**Chest Disease Detection using CNN AI Algorithm**

**1. INTRODUCTION**

X-ray radiographies are an affordable and non-invasive method of examining different organs of the body. Recognized as a valuable diagnosis tool for many disorders and abnormalities, X-rays can also be used to monitor diseases during treatment. Around 3.6 billion X-ray images are taken every year worldwide. This number includes over 150 million chest X-ray radiographies (CXR) performed in the United States only. The World Health Organization (WHO) stated that CXRs are the most commonly performed clinical imaging technique worldwide. CXRs are grayscale images generally produced by projecting X-rays onto the human body positioned against a metallic plate. Although CXRs play a crucial role in the diagnosis of thoracic disease, visual inspection by radiologists remains complex and error-prone. Previous studies have shown that the risk of misdiagnosis increases with the amount of time it takes for a radiologist to interpret CXR images. In addition, even experienced radiologists were at greater risk of misdiagnosis because of hidden lesions and symptoms in soft tissue and bones. The WHO reports that many chest diseases can be life threatening and lead to the death of millions of people if not accurately and timely treated. Some chest diseases are of high mortality rates such as tuberculosis that kills around 1.4 million people annually, pneumonia that kills 9 million children under the age of 5 years being the world’s leading killer disease, and COVID-19 which caused the death of over 6 million people all over the world as of November 2022.

* 1. **Objective of the Project**

Pneumonia is a life-threatening infectious disease affecting one or both lungs in humans commonly caused by bacteria called Streptococcus pneumoniae. One in three deaths in India is caused due to pneumonia as reported by World Health Organization (WHO). Chest X-Rays which are used to diagnose pneumonia need expert radiotherapists for evaluation. Thus, developing an automatic system for detecting pneumonia would be beneficial for treating the disease without any delay particularly in remote areas. Due to the success of deep learning algorithms in analyzing medical images, Convolutional Neural Networks (CNNs) have gained much attention for disease classification. In addition, features learned by pre-trained CNN models on large-scale datasets are much useful in image classification tasks. In this work, we appraise the functionality of pre-trained CNN models utilized as feature-extractors followed by different classifiers for the classification of abnormal and normal chest X-Rays. We analytically determine the optimal CNN model for the purpose. Statistical results obtained demonstrates that pretrained CNN models employed along with supervised classifier algorithms can be very beneficial in analyzing chest X-ray images, specifically to detect Pneumonia.

**2.LITERATURE SURVEY**

**"Machine learning approach for automated screening of malaria parasite using light microscopic images"**

Dev Kumar Das, Madhumala Ghosh, Mallika Pal, Asok K Maiti and Chandan Chakraborty, 2013.

The aim of this paper is to address the development of computer assisted malaria parasite characterization and classification using machine learning approach based on light microscopic images of peripheral blood smears. In doing this, microscopic image acquisition from stained slides, illumination correction and noise reduction, erythrocyte segmentation, feature extraction, feature selection and finally classification of different stages of malaria (Plasmodium vivax and Plasmodium falciparum) have been investigated. The erythrocytes are segmented using marker controlled watershed transformation and subsequently total ninety six features describing shape-size and texture of erythrocytes are extracted in respect to the parasitemia infected versus non-infected cells. Ninety four features are found to be statistically significant in discriminating six classes. Here a feature selection-cum-classification scheme has been devised by combining F-statistic, statistical learning techniques i.e., Bayesian learning and support vector machine (SVM) in order to provide the higher classification accuracy using best set of discriminating features. Results show that Bayesian approach provides the highest accuracy i.e., 84% for malaria classification by selecting 19 most significant features while SVM provides highest accuracy i.e., 83.5% with 9 most significant features. Finally, the performance of these two classifiers under feature selection framework has been compared toward malaria parasite classification.

**"Image analysis and machine learning for detecting malaria"**

Mahdieh Poostchi, Kamolrat Silamut, Richard Maude, Stefan Jaeger and George Thoma, 2018.

Malaria remains a major burden on global health, with roughly 200 million cases worldwide and more than 400,000 deaths per year. Besides biomedical research and political efforts, modern information technology is playing a key role in many attempts at fighting the disease. One of the barriers toward a successful mortality reduction has been inadequate malaria diagnosis in particular. To improve diagnosis, image analysis software and machine learning methods have been used to quantify parasitemia in microscopic blood slides. This article gives an overview of these techniques and discusses the current developments in image analysis and machine learning for microscopic malaria diagnosis. We organize the different approaches published in the literature according to the techniques used for imaging, image preprocessing, parasite detection and cell segmentation, feature computation, and automatic cell classification. Readers will find the different techniques listed in tables, with the relevant articles cited next to them, for both thin and thick blood smear images. We also discussed the latest developments in sections devoted to deep learning and smartphone technology for future malaria diagnosis.

**"Automated image processing method for the diagnosis and classification of malaria on thin blood smears"**

Nicholas E Ross, Charles J Pritchard, David M Rubin and Adriano G Duse, 2006.

Malaria is a serious global health problem, and rapid, accurate diagnosis is required to control the disease. An image processing algorithm to automate the diagnosis of malaria on thin blood smears is developed. The image classification system is designed to positively identify malaria parasites present in thin blood smears, and differentiate the species of malaria. Images are acquired using a charge-coupled device camera connected to a light microscope. Morphological and novel threshold selection techniques are used to identify erythrocytes (red blood cells) and possible parasites present on microscopic slides. Image features based on colour, texture and the geometry of the cells and parasites are generated, as well as features that make use of a priori knowledge of the classification problem and mimic features used by human technicians. A two-stage tree classifier using backpropogation feedforward neural networks distinguishes between true and false positives, and then diagnoses the species (Plasmodium falciparum, P. vivax, P. ovale or P. malariae) of the infection. Malaria samples obtained from the Department of Clinical Microbiology and Infectious Diseases at the University of the Witwatersrand Medical School are used for training and testing of the system. Infected erythrocytes are positively identified with a sensitivity of 85% and a positive predictive value (PPV) of 81%, which makes the method highly sensitive at diagnosing a complete sample provided many views are analysed. Species were correctly determined for 11 out of 15 samples.

**"CNN features off-the-shelf: an astounding baseline for recognition"**

Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan and Stefan Carlsson, 2014.

Recent results indicate that the generic descriptors extracted from the convolutional neural networks are very powerful. This paper adds to the mounting evidence that this is indeed the case. We report on a series of experiments conducted for different recognition tasks using the publicly available code and model of the OverFeat network which was trained to perform object classification on ILSVRC13. We use features extracted from the OverFeat network as a generic image representation to tackle the diverse range of recognition tasks of object image classification, scene recognition, fine grained recognition, attribute detection and image retrieval applied to a diverse set of datasets. We selected these tasks and datasets as they gradually move further away from the original task and data the OverFeat network was trained to solve. Astonishingly, we report consistent superior results compared to the highly tuned state-of-the-art systems in all the visual classification tasks on various datasets. For instance retrieval it consistently outperforms low memory footprint methods except for sculptures dataset. The results are achieved using a linear SVM classifier (or L2 distance in case of retrieval) applied to a feature representation of size 4096 extracted from a layer in the net. The representations are further modified using simple augmentation techniques e.g. jittering. The results strongly suggest that features obtained from deep learning with convolutional nets should be the primary candidate in most visual recognition tasks.

**"Imagenet classification with deep convolutional neural networks"**

Alex Krizhevsky, Ilya Sutskever and Geoffrey E Hinton, 2012.

We trained a large, deep convolutional neural network to classify the 1.3 million high-resolution images in the LSVRC-2010 ImageNet training set into the 1000 different classes. On the test data, we achieved top-1 and top-5 error rates of 39.7\% and 18.9\% which is considerably better than the previous state-of-the-art results. The neural network, which has 60 million parameters and 500,000 neurons, consists of five convolutional layers, some of which are followed by max-pooling layers, and two globally connected layers with a final 1000-way softmax. To make training faster, we used non-saturating neurons and a very efficient GPU implementation of convolutional nets. To reduce overfitting in the globally connected layers we employed a new regularization method that proved to be very effective.

**"Very deep convolutional networks for large-scale image recognition"**

Karen Simonyan and Andrew Zisserman, 2014.

In this work we investigate the effect of the convolutional network depth on its accuracy in the large-scale image recognition setting. Our main contribution is a thorough evaluation of networks of increasing depth, which shows that a significant improvement on the prior-art configurations can be achieved by pushing the depth to 16-19 weight layers. These findings were the basis of our ImageNet Challenge 2014 submission, where our team secured the first and the second places in the localisation and classification tracks respectively.

**"Xception: deep learning with separable convolutions"**

F Chollet, 2016.

Malaria is a blood disease caused by the Plasmodium parasites transmitted through the bite of female Anopheles mosquito. Microscopists commonly examine thick and thin blood smears to diagnose disease and compute parasitemia. However, their accuracy depends on smear quality and expertise in classifying and counting parasitized and uninfected cells. Such an examination could be arduous for large-scale diagnoses resulting in poor quality. State-of-the-art image-analysis based computeraided diagnosis (CADx) methods using machine learning (ML) techniques, applied to microscopic images of the smears using hand-engineered features demand expertise in analyzing morphological, textural, and positional variations of the region of interest (ROI). In contrast, Convolutional Neural Networks (CNN), a class of deep learning (DL) models promise highly scalable and superior results with end-to-end feature extraction and classification. Automated malaria screening using DL techniques could, therefore, serve as an effective diagnostic aid. In this study, we evaluate the performance of pre-trained CNN based DL models as feature extractors toward classifying parasitized and uninfected cells to aid in improved disease screening. We experimentally determine the optimal model layers for feature extraction from the underlying data. Statistical validation of the results demonstrates the use of pre-trained CNNs as a promising tool for feature extraction for this purpose.

**"Deep residual learning for image recognition"**

Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun, 2016.

Deeper neural networks are more difficult to train. We present a residual learning framework to ease the training of networks that are substantially deeper than those used previously. We explicitly reformulate the layers as learning residual functions with reference to the layer inputs, instead of learning unreferenced functions. We provide comprehensive empirical evidence showing that these residual networks are easier to optimize, and can gain accuracy from considerably increased depth. On the ImageNet dataset we evaluate residual nets with a depth of up to 152 layers - 8× deeper than VGG nets [40] but still having lower complexity. An ensemble of these residual nets achieves 3.57% error on the ImageNet test set. This result won the 1st place on the ILSVRC 2015 classification task. We also present analysis on CIFAR-10 with 100 and 1000 layers. The depth of representations is of central importance for many visual recognition tasks. Solely due to our extremely deep representations, we obtain a 28% relative improvement on the COCO object detection dataset. Deep residual nets are foundations of our submissions to ILSVRC & COCO 2015 competitions1, where we also won the 1st places on the tasks of ImageNet detection, ImageNet localization, COCO detection, and COCO segmentation.

**"Densely Connected Convolutional Networks"**

Gao Huang, Zhuang Liu, Laurens Van Der Maaten and Kilian Q Weinberger, 2017

Recent work has shown that convolutional networks can be substantially deeper, more accurate, and efficient to train if they contain shorter connections between layers close to the input and those close to the output. In this paper, we embrace this observation and introduce the Dense Convolutional Network (DenseNet), which connects each layer to every other layer in a feed-forward fashion. Whereas traditional convolutional networks with L layers have L connections - one between each layer and its subsequent layer - our network has L(L+1)/2 direct connections. For each layer, the feature-maps of all preceding layers are used as inputs, and its own feature-maps are used as inputs into all subsequent layers. DenseNets have several compelling advantages: they alleviate the vanishing-gradient problem, strengthen feature propagation, encourage feature reuse, and substantially reduce the number of parameters. We evaluate our proposed architecture on four highly competitive object recognition benchmark tasks (CIFAR-10, CIFAR-100, SVHN, and ImageNet). DenseNets obtain significant improvements over the state-of-the-art on most of them, whilst requiring less memory and computation to achieve high performance. Code and models are available at https://github.com/liuzhuang13/DenseNet.

**"Classification using deep learning neural networks for brain tumors"**

Heba Mohsen, El-Sayed A El-Dahshan, El-Sayed M El-Horbaty and AbdelBadeeh M Salem, 2017**.**

Pneumonia is a life-threatening infectious disease affecting one or both lungs in humans commonly caused by bacteria called Streptococcus pneumonia. The present study aimed to examine the risk factors for death due to pneumonia in young children. One or more in three deaths in Asia is caused due to pneumonia as reported by World Health Organization (WHO). Chest X-Rays which are used to diagnose pneumonia need expert radiotherapists for evaluation. Thus, developing an automatic system for detecting pneumonia would be beneficial and it can save lots of peoples life and help stopping and curing and controll for treating the disease without any delay particularly in remote areas. Due to the success of deep learning algorithms in analyzing medical images, Convolutional Neural Networks (CNNs) have gained much attention for disease classification. In addition, features learned by pre-trained CNN models on large-scale datasets are much useful in image classification tasks. In this work, we appraise the functionality of pre-trained CNN models utilized as feature-extractors followed by different classifiers for the classification of abnormal and normal chest X-Rays. We analytically determine the optimal CNN model for the purpose. Statistical results obtained demonstrates that pretrained CNN models employed along with supervised classifier algorithms can be very beneficial in analyzing chest X-ray images, specifically to detect Pneumonia

**3. SYSTEM ANALYSIS**

**3.1 Existing System**

In recent time, exploration of Machine learning (ML) algorithms in detecting thoracic diseases has gained attention in research area of medical image classification. Over the recent years, Computer Aided Designs (CAD) have become the major research domain in machine learning. The subsisting CAD systems have already been proved to facilitate the medical area primarily in detection of breast cancer, mammograms, lung nodules etc. In the procedure of employing Machine Learning (ML) techniques to medical images, significant features are of uppermost importance. For this reason, most of the previous algorithms used hand crafted features for developing CAD systems based on examining images. However, the hand crafted features with limitations varying according to tasks were not capable of supplying much meaningful features.

**Disadvantages**

1. It takes more time
2. Less accuracy

**3.2 Proposed System**

In this project as per your request we have employed CNN and traditional machine learning algorithms like Random Forest and Decision tree to detect various viral chest disease infections. Each algorithm performance is evaluated in terms of precision, recall, accuracy, confusion matrix and FSCORE. Among all algorithms CNN is giving best detection accuracy. The primary goal of using Convolutional Neural Network in most of the image classification tasks is to reduce the computational complexity of the model which is likely to increase if the input are images . The original 3-channel images were resized from 1024×1024 into 224×224 pixels to reduce the heavy computation and for faster processing. All of the further techniques has been applied over these downsized images.

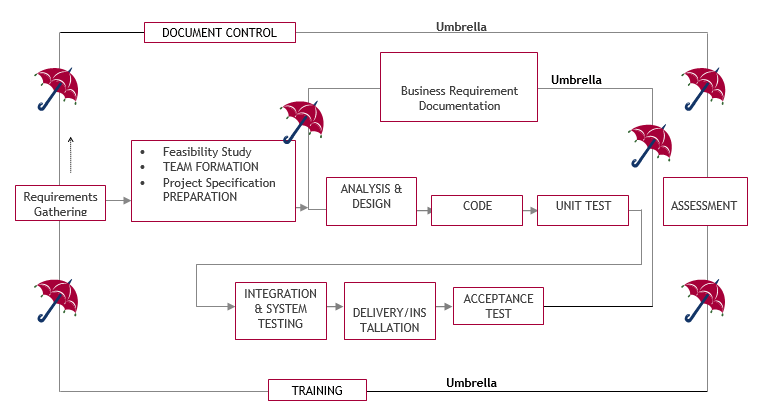
**Advantages**

1.It takes less time

2.High prediction result

**3.4. Process Model Used With Justification**

**SDLC (Umbrella Model):**

****

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

**Stages in SDLC:**

1. Requirement Gathering
2. Analysis
3. Designing
4. Coding
5. Testing
6. Maintenance
7. **Requirements Gathering** **stage:**

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



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

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

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

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

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

1. **Analysis Stage:**

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



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

1. **Designing Stage:**

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



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

1. **Development (Coding) Stage:**

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



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

1. **Integration & Test Stage:**

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



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

**Installation & Acceptance Test:**

During the installation and acceptance stage, the software artefacts, online help, and initial production data are loaded onto the production server. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite is a prerequisite to acceptance of the software by the customer. After customer personnel have verified that the initial production data load is correct and the test suite has been executed with satisfactory results, the customer formally accepts the delivery of the software.



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

1. **Maintenance:**

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

**3.5. Software Requirement Specification**

**3.5. 1. Overall Description**

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

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

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

* **ECONOMIC FEASIBILITY**

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

* **Operational Feasibility**

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

* **TECHNICAL FEASIBILITY**

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

**3.5. 2. External Interface Requirements**

1. **User Interface**

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

1. **Hardware Interfaces**

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

1. **Software Interfaces**

The required software is python.

1. **Operating Environment**

Windows XP.

1. **Hardware Requirements:**

# Processor - Intel i3(min)

* Speed - 1.1 GHz
* RAM - 256 MB(min)
* Hard Disk - 20 GB
* Key Board - Standard Windows Keyboard
* Mouse - Two or Three Button Mouse

1. **Software Requirements:**

* Operating System - Windows7/8
* Programming Language - Python

**4. SYSTEM DESIGN**

**4.1 UML Diagram:**

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

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

* + **User Model View**

1. This view represents the system from the user’s perspective.
2. The analysis representation describes a usage scenario from the end-user’s perspective.
   * **Structural Model view**
3. In this model the data and functionality are arrived from inside the system.
4. This model view models the static structures.

* **Behavioural Model View**

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

* **Implementation Model View**

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

* **Environmental Model View**

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

1. **Class Diagram:**

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

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



1. **Use case Diagram:**

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



1. **Sequence diagram:**

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



1. **Collaboration diagram:**

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



1. **Component Diagram:**

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

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



1. **Deployment Diagram:**

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

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

****

1. **Activity Diagram:**

Activity diagram is another important diagram in UML to describe dynamic aspects of the system. It is basically a flow chart to represent the flow form one activity to another activity. The activity can be described as an operation of the system. So the control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent.

load Dataset()

Preprocessing()

train CNN()

train Random forest()

Performance Graph()

train Decision tree()

Predict()

1. **Data Flow Diagram:**

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

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

User

1. load Dataset 2. Successfully loaded Dataset

3. Preprocessing 4. Successfully Preprocessing

5. train CNN 6. Successfully train CNN

7. train Random forest 8. Successfully train Random forest

9. train Decision tree 10. Succesfully train Decision tree

11. Performance Graph 12. Successfully Performance Graph

13. Predict 14. Successfully Predict

**5. IMPLEMETATION**

**5.1 Python**

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.

**5.1.1 History of Python:**

Python was developed by Guido van Rossum 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](https://www.tutorialspoint.com/cprogramming/index.htm), [C++](https://www.tutorialspoint.com/cplusplus/index.htm), Algol-68, Smalltalk, and [Unix](https://www.tutorialspoint.com/unix/index.htm) shell and other [scripting languages](https://www.tutorialspoint.com/scripting_language_tutorials.htm).

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

For many uninitiated people, the word Python is related to a species of snake. Rossum though attributes the choice of the name Python to a popular comedy series **Monty Python's Flying Circus** on BBC.

Being the principal architect of Python, the developer community conferred upon him the title of **Benevolent Dictator for Life** (BDFL). However, in 2018, Rossum relinquished the title. Thereafter, the development and distribution of the reference implementation of Python is handled by a nonprofit organization **Python Software Foundation**.

* + 1. **Features of Python:**

1. **Easy to Learn**

This is one of the most important reasons for the popularity of Python. Python has a limited set of keywords. Its features such as simple [syntax](https://www.tutorialspoint.com/python/python_basic_syntax.htm), usage of indentation to avoid clutter of curly brackets and dynamic typing that doesn't necessitate prior declaration of variable help a beginner to learn Python quickly and easily.

1. **Dynamically Typed**

Python is a dynamically typed programming language. In Python, you don't need to specify the variable time at the time of the variable declaration. The types are specified at the runtime based on the assigned value due to its dynamically typed feature.

1. **Interpreter Based**

Instructions in any programming languages must be translated into machine code for the processor to execute them. Programming languages are either compiler based or interpreter based.

In case of a compiler, a [machine language](https://www.tutorialspoint.com/machine_learning/index.htm) version of the entire source program is generated. The conversion fails even if there is a single erroneous statement. Hence, the development process is tedious for the beginners. The C family languages (including [C](https://www.tutorialspoint.com/cprogramming/index.htm), [C++](https://www.tutorialspoint.com/cplusplus/index.htm), [Java](https://www.tutorialspoint.com/java/index.htm), [C#](https://www.tutorialspoint.com/csharp/index.htm) etc) are compiler based.

Python is an interpreter based language. The interpreter takes one instruction from the source code at a time, translates it into machine code and executes it. Instructions before the first occurrence of error are executed. With this feature, it is easier to debug the program and thus proves useful for the beginner level programmer to gain confidence gradually. Python therefore is a beginner-friendly language.

1. **Interactive**

Standard Python distribution comes with an interactive shell that works on the principle of REPL (Read – Evaluate – Print – Loop). The shell presents a Python prompt >>>. You can type any valid Python expression and press Enter. Python interpreter immediately returns the response and the prompt comes back to read the next expression.

>>> 2\*3+1

7

>>> print ("Hello World")

Hello World

The interactive mode is especially useful to get familiar with a library and test out its functionality. You can try out small code snippets in interactive mode before writing a program.

1. **Multi-paradigm**

Python is a completely [object-oriented](https://www.tutorialspoint.com/python/python_oops_concepts.htm) language. Everything in a Python program is an [object](https://www.tutorialspoint.com/python/python_object_classes.htm). However, Python conveniently encapsulates its object orientation to be used as an imperative or procedural language – such as C. Python also provides certain functionality that resembles functional programming. Moreover, certain third-party tools have been developed to support other programming paradigms such as aspect-oriented and logic programming.

1. **Standard Library**

Even though it has a very few keywords (only Thirty Five), Python software is distributed with a standard library made of large number of modules and packages. Thus Python has out of box support for programming needs such as serialization, data compression, internet data handling, and many more. Python is known for its batteries included approach.

Some of the Python's popular modules are:

* [NumPy](https://www.tutorialspoint.com/numpy/index.htm)
* [Pandas](https://www.tutorialspoint.com/python_pandas/index.htm)
* [Matplotlib](https://www.tutorialspoint.com/matplotlib/index.htm)
* Tkinter
* [Math](https://www.tutorialspoint.com/python/python_maths.htm)

## **Open Source and Cross Platform**

Python's standard distribution can be downloaded from <https://www.python.org/downloads/> without any restrictions. You can download pre-compiled binaries for various operating system platforms. In addition, the source code is also freely available, which is why it comes under open source category.

Python software (along with the documentation) is distributed under Python Software Foundation License. It is a BSD style permissive software license and compatible to GNU GPL (General Public License).

Python is a cross-platform language. Pre-compiled binaries are available for use on various operating system platforms such as [Windows](https://www.tutorialspoint.com/windows10/index.htm), [Linux](https://www.tutorialspoint.com/unix/index.htm), Mac OS, [Android OS](https://www.tutorialspoint.com/android/index.htm). The reference implementation of Python is called CPython and is written in C. You can download the source code and compile it for your OS platform.

A Python program is first compiled to an intermediate platform independent byte code. The virtual machine inside the interpreter then executes the byte code. This behaviour makes Python a cross-platform language, and thus a Python program can be easily ported from one OS platform to other.

## **GUI Applications**

Python's standard distribution has an excellent graphics library called TKinter. It is a Python port for the vastly popular GUI toolkit called TCL/Tk. You can build attractive user-friendly GUI applications in Python. GUI toolkits are generally written in C/C++. Many of them have been ported to Python. Examples are [PyQt](https://www.tutorialspoint.com/pyqt/index.htm" \t "_blank), [WxWidgets](https://www.tutorialspoint.com/wxpython/index.htm" \t "_blank), [PySimpleGUI](https://www.tutorialspoint.com/pysimplegui/index.htm" \t "_blank) etc.

## **Database Connectivity**

Almost any type of database can be used as a backend with the Python application. DB-API is a set of specifications for database driver software to let Python communicate with a relational database. With many third party libraries, Python can also work with NoSQL databases such as [MongoDB](https://www.tutorialspoint.com/mongodb/index.htm).

## **Extensible**

The term extensibility implies the ability to add new features or modify existing features. As stated earlier, CPython (which is Python's reference implementation) is written in C. Hence one can easily write modules/libraries in C and incorporate them in the standard library. There are other implementations of Python such as Jython (written in Java) and [IPython](https://www.tutorialspoint.com/jupyter/ipython_introduction.htm" \t "_blank) (written in C#). Hence, it is possible to write and merge new functionality in these implementations with Java and C# respectively.

## **Active Developer Community**

As a result of Python's popularity and open-source nature, a large number of Python developers often interact with online forums and conferences. Python Software Foundation also has a significant member base, involved in the organization's mission to "**Promote, Protect, and Advance the Python Programming Language**"

Python also enjoys a significant institutional support. Major IT companies Google, Microsoft, and Meta contribute immensely by preparing documentation and other resources.

**5.1.3 Applications of Python:**

[Python](https://www.tutorialspoint.com/python/python_overview.htm) is a general-purpose programming language. It is suitable for the development of a wide range of software applications. Over the last few years Python has been the preferred language of choice for developers in the following application areas −

* [Data Science](https://www.tutorialspoint.com/python/python_application_areas.htm#data_science)
* [Machine Learning](https://www.tutorialspoint.com/python/python_application_areas.htm#machine_learning)
* [Web Development](https://www.tutorialspoint.com/python/python_application_areas.htm#web_development)
* [Computer Vision and Image processing](https://www.tutorialspoint.com/python/python_application_areas.htm#computer_vision_and_image_processing)
* [Embedded Systems and IoT](https://www.tutorialspoint.com/python/python_application_areas.htm#embedded_systems_and_iot)
* [Job Scheduling and Automation](https://www.tutorialspoint.com/python/python_application_areas.htm#job_scheduling_and_automation)
* [Desktop GUI Applications](https://www.tutorialspoint.com/python/python_application_areas.htm#desktop_gui_applications)
* [Console-based Applications](https://www.tutorialspoint.com/python/python_application_areas.htm#console_based_applications)
* [CAD Applications](https://www.tutorialspoint.com/python/python_application_areas.htm#cad_applications)
* [Game Development](https://www.tutorialspoint.com/python/python_application_areas.htm#game_development)

## **Data Science**

Python's recent meteoric rise in the popularity charts is largely due to its Data science libraries. Python has become an essential skill for data scientists. Today, real time web applications, mobile applications and other devices generate huge amount of data. Python's data science libraries help companies generate business insights from this data.

Libraries like [NumPy](https://www.tutorialspoint.com/numpy/index.htm), [Pandas](https://www.tutorialspoint.com/python_pandas/index.htm), and [Matplotlib](https://www.tutorialspoint.com/matplotlib/index.htm) are extensively used to apply mathematical algorithms to the data and generate [visualizations](https://www.tutorialspoint.com/python_pandas/python_pandas_visualization.htm). Commercial and community Python distributions like Anaconda and ActiveState bundle all the essential libraries required for data science.

1. **Machine Learning**

Python libraries such as [Scikit-learn](https://www.tutorialspoint.com/scikit_learn/index.htm) and [TensorFlow](https://www.tutorialspoint.com/tensorflow/index.htm) help in building models for prediction of trends like customer satisfaction, projected values of stocks etc. based upon the past data. [Machine learning](https://www.tutorialspoint.com/machine_learning/index.htm) applications include (but not restricted to) medical diagnosis, statistical arbitrage, basket analysis, sales prediction etc.

1. **Web Development**

Python's web frameworks facilitate rapid web application development. [Django](https://www.tutorialspoint.com/django/index.htm), [Pyramid](https://www.tutorialspoint.com/python_pyramid/index.htm), [Flask](https://www.tutorialspoint.com/flask/index.htm) are very popular among the web developer community. etc. make it very easy to develop and deploy simple as well as complex web applications.

Latest versions of Python provide asynchronous programming support. Modern web frameworks leverage this feature to develop fast and high performance web apps and APIs.

1. **Computer Vision and Image processing**

[OpenCV](https://www.tutorialspoint.com/opencv_python/index.htm) is a widely popular library for capturing and processing images. Image processing algorithms extract information from images, reconstruct image and video data. Computer Vision uses image processing for face detection and pattern recognition. OpenCV is a C++ library. Its Python port is extensively used because of its rapid development feature.

Some of the application areas of computer vision are [robotics](https://www.tutorialspoint.com/artificial_intelligence/artificial_intelligence_robotics.htm), industrial surveillance, automation, and [biometrics](https://www.tutorialspoint.com/biometrics/index.htm) etc

* 1. **Python Libraries**

1. **NumPy**

**NumPy**, which stands for **Numerical Python**, is an open-source [Python](https://www.tutorialspoint.com/python/index.htm) library consisting of multidimensional and single-dimensional array elements. It's a standard that computes numerical data in Python. NumPy is most widely used in almost every domain where numerical computation is required, like science and engineering; hence, the NumPy API functionalities are highly utilized in data science and scientific [Python packages](https://www.tutorialspoint.com/python/python_packages.htm), including [Pandas](https://www.tutorialspoint.com/python_pandas/index.htm), [SciPy](https://www.tutorialspoint.com/scipy/index.htm), [Matplotlib](https://www.tutorialspoint.com/matplotlib/index.htm), [scikit-learn](https://www.tutorialspoint.com/scikit_learn/index.htm), scikit-image, and many more.

**Why NumPy - Need of NumPy**

NumPy is a fundamental package for numerical computation in Python. It provides mathematical functions to compute data as well as functions to operate multi-dimensional arrays and matrices efficiently. Here are some reasons why NumPy is essential:

* NumPy includes a wide range of mathematical functions for basic arithmetic, linear algebra, Fourier analysis, and more.
* NumPy performs numerical operations on large datasets efficiently.
* NumPy supports multi-dimensional arrays, allowing for the representation of complex data structures such as images, sound waves, and tensors in [machine learning](https://www.tutorialspoint.com/machine_learning/index.htm) models.
* It supports the writing of concise and readable code for complex mathematical computations.
* NumPy integrates with other libraries to do scientific computation; these are SciPy (for scientific computing), Pandas (for data manipulation and analysis), and scikit-learn (for machine learning).
* Many scientific and numerical computing libraries and tools are built on top of NumPy.
* Its widespread adoption and stability make it a standard choice for numerical computing tasks.

NumPy plays a crucial role in the Python ecosystem for scientific computing, data analysis, machine learning, and more. Its efficient array operations and extensive mathematical functions make it an indispensable tool for working with numerical data in Python.

**Installation command:**

**pip install numpy==1.19.2**

**Example:**

import numpy as np

a = np.array([0, 30, 45, 60, 90])

print('Array containing sine values:')

sin = np.sin(a \* np.pi / 180)

print(sin)

print('\n')

print('Compute sine inverse of angles. Returned values are in radians.')

inv = np.arcsin(sin)

print(inv)

print('\n')

print('Check result by converting to degrees:')

print(np.degrees(inv))

print('\n')

print('arccos and arctan functions behave similarly:')

cos = np.cos(a \* np.pi / 180)

print(cos)

print('\n')

print('Inverse of cos:')

inv = np.arccos(cos)

print(inv)

print('\n')

print('In degrees:')

print(np.degrees(inv))

print('\n')

print('Tan function:')

tan = np.tan(a \* np.pi / 180)

print(tan)

print('\n')

print('Inverse of tan:')

inv = np.arctan(tan)

print(inv)

print('\n')

print('In degrees:')

print(np.degrees(inv))

**Its output is as follows −**

Array containing sine values:

[ 0. 0.5 0.70710678 0.8660254 1. ]

Compute sine inverse of angles. Returned values are in radians.

[ 0. 0.52359878 0.78539816 1.04719755 1.57079633]

Check result by converting to degrees:

[ 0. 30. 45. 60. 90.]

arccos and arctan functions behave similarly:

[ 1.00000000e+00 8.66025404e-01 7.07106781e-01 5.00000000e-01

6.12323400e-17]

Inverse of cos:

[ 0. 0.52359878 0.78539816 1.04719755 1.57079633]

In degrees:

[ 0. 30. 45. 60. 90.]

Tan function:

[ 0.00000000e+00 5.77350269e-01 1.00000000e+00 1.73205081e+00

1.63312394e+16]

Inverse of tan:

[ 0. 0.52359878 0.78539816 1.04719755 1.57079633]

In degrees:

[ 0. 30. 45. 60. 90.]

1. **Pandas**

**Pandas** is an open-source, BSD-licensed Python library providing high-performance, easy-to-use data structures and data analysis tools for the [Python programming language](https://www.tutorialspoint.com/python/index.htm). This **Pandas tutorial** has been prepared for those who want to learn about the foundations and advanced features of the Pandas Python package. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc. In this tutorial, we will learn the various features of Python Pandas and how to use them in practice.

**What is Pandas?**

**Pandas** is a powerful Python library that is specifically designed to work on data frames that have "relational" or "labeled" data. Its aim aligns with doing real-world data analysis using Python. Its flexibility and functionality make it indispensable for various data-related tasks. Hence, this Python package works well for data manipulation, operating a dataset, exploring a data frame, data analysis, and machine learning-related tasks. To work on it we should first install it using a pip command like "pip install pandas" and then import it like "import pandas as pd". After successfully installing and importing, we can enjoy the innovative functions of pandas to work on datasets or data frames. Pandas versatility and ease of use make it a go-to tool for working with structured data in Python.

Generally, Pandas operates a data frame using [Series](https://www.tutorialspoint.com/python_pandas/python_pandas_series.htm) and [DataFrame](https://www.tutorialspoint.com/python_pandas/python_pandas_dataframe.htm" \t "_blank); where Series works on a one-dimensional labeled array holding data of any type like [integers](https://www.tutorialspoint.com/python/python_numbers.htm), [strings](https://www.tutorialspoint.com/python/python_strings.htm), and [objects](https://www.tutorialspoint.com/python/python_object_classes.htm), while a DataFrame is a two-dimensional data structure that manages and operates data in tabular form (using rows and columns).

**Why Pandas?**

The beauty of Pandas is that it simplifies the task related to data frames and makes it simple to do many of the time-consuming, repetitive tasks involved in working with data frames, such as:

* **Import datasets** - available in the form of spreadsheets, comma-separated values (CSV) files, and more.
* **Data cleansing** - dealing with missing values and representing them as NaN, NA, or NaT.
* **Size mutability** - columns can be added and removed from DataFrame and higher-dimensional objects.
* **Data normalization** – normalize the data into a suitable format for analysis.
* **Data alignment** - objects can be explicitly aligned to a set of labels.  
  Intuitive merging and joining data sets – we can merge and join datasets.
* **Reshaping and pivoting of datasets** – datasets can be reshaped and pivoted as per the need.
* **Efficient manipulation and extraction** - manipulation and extraction of specific parts of extensive datasets using intelligent label-based slicing, indexing, and subsetting techniques.
* **Statistical analysis** - to perform statistical operations on datasets.
* **Data visualization** - Visualize datasets and uncover insights.

**Installation command:**

**pip install pandas==0.25.3**

**Example**

import pandas as pd

import numpy as np

df = pd.DataFrame(np.random.randn(5, 3), index=['a', 'c', 'e', 'f',

'h'],columns=['one', 'two', 'three'])

df = df.reindex(['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'])

print df['one'].notnull()

**Its output is as follows −**

a True

b False

c True

d False

e True

f True

g False

h True

Name: one, dtype: bool

1. **Sci Py**

SciPy, pronounced as Sigh Pi, is a scientific python open source, distributed under the BSD licensed library to perform Mathematical, Scientific and Engineering Computations.

The SciPy library depends on NumPy, which provides convenient and fast N-dimensional array manipulation. The SciPy library is built to work with NumPy arrays and provides many user-friendly and efficient numerical practices such as routines for numerical integration and optimization. Together, they run on all popular operating systems, are quick to install and are free of charge. NumPy and SciPy are easy to use, but powerful enough to depend on by some of the world's leading scientists and engineers.

**Installation command:**

**pip install scipy==1.7.3**

**Example**

import scipy.io as sio

import numpy as np

#Save a mat file

vect = np.arange(10)

sio.savemat('array.mat', {'vect':vect})

#Now Load the File

mat\_file\_content = sio.loadmat(‘array.mat’)

Print(mat\_file\_content)

**The above program will generate the following output.**

{

'vect': array([[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]]), '\_\_version\_\_': '1.0',

'\_\_header\_\_': 'MATLAB 5.0 MAT-file Platform: posix, Created on: Sat Sep 30

09:49:32 2017', '\_\_globals\_\_': []

}

We can see the array along with the Meta information. If we want to inspect the contents of a MATLAB file without reading the data into memory, use the **command** as shown below.

import scipy.io as sio

mat\_file\_content = sio.whosmat(‘array.mat’)

print mat\_file\_content

**The above program will generate the following output.**

[('vect', (1, 10), 'int64')]

1. **Matplotlib**

What Is Matplotlib?

**Matplotlib** is one of the most popular Python packages used for data visualization. It is a cross-platform library for making 2D plots from data in arrays. It provides an object-oriented API that helps in embedding plots in applications using Python GUI toolkits such as [PyQt](https://www.tutorialspoint.com/pyqt/index.htm" \t "_blank), [WxPython](https://www.tutorialspoint.com/wxpython/index.htm" \t "_blank), or [Tkinter](https://www.tutorialspoint.com/python/python_gui_programming.htm" \t "_blank). It can be used in Python and IPython shells, [Jupyter notebook](https://www.tutorialspoint.com/matplotlib/matplotlib_jupyter_notebook.htm" \t "_blank) and web application servers also.

Matplotlib is a [Python](https://www.tutorialspoint.com/python/index.htm) library that is specifically designed to do effective data visualization. It's a cornerstone of plotting libraries in Python which empowers beginners to dive into the world of attractive data visualization. Matplotlib is an open-source Python library that offers various data visualization (like Line plots, histograms, scatter plots, bar charts, Scatter plots, Pie Charts, and Area Plot etc). A beauty of the Python matplotlib library is its Python code. Its script is structured which denotes that a few lines of code are all that are required in most instances to generate a visual data plot.

**Matplotlib and Pyplot**

**Matplotlib** is a versatile toolkit that allows for the creation of static, animated, and interactive visualizations in the Python programming language.

Generally, matplotlib overlays two APIs:

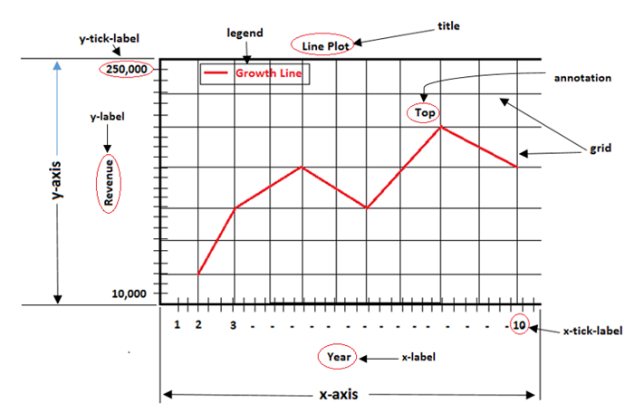
* **The pyplot API**: to make plot using **matplotlib.pyplot**.
* **Object-Oriented API**: A group of objects assembled with greater flexibility than [pyplot](https://www.tutorialspoint.com/matplotlib/matplotlib_pyplot_api.htm" \t "_blank). It provides direct access to Matplotlib’s backend layers.

Matplotlib simplifies simple tasks and enables complex tasks to be accomplished. Following are the key aspects of matplotlib:

* Matplotlib offers to create quality plots.
* Matplotlib offers interactive figures and customizes their visual style that can be manipulated as per need.
* Matplotlib offers export to many file formats.

The most common way to use Matplotlib is through its pyplot module.

**Components of Matplotlib**



**Installation command:**

**pip install matplotlib==3.1.1**

**Example**

import matplotlib.pyplot as plt

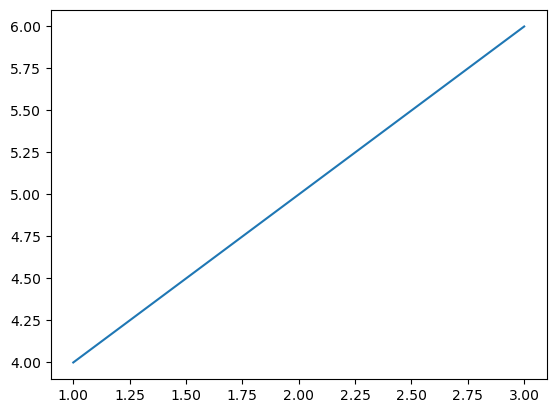
# Create a new figure

fig = plt.figure()

# Add a plot or subplot to the figure

plt.plot([1, 2, 3], [4, 5, 6])

plt.show()



1. **sklearn:**

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

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

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

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

**Installation command:**

**pip install sklearn==0.0**

**Example:**

*# load the iris dataset as an example*

from sklearn.datasets import load\_iris

iris = load\_iris()

*# store the feature matrix (X) and response vector dgfdg(y)*

X = iris.data

y = iris.target

*# store the feature and target names*

feature\_names = iris.feature\_names

target\_names = iris.target\_names

*# printing features and target names of our dataset*

print("Feature names:", feature\_names)

print("Target names:", target\_names)

*# X and y are numpy arrays*

print("\nType of X is:", type(X))

*# printing first 5 input rows*

print("\nFirst 5 rows of X:\n", X[:5])

**Output:**

Feature names: ['sepal length (cm)','sepal width (cm)', 'petal length (cm)','petal width (cm)']

Target names: ['setosa' 'versicolor' 'virginica']

Type of X is:

First 5 rows of X:

[[ 5.1 3.5 1.4 0.2]  
 [ 4.9 3. 1.4 0.2]  
 [ 4.7 3.2 1.3 0.2]  
 [ 4.6 3.1 1.5 0.2]  
 [ 5. 3.6 1.4 0.2]]

1. **Keras:**

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

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

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

* User-Friendly API
* Multi-backend Support
* Wide Range of Neural Network Architectures
* Pre-trained Models and Transfer Learning:
* Easy Model Training and Evaluation:
* GPU Support:
* **Installation command:**
* **pip install keras==2.3.1**

**Example**

from keras.models import Sequential

from keras.layers import Dense, Activation

model = Sequential()

model.add(Dense(units=64, input\_dim=100))

model.add(Activation('relu'))

model.add(Dense(units=10))

model.add(Activation('softmax'))

1. **h5py:**

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

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

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

* + HDF5 File Access
  + Dataset Handling:
  + Group Organization:
  + Attributes:
  + Compatibility with NumPy
  + Performance
* **Installation command:**
* **pip install h5py==2.10.0**

**Example:**

# Python program to demonstrate

# HDF5 file

**import** numpy as np

**import** h5py

# initializing a random numpy array

arr **=** np.random.randn(1000)

# creating a file

with h5py.File('test.hdf5', 'w') as f:

    dset **=** f.create\_dataset("default", data **=** arr)

1. **Tensor flow**

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

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

Here are some key features and functionalities of TensorFlow:

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

**Installation command:**

* **pip install tensorflow==1.14.0**

**Example**

# importing tensorflow

**import** tensorflow as tf

# creating nodes in computation graph

node **=** tf.Variable(tf.zeros([2,2]))

# running computation graph

with tf.Session() as sess:

    # initialize all global variables

    sess.run(tf.global\_variables\_initializer())

    # evaluating node

    print("Tensor value before addition:\n",sess.run(node))

    # elementwise addition to tensor

    node **=** node.assign(node **+** tf.ones([2,2]))

    # evaluate node again

    print("Tensor value after addition:\n", sess.run(node))

1. **Tkinter**

**Tkinter** is a standard [Python GUI (Graphical User Interface) library](https://www.geeksforgeeks.org/python3-gui-application-overview/) that provides a set of tools and widgets to create desktop applications with graphical interfaces. Tkinter is included with most Python installations, making it easily accessible for developers who want to build GUI applications without requiring additional installations or libraries.

**Full Form of Tkinter**

The name “**Tkinter**” comes from “**Tk interface**“, referring to the Tk GUI toolkit that Tkinter is based on. Tkinter provides a way to create windows, buttons, labels, text boxes, and other GUI components to build interactive applications.

**Significance of Tkinter**

**Tkinter** is the inbuilt python module that is used to create GUI applications. It is one of the most commonly used modules for creating GUI applications in Python as it is simple and easy to work with. You don’t need to worry about the installation of the Tkinter module separately as it comes with Python already. It gives an object-oriented interface to the Tk GUI toolkit. Among all, Tkinter is most widely used

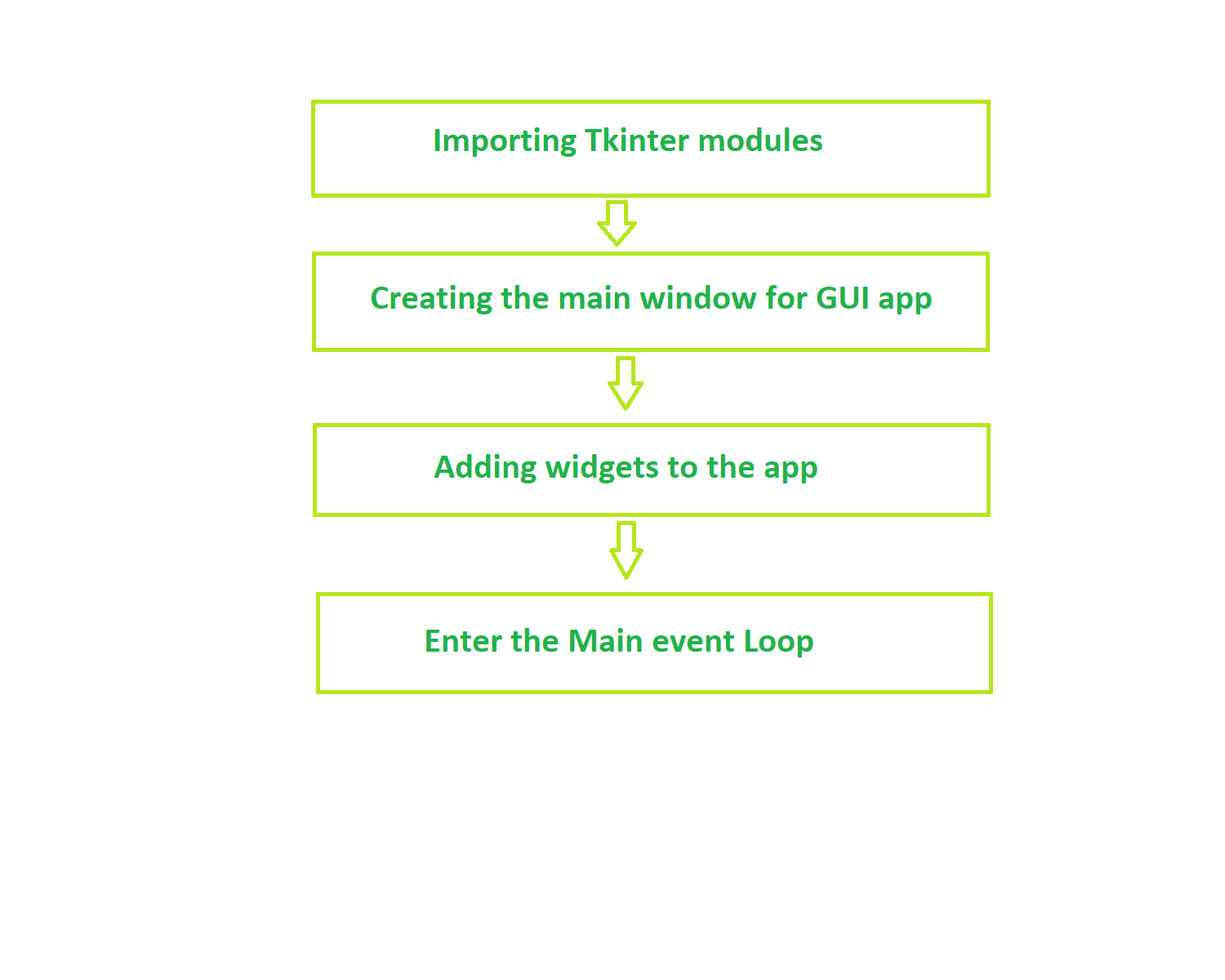
**Where is Python Tkinter used?**

Here are some common use cases for Tkinter:

1. **Creating windows and dialog boxes**: Tkinter can be used to create windows and dialog boxes that allow users to interact with your program. These can be used to display information, gather input, or present options to the user.
2. **Building a GUI for a desktop application**: Tkinter can be used to create the interface for a desktop application, including buttons, menus, and other interactive elements.
3. **Adding a GUI to a command-line program**: Tkinter can be used to add a GUI to a command-line program, making it easier for users to interact with the program and input arguments.
4. **Creating custom widgets**: Tkinter includes a variety of built-in widgets, such as buttons, labels, and text boxes, but it also allows you to create your own custom widgets.
5. **Prototyping a GUI**: Tkinter can be used to quickly prototype a GUI, allowing you to test and iterate on different design ideas before committing to a final implementation.

Tkinter is a useful tool for creating a wide variety of graphical user interfaces, including windows, dialog boxes, and custom widgets. It is particularly well-suited for building desktop applications and adding a GUI to command-line programs.

**Fundamental structure of Tkinter program**



**Example:**

# import the modules

**import** tkinter

**import** random

# list of possible colour.

colours **=** ['Red','Blue','Green','Pink','Black',

           'Yellow','Orange','White','Purple','Brown']

score **=** 0

 # the game time left, initially 30 seconds.

timeleft **=** 30

 # function that will start the game.

**def** startGame(event):

**if** timeleft **==** 30:

        # start the countdown timer.

        countdown()

    # run the function to

    # choose the next colour.

    nextColour()

# Function to choose and

# display the next colour.

**def** nextColour():

    # use the globally declared 'score'

    # and 'play' variables above.

**global** score

**global** timeleft

    # if a game is currently in play

**if** timeleft > 0:

        # make the text entry box active.

        e.focus\_set()

        # if the colour typed is equal

        # to the colour of the text

**if** e.get().lower() **==** colours[1].lower():

            score **+=** 1

         # clear the text entry box.

        e.delete(0, tkinter.END)

        random.shuffle(colours)

        # change the colour to type, by changing the

        # text \_and\_ the colour to a random colour value

        label.config(fg **=** str(colours[1]), text **=** str(colours[0]))

        # update the score.

        scoreLabel.config(text **=** "Score: " **+** str(score))

# Countdown timer function

**def** countdown():

**global** timeleft

    # if a game is in play

**if** timeleft > 0:

         # decrement the timer.

        timeleft **-=** 1

        # update the time left label

        timeLabel.config(text **=** "Time left: "

**+** str(timeleft))

        # run the function again after 1 second.

        timeLabel.after(1000, countdown)

**# Driver Code**

# create a GUI window

root **=** tkinter.Tk()

 # set the title

root.title("COLORGAME")

# set the size

root.geometry("375x200")

# add an instructions label

instructions **=** tkinter.Label(root, text **=** "Type **in** the colour"

                        "of the words, **and** **not** the word text!",

                                      font **=** ('Helvetica', 12))

instructions.pack()

# add a score label

scoreLabel **=** tkinter.Label(root, text **=** "Press enter to start",

                                      font **=** ('Helvetica', 12))

scoreLabel.pack()

# add a time left label

timeLabel **=** tkinter.Label(root, text **=** "Time left: " **+**

              str(timeleft), font **=** ('Helvetica', 12))

timeLabel.pack()

# add a label for displaying the colours

label **=** tkinter.Label(root, font **=** ('Helvetica', 60))

label.pack()

# add a text entry box for

# typing in colours

e **=** tkinter.Entry(root)

# run the 'startGame' function

# when the enter key is pressed

root.bind('<Return>', startGame)

e.pack()

# set focus on the entry box

e.focus\_set()

# start the GUI

root.mainloop()

1. **OpenCV**

OpenCV stands for **Open Source Computer Vision** and is a library of functions which is useful in real time computer vision application programming. The term Computer vision is used for a subject of performing the analysis of digital images and videos using a computer program. Computer vision is an important constituent of modern disciplines such as artificial intelligence and machine learning.

Originally developed by Intel, OpenCV is a cross platform library written in C++ but also has a C Interface Wrappers for OpenCV which have been developed for many other programming languages such as Java and Python. In this tutorial, functionality of OpenCV’s Python library will be described.

**OpenCV-Python** is a Python wrapper around C++ implementation of OpenCV library. It makes use of NumPy library for numerical operations and is a rapid prototyping tool for computer vision problems.

OpenCV-Python is a cross-platform library, available for use on all Operating System (OS) platforms including, Windows, Linux, MacOS and Android. OpenCV also supports the Graphics Processing Unit (GPU) acceleration.

This tutorial is designed for the computer science students and professionals who wish to gain expertise in the field of computer vision applications. Prior knowledge of Python and NumPy library is essential to understand the functionality of OpenCV-Python.

**Installation command:**

**pip install opencv-python==4.1.1.26**

**Example:**

# Python code to read image

import cv2

# To read image from disk, we use

# cv2.imread function, in below method,

img = cv2.imread("geeks.png", cv2.IMREAD\_COLOR)

print(img)

**5.3 Algorithms**

**1.Convolutional Neural Network (CNN)**

A **Convolutional Neural Network (CNN)** is a type of Deep Learning neural network architecture commonly used in Computer Vision. Computer vision is a field of Artificial Intelligence that enables a computer to understand and interpret the image or visual data.

When it comes to Machine Learning, Artificial Neural Networks perform really well. Neural Networks are used in various datasets like images, audio, and text. Different types of Neural Networks are used for different purposes, for example for predicting the sequence of words we use **Recurrent Neural Networks** more precisely an LSTM, similarly for image classification we use Convolution Neural networks. In this blog, we are going to build a basic building block for CNN.

In a regular Neural Network there are three types of layers:

1. **Input Layers:** It’s the layer in which we give input to our model. The number of neurons in this layer is equal to the total number of features in our data (number of pixels in the case of an image).
2. **Hidden Layer:** The input from the Input layer is then fed into the hidden layer. There can be many hidden layers depending on our model and data size. Each hidden layer can have different numbers of neurons which are generally greater than the number of features. The output from each layer is computed by matrix multiplication of the output of the previous layer with learnable weights of that layer and then by the addition of learnable biases followed by activation function which makes the network nonlinear.
3. **Output Layer:** The output from the hidden layer is then fed into a logistic function like sigmoid or softmax which converts the output of each class into the probability score of each class.

The data is fed into the model and output from each layer is obtained from the above step is called [**feedforward**](https://www.geeksforgeeks.org/understanding-multi-layer-feed-forward-networks/), we then calculate the error using an error function, some common error functions are cross-entropy, square loss error, etc. The error function measures how well the network is performing. After that, we backpropagate into the model by calculating the derivatives. This step is called **[Backprzopagation](https://www.geeksforgeeks.org/backpropagation-in-data-mining/)** which basically is used to minimize the loss.

**Convolution Neural Network**

Convolutional Neural Network (CNN) is the extended version of [artificial neural networks (ANN)](https://www.geeksforgeeks.org/artificial-neural-networks-and-its-applications/) which is predominantly used to extract the feature from the grid-like matrix dataset. For example visual datasets like images or videos where data patterns play an extensive role.

**CNN architecture**

Convolutional Neural Network consists of multiple layers like the input layer, Convolutional layer, Pooling layer, and fully connected layers.

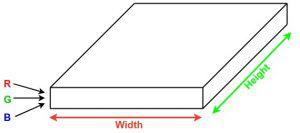


*Simple CNN architecture*

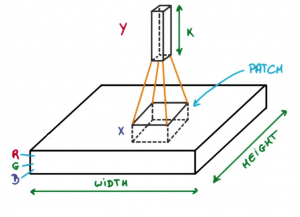
The Convolutional layer applies filters to the input image to extract features, the Pooling layer downsamples the image to reduce computation, and the fully connected layer makes the final prediction. The network learns the optimal filters through backpropagation and gradient descent.

**How Convolutional Layers works**

Convolution Neural Networks or covnets are neural networks that share their parameters. Imagine you have an image. It can be represented as a cuboid having its length, width (dimension of the image), and height (i.e the channel as images generally have red, green, and blue channels).



Now imagine taking a small patch of this image and running a small neural network, called a filter or kernel on it, with say, K outputs and representing them vertically. Now slide that neural network across the whole image, as a result, we will get another image with different widths, heights, and depths. Instead of just R, G, and B channels now we have more channels but lesser width and height. This operation is called **Convolution**. If the patch size is the same as that of the image it will be a regular neural network. Because of this small patch, we have fewer weights.



*Image source: Deep Learning Udacity*

Now let’s talk about a bit of mathematics that is involved in the whole convolution process.

* Convolution layers consist of a set of learnable filters (or kernels) having small widths and heights and the same depth as that of input volume (3 if the input layer is image input).

1. For example, if we have to run convolution on an image with dimensions 34x34x3. The possible size of filters can be axax3, where ‘a’ can be anything like 3, 5, or 7 but smaller as compared to the image dimension.
2. During the forward pass, we slide each filter across the whole input volume step by step where each step is called **stride** (which can have a value of 2, 3, or even 4 for high-dimensional images) and compute the dot product between the kernel weights and patch from input volume.
3. As we slide our filters we’ll get a 2-D output for each filter and we’ll stack them together as a result, we’ll get output volume having a depth equal to the number of filters. The network will learn all the filters.

**Layers used to build ConvNets**

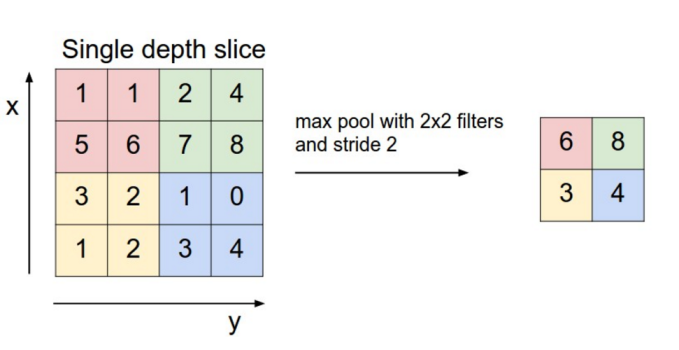
A complete Convolution Neural Networks architecture is also known as covnets. A covnets is a sequence of layers, and every layer transforms one volume to another through a differentiable function.

**Types of layers:**

datasets  
Let’s take an example by running a covnets on of image of dimension 32 x 32 x 3.

* **Input Layers:** It’s the layer in which we give input to our model. In CNN, Generally, the input will be an image or a sequence of images. This layer holds the raw input of the image with width 32, height 32, and depth 3.

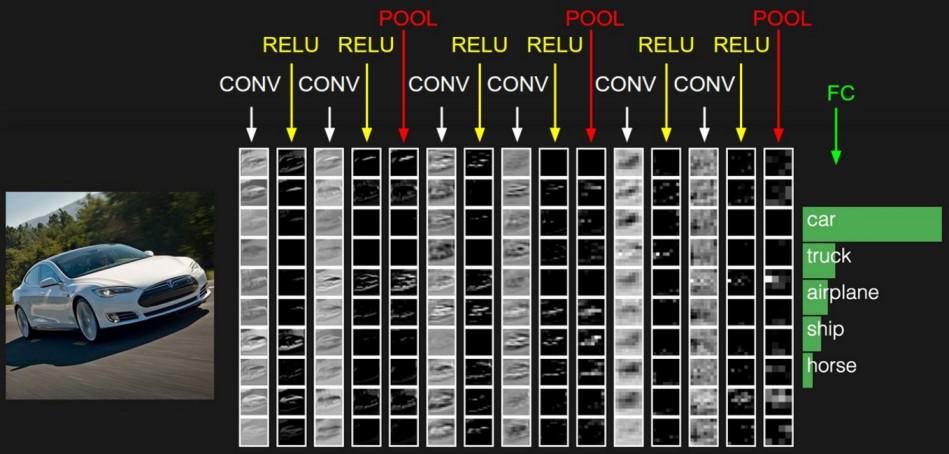
1. **Convolutional Layers:**This is the layer, which is used to extract the feature from the input dataset. It applies a set of learnable filters known as the kernels to the input images. The filters/kernels are smaller matrices usually 2×2, 3×3, or 5×5 shape. it slides over the input image data and computes the dot product between kernel weight and the corresponding input image patch. The output of this layer is referred as feature maps. Suppose we use a total of 12 filters for this layer we’ll get an output volume of dimension 32 x 32 x 12.
2. **Activation Layer:**By adding an activation function to the output of the preceding layer, activation layers add nonlinearity to the network. it will apply an element-wise activation function to the output of the convolution layer. Some common activation functions are **RELU**: max(0, x),  **Tanh**, **Leaky RELU**, etc. The volume remains unchanged hence output volume will have dimensions 32 x 32 x 12.
3. **Pooling layer:** This layer is periodically inserted in the covnets and its main function is to reduce the size of volume which makes the computation fast reduces memory and also prevents overfitting. Two common types of pooling layers are **max pooling** and **average pooling**. If we use a max pool with 2 x 2 filters and stride 2, the resultant volume will be of dimension 16x16x12.



*Image source: cs231n.stanford.edu*

* **Flattening:**The resulting feature maps are flattened into a one-dimensional vector after the convolution and pooling layers so they can be passed into a completely linked layer for categorization or regression.

1. **Fully Connected Layers:**It takes the input from the previous layer and computes the final classification or regression task.



* **Output Layer:** The output from the fully connected layers is then fed into a logistic function for classification tasks like sigmoid or softmax which converts the output of each class into the probability score of each class.

**Example:**

Let’s consider an image and apply the convolution layer, activation layer, and pooling layer operation to extract the inside feature.

**Input image:**



*Input image*

**Step:**

* import the necessary libraries

1. set the parameter
2. define the kernel
3. Load the image and plot it.
4. Reformat the image
5. Apply convolution layer operation and plot the output image.
6. Apply activation layer operation and plot the output image.
7. Apply pooling layer operation and plot the output image.

**Python3**

|  |
| --- |
| # import the necessary libraries  import numpy as np  import tensorflow as tf  import matplotlib.pyplot as plt  from itertools import product    # set the param  plt.rc('figure', autolayout=True)  plt.rc('image', cmap='magma')    # define the kernel  kernel = tf.constant([[-1, -1, -1],                      [-1,  8, -1],                      [-1, -1, -1],                     ])    # load the image  image = tf.io.read\_file('Ganesh.jpg')  image = tf.io.decode\_jpeg(image, channels=1)  image = tf.image.resize(image, size=[300, 300])    # plot the image  img = tf.squeeze(image).numpy()  plt.figure(figsize=(5, 5))  plt.imshow(img, cmap='gray')  plt.axis('off')  plt.title('Original Gray Scale image')  plt.show();      # Reformat  image = tf.image.convert\_image\_dtype(image, dtype=tf.float32)  image = tf.expand\_dims(image, axis=0)  kernel = tf.reshape(kernel, [\*kernel.shape, 1, 1])  kernel = tf.cast(kernel, dtype=tf.float32)    # convolution layer  conv\_fn = tf.nn.conv2d    image\_filter = conv\_fn(      input=image,      filters=kernel,      strides=1, # or (1, 1)      padding='SAME',  )    plt.figure(figsize=(15, 5))    # Plot the convolved image  plt.subplot(1, 3, 1)    plt.imshow(      tf.squeeze(image\_filter)  )  plt.axis('off')  plt.title('Convolution')    # activation layer  relu\_fn = tf.nn.relu  # Image detection  image\_detect = relu\_fn(image\_filter)    plt.subplot(1, 3, 2)  plt.imshow(      # Reformat for plotting      tf.squeeze(image\_detect)  )    plt.axis('off')  plt.title('Activation')    # Pooling layer  pool = tf.nn.pool  image\_condense = pool(input=image\_detect,                               window\_shape=(2, 2),                               pooling\_type='MAX',                               strides=(2, 2),                               padding='SAME',                              )    plt.subplot(1, 3, 3)  plt.imshow(tf.squeeze(image\_condense))  plt.axis('off')  plt.title('Pooling')  plt.show() |

**Output**:



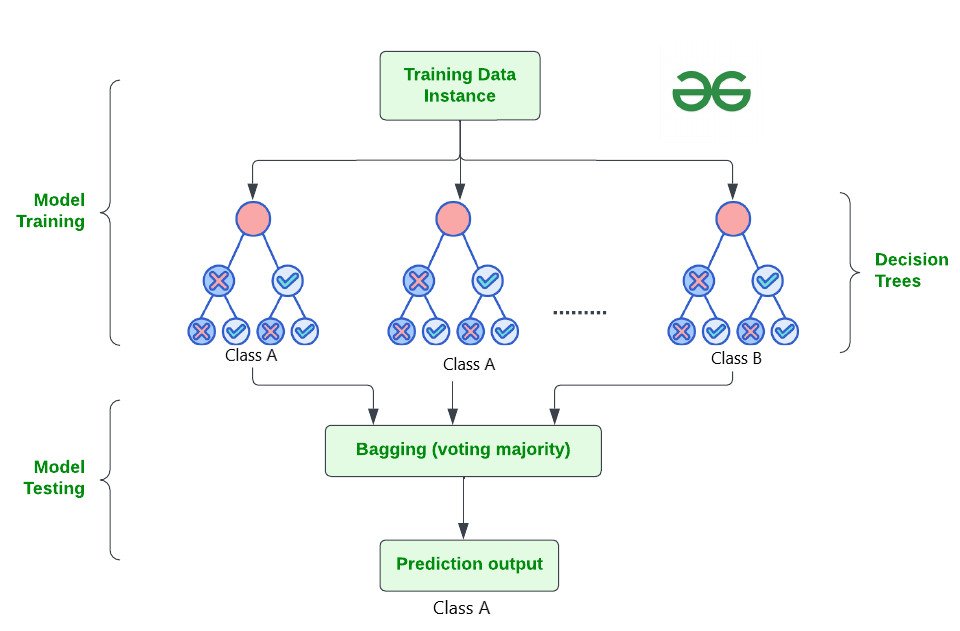
*Original Grayscale image*

# Screenshot-from-2023-03-20-15-07-10-(2)

**2.Random Forest:**

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

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



**How Does Random Forest Work?**

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

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

**Key Features of Random Forest**

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

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

## Implement Random Forest for Classification :

# Import necessary libraries

import pandas as pd

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

from sklearn.ensemble import RandomForestClassifier

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

import warnings

warnings.filterwarnings('ignore')

# Load the Titanic dataset

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

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

# Drop rows with missing target values

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

# Select relevant features and target variable

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

y = titanic\_data['Survived']

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

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

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

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

# Split the dataset into training and testing sets

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

# Create a Random Forest Classifier

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

# Train the classifier

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

# Make predictions on the test set

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

# Evaluate the model

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

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

# Print the results

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

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

**Output:**

Accuracy: 0.80

Classification Report:

precision recall f1-score support

0 0.82 0.85 0.83 105

1 0.77 0.73 0.75 74

accuracy 0.80 179

macro avg 0.79 0.79 0.79 179

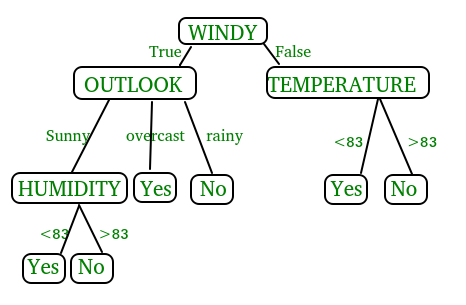
weighted avg 0.80 0.80 0.80 179

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

**3.Decision Tree:**

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

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



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

**Key Components of Decision Trees in Python**

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

**Assumptions we make while using Decision tree**

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

Implementing Decision Tree Using Python:

from sklearn.datasets import load\_iris

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

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy\_score

# Load the Iris dataset

iris = load\_iris()

X = iris.data

y = iris.target

# Split the dataset into training and testing sets

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

# Initialize the decision tree classifier

clf = DecisionTreeClassifier()

# Train the classifier on the training data

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

# Make predictions on the testing data

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

# Calculate the accuracy of the classifier

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

print("Accuracy:", accuracy)

In This Example :

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

**5.4 Sample Code:**

import os

import cv2

import numpy as np

from keras.utils.np\_utils import to\_categorical

from keras.layers import MaxPooling2D

from keras.layers import Dense, Dropout, Activation, Flatten

from keras.layers import Convolution2D

from keras.models import Sequential, load\_model, Model

import pickle

from sklearn.metrics import precision\_score

from sklearn.metrics import recall\_score

from sklearn.metrics import f1\_score

from sklearn.metrics import accuracy\_score

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

from keras.callbacks import ModelCheckpoint

import keras

from sklearn.metrics import confusion\_matrix

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.ensemble import RandomForestClassifier

from sklearn.tree import DecisionTreeClassifier

#define and load class labels found in dataset

path = "Dataset"

labels = []

X = []

Y = []

for root, dirs, directory in os.walk(path):

for j in range(len(directory)):

name = os.path.basename(root)

if name not in labels:

labels.append(name.strip())

print("Chest Disease Class Labels : "+str(labels))

#define function to get class label of given image

def getLabel(name):

index = -1

for i in range(len(labels)):

if labels[i] == name:

index = i

break

return index

#load dataset image and process them

if os.path.exists("model/X.txt.npy"):

X = np.load('model/X.txt.npy')

Y = np.load('model/Y.txt.npy')

else: #if images not process then read and process image pixels

for root, dirs, directory in os.walk(path):#connect to dataset folder

for j in range(len(directory)):#loop all images from dataset folder

name = os.path.basename(root)

if 'Thumbs.db' not in directory[j]:

img = cv2.imread(root+"/"+directory[j])#read images

img = cv2.resize(img, (32, 32))#resize image

X.append(img) #add image pixels to X array

label = getLabel(name)#get image label id

Y.append(label)#add image label

X = np.asarray(X)#convert array as numpy array

Y = np.asarray(Y)

np.save('model/X.txt',X)#save process images and labels

np.save('model/Y.txt',Y)

print("Dataset images loaded")

print("Total images found in dataset : "+str(X.shape[0]))

print()

#visualizing class labels count found in dataset

names, count = np.unique(Y, return\_counts = True)

height = count

bars = labels

y\_pos = np.arange(len(bars))

plt.figure(figsize = (6, 3))

plt.bar(y\_pos, height)

plt.xticks(y\_pos, bars)

plt.xlabel("Dataset Class Label Graph")

plt.ylabel("Count")

plt.show()

#preprocess images like shuffling and normalization

X = X.astype('float32')

X = X/255 #normalized pixel values between 0 and 1

indices = np.arange(X.shape[0])

np.random.shuffle(indices)#shuffle all images

X = X[indices]

Y = Y[indices]

Y = to\_categorical(Y)

#split dataset into train and test

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, Y, test\_size=0.2)

print("Dataset Image Processing & Normalization Completed")

print("80% images used to train algorithms : "+str(X\_train.shape[0]))

print("20% image used to train algorithms : "+str(X\_test.shape[0]))

#define global variables to save accuracy and other metrics

accuracy = []

precision = []

recall = []

fscore = []

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

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

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

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

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

accuracy.append(a)

precision.append(p)

recall.append(r)

fscore.append(f)

print(algorithm+" Accuracy : "+str(a))

print(algorithm+" Precision : "+str(p))

print(algorithm+" Recall : "+str(r))

print(algorithm+" FScore : "+str(f))

conf\_matrix = confusion\_matrix(y\_test, predict)

plt.figure(figsize =(6, 5))

ax = sns.heatmap(conf\_matrix, xticklabels = labels, yticklabels = labels, annot = True, cmap="viridis" ,fmt ="g");

ax.set\_ylim([0,len(labels)])

plt.title(algorithm+" Confusion matrix")

plt.xticks(rotation=90)

plt.ylabel('True class')

plt.xlabel('Predicted class')

plt.show()

#train CNN algorithm

cnn\_model = Sequential()

#defining cnn layer with 3 X 3 matrix to filter dataset using 32 neurons

cnn\_model.add(Convolution2D(64, (3 , 3), input\_shape = (X\_train.shape[1], X\_train.shape[2], X\_train.shape[3]), activation = 'relu'))

cnn\_model.add(MaxPooling2D(pool\_size = (2, 2)))

cnn\_model.add(Convolution2D(32, (3, 3), activation = 'relu'))

cnn\_model.add(MaxPooling2D(pool\_size = (2, 2)))

cnn\_model.add(Flatten())

cnn\_model.add(Dense(units = 256, activation = 'relu'))

cnn\_model.add(Dense(units = y\_train.shape[1], activation = 'softmax'))

#compiling, training and loading model

cnn\_model.compile(optimizer = 'adam', loss = 'categorical\_crossentropy', metrics = ['accuracy'])

if os.path.exists("model/cnn\_weights.hdf5") == False:

model\_check\_point = ModelCheckpoint(filepath='model/cnn\_weights.hdf5', verbose = 1, save\_best\_only = True)

hist = cnn\_model.fit(X\_train, y\_train, batch\_size = 64, epochs = 60, validation\_data=(X\_test, y\_test), callbacks=[model\_check\_point], verbose=1)

f = open('model/cnn\_history.pckl', 'wb')

pickle.dump(hist.history, f)

f.close()

else:

cnn\_model.load\_weights("model/cnn\_weights.hdf5")

#perform prediction on test data

predict = cnn\_model.predict(X\_test)

predict = np.argmax(predict, axis=1)

y\_test1 = np.argmax(y\_test, axis=1)

calculateMetrics("CNN Algorithm", predict, y\_test1)#calculate accuracy and other metrics

#training Random Forest machine learning algorithm

y\_test = np.argmax(y\_test, axis=1)

y\_train = np.argmax(y\_train, axis=1)

X\_train = np.reshape(X\_train, (X\_train.shape[0], (X\_train.shape[1] \* X\_train.shape[2] \* X\_train.shape[3])))

X\_test = np.reshape(X\_test, (X\_test.shape[0], (X\_test.shape[1] \* X\_test.shape[2] \* X\_test.shape[3])))

rf = RandomForestClassifier()

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

predict = rf.predict(X\_test)

calculateMetrics("Random Forest Algorithm", predict, y\_test1)#calculate accuracy and other metrics

#training decision tree algorithm

dt = DecisionTreeClassifier()

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

predict = dt.predict(X\_test)

calculateMetrics("Decision Tree Algorithm", predict, y\_test1)#calculate accuracy and other metrics

#plot all algorithm performance in tabukar format

import pandas as pd

df = pd.DataFrame([['CNN Algorithm','Accuracy',accuracy[0]],['CNN Algorithm','Precision',precision[0]],['CNN Algorithm','Recall',recall[0]],['CNN Algorithm','FSCORE',fscore[0]],

['Random Forest','Accuracy',accuracy[1]],['Random Forest','Precision',precision[1]],['Random Forest','Recall',recall[1]],['Random Forest','FSCORE',fscore[1]],

['Decision Tree','Accuracy',accuracy[2]],['Decision Tree','Precision',precision[2]],['Decision Tree','Recall',recall[2]],['Decision Tree','FSCORE',fscore[2]],

],columns=['Parameters','Algorithms','Value'])

df.pivot("Parameters", "Algorithms", "Value").plot(kind='bar', figsize=(6, 3))

plt.title("All Algorithms Performance Graph")

plt.show()

#display all algorithm performnace

algorithms = ['CNN Algorithm', 'Random Forest', 'Decision Tree']

data = []

for i in range(len(accuracy)):

data.append([algorithms[i], accuracy[i], precision[i], recall[i], fscore[i]])

data = pd.DataFrame(data, columns=['Algorithm Name', 'Accuracy', 'Precision', 'Recall', 'FSCORE'])

data

#use this function to predict fish species uisng extension model

def predict(image\_path):

image = cv2.imread(image\_path)#read test image

img = cv2.resize(image, (28, 28))#resize image

im2arr = np.array(img)

im2arr = im2arr.reshape(1,28,28,3)#convert image as 4 dimension

img = np.asarray(im2arr)

img = img.astype('float32')#convert image features as float

img = img/255 #normalized image

predict = cnn\_model.predict(img)#now predict dog breed

predict = np.argmax(predict)

img = cv2.imread(image\_path)

img = cv2.resize(img, (500,300))#display image with predicted output

img = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)

cv2.putText(img, 'Chest Disease : '+labels[predict], (10, 25), cv2.FONT\_HERSHEY\_SIMPLEX,0.7, (255, 0, 0), 2)

plt.imshow(img)

#call this function with test image to predict dog breed

predict("testImages/cavitating Pneumonia.jpeg")

#call this function with test image to predict dog breed

predict("testImages/WhatsApp Image 2024-06-28 at 3.46.38 PM (1).jpeg")

#call this function with test image to predict dog breed

predict("testImages/WhatsApp Image 2024-06-28 at 3.47.09 PM.jpeg")

#call this function with test image to predict dog breed

predict("testImages/3.png")

**6. TESTING**

**Implementation and Testing:**

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

**A. Implementation**

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

## **Testing**

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

### **System Testing**

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

1. **Module Testing**

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

1. **Integration Testing**

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

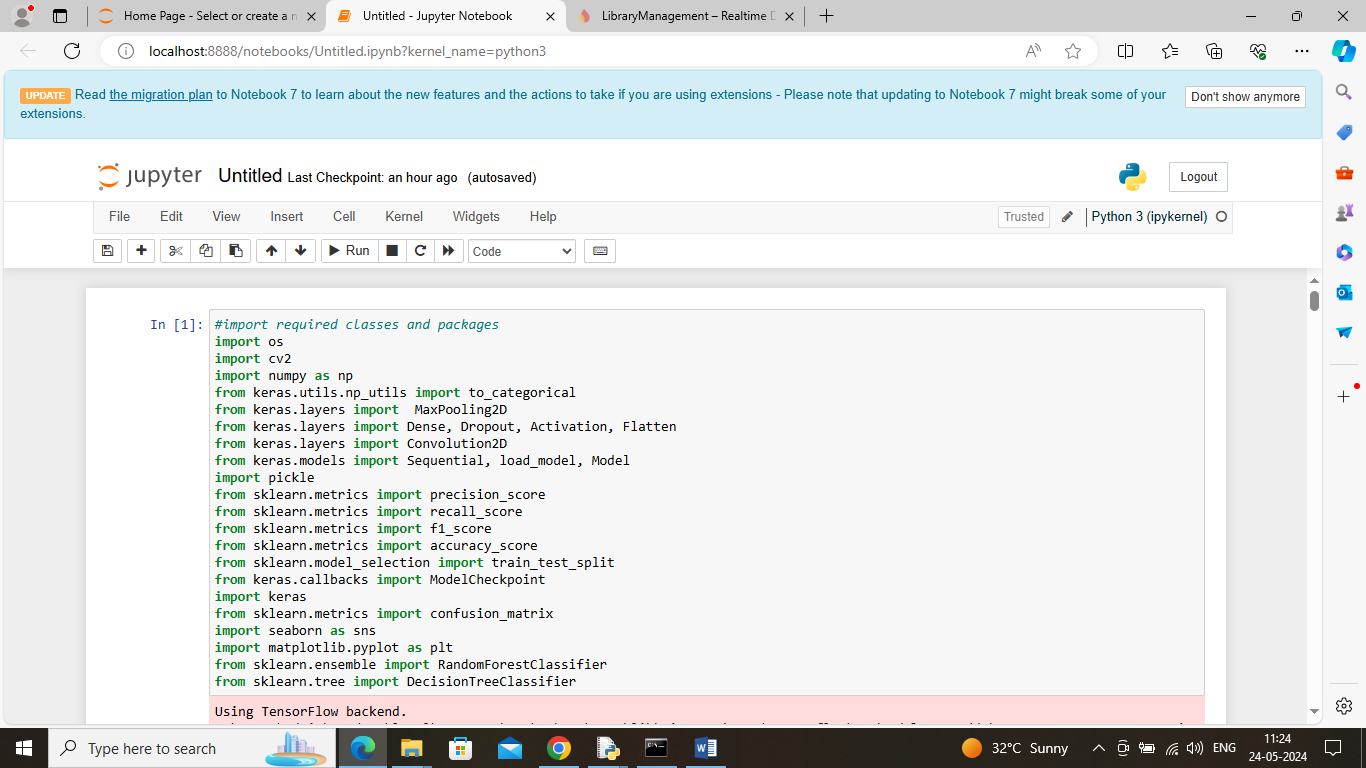
1. **Acceptance Testing**

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

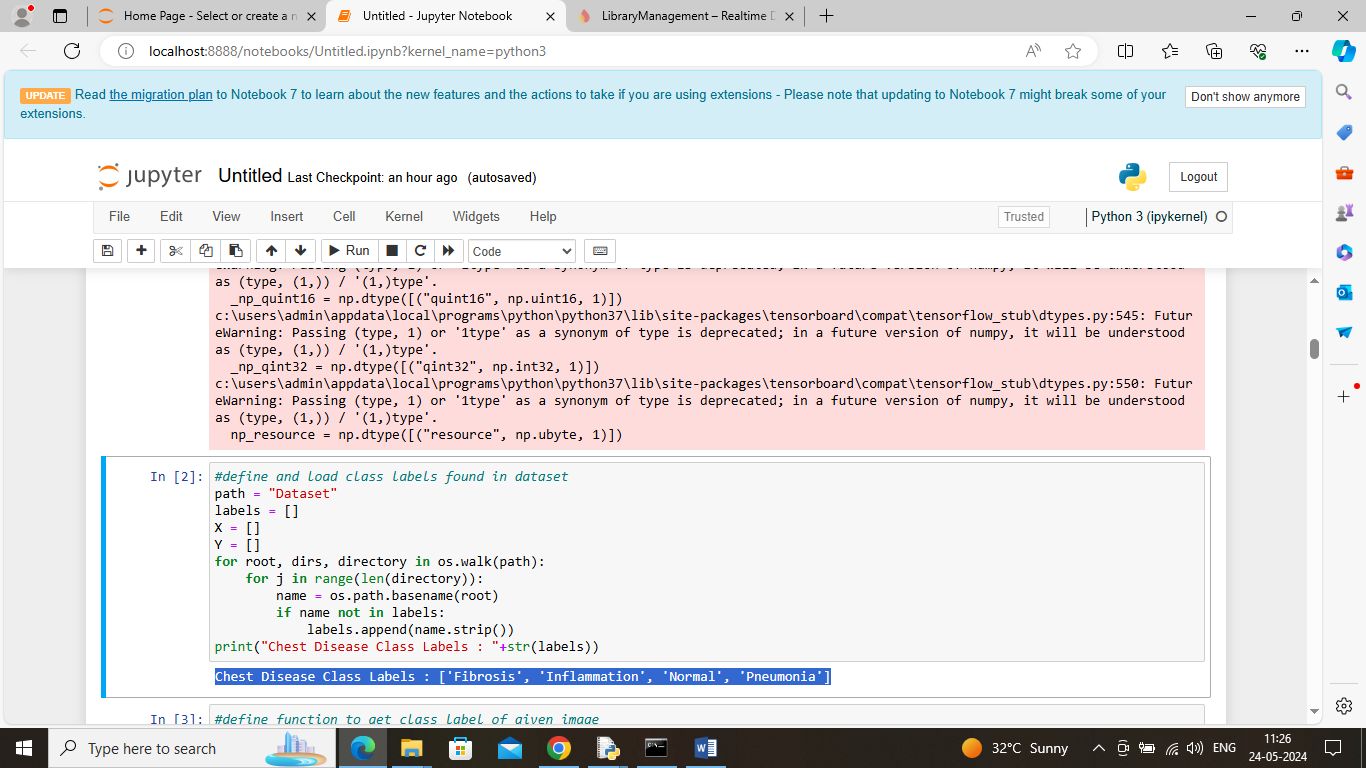
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case Id** | **Test Case Name** | **Test Case Desc.** | **Test Steps** | | | **Test Case Status** | **Test Priority** |
| **Step** | **Expected** | **Actual** |
| 01 | load Dataset | Test whether the Dataset Loaded or not into the system | Dataset may not loaded | we cannot do further operations | Dataset loaded we will do further operations | High | High |
| 02 | Preprocessing | Verify the Preprocessing  or not | Without Text Preprocessing | Without Preprocessing We cannot do further operations | We Can Preprocessing successfully  We will do further operations | High | High |
| 03 | train CNN | Check CNN train or not | Without train CNN | Without train CNN  We cannot do further operations | we can train CNN  We will do further operations | High | High |
| 04 | train Random forest | Check Random forest train run or not | Without train Random Forest | Without train Random forest  We cannot do further operations | we can train Random forest  We will do further operations | High | High |
| 05 | train Decision tree | Check Decision tree train  or not | Without train Decision tree | Without train Decision tree  We cannot do further operations | we can train Decision tree  We will do further operations | High | High |
| 06 | Performance Graph | Check Performance Graph or not | Without Performance Graph | Without Performance Graph  We cannot do further operations | we can Performance Graph  We will do further operations | High | High |
| 07 | Predict | Predict or not | Without Predict | Without Predict  We cannot do further operations | we can Predict  We will do further operations | High | High |

**7. SCREENSHOTS**

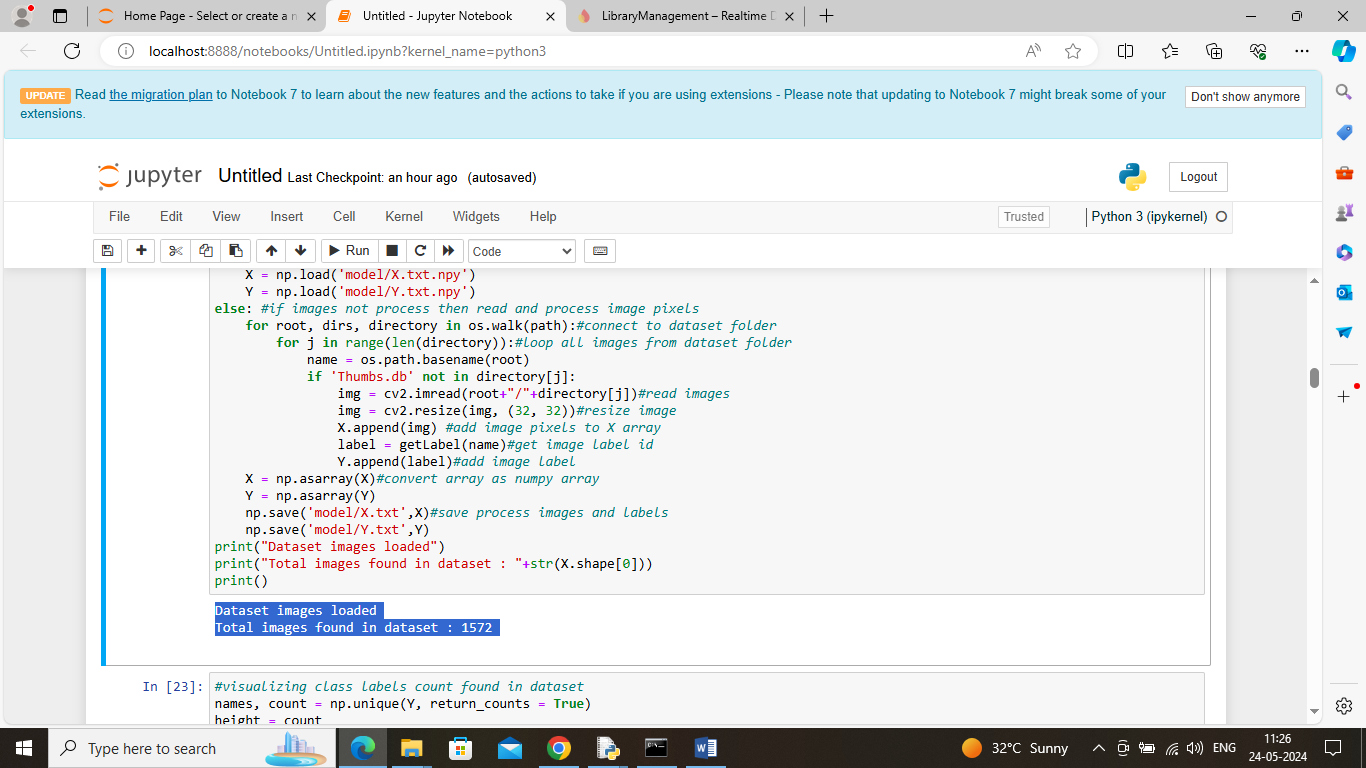
We have coded this project using JUPYTER notebook and below are the code and output screens with blue color comments



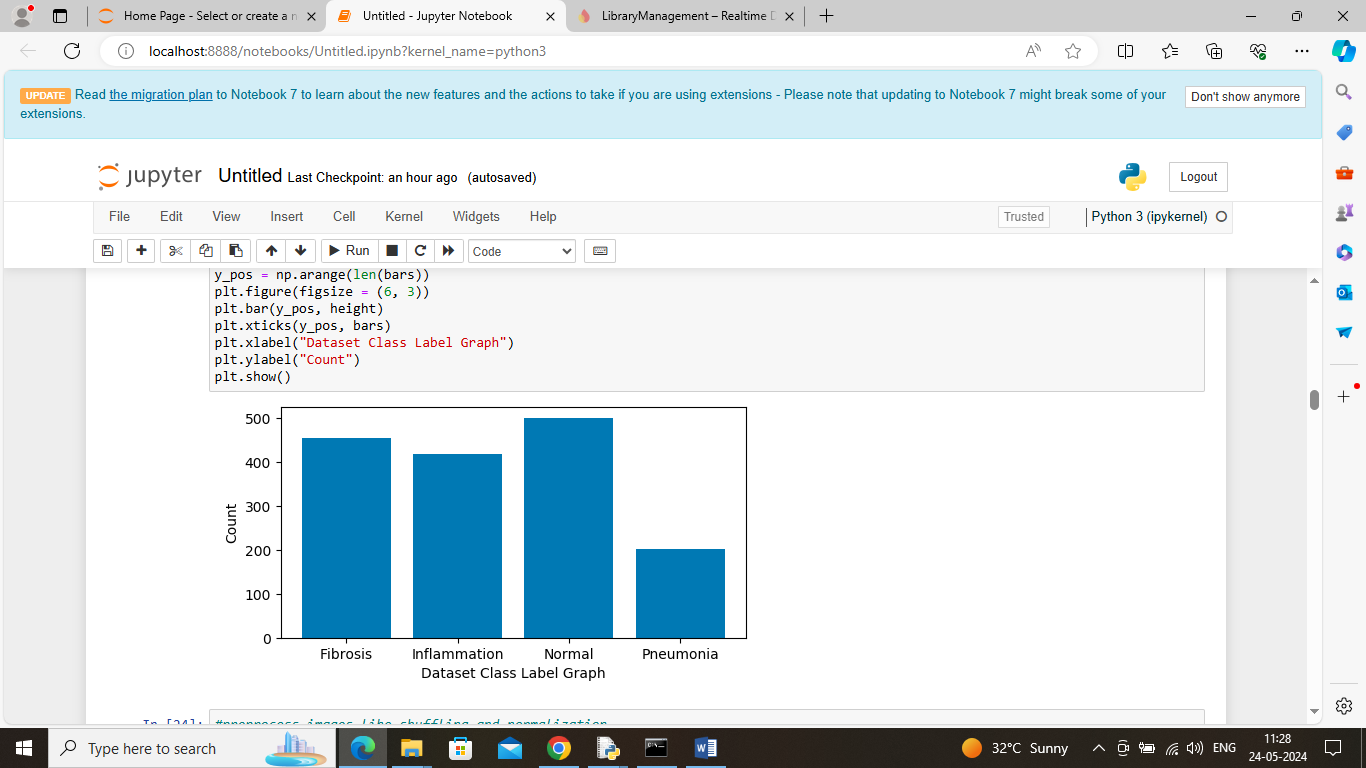
In above screen importing required classes and packages



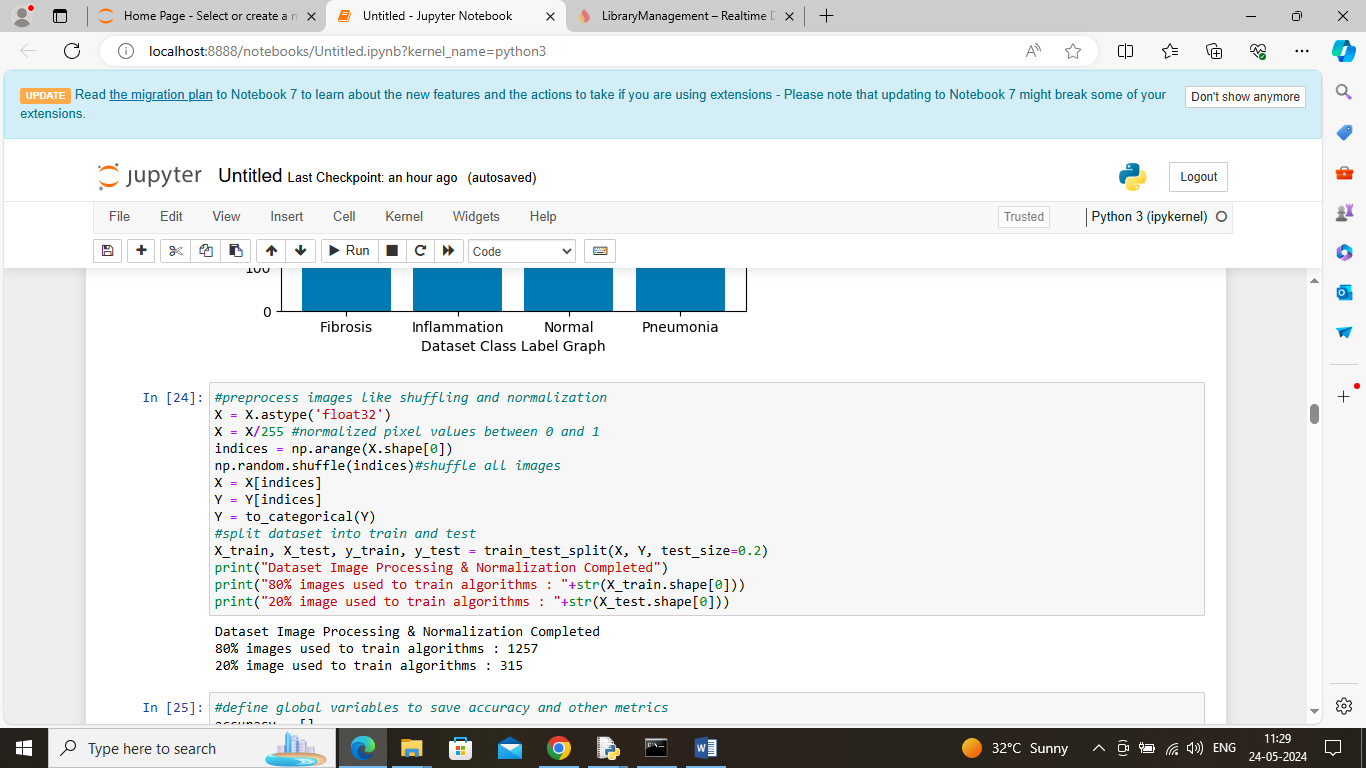
In above screen defining function to identify class labels available in dataset



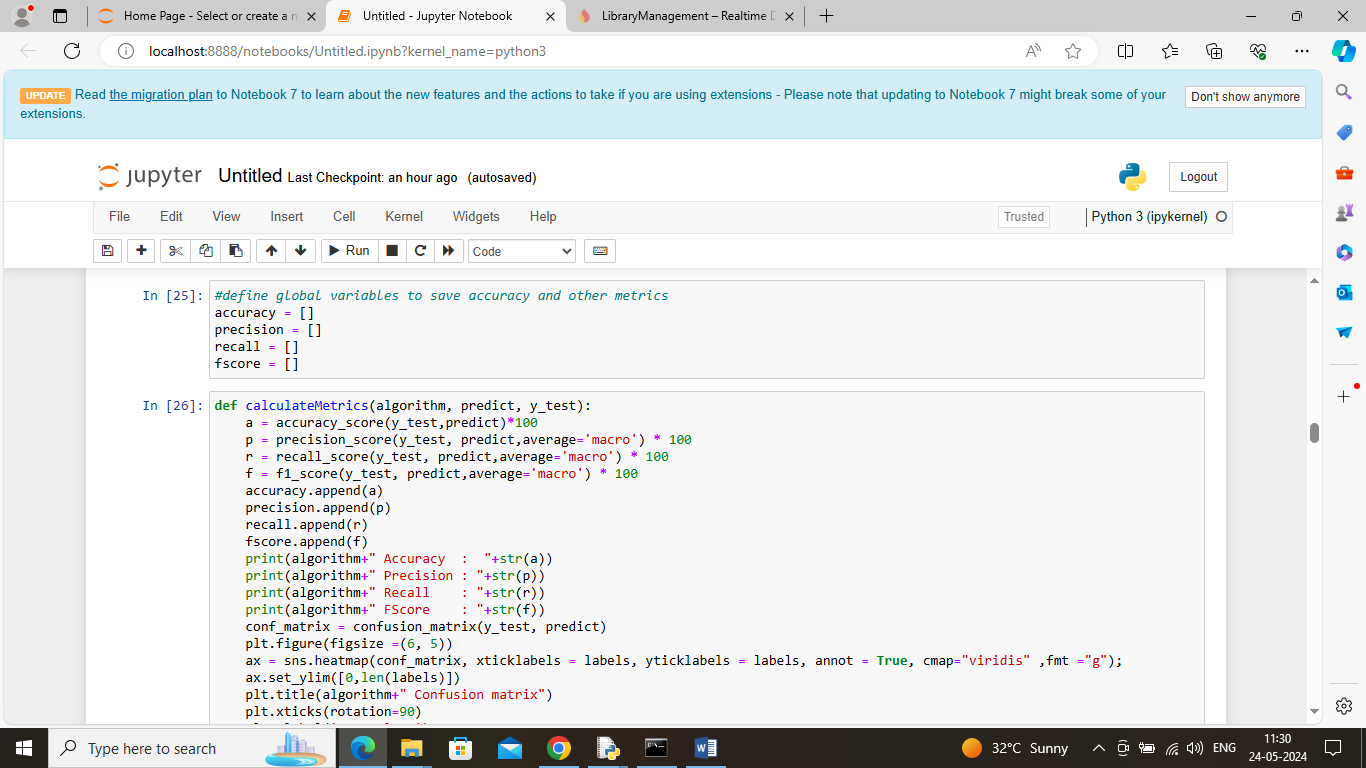
In above screen connecting to dataset folder and then looping and reading all images and then resizing to equal size and then creating X and Y training array where X will contains image features and Y will contains labels and then in blue color text can see total number of images loaded



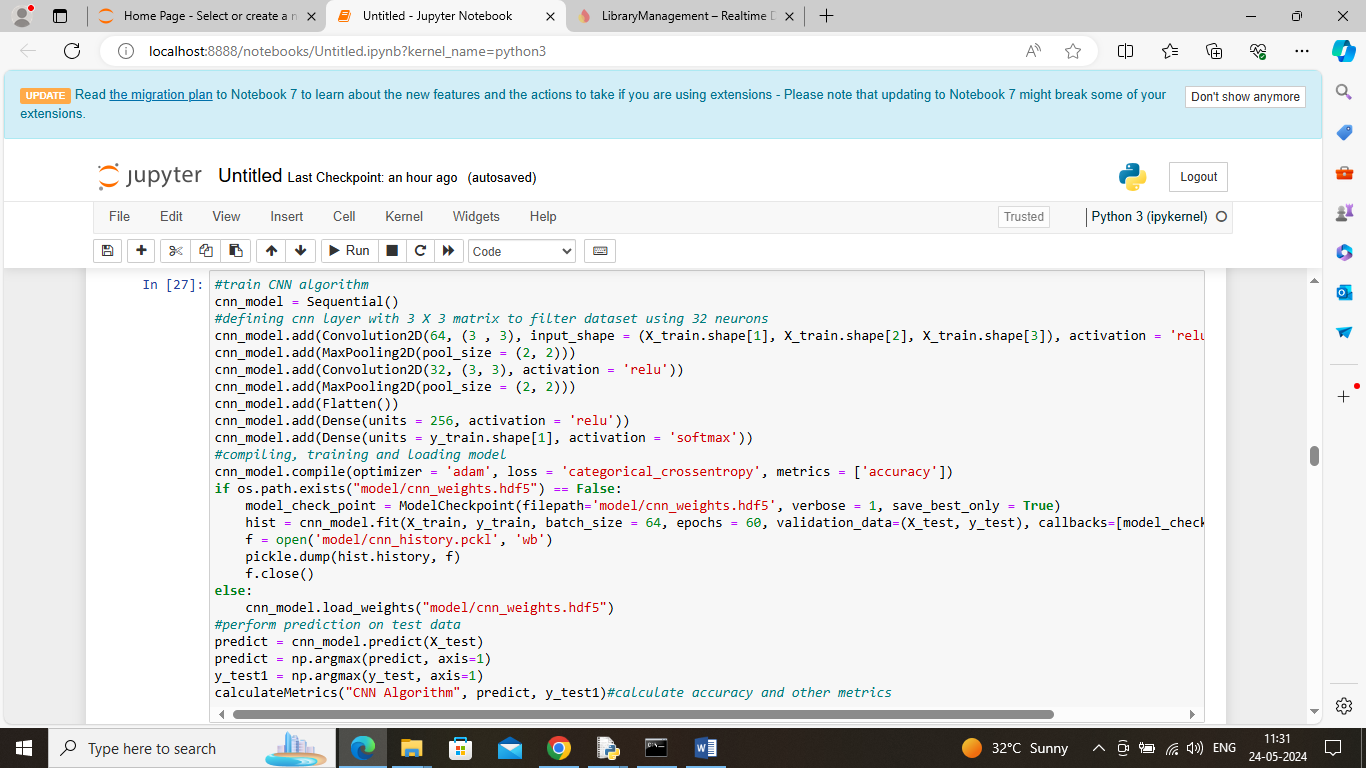
In above screen displaying graph with different class labels and there image counts and in above graph x-axis represents Class label and y-axis represents image count found in that class label



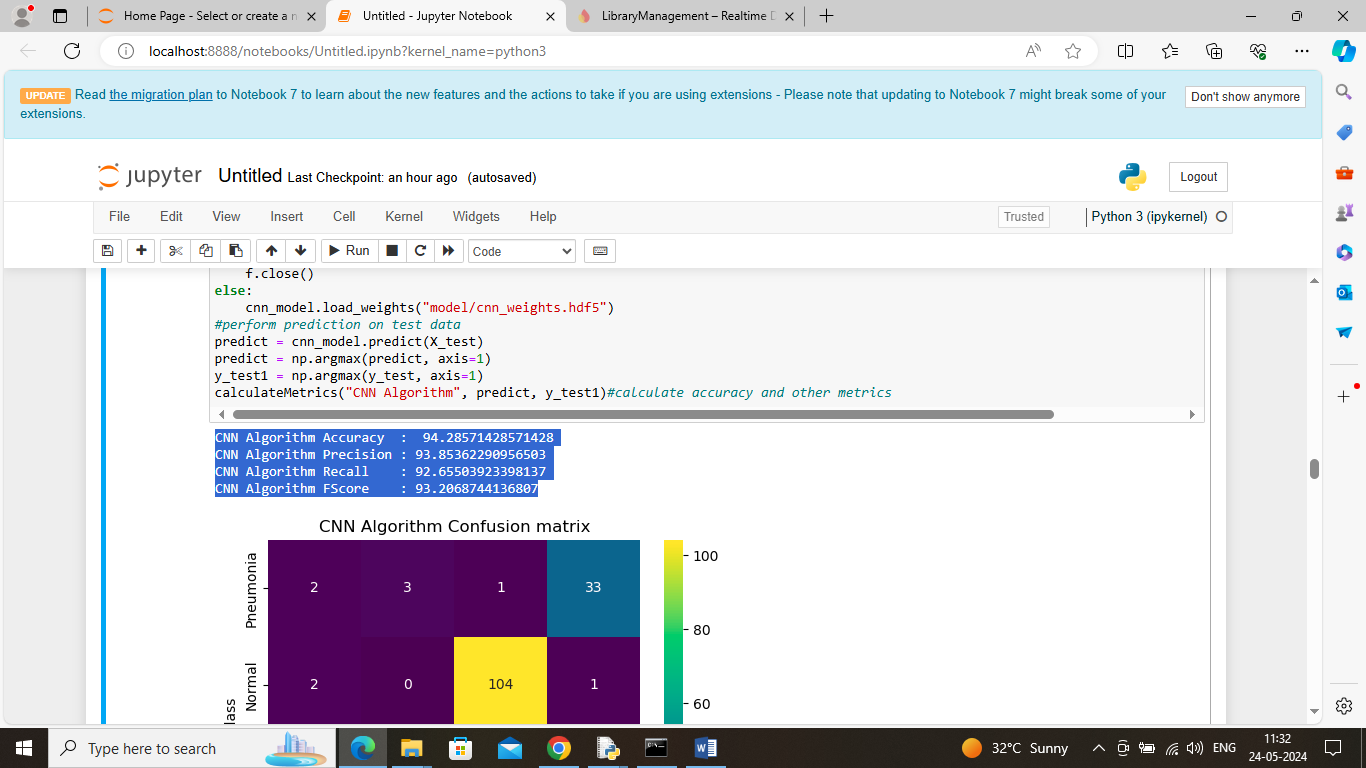
In above screen applying image processing techniques like shuffling, normalizing and then splitting all images into train and test where application using 80% images for training and 20% for testing and then can see train and test size



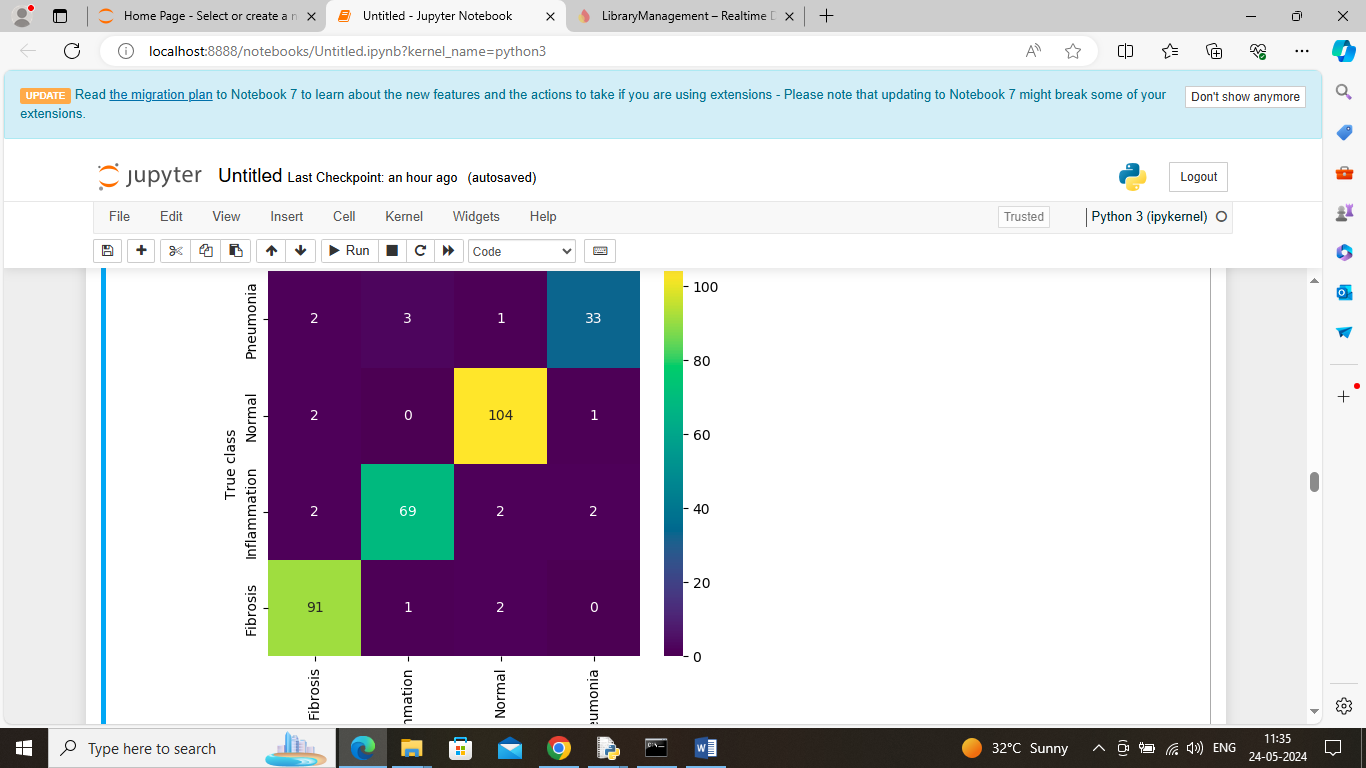
In above screen defining function to calculate accuracy and other metrics



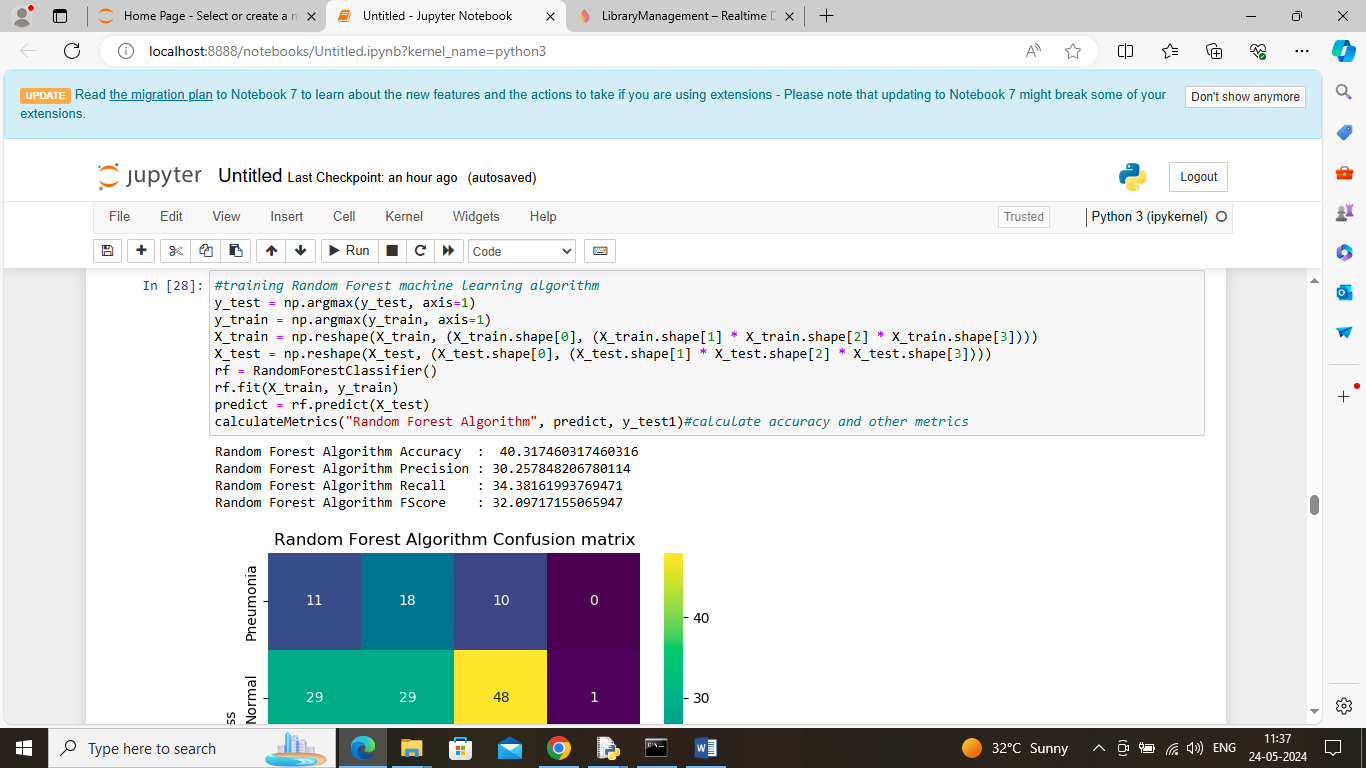
In above screen defining CNN algorithm layers for features filtration and training and then generate a train model and this model will be applied on test data to get below output



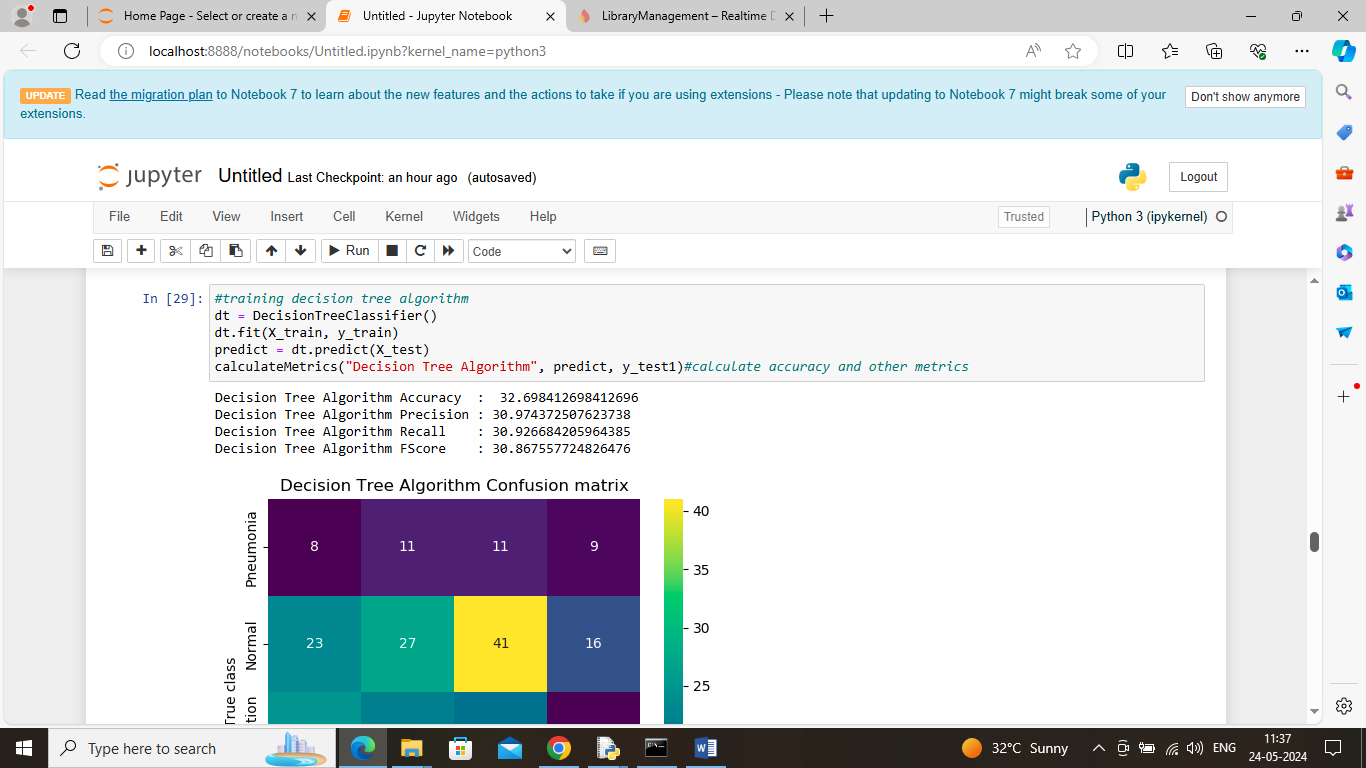
In above screen CNN got 94% accuracy and can see other metrics like precision, recall and FSCORE. Below is the confusion matrix graph



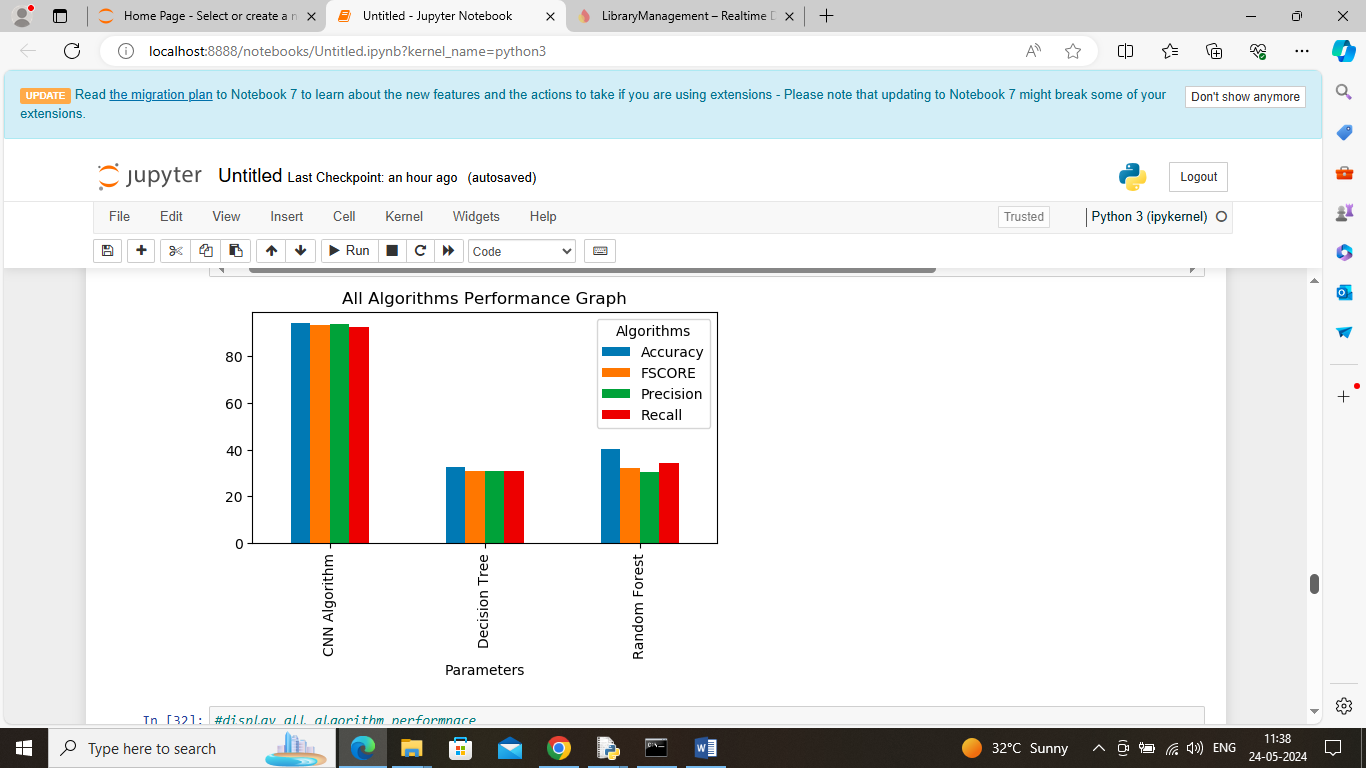
In above confusion matrix graph x-axis represents ‘Predicted Labels’ and y-axis represents true labels and then all different color boxes in diagnol represents correct prediction count and remaining blue boxes represents incorrect prediction count which are very few

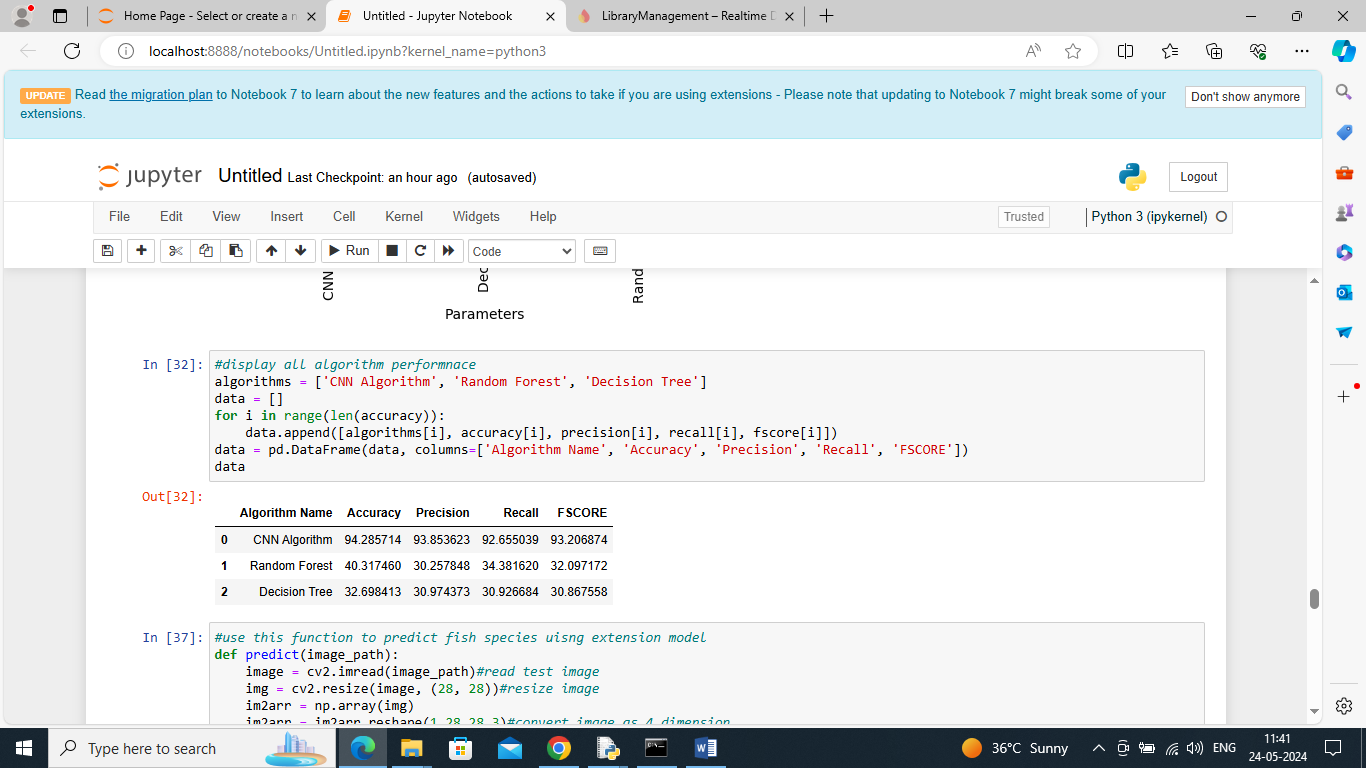


In above screen Random Forest got 40% accuracy and can se other metrics also

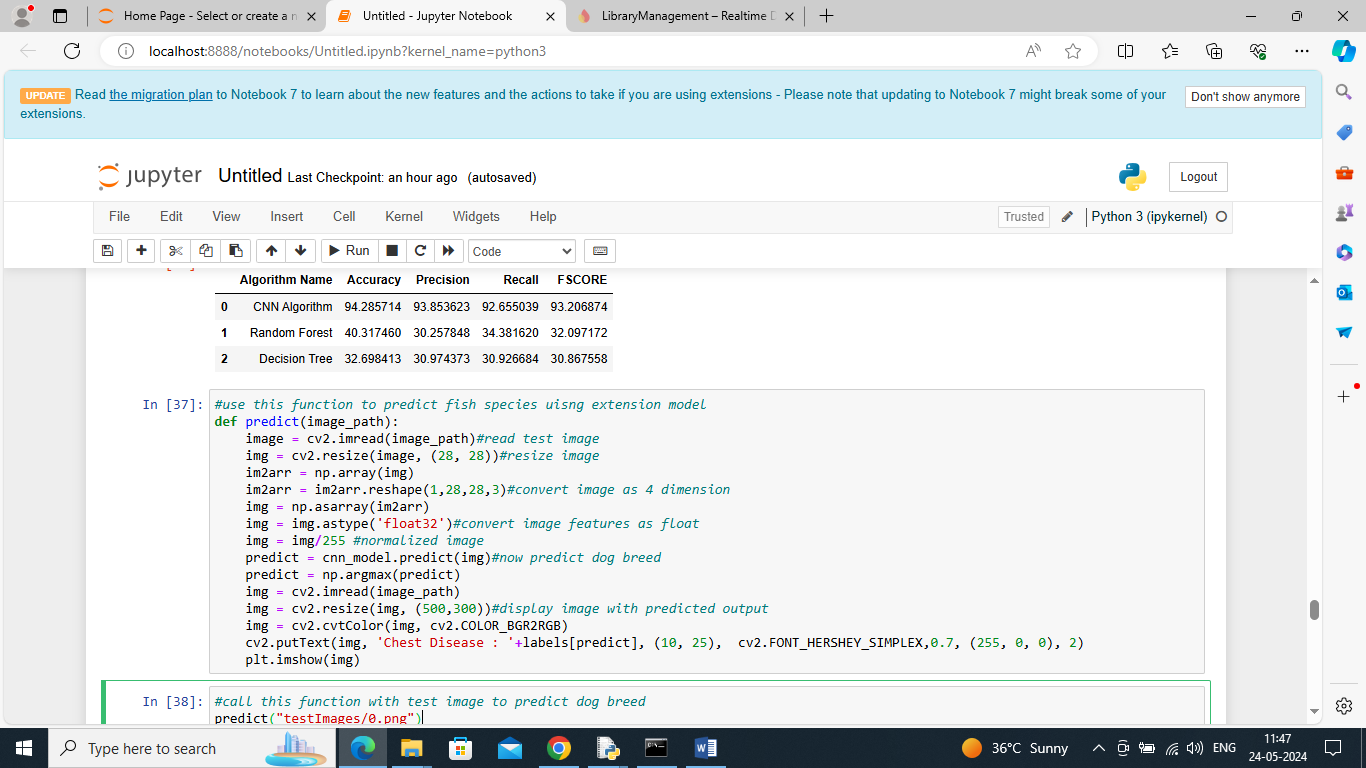


In above screen decision tree got 32% accuracy and can see other metrics also

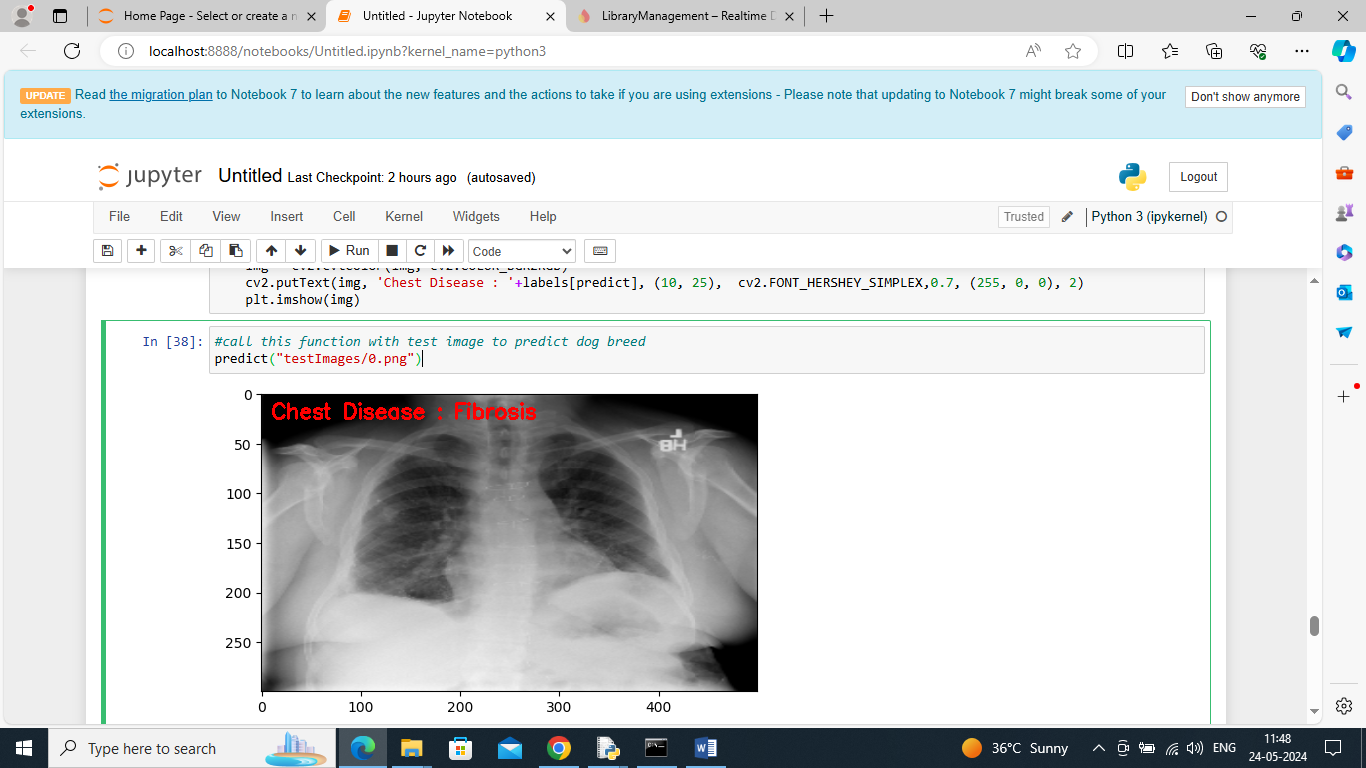
in above graph showing comparison between all algorithms where x-axis represents algorithm names and y-axis represents accuracy and other metrics in different color bars and in all algorithms CNN got best performance



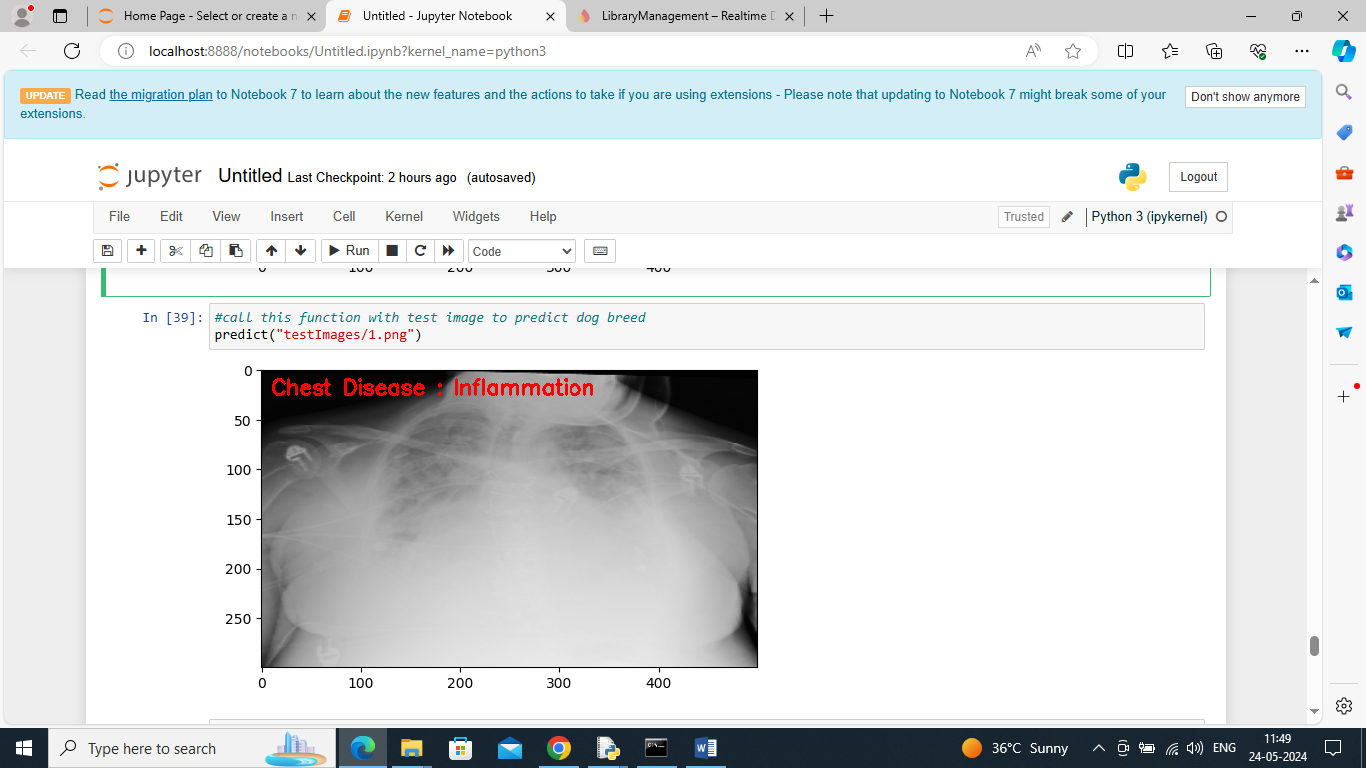
In above screen displaying all algorithm performance in tabular format



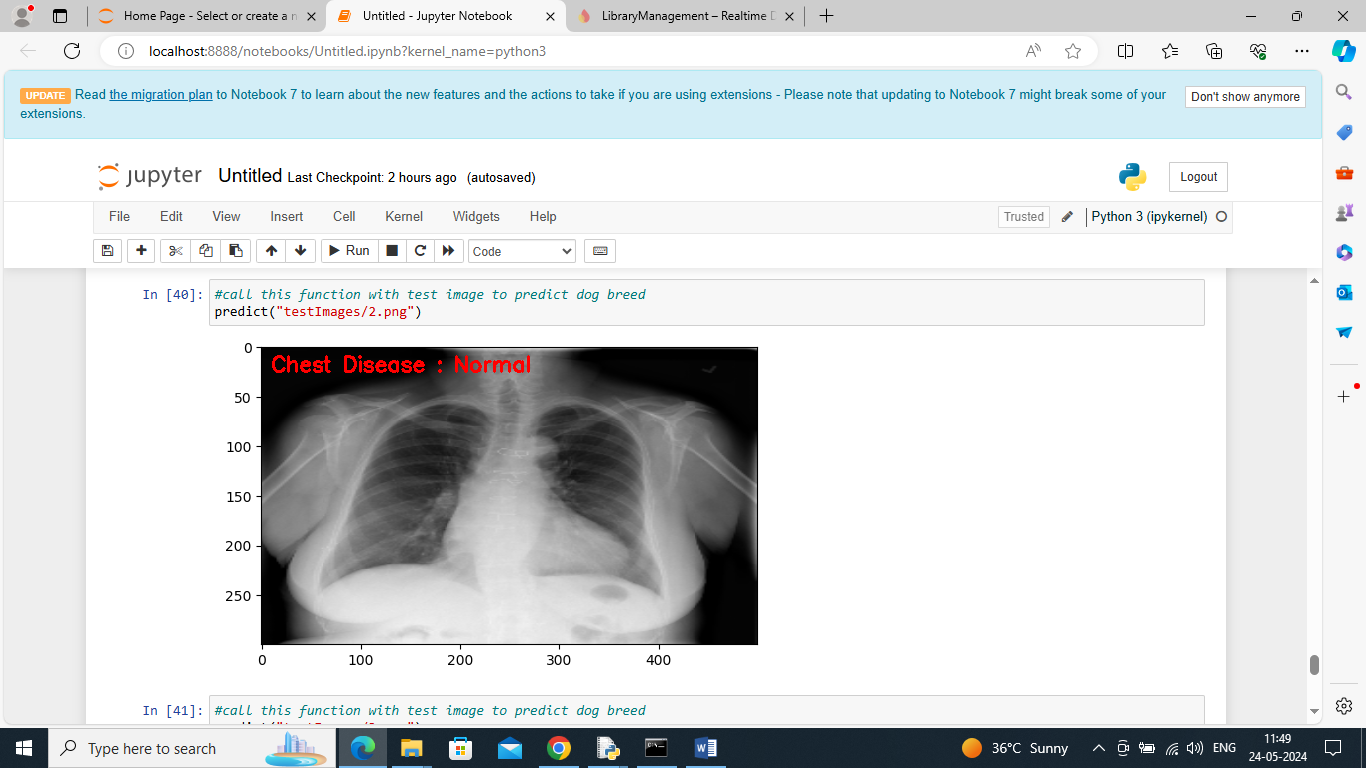
In above screen defining predict function which will take image path as input and then extract features and then apply CNN model to predict type of chest infection from x-ray image

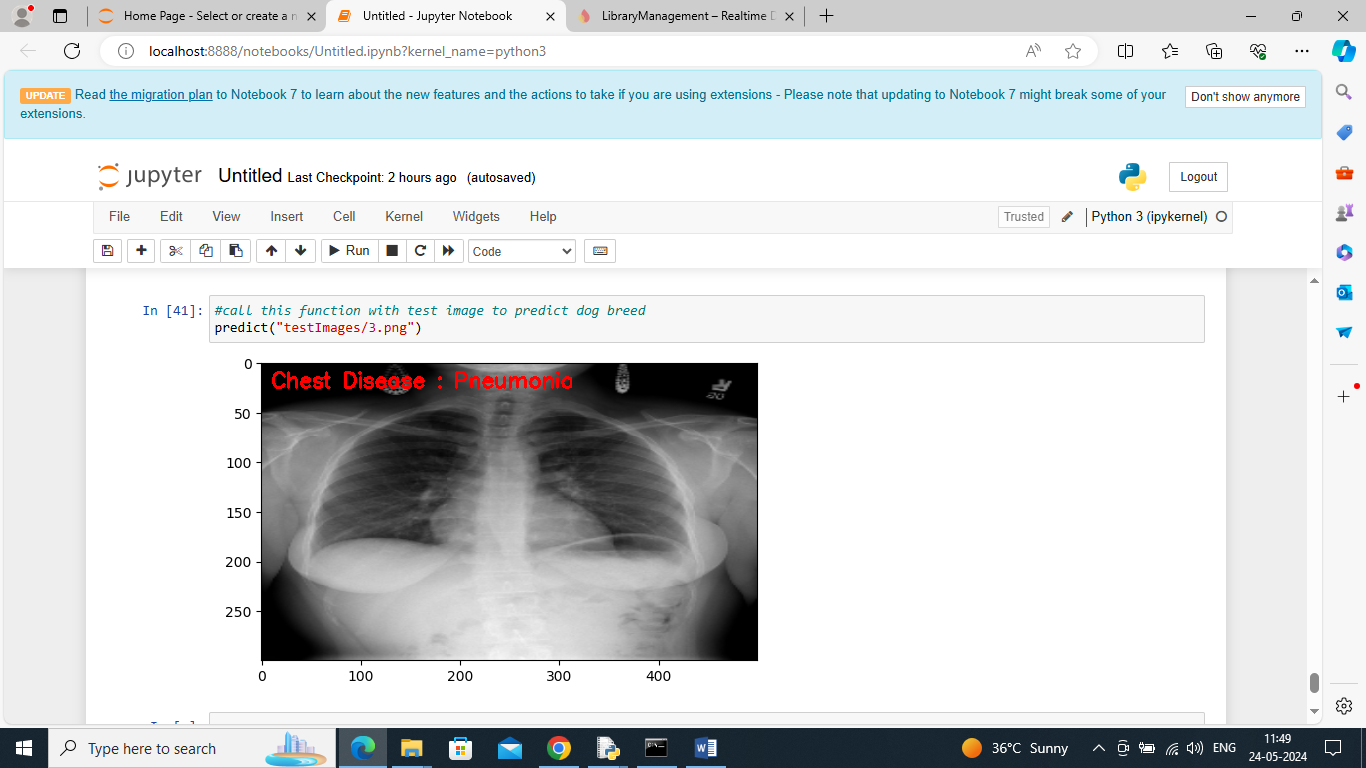


In above screen calling predict function with image path and then CNN detected ‘Fibrosis’ from given image



In above screen inflammation detected

above screen normal detected



In above screen pneumonia detected.

Similarly by giving input image path CNN model can detect type of chest infection

**8. CONCLUSION**

In this paper, we proposed a novel approach composed of two steps for chest disease classifcation using new DL architectures. In the frst step (multi-classifcation), we classifed chest X-ray (CXR) images into three classes (normal, lung disease, and heart disease). In the second step (binary classifcation), we classifed CXR images into specifc diseases to predict whether it is a normal or abnormal case. A dataset of 26,316 CXR images was consolidated by merging images from two public datasets (VinDr-CXR and CheXpert) to train, validate, and test our methods. For this work, we implemented two DL methods to perform our two-step classifcation approach. The frst is called DC-ChestNet which is based on an ensemble learning (EL) of three deep convolutional neural network (DCNN) models. The second method named VT-ChestNet is based on a modifed Swin transformer (M-Swin). Our two methods showed high performance outperforming state-of-the-art models trained and tested individually on our consolidated dataset including, DenseNet121, DenseNet201, EfcientNetB5, and Xception. VT-ChestNet outperformed DC-ChestNet by obtaining the best results on our dataset for the two-steps of our approach. VT-ChestNet achieved an area under curve (AUC) of 95.13% for the frst step. For the second step, it obtained an average AUC of 99.26% for heart diseases and an average AUC of 99.57% for lung diseases. In future work, we intend to investigate the use of different datasets with multi-labeled images. We plan to examine the potential of more transformer-based architectures and implement an explainability algorithm to show the features that our models focused on to classify the images. We aim to explore the ability of the proposed methods to be generalized to other diseases and test their ability to perform multiple classifcation of chest diseases

**9.REFERENCES**

1. Ambati A, Dubey SR. Ac-covidnet: Attention guided contrastive cnn for recognition of covid-19 in chest x-ray images. In: International Conference on Computer Vision and Image Processing, pp. 71–82 (2022)

2. Russo P. Handbook of X-ray Imaging: Physics and Technology. Series in Medical Physics and Biomedical Engineering. CRC Press, New York (2017)

3. World Health Organization: To X-ray or not to X-ray. https://www. who.int/news-room/feature-stories/detail/to-x-ray-or-not-to-x-ray- (Accessed: November 2022)

4. Nasser AA, Akhlouf M. Chest diseases classifcation using cxr and deep ensemble learning. In: 19th International Conference on Content-based Multimedia Indexing, pp. 116–120 (2022)

5. Marciniuk DD, Schraufnage DE, Ferkol T, Fong KM, Joos G, Varela VL, Zar H. The Global Impact of Respiratory Disease. European Respiratory Society, ??? (2017)

6. Tsuneki M. Deep learning models in medical image analysis. J Oral Biosci. 2022;64(3):312–20

7. Demner-Fushman D, Kohli MD, Rosenman MB, Shooshan SE, Rodriguez L, Antani S, Thoma GR, McDonald CJ. Preparing a collection of radiology examinations for distribution and retrieval. J Am Med Inform Assoc. 2016;23(2):304–10.

8. Wang X, Peng Y, Lu L, Lu Z, Bagheri M, Summers RM. Chestxray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classifcation and localization of common thorax diseases. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017; pp. 2097–2106

9. Nguyen H, Lam K, Le L, Pham H, Tran D, Nguyen D, Le D, Pham C, Tong H, Dinh D, et al. Vindr-cxr: An open dataset of chest x-rays with radiologist’s annotations. 2021; preprint arXiv:2012. 15029

10. Irvin J, Rajpurkar P, Ko M, Yu Y, Ciurea-Ilcus S, Chute C, Marklund H, Haghgoo B, Ball R, Shpanskaya K, Seekins J, Mong D, Halabi S, Sandberg J, Jones R, Larson D, Langlotz C, Patel B, Lungren M, Ng A. Chexpert : A large chest radiograph dataset with uncertainty labels and expert comparison. In: AAAI Conference on Artifcial Intelligence, 2019; pp. 590–597

11. Zhu CS, Pinsky PF, Kramer BS, Prorok PC, Purdue MP, Berg CD, Gohagan JK. The prostate, lung, colorectal, and ovarian cancer screening trial and its associated research resource. JNCI. 2013;105(22):1684–93.

12. Johnson AE, Pollard TJ, Berkowitz SJ, Greenbaum NR, Lungren MP, Deng C-Y, Mark RG, Horng S. Mimic-cxr, a de-identifed publicly available database of chest radiographs with free-text reports. Sci Data. 2019;6(1):1–8.

13. Rajpurkar P, Irvin J, Zhu K, Yang B, Mehta H, Duan T, Ding DY, Bagul A, Langlotz CP, Shpanskaya KS, Lungren MP, Ng AY. Chexnet: Radiologist-level pneumonia detection on chest X-rays with deep learning. 2017; arXiv preprint arXiv:1711.05225

14. Sze-To A, Wang Z. tchexnet: Detecting pneumothorax on chest x-ray images using deep transfer learning. In: International Conference on Image Analysis and Recognition, 2019; pp. 325–332

15. Khoiriyah SA, Basof A, Fariza A. Convolutional neural network for automatic pneumonia detection in chest radiography. In: International Electronics Symposium (IES), 2020; pp. 476–480

16. Wang H, Xia Y. Chestnet: A deep neural network for classifcation of thoracic diseases on chest radiography. 2018; arXiv preprint arXiv:1807.03058

17. Pham HH, Le TT, Tran DQ, Ngo DT, Nguyen HQ. Interpreting chest X-rays via cnns that exploit hierarchical disease dependencies and uncertainty labels. Neurocomputing. 2021;437:186–94.

18. Guendel S, Grbic S, Georgescu B, Liu S, Maier A, Comaniciu D. Learning to recognize abnormalities in chest X-rays with locationaware dense networks. In: Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications, 2019; pp. 757–765.

19. Kim S, Rim B, Choi S, Lee A, Min S, Hong M. Deep learning in multi-class lung diseases’ classifcation on chest X-ray images. Diagnostics. 2022;12(4):915.

20. Blais M-A, Akhlouf MA. Deep learning and binary relevance classifcation of multiple diseases using chest X-ray images. In: 43rd Annual International Conference of the IEEE Engineering in Medicine Biology Society (EMBC), 2021; pp. 2794–2797.

21. Huang G, Liu Z, Van Der Maaten L, Weinberger KQ. Densely connected convolutional networks. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017; pp. 4700–4708

22. Deng J, Dong W, Socher R, Li L-J, Li K, Fei-Fei L. Imagenet: A large-scale hierarchical image database. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2009; pp. 248–255

23. Chollet F. Xception: Deep learning with depthwise separable convolutions. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017; pp. 1251–1258

24. Tan M, Le Q. Efcientnet: Rethinking model scaling for convolutional neural networks. In: International Conference on Machine Learning, vol. 97, 2019; pp. 6105–6114

25. Krizhevsky A, Sutskever I. H. geofrey e.,“alex net,”. Adv. Neural Inf. Process. Syst 2012; 25, 1–9

26. Dosovitskiy A, Beyer L, Kolesnikov A, Weissenborn D, Zhai X, Unterthiner T, Dehghani M, Minderer M, Heigold G, Gelly S, Uszkoreit J, Houlsby N. An image is worth 16x16 words: Transformers for image recognition at scale. 2020; arXiv preprint arXiv:2010.11929

27. Touvron H, Cord M, Douze M, Massa F, Sablayrolles A, Jégou H. Training data-efcient image transformers & distillation through attention. In: International Conference on Machine Learning, 2021; pp. 10347–10357

28. Hong W, Lao J, Ren W, Wang J, Chen J, Chu W. Training object detectors from scratch: An empirical study in the era of vision transformer. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022; pp. 4662–4671

29. Ghali R, Akhlouf MA, Mseddi WS. Deep learning and transformer approaches for UAV-based wildfre detection and segmentation. Sensors. 2022;22(5):1977.

30. Liu Z, Lin Y, Cao Y, Hu H, Wei Y, Zhang Z, Lin S, Guo B. Swin transformer: Hierarchical vision transformer using shifted windows. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 2021; pp. 10012–10022.