|  |
| --- |
| **Department of E& TC Engineering**  **Vishwakarma Institute of Technology, Pune** |
| Vishwakarma Institute of Technology, Pune  **COURSE PROJECT REPORT**  **COMPUTER VISION LAB (EL 3221)**  **TITLE**  **Plant Disease Detection Using Image processing**  **BATCH**  **A3**  **Name: Pranesh Shridhar Kulkarni**  **Div.: ET-A**  **Roll No.: 66**  **Gr. No.: 11810198**    **Batch guide**  **Prof. Dr. Medha Wyawahare** |

**Index**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **CONTENT** | **Page No** |
| 1 | Title of project | 3 |
| 2 | Introduction | 3 |
| 3 | Background/ Literature survey | 4 |
| 4 | Algorithm / Methodology used | 6 |
| 5 | Performance Analysis | 12 |
| 6 | Python Code | 15 |
| 8 | Output images | 26 |
| 9 | Applications | 28 |
| 10 | Conclusion | 29 |
| 11 | References | 30 |
|  |  |  |

**Plant Disease Detection Using Image Processing and Machine Learning**

1. **Introduction**

In India about 70% of the populace relies on agriculture. Identification of the plant diseases is important in order to prevent the losses within the yield. It's terribly troublesome to observe the plant diseases manually. It needs tremendous quantity of labor, expertize within the plant diseases, and conjointly need the excessive time interval. Hence, image processing and machine learning models can be employed for the detection of plant diseases. In this project, we have described the technique for the detection of plant diseases with the help of their leaves pictures. Image processing is a branch of signal processing which can extract the image properties or useful information from the image. Machine learning is a sub part of artificial intelligence which works automatically or give instructions to do a particular task. The main aim of machine learning is to understand the training data and fit that training data into models that should be useful to the people. So it can assist in good decisions making and predicting the correct output using the large amount of training data. The color of leaves, amount of damage to leaves, area of the leaf, texture parameters are used for classification. In this project we have analyzed different image parameters or features to identifying different plant leaves diseases to achieve the best accuracy. Previously plant disease detection is done by visual inspection of the leaves or some chemical processes by experts. For doing so, a large team of experts as well as continuous observation of plant is needed, which costs high when we do with large farms. In such conditions, the recommended system proves to be helpful in monitoring large fields of crops. Automatic detection of the diseases by simply seeing the symptoms on the plant leaves makes it easier as well as cheaper. The proposed solution for plant disease detection is computationally less expensive and requires less time for prediction than other deep learning based approaches since it uses statistical machine learning and image processing algorithm.

1. **Literature Survey**

**2.1. Plant Disease Detection Using Image Processing (2015) [1]**

In 2015, S. Khirade et Al. tackled the problem of plant disease detection using digital image processing techniques and back propagation neural network (BPNN). Authors have elaborated different techniques for the detection of plant disease using the images of leaves. They have implemented Otsu’s thresholding followed by boundary detection and spot detection algorithm to segment the infected part in leaf. After that they have extracted the features such as color, texture, morphology, edges etc. for classification of plant disease. BPNN is used for classification i.e. to detect the plant disease.

**2.2. Plant Disease Identification: A Comparative Study (2017) [2]**

Shiroop Madiwalar and Medha Wyawahare analyzed different image processing approaches for plant disease detection in their research. Authors analyzed the color and texture features for the detection of plant disease. They have experimented their algorithms on the dataset of 110 RGB images. The features extracted for classification were mean and standard deviation of RGB and YCbCr channels, grey level co-occurrence matrix (GLCM) features, the mean and standard deviation of the image convolved with Gabor filter. Support vector machine classifier was used for classification. Authors concluded that GCLM features are effective to detect normal leaves. Whereas color features and Gabor filter features are considered as best for detecting anthracnose affected leaves and leaf spot respectively. They have achieved highest accuracy of 83.34% using all the extracted features.

**2.3. Plant Disease Detection using Hyperspectral Imaging (2017) [3]**

Peyman Moghadam et Al. demonstrated the application of hyperspectral imaging in plant disease detection task. visible and near-infrared (VNIR) and short-wave infrared (SWIR) spectrums were used in this research. Authors have used k-means clustering algorithm in spectral domain for the segmentation of leaf. They have proposed a novel grid removal algorithm to remove the grid from hyperspectral images. Authors have achieved the accuracy of 83% with vegetation indices in VNIR spectral range and 93% accuracy with full spectrum. Though the proposed method achieved higher accuracy, it requires the hyperspectral camera with 324 spectral bands so the solution becomes too costly.

**2.4. Image based Plant Disease Detection in Pomegranate Plant for Bacterial Blight (2019) [4]**

Sharath D. M. et Al. developed the Bacterial Blight detection system for Pomegranate plant by using features such as color, mean, homogeneity, SD, variance, correlation, entropy, edges etc. Authors have implemented grab cut segmentation for segmenting the region of interest in the image. Canny edge detector was used to extract the edges from the images. Authors have successfully developed a system which can predict the infection level in the fruit.

**2.5. Plant Disease Detection Using CNN (2020) [5]**

Garima Shrestha et Al. deployed the convolutional neural network to detect the plant disease. Authors have successfully classified 12 plant diseases with 88.80% accuracy. The dataset of 3000 high resolution RGB images were used for experimentation. The network has 3 blocks of convolution and pooling layers. This makes the network computationally expensive. Also the F1 score of the model is 0.12 which is very low because of higher number of false negative predictions.

1. **Methodology**

**3.1. Dataset**

For this project we have used public dataset for plant leaf disease detection called PlantVillage curated by Sharada P. Mohanty et Al. [6]. The dataset consists of 87000 RGB images of healthy and unhealthy plant leaves having 38 classes out of which We have selected only 25 classes for experimentation of our algorithm These classes are shown in Table 1.

|  |  |  |
| --- | --- | --- |
| **Plant** | **Disease Name** | **No. of images** |
| **Apple** | Healthy | 2008 |
| Diseased: Scab | 2016 |
| Diseased: Black rot | 1987 |
| Diseased: Cedar apple rust | 1760 |
| **Corn** | Healthy | 1859 |
| Diseased: Cercospora leaf spot Gray leaf spot | 1642 |
| Diseased: Common rust | 1907 |
| Diseased: Northern Leaf Blight | 1908 |
| **Grapes** | Healthy | 1692 |
| Diseased: Black rot | 1888 |
| Diseased: Esca (Black Measles) | 1920 |
| Diseased: Leaf blight (Isariopsis Leaf Spot) | 1722 |
| **Potato** | Healthy | 1824 |
| Diseased: Early blight | 1939 |
| Diseased: Late blight | 1939 |
| **Tomato** | Healthy | 1926 |
| Diseased: Bacterial spot | 1702 |
| Diseased: Early blight | 1920 |
| Diseased: Late blight | 1851 |
| Diseased: Leaf Mold | 1882 |
| Diseased: Septoria leaf spot | 1745 |
| Diseased: Spider mites Two-spotted spider mite | 1741 |
| Diseased: Target Spot | 1827 |
| Diseased: Tomato Yellow Leaf Curl Virus | 1961 |
| Diseased: Tomato mosaic virus | 1790 |

Table 1: Dataset Specifications

Some samples from the datset are shown in Fig. 1.



Fig. 1. Sample images in the dataset [1]

**3.2. Data preprocessing and feature extraction**

Data preprocessing is important task in any computer vision based system. Fig. 2 illustrates the preprocessing steps for each image. To get precise results, some background noise should be removed before extraction of features. So first the RGB image is converted to greyscale and then Gaussian filter is used for smoothening of the image. Then to binaries the image, Otsu’s thresholding algorithm is implemented. Then morphological transform is applied on binarised image to close the small holes in the foreground part. Now after foreground detection, the bitwise AND operation on binarised image and original color image is performed to get RGB image of segmented leaf. Now after image segmentation shape, texture and color features are extracted from the image.

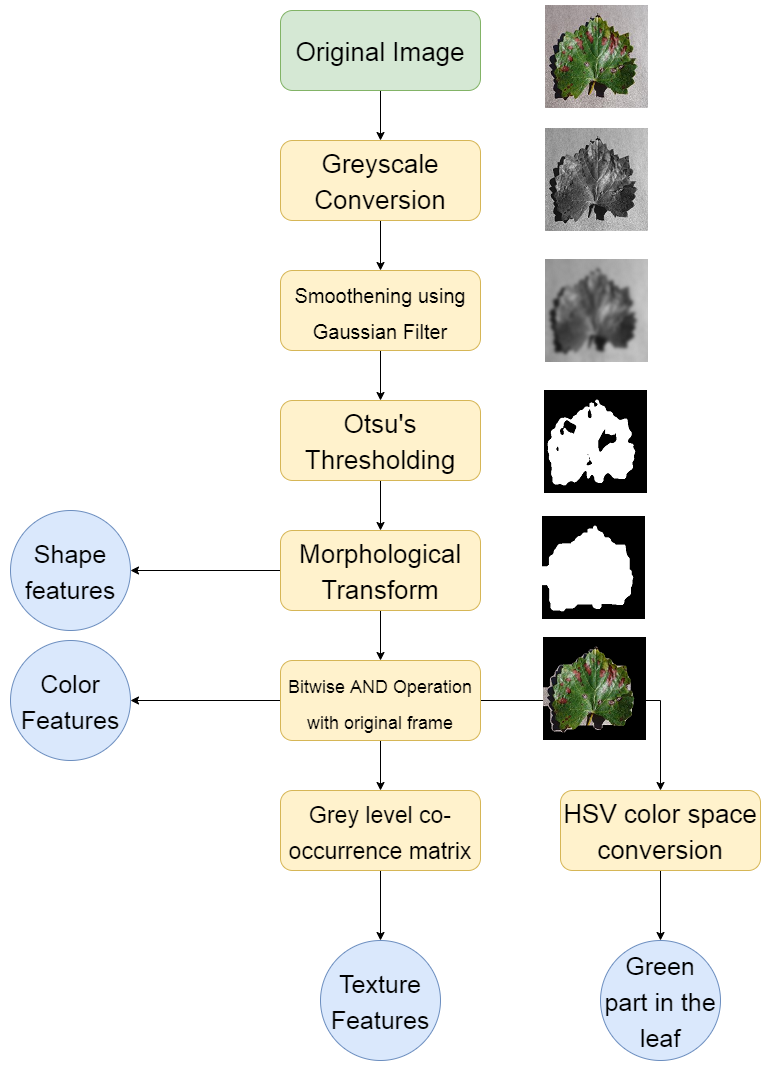


Fig. 2. Steps for data preprocessing and feature extraction

By using contours, area of the leaf and perimeter of the leaf is calculated. Contours are the line that joins all the points along the edges of objects having same color or intensity. Mean and standard deviation of each channel in RGB image is also estimated. To obtain amount of green color in the image, image is first converted to HSV color space and we have calculated the ratio of number of pixels having pixel intensity of hue (H) channel in between 30 and 70 and total number of pixels in one channel. Non green part of image is calculated by subtracting green color part from 1. After extracting color features from the image, we have extracted texture features from grey level co-occurrence matrix (GLCM) of the image [7]. GLCM is the spacial relationship of pixels in the image. Extracting texture features from GCLM is one of the tradition method in computer vision. We have extracted following features from GCLM:

* Contrast
* Dissimilarity
* Homogeneity
* Energy
* Correlation

After extracting all the features from all the images in the dataset, feature selection task is performed.

**3.3. Feature selection**

Feature selection is an important step in all machine learning problems. In this project we are selecting the features on the basis of correlation of variables with target variable. Fig. 3 shows the correlation of each variable with each other for apple dataset. The correlation of feature green part of leaf (F1) and green part of leaf (F2) is very high (1) which means both variables are dependent on each other. So we have dropped one of them (F2). Now for apple disease prediction, less correlated features such as green channel mean, red channel standard deviation, blue channel standard deviation, dissimilarity (f5) and correlation (f8) will not contribute too much in model development. So we have dropped these variables also. After feature selection, the data is now parsed to machine learning classifiers to find the patterns in the data.

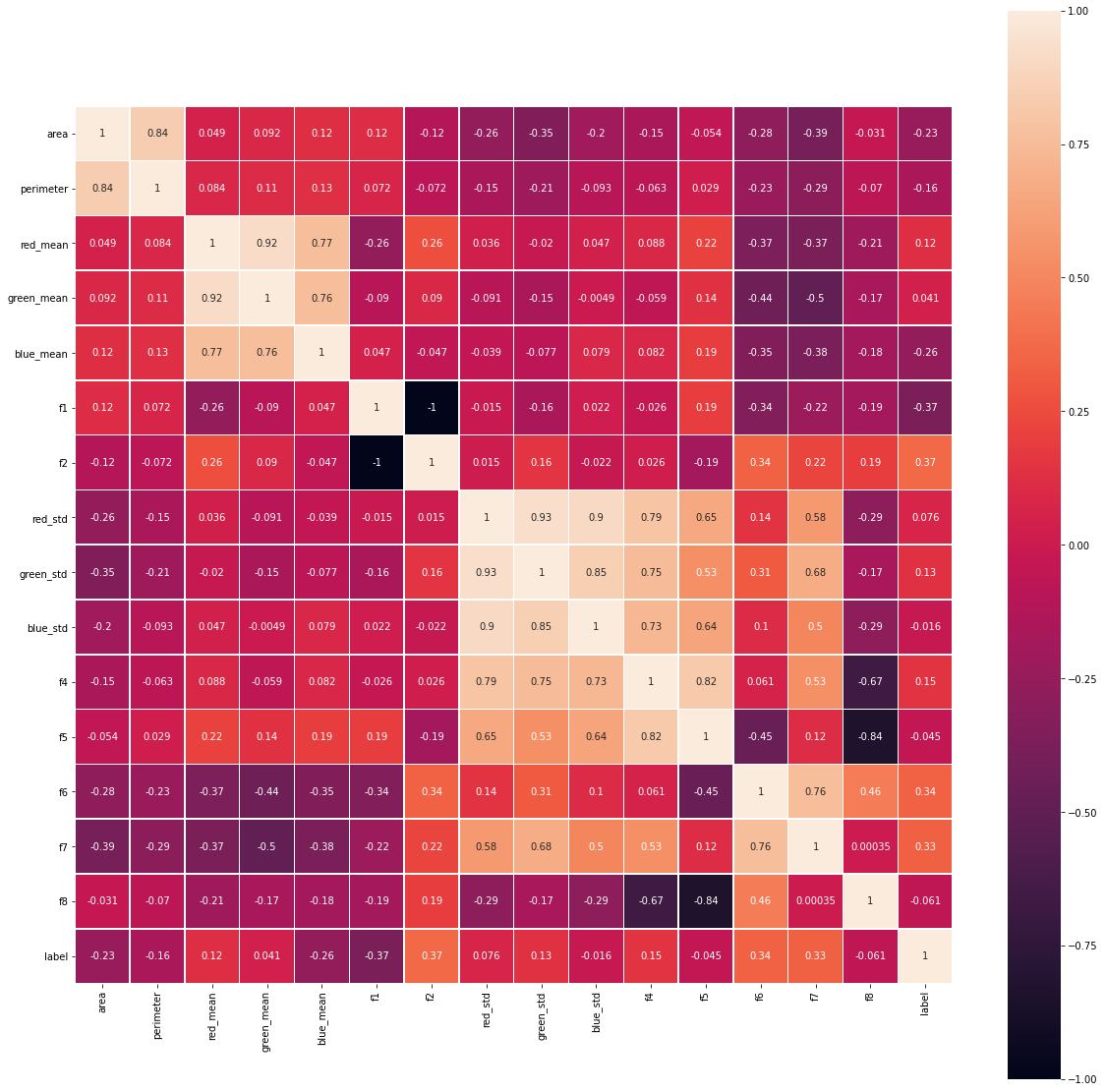


Fig. 3. Correlation plot for Apple dataset

**3.4. Classification Algorithm**

Random forest classifier has been used for classification or detection task. It is the part of ensemble learning, where the output is predicted from multiple base estimators [8]. Generally, to achieve higher accuracies, decision trees are used. But they are prone to overfitting problems. So to overcome this issue, random forest classifier is used which is a combination of multiple decision trees. Each tree is trained by using different subsets of the whole dataset, this can reduce the overfitting and improves the accuracy of the classifier. We have splitted the dataset into train set (80%) for fitting the model and test set (20%) for validation. K-fold cross validation technique is implemented to find the accuracy score. This method can find the accuracy on whole dataset without any bias. After fitting the data, f1 score, precision, recall, accuracy has been calculated from test data to analyze the performance of the model. ROC curve and confusion matrix was plotted to analyze false positives and false negatives.

1. **Performance Analysis**

Table 2 shows the performance matrices for each model developed for each of the plant. We can observe that the accuracy scores are nearly equal to f1 scores. This is because of balanced number of false negative and false positive predictions. This is considered as best case for any machine learning algorithm. The average accuracy was 93%.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Plant** | **Accuracy** | **Precision** | **Recall** | **F1 Score** | **No. of Testing datapoints** |
| **Apple** | 0.91 | 0.91 | 0.91 | 0.91 | 1549 |
| **Corn** | 0.94 | 0.94 | 0.94 | 0.94 | 1459 |
| **Grapes** | 0.95 | 0.95 | 0.95 | 0.95 | 1433 |
| **Potato** | 0.98 | 0.98 | 0.98 | 0.98 | 1135 |
| **Tomato** | 0.87 | 0.87 | 0.87 | 0.87 | 3661 |

Table 2. Performance matric for all models.

Fig. 4 shows the confusion matrices for each of the model. With the help of confusion matrices, number of false negatives, false positives, true predictions can be analyzed. Fig. 5 shows the receiver operating characteristic (ROC) curve for each of the model. An ROC curve is a graph showing the performance of a classification model at all classification thresholds. It depends upon two parameters, true positive rate and false positive rate.

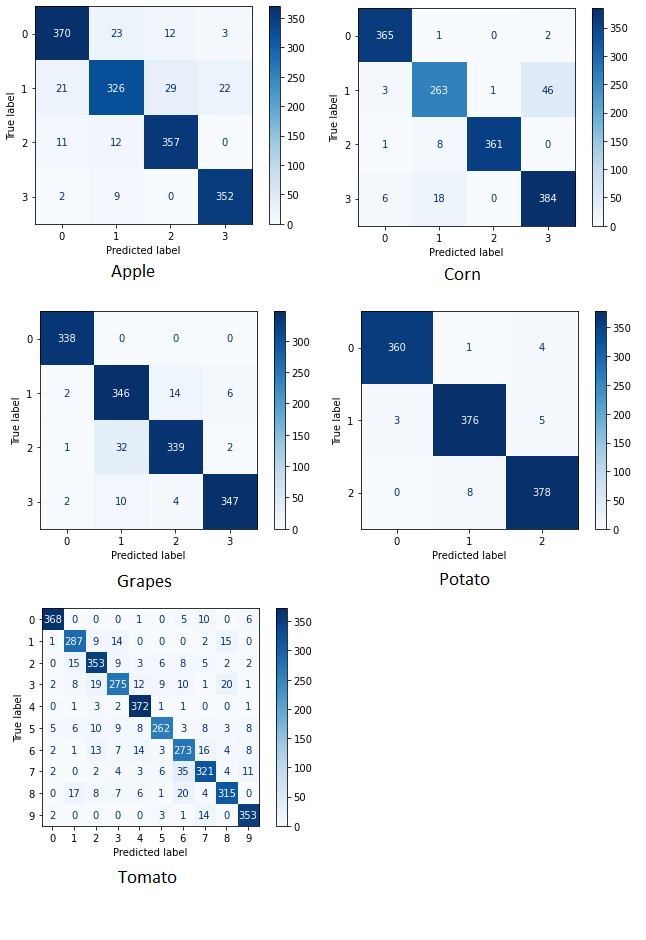


Fig. 4. Confusion matrices for all the models

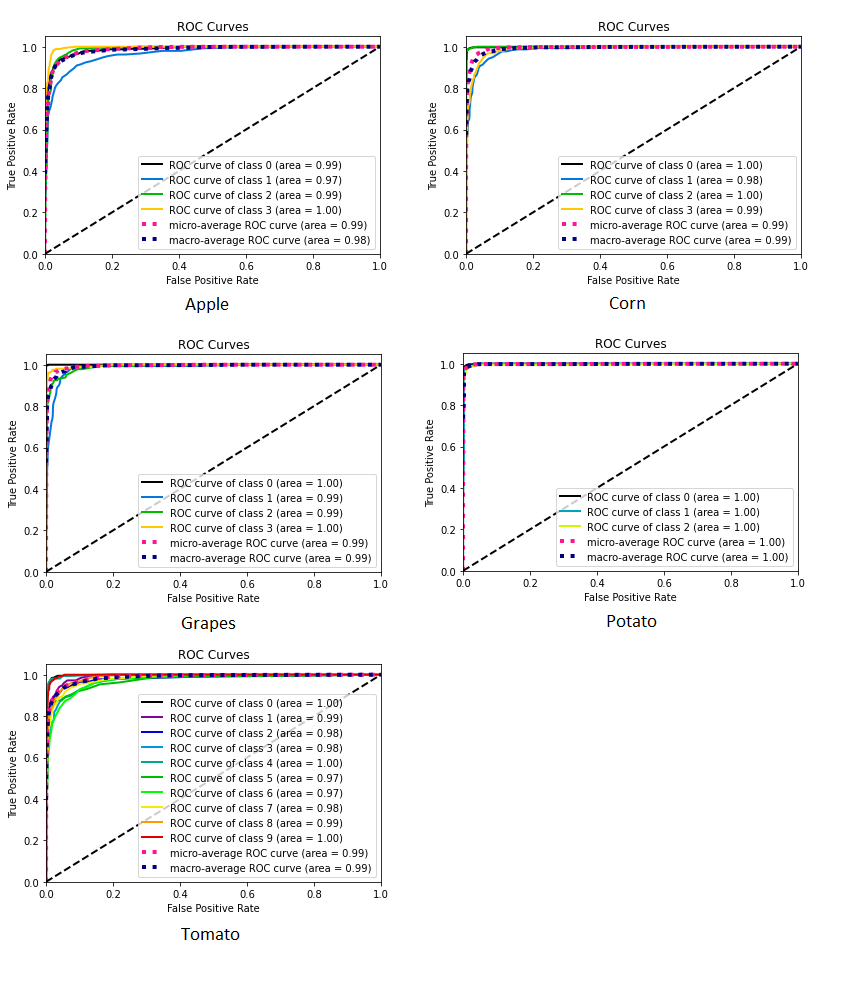


Fig. 5. ROC curves for all the models.

1. **Code**

**5.1. Image Preprocessing and Feature extraction code:**

import os

import cv2

import numpy as np

import pandas as pd

from skimage.feature import local\_binary\_pattern , greycomatrix , greycoprops

from matplotlib import pyplot as plt

%matplotlib inline

from os import listdir

from os.path import isfile, join

def create\_empty\_df():

    df = pd.DataFrame()

    df['area'] = None

    df['perimeter'] = None

    df['red\_mean'] = None

    df['green\_mean'] = None

    df['blue\_mean'] = None

    df['f1'] = None

    df['f2'] = None

    df['red\_std'] = None

    df['green\_std'] = None

    df['blue\_std'] = None

    df['f4'] = None

    df['f5'] = None

    df['f6'] = None

    df['f7'] = None

    df['f8'] = None

    df['label'] = None

    return df

def feature\_extractor(filename):

    '''

    input params:

    filename : path of the file that we want to process

    Output params:

    l : Feature vector

    '''

    try:

        main\_img = cv2.imread(filename)

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

    except:

        return "Invalid"

    #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((25,25),np.uint8)

    closing = cv2.morphologyEx(im\_bw\_otsu, cv2.MORPH\_CLOSE, kernel)

    #Shape features

    contours, \_ = cv2.findContours(closing,cv2.RETR\_TREE,cv2.CHAIN\_APPROX\_SIMPLE)

    cnt = contours[0]

    M = cv2.moments(cnt)

    area = cv2.contourArea(cnt)

    if area==0:

        return "Invalid"

    perimeter = cv2.arcLength(cnt,True)

    current\_frame = main\_img

    filtered\_image = closing/255

    #Elementwise Multiplication of range bounded filtered\_image with current\_frame

    current\_frame[0:current\_frame.shape[0], 0:current\_frame.shape[1], 0] = np.multiply(current\_frame[0:current\_frame.shape[0], 0:current\_frame.shape[1], 0], filtered\_image) #B channel

    current\_frame[0:current\_frame.shape[0], 0:current\_frame.shape[1], 1] = np.multiply(current\_frame[0:current\_frame.shape[0], 0:current\_frame.shape[1], 1], filtered\_image) #G channel

    current\_frame[0:current\_frame.shape[0], 0:current\_frame.shape[1], 2] = np.multiply(current\_frame[0:current\_frame.shape[0], 0:current\_frame.shape[1], 2], filtered\_image) #R channel

    img = current\_frame

    #Color features

    #a higher number in order to see the actual color.

    red\_channel = img[:,:,0]

    green\_channel = img[:,:,1] #show the intensities of green channe

    blue\_channel = img[:,:,2]

    red\_mean = np.mean(red\_channel)

    green\_mean = np.mean(green\_channel)

    blue\_mean = np.mean(blue\_channel)

    #standard deviation for colour feature from the image.

    red\_std = np.std(red\_channel)

    green\_std = np.std(green\_channel)

    blue\_std = np.std(blue\_channel)

    #amt.of green color in the image

    gr = cv2.cvtColor(img, cv2.COLOR\_BGR2HSV)

    boundaries = [([30,0,0],[70,255,255])]

    for (lower, upper) in boundaries:

        mask = cv2.inRange(gr, (36, 0, 0), (70, 255,255))

        ratio\_green = cv2.countNonZero(mask)/(img.size/3)

        f1=np.round(ratio\_green, 2)

    #amt. of non green part of the image

    f2=1-f1

    #Texture features using grey level cooccurance matrix

    img=cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

    g=greycomatrix(img, [1], [0, np.pi/4, np.pi/2, 3\*np.pi/4])

    #with the help of glcm find the contrast

    contrast = greycoprops(g, 'contrast')

    f4=contrast[0][0]+contrast[0][1]+contrast[0][2]+contrast[0][3]

    #[0][3] represent no. of times grey level 3 appears at the right of 0

    #with the help of glcm find the dissimilarity

    dissimilarity = greycoprops(g, prop='dissimilarity')

    f5=dissimilarity[0][0]+dissimilarity[0][1]+dissimilarity[0][2]+dissimilarity[0][3]

    #with the help of glcm find the homogeneity

    homogeneity = greycoprops(g, prop='homogeneity')

    f6=homogeneity[0][0]+homogeneity[0][1]+homogeneity[0][2]+homogeneity[0][3]

    #with the help of glcm find the energy.

    energy = greycoprops(g, prop='energy')

    f7=energy[0][0]+energy[0][1]+energy[0][2]+energy[0][3]

    #with the help of glcm find the correlation

    correlation = greycoprops(g,prop= 'correlation')

    f8=correlation[0][0]+correlation[0][1]+correlation[0][2]+correlation[0][3]

    l = [area, perimeter, red\_mean, green\_mean, blue\_mean,

         f1, f2, red\_std, green\_std, blue\_std,

        f4,f5,f6,f7,f8]

    return l

def process\_folder(folderpath,df\_f,label\_f):

    '''

    input params:

    folderpath : Path of the folder that we want to process

    df\_f = dataframe for specific disease

    label\_f : label corresponding to the specific disease

    Output params:

    df\_f = Dataframe consisting processed vectors

    '''

    imagelist = os.listdir(folderpath)

    for image in imagelist:

        imagepath = os.path.join(folderpath, image)

        im\_feature = feature\_extractor(imagepath)

        if im\_feature == "Invalid":

            continue

        im\_feature.append(label\_f)

        df\_f.loc[len(df\_f)] = im\_feature

        if len(df\_f)%500 ==0:

            print(len(df\_f))

    return df\_f

def process\_plant(folderpaths, labels, savepath):

    '''

    input params:

    folderpaths : List of the folderpaths for specific Plant

    labels : List of labels

    savepath : Path to export datasheet

    Output params:

    None

    '''

    datasheet = create\_empty\_df()

    for i in range(len(folderpaths)):

        datasheet = process\_folder(folderpaths[i],datasheet,labels[i])

    datasheet.to\_excel(savepath)

    return None

folderpaths = ['/content/drive/MyDrive/Plant Disease Detection /Raw\_Dataset/New Plant Diseases Dataset(Augmented)/New Plant Diseases Dataset(Augmented)/train/Apple\_\_\_healthy',

               '/content/drive/MyDrive/Plant Disease Detection /Raw\_Dataset/New Plant Diseases Dataset(Augmented)/New Plant Diseases Dataset(Augmented)/train/Apple\_\_\_Apple\_scab',

               '/content/drive/MyDrive/Plant Disease Detection /Raw\_Dataset/New Plant Diseases Dataset(Augmented)/New Plant Diseases Dataset(Augmented)/train/Apple\_\_\_Black\_rot',

               '/content/drive/MyDrive/Plant Disease Detection /Raw\_Dataset/New Plant Diseases Dataset(Augmented)/New Plant Diseases Dataset(Augmented)/train/Apple\_\_\_Cedar\_apple\_rust'

]

labels = [0,1,2,3]

savepath = '/content/drive/MyDrive/Plant Disease Detection /Processed\_data&models/Apple/dataset.xlsx'

process\_plant(folderpaths, labels, savepath)

**5.2. Data Cleaning and Machine Learning Model training and testing**

# EDA MOdule importing

import shutil

shutil.copy('/content/drive/MyDrive/DA\_Library/EDA.py','EDA.py')

import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

import seaborn as sns

import EDA as eda

raw\_data = eda.readfile('/content/drive/MyDrive/Plant Disease Detection /Processed\_data&models/Apple/dataset\_apple.xlsx')

raw\_data.drop(['Unnamed: 0'],axis = 1, inplace=True)

eda.correlation(raw\_data)

eda.correlationlist(raw\_data)

cleaned\_data = raw\_data.drop(['green\_mean', 'red\_std', 'blue\_std', 'f5', 'f8','f1'],axis = 1, inplace=False)

cleaned\_data = eda.removenullrows(cleaned\_data)

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

from sklearn.metrics import accuracy\_score

from sklearn.preprocessing import StandardScaler

from sklearn.decomposition import PCA

from sklearn.svm import SVC

from sklearn.metrics import roc\_curve, auc

from sklearn.model\_selection import KFold

df = cleaned\_data.reset\_index()

X = df.drop(['index','label'],axis = 1, inplace=False)

y = df['label']

print(X.shape)

print(y.shape)

k = 5

kf = KFold(n\_splits=k, random\_state=9, shuffle = True)

from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier(random\_state = 50,n\_estimators = 50,max\_samples = 0.7)

acc\_score = []

for train\_index , test\_index in kf.split(X):

    X\_train , X\_test = X.iloc[train\_index,:],X.iloc[test\_index,:]

    y\_train , y\_test = y[train\_index] , y[test\_index]

    model = RandomForestClassifier(random\_state = 50,n\_estimators = 50,max\_samples = 0.7)

    model.fit(X\_train,y\_train)

    pred\_values = model.predict(X\_test)            # classification

    acc = accuracy\_score(pred\_values , y\_test)     # Classification

    #acc = model.score(X\_test,y\_test)                # Regression

    acc\_score.append(acc)

avg\_acc\_score = sum(acc\_score)/k

print('Score of each fold - {}'.format(acc\_score))

print('Avg Score : {}'.format(avg\_acc\_score))

**5.3. Evaluation**

!pip install scikit-plot

import scikitplot as skplt

import matplotlib.pyplot as plt

y\_true = y\_test

y\_probas = model.predict\_proba(X\_test)

skplt.metrics.plot\_roc\_curve(y\_true, y\_probas)

plt.show()

from sklearn.metrics import plot\_confusion\_matrix

y\_pred = model.predict(X\_test)

plot\_confusion\_matrix(model, X\_test, y\_test, values\_format = 'd',cmap = 'Blues',display\_labels = ['0','1','2','3'])

from sklearn.metrics import classification\_report

print(classification\_report(y\_test, y\_pred, [0,1,2,3]))

from sklearn.ensemble import RandomForestClassifier

lm = RandomForestClassifier(random\_state = 50,n\_estimators = 50,max\_samples = 0.7)

lm.fit(X,y)

print('Training Score: ',lm.score(X,y))

lm.feature\_importances\_

import pickle

filename = '/content/drive/MyDrive/Plant Disease Detection /Processed\_data&models/Apple/Results/Applemodel\_V1.sav'

pickle.dump(lm, open(filename, 'wb'))

**5.4. Inference**

import numpy as np

def apple\_p(feature\_vector,model):

    processed\_vector = np.array(feature\_vector).reshape(1, -1)

    output = model.predict(processed\_vector)

    output = int(output)

    label\_dict = {0 :'Apple\_\_\_healthy', 1: 'Apple\_\_\_Apple\_scab', 2: 'Apple\_\_\_Black\_rot', 3: 'Apple\_\_\_Cedar\_apple\_rust'}

    output = label\_dict[output]

    return output

def corn\_p(feature\_vector,model):

    processed\_vector = np.array(feature\_vector).reshape(1, -1)

    output = model.predict(processed\_vector)

    output = int(output)

    label\_dict = {0: 'Corn\_(maize)\_\_\_healthy',

    1: 'Corn\_(maize)\_\_\_Cercospora\_leaf\_spot Gray\_leaf\_spot',

    2: 'Corn\_(maize)\_\_Common\_rust',

    3: 'Corn\_(maize)\_\_\_Northern\_Leaf\_Blight'}

    output = label\_dict[output]

    return output

def grapes\_p(feature\_vector,model):

    processed\_vector = np.array(feature\_vector).reshape(1, -1)

    output = model.predict(processed\_vector)

    output = int(output)

    label\_dict = {0 : 'Grape\_\_\_healthy',

    1 : 'Grape\_\_\_Black\_rot',

    2 : 'Grape\_\_\_Esca\_(Black\_Measles)',

    3 : 'Grape\_\_\_Leaf\_blight\_(Isariopsis\_Leaf\_Spot)'}

    output = label\_dict[output]

    return output

def potato\_p(feature\_vector,model):

    processed\_vector = np.array(feature\_vector).reshape(1, -1)

    output = model.predict(processed\_vector)

    output = int(output)

    label\_dict = {0: 'Potato\_\_\_healthy',

    1: 'Potato\_\_\_Early\_blight',

    2: 'Potato\_\_\_Late\_blight'}

    output = label\_dict[output]

    return output

def tomato\_p(feature\_vector,model):

    processed\_vector = np.array(feature\_vector).reshape(1, -1)

    output = model.predict(processed\_vector)

    output = int(output)

    label\_dict = {0 : 'Tomato\_\_\_healthy',

    1 : 'Tomato\_\_\_Bacterial\_spot',

    2 : 'Tomato\_\_\_Early\_blight',

    3 : 'Tomato\_\_\_Late\_blight',

    4 : 'Tomato\_\_\_Leaf\_Mold',

    5 : 'Tomato\_\_\_Septoria\_leaf\_spot',

    6 : 'Tomato\_\_\_Spider\_mites Two-spotted\_spider\_mite',

    7 : 'Tomato\_\_\_Target\_Spot',

    8 : 'Tomato\_\_\_Tomato\_Yellow\_Leaf\_Curl\_Virus',

    9 : 'Tomato\_\_\_Tomato\_mosaic\_virus'}

    output = label\_dict[output]

    return output

**5.4. API**

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

# Plant disease detection

# API V2

# Version 1.1

# API Main Script

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

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

# Importing required Libraries and modules

import imageprocess

import predictor

import pickle

import os

from flask import Flask

import flask

from flask import request

from flask import render\_template

import numpy as np

import cv2

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

app = Flask(\_\_name\_\_)

applemodelpath = 'models/Applemodel\_V1.sav'

apple\_model = pickle.load(open(applemodelpath, 'rb'))

cornmodelpath = 'models/cornmodel\_V1.sav'

corn\_model = pickle.load(open(cornmodelpath, 'rb'))

grapesmodelpath = 'models/grapesmodel\_V1.sav'

grapes\_model = pickle.load(open(grapesmodelpath, 'rb'))

potatomodelpath = 'models/potatomodel\_V1.sav'

potato\_model = pickle.load(open(potatomodelpath, 'rb'))

tomatomodelpath = 'models/Tomatomodel\_V1.sav'

tomato\_model = pickle.load(open(tomatomodelpath, 'rb'))

@app.route("/")

def home():

    version = "1.1"

    return render\_template('index.html', version1 = version)

@app.route("/predict", methods = ['GET', 'POST'])

def submit():

    imagefile = flask.request.files["data\_file"].read()

    dname = request.form.get('Name')

    dname = str(dname)

    response = dname[0]

    npimg = np.frombuffer(imagefile, np.uint8)

    # convert numpy array to image

    img = cv2.imdecode(npimg,cv2.IMREAD\_COLOR)

    f\_vector = imageprocess.feature\_extractor(img)

    if response=='n':

        res = "Please select the appropriate plant from the list"

        return '<h1>'+res+'</h1>'

    if response=='a':

        p\_vector = [f\_vector['area'],f\_vector['perimeter'],f\_vector['red\_mean'],f\_vector['blue\_mean'],f\_vector['f2'],f\_vector['green\_std'],

        f\_vector['f4'],f\_vector['f6'],f\_vector['f7']]

        res = predictor.apple\_p(p\_vector,apple\_model)

        return '<h1>'+res+'</h1>'

    if response=='c':

        p\_vector = [f\_vector['red\_mean'], f\_vector['green\_mean'], f\_vector['blue\_mean'], f\_vector['f2'], f\_vector['red\_std'], f\_vector['blue\_std'],

        f\_vector['f7'], f\_vector['f8']]

        res = predictor.corn\_p(p\_vector,corn\_model)

        return '<h1>'+res+'</h1>'

    if response=='g':

        p\_vector = [f\_vector['area'], f\_vector['perimeter'], f\_vector['red\_mean'], f\_vector['green\_mean'], f\_vector['blue\_mean'], f\_vector['f2'],

           f\_vector['red\_std'], f\_vector['green\_std'], f\_vector['blue\_std'], f\_vector['f4'], f\_vector['f5'], f\_vector['f6'], f\_vector['f7'], f\_vector['f8']]

        res = predictor.grapes\_p(p\_vector,grapes\_model)

        return '<h1>'+res+'</h1>'

    if response=='p':

        p\_vector = [f\_vector['area'], f\_vector['perimeter'], f\_vector['green\_mean'], f\_vector['blue\_mean'], f\_vector['f2'], f\_vector['red\_std'],

           f\_vector['green\_std'], f\_vector['blue\_std'], f\_vector['f4'], f\_vector['f5'], f\_vector['f7'], f\_vector['f8']]

        res = predictor.potato\_p(p\_vector,potato\_model)

        return '<h1>'+res+'</h1>'

    if response=='t':

        del f\_vector["f1"]

        p\_vector = list(f\_vector.values())

        res = predictor.tomato\_p(p\_vector,tomato\_model)

        return '<h1>'+res+'</h1>'

if \_\_name\_\_ == "\_\_main\_\_":

    app.run()

1. **Outputs**

We have developed a flask based web application for detecting the plant disease and deployed it on heroku (free cloud hosting server). Fig 6 shows the homepage of deployed web application and Fig 7 shows the input images and their corresponding predictions made by our system. It shows that the system successfully detected the disease of leaf.

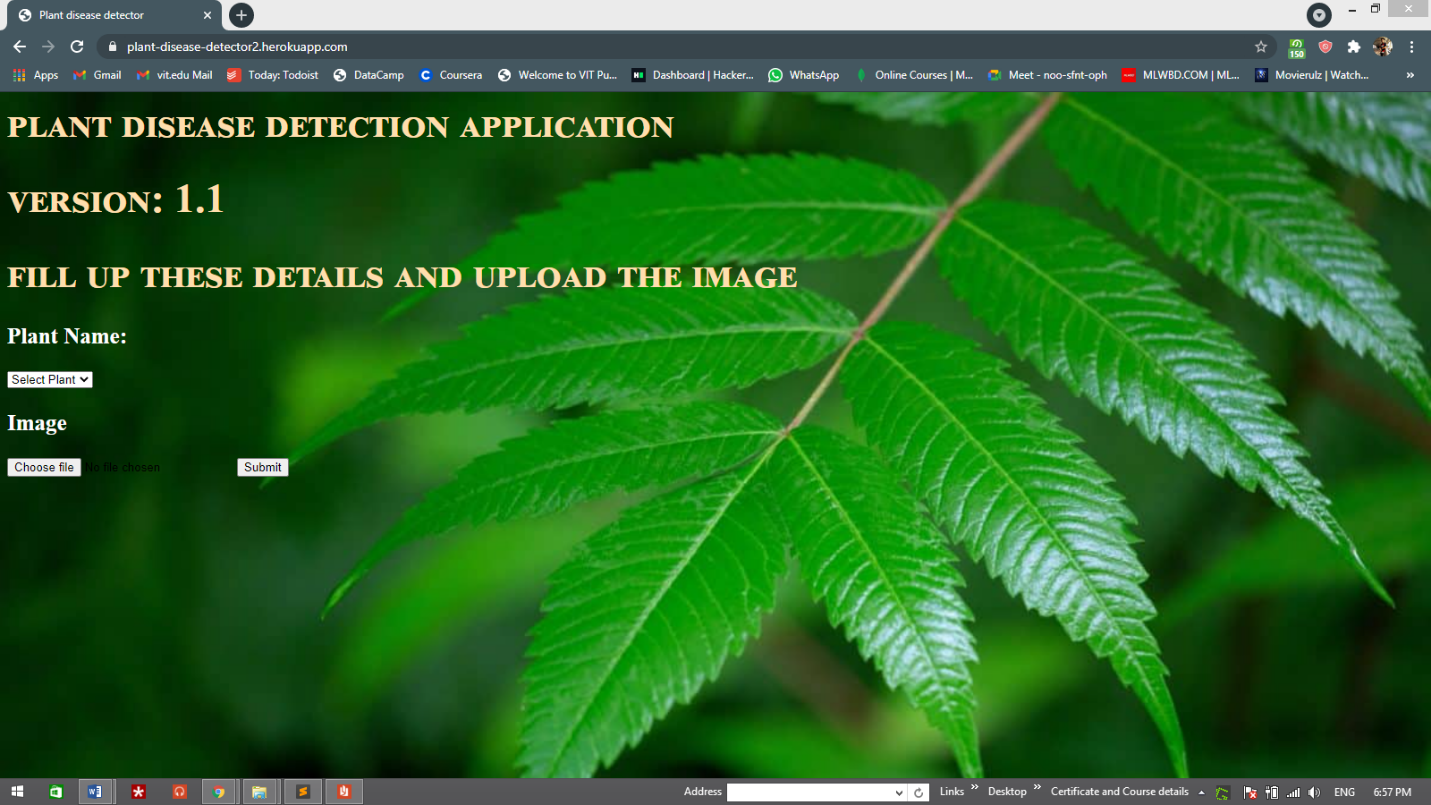


Fig. 6. Homepage of deployed API

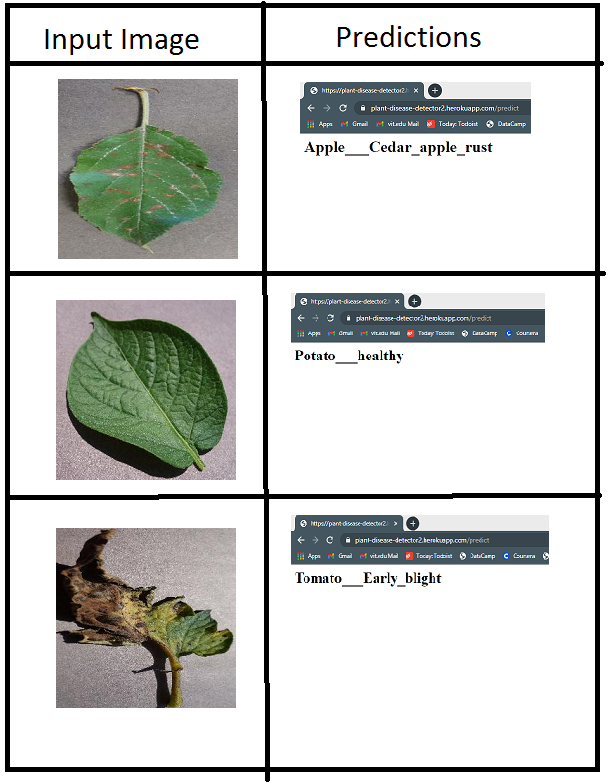


Fig. 7. Images and Outputs generated by system

**7. Applications**

**7.1. Intelligent Robot for plant disease detection:**

We can deploy a intelligent robot vehicle with high end processor attached to it for real time plant disease detection. This system can detect the diseased plants in the agricultural site. Even we can automate the process of spreading the fertilizers by using such robots. Our proposed algorithm is computationally inexpensive, so it can detect the plant disease in efficient manner

**7.2. Website or application for testing the plant disease:**

Sometimes it happens that the farmer also could not identify the disease of the plant. So they need an expert advice. So we can deploy a website which can detect the plant disease based on images captured and uploaded by farmer and can give suggestions or can suggest some fertilizers based on detected disease.

**8. Conclusion**

We have successfully developed a computer vision based system for plant disease detection with average 93% accuracy and 0.93 F1 score. Also the proposed system is computationally efficient because of the use of statistical image processing and machine learning model. Table 3 illustrates the overall benefits of our system over the other approaches.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Author** | **S. Khirade et Al. (2015)** | **Shiroop Madiwalar et Al. (2017)** | **Peyman Moghadam et Al. (2017)** | **Sharath D. M. et Al. (2019)** | **Garima Shrestha et Al. (2020)** | **Proposed Method** |
| **Algorithm** | Digital image processing and BPNN | Digital Image processing and SVM | Hyperspectral imaging and SVM | Digital image processing | CNN | Digital image processing and random forest classifier |
| **Accuracy** | - | 83.34% | 93% | - | 88.80% | 93% |
| **Computationally efficient** | ✘ | ✔ | ✘ | ✔ | ✘ | ✔ |
| **Specialized hardware requirement** | ✘ | ✘ | ✔ | ✘ | ✘ | ✘ |

Table 3. Comparison of Proposed system with other existing systems.

We can observe that our technique is accurate and efficient compared with other systems. Also it won't require a specialized hardware, makes it cost effective solution.

**9. References**

1. S. D. Khirade and A. B. Patil, "Plant Disease Detection Using Image Processing," 2015 International Conference on Computing Communication Control and Automation, 2015, pp. 768-771, doi: 10.1109/ICCUBEA.2015.153.
2. S. C. Madiwalar and M. V. Wyawahare, "Plant disease identification: A comparative study," 2017 International Conference on Data Management, Analytics and Innovation (ICDMAI), 2017, pp. 13-18, doi: 10.1109/ICDMAI.2017.8073478.
3. P. Moghadam, D. Ward, E. Goan, S. Jayawardena, P. Sikka and E. Hernandez, "Plant Disease Detection Using Hyperspectral Imaging," 2017 International Conference on Digital Image Computing: Techniques and Applications (DICTA), 2017, pp. 1-8, doi: 10.1109/DICTA.2017.8227476.
4. S. D.M., Akhilesh, S. A. Kumar, R. M.G. and P. C., "Image based Plant Disease Detection in Pomegranate Plant for Bacterial Blight," 2019 International Conference on Communication and Signal Processing (ICCSP), 2019, pp. 0645-0649, doi: 10.1109/ICCSP.2019.8698007.
5. G. Shrestha, Deepsikha, M. Das and N. Dey, "Plant Disease Detection Using CNN," 2020 IEEE Applied Signal Processing Conference (ASPCON), 2020, pp. 109-113, doi: 10.1109/ASPCON49795.2020.9276722.
6. Mohanty SP, Hughes DP and Salathé M (2016) Using Deep Learning for Image-Based Plant Disease Detection. Front. Plant Sci. 7:1419. doi: 10.3389/fpls.2016.01419
7. R. M. Haralick, K. Shanmugam and I. Dinstein, "Textural Features for Image Classification," in IEEE Transactions on Systems, Man, and Cybernetics, vol. SMC-3, no. 6, pp. 610-621, Nov. 1973, doi: 10.1109/TSMC.1973.4309314.
8. Breiman, L. Random Forests. Machine Learning 45, 5–32 (2001). https://doi.org/10.1023/A:1010933404324