PLANT LEAF IDENTIFICATION

A Project Report submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

Submitted by

B Sarat Teja, (1215316462)

N Tejaswini Yadav, (1215316435)

DVL Sriram, (1215316417)

S Krishna Sai Reddy, (1215316448)

Under the esteemed guidance of

Mr. D. Veera Reddy

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GITAM INSTITUTE OF TECHNOLOGY

GITAM
(Deemed to be University)
VISAKHAPATNAM – 530045

April, 2020

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GITAM INSTITUTE OF TECHNOLOGY

GITAM

VISAKHAPATNAM – 530045

DECLARATION

We, hereby declare that the project review entitled "**PLANT LEAF IDENTIFICATION**" is an original work done in the Department of Computer Science and Engineering, GITAM Institute of Technology, GITAM (Deemed to be University) submitted in partial fulfilment of the requirements for the award of the degree of B.Tech. in Computer Science and Engineering. The work has not been submitted to any other college or University for the award of any degree or diploma.

Date:

Registration No(s)	**Name(s)**
1215316462	**B. Sarat Teja**
1215316435	**N. Tejaswini Yadav**
1215316417	**D.V.L Sriram**
1215316448	**S. Krishna Sai Reddy**

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GITAM INSTITUTE OF TECHNOLOGY

GITAM

(Deemed to be University)
(Estd. u/s 3 of the UGC Act, 1956)

VISAKHAPATNAM – 530045

CERTIFICATE

This is to certify that the project work entitled '**PLANT LEAF IDENTIFICATION**' is a bona-fide work carried out by **B. Sarat Teja(1215316462), N. Tejaswini Yadav(1215316435), D.V.L Sriram(1215316417), S. Krishna Sai Reddy(1215316448**) submitted in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and engineering, GITAM Institute of Technology, GITAM University, Visakhapatnam during the academic year 2019-2020.

Project Guide	**Head of the Department**
Mr. D. Veera Reddy	**Dr. K. Thammi Reddy**
Assistant professor	**Professor**

ACKNOWLEDGEMENTS

The project opportunity was a great platform for us in enhancing our learning and professional development. Therefore, we consider ourselves to be very lucky to have been provided with an opportunity to be a part of it. Bearing in mind previous, we are using this opportunity to express our deepest gratitude and special thanks to our project guide, Asst Prof. D. Veera Reddy sir, who in spite of being extraordinarily busy with her duties, took time out to listen, give us necessary advice and keep us on the correct path with valuable insights that propelled our decision making process whilst allowing us to carry out our work. We choose this moment to acknowledge her contribution as our mentor.

We also express our gratitude to the project reviewers K. Srinivasa Rao, Associate Professor and Y. Jhansi, Assistant Professor, Department of CSE, GITAM (Deemed to be University) for their valuable suggestions and guidance in the completion of our project.

We express our deepest thanks to Dr. K. Thammi Reddy sir, Head of the Department, Computer Science and Engineering, GITAM Institute Technology for giving us a great opportunity to explore our boundaries beyond the classrooms and develop this project. We choose this moment to acknowledge his contribution to gratefully. We would also like to thank Asst Prof. Sandeep Varma sir, A.M.C who helped us a lot in the successful completion of our project.

We perceive this opportunity as a big milestone in our career development. We will strive to use gained skills and knowledge in the best possible way, and continue to work on their improvement, in order to attain our desired career objectives. Hope to continue cooperation with all of you in the future.

Sincerely,

B. Sarat Teja

N. Tejaswini Yadav

D.V.L. Sriram

S. Krishna Sai Reddy

TABLE OF CONTENTS

1.ABSTRACT

This project aims to show the usage of machine learning and it's techniques in providing a solution to the identification of plants through plant leaves on the basis of their shape, colour and texture features using digital image processing technique. Knowing the identification and specific attributes of the various forage plants is the start to a healthy animal diet. Better diets directly lead to improved performance and indirectly to better human health. Understanding what will grow best in an area also benefits conservation and ecological efforts. Identification helps the medicinal uses and the availability of a particular tree's existence in a region. It helps to extract the genes and DNA of species of plants that were extinct. Plant identification is not exclusively the job of botanists and plant ecologists. It is required or useful for large parts of society, from professionals to the general public. But the identification of plants by conventional means is difficult, time consuming and frustrating for novices. This creates a hard-to-overcome hurdle for novices interested in acquiring species knowledge. In recent years, computer science research, especially image processing and pattern recognition techniques have been introduced into plant taxonomy to eventually make up for the deficiency in people's identification abilities. On computer vision approaches for plant species identification, highlight the main research challenges to overcome in providing applicable tools, and conclude with a discussion of open and future research thrusts.

2.INTRODUCTION

Biodiversity is declining steadily throughout the world. The current rate of extinction is largely the result of direct and indirect human activities. Building accurate knowledge of the identity and the geographic distribution of plants is essential for future biodiversity conservation. Therefore, rapid and accurate plant identification is essential for effective study and management of biodiversity. Recently, taxonomists started searching for more efficient methods to meet species identification requirements, such as developing digital image processing and pattern recognition techniques. The rich development and ubiquity of relevant information technologies, such as digital cameras and portable devices, has brought these ideas closer to reality. Digital image processing refers to the use of algorithms and procedures for operations such as image enhancement, image compression, image analysis, mapping, and geo-referencing. The influence and impact of digital images on modern society is tremendous and is considered a critical component in a variety of application areas including pattern recognition, computer vision, industrial automation, and healthcare industries. Image-based methods are considered a promising approach for species identification. A user can take a picture of a plant in the field with the build-in camera of a mobile device and analyze it with an installed recognition application to identify the species or at least to receive a list of possible species if a single match is impossible. By using a computer-aided plant identification system also non-professionals can take part in this process. Therefore, it is not surprising that large numbers of research studies are devoted to automating the plant species identification process.

3.PROBLEM STATEMENT

Build a model, which can identify the plant through the plant leaves of a random plant based on their given features such as shape, colour and texture using digital image processing.

The study aims to show the usage of machine learning and its technique in providing an accurate result of the plant leaf identification for storing the information.

4.LITERATURE REVIEW

Here we have presented the review of our working related area of plant leaf classification and also present the methods used to classify the images.

1. Hang zhang, Paul Yanne, Shangsong Liang proposed Plant Species Classification Using Leaf Shape And Texture which presents the new method to generate the feature space that combines local texture features using wavelet decomposition and co-occurrence matrix statistics and global shape features. A three level 2-D DWT(Discrete Wavelet Transform) of a gray scale image will be employed to decompose the image, then nine statistical features of co occurrence matrix, e.g., contrast, energy, entropy, homogeneity, etc., are computed out of different sub-bands in approximation and detail regions. Combined with geometrical features, e.g., aspect ratio, solidity and seven Hu moments, a feature set for classification is then built for classification support vector machine is used to classify plant species. The prediction accuracy of SVM is 93.8%

2. ArunPriya C, Balasaravanan T., Antony Selvadoss Thanamani proposed An Efficient Leaf Recognition Algorithm for Plant Classification Using Support Vector Machine.The paper presents the application of SVM and on image processing particularly for understanding leaf image features Therefore two techniques have been combined namely; Support Vector Machine (SVM) and KNN. The study shows that SVM obtains a higher percentage of accuracy among K-NN techniques. Color extraction is used to extract the feature. An RGB image is firstly converted into a grayscale image and boundary enhancement. Classifier tested with flavia dataset and a real dataset. The accuracy obtained by the k-NN is 78% In the flavia dataset and in real dataset the accuracy of k-NN is 81.3%. The proposed approach produces very high accuracy and takes very less execution time.

5.SYSTEM REQUIREMENTS

System requirements describe what the system should do, the information services provided for the users. It should include everything the user of the system must need to know regarding what the system does. It can be further categorized as What information the system should explain, The data from the users, The timing and synchronization category of functional requirement is of most importance in hard real time systems that do such things as controlled hardware devices.

1. **Software Requirements**
   - Operating System - Windows
   - Programming Language - Python
2. **Hardware Requirements**
   - System: Intel core i3
   - Hard Disk: 40gb
   - Monitor: 15 inch vga color
   - Mouse: logitech Mouse
   - RAM: 8gb
   - Keyboard: Standard Keyboard

The major non functional requirements of the system are as follows

1. **Performance**
   - Response Times
   - Processing Times
2. **Capacity & Scalability**
   - Throughput
   - Storage

6.OVERVIEW OF TECHNOLOGIES & METHODOLOGIES

1.OVERVIEW OF TECHNOLOGY

PYTHON

Python is an interpreted, Object oriented, High level programming language with dynamic semantics. Its high level built in data structures, combined with dynamic typing and dynamic binding, makes it very attractive for rapid application development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and the code reuse. The python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms and can be freely distributed.

Often, programmers fall in love with python because of the increased productivity it provides. Since there is no compilation step, the edit test debug cycle is incredibly fast. Debugging Python programmes is easy; a bug or bad input will never cause a segmentation fault. Instead when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time and so on. The debugger is written in python itself testifying to pythons introspective power. On the other hand the quickest way to debug a program is to add a few print statements to the source.

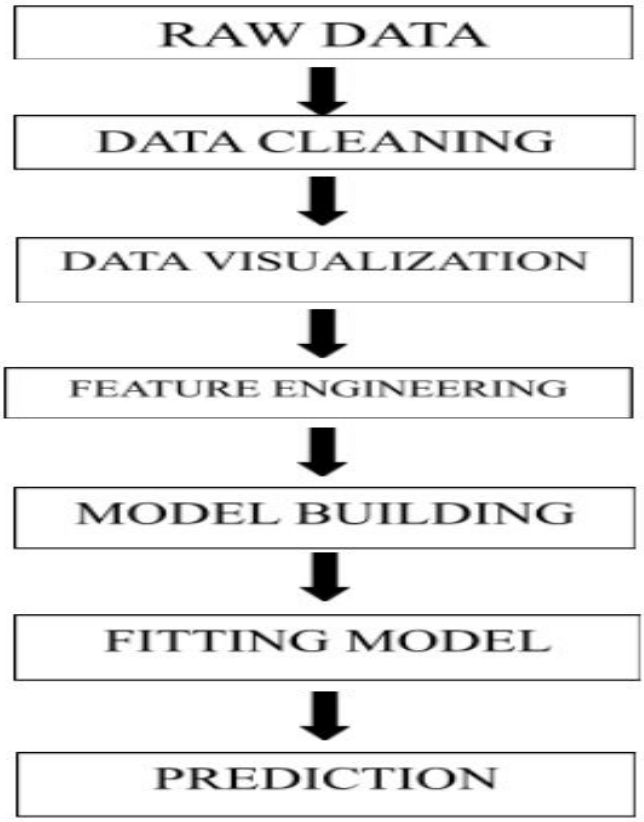
The various technologies used are:

MACHINE LEARNING

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly

programmed. Machine learning focuses on the development of computer programs that can access data and use it to learn for themselves.

ARCHITECTURE OF MACHINE LEARNING

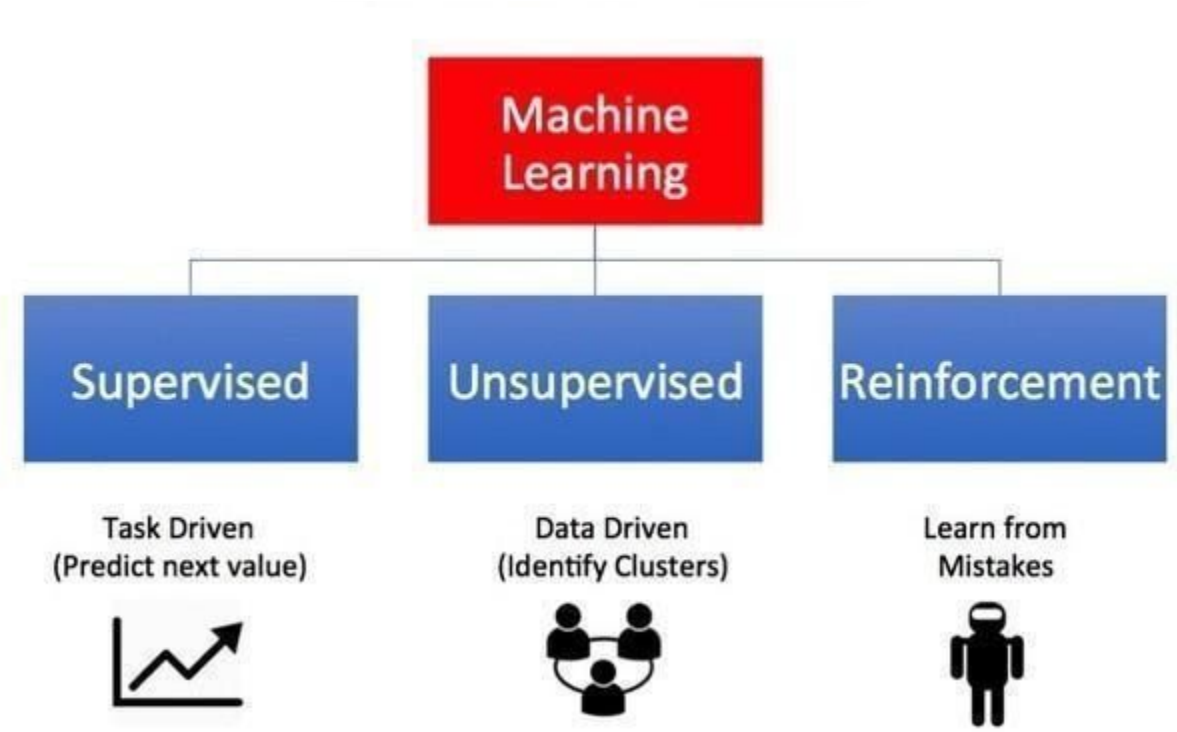


The unstructured data or the raw data contains missing, noisy or non understandable data which is cleaned and removed in the Data Cleaning process resulting in a structured data.

It is then represented by using Data Visualization techniques like scatter plots , graphs , charts etc. The next step is feature engineering which is the process of using domain knowledge of the data to create features that make machine learning algorithms work.

The model is then built by using algorithms like KNN Classifier. Fitting is a measure of how well a machine learning model generalizes to similar data to that on which it was trained. During the fitting process, we run an algorithm on data for which we know the target variable, known as labeled data, and produce a machine learning model. The final prediction is then produced

TYPES OF MACHINE LEARNING



1. Supervised Learning

This algorithm consists of a target / outcome variable (or dependent variable) which is to be predicted from a given set of predictors (independent variables). Using these set of variables,

we generate a function that map inputs to desired outputs. The training process continues until the model achieves a desired level of accuracy on the training data.

Examples of Supervised Learning: Regression, Decision Tree, Random Forest, KNN, Logistic Regression etc.

Real time example: predicting housing price from a given dataset of some or lot of given existing house's price

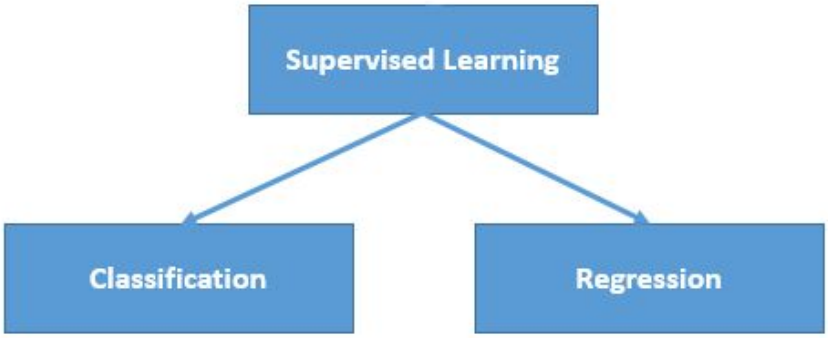
2. Unsupervised Learning

In this algorithm, we do not have any target or outcome variable to predict / estimate. It is used for clustering population in different groups, which is widely used for segmenting customers in different groups for specific intervention. Examples of Unsupervised Learning: Apriori algorithm, K-means. Real time example: predicting or understanding handwritten digits.

3. Reinforcement Learning

Using this algorithm, the machine is trained to make specific decisions. It works this way: the machine is exposed to an environment where it trains itself continually using trial and error. This machine learns from past experience and tries to capture the best possible knowledge to make accurate business decisions. It is a technique to allow an agent to take actions and interact with an environment so as to maximize the total rewards. Example of Reinforcement Learning: Markov Decision Process.

SUPERVISED LEARNING



CLASSIFICATION

Classification algorithms attempt to estimate the mapping function (f) from the input variables (x) to discrete or categorical output variables (y).

In this case, y is a category that the mapping function predicts. If provided with a single or several input variables, a classification model will attempt to predict the value of a single or several conclusions.

For example, when provided with a dataset about houses, a classification algorithm can try to predict whether the prices for the houses "sell more or less than the recommended retail price."

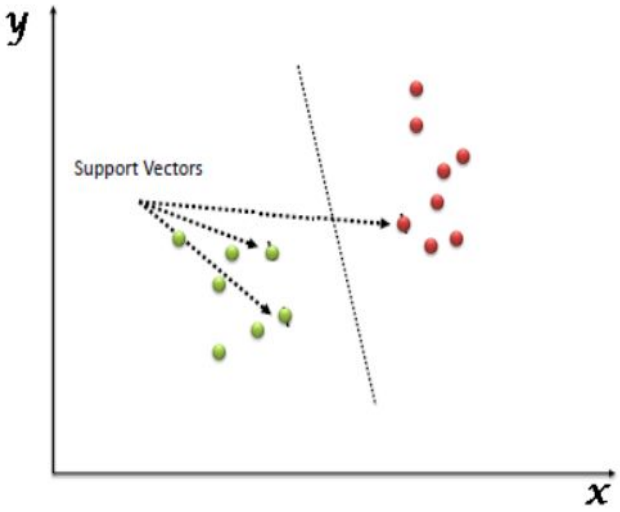
Here, the houses will be classified whether their prices fall into two discrete categories: above or below the said price.

Examples of the common classification algorithms include logistic regression, Naïve Bayes, decision trees, and K Nearest Neighbors.

1. Support Vector Machine(SVM)

"Support Vector Machine" (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems. In the SVM algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiates the two classes very well.

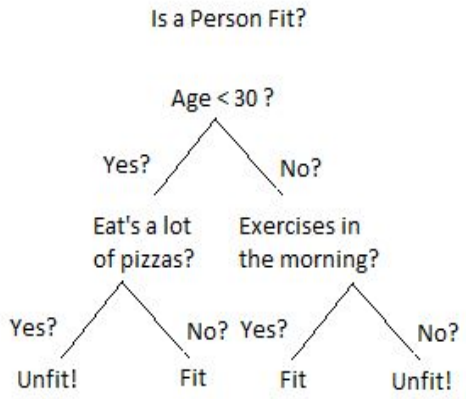
For Example :



2. Decision Tree Classification

Decision trees build classification or regression models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes. It follows the Iterative Dichotomiser 3(ID3) algorithm structure for determining the split.

For Example :

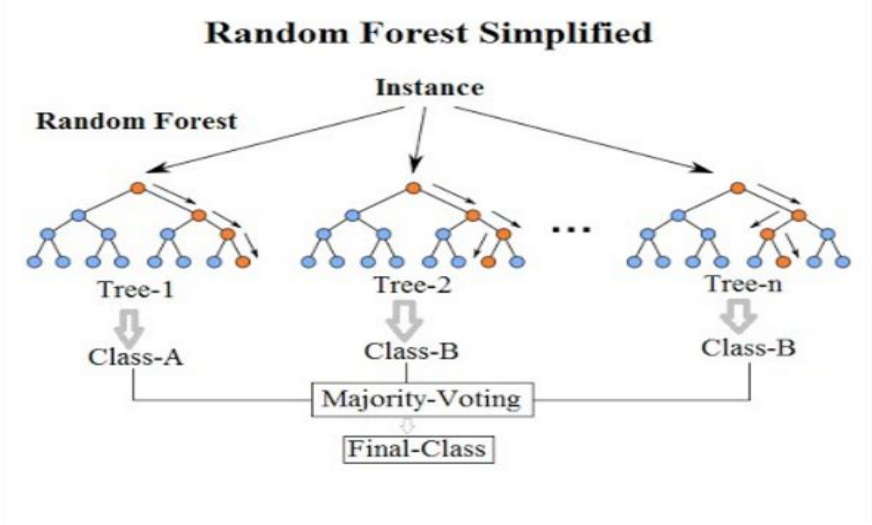


Ensemble Methods for Classification

An ensemble model is a team of models. Technically, ensemble models comprise of several supervised learning models that are individually trained, and the results are merged in various ways to achieve the final prediction. This result has higher predictive power than the results of any of its constituting learning algorithms independently.

Random Forest Classification

Random Forest is a combination of multiple trees. Each tree is provided with some random sample of data with replacement and gives different classifications. It takes votes from the result of all the trees and chooses the classification having maximum votes and when the dependent variable is continuous, it takes the mean from the outputs given by different trees. The number of attributes given to the trees for classification is considered by taking the square root of the total number of attributes. Although each tree works on different attributes. Thus each tree will have a different root node and split. So for the final result, the output of all the trees are considered. The number of trees to be taken can be tuned. One important feature of random forest is that it shows the most important as well as the least important variables in the dataset The random forest algorithm can be used to solve both classification and regression problems. It can handle large dataset with n number of input variables. It automatically takes care of the missing values. Since it is a combination of multiple trees, the accuracy is expected to be much higher than the single decision tree. Random forest is not as good for regression problems as it is for the classification problems.



MACHINE LEARNING WITH PYTHON

Machine learning is a type of artificial intelligence (AI) that provides computers with the ability to learn without being explicitly programmed. Machine learning focuses on the development of Computer Programs that can change when exposed to new data. we will be using numpy, scipy and scikit-learn modules, we can install them by using cmt command.

```
pip install numpy scipy scikit-learn
```

Machine learning involves computers to get trained using a given data set, and use this training to predict the properties of a given new data.

Numpy:

NumPy is the fundamental package for scientific computing with Python. It contains among other things:

- a powerful N-dimensional array object
- sophisticated (broadcasting) functions
- tools for integrating C/C++ and Fortran code
- useful linear algebra, Fourier transform, and random number capabilities

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

```
import numpy as np
```

Pandas:

Pandas are an open-source Python Library providing high-performance data manipulation and analysis tools using its powerful data structures. The name Pandas is derived from the word Panel Data – an Econometrics from Multidimensional data.

```
import pandas as pd
```

OpenCV:

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in commercial products. The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc.

Matplotlib:

Matplotlib is a plotting library for Python. It is used along with Numpy to provide an environment that is an effective open source alternative for MatLab.

```
import matplotlib.pyplot as plt
```

Scikit Learn:

It features various algorithms like support vector machine, random forests, and k-neighbours, and it also supports Python numerical and scientific libraries like NumPy and SciPy. It provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python.

Mahotas:

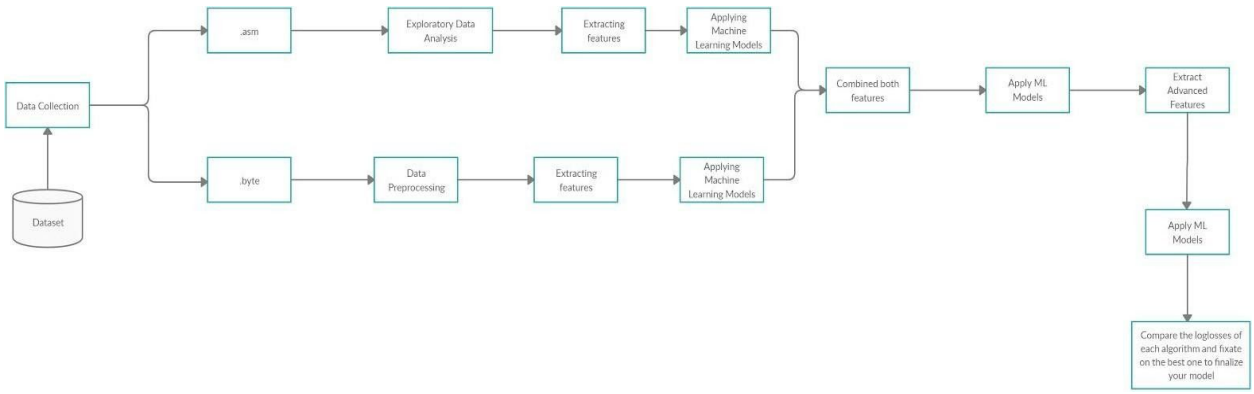
Mahotas is a computer vision and image processing library for Python. It includes many algorithms implemented in C++ for speed while operating in numpy arrays and with a very clean Python interface. Mahotas currently has over 100 functions for image processing and computer vision and it keeps growing.

DATA OVERVIEW

The data set includes figures of plant leaf of about 32 different species of plants on the basis of their leaves using digital image processing technique. The images are first preprocessed and then their shape, color and texture based features are extracted from the processed image

7.IMPLEMENTATION

FLOW CHART



7.1 CODING

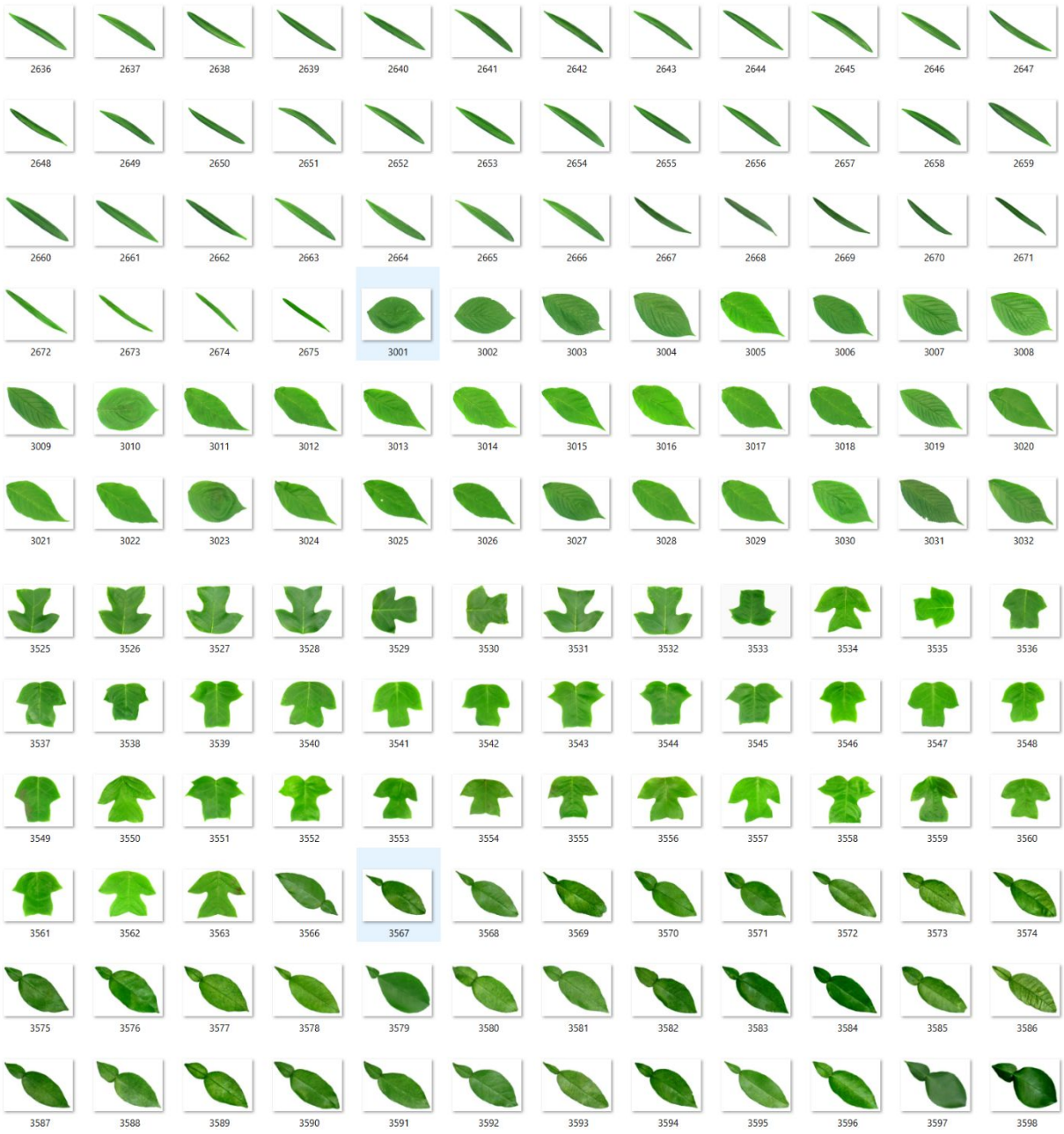
Dataset

The dataset used is the **Flavia leaves dataset** having around 4000 flavia leaves.

.The plant leaf species are 2'Chinese horse chestnut','Anhui Barberry','Chinese redbud','true indigo','Japanese maple','Nanmu',castor aralia','Chinese cinnamon',’goldenrain tree','Big-fruited Holly','Japanese cheesewood',,'wintersweet','camphortree','Japan Arrowwood','sweet osmanthus','deodar','ginkgo’ ,’maidenhair tree', 'Crape myrtle, ’Crepe myrtle','oleander','yew plum pine',’Japanese Flowering Cherry','Glossy Privet','Chinese Toon','peach','Ford Woodlotus','trident maple','Beales barberry','southern magnolia','Canadian poplar','Chinese tulip tree,'tangerine'.With help of the Dataset the features are extracted from the leaf based upon their Shape,Color,Texture Features.The features are stored in CSV(Comma Separated Values) to form a dataset in python use for


1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492


2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512



Project Structure

Preprocess_extract_dataset_flavia.ipynb: contains create_dataset() function which performs image pre-processing and feature extraction on the dataset. The dataset is stored in Flavia_features.csv

Code:

Importing Libraries

Preprocess and Feature Extraction - Flavia dataset

Extracted features are saved in file named "Flavia_features.csv"

In [1]:
```
import os
import cv2
import numpy as np
import pandas as pd
import mahotas as mt
from matplotlib import pyplot as plt
%matplotlib inline
```

Creating Path

In [2]:
```
ds_path = r"C:\Users\sarat\Anaconda3\Plant-Leaf-Identification-master\Flavia leaves dataset"
img_files = os.listdir(ds_path)
```

Creating Datasets

names = ['area','perimeter','physiological_length','physiological_width','aspect_ratio','rectangularity','circularity', \

'mean_r','mean_g','mean_b','stddev_r','stddev_g','stddev_b', \

'contrast','correlation','inverse_difference_moments','entropy'

In [3]:
```
def create_dataset():
    names = ['area','perimeter','physiological_length','physiological_width','aspect_ratio','rectangularity','circularity', \
             'mean_r','mean_g','mean_b','stddev_r','stddev_g','stddev_b', \
             'contrast','correlation','inverse_difference_moments','entropy'
            ]
    df = pd.DataFrame([], columns=names)
    for file in img_files:
        imgpath = ds_path + "\\" + file
        main_img = cv2.imread(imgpath)
```

Preprocessing

```
#Preprocessing
img = cv2.cvtColor(main_img, cv2.COLOR_BGR2RGB)
gs = cv2.cvtColor(img,cv2.COLOR_RGB2GRAY)
blur = cv2.GaussianBlur(gs, (25,25),0)
ret_otsu,im_bw_otsu = cv2.threshold(blur,0,255,cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)
kernel = np.ones((50,50),np.uint8)
closing = cv2.morphologyEx(im_bw_otsu, cv2.MORPH_CLOSE, kernel)
```

Shape based features

contours, image = cv2.findContours(closing,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)

cnt = contours[0]

```
#Shape features
contours, image = cv2.findContours(closing,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)
cnt = contours[0]
M = cv2.moments(cnt)
area = cv2.contourArea(cnt)
perimeter = cv2.arcLength(cnt,True)
x,y,w,h = cv2.boundingRect(cnt)
aspect_ratio = float(w)/h
rectangularity = w*h/area
circularity = ((perimeter)**2)/area
```

Color features

red_channel = img[:,:,0]

green_channel = img[:,:,1

```
        #Color features
        red_channel = img[:,:,0]
        green_channel = img[:,:,1]
        blue_channel = img[:,:,2]
        blue_channel[blue_channel == 255] = 0
        green_channel[green_channel == 255] = 0
        red_channel[red_channel == 255] = 0

        red_mean = np.mean(red_channel)
        green_mean = np.mean(green_channel)
        blue_mean = np.mean(blue_channel)

        red_std = np.std(red_channel)
        green_std = np.std(green_channel)
        blue_std = np.std(blue_channel)
```

Texture features

textures = mt.features.haralick(gs)

ht_mean = textures.mean(axis=0)

```
        #Texture features
        textures = mt.features.haralick(gs)
        ht_mean = textures.mean(axis=0)
        contrast = ht_mean[1]
        correlation = ht_mean[2]
        inverse_diff_moments = ht_mean[4]
        entropy = ht_mean[8]

        vector = [area,perimeter,w,h,aspect_ratio,rectangularity,circularity,\
                  red_mean,green_mean,blue_mean,red_std,green_std,blue_std,\
                  contrast,correlation,inverse_diff_moments,entropy
                 ]

        df_temp = pd.DataFrame([vector],columns=names)
        df = df.append(df_temp)
        print(file)
    return df
```

Creation of DataSet

```
In [*]: dataset = create_dataset()
```

```
1001.jpg
1002.jpg
1003.jpg
1004.jpg
1005.jpg
1006.jpg
1007.jpg
1008.jpg
1009.jpg
1010.jpg
1011.jpg
1012.jpg
1013.jpg
1014.jpg
1015.jpg
1016.jpg
1017.jpg
1018.jpg
1019.jpg
1020.jpg
1021.jpg
1022.jpg
1023.jpg
1024.jpg
1025.jpg
1026.jpg
1027.jpg
1028.jpg
1029.jpg
1030.jpg
1031.jpg
1032.jpg
1033.jpg
1034.jpg
1035.jpg
```

DataSet Shape

```
In [6]: dataset.shape

Out[6]: (1907, 17)
```

Type of DataSet

```
In [7]: type(dataset)

Out[7]: pandas.core.frame.DataFrame
```

Converting into CSV

```
In [8]: dataset.to_csv("Flavia_features.csv")

In [ ]:
```

		area	perimeter	physiologi	physiologi	aspect_rat	rectangula	circularity	mean_r	mean_g	mean_b	stddev_r	stddev_g	stddev_b	contrast	correlatio	inverse_di	entropy
2	0	197484	3479.036	1416	759	1.865613	5.442183	61.28948	6.395667	13.64341	4.388007	24.02533	40.20093	21.44841	12.63969	0.997666	0.911738	1.688689
3	0	101248	2490.382	1190	130	9.153846	1.527931	61.25555	7.049316	9.232018	10.87607	33.81621	37.38222	46.47923	8.137424	0.997191	0.944818	1.193795
4	0	86570.5	2290.683	1095	119	9.201681	1.505189	60.61222	3.434303	6.371511	2.644757	19.9757	29.05737	19.27505	8.553729	0.99661	0.959023	0.848758
5	0	190214	2856.479	1318	254	5.188976	1.759976	42.89629	7.670415	13.3036	6.049157	28.82289	40.22185	26.9486	8.440064	0.998419	0.914331	1.673914
6	0	227727	2917.249	1324	286	4.629371	1.662798	37.3708	8.992028	16.67117	6.294281	30.96716	45.0402	28.59533	8.641447	0.998568	0.898644	1.968081
7	0	233724	3689.81	1434	953	1.504722	5.847076	58.25117	7.31924	15.73062	4.432931	24.81753	42.22132	21.53427	14.56146	0.997708	0.895654	1.957151
8	0	258395	3543.678	1396	874	1.597254	4.721856	48.59866	9.674849	18.43227	6.657215	28.70796	46.43166	24.42757	11.79534	0.998101	0.884639	2.126015
9	0	244401	3732.957	1479	912	1.621711	5.518995	57.01682	9.033226	16.53425	6.947844	29.5701	43.37798	28.11403	13.62784	0.997945	0.889018	2.070195
10	0	223690	3142.317	1404	388	3.618557	2.435299	44.14214	8.594646	16.13478	5.935715	28.13373	43.99787	24.5651	12.59348	0.997737	0.896627	1.891695
11	0	288344.5	3083.27	1329	450	2.953333	2.074082	32.96944	9.552538	19.34629	6.716408	27.72883	45.99022	24.64435	10.30296	0.998595	0.874376	2.286608
12	0	325751.5	3438.225	1371	771	1.77821	3.244931	36.2896	10.79317	23.06702	6.985511	28.37241	50.83093	23.94309	13.0318	0.998273	0.855683	2.586471
13	0	280822	3638.732	1417	914	1.550328	4.611953	47.14861	10.49335	19.79898	8.049866	30.77275	47.62328	28.25007	12.60749	0.998187	0.874752	2.292682
14	0	326785	3338.297	1410	564	2.5	2.433527	34.10262	9.732556	23.08702	5.95837	25.73969	50.9153	21.00212	15.70226	0.997933	0.855815	2.588301
15	0	248368	3399.889	1342	811	1.654747	4.382054	46.54079	8.495395	17.53864	5.821838	26.50284	45.375	23.25148	14.59665	0.997623	0.888203	2.092772
16	0	295805.5	3617.719	1413	899	1.571746	4.294332	44.24493	11.03401	20.44471	8.728894	31.28918	47.78207	29.24721	13.05082	0.998204	0.868841	2.376754
17	0	235298	3670.922	1418	966	1.467909	5.821503	57.27066	7.761884	15.32943	5.705246	26.47336	41.27142	25.07085	13.60631	0.997937	0.894009	1.993244
18	0	209500	3489.369	1399	821	1.704019	5.482477	58.11788	8.131403	15.88859	6.078087	28.83397	44.77871	27.20457	9.484338	0.998218	0.904935	1.833117
19	0	278017	3384.658	1312	841	1.560048	3.968793	41.2058	10.0387	19.93544	7.347536	28.6579	48.06238	25.47056	11.85196	0.998185	0.87716	2.241877
20	0	199010	3598.624	1423	861	1.652729	6.15649	65.07258	6.899117	14.72633	5.054159	26.06508	42.93309	24.96767	11.90764	0.997733	0.907537	1.798283
21	0	193386	3380.266	1360	758	1.794195	5.330686	59.08495	5.766157	14.24519	3.447284	22.5856	42.26467	20.10946	12.32236	0.997586	0.911058	1.704064
22	0	330048	3600.305	1387	916	1.514192	3.849416	39.27367	12.83173	27.15739	6.820429	32.12817	58.60535	25.73165	14.20206	0.997771	0.84893	2.700594
23	0	359163.5	3596.44	1397	878	1.591116	3.415063	36.01252	12.53174	26.06077	8.825434	30.36366	53.93353	26.0796	10.82811	0.998609	0.848352	2.737034
24	0	219832	3512.055	1385	844	1.640995	5.317424	56.10891	8.538715	16.16851	6.102978	28.17918	44.52646	25.06479	10.79461	0.998002	0.902526	1.871212
25	0	231084.5	3534.548	1371	900	1.523333	5.339605	54.0626	8.632995	16.82743	6.517402	28.42363	45.06641	27.15636	12.10391	0.997908	0.896733	1.944669
26	0	229428.5	3531.66	1364	912	1.495614	5.422029	54.36389	8.93116	16.51889	7.334003	29.57161	44.50895	28.92823	13.00753	0.997786	0.895131	1.961746

background_substract_camera_capture_leaf_file.ipynb : contains exploration of techniques to create a background subtraction function to remove background from mobile camera captured leaf images

Code:

Importing necessary libraries

Image background subtraction - Testfile

This file explores the method of background subtraction of plant leaf images captured from mobile camera. Background subtracted images will then be treated as input images to the plant leaf identification system.

Importing necessary libraries

In [1]:
```
import os
import cv2
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
%matplotlib inline
```

Reading the image

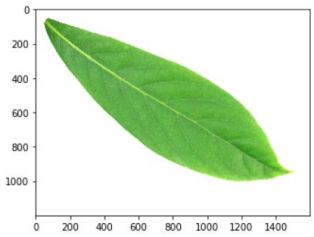
Reading the image

Note: 'mobile captures' folder must be in the project root

In [2]:
```
test_img_path = 'mobile captures\\' + 'Test.jpg'
```

In [3]:
```
main_img = cv2.imread(test_img_path)
img = cv2.cvtColor(main_img, cv2.COLOR_BGR2RGB)
plt.imshow(img,cmap="Greys_r")
```

Out[3]: `<matplotlib.image.AxesImage at 0x199a1b80390>`



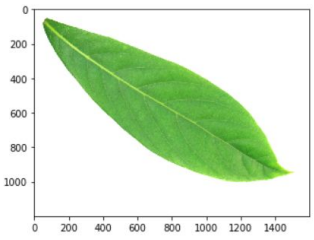
Resizing image to (1600,1200)

Resizing the image to (1600,1200) - Optional

This is done as all the images in the flavia dataset were of size (1600,1200)

In [5]:
```
resized_image = cv2.resize(img, (1600, 1200))
plt.imshow(resized_image,cmap="Greys_r")
```

Out[5]: `<matplotlib.image.AxesImage at 0x22b60496f98>`



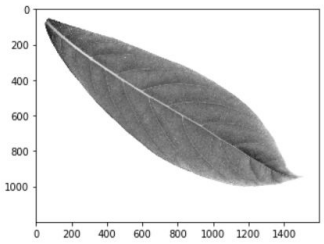
In [6]:
```
y,x,_ = img.shape
```

Converting Image to GrayScale

Converting image to grayscale

```
In [7]: gs = cv2.cvtColor(resized_image,cv2.COLOR_RGB2GRAY)
        plt.imshow(gs,cmap="Greys_r")
```

```
Out[7]: <matplotlib.image.AxesImage at 0x22b611330f0>
```

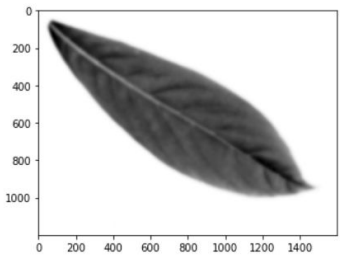


Smoothing image using Gaussian filter of size (55,55)

Smoothing image using Guassian filter of size (55,55)

```
In [8]: blur = cv2.GaussianBlur(gs, (55,55),0)
        plt.imshow(blur,cmap="Greys_r")
```

```
Out[8]: <matplotlib.image.AxesImage at 0x22b6174dc88>
```



Adaptive image thresholding using Otsu's thresholding method

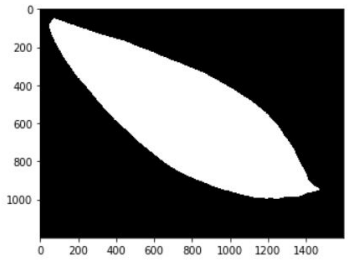
Otsu's Method:

Otsu's thresholding method involves iterating through all the possible threshold values and calculating a measure of spread for the pixel levels each side of the threshold, i.e. the pixels that either fall in foreground or background. The aim is to find the threshold value where the sum of foreground and background spreads is at its minimum.

Adaptive image thresholding using Otsu's thresholding method

```
In [7]: ret_otsu,im_bw_otsu = cv2.threshold(blur,0,255,cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)
        plt.imshow(im_bw_otsu,cmap='Greys_r')
```

```
Out[7]: <matplotlib.image.AxesImage at 0x199a4693940>
```



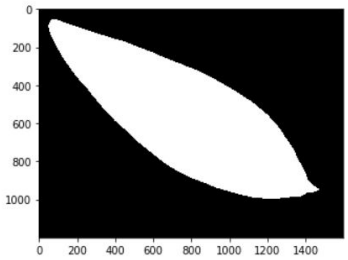
Closing of holes using Morphological Transformation

Closing of holes using Morphological Transformation

Performed so as to close any holes present in the leaf

```
In [8]: kernel = np.ones((50,50),np.uint8)
        closing = cv2.morphologyEx(im_bw_otsu, cv2.MORPH_CLOSE, kernel)
        plt.imshow(closing,cmap="Greys_r")
```

```
Out[8]: <matplotlib.image.AxesImage at 0x199a4c955c0>
```



Finding contours

Finding contours

```
In [9]: contours, hierarchy = cv2.findContours(closing,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)
```

```
In [10]: len(contours)
```

```
Out[10]: 1
```

Finding the correct leaf contour from the list of contours

Here we find the correct leaf contour from the list from the given length.

Finding the correct leaf contour from the list of contours

The following function finds the correct leaf contour by taking any coordinate point of the leaf (default - center point) and checks whether the current contour contains that point or not. Returns the index of the correct contour.

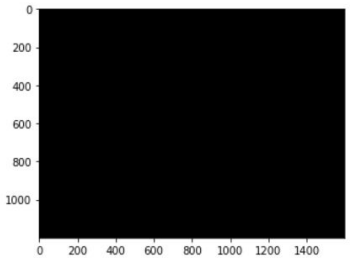
In [11]:
```
def find_contour(cnts):
    contains = []
    y_ri,x_ri, _ = resized_image.shape
    for cc in cnts:
        yn = cv2.pointPolygonTest(cc,(x_ri//2,y_ri//2),False)
        contains.append(yn)

    val = [contains.index(temp) for temp in contains if temp>0]
    print(contains)
    return val[0]
```

Creating mask image for background subtraction using leaf contour

Creating mask image for background subtraction using leaf contour

In [10]:
```
black_img = np.empty([1200,1600,3],dtype=np.uint8)
black_img.fill(0)
plt.imshow(black_img,cmap="Greys_r")
```

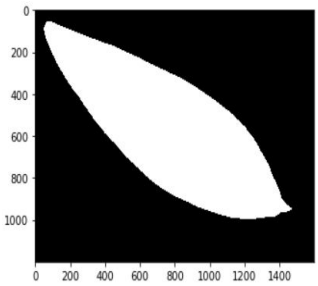
Out[10]: <matplotlib.image.AxesImage at 0x22b61992c18>



In [26]:
```
index = find_contour(contours)
cnt = contours[index]
mask = cv2.drawContours(black_img, [cnt] , 0, (255,255,255), -1)
plt.imshow(mask)
```

```
[1.0]
```

Out[26]: <matplotlib.image.AxesImage at 0x22b61692630>



Performing masking operation on the original image

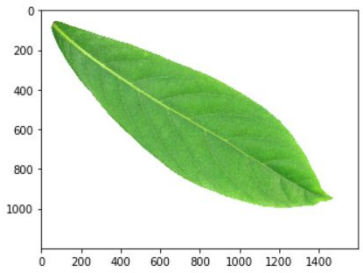
Performing masking operation on the original image

```
In [27]: maskedImg = cv2.bitwise_and(resized_image, mask)

In [28]: white_pix = [255,255,255]
         black_pix = [0,0,0]

         final_img = maskedImg
         h,w,channels = final_img.shape
         for x in range(0,w):
             for y in range(0,h):
                 channels_xy = final_img[y,x]
                 if all(channels_xy == black_pix):
                     final_img[y,x] = white_pix
```

Background subtracted image

Background subtracted image

```
In [29]: plt.imshow(final_img)
Out[29]: <matplotlib.image.AxesImage at 0x22b616f6c50>
```



- **single_image_process_file.ipynb**: contains exploration of preprocessing and feature extraction techniques by operating on a single image : contains exploration of preprocessing and feature extraction techniques by operating on a single image

Code :

Importing necessary libraries

Single image preprocessing and feature extraction - Testfile

This file explores the techniques to be used for preprocessing and feature extraction for the Flavia leaves dataset images.

Importing necessary libraries

In [1]:
```
import os
import cv2
import numpy as np
from matplotlib import pyplot as plt
%matplotlib inline
```

Reading the image

Reading the image

Note: 'Flavia leaves dataset' should be in the project root containing Flavia images.

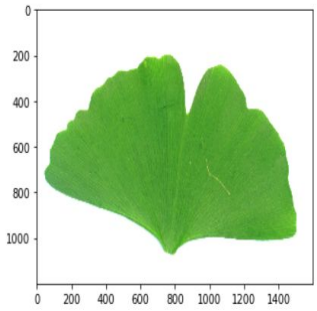
In [5]:
```
ds_path = "Flavia leaves dataset"
```

In [6]:
```
test_img_path = ds_path + "\\2456.jpg"
test_img_path
```

Out[6]: 'Flavia leaves dataset\\2456.jpg'

In [7]:
```
main_img = cv2.imread(test_img_path)
img = cv2.cvtColor(main_img, cv2.COLOR_BGR2RGB)
plt.imshow(img)
```

Out[7]: <matplotlib.image.AxesImage at 0x1c7d3e8b278>

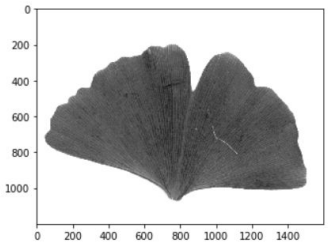


Converting image to grayscale

Converting image to grayscale

```
In [8]: gs = cv2.cvtColor(img,cv2.COLOR_RGB2GRAY)
        plt.imshow(gs,cmap='Greys_r')
```

```
Out[8]: <matplotlib.image.AxesImage at 0x1c7d44c9eb8>
```




```
In [9]: gs.shape
```

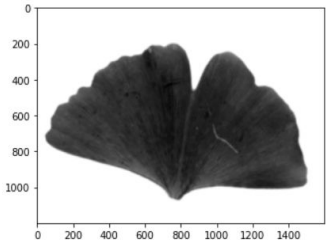
```
Out[9]: (1200, 1600)
```

Smoothing image using Gaussian filter of size (25,25)

Smoothing image using Guassian filter of size (25,25)

```
In [10]: blur = cv2.GaussianBlur(gs, (25,25),0)
         plt.imshow(blur,cmap='Greys_r')
```

```
Out[10]: <matplotlib.image.AxesImage at 0x1c7d49f4fd0>
```

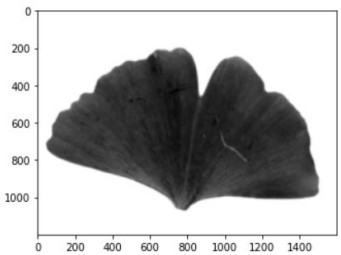


Adaptive image thresholding using Otsu's thresholding method

Smoothing image using Guassian filter of size (25,25)

```
In [10]: blur = cv2.GaussianBlur(gs, (25,25),0)
         plt.imshow(blur,cmap='Greys_r')
```

```
Out[10]: <matplotlib.image.AxesImage at 0x1c7d49f4fd0>
```



Closing of holes using Morphological Transformation

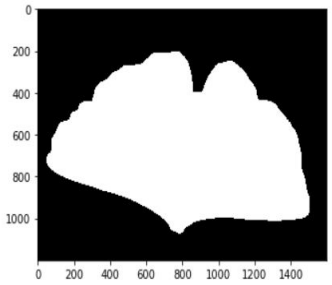
Closing of holes using Morphological Transformation

Performed so as to close any holes present in the leaf

```
In [14]: kernel = np.ones((50,50),np.uint8)
         closing = cv2.morphologyEx(im_bw_otsu, cv2.MORPH_CLOSE, kernel)
```

```
In [15]: plt.imshow(closing,cmap='Greys_r')
```

```
Out[15]: <matplotlib.image.AxesImage at 0x1c7d5282278>
```



Boundary Extraction:

Boundary extraction using sobel filters – Not effective

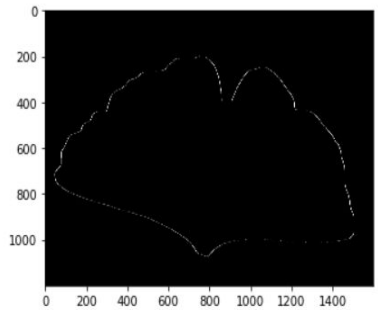
Boundary extraction using sobel filters - Not effective

Trying to extract the boundary of the leaf using sobel filters. The image after edge extraction is thresholded using Otsu's method. Then the gaps were closed using Closing operation of Morphological Transformation.

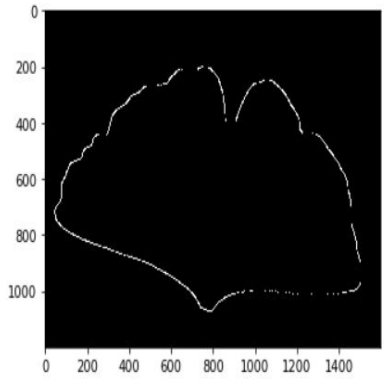
This method is not effective as even after performing morphological transformation, gaps still persist.

```
In [16]: sobelx64f = cv2.Sobel(closing,cv2.CV_64F,1,0,ksize=5)
         abs_sobel64f = np.absolute(sobelx64f)
         sobel_8u = np.uint8(abs_sobel64f)
         plt.imshow(abs_sobel64f,cmap='Greys_r')
```

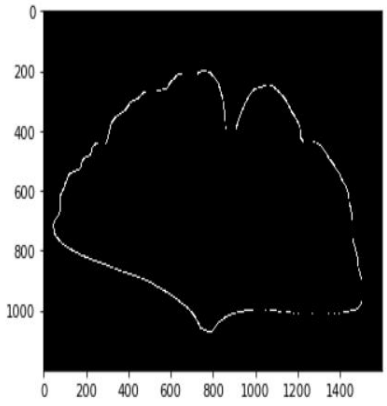
```
Out[16]: <matplotlib.image.AxesImage at 0x1c7d52e52e8>
```



```
In [40]: ret_sobel,im_bw_sobel = cv2.threshold(sobel_8u,1,255,cv2.THRESH_BINARY)
         plt.imshow(im_bw_sobel,cmap='Greys_r')

Out[40]: <matplotlib.image.AxesImage at 0x1dabd628fd0>
```




```
In [19]: kernel_edge = np.ones((15,15),np.uint8)
         closing_edge = cv2.morphologyEx(im_bw_sobel, cv2.MORPH_CLOSE, kernel_edge)
         plt.imshow(closing_edge,cmap='Greys_r')

Out[19]: <matplotlib.image.AxesImage at 0x1c7d53bea58>
```



Boundary extraction using contours

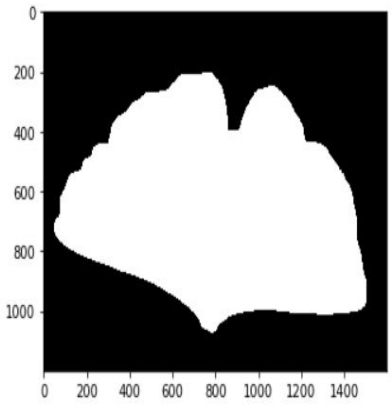
Here the boundary extraction is done after the image is in grayscale form and closing the holes or gaps in the leaves for a better accuracy.

Boundary extraction using contours - Effective

Contours are used to extract leaf boundaries. They are continous, sharp and there are no gaps between the boundary pixels

In [20]:
```
plt.imshow(closing,cmap="Greys_r")
```

Out[20]: <matplotlib.image.AxesImage at 0x1c7d5422240>



In [21]:
```
contours, hierarchy = cv2.findContours(closing,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)
```

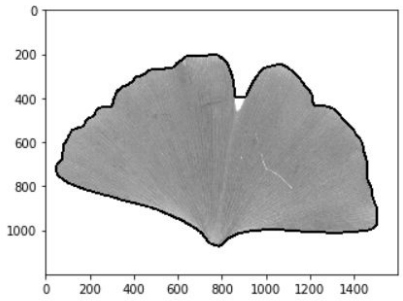
In [22]:
```
len(contours)
```

Out[22]: 1

In [23]:
```
cnt = contours[0]
len(cnt)
print(cnt)
```

```
[[[753 201]]

 [[752 202]]

 [[749 202]]

 ...

 [[771 202]]

 [[762 202]]

 [[761 201]]]
```

```
In [29]: plottedContour = cv2.drawContours(gs,contours,-1,(0,255,0),10)
         plt.imshow(plottedContour,cmap="Greys_r")

Out[29]: <matplotlib.image.AxesImage at 0x1c7da682978>
```



Morphological Processing

1.Shape based features

Calculating moments using contour

Morphological processing

1. Shape based features

Calculating moments using contours

```
In [54]: M = cv2.moments(cnt)
         M

Out[54]: {'m00': 831260.5,
          'm10': 678825250.5,
          'm01': 540788221.8333333,
          'm20': 658736623388.0833,
          'm11': 453174341096.625,
          'm02': 384443942913.0833,
          'm30': 705846927495854.4,
          'm21': 450755997854018.3,
          'm12': 330450154769313.06,
          'm03': 291772518407976.56,
          'mu20': 104393284908.28333,
          'mu11': 11555017005.726807,
          'mu02': 32626551284.756836,
          'mu30': -2591220282978.375,
          'mu21': 3333460820893.0703,
          'mu12': 1470382422685.996,
          'mu03': -784225946214.75,
          'nu20': 0.15107697325726974,
          'nu11': 0.016722311178302324,
          'nu02': 0.047216836027860594,
          'nu30': -0.004113026183512887,
          'nu21': 0.005291179498752782,
          'nu12': 0.002333927935039665,
          'nu03': -0.001244796465881305}
```

Area

```
In [30]: area = cv2.contourArea(cnt)
         area

Out[30]: 831260.5
```

Perimeter

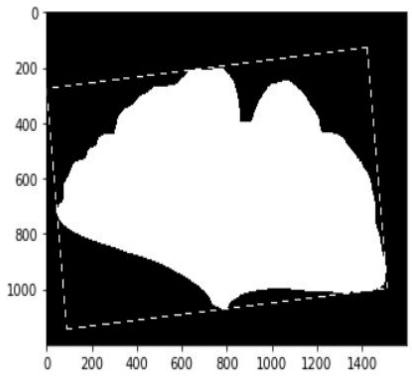
```
In [31]: perimeter = cv2.arcLength(cnt,True)
         perimeter

Out[31]: 4221.749678015709
```

Fitting in the best-fit rectangle and ellipse

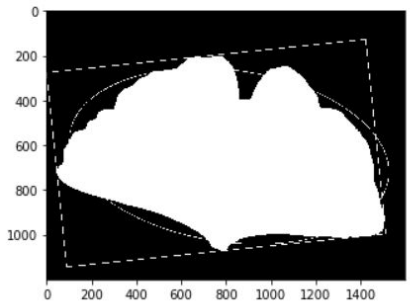
Fitting in the best-fit rectangle and ellipse

The best-fit rectangle is chosen and not ellipse as removes (leaves out) some portion at the extreme ends of the leaf image.

```
In [32]: rect = cv2.minAreaRect(cnt)
         box = cv2.boxPoints(rect)
         box = np.int0(box)
         contours_im = cv2.drawContours(closing,[box],0,(255,255,255),2)
         plt.imshow(contours_im,cmap="Greys_r")

Out[32]: <matplotlib.image.AxesImage at 0x1c7da8c6a58>
```



```
In [33]: ellipse = cv2.fitEllipse(cnt)
         im = cv2.ellipse(closing,ellipse,(255,255,255),2)
         plt.imshow(closing,cmap="Greys_r")

Out[33]: <matplotlib.image.AxesImage at 0x1c7dab0e320>
```



Shape based features calculated- Aspect ratio,rectangularity,circularity etc.

Shape based features calculated - Aspect ratio, rectangularity, circularity etc.

```
In [34]: x,y,w,h = cv2.boundingRect(cnt)
         aspect_ratio = float(w)/h
         aspect_ratio

Out[34]: 1.6739380022962111

In [35]: rectangularity = w*h/area
         rectangularity

Out[35]: 1.5277016049722079

In [36]: circularity = ((perimeter)**2)/area
         circularity

Out[36]: 21.441137097005985

In [37]: equi_diameter = np.sqrt(4*area/np.pi)
         equi_diameter

Out[37]: 1028.7826498227523

In [38]: (x,y),(MA,ma),angle = cv2.fitEllipse(cnt)
```

2.Color based features

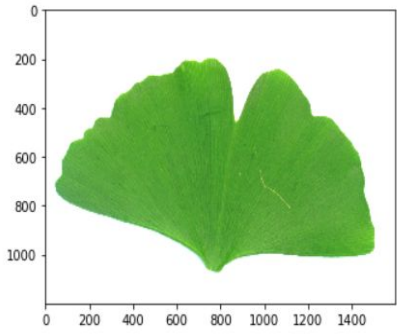
Calculating color based features – mean,std-dev of the RGB channels

2. Color based features

Calculating color based features - mean, std-dev of the RGB channels

```
In [64]: plt.imshow(img,cmap="Greys_r")
```

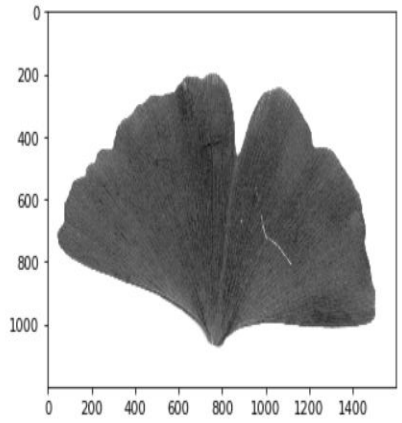
```
Out[64]: <matplotlib.image.AxesImage at 0x1dac1d91f98>
```



Red Channel

```
In [43]: red_channel = img[:,:,0]
         plt.imshow(red_channel,cmap="Greys_r")
```

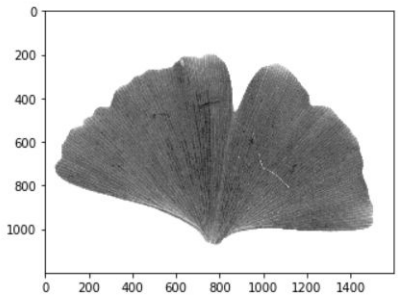
```
Out[43]: <matplotlib.image.AxesImage at 0x1c7dad55240>
```



Green Channel

```
In [66]: green_channel = img[:,:,1]
         plt.imshow(green_channel,cmap="Greys_r")
```

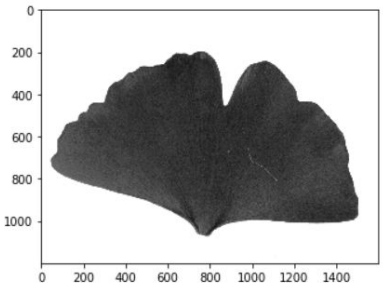
```
Out[66]: <matplotlib.image.AxesImage at 0x1dac1e5aa20>
```



Blue Channel

```
In [44]: blue_channel = img[:,:,2]
         plt.imshow(blue_channel,cmap="Greys_r")
```

```
Out[44]: <matplotlib.image.AxesImage at 0x1c7daf8fc88>
```




```
In [68]: np.mean(blue_channel)
```

```
Out[68]: 167.72929895833335
```

```
In [69]: blue_channel[blue_channel == 255] = 0
         green_channel[green_channel == 255] = 0
         red_channel[red_channel == 255] = 0
```

Red Mean

```
In [35]: red_mean = np.mean(red_channel)
         red_mean

Out[35]: 46.304093229166668
```

Green Mean

```
In [36]: green_mean = np.mean(green_channel)
         green_mean

Out[36]: 99.320648958333337
```

Blue Mean

```
In [37]: blue_mean = np.mean(blue_channel)
         blue_mean

Out[37]: 27.545344791666668
```

```
In [38]: red_var = np.std(red_channel)
         red_var

Out[38]: 50.659233268268196
```

3.Texture based features

Using Haralick moments –

3. Texture based features

Using Haralick moments - calculating texture based features such as contrast, correlation, entropy

```
In [45]: import mahotas as mt

In [46]: textures = mt.features.haralick(gs)
         ht_mean = textures.mean(axis=0)
         ht_mean

Out[46]: array([ 3.05588455e-01,  1.48137947e+02,  9.84563299e-01,  4.79813905e+03,
                 6.75625867e-01,  3.91461353e+02,  1.90444182e+04,  3.77516810e+00,
                 5.17651411e+00,  1.52647581e-03,  2.25908957e+00, -4.53976753e-01,
                 9.75623519e-01])

In [47]: print(ht_mean[1]) #contrast
         print(ht_mean[2]) #correlation
         print(ht_mean[4]) #inverse difference moments
         print(ht_mean[8]) #entropy

         148.13794663972374
         0.9845632987704573
         0.6756258672392551
         5.176514107685743
```

Plant Leaf Classification

Importing Necessary Libraries

Plant Leaf Classification

Applying machine learning models for classification of plant leaf images

Importing necessary libraries ¶

```
In [1]: import numpy as np
        import pandas as pd
        import os
        import string
```

Reading the Dataset

Reading the dataset

```
In [2]: dataset = pd.read_csv("Flavia_features.csv")

In [3]: dataset.head(5)
```

Out[3]:

	Unnamed: 0	area	perimeter	physiological_length	physiological_width	aspect_ratio	rectangularity	circularity	mean_r	mean_g	mean_b	s
0	0	197484.0	3479.036038	1416	759	1.865613	5.442183	61.289480	6.395667	13.643413	4.388007	24
1	0	101248.0	2490.381812	1190	130	9.153846	1.527931	61.255546	7.049316	9.232018	10.876066	33
2	0	86570.5	2290.683327	1095	119	9.201681	1.505189	60.612219	3.434303	6.371511	2.644757	19
3	0	190214.0	2856.479353	1318	254	5.188976	1.759976	42.896287	7.670415	13.303599	6.049157	28
4	0	227727.0	2917.248904	1324	286	4.629371	1.662798	37.370804	8.992028	16.671173	6.294281	30

```
In [4]: type(dataset)
```

Out[4]: pandas.core.frame.DataFrame

```
In [5]: maindir = r'C:\\Users\\sarat\Anaconda3\\Plant-Leaf-Identification-master'
        ds_path = maindir + "\\Flavia leaves dataset"
        img_files = os.listdir(ds_path)
```

Creating target labels

Break Points – is the value corresponding to the number of points k for which a data set exists.

Creating target labels

Breakpoints are used alongside the image file to create a vector of target labels. The breakpoints are specified in Flavia leaves dataset website.

```
In [8]: breakpoints = [1001,1059,1060,1122,1552,1616,1123,1194,1195,1267,1268,1323,1324,1385,1386,1437,1497,1551,1438,1496,2001,2050,
```

Creation of Target Labels

```
In [9]: target_list = []
for file in img_files:
    target_num = int(file.split(".")[0])
    flag = 0
    i = 0
    for i in range(0,len(breakpoints),2):
        if((target_num >= breakpoints[i]) and (target_num <= breakpoints[i+1])):
            flag = 1
            break
    if(flag==1):
        target = int((i/2))
        target_list.append(target)
```

```
In [9]: y = np.array(target_list)
        y

Out[9]: array([ 0,  0,  0, ..., 31, 31, 31])
```

```
In [10]: X = dataset.iloc[:,1:]

In [11]: X.head(5)
```

Out[11]:

	area	perimeter	physiological_length	physiological_width	aspect_ratio	rectangularity	circularity	mean_r	mean_g	mean_b	stddev_r	st
0	197484.0	3479.036038	1416	759	1.865613	5.442183	61.289480	6.395667	13.643413	4.388007	24.025329	40.
1	101248.0	2490.381812	1190	130	9.153846	1.527931	61.255546	7.049316	9.232018	10.876066	33.816205	37.
2	86570.5	2290.683327	1095	119	9.201681	1.505189	60.612219	3.434303	6.371511	2.644757	19.975699	29.
3	190214.0	2856.479353	1318	254	5.188976	1.759976	42.896287	7.670415	13.303599	6.049157	28.822885	40.
4	227727.0	2917.248904	1324	286	4.629371	1.662798	37.370804	8.992028	16.671173	6.294281	30.967158	45.

```
In [15]: y[0:5]
```

Out[15]: array([0, 0, 0, 0, 0])

Train test split

Importing Test Train Split from Sklearn

```
In [14]: from sklearn.model_selection import train_test_split

In [15]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state = 142)
```

```
In [16]: X_train.head(5)
```

Out[16]:

	area	perimeter	physiological_length	physiological_width	aspect_ratio	rectangularity	circularity	mean_r	mean_g	mean_b	stddev_
147	816496.5	3599.018794	1101	1083	1.016620	1.460365	15.864044	36.740296	70.312759	25.082371	45.40749
837	921134.0	4097.732429	1377	1038	1.326590	1.551703	18.229064	34.696545	64.960450	23.699098	46.63946
358	552379.0	3850.814822	1398	1049	1.332698	2.654884	26.845291	29.863131	49.429785	23.717367	48.48232
962	38454.5	3035.410698	1176	822	1.430657	25.138072	239.600518	4.282973	5.213605	4.262034	27.77227
1101	941172.0	3719.580197	1303	987	1.320162	1.366446	14.700051	43.594426	72.467562	32.378145	47.40852

```
In [18]: y_train[0:5]
```

Out[18]: array([3, 13, 6, 16, 18])

Feature Scaling

from sklearn.preprocessing import StandardScaler

Feature Scaling

```
In [19]: from sklearn.preprocessing import StandardScaler

In [20]: sc_X = StandardScaler()
         X_train = sc_X.fit_transform(X_train)
         X_test = sc_X.transform(X_test)

In [21]: X_train[0:2]

Out[21]: array([[ 0.83193224, -0.17692269, -0.90777284,  0.86491733, -0.9528451 ,
                 -0.33789941, -0.42611334,  0.01198242,  0.22802428, -0.06515913,
                 -0.03038517,  0.66516332, -0.17255204, -0.1745652 ,  0.2744626 ,
                 -0.97141055,  0.72845006],
                [ 1.22949141,  0.46823954,  0.25932835,  0.67383324, -0.45412004,
                  -0.31773833, -0.37146503, -0.04156278,  0.1082232 , -0.10299271,
                   0.04545738, -0.01456099, -0.11043539,  2.35360002, -1.92405329,
                  -1.86068867,  1.7415609 ]])

In [22]: y_train[0:2]

Out[22]: array([ 3, 13])
```

Applying SVM Classifier Model

from sklearn import svm

Applying SVM classifier model

```
In [23]: from sklearn import svm

In [24]: clf = svm.SVC()
         clf.fit(X_train,y_train)

Out[24]: SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
             decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
             kernel='rbf', max_iter=-1, probability=False, random_state=None,
             shrinking=True, tol=0.001, verbose=False)

In [25]: y_pred = clf.predict(X_test)

In [26]: from sklearn import metrics

In [27]: metrics.accuracy_score(y_test, y_pred)

Out[27]: 0.7975567190226877
```

Metrics Accuracy,Precision,Recall,F1 Score,Support

```
In [28]: print(metrics.classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
0	0.83	0.28	0.42	18
1	0.65	0.72	0.68	18
2	0.90	0.86	0.88	22
3	0.81	0.96	0.88	27
4	0.96	1.00	0.98	24
5	0.89	1.00	0.94	16
6	0.73	0.95	0.83	20
7	0.83	0.71	0.77	14
8	0.35	0.50	0.41	12
9	0.93	0.93	0.93	15
10	0.65	0.94	0.77	18
11	0.92	0.61	0.73	18
12	0.58	0.78	0.67	18
13	0.94	0.71	0.81	24
14	0.60	0.46	0.52	13
15	0.44	0.22	0.30	18
16	1.00	1.00	1.00	26
17	0.94	1.00	0.97	17
18	0.94	0.81	0.87	21
19	0.82	0.93	0.87	15
20	0.62	0.93	0.74	14
21	0.62	0.87	0.72	15
22	1.00	0.93	0.96	14
23	0.70	0.88	0.78	16
24	0.92	0.69	0.79	16
25	0.75	0.69	0.72	13
26	0.93	0.93	0.93	15
27	0.86	0.86	0.86	22
28	0.62	0.71	0.67	14
29	0.85	0.81	0.83	21
30	0.91	0.67	0.77	15
31	0.95	0.79	0.86	24
accuracy			0.80	573
macro avg	0.80	0.79	0.78	573
weighted avg	0.81	0.80	0.79	573

Performing parameter tuning of the m

Performing parameter tuning of the model

```
In [29]: from sklearn.model_selection import GridSearchCV

In [30]: parameters = [{'kernel': ['rbf'],
                        'gamma': [1e-4, 1e-3, 0.01, 0.1, 0.2, 0.5],
                        'C': [1, 10, 100, 1000]},
                       {'kernel': ['linear'], 'C': [1, 10, 100, 1000]}
                      ]

In [31]: svm_clf = GridSearchCV(svm.SVC(decision_function_shape='ovr'), parameters, cv=5)
         svm_clf.fit(X_train, y_train)

         C:\Users\sarat\Anaconda3\lib\site-packages\sklearn\model_selection\_search.py:813: DeprecationWarning: The default of the `i
         id` parameter will change from True to False in version 0.22 and will be removed in 0.24. This will change numeric results w
         hen test-set sizes are unequal.
           DeprecationWarning)

Out[31]: GridSearchCV(cv=5, error_score='raise-deprecating',
                      estimator=SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
                                    decision_function_shape='ovr', degree=3,
                                    gamma='auto_deprecated', kernel='rbf', max_iter=-1,
                                    probability=False, random_state=None, shrinking=True,
                                    tol=0.001, verbose=False),
                      iid='warn', n_jobs=None,
                      param_grid=[{'C': [1, 10, 100, 1000],
                                   'gamma': [0.0001, 0.001, 0.01, 0.1, 0.2, 0.5],
                                   'kernel': ['rbf']},
                                  {'C': [1, 10, 100, 1000], 'kernel': ['linear']}],
                      pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
                      scoring=None, verbose=0)

In [32]: svm_clf.best_params_

Out[32]: {'C': 100, 'kernel': 'linear'}
```

odel using Grid Search CV

```
In [32]: means = svm_clf.cv_results_['mean_test_score']
         stds = svm_clf.cv_results_['std_test_score']
         for mean, std, params in zip(means, stds, svm_clf.cv_results_['params']):
             print("%0.3f (+/-%0.03f) for %r" % (mean, std * 2, params))

         0.076 (+/-0.005) for {'C': 1, 'gamma': 0.0001, 'kernel': 'rbf'}
         0.198 (+/-0.023) for {'C': 1, 'gamma': 0.001, 'kernel': 'rbf'}
         0.603 (+/-0.061) for {'C': 1, 'gamma': 0.01, 'kernel': 'rbf'}
         0.833 (+/-0.048) for {'C': 1, 'gamma': 0.1, 'kernel': 'rbf'}
         0.843 (+/-0.039) for {'C': 1, 'gamma': 0.2, 'kernel': 'rbf'}
         0.859 (+/-0.035) for {'C': 1, 'gamma': 0.5, 'kernel': 'rbf'}
         0.199 (+/-0.021) for {'C': 10, 'gamma': 0.0001, 'kernel': 'rbf'}
         0.602 (+/-0.067) for {'C': 10, 'gamma': 0.001, 'kernel': 'rbf'}
         0.839 (+/-0.035) for {'C': 10, 'gamma': 0.01, 'kernel': 'rbf'}
         0.888 (+/-0.027) for {'C': 10, 'gamma': 0.1, 'kernel': 'rbf'}
         0.882 (+/-0.014) for {'C': 10, 'gamma': 0.2, 'kernel': 'rbf'}
         0.872 (+/-0.017) for {'C': 10, 'gamma': 0.5, 'kernel': 'rbf'}
         0.599 (+/-0.068) for {'C': 100, 'gamma': 0.0001, 'kernel': 'rbf'}
         0.841 (+/-0.043) for {'C': 100, 'gamma': 0.001, 'kernel': 'rbf'}
         0.891 (+/-0.027) for {'C': 100, 'gamma': 0.01, 'kernel': 'rbf'}
         0.888 (+/-0.008) for {'C': 100, 'gamma': 0.1, 'kernel': 'rbf'}
         0.875 (+/-0.012) for {'C': 100, 'gamma': 0.2, 'kernel': 'rbf'}
         0.874 (+/-0.018) for {'C': 100, 'gamma': 0.5, 'kernel': 'rbf'}
         0.843 (+/-0.043) for {'C': 1000, 'gamma': 0.0001, 'kernel': 'rbf'}
         0.891 (+/-0.024) for {'C': 1000, 'gamma': 0.001, 'kernel': 'rbf'}
         0.903 (+/-0.013) for {'C': 1000, 'gamma': 0.01, 'kernel': 'rbf'}
         0.888 (+/-0.022) for {'C': 1000, 'gamma': 0.1, 'kernel': 'rbf'}
         0.877 (+/-0.015) for {'C': 1000, 'gamma': 0.2, 'kernel': 'rbf'}
         0.875 (+/-0.020) for {'C': 1000, 'gamma': 0.5, 'kernel': 'rbf'}
         0.873 (+/-0.036) for {'C': 1, 'kernel': 'linear'}
         0.896 (+/-0.015) for {'C': 10, 'kernel': 'linear'}
         0.904 (+/-0.016) for {'C': 100, 'kernel': 'linear'}
         0.904 (+/-0.028) for {'C': 1000, 'kernel': 'linear'}
```

```
In [33]: y_pred_svm = svm_clf.predict(X_test)

In [34]: metrics.accuracy_score(y_test, y_pred_svm)
Out[34]: 0.900523560209424
```

```
In [35]: print(metrics.classification_report(y_test, y_pred_svm))
```

	precision	recall	f1-score	support
0	0.93	0.78	0.85	18
1	0.82	0.78	0.80	18
2	1.00	0.91	0.95	22
3	0.93	0.96	0.95	27
4	1.00	1.00	1.00	24
5	0.94	1.00	0.97	16
6	0.73	0.95	0.83	20
7	0.87	0.93	0.90	14
8	0.54	0.58	0.56	12
9	1.00	1.00	1.00	15
10	0.94	0.89	0.91	18
11	0.72	0.72	0.72	18
12	0.94	0.89	0.91	18
13	0.81	0.88	0.84	24
14	0.92	0.85	0.88	13
15	0.81	0.72	0.76	18
16	1.00	1.00	1.00	26
17	0.94	1.00	0.97	17
18	0.88	1.00	0.93	21
19	0.88	0.93	0.90	15
20	1.00	1.00	1.00	14
21	0.88	1.00	0.94	15
22	1.00	1.00	1.00	14
23	0.84	1.00	0.91	16
24	0.93	0.81	0.87	16
25	0.90	0.69	0.78	13
26	0.94	1.00	0.97	15
27	1.00	0.86	0.93	22
28	0.93	0.93	0.93	14
29	0.90	0.90	0.90	21
30	0.92	0.73	0.81	15
31	0.96	0.92	0.94	24
accuracy			0.90	573
macro avg	0.90	0.89	0.89	573
weighted avg	0.90	0.90	0.90	573

Applying Random Forest Classifier Model

Training and Testing Model

```
In [40]: from sklearn.ensemble import RandomForestClassifier
         model = RandomForestClassifier(n_estimators = 100,random_state = 42)
         model.fit(X_train,y_train)

Out[40]: RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                                max_depth=None, max_features='auto', max_leaf_nodes=None,
                                min_impurity_decrease=0.0, min_impurity_split=None,
                                min_samples_leaf=1, min_samples_split=2,
                                min_weight_fraction_leaf=0.0, n_estimators=100,
                                n_jobs=None, oob_score=False, random_state=42, verbose=0,
                                warm_start=False)
```

Model Score

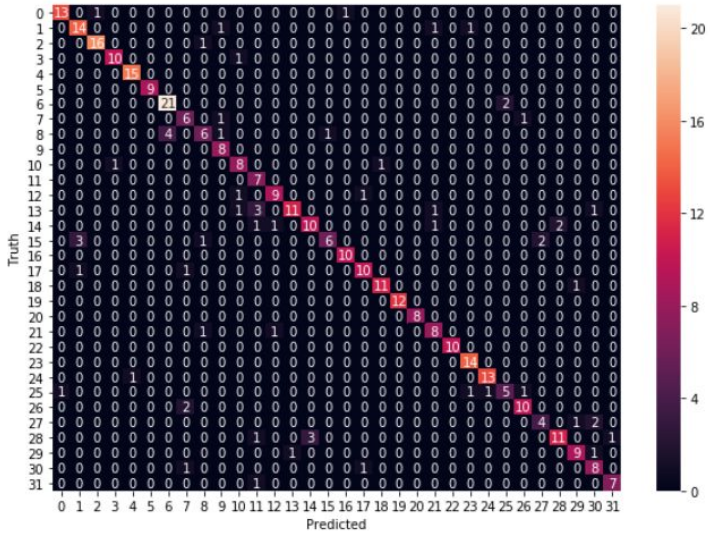
```
In [41]: model.score(X_test,y_test)
Out[41]: 0.8507853403141361
```

```
In [44]: y_predicted = model.predict(X_test)
```

```
In [45]: from sklearn.metrics import confusion_matrix
         cm = confusion_matrix(y_test, y_predicted)
         cm
Out[45]: array([[11,  0,  0, ...,  0,  0,  0],
                [ 0, 10,  0, ...,  0,  0,  0],
                [ 0,  0, 16, ...,  0,  0,  0],
                ...,
                [ 0,  0,  0, ..., 11,  0,  0],
                [ 0,  0,  0, ...,  2, 12,  0],
                [ 0,  0,  0, ...,  0,  0,  9]], dtype=int64)
```

Heat Map

```
In [37]: %matplotlib inline
         import matplotlib.pyplot as plt
         import seaborn as sn
         plt.figure(figsize=(10,7))
         sn.heatmap(cm, annot=True)
         plt.xlabel('Predicted')
         plt.ylabel('Truth')
Out[37]: Text(69.0, 0.5, 'Truth')
```



Metrics Accuracy,Precision,Recall,F1 Score,Support

In [47]:
```
print(classification_report(y_test,y_predicted))
print(accuracy_score(y_test, y_predicted))
```

```
              precision    recall  f1-score   support

           0       0.85      0.92      0.88        12
           1       0.83      0.77      0.80        13
           2       1.00      0.89      0.94        18
           3       0.91      1.00      0.95        10
           4       1.00      1.00      1.00        16
           5       1.00      1.00      1.00        14
           6       0.80      1.00      0.89        12
           7       0.83      1.00      0.91        10
           8       0.45      0.50      0.48        10
           9       1.00      1.00      1.00        13
          10       0.45      0.71      0.56         7
          11       0.69      0.85      0.76        13
          12       0.67      0.77      0.71        13
          13       0.90      0.64      0.75        14
          14       0.82      0.64      0.72        14
          15       0.80      0.80      0.80        10
          16       1.00      1.00      1.00        12
          17       0.93      1.00      0.96        13
          18       1.00      0.92      0.96        12
          19       1.00      1.00      1.00        10
          20       0.90      1.00      0.95         9
          21       1.00      0.78      0.88         9
          22       1.00      0.91      0.95        11
          23       0.85      0.79      0.81        14
          24       0.80      0.57      0.67         7
          25       0.78      0.58      0.67        12
          26       1.00      0.89      0.94         9
          27       1.00      0.69      0.82        13
          28       0.76      0.81      0.79        16
          29       0.73      1.00      0.85        11
          30       0.92      0.80      0.86        15
          31       0.69      0.90      0.78        10

    accuracy                           0.85       382
   macro avg       0.86      0.85      0.84       382
weighted avg       0.86      0.85      0.85       382

0.8507853403141361
```

Parameter Tuning of the Model

In [49]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.datasets import make_classification
from sklearn.ensemble import RandomForestClassifier
# Build a classification task using 3 informative features
X, y = make_classification(n_samples=1000,
                           n_features=10,
                           n_informative=3,
                           n_redundant=0,
                           n_repeated=0,
                           n_classes=2,
                           random_state=0,
                           shuffle=False)


rfc = RandomForestClassifier(n_jobs=-1,max_features= 'sqrt' ,n_estimators=50, oob_score = True)

param_grid = {
    'n_estimators': [200, 700],
    'max_features': ['auto', 'sqrt', 'log2']
}

CV_rfc = GridSearchCV(estimator=rfc, param_grid=param_grid, cv= 5)
CV_rfc.fit(X, y)
print(CV_rfc.best_params_)
```

```
{'max_features': 'log2', 'n_estimators': 700}
```

Metrics of Model

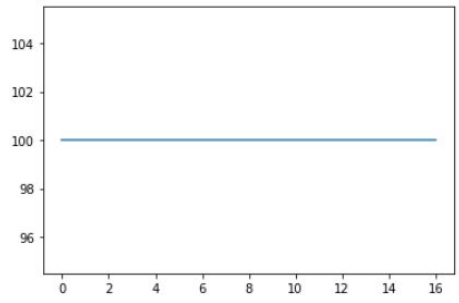
```
In [54]: model.fit(X_train,y_train)

Out[54]: RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                                max_depth=None, max_features='log2', max_leaf_nodes=None,
                                min_impurity_decrease=0.0, min_impurity_split=None,
                                min_samples_leaf=1, min_samples_split=2,
                                min_weight_fraction_leaf=0.0, n_estimators=700,
                                n_jobs=None, oob_score=False, random_state=None,
                                verbose=0, warm_start=False)

In [55]: model.score(X_test,y_test)

Out[55]: 0.8534031413612565
```

Dimensionality Reduction using PCA

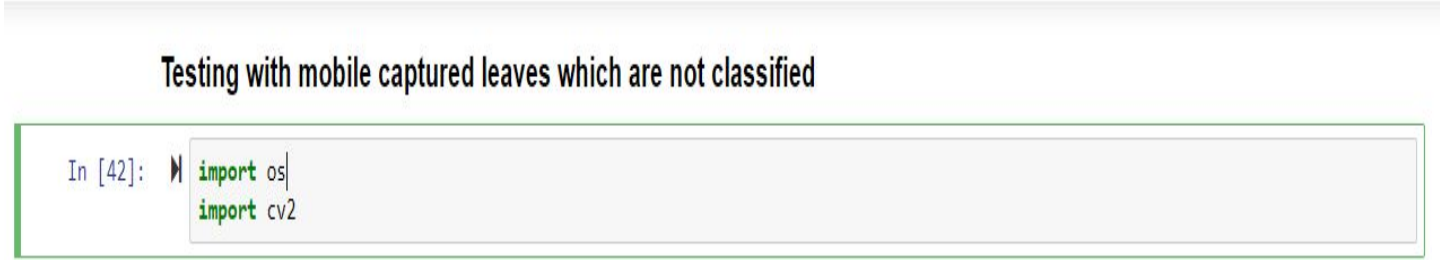
Dimensionality Reduction using PCA

```
In [36]: from sklearn.decomposition import PCA

In [37]: pca = PCA()

In [38]: pca.fit(X)

Out[38]: PCA(copy=True, iterated_power='auto', n_components=None, random_state=None,
             svd_solver='auto', tol=0.0, whiten=False)

In [39]: var= pca.explained_variance_ratio_
         var

Out[39]: array([9.99992897e-01, 6.41435225e-06, 3.92823849e-07, 2.08041223e-07,
                6.15523849e-08, 1.38782098e-08, 8.94936911e-09, 2.51985981e-09,
                8.66425534e-10, 2.10140828e-10, 1.07067508e-10, 1.32409473e-11,
                2.22566471e-12, 1.41983290e-12, 5.16455826e-13, 3.39152588e-15,
                1.00671075e-17])
```

```
In [39]: import matplotlib.pyplot as plt
         %matplotlib inline

In [40]: var1=np.cumsum(np.round(pca.explained_variance_ratio_, decimals=4)*100)
         plt.plot(var1)

Out[40]: [<matplotlib.lines.Line2D at 0x2b7dc7f64a8>]
```



Testing with mobile captured leaves which are not classified



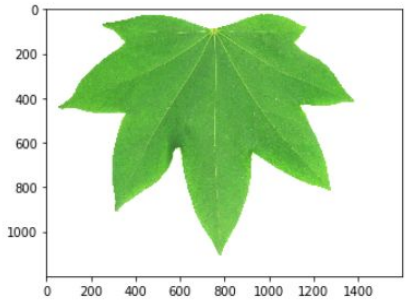
In [41]:

```
def bg_sub(filename):
    test_img_path = r'C:\\Users\\sarat\\Anaconda3\\Plant-Leaf-Identification-master\\mobile captures' + filename
    main_img = cv2.imread(test_img_path)
    img = cv2.cvtColor(main_img,cv2.COLOR_BGR2RGB)
    resized_image = cv2.resize(img, (1600, 1200))
    size_y,size_x,_ = img.shape
    gs = cv2.cvtColor(resized_image,cv2.COLOR_RGB2GRAY)
    blur = cv2.GaussianBlur(gs, (55,55),0)
    ret_otsu,im_bw_otsu = cv2.threshold(blur,0,255,cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)
    kernel = np.ones((50,50),np.uint8)
    closing = cv2.morphologyEx(im_bw_otsu, cv2.MORPH_CLOSE, kernel)

    contours, hierarchy = cv2.findContours(closing,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)

    contains = []
    y_ri,x_ri, _ = resized_image.shape
    for cc in contours:
        yn = cv2.pointPolygonTest(cc,(x_ri//2,y_ri//2),False)
        contains.append(yn)

    val = [contains.index(temp) for temp in contains if temp>0]
    index = val[0]

    black_img = np.empty([1200,1600,3],dtype=np.uint8)
    black_img.fill(0)

    cnt = contours[index]
    mask = cv2.drawContours(black_img, [cnt] , 0, (255,255,255), -1)

    maskedImg = cv2.bitwise_and(resized_image, mask)
    white_pix = [255,255,255]
    black_pix = [0,0,0]

    final_img = maskedImg
    h,w,channels = final_img.shape
    for x in range(0,w):
        for y in range(0,h):
            channels_xy = final_img[y,x]
            if all(channels_xy == black_pix):
                final_img[y,x] = white_pix

    return final_img
```

Importing Image

We import the images of the leaves that has to be calculated.

```
In [47]: filename = "\\Test.jpg"
         bg_rem_img = bg_sub(filename)

In [48]: plt.imshow(bg_rem_img)
Out[48]: <matplotlib.image.AxesImage at 0x2b7dcb74d68>
```



Importing Library Mahatos

```
In [49]: import mahotas as mt

In [50]: def feature_extract(img):
```

Feature Extraction- COLOUR,SHAPE,TEXTURE

In Feature Extraction preprocessing,shape,color,texture features are extracted

Preprocessing

```
In [50]: def feature_extract(img):
             names = ['area','perimeter','pysiological_length','pysiological_width','aspect_ratio','rectangularity','circularity', \
                      'mean_r','mean_g','mean_b','stddev_r','stddev_g','stddev_b', \
                      'contrast','correlation','inverse_difference_moments','entropy'
                     ]
             df = pd.DataFrame([], columns=names)

             #Preprocessing
             gs = cv2.cvtColor(img,cv2.COLOR_RGB2GRAY)
             blur = cv2.GaussianBlur(gs, (25,25),0)
             ret_otsu,im_bw_otsu = cv2.threshold(blur,0,255,cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)
             kernel = np.ones((50,50),np.uint8)
             closing = cv2.morphologyEx(im_bw_otsu, cv2.MORPH_CLOSE, kernel)
```

Shape Features

Shape features contain area, perimeter, aspect ratio, rectangularity and circularity

```
#Shape features
contours, image = cv2.findContours(closing,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)
cnt = contours[0]
M = cv2.moments(cnt)
area = cv2.contourArea(cnt)
perimeter = cv2.arcLength(cnt,True)
x,y,w,h = cv2.boundingRect(cnt)
aspect_ratio = float(w)/h
rectangularity = w*h/area
circularity = ((perimeter)**2)/area
```

Color Features

```
#Color features
red_channel = img[:,:,0]
green_channel = img[:,:,1]
blue_channel = img[:,:,2]
blue_channel[blue_channel == 255] = 0
green_channel[green_channel == 255] = 0
red_channel[red_channel == 255] = 0

red_mean = np.mean(red_channel)
green_mean = np.mean(green_channel)
blue_mean = np.mean(blue_channel)

red_std = np.std(red_channel)
green_std = np.std(green_channel)
blue_std = np.std(blue_channel)
```

Texture Features

```
#Texture features
textures = mt.features.haralick(gs)
ht_mean = textures.mean(axis=0)
contrast = ht_mean[1]
correlation = ht_mean[2]
inverse_diff_moments = ht_mean[4]
entropy = ht_mean[8]

vector = [area,perimeter,w,h,aspect_ratio,rectangularity,circularity,\
          red_mean,green_mean,blue_mean,red_std,green_std,blue_std,\
          contrast,correlation,inverse_diff_moments,entropy
         ]

df_temp = pd.DataFrame([vector],columns=names)
df = df.append(df_temp)

return df
```

Leaf Feature Extracted

```
In [51]: features_of_img = feature_extract(bg_rem_img)
         features_of_img
```

Out[51]:

	area	perimeter	pysiological_length	pysiological_width	aspect_ratio	rectangularity	circularity	mean_r	mean_g	mean_b	stddev_r	stdc
0	750300.0	5257.771098	1322	1102	1.199637	1.941682	36.844138	37.941786	70.604677	26.328703	48.310965	88.2

Scaling of the Features

```
In [54]: scaled_features = sc_X.transform(features_of_img)
         print(scaled_features)
         # y_pred_mobile = svm_clf.predict(features_of_img)
         y_pred_mobile = svm_clf.predict(scaled_features)
         y_pred_mobile[0]

[[ 0.5804256   1.96892668  0.02675383  0.94559728 -0.6583805  -0.23165749
   0.05867169  0.04346084  0.23455829 -0.031071    0.14835805  1.07395551
  -0.19960066  0.46683473 -1.48247762 -0.72744903  0.44841727]]

Out[54]: 7
```

Prediction of Leaf

1.'pubescent bamboo'

2'Chinese horse chestnut'

3.'Anhui Barberry',

4.'Chinese redbud',

5.'true indigo'

6.'Japanese maple'

7.'Nanmu'

8.castor aralia'

9.'Chinese cinnamon'

10.’goldenrain tree'

11.'Big-fruited Holly',

12.'Japanese cheesewood',

13.'wintersweet'

14.'camphortree'

15.'Japan Arrowwood'

16.'sweet osmanthus'

17.'deodar'

18.'ginkgo’

19.’maidenhair tree',

20.'Crape myrtle,

21.Crepe myrtle'

22.'oleander'

23.'yew plum pine'

24.’Japanese Flowering Cherry'

25.'Glossy Privet'

26. 'Chinese Toon'

27.'peach'

28.'Ford Woodlotus'

29.'trident maple'

30.'Beale's barberry'

31.'southern magnolia',

32.'Canadian poplar',

34'Chinese tulip tree'

35.'tangerine'

```
In [53]: common_names = ['pubescent bamboo','Chinese horse chestnut','Anhui Barberry', \
                         'Chinese redbud','true indigo','Japanese maple','Nanmu',' castor aralia', \
                         'Chinese cinnamon','goldenrain tree','Big-fruited Holly','Japanese cheesewood', \
                         'wintersweet','camphortree','Japan Arrowwood','sweet osmanthus','deodar','ginkgo, maidenhair tree', \
                         'Crape myrtle, Crepe myrtle','oleander','yew plum pine','Japanese Flowering Cherry','Glossy Privet',\
                         'Chinese Toon','peach','Ford Woodlotus','trident maple','Beales barberry','southern magnolia',\
                         'Canadian poplar','Chinese tulip tree','tangerine'
                        ]
         common_names[y_pred_mobile[0]]

Out[53]: ' castor aralia'
```

TYPES OF TESTING:

In order to make sure that the system does not have errors, the different levels of testing strategies that are applied at differing phases of software development are:

Unit Testing: Unit Testing is done on individual modules as they are completed and become executable. It is confined only to the designer's requirements.Each module can be tested using the following two strategies:

Black Box Testing: In this strategy some test cases are generated as input conditions that fully execute all functional requirements for the program.

White Box testing:

In this the test cases are generated on the logic of each module by drawing flow graphs of that module and logical decisions are tested on all the cases.

It has been uses to generate the test cases in the following cases:

1.Guarantee that all independent paths have been executed.
2.Execute all logical decisions on their true and false sides.
3.Execute all loops at their boundaries and within their operational bounds.
4.Execute internal data structures to ensure their valid

2. Integrating Testing:

Integration testing ensures that software and subsystems work together as a whole. It tests the interface of all the modules to make sure that the modules behave properly when integrated together.

3. System Testing:

Involves in-house testing of the entire system before delivery to the user. Its aim is to satisfy the user that the system meets all requirements of the client's specifications.

4. Acceptance Testing:

It is a pre-delivery testing in which the entire system is tested at the client's site on real world

data to find errors.

Validation: The system has been tested and implemented successfully and thus ensured that all the requirements as listed in the software requirements specification are completely fulfilled. In case of erroneous input corresponding error messages are displayed.

8.RESULTS & DISCUSSION

After having done the analysis the proposed method is tested on the Flavia dataset which contains leaves of 32 plant species with over 3900 samples. Out of this total sample, 2500 were used as a training set and 100 were reserved for testing. All the input leaf images were either 1600x1200 or 800x600 which were resized to the later dimension. The feature set is based on 12 shape feature, 5 vain feature and fourier descriptor. An implementation of multiclass SVM and Random Forest is used for classification after dimensionality reduction using principal component analysis. 100 test images belonging to different classes were used to test the accuracy of the proposed method. The algorithms resulted in an aggregate accuracy of 90.50% and 83.5% respectively. This performance is good at this level, however the accuracy will be improved further by optimizing the feature vector and classifier.

9.CONCLUSION & FUTURE SCOPE

Plant leaf recognition is useful to identify the plant type. The methods used to extract plant leaf features are based on color, shape and texture etc. Classifiers play an important role to test the data and check the accuracy of classification algorithms. Supervised classification gives higher accuracy as compared to unsupervised classification algorithms.SVM and NN gives better results as compared to other classifiers. To identify different plant leaf images based on its surface parameter is a challenging and most expensive task. Plant leaf image surface parameters are color, texture and shape. The combined feature extracted from each of its parameters is used to identify plant type and gives better results as compared to using a single parameter.

FUTURE SCOPE

1. Improved segmentation by adding thresholding, region growing and clustering based approaches.
2. Extension of feature vector by adding shape feature and some texture features.
3. Adding weighted normalization by testing the retrieval accuracy of individual features and assigning more weight to it in the feature vector.
4. Testing of other classifiers or implementation of binary tree based SVM.
5. Selection of first five classifier outputs and weighting them accordingly to determine the confidence based label for reduction of false classification.

10.REFERENCES

http://en.wikipedia.org/wiki/Digital_image_processing

[2] http://en.wikipedia.org/wiki/Machine_learning

[3]http://en.wikipedia.org/wiki/List_of_machine_learning_concepts

[4] J.-X. Du, X.-F. Wang, and G.-J. Zhang, "Leaf shape based plant species recognition,"Applied Mathematics and Computation, vol. 185, 2007.

[5] Y. Ye, C. Chen, C.-T. Li, H. Fu, and Z. Chi, "A computerized plant species recognition system," inProceedings of 2004 International Sym-posium on Intelligent Multimedia, Video and Speech Processing, HongKong, October 2004.

[6] Z. Miao, M.-H. Gandelin, and B. Yuan, "An oopr-based rosevarietyrecognition system,"Engineering Applications of Artificial Intelligence,vol. 19, 2006

.[7] R. de Oliveira Plotze, M. Falvo, J. G. Padua, L. C. Bernacci, M. L. C.Vieira, G. C. X. Oliveira, and O. M. Bruno, "Leaf shape analysis using the multiscale minkowski fractal dimension, a new morphometric method: a study with passiflora (passifloraceae),"Canada Journal ofBotany, vol. 83, 2005.

[8] M. J. Dallwitz, "A general system for coding taxonomic descriptions,"Taxon, vol. 29, 1980.