Assignment 1: Raw Data to Feature Space

Priyanka Budavi
University of North Carolina Greenboro
Email: p_budavi@uncg.edu

1 Build your programming environment!

1.1 Install Anaconda 3

Installed Anaconda 3 as the software is easy to use and also it comes with an inbuilt Python, Jupyter Notebook and Spyder packages which is required as a part of our assignment.

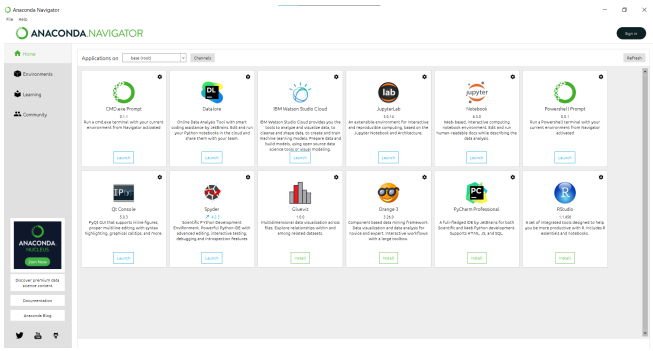


Figure 1: Installed Anaconda 3

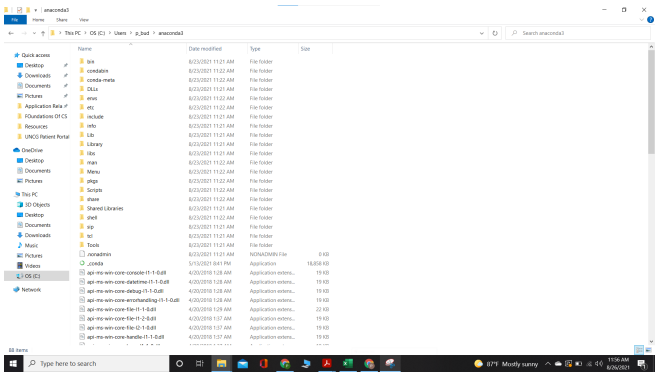


Figure 2: Workspace for the Anaconda 3

1.2 Install the Spyder IDE

The Spyder package was already present in Ananconda 3. We can access this by clicking on the launch button present on anaconda home page.

1.3 Install Jupyter Notebook

Anaconda 3 has Jupyter Notebook package and hence we can access it by clicking on windows and launching the Jupyter Notebook.

1.4 Download OpenCV

I downloaded openCV via Anaconda prompt by using the command conda **install -c conda-forge opencv**. I followed the steps from https://www.javatpoint.com/opencv-installation website.

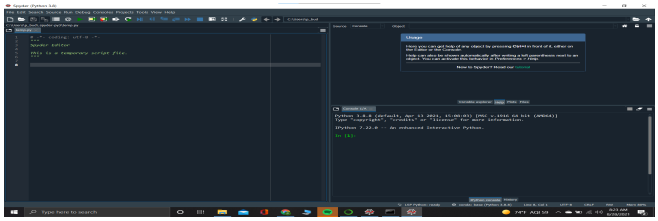


Figure 3: Installed Spyder IDE

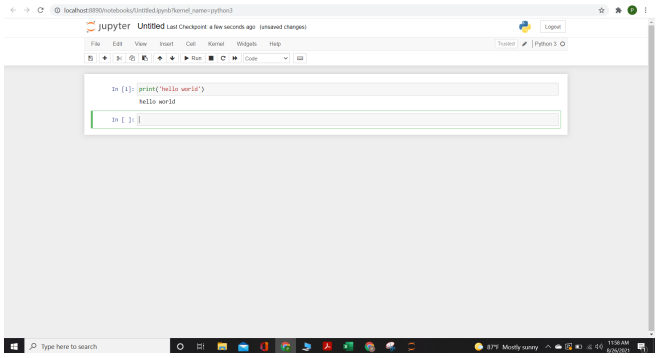


Figure 4: Sample program in Jupyter Notebook

2 Download or generate a fruits and vegetables image dataset!

2.1 Download Tropical Fruits DataSet

Downloaded the Tropical fruits from http://www.ic.unicamp.br/ rocha/pub/downloads/tropical-fruits-DB-1024x768.tar.gz. Figure 6 shows the data set stored in my workspace.

2.2 Download the FIDS30 DataSet

Downloaded the dataset from http://data.vicos.si/datasets/FIDS30/FIDS30.zip. Figure 7 shows the data set stored in my workspace.

2.3 Choose three fruits

The fruits I am choosing are Banana , Apple and Guava. I chose banana and apple just as single image to see how the data set would be generated because of the background noise. Guava has less background noise so I chose that image to compare all of them when combined together. I have assigned banana as Image0 , apple as Image1 and Guava as Image2. Figures 8-10 show the images of the fruits that I will be working on for the assignment.

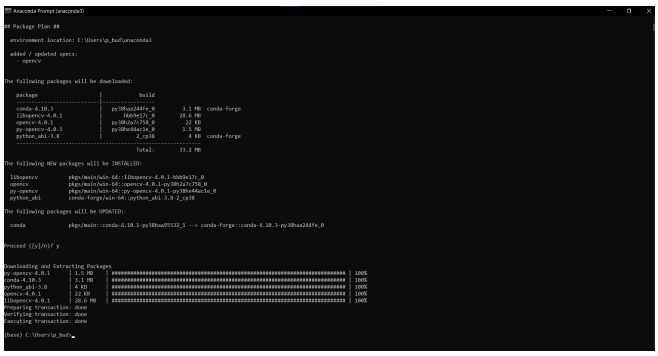


Figure 5: Installed OpenCV

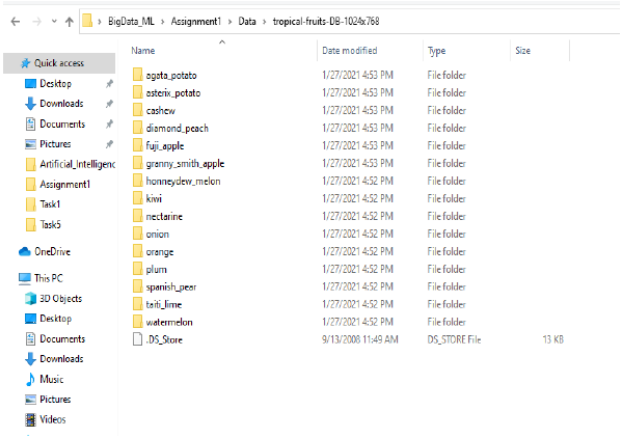


Figure 6: Tropical Fruits DataSet

3 Read your selected images and display them on the environment

3.1 Read RBG values of the selected Images

I wrote a python code using openCV libraries to read the images and display the R, G, B values for Image0 , Image1, Image2. I used CV2 library to read and get the dimensions of the images and matplotlib.pyplot to display the images. Figure 11 shows the code for the same.

3.2 Convert the color images to grayscale and display them while printing their dimensions

I wrote a python code using CV2 library. Firstly, the original images were converted to grayscale by using the function `cvtColor`.Then I derived the dimensions of the converted gray image by using the function shape. Figure 12 shows the code for all the fruits.Figure 13 - 15 are the gray images of the chosen fruits.

4 Resize the images to reduce their dimensions!

4.1 Create a function so that the dimensions are divisible by 8

I created a function which takes dimension as a parameter. The code checks if the passed argument is divisible by 8. I wrote an if condition to check if the remainder is equal to zero. If the remainder is zero that means it is divisible

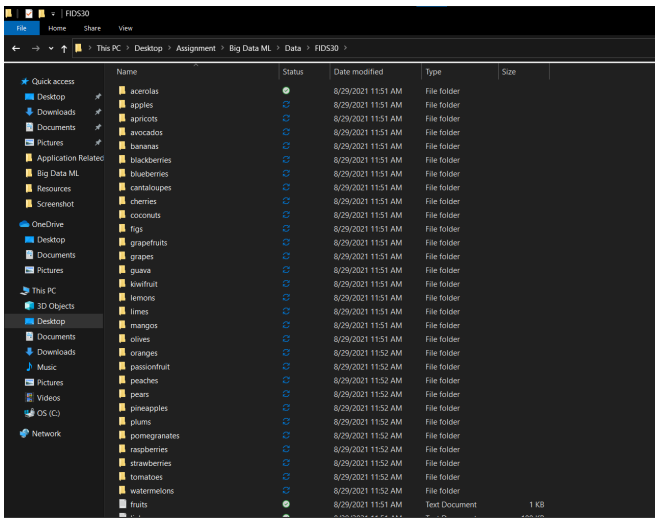


Figure 7: FIDS30 DataSet

Figure 8: Image0

by 8. If the dimension is not divisible by 8, I adjust the value such that its a multiple of 8. Check the code for logic.Figure 16 shows the python code.

4.2 Convert the image to standard height 256

Figure 17 shows the code to convert the image to a standard height of 256. I set the standard height to 256 and used a function to get the new width without changing the aspect ratio of the dimension. The new width is rounded off to the nearest value divisible by 8. I performed this task for Image0 , Image1 and Image2 and saved the new dimension images in a different file with Image0 as 256 * 256 , Image1 as 256 * 256 , Image2 as 256 * 256. Refer Figure 17 for the code.

5 Generate block-feature vectors!

5.1 Create non-overlapping feature vector of 8*8 size

I reused the code of Task 4 to convert the image to grayscale and make its width divisible by 8 with a standard height of 256. Figure 18 shows the non-overlapping code

Figure 9: Image1

Figure 10: Image2

to generate feature vectors of 8*8 size. I created a CSV file to store the pixel values and labeling each Image as 0 for Image0 , 1 for Image1 and 2 for Image2. Image 0 has 64 feature and each feature is a pixel of size 8*8. Image 0 has 1024 sub-blocks of size 8*8 and same are for Image 1 and Image 2. Figure 18 Shows the code for non-overlapping feature extraction. Figure 19-21 show the CSV files generated.

6 Generate sliding block-feature vectors!

In generating sliding block feature, an image is converted to a gray scale of height 256 and resized width. Image0 is labeled as 0 , Image1 as 1 and Image2 as 2. Each Image CSV files are generated by the code in Figure 22. Image0 has 79930 sub-locks of size 8*8 and 64 features. Image1 has 620002 sub blocks of size 8*8 and 64 features. Image2 has 62002 sublocks of size 8*8.

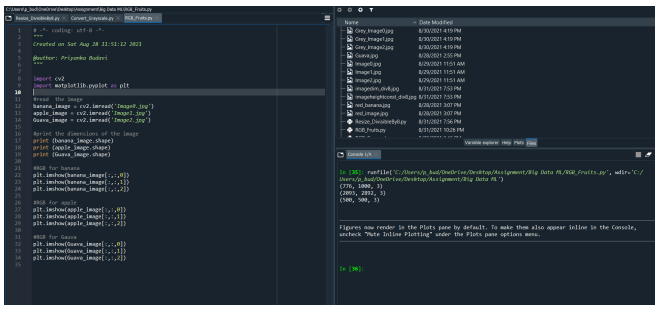


Figure 11: Read the images and display their R, G, B values

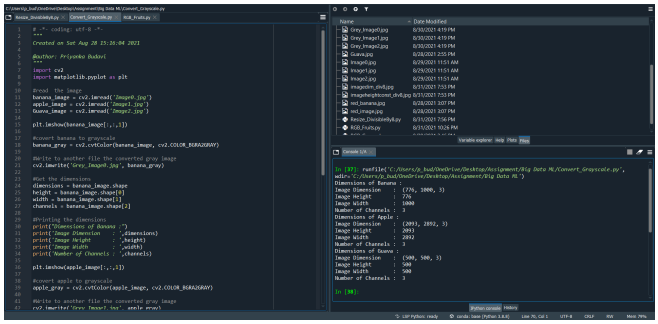


Figure 12: Grayscale image dimensions

7 Derive statistical descriptors

7.1 Number of Observations

Image0 has 1024 observation and 64 features, Image1 has 1024 observation and 64 features and Image2 has 1024 observation and 64 features where each feature in all images is of size 8 * 8. The number of observation in Image0 is more than Image1 and Image2 while observing the overlapping dataset.

7.2 Mean of the Images and the their plot

To answer the statistical questions, I took the mean of each image in the CSV file. Using the Matplotlib.pyplot I plotted the values of each image on the graph by using red color to banana , blue to apple and green to guava. Figure 23 shows the code for generating the plot. The mean value for banana falls between 217-219 , for apple 138- 139 and for guava its between 121-124. Figure 24 shows the mean plot for overlapping data.

7.3 Standard Deviation of the Images and the their plot

I took the standard deviation of the images and plotted a graph against them. Image0 ranges between 38-40 , Image1 ranges between 50-55 and Image2 between 89 -90. These are the values for non overlapping. Figure 26-28 show the required plot and code. Thus I conclude saying that there is an increasing trend between the data sets. Also the co variance between the features is positive that's why is trending up.

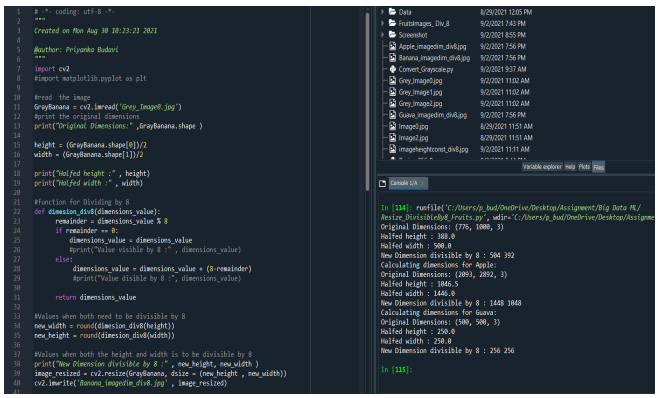
7.4 Scatter Plot of the Images

I plotted scatter plot for Image0 by assigning red as the color of the data sets. I selected Feature 20 and Feature

Figure 13: Image 0

Figure 14: Image 1

Figure 15: Image 2



```
# -*- coding: utf-8 -*-
"""
Created on Mon Aug 30 10:23:21 2021

@author: Priyanka Badavi
"""
import cv2
#import matplotlib.pyplot as plt

#read  the image
GrayBanana = cv2.imread('Grey_Image0.jpg')
#print the original dimensions
print("Original Dimensions:" ,GrayBanana.shape )

height = (GrayBanana.shape[0])/2
width = (GrayBanana.shape[1])/2

print("Halfed height :" , height)
print("Halfed width :" , width)

#Function for Dividing by 8
def dimension_div8(dimensions_value):
    remainder = dimensions_value % 8
    if remainder == 0:
        dimensions_value = dimensions_value
        #print("Value visible by 8 :" , dimensions_value)
    else:
        dimensions_value = dimensions_value + (8-remainder)
        #print("Value disible by 8 :", dimensions_value)

    return dimensions_value

#Values when both need to be divisible by 8
new_width = round(dimension_div8(height))
new_height = round(dimension_div8(width))

#Values when both the height and width is to be divisible by 8
print("New Dimension divisible by 8 :" , new_height, new_width )
image_resized = cv2.resize(GrayBanana, dsize = (new_height , new_width))
cv2.imwrite('Banana_imagedim_div8.jpg' , image_resized)
```

```
Data                                  8/29/2021 12:05 PM
FruitsImages_Div_8                    9/2/2021 7:43 PM
Screenshot                            9/2/2021 8:55 PM
Apple_imagedim_div8.jpg               9/2/2021 7:56 PM
Banana_imagedim_div8.jpg              9/2/2021 7:56 PM
Convert_Grayscale.py                  9/2/2021 9:37 AM
Grey_Image0.jpg                       9/2/2021 11:02 AM
Grey_Image1.jpg                       9/2/2021 11:02 AM
Grey_Image2.jpg                       9/2/2021 11:02 AM
Guava_imagedim_div8.jpg               9/2/2021 7:56 PM
Image0.jpg                            8/29/2021 11:51 AM
Image2.jpg                            8/29/2021 11:51 AM
imageheightconst_div8.jpg             9/2/2021 11:11 AM

In [114]: runfile('C:/Users/p_bud/OneDrive/Desktop/Assignment/Big Data ML/
Resize_DivisibleBy8_Fruits.py', wdir='C:/Users/p_bud/OneDrive/Desktop/Assignme
Original Dimensions: (770, 1000, 3)
Halfed height : 388.0
Halfed width : 500.0
New Dimension divisible by 8 : 504 392
Calculating dimensions for Apple:
Original Dimensions: (2893, 2892, 3)
Halfed height : 1040.5
Halfed width : 1446.0
New Dimension divisible by 8 : 1448 1048
Calculating dimensions for Guava:
Original Dimensions: (500, 500, 3)
Halfed height : 250.0
Halfed width : 250.0
New Dimension divisible by 8 : 256 256

In [115]:
```

Figure 16: Code for dimensions divisible by 8

40 of Image0 to plot the graph. For Image1 I chose green color and selected Feature 10 and Feature 40 from the Image1.csv file. Similarly for Image2 I chose blue and selected Feature 15 and Feature 40 from the Image2.csv file. Figure 29 - 35 show the overlapping and non-overlapping graph of the images.

7.5 Histogram of the images

I took histogram plot for Image0 , Image1 , Image2. For Image0 all the features are present towards the right end and rest other features are skewed for both overlapping and non overlapping. In Image1 the feature are distributed everywhere but its skewed in the middle for both overlapping and non overlapping. Image2 the features are distributed more towards the centre than at the edges.

7.6 Is the dataset imbalanced, inaccurate or incomplete ?

The no of observations in Image0 , Image1 and Image2 for non-overlapping data set are the same. Thus the data set is balanced. But for overlapping data set , the number of observations in Image0 is more than Image1 and Image2 so data is imbalanced . There was no missing data in the CSV file and so it wasn't incomplete. The labels in the csv file were generated correctly hence the data is accurate.

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

It is trivial data. The total images that we are considering for analysis is just three. So we have only 3 labels or classes. The number of observation generated from the images is more than the number of features so it is a low dimensional data set. For the data to be a big data it needs to exponentially increase time

7.8 Does it have scalability problem? Are they high dimensional?

It is a low dimensional data.Each feature is of size 8*8. So if the features increase then the scalability also increases. But I don't see high dimensionality and scalabilty problem with the data set generated from three fruit images.

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

Yes it need to be standardized and normalized. For example, A variable that ranges between 0 and 500 will out-

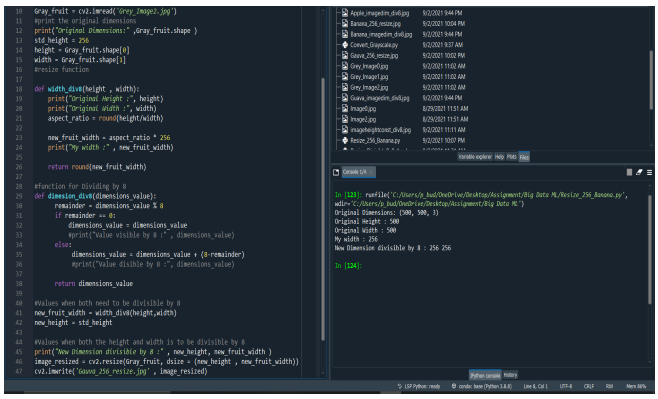


Figure 17: Code to set standard height to 256

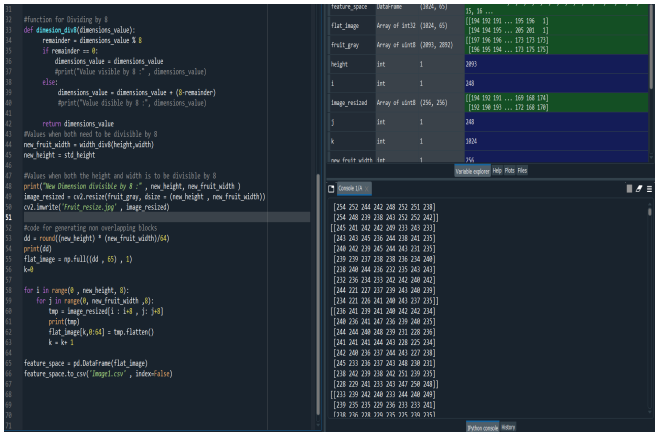


Figure 18: Non-Overlapping feature block

Figure 19: Non-Overlapping Banana with label 0

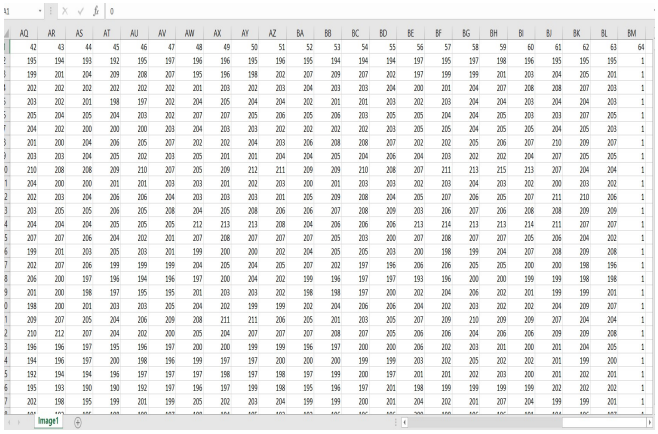


Figure 20: Non-Overlapping apple with label 1

weigh a variable that ranges between 0 and 1. Using these variables without standardization/normalization will give the variable with the larger range weight of 500 in the analysis. Thus its necessary the variables fall in the same normal distribution curve to receive good accuracy.

7.10 How do they affect the data characteristics?

The data characteristics which is volume , velocity and variety seems to have no effect on the dataset.

8 Construct a feature space!

I merged the Image0 and Image1 to produce a concatenated file i.e Image01.csv. Figure 42 shows the image of the code which merges and uses random function to distribute the data in the excel randomly. The data is randomized row wise to get jumble the data. Similary the Image012.csv was obtained by cancatenating Image01.csv with Image2. Did the same for overlapping dataset.Figure 43-44 show the merged data set.

9 Display sub spaces

9.1 Plot two dimension feature space

I assigned Image0 , Image1 and Image2 with red , green and blue color to plot the scatter. Then I chose two features from each image to obtain a 2D feature space. I chose feature20 and feature 40 from Image0 and Image1 ,feature 10 and feature 40 of Image0 and Image2 and feature 25 30 from Image1 and Image2. I followed the same steps to obtain the Overlapping feature space scatter plot. Figure 45 - 51 show the images of the 2D feature space of all Images and the respective code.

9.2 Plot three dimensional feature space

I selected feature 10 , feature 20 , feature 40 from Image0 , Image1 and Image2 to plot a three dimensional scatter plot. Figure 52-53 shows the plot and code for the same. The points are overlapping on each other and thus make it difficult to classify them. Thus it is inseparable classes.

9.3 Discuss the observation from the figures

We observe that the classes are overlapping on each other this makes the class not separable easily. Even though towards the left we see two classes which looks easy to distinguish but that alone will not help. So the complexity of classifying them is high.

10 Read multiple files from a folder

I reused the code from the task 6 and wrote a function which will read all the images and create their respective CSV files. Figure 54 -55 show the code and the CSV files generated by reading all of them.

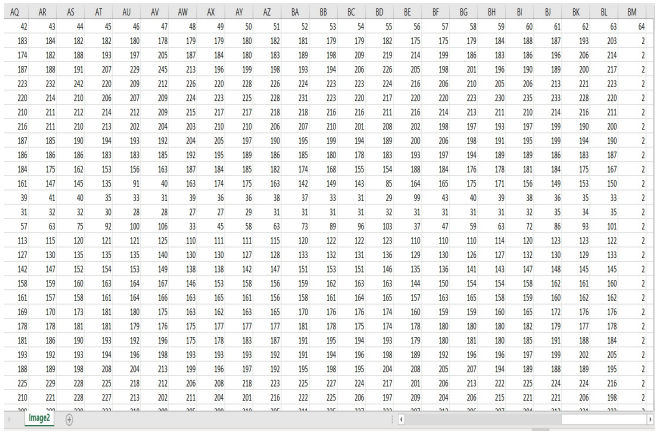


Figure 21: Non-Overlapping Guava with label 2

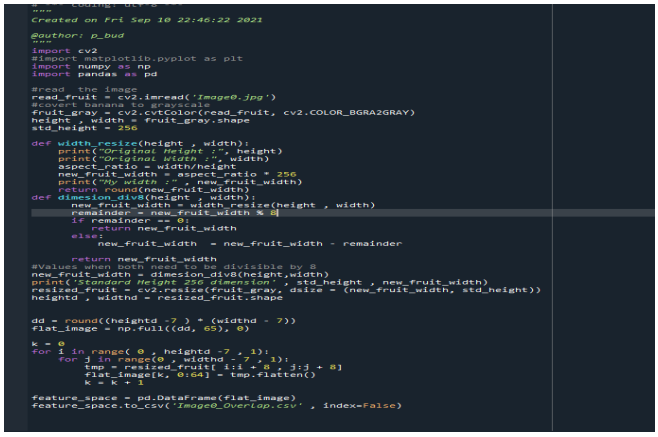


Figure 22: Overlapping feature block

11 Describe the effects of block size on the dimensionality of the feature space and the number of vectors in the domain

The images that we generated are of size 8*8 each pixel. Thus the feature vector is of size 64. Suppose we decrease the size by 4*4 then the number of observations will increase and if we increase by 16 * 16 the features will increase and this leads to the reduction in the number of observation. Hence I conclude that this would lead to high dimension or low dimension issue which doesn't give accurate results.

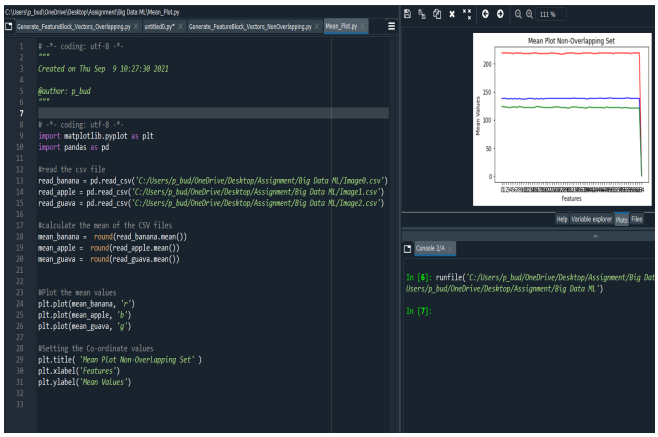


Figure 23: Mean Plot

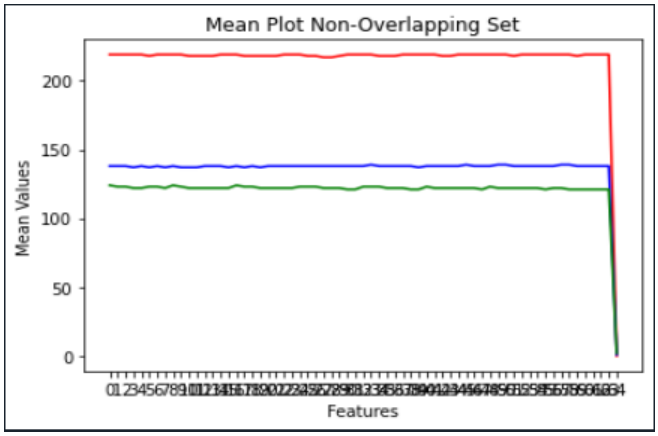


Figure 24: Mean Plot Non-Overlapping

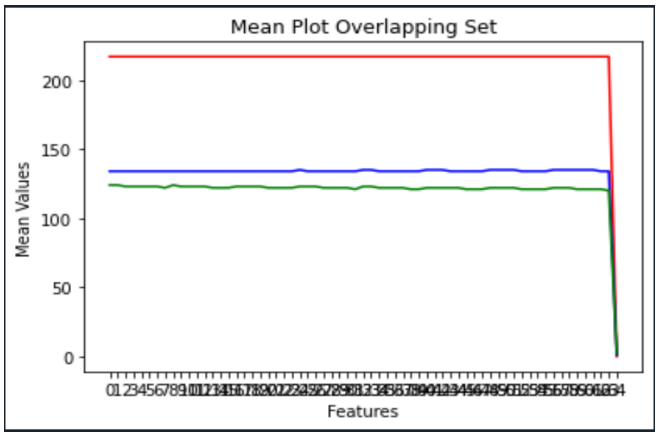


Figure 25: Mean Plot Overlapping

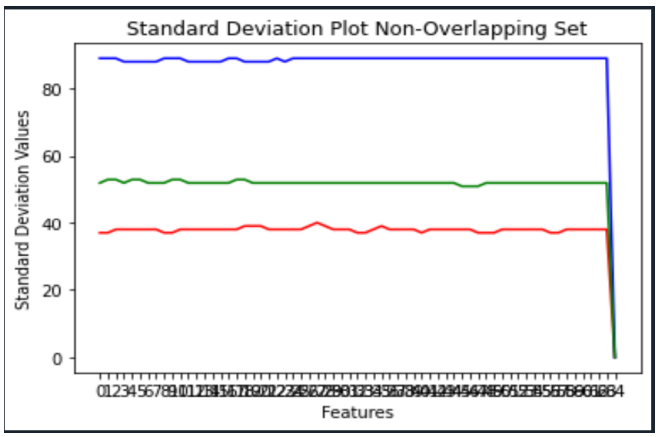


Figure 26: Standard Deviation Plot Non Overlapping

```
# -*- coding: utf-8 -*-
"""
Created on Mon Sep 13 12:00:09 2021

@author: p_bud
"""

import matplotlib.pyplot as plt
import pandas as pd

#read the csv file
read_fruit = pd.read_csv('Image0.csv')

f20 = read_fruit['20']
f40 = read_fruit['40']


plt.scatter(f20, f40, color ='red', s=1)
plt.show()
```

Figure 29: Scatter Plot Code

```
# -*- coding: utf-8 -*-
import matplotlib.pyplot as plt
import pandas as pd

#read the csv file
read_banana = pd.read_csv('C:/Users/p_bud/OneDrive/Desktop/Assignment/Big Data ML/Image0.csv')
read_apple = pd.read_csv('C:/Users/p_bud/OneDrive/Desktop/Assignment/Big Data ML/Image1.csv')
read_guava = pd.read_csv('C:/Users/p_bud/OneDrive/Desktop/Assignment/Big Data ML/Image2.csv')

#calculate the mean of the CSV files
std_banana =  round(read_banana.std())
std_apple =  round(read_apple.std())
std_guava =  round(read_guava.std())

print(" Banana ", std_banana)
print(" Apple ", std_apple)
print(" Guava ", std_guava)

#Plot the mean values
plt.plot(std_banana, 'r')
plt.plot(std_apple, 'b')
plt.plot(std_guava, 'g')

#Setting the Co-ordinate values
plt.title( 'Standard Deviation Plot Non-Overlapping Set' )
plt.xlabel('Features')
plt.ylabel('Standard Deviation Values')
```

Figure 27: Standard Deviation Non Overlapping Code

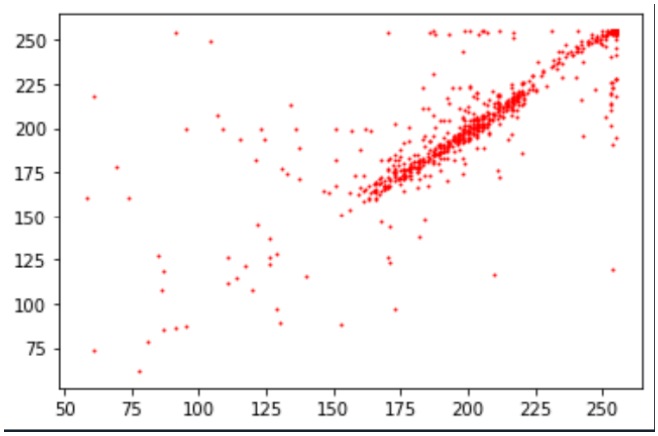


Figure 30: Non-Overlapping Scatter Plot Banana Feature 20 and Feature 40

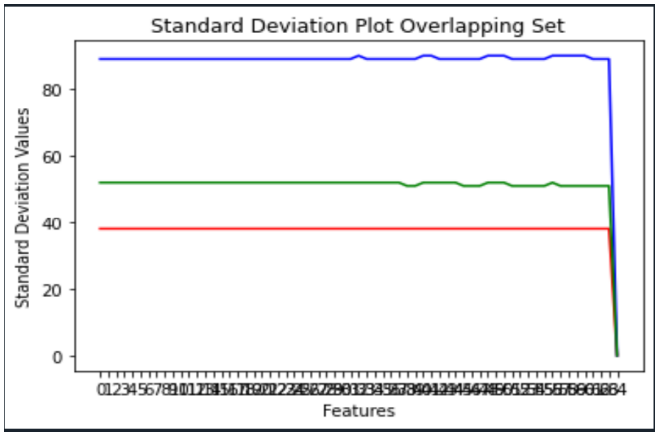


Figure 28: Standard Deviation Overlapping Plot

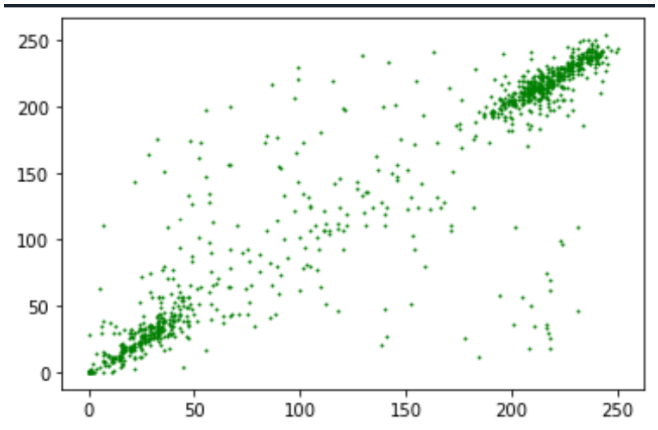


Figure 31: Non-Overlapping Scatter Plot apple Feature 10 and Feature 40

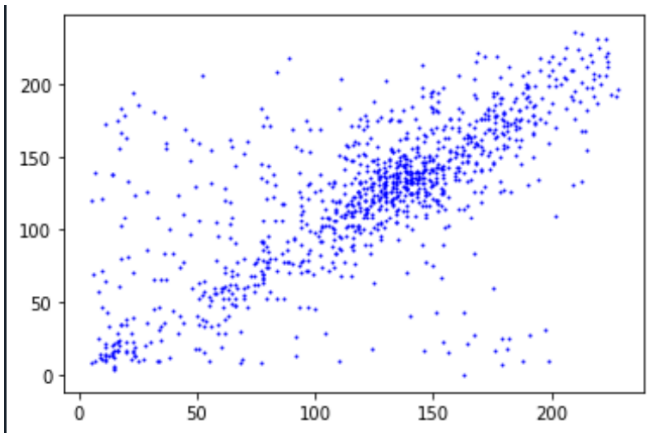


Figure 32: Non -Overlapping Scatter Plot Guava Feature 15 and Feature 40

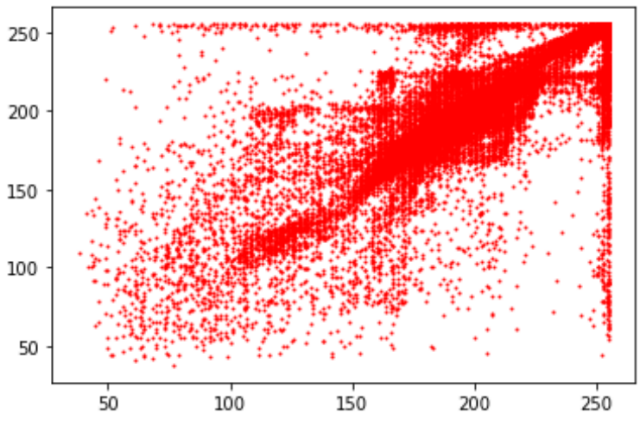


Figure 33: Overlapping Scatter Plot Banana Feature 20 and Feature 40

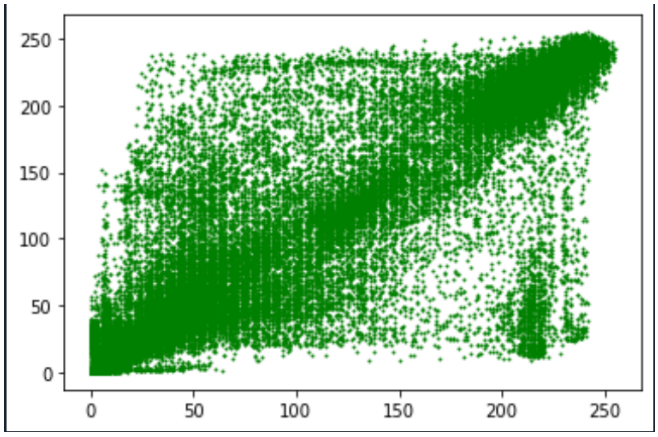


Figure 34: Overlapping Scatter Plot apple Feature 10 and Feature 40

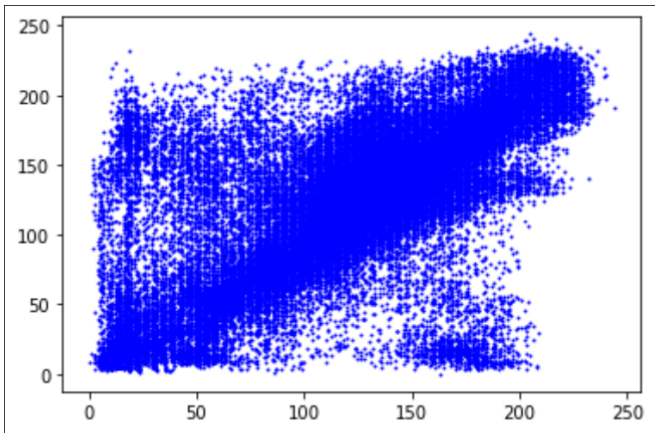


Figure 35: Overlapping Scatter Plot Guava Feature 15 and Feature 40

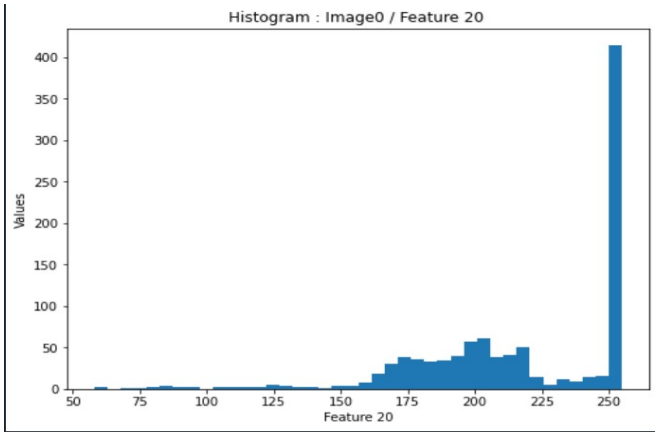


Figure 36: Histogram of Feature 20 for Image0

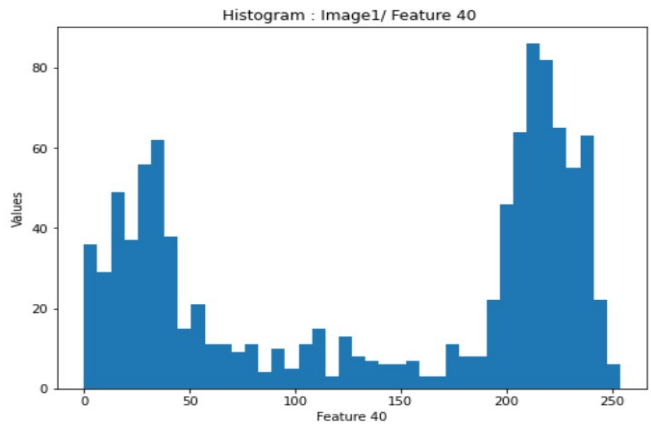


Figure 37: Histogram of feature 40 for Image1

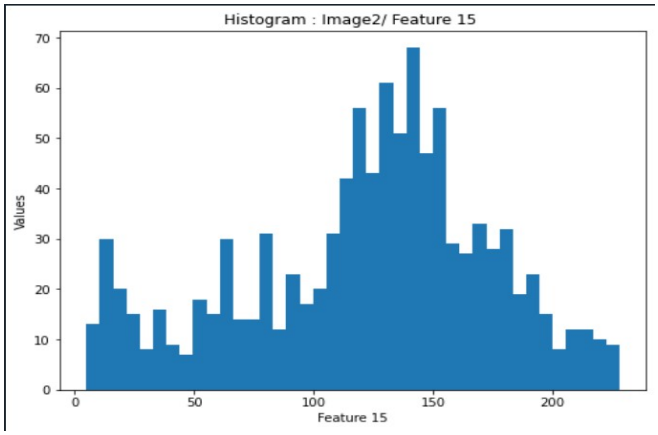


Figure 38: Histogram of Feature 15 for Image2

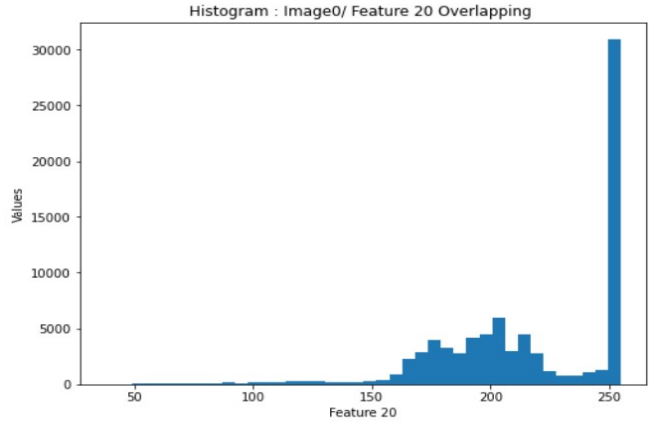


Figure 39: Histogram of feature 20 for Image0: Overlap

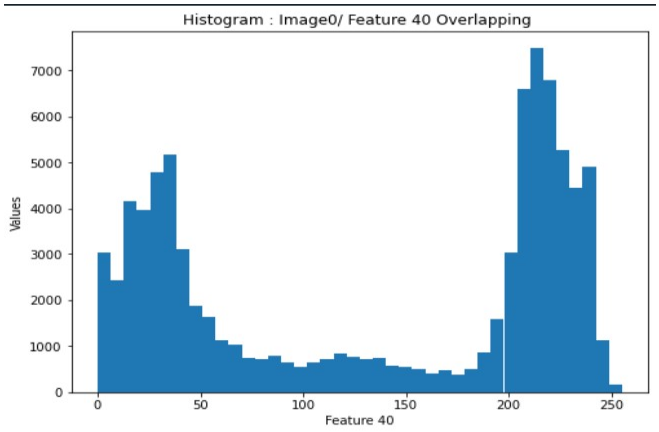


Figure 40: Histogram of feature 40 for Image1: Overlap

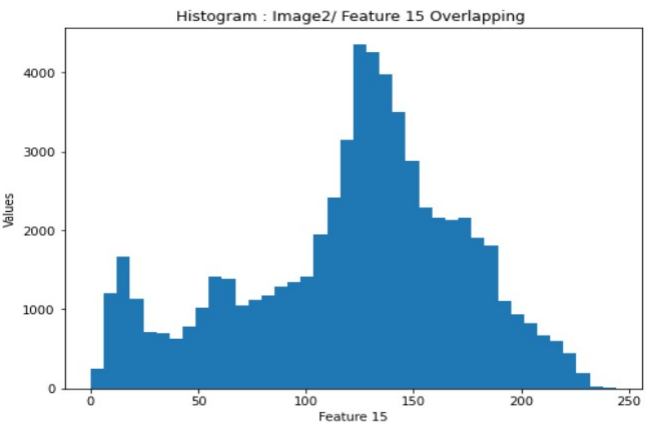


Figure 41: Histogram of feature 15 for Image2: Overlap

```
# -*- coding: utf-8 -*-
"""
Created on Sat Sep 11 18:32:34 2021

@author: p_bud
"""

import pandas as pd
import numpy as np

Image0_read = pd.read_csv('Image01_Overlap.csv')
Image1_read= pd.read_csv('Image2_Overlap.csv')


frames = [Image0_read , Image1_read]
merged = pd.concat(frames)

indx = np.arange(len(merged))
rndmged = np.random.permutation(indx)
rndmged= merged.sample(frac=1).reset_index(drop=True)
rndmged.to_csv('C:/Users/p_bud/OneDrive/Desktop/Assignment/Big Data ML/Image012_Overlap.csv', index=False)

```

Figure 42: Merge and Randomize

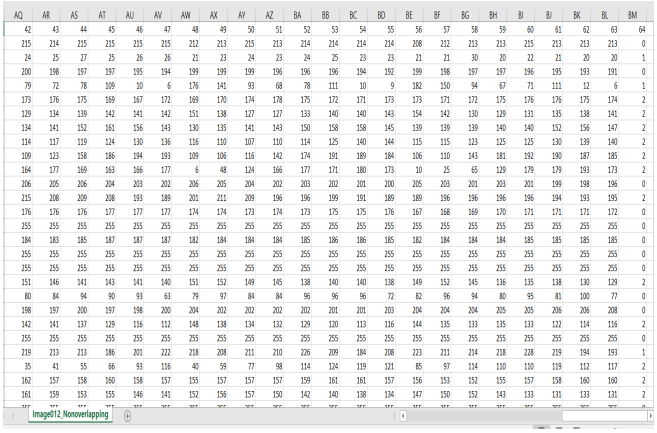


Figure 43: Feature Space of Non-Overlapping DataSet

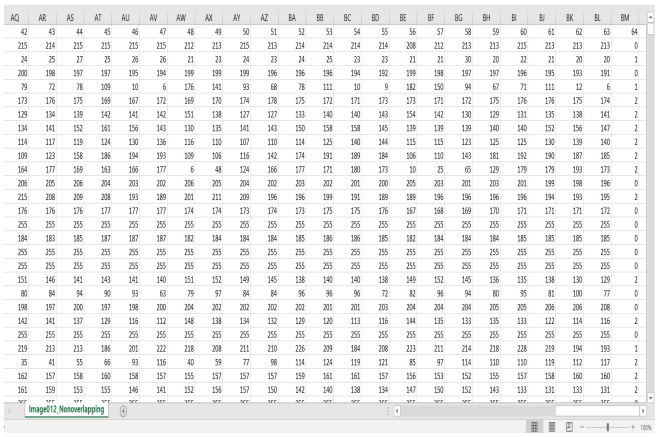


Figure 44: Feature Space of Overlapping Data Set

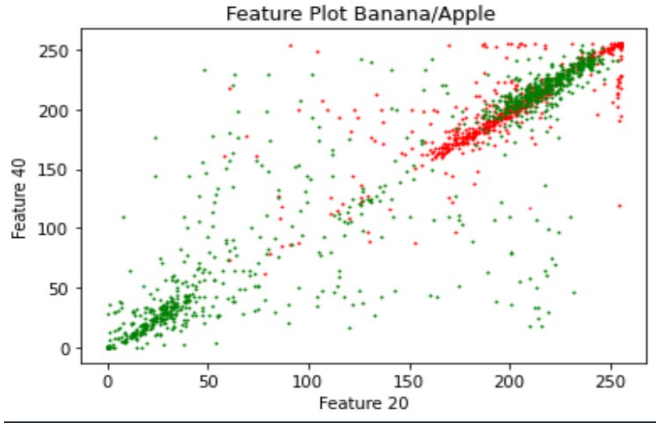


Figure 45: Feature Space of Image0 and Image1

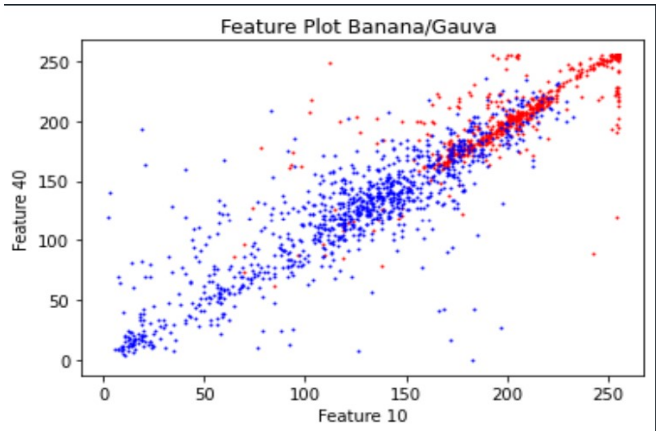


Figure 46: Feature Space of Image0 and Image2

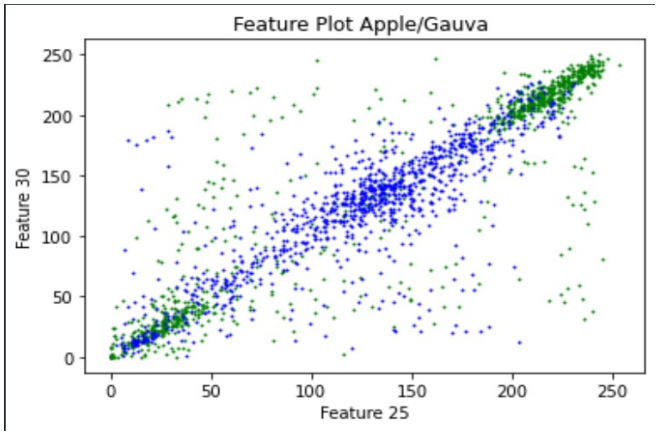


Figure 47: Feature Space of Image1 and Image2

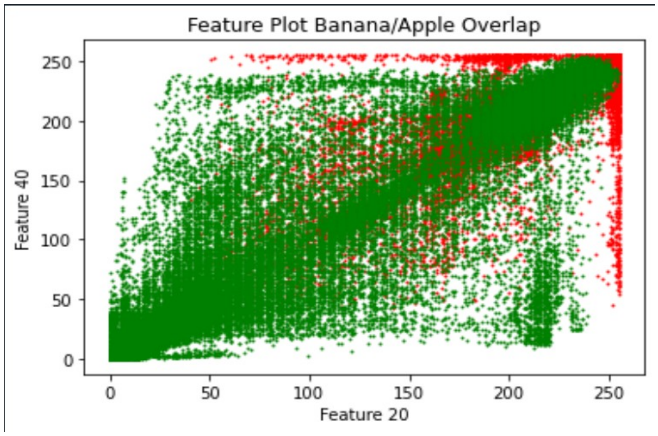


Figure 48: Overlapping Feature Space of Image0 and Image1

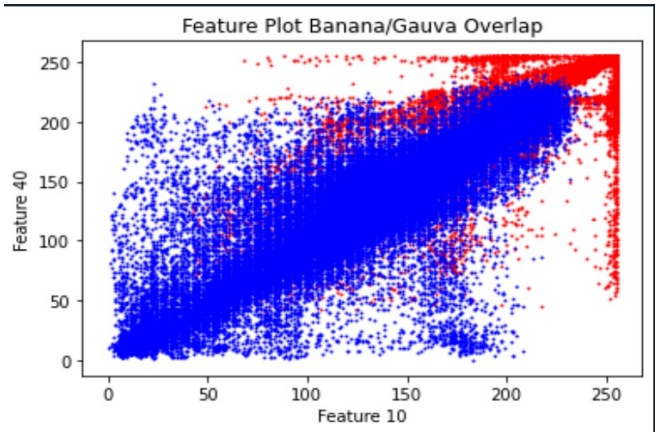


Figure 49: Overlapping Feature Space of Image0 and Image2

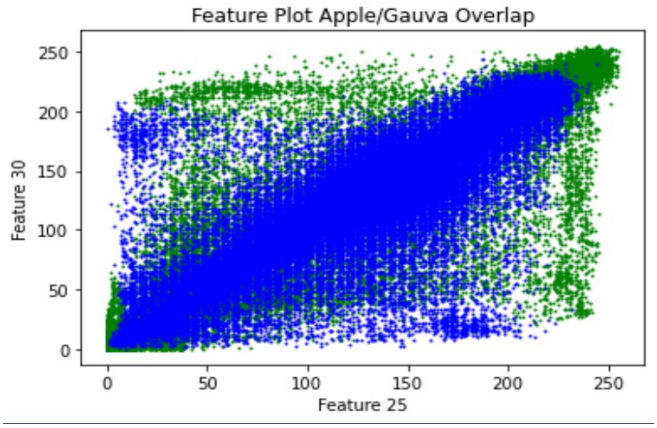


Figure 50: Overlapping Feature Space of Image1 and Image2

```
# -*- coding: utf-8 -*-
"""
Created on Tue Sep 14 10:24:31 2021

@author: p_bud
"""


import matplotlib.pyplot as plt
import pandas as pd


#read the csv file
read_banana = pd.read_csv('Image0.csv')
read_apple = pd.read_csv('Image1.csv')
read_gauava = pd.read_csv('Image2.csv')


f20 = read_banana['20']
f40 = read_banana['40']

G20 = read_apple['20']
G40 = read_apple['40']

plt.scatter(f20, f40, color =['red'] , s=1 )
plt.scatter(G20, G40, color =['green'] , s=1 )

plt.title("Feature Plot Banana/Apple ")
plt.xlabel('Feature 20')
plt.ylabel('Feature 40')
plt.show()
```

Figure 51: Code for 2D scatter plot

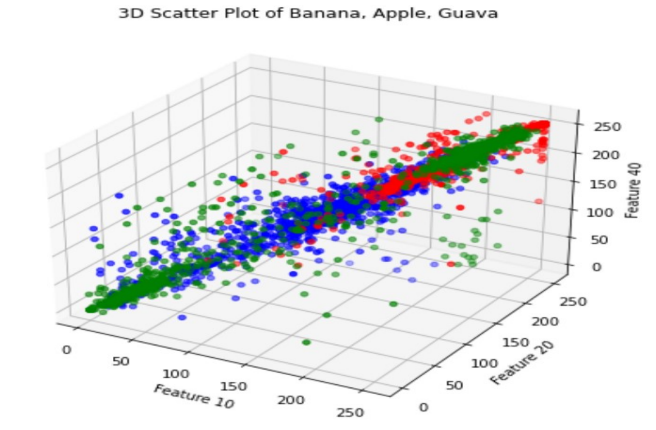


Figure 52: Three Dimensional feature space

```
"""
Created on Wed Sep 15 14:22:14 2021

@author: p_bud
"""


import matplotlib.pyplot as plt
import pandas as pd

read_banana = pd.read_csv('C:/Users/p_bud/OneDrive/Desktop/Assignment/Big Data ML/CSV files/Image0.csv')
read_apple = pd.read_csv('C:/Users/p_bud/OneDrive/Desktop/Assignment/Big Data ML/CSV files/Image1.csv')
read_gauava = pd.read_csv('C:/Users/p_bud/OneDrive/Desktop/Assignment/Big Data ML/CSV files/Image2.csv')

f10 = read_banana['10']
f20 = read_banana['20']
f40 = read_banana['40']

A10 = read_apple['10']
A20 = read_apple['20']
A40 = read_apple['40']

G10 = read_gauava['10']
G20 = read_gauava['20']
G40 = read_gauava['40']

# Creating figure
fig = plt.figure(figsize = (40, 7))
ax = plt.axes(projection ="3d")

# Creating plot
ax.scatter3D(f10, f20, f40, color = "red")
ax.scatter3D(A10, A20, A40, color = "green")
ax.scatter3D(G10,G20, G40, color = "blue")

plt.title("3D Scatter Plot of Banana, Apple, Guava")
plt.xlabel('Feature 10')
plt.ylabel('Feature 20')
ax.set_zlabel('Feature 40')
plt.show()
```

Figure 53: Code for 3D scatter plot

```
def read_multiple_files( image , filename):
    fruit_gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    height , width = fruit_gray.shape
    print("Dimension of Gray image: ", height , width)
    new_fruit_width = width_div8(height,width)

    #Values when both the height and width is to be divisible by 8
    print("New Dimension divisible by 8 :" , std_height, new_fruit_width )
    image_resized = cv2.resize(fruit_gray, dsize = (std_height , new_fruit_width))
    heightd , widthd = image_resized.shape
    dd = round((heightd) * (widthd)/64)
    print(dd)
    flat_image = np.full((dd , 65) , 1)
    k=0
    for i in range(0 , heightd, 8):
        for j in range(0, widthd ,8):
            tmp = image_resized[i : i+8 , j: j+8]
            print(tmp)
            flat_image[k,0:64] = tmp.flatten()
            k = k+ 1

    feature_space = pd.DataFrame(flat_image)
    feature_space.to_csv(filename, index=False)

    return feature_space


folder_path = 'C:/Users/p_bud/OneDrive/Desktop/Assignment/Big Data ML/Priyanka_Budavi_Submission_ML/Data/FIDS30/apples'
images = []
for imagename in os.listdir(folder_path):
 img = cv2.imread(os.path.join(folder_path,imagename))
 if img is not None:
    images.append(img)
    splitted_filename = imagename.split('.')
    rename_csvfile = 'AppleDataset'+splitted_filename[0]+'.csv'
    read_multiple_files(img,rename_csvfile)
```

Figure 54: Code for reading multiple files

Name	Status	Date modified	Type	Size
AppleDataset0	✓	9/18/2021 5:17 PM	Microsoft Excel Co...	135 KB
AppleDataset1	✓	9/18/2021 5:17 PM	Microsoft Excel Co...	136 KB
AppleDataset2	✓	9/18/2021 5:17 PM	Microsoft Excel Co...	136 KB
AppleDataset3	✓	9/18/2021 5:17 PM	Microsoft Excel Co...	135 KB
AppleDataset4	✓	9/18/2021 5:17 PM	Microsoft Excel Co...	136 KB
AppleDataset7	✓	9/18/2021 5:17 PM	Microsoft Excel Co...	136 KB
AppleDataset8	✓	9/18/2021 5:17 PM	Microsoft Excel Co...	136 KB
AppleDataset9	✓	9/18/2021 5:17 PM	Microsoft Excel Co...	136 KB
AppleDataset11	✓	9/18/2021 5:17 PM	Microsoft Excel Co...	136 KB
AppleDataset12	✓	9/18/2021 5:17 PM	Microsoft Excel Co...	135 KB
AppleDataset13	✓	9/18/2021 5:17 PM	Microsoft Excel Co...	135 KB
AppleDataset14	✓	9/18/2021 5:17 PM	Microsoft Excel Co...	136 KB
AppleDataset15	✓	9/18/2021 5:17 PM	Microsoft Excel Co...	136 KB
AppleDataset16	✓	9/18/2021 5:17 PM	Microsoft Excel Co...	136 KB
AppleDataset17	✓	9/18/2021 5:17 PM	Microsoft Excel Co...	136 KB
AppleDataset19	✓	9/18/2021 5:17 PM	Microsoft Excel Co...	136 KB
AppleDataset20	✓	9/18/2021 5:17 PM	Microsoft Excel Co...	136 KB
AppleDataset22	✓	9/18/2021 5:17 PM	Microsoft Excel Co...	136 KB
AppleDataset23	✓	9/18/2021 5:17 PM	Microsoft Excel Co...	271 KB
AppleDataset24	✓	9/18/2021 5:17 PM	Microsoft Excel Co...	136 KB
AppleDataset25	✓	9/18/2021 5:17 PM	Microsoft Excel Co...	134 KB
AppleDataset26	✓	9/18/2021 5:17 PM	Microsoft Excel Co...	135 KB
AppleDataset28	✓	9/18/2021 5:17 PM	Microsoft Excel Co...	134 KB
AppleDataset32	✓	9/18/2021 5:17 PM	Microsoft Excel Co...	136 KB
AppleDataset34	✓	9/18/2021 5:17 PM	Microsoft Excel Co...	136 KB
AppleDataset35	✓	9/18/2021 5:17 PM	Microsoft Excel Co...	271 KB
AppleDataset38	✓	9/18/2021 5:17 PM	Microsoft Excel Co...	271 KB

Figure 55: Multiple CSV files are created

12 ASSIGNMENT 2 : FEATURE SPACE TO A CLASSIFIER

12.1 Split the data into 80:20 and save the file

I used the dataset created in the previous assignment and split them into 80:20 ratio using the code and later saved the files into train dataset and test data set with respective file names. Figure 56 show the code for splitting the data into training and testing dataset.

```
# -*- coding: utf-8 -*-
"""
Created on Wed Oct 20 14:42:25 2021

@author: p_bud
"""

import pandas as pd
import numpy as np

# Read a feature space
input_data = pd.read_csv("C:/Users/p_bud/OneDrive/Desktop/Assignment/Big Data ML/Priyanka_Budavi_Submission_ML/Data/CSV files/Image01.csv" , header=None)

#Store the value of the CSC into X
X = input_data
row, col = X.shape

#Split the data into 80% - Training Dataset
TR = round(row*0.8)
X1 = np.array(X)

#remaining is the Testing Dataset
TT = row-TR

#Training and Testing Dataset
X_train = X1[0:TR,:]
X_test = X1[TR:row,:]

#saveTraining dataset
data_train = pd.DataFrame(X_train)
data_train.to_csv('C:/Users/p_bud/OneDrive/Desktop/Assignment/Big Data ML/Priyanka_Budavi_Submission_ML/Assignment2/Data/Train_Image01_NonOverlap.csv')

#save testing Dataset
data_test = pd.DataFrame(X_test)
data_test.to_csv('C:/Users/p_bud/OneDrive/Desktop/Assignment/Big Data ML/Priyanka_Budavi_Submission_ML/Assignment2/Data/Test_Image01_NonOverlap.csv', index=False)
```

Figure 56: Code to split the data into 80:20 training and testing dataset

12.2 Select two features in each category of training and testing datasets and plot their histograms to see if they follow the same distribution

I selected two features from each category of the dataset and tried to plot histogram for each feature. If you observe the train and test dataset, the two features follow the same gaussian curve. Then I tried to calculate the mean of the two features to verify the same. Figure 57-61 shows the images of the histogram representing different category of the dataset and how they are distributed. Figure 62-65 show the mean values of different category. I chose the features such that they dont differ much in the values.

```
Created on Wed Oct 20 22:28:29 2021

@author: p_bud
"""


import matplotlib.pyplot as plt
import pandas as pd


#read the file
input_data = pd.read_csv("C:/Users/p_bud/OneDrive/Desktop/Assignment/Big Data ML/Priyanka_Budavi_Submission_ML/Assignment2/Data/Training Dataset/Train_I

f61 = input_data['61']
f62 = input_data['62']

nbins = 40
plt.title('HIstogram of Non Overlapping Feature 61 Train data set/image012')
plt.hist(f61 ,nbins ,  color='g', edgecolor='k')
plt.axvline(f61.mean(), color='k', linestyle='dashed', linewidth=1)
plt.text(f61.mean(), 250 ,r'mean')
plt.show()


plt.title('HIstogram of  Non Overlapping Feature 62 Train data set/image012')
plt.hist(f62 ,nbins ,  color='g', edgecolor='k')
plt.axvline(f62.mean(), color='k', linestyle='dashed', linewidth=1)
plt.text(f62.mean(), 250 ,r'mean')
plt.show()
```

Figure 57: Code for histogram

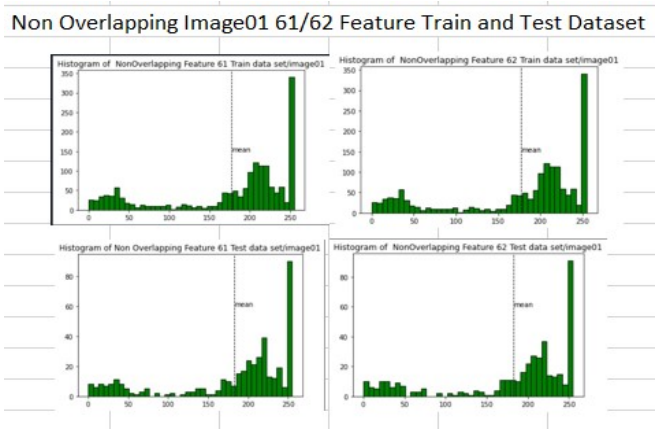


Figure 58: Combined Non Overlapping Train/Test dataset of Image01

Figure 59: Combined Overlapping Train/Test dataset of Image01

12.3 Use the same two features to generate scatter plots for each category of training and testing datasets

Figure 66 shows the code to generate the scatter plots for feature 61 and feature 62 of Image01 and Image012 for both overlapping and non overlapping datasets. Figure 67-74 shows the scatter plots of all category of dataset.

13 Implement a regression-based model

13.1 Implement and train lasso regression using the training sets of the overlapping and non-overlapping feature vectors (feature spaces)

I used the formula from the text book to calculate the coefficients to implement the lasso regression. I used the training data-set to train the model and calculated the coefficients. The value of coefficients is then multiplied to the training dataset to get the predicted values. The values obtained are in the range from 0 to 1 (Two class classifier). Now use the testing data-set and multiply the data with the co-efficients obtained previously and predict the values. Also, I tried to alter the lambda value till the accuracy is better. Lambda for non overlapping data set ranges from 0.1 to 150 and the accuracy is almost same. But after 150

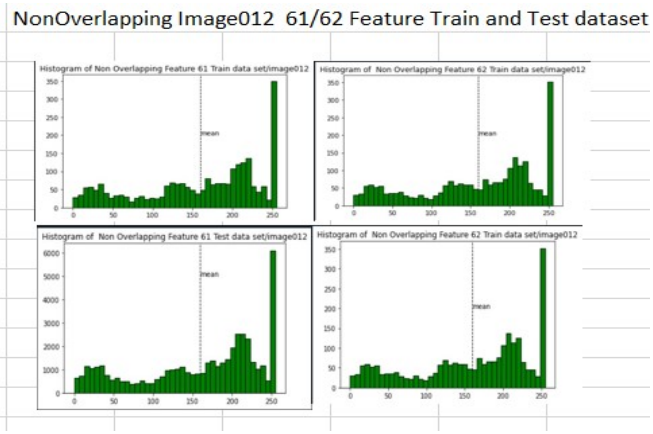


Figure 60: Combined Non Overlapping Train/Test dataset of Image012

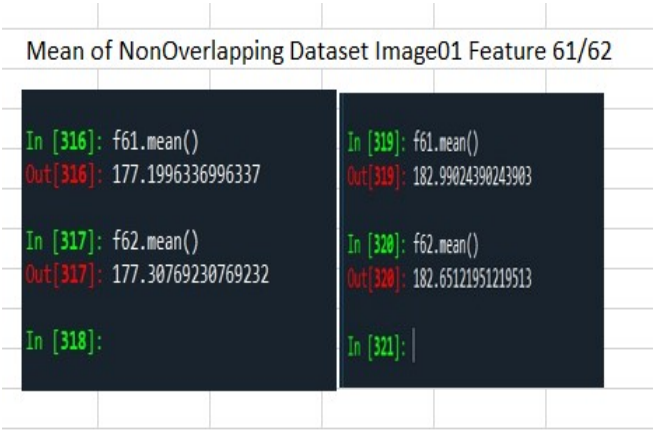


Figure 62: Combined mean values of Non Overlapping Train/Test dataset of Image01

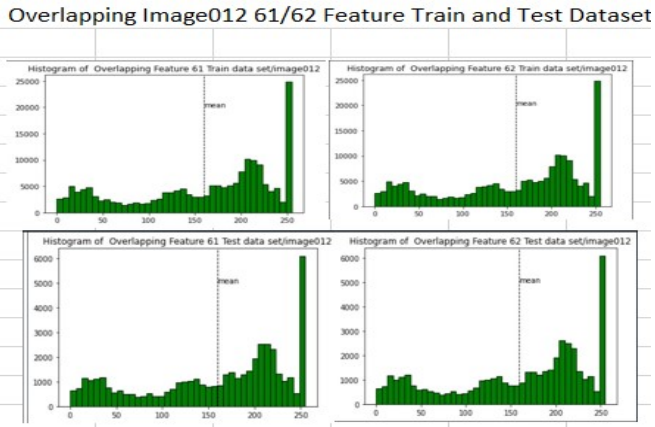


Figure 61: Combined Overlapping Train/Test dataset of Image012

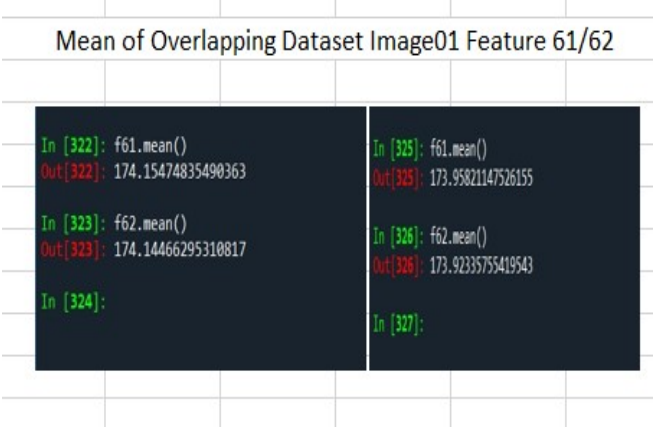


Figure 63: Combined mean values of Overlapping Train/Test dataset of Image01

the accuracy decreases. I did the same for both overlapping and non overlapping data-set. The lambda value used while getting the co-efficient is 0.1. Figure 75-77 shows the code for implementing lasso regression and also predicted the labels. The highlighted are the actual values and predicted values.

13.2 Adding the predicted values to the excel

The code in figure 75 saves the predicted values of the dataset and the predicted value is added to the 66th column of the dataset. Figure 76-77 show the saved columns.

13.3 Construct confusion matrices using the actual and predicted labels

I contructed the confusion matrix for the actual and the predicted values and can be seen in the figure 78 . The class zero is considered as a positive class. Figure 79-80 show the confusion matrix for overlapping and non overlapping dataset.

13.4 Comparing performance of Data sets

Calculated the measures from the overlapping and non-overlapping dataset for Image01. I selected accuracy and precision as the two measures. The performance of non overlapping dataset is better.For overlapping dataset the accuracy is 29% and the precision score is 36%. In the non overlapping dataset, accuracy is 40% and precision score is 42%. Figure 81 - 82 shows the measures for both the datasets.

14 Implement random forest or deep learning

14.1 Training Random Forest for Two Class Classifier

I have used RandomClassifier from sklearn.ensemble library to perform two class classification on merged dataset of image0 and image1. Figure 83 shows the implementation of the RandomForest. I used the two class data set to train the model i.e x_train stores my training data. I drop the label column from the dataset and store in X1. Now the data in X1 is my dataset that will be fed to the RandomForest object to fit the model. I then used the testing data set x_test to test the trained model. Using the predict() I feed only the x_test data without the labels to predict my y_hat values (Predicted Values/labels).I saved the obtained values in a separate files. I followed the same process for both overlapping and non-overlapping data set. I also generated the random forest tree to visualize how the data set

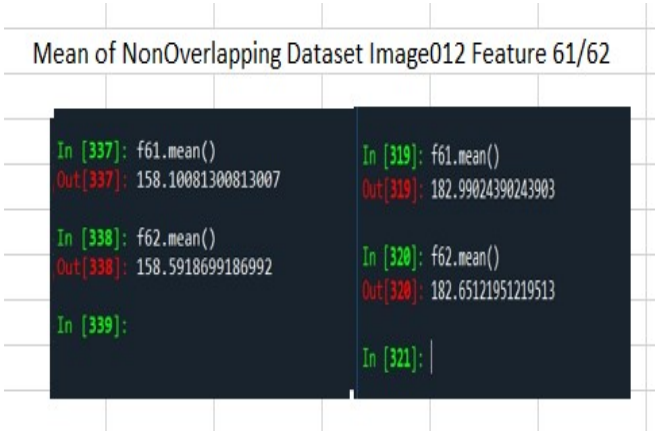


Figure 64: Combined mean values of Non Overlapping Train/Test dataset of Image012

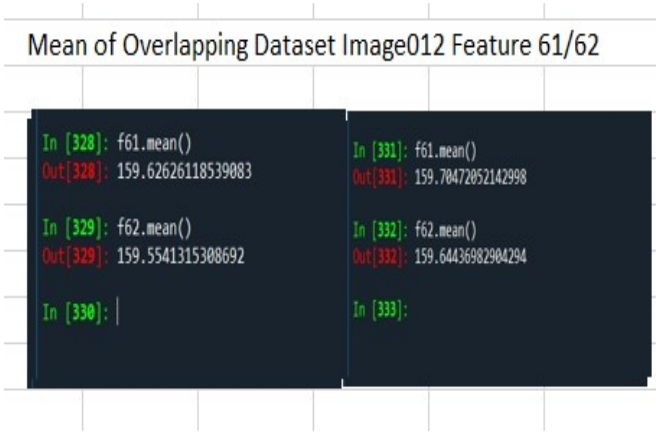


Figure 65: Combined mean values of Overlapping Train/Test dataset of Image012

```
# -*- coding: utf-8 -*-
"""
Created on Mon Oct 25 13:59:51 2021

@author: p_bud
"""


import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
df = pd.read_csv('C:/Users/p_bud/OneDrive/Desktop/Assignment/Big Data ML/Priyanka_Budavi_Submission_ML/

f61 = df['61']
f62 = df['62']
label = df['64']


fig = plt.figure(figsize = (20 , 20))
ax1 = fig.add_subplot(221)
ax1.set_title(" Image012 Non Overlapping Train dataset")
ax1.set_xlabel("Feature 61")
ax1.set_ylabel("Feature 62")
sns.scatterplot(x = f61 , y= f62 , hue= label , palette= "tab10")
plt.show()
```

Figure 66: Code for creating Scatter plot of Non Overlapping and Overlapping Dataset

Figure 67: Scatter plot for Feature 61 and 62 of NonOverlapping Image01 Train Dataset

is being split and values are predicted.

14.2 Training Random Forest for Three Class Classifier

For a three class classifier I used the merged data-set of Image0 , Image1 and Image2. The merged data set is then split into 80:20 ratio to train and test the model. I followed the same process as described for the two class classifier and saved the predicted values in a separate file. I also generated the Random Forest Trees and saved it as a pdf file. Figure 85 - 86 show the graphical visualization of the trees.

14.3 Construct confusion matrices using the actual and predicted label

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, 87 and 88 shows two class Confusion matrices respectively. Same is followed for a three class dataset.

14.4 Performance evaluation of the random forest model

I calculated the measures using the formula as well as the inbuilt functions. Figure 90 shows the measures using the formula. The class 0 is considered as a positive class while calculating using the formula. While using the inbuilt function which class is positive is not known. I choose accuracy and precision to determine the performance. In two class classifier the performance of overlapping data set is good as it is 98%. In three class the overlapping data set performed better with an accuracy of 99%. Also the precision for two class and three class overlapping data set performed better i.e 99%. Refer figure 89

15 Evaluation of the learning models

15.1 Evaluating of models using built-in measures

For this task, I am using inbuilt library accuracy_score, precision_score, recall_score from sklearn.metrics to evaluate the performance of all four types of data sets. Figure 90 show the execution of Lasso regression for in-built performance evaluation. All other data sets are executed in same way to get the results. Refer figure 91- 92 for other models.

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

The chart shows the performance of Lasso Regression and Random forest using Confusion matrix and in-built library

Figure 68: Scatter plot for Feature 61 and 62 of Overlapping Image01 Train Dataset

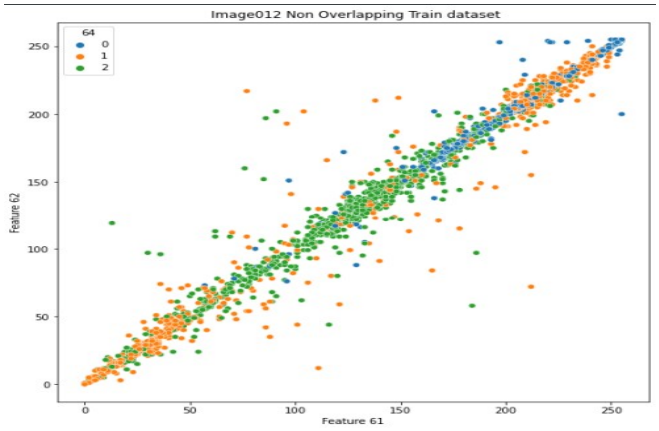


Figure 69: Scatter plot for Feature 61 and 62 of Non Overlapping Image012 Train Dataset

as shown in Figure 93. Looking at the results in figure 93, there is only slight difference in values generated by confusion based matrix and in-built metrics. Confusion based metrics has given better accuracy levels compared to in-built metrics. Also, in in-built metrics the positive class is unknown it always considered the default where as in confusion based we explicitly mention the positive class.

15.3 Comparing the model to find the best one

From Figure 93, 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. In overlapping data set we get many observation which results in better classification when random forest is considered. The accuracy is 98% for overlapping two class classifier and 99% for overlapping three class classifier.

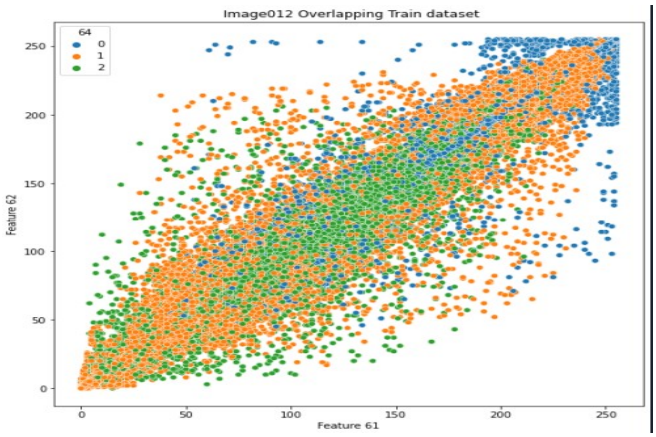


Figure 70: Scatter plot for Feature 61 and 62 of Overlapping Image012 Train Dataset

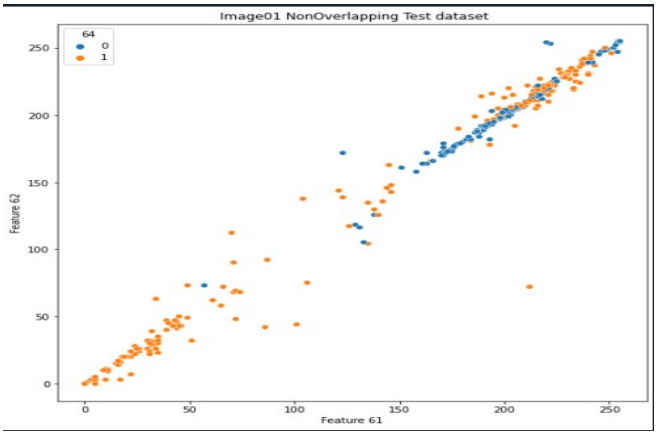


Figure 71: Scatter plot for Feature 61 and 62 of Non Overlapping Image01 Test Dataset

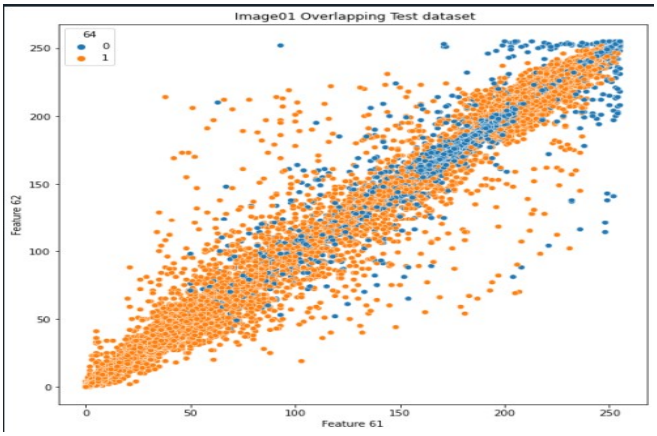


Figure 72: Scatter plot for Feature 61 and 62 of Overlapping Image01 Test Dataset

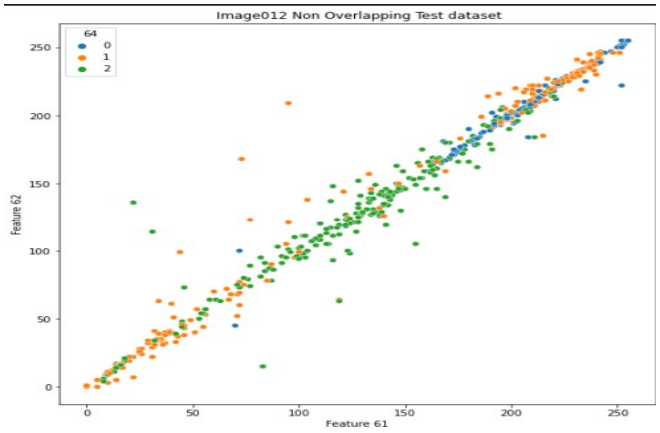


Figure 73: Scatter plot for Feature 61 and 62 of Non Overlapping Image012 Test Dataset

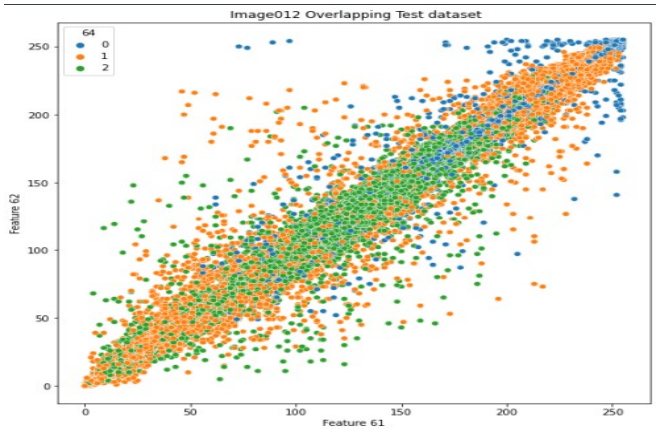


Figure 74: Scatter plot for Feature 61 and 62 of Overlapping Image012 Test Dataset

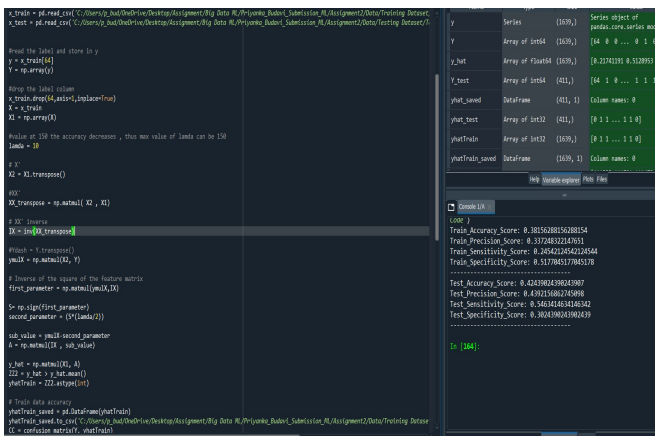


Figure 75: Implementation of Lasso Regression

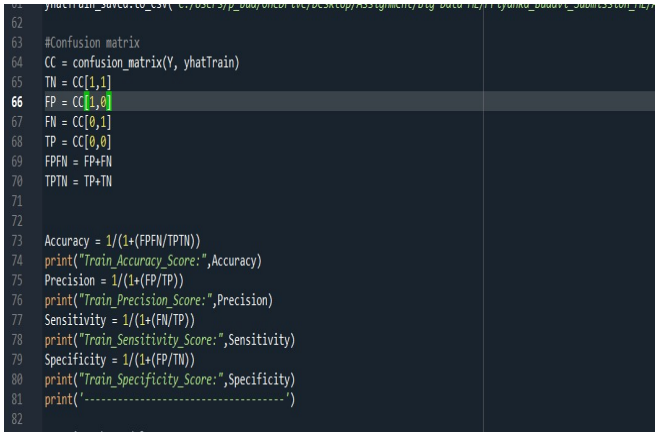


Figure 78: Confusion Matrix code

Figure 76: The Actual values and Predicted values for Non-Overlapping Dataset

Figure 77: The Actual values and Predicted values for Overlapping Dataset

```
In [120]:
    ...: print(" Confusion Matrix Train Dataset")
 Confusion Matrix Train Dataset

In [121]: CC
Out[121]:
array([[186, 633],
       [421, 398]], dtype=int64)

In [122]:
    ...: print(" Confusion Matrix Test Dataset")
 Confusion Matrix Test Dataset

In [123]: CC_test
Out[123]:
array([[ 44, 161],
       [ 97, 108]], dtype=int64)
```

Figure 79: Confusion Matrix for Non overlapping Dataset

```
Train_Accuracy_Score: 0.38705738705738707
Train_Precision_Score: 0.35569422776911075
Train_Sensitivity_Score: 0.2783882783882784
Train_Specificity_Score: 0.49572649572649574
-------------------------------------
Test_Accuracy_Score: 0.4097560975609756
Test_Precision_Score: 0.4280155642023346
Test_Sensitivity_Score: 0.5365853658536585
Test_Specificity_Score: 0.2829268292682927
-------------------------------------
```

Figure 82: Quantitative Measure of Non-Overlapping dataset

```
    ...: print(" Confusion Matrix Train Dataset")
 Confusion Matrix Train Dataset

In [127]: CC
Out[127]:
array([[ 7215, 56702],
       [36338, 32412]], dtype=int64)

In [128]:
    ...: print(" Confusion Matrix Test Dataset")
 Confusion Matrix Test Dataset

In [129]: CC_test
Out[129]:
array([[ 1784, 14228],
       [ 9160,  7995]], dtype=int64)

In [130]:
```

Figure 80: Confusion Matrix for Overlapping Dataset

```
x_train = pd.read_csv('C:/Users/p_bud/OneDrive/Desktop/Assignment/Big Data ML/Priyanka_Budavi_Submission_ML/Assignment2/Data/Tr
x_test = pd.read_csv('C:/Users/p_bud/OneDrive/Desktop/Assignment/Big Data ML/Priyanka_Budavi_Submission_ML/Assignment2/Data/Te

y = x_train[64]
Y = np.array(y)
x_train.drop(64,axis=1,inplace=True)
X = x_train
X1 = np.array(X)

# Train the model
randomFClassifier = RandomForestClassifier(random_state=0,n_estimators=50,oob_score=True, n_jobs=1)
randomFClassifier = randomFClassifier.fit(X1, Y)


#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('Predicted_01_NO.csv', index=False)


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]

FPFN = FP+FN
TPTN = TP+TN
```

Figure 83: Code for Random Forest

```
Train_Accuracy_Score: 0.29904949987562846
Train_Precision_Score: 0.1659466911764706
Train_Sensitivity_Score: 0.11299028427491903
Train_Specificity_Score: 0.47202909090909095
-------------------------------------
Test_Accuracy_Score: 0.2949618596798022
Test_Precision_Score: 0.3598650168728909
Test_Sensitivity_Score: 0.4662197610026232
Test_Specificity_Score: 0.11147889083187609
-------------------------------------
```

Figure 81: Quantitative Measure of Overlapping dataset

```
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]

FPFN = FP+FN
TPTN = TP+TN


#Qualitative measures of performance calculation using Confusion matrix
Accuracy = 1/(1+(FPFN/TPTN))
print("Test_Accuracy_Score:",Accuracy)
Precision = 1/(1+(FP/TP))
print("Test_Precision_Score:",Precision)
Sensitivity = 1/(1+(FN/TP))
print("Test_Sensitivity_Score:",Sensitivity)
Specificity = 1/(1+(FP/TN))
print("Test_Specificity_Score:",Specificity)

print("-----------------------------------------------------------")
print("Measures using Library")

print("-----------------------------------------------------------")
#Qualitative measures of performance calculation using sklearn metrics
print('sklearn.metrics Accuracy',accuracy_score(test_y, y_pred))
print('sklearn.metrics precision',precision_score(test_y, y_pred))
print('sklearn.metrics sensitivity',recall_score(test_y, y_pred))
```

Figure 84: Confusion Matrix for Random Forest

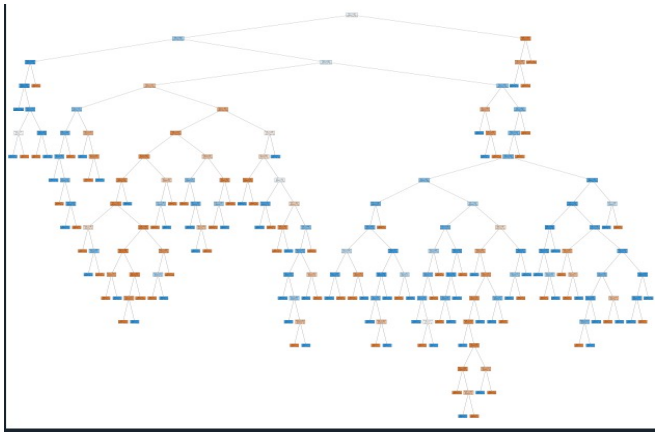


Figure 85: Random Forest Tree for Image01 Non Overlapping Dataset

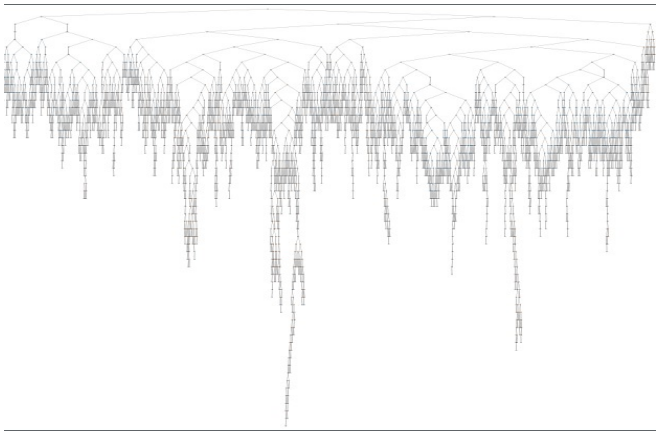


Figure 86: Random Forest Tree for Image01 Overlapping Dataset

```
In [215]: CC_test
Out[215]:
array([[194,  11],
       [ 15, 190]], dtype=int64)

In [216]:
```

Figure 87: Confusion Matrix for two class NonOverlapping Dataset

```
In [217]: CC_test
Out[217]:
array([[15801,   211],
       [  150, 17005]], dtype=int64)

In [218]:
```

Figure 88: Confusion Matrix for two class Overlapping Dataset

TWO CLASS OVERLAPPING				
N_ESTIMATORS	Accuracy	Precision	Sensitivity	Specificity
50	0.98	0.99	0.98	0.99
100	0.98	0.99	0.98	0.99
TWO CLASS NON OVERLAPPING				
N_ESTIMATORS	Accuracy	Precision	Sensitivity	Specificity
50	0.936	0.928	0.946	0.926
100	0.936	0.928	0.946	0.926
THREE CLASS OVERLAPPING				
N_ESTIMATORS	Accuracy	Precision	Sensitivity	Specificity
50	0.991	0.992	0.991	0.992
100	0.93	0.938	0.938	0.938
THREE CLASS NON OVERLAPPING				
N_ESTIMATORS	Accuracy	Precision	Sensitivity	Specificity
50	0.924	0.918	0.933	0.9166
100	0.93	0.938	0.938	0.938

Figure 89: Performance Measures of Two Class and Three class dataset

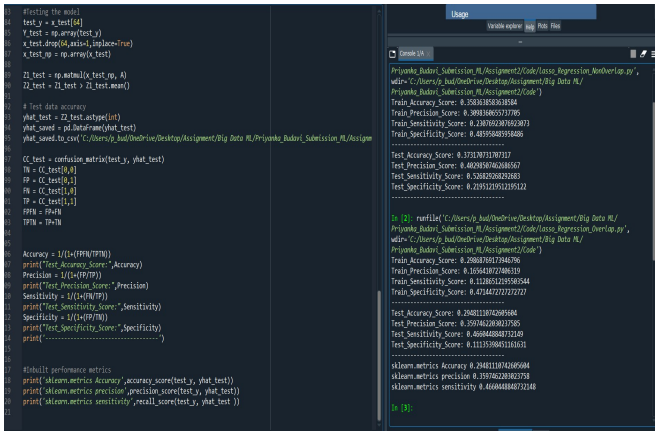


Figure 90: Lasso Regression Overlapping and Non Overlapping measures

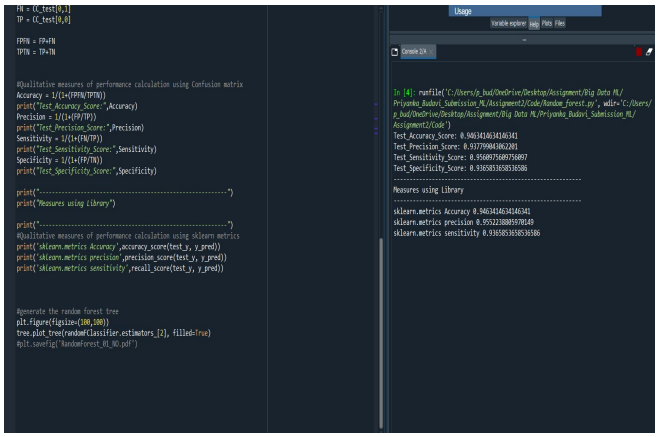


Figure 91: Random Forest two class measures

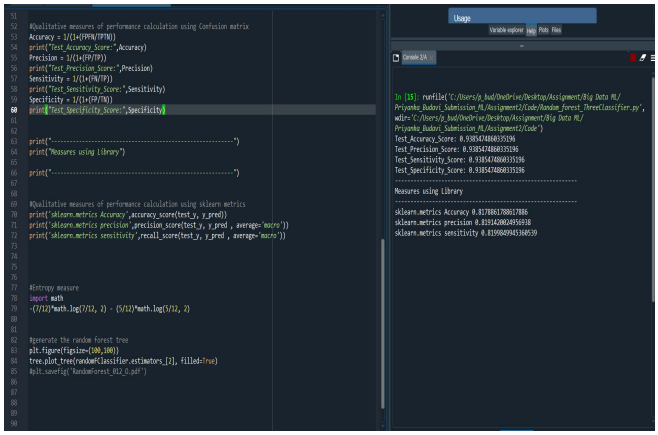


Figure 92: Random Forest three class measures

	Inbuilt Acuracy	Confusion Based Acuracy	Inbuilt Precision	Confusion Based Precision	Inbuilt Specificity	Confusion Based Specificity	Inbuilt Sensitivity	Confusion Based Sensitivity
RF Two class Nonoverlap(01)	0.936	0.936	0.945	0.928	0.926	0.946	0.9268	0.9268
RF Two class Overlap(01)	0.989	0.989	0.987	0.99	0.991	0.986	0.991	0.991
RF Two class Nonoverlap(012)	0.804	0.924	0.805	0.918	0.803	0.933	0.807	0.916
RF Two class Overlap(012)	0.95	0.991	0.958	0.992	0.956	0.992	0.57	0.991
Lasso Two Class Nonoverlap(01)	0.3731	0.373	0.59	0.4	0.373	0.826	0.373	0.485
Lasso Two Class overlap(01)	0.294	0.294	0.359	0.359	0.466	0.466	0.466	0.466
Lasso Two Class Nonoverlap(012)	0.195	0.29	0.119	0.375	0.119	0.36	0.192	0.509
Lasso Two Class overlap(012)	0.2013	0.276	0.119	0.355	0.179	0.491	0.19	0.46

Figure 93: Built-in measures and confusion based measure

16 ASSIGNMENT 3: A CLASSIFIER ON A BIG DATA SYSTEM

17 Completed tasks from Assignment 1 and Assignment 2

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

1. 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.

2. 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 was selected.

18 Understanding and using the Data-bricks

18.1 Signing up Data-bricks community version

I created an account to sign in for community version of data-bricks. I used my email ID too create the same.

18.2 Exploring Quick-start tutorial and setting up Big data environment in Data-bricks community

I created my community Data-bricks account which gives around 15GB of free memory. Then, created cluster CSC610 using "Create Cluster" option. I chose 7.3 ML version Apache spark 3.0.1, Scala 2.12 as there was no 7.2 version present in the list. I refered the video uploaded on data-bricks and created the tables as shown in the video. I created 4 data-set that is overlapping and non-overlapping data-set for two class and three class.

18.3 Cluster creation

I created my cluster and named it as CSC610 and it took around 5 min to create one. Figure 94 shows the creation of a cluster.

18.4 Table creation

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. The data for image01_nonoverlap , image012_nonoverlap , image01_overlap , image012_overlap was created as shown in the figure 95. The data was set huge for overlapping and also I was getting error, so I reduced the data and created them using the UI way.

19 Converting the code to run your data sets

For this task, I am using the random forest model which I implemented in assignment 2 . Previously I calculated the

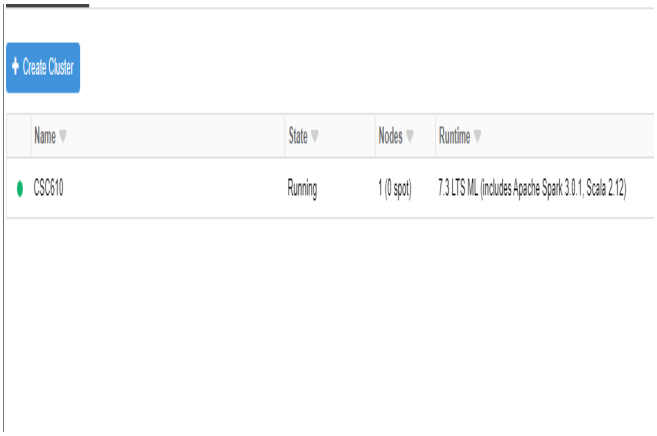


Figure 94: Cluster Creation

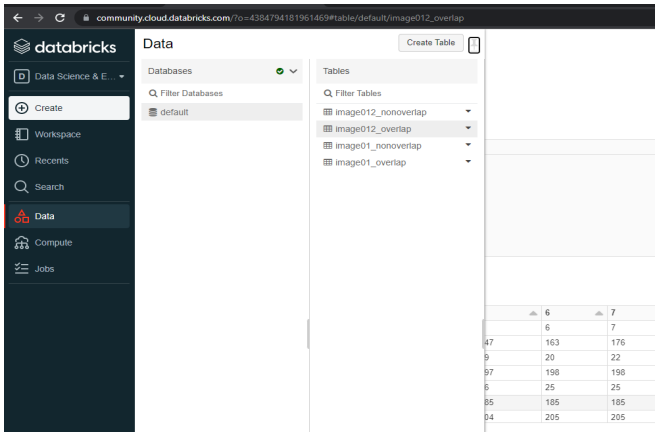


Figure 95: Data-frame creation

accuracy by changing the n_estimators and then chose the best model amongst all of them. So I will use the same data in data-bricks environment and compute the accuracy. Below are the general steps followed to calculate the same. I showed the execution for one data-set Same process is followed for the other three data-sets.

19.1 Pyspark

Imported the pyspark as shown in the figure 96 for computational purpose and fetching the data from pyspark.sql.dataframe.Dataframe which was uploaded earlier in Task2 using the select query as shown in the video.

19.2 Splitting the data in train and test

After importing the input data we should check if there are any null values present and this can be done by using isna() function. Then, split the data into 80% Training data-set and 20% Testing data-set. Splitting is done using train_test_split() function imported from sklearn.model_selection module. I split the data into 80:20 by setting the value of test_size = 0.2 Figure 97 shows the code for splitting the data.

19.3 Histogram and Box-Plot for the data-set

Figure 98 and 99 shows the histogram and box-plots respectively for Feature 30 using sea-born package

Figure 96: Import Pyspark

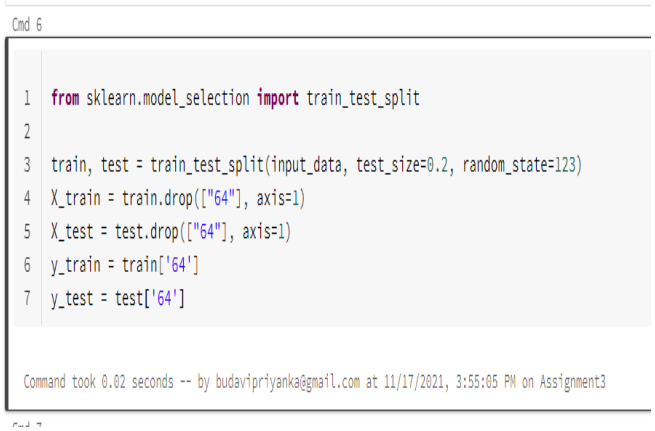


Figure 97: Code for splitting the data

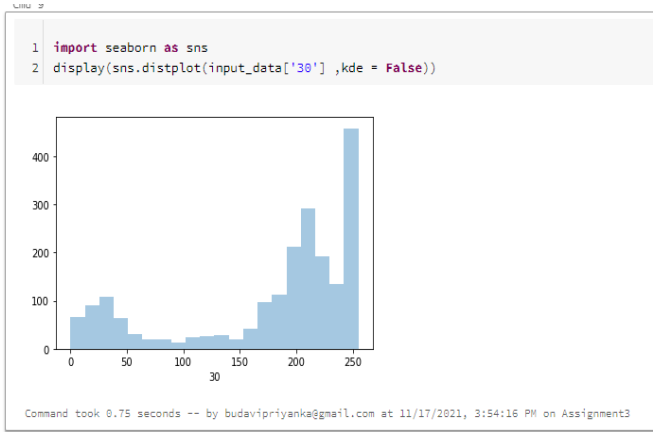


Figure 98: Histogram for feature 30

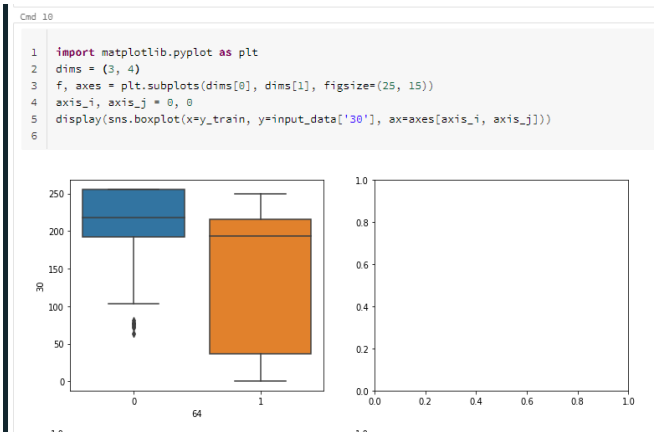


Figure 99: Box-plot for feature 30

19.4 Random-Forest Classifier : Training Model

RandomForestClassifier is trained using 80% Train data set for 50 estimators(as this was the best predicted model in this scenario). The time take to train the model took 0.38 sec on data-bricks. Using x_test we predict the labels for testing data-set. This command took 0.12 seconds to predict the values. Figure 100 show the code for 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 yaxis and fpr along xaxis. Figure 101 and 102 shows the generated ROC curve.

19.6 Feature Importance

Feature importance refers to a class of techniques for assigning scores to input features to a predictive model that indicates the relative importance of each feature when making a prediction. Importance of every feature is calculated to find the feature that is maximum impacting in classification. Figure 103 shows feature 63 seems to have highest importance compared to other features. All the features are sorted to know the importance of the highest feature.

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 104 shows the execution of accuracy and confusion matrix.

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

20.1 Local System Results

Two class Nonoverlap data set executed the model in just 0.16 seconds. Three class Nonoverlap data set just took 6.62 seconds to execute or train the model. Two class overlap dataset took 25.32 seconds and three class overlap data set took 7.42 minutes. The overlapping dataset took longer time to train the model.

20.2 Biga Data System Results

Two class Nonoverlap data set executed the model in just 0.44 seconds. Three class Nonoverlap data set took 11.7 seconds to execute or train the model. Two class overlap dataset took 37.71 seconds and three class overlap data set took 8.89 minutes. The overlapping dataset took longer

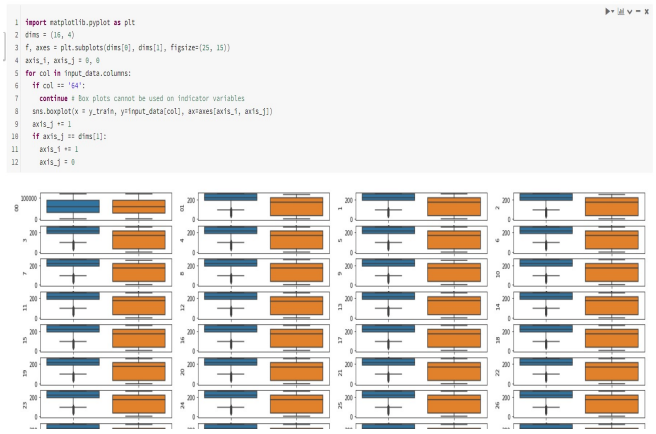


Figure 100: Box-plot for Multiple features

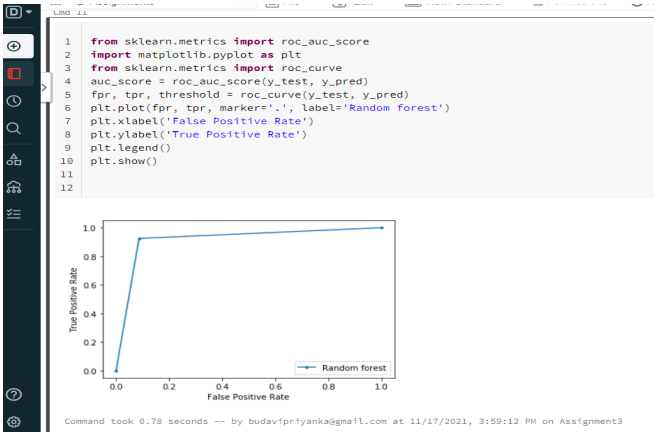


Figure 102: ROC curve

Figure 101: Using the RandomForest Model

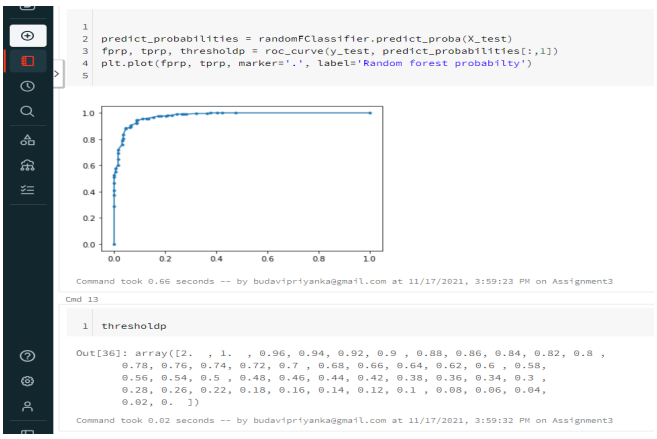


Figure 103: ROC curve with multiple threshold values

time to train the model. Figure 105 shows the observations for local system and big data system. Executed all the datasets and noted the time taken to train the model. Figure 106 shows the code implementation of Random Forest and displaying the time taken to train the model.

20.3 Comparing the Local system and Big data system

From the table i.e Figure 105 we observe that the time taken to train the model for two class overlap and non overlapping data-set, the local system performed better. This means that for small data-set big data system took longer time to train the model. The data-set for overlapping three class classifier, the number of observations are many and hence the accuracy will be great and also it will take more time to compute compared to other data-sets. To check this I chose three class overlapping data-set and increased the n_estimators to 1500. Till now estimator with 1000 the local system performed better. As the value of n_estimators increased to 1500 the local system took more time to compute i.e it took 12.87 minutes. While the big data system took 12.71 minutes. Thus by this we can conclude that as the number of data increases the big data performs better than the local system. Figure 107 and 108 shows the computation time with 1500 n_estimators. Figure 110 shows the table of computed time to run on local and big-data. With 1GB data we observe that the system failed and showed error when trying to deal with large data-set. During the creation of data-set I was getting error uploading the UI way but I was able to upload the data via notebook and created the table for the same. I used multiple three class overlapping data-set to create 1GB data. I used this data-set because this would be the best fit to test the time taken to train model on Big data and Local System as it was taking more time to train as showed in the Figure 106. After running the model I conclude that, as the number of observations increases the system will perform low and further fail due to large data-set and in such situation big data performs better.

References

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

```
import pandas as pd
feature_importances = pd.DataFrame(randomFClassifier.feature_importances_, index=X_train.columns.tolist(), columns=['importance'])
feature_importances.sort_values('importance', ascending=False)
```

Out[57]:

	importance
63	0.036992
62	0.035737
60	0.034631
15	0.034506
14	0.030171
...	...
36	0.004839
4	0.004760
29	0.004606
20	0.003778
37	0.003513

64 rows × 1 columns

Figure 104: ROC curve with multiple threshold values

Cmd 15

```
from sklearn.metrics import accuracy_score
auccuracy = accuracy_score(y_test, y_pred)
print(auccuracy)
```

```
0.9195121951219513
```

Command took 0.02 seconds -- by budavipriyanka@gmail.com at 11/17/2021, 3:59:42 PM on Assignment3

Cmd 16

```
from sklearn.metrics import confusion_matrix
CC_test = confusion_matrix(y_test, y_pred)
print(CC_test)
```

```
[[179  17]
 [ 16 198]]
```

Command took 0.03 seconds -- by budavipriyanka@gmail.com at 11/17/2021, 3:59:48 PM on Assignment3

Cmd 17

```
from sklearn.metrics import roc_auc_score
auc_score = roc_auc_score(y_test, y_pred)
print(auc_score)
```

```
0.919249475491131
```

Command took 0.03 seconds -- by budavipriyanka@gmail.com at 11/17/2021, 3:59:51 PM on Assignment3

Figure 105: Accuracy , Confusion Matrix , AUC score

Local and Big Data execution Random Forest				
	NonOverlap01	NonOverlap012	Overlap01	Overlap012
Estimators	50	1000	100	1000
Big Data Environment	0.44	11.7	37.71	8.89 minutes
Local	0.16	6.62	25.32	7.42 minutes
Accuracy	0.91	0.8276	0.98	0.9517
Values are in seconds				

Figure 106: Comparing local and Big-data environment

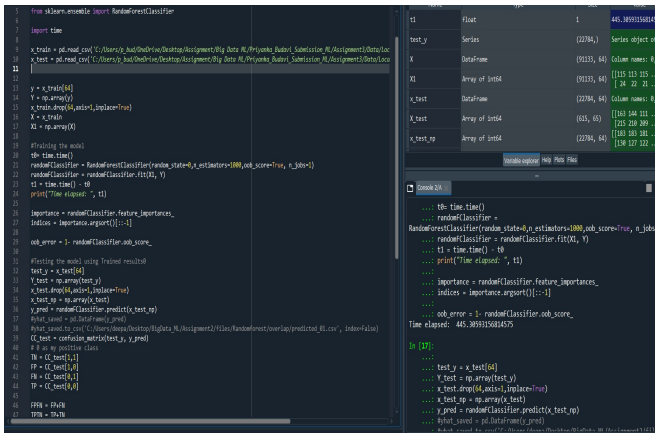
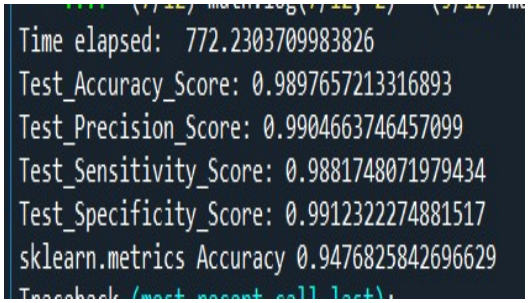


Figure 107: Random Forest code to display the time take to train the model



```
Time elapsed:  772.2303709983826
Test_Accuracy_Score: 0.9897657213316893
Test_Precision_Score: 0.9904663746457099
Test_Sensitivity_Score: 0.9881748071979434
Test_Specificity_Score: 0.9912322274881517
sklearn.metrics Accuracy 0.9476825842696629
```

Figure 108: Local Computation Time

```
from sklearn.ensemble import RandomForestClassifier
randomFClassifier = RandomForestClassifier(random_state=0,n_estimators=1500,oob_score=True, n_jobs=1)
randomFClassifier = randomFClassifier.fit(X_train, y_train)
```

Command took 12.71 minutes -- by budavipriyanka@gmail.com at 11/19/2021, 1:54:08 PM on Assignment3

Figure 109: Big data computation Time

Overlap three class (huge Dataset)				
Estimator value	Big Data Time (min)	Local Time (min)	Local Accuracy	Big data Accuracy
500	40.8	Error	NA	89.5
250	24.86	32.85	88.853	87
150	18.38	20.56	88.93	88.98
100	9.74	10.35	89.3	88.87

Figure 110: System Error with huge Dataset(1GB)