

Configuration Manual

MSc Research Project
Data Analytics

Sai Rajasekhar Reddy Evuri
Student ID: x20250151

School of Computing
National College of Ireland

Supervisor: Abubakr Siddig

**National College of Ireland
Project Submission Sheet
School of Computing**



Student Name:	Sai Rajasekhar Reddy Evuri
Student ID:	x20250151
Programme:	Data Analytics
Year:	2022
Module:	MSc Research Project
Supervisor:	Abubakr Siddig
Submission Due Date:	20/12/2018
Project Title:	Configuration Manual
Word Count:	XXX
Page Count:	19

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Sai Rajasekhar Reddy Evuri
Date:	19th September 2022

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	<input type="checkbox"/>
Attach a Moodle submission receipt of the online project submission , to each project (including multiple copies).	<input type="checkbox"/>
You must ensure that you retain a HARD COPY of the project , both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	<input type="checkbox"/>

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Sai Rajasekhar Reddy Evuri
x20250151

1 Introduction

It describes all requirements required for reproducing the research and its results in the individual environment in this Configuration Manual. Software with the hardware components needed along with imported data and Exploratory Data Analysis, Data Pre-processing, Label Encoding, Feature Selection, all the support vector machine model with validating other methods and their respective evaluation are included. This report follows the following structure, Section 2 tells the information about environment configuration.

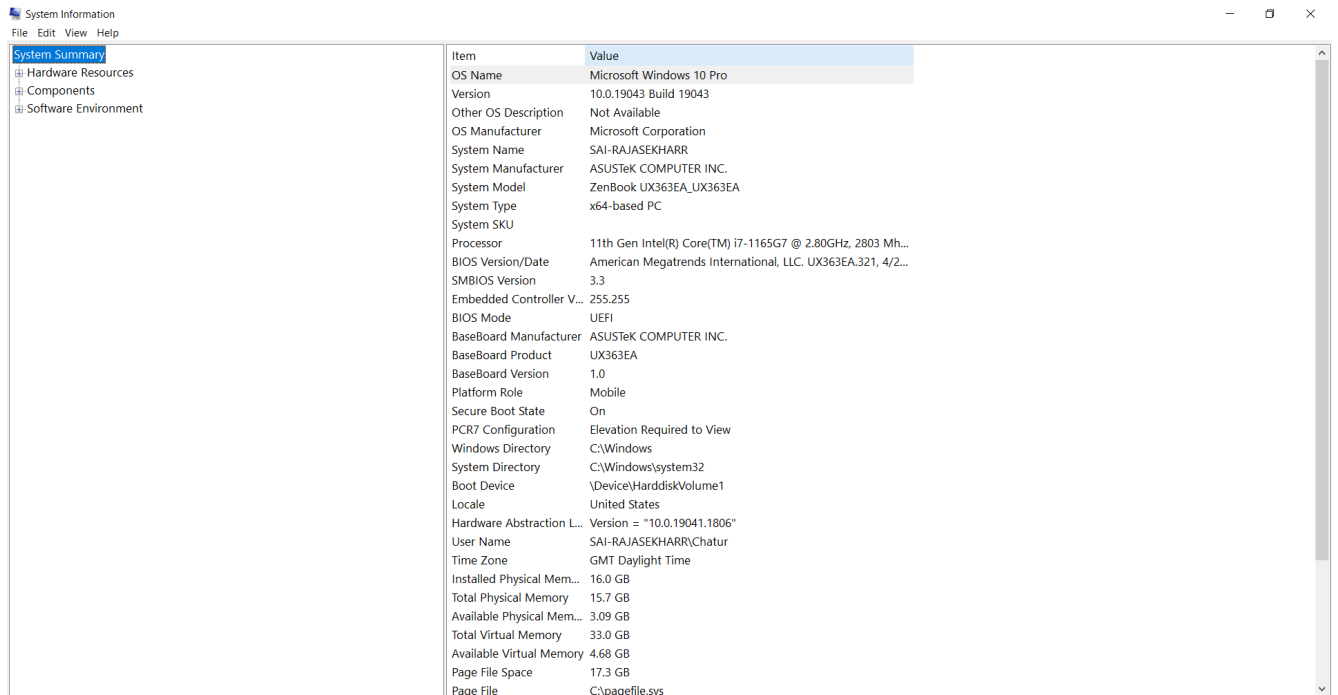
Section 3, gives detailed data acquisition. Section 4 consists of Pre-processing of the given data and Exploratory Data Analysis(EDA). Data Augmentation is showed in section 5. Section 6 shows in detail about GLCM Features. Section 7 shows in detail about NGDTM Features. Section 8 shows in detail about GLCM + NGDTM Features. Section 9 shows in detail about the SVM model. Section 10, explains about the results
float

2 System Specifications

A detailed description of the hardware and software requirements for implementing the research is provided in this section.

2.1 Hardware Specifications

Below image, shows the hardware that is required. IntelCore CPU @ 2.80 GHz Intel i7-1156G7 is the 11th Generation, 16 GB of the DDR4 RAM Memory at speed of 3600 Mhz, Operating system is windows 10 home edition with 64 bit processor, 1024GB SSD
.



The screenshot shows the Windows System Information window. On the left, there is a navigation pane with 'System Summary' selected. The main area displays a list of system properties and their values.

Item	Value
OS Name	Microsoft Windows 10 Pro
Version	10.0.19043 Build 19043
Other OS Description	Not Available
OS Manufacturer	Microsoft Corporation
System Name	SAI-RAJASEKHARR
System Manufacturer	ASUSTeK COMPUTER INC.
System Model	ZenBook UX363EA_UX363EA
System Type	x64-based PC
System SKU	
Processor	11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz, 2803 Mh...
BIOS Version/Date	American Megatrends International, LLC. UX363EA.321, 4/2...
SMBIOS Version	3.3
Embedded Controller V...	255.255
BIOS Mode	UEFI
BaseBoard Manufacturer	ASUSTeK COMPUTER INC.
BaseBoard Product	UX363EA
BaseBoard Version	1.0
Platform Role	Mobile
Secure Boot State	On
PCR7 Configuration	Elevation Required to View
Windows Directory	C:\Windows
System Directory	C:\Windows\system32
Boot Device	\Device\HarddiskVolume1
Locale	United States
Hardware Abstraction L...	Version = "10.0.19041.1806"
User Name	SAI-RAJASEKHARR\Chatur
Time Zone	GMT Daylight Time
Installed Physical Mem...	16.0 GB
Total Physical Memory	15.7 GB
Available Physical Mem...	3.09 GB
Total Virtual Memory	33.0 GB
Available Virtual Memory	4.68 GB
Page File Space	17.3 GB
Page File	C:\pagefile.sys

Figure 1: System Configuration

2.2 Software Requirements

1. Anaconda 3 (Windows operating system)
2. Jupyter Notebook (Latest)
3. Python Version 3.x

3 Data Collection

The main dataset is obtained from Mendely Data public cloud repository. <https://data.mendeley.com/datasets/5kxw8/1> is the link for the dataset. There are 1,288 pictures of banana leaf falling in three categories as Healthy banana leaf, Xanthomonas infected leaves and Sigatoka infected leaves.

4 Data Exploration

The project requires all the Python libraries listed below in Figure 2. float

```

import glob, random, re
import os, sys
import pandas as pd
import shutil
import cv2
from sklearn.preprocessing import MinMaxScaler
import matplotlib.pyplot as plt
import seaborn as sb
import numpy as np
import warnings
warnings.filterwarnings("ignore")
from sklearn.decomposition import PCA
from skimage.feature import greycomatrix, greycoprops
from scipy import signal
from collections import Counter
from imblearn.over_sampling import SMOTE
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn.metrics import confusion_matrix, accuracy_score

#Preprocessing of Images
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.preprocessing.image import load_img
from tensorflow.keras.utils import to_categorical
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split

#Sharpening of images
from skimage.io import imshow, imread
from skimage.color import rgb2yuv, rgb2hsv, rgb2gray, yuv2rgb, hsv2rgb
from scipy.signal import convolve2d

```

Figure 2: Python Libraries required

```

data1="./Banana_Leaf_Images/healthy"
data2="./Banana_Leaf_Images/segatoka"
data3="./Banana_Leaf_Images/xamthomonas"
count_healthy=0
count_segatoka=0
count_xamthomonas=0

```

```

for i in os.listdir(data1):
    img_path=os.path.join(data1, i)
    count_healthy=count_healthy+1
for i in os.listdir(data2):
    img_path=os.path.join(data2, i)
    count_segatoka=count_segatoka+1
for i in os.listdir(data3):
    img_path=os.path.join(data3, i)
    count_xamthomonas=count_xamthomonas+1

```

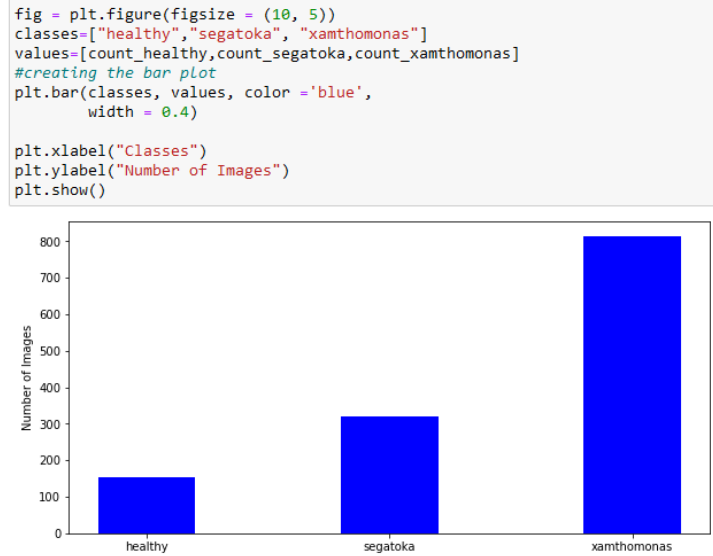


Figure 3: Generating images list based on categories

```
healthy = glob.glob("./Banana_Leaf_Images/healthy/*.jpg")
# Print out the first 5 file names to verify we're in the right folder.
print ("Total of %d images.\nFirst 5 filenames:" % len(healthy))
print ('\n'.join(healthy[:5]))
```

```
Total of 155 images.
First 5 filenames:
./Banana_Leaf_Images/healthy\20210218_154608.jpg
./Banana_Leaf_Images/healthy\20210218_154706.jpg
./Banana_Leaf_Images/healthy\20210218_160901.jpg
./Banana_Leaf_Images/healthy\20210218_160903.jpg
./Banana_Leaf_Images/healthy\20210218_160916.jpg
```

```
segatoka = glob.glob("./Banana_Leaf_Images/segatoka/*.jpg")
# Print out the first 5 file names to verify we're in the right folder.
print ("Total of %d images.\nFirst 5 filenames:" % len(segatoka))
print ('\n'.join(segatoka[:5]))
```

```
Total of 320 images.
First 5 filenames:
./Banana_Leaf_Images/segatoka\20210218_160016.jpg
./Banana_Leaf_Images/segatoka\20210219_112205.jpg
./Banana_Leaf_Images/segatoka\20210219_112208.jpg
./Banana_Leaf_Images/segatoka\20210219_112216.jpg
./Banana_Leaf_Images/segatoka\20210219_112219.jpg
```

```
xamthomonas = glob.glob("./Banana_Leaf_Images/xamthomonas/*.jpg")
# Print out the first 5 file names to verify we're in the right folder.
print ("Total of %d images.\nFirst 5 filenames:" % len(xamthomonas))
print ('\n'.join(xamthomonas[:5]))
```

```
Total of 814 images.
First 5 filenames:
./Banana_Leaf_Images/xamthomonas\20210218_154633.jpg
./Banana_Leaf_Images/xamthomonas\20210218_155926.jpg
./Banana_Leaf_Images/xamthomonas\20210218_155929.jpg
./Banana_Leaf_Images/xamthomonas\20210218_165151.jpg
./Banana_Leaf_Images/xamthomonas\20210218_170251.jpg
```

Figure 4: Generating images list based on categories

Defining a Sharpen Filter

```
sharpen = np.array([[0, -1, 0],
                    [-1, 5, -1],
                    [0, -1, 0]])

#Sharpening of images
from skimage.io import imshow, imread
from skimage.color import rgb2yuv, rgb2hsv, rgb2gray, yuv2rgb, hsv2rgb
from scipy.signal import convolve2d

data_dir = './Banana_Leaf_Images/'
categories = os.listdir(data_dir)
print(categories)

['healthy', 'segatoka', 'xanthomonas']

labels=[i for i in range(len(categories))]
labels
[0, 1, 2]
```

Figure 5: Defining Sharpening Image

```
og_image = imread(data_dir+'segatoka/20210218_160016.jpg')
imshow(og_image);
```



Figure 6: Reading Images

```
def multi_convolver(image, kernel, iterations):
    for i in range(iterations):
        image = convolve2d(image, kernel, 'same', boundary = 'fill',
                           fillvalue = 0)
    return image

def convolver_rgb(image, kernel, iterations = 1):
    img_yuv = rgb2yuv(image)
    img_yuv[:, :, 0] = multi_convolver(img_yuv[:, :, 0], kernel,
                                       iterations)
    final_image = yuv2rgb(img_yuv)
    return final_image
```

```
final_image = convolver_rgb(og_image, sharpen, iterations = 1)
imshow(final_image);
```

Clipping input data to the valid range for imshow with RGB data (

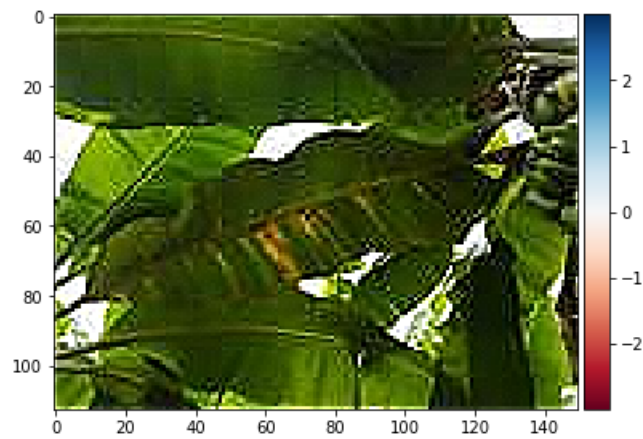


Figure 7: Image Convulsion

- Resizing to 224,224
- Sharpening

```
data_list=[] #data_list- storing the images
labels_list=[] #label_list - storing the class labels
```

```
label_dict=dict(zip(categories, labels))
label_dict
```

```
{'healthy': 0, 'segatoka': 1, 'xanthomonas': 2}
```

```
from PIL import Image
for i in categories:
    folder_path=os.path.join(data_dir, i) #path to each disease folder
    img_names=os.listdir(folder_path) #all images in each disease folder
    for img_name in img_names:
        img_path=os.path.join(folder_path, img_name)
        if(img_path=='./Banana_Leaf_Images/healthy' or img_path=='./Banana_Leaf_Images/segatoka' or img_path=='./Banana_Leaf_Images/xanthomonas'):
            continue
        img = Image.open(img_path)
        img = img.resize((224,224))
        img = np.array(img)
        img = convolver_rgb(img, sharpen, iterations = 1)
        data_list.append(img)
        labels_list.append(label_dict[i])
```

```
data_list[0].shape
```

```
(224, 224, 3)
```

Figure 8: Converting all images to rgb based on all categories


```

#Preprocessing of Images
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.preprocessing.image import load_img
from tensorflow.keras.utils import to_categorical
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split

lb = LabelEncoder()
labels_list = lb.fit_transform(labels_list)
labels_list = to_categorical(labels_list)

data = np.array(data_list)
labels = np.array(labels_list)
print(data.shape)

(1289, 224, 224, 3)

```

Figure 9: Label encoding images

```

plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(data[np.random.randint(224)], cmap=plt.cm.binary)

plt.show()

```

```

imgs = [] #list image matrix
labels = []
for filename in healthy:
    img = cv2.imread(filename)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    imgs.append(gray)
    labels.append(0) # healthy
for filename in segatoka:
    img = cv2.imread(filename)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    imgs.append(gray)
    labels.append(1) # "segatoka"
for filename in xamthomonas:
    img = cv2.imread(filename)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    imgs.append(gray)
    labels.append(2) # "xamthomonas"

```

```

def plotImages(images_arr):
    fig, axes = plt.subplots(3, 4, figsize=(20,20))
    axes = axes.flatten()
    for img, ax in zip( images_arr, axes):
        ax.imshow(img)
    plt.tight_layout()
    plt.show()

plotImages(imgs)

```

Figure 10: Visualising pre-processed images

5 Image Augmentation

This section explains the steps taken in data augmentation.

```
aug = ImageDataGenerator(
    rotation_range=20,
    zoom_range=0.15,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.15,
    horizontal_flip=True,
    fill_mode="nearest")
```

Figure 11: Data Augmentation

6 GLCM

This section show the implementation of features extracted by GLCM feature extractor algorithm.

```
def generateGlcmaGls(img, label, props, dists=[5], agls=[0, np.pi/4, np.pi/2, 3*np.pi/4], lvl=256, sym=True, norm=True):
    glcm = greycomatrix(img, distances=dists, angles=agls, levels=lvl, symmetric=sym, normed=norm)
    feature = []
    propsGlcma = [prop for name in props for prop in greycoprops(glcm, name)[0]]
    for prop in propsGlcma:
        feature.append(prop)
    feature.append(label)
    return feature
```

Figure 12: GLCM function

```
props = ['dissimilarity', 'correlation', 'homogeneity', 'contrast', 'ASM', 'energy']

glcmFeatures = []
for img, label in zip(imgs, labels):
    glcmFeatures.append(generateGlcmaGls(img, label, props=props))

columns = []
angles = ['0', '45', '90', '135']
for name in props :
    for ang in angles:
        columns.append(name + "_" + ang)
columns.append("label")
```

Figure 13: Extracting features using GLCM

```
# Create the pandas DataFrame for GLCM features data
glcmData = pd.DataFrame(glcmFeatures, columns = columns)

glcmData.head(15)
```

Figure 14: Converting NGDTM extracted features into dataframe

```
glcmData.isnull().sum()
```

```

dissimilarity_0      0
dissimilarity_45     0
dissimilarity_90     0
dissimilarity_135    0
correlation_0        0
correlation_45       0
correlation_90       0
correlation_135      0
homogeneity_0        0
homogeneity_45       0
homogeneity_90       0
homogeneity_135      0
contrast_0           0
contrast_45          0
contrast_90          0
contrast_135         0
ASM_0                0
ASM_45               0
ASM_90               0
ASM_135              0
energy_0             0
energy_45            0
energy_90            0
energy_135           0
label                0
dtype: int64
```

```
glcmData.describe()
```

Figure 15: Missing data handling and data description

```

features = glcmData.drop(['label'], axis='columns')
target= glcmData['label']
features

```

Figure 16: Feature selection

```

scaler = MinMaxScaler().fit(features)
features = scaler.transform(features)

```

```

oversample = SMOTE()
features,target = oversample.fit_resample(features,target)
counter = Counter(target)
print(counter)

```

```

Counter({0: 2442, 1: 2442, 2: 2442})

```

```

x_train, x_test, y_train, y_test = train_test_split(features,target, test_size=0.15, random_state=42)

```

Figure 17: Scaling, Class balancing and data split

7 NGDTM

This section show the implementation of features extracted by NGDTM feature extractor algorithm

```

def imageXOR(img):
    img = img.astype(np.uint8)
    mask = np.ones(img.shape, np.uint8)
    xorImg = np.zeros(img.shape, np.uint8)
    for i in range(img.shape[0]):
        for j in range(img.shape[1]):
            xorImg[i,j] = img[i,j] ^ mask[i,j]
    return xorImg

def genNgtdm(img, mask, d, Ng=256):
    img = img.astype(np.double)
    N1, N2 = img.shape
    oneskernl = np.ones((2*d+1,2*d+1))
    kernel = oneskernl.copy()
    kernel[d,d] = 0
    W = (2*d + 1)**2
    mask = imageXOR(mask)
    mask = signal.convolve2d(mask, oneskernl, 'same')
    mask = abs(np.sign(mask)-1)
    A = signal.convolve2d(img, kernel, 'same') / (W-1)
    diff = abs(img-A)
    S = np.zeros(Ng, np.double)
    N = np.zeros(Ng, np.double)
    for x in range(d, (N1-d)):
        for y in range(d, (N2-d)):
            if mask[x,y] > 0:
                index = img[x,y].astype('i')
                S[index] = S[index] + diff[x,y]
                N[index] += 1
    R = sum(N)
    return S, N, R

```

Figure 18: NGDTM function

```

def generateNGTDMFeatures(img, mask, label, d=1):
    if mask is None:
        mask = np.ones(img.shape)

    img = img.astype(np.uint8)
    mask = mask.astype(np.uint8)
    Ng = 256

    S, N, R = genNgtdm(img, mask, d, Ng)

    features = np.zeros(6, np.double)
    Ni, Nj = np.meshgrid(N, N)
    Si, Sj = np.meshgrid(S, S)
    Ngi, Ngj = np.meshgrid(np.arange(Ng), np.arange(Ng))
    Ngdiffsq = ((Ngi - Ngj)**2).astype(np.double)
    Ni = np.multiply(Ni, abs(np.sign(Nj)))
    Nj = np.multiply(Nj, abs(np.sign(Ni)))
    features[0] = R*R / sum(np.multiply(N, S))
    features[1] = sum(S)*sum(sum(np.multiply(np.multiply(Ni, Nj), Ngdiffsq))) / R**3 / Ng / (Ng-1)
    mult = np.multiply(Ngi, Ni) - np.multiply(Ngj, Nj)
    features[2] = sum(np.multiply(N, S)) / sum(sum(abs(mult))) / R
    mult = np.multiply(Ni, Si) + np.multiply(Nj, Sj)
    mult2 = np.multiply(abs(Ngi - Ngj), mult)
    mult3 = np.divide(mult2, Ni + Nj + 1e-16)
    features[3] = sum(sum(mult3)) / R
    features[4] = sum(sum(np.multiply(Ni + Nj, Ngdiffsq))) / (sum(S) + 1e-16)
    features[5] = label
    return features

```

Figure 19: NGDTM function

```

columns = ["NGTDM_Coarseness", "NGTDM_Contrast", "NGTDM_Busyness", "NGTDM_Complexity", "NGTDM_Strngth", 'label']

ngtdmFeatures = []
for img, label in zip(imgs, labels):
    ngtdmFeatures.append(generateNGTDMFeatures(img, img>0, label))

# Create the pandas DataFrame for NGTDM features data
ngtdmData = pd.DataFrame(ngtdmFeatures, columns = columns)

ngtdmData.head(15)

```

	NGTDM_Coarseness	NGTDM_Contrast	NGTDM_Busyness	NGTDM_Complexity	NGTDM_Strngth	label
0	22.486973	0.106863	0.000001	87913.936405	343132.709713	0.0
1	26.364381	0.281524	0.000001	102506.396257	568446.542874	0.0
2	27.186464	0.123899	0.000002	91760.925851	506395.161956	0.0
3	27.538203	0.133526	0.000002	94270.372437	488887.552588	0.0

Figure 20: Extracting features using GLCM and Converting them into dataframe

```

: ngtdmData.isnull().sum()
: NGTDM_Coarseness    3
: NGTDM_Contrast      3
: NGTDM_Busyness      3
: NGTDM_Complexity    3
: NGTDM_Strngth       0
: label               0
: dtype: int64

: ngtdmData = ngtdmData.fillna(0)

: ngtdmData.describe()

```

	NGTDM_Coarseness	NGTDM_Contrast	NGTDM_Busyness	NGTDM_Complexity	NGTDM_Strngth	label
count	3867.000000	3867.000000	3867.000000	3867.000000	3.867000e+03	3867.000000
mean	18.411548	0.469811	0.000003	192202.905078	4.453072e+05	1.511249
std	5.047935	0.335897	0.000003	94446.762237	1.839699e+05	0.700355
min	0.000000	0.000000	0.000000	0.000000	0.000000e+00	0.000000
25%	14.689241	0.202504	0.000002	118156.243159	3.288004e+05	1.000000
50%	17.783632	0.396389	0.000003	189328.849896	4.117100e+05	2.000000
75%	21.169409	0.675807	0.000004	257156.667216	5.229258e+05	2.000000
max	43.118379	1.792014	0.000105	485214.700223	2.416768e+06	2.000000

Figure 21: Missing data handling and data description

```

: ngtdmData.label.value_counts()
: 2.0    2442
: 1.0     960
: 0.0     465
: Name: label, dtype: int64

: features = ngtdmData.drop(['label'], axis='columns')
: target= ngtdmData['label']
: features

```

	NGTDM_Coarseness	NGTDM_Contrast	NGTDM_Busyness	NGTDM_Complexity	NGTDM_Strngth
0	22.486973	0.106863	0.000001	87913.936405	343132.709713
1	26.364381	0.281524	0.000001	102506.396257	568446.542874
2	27.186464	0.123899	0.000002	91760.925851	506395.161956
3	27.538203	0.133526	0.000002	94270.372437	488887.552588
4	18.271437	0.448775	0.000003	203003.648994	503295.348991

Figure 22: Checking of class balance and feature and target split

```

scaler = MinMaxScaler().fit(features)
features = scaler.transform(features)

oversample = SMOTE()
features,target = oversample.fit_resample(features,target)
counter = Counter(target)
print(counter)

Counter({0.0: 2442, 1.0: 2442, 2.0: 2442})

x_train, x_test, y_train, y_test = train_test_split(features,target, test_size=0.20, random_state=42)

```

Figure 23: Scaling, Class balancing and data split

8 GLCM+NGTDM

This section show the implementation of features extracted GLCM and NGDTM and concatenated into one dataframe.

```

data = pd.concat([glcmData, ngtdmData], axis=1)
data = data.T.drop_duplicates().T

```

Figure 24: Merging GLCM and NGDTM data


```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3867 entries, 0 to 3866
Data columns (total 30 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   dissimilarity_0                       3867 non-null   float64
1   dissimilarity_45                      3867 non-null   float64
2   dissimilarity_90                      3867 non-null   float64
3   dissimilarity_135                    3867 non-null   float64
4   correlation_0                        3867 non-null   float64
5   correlation_45                       3867 non-null   float64
6   correlation_90                       3867 non-null   float64
7   correlation_135                      3867 non-null   float64
8   homogeneity_0                        3867 non-null   float64
9   homogeneity_45                      3867 non-null   float64
10  homogeneity_90                       3867 non-null   float64
11  homogeneity_135                      3867 non-null   float64
12  contrast_0                           3867 non-null   float64
13  contrast_45                          3867 non-null   float64
14  contrast_90                          3867 non-null   float64
15  contrast_135                         3867 non-null   float64
16  ASM_0                                3867 non-null   float64
17  ASM_45                               3867 non-null   float64
18  ASM_90                               3867 non-null   float64
19  ASM_135                              3867 non-null   float64
20  energy_0                             3867 non-null   float64
21  energy_45                            3867 non-null   float64
22  energy_90                            3867 non-null   float64
23  energy_135                           3867 non-null   float64
24  label                                3867 non-null   float64
25  NGTDM_Coarseness                     3867 non-null   float64
26  NGTDM_Contrast                       3867 non-null   float64
27  NGTDM_Busyness                       3867 non-null   float64
28  NGTDM_Complexity                     3867 non-null   float64
29  NGTDM_Strngth                        3867 non-null   float64
dtypes: float64(30)
memory usage: 906.5 KB
```

```
features = data.drop(['label'], axis='columns')
target= data['label']
features
```

Figure 25: Data information and feature and target split

```

scaler = MinMaxScaler().fit(features)
features = scaler.transform(features)

oversample = SMOTE()
features,target = oversample.fit_resample(features,target)
counter = Counter(target)
print(counter)

Counter({0.0: 2442, 1.0: 2442, 2.0: 2442})

aug = ImageDataGenerator(
    rotation_range=20,
    zoom_range=0.15,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.15,
    horizontal_flip=True,
    fill_mode="nearest")

x_train, x_test, y_train, y_test = train_test_split(features,target, test_size=0.20, random_state=42)

```

Figure 26: Scaling, Class balancing and data split

9 Machine Learning Models

9.1 SVM for GLCM

```

model= svm.SVC(gamma='scale', C=20, kernel = 'linear')

model.fit(x_train,y_train)

accuracy = model.score(x_test,y_test)*100
accuracy

y_predicted = model.predict(x_test)
cm = confusion_matrix(y_test,y_predicted)
plt.figure(figsize = (8,7))
sb.heatmap(cm, annot=True)
plt.xlabel('Predicted')
plt.ylabel('Truth')

score = []
score.append(["Model" , "Accuracy"])
score.append(["GLCM" , accuracy])

```

Figure 27: Implementation of SVM for GLCM

9.2 SVM for NGTDM

```
model= svm.SVC(C=10, gamma=50)
```

```
model.fit(x_train,y_train)
```

```
accuracy = model.score(x_test,y_test)*100  
accuracy
```

```
y_predicted = model.predict(x_test)  
cm = confusion_matrix(y_test,y_predicted)  
plt.figure(figsize = (10,7))  
sb.heatmap(cm, annot=True)  
plt.xlabel('Predicted')  
plt.ylabel('Truth')
```

```
score.append(["NGTDM" , accuracy])
```

Figure 28: Implementation of SVM for NGTDM

9.3 9.3 SVM for NGTDM+GLCM

```
model= svm.SVC(C=10, gamma=50)

model.fit(x_train,y_train)

accuracy = model.score(x_test,y_test)*100
accuracy

y_predicted = model.predict(x_test)
cm = confusion_matrix(y_test,y_predicted)
plt.figure(figsize = (10,7))
sb.heatmap(cm, annot=True)
plt.xlabel('Predicted')
plt.ylabel('Truth')

score.append(["GLCM + NGTDM" , accuracy])
```

Figure 29: Implementation of SVM for NGTDM+GLCM

10 Model result

This section explains the performance of the models.

```
score.append(["GLCM + NGTDM" , accuracy])
score = pd.DataFrame(score)
score
```

	0	1
0	Model	Accuracy
1	GLCM	77.929155
2	NGTDM	70.961145
3	GLCM + NGTDM	89.775051
4	GLCM + NGTDM	89.775051

Figure 30: Figure 37: Model Performance

11 References

<https://docs.w3cub.com/scikit-learn/modules/generated/sklearn.svm.svc>

<https://www.codegrepper.com/code-examples/python/train-test-split+in+python+svm>

<https://stackoverflow.com/questions/69796388/extracting-glcmm-features>

<https://www.geeksforgeeks.org/python-opencv-cv2-cvtColor-method/>

<https://ggplot2.tidyverse.org/reference/aesposition.html>

<https://scikit-learn.org/stable/auto-examples/svm/plot-rbf-parameters.html>