Assignment 1 : Raw Data to Feature Space

Deepa Jayanna

1 BUILDING PROGRAMMING ENVIRONMENT

1.1 Download and install python environment

Python (version 3.8.5) was downloaded and installed from the website https://docs.anaconda.com/anaconda. I preferred python as it has huge follower base and it is easier to find related resources. Also, I was comfortable with Java programming language so I thought it would be easier to learn python. Plus, the Professor was also using python code to explain the programs in the class and that it seemed I would be able to understand the concepts easier. Figure 1 shows the Anaconda installed in my system.

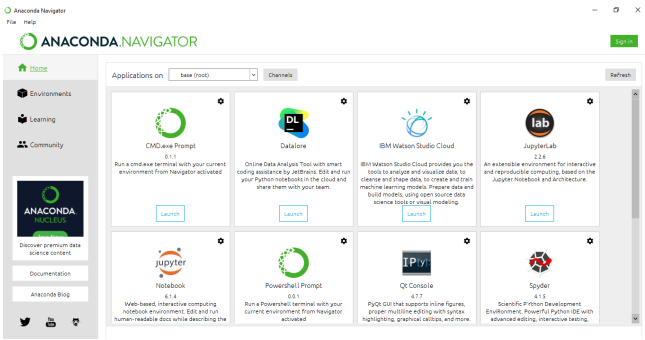


Figure 1: Anaconda navigator

1.2 Install the Spyder IDE

Spyder IDE was already present in the Ananconda package installed in sub task 1.1 and I accessed it via Ananconda navigator. Figure 2 represents the Spyder IDE set up in my system.

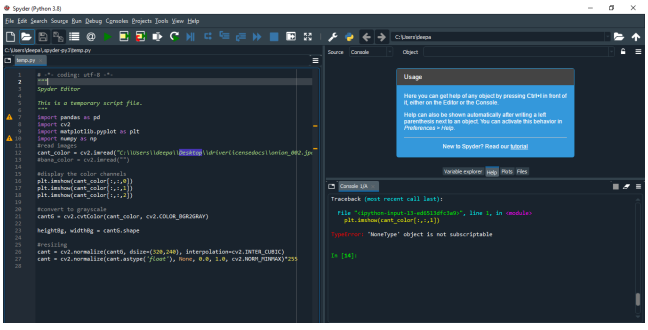


Figure 2: Spyder work space

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
Conference'17, July 2017, Washington, DC, USA
© 2021 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1.3 Install Jupyter

Jupyter was already installed when Ananconda was installed. Figure 3 shows the Jupyter package installed in my system.

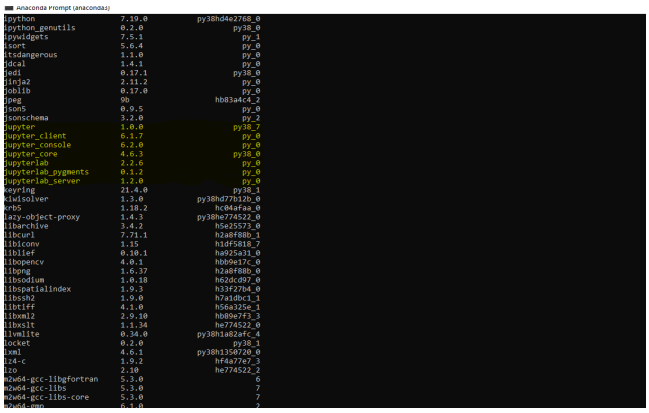


Figure 3: Jupyter installion

1.4 Install the OpenCV

Initially when I downloaded Anaconda, opencv was not present, even though it used be automatically available in older versions of python. I installed opencv through Anaconda command prompt by using command **conda install -c conda-forge opencv** as suggested by the professor during lecture. Figure 4 shows the opencv installed in my system.

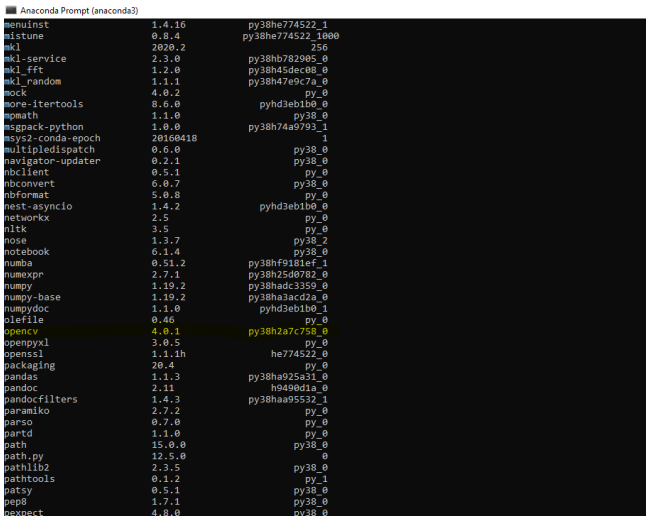


Figure 4: Opencv installed

2 DOWNLOAD OR GENERATE A FRUITS AND VEGETABLES IMAGE DATA SET!

2.1 Download FIDS30 Data Set

Downloaded and unzipped the FIDS30 data from the https://www.vicos.si/Downloads/FIDS30/. [1] Figure 5 shows the screenshot of the data folder in my local system.

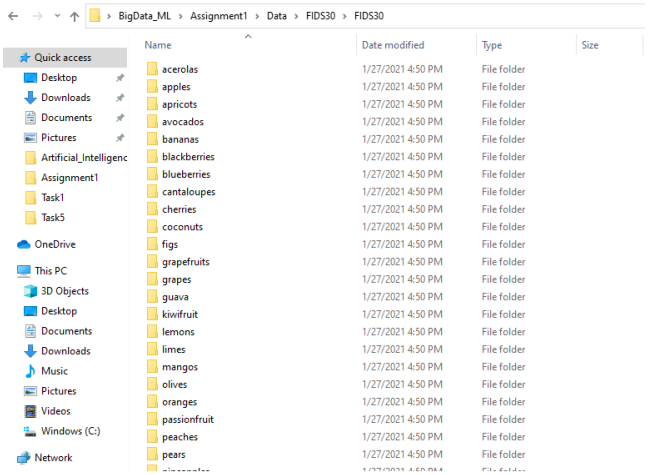


Figure 5: FIDS30 Data folder

2.2 Download Tropical Fruits Data set

Downloaded Tropical fruits data from http://www.ic.unicamp.br/~rocha/pub/downloads/tropical-fruits-DB-1024-tar.gz, unzipped the folder and stored in my local system. Figure 6 shows the screenshot the data folder I have created in my system.

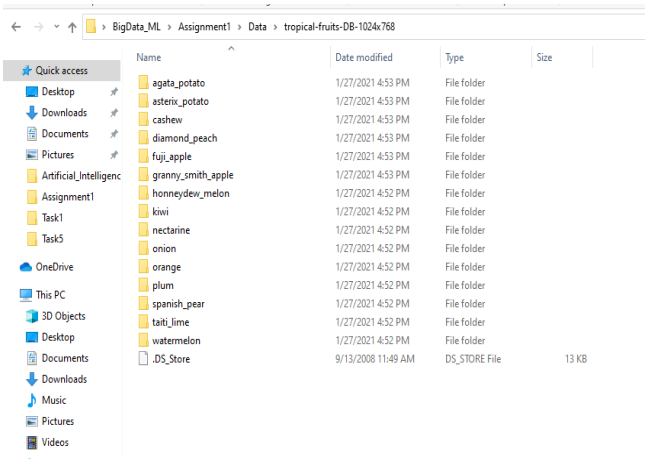


Figure 6: Tropical fruits Data folder

2.3 Choose three fruits.

I am choosing three fruits Banana, Guava and Pineapple. I am assigning labels image0 to banana, image1 to guava and image2 to pineapple to explain the rest of the assignment. Figure 7 to Figure 9 shows the images of the fruits I will be using for the feature extraction process.

Figure 7: image0

Figure 8: image1

Figure 9: image2

```python
import cv2
import matplotlib.pyplot as plt

#read the image
banana_color = cv2.imread('banana.jpg')
guava_color = cv2.imread('guava.jpg')
pineapple_color = cv2.imread('pineapple.jpg')

#display the color channels
plt.axis('off')

#RGB for banana
plt.imshow(banana_color[:,:,0])
plt.imshow(banana_color[:,:,1])
plt.imshow(banana_color[:,:,2])
#RGB for guava
plt.imshow(guava_color[:,:,0])
plt.imshow(guava_color[:,:,1])
plt.imshow(guava_color[:,:,2])
#RGB for pineapple
plt.imshow(pineapple_color[:,:,0])
plt.imshow(pineapple_color[:,:,1])
plt.imshow(pineapple_color[:,:,2])
```

Figure 10: Code for generating RGB scale images

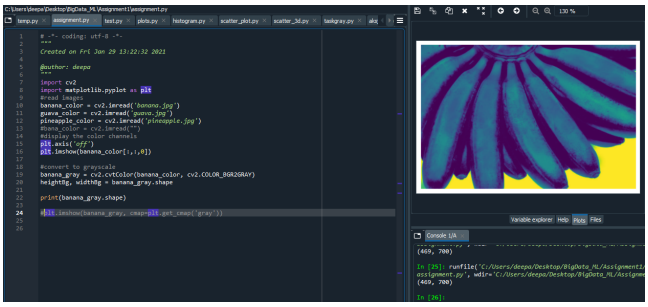


Figure 11: Red scale for image0

3 WRITE A SIMPLE CODE TO READ YOUR SELECTED IMAGES AND DISPLAY THEM ON THE ENVIRONMENT!

I am using opencv to read the images and matplotlib.pyplot to plot the images in python environment. I have generated RGB images for image0, image1 and image2 using the code shown in Figure 10. I have ran the code separately once passing R, then G and then B to get the respective scales for each image. Gray scale images for each fruit is generated separately by executing the code three times and corresponding dimensions are printed in console. Fig 11 to Figure 19 shows the execution of RGB code and Figure 20 to 22 corresponds to the Gray scale image code executed.

4 RESIZE THE IMAGES TO REDUCE THEIR DIMENSIONS!

4.1 Convert Image dimension divisible by 8

I have written a parametric function to calculate the divisibility by 8. Initially I have halved the dimensions in order to reduce the dimension. Then, I am passing these halved dimensions to the

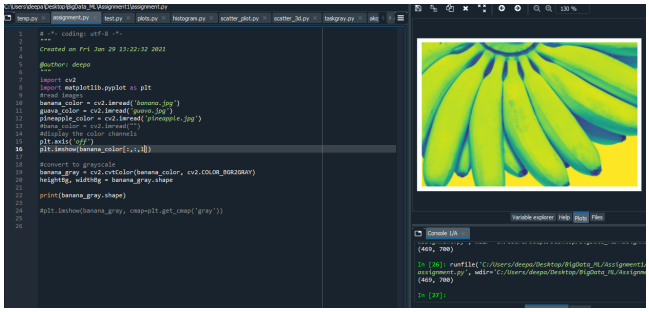


Figure 12: Green scale for image0

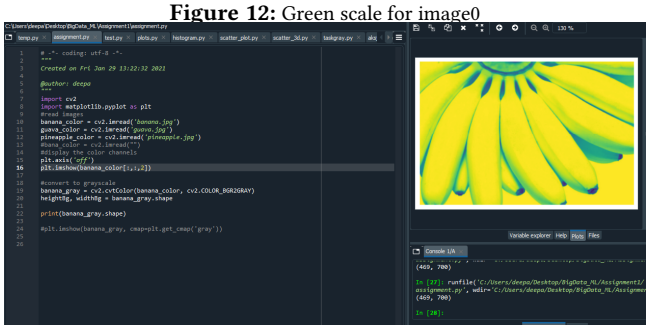


Figure 13: Blue scale for image0

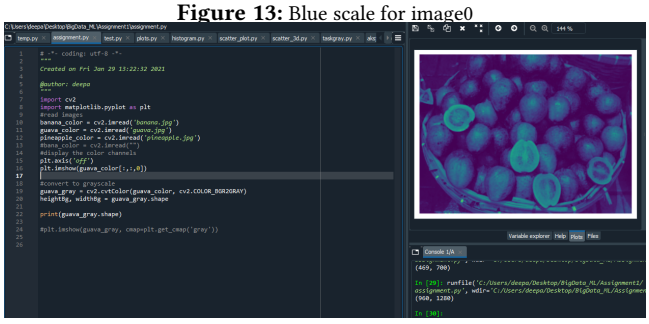


Figure 14: Red scale for image1

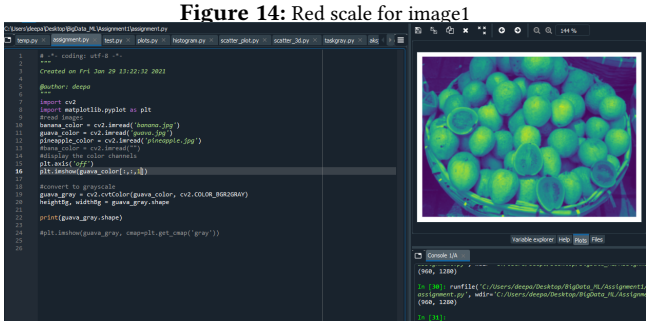


Figure 15: Green scale for image1

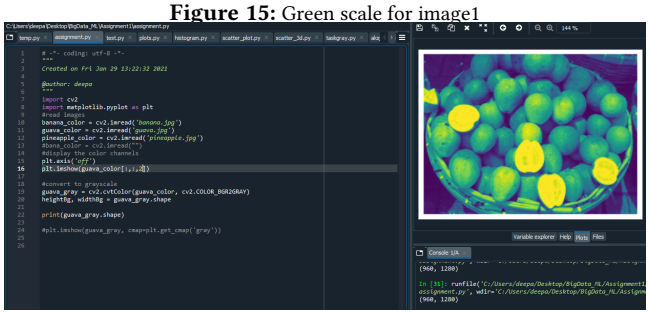


Figure 16: Blue scale for image1

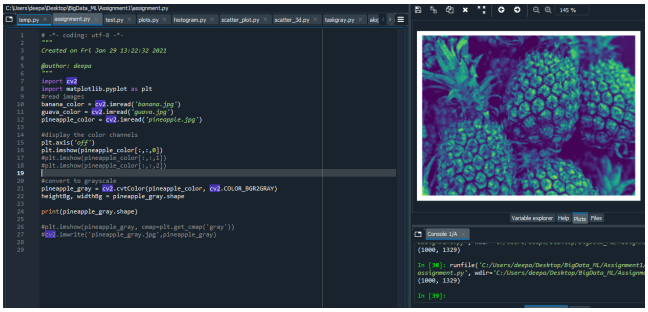


Figure 17: Red scale for image2

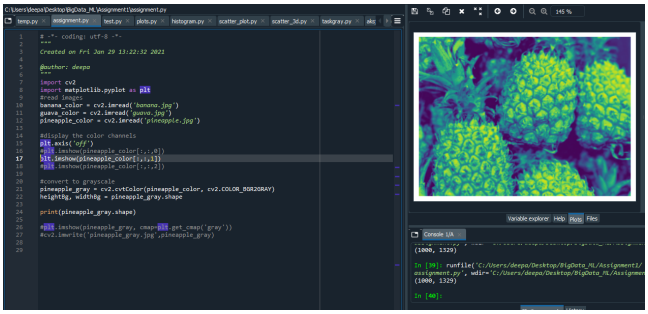


Figure 18: Green scale for image2

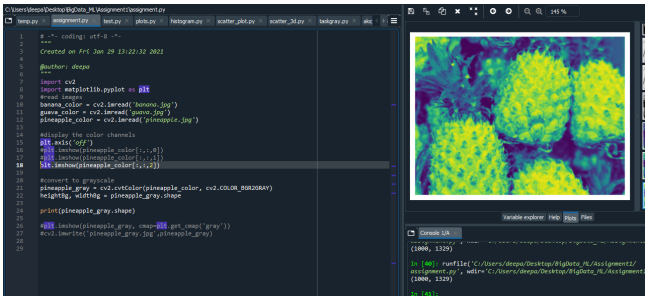


Figure 19: Blue scale for image2

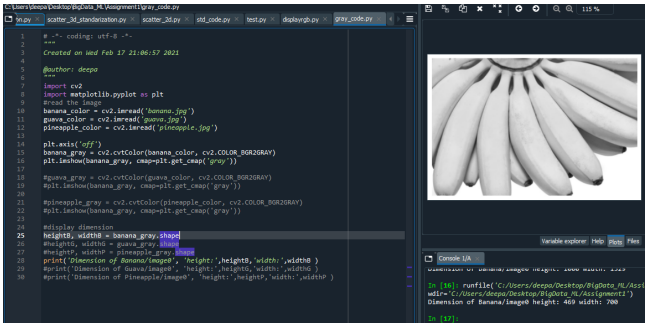


Figure 20: Gray image Banana with dimension printed

function to check their divisibility by 8. If they are not divisible, by using the remainder I am rounding them off them to nearest values that are divisible by 8. Figure 23 shows the code I have used to accomplish the task. Initial dimension of image0 was 700*469, image1 was 1280*960 and image2 was 1329*1000 and the resized image values are 352*232, 640*480 and 664*496 respectively. These images with new dimensions are uploaded in Data folder as proof of this task completion.

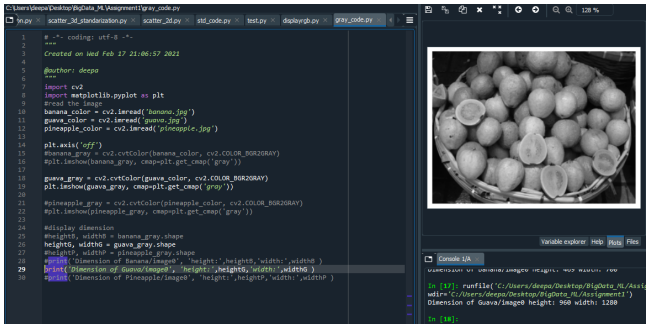


Figure 21: Gray image Guava with dimension printed

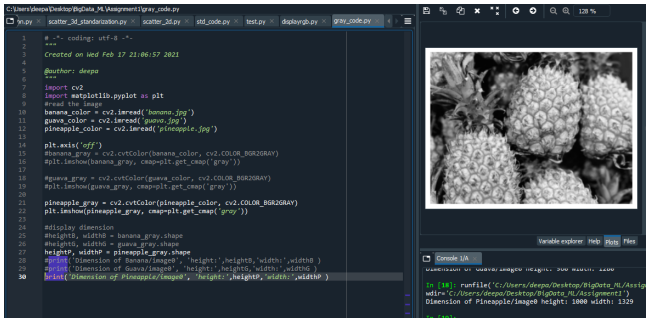


Figure 22: Gray image Pineapple with dimension printed

```
import cv2
#read images
fruitGray = cv2.imread('guava_gray.jpg')
#Displaying images in gray scale
width_fruit = int(fruitGray.shape[1]/2)
height_fruit = int(fruitGray.shape[0]/2)
print('Original dimension:',fruitGray.shape)
def hw_divby8(dimension):
    dimension_rem = dimension % 8
    if dimension_rem == 0:
        dimension = dimension

    elif dimension_rem >=5:
        dimension = dimension + (8-dimension_rem)
    else:
        dimension = dimension-dimension_rem
    return dimension

new_height = round(hw_divby8(height_fruit))
new_width = round(hw_divby8(width_fruit))
print('Divisible by 8 and reduced dimension:',new_height,new_width)

resized_fruit = cv2.resize(fruitGray, dsize=(new_width, new_height), interpolation=cv2.INTER_CUBIC)
cv2.imwrite('guava_div_8.png',resized_fruit)
```

Figure 23: Divisibility by 8 code

4.2 Convert the image to standard height 256

For this, I am reusing the code in the sub task 4.1. I am setting standard height to 256, and passing standard height and width to the function and calculating the new width as per the aspect ratio with the help of another function. The new width will be rounded off to the nearest value divisible by 8 according to the remainder generated. Then these values are passed through resize function and saved as new image. This process is repeated for image0, image1 and image2 to generate resized images. Resized image dimensions are 384*256 for image0, 344*256 for image1 and 336*256 for image2. Figure 25 shows the new images along with their dimensions saved in a folder.

5 GENERATE BLOCK FEATURE VECTORS

Each image is converted to gray scale image, converted in to standard height of 256 and resized width, then divided into non overlapping blocks of size 8*8 pixels to generate the block feature vectors. Each pixel is called Feature. image0 is split into 1536 Non

```python
import cv2
#read images
fruitGray = cv2.imread('guava_gray.jpg')
#Displaying images in gray scale
width_fruit = int(fruitGray.shape[1])
height_fruit = int(fruitGray.shape[0])
std_height = 256

def resize_get_width(height_fruit,width_fruit):
    print('original width',width_fruit)
    print('original height',height_fruit)
    aspect_ratio = width_fruit/height_fruit
    new_fruit_width = aspect_ratio*std_height
    print('My width',new_fruit_width)
    return round(new_fruit_width)

def width_divisibilityBy8(height_fruit,width_fruit):
    new_fruit_width = resize_get_width(height_fruit, width_fruit)
    dimension_rem = new_fruit_width % 8
    if dimension_rem == 0:
        return new_fruit_width
    elif dimension_rem >=5:
        new_fruit_width = new_fruit_width + (8-dimension_rem)
    else:
        new_fruit_width = new_fruit_width-dimension_rem
    return new_fruit_width

new_fruit_width = width_divisibilityBy8(height_fruit,width_fruit)
print('standard Height 256 dimension',std_height,new_fruit_width)
resized_fruit = cv2.resize(fruitGray, dsize=(new_fruit_width, std_height), interpolation=cv2.INTER_(
cv2.imwrite('resize_guava_256.png',resized_fruit)
```

Figure 24: Code to convert images to standard height 256

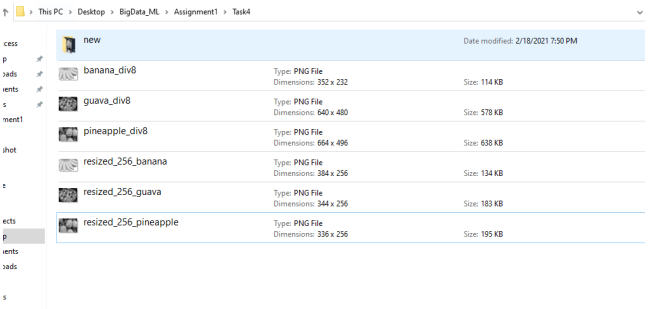


Figure 25: Saved new images dimensions divisible by 8

overlapping sub blocks of size 8*8 pixels, image1 is split into 1376 Non overlapping sub blocks of size 8*8 and image2 is split into 1346 Non overlapping sub blocks of size 8*8 pixels. Hence, there are 64 features vectors for each image. Figure 26 shows the code used to generate the non overlapping feature vectors.image0.csv, image1.csv and image2.csv are generated by executing the code separately by passing each image at a time. image0 is filled with 0, image1 with 1 and image2 with 2 in 64th column of spreadsheet. This is represented as class name or label.

```python
"""
import cv2
import numpy as np
import pandas as pd
fruit_read = cv2.imread('image1.jpg')
fruit_gray = cv2.cvtColor(fruit_read, cv2.COLOR_BGR2GRAY)
heightcG, widthcG = fruit_gray.shape
std_height = 256
def resize_get_width(heightcG,widthcG):
    aspect_ratio = widthcG/heightcG
    new_fruit_width = aspect_ratio*std_height
    return round(new_fruit_width)
def width_divisibilityBy8(height_fruit,width_fruit):
    new_fruit_width = resize_get_width(height_fruit, width_fruit)
    dimension_rem = new_fruit_width % 8
    if dimension_rem == 0:
        return new_fruit_width
    elif dimension_rem >=5:
        new_fruit_width = new_fruit_width + (8-dimension_rem)
    else:
        new_fruit_width = new_fruit_width-dimension_rem
    return new_fruit_width
new_fruit_width = width_divisibilityBy8(heightcG,widthcG)
resized_fruit = cv2.resize(fruit_gray, dsize=(new_fruit_width, std_height), interpolation=c
heightc, widthc = resized_fruit.shape
cc = round((heightc)*(widthc)/64)
flatc = np.ones((cc, 65), np.uint8)
k = 0
for i in range(0,heightc,8):
    for j in range(0,widthc,8):
        crop_tmp1 = resized_fruit[i:i+8,j:j+8]
        flatc[k,0:64] = crop_tmp1.flatten()
        k = k + 1
fspaceC = pd.DataFrame(flatc)
fspaceC.to_csv('image1.csv', index=False)
```

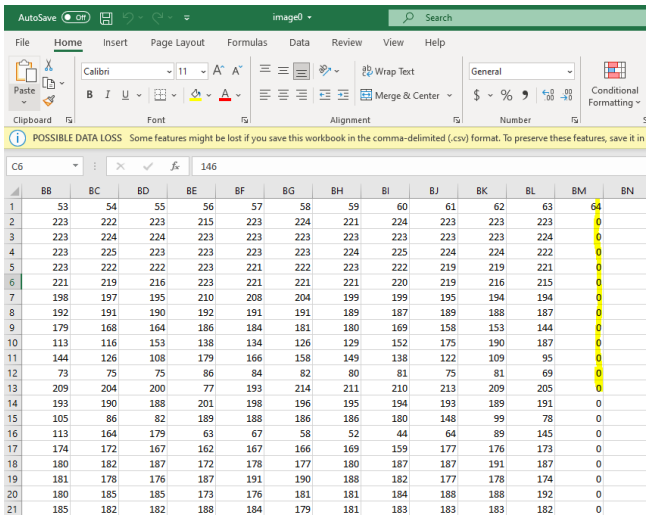
Figure 26: Non Overlapping Feature Vector code

6 TASK 6: GENERATE SLIDING BLOCK FEATURE VECTORS

Each image is converted to gray scale image, converted in to standard height of 256 and resized width, then divided into non overlapping blocks of size 8*8 pixels to generate the block feature vectors. Image1 is split into 93873 overlapping sub blocks of size 8*8 and image2 is split into 83913 overlapping sub blocks of size 8*8, image3 is split into 81921 overlapping sub blocks of size 8*8 and corresponding csv are generated as image0.csv, image1.csv and image2.csv. Image1.csv is assigned label 0 , image2 with label 1 and image3 with label 2 in the 64th column of the spreadsheet. Code written in Figure 27 is used to accomplish this task and Figure 28 - 30 shows the results obtained with class labels highlighted.

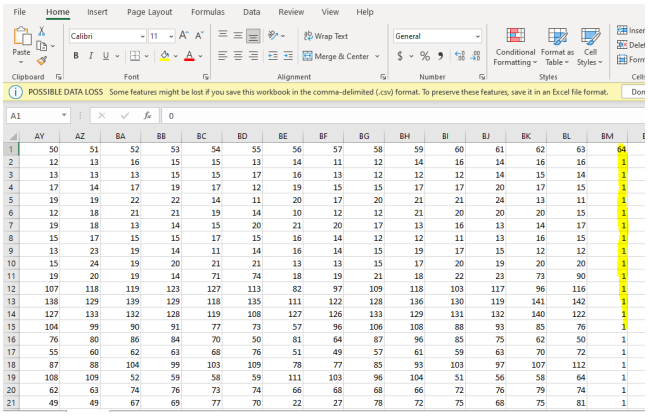
```python
import cv2
import numpy as np
import pandas as pd
fruit_read = cv2.imread('image1.jpg')
fruit_gray = cv2.cvtColor(fruit_read, cv2.COLOR_BGR2GRAY)
heightcG, widthcG = fruit_gray.shape
std_height = 256
def resize_get_width(heightcG,widthcG):
    aspect_ratio = widthcG/heightcG
    new_fruit_width = aspect_ratio*std_height
    return round(new_fruit_width)
def width_divisibilityBy8(height_fruit,width_fruit):
    new_fruit_width = resize_get_width(height_fruit, width_fruit)
    dimension_rem = new_fruit_width % 8
    if dimension_rem == 0:
        return new_fruit_width
    elif dimension_rem >=5:
        new_fruit_width = new_fruit_width + (8-dimension_rem)
    else:
        new_fruit_width = new_fruit_width-dimension_rem
    return new_fruit_width
new_fruit_width = width_divisibilityBy8(heightcG,widthcG)
resized_fruit = cv2.resize(fruit_gray, dsize=(new_fruit_width, std_height), interpolation=c
heightc, widthc = resized_fruit.shape
cc = round((heightc-7)*(widthc-7))
flatc = np.ones((cc, 65), np.uint8)
k = 0
for i in range(0,heightc-7,1):
    for j in range(0,widthc-7,1):
        crop_tmp1 = resized_fruit[i:i+8,j:j+8]
        flatc[k,0:64] = crop_tmp1.flatten()
        k = k + 1
fspaceC = pd.DataFrame(flatc)
fspaceC.to_csv('image1.csv', index=False)
```

Figure 27: Overlapping Feature Vector code



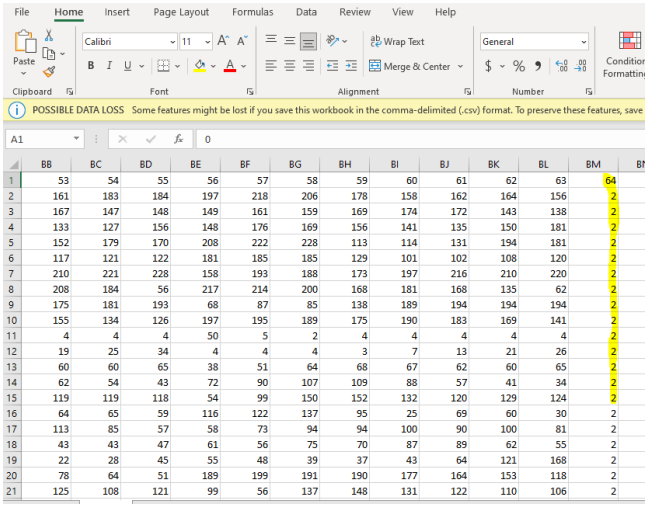
	BB	BC	BD	BE	BF	BG	BH	BI	BJ	BK	BL	BM	BN
1	53	54	55	56	57	58	59	60	61	62	63	64	
2	223	222	223	215	223	224	221	224	223	223	223	0	
3	223	224	224	223	223	223	223	223	223	223	224	0	
4	223	225	223	223	223	223	224	225	224	224	222	0	
5	223	222	222	223	221	222	223	222	219	219	221	0	
6	221	219	216	223	221	221	221	220	219	216	215	0	
7	198	197	195	210	208	204	199	199	195	194	194	0	
8	192	191	190	192	191	191	189	187	189	188	187	0	
9	179	168	164	186	184	181	180	169	158	153	144	0	
10	113	116	153	138	134	126	129	152	175	190	187	0	
11	144	126	108	179	166	158	149	138	122	109	95	0	
12	73	75	75	86	84	82	80	81	75	81	69	0	
13	209	204	200	77	193	214	211	210	213	209	205	0	
14	193	190	188	201	198	196	195	194	193	189	191	0	
15	105	86	82	189	188	186	186	180	148	99	78	0	
16	113	164	179	63	67	58	52	44	64	89	145	0	
17	174	172	167	162	167	166	169	159	177	176	173	0	
18	180	182	187	172	178	177	180	187	187	191	187	0	
19	181	178	176	187	191	190	188	182	177	178	174	0	
20	180	185	185	173	176	181	181	184	188	188	192	0	
21	185	182	182	188	184	179	181	183	183	183	182	0	

Figure 28: image0.csv with 0 as label in 64th column



	AY	AZ	BA	BB	BC	BD	BE	BF	BG	BH	BI	BJ	BK	BL	BM
1	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64
2	12	13	16	15	15	13	14	11	12	14	16	14	16	16	1
3	13	13	13	15	15	17	16	13	12	12	12	14	15	14	1
4	17	14	17	19	17	12	19	15	15	17	17	20	17	15	1
5	19	19	22	22	14	11	20	17	20	21	21	24	13	11	1
6	12	18	21	21	19	14	10	12	12	21	20	20	20	15	1
7	19	18	13	14	15	20	21	20	17	13	16	13	14	17	1
8	15	17	15	15	17	15	16	14	12	12	11	13	16	15	1
9	13	23	19	14	11	14	16	14	15	19	17	15	12	12	1
10	15	24	19	20	21	21	13	13	15	17	20	19	20	20	1
11	19	20	19	14	71	74	18	19	21	18	22	23	73	90	1
12	107	118	119	123	127	113	82	97	109	118	103	117	96	116	1
13	138	129	139	129	118	135	111	122	128	136	130	119	141	142	1
14	127	133	132	128	119	108	127	126	133	129	131	132	140	122	1
15	104	99	90	91	77	73	57	96	106	108	88	93	85	76	1
16	76	80	86	84	70	50	81	64	87	96	85	75	62	50	1
17	55	60	62	63	68	76	51	49	57	61	59	63	70	72	1
18	87	88	104	99	103	109	78	77	85	93	103	97	107	112	1
19	108	109	52	59	58	59	111	103	96	104	51	56	58	64	1
20	62	63	74	76	73	74	66	68	68	66	72	76	79	74	1
21	49	49	67	69	77	70	22	27	78	72	75	68	75	81	1

Figure 29: image1.csv with 1 as label in 64th column



	BB	BC	BD	BE	BF	BG	BH	BI	BJ	BK	BL	BM
1	53	54	55	56	57	58	59	60	61	62	63	64
2	161	183	184	197	218	206	178	158	162	164	156	2
3	167	147	148	149	161	159	169	174	172	143	138	2
4	133	127	156	148	176	169	156	141	135	150	181	2
5	152	179	170	208	222	228	113	114	131	194	181	2
6	117	121	122	181	185	185	129	101	102	108	120	2
7	210	221	228	158	193	188	173	197	216	210	220	2
8	208	184	56	217	214	200	168	181	168	135	62	2
9	175	181	193	68	87	85	138	189	194	194	194	2
10	155	134	126	197	195	189	175	190	183	169	141	2
11	4	4	4	50	5	2	4	4	4	4	4	2
12	19	25	34	4	4	4	3	7	13	21	26	2
13	60	60	65	38	51	64	68	67	62	60	65	2
14	62	54	43	72	90	107	109	88	57	41	34	2
15	119	119	118	54	99	150	152	132	120	129	124	2
16	64	65	59	116	122	137	95	25	69	60	30	2
17	113	85	57	58	73	94	94	100	90	100	81	2
18	43	43	47	61	56	75	70	87	89	62	55	2
19	22	28	45	55	48	39	37	43	64	121	168	2
20	78	64	51	189	199	191	190	177	164	153	118	2
21	125	108	121	99	56	137	148	131	122	110	106	2

Figure 30: image2.csv with 2 as label in 64th column

7 DERIVE STATISTICAL DESCRIPTORS!

7.1 Extract statistical information

In this task, Statistical information are obtained using Counting, calculating Mean of each feature, standard deviation, scatter plot and histogram.

7.2 Counting:

By looking at the csv files, below are the observations presented.
Non Overlapping data set. image0 has 64 features and 1536 Observations, 384*256 dimension and labelled as 0. image0 has 64 features and 1376 Observations, 344*256 dimension and labelled as 1. image2 has 64 features and 1346 observations, 336*256 dimension and labelled as 2. Cumulatively, there are 4258 non overlapping blocks of size 8*8, 3 classes and 64 features and hence volume of the data is 4258.
Overlapping Data Set statistical information: image0 has 64 features and 93873 Observations, 384*256 dimension and labelled as 0. image0 has 64 features and 83913 Observations, 344*256 dimension and labelled as 1. image2 has 64 features and 81921 observations, 336*256 dimension and labelled as 2. Cumulatively, there are 259707 overlapping blocks of size 8*8, 3 classes and 64 features and hence the volume of the data is 259707

7.3 Mean and STD plot:

Figure 31 to 36 shows the Mean, STD plots and for image0, 1 and 2 for overlapping and Non overlapping data sets and the code used to generate the plots. Mean for image0 in Non overlapping Data set ranges between 198 to 200, image1 ranges between 103 to 104 and image2 ranges between 119 to 122. Mean for image0 in Overlapping Data set ranges between 198 to 200, image1 is 106, image2 ranges between 120 to 121. STD for image 0 in Non Overlapping Data set ranges between 43-44, image1 - 58, image2 - 65-66 STD for image 0 in Overlapping Data set ranges between 43, image1 - 58, image2 - 65-66

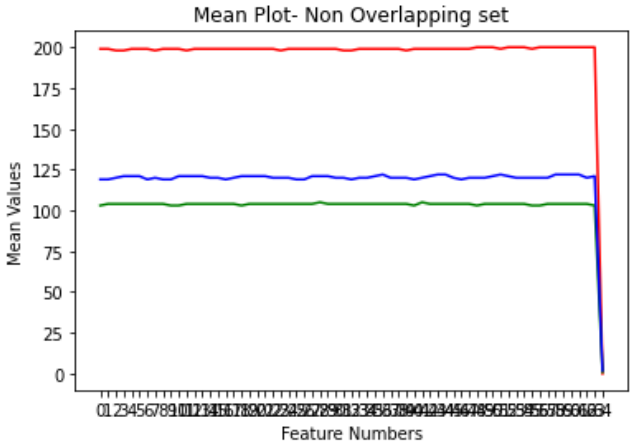


Figure 31: Mean Plot of all features for Non overlapping data set

Figure 32: Mean Plot of all features for overlapping data set

7.4 Scatter Plot:

Scatter plots are mainly used to observe and determine the relationship between two features. I have selected feature10 and feature30 from image0 initially to analyze if there is any correlation. In Figure 37, I have plotted feature 10 on X-axis and feature 30 on Y-axis.

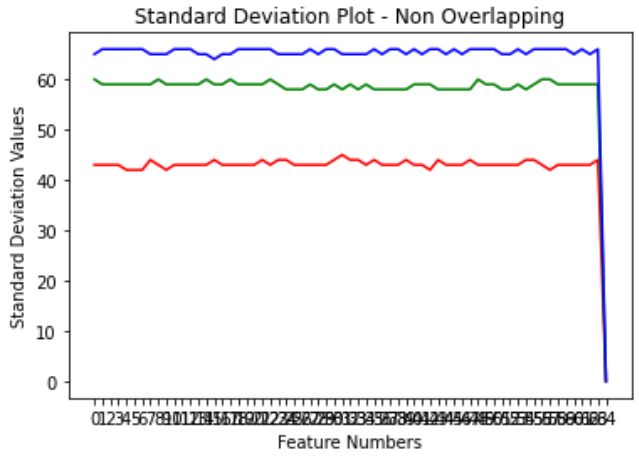


Figure 33: STD Plot of all features for Non overlapping data set

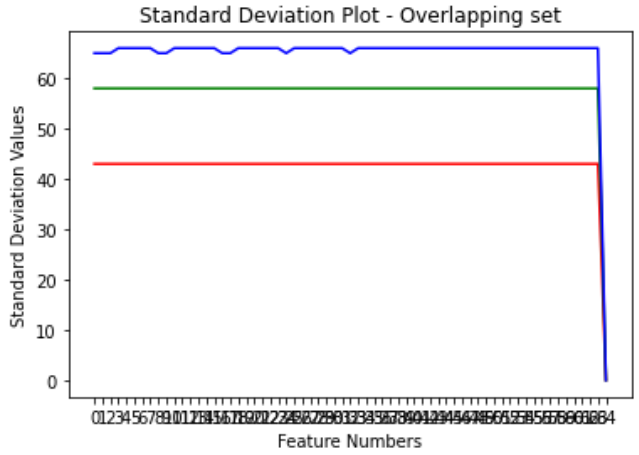
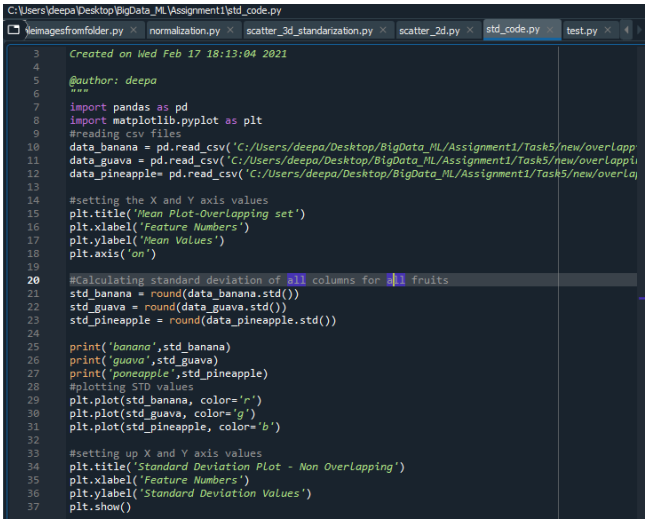


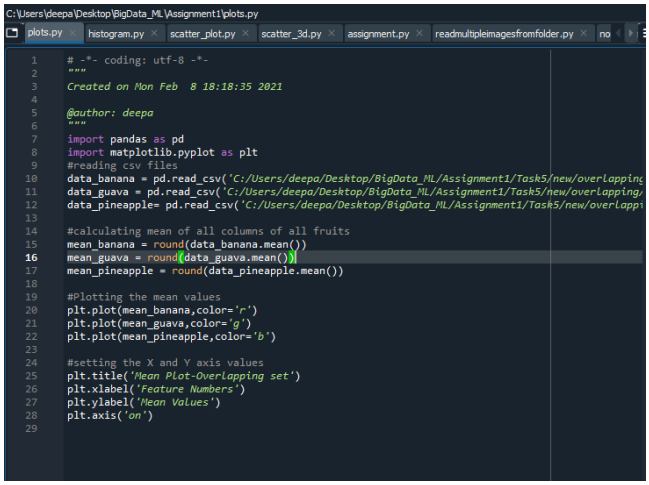
Figure 34: STD Plot of all features for Non overlapping data set



```
C:\Users\deepa\Desktop\BigData_ML\Assignment1\std_code.py
Created on Wed Feb 17 18:13:04 2021

@author: deepa
"""
import pandas as pd
import matplotlib.pyplot as plt
#reading csv files
data_banana = pd.read_csv('C:/Users/deepa/Desktop/BigData_ML/Assignment1/Task5/new/overlapp
data_guava = pd.read_csv('C:/Users/deepa/Desktop/BigData_ML/Assignment1/Task5/new/overlappi
data_pineapple= pd.read_csv('C:/Users/deepa/Desktop/BigData_ML/Assignment1/Task5/new/overla

#setting the X and Y axis values
plt.title('Mean Plot-Overlapping set')
plt.xlabel('Feature Numbers')
plt.ylabel('Mean Values')
plt.axis('on')

#Calculating standard deviation of all columns for all fruits
std_banana = round(data_banana.std())
std_guava = round(data_guava.std())
std_pineapple = round(data_pineapple.std())

print('banana',std_banana)
print('guava',std_guava)
print('poneapple',std_pineapple)
#plotting STD values
plt.plot(std_banana, color='r')
plt.plot(std_guava, color='g')
plt.plot(std_pineapple, color='b')

#setting up X and Y axis values
plt.title('Standard Deviation Plot - Non Overlapping')
plt.xlabel('Feature Numbers')
plt.ylabel('Standard Deviation Values')
plt.show()
```

Figure 35: Code to generate STD of all images

By looking at the plot, with the increase of x-axis values, there is increase in Y-axis values establishing a positive relation with each



```
C:\Users\deepa\Desktop\BigData_ML\Assignment1\plots.py
# -*- coding: utf-8 -*-
"""
Created on Mon Feb  8 18:18:35 2021

@author: deepa
"""
import pandas as pd
import matplotlib.pyplot as plt
#reading csv files
data_banana = pd.read_csv('C:/Users/deepa/Desktop/BigData_ML/Assignment1/Task5/new/overlappin
data_guava = pd.read_csv('C:/Users/deepa/Desktop/BigData_ML/Assignment1/Task5/new/overlapping/
data_pineapple= pd.read_csv('C:/Users/deepa/Desktop/BigData_ML/Assignment1/Task5/new/overlappi

#calculating mean of all columns of all fruits
mean_banana = round(data_banana.mean())
mean_guava = round(data_guava.mean())
mean_pineapple = round(data_pineapple.mean())

#Plotting the mean values
plt.plot(mean_banana,color='r')
plt.plot(mean_guava,color='g')
plt.plot(mean_pineapple,color='b')

#setting the X and Y axis values
plt.title('Mean Plot-Overlapping set')
plt.xlabel('Feature Numbers')
plt.ylabel('Mean Values')
plt.axis('on')
```

Figure 36: Code to generate Mean of all images

other with few outliers and the data is more concentrated at the top right. Figure 38 shows the same plot with overlapping data. Similarly, scatter plots are plotted for image1 (Feature 10 and Feature 30) and image2 (Feature 15 and Feature 30). From the plots in Figure 39(image1 scatter plot), I can see that it is following a moderate, linear relationship with small clustering at the bottom left. Figure 40 shows the similar plot for overlapping data set. For image3 (Figure 41 and Figure 42), I have plotted scatter plot for feature 15 and 30 which also shows moderate linear relationship with few outliers.

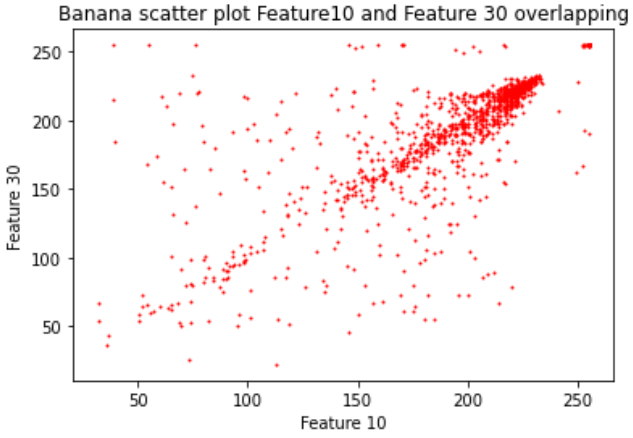


Figure 37: Scatter Plot of image0 feature10 and feature30 non sliding

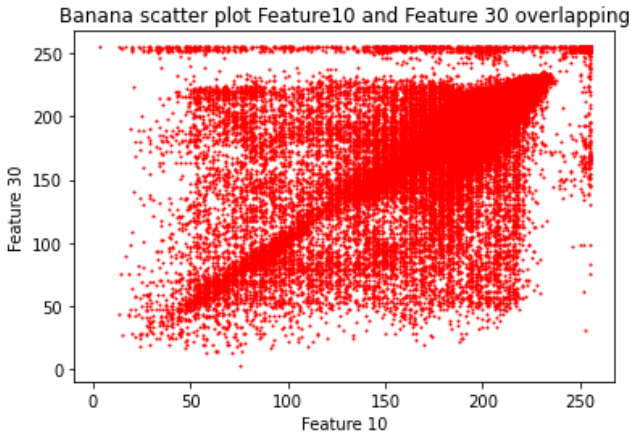


Figure 38: Scatter Plot of image0 feature10 and feature30 Overlapping

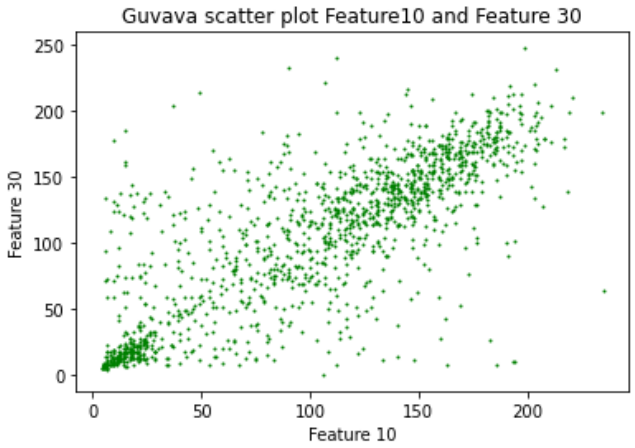


Figure 39: Scatter Plot of image1 feature10 and feature30 non sliding data

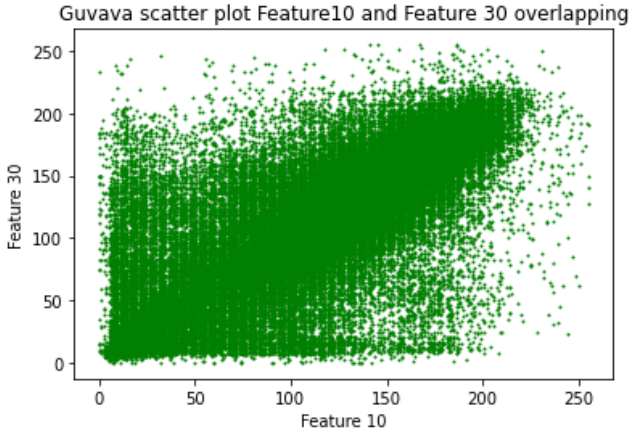


Figure 40: Scatter Plot of image1 feature10 and feature30 Overlapping data set

7.5 Histogram:

Histograms helps to visualize the distribution of the data numerically. I have plotted histograms to understand if the data follows Gaussian distribution or not. Figure 43 and Figure 44 are histogram

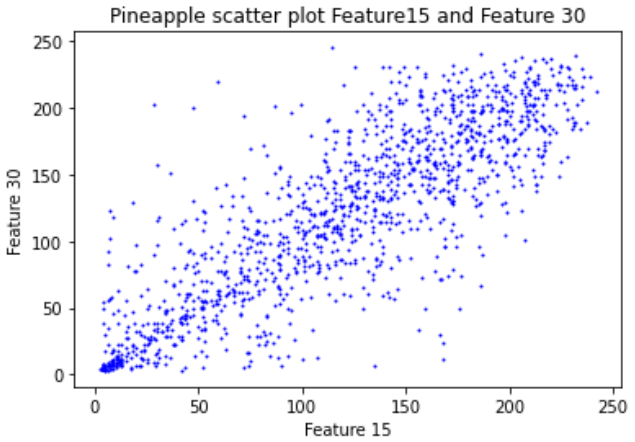


Figure 41: Scatter Plot of image2 feature15 and feature30 non sliding data

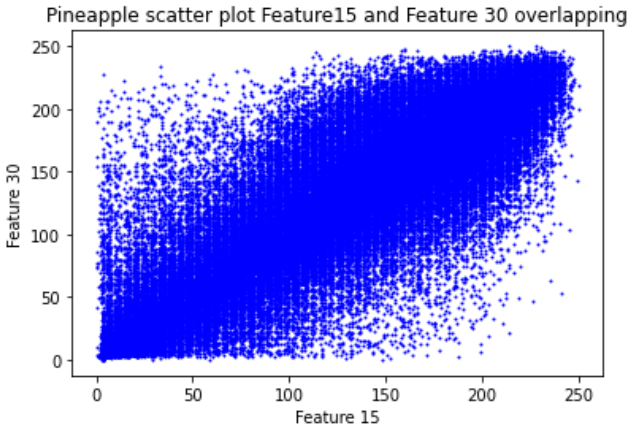


Figure 42: Scatter Plot of image2 feature15 and feature30 overlapping data set

plots for non sliding and sliding data set for image0 feature 10. The plot does not follow Gaussian distribution and is skewed on the left end and most of the data points are clustered on right end and the skewness could help differentiating classes. In histogram plot of Figure 45, feature 15, I can see that there are two peak values in non sliding data and data concentrated more in the middle. For image2, histogram plot for feature 15 in Figure 47 and 48, it is evident that the plot is almost equally distributed. So, none of the plots doesn't follow Gaussian distribution. Figure 49 has the code for generating histogram.

7.6 Is the data set imbalanced, inaccurate or incomplete?

The data set is imbalanced. For non sliding data set, the number of observations of image0 is greater than that of image1. Similarly, the observations of image1 is more than the observations in image2. For overlapping data set, image0 has maximum observations compared to image1 and image2. So it's an imbalanced data set. I have randomly selected feature 10 and 30 of image0 and feature15 and 25 from image0 and image2 to plot scatter plots. Figure 51 and 52 shows the plots obtained and they are imbalanced with some

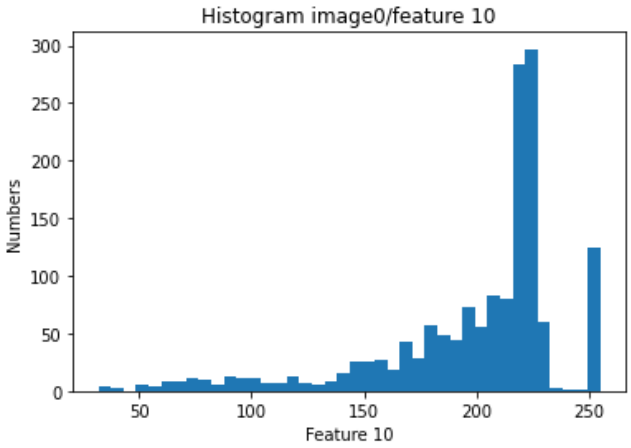


Figure 43: Histogram for feature10 of image0 non sliding data

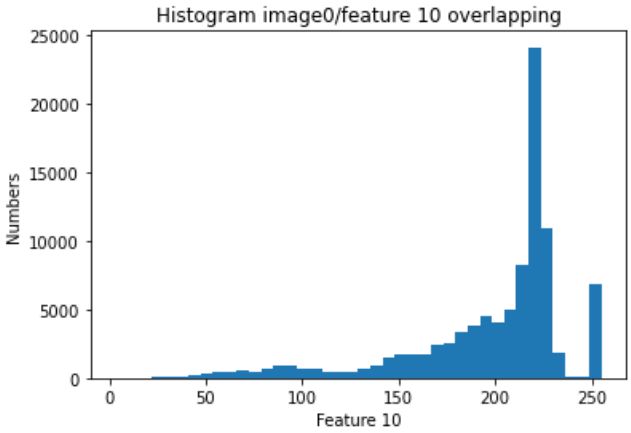


Figure 44: Histogram for feature10 of image0 non sliding data

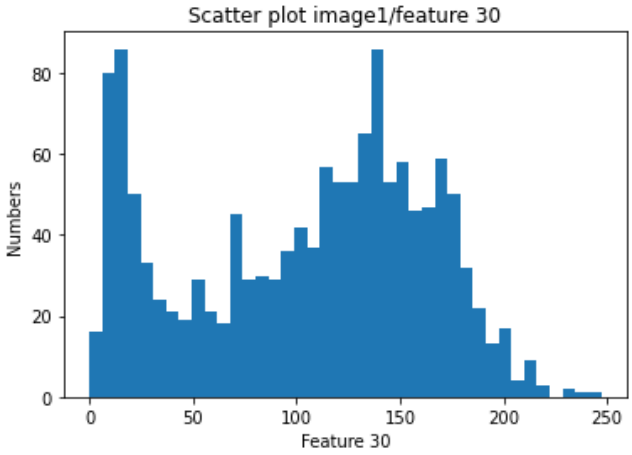


Figure 45: Histogram for feature 30 of image0 non sliding data

complex structures hidden. Also, none of the Histograms plotted follow Gaussian distribution, thus it makes them imbalanced data.

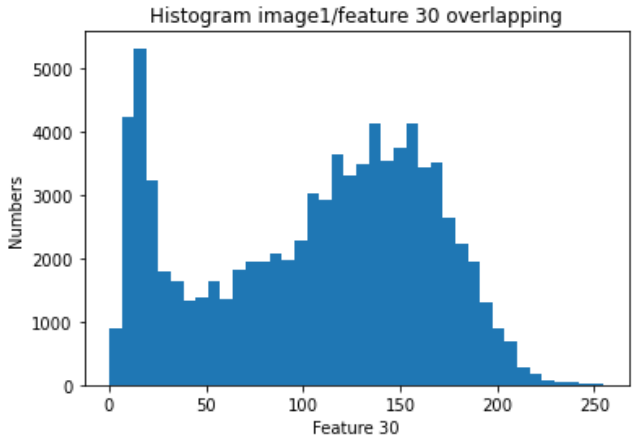


Figure 46: Histogram for feature 30 of image0 non sliding data

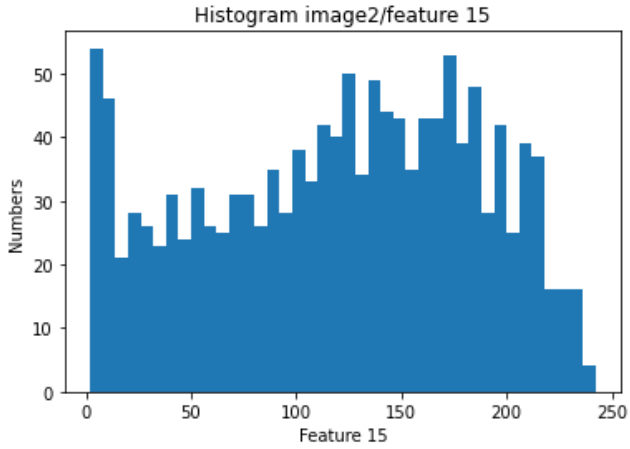


Figure 47: Histogram for feature10 of image0 non sliding data

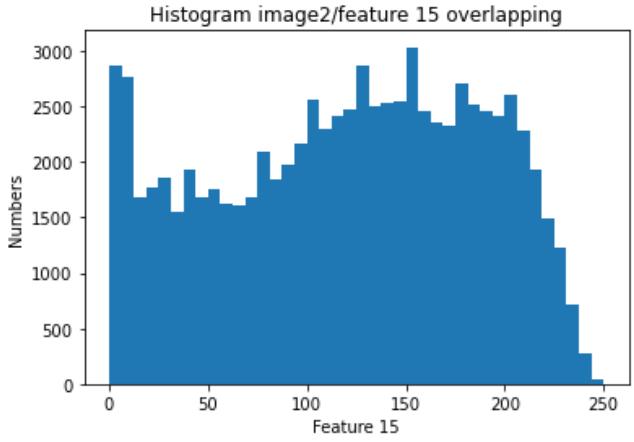


Figure 48: Histogram for feature10 of image0 non sliding data

7.7 Is it a trivial data or possibly a big data? Is it high dimensional?

This is trivial data and low dimensional because number of observations are less than number of features.

```
# -*- coding: utf-8 -*-
"""
Created on Wed Feb 10 19:15:46 2021

@author: deepa
"""
import pandas as pd
import matplotlib.pyplot as plt

data_banana = pd.read_csv('C:/Users/deepa/Desktop/BigData_ML/Assignment1/Task5/new/nonsliding/
data_guava = pd.read_csv('C:/Users/deepa/Desktop/BigData_ML/Assignment1/Task5/new/nonsliding/i
data_pineapple = pd.read_csv('C:/Users/deepa/Desktop/BigData_ML/Assignment1/Task5/new/nonslidi
f10=data_banana['10']
f30=data_guava['30']
p15=data_pineapple['15']
plt.xlabel('Feature 30')
plt.ylabel('Numbers')
plt.title('Scatter plot image1/feature 30')
nbins = 40
plt.hist(f30,nbins)
plt.show()
```

Figure 49: Code to generate Histogram

```
C:\Users\deepa\Desktop\BigData_ML\Assignment1\mergeframes.py
...\pleimagesfromfolder.py | normalization.py | scatter_3d_standarization.py | scatter_2d.py | std_code.py | test.py | mergeframes.py

# -*- coding: utf-8 -*-
"""
Created on Fri Feb 12 22:20:33 2021

@author: deepa
"""
import pandas as pd
import numpy as np

read_image0 = pd.read_csv('C:/Users/deepa/Desktop/BigData_ML/Assignment1/Task5/new/overlapping/image0.csv')
read_image1 = pd.read_csv('C:/Users/deepa/Desktop/BigData_ML/Assignment1/Task5/new/overlapping/image1.csv')
read_image2 = pd.read_csv('C:/Users/deepa/Desktop/BigData_ML/Assignment1/Task5/new/overlapping/image2.csv')
frames = [read_image0, read_image1]
mged = pd.concat(frames)
indx = np.arange(len(mged))
rndmged = np.random.permutation(indx)
rndmged=mged.sample(frac=1).reset_index(drop=True)
rndmged.to_csv('C:/Users/deepa/Desktop/BigData_ML/Assignment1/Task5/new/overlapping/image01.csv', index=False)
```

Figure 50: Code to merge and randomize feature space

7.8 Does it have scalability problem?

No, since it is a low dimensional data, there will not be any scalability problem and we know that scalability occur only in high-dimensional data.

7.9 Do you need to standardize? Do you need to normalize?

The data needs to be normalized since it does not follow Gaussian distribution.

7.10 How do they affect the data characteristics?

Data characteristics velocity, volume and variety seems to have no effect from above data.

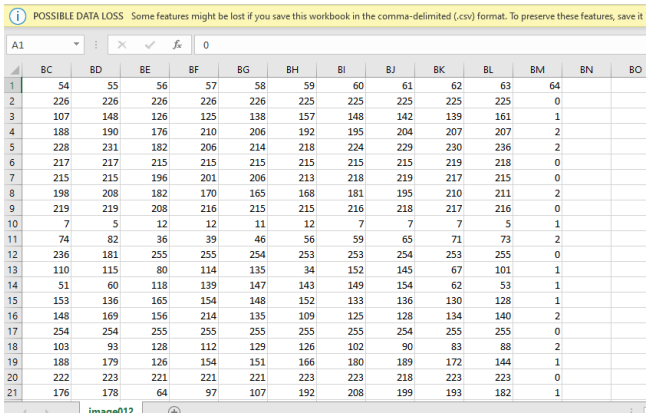
8 CONSTRUCT A FEATURE SPACE!

8.1 merge feature vectors

image0.csv and image1.csv files generated from Task5 are merged to generate image01.csv and image0.csv, image1.csv and image2.csv are merged to obtain image012.csv. The same procedure is repeated to generate merged feature space for both Overlapping and Nonoverlapping data sets.

8.2 Randomize the placement

Rows in the merged data sets are randomized by using the code presented in Figure 50 and screenshot of merged data is presented in Figure 51.



	BC	BD	BE	BF	BG	BH	BI	BJ	BK	BL	BM	BN	BO
1	54	55	56	57	58	59	60	61	62	63	64		
2	226	226	226	226	226	225	225	225	225	225	0		
3	107	148	126	125	138	157	148	142	139	161	1		
4	188	190	176	210	206	192	195	204	207	207	2		
5	228	231	182	206	214	218	224	229	230	236	2		
6	217	217	215	215	215	215	215	215	219	218	0		
7	215	215	196	201	206	213	218	219	217	215	0		
8	198	208	182	170	165	168	181	195	210	211	2		
9	219	219	208	216	215	215	216	218	217	216	0		
10	7	5	12	12	11	12	7	7	7	5	1		
11	74	82	36	39	46	56	59	65	71	73	2		
12	236	181	255	255	254	253	253	254	253	255	0		
13	110	115	80	114	135	34	152	145	67	101	1		
14	51	60	118	139	147	143	149	154	62	53	1		
15	153	136	165	154	148	152	133	136	130	128	1		
16	148	169	156	214	135	109	125	128	134	140	2		
17	254	254	255	255	255	255	255	254	255	255	0		
18	103	93	128	112	129	126	102	90	83	88	2		
19	188	179	126	154	151	166	180	189	172	144	1		
20	222	223	221	221	221	223	223	218	223	223	0		
21	176	178	64	97	107	192	208	199	193	182	1		

Figure 51: Merged-randomized feature space

9 DISPLAY SUBSPACES

9.1 Plot two-dimensional Feature space

For this task, I have chosen feature 10 and feature 30 from image0 and image1, feature 15 and feature 25 from image0 and image2 and plotted scatter plot by color coding red to image0, green to image1 and blue to image2. Generated scatter plot is as shown in Figure 52 and 53.

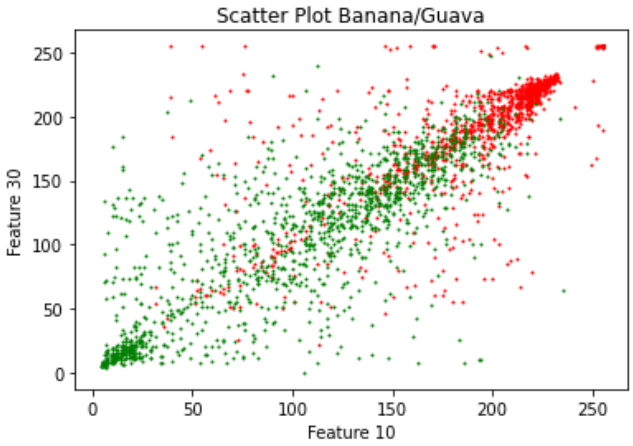


Figure 52: Two dimensional feature space image0 and image1 feature 10 and feature 30

9.2 Plot three-dimensional Feature space

For this task, I have chosen feature 10, feature 30 and feature 60 from image0, image1 and image2 and plotted scatter plot by labelling red as image0, green as image1 and blue as image2. Generated scatter plot is as shown in Figure 54.

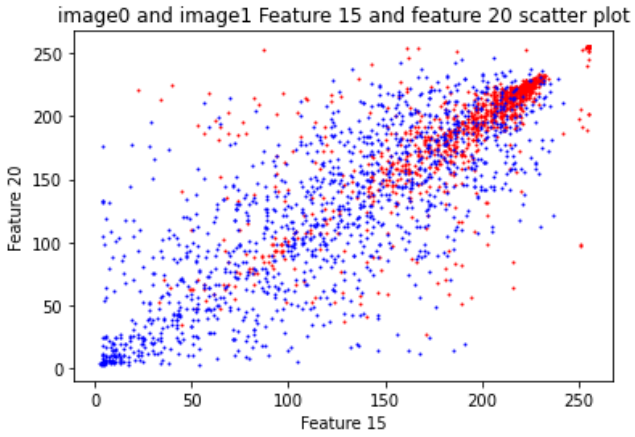


Figure 53: Two dimensional feature space image0 and image2 feature 15 and feature 25

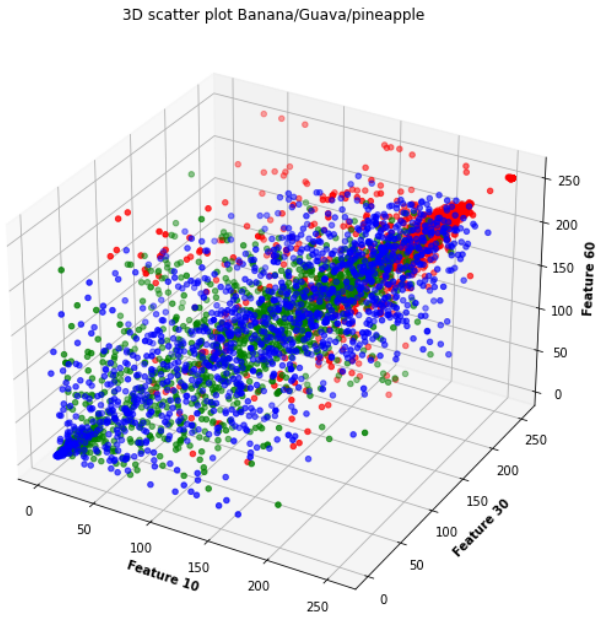


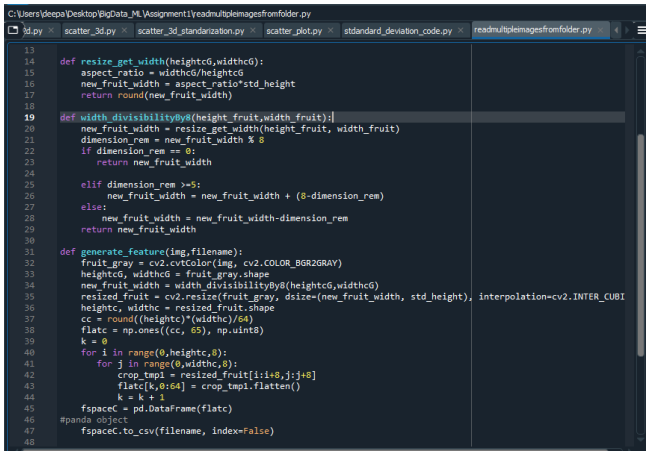
Figure 54: Three dimensional feature space

9.3 Discuss the figures:

By looking at the scatter plots in Figure 52, 53 and 54, the features are overlapping on each other making the class separation complex. Even though there are some clustering of image0 at the top right and image1 at bottom left, only that proof will not help in classification. The data needs to be randomized to make it balanced.

10 READ ANY NUMBERS OF IMAGES AND GENERATE FEATURE SPACE

Figure 55 shows the code use to read multiple images from folder and generate feature spaces. The code used in all the earlier tasks are combined and a new csv filename is generated and passed along with the image data to generate a feature space. For each image, csv file is generated with data appended with imagename. Figure 56



```
def resize_get_width(heightcG,widthcG):
    aspect_ratio = widthcG/heightcG
    new_fruit_width = aspect_ratio*std_height
    return round(new_fruit_width)

def width_divisibilityBy8(height_fruit,width_fruit):
    new_fruit_width = resize_get_width(height_fruit, width_fruit)
    dimension_rem = new_fruit_width % 8
    if dimension_rem == 0:
        return new_fruit_width

    elif dimension_rem >=5:
        new_fruit_width = new_fruit_width + (8-dimension_rem)
    else:
        new_fruit_width = new_fruit_width-dimension_rem
    return new_fruit_width

def generate_feature(img,filename):
    fruit_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    heightcG, widthcG = fruit_gray.shape
    new_fruit_width = width_divisibilityBy8(heightcG,widthcG)
    resized_fruit = cv2.resize(fruit_gray, dsize=(new_fruit_width, std_height), interpolation=cv2.INTER_CUBI
    heightc, widthc = resized_fruit.shape
    cc = round((heightc)*(widthc)/64)
    flatc = np.ones((cc, 65), np.uint8)
    k = 0
    for i in range(0,heightc,8):
        for j in range(0,widthc,8):
            crop_tmp1 = resized_fruit[i:i+8,j:j+8]
            flatc[k,0:64] = crop_tmp1.flatten()
            k = k + 1
    fspaceC = pd.DataFrame(flatc)
#panda object
    fspaceC.to_csv(filename, index=False)
```

Figure 55: Code to read multiple images from folder

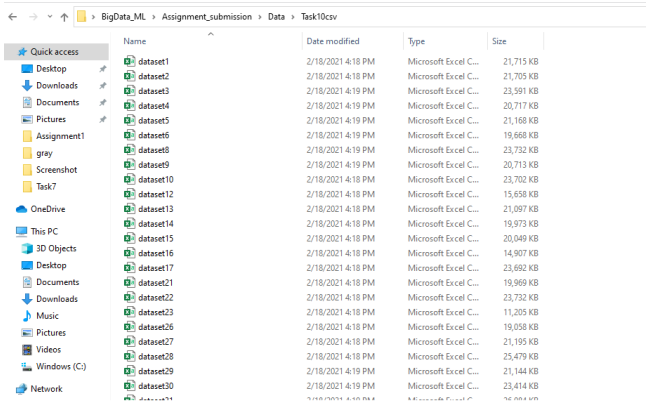


Figure 56: multiple csv files generated

shows the resulted overlapping data in csv files. Non overlapping data sets are generated and csv files are uploaded for reference.

11 DESCRIBE THE EFFECTS OF BLOCK SIZE ON DIMENSIONALITY

I have considered blocks of size 8*8 which generates feature vector of size 64. But, if I were to chose 4*4 blocks that would result in a 16 vector size and the number of observations would increase adding more volume and complexity to the data. More observations will make it difficult to understand and predict the data which will finally pose problems to classification of data.

REFERENCES

[1] Š. Marko, "Automatic fruit recognition using computer vision," *Mentor: Matej Kristan), Fakulteta za racunalništvo in informatiko, Univerza v Ljubljani*, 2013.

12 COMPLETE AND EXTEND THE TASKS OF ASSIGNMENT 1

All the tasks in Assignment 1 are completed. Overlapping feature vectors for two class, non-overlapping feature vectors for two class, overlapping feature vectors for three class and non-overlapping feature vectors for three class are prepared for carrying out Machine learning process.

12.1 Dividing the Data set to 80:20

For machine learning purpose, complete Data has to be divided into 80 percent as Training set and 20 percent has Testing set. This 80 percent data helps to Train the model so that we can Test the Model using the other 20 percent data. I have written Python code and saved the data sets as X_Train and X_Test csv files separately for image01 and image012 merged data sets. Figure 57 shows the code used to achieve the task and Fig 58 shows the generated files stored in a folder.

```
import pandas as pd
import numpy as np

image01_N = pd.read_csv('C:/Users/deepa/Desktop/BigData_ML/Assignment2/files/nonoverlap/image01.cs
image012_N = pd.read_csv('C:/Users/deepa/Desktop/BigData_ML/Assignment2/files/nonoverlap/image012.
image01_O = pd.read_csv('C:/Users/deepa/Desktop/BigData_ML/Assignment2/files/overlap/image01.csv',
image012_O = pd.read_csv('C:/Users/deepa/Desktop/BigData_ML/Assignment2/files/overlap/image012.csv


X = image012_N
row, col = X.shape
TR = round(row*0.8)
X1 = np.array(X)
TT = row-TR
X_train = X1[0:TR,:]
X_test = X1[TR:row,:]


data_train = pd.DataFrame(X_train)
data_train.to_csv('C:/Users/deepa/Desktop/BigData_ML/Assignment2/files/nonoverlap/X_Train_012.csv
data_test = pd.DataFrame(X_test)
data_test.to_csv('C:/Users/deepa/Desktop/BigData_ML/Assignment2/files/nonoverlap/X_Test_012.csv',
```

Figure 57: Code executed to generate Train and Test Data set

s PC › Desktop › BigData_ML › Assignment2 › files › nonoverlap

Name	Date modified	Type	Size
image0	3/10/2021 11:09 AM	Microsoft Excel C...	339 KB
image01	3/10/2021 12:09 PM	Microsoft Excel C...	669 KB
image1	3/10/2021 12:07 PM	Microsoft Excel C...	330 KB
image2	2/12/2021 9:56 PM	Microsoft Excel C...	305 KB
image012	3/12/2021 9:31 AM	Microsoft Excel C...	973 KB
X_Test_01	3/19/2021 12:24 PM	Microsoft Excel C...	134 KB
X_Test_012	3/19/2021 10:13 AM	Microsoft Excel C...	196 KB
X_Train_01	3/19/2021 12:18 PM	Microsoft Excel C...	535 KB
X_Train_012	3/19/2021 10:30 AM	Microsoft Excel C...	778 KB

Figure 58: Folder where Train and Test Data sets are saved locally

12.2 Draw histograms of Two features from Training and Test Data set

I have selected Feature 30 and Feature 60 as there was no huge variation in the mean and Standard deviation of all features. From Figure 59 to 62, we can observe that histogram of Train data set of overlapping/non-overlapping of two class and overlapping and non-overlapping data set of three class follow the same distribution as that of their respective Test data sets. The Mean and Standard deviation of the features are calculated and a comparison table is drawn in Figure 63 and 64.

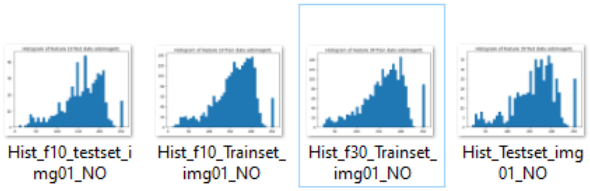


Figure 59: Histogram of Feature 10 and Feature 30 01 Non oVerlapping Data set

Figure 60: Histogram of Feature 10 and Feature 30 01 oVerlapping Data set

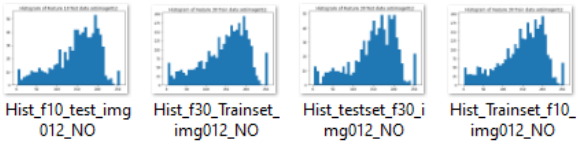


Figure 61: Histogram of Feature 10 and Feature 30 012 Non oVerlapping Data set

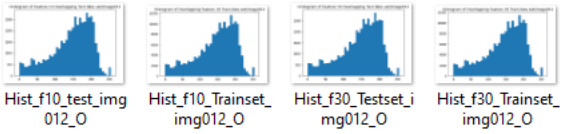


Figure 62: Histogram of Feature 10 and Feature 30 012 oVerlapping Data set

12.3 Draw Scatter plot of Two features from Training and Test Data set

Figure 65 to 72 to shows scatter plot of Feature 10 and 30 for non-overlapping and overlapping data sets.

13 IMPLEMENTING REGRESSION BASED MODEL

13.1 Implementing Lasso Regression Model

I have implemented Lasso regression model. Trained the model using X_Train_01 sets of overlapping and non overlapping data sets generated in earlier task. Training generates the coefficients for each feature/column in the data set. These coefficients are multiplied with feature matrix to get the predicted labels. Initially lamda is set to 0 during execution and increased up to 17. But the accuracy of the model started reducing after 17, so I have kept the lamda

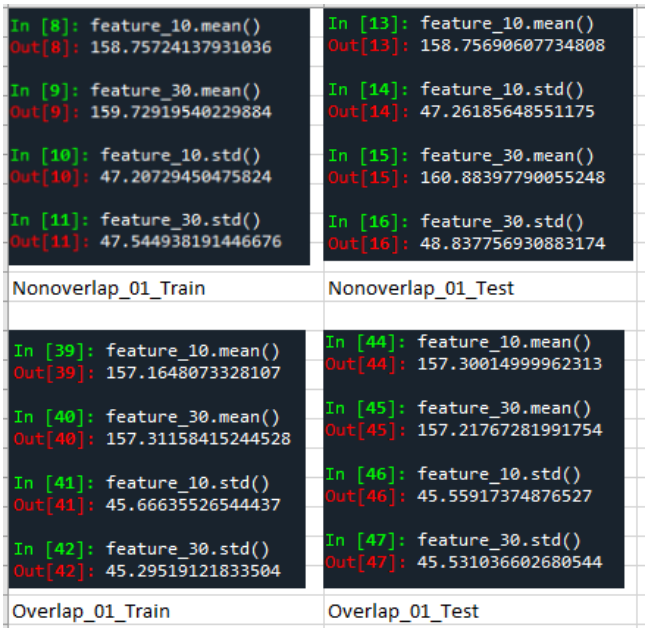


Figure 63: Comparison Table of Mean and STD for image01 data set

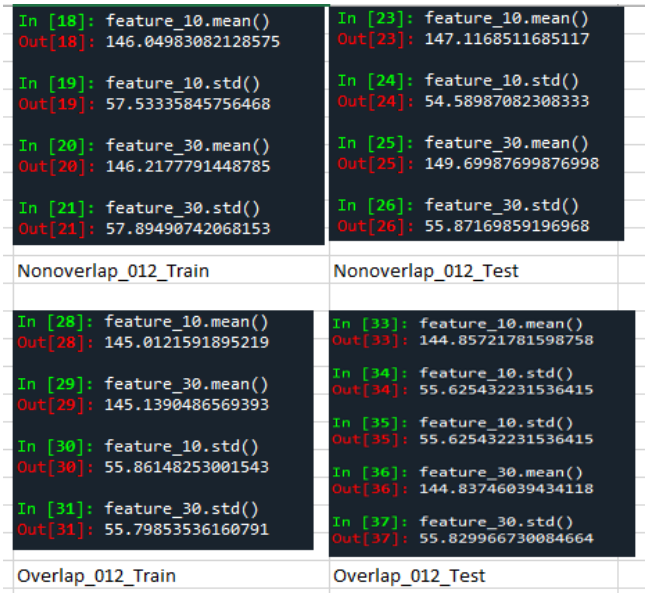


Figure 64: Comparison Table of Mean and STD for image012 data set

value as 17 for analysing accuracy measures. Figure 73 shows the screenshot of code and execution in the local environment.

13.2 Testing and adding predicted labels

In the above step model is trained using Train data set and now I am testing the accuracy of the model for Test data set. Testing is done by matrix multiplication of parameters or coefficients obtained through Training and X_Test_01 feature vectors. This process gives us the predicted labels for Test data. Predicted labels are saved in a lasso_predicted.csv file and later placing them as 66th column in X_Test_01 files of both overlapping and non-overlapping data sets. I started off the testing by initially setting the lamda value to 0 and

Figure 65: Scatter Plot 01 Non overlapping Train Dataset

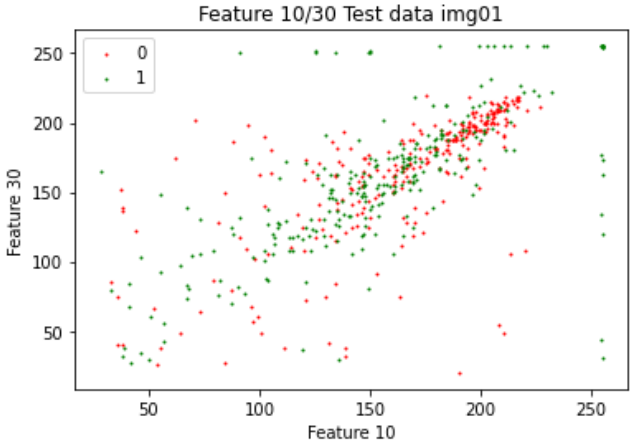


Figure 66: Scatter Plot 01 overlapping Test Dataset

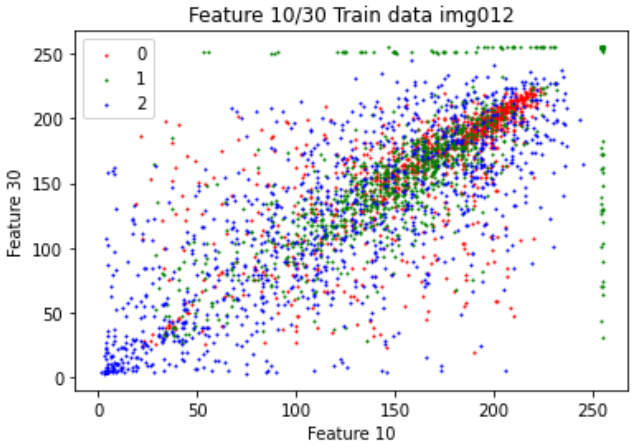


Figure 67: Scatter Plot 012 Non overlapping Train Dataset

gradually increasing up to 17. But, there was only slight increase in the accuracy level and it started decreasing after value 17.

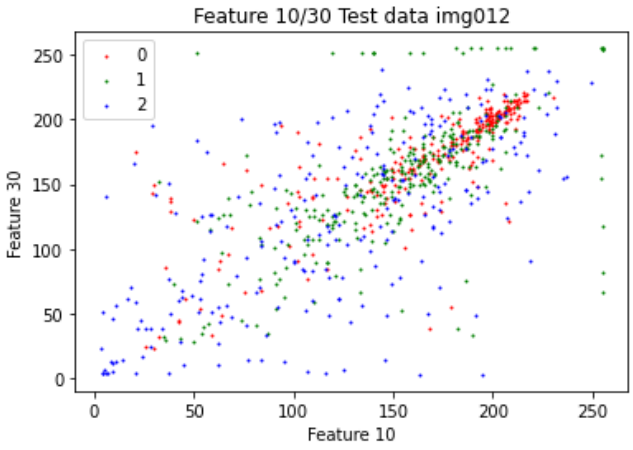


Figure 68: Scatter Plot 012 Non overlapping Train Dataset

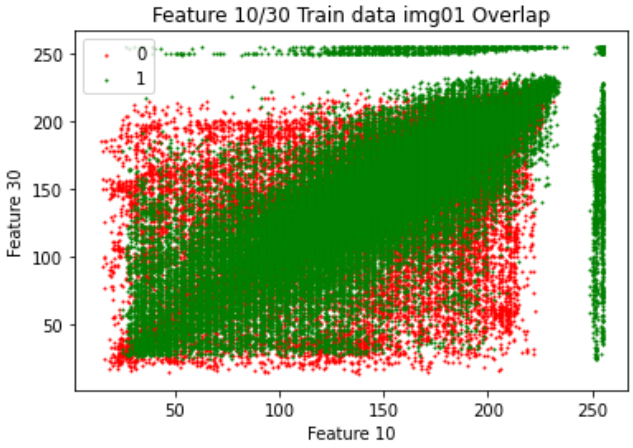


Figure 69: Scatter Plot of img01 Overlapping Train Data set

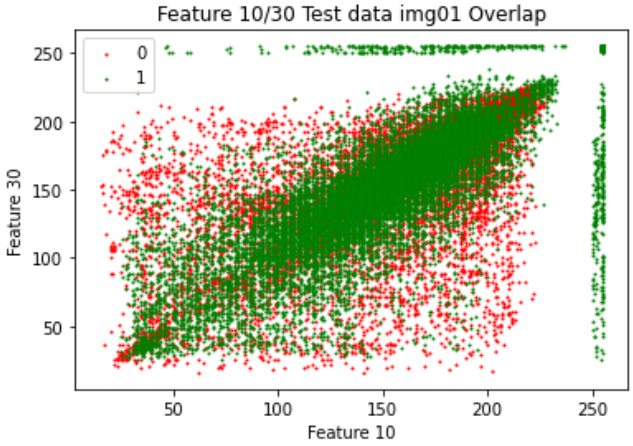


Figure 70: Scatter Plot of img01 Overlapping Test Data set

13.3 Create Confusion Matrices

Confusion matrix is created by passing 65th column and 66th column generated in previous task to the confusion matrix function of sklearn.metrics library. The matrix is printed on the command

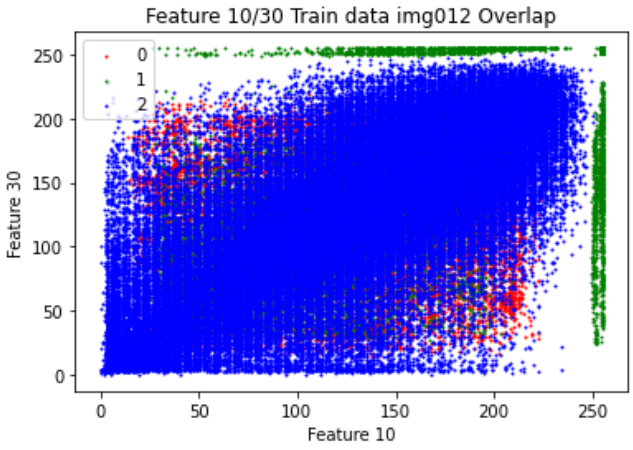


Figure 71: Scatter Plot of img012 Overlapping Train Data set

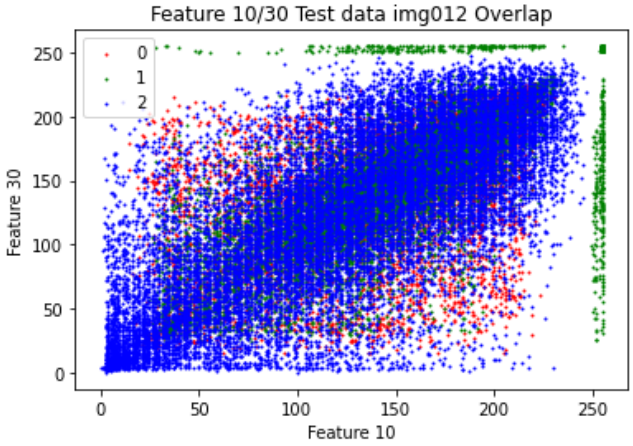


Figure 72: Scatter Plot of img012 Overlapping Test Data set

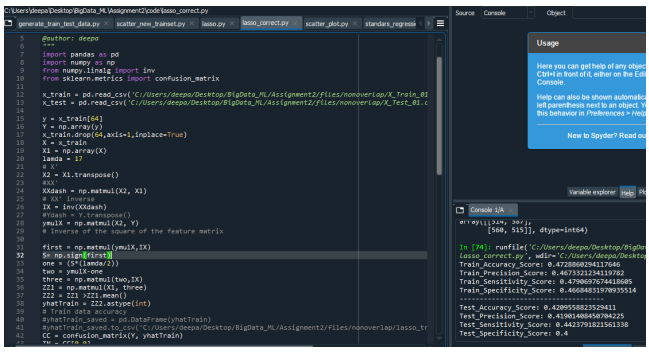
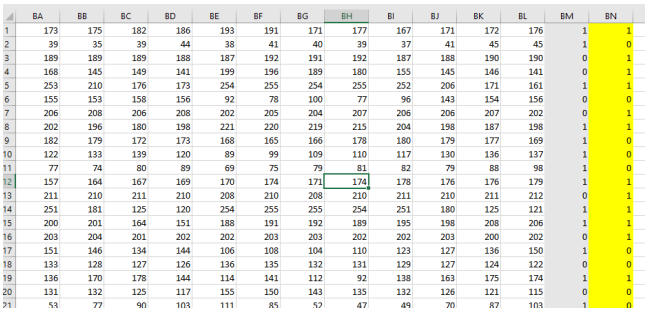


Figure 73: Lasso regression execution in local environment

prompt. Figure 76 shows the code of confusion matrix. I am considering class 0 as the True positive class and qualitative analysis is performed based on it.

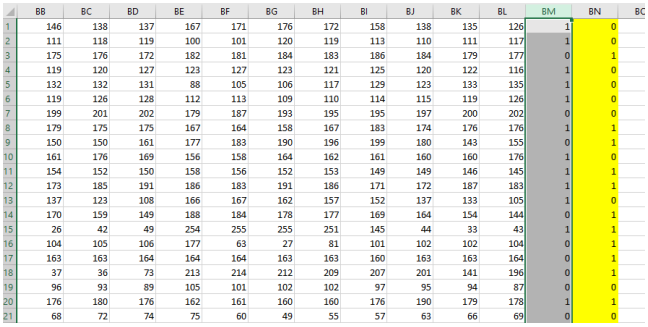
13.4 Comparing performance of Data sets

By using Confusion matrix, accuracy, precision, sensitivity and specificity is calculated for nonoverlapping and overlapping sets



	BA	BB	BC	BD	BE	BF	BG	BH	BI	BJ	BK	BL	BM	BN
1	173	175	182	186	193	191	171	177	167	171	172	176	1	1
2	39	35	39	44	38	41	40	39	37	41	45	45	1	0
3	189	189	189	188	187	192	191	192	187	188	190	190	0	1
4	168	145	149	141	199	196	189	180	155	145	146	141	0	1
5	253	210	176	173	254	255	254	255	252	206	171	161	1	1
6	155	153	158	156	92	78	100	77	96	143	154	156	0	0
7	206	208	206	208	202	205	204	207	206	206	207	202	0	1
8	202	196	180	198	221	220	219	215	204	198	187	198	1	1
9	182	179	172	173	168	165	166	178	180	179	177	169	1	0
10	122	133	139	120	89	99	109	110	117	130	136	137	1	0
11	77	74	80	89	69	75	79	81	82	79	88	98	1	0
12	157	164	167	169	170	174	171	174	178	176	176	179	1	1
13	211	210	211	210	208	210	208	210	211	210	211	212	0	1
14	251	181	125	120	254	255	255	254	251	180	125	121	1	1
15	200	201	164	151	188	191	192	189	195	198	208	206	1	1
16	203	204	201	202	202	203	203	202	202	203	200	202	0	1
17	151	146	134	144	106	108	104	110	123	127	136	150	1	0
18	133	128	127	126	136	135	132	131	129	127	124	122	0	0
19	136	170	178	144	114	141	112	92	138	163	175	174	1	1
20	131	132	125	117	155	150	143	135	132	126	121	115	0	0
21	53	77	90	103	111	85	52	47	49	70	87	103	1	0

Figure 74: Predicted labels comparison Non overlapping set



	BB	BC	BD	BE	BF	BG	BH	BI	BJ	BK	BL	BM	BN	BO
1	146	138	137	167	171	176	172	158	138	135	126	1	0	
2	111	118	119	100	101	120	119	113	110	111	117	1	0	
3	175	176	172	182	181	184	183	186	184	179	177	0	1	
4	119	120	127	123	127	123	121	125	120	122	116	1	0	
5	132	132	131	88	105	106	117	129	123	133	135	1	0	
6	119	126	128	112	113	109	110	114	115	119	126	1	0	
7	199	201	202	179	187	193	195	195	197	200	202	0	0	
8	179	175	175	167	164	158	167	183	174	176	176	1	1	
9	150	150	161	177	183	190	196	199	180	143	155	0	1	
10	161	176	169	156	158	164	162	161	160	160	176	1	0	
11	154	152	150	158	156	152	153	149	149	146	145	1	1	
12	173	185	191	186	183	191	186	171	172	187	183	1	1	
13	137	123	108	166	167	162	157	152	137	133	105	1	0	
14	170	159	149	188	184	178	177	169	164	154	144	0	1	
15	26	42	49	254	255	255	251	145	44	33	43	1	1	
16	104	105	106	177	63	27	81	101	102	102	104	0	1	
17	163	163	164	164	164	163	163	160	163	163	164	0	1	
18	37	36	73	213	214	212	209	207	201	141	196	0	1	
19	96	93	89	105	101	102	102	97	95	94	87	0	0	
20	176	180	176	162	161	160	160	176	190	179	178	1	1	
21	68	72	74	75	60	49	55	57	63	66	69	0	0	

Figure 75: Predicted labels comparison Non overlapping set

Figure 76: Confusion matrix code

of image01 data set. I am using accuracy and sensitivity for performance evaluation. By observing the accuracy and precision in Figure 77 and 78, performance of nonoverlapping set is better than that of overlapping dataset. For nonoverlapping data set, test accuracy and precision is measured to be 42% and 41% respectively and for overlapping data accuracy is measured to be 41.025% and precision at 41%.

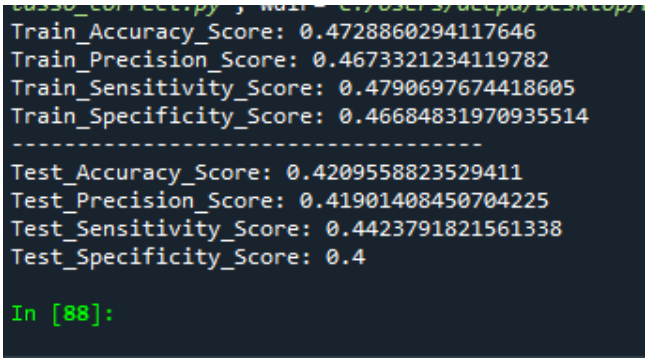


Figure 77: Performance of Train and Test set Nonoverlap

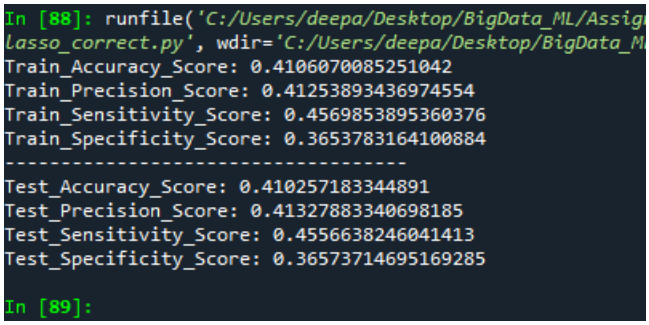


Figure 78: Performance of Train and Test set Overlap

14 IMPLEMENT RANDOM FOREST OR DEEP LEARNING

I am implementing Random Forest as two class and three class classifier.

14.1 Training Random forest for two class classifier

I have used RandomClassifier from sklearn.ensemble library to perform two class classification on merged data set of image0 and image1. Train set X_Train_01 created in task 12 is used to train the RandomClassifier. Figure shows the execution of the random classifier model. Model is trained for multiple n_estimators(number of trees) starting from 50, 100, 500 till 1000. Performance evaluation of all these multiple is discussed in task 15. Figure 80 shows the sample Tree generated for Non-overlapping image01 data set with n_estimators as 50 and figure 81 is the zoomed version of small part of the tree just to show the values calculated during splitting.

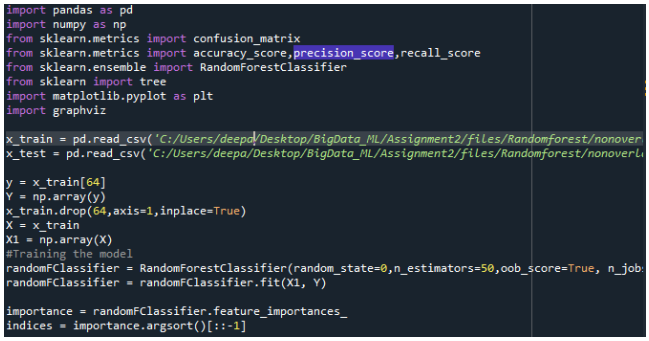


Figure 79: Training the Random Classifier Code snippet

14.2 Training Random forest for three class classifier

Merged data of image0, image1 and image2 is used for Training the model as Three class classifier. Train data set X_Train_012 of both overlap and non-overlap data set is used to Train the model. In a similar way to training the two class model, this model has also been trained for different estimator values and the best model is chosen. Figure 82 shows the code snippet used for execution.

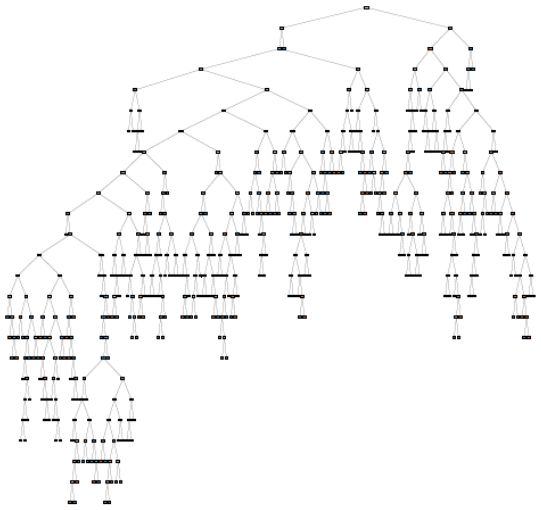


Figure 80: Random Forest tree for non overlapping data as two class classifier

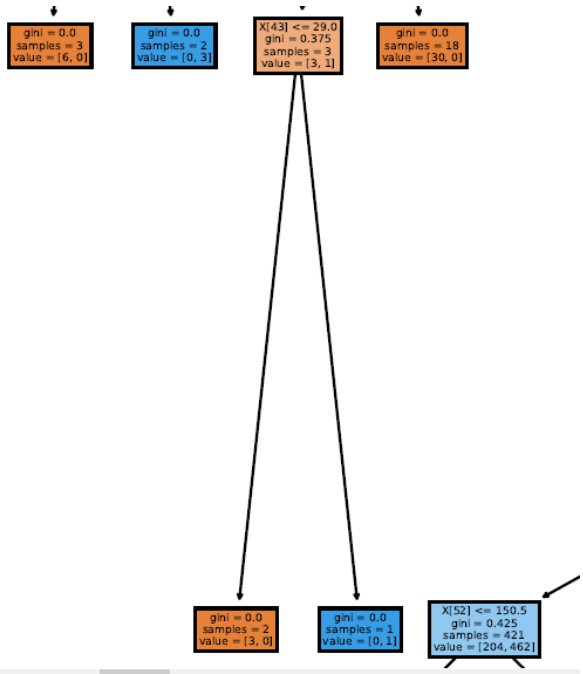


Figure 81: Zoomed in tree

14.3 Applying trained model to Test data

Trained data of both two class and three class classifier is applied to Test data to evaluate the performance of the model. Figure 83 has the code used for testing. Model is trained and tested by changing the n_estimators like 50, 100, 500 and 1000 to see the variations in performance. The best model's predicted labels are saved in 66th column of Test set.

```
import pandas as pd
import numpy as np
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score,precision_score,recall_score
from sklearn.metrics import classification_report
from sklearn.ensemble import RandomForestClassifier

x_train = pd.read_csv('C:/Users/deepa/Desktop/BigData_ML/Assignment2/files/Randomforest/overlap/
x_test = pd.read_csv('C:/Users/deepa/Desktop/BigData_ML/Assignment2/files/Randomforest/overlap/

y = x_train[64]
Y = np.array(y)
x_train.drop(64,axis=1,inplace=True)
X = x_train
X1 = np.array(X)
#Training the model
randomFClassifier = RandomForestClassifier(random_state=0,n_estimators=1000,oob_score=True, n_j
model = randomFClassifier.fit(X1, Y)

importance = model.feature_importances_
indices = importance.argsort()[::-1]

oob_error = 1- randomFClassifier.oob_score_
```

Figure 82: Training the Random Classifier Code snippet for three class classifier

```
#Testing the model using Trained results0
test_y = x_test[64]
Y_test = np.array(test_y)
x_test.drop(64,axis=1,inplace=True)
x_test_np = np.array(x_test)
y_pred = randomFClassifier.predict(x_test_np)
#yhat_saved = pd.DataFrame(y_pred)
#yhat_saved.to_csv('C:/Users/deepa/Desktop/BigData_ML/Assignment2/files/Randomforest/nonoverlap,
```

Figure 83: Testing the model code snippet

14.4 Constructing confusion matrices

After testing the model, the predicted results in 66th column and original values in 65th column are fed to Confusion matrix. I am taking class 0 as the true positive class and the evaluation is done based on that. Figure 84 and 85 shows the code snippet for both two class and three class Confusion matrices respectively.

```
CC_test = confusion_matrix(test_y, y_pred)
# 0 as my positive class
TN = CC_test[1,1]
FP = CC_test[1,0]
FN = CC_test[0,1]
TP = CC_test[0,0]
```

Figure 84: Constructing Confusion matrices code snippet Two class

```
TP = CC_test[0,0]
TN = CC_test[1,1]+CC_test[1,2]+CC_test[2,1]+CC_test[2,2]
FP = CC_test[0,1]+CC_test[0,2]
FN = CC_test[1,0] + CC_test[2,0]
```

Figure 85: Constructing Confusion matrices code snippet three class

14.5 Performance evaluation of the random forest model

After constructing the confusion matrices, the important task is evaluating the model for it's effectiveness in terms on accuracy, precision, specificity and sensitivity. All the classifiers i.e two class and three class are trained and tested by setting number of trees option to 50, 100, 500 and 1000. The results obtained are shown in Figure 86 to Figure 88. Figure shows the two class classifier for non overlapping data set. Here we can see that as the number of estimators increases, performance was decreasing slightly. In contrast overlapping data set performed much better with increasing estimators. In three class non overlapping data set in Figure 86 demonstrates when estimator value is 100 it is giving good results and as the estimator value increased performance got slightly down. For three class overlapping classifier, as the estimators increased,

performance increased a bit, but got stagnant after that. So post 500 estimator, performance was as it was. I am saving the best results of all the four as 66th Column of the data set. On comparison from Figure 88, overlap data set is giving better results than non-overlapping. So the best model is the model with 1000 estimators applied on overlap data set for two class and 500 estimators for three class classifier applied on overlapping data set.

	TWO CLASS NONOVERLAP(01)			
Number of Estimators	ACCURACY	PRECISION	SENSITIVITY	SPECIFICITY
50	0.803	0.791	0.817	0.789
100	0.803	0.793	0.814	0.792
500	0.801	0.792	0.81	0.792
1000	0.799	0.789	0.81	0.789
	THREE CLASS NONOVERLAP(012)			
Number of Estimators	ACCURACY	PRECISION	SENSITIVITY	SPECIFICITY
50	0.81	0.714	0.7148	0.857
100	0.821	0.721	0.74	0.861
500	0.817	0.698	0.747	0.849
1000	0.814	0.692	0.746	0.846

Figure 86: Performance evaluation of Two and three class Non overlap set

	TWO CLASS OVERLAP(01)			
Number of Estimators	ACCURACY	PRECISION	SENSITIVITY	SPECIFICITY
50	0.923	0.911	0.937	0.91
100	0.926	0.912	0.942	0.91
500	0.929	0.911	0.948	0.91
1000	0.929	0.912	0.949	0.91
	THREE CLASS OVERLAP(012)			
Number of Estimators	ACCURACY	PRECISION	SENSITIVITY	SPECIFICITY
50	0.937	0.882	0.926	0.942
100	0.94	0.883	0.934	0.942
500	0.942	0.886	0.938	0.944
1000	0.942	0.886	0.939	0.944

Figure 87: Performance evaluation of Two and three class Overlap set

TYPE OF FEATURE SPACE	ACCURACY	PRECISION	SENSITIVITY	SPECIFICITY
TWO CLASS NONOVERLAP(01)	0.803	0.791	0.817	0.789
TWO CLASS OVERLAP(01)	0.929	0.912	0.949	0.91
THREE CLASS NONOVERLAP(012)	0.821	0.721	0.74	0.861
THREE CLASS OVERLAP(012)	0.942	0.886	0.938	0.944

Figure 88: Finding the best model of Random Search

15 EVALUATION OF THE LEARNING MODELS

15.1 Evaluating the Learning models using In-built libraries

For this task, I am using inbuilt library accuracy_score, precision_score, recall_score and classification_report from sklearn.metrics to evaluate the performance of all four types of data sets. Code snippet showing the execution of Lasso regression for in-built performance evaluation and few results are shown from Figure 89 to 91. All other data sets are executed in same way to get the results.

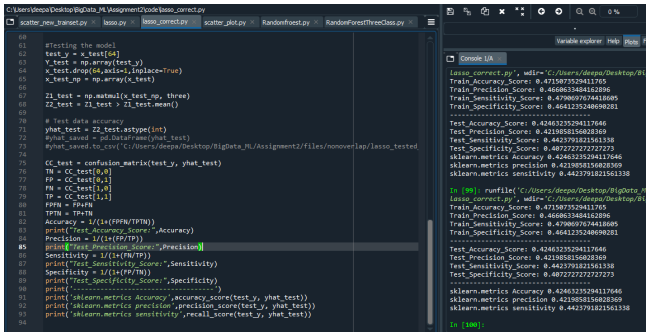


Figure 89: Code snippet and execution of Performance evaluation using sklearn,metrics libraries

	precision	recall	f1-score	support
0	0.94	0.88	0.91	16666
1	0.89	0.92	0.91	16312
2	0.93	0.97	0.95	16573
accuracy			0.92	49551
macro avg	0.92	0.92	0.92	49551
weighted avg	0.92	0.92	0.92	49551

Figure 90: Results of Random Forest as three class classifier

	precision	recall	f1-score	support
0	0.74	0.72	0.73	272
1	0.76	0.70	0.73	291
2	0.82	0.90	0.86	250
accuracy			0.77	813
macro avg	0.77	0.78	0.77	813
weighted avg	0.77	0.77	0.77	813

Figure 91: Results of Random Forest as three class classifier Overlap data set

15.2 Comparing the performance of models using inbuilt accuracy measures and confusion-based

The chart showing the performance of Lasso Regression and Random forest Confusion using Confusion matrix and in-built library is shown in Figure 92. Looking at the results in figure 92, there is only slight difference in values generated by confusion based an in-built metrics. Confusion based metrics has given better accuracy levels compared to in-built metrics. For precision, sometimes there has been exact same results and sometimes in-built metrics has given better numbers.

	Confusion Based	In-built	Confusion Based	In-built	Confusion Based	In-built
TYPE OF FEATURE SPACE	ACCURACY	ACCURACY	PRECISION	PRECISION	SENSITIVITY	SENSITIVITY
RF TWO CLASS NONOVERLAP(01)	0.803	0.803	0.791	0.79	0.817	0.817
RF TWO CLASS OVERLAP(01)	0.929	0.929	0.912	0.912	0.949	0.949
RF THREE CLASS NONOVERLAP(01	0.821	0.77	0.721	0.74	0.74	0.72
RF THREE CLASS OVERLAP(012)	0.942	0.92	0.886	0.94	0.938	0.88
TWO CLASS NON OVERLAP LASSO	0.4246	0.424	0.421	0.421	0.442	0.442
TWO CLASS OVERLAP LASSO	0.41	0.41	0.413	0.413	0.455	0.455

Figure 92: Chart for comparing performance of all models Non overlapping data set

15.3 Comparing the model to find the best one

From Figure 92, we can see that random forest model is performing better than Lasso regression model. There is only slight difference in the calculations of in-built Confusion based metrics and imported libraries. In most cases, both of them gave the same results. Random forest gave best results for overlapping data set compared to non-overlapping data set. Random forest model trained using 500 estimators for overlapping data for three class classifier was the best model among all others with 94.2% accuracy.

16 ASSIGNMENT 3: A CLASSIFIER ON A BIG DATA SYSTEM

17 MAKING SURE ALL TASKS OF ASSIGNMENT 1 AND ASSIGNMENT 2 ARE COMPLETED

All the below tasks related to assignment 1 and assignment 2 are completed.

- In assignment 1, all four feature vectors/ data frames of two class non-overlapping, two class overlapping, three class non-overlapping and three class overlapping are generated and stored in respective csv files.
- In assignment 2, implementation of Lasso regression as two class classifier and random forest as two and three class classifier is performed in local environment and the best model and the best feature set is selected.
- In addition to above tasks, I have generated single csv file with multiple feature features generated from folder of image0 (banana) and image1 (guava). Then, I have merged all these multiple feature vectors to generate a large volume data set to experiment on Big data environment. So, this data frame is prepared with around 40 banana and 40 guava images I took from FIDS30 folder. I will be trying to run this data around 1 GB in local as well as big data environment to analyze the performance.

18 UNDERSTANDING AND USING BIGDATA SYSTEM USING DATABRICKS SYSTEM

18.1 Signing up Databricks community version

Created my account by signing up to Databricks community using UNCG email id. I chose Google Cloud platform as the backend.

18.2 Exploring Quickstart tutorial and setting up Big data environment in Databricks community

I created my community Databricks account by choosing gcp as backend platform which gives around 15GB of free memory. Then, created cluster CSC610 using "Create Cluster" option. I chose 7.4 ML version apache spark 3.0.1, scala 2.12 as there was no 7.3 version present in the list. It took around 5 minutes for the cluster to be up and running. There were two options available to create tables, one was using UI , other was using notebook. I uploaded all the data set generated in assignment 1 and created the tables using notebook option since there was an internal error thrown while trying to create table using UI. Figure 95 and 97 shows the creation

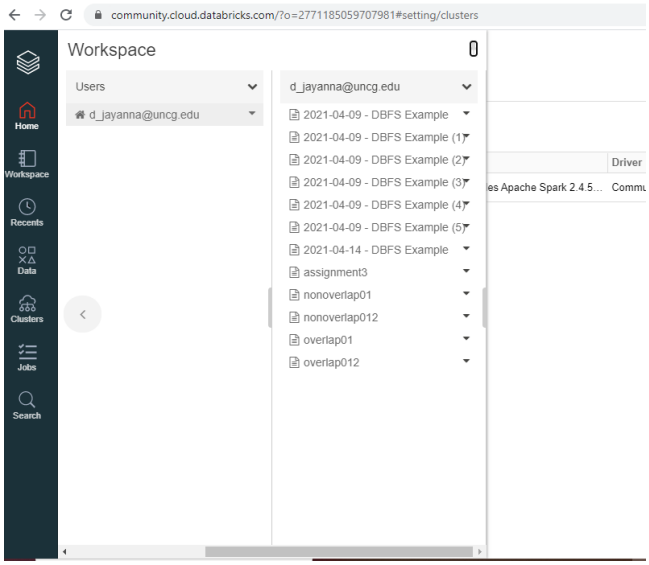


Figure 93: Account created in Databricks

of tables using notebook. I first selected the required parameters like selecting the cluster, table name, first row header checkbox and infer schema checked using UI option itself. After selecting all these parameters when I hit create table, an Internal error was thrown and it persisted for other table creations as well. So, I chose create table with notebook and ran all the queries to generate tables. I followed the same procedure to create four tables by uploading 4 csv file, overlapping two class and three class and non overlapping two class and three class.Next step was to create notebook in order to write the code and run the models. I created 4 different notebooks to execute models all 4 types of dataset.

18.3 Cluster creation:

Cluster is created with 7.4 ML version apache spark 3.0.1, scala 2.12 since version 7.2 was not available in the list. Figure 94 shows the screenshot of cluster created in Databricks community version.

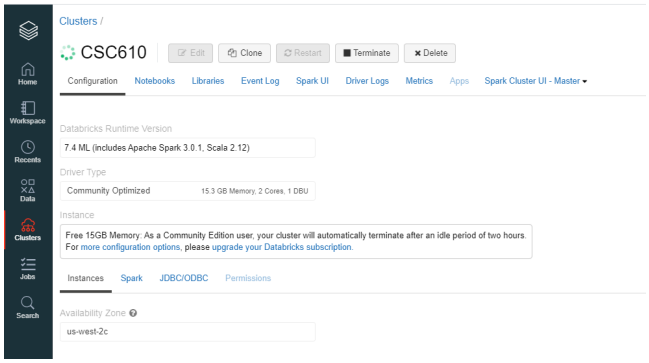


Figure 94: Cluster creation

18.4 Create Dataframes

Four data frames are created by uploading two class overlapping and non-overlapping, three class overlap and non-overlap feature vectors into big data system. Theses are the same feature vectors

used in previous assignments. Overlap01, overlap012, nonoverlap01 and nonoverlap012 tables are created. Similarly, large_dataset folder which is around 1.5 GB is also uploaded to see the performance of big data system. An end to end process for generating a table for one of the dataset is shown from figure 95 to 97. Same code and procedure is used to upload other data sets as well.

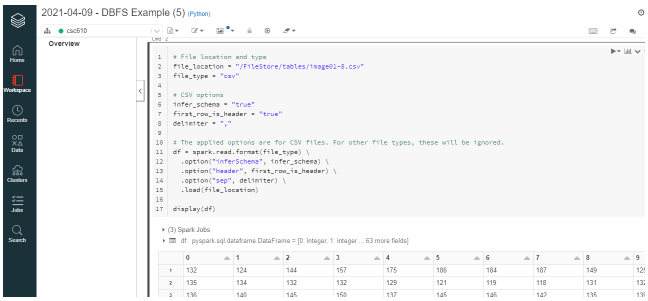


Figure 95: Dataframe creation part1

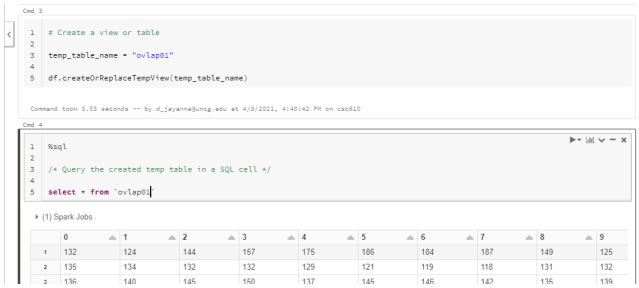


Figure 96: Dataframe creation part2

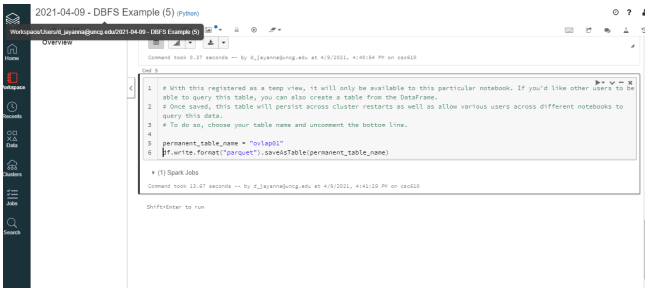


Figure 97: Dataframe creation part3

19 CONVERTING THE CODE TO RUN YOUR DATA SETS

In assignment 2, I had ran multiple combinations by changing the n_estimators values to analyse it's impact on accuracy and to find out the best model among each. The best model in each category is predicted as below.

- Non overlap two class - 50 estimators with 80% accuracy.
- Non overlap three class - 100 estimators with 82% accuracy.
- Overlap two class - 1000 estimators with 92.9% accuracy.
- Overlap three class - 500 estimators with 94.2% accuracy.

For this task, I am executing aforementioned models, in the databricks platform as well as in local environment. General steps followed to execute the model are as follows:

19.1 pyspark

Imported pyspark as shown in Figure 98 for computational purpose and fetching the data from pyspark.sql.dataframe.Dataframe uploaded earlier in Task2 using the select query.

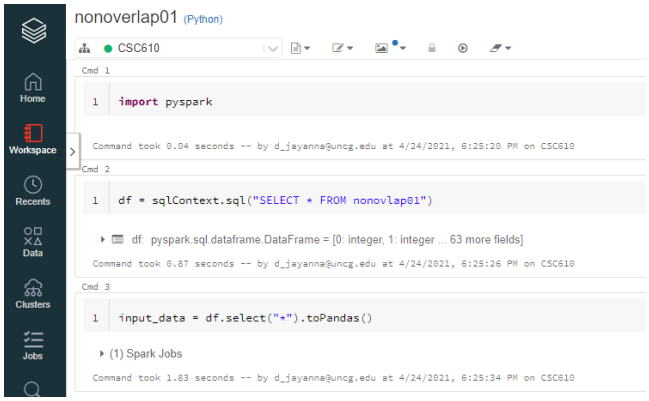


Figure 98: importing pyspark and fetching Dataframe

19.2 Generating histograms and boxplots

Figure 99 and 100 shows the histogram and boxplots respectively for Feature 30 using seaborn package.

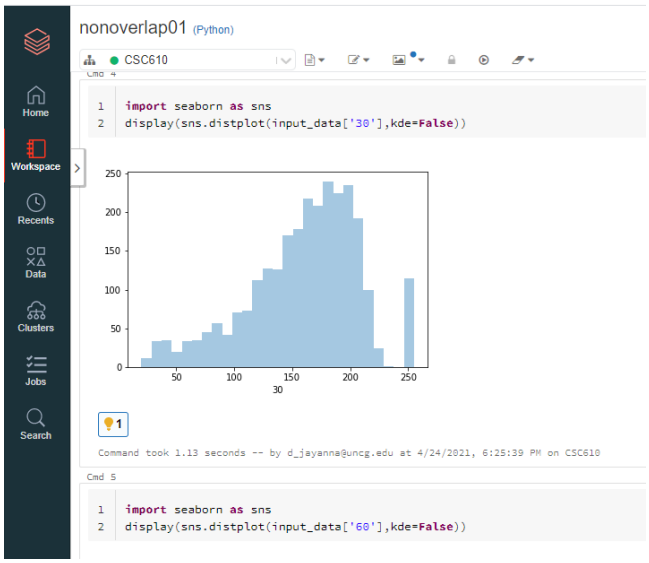


Figure 99: Histogram for Feature 30

19.3 Splitting the data

Checked the input data for any missing values before applying the model using isna() function. Then, splitting the data into 80% Train and 20% Test set into by assigning test_split parameter to 0.2. Splitting is done using train_test _split imported from sklearn.model_selection module. Figure 101 shows the execution of this task.

19.4 Training RandomForest Classifier

RandomForestClassifier is trained using 80% Train data set for 50 estimators(as this was the best predicted model in this scenario).

Figure 100: Boxplot for feature 30

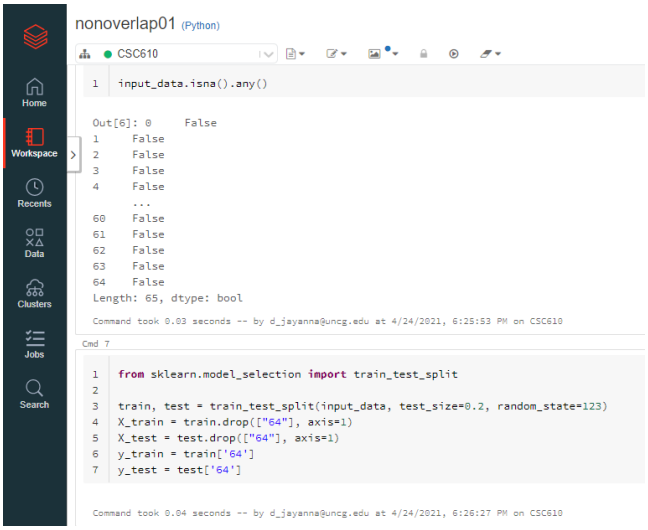


Figure 101: Splitting the data to Train and Test data

This process took around 0.64 seconds in Databricks environment for training the model. Then the model is tested for X_Test values to predict the output values. The proof of execution is as shown in Figure 102.

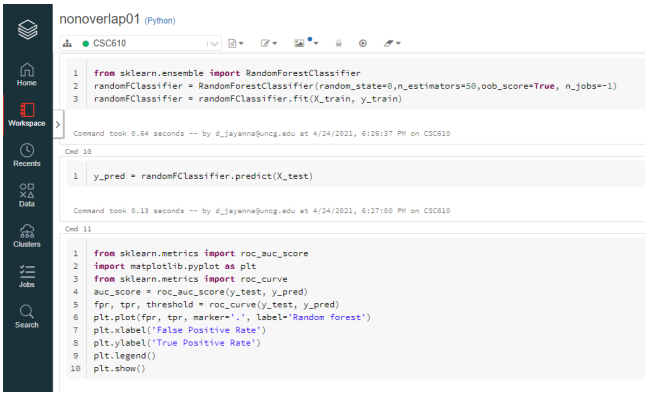


Figure 102: Random Forest classifier

19.5 Generating ROC Curve

ROC curve is drawn using roc_auc_score by passing test and predicted outputs. Also, predict_probalities with multiple threshold values among tpr along y-axis and fpr along x-axis. Figure 103 and 104 shows the generated ROC curve.

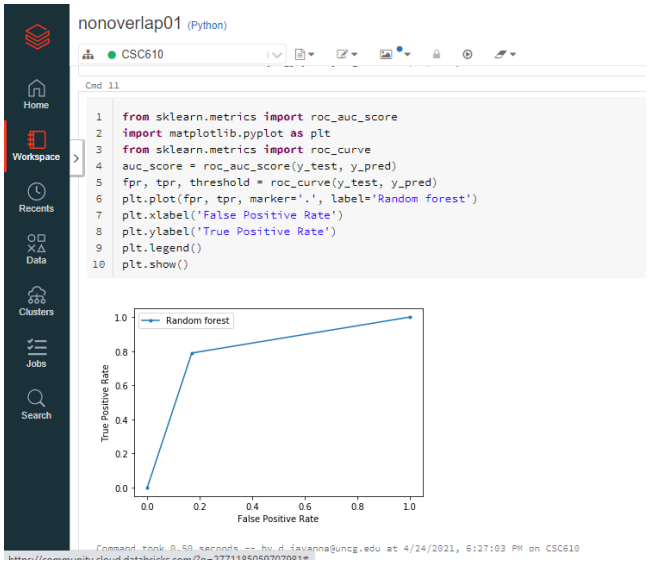


Figure 103: ROC curve

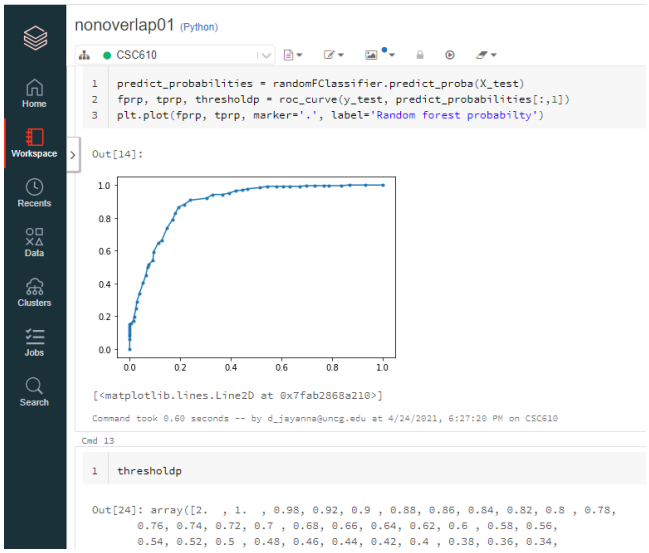


Figure 104: ROC curve with multiple threshold values

19.6 Calculating the feature importance

Importance of every feature is calculated to find the feature that is maximum impacting in classification. Looking in to figure 105, feature 63 seems to have highest importance compared to other features.

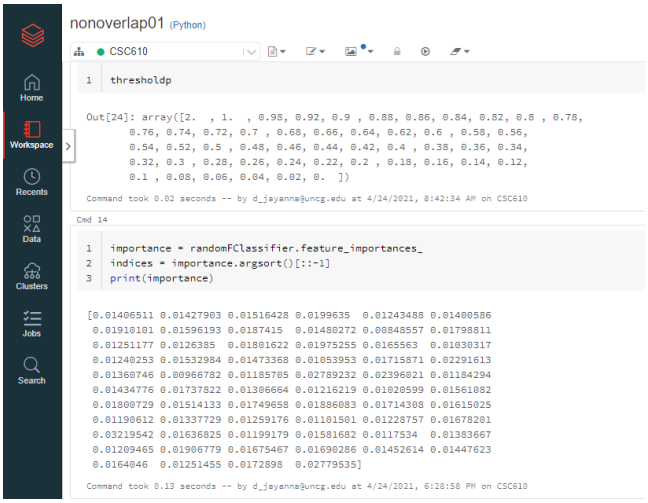


Figure 105: Feature importance evaluation

19.7 Calculating Accuracy scores, AUC scores and Confusion matrix

Accuracy of the model is calculated using sklearn.metric accuracy_score, AUC score is calculated and confusion matrix is printed. Figure 106 shows the execution of accuracy and confusion matrix. Similar model (100 estimators and merged files data set) ran in both

Figure 106: Accuracy and Confusion matrix

big data system and local system which has good results i.e where the big data system is performing well is shown from Figure 107 to Figure 116.

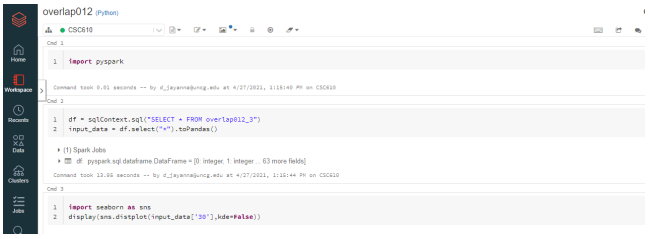


Figure 107: Reading the tables

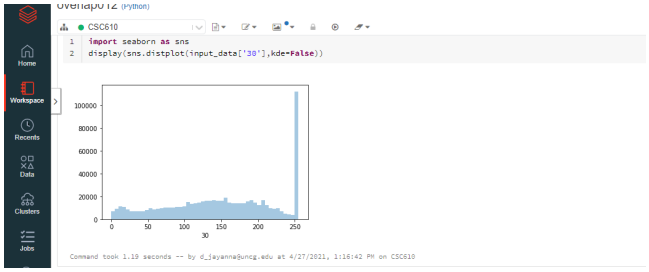


Figure 108: Histogram

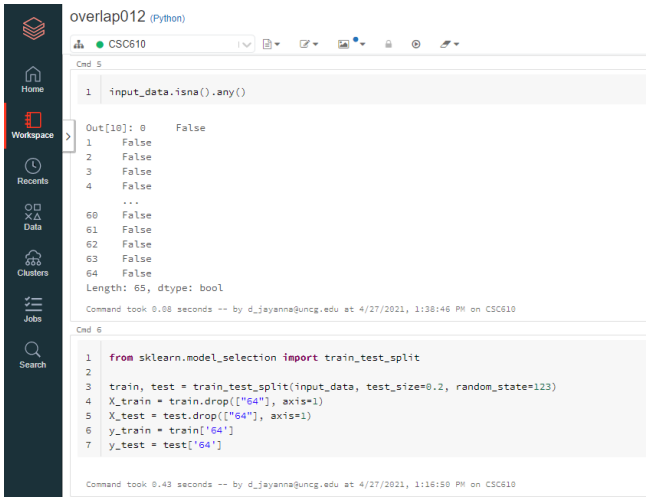


Figure 109: Splitting the data set

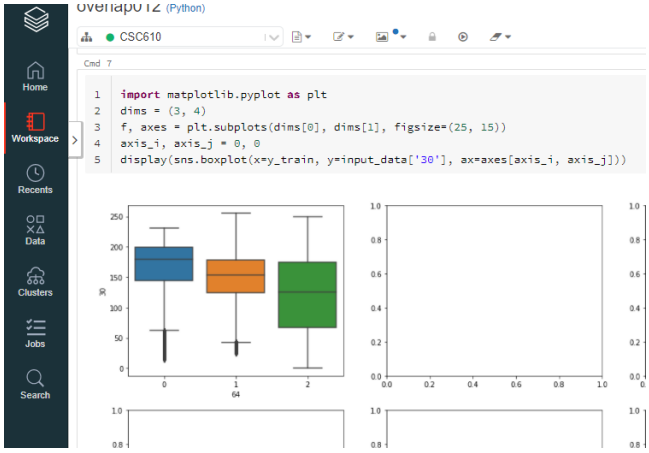


Figure 110: Box plot

20 COMPARING THE RESULTS ON A LOCAL SYSTEM VERSUS A BIG DATA SYSTEM

20.1 Compiling the results from Local system

Two class Nonoverlap data set executed the model in just 0.169 seconds. Three class Nonoverlap data set just took 0.44 seconds to execute or train the model. Two class overlap dataset took 5.4 minutes and three class overlap data set took 11.50 minutes and three class overlap took 27.26 minutes. I increased the data set by merging 9 files i.e 3 data set from all three categories to increase the overall data set size. This particular model with above said data set

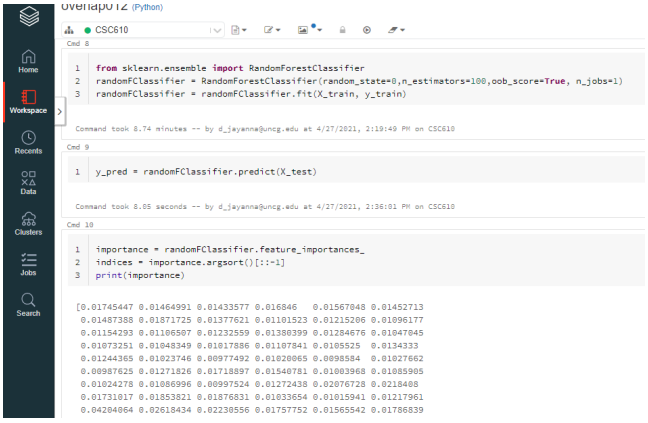


Figure 111: Training and predicting the model

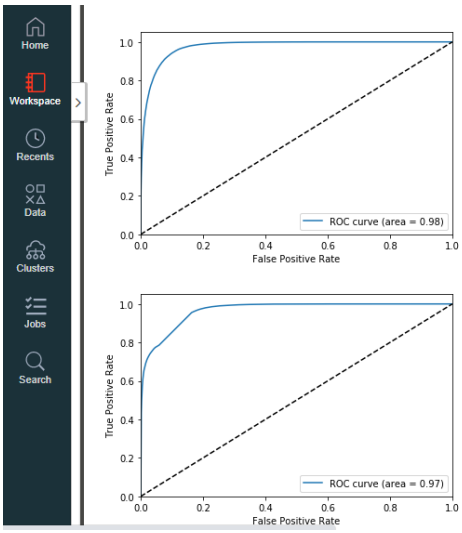


Figure 112: ROC and AUC curve class1 and class2

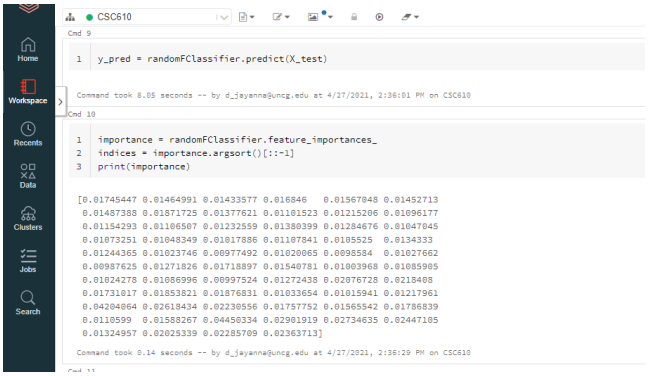


Figure 113: Feature importance

took 31.15 minutes, 20.65 minutes, 10.21 minutes for 300, 200 and 100 estimator values respectively. But, it failed to allocate memory for 500 estimator value. All the executed models screenshots are attached in the uploaded screenshot folder.

Figure 114: ROC and AUC class0

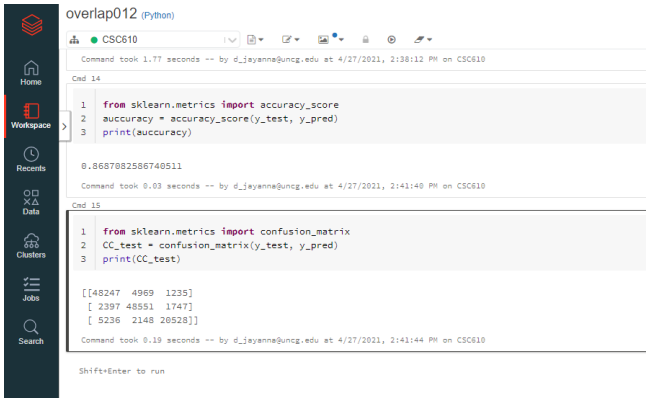


Figure 115: Accuracy and Confusion matrix

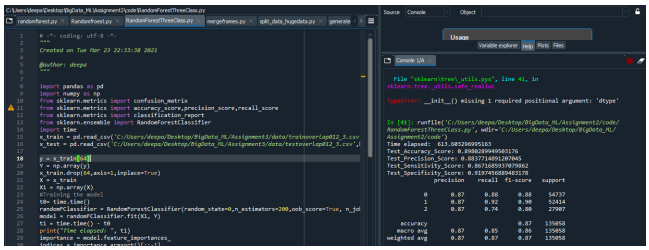


Figure 116: Local environment results

20.2 Compiling the results from Big data system

A similar experiments are conducted in big data system as local system. Two class Nonoverlap executed the model in just 0.169 seconds.Three class Nonoverlap - just took 0.44 seconds to execute or train the model.Two class overlap - took 5.4 minutes and three class overlap took 11.50 minutes. Three class overlap took 27.26 minutes. I increased the data set by merging 9 files i.e 3 data set from all three categories to increase the overall data set size. This particular model with above said data set took 31.15 minutes, 20.65 minutes, 10.21 minutes for 300, 200 and 100 estimator values respectively. But, it failed to allocate memory for 500 estimator value. All the executed models screenshots are attached in the uploaded screenshot folder.

20.3 Comparing the results

I started off with the best models I summarized in assignment 2 shown in Figure 87 and Figure 88. For Nonoverlap two class, local system took 0.169 second to complete training the model for 50 estimators. And the big data system took 0.64 seconds. For Nonoverlap three class, model was trained with 100 estimators. Local system took 0.44 seconds and bigdata system took 1.63 seconds. Overlap

two class was trained for 1000 estimators. Local system took 5.4 minutes and big data system took 12.78 minutes. Overlap three class trained for 1000 estimators. Local system took around 28 minutes and big data system took 25 minutes for execution. The table showing the summary of these results is shown in figure 117. I

	Nonoverlap01	Nonoverlap012	Overlap01	Overlap012
Estimators	50	100	1000	1000
Local env	0.169s	0.44s	5.4m	27.26m
Bigdata env	0.64s	1.63s	12.78m	25m
Accuracy	80.33	82.1	92.95	93.8

Figure 117: Comparing local and Bigdata environment

observed local system was performing better than big data system for small data set like nonoverlap two, three class and two class overlap data set. Overlap three class data set has way more observations compared to others. This set started performing better than local system. So I chose to increase this specific dataset with more observations. This got me to an understanding that big data system really needs to have large data set to perform better than local systems. I created a new merged data set of 9 files, three from each category of fruits. Then, I carried few experiments to have a better understanding on Big data environment performance. Four

Overlap three class multiple files merged data				
Estimator values	Bigdata	Local	Accuracy Local	Accuracy Bigdata
500	44.7	Memory error	NA	87.5
300	25.48	31.15	87	87
200	17.38	20.65	89.8	86.94
100	8.74	10.21	89.8	86.87
All numbers are in minutes				

Figure 118: Comparing performance of local and big data system with overlap data

experiments are conducted on the above said dataset by changing the estimator values and setting n_jobs to 1.

- Experiment 1:- 100 estimators - In this trail, both local and big data system are trained with 100 estimators. Local system ended up taking 10.21 minutes and big data system took 8.74.
- Experiment 2: 200 estimators - In this trail, both local and big data system are trained with 200 estimators. Local system ended up taking 20.65 minutes and big data system took 17.38 min.
- Experiment 3: 300 estimators - In this trail, both local and big data system are trained with 300 estimators. Local system ended up taking 31.15 minutes and big data system took 25.48 min.
- Experiment 4: 500 estimators - In this trail, both local and big data system are trained with 400 estimators. Local system failed in the middle throwing "Could not allocate memory". But, the big data environment successfully executed the model in 44.7 minutes.

A cumulative table with all the execution time and the accuracy of specific models of both big data and local system is summarized in Figure 118. In conclusion, big data environment started performing better after increasing the data set, as well as increasing the estimator values. From figure 117 and 118, we can see that bigdata environment time increased by a lesser number compared to local system while increasing the estimator values as well.