**A Project Report On**

FLORA IDENTIFICATION

**A Dissertation submitted to JNTU Hyderabad in partial fulfillment of the**

**academic requirements for the award of the degree**

**Bachelor of Technology**

**in**

## **Computer Science and Engineering**

Submitted by

**B. PRAVEEN KUMAR**

**(20H51A05D8)**

**B. NARESH**

**(20H51A0532)**

**V.VAMSHI**

**(20H51A05M6)**

**Under the esteemed guidance of**

**Dr.P. Senthil**

**(Assistant Professor)**



**Department of Computer Science and Engineering**

**CMR COLLEGE OF ENGINEERING & TECHNOLOGY**

(UGC Autonomous)

\*Approved by AICTE \*Affiliated to JNTUH \*NAAC Accredited with A+ Grade

**KANDLAKOYA,MEDCHAl ROAD, HYDERABAD -501401**.

2020 - 2024

### **CMR COLLEGE OF ENGINEERING & TECHNOLOGY**

KANDLAKOYA, MEDCHAL ROAD, HYDERABAD – 501401

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

## 

## **CERTIFICATE**

This is to certify that the Project report **FLORA IDENTIFICATION** being submitted by B. Praveen kumar (20H51A05D8), B. Naresh (20H51A0532), V. Vamshi (20H51A05M6), in partial fulfillment for the award of **Bachelor of Technology** in Computer Science and Engineering record of bonafide work carried out his/her under my guidance and supervision.

e

The results embody in this project report have not been submit any other University or Institute for the award of any Degree.

**Dr.P. Senthil Dr. Siva Skandha Sanagala**

**Assistant Professor Associate Professor and HOD**

**Dept. of CSE Dept. of CSE**

## **ACKNOWLEDGEMENT**

## 

With great pleasure we want to take this opportunity to express my heartfelt gratitude to all the people who helped in making this project work a grand success.

We are grateful to **Dr.P. Senthil, Assistant professor**, Department of Computer Science and Engineering for his valuable technical suggestions and guidance during the execution of this project work.

We would like to thank **Dr. Siva Skandha Sanagala,** Head of the Department of Computer Science and Engineering, CMR College of Engineering and Technology, who is the major driving forces to complete my project work successfully.

We are very grateful to **Dr. Vijaya Kumar Koppula**, Dean-Academics, CMR College of Engineering and Technology, for his constant support and motivation in carrying out the project work successfully.

We are highly indebted to **Major Dr. V A Narayana,** Principal, CMR College of Engineering and Technology, for giving permission to carry out this project in a successful and fruitful way.

We would like to thank the **Teaching & Non- teaching** staff of Department of Computer Science and Engineering for their co-operation

We express our sincere thanks to **Shri. Ch. Gopal Reddy**, Secretary, CMR Group of Institutions, for his continuous care.

Finally, we extend thanks to our parents who stood behind us at different stages of this Project. We sincerely acknowledge and thank all those who gave support directly and indirectly in completion of this project work.

B. Praveen kumar-20H51A05D8

B. Naresh-20H51A0532

V.Vamshi-20H51A05M6

**TABLE OF CONTENT**

**CHAPTER NO : PAGE NO**

**CHAPTER -1**

1.1 Purpose of the Project 08

1.2 Scope of the Project 09

1.3 Architecture Diagram 09

**CHAPTER-2**

2.SOFTWARE REQUIREMENTS SPECIFICATIONS 11-13

2.1Requirements Specification Document 11

2.1.1Functional Requirements 12

2.1.2Non-Functional Requirements 13

2.2Software Requirements 13

2.3Hardware Requirements 13

**CHAPTER-3**

3.SYSTEM ANALYSIS 15-16

3.1Existing System 15

3.2Proposed System 16

**CHAPTER-4**

4. SYSTEM DESIGN 18-20

4.1 Introduction to UML 18

4.2 UML Diagrams 18-20

4.2.1 Use Case diagram 18

4.2.2 Sequence diagram 19

4.2.3 Class diagram 20

**CHAPTER-5**  21-27

5.1 Pseudo code 22

5.2 Source code 23-27

**CHAPTER-6**

6. SYSTEM ANALYSIS 28-29

6.1 Introduction to Analysis 29

6.2 Learning Techniques 29

6.3 Supervised Machine Learning 29

6.4 Training Data and test data 29

**CHAPTER-7**

7. SCREENSHOTS 30-32

7.1 Interface 31

7.2 Example Results 32

**CONCLUSION**  33-34

**REFERENCES**  35-36

**ABSTRACT**

Plant plays an important role in agricultural, industrial, medicine, environmental and ecological protection. Recently, with global warming, biodiversity loss, rapid urban development and environmental damage, people have been seriously destroying the natural environments, which results in that a large number of plant species constantly dying and even dying out every year. It is essential to protect plant species. The first step of protecting plants is to recognize them and understand what they are and where they come from. But there are a large number of plant species that have been named on Earth, and many are still unknown yet, it is difficult to identify each species. To handle such huge information, develop a quick and efficient classification method has become significant research.

Plant species can be recognized by its leaf, flower, skin, fruit and seed, etc. Relatively speaking, using leaf to recognize plant species is very simple and convenient, and many leaf based plant species recognition methods have been proposed. In this paper, we mainly summarize the existing leaf based plant species identification methods, including plant leaf characteristic, public databases, feature extraction-based methods, subspace learning based methods, sparse representation-based methods, and deep learning-based methods. The aim is to emphasize the importance of plant species identification, train people to know about plant species, and provide guidance and comprehensive study for the beginners in this field, in turn, to treasure and protect plant species.

**CHAPTER-1**

**INTRODUCTION** **1. INTRODUCTION**

**1.1 Purpose of Project**

Plant species recognition has received a great attention from the machine learning and machine vision communities. There are many challenges that need to be overcome in order to have a reliable plant recognition method that could be used in serious industrial applications. The problem of plant species recognition can be seen as two separate problems, based on the goal that we want to achieve. The first problem can be defined as recognizing the presence of a certain plant on an image (retrieval problem), and the second problem is finding a certain type of plant in an image and segmenting it from the image.

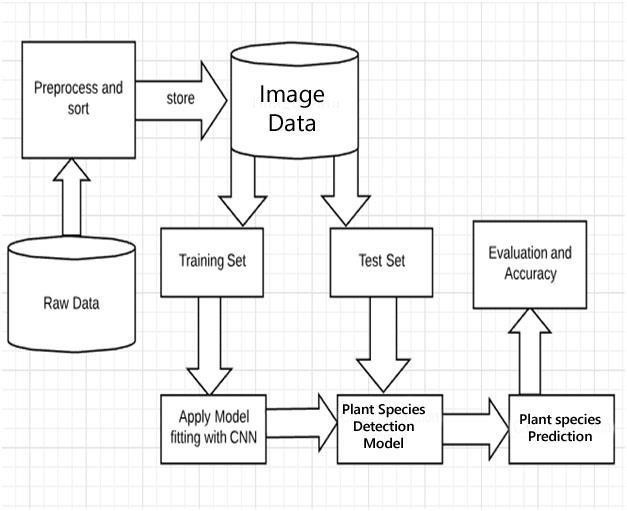
Plant Taxonomy is a science to separate plants into similar groups based on the characteristics like color of the flower, shape of the flower, leaf shape and form, fruits, bark of the stem etc. Plant recognition when done manually by specialized taxonomists, suffers from perceptual biasness, cost of hiring of experts and shortage of experts. The process of manual taxonomy becomes time consuming and tedious when more and more images are added to the database. Automation of the recognition process can improve the time, efficiency, accuracy and cost associated with the recognition process. This has given rise to the demand of automatic tools for plant species recognition and classification.

Major support system for the automation is availability of digitized databases with high resolution plant images annotated with species names and metadata like date and time, lat long information aided by high resolution cameras available on handheld devices. The current approaches for plant species identification utilize only leaf form and shape. The reason being that leaves due to their shape and size, have the advantage to be easily observed, captured and described and are less effected by seasonal changes. Some of the approaches use combination of leaf and bark or color, shape and texture of the flowers to identify the species of the plant. The method uses content as well as the metadata associated with the image like to which part the image is associated and latitude and longitude of the place where image is captured We have used different techniques for different parts of the plant by extracting different features according to the part of the plant.

#### **1.2 Scope of the Project**

This project offers global significance in the field of plant species recognition. It can be used in forests, mountains and dense regions around the world to identify plant species in images greatly reducing the efforts for research work and easily identifying plant species.

#### **1.3 Architecture Diagram**



##### Figure 1.3.1: Architecture Diagram

**CHAPTER 2**

**REQUIREMENTS**

**2. SYSTEM REQUIREMENT SPECIFICATIONS**

**Problem/Requirement Analysis:**

The process is order and more nebulous of the two, deals with understand the problem, the goal and constraints.

**Requirement Specification:**

Here, the focus is on specifying what has been found giving analysis such as representation, specification languages and tools, and checking the specifications are addressed during this activity.

The Requirement phase terminates with the production of the validate SRS document. Producing the SRS document is the basic goal of this phase.

**2.1 Requirements Specification Document**

A Software Requirements Specification (SRS) is a document that describes the nature of a project, software or application. In simple words, SRS document is a manual of a project provided it is prepared before you kick-start a project/application. This document is also known by the names SRS report, software document. A software document is primarily prepared for a project, software or any kind of application.

There are a set of guidelines to be followed while preparing the software requirement specification document. This includes the purpose, scope, functional and nonfunctional requirements, software and hardware requirements of the project. In addition to this, it also contains the information about environmental conditions required, safety and security requirements, software quality attributes of the project etc.

The purpose of SRS (Software Requirement Specification) document is to describe the external behaviour of the application developed or software. It defines the operations, performance and interfaces and quality assurance requirement of the application or software. The complete software requirements for the system are captured by the SRS.

**2.1.1 Functional Requirements**

For documenting the functional requirements, the set of functionalities supported by the system are to be specified. A function can be specified by identifying the state at which data is to be input to the system, its input data domain, the output domain, and the type of processing to be carried on the input data to obtain the output data.

Functional requirements define specific behaviour or function of the application. Following are the functional requirements:

The input design is the link between the information system and user. It compromises the developing specification and procedures for data preparation and those steps are necessary to put transaction data into a usable form for processing can be achieved by inspecting the computer to read data from a written or printed documented or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy.

**2.1.1.1.** Input design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management correct information from the computerized system.

**2.1.1.2.** It is achieved by creating user-friendly screens for data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulations can be performed.

It also provides viewing facilities.

#### **2.1.2 Non-Functional Requirements**

A non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviour. Especially these are the constraints the system must work within. Following are the non-functional requirements:

**Performance:**

The performance of the developed applications can be calculated by using following methods:

Measuring enables you to identify how the performance of your application stands in relation to your defined performance goals and helps you to identify the bottlenecks that affect your application performance. It helps you identify whether your application is moving toward or away from your performance goals. Defining what you will measure, that is, your metrics, and defining the objectives for each metric is a critical part of your testing plan.

Performance objectives include the following:

* Response time or latency

* Throughput

* Resource utilization

### **2.2 Software Requirements**

* Jupyter Notebook
* Internet Browser

### **2.3 Hardware Requirements**

* Processor - Intel i5 (Min 2.4 GHz)
* RAM-8GB
* Disk Space - 10 GB • GPU - Nvidia 640Mx or Better

## **CHAPTER-3**

## **SYSTEM**

## **ANAYLSIS**

### **3. SYSTEM ANALYSIS**

#### **3.1 EXISTING SYSTEM**

Plant Taxonomy is a science to separate plants into similar groups based on the characteristics like color of the flower, shape of the flower, leaf shape and form, fruits, bark of the stem etc. Plant recognition when done manually by specialized taxonomists, suffers from perceptual biasness, cost of hiring of experts and shortage of experts. The process of manual taxonomy becomes time consuming and tedious when more and more images are added to the database. Automation of the recognition process can improve the time, efficiency, accuracy and cost associated with the recognition process. This has given rise to the demand of automatic tools for plant species recognition and classification. Major support system for the automation is availability of digitized databases with high resolution plant images annotated with species names and metadata like date and time, lat long information aided by high resolution cameras available on handheld devices. The current approaches for plant species identification utilize only leaf form and shape. The reason being that leaves due to their shape and size, have the advantage to be easily observed, captured and described and are less effected by seasonal changes.

**Problems with Existing System**

Plant species recognition based on flower identification remain a challenge in Image processing and Computer Vision community mainly because of their vast existence, complex structure and unpredictable variety of classes in nature. Because of these natural complexities, it is highly undesirable to perform normal segmentation or feature extraction or combining shape, texture and color features which results in moderate accuracy on benchmark datasets. Although some feature extraction techniques combining global and local feature descriptors reaches state of the art accuracy in classifying flowers, still there is a need for a robust and efficient system to automatically identify and recognize flower species at a larger scale in complex environment.

With a large number of images to identify and a lack in manpower, a method to achieve the same in easier ways is one of the leading researches in the world right now.

#### **3.2 PROPOSED SYSTEM**

Instead, the image data set is collected whit different plant names then after that feature extraction is done. Using deep learning we train to get new image and whenever new image is provided, we try finds new species of the plant. Thus, the prediction is done using Machine Learning .

Proposed system features are:

1.User Interface - A UI is created to enable uploading of image(s) in which the plant species are to be detected.

2.Reduced Human Errors - Using a machine drastically reduces chances of errors by human errors.

#### **Technologies Used:**

**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.

**Machine Learning Models Studied:**

**1. Convolution Neural Networks**

A Convolutional Neural Network is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other.

The per-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.

**CHAPTER-4**

**SYSTEM DESIGN**

## **4. SYSTEM DESIGN**

#### **4.1 Introduction to UML**

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.

**4.2 UML Diagrams**

**4.2.1 Use Case Diagram**

To model a system, the most important aspect is to capture the dynamic behaviour. To clarify a bit in details, dynamic behaviour means the behaviour of the system when it is running /operating.

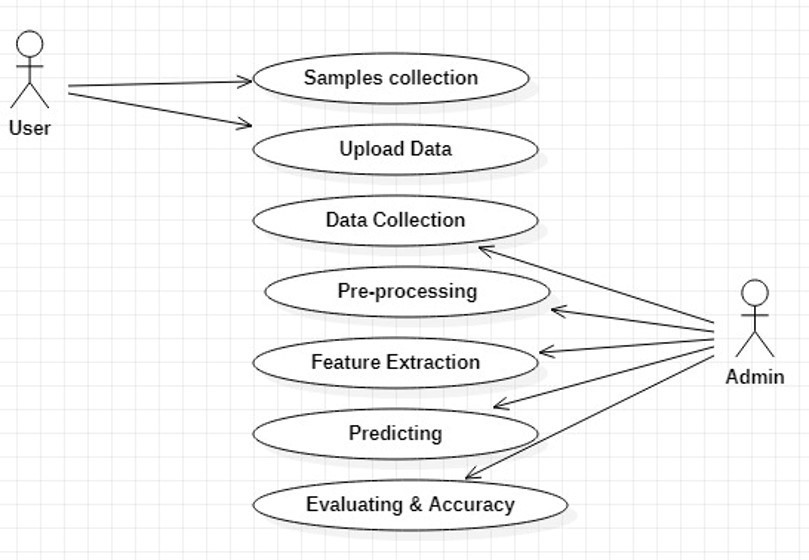


Figure 4.2.1: Use case Diagram for entire application functionality

##### **4.2.2 Sequence Diagram**

Sequence diagrams describe interactions among classes in terms of an exchange of messages over time. They're also called event diagrams. A sequence diagram is a good way to visualize and validate various runtime scenarios. These can help to predict how a system will behave and to discover responsibilities a class may need to have in the process of modeling a new system.

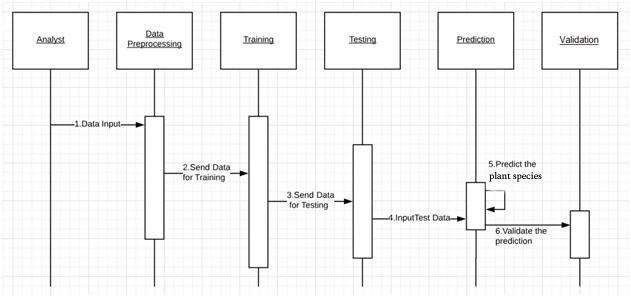


Figure 4.2.2: Sequence Diagram for entire application functionality

##### **4.2.3 Class Diagram**

Class diagrams are the main building blocks of every object oriented methods. The class diagram can be used to show the classes, relationships, interface, association, and collaboration. UML is standardized in class diagrams. Since classes are the building block of an application that is based on OOPs, so as the class diagram has appropriate structure to represent the classes, inheritance, relationships, and everything that OOPs have in its context. It describes various kinds of objects and the static relationship in between them.

The main purpose to use class diagrams are:

* This is the only UML which can appropriately depict various aspects of OOPs concept.
* Proper design and analysis of application can be faster and efficient.
* It is base for deployment and component diagram.

Each class is represented by a rectangle having a subdivision of three compartments name, attributes and operation.

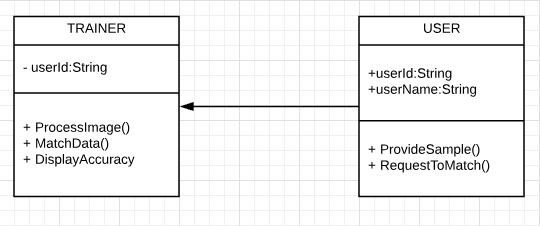


Figure 4.2.3: Class Diagram for entire application functionality

## **CHAPTER 5**

## **IMPLEMENTATION**

**5. IMPLEMENTATION**

#### **5.1 Pseudo code**

Step 1: Select the functionality for which the script must be written.

Step 2: Import the necessary packages.

Step 3: Pre-process the data

Use LabelEncoder to convert the categorical Y\_train, Y\_val sets

Use OneHotEncoder to convert the encoded Y\_train, Y\_val sets to binary format Step 4:

Create the model and train it.

Step 5: Run the console.

Step 6: Provide the input file.

Case 1: Valid Image -> Display result.

Case 2: Invalid file format -> Display Error.

Step 7: Check the result.

Case 1: Image is correctly classified.

Case 2: Image is wrongly classified.

Step 8: Stop.

**SOURCE CODE:**

from django.shortcuts import render,redirect

from django.http import HttpResponse

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

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

from sklearn.feature\_extraction.text import TfidfVectorizer

import itertools

from sklearn.naive\_bayes import MultinomialNB

from sklearn import metrics

from sklearn.linear\_model import PassiveAggressiveClassifier

import os

import seaborn as sns

from sklearn.linear\_model import LogisticRegression

from sklearn.svm import SVC

from sklearn.tree import DecisionTreeClassifier

from sklearn.neighbors import KNeighborsClassifier

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

from sklearn.metrics import confusion\_matrix

# Input data files are available in the "../input/" directory.

# For example, running this (by clicking run or pressing Shift+Enter) will list the files in the input directory

from fakenews.models import User

################ Home #################

def home(request):

    return render(request,'home1.html')

def login(request):

    return render(request,'loginform.html')

def loginCheck(request):

    if request.method == 'POST':

        firstname = request.POST.get('username')

        password = request.POST.get('email')

        request.session['firstname'] = firstname

        print(firstname)

        print(password)

        user\_object=User.objects.get(firstname=firstname,password=password)

        print('--------------------')

        print(user\_object)

        try:

            print('hi')

            user\_object=User.objects.get(firstname=firstname,password=password)

            print(user\_object)

            if user\_object is not None:

                print('hiiiiiiii')

                request.session['useremail'] = user\_object.email

                return redirect('home')

        except:

            #user\_object = None

            print('hello')

            return redirect('login')

    return render(request,'home.html')

def logout(request):

    return render(request,'index.html')

def reg(request):

    return render(request,'register.html')

######## SVM ######

def save(request):

    if request.method == 'POST':

        print('printtttttttttttttttttttttttttttttttt')

        print('checkkkkkkkkkkkkkkkkk')

        username= request.POST.get('username')

        password= request.POST.get('password')

        address= request.POST.get('address')

        email= request.POST.get('email')

        age= request.POST.get('age')

        gender= request.POST.get('gender')

        phone= request.POST.get('phone')

        user=User()

        user.firstname= request.POST.get('username')

        user.password= request.POST.get('password')

        user.address= request.POST.get('address')

        user.email= request.POST.get('email')

        user.age= request.POST.get('age')

        user.gender= request.POST.get('gender')

        user.phone= request.POST.get('phone')

        user.save()

        return render(request,'loginform.html')

    return render(request,'loginform.html')

######## SVM ######

def nvb(request):

    return render(request,'pacweb1.html')

def pac(request):

    return render(request,'result.html')

def svm(request):

    return render(request,'acc1.html')

def dec(request):

    return render(request,'acc1.html')

def randomf(request):

    return render(request,'acc1.html')

def mnb(request):

    return render(request,'acc1.html')

def graph(request):

    if request.method == 'POST' and request.FILES['myfile']:

        # Importing libraries

        import os

        import pandas as pd

        import numpy as np

        import matplotlib.pyplot as plt

        from matplotlib.image import imread

        import seaborn as sns

        import random

        from PIL import Image

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

        from tensorflow.keras.utils import to\_categorical

        import tensorflow as tf

        from tensorflow.keras.models import Sequential, model\_from\_json

        from tensorflow.keras.layers import Dense, Flatten, Dropout, Conv2D, MaxPool2D

        myfile = request.FILES['myfile']

        file1=myfile.name

        ######

        ##########################

        ################################

        import sys,os

        print('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*')

        print(os.path.join(os.getcwd(), 'img', file1))

        print('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*')

        import os

        dirname = os.path.dirname(\_\_file\_\_)

        filename = os.path.join(dirname, myfile.name)

        print(filename)

        global model

        from keras.models import model\_from\_json

        dim1 = []

        dim2 = []

        for i in range(0,3):

            labels = 'C:/Users/91939/OneDrive/Desktop/leaf\_detect/images' + '/{0}'.format(i)

            image\_path = os.listdir(labels)

            for x in image\_path:

                img = imread(labels + '/' + x)

                dim1.append(img.shape[0])

                dim2.append(img.shape[1])

        print("Dimension 1 Mean : ",np.mean(dim1), " Dimension 2 Mean : ",np.mean(dim2))

        images = []

        label\_id = []

        for i in range(3):

            labels = 'images' + '/{0}'.format(i)

            image\_path = os.listdir(labels)

            for x in image\_path:

                img = Image.open(labels + '/' + x)

                img = img.resize((50,50))

                img = np.array(img)

                images.append(img)

                label\_id.append(i)

        images = np.array(images)

        images = images/255

        label\_id = np.array(label\_id)

        label\_id.shape

        images.shape

        label\_counts = pd.DataFrame(label\_id).value\_counts()

        label\_counts.head()

        x\_train, x\_val, y\_train, y\_val = train\_test\_split(images, label\_id , test\_size = 0.2, random\_state = 42)

        y\_train\_cat = to\_categorical(y\_train)

        y\_val\_cat = to\_categorical(y\_val)

        model = Sequential()

        model.add(Conv2D(filters = 64, kernel\_size = (3,3), input\_shape = x\_train.shape[1:], activation = 'relu', padding = 'same'))

        model.add(MaxPool2D(pool\_size=(2,2)))

        model.add(Dropout(0.5))

        model.add(Conv2D(filters = 64, kernel\_size = (3,3), activation = 'relu'))

        model.add(MaxPool2D(pool\_size=(2,2)))

        model.add(Dropout(0.5))

        model.add(Flatten())

        model.add(Dense(128, activation = 'relu'))

        model.add(Dropout(0.5))

        model.add(Dense(4, activation = 'softmax'))

        model.compile(loss = 'sparse\_categorical\_crossentropy', optimizer = 'adam', metrics = ['accuracy'])

        model.summary()

        model.fit(x\_train, y\_train, epochs = 50, batch\_size = 128, validation\_data = (x\_val, y\_val), verbose = 2)

        model\_json = model.to\_json()

        test\_path1 = 'test\_images/'+file1

        img = Image.open(test\_path1)

        img = img.resize((50,50))

        img = np.array(img)

        im2arr = img.reshape(1,50,50,3)

        test = np.asarray(im2arr)

        test = test.astype('float32')

        test = test/255

        y\_pred1 = model.predict\_classes(test);

        print(y\_pred1)

        testimg\_path="test\_images/"+file1

        if y\_pred1==0:

            import cv2

            from imutils import paths

            import imutils

            imagedisplay = cv2.imread(test\_path1)

            orig = imagedisplay.copy()

            output = imutils.resize(orig, width=400)

            cv2.putText(output, "Pepper\_\_bell\_\_\_leaf", (10, 25),    cv2.FONT\_HERSHEY\_SIMPLEX,0.7, (0, 255, 0), 2)

            cv2.imshow("Predicted Image Result", output)

            cv2.waitKey(0)

            return render(request, 'home1.html')

        elif y\_pred1==1:

            import cv2

            from imutils import paths

            import imutils

            imagedisplay = cv2.imread(test\_path1)

            orig = imagedisplay.copy()

            output = imutils.resize(orig, width=400)

            cv2.putText(output, "Potato\_\_\_leaf", (10, 25),   cv2.FONT\_HERSHEY\_SIMPLEX,0.7, (0, 255, 0), 2)

            cv2.imshow("Predicted Image Result", output)

            cv2.waitKey(0)

        elif y\_pred1==2:

            import cv2

            from imutils import paths

            import imutils

            imagedisplay = cv2.imread(test\_path1)

.

**CHAPTER-6**

**ANALYSIS**

### **6. SYSTEM ANALYSIS**

#### **6.1 Introduction to Analysis**

Many of the machine vision systems used in industrial applications employ well known image processing algorithms to discriminate between good and bad parts. Algorithms such as threshold, blob analysis and edge detection, for example, can be found in every machine vision software vendor's toolbox since they can be used in numerous applications to solve a relatively large number of imaging tasks.

To determine whether a part is good or bad, features such as color, length and area are extracted and compared with known good data. In this way, the vision system can, to an acceptable degree, be used to accept good parts and reject bad parts.

**6.2 Learning techniques:**

Image classification may be performed using supervised, unsupervised or semi-supervised learning techniques. In supervised learning, the system is presented with numerous examples of images that must be manually labeled. Using this training data, a learned model is then generated and used to predict the features of unknown images. Such traditional supervised learning techniques can use either generative or discriminative models to perform this task.

#### **6.3 Supervised Machine Learning**

The majority of practical machine learning uses supervised learning.Supervised learning is where the input variables (x) and an output variable (Y) are provided and an algorithm is used to learn the mapping function from the input to the output.

#### **6.4 Training data and test data**

The observations in the training set comprise the experience that the algorithm uses to learn. In supervised learning problems, each observation consists of an observed response variable and one or more observed explanatory variables.

## **CHAPTER-7**

## **SCREENSHORTS**

### **7. SCREENSHOTS**

#### **7.1 Interface**

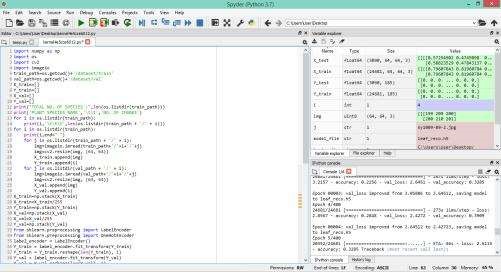
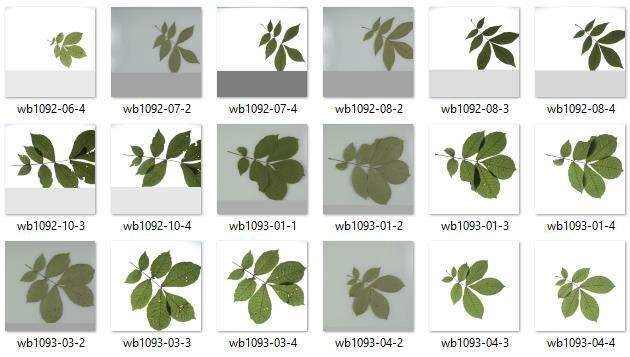
Figure 7.1.1: spyder Interface to run modules 

Figure 7.1.2: Sample image data

#### 7.2 Example Results

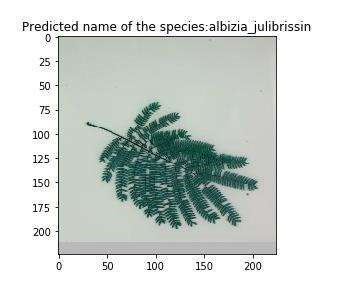
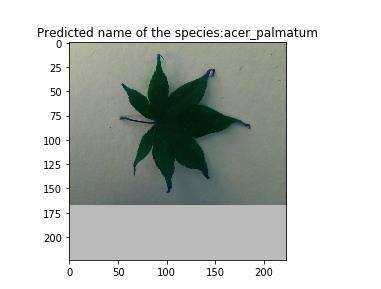


Figure 18: Valid image with correct prediction of species

Figure 7.2.1:Valid image with correct prediction of species

Figure19:Validimage with correct prediction of species

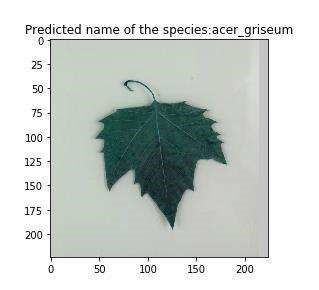


Figure 7.2.2: Valid image with wrong prediction of species

**CONCLUSION**

**CONCLUSION**

Various techniques were used to predict the plant species in a metastatic image, based on visual observation and human experience, these predictions were often not very accurate.

However, in recent years, with the advancement in technology, it has been possible to study metastatic images correctly using machine learning techniques namely Convolution Neural Networks and NasNet.

In our project, we implemented these algorithms to predict the plant species by training the machine using a large dataset of collected metastatic scans and determined the roc curve for the best working algorithm.

**REFERENCES**

**REFERENCES**

1. https://www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networksunderstand-the-basics/
2. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neuralnetworks-the-eli5-way-3bd2b1164a53
3. https://ieeexplore.ieee.org/document/7411268
4. <http://en.m.wikipedia.org/wiki/Plant_taxonomy>
5. <http://www.frontiersin.org/articles/10.3389/fpls.2022.787527/full>