A
MAJOR PROJECT REPORT

ON

DEEP LEARNING BASED BLOOD GROUP DETECTION USING FINGERPRINT

Submitted to in partial fulfillment of the

requirements for the reward of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

SUBMITTED

BY

P.JHANSI RANI

(22M85A0503)

Under the Esteemed Guidance of

P. VISHNU VARDHAN M.Tech

Assistant Professor, CSE

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

SANA ENGINEERING COLLEGE

(Accredited by NAAC, Approved by AICTE, New Delhi & Affiliated to JNTUH, Hyderabad.)

N.H. – 65, KODAD – 508206, SURYAPET DIST. TELANGANA.

2025

SANA ENGINEERING COLLEGE

(Accredited by NAAC, Approved by AICTE ,New Delhi & Affiliated to JNTUH, Hyderabad.)

N.H. – 65, KODAD – 508206, SURYAPET DIST. TELANGANA

2025

CERTIFICATE

This is to certify that the dissertation work entitled **"DEEP LEARNING BASED BLOOD GROUP DETECTION USING FINGERPRINT"**is the bonafide work done by **P.JHANSI RANI(22M85A0503),**submitted in partial fulfillment of the requirements for the award of **"BACHELOR OF TECHNOLOGY"** in **"COMPUTER SCIENCE AND ENGINEERING"** during the academic year 2024-25. This is further certified that the Major Project presented under our guidance and report made have not submitted elsewhere for the award of any other degree or diploma.

INTERNAL GUIDE
P.VISHNU VARDHAN **M .Tech**

HEAD OF THE DEPARTMENT
MD. JANI PASHA **M. Tech**

EXTERNAL EXAMINER

PRINCIPAL
DR.CH.S.NAGA PRASAD
BE,M.Tech,Phd,LMISTE

ACKNOWLEDGEMENT

First of all, I’m grateful to the Almighty God for establishing us to complete this project. This report will certainly not be completed without due acknowledgements paid to all those who have helped us in doing our project work.

It is great pleasure to us to express our sincere thanks to our management **Mr.MD.NAZEERUDIN** chairman, **SANA GROUP OF INSTITUTIONS** for providing the required infrastructure and lab facility which contributed a lot in carrying this project.

I derive great pleasure in expressing my sincere gratitude to our principal **Dr.CH.S.NAGA PRASAD PRINCIPAL & PROFESSOR (BE, M.Tech, Ph-D, LMISTE)** .For his timely suggestions which helped us to complete this work successfully.

It is privilege to thank our head of the Computer Science and Engineering Department, **Mr.MD.JANI PASHA M.Tech**, his encouragement during the progress of this project work and moral support, kind attention and valuable guidance to us throughout this project work.

I express my sincere thanks to our guide **MR. P. VISHNU VARDHAN** for giving us moral support, kind attention and valuable guidance to us throughout this project work.

I’m thank full to both teaching and non-teaching staff members of **COMPUTER SCIENCE AND ENGINEERING DEPARTMENT** for their kind co-operation and all sorts of help to our bring out this project work successfully.

I would like to thank our parents & our friends for being supportive all the time and we are very much obliged to them.

P.JHANSI RANI

(22M85A0503)

DECLARATION

I hereby declare that the Major Project report entitled "**DEEP LEARNING BASED BLOOD GROUP DETECTION USING FINGERPRINT**" Submitted to Department of **Computer Science Engineering** **SANA ENGINEERING COLLEGE, KODAD** affiliated to **JNTUH Hyderabad** in a partial fulfillment of the requirements for the award of the Bachelor Technology in **COMPUTER SCIENCE ENGINEERING** is my original work under the supervision and guidance of

Mr. P.VISHNU VARDHAN M.Tech Assistant professor in **Sana Engineering College**. It has not previously formed the basis for the award of any Degree, Diploma, Associate ship, fellowship or other similar title.

P.JHANSIRANI

(22M85A0503)

ABSTRACT

Blood group identification is a fundamental aspect of medical diagnostics, traditionally performed through invasive methods such as blood sampling and laboratory testing. This project proposes a novel, non-invasive approach to predict an individual's blood group using fingerprint pattern analysis and machine learning techniques. The hypothesis is based on emerging studies that suggest potential correlations between dermatoglyphic features and blood types. In this study, fingerprint images are collected, preprocessed, and analyzed to extract unique features such as ridge count, minutiae points, and texture patterns. These features are then used to train various machine learning models, including Support Vector Machines (SVM), Random Forest, and Convolutional Neural Networks (CNN), to classify blood groups. The goal is to develop a reliable and efficient system that can serve as a supplementary tool in scenarios where traditional blood testing may not be feasible. The proposed system aims to enhance healthcare accessibility, especially in emergency and resource-limited settings.

INDEX

LIST OF FIGURES

DEEP LEARNING BASED BLOOD GROUP DETECTION USING FINGERPRINT

1. INTRODUCTION

Blood group determination is a crucial component in various medical procedures such as blood transfusions, organ transplants, and prenatal care. Traditionally, identifying an individual's blood group requires blood sampling and laboratory analysis, which can be time-consuming, invasive, and dependent on the availability of medical facilities. In emergency scenarios or rural regions where such resources may be limited, the need for a rapid, reliable, and non-invasive method becomes essential.

Fingerprints have long been used as a reliable biometric identifier due to their uniqueness and permanence. Interestingly, several studies have suggested a possible correlation between fingerprint patterns (dermatoglyphics) and blood groups. Dermatoglyphics, the scientific study of fingerprints, has revealed that certain fingerprint ridge patterns may be statistically associated with specific blood groups.

This project aims to explore the feasibility of using fingerprint images to predict an individual's blood group through machine learning techniques. By extracting key features from fingerprint patterns—such as ridge density, minutiae points, and texture—and training classification models, the system attempts to identify blood groups without the need for invasive procedures.

The motivation behind this approach lies in developing an accessible and fast diagnostic tool, especially for situations where conventional blood testing may not be available. If successful, this research could pave the way for integrating biometric systems with preliminary medical diagnostics, thereby improving healthcare delivery and emergency response systems.

2. LITERATURE SURVEY

1. Dermatoglyphic Patterns and Their Relationship with Blood Groups

Author(s): Dr. N. R. Bhat, Dr. A. H. Rajasab

Description:

This study explores the relationship between fingerprint patterns and ABO blood groups. It analyzed dermatoglyphic data from a large population and observed that specific fingerprint types (loops, whorls, arches) had a notable distribution among different blood groups. The findings suggested that loops were more frequent in individuals with blood group O, while whorls were more common among those with blood group B. Though not conclusive, this study laid the groundwork for further investigation into biometric correlations with genetic markers.

2. Predicting Blood Groups Using Fingerprint Patterns – A Data Mining Approach

Author(s): R. S. Mehta, A. R. Sharma

Description:

In this work, fingerprint images were analyzed using data mining algorithms to predict blood groups. The researchers used a dataset containing fingerprint images along with known blood groups and applied decision trees and Naive Bayes classifiers to build predictive models. The results showed an average accuracy of over 70%, indicating that there is a significant potential for blood group prediction using fingerprint features when combined with machine learning techniques.

3. Biometric Systems and Their Application in Healthcare

Author(s): L. Zhang, K. Jain

Description:

This paper discusses the applications of biometric systems, particularly fingerprints, in the healthcare domain. The authors review existing research on using biometric traits for patient identification and suggest that combining biometrics with diagnostic tools can enhance efficiency and accuracy in healthcare delivery. The study supports the idea that fingerprint analysis can be extended beyond identification to include health-related predictions, including genetic traits like blood groups.

4. Fingerprint-Based Health Prediction Models Using Deep Learning

Author(s): Priya A., Senthil Kumar R.

Description:

This recent work leverages convolutional neural networks (CNNs) to analyze fingerprint images for predicting health markers. The study demonstrated that CNNs could effectively classify various traits by learning complex patterns in the fingerprint ridges. Although blood group prediction was not the primary focus, the model's success in related predictions reinforces the viability of using deep learning for fingerprint-based medical classification.

5. A Review on the Correlation Between Fingerprint Patterns and Genetic Traits

Author(s): M. Kalpana, D. Srinivas

Description:

This literature review compiles various studies on the correlation between dermatoglyphics and genetic attributes such as blood groups, gender, and even certain diseases. The review highlights that while fingerprint patterns are primarily determined by genetic factors, further research is needed to establish robust models linking these patterns with specific biological markers. The paper calls for more empirical studies using modern machine learning tools to test and validate these hypotheses.

3. SYSTEM REQUIREMENTS SPECIFICATIONS

EXISTING SYSTEM:

Current methods for blood group detection are primarily based on serological testing, which involves collecting a blood sample and reacting it with specific antibodies to identify the presence or absence of antigens (A, B, and Rh). While these methods are accurate and widely accepted, they are invasive, require trained professionals, and depend on the availability of laboratory equipment and sterile conditions. In emergency situations or resource-limited settings, such methods may not be immediately accessible, leading to delays in treatment.

Biometric systems, particularly fingerprint recognition, have been widely used for identification and authentication purposes in various fields such as law enforcement, banking, and personal devices. These systems rely on the unique patterns present in fingerprints, including loops, whorls, arches, and minutiae points, which are highly distinct for every individual.

In summary, the existing systems for blood group detection are primarily invasive and resource-dependent, while the use of biometric features like fingerprints for predicting health traits is still an emerging area of research. At present, there is no robust, widely deployed system capable of predicting blood groups from fingerprints in real time. This gap presents a valuable opportunity to explore innovative, non-invasive approaches by leveraging fingerprint biometrics and machine learning techniques to develop a reliable and accessible diagnostic tool.

Disadvantages of Existing Systems:

1. Invasive Nature of Traditional Blood Group Testing
2. Dependence on Specialized Infrastructure
3. -Based Systems in Medical Diagnostics
4. Lack of Large-Scale, Validated Datasets
5. Low Accuracy and Reliability of Preliminary Models

PROPOSED SYSTEM:

The proposed system introduces a novel, non-invasive approach to blood group detection using fingerprint image analysis combined with machine learning techniques. Instead of relying on traditional blood sampling and laboratory procedures, this system leverages the unique ridge patterns in fingerprints to predict an individual's blood group. The goal is to develop a fast, accessible, and reliable method that can function effectively even in low-resource or emergency environments.

In this system, fingerprint images are collected using a biometric scanner or high-resolution camera. From the processed fingerprint, a range of features such as ridge count, minutiae points, and texture descriptors (e.g., Local Binary Patterns, Gabor features) are extracted, which may have statistical correlations with blood groups.

The extracted features are then fed into machine learning models to predict the blood group. Algorithms like Support Vector Machines (SVM), Random Forest, K-Nearest Neighbors (KNN), and Logistic Regression are used in the initial phases. Additionally, deep learning architectures such as Convolutional Neural Networks (CNNs) can be implemented to automatically learn complex patterns in fingerprint images and improve classification accuracy.

Ultimately, this proposed system offers a significant advancement in healthcare diagnostics by providing a contactless, rapid, and cost-effective alternative to traditional blood group detection. It has the potential to be integrated into emergency medical kits, rural healthcare devices, and biometric authentication systems, enhancing both convenience and accessibility.

Advantages of the Proposed System:

1. Non-Invasive and Painless
2. Faster Results
3. Cost-Effective and Resource-Efficient
4. Portable and Easy to Use
5. Potential for Integration with Biometric Identity Systems
6. Environmentally Friendly

SYSTEM ANALYSIS:

The core objective of this system is to analyze the relationship between fingerprint patterns and blood groups using computational methods. The analysis begins by understanding the biological and genetic basis of fingerprints and blood groups. While fingerprints are unique and genetically influenced, they do not change throughout a person’s life. Blood groups, determined by the ABO and Rh systems, are also genetic traits. Several studies suggest there might be correlations between dermatoglyphic patterns and blood types, making this analysis biologically plausible.

From a technical perspective, system analysis involves identifying the appropriate hardware and software requirements to support fingerprint acquisition and processing. The system needs high-resolution fingerprint scanners or cameras to capture detailed ridge patterns, and preprocessing algorithms to clean and enhance these images for further analysis. Image processing libraries such as OpenCV and machine learning frameworks like TensorFlow or scikit-learn are integrated into the system to handle complex operations.

Feature extraction is another critical part of the analysis. In this system, key fingerprint features such as ridge count, ridge flow, minutiae points, and ridge frequency are identified and quantified. These features form the input vectors for machine learning models.

The next stage of system analysis focuses on the selection and evaluation of machine learning models. Algorithms such as SVM, Random Forest, and CNNs are tested for their ability to learn and classify patterns associated with blood groups. Each model's performance is analyzed using metrics like accuracy, confusion matrix, precision, recall, and F1-score.. Model selection is based not only on accuracy but also on computational efficiency and ease of deployment.

Lastly, the system's usability and scalability are assessed. For practical use, the application should have a simple and intuitive user interface, minimal setup requirements, and fast response times. It should also be scalable—capable of handling large numbers of users and being deployable on various platforms such as desktops, tablets, or mobile devices. System security and data privacy are also analyzed, especially if biometric data is stored or transmitted, to ensure compliance with data protection standards.

HARDWARE & SOFTWARE REQUIREMENTS:

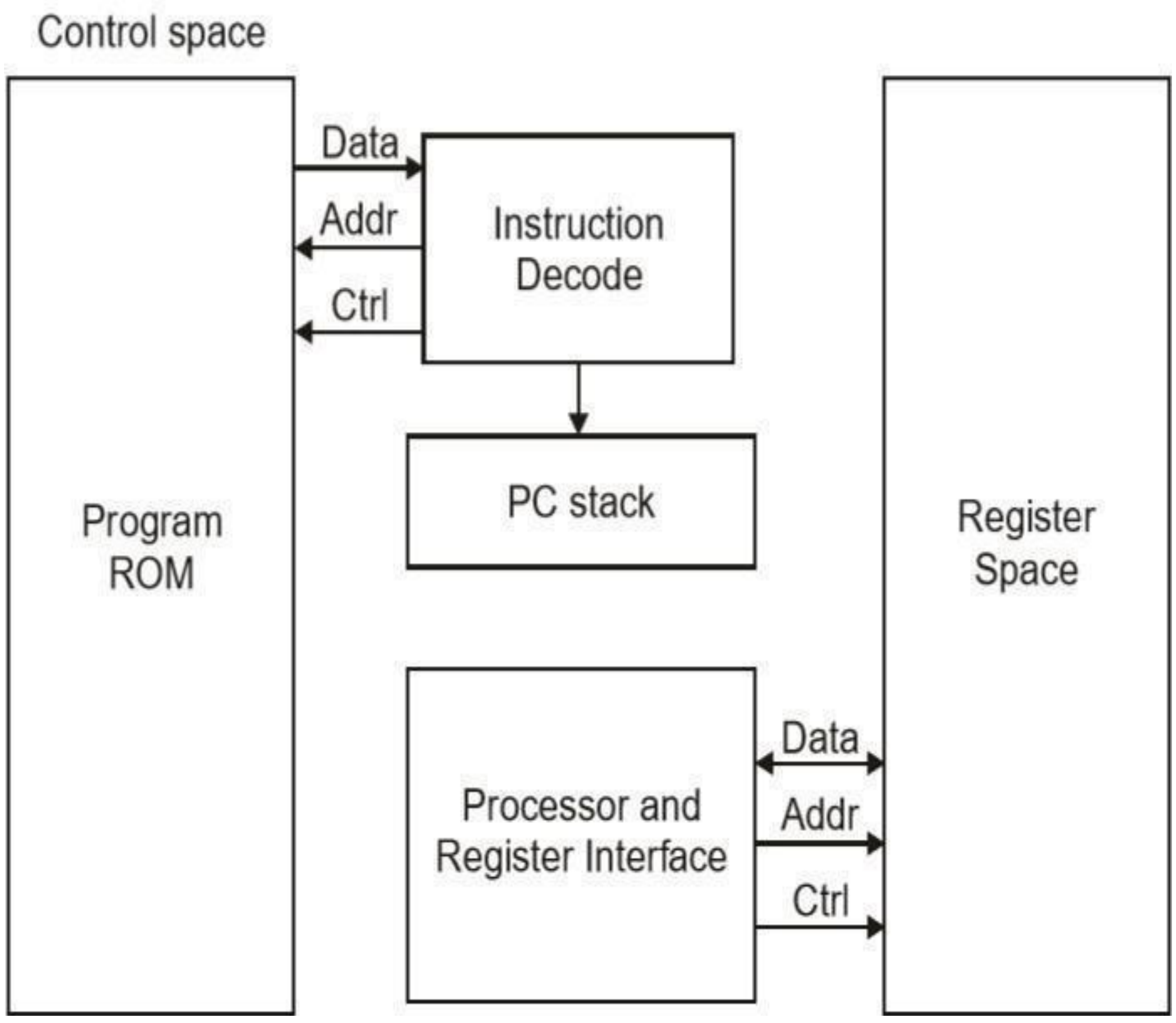
HARDWARE REQUIREMENTS:

• System	: Pentium IV 2.4 GHz.
• Hard Disk	: 40 GB.
• Ram	: 512 Mb.

SOFTWARE REQUIREMENTS:

• Operating system	: - Windows.
• Coding Language	: python.

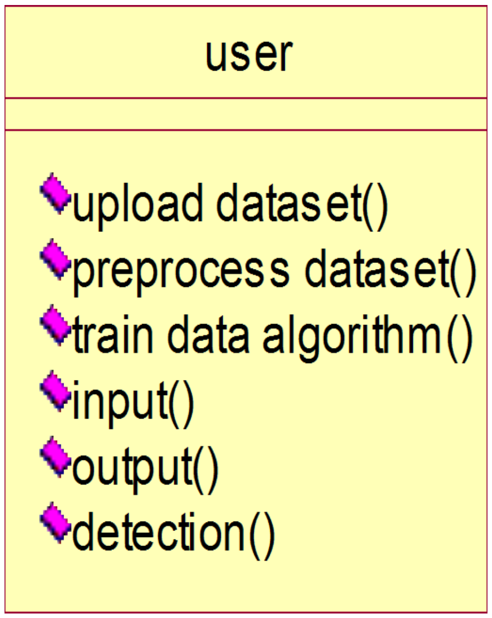
4.SYSTEM ARCHITECTURE



UML DIAGRAM'S :

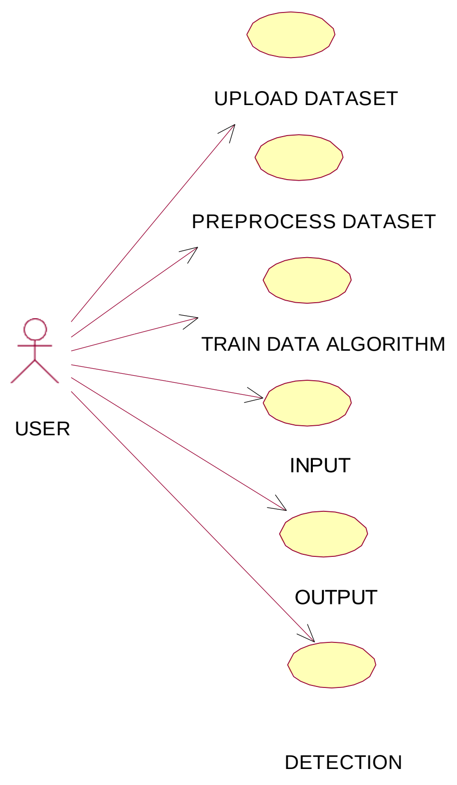
CLASS DIAGRAM:

The class diagram is used to refine the use case diagram and define a detailed design of the system. The class diagram classifies the actors defined in the use case diagram into a set of interrelated classes. The relationship or association between the classes can be either an "is-a" or "has-a" relationship. Each class in the class diagram may be capable of providing certain functionalities. These functionalities provided by the class are termed "methods" of the class. Apart from this, each class may have certain "attributes" that uniquely.



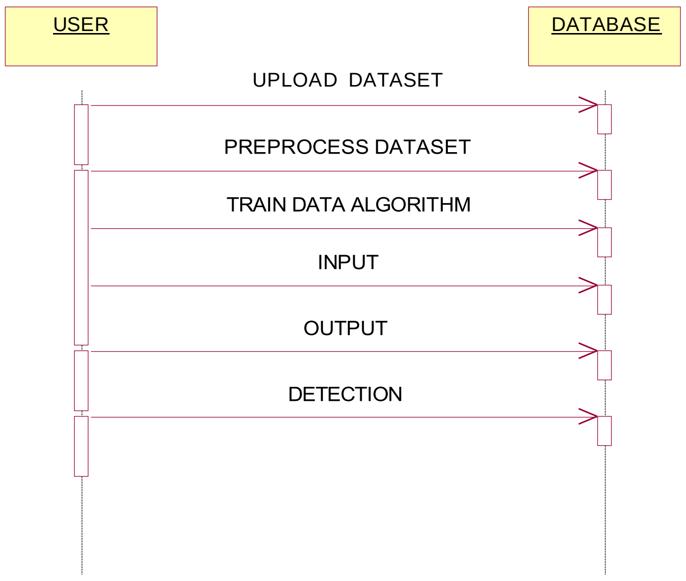
USE CASE DIAGRAM:

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



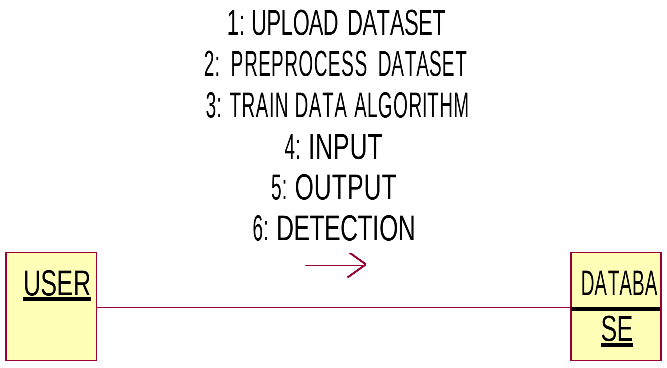
SEQUENCE DIAGRAM:

A sequence diagram represents the interaction between different objects in the system. The important aspect of a sequence diagram is that it is time-ordered. This means that the exact sequence of the interactions between the objects is represented step by step. Different objects in the sequence diagram interact with each other by passing "messages".



COLLABORATIVE DIAGRAM:

A collaboration diagram groups together the interactions between different objects. The interactions are listed as numbered interactions that help to trace the sequence of the interactions. The collaboration diagram helps to identify all the possible interactions that each object has with other objects.



5. SYSTEM IMPLEMENTATIONS

The implementation of the fingerprint-based blood group detection system involves the integration of biometric data acquisition, image processing,and user interface components. The system is designed to take a fingerprint as input, process it using image enhancement techniques, extract relevant features, and predict the individual’s blood group using model. programming tools such as Python, OpenCV, and machine learning libraries like scikit-learn and TensorFlow.

The first step in implementation is **data collection**. Fingerprint images are collected from individuals with known blood groups. Each fingerprint image is labeled with the corresponding blood group (A, B, AB, O with Rh+ or Rh−) and stored in a structured dataset for further processing.

Once the data is collected, **image preprocessing** is performed to improve fingerprint clarity and remove noise. This step includes grayscale conversion, histogram equalization, binarization, and ridge enhancement.After preprocessing, **feature extraction** techniques such as Minutiae extraction, Gabor filters, and Local Binary Patterns (LBP) are applied to transform the image into a feature vector that can be used for classification.

The next phase involves **training and testing the machine learning model**. The dataset is divided into training and testing sets using methods such as k-fold cross-validation to ensure model robustness. Various classifiers including Support Vector Machine (SVM), Random Forest, K-Nearest Neighbors (KNN), and Convolutional Neural Networks (CNN) are tested. These models are trained on the extracted fingerprint features to learn the mapping patterns blood group labels.

Finally, the system is deployed through a **Graphical User Interface (GUI)** developed using tools like Tkinter or PyQt. The GUI allows users to upload or capture a fingerprint, click a button to run the prediction, and receive the blood group result almost instantly. The implementation also includes logging features to record predictions for future analysis. The system is packaged for deployment on standalone PCs or embedded devices like Raspberry Pi for portability and use in field applications.

6. SYSTEM ENVIRONMENT

What is Python :-

Below are some facts about Python.

Python is currently the most widely used multi-purpose, high-level programming language.

Python allows programming in Object-Oriented and Procedural paradigms. Python programs generally are smaller than other programming languages like Java.

Programmers have to type relatively less and indentation requirement of the language, makes them readable all the time.

Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber… etc.

The biggest strength of Python is huge collection of standard library which can be used for the following .

- Machine Learning
- GUI Applications (like Kivy, Tkinter, PyQt etc.)
- Web frameworks like Django (used by YouTube, Instagram, Dropbox)
- Image processing (like Opencv, Pillow)
- Web scraping (like Scrapy, BeautifulSoup, Selenium)
- Test frameworks
- Multimedia

Advantages of Python :-

1. Extensive Libraries

Python downloads with an extensive library and it *contain code for various purposes like regular expressions, documentation-generation, unit-testing, web browsers, threading,* *databases, CGI, email, image manipulation, and more.*

2. Extensible

As we have seen earlier, Python can be **extended to other languages**. You can write some of your code in languages like C++ or C. This comes in handy, especially in projects.

3. Embeddable

Complimentary to extensibility, Python is embeddable as well. You can put your Python code in your source code of a different language, like C++. This lets us add **scripting capabilities** to our code in the other language.

4. Improved Productivity

The language's simplicity and extensive libraries render programmers **more productive** than languages like Java and C++ do. Also, the fact that you need to write less and get more things done.

5. IOT Opportunities

Since Python forms the basis of new platforms like Raspberry Pi, it finds the future bright for the Internet Of Things. This is a way to connect the language with the real world.

6. Simple and Easy

When working with Java, you may have to create a class to print **'Hello World'**. But in Python, just a print statement will do. It is also quite **easy to learn, understand,** and **code.** This is why when people pick up Python, they have a hard time adjusting to other more verbose languages like Java.

7. Readable

Because it is not such a verbose language, reading Python is much like reading English. This is the reason why it is so easy to learn, understand, and code. It also does not need curly braces to define blocks, and **indentation is mandatory.** This further aids the readability of the code.

8. Object-Oriented

This language supports both the **procedural and object-oriented** programming paradigms. While functions help us with code reusability, classes and objects let us model the real world. A class allows the **encapsulation of data** and functions into one.

9. Free and Open-Source

Like we said earlier, Python is **freely available.** But not only can you **download Python** for free, but you can also download its source code, make changes to it, and even distribute it. It downloads with an extensive collection of libraries to help you with your tasks.

10. Portable

When you code your project in a language like C++, you may need to make some changes to it if you want to run it on another platform. But it isn't the same with Python. Here, you need to **code only once**, and you can run it anywhere. This is called **Write Once Run Anywhere (WORA)**. However, you need to be careful enough not to include any system-dependent features.

11. Interpreted

Lastly, we will say that it is an interpreted language. Since statements are executed one by one, **debugging is easier** than in compiled languages.

Advantages of Python Over Other Languages:

1. Less Coding

Almost all of the tasks done in Python requires less coding when the same task is done in other languages. Python also has an awesome standard library support, so you don't have to search for any third-party libraries to get your job done. This is the reason that many people suggest learning Python to beginners.

2. Affordable

Python is free therefore individuals, small companies or big organizations can leverage the free available resources to build applications. Python is popular and widely used so it gives you better community support.

The 2019 Github annual survey showed us that Python has overtaken Java in the most popular programming language category.

3. Python is for Everyone

Python code can run on any machine whether it is Linux, Mac or Windows. Programmers need to learn different languages for different jobs but with Python, you can professionally build web apps, perform data analysis and **machine learning**, automate things, do web scraping and also build games and powerful visualizations. It is an all-rounder programming language.

Disadvantages:

1. Speed Limitations
2. Weak in Mobile Computing and Browsers
3. Design Restrictions
4. Underdeveloped Database Access Layers
5. High Memory Consumption

History of Python : -

What do the alphabet and the programming language Python have in common? Right, both start with ABC. If we are talking about ABC in the Python context, it's clear that the programming language ABC is meant. ABC is a general-purpose programming language and programming environment, which had been developed in the Netherlands, Amsterdam, at the CWI (Centrum Wiskunde &Informatica). The greatest achievement of ABC was to influence the design of Python.Python was conceptualized in the late 1980s. Guido van Rossum worked that time in a project at the CWI, called Amoeba, a distributed operating system. In an interview with Bill Venners[1], Guido van Rossum said: "In the early 1980s, I worked as an implementer on a team building a language called ABC at Centrum voor Wiskunde en Informatica (CWI).

I don't know how well people know ABC's influence on Python. I try to mention ABC's influence because I'm indebted to everything I learned during that project and to the people who worked on it."Later on in the same Interview, Guido van Rossum continued: "I remembered all my experience and some of my frustration with ABC. I decided to try to design a simple scripting language that possessed some of ABC's better properties, but without its problems. So I started typing. I created a simple virtual machine, a simple parser, and a simple runtime. I made my own version of the various ABC parts that I liked. I created a basic syntax, used indentation for statement grouping instead of curly braces or begin-end blocks, and developed a small number of powerful data types: a hash table (or dictionary, as we call it), a list, strings, and numbers."

What is Machine Learning : -

Before we take a look at the details of various machine learning methods, let's start by looking at what machine learning is, and what it isn't. Machine learning is often categorized as a subfield of artificial intelligence, but I find that categorization can often be misleading at first brush. The study of machine learning certainly arose from research in this context, but in the data science application of machine learning methods, it's more helpful to think of machine learning as a means of *building models of data*.

Once these models have been fit to previously seen data, they can be used to predict and understand aspects of newly observed data. I'll leave to the reader the more philosophical digression regarding the extent to which this type of mathematical, model-based "learning" is similar to the "learning" exhibited by the human brain.Understanding the problem setting in machine learning is essential to using these tools effectively, and so we will start with some broad categorizations of the types of approaches we'll discuss here.

Categories Of Machine Leaning :-

At the most fundamental level, machine learning can be categorized into two main types: supervised learning and unsupervised learning.

Supervised learning involves somehow modeling the relationship between measured features of data and some label associated with the data; once this model is determined, it can be used to apply labels to new, unknown data. This is further subdivided into *classification* tasks and *regression* tasks: in classification, the labels are discrete categories, while in regression, the labels are continuous quantities.

Unsupervised learning involves modeling the features of a dataset without reference to any label, and is often described as "letting the dataset speak for itself." These models include tasks such as *clustering* and *dimensionality reduction.*

Need for Machine Learning:-

Human beings, at this moment, are the most intelligent and advanced species on earth because they can think, evaluate and solve complex problems. On the other side, AI is still in its initial stage and haven't surpassed human intelligence in many aspects. Then the question is that what is the need to make machine learn? The most suitable reason for doing this is, "to make decisions, based on data, with efficiency and scale".

Lately, organizations are investing heavily in newer technologies like Artificial Intelligence, Machine Learning and Deep Learning to get the key information from data to perform several real-world tasks and solve problems.We can call it data-driven decisions taken by machines.

Challenges in Machines Learning :-

While Machine Learning is rapidly evolving, making significant strides with cybersecurity and autonomous cars, this segment of AI as whole still has a long way to go. The reason behind is that ML has not been able to overcome number of challenges. The challenges that ML is facing currently are −

Quality of data − Having good-quality data for ML algorithms is one of the biggest challenges. Use of low-quality data leads to the problems related to data preprocessing and feature extraction.

Time-Consuming task − Another challenge faced by ML models is the consumption of time especially for data acquisition, feature extraction and retrieval.

Lack of specialist persons − As ML technology is still in its infancy stage, availability of expert resources is a tough job.

No clear objective for formulating business problems − Having no clear objective and well-defined goal for business problems is another key challenge for ML because this technology is not that mature yet.

Issue of overfitting & underfitting − If the model is overfitting or underfitting, it cannot be represented well for the problem.

Applications of Machines Learning :-

Machine Learning is the most rapidly growing technology and according to researchers we are in the golden year of AI and ML. It is used to solve many real-world complex problems which cannot be solved with traditional approach. Following are some real-world applications of ML –

- Emotion analysis
- Sentiment analysis
- Error detection and prevention
- Weather forecasting and prediction
- Stock market analysis and forecasting
- Speech synthesis
- Speech recognition
- Customer segmentation
- Object recognition
- Fraud detection
- Fraud prevention
- Recommendation of products to customer in online shopping

How to Start Learning Machine Learning?

Arthur Samuel coined the term **"Machine Learning"** in 1959 and defined it as a **"Field of study that gives computers the capability to learn without being explicitly programmed".** And that was the beginning of Machine Learning! In modern times, Machine Learning is one of the most popular (if not the most!) career choices. According to Indeed, Machine Learning Engineer Is The Best Job of 2019 with a *344%* growth and an average base salary of **$146,085** per year.

But there is still a lot of doubt about what exactly is Machine Learning and how to start learning it? So this article deals with the Basics of Machine Learning and also the path you can follow to eventually become a full-fledged Machine Learning Engineer. Now let's get started!!!

How to start learning ML?

This is a rough roadmap you can follow on your way to becoming an insanely talented Machine Learning Engineer. Of course, you can always modify the steps according to your needs to reach your desired end-goal!

Step 1 – Understand the Prerequisites

In case you are a genius, you could start ML directly but normally, there are some prerequisites that you need to know which include Linear Algebra, Multivariate Calculus, Statistics, and Python. And if you don't know these, never fear! You don't need a Ph.D. degree in these topics to get started but you do need a basic understanding.

(a) Learn Linear Algebra and Multivariate Calculus

Both Linear Algebra and Multivariate Calculus are important in Machine Learning. However, the extent to which you need them depends on your role as a data scientist. If you are more focused on application heavy machine learning, then you will not be that heavily focused on maths as there are many common libraries available. But if you want to focus on R&D in Machine Learning, then mastery of Linear Algebra and Multivariate Calculus is very important as you will have to implement many ML algorithms from scratch.

(b) Learn Statistics

Data plays a huge role in Machine Learning. In fact, around 80% of your time as an ML expert will be spent collecting and cleaning data. And statistics is a field that handles the collection, analysis, and presentation of data. So it is no surprise that you need to learn it!!! Some of the key concepts in statistics that are important are Statistical Significance, Probability

Distributions, Hypothesis Testing, Regression, etc. Also, Bayesian Thinking is also a very important part of ML which deals with various concepts like Conditional Probability, Priors, and Posteriors, Maximum Likelihood, etc.

(c) Learn Python

Some people prefer to skip Linear Algebra, Multivariate Calculus and Statistics and learn them as they go along with trial and error. But the one thing that you absolutely cannot skip is Python! While there are other languages you can use for Machine Learning like R, Scala, etc. Python is currently the most popular language for ML. In fact, there are many Python libraries that are specifically useful for Artificial Intelligence and Machine Learning such as Keras, TensorFlow, Scikit-learn, etc.
So if you want to learn ML, it's best if you learn Python! You can do that using various online resources and courses such as **Fork Python** available Free on GeeksforGeeks.

Step 2 – Learn Various ML Concepts

Now that you are done with the prerequisites, you can move on to actually learning ML (Which is the fun part!!!) It's best to start with the basics and then move on to the more complicated stuff. Some of the basic concepts in ML are:

(a) Terminologies of Machine Learning

- **Model –** A model is a specific representation learned from data by applying some machine learning algorithm. A model is also called a hypothesis.
- **Feature –** A feature is an individual measurable property of the data. A set of numeric features can be conveniently described by a feature vector. Feature vectors are fed as input to the model. For example, in order to predict a fruit, there may be features like color, smell, taste, etc.
- **Target (Label) –** A target variable or label is the value to be predicted by our model. For the fruit example discussed in the feature section, the label with each set of input would be the name of the fruit like apple, orange, banana, etc.

- **Training –** The idea is to give a set of inputs(features) and it's expected outputs(labels), so after training, we will have a model (hypothesis) that will then map new data to one of the categories trained on.
- **Prediction –** Once our model is ready, it can be fed a set of inputs to which it will provide a predicted output(label).

(b) Types of Machine Learning

- **Supervised Learning –** This involves learning from a training dataset with labeled data using classification and regression models. This learning process continues until the required level of performance is achieved.
- **Unsupervised Learning –** This involves using unlabelled data and then finding the underlying structure in the data in order to learn more and more about the data itself using factor and cluster analysis models.
- **Semi-supervised Learning –** This involves using unlabelled data like Unsupervised Learning with a small amount of labeled data. Using labeled data vastly increases the learning accuracy and is also more cost-effective than Supervised Learning.
- **Reinforcement Learning –** This involves learning optimal actions through trial and error. So the next action is decided by learning behaviors that are based on the current state and that will maximize the reward in the future.

Advantages of Machine learning :-

1. Easily identifies trends and patterns -

Machine Learning can review large volumes of data and discover specific trends and patterns that would not be apparent to humans. For instance, for an e-commerce website like Amazon, it serves to understand the browsing behaviors and purchase histories of its users to help cater to the right products, deals, and reminders relevant to them. It uses the results to reveal relevant advertisements to them.

2. No human intervention needed (automation)

With ML, you don't need to babysit your project every step of the way. Since it means giving machines the ability to learn, it lets them make predictions and also improve the algorithms on their own. A common example of this is anti-virus softwares; they learn to filter new threats as they are recognized. ML is also good at recognizing spam.

3. Continuous Improvement

As **ML algorithms** gain experience, they keep improving in accuracy and efficiency. This lets them make better decisions. Say you need to make a weather forecast model. As the amount of data you have keeps growing, your algorithms learn to make more accurate predictions faster.

4. Handling multi-dimensional and multi-variety data

Machine Learning algorithms are good at handling data that are multi-dimensional and multi-variety, and they can do this in dynamic or uncertain environments.

5. Wide Applications

You could be an e-tailer or a healthcare provider and make ML work for you. Where it does apply, it holds the capability to help deliver a much more personal experience to customers while also targeting the right customers.

Disadvantages of Machine Learning :-

1. Data Acquisition

Machine Learning requires massive data sets to train on, and these should be inclusive/unbiased, and of good quality. There can also be times where they must wait for new data to be generated.

2. Time and Resources

ML needs enough time to let the algorithms learn and develop enough to fulfill their purpose with a considerable amount of accuracy and relevancy. It also needs massive resources to function. This can mean additional requirements of computer power for you.

3. Interpretation of Results

Another major challenge is the ability to accurately interpret results generated by the algorithms. You must also carefully choose the algorithms for your purpose.

4. High error-susceptibility

Machine Learning is autonomous but highly susceptible to errors. Suppose you train an algorithm with data sets small enough to not be inclusive. You end up with biased predictions coming from a biased training set. This leads to irrelevant advertisements being displayed to customers. In the case of ML, such blunders can set off a chain of errors that can go undetected for long periods of time. And when they do get noticed, it takes quite some time to recognize the source of the issue, and even longer to correct it.

Python Development Steps : -

Guido Van Rossum published the first version of Python code (version 0.9.0) at alt.sources in February 1991. This release included already exception handling, functions, and the core data types of list, dict, str and others. It was also object oriented and had a module system. Python version 1.0 was released in January 1994. The major new features included in this release were the functional programming tools lambda, map, filter and reduce, which Guido Van Rossum never liked.Six and a half years later in October 2000, Python 2.0 was introduced. This release included list comprehensions, a full garbage collector and it was supporting unicode.Python flourished for another 8 years in the versions 2.x before the next major release as Python 3.0 (also known as "Python 3000" and "Py3K") was released. Python 3 is not backwards compatible with Python 2.x.

The emphasis in Python 3 had been on the removal of duplicate programming constructs and modules, thus fulfilling or coming close to fulfilling the 13th law of the Zen of Python: "There should be one -- and preferably only one -- obvious way to do it."Some changes in Python 7.3:

- Print is now a function
- Views and iterators instead of lists
- The rules for ordering comparisons have been simplified. E.g. a heterogeneous list cannot be sorted, because all the elements of a list must be comparable to each other.
- The division of two integers returns a float instead of an integer. "//" can be used to have the "old" behaviour.
- Text Vs. Data Instead Of Unicode Vs. 8-bit

Purpose :-

We demonstrated that our approach enables successful segmentation of intra-retinal layers—even with low-quality images containing speckle noise, low contrast, and different intensity ranges throughout—with the assistance of the ANIS feature.

Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

- 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 also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

Modules Used in Project :-

Tensorflow

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.

TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache 2.0 open-source license on November 9, 2015.

Numpy

Numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined using Numpy which allows Numpy to seamlessly and speedily integrate with a wide variety of databases.

Pandas

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

Scikit – learn

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use.

Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

- 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 also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels.

All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

Install Python Step-by-Step in Windows and Mac :

Python a versatile programming language doesn’t come pre-installed on your computer devices. Python was first released in the year 1991 and until today it is a very popular high-level programming language. Its style philosophy emphasizes code readability with its notable use of great whitespace.

How to Install Python on Windows and Mac :

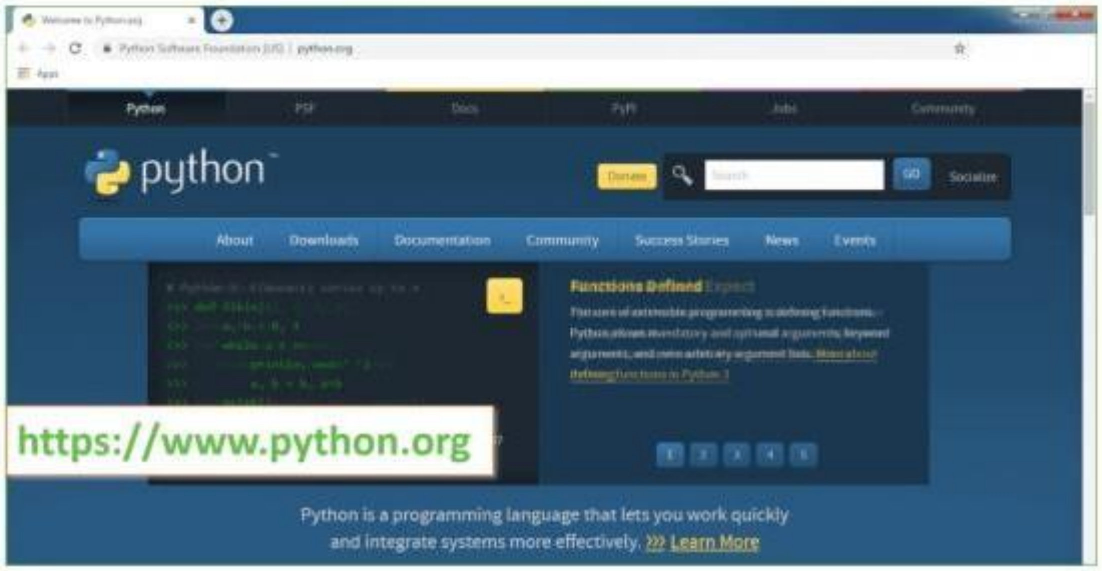
There have been several updates in the Python version over the years. The question is how to install Python? It might be confusing for the beginner who is willing to start learning Python but this tutorial will solve your query. The latest or the newest version of Python is version 3.7.4 or in other words, it is Python 3.

Note: The python version 3.7.4 cannot be used on Windows XP or earlier devices.

Before you start with the installation process of Python. First, you need to know about your **System Requirements**. Based on your system type i.e. operating system and based processor, you must download the python version. My system type is a **Windows 64-bit operating system**. So the steps below are to install python version 3.7.4 on Windows 7 device or to install Python 3. Download the Python Cheatsheet here.The steps on how to install Python on Windows 10, 8 and 7 are **divided into 4 parts** to help understand better.

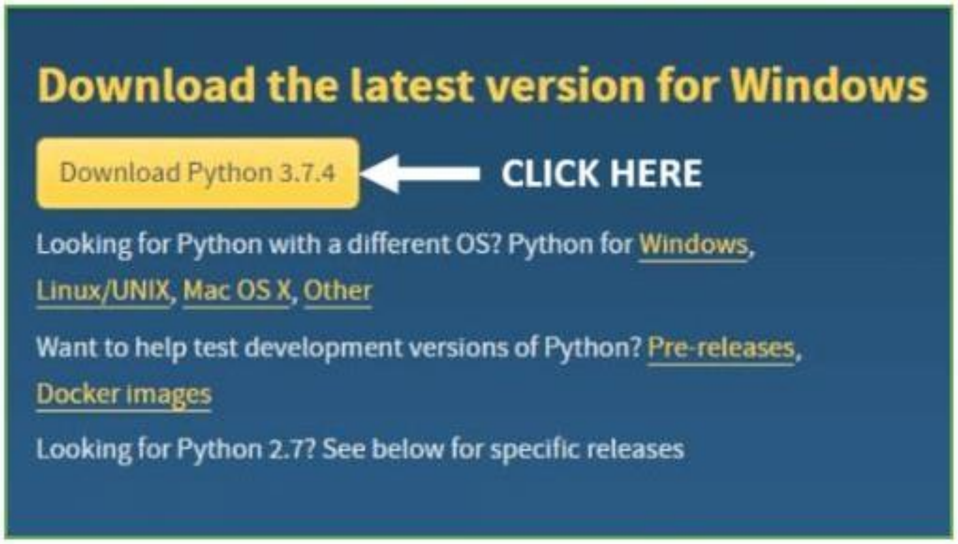
Download the Correct version into the system

Step 1: Go to the official site to download and install python using Google Chrome or any other web browser. OR Click on the following link: https://www.python.org


python
About
Downloads
Documentation
Community
Success Stories
News
Events
Functions Defined
https://www.python.org
Python is a programming language that lets you work quickly and integrate systems more effectively. >>> Learn More

Now, check for the latest and the correct version for your operating system.

Step2: ClickontheDownloadTab.



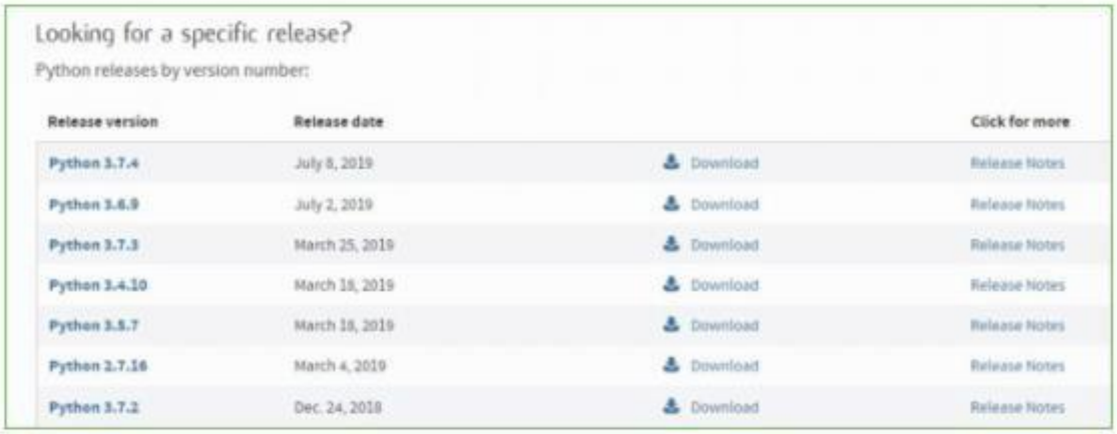
Download the latest version for Windows

Looking for Python with a different OS? Python for Windows, Linux/UNIX, Mac OS X, Other

Want to help test development versions of Python? Pre-releases, Docker images

Looking for Python 2.7? See below for specific releases

Step 3: You can either select the Download Python for windows 3.7.4 button in Yellow Color or you can scroll further down and click on download with respective to their version. Here, we are downloading the most recent python version for windows 3.7.4



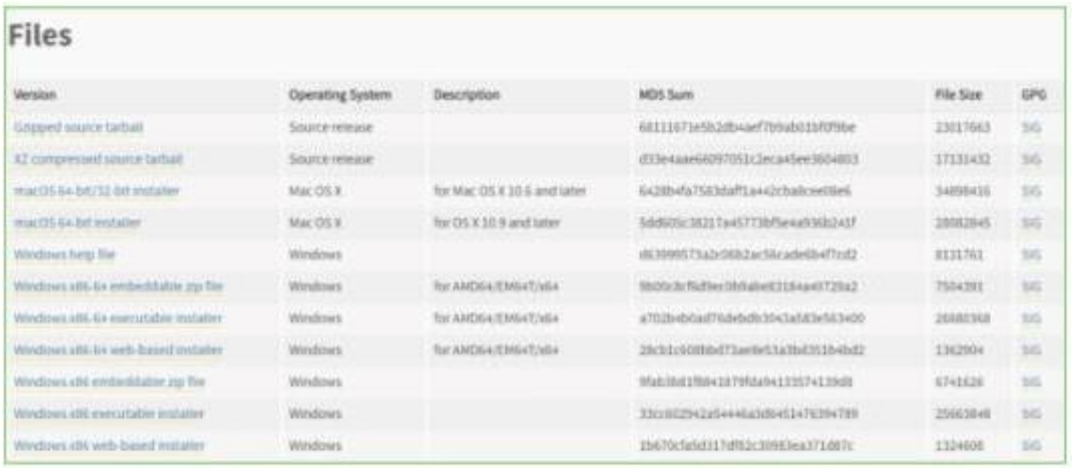
Looking for a specific release?

Python releases by version number:

Release version	Release date		Click for more
Python 3.7.4	July 8, 2019	Download	Release Notes
Python 3.6.9	July 2, 2019	Download	Release Notes
Python 3.7.3	March 25, 2019	Download	Release Notes
Python 3.4.10	March 18, 2019	Download	Release Notes
Python 3.5.7	March 18, 2019	Download	Release Notes
Python 2.7.16	March 4, 2019	Download	Release Notes
Python 3.7.2	Dec. 24, 2018	Download	Release Notes

Step 4: Scroll down the page until you find the Files option.

Step 5: Here you see a different version of python along with the operating system.



Files

Version	Operating System	Description	MD5 Sum	File Size	GPG
Gzipped source tarball	Source release		[illegible]	23017663	SIG
XZ compressed source tarball	Source release		[illegible]	17131432	SIG
macOS 64-bit/32-bit installer	Mac OS X	for Mac OS X 10.6 and later	[illegible]	34898416	SIG
macOS 64-bit installer	Mac OS X	for OS X 10.9 and later	[illegible]	28082845	SIG
Windows help file	Windows		[illegible]	8131761	SIG
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64T/x64	[illegible]	7504391	SIG
Windows x86-64 executable installer	Windows	for AMD64/EM64T/x64	[illegible]	26680368	SIG
Windows x86-64 web-based installer	Windows	for AMD64/EM64T/x64	[illegible]	1362904	SIG
Windows x86 embeddable zip file	Windows		[illegible]	6741626	SIG
Windows x86 executable installer	Windows		[illegible]	25663848	SIG
Windows x86 web-based installer	Windows		[illegible]	1324608	SIG

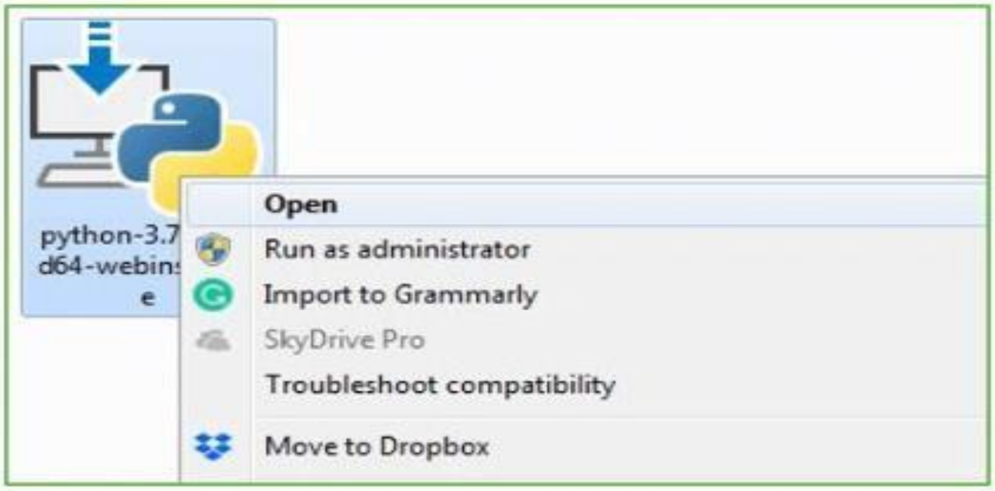
• To download Windows 32-bit python, you can select any one from the three options: Windows x86 embeddable zip file, Windows x86 executable installer or Windows x86 web-based installer.
•To download Windows 64-bit python, you can select any one from the three options: Windows x86-64 embeddable zip file, Windows x86-64 executable installer or Windows x86-64 web-based installer.

Here we will install Windows x86-64 web-based installer. Here your first part regarding which version of python is to be downloaded is completed. Now we move ahead with the second part in installing python i.e. Installation

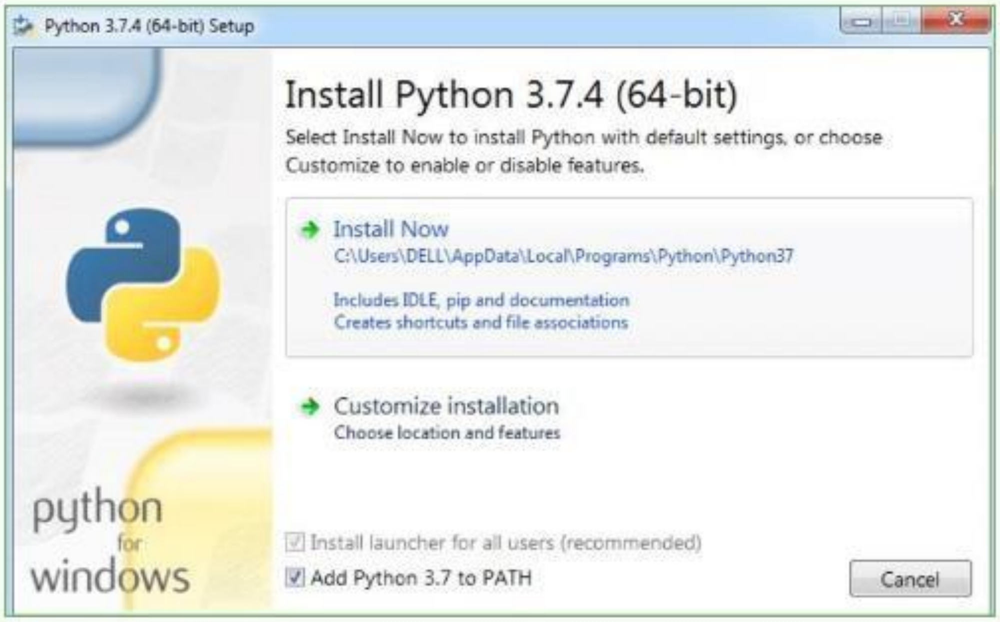
Note: To know the changes or updates that are made in the version you can click on the Release Note Option.

Installation of python

Step1:Go to Download and Open the downloaded python vesion to carry out the installation process



Step 2: Before you click on Install Now, Make sure to put a tick on Add Python 3.7 to PATH.



Python 3.7.4 (64-bit) Setup

Install Python 3.7.4 (64-bit)

Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

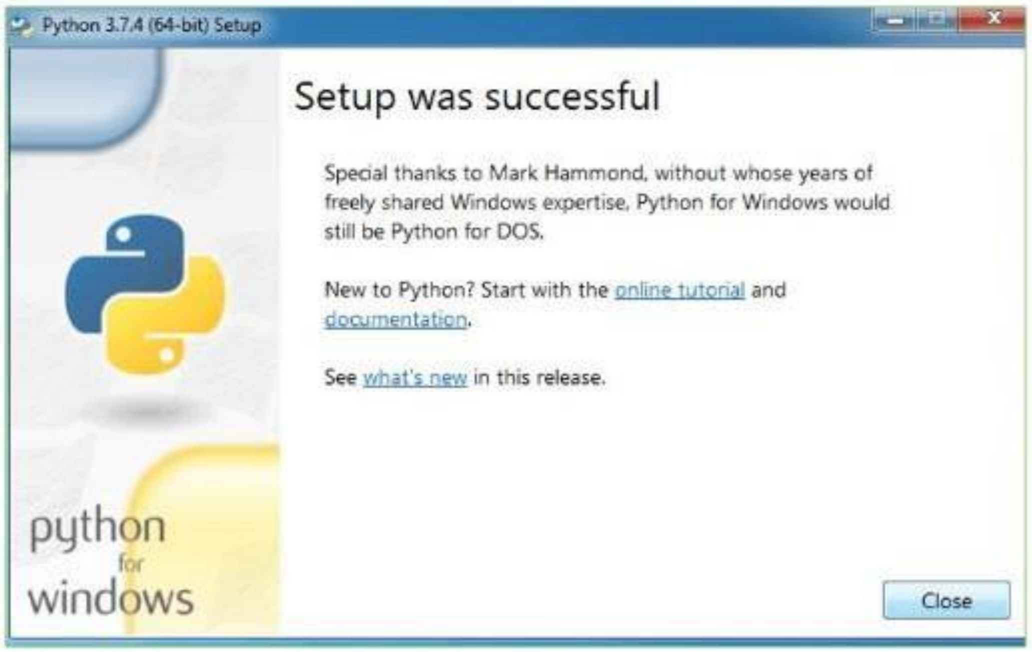
Install Now
C:\Users\DELL\AppData\Local\Programs\Python\Python37

Includes IDLE, pip and documentation
Creates shortcuts and file associations

Customize installation
Choose location and features

☑ Install launcher for all users (recommended)
☑ Add Python 3.7 to PATH

Step 3: Click on Install NOW After the installation is successful. Click on Close.



Python 3.7.4 (64-bit) Setup

Setup was successful

Special thanks to Mark Hammond, without whose years of freely shared Windows expertise, Python for Windows would still be Python for DOS.

New to Python? Start with the online tutorial and documentation.

See what's new in this release.

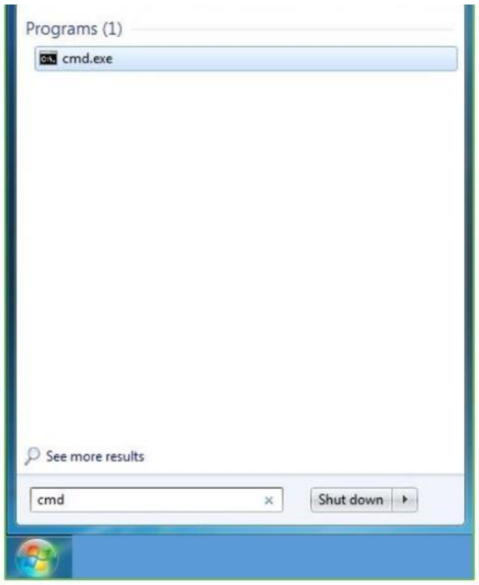
With these above three steps on python installation, you have successfully and correctly installed Python. Now is the time to verify the installation.

Note: The installation process might take a couple of minutes.

Verify the Python Installation

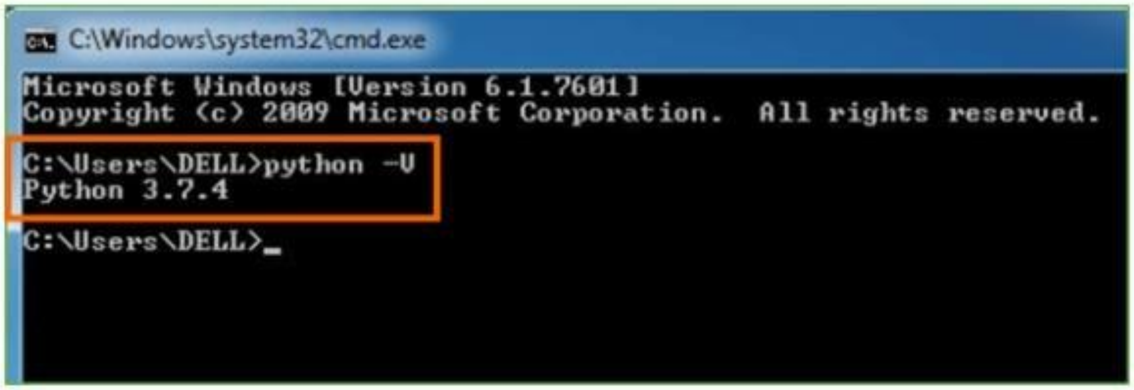
Step 1: Click on Start

Step 2: In the Windows Run Command, type “cmd”.



Step 3: Open the Command prompt option.

Step 4: Let us test whether the python is correctly installed. Type **python –V** and press Enter.



```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\DELL>python -V
Python 3.7.4

C:\Users\DELL>_
```

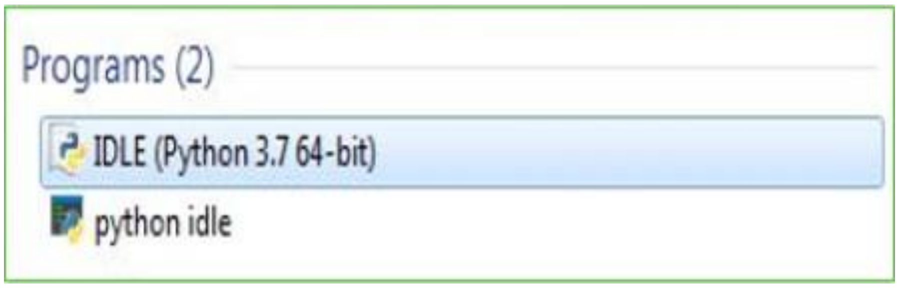
Step 5: You will get the answer as 3.7.4

Note: If you have any of the earlier versions of Python already installed. You must first uninstall the earlier version and then install the new one.

Check how the Python IDLE works

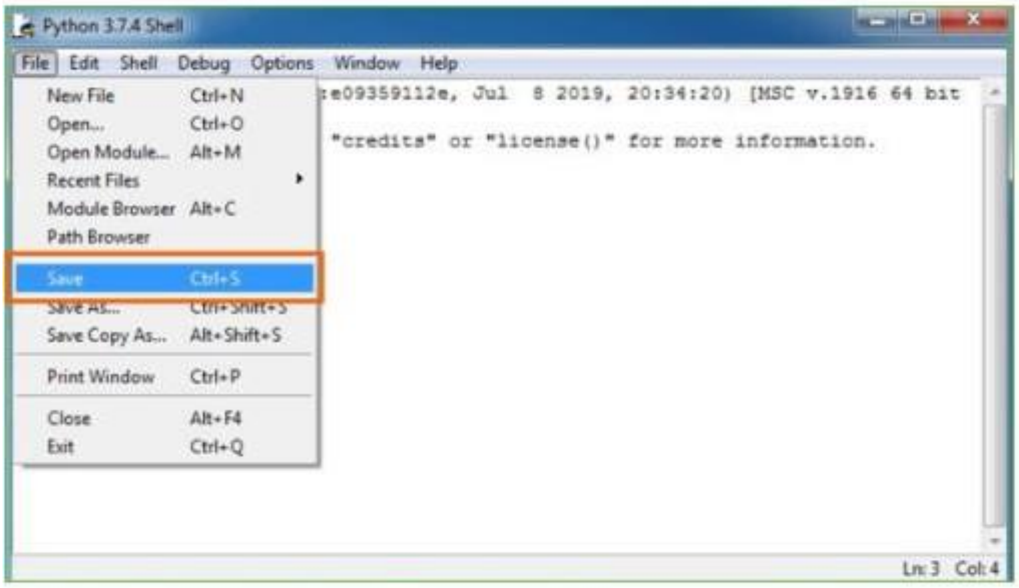
Step 1: Click on Start

Step 2: In the Windows Run command, type "python idle".



Step 3: Click on IDLE (Python 3.7 64-bit) and launch the program

Step 4: To go ahead with working in IDLE you must first save the file. **Click on File > Click on Save**



```
e09359112e, Jul  8 2019, 20:34:20) [MSC v.1916 64 bit
"credits" or "license()" for more information.
```

Step 5: Name the file and save as type should be Python files. Click on SAVE. Here I have named the files as Hey World.

Step 6: Now for e.g. **enter print**

7. SYSTEM TESTING

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

TYPES OF TESTS

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. 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.

Integration testing

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

Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals. Functional testing is centered on the following items:

Valid Input	: identified classes of valid input must be accepted.
Invalid Input	: identified classes of invalid input must 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. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing

System Test

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

White Box Testing

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

Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document.It is a testing in which the software under test is treated, as a black box .you cannot "see" into it. The test provides responds to outputs without considering how the software works.

Unit Testing

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

Test strategy and approach

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

Test objectives

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

Features to be tested

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

Integration Testing

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

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

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

Acceptance Testing

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

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

8. TEST CASES

Test cases 1: Test case for Login form:

FUNCTION:	**LOGIN**
EXPECTED RESULTS:	Should Validate the user and check his existence in database
ACTUAL RESULTS:	Validate the user and checking the user against the database
LOW PRIORITY	**No**
HIGH PRIORITY	**Yes**

Case2: Test case for User Registration form:

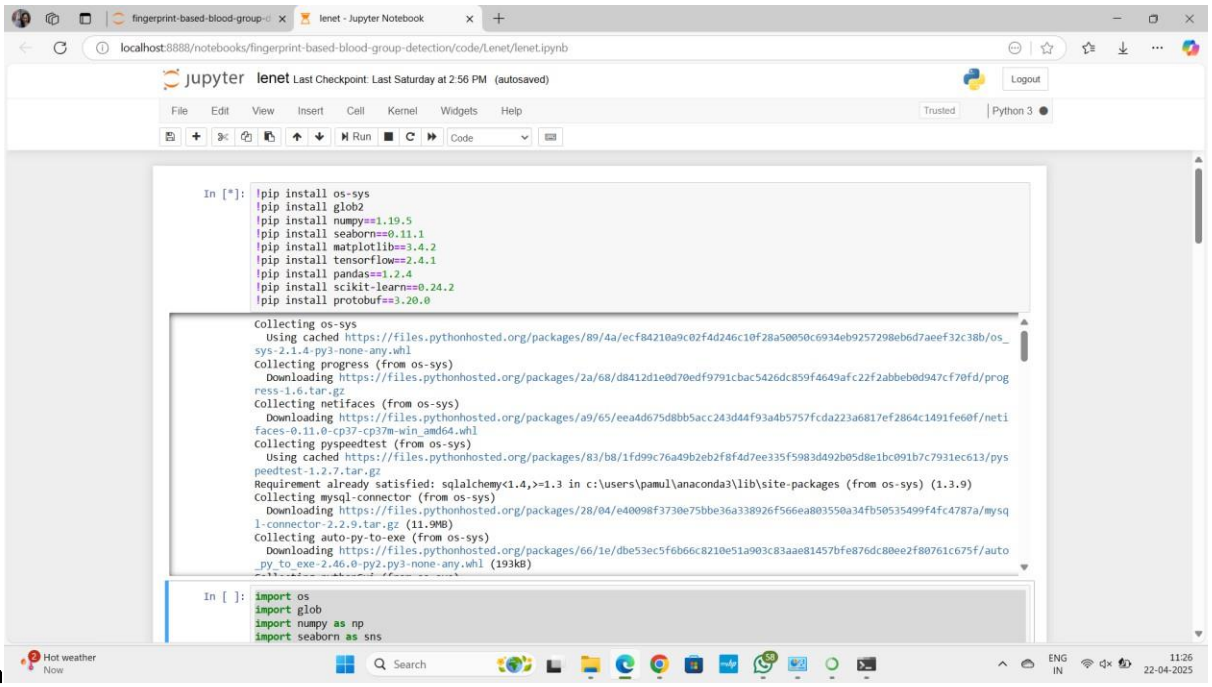
FUNCTION:	**USER REGISTRATION**
EXPECTED RESULTS:	Should check if all the fields are filled by the user and saving the user to database.
ACTUAL RESULTS:	Checking whether all the fields are field by user or not through validations and saving user.
LOW PRIORITY	**No**
HIGH PRIORITY	**Yes**

Test case3: Test case for Change Password:

When the old password does not match with the new password ,then this results in displaying an error message as “ OLD PASSWORD DOES NOT MATCH WITH THE NEW PASSWORD”.

FUNCTION:	**Change Password**
EXPECTED RESULTS:	Should check if old password and new password fields are filled by the user and saving the user to database.
ACTUAL RESULTS:	Checking whether all the fields are field by user or not through validations and saving user.
LOW PRIORITY	**No**
HIGH PRIORITY	**Yes**

9. SCREEN SHOTS

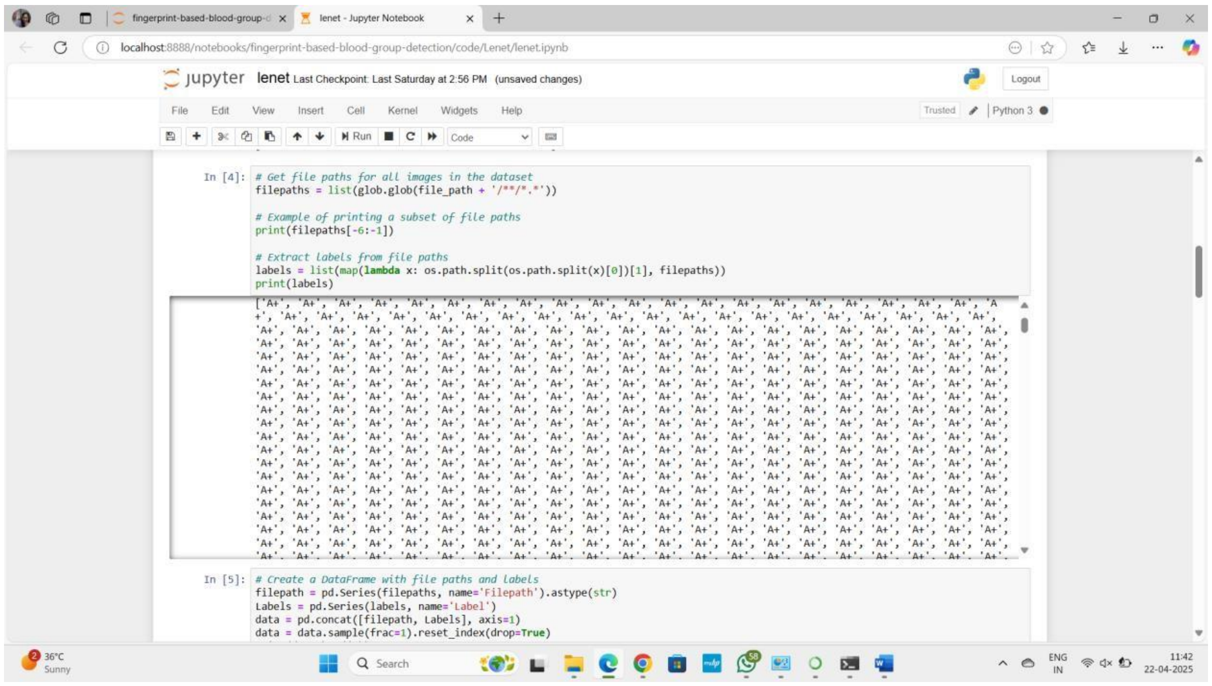


```
In [*]: !pip install os-sys
        !pip install glob2
        !pip install numpy==1.19.5
        !pip install seaborn==0.11.1
        !pip install matplotlib==3.4.2
        !pip install tensorflow==2.4.1
        !pip install pandas==1.2.4
        !pip install scikit-learn==0.24.2
        !pip install protobuf==3.20.0
```

```
Collecting os-sys
  Using cached https://files.pythonhosted.org/packages/89/4a/ecf84210a9c02f4d246c10f28a50050c6934eb9257298eb6d7aeef32c38b/os_
sys-2.1.4-py3-none-any.whl
Collecting progress (from os-sys)
  Downloading https://files.pythonhosted.org/packages/2a/68/d8412d1e0d70edf9791cbac5426dc859f4649afc22f2abbeb0d947cf70fd/prog
ress-1.6.tar.gz
Collecting netifaces (from os-sys)
  Downloading https://files.pythonhosted.org/packages/a9/65/eea4d675d8bb5acc243d44f93a4b5757fcda223a6817ef2864c1491fe60f/neti
faces-0.11.0-cp37-cp37m-win_amd64.whl
Collecting pyspeedtest (from os-sys)
  Using cached https://files.pythonhosted.org/packages/83/b8/1fd99c76a49b2eb2f8f4d7ee335f5983d492b05d8e1bc091b7c7931ec613/pys
peedtest-1.2.7.tar.gz
Requirement already satisfied: sqlalchemy<1.4,>=1.3 in c:\users\pamul\anaconda3\lib\site-packages (from os-sys) (1.3.9)
Collecting mysql-connector (from os-sys)
  Downloading https://files.pythonhosted.org/packages/28/04/e40098f3730e75bbe36a338926f566ea803550a34fb50535499f4fc4787a/mysq
l-connector-2.2.9.tar.gz (11.9MB)
Collecting auto-py-to-exe (from os-sys)
  Downloading https://files.pythonhosted.org/packages/66/1e/dbe53ec5f6b66c8210e51a903c83aae81457bfe876dc80ee2f80761c675f/auto
_py_to_exe-2.46.0-py2.py3-none-any.whl (193kB)
```

```
In [ ]: import os
        import glob
        import numpy as np
        import seaborn as sns
```

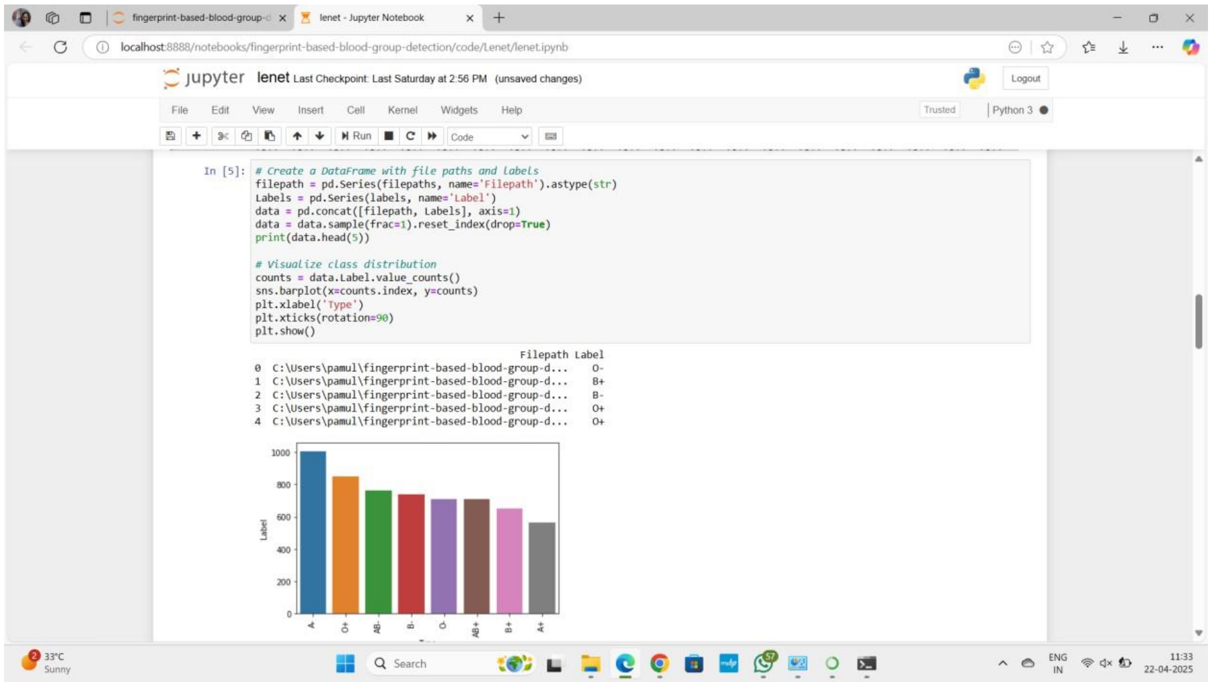
n



```
In [4]: # Get file paths for all images in the dataset
        filepaths = list(glob.glob(file_path + '/**/*.*'))

        # Example of printing a subset of file paths
        print(filepaths[-6:-1])

        # Extract labels from file paths
        labels = list(map(lambda x: os.path.split(os.path.split(x)[0])[1], filepaths))
        print(labels)
```

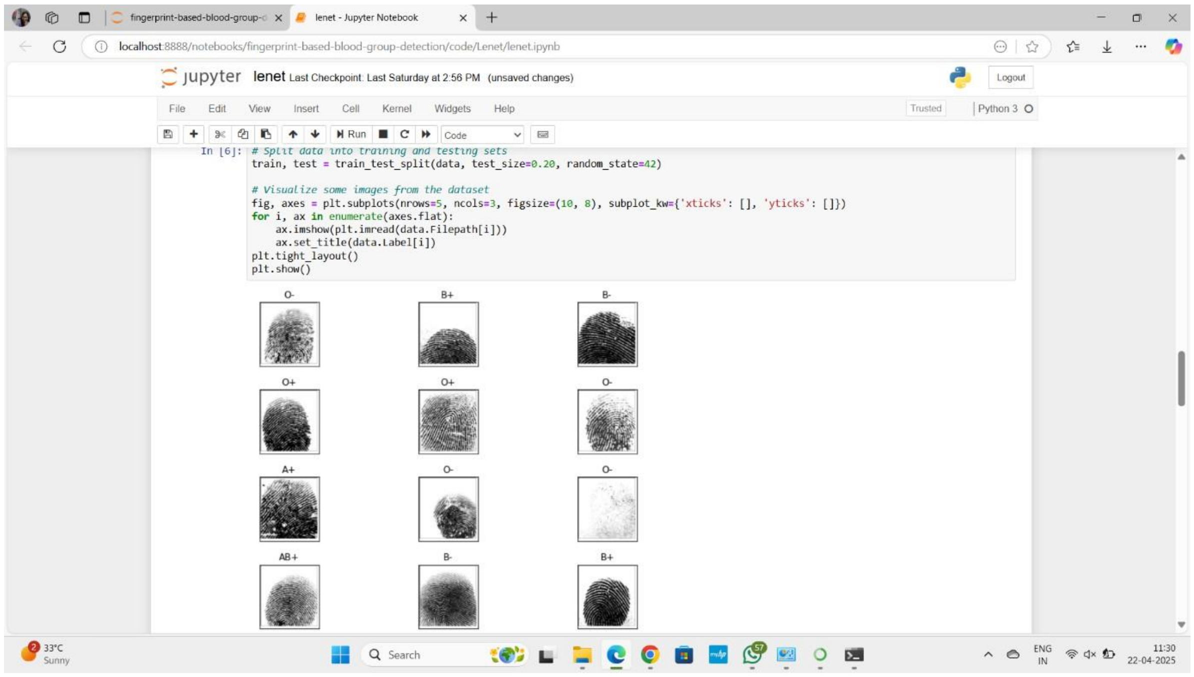
```
['A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A
+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+',
'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+',
'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+',
'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+',
'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+',
'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+',
'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+',
'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+',
'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+',
'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+',
'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+',
'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+',
'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+',
'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+',
'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+',
'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+',
'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+',
'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+', 'A+',
```

```
In [5]: # Create a DataFrame with file paths and labels
        filepath = pd.Series(filepaths, name='Filepath').astype(str)
        Labels = pd.Series(labels, name='Label')
        data = pd.concat([filepath, Labels], axis=1)
        data = data.sample(frac=1).reset_index(drop=True)
```

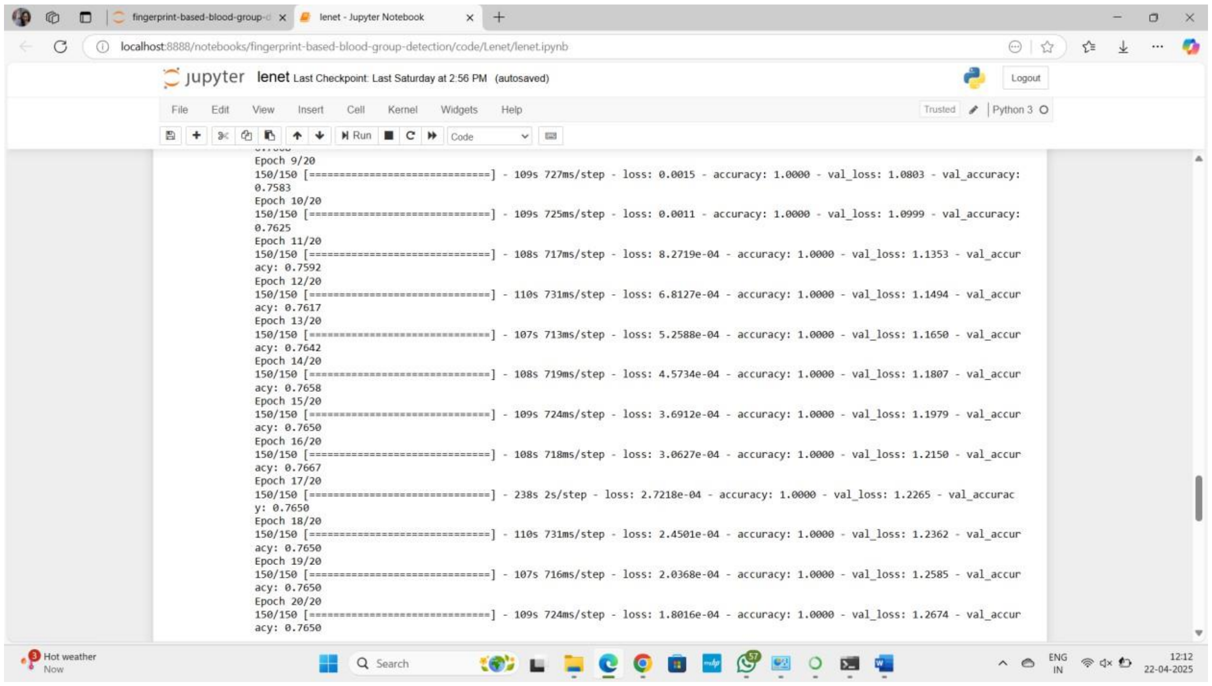


```
In [5]: # Create a DataFrame with file paths and labels
        filepath = pd.Series(filepaths, name='Filepath').astype(str)
        Labels = pd.Series(labels, name='Label')
        data = pd.concat([filepath, Labels], axis=1)
        data = data.sample(frac=1).reset_index(drop=True)
        print(data.head(5))

        # Visualize class distribution
        counts = data.Label.value_counts()
        sns.barplot(x=counts.index, y=counts)
        plt.xlabel('Type')
        plt.xticks(rotation=90)
        plt.show()
```

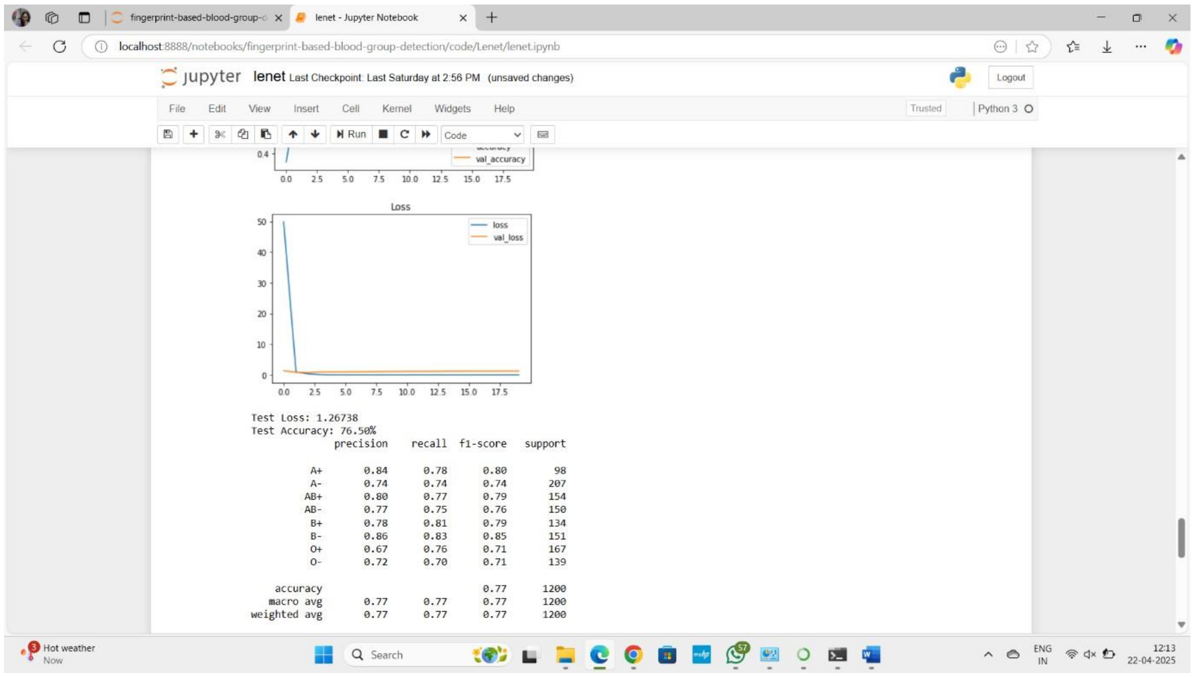
```
                                            Filepath Label
0  C:\Users\pamul\fingerprint-based-blood-group-d...    O-
1  C:\Users\pamul\fingerprint-based-blood-group-d...    B+
2  C:\Users\pamul\fingerprint-based-blood-group-d...    B-
3  C:\Users\pamul\fingerprint-based-blood-group-d...    O+
4  C:\Users\pamul\fingerprint-based-blood-group-d...    O+
```



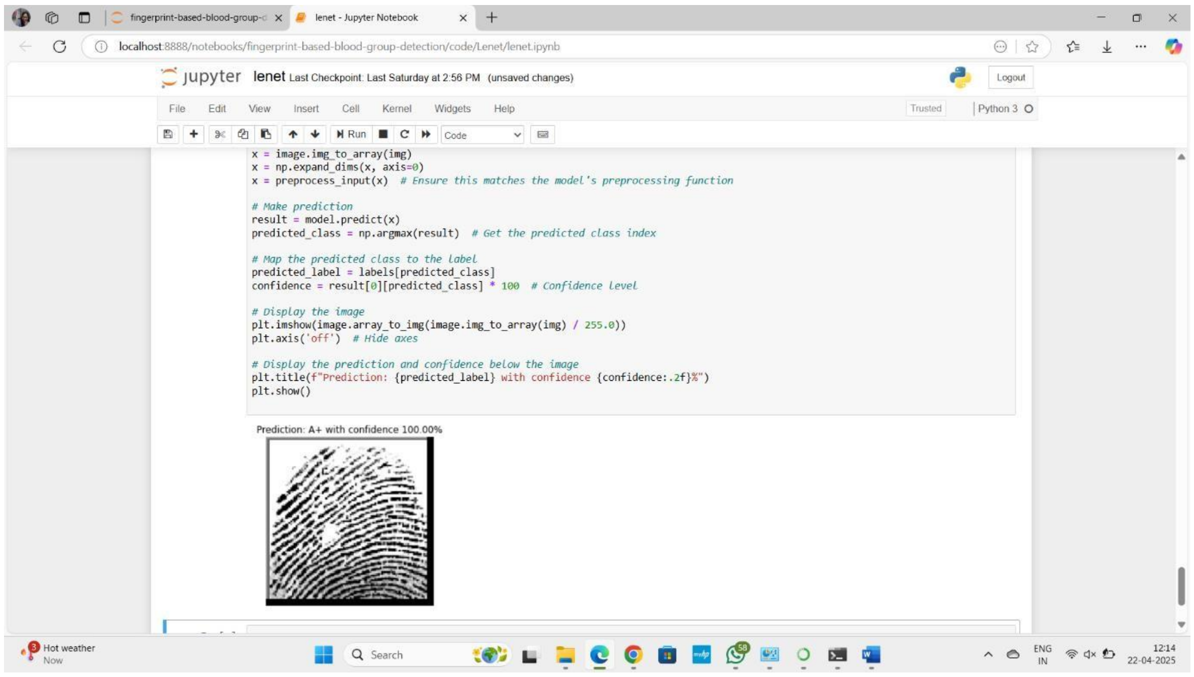
```
In [6]: # Split data into training and testing sets
        train, test = train_test_split(data, test_size=0.20, random_state=42)

        # Visualize some images from the dataset
        fig, axes = plt.subplots(nrows=5, ncols=3, figsize=(10, 8), subplot_kw={'xticks': [], 'yticks': []})
        for i, ax in enumerate(axes.flat):
            ax.imshow(plt.imread(data.Filepath[i]))
            ax.set_title(data.Label[i])
        plt.tight_layout()
        plt.show()
```



```
Epoch 9/20
150/150 [==============================] - 109s 727ms/step - loss: 0.0015 - accuracy: 1.0000 - val_loss: 1.0803 - val_accuracy:
0.7583
Epoch 10/20
150/150 [==============================] - 109s 725ms/step - loss: 0.0011 - accuracy: 1.0000 - val_loss: 1.0999 - val_accuracy:
0.7625
Epoch 11/20
150/150 [==============================] - 108s 717ms/step - loss: 8.2719e-04 - accuracy: 1.0000 - val_loss: 1.1353 - val_accur
acy: 0.7592
Epoch 12/20
150/150 [==============================] - 110s 731ms/step - loss: 6.8127e-04 - accuracy: 1.0000 - val_loss: 1.1494 - val_accur
acy: 0.7617
Epoch 13/20
150/150 [==============================] - 107s 713ms/step - loss: 5.2588e-04 - accuracy: 1.0000 - val_loss: 1.1650 - val_accur
acy: 0.7642
Epoch 14/20
150/150 [==============================] - 108s 719ms/step - loss: 4.5734e-04 - accuracy: 1.0000 - val_loss: 1.1807 - val_accur
acy: 0.7658
Epoch 15/20
150/150 [==============================] - 109s 724ms/step - loss: 3.6912e-04 - accuracy: 1.0000 - val_loss: 1.1979 - val_accur
acy: 0.7650
Epoch 16/20
150/150 [==============================] - 108s 718ms/step - loss: 3.0627e-04 - accuracy: 1.0000 - val_loss: 1.2150 - val_accur
acy: 0.7667
Epoch 17/20
150/150 [==============================] - 238s 2s/step - loss: 2.7218e-04 - accuracy: 1.0000 - val_loss: 1.2265 - val_accurac
y: 0.7650
Epoch 18/20
150/150 [==============================] - 110s 731ms/step - loss: 2.4501e-04 - accuracy: 1.0000 - val_loss: 1.2362 - val_accur
acy: 0.7650
Epoch 19/20
150/150 [==============================] - 107s 716ms/step - loss: 2.0368e-04 - accuracy: 1.0000 - val_loss: 1.2585 - val_accur
acy: 0.7650
Epoch 20/20
150/150 [==============================] - 109s 724ms/step - loss: 1.8016e-04 - accuracy: 1.0000 - val_loss: 1.2674 - val_accur
acy: 0.7650
```



```
Test Loss: 1.26738
Test Accuracy: 76.50%
              precision    recall  f1-score   support

          A+       0.84      0.78      0.80        98
          A-       0.74      0.74      0.74       207
         AB+       0.80      0.77      0.79       154
         AB-       0.77      0.75      0.76       150
          B+       0.78      0.81      0.79       134
          B-       0.86      0.83      0.85       151
          O+       0.67      0.76      0.71       167
          O-       0.72      0.70      0.71       139

    accuracy                           0.77      1200
   macro avg       0.77      0.77      0.77      1200
weighted avg       0.77      0.77      0.77      1200
```



```
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x)  # Ensure this matches the model's preprocessing function

# Make prediction
result = model.predict(x)
predicted_class = np.argmax(result)  # Get the predicted class index

# Map the predicted class to the label
predicted_label = labels[predicted_class]
confidence = result[0][predicted_class] * 100  # Confidence level

# Display the image
plt.imshow(image.array_to_img(image.img_to_array(img) / 255.0))
plt.axis('off')  # Hide axes

# Display the prediction and confidence below the image
plt.title(f"Prediction: {predicted_label} with confidence {confidence:.2f}%")
plt.show()
```

```
Prediction: A+ with confidence 100.00%
```

localhost:8888/notebooks/fingerprint-based-blood-group-detection/code/Lenet/lenet.ipynb
jupyter lenet Last Checkpoint: Last Saturday at 2:56 PM (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

10. CONCLUSION

The fingerprint-based blood group detection system presents a significant step forward in non-invasive, rapid, and cost-effective medical diagnostics. By leveraging the uniqueness of fingerprint patterns and the power of machine learning, the proposed system offers an innovative alternative to traditional blood group testing, which is often invasive, resource-intensive, and time-consuming. This project demonstrates that with the right combination of biometric analysis and artificial intelligence, it is possible to explore previously untapped relationships between physical traits and biological markers.

Through careful system design, feature extraction, and model training, the system is capable of predicting an individual's blood group based solely on fingerprint images with promising accuracy.The system can be especially useful in emergency situations, rural healthcare environments, and large-scale identification scenarios.

While the results are encouraging, it is also acknowledged that the system has certain limitations, including the need for larger datasets and deeper genetic analysis to improve accuracy and reliability further. Nonetheless, this work lays the groundwork for future research in biometric-based health diagnostics, where fingerprints may help predict not only blood groups but potentially other medically relevant information.

In conclusion, the proposed system represents a fusion of biometrics and artificial intelligence, aimed at enhancing healthcare delivery. With further development, validation, and integration it holds great potential to become a practical tool in modern medical and emergency response systems, revolutionizing how basic yet crucial health information is obtained.

Future Work

The current implementation of fingerprint-based blood group detection opens up multiple avenues for further research, development, and real-world application. While the system shows promising results, several enhancements can significantly improve its accuracy, reliability, and scalability in future iterations.

One key direction for future work is the **expansion of the dataset**. A larger and more diverse dataset with balanced representation across all blood groups and fingerprint patterns is essential for training more generalized models. Collaboration with hospitals, biometric centers, and research institutions can help build a standardized fingerprint-blood group dataset to enhance model training and validation.

Another important area involves the **integration of advanced deep learning models**. While initial models like SVM and Random Forest are effective, leveraging state-of-the-art architectures such as Convolutional Neural Networks (CNNs), Transfer Learning models, or hybrid deep learning approaches can significantly improve feature extraction and classification accuracy. These models can automatically learn complex patterns from raw fingerprint images, eliminating the need for manual feature engineering.

Additionally, **cross-modal analysis** could be explored by combining fingerprint data with other biometric or physiological data such as iris patterns, facial features,or genetic information to improve the reliability of predictions. This multimodal approach may lead to a more robust healthcare diagnostic tool that not only predicts blood group but could also aid in disease prediction or early health screening.

Future versions of the system can also be **integrated into portable health kits or mobile applications**, enabling real-time diagnostics in rural areas, disaster zones, or military environments. With advancements in embedded systems and edge computing, the system could run on low-power devices like Raspberry Pi or smartphones, making it truly accessible and scalable.

Another potential improvement is the **integration of multiple biometric traits**, such as combining fingerprint data with iris patterns or facial recognition features. This multimodal approach may increase the robustness and reliability of blood group predictions. Furthermore, the development of a real-time mobile or handheld application can make this system more accessible and practical for use in emergency situations, rural healthcare environments, and large-scale identification scenarios.

To further **validate the scientific basis of the system**, future research could explore correlations between genetic markers and fingerprint patterns, thereby strengthening the biological relationship underlying the model's predictions. Improved preprocessing techniques such as advanced noise reduction, image enhancement, and segmentation algorithms can also contribute to better input quality and model performance.

Finally, **regulatory validation and ethical compliance** should be part of future development. Clinical trials and medical certifications will be necessary for real- world deployment. Ensuring data privacy, biometric security, and user consent will be critical to gaining user trust and meeting data protection regulations.

11. REFERENCES

- Ahmed, A. A., & Osman, I. H. (2016). *Correlation between Fingerprint Patterns and ABO Blood Group*. International Journal of Science and Research (IJSR), 5(3), 1413–1416.
- Kanchan, T., Krishan, K., & Sharma, A. (2010). *Relationship of fingerprint patterns with blood group and gender – A study in a North Indian population*. Journal of Forensic and Legal Medicine, 17(8), 431–435.
- Jain, A. K., Flynn, P., & Ross, A. (2007). *Handbook of Biometrics*. Springer Science & Business Media.
- Mallikarjun, B. (2014). *Fingerprint Classification and Matching for Identification*. International Journal of Innovative Research in Science, Engineering and Technology, 3(3), 10143–10149.
- Gonzalez, R. C., & Woods, R. E. (2018). *Digital Image Processing* (4th ed.). Pearson Education.
- Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.
- Cortes, C., & Vapnik, V. (1995). *Support-vector networks*. Machine Learning, 20(3), 273–297.
- OpenCV Library
- Scikit-learn: Machine Learning in Python
- TensorFlow: Open-source platform for machine learning