Acknowledgements

It is with extreme pleasure and pride that we present our B-Tech. dissertation titled "***Determination of medicinal leaf properties using machine learning***". We would like to express our sincere thanks and gratitude to the following people, who stood by us throughout, helping us withmuch required inputs, guidance, knowledge and supported us.

We take great pleasure to express our sincere thanks and gratitude to academic project guide **DR. PADMAPRIYA DHARSHINI** Asst. Professor Department of CSE, for his support, guidance andsuggestions throughout the project which is leading this project for the completion.

We express our sincere thanks to**, DR. DILIP KUMAR MAHANTY**, our respected Dean and to**DR. RINKI SHARMA,** Head of Department of Computer Science and Engineering, for theirkind cooperation and support toward out dissertation, and to the management of Ramaiah University of Applied Science for their continued support. We are thankful to the staff members of the Computer Science and Engineering, RUAS for giving us good support and suggestion.

Lastly, we would like to thanks our parents and friends for their continued support, encouragementand motivation and God for paving our way of success in this object.

Table of Contents

Acknowledgements 1

Abstract 8

List of Figures 6

List of Tables 7

Chapter-**1: Introduction** 9

1.1 Introduction 9

1.2 Literature Survey 10

1.2.1 Paper 1 10-12

1.2.2 Paper 2 12-13

Chapter-**2: Background Theory** **14**

2.1 Background Theory: 14

2.1.2Python 14

2.1.3 Tkinter 14-15

Chapter-**3: Aim and Objectives** 16

3.1 Title 16

3.2 Aim 16

3.3 Objectives 16

3.4 Functional Requirements 17

3.5 Non- functional requirements 17

3.6 Method and Methodology 18

Chapter-**4: Problem Solving** 19

4.1 CNN 19

4.2 Design 20

4.2.1 Algorithm 20

4.2.2 Flow chart for training 20

4.2.3 Flow chart for testing 21

4.2.4 Uml diagrams 21

4.2.5 Use case diagrams 22

4.2.6 Sequence diagram 22-2

4.2.7 Activity diagram 23

4.2.8 Object diagram	24
4.2.9 Components diagram	24
4.3 Libraris used in this model	25
4.4 Applications of python	25
4.4.1 GUI-Based desktop applications	25-26
4.4.2 Web frame works and web applications	26
4.4.3 Enterprise and business applications	26
4.4.4 Operating system	26
4.4.5 Language development	26
4.4.6 Prototyping	27
4.5 Python machine learning	27
4.5.1 Machine learning	27-28
4.5.2 Concepts of learning	28
4.5.2.1 supervised learning	29-30
4.5.2.2 unsupervised learning	30-31
4.6 system configuration	32
4.6.1 H/W system configuration	32
4.6.2 S/W system configuration	32
4.7 Source code	33-37
4.8 Implementation	37-42
Chapter 5 :Testing	
5.1 Test objectives	43
5.2 Testing design	44
5.2.1 White box testing	44

5.2.2 Black box testing 44

5.2.3 Testing strategies 44-45

5.2.4 Levels of testing 45

5.2.4.1 Uint testing 45

5.2.4.2 integration testing 46

5.2.4.3 Function testing 46-48

5.2.4.4 Validation testing 48

5.2.4.5 Alpha testing 49

5.2.4.6 Beta testing 49

5.2.4.7 System testing Acceptance testing 50

Chapter-**6 Conclusion** **51**

References 52

List of Figures

Figure 1: Architecture of the Proposed Model
Figure 2: Flow chart for Training
Figure 3: Flow chart for Training
Figure 4: Usecase Diagram
Figure 5: Sequence Diagram
Figure 6: Activity Diagram
Figure 7: Object diagram
Figure 8: Components Diagram
Figure 9: Source code of Training
Figure 10: Source code of Testing
Figure 11: Dataset of leafs

Figure 12: Graph of Accuracy

Figure 13: Graph of Loss model

Figure 14: Trained model with Validation loss and Accuracy

Figure 15: Click here to upload the image

Figure 16: Select & Upload the Image
Figure 17: Predicting the image
Figure 18: Name of the predicted Leaf
Figure 19: Properties & uses of predicted Leaf

Figure 20: Result

List of Tables

Table 1: Method and Methodology

Table 2: Hardware Configuration

Table 3: Software Configuration

Table 4: Test Cases 1

Table 5: Test Cases 2

Table 6: Test Cases 3

ABSTRACT

Abstract:- Classifying plant species has taken much attention in the research area to help people recognizing plants easily. In recent years, the convolutional neural networks (CNN) have achieved tremendous computer vision results, especially in image classification. Usually, humans find it difficult to recognize proper medicinal plants. It requires the intuition of an expert botanist, which is a time consuming manual task. In this research, we proposed an automated system for the medicinal plant classification, which will help people identify useful plant species quickly. A new dataset of medicinal plants of India is introduced, collected from different regions across the country, and some state-of-the images collected from different sources. After that, convolutional neural network is employed to extract the high-level features for the classification trained with the data augmentation technique with SVM algorithms. The training process was done on some images, and the experimental result on another some images proved that this method is quite feasible and effective.

Keywords— Computer Vision, Image Recognition, Leaf Classification, Convolutional Neural Network SVM.

CHAPTER 1

INTRODUCTION

1.1: INTRODUCTION:-

There are almost some plant species around the world. Among all the plants, some used in medicine, which provides many drugs from the ancient time to the present. Within the context of India, there are about 449 enlisted medicinal plants. Among them, a lot of traditional and modern medicine exists which can be derived from these plants. Considering this huge number, the medicinal plant classification is a fairly difficult task and lengthy process even for experienced botanists. Because it relies much on the inherited knowledge of an expert botanist. Also, the plants are hard to recognize because of their almost similar shape and color. So, it is important to study and classify plants correctly for further use. As manually identifying these plants requires a lot of time, an automation process is needed to automate this plant classification system so that the people with no botanical knowledge can identify plant easily. Our objective of this research is to make the process of plant classification easy for a human. In this regards, we propose an approach for classification of India medicinal plants. This method offers an alternative solution to the traditional way of identifying medicinal plant images by the botanist and reduces time, the manpower and associated costs for the whole process. To the best of our knowledge, there is no availability of medicinal plant dataset in India for research. In this paper, we introduce an image dataset of 6 classes of India medicinal plants which were taken in different conditions. Also, a convolutional neural network is applied to classify the plants. In CNN transfer learning, there are feature maps which capture the result of the filters to an input image.

1.2 Literature Review:-

1.2.1: Paper1

Convolution Neural Network in Precision Agriculture for Plant Image Recognition and Classification

Author : Halimatu Sadiyah Abdullahi, Ray E. Sheriff, Fatima Mahieddine

Key words: CNN ,Convolution, Feature extraction, Image analysis, Validation and Precision Agriculture.

Abstract— Agriculture is essential to the continued existence of human life as they directly depend on it for the production of food. The exponential rise in population calls for a rapid increase in food with the application of technology to reduce the laborious work, maximize production and reduce the impact of pressure on the environment. Precision Agriculture is thought to be solution required to achieve the production rate required. There has been a significant improvement in the area of image processing and data processing which has being a major challenge previously in the practice of precision Agriculture. A database of images is collected using remote sensing techniques and the images are analyzed to develop a model to determine the right treatment plans for different crop types and different regions. Features of images from vegetation's need to be extracted, classified, segmented and finally fed into the model. Different techniques have been applied to the processes from the use of the neural network, support vector machine, fuzzy logic approach and recently, the most effective approach generating fast and excellent results using the deep learning approach of convolution neural network for image classifications. Deep Convolution neural network is used in plant images recognition and classification to optimize production on a maize plantation. The experimental results on the developed model yielded results with an average accuracy of 99.58%

INTRODUCTION

Food is an essential necessity of human life and requires its continuous supply and production to cater the needs of the growing population through sustainable agriculture. This can be achieved with the application of emerging technologies in the sector to maximize production across a vegetation. Technology can aid/improve agriculture in several ways through pre- planning and post-harvest by the use of computer vision technology through image processing to determine the soil nutrient composition, right amount, right time, right place application of farm input resources like fertilizers, herbicides, water, weed detection, early detection of pest and diseases etc. To achieve sustainable agriculture, Precision agriculture (PA) is used and it is the technology that enhances farming techniques. It prepares the land before planting, ensures an equally fertile vegetation across the field, monitors plantations during in-season growth, detects of early onset of pest and diseases, right amount, right time and right application of farm input resources to the right location through harvest and post- harvest processes [1][2]. PA involves remotes sensing, the use of Geographical Information System (GIS), Global Positioning System (GPS) and data analysis. This technology enables the collection of field data in a known- destructive way and making the data available for analysis and implementation on the field [3].

The technology is made necessary due to a spatial and temporal variability of the field revealing information as, patterns and spatial relationships. The stages of remote sensing involve energy interaction with the atmosphere and interaction with the target, then the interaction of the energy with the sensor (camera), data transmission and processing, image analysis and finally the application of results to treatment areas required [4][5]. Simple applications on the farm involve determining the location of sampling sites, plotting maps for use in the field, examining the distribution of soil types in relation to yields and productivity. Other applications take advantage of the analytical capabilities of GIS and RS software for vegetation classification to predict crop yield, environmental impacts, modeling of surface water drainage patterns, tracking animal migration patterns and other ranges of applications [6].

CONCLUSION

The use of off- the- shelf ConvNet representations for the problem of estimating plant health on a maize plantation was used with an average prediction accuracy of 99.58%. So far this is the best result achieved when compared with other techniques and proves the ability of the model to accurately predict the treatment solutions to produce an equally fertile land for optimization of production. This model can be developed for different regions with separate soil composition, weather and other factors from one geographical region to another for making informed decisions on a plantation.

1.2.2: PAPER 2

Deep Residual Networks for Plant Identification

Authors: Vinit Bodhwani , D. P. Acharjya a , Umesh Bodhwani

ABSTRACT

Advancing the knowledge and understanding of plants growing around us plays a very important and crucial role in medicinally, economically and in a sustainable agriculture. The identification of plant images in the field of computer vision has become an interdisciplinary focus. Taking benign conditions of quick advancement in computer vision and deep learning algorithms, convolutional neural networks (CNN) approach is taken into consideration to learn feature representation of 185 classes of leaves taken from Columbia University, the University of Maryland, and the Smithsonian Institution. For the large-scale classification of plants in the natural environment, a 50-layer deep residual learning framework consisting of 5 stages is designed. The proposed model achieves a recognition rate of 93.09 percent as the accuracy of testing on the LeafSnap dataset, illustrating that deep learning is a highly promising forestry technology.

INTRODUCTION

Plants are the essential resource for human well-being providing us with oxygen and food. So, the researchers along with breeding industry are making great efforts to continue agriculture for a long period without any interruptions. A good knowledge of plants is essential to help fully recognize new, different or uncommon plant species in order to support the ecosystem, promote the drug industry, and promote sustainability and productivity in agriculture. Different modern and advanced models have been proposed for automatic plant identification as the deep learning technology advances. In addition, the classification and learning of an item in an image in computer vision is a really challenging task. Today, most researchers use different variants in leaf properties as a comparable method for the study of new plants, and some leaf datasets, including Flavia, Swedish and ICL, have been transmitted, and fewer attempts have been made to extract local features of leaf, flower or fruit. However, much work has been conveyed towards identification and prediction in different applications. Additionally, computational intelligence techniques play a vital role in identification and prediction applications. For example, rough computing is hybridized with neural network [1, 2], genetic algorithm [3, 4], and soft set [5]. Deep convolutional neural networks play a vital role in order to learn distinct features of an image using image classification techniques. To enhance the quality of images, we need to increase their resolution, which results to an establishment of deeper neural networks and increasing the number of stacked layers of the network become very crucial as mentioned in recent evidence [6,7]. These large number of hidden layers establish the problem of vanishing gradient descent [8] which could not be resolved by traditional machine learning techniques. To overcome aforementioned challenges and invigorated by the deep learning breakthrough in image recognition, a 50-layer deep learning model using residual networks is designed for uncontrolled plant identification on the LeafSnap dataset which consists of 185 different tree species. The model proposed achieves an accuracy rate of 93.09 percent with 0.24 percent error.

CONCLUSION

This paper examined a methodology of deep learning using CNN to learn discriminatory characteristics for plant categorization from leaf images. We demonstrated how skip-connections in residual networks help to address the Vanishing Gradient problem which resulted in 93.09% accuracy in the test set, manifesting that deep learning is the promising technology for large-scale plant classification in the natural environment. We plan to try to evaluate different CNN architectures in our future work to increase performance. In particular, we plan to extend the deep learning model to include prediction, disease segmentation, insect detection and so on from the classification task.

CHAPTER 2

BACKGROUND THEORY

2.1 Background Theory:

This section gives complete knowledge and understanding of different technologies and framework that were required in this project for its completion. The background Theory is listed below:

2.1.2 Python

Python is used for server-side web development, software development, mathematics, and system scripting, and is popular for Rapid Application Development and as a scripting or glue language to tie existing components because of its high-level, built- in data structures, dynamic typing, and dynamic binding. Program maintenance costs are reduced with Python due to the easily learned syntax and emphasis on readability. Additionally, Python's support of modules and packages facilitates modular programs and reuse of code. Python is an open source community language, so numerous independent programmers are continually building libraries and functionality for it

2.1.3:Tkinter

Tkinter is a standard Python library used to create graphical user interfaces (GUIs) in Python. It is based on the Tk GUI toolkit and is available on most operating systems. Tkinter provides a wide range of GUI widgets, such as buttons, labels, text boxes, check boxes, radio buttons, menus, and more.

Some of the features of Tkinter include:

- Easy to use: Tkinter is easy to learn and use, making it a great choice for beginners.
- Cross-platform: Tkinter works on most operating systems, including Windows, macOS, and Linux.
- Customizable: You can customize the appearance and behavior of the widgets using various options and methods.
- Integration with Python: Tkinter is a part of the standard Python library, making it easy to integrate with other Python modules and libraries.
- Extensible: You can extend the functionality of Tkinter by creating custom widgets or by using other Python libraries that work with Tkinter.

Tkinter is a popular choice for creating simple GUI applications in Python, but for more complex applications, you may need to use other GUI frameworks such as PyQt or wxPython.

CHAPTER 3

AIM AND OBJECTIVES

This chapter focuses on defined title and Aim of the project correctly and clearly. Later this chapter also includes the required objectives that needed to be fulfilled in order to complete this project. Functional Requirements are well documented in this section since it is required to design different diagrams leading to complete view of the project. This is followed by method and methodologies that tabulates the procedure at will be followed in order to complete the objectives. This section then ends with a summary.

3.1Title

MEDICINAL LEAF DETECTION USING MACHINE LEARNING

3.2 Aim

The aim of the determination of medicinal leaf properties is to identify and understand the medicinal compounds present in the leaves. This information can be used to develop new medicines or to improve existing treatments additionally.

3.3 Objectives

- To review the literature and identify the various properties of plants and there by gathering user requirements.
- To identify the application requirements and user requirements based on the literature review.
- To develop an ML based model that provides in-depth information about medicinal leaves and herbs.
- To test and validate the functionality of the application using suitable test cases against the requirements specified.

3.4Functional requirements

- The system should collect data on the medicinal leaf properties and it uses.
- The system should pre-process the collected data, such as cleaning and formatting, to prepare it for machine learning analysis.
- The system should extract relevant features from the pre-processed data that are important for predicting medicinal leaf properties.
- The system should select machine learning model that are capable of predicting medicinal properties accurately based on the type of data and the prediction task.
- The system should deploy the trained model for practical use, such as predicting medicinal properties of new leaf samples.
- The system should able to identify the leaf name and display the advantages and disadvantages associated with it.

3.5 Non-functional requirements

- Accuracy: The ML model must be able to accurately determine the medicinal properties of a given leaf sample. This includes both the ability to correctly identify the active compounds present in the sample, as well as the ability to predict the leaf properties and uses.
- Speed: The ML model should be able to process samples quickly, particularly if it is intended for use
- Reliability: The ML model should be reliable and consistent, with a low error rate and minimal false rate.
- Scalability: The ML model should be able to handle large volumes of data, as well as varying levels of complexity in the input samples.
- Security: The ML model should be designed in a secured manner
- Interpretability: The ML model should be able to provide clear and understandable explanations for its predictions, in order to facilitate further research and development.
- Usability: The ML model should be user-friendly, with a clear and intuitive interface that allows users to easily input data, view results, and adjust parameters as needed.
- Maintainability: The ML model should be designed with maintainability in mind, with clear documentation, modular code structure, and the ability to easily update and improve the model over time.

3.5 METHODOLOGY

METHODOLOGY	**DECSCRIPTION**
Sample collection	Leaves from different plant species are collected and prepared for analysis.
Data acquisition	The medicinal properties present in the leaves are analysed using the existing dataset.
Data pre-processing	The acquired data is cleaned and processed for further analysis.
Model training	The machine learning model is trained on a large dataset of labelled samples, where each sample is associated with a set of medicinal properties.
Model validation	The model is validated using a separate dataset of samples to check for its performance and accuracy.
Model deployment	After successful training and pre-processing, comparison of the test image with the trained model takes place.

Table 1: Methodology

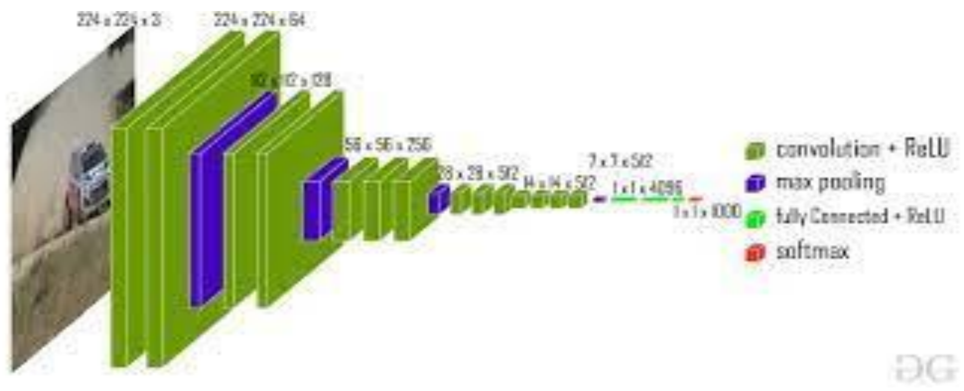
CHAPTER 4

PROBLEM SOLVING

In this section, the actual dissection of project is done and each module is built piece by piece in order to complete the project. In design section, the application has been built in accordance with functional requirements based on which diagrams like Use Case and Low Level Sequence Diagram is drawn. In implementation section, snips of important code is displayed with their explanation given below. In testing section, all functional requirements are tested and the result is analysed resulting in status of test condition. In the ending section, performance analysis is done on various factors such as battery, booting time and time requirement for major operation.

4.1 CNN:-

Convolutional Neural Networks are a complex neural network chain which work to get the features of an image from a dataset which is trained and classify them to get the required output. It trains the neural networks by using the dataset images and changing them to numerical values. The main advantage of CNN compared to its predecessors is that it automatically detects the important features without any human supervision....ConvNets are more powerful than machine learning algorithms and are also computationally efficient. These numerical values are then put into numerical arrays based on their categorized characteristics. These arrays are then put into different nodes in the network and passed through multiple iterations based on the input given. The CNN models are used for geographical classification in multiple companies which require data to be classified in a quick and secure way it almost acts like a filter removing dust and separates the features of the images.



VGG16 architecture

4.2 Design

Design is necessary when it comes to development since it acts as a blueprint for entire process from requirement making to finished (final product). Hence, in this section designs specific to this project like use case, sequence diagram etc. are attached

4.2.1Algorithm:-

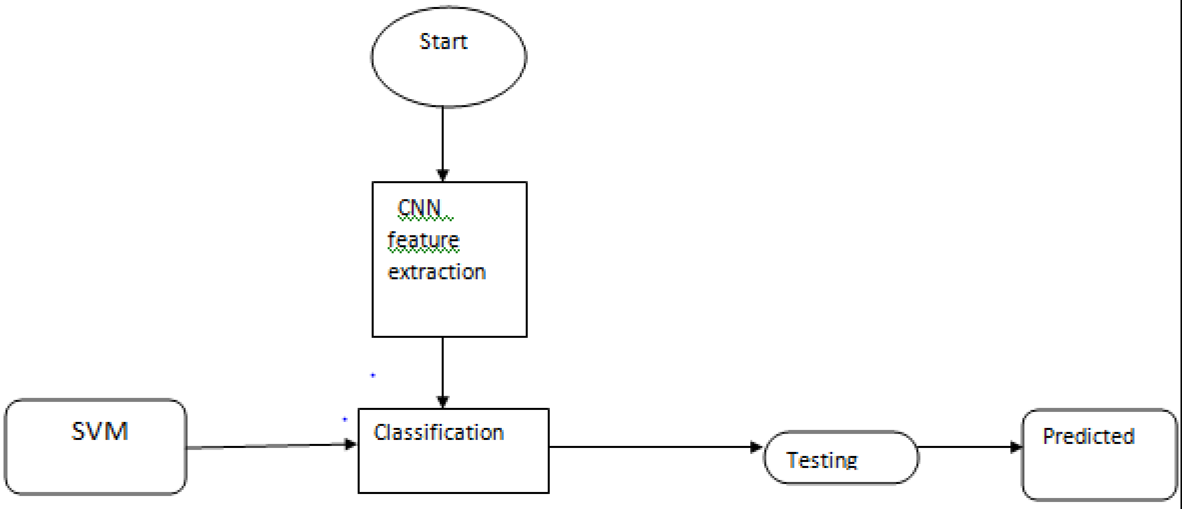


Fig1: Architecture of the proposed model

4.2.2Flow chart for training.

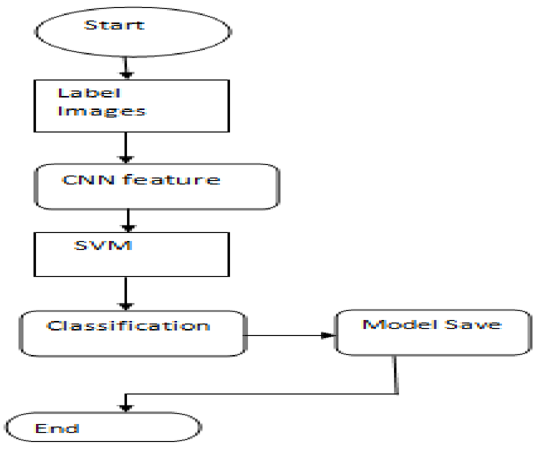


Fig2:Flow chart for training.

4.2.3 Flow chart for testing.

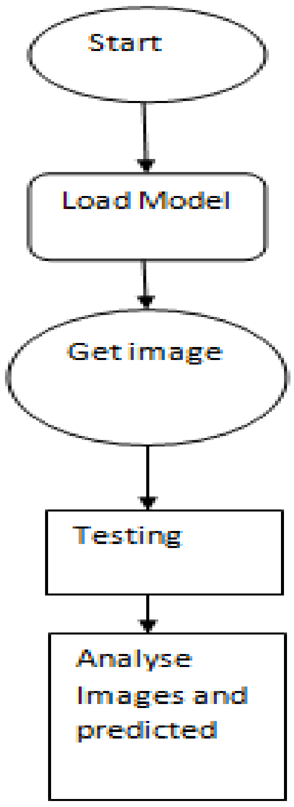


Fig: 3 Flow chart for Testing

4.2.4 UML Diagrams

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

4.2.5 Use Case Diagrams

A use case diagram at its simplest is a graphical 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.

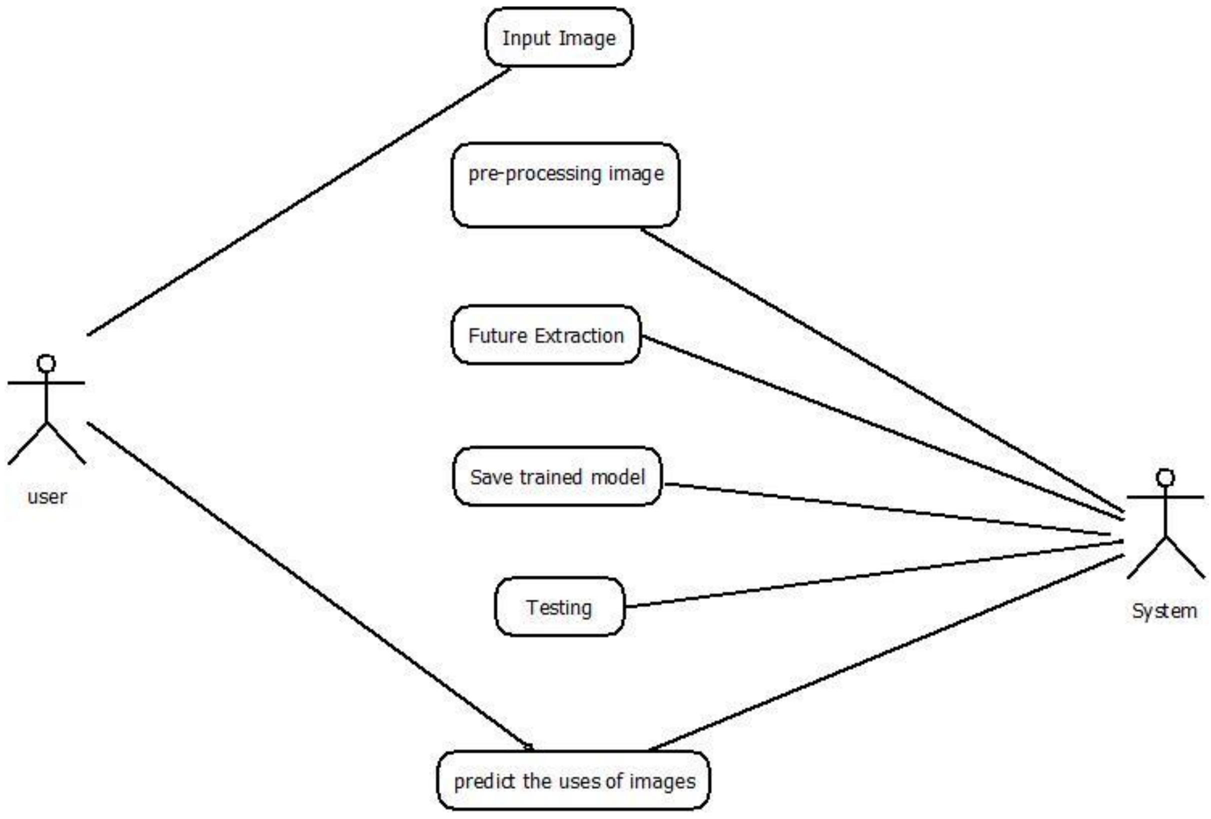


Fig4 : Use Case Diagram

4.2.6 Sequence Diagram

A sequence diagram is a type of interaction diagram because it describes how—and in what order—a group of objects works together. These diagrams are used by software developers and business professionals to understand requirements for a new system or to document an existing process.

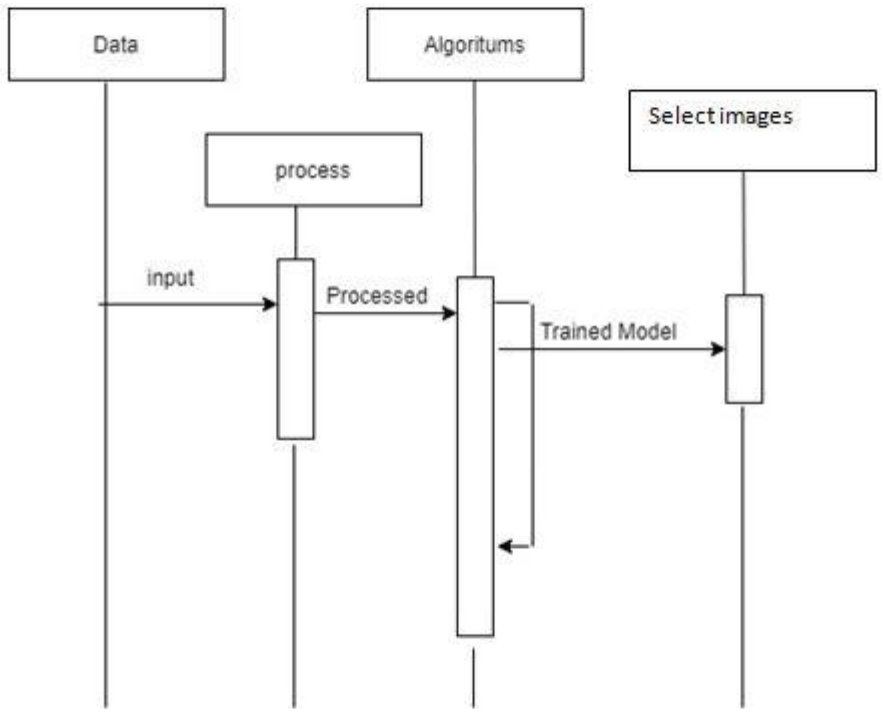


Fig 5:Sequence Diagram

4.2.7Activity diagram:

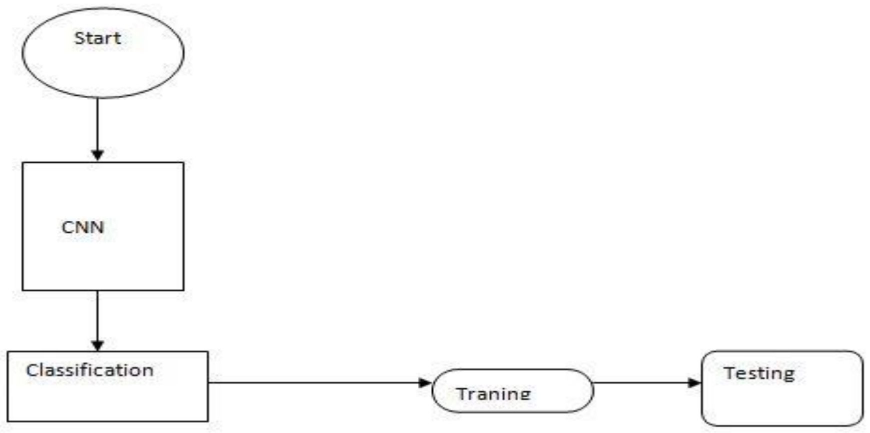


Fig 6:Activity Diagram

4.2.8 Object diagram:-

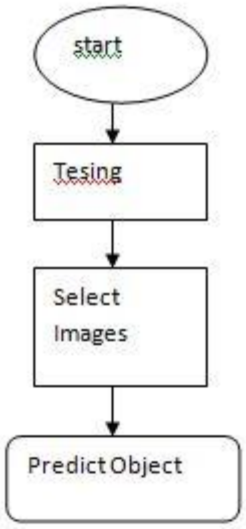


Fig 7:Object Diagram

4.2.9Components Diagram:-

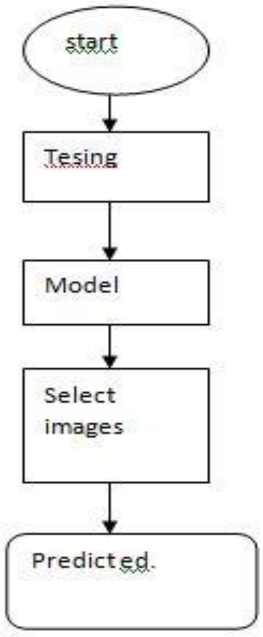


Fig 8:Components Diagram

4.3 Libraries Used for this model:

Matplotlib: It is an amazing visualization library in Python for 2D plots of arrays, It is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack.[10]

Seaborn: This library sits on top of matplotlib. In a sense, it has some flavors of matplotlib while from the visualization point, it is much better than matplotlib and has added features as well.[10]

Pandas: Pandas are an important library for data scientists. It is an open-source machine learning library that provides flexible high-level data structures and a variety of analysis tools. It eases data analysis, data manipulation, and cleaning of data. Pandas support operations like Sorting, Re-indexing, Iteration, Concatenation, Conversion of data, Visualizations, Aggregations, etc[9]

Numpy : The name “Numpy” stands for “Numerical Python”. It is the commonly used library. It is a popular machine learning library that supports large matrices and multi-dimensional data. It consists of in-built mathematical functions for easy computations. Even libraries like TensorFlow use Numpy internally to perform several operations on tensors. Array Interface is one of the key features of this library.[9]

4.4 Applications of Python

4.4.1 GUI-Based Desktop Applications:

Python has simple syntax, modular architecture, rich text processing tools and the ability to work on multiple operating systems which make it a desirable choice for developing desktop-based applications. There are various GUI toolkits like wxPython, PyQt or PyGtk available which help developers create highly functional Graphical User Interface (GUI). The various applications developed using Python includes:

- Image Processing and Graphic Design Applications:

Python has been used to make 2D imaging software such as Inkscape, GIMP, Paint Shop Pro and Scribus. Further, 3D animation packages, like Blender, 3ds Max, Cinema 4D, Houdini, Lightwave and Maya, also use Python in variable proportions.

- Scientific and Computational Applications:

The higher speeds, productivity and availability of tools, such as Scientific Python and Numeric Python, have resulted in Python becoming an integral part of applications involved in computation and processing of scientific data. 3D modeling software, such as FreeCAD, and finite element method software, such as Abaqus, are coded in Python.

• Games:

Python has various modules, libraries and platforms that support development of games. For example, PySoy is a 3D game engine supporting Python 3, and PyGame provides functionality and a library for game development. There have been numerous games built using Python including Civilization-IV, Disney's Toontown Online, Vega Strike etc.

4.4.2. Web Frameworks and Web Applications:

Python has been used to create a variety of web-frameworks including CherryPy, Django, TurboGears, Bottle, Flask etc. These frameworks provide standard libraries and modules which simplify tasks related to content management, interaction with database and interfacing with different internet protocols such as HTTP, SMTP, XML-RPC, FTP and POP. Plone, a content management system; ERP5, an open source ERP which is used in aerospace, apparel and banking; Odoo – a consolidated suite of business applications; and Google App engine are a few of the popular web applications based on Python.

4.4.3. Enterprise and Business Applications:

With features that include special libraries, extensibility, scalability and easily readable syntax, Python is a suitable coding language for customizing larger applications. Reddit, which was originally written in Common Lips, was rewritten in Python in 2005. Python also contributed in a large part to functionality in YouTube.

4.4.4. Operating Systems:

Python is often an integral part of Linux distributions. For instance, Ubuntu's Ubiquity Installer, and Fedora's and Red Hat Enterprise Linux's Anaconda Installer are written in Python. Gentoo Linux makes use of Python for Portage, its package management system.

4.4.5. Language Development:

Python's design and module architecture has influenced development of numerous languages. Boo language uses an object model, syntax and indentation, similar to Python. Further, syntax of languages like Apple's Swift, CoffeeScript, Cobra, and OCaml all share similarity with Python.

4.4.6. Prototyping:

Besides being quick and easy to learn, Python also has the open source advantage of being free with the support of a large community. This makes it the preferred choice for prototype development. Further, the agility, extensibility and scalability and ease of refactoring code associated with Python allow faster development from initial prototype.Since its origin in 1989, Python has grown to become part of a plethora of web-based, desktop-based, graphic design, scientific, and computational applications. With Python available for Windows, Mac OS X and Linux / UNIX, it offers ease of development for enterprises. Additionally, the latest release Python 3.4.3 builds on the existing strengths of the language, with drastic improvement in Unicode support, among other new features.

Versions of python

- Python 1.0 Introduced in jan 1994
- Python 2.0 Introduced in oct 2000
- Python 3.0 introduced in dec 2008

latest version

python 3.6.3 è 2016

python 3.7

Any new version should provide support for old version programs

- There is no- backward compatibility support
- Python 3 is not support to python 2 program

4.5 Python Machine Learning

4.5.1 Machine Learning (ML) is an automated learning with little or no human intervention. It involves programming computers so that they learn from the available inputs. The main purpose of machine learning is to explore and construct algorithms that can learn from the previous data and make predictions on new input data.

The **input** to a learning algorithm is training data, representing experience, and the **output** is any expertise, which usually takes the form of another algorithm that can perform a task. The input data to a machine learning system can be numerical, textual, audio, visual, or multimedia. The corresponding output data of the system can be a floating-point number, for instance, the velocity of a rocket, an integer representing a category or a class, for example, a pigeon or a sunflower from image recognition.

In this chapter, we will learn about the training data our programs will access and how learning process is automated and how the success and performance of such machine learning algorithms is evaluated.

4.5.2 Concepts of Learning

Learning is the process of converting experience into expertise or knowledge.
Learning can be broadly classified into three categories, as mentioned below, based on the nature of the learning data and interaction between the learner and the environment.

- Supervised Learning
- Unsupervised Learning
- Semi-supervised learning

Similarly, there are four categories of machine learning algorithms as shown below:

- Supervised learning algorithm
- Unsupervised learning algorithm
- Semi-supervised learning algorithm
- Reinforcement learning algorithm

However, the most commonly used ones are **supervised** and **unsupervised learning**.

4.5.2.1Supervised Learning

Supervised learning is commonly used in real world applications, such as face and speech recognition, products or movie recommendations, and sales forecasting. Supervised learning can be further classified into two types: **Regression** and **Classification**.

Regression trains on and predicts a continuous-valued response, for example predicting real estate prices.

Python Machine Learning – Types of Learning

Classification attempts to find the appropriate class label, such as analyzing positive/negative sentiment, male and female persons, benign and malignant tumors, secure and unsecure loans etc.

In supervised learning, learning data comes with description, labels, targets or desired outputs and the objective is to find a general rule that maps inputs to outputs. This kind of learning data is called **labeled data**. The learned rule is then used to label new data with unknown outputs.

Supervised learning involves building a machine learning model that is based on **labeled samples**. For example, if we build a system to estimate the price of a plot of land or a house based on various features, such as size, location, and so on, we first need to create a database and label it. We need to teach the algorithm what features correspond to what prices. Based on this data, the algorithm will learn how to calculate the price of real estate using the values of the input features.

Supervised learning deals with learning a function from available training data. Here, a learning algorithm analyzes the training data and produces a derived function that can be used for mapping new examples. There are many **supervised learning algorithms** such as Logistic Regression, Neural networks, Support Vector Machines (SVMs), and Naive Bayes classifiers.

Common **examples** of supervised learning include classifying e-mails into spam and not-spam categories, labeling webpages based on their content, and voice recognition.

4.5.2.2Unsupervised Learning

Unsupervised learning is used to detect anomalies, outliers, such as fraud or defective equipment, or to group customers with similar behaviors for a sales campaign. It is the opposite of supervised learning. There is no labeled data here.

When learning data contains only some indications without any description or labels, it is up to the coder or to the algorithm to find the structure of the underlying data, to discover hidden patterns, or to determine how to describe the data. This kind of learning data is called **unlabeled data**.

Suppose that we have a number of data points, and we want to classify them into several groups. We may not exactly know what the criteria of classification would be. So, an unsupervised learning algorithm tries to classify the given dataset into a certain number of groups in an optimum way. Unsupervised learning algorithms are extremely powerful tools for analyzing data and for identifying patterns and trends. They are most commonly used for clustering similar input into logical groups. Unsupervised learning algorithms include Kmeans, Random Forests, Hierarchical clustering and so on. If some learning samples are labeled, but some other are not labeled, then it is semi-supervised learning. It makes use of a large amount of **unlabeled data for training** and a small amount of **labeled data for testing**. Semi-supervised learning is applied in cases where it is expensive to acquire a fully labeled dataset while more practical to label a small subset. For example, it often requires skilled experts to label certain remote sensing images, and lots of field experiments to locate oil at a particular location, while acquiring unlabeled data is relatively easy. **Reinforcement Learning**

Here learning data gives feedback so that the system adjusts to dynamic conditions in order to achieve a certain objective. The system evaluates its performance based on the feedback responses and reacts accordingly. The best known instances include self-driving cars and chess master algorithm AlphaGo.

Purpose of Machine Learning

Machine learning can be seen as a branch of AI or Artificial Intelligence, since, the ability to change experience into expertise or to detect patterns in complex data is a mark of human or animal intelligence.

As a field of science, machine learning shares common concepts with other disciplines such as statistics, information theory, game theory, and optimization.

As a subfield of information technology, its objective is to program machines so that they will learn.

Python Machine Learning – Environment Setup

Similarly, we can download and install necessary libraries like numpy, matplotlib etc. individually using installers like pip. For this purpose, you can use the commands shown here:

```
pip install numpy

pip install matplotlib

pip install pandas

pip install seaborn

pip install tensorflow

pip install opencv python
```

4.6 Configuration

4.6.1 H/W System Configuration:

Processor	Dual Core.
Speed	1.1 G Hz.
RAM	8 GB (min).
Hard Disk	20 GB.

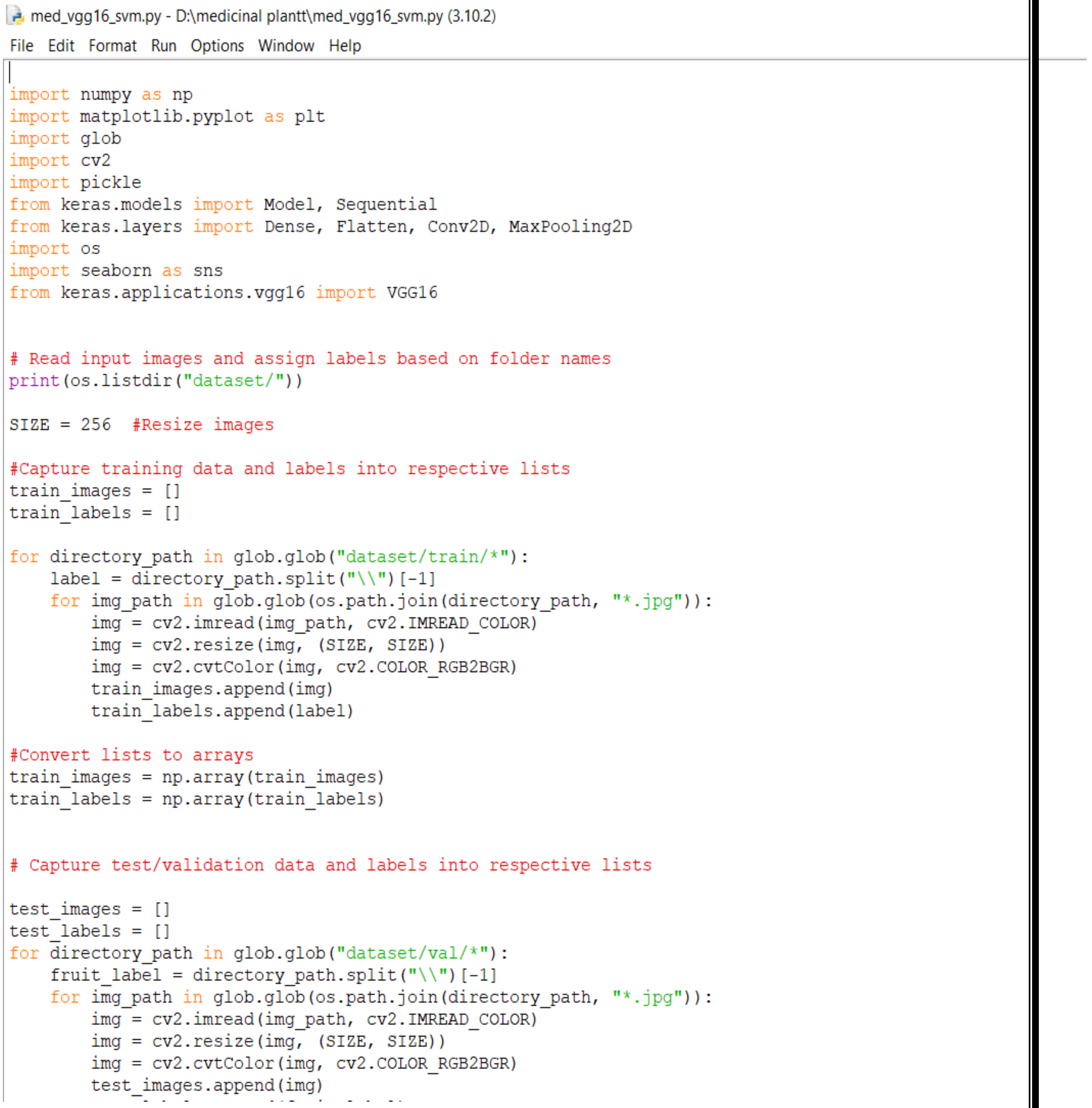
Table 2: Hardware Configuration

4.6.2 S/W System Configuration:

Operating System	Windows 10.
Technology	Machine Learning and AI
Front End	GUI-tkinter
IDLE	Python 3.7 or higher

Table 3: Software Configuration

SOURCE CODE(TRAINING)



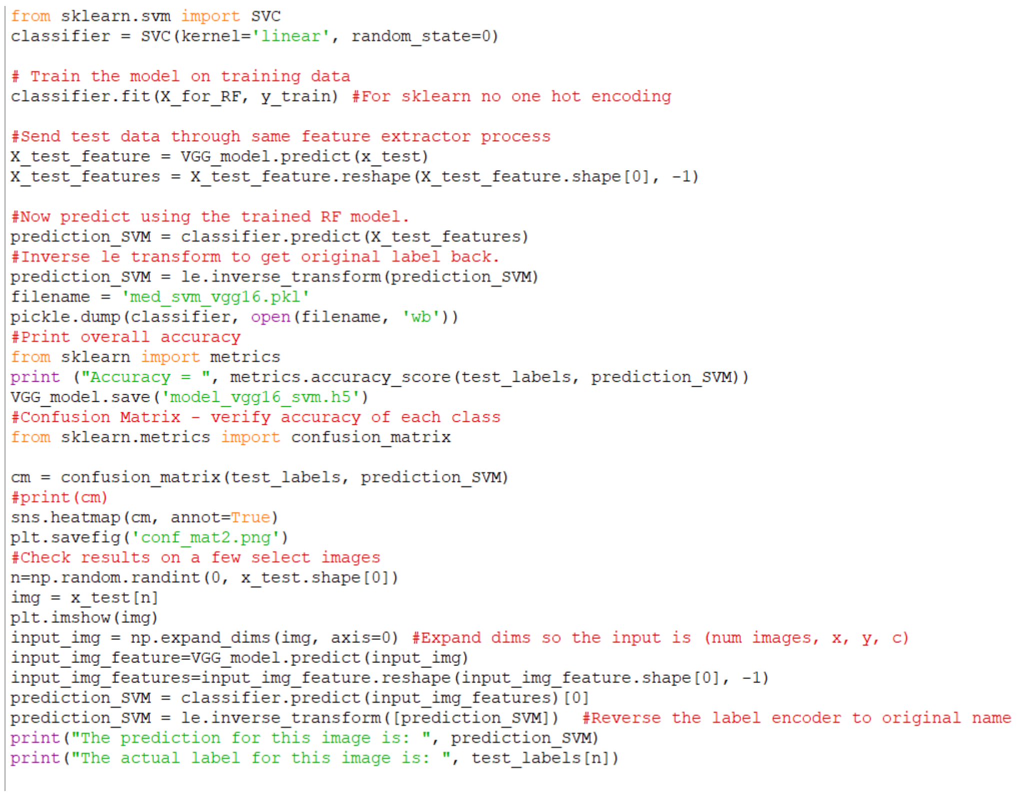
med_vgg16_svm.py - D:\medicinal plantt\med_vgg16_svm.py (3.10.2)

File Edit Format Run Options Window Help

```
import numpy as np
import matplotlib.pyplot as plt
import glob
import cv2
import pickle
from keras.models import Model, Sequential
from keras.layers import Dense, Flatten, Conv2D, MaxPooling2D
import os
import seaborn as sns
from keras.applications.vgg16 import VGG16


# Read input images and assign labels based on folder names
print(os.listdir("dataset/"))

SIZE = 256  #Resize images

#Capture training data and labels into respective lists
train_images = []
train_labels = []

for directory_path in glob.glob("dataset/train/*"):
    label = directory_path.split("\\")[-1]
    for img_path in glob.glob(os.path.join(directory_path, "*.jpg")):
        img = cv2.imread(img_path, cv2.IMREAD_COLOR)
        img = cv2.resize(img, (SIZE, SIZE))
        img = cv2.cvtColor(img, cv2.COLOR_RGB2BGR)
        train_images.append(img)
        train_labels.append(label)

#Convert lists to arrays
train_images = np.array(train_images)
train_labels = np.array(train_labels)


# Capture test/validation data and labels into respective lists

test_images = []
test_labels = []
for directory_path in glob.glob("dataset/val/*"):
    fruit_label = directory_path.split("\\")[-1]
    for img_path in glob.glob(os.path.join(directory_path, "*.jpg")):
        img = cv2.imread(img_path, cv2.IMREAD_COLOR)
        img = cv2.resize(img, (SIZE, SIZE))
        img = cv2.cvtColor(img, cv2.COLOR_RGB2BGR)
        test_images.append(img)
```

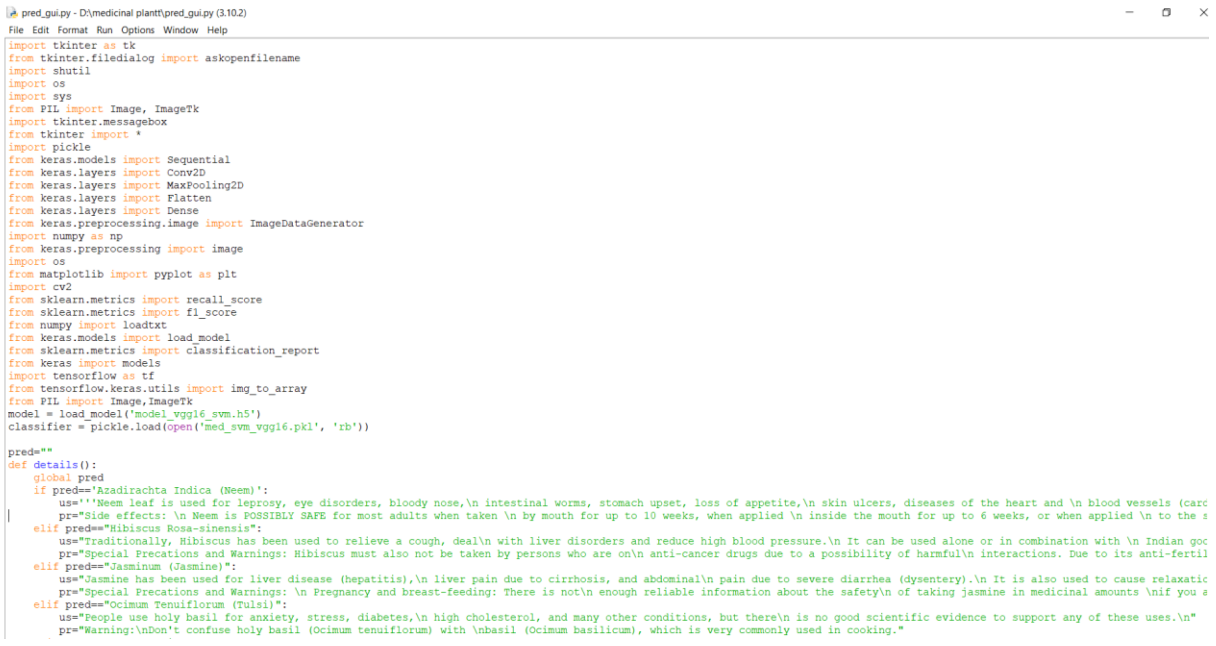
Fig 9: Source code of Training

```
        test_images.append(img)
        test_labels.append(fruit_label)

#Convert lists to arrays
test_images = np.array(test_images)
test_labels = np.array(test_labels)

#Encode labels from text to integers.
from sklearn import preprocessing
le = preprocessing.LabelEncoder()
le.fit(test_labels)
test_labels_encoded = le.transform(test_labels)
le.fit(train_labels)
train_labels_encoded = le.transform(train_labels)

#Split data into test and train datasets (already split but assigning to meaningful convention)
x_train, y_train, x_test, y_test = train_images, train_labels_encoded, test_images, test_labels_encoded

###################################################################
# Normalize pixel values to between 0 and 1
x_train, x_test = x_train / 255.0, x_test / 255.0

#One hot encode y values for neural network.
from tensorflow.keras.utils import to_categorical
y_train_one_hot = to_categorical(y_train)
y_test_one_hot = to_categorical(y_test)

#############################
#Load model wothout classifier/fully connected layers
VGG_model = VGG16(weights='imagenet', include_top=False, input_shape=(SIZE, SIZE, 3))

#Make loaded layers as non-trainable. This is important as we want to work with pre-trained weights
for layer in VGG_model.layers:
        layer.trainable = False

VGG_model.summary()  #Trainable parameters will be 0


#Now, let us use features from convolutional network for RF
feature_extractor=VGG_model.predict(x_train)

features = feature_extractor.reshape(feature_extractor.shape[0], -1)

X_for_RF = features #This is our X input to RF

# "Support vector classifier"
from sklearn.svm import SVC
```



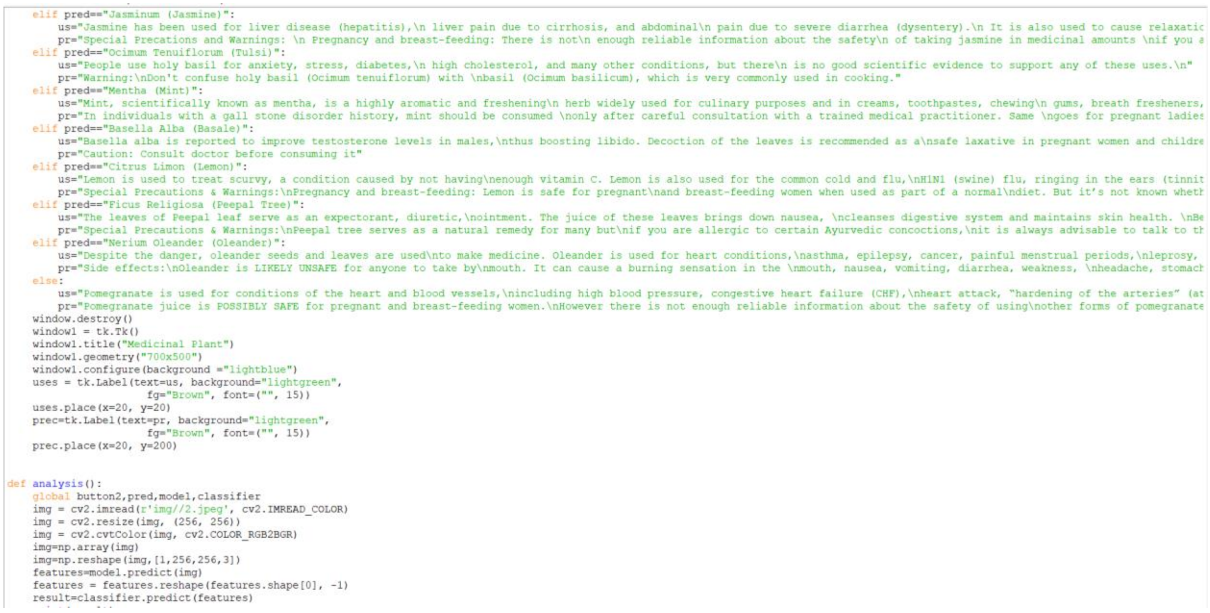
```
from sklearn.svm import SVC
classifier = SVC(kernel='linear', random_state=0)

# Train the model on training data
classifier.fit(X_for_RF, y_train) #For sklearn no one hot encoding

#Send test data through same feature extractor process
X_test_feature = VGG_model.predict(x_test)
X_test_features = X_test_feature.reshape(X_test_feature.shape[0], -1)

#Now predict using the trained RF model.
prediction_SVM = classifier.predict(X_test_features)
#Inverse le transform to get original label back.
prediction_SVM = le.inverse_transform(prediction_SVM)
filename = 'med_svm_vgg16.pkl'
pickle.dump(classifier, open(filename, 'wb'))
#Print overall accuracy
from sklearn import metrics
print ("Accuracy = ", metrics.accuracy_score(test_labels, prediction_SVM))
VGG_model.save('model_vgg16_svm.h5')
#Confusion Matrix - verify accuracy of each class
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(test_labels, prediction_SVM)
#print(cm)
sns.heatmap(cm, annot=True)
plt.savefig('conf_mat2.png')
#Check results on a few select images
n=np.random.randint(0, x_test.shape[0])
img = x_test[n]
plt.imshow(img)
input_img = np.expand_dims(img, axis=0) #Expand dims so the input is (num images, x, y, c)
input_img_feature=VGG_model.predict(input_img)
input_img_features=input_img_feature.reshape(input_img_feature.shape[0], -1)
prediction_SVM = classifier.predict(input_img_features)[0]
prediction_SVM = le.inverse_transform([prediction_SVM])  #Reverse the label encoder to original name
print("The prediction for this image is: ", prediction_SVM)
print("The actual label for this image is: ", test_labels[n])
```

SOURCE CODE(TESTING)

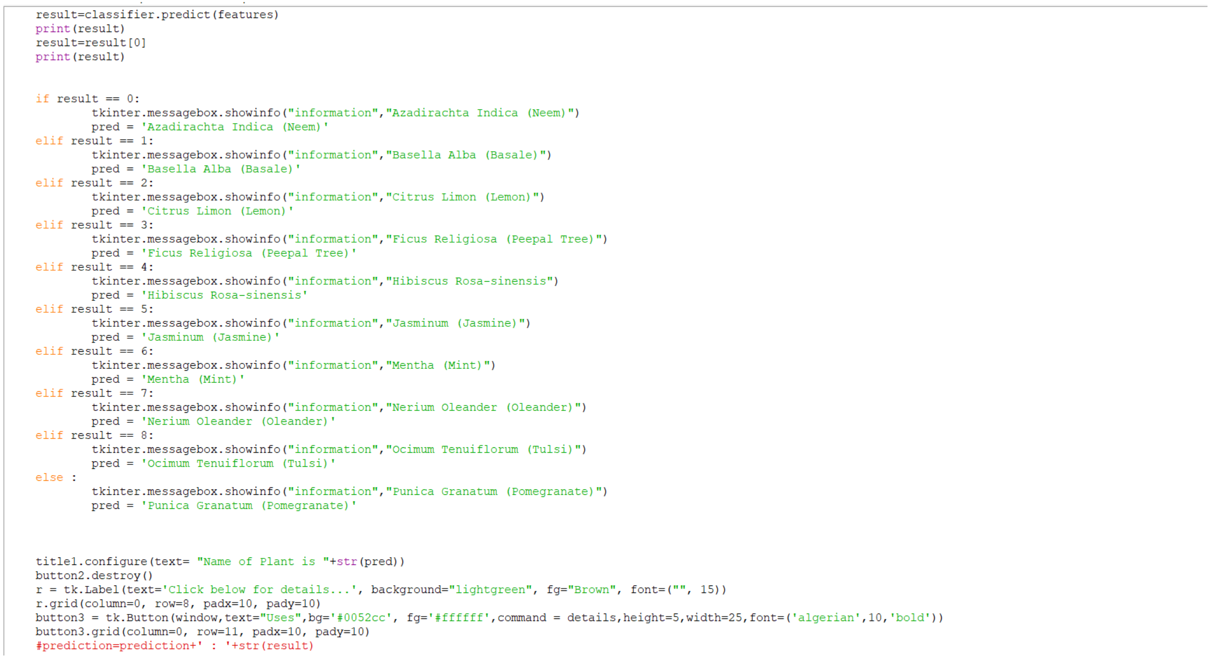

pred_gui.py - D:\medicinal plantt\pred_gui.py (3.10.2)
File Edit Format Run Options Window Help

```
import tkinter as tk
from tkinter.filedialog import askopenfilename
import shutil
import os
import sys
from PIL import Image, ImageTk
import tkinter.messagebox
from tkinter import *
import pickle
from keras.models import Sequential
from keras.layers import Conv2D
from keras.layers import MaxPooling2D
from keras.layers import Flatten
from keras.layers import Dense
from keras.preprocessing.image import ImageDataGenerator
import numpy as np
from keras.preprocessing import image
import os
from matplotlib import pyplot as plt
import cv2
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
from numpy import loadtxt
from keras.models import load_model
from sklearn.metrics import classification_report
from keras import models
import tensorflow as tf
from tensorflow.keras.utils import img_to_array
from PIL import Image,ImageTk
model = load_model('model_vgg16_svm.h5')
classifier = pickle.load(open('med_svm_vgg16.pkl', 'rb'))

pred=""
def details():
    global pred
    if pred=='Azadirachta Indica (Neem)':
        us='''Neem leaf is used for leprosy, eye disorders, bloody nose,\n intestinal worms, stomach upset, loss of appetite,\n skin ulcers, diseases of the heart and \n blood vessels (card
        pr="Side effects: \n Neem is POSSIBLY SAFE for most adults when taken \n by mouth for up to 10 weeks, when applied \n inside the mouth for up to 6 weeks, or when applied \n to the s
    elif pred=="Hibiscus Rosa-sinensis":
        us="Traditionally, Hibiscus has been used to relieve a cough, deal\n with liver disorders and reduce high blood pressure.\n It can be used alone or in combination with \n Indian goc
        pr="Special Precations and Warnings: Hibiscus must also not be taken by persons who are on\n anti-cancer drugs due to a possibility of harmful\n interactions. Due to its anti-fertil
    elif pred=="Jasminum (Jasmine)":
        us="Jasmine has been used for liver disease (hepatitis),\n liver pain due to cirrhosis, and abdominal\n pain due to severe diarrhea (dysentery).\n It is also used to cause relaxatic
        pr="Special Precations and Warnings: \n Pregnancy and breast-feeding: There is not\n enough reliable information about the safety\n of taking jasmine in medicinal amounts \nif you a
    elif pred=="Ocimum Tenuiflorum (Tulsi)":
        us="People use holy basil for anxiety, stress, diabetes,\n high cholesterol, and many other conditions, but there\n is no good scientific evidence to support any of these uses.\n"
        pr="Warning:\nDon't confuse holy basil (Ocimum tenuiflorum) with \nbasil (Ocimum basilicum), which is very commonly used in cooking."
```

Fig 10: Source code of Testing



```
    elif pred=="Jasminum (Jasmine)":
        us="Jasmine has been used for liver disease (hepatitis),\n liver pain due to cirrhosis, and abdominal\n pain due to severe diarrhea (dysentery).\n It is also used to cause relaxatio
        pr="Special Precations and Warnings: \n Pregnancy and breast-feeding: There is not\n enough reliable information about the safety\n of taking jasmine in medicinal amounts \nif you a
    elif pred=="Ocimum Tenuiflorum (Tulsi)":
        us="People use holy basil for anxiety, stress, diabetes,\n high cholesterol, and many other conditions, but there\n is no good scientific evidence to support any of these uses.\n"
        pr="Warning:\nDon't confuse holy basil (Ocimum tenuiflorum) with \nbasil (Ocimum basilicum), which is very commonly used in cooking."
    elif pred=="Mentha (Mint)":
        us="Mint, scientifically known as mentha, is a highly aromatic and freshening\n herb widely used for culinary purposes and in creams, toothpastes, chewing\n gums, breath fresheners,
        pr="In individuals with a gall stone disorder history, mint should be consumed \nonly after careful consultation with a trained medical practitioner. Same \ngoes for pregnant ladies
    elif pred=="Basella Alba (Basale)":
        us="Basella alba is reported to improve testosterone levels in males,\nthus boosting libido. Decoction of the leaves is recommended as a\nsafe laxative in pregnant women and childre
        pr="Caution: Consult doctor before consuming it"
    elif pred=="Citrus Limon (Lemon)":
        us="Lemon is used to treat scurvy, a condition caused by not having\nenough vitamin C. Lemon is also used for the common cold and flu,\nH1N1 (swine) flu, ringing in the ears (tinnit
        pr="Special Precautions & Warnings:\nPregnancy and breast-feeding: Lemon is safe for pregnant\nand breast-feeding women when used as part of a normal\ndiet. But it’s not known wheth
    elif pred=="Ficus Religiosa (Peepal Tree)":
        us="The leaves of Peepal leaf serve as an expectorant, diuretic,\nointment. The juice of these leaves brings down nausea, \ncleanses digestive system and maintains skin health. \nBe
        pr="Special Precautions & Warnings:\nPeepal tree serves as a natural remedy for many but\nif you are allergic to certain Ayurvedic concoctions,\nit is always advisable to talk to th
    elif pred=="Nerium Oleander (Oleander)":
        us="Despite the danger, oleander seeds and leaves are used\nto make medicine. Oleander is used for heart conditions,\nasthma, epilepsy, cancer, painful menstrual periods,\nleprosy,
        pr="Side effects:\nOleander is LIKELY UNSAFE for anyone to take by\nmouth. It can cause a burning sensation in the \nmouth, nausea, vomiting, diarrhea, weakness, \nheadache, stomach
    else:
        us="Pomegranate is used for conditions of the heart and blood vessels,\nincluding high blood pressure, congestive heart failure (CHF),\nheart attack, “hardening of the arteries” (at
        pr="Pomegranate juice is POSSIBLY SAFE for pregnant and breast-feeding women.\nHowever there is not enough reliable information about the safety of using\nother forms of pomegranate
    window.destroy()
    window1 = tk.Tk()
    window1.title("Medicinal Plant")
    window1.geometry("700x500")
    window1.configure(background ="lightblue")
    uses = tk.Label(text=us, background="lightgreen",
                         fg="Brown", font=("", 15))
    uses.place(x=20, y=20)
    prec=tk.Label(text=pr, background="lightgreen",
                         fg="Brown", font=("", 15))
    prec.place(x=20, y=200)


def analysis():
    global button2,pred,model,classifier
    img = cv2.imread(r'img//2.jpeg', cv2.IMREAD_COLOR)
    img = cv2.resize(img, (256, 256))
    img = cv2.cvtColor(img, cv2.COLOR_RGB2BGR)
    img=np.array(img)
    img=np.reshape(img,[1,256,256,3])
    features=model.predict(img)
    features = features.reshape(features.shape[0], -1)
    result=classifier.predict(features)
```



```
    result=classifier.predict(features)
    print(result)
    result=result[0]
    print(result)


    if result == 0:
            tkinter.messagebox.showinfo("information","Azadirachta Indica (Neem)")
            pred = 'Azadirachta Indica (Neem)'
    elif result == 1:
            tkinter.messagebox.showinfo("information","Basella Alba (Basale)")
            pred = 'Basella Alba (Basale)'
    elif result == 2:
            tkinter.messagebox.showinfo("information","Citrus Limon (Lemon)")
            pred = 'Citrus Limon (Lemon)'
    elif result == 3:
            tkinter.messagebox.showinfo("information","Ficus Religiosa (Peepal Tree)")
            pred = 'Ficus Religiosa (Peepal Tree)'
    elif result == 4:
            tkinter.messagebox.showinfo("information","Hibiscus Rosa-sinensis")
            pred = 'Hibiscus Rosa-sinensis'
    elif result == 5:
            tkinter.messagebox.showinfo("information","Jasminum (Jasmine)")
            pred = 'Jasminum (Jasmine)'
    elif result == 6:
            tkinter.messagebox.showinfo("information","Mentha (Mint)")
            pred = 'Mentha (Mint)'
    elif result == 7:
            tkinter.messagebox.showinfo("information","Nerium Oleander (Oleander)")
            pred = 'Nerium Oleander (Oleander)'
    elif result == 8:
            tkinter.messagebox.showinfo("information","Ocimum Tenuiflorum (Tulsi)")
            pred = 'Ocimum Tenuiflorum (Tulsi)'
    else :
            tkinter.messagebox.showinfo("information","Punica Granatum (Pomegranate)")
            pred = 'Punica Granatum (Pomegranate)'



    title1.configure(text= "Name of Plant is "+str(pred))
    button2.destroy()
    r = tk.Label(text='Click below for details...', background="lightgreen", fg="Brown", font=("", 15))
    r.grid(column=0, row=8, padx=10, pady=10)
    button3 = tk.Button(window,text="Uses",bg='#0052cc', fg='#ffffff',command = details,height=5,width=25,font=('algerian',10,'bold'))
    button3.grid(column=0, row=11, padx=10, pady=10)
    #prediction=prediction+' : '+str(result)
    #prediction=prediction+' : '+str(result)


window = tk.Tk()

window.title("Medicinal Plant")

window.geometry("500x350")
window.configure(background ="lightblue")

title = tk.Label(text = "Upload a plant image", background = "lightblue", fg="Brown", font=("Lucida Grande", 15))
title.grid()
button2 = tk.Button(window,text="Analyse Image",bg='#0052cc', fg='#ffffff',command = analysis,height=5,width=25,font=('algerian',10,'bold'))
var = StringVar(window)
def openphoto():
    global button2,var
    dirPath = "img"
    fileList = os.listdir(dirPath)
    for fileName in fileList:
        os.remove(dirPath + "/" + fileName)
     # C:/Users/Downloads/images is the location of the image which you want to test..... you can change it according to the image location you have
    fileName = askopenfilename(initialdir='test dataset', title='Select image for analysis ',filetypes=[('All files', '.jpg')])
    dst = "img\\2.jpeg"
    shutil.copy(fileName, dst)
    title.destroy()
    button1.destroy()
    button2 = tk.Button(window,text="Analyse Image",bg='#0052cc', fg='#ffffff',command = analysis,height=5,width=25,font=('algerian',10,'bold'))
    button2.grid(column=0, row=500, padx=40, pady = 10)

button1 = tk.Button(text="Click here to Upload", command = openphoto,height=5,width=25,bg='#0052cc', fg='#ffffff',font=('algerian',10,'bold'))
button1.grid(column=0, row=500, padx=40, pady = 10)
title1 = tk.Label(text = "", background = "lightblue", fg="Brown", font=("Lucida Grande", 15))
title1.grid(column=0, row=2, padx=10, pady=10)

window.mainloop()
```

IMPLEMENTATION

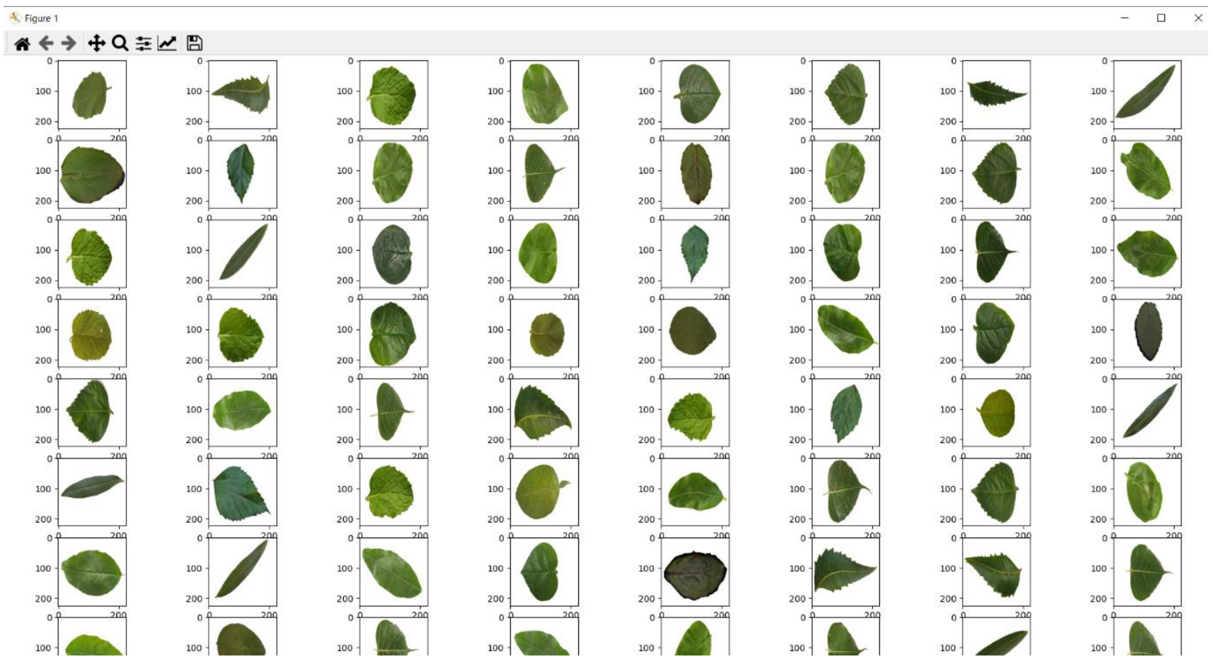


Fig 11:Dataset of leafs

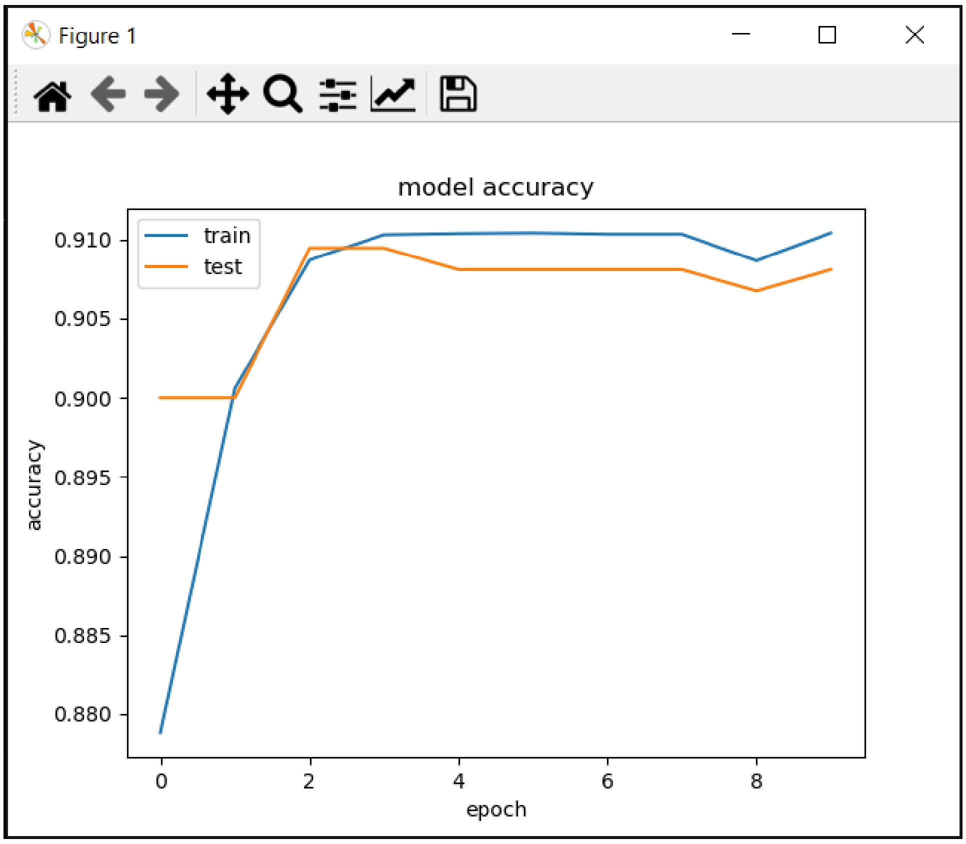


Fig 12: Graph of Model Accuracy

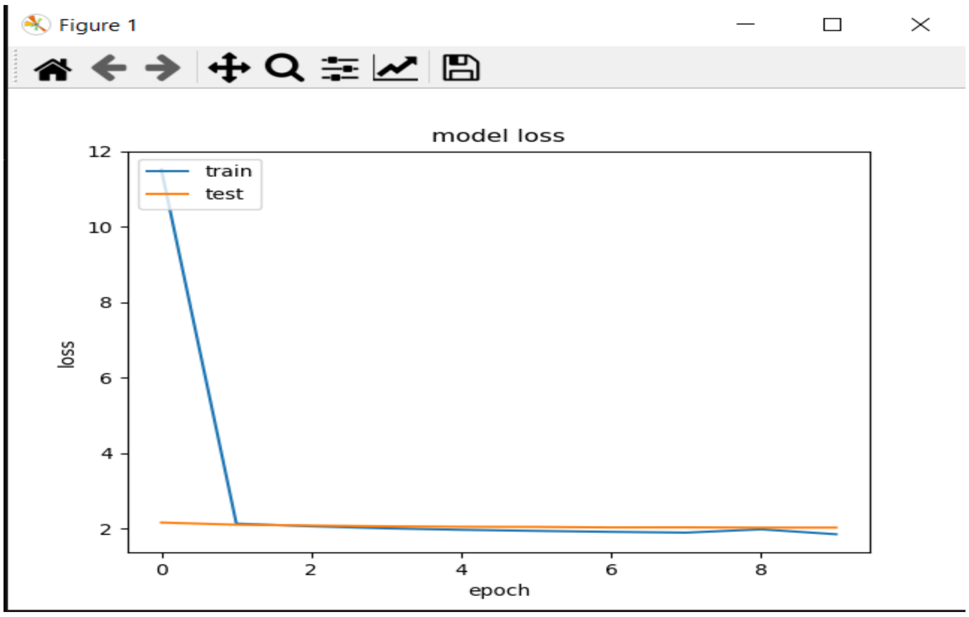
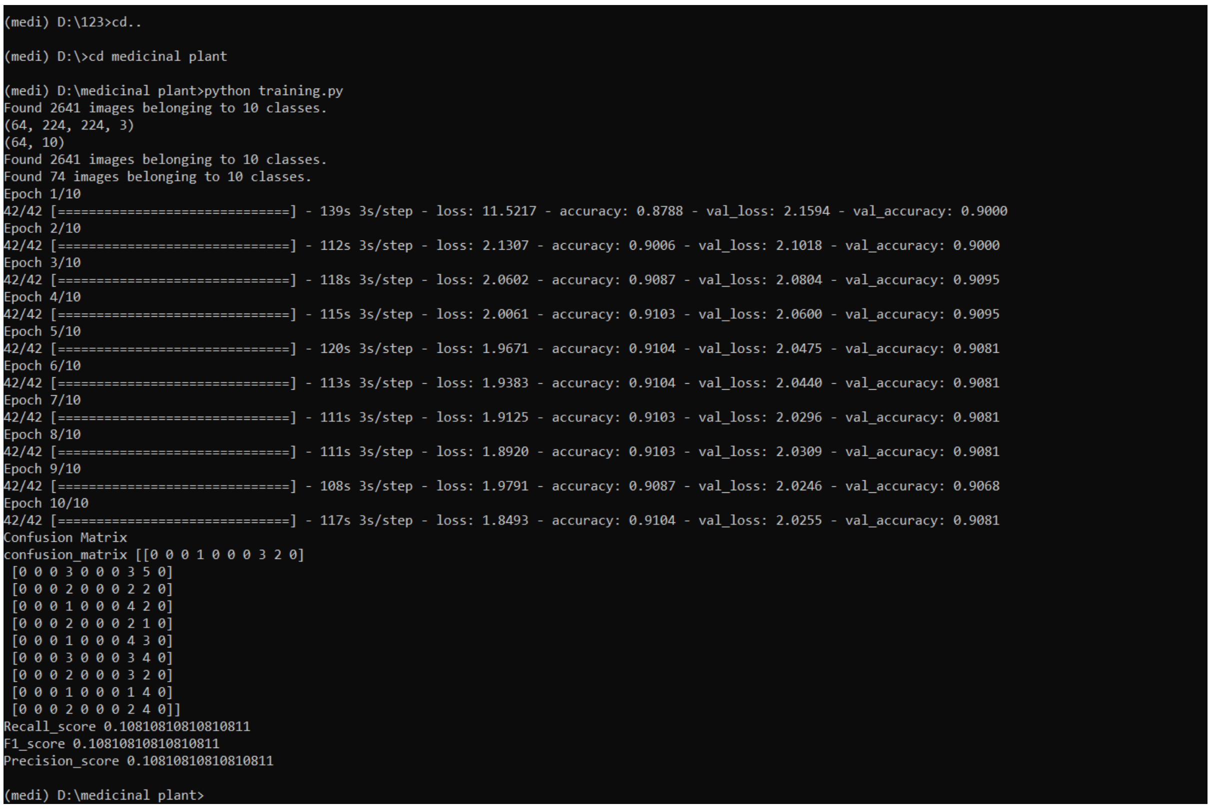


Fig 13: Graph of Loss model



```
(medi) D:\123>cd..

(medi) D:\>cd medicinal plant

(medi) D:\medicinal plant>python training.py
Found 2641 images belonging to 10 classes.
(64, 224, 224, 3)
(64, 10)
Found 2641 images belonging to 10 classes.
Found 74 images belonging to 10 classes.
Epoch 1/10
42/42 [==============================] - 139s 3s/step - loss: 11.5217 - accuracy: 0.8788 - val_loss: 2.1594 - val_accuracy: 0.9000
Epoch 2/10
42/42 [==============================] - 112s 3s/step - loss: 2.1307 - accuracy: 0.9006 - val_loss: 2.1018 - val_accuracy: 0.9000
Epoch 3/10
42/42 [==============================] - 118s 3s/step - loss: 2.0602 - accuracy: 0.9087 - val_loss: 2.0804 - val_accuracy: 0.9095
Epoch 4/10
42/42 [==============================] - 115s 3s/step - loss: 2.0061 - accuracy: 0.9103 - val_loss: 2.0600 - val_accuracy: 0.9095
Epoch 5/10
42/42 [==============================] - 120s 3s/step - loss: 1.9671 - accuracy: 0.9104 - val_loss: 2.0475 - val_accuracy: 0.9081
Epoch 6/10
42/42 [==============================] - 113s 3s/step - loss: 1.9383 - accuracy: 0.9104 - val_loss: 2.0440 - val_accuracy: 0.9081
Epoch 7/10
42/42 [==============================] - 111s 3s/step - loss: 1.9125 - accuracy: 0.9103 - val_loss: 2.0296 - val_accuracy: 0.9081
Epoch 8/10
42/42 [==============================] - 111s 3s/step - loss: 1.8920 - accuracy: 0.9103 - val_loss: 2.0309 - val_accuracy: 0.9081
Epoch 9/10
42/42 [==============================] - 108s 3s/step - loss: 1.9791 - accuracy: 0.9087 - val_loss: 2.0246 - val_accuracy: 0.9068
Epoch 10/10
42/42 [==============================] - 117s 3s/step - loss: 1.8493 - accuracy: 0.9104 - val_loss: 2.0255 - val_accuracy: 0.9081
Confusion Matrix
confusion_matrix [[0 0 0 1 0 0 0 3 2 0]
 [0 0 0 3 0 0 0 3 5 0]
 [0 0 0 2 0 0 0 2 2 0]
 [0 0 0 1 0 0 0 4 2 0]
 [0 0 0 2 0 0 0 2 1 0]
 [0 0 0 1 0 0 0 4 3 0]
 [0 0 0 3 0 0 0 3 4 0]
 [0 0 0 2 0 0 0 3 2 0]
 [0 0 0 1 0 0 0 1 4 0]
 [0 0 0 2 0 0 0 2 4 0]]
Recall_score 0.10810810810810811
F1_score 0.10810810810810811
Precision_score 0.10810810810810811

(medi) D:\medicinal plant>
```

Fig 14: model Trained output with validation Loss & Accuracy

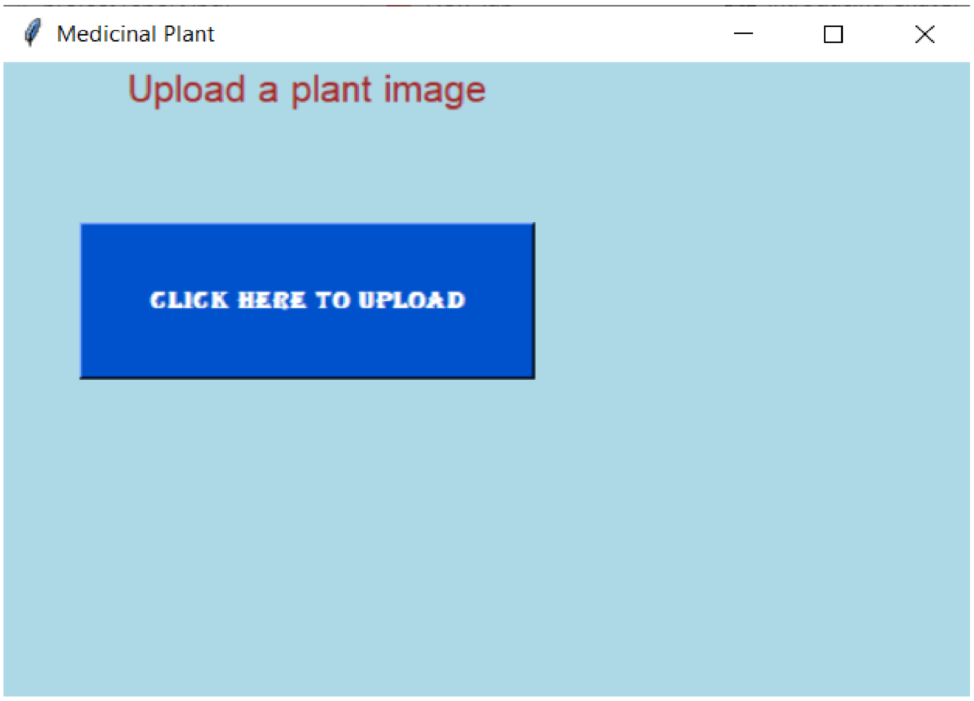
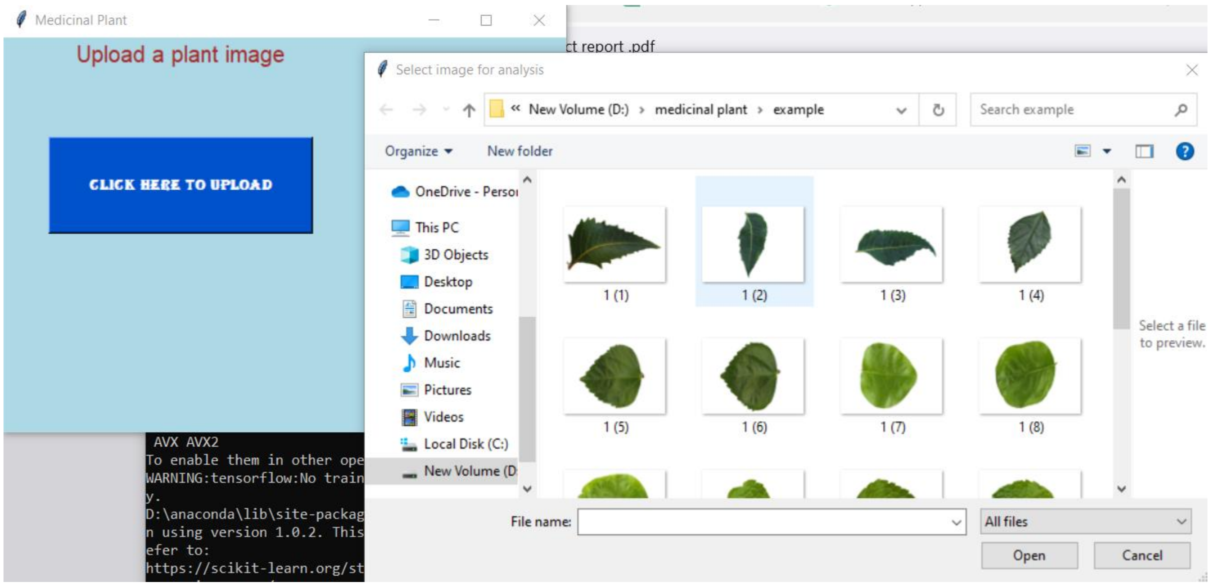
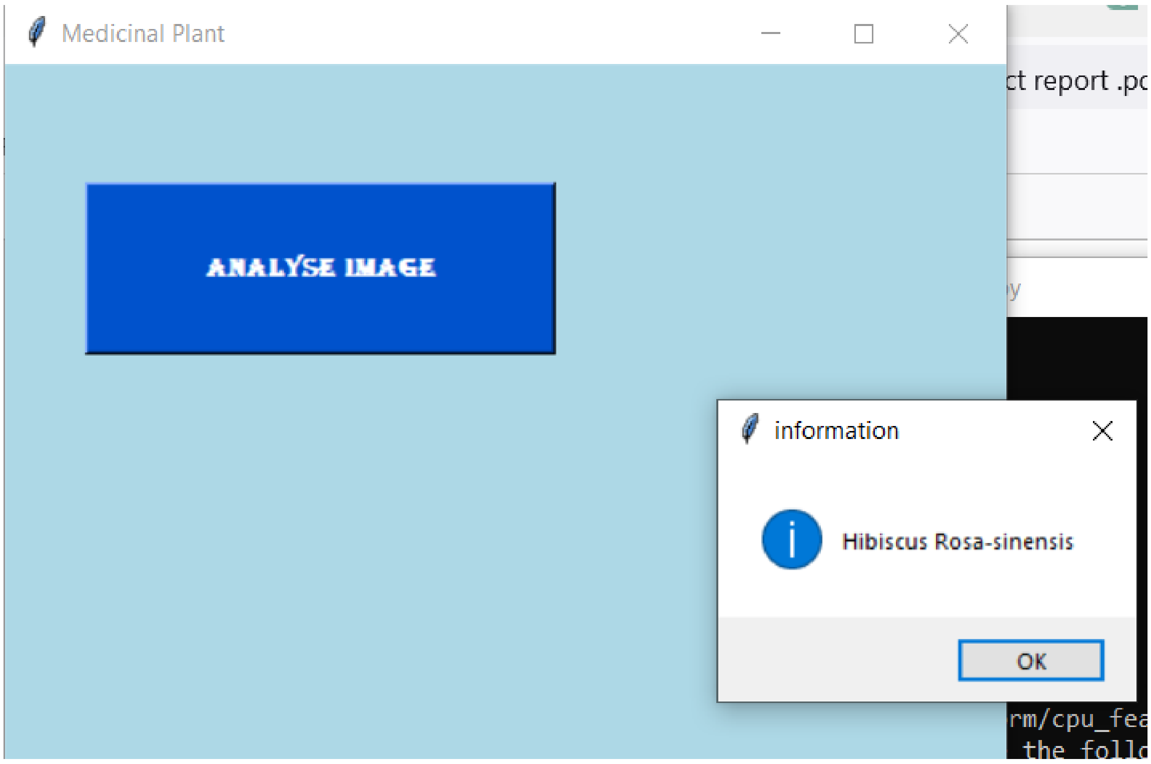


Fig 15: Click here to upload the image



```
 AVX AVX2
To enable them in other ope
WARNING:tensorflow:No train
y.
D:\anaconda\lib\site-packag
n using version 1.0.2. This
efer to:
https://scikit-learn.org/st
```

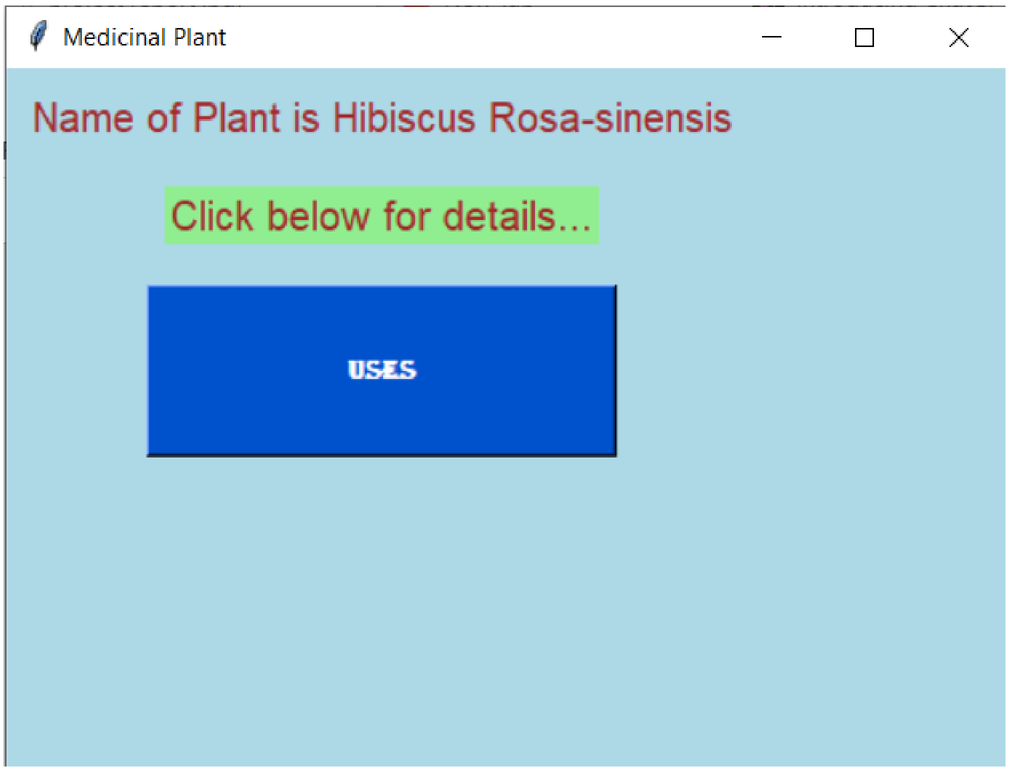
Fig 16: Select & upload the image



Hibiscus Rosa-sinensis

```
rm/cpu_fea
 the follo
```

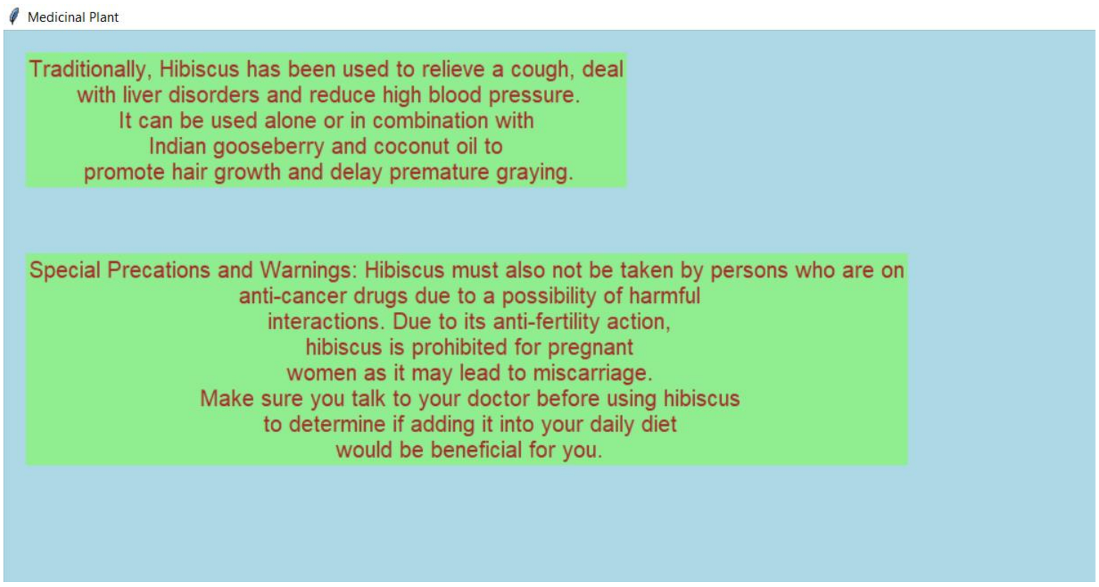
Fig 17: Predicting the Image



Name of Plant is Hibiscus Rosa-sinensis

Click below for details...

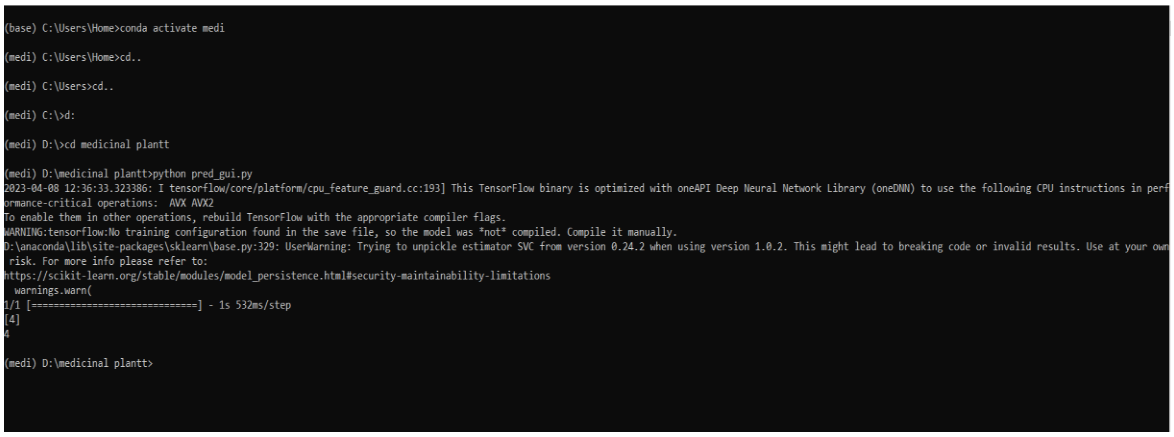
Fig 18 :Name of the predicted Leaf Name



Traditionally, Hibiscus has been used to relieve a cough, deal with liver disorders and reduce high blood pressure. It can be used alone or in combination with Indian gooseberry and coconut oil to promote hair growth and delay premature graying.

Special Precations and Warnings: Hibiscus must also not be taken by persons who are on anti-cancer drugs due to a possibility of harmful interactions. Due to its anti-fertility action, hibiscus is prohibited for pregnant women as it may lead to miscarriage. Make sure you talk to your doctor before using hibiscus to determine if adding it into your daily diet would be beneficial for you.

Fig 19: Properties & Uses of Predicted Leaf



```
(base) C:\Users\Home>conda activate medi

(medi) C:\Users\Home>cd..

(medi) C:\Users>cd..

(medi) C:\>d:

(medi) D:\>cd medicinal plantt

(medi) D:\medicinal plantt>python pred_gui.py
2023-04-08 12:36:33.323386: I tensorflow/core/platform/cpu_feature_guard.cc:193] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in perf
ormance-critical operations:  AVX AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
WARNING:tensorflow:No training configuration found in the save file, so the model was *not* compiled. Compile it manually.
D:\anaconda\lib\site-packages\sklearn\base.py:329: UserWarning: Trying to unpickle estimator SVC from version 0.24.2 when using version 1.0.2. This might lead to breaking code or invalid results. Use at your own
 risk. For more info please refer to:
https://scikit-learn.org/stable/modules/model_persistence.html#security-maintainability-limitations
  warnings.warn(
1/1 [==============================] - 1s 532ms/step
[4]
4

(medi) D:\medicinal plantt>
```

Fig 20**: Result**

CHAPTER 5

TESTING

TESTING:

Testing is a critical element which assures quality and effectiveness of the proposed system in (satisfying) meeting its objectives. Testing is done at various stages in the System designing and implementation process with an objective of developing an transparent, flexible and secured system. Testing is an integral part of software development. Testing process, in a way certifies, whether the product, that is developed, complies with the standards, that it was designed to. Testing process involves building of test cases, against which, the product has to be tested.

5.1 Test objectives

- Testing is a process of executing a program with the intent of finding an error.
- A good case is one that has a high probability of finding an undiscovered error.
- A successful test is one that uncovers a yet undiscovered error. If testing is conducted successfully (according to the objectives) it will uncover errors in the software. Testing can't show the absences of defects are present. It can only show that software defects are present.

Testing principles

Before applying methods to design effective test cases, a software engineer must understand the basic principle that guides software testing. All the tests should be traceable to customer requirements.

5.2 Testing design

Any engineering product can be tested in one of two ways:

5.2.1 White box Testing

This testing is also called as glass box testing. In this testing, by knowing the specified function that a product has been designed to perform test can be conducted that demonstrates each function is fully operation at the same time searching for errors in each function. it is a test case design method that uses the control structure of the procedural design to derive test cases.

5.2.2 Black box Testing

In this testing by knowing the internal operation of a product, tests can be conducted to ensure that "all gears mesh", that is the internal operation performs according to specification and all internal components have been adequately exercised. It fundamentally focuses on the functional requirements of the software.

The steps involved in black box test case design are:

- Graph based testing methods
- Equivalence partitioning
- Boundary value analysis
- Comparison testing

5.2.3 Testing strategies

A software testing strategy provides a road map for the software developer. Testing is a set of activities that can be planned in advanced and conducted systematically. For this reason a template for software testing a set of steps into which we can place specific test case design methods should be defined for software engineering process.

Any software testing strategy should have the following characteristics:

a. Testing begins at the module level and works outward toward the integration of the entire computer based system.
b. Different testing techniques are appropriate at different points in time.
c. The developer of the software and an independent test group conducts testing.
d. Testing and debugging are different activities but debugging must be accommodated in any testing strategy.
e.

5.2.4 Levels of Testing

Testing can be done in different levels of SDLC. They are:

5.2.4.1 Unit Testing

The first level of testing is called unit testing. Unit testing verifies on the smallest unit of software designs-the module. The unit test is always white box oriented. In this, different modules are tested against the specifications produced during design for the modules. Unit testing is essentially for verification of the code produced during the coding phase, and hence the goal is to test the internal logic of the modules. It is typically done by the programmer of the module. Due to its close association with coding, the coding phase is frequently called “coding and unit testing.” The unit test can be conducted in parallel for multiple modules.

5.2.4.2 Integration Testing

The second level of testing is called integration testing. Integration testing is a systematic technique for constructing the program structure while conducting tests to uncover errors associated with interfacing. In this, many tested modules are combined into subsystems, which are then tested. The goal here is to see if all the modules can be integrated properly.

There are three types of integration testing:

- *Top-Down Integration*: Top down integration is an incremental approach to construction of program structures. Modules are integrated by moving downwards throw the control hierarchy beginning with the main control module.
- *Bottom-Up Integration*: Bottom up integration as its name implies, begins Construction and testing with automatic modules.
- *Regression Testing*: In this contest of an integration test strategy, regression testing is the re execution of some subset of test that have already been conducted to ensure that changes have not propagatedunintended side effects.

5.2.4.3Functional test

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

Unit test case 1

Test case ID	Unit test case 1
Description	Train the model
Input	Code in python
Expected output	Model trained with good accuracy
Actual output	Got the expected output
Passed(?)	Yes

Table 4: Testcase 1

UNIT TEST CASE 2

Test case ID	Unit test case 2
Description	Leaf identification
Input	Check if the model can correctly identify the leaf
Expected output	Identify the image what we give in randomly
Actual output	Got the expected output
Passed(?)	yes

Table 5: Testcase 2

UNIT TEST CASE 3

Test case ID	Unit test case 3
Description	Leaf name and it uses
Input	Check if the model can correctly identify the leaf name and uses
Expected output	Give Leaf uses properly
Actual output	Got the expected output
Passed(?)	yes

Table 6: Testcase 3

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

5.2.4.4Validation testing

At the culmination of integration testing, software is completely assembled as a package; interfacing errors have been covered and corrected, and final series of software tests-validating testing may begin. Validation can be defined in many ways, but a simple definition is that validation succeeds when software functions in a manner that can be reasonably expected by customers. Reasonable expectation is defined in the software requirement specification- a document that describes all user visible attributes of the software. The specification contains a section title "validation criteria". Information contained in that section forms the basis for validation testing approach

5.2.4.5Alpha testing

It is virtually impossible for a software developer to forsee how the customer will really use a program. Instructions for use may be misinterpreted; strange combination of data may be regularly used and output that seemed clear to the tester may be unintelligible to a user in field.

When custom software is built for one customer, a series of acceptance tests are conducted to enable the customer to validate all requirements by the end user rather than system developer and acceptable test can range from an informal “test drive” to a planned and systematically executed series of tests. In fact, acceptance testing can be conducted over a period of weeks or months, thereby uncovering cumulative errors that might degrade the system over time. If software is developed as a product to be used by many customers, it is impractical to perform formal acceptance test with each one. Most software product builders use a process called alpha and beta testing to uncover errors that only the end user seems able to find.

A customer conducts the alpha test at the developer’s site. The software is used in a natural setting with the developer “Looking over the shoulder” of the user and recording errors and usage problems. Alpha tests are conducted in controlled environment.

5.2.4.6Beta testing

The beta test is conducted at one or more customer sites by the end user of the software. Unlike alpha testing, the developer is generally not present. Therefore, the beta test is a “live” application of the software in an environment that cannot be controlled by the developer. The customer records all problems that are encountered during beta testing and reports these to the developer at regular intervals. As a result of problems reported during beta test, the software developer makes modification and then prepares for release of the software product to the entire customer base.

5.2.4.7System Testing and Acceptance Testing

System testing is actually a series of different tests whose primary purpose is to fully exercise the computer-based system. Include recovery testing during crashes, security testing for unauthorized user, etc.

Acceptance testing is sometimes performed with realistic data of the client to demonstrate that the software is working satisfactorily. This testing in FDAC focuses on the external behavior of the system.

CHAPTER 6

CONCLUSION

CONCLUSION:-

In this work, we introduced an attention architecture for the task of extracting features for plant classification and identified medicinal plants from leaf images. We collected a dataset of India medicinal plants, preprocessed them, and classified the plant species using a deep learning technique. In our paper, we described the methodology of the architecture. We analyzed the results based on both training and testing sets, and our result is quite impressive, which correctly identifies of leaves. Besides, the results of the proposed architecture are promising and comparable to other existing methodology through in our dataset, we have both single and compound leaf images. In the future, we will focus on exploring the CNN feature extraction and SVM algorithm model for better performance in both single and compound images as well as we will extend our dataset. Finally, we aim people can be able to identify medicinal plants more easily.

REFERENCES

REFERENCES:-

1.R.Janani, A.Gopal. "Identification of Selected Medicinal Plant Leaves Using Image Features and ANN". International Conference on Advanced Electronic Systems (ICAES), 2013.

2.Christenhusz, Maarten J.M, Byng James W. "The number of known plants species in the world and its annual increase", Phytotaxa, [S.l.], 2016.

3. Ghani, A. "Medicinal plants of Bangladesh: Chemical constituents and uses." (1998). [3] Dyk D A V, Meng X. "The Art of Data Augmentation," Journal of Computational & Graphical Statistics 10(1) pp. 1-50, 2001.

4. Pawara, Pornntiwa, Emmanuel Okafor, Lambert Schomaker, and Marco Wiering. "Data augmentation for plant classification." In International Conference on Advanced Concepts for Intelligent Vision Systems, pp. 615-626. Springer, Cham, 2017.

5. Ganschow, Lene, Tom Thiele, Niklas Deckers, and Ralf Reulke. "Classification of Tree Species on the Basis of Tree Bark Texture." International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences-ISPRS Archives 42, no. W13 (2019). 6.

Seeland, Marco, Michael Rzanny, Nedal Alaqraa, Jana Wäldchen, and Patrick Mäder. "Plant species classification using flower images—A comparative study of local feature representations." PloS one 12, no. 2 (2017): e0170629.

7. Anant Bhardwaj, Manpreet Kaur, and Anupam Kumar. "Recognition of plants by Leaf Image using Moment Invariant and Texture Analysis", International Journal of Innovation and Applied Studies, 2013.

8.M. Jogin, Mohana, M. S. Madhulika, G. D. Divya, R. K. Meghana and S. Apoorva, "Feature Extraction using Convolution Neural Networks (CNN) and Deep Learning," 2018 3rd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT), Bangalore, India, pp. 2319-2323, 2018.

9. Herdiyeni, Y. and Wahyuni, N.K.S., “Mobile Application for Indonesian Medicinal Plants Identification using Fuzzy Local Binary Pattern and Fuzzy Color Histogram”. International Conference on Advanced Computer Science and Information Systems (ICACSIS), West Java, Indonesia, 301-306. , 2012.

10. Mohd Shamrie Sainin , Taqiyah Khadijah Ghazali and Rayner Alfred. “Malaysian Medicinal Plant Leaf Shape Identification and Classification”. Knowledge Management International Conference (KMICe) , 12 – 15 August 2014, Malaysia.