Nehru Technological University Hyderabad, is a record of bonafide work carried out by them under our guidance and supervision during the year 2022-23.

The results embodied in this thesis have not been submitted to any other University or Institute for the award of any degree or diploma.

G. **POORNIMA**
(Assistant Professor)
INTERNAL GUIDE

Dr. A. Raji Reddy
DIRECTOR

Dr. K. Srujan Raju
HOD

EXTERNAL EXAMINER

Submitted for viva voice Examination held on _

ACKNOWLEDGEMENT

Apart from the efforts of us, the success of any project depends largely on the encouragement and guidelines of many others. We take this opportunity to express our gratitude to the people who have been instrumental in the successful completion of this project.

We take this opportunity to express my profound gratitude and deep regard to my guide **G.POORNIMA ,** Assistant Professor for his exemplary guidance, monitoring and constant encouragement throughout the project work. The blessing, help and guidance given by him shall carry us a long way in the journey of life on which we are about to embark.

We also take this opportunity to express a deep sense of gratitude to the Project Review Committee (PRC) **Dr. Punyaban Patel, Ms. Shilpa, Dr.M . Subha Mastan Rao & J. Narasimharao** for their cordial support, valuable information and guidance, which helped us in completing this task through various stages.

We are also thankful to **Dr. K. Srujan Raju,** Head, Department of Computer Science and Engineering for providing encouragement and support for completing this project successfully.

We are obliged to **Dr. A. Raji Reddy,** Director for being cooperative throughout the course of this project. We also express our sincere gratitude to Sri. **Ch. Gopal Reddy,** Chairman for providing excellent infrastructure and a nice atmosphere throughout the course of this project.

The guidance and support received from all the members of **CMR Technical Campus** who contributed to the completion of the project. We are grateful for their constant support and help.

Finally, we would like to take this opportunity to thank our family for their constant encouragement, without which this assignment would not be completed. We sincerely acknowledge and thank all those who gave support directly and indirectly in the completion ofthis project.

ANJALI (197R1A0518)
BINDHU KOMMU (197R1A0521)

ABSTRACT

Now-a-days, Online Social Media is dominating the world in several ways. Day by day the number of users using social media is increasing drastically. The main advantage of online social media is that we can connect to people easily and communicate with them in a better way. This provided a new way of a potential attack, such as fake identity, false information, etc. A recent survey suggest that the number of accounts present in the social media is much greater than the users using it. This suggest that fake accounts have been increased in the recent years. Online social media providers face difficulty in identifying these fake accounts. The need for identifying these fake accounts is that social media is flooded with false information, advertisements, etc. Traditional methods cannot distinguish between real and fake accounts efficiently. Improvement in fake account creation made the previous works outdated. The new models created used different approaches such as automatic posts or comments, spreading fake news, false information or spam with advertisements to identify fake accounts. Due to the increase in the creation of the fake accounts different algorithms with different attributes are use. Previously use algorithms like naïvebayes, support vector machine, random forest has become inefficient in finding the fake accounts. In this research, we came up with an innovative method to identify fake accounts. We used ANN model, random forest algorithm, gradient boosting algorithm with decision tree containing three attributes. Those attributes are spam commenting, artificial activity and engagement .

LIST OF FIGURES/TABLES

LIST OF SCREENSHOTS

TABLE OF CONTENTS

1. INTRODUCTION

1. INTRODUCTION

1.1 PROJECT SCOPE

This project is titled “Fake account detection on social media using machine learning”. the project scope is to detect the fake accounts on social media. So many people are using online social media for spreading fake news, false information or spam with advertisements.So, it is very important to detect the fake accounts. so we are using ANN model,Gradient Boosting and Random Forest algorithms

1.2 PROJECT PURPOSE

This project has been developed to verify whether the account is fake or Geniue. Day by day the number of users using social media is increasing drastically. The main advantage of online social media is that we can connect to people easily and communicate with them in a better way. This provided a new way of a potential attack, such as fake identity, false information, etc.It is very important to detect the fake accounts. so we are using ANN model,Gradient Boosting and Random Forest algorithms.

1.3 PROJECT FEATURE

The main feature of this project is that we can easily identify the account is fake or genuine. ANN Model,random forest algorithm and gradient boosting algorithm is used in this project to detect fake accounts accurately. Identifying these fake accounts is that social media is flooded with false information ,advertisements. As proposed earlier, this detection method uses gradient boost,random forest algorithm extreme gradient boosting algorithms and ANN model to detect fake accounts

2.SYSTEM ANALYSIS

2. SYSTEM ANALYSIS

SYSTEM ANALYSIS

System Analysis is the important phase in the system development process. The System is studied to the minute details and analyzed. The system analyst plays an important role of an interrogator and dwells deep into the working of the present system. In analysis, a detailed study of these operations performed by the system and their relationships within and outside the system is done. A key question considered here is, “what must be done to solve the problem?” The system is viewed as a whole and the inputs to the system are identified. Once analysis is completed the analyst has a firm understanding of what is to be done.

2.1 PROBLEM DEFINITION

A general statement of fake accounts are so many peoples are misusing such as spreading false information, division and distructs etc., So to overcome this we are using ANN model random forest algorithm and gredient bboosting algorithm

2.2 EXISTING SYSTEM

The existing system use very fewer factors to decide weather an account is fake or not. The factors largely affect the way decision making occurs. When the number of factors is low, the accuracy of the decision making is reduced significantly. There is an exceptional improvement in fake account creation, which is unmatched by the software or application used to detect the fake account. Due to advancement in creation of fake account ,existing methods have turned obsolete. The most common algorithm used by fake account detection applications is the random forest algorithm . The algorithm has few downsides such as inefficiency to handle the categorical f increase in the number of trees, the algorithm’s time efficiency takes a hit.

2.2.1 DISADVANTAGES OF EXISTING SYSTEM

Following are the disadvantages of existing system:

- Fake accounts serve very different purposes: they are used to stirrup political quarrels, spread fake news, create division and distrust. they can promote disinformation about brands to mislead customers.
- In existing system uses very fewer factors to decide weather the account is fake or not .
- Inefficient and Unstable.
- Time consuming process

2.3 PROPOSED SYSTEM

The existing system uses random forest algorithm to identify the fake account. It is efficient when it has the correct inputs and when it has all the inputs. When some of the inputs are missing it becomes difficult for the algorithm to produce the output. To overcome such difficulties in the proposed systems we used a gradient boosting algorithm, naïve bayes algorithm and random forest algorithm . Gradient boosting algorithm is like random forest algorithm which uses decision trees as its main component. We also changed the way we find the fake accounts i.e., we introduced new methods to find the account. The methods used are spam commenting, engagement rate and artificial activity. These inputs are used to form decision trees that are used in the gradient boosting algorithm. This algorithm gives us an output even if some inputs are missing. This is the major reason for choosing this algorithm. Due to the use of this algorithm we were able to get highly accurate results.

2.3.1 ADVANTAGES OF THE PROPOSED SYSTEM

- The issues are privacy, online bulling, potential for misuse,trolling, etc. These are done mostly by using fake accounts.

- In this project, we came up with a framework through which we can detect a fake accounts using machine learning algorithms so that the social life of people also become secured

- It consume less time

2.4 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and a 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. Three key considerations involved in the feasibility analysis:

- EconomicFeasibility
- TechnicalFeasibility
- SocialFeasibility

2.4.1 ECONOMIC FEASIBILITY

The developing system must be justified by cost and benefit. Criteria to ensure that effort is concentrated on a project, which will give best, return at the earliest. One of the factors, which affect the development of a new system, is the cost it would require.

The following are some of the important financial questions asked during preliminary investigation:

- The costs conduct a full system investigation.
- The cost of the hardware and software.
- The benefits in the form of reduced costs or fewer costly errors.

Since the system is developed as part of project work, there is no manual cost to spend for the proposed system. Also all the resources are already available, it give an indication that the system is economically possible for development.

2.4.2 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. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

2.4.3 BEHAVIORAL FEASIBILITY

This includes the following questions:

- Is there sufficient support for the users?
- Will the proposed system cause harm?

The project would be beneficial because it satisfies the objectives when developed and installed. All behavioral aspects are considered carefully and conclude that the project is behaviorally feasible

2.5 HARDWARE & SOFTWARE REQUIREMENTS

2.5.1 HARDWARE REQUIREMENTS:

Hardware interfaces specify the logical characteristics of each interface between the software product and the hardware components of the system. The following are some hardware requirements.

- Processor : Intel Dual Core I5 and above
- Hard disk : 8GB and above
- RAM : 8GB and above
- Input devices : Keyboard, mouse.

2.5.2 SOFTWARE REQUIREMENTS:

Software Requirements specifies the logical characteristics of each interface and software components of the system. The following are some software requirements,

- Operating system : Windows 8 and above
- Languages : Python
- Tools : Python IDEL3.7 version, Anaconda - Jupyter, Spyder

3. **ARCHITECTURE**

3. ARCHITECTURE

3.1 PROJECT ARCHITECTURE

This project architecture shows the procedure followed for classification.

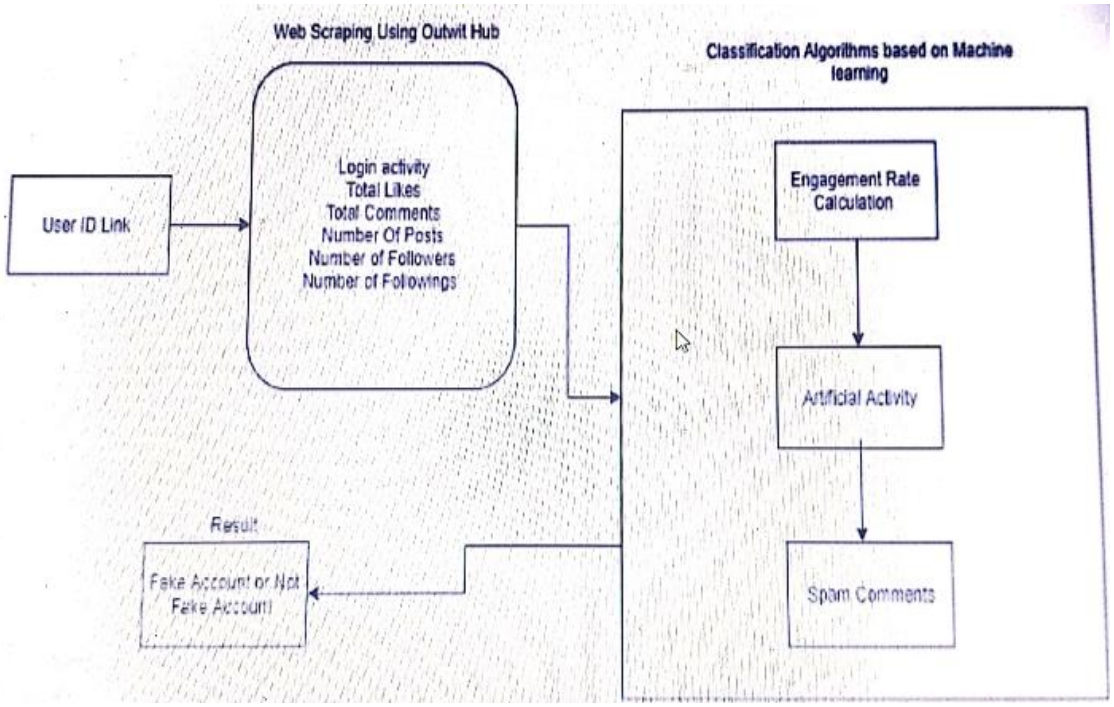


Figure 3.1: Project Architecture of Fake account detection

This project is totally based upon detecting the fake accounts. In this step, we combine all the data we extracted from the website. In this paper we mainly focus on engagement rate, artificial activity and spam comments. The data collected using web scraper is used to compute the values for the factors mentioned above. Using these factors different decision tress is formed. Using gradient boosting algorithm and with the formed decision trees fake accounts are detected.

3.2 MODULE

•DATA COLLECTION: This is the real step towards the real development of a machine learning model,collecting data. This isa critical step that will cascade in hoe goog the model will be, the more and better data that we get, the better our model will perform.

• DATASET: The dataset consists of 1338 individual data. There are 9 columns in the dataset.

1)User Id

2)No of Abuse Report

3)No of Rejected Friend Requests

4)No of friends

5)No of Followers

6)No of Likes To Unkown Accounts

7)No of Comments per Day

8)Fake Or Not Category

•DATA PREPARATION:

1)This module will transform the data. By gitting rid of missing data and removing some coloumns.

2)First we will create a list of column names that we want to keep or retain.

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

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

•MODEL SELECTION:

While creating a machine learning model, we need two dataset, one for training and other from testing. But now we have only one. So lets split this in two with a ratio of 80:20. we will divide the dataframes into feature colmn and label column. We will use gradient boosting algorith.

•ANALYSE AND PREDICTION:

In the actual dataset, we choose only 7 features:

1)User Id

2)No of Abuse Report

3)No of Rejected Friend Requests

4)No of Friends

5)No of Followers

6)No of Likes To Unknown Account

7)No Comments Per Day

•ACCURACY ON TESTSET: Accuracy on test would be calculated as per the dataset has given.

•SAVING THE TRAIN 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 model and dump the model into .pkl file

3.3USE CASE DIAGRAM

In the use case diagram, we have basically one actor who is the user in the trained model. A use case diagram is a graphical depiction of a user's possible interactions with a system. A use case diagram shows various use cases and different types of users the system has. The use cases are represented by either circles or ellipses. The actors are often shown as stick figures.

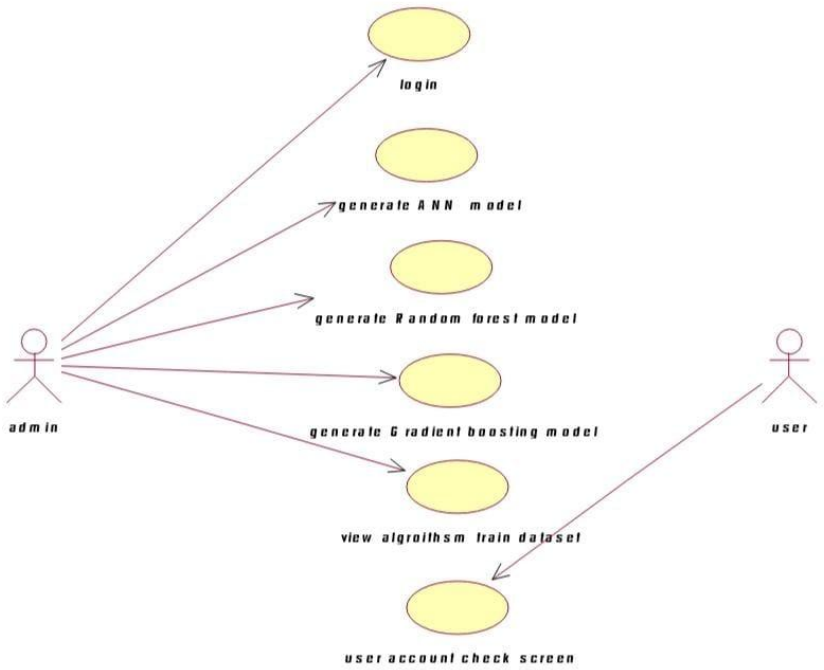


Fig.1: usecase diagram

Figure 3.3: Use Case Diagram

3.4CLASS DIAGRAM

Class diagram is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations(or methods), and the relationships among objects.

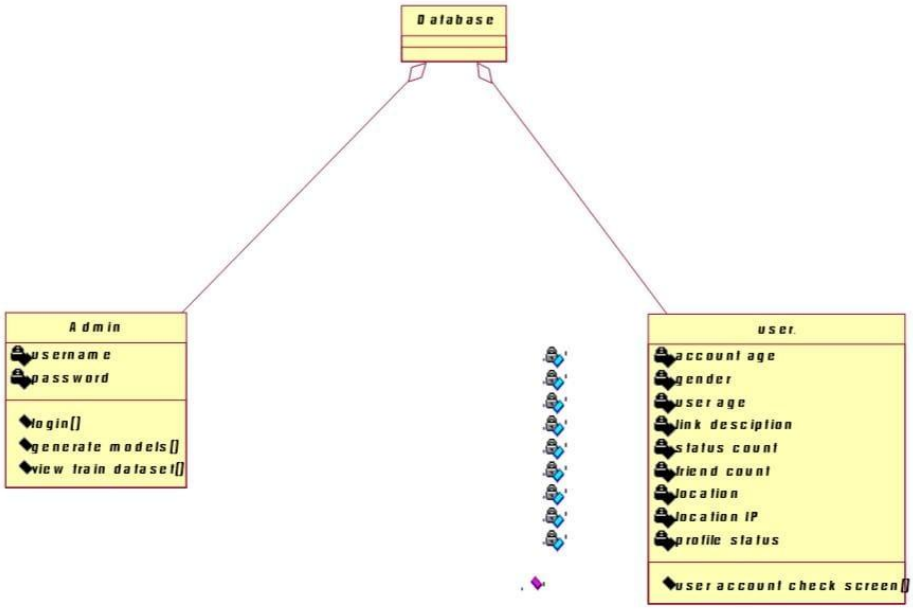
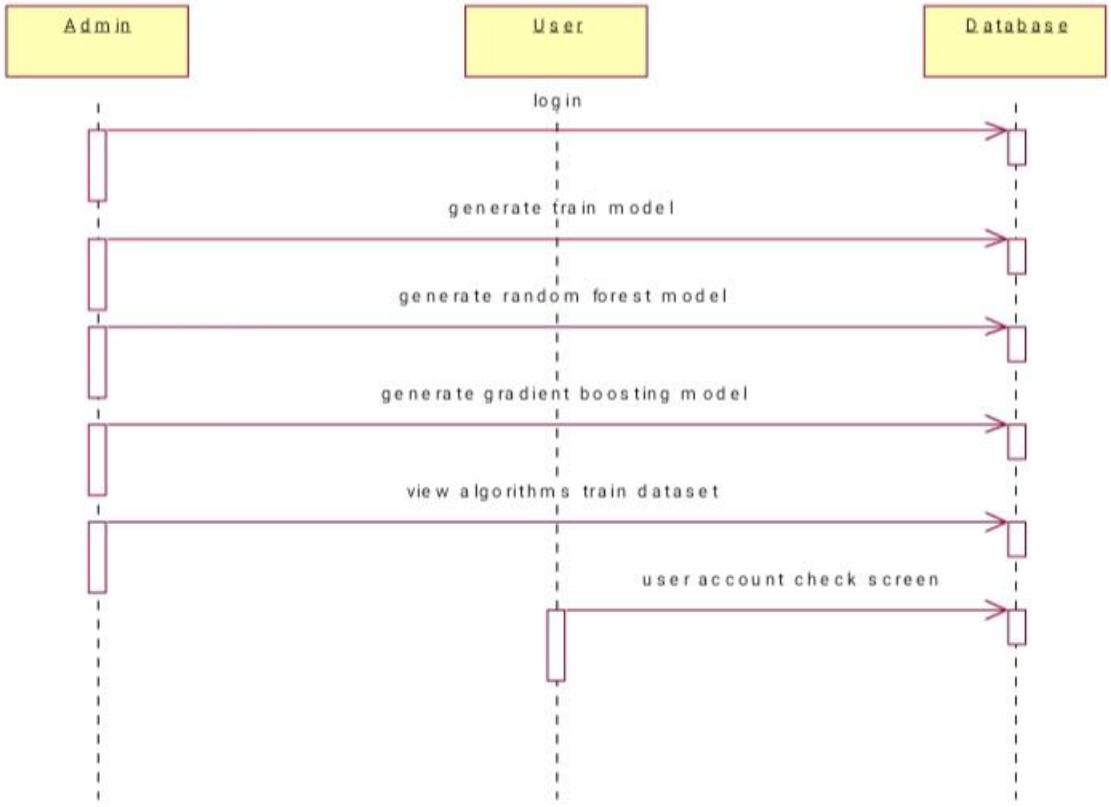


figure 3.4: Class Diagram for fake account detection

3.5 SEQUENCE DIAGRAM

A sequence diagram shows object interactions arranged in time sequence. It depicts the objects involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the logical view of the system under development.



3.6 ACTIVITY DIAGRAM

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. They can also include elements showing the flow of data between activities through one or more data stores.

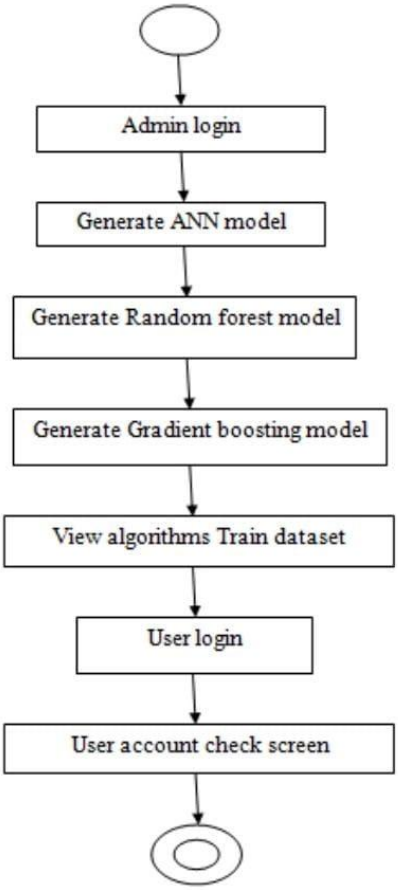


Figure 3.6: Activity Diagram

3.7 INTRODUCTION TO PYTHON

Script

What Is A Script?

Up to this point, I have concentrated on the interactive programming capability of Python. This is a very useful capability that allows you to type in a program and to have it executed immediately in an interactive mode.

Scripts are reusable

Basically, a script is a text file containing the statements that comprise a Python program. Once you have created the script, you can execute it over and over without having to retype it each time.

Scripts are editable

Perhaps, more importantly, you can make different versions of the script by modifying the statements from one file to the next using a text editor. Then you can execute each of the individual versions. In this way, it is easy to create different programs with a minimum amount of typing.

You will need a text editor

Just about any text editor will suffice for creating Python script files.

You can use *Microsoft Notepad, Microsoft WordPad, Microsoft Word,* or just about any word processor if you want to.

Difference between a script and a program Script:

Scripts are distinct from the core code of the application, which is usually written in a different language, and are often created or at least modified by the end-user. Scripts are often interpreted from source code or byte code, whereas the applications they control are traditionally compiled to native machine code.

Program:

The program has an executable form that the computer can use directly to execute the instructions.

The same program in its human-readable source code form, from which executable programs are derived (e.g., compiled).

Python

What is Python? Chances you are asking yourself this. You may have found this book because you want to learn to program but don't know anything about programming languages. Or you may have heard of programming languages like C, C++, C#, or Java and want to know what Python is and how it compares to "big name" languages. Hopefully I can explain it for you.

Python concepts

If you're not interested in the the hows and whys of Python, feel free to skip to the next chapter. In this chapter I will try to explain to the reader why I think Python is one of the best languages available and why it's a great one to start programming with.

- Open source general-purpose language.
- Object Oriented, Procedural, Functional.
- Easy to interface with C/ObjC/Java/Fortran.
- Easy-is to interface with C++ (via SWIG).
- Great interactive environment.

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.

Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python is Object-Oriented − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

Python is a Beginner's Language − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

History of Python Python was developed by Guido van Possum in the late eighties and early nineties at the

National Research Institute for Mathematics and Computer Science in the Netherlands. Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, Smalltalk, and UNIX shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Possum still holds a vital role in directing its progress.

Python Features

Python's features include −

- **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
- **A broad standard library** − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- **Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- **Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- **Extendable** − you can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- **Databases** − Python provides interfaces to all major commercial databases.

- **GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
- **Scalable** − Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below −

- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte-code for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.
- IT supports automatic garbage collection.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

Dynamic vs. Static

Types Python is a dynamic-typed language. Many other languages are static typed, such as C/C++ and Java. A static typed language requires the programmer to explicitly tell the computer what type of “thing” each data value is:

For example, in C if you had a variable that was to contain the price of something, you would have to declare the variable as a “float” type. This tells the compiler that the only data that can be used for that variable must be a floating point number, i.e. a number with a decimal point.If any other data value was assigned to that variable, the compiler would give an error when trying to compile the program.

Python, however, doesn’t require this. You simply give your variables names and assign values to them. The interpreter takes care of keeping track of what kinds of objects your

program is using. This also means that you can change the size of the values as you develop the program. Say you have another decimal number (a.k.a. a floating point number) you need in your program.

With a static typed language, you have to decide the memory size the variable can take when you first initialize that variable. A double is a floating point value that can handle a much larger number than a normal float (the actual memory sizes depend on the operating environment).

If you declare a variable to be a float but later on assign a value that is too big to it, your program will fail; you will have to go back and change that variable to be a double. With Python, it doesn't matter. You simply give it whatever number you want and Python will take care of manipulating it as needed. It even works for derived values.

Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory.

Based on the data type of a variable, the interpreter allocates memory and decides what can be stored in the reserved memory. Therefore, by assigning different data types to variables, you can store integers, decimals or characters in these variables.

Standard Data Types

The data stored in memory can be of many types. For example, a person's age is stored as a numeric value and his or her address is stored as alphanumeric characters. Python has various standard data types that are used to define the operations possible on them and the storage method for each of them.

Python has five standard data types −

- Numbers
- String
- List

- Tuple
- Dictionary

Python Numbers

Number data types store numeric values. Number objects are created when you assign a value to them.

Python Strings

Strings in Python are identified as a contiguous set of characters represented in the quotation marks. Python allows for either pairs of single or double quotes. Subsets of strings can be taken using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the string and working their way from -1 at the end.

Python Lists

Lists are the most versatile of Python's compound data types. A list contains items separated by commas and enclosed within square brackets ([]). To some extent, lists are similar to arrays in C. One difference between them is that all the items belonging to a list can be of different data type.

The values stored in a list can be accessed using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the list and working their way to end -1. The plus (+) sign is the list concatenation operator, and the asterisk (*) is the repetition operator.

Python Tuples

A tuple is another sequence data type that is similar to the list. A tuple consists of a number of values separated by commas. Unlike lists, however, tuples are enclosed within parentheses.

The main differences between lists and tuples are: Lists are enclosed in brackets ([ ]) and their elements and size can be changed, while tuples are enclosed in parentheses (( )) and cannot be updated. Tuples can be thought of as **read-only** lists.

Python Dictionary

Python's dictionaries are kind of hash table type. They work like associative arrays or hashes found in Perl and consist of key-value pairs. A dictionary key can be almost any Python type, but are usually numbers or strings. Values, on the other hand, can be any arbitrary Python object.

Dictionaries are enclosed by curly braces ({ }) and values can be assigned and accessed using square braces ([]).

Different modes in python

Python has two basic modes: normal and interactive. The normal mode is the mode where the scripted and finished .pie files are run in the Python

interpreter. Interactive mode is a command line shell which gives immediate feedback for each statement, while running previously fed statements in active memory. As new lines are fed into the interpreter, the fed program is evaluated both in part and in whole.

Python libraries

1. Requests. The most famous http library written by Kenneth remits. It's a must have for every python developer.
2. Scrappy. If you are involved in web scraping then this is a must have library for you. After using this library you won't use any other.
3. Python. A guy toolkit for python. I have primarily used it in place of tinder. You will really love it.
4. Pillow. A friendly fork of PIL (Python Imaging Library). It is more user friendly than PIL and is a must have for anyone who works with images.
5. SQLAlchemy. A database library. Many love it and many hate it. The choice is yours.
6. Beautiful Soup. I know it's slow but this xml and html parsing library is very useful for beginners.
7. Twisted. The most important tool for any network application developer. It has a very beautiful ape and is used by a lot of famous python developers.
8. Numbly. How can we leave this very important library? It provides some advance math functionalities to python.
9. Skippy. When we talk about numbly then we have to talk about spicy. It is a library of algorithms and mathematical tools for python and has caused many scientists to switch from ruby to python.
10. Matplotlib. A numerical plotting library. It is very useful for any data scientist or any data analyzer.
11. Pygmy. Which developer does not like to play games and develop them? This library will help you achieve your goal of 2d game development.
12. Piglet. A 3d animation and game creation engine. This is the engine in which the famous python port of mine craft was made

13. Pit. Another python GUI library. It is the same library in which the famous Bit torrent client is created.

14. Scaly. A packet sniffer and analyzer for python made in python.

15. Pywin32. A python library which provides some useful methods and classes for interacting with windows.

17. Notch. Natural Language Toolkit – I realize most people won't be using this one, but it's generic enough. It is a very useful library if you want to manipulate strings. But its capacity is beyond that. Do check it out.

18. Nose. A testing framework for python. It is used by millions of python developers. It is a must have if you do test driven development.

19. Simply. Simply can do algebraic evaluation, differentiation, expansion, complex numbers, etc. It is contained in a pure Python distribution.

20. I Python. I just can't stress enough how useful this tool is. It is a python prompt on steroids. It has completion, history, shell capabilities, and a lot more. Make sure that you take a look at it.

Numpy

Numpy's main object is the homogeneous multidimensional array. It is a table of elements (usually numbers), all of the same type, indexed by a tuple of positive integers. In numbly dimensions are called *axes*. The number of axes is *rank*.

- Offers Matlab-ish capabilities within Python
- Fast array operations
- 2D arrays, multi-D arrays, linear algebra etc.

Matplotlib

- High quality plotting library.

Python class and objects

These are the building blocks of OOP. Class creates a new object. This object can be anything, whether an abstract data concept or a model of a

physical object, e.g. a chair. Each class has individual characteristics unique to that class, including variables and methods. Classes are very powerful and currently "the big thing" in most programming languages. Hence, there are several chapters dedicated to OOP later in the book.

The class is the most basic component of object-oriented programming. Previously, you learned how to use functions to make your program do something.

Now will move into the big, scary world of Object-Oriented Programming (OOP). To be honest, it took me several months to get a handle on objects.

When I first learned C and C++, I did great; functions just made sense for me.

Having messed around with BASIC in the early '90s, I realized functions were just like subroutines so there wasn't much new to learn.

However, when my C++ course started talking about objects, classes, and all the new features of OOP, my grades definitely suffered.

Once you learn OOP, you'll realize that it's actually a pretty powerful tool. Plus many Python libraries and APIs use classes, so you should at least be able to understand what the code is doing.

One thing to note about Python and OOP: it's not mandatory to use objects in your code in a way that works best; maybe you don't need to have a full-blown class with initialization code and methods to just return a calculation. With Python, you can get as technical as you want.As you've already seen, Python can do just fine with functions. Unlike languages such as Java, you aren't tied down to a single way of doing things; you can mix functions and classes as necessary in the same program. This lets you build the code

Objects are an encapsulation of variables and functions into a single entity. Objects get their variables and functions from classes. Classes are essentially a template to create your objects. Here's a brief list of Python OOP ideas:

• The class statement creates a class object and gives it a name. This creates a new namespace.

• Assignments within the class create class attributes. These attributes are accessed by qualifying the name using dot syntax: ClassName.Attribute.

• Class attributes export the state of an object and its associated behavior. These attributes are shared by all instances of a class.

• Calling a class (just like a function) creates a new instance of the class.

This is where the multiple copies part comes in.

• Each instance gets ("inherits") the default class attributes and gets its own namespace. This prevents instance objects from overlapping and confusing the program.

• Using the term self identifies a particular instance, allowing for per-instance attributes. This allows items such as variables to be associated with a particular instance.

Inheritance

First off, classes allow you to modify a program without really making changes to it. To elaborate, by sub classing a class, you can change the behavior of the program by simply adding new components to it rather than rewriting the existing components. As we've seen, an instance of a class inherits the attributes of that class.

However, classes can also inherit attributes from other classes. Hence, a subclass inherits from a superclass allowing you to make a generic superclass that is specialized via subclasses. The subclasses can override the logic in a superclass, allowing you to change the behavior of your classes without changing the superclass at all.

Operator Overloads

Operator overloading simply means that objects that you create from classes can respond to actions (operations) that are already defined within Python, such as addition, slicing, printing, etc.Even though these actions can be implemented via class methods, using overloading ties the behavior closer

to Python's object model and the object interfaces are more consistent to Python's built-in objects, hence overloading is easier to learn and use.User-made classes can override nearly all of Python's built-in operation methods.

Exceptions

I've talked about exceptions before but now I will talk about them in depth. Essentially, exceptions are events that modify program's flow, either intentionally or due to errors. They are special events that can occur due to an error, e.g. trying to open a file that doesn't exist, or when the program reaches a marker, such as the completion of a loop.

Exceptions, by definition, don't occur very often; hence, they are the "exception to the rule" and a special class has been created for them. Exceptions are everywhere in Python.Virtually every module in the standard Python library uses them, and Python itself will raise them in a lot of different circumstances.

Here are just a few examples:

- Accessing a non−existent dictionary key will raise a Key Error exception.
- Searching a list for a non−existent value will raise a Value Error exception.
- Calling a non−existent method will raise an Attribute Error exception.
- Referencing a non−existent variable will raise a Name Error exception.
- Mixing data types without coercion will raise a Type Error exception.

One use of exceptions is to catch a fault and allow the program to continue working; we have seen this before when we talked about files. This is the most common way to use exceptions. When programming with the Python command line interpreter, you don't need to worry about catching exceptions. Your program is usually short enough to not be hurt too much if an exception occurs. Plus, having the exception occur at the command line is a quick and easy way to tell if your code logic has a problem.

However, if the same error occurred in your real program, it will fail and stop working. Exceptions can be created manually in the code by raising an exception.It operates exactly as a system-caused exceptions, except that the programmer is doing it on purpose. This can be for a number of reasons. One of

the benefits of using exceptions is that, by their nature, they don't put any overhead on the code processing. Because exceptions aren't supposed to happen very often, they aren't processed until they occur.

Exceptions can be thought of as a special form of the if/elf statements. You can realistically do the same thing with if blocks as you can with exceptions. However, as already mentioned, exceptions aren't processed until they occur; if blocks are processed all the time.Proper use of exceptions can help the performance of your program. The more infrequent the error might occur, the better off you are to use exceptions; using if blocks requires Python to always test extra conditions before continuing. Exceptions also make code management easier: if your programming logic is mixed in with error-handling if statements, it can be difficult to read, modify, and debug your program.

User-Defined Exceptions

I won't spend too much time talking about this, but Python does allow for a programmer to create his own exceptions. You probably won't have to do this very often but it's nice to have the option when necessary. However, before making your own exceptions, make sure there isn't one of the built-in exceptions that will work for you.They have been "tested by fire" over the years and not only work effectively, they have been optimized for performance and are bug-free. Making your own exceptions involves object-oriented programming, which will be covered in the next chapter.To make a custom exception, the programmer determines which base exception to use as the class to inherit from, e.g. making an exception for negative numbers or one for imaginary numbers would probably fall under the Arithmetic Error exception class.To make a custom exception, simply inherit the base exception and define what it will do.

Python modules

Python allows us to store our code in files (also called modules). This is very useful for more serious programming, where we do not want to retype a long function definition from the very beginning just to change one mistake. In

doing this, we are essentially defining our own modules, just like the modules defined already in the Python library.

To support this, Python has a way to put definitions in a file and use them in a script or in an interactive instance of the interpreter. Such a file is called a *module*; definitions from a module can be *imported* into other modules or into the *main* module.

Testing code

- As indicated above, code is usually developed in a file using an editor.
- To test the code, import it into a Python session and try to run it.
- Usually there is an error, so you go back to the file, make a correction, and test again.
- This process is repeated until you are satisfied that the code works.
- This entire process is known as the development cycle.
- There are two types of errors that you will encounter. Syntax errors occur when the form of some command is invalid.
- This happens when you make typing errors such as misspellings, or call something by the wrong name, and for many other reasons. Python will always give an error message for a syntax error.

Functions in Python

It is possible, and very useful, to define our own functions in Python. Generally speaking, if you need to do a calculation only once, then use the interpreter. But when you or others have need to perform a certain type of calculation many times, then define a function.

You use functions in programming to bundle a set of instructions that you want to use repeatedly or that, because of their complexity, are better self-contained in a sub-program and called when needed. That means that a function is a piece of code written to carry out a specified task. To carry out that specific task, the function might or might not need multiple inputs. When the task is carved out, the function can or cannot return one or more values.There are three types of functions in python: Help (), min (), print ().

Python Namespace

Generally speaking, a **namespace** (sometimes also called a context) is a naming system for making names unique to avoid ambiguity. Everybody knows a name spacing system from daily life, i.e. the naming of people in first name and family name (surname).

An example is a network: each network device (workstation, server, printer,) needs a unique name and address. Yet another example is the directory structure of file systems. The same file name can be used in different directories, the files can be uniquely accessed via the pathnames. Many programming languages use namespaces or contexts for identifiers. An identifier defined in a namespace is associated with that namespace.This way, the same identifier can be independently defined in multiple namespaces. (Like the same file names in different directories) Programming languages, which support namespaces, may have different rules that determine to which namespace an identifier belongs. Namespaces in Python are implemented as Python dictionaries, this means it is a mapping from names (keys) to objects (values). The user doesn't have to know this to write a Python program and when using namespaces.

Some namespaces in Python:

- **global names** of a module
- **local names** in a function or method invocation
- **built-in names**: this namespace contains built-in functions (e.g. abs(), camp(), ...) and built-in exception names

Garbage Collection

Garbage Collector exposes the underlying memory management mechanism of Python, the automatic garbage collector. The module includes functions for controlling how the collector operates and to examine the objects known to the system, either pending collection or stuck in reference cycles and unable to be freed.

Python XML Parser

XML is a portable, open source language that allows programmers to develop applications that can be read by other applications, regardless of operating system and/or developmental language.

What is XML?

The Extensible Markup Language XML is a markup language much like HTML or SGML. This is recommended by the World Wide Web Consortium and available as an open standard. XML is extremely useful for keeping track of small to medium amounts of data without requiring a SQL-based backbone. XML Parser Architectures and APIs the Python standard library provides a minimal but useful set of interfaces to work with XML.The two most basic and broadly used APIs to XML data are the SAX and DOM interfaces. Simple API for XML SAX: Here, you register callbacks for events of interest and then let the parser proceed through the document. This is useful when your documents are large or you have memory limitations, it parses the file as it reads it from disk and the entire file is never stored in memory. Document Object Model DOM API : This is a World Wide Web Consortium recommendation wherein the entire file is read into memory and stored in a hierarchical tree − based form to represent all the features of an XML document.

SAX obviously cannot process information as fast as DOM can when working with large files. On the other hand, using DOM exclusively can really kill your resources, especially if used on a lot of small files.SAX is read-only, while DOM allows changes to the XML file. Since these two different APIs literally complement each other, there is no reason why you cannot use them both for large projects.

Python Web Frameworks

A web framework is a code library that makes a developer's life easier when building reliable, scalable and maintainable web applications.

Why are web frameworks useful?

Web frameworks encapsulate what developers have learned over the past twenty years while programming sites and applications for the web. Frameworks make it easier to reuse code for common HTTP operations and to structure projects so other developers with knowledge of the framework can quickly build and maintain the application.

Common web framework functionality

Frameworks provide functionality in their code or through extensions to perform common operations required to run web applications. These common operations include:

1. URL routing
2. HTML, XML, JSON, and other output format tinplating
3. Database manipulation
4. Security against Cross-site request forgery (CSRF) and other attacks
5. Session storage and retrieval

Not all web frameworks include code for all of the above functionality. Frameworks fall on the spectrum from executing a single use case to providing every known web framework feature to every developer. Some frameworks take the "batteries-included" approach where everything possible comes bundled with the framework while others have a minimal core package that is
amenable to extensions provided by other packages.

Comparing web frameworks

There is also a repository called compare-python-web-frameworks where the same web application is being coded with varying Python web frameworks, tinplating engines and object.

Web framework resources

- When you are learning how to use one or more web frameworks it's helpful to have an idea of what the code under the covers is doing.
- Frameworks is a really well done short video that explains how to choose between web frameworks. The author has some particular opinions about what should be in a framework. For the most part I agree although I've found sessions and database ORMs to be a helpful part of a framework when done well.

- What is a web framework? Is an in-depth explanation of what web frameworks are and their relation to web servers?
- Jingo vs. Flash vs. Pyramid: Choosing a Python web framework contains background information and code comparisons for similar web applications built in these three big Python frameworks.
- This fascinating blog post takes a look at the code complexity of several Python web frameworks by providing visualizations based on their code bases.
- Python's web frameworks benchmarks is a test of the responsiveness of a framework with encoding an object to JSON and returning it as a response as well as retrieving data from the database and rendering it in a template. There were no conclusive results but the output is fun to read about nonetheless.
- What web frameworks do you use and why are they awesome? Is a language agnostic Reedit discussion on web frameworks? It's interesting to see what programmers in other languages like and dislike about their suite of web frameworks compared to the main Python frameworks.
- This user-voted question & answer site asked "What are the best general purpose Python web frameworks usable in production?" The votes aren't as important as the list of the many frameworks that are available to Python developers.

4.IMPLEMENTATION

4.1 SAMPLE CODE

```
from django.shortcuts import render
from django.template import RequestContext
from django.contrib import messages
from django.http import HttpResponse
import pandas as pd
from sklearn.model_selection import train_test_split
from keras.models import Sequential
from keras.layers.core import Dense,Activation,Dropout
from keras.callbacks import EarlyStopping
from sklearn.preprocessing import OneHotEncoder
from keras.optimizers import Adam
from sklearn.metrics import accuracy_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier

global model

def index(request):
    if request.method == 'GET':
        return render(request, 'index.html', {})

def User(request):
    if request.method == 'GET':
        return render(request, 'User.html', {})



def Admin(request):
    if request.method == 'GET':
        return render(request, 'Admin.html', {})
```

```
def AdminLogin(request):
    if request.method == 'POST':
      username = request.POST.get('username', False)
      password = request.POST.get('password', False)
      if username == 'admin' and password == 'admin':
       context= {'data':'welcome '+username}
       return render(request, 'AdminScreen.html', context)
      else:
       context= {'data':'login failed'}
       return render(request, 'Admin.html', context)



def importdata():
    balance_data = pd.read_csv('C:/Users/navee/OneDrive/Desktop/Use of Artificial Neural Networks to Identify FakeProfiles/Profile/dataset/dataset.txt')
    balance_data = balance_data.abs()
    rows = balance_data.shape[0]  # gives number of row count
    cols = balance_data.shape[1]  # gives number of col count
    return balance_data


def splitdataset1():
    data = pd.read_csv('C:/Users/navee/OneDrive/Desktop/Use of Artificial Neural Networks to Identify FakeProfiles/Profile/dataset/dataset.txt')
    X = data.values[:, 0:8]
    y= data.values[:, 8]


    train_x, test_x, train_y, test_y = train_test_split(X, y, test_size=0.2)
    return train_x, test_x, train_y, test_y


def splitdataset(balance_data):
    X = balance_data.values[:, 0:8]
    y_= balance_data.values[:, 8]
```

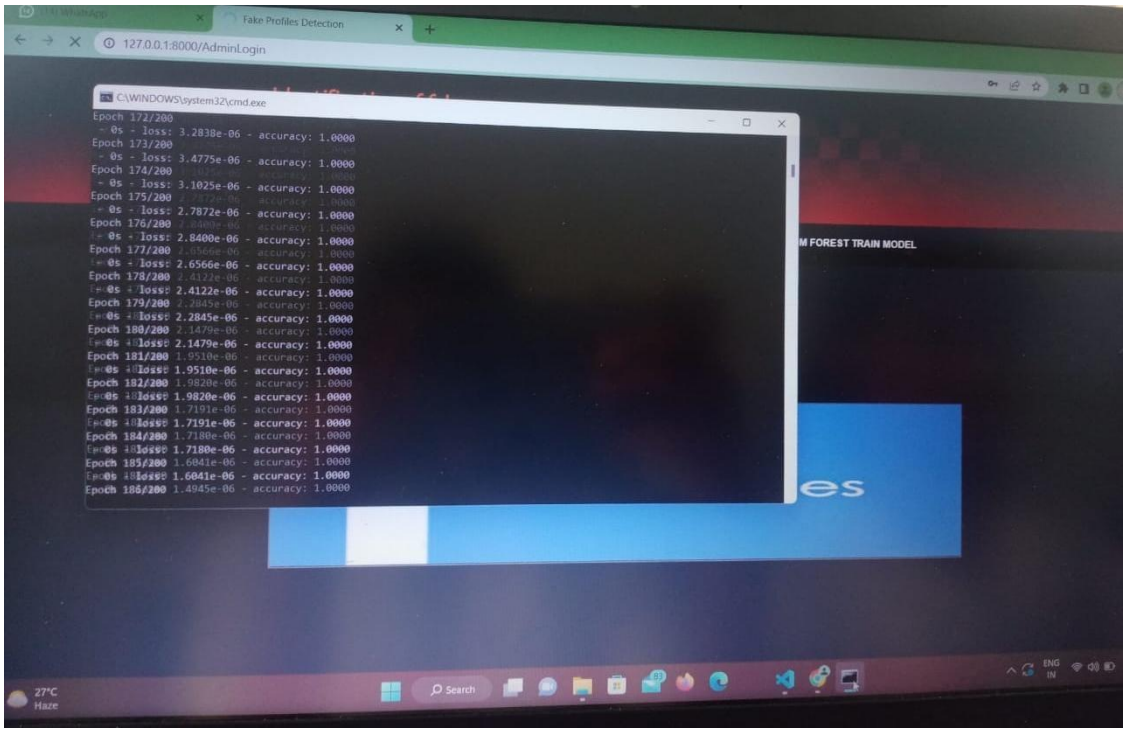
```
    y_ = y_.reshape(-1, 1)
    encoder = OneHotEncoder(sparse=False)
    Y = encoder.fit_transform(y_)
    print
    train_x, test_x, train_y, test_y = train_test_split(X, Y, test_size=0.2)
    return train_x, test_x, train_y, test_y


def UserCheck(request):
    if request.method == 'POST':
      data = request.POST.get('t1', False)
      input = 'Account_Age,Gender,User_Age,Link_Desc,Status_Count,Friend_Count,Location,Location_IP\n';
      input+=data+"\n"
      f = open("C:/Users/navee/OneDrive/Desktop/Use of Artificial Neural Networks to Identify FakeProfiles/Profile/dataset/test.txt", "w")
      f.write(input)
      f.close()
      test = pd.read_csv('C:/Users/navee/OneDrive/Desktop/Use of Artificial Neural Networks to Identify FakeProfiles/Profile/dataset/test.txt')
      test = test.values[:, 0:8]
      predict = model.predict_classes(test)
      print(predict[0])
      msg = ''
      if str(predict[0]) == '0':
        msg = "Given Account Details Predicted As Genuine"
      if str(predict[0]) == '1':
        msg = "Given Account Details Predicted As Fake"
      context= {'data':msg}
      return render(request, 'User.html', context)


def GenerateModel(request):
    global model
    data = importdata()
```

```
    train_x, test_x, train_y, test_y = splitdataset(data)
    model = Sequential()
    model.add(Dense(200, input_shape=(8,), activation='relu', name='fc1'))
    model.add(Dense(200, activation='relu', name='fc2'))
    model.add(Dense(2, activation='softmax', name='output'))
    optimizer = Adam(lr=0.001)
    model.compile(optimizer, loss='categorical_crossentropy', metrics=['accuracy'])
    print('CNN Neural Network Model Summary: ')
    print(model.summary())
    model.fit(train_x, train_y, verbose=2, batch_size=5, epochs=200)
    results = model.evaluate(test_x, test_y)
    ann_acc = results[1] * 100
    print(ann_acc)
    context= {'data':'ANN Accuracy : '+str(ann_acc)}
    return render(request, 'AdminScreen.html', context)


def Gradient(request):
    global model
    train_x, test_x, train_y, test_y = splitdataset1()
    gra = GradientBoostingClassifier(n_estimators=100, learning_rate=1.0,max_depth=1,
random_state=0)
    gra.fit(train_x, train_y)
    var = gra.predict(test_x)
    acc = accuracy_score(test_y, var)
    print(acc)
    context = {'data':'Gradient Boosting Accuracy : '+str(acc)}
    return render(request, 'AdminScreen.html', context)


def Random(request):
    global model
    train_x, test_x, train_y, test_y = splitdataset1()
    gra = RandomForestClassifier(max_depth=2, random_state=0)
    gra.fit(train_x, train_y)
```

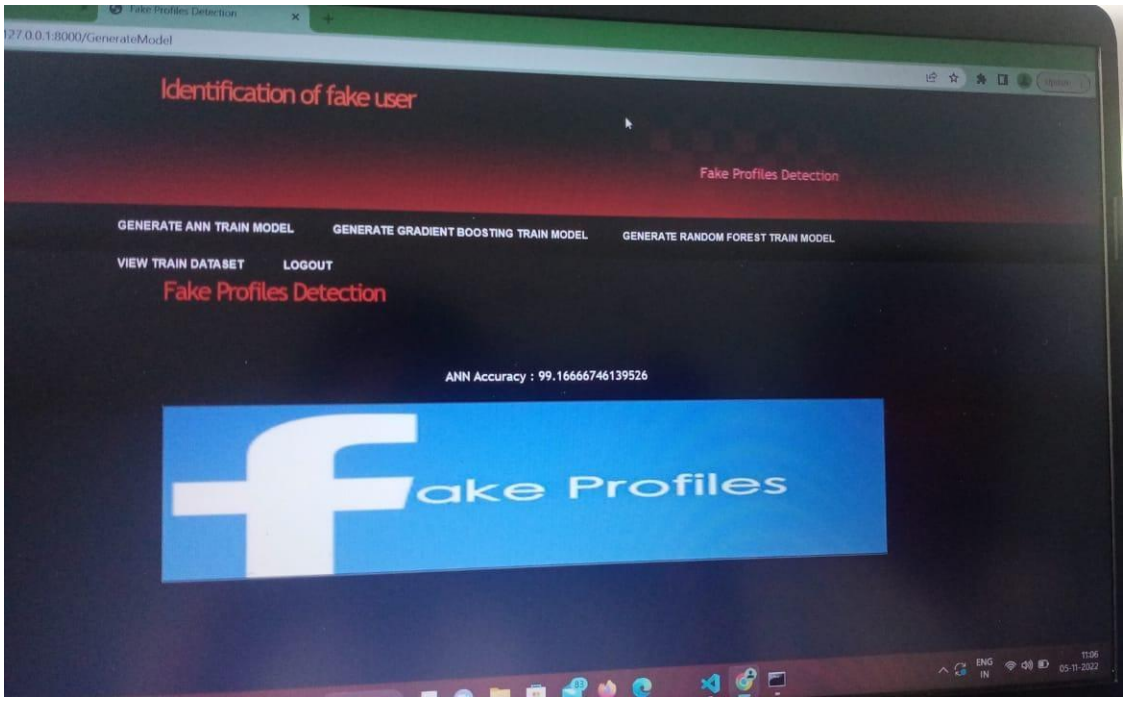
```
    var = gra.predict(test_x)
    acc = accuracy_score(test_y, var)
    print(acc)
    context = {'data':'Random Forest Accuracy : '+str(acc)}
    return render(request, 'AdminScreen.html', context)
```

```
def ViewTrain(request):
    if request.method == 'GET':
        strdata = '<table border=1 align=center width=100%><tr><th><font size=4 color=white>Account Age</th><th><font size=4 color=white>Gender</th><th><font size=4 color=white>User Age</th><th><font size=4 color=white>Link Description</th> <th><font size=4 color=white>Status Count</th><th><font size=4 color=white>Friend Count</th><th><font size=4 color=white>Location</th><th><font size=4 color=white>Location IP</th><th><font size=4 color=white>Profile Status</th></tr><tr>'
        data = pd.read_csv('C:/Users/navee/OneDrive/Desktop/Use of Artificial Neural Networks to Identify FakeProfiles/Profile/dataset/dataset.txt')
        rows = data.shape[0]  # gives number of row count
        cols = data.shape[1]  # gives number of col count
        for i in range(rows):
            for j in range(cols):
                strdata+='<td><font size=3 color=white>'+str(data.iloc[i,j])+'</font></td>'
            strdata+='</tr><tr>'
        context= {'data':strdata}
        return render(request, 'ViewData.html', context)
```

5. **SCREENSHOTS**

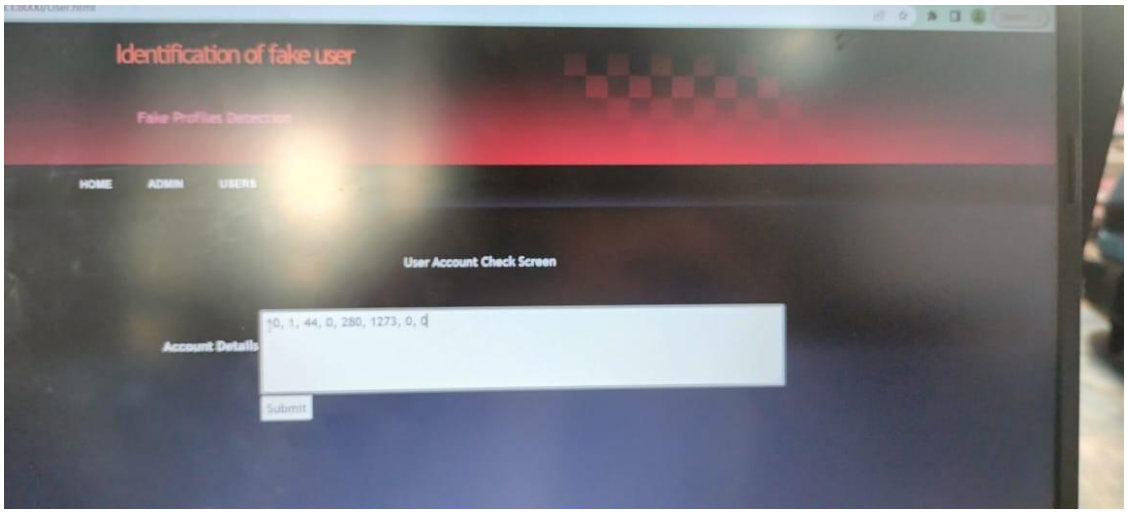


Screenshot 5.1: train the data

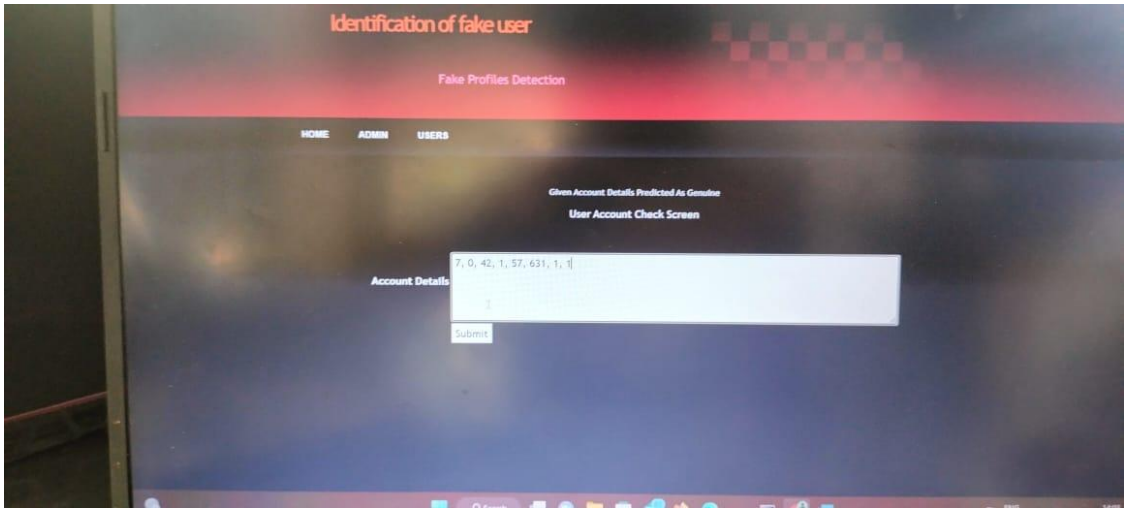


S

screenshot 5.2: train accuracy



Screenshot 5.3: inputs



Screenshot 5.4: fake are jenuine

6. **TESTING**

6. TESTING

6.1 INTRODUCTION TO 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, subassemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

6.2 TYPES OF TESTING

6.2.1 UNIT TESTING

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.

6.2.2 **INTEGRATION TESTING**

Integration tests are designed to test integrated software components to determine if they actually run as one program. Integration tests demonstrate that although the components were individually satisfactory, 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.

6.2.3 **FUNCTIONAL TESTING**

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

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid : identified classes of invalid input must Input be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked. Organization and preparation of functional tests is focused on requirements, key functions, or special test cases.

6.3 **TEST CASES**

6.3.1 CLASSIFICATION

Test case ID	Test case name	Purpose	Input	Output
1	Detecting fake accounts	To detect fake accounts	The user gives the input as account age, friends,likes, comments.	An output is It will show whether the account is fake or genuine
2	Detecting fake accounts	To detect fake accounts	The user gives input as account age, friends,likes, comments.	An output isIt will show account is fake or genuine.

7. CONCLUSION

7. CONCLUSION & FUTURE SCOPE

7.1 PROJECT CONCLUSION

In this research, We have come up with an ingenious way to detect fake accounts on OSNs By using machine learning algorithms to its full extent, we have eliminated the need for manual prediction of a fake account, which needs a lot of human resources and is also a time-consuming process. Existing systems have become obsolete due to the advancement in the creation of fake accounts. The factors that the existing system relayed upon is unstable. In this research, we used stable factors such as engagement rate, artificial activity to increase the accuracy of the prediction.

7.2 FUTURE SCOPE

The potential idea behind the idea of the project is by using the ANN model, random forest algorithm and gradient boosting algorithm we can easily decide whether the accout is fake or genuine.The additional features of this project, it is sufficient, stable and fast.

8.BIBLIOGRAPHY

8. REFERENCES

[1] Rao, K. Sreenivasa, N. Swapna, and P. Praveen Kumar. "Educational data mining for student placement prediction using machine learning algorithms." Int. J. Eng. Technol. Sci 7.1.2 (2018): 43-46.

[2] Y. Boshmaf, D. Logothetis, G. Siganos, J. Lería, J. Lorenzo, M. Ripeanu, K. Beznosov, H. Halawa, "Íntegro: Leveraging victim prediction for robust fake account detection in large scale osns", Computers & Security, vol. 61, pp. 142-168, 2016.

[3] N. Singh, T. Sharma, A. Thakral and T. Choudhury, "Detection of Fake Profile in Online Social
Networks Using Machine Learning," 2018 International Conference on Advances in Computing and Communication Engineering (ICACCE), Paris, 2018, pp. 231-234.

[4] D. M. Freeman, "Detecting clusters of fake accounts in online social networks", 8th ACM Workshop on Artificial Intelligence and Security, pp. 91-101.

8.1 GITHUB LINK

https://github.com/katakamAnjali/socialmediafakeaccountdetectionusingML