Big Data Assignment1: Feature Space Generation

Manish Shah
University of North Carolina at Greensboro
m_shah2@uncg.edu

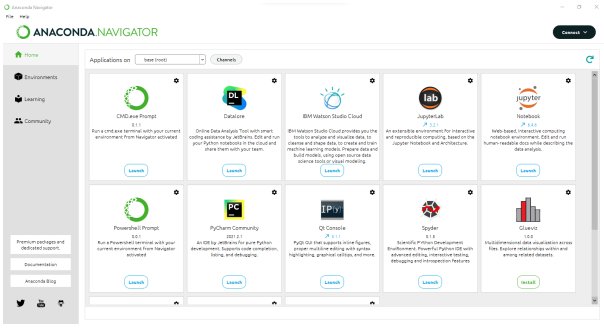


Fig. 1: Anaconda GUI

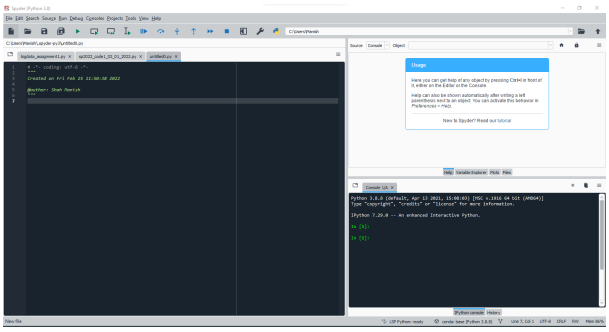


Fig. 2: SpyderIDE

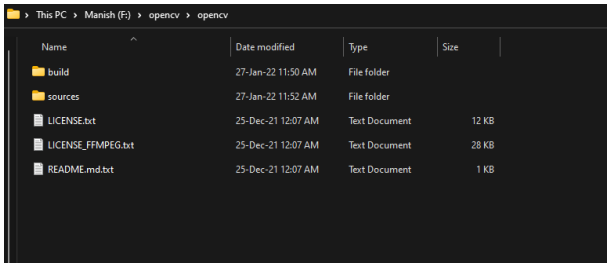


Fig. 3: OpenCV

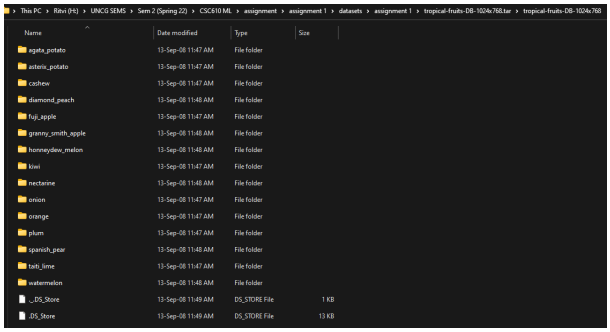


Fig. 4: Tropical Fruits Folder

I. Task 1: Build your programming environment!

1.1 Download and install Anaconda software.

I have downloaded and install the latest version of Anaconda software on local machine. It has inbuilt python packages. Also JupyterNotebook and Spyder can be installed using Anaconda. The Figure 1 shows the anaconda installation.

1.2 Install Spyder IDE.

The latest version of Spyder can be downloaded from Anaconda software. It can be accessed from Anaconda bundle and also by directly going inside the Spyder directory in the local machine. The Figure 2 shows sypder IDE installed in the local machine.

1.3 Install Jupyter Notebook.

Similar to Spyder, the latest version of JupyterNotebook can be downloaded from Anaconda software. It can be accessed from Anaconda bundle and also by directly searching Jupyter-Notebook in the local machine.

1.4 Download and install OpenCV.

OpenCV can be dowloaded online or via anaconda prompt. In my case, I downloaded it online from here and added it to the system variables. The Figure 3 shows OpenCV installed on the local machine.

II. Task 2: Download or generate a fruits and vegetables image dataset!

2.1 Tropical fruit dataset.

This dataset can be downloaded from here and contains images of tropical fruits. The folder is shown in Figure 4.

2.2 FIDS30 Dataset.

The FIDS dataset can be downloaded from here. The folder contains images of 32 kinds of fruits and is shown in Figure 5.

2.3 Image selection.

For this assignment, I am selecting images from FIDS30 dataset. I have selected cantaloupe, banana and tomato images. The images are selected such that there is no background except the fruit images. I am representing cantaloupe by 16.jpg, banana by 43.jpg and tomato by 5.jpg in the project. The selected images are displayed in Figure 6-Figure 8.

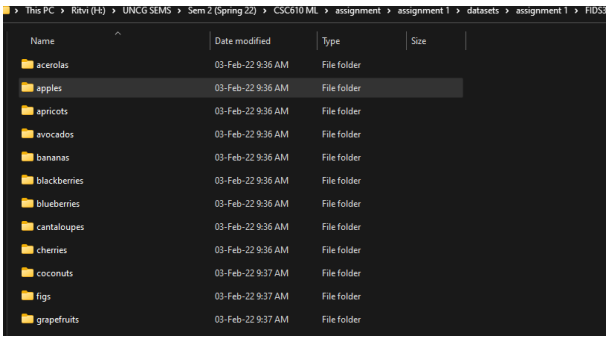


Fig. 5: Tropical Fruits Folder

Fig. 6: Cantaloupe image

III. Task 3:Read images and display in the programming environment

3.1 Read image and display R G B channel.

For this assignment I am using Spyder IDE. OpenCV library is used to read color images of fruits and display the R,G,B color channels for each image and plot the images using matplotlib library. The code snippet is shown in Figure 9.

3.2 Convert images to gray scale and display dimensions.

Fig. 7: Banana image

Fig. 8: Tomato image

```
import pandas as pd
import cv2
import matplotlib.pyplot as plt
import numpy as np

# Read Images (e.g, .jpg, .png, and .tif)
cant_color = cv2.imread("C:/Users/Manish/Desktop/bigdata_assignment1/16.jpg")
bana_color = cv2.imread("C:/Users/Manish/Desktop/bigdata_assignment1/43.jpg")
toma_color = cv2.imread("C:/Users/Manish/Desktop/bigdata_assignment1/5.jpg")

# cant_color = cv2.imread("C:/Users/s_suthah/Desktop/Images/02_03.png")
# bana_color = cv2.imread("C:/Users/s_suthah/Desktop/Images/s_02-_04.tif")

# Display the color channels
plt.imshow(cant_color[:,:,0])
plt.imshow(cant_color[:,:,1])
plt.imshow(cant_color[:,:,2])

plt.imshow(bana_color[:,:,0])
plt.imshow(bana_color[:,:,1])
plt.imshow(bana_color[:,:,2])

plt.imshow(toma_color[:,:,0])
plt.imshow(toma_color[:,:,1])
plt.imshow(toma_color[:,:,2])
```

Fig. 9: Code to display R,G,B values of fruits images

In this step, first colored images are converted to grayscale using cvtColor function of openCV library. To display dimensions of grayscale images, image.shape function is used. The code is displayed in Figure 10 and grayscale images are shown in Figure 11 - Figure 13.

```
@author: Shah Manish
"""
import pandas as pd
import cv2
import matplotlib.pyplot as plt
import numpy as np

# Read Images (e.g, .jpg, .png, and .tif)
cant_color = cv2.imread("C:/Users/Manish/Desktop/bigdata_assignment1/16.jpg")
bana_color = cv2.imread("C:/Users/Manish/Desktop/bigdata_assignment1/43.jpg")
toma_color = cv2.imread("C:/Users/Manish/Desktop/bigdata_assignment1/5.jpg")

# Convert to grayscale.
cantG = cv2.cvtColor(cant_color, cv2.COLOR_BGR2GRAY)
heightcG, widthcG = cantG.shape

banaG = cv2.cvtColor(bana_color, cv2.COLOR_BGR2GRAY)
heightbG, widthbG = banaG.shape

tomaG = cv2.cvtColor(toma_color, cv2.COLOR_BGR2GRAY)
heighttG, widthtG = tomaG.shape
```

Fig. 10: Code to convert the images to grayscale

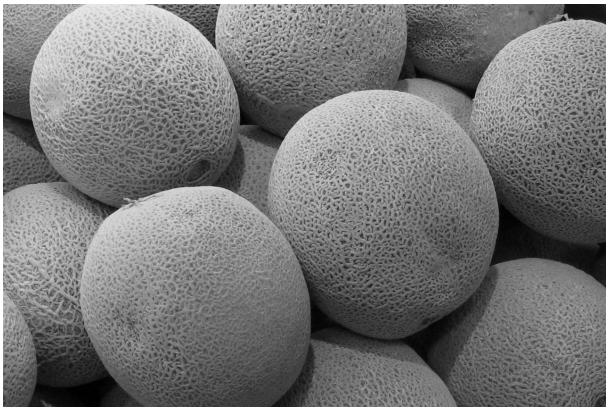


Fig. 11: Cantaloupe gray

Fig. 12: Banana gray

Fig. 13: Tomato gray

```
@author: Shah Manish
"""
import cv2
#read  the image
cantG = cv2.imread('C:/Users/Manish/Desktop/bigdata_assignment1/output_new/16_gray.jpg')
#print the original dimensions
print("Original Dimensions:" ,cantG.shape )
std_height = 261
height = cantG.shape[0]
width = cantG.shape[1]

#resize function
def newwidth(height , width):
    print("Original Height :", height)
    print("Original Width :", width)
    aspect_ratio = round(height/width)
    new_width = aspect_ratio * 261
    print("My width :" , new_width)
    return round(new_width)

#function to find divisble by 9
def div9(dim):
      remainder = dim % 9
      if remainder == 0:
          dim = dim
      else:
          dim = dim + (9-remainder)
      return dim

#Values when both need to be divisible by 9
new_fruit_width = newwidth(height,width)
new_height = std_height

#Values when both the height and width is to be divisible by 9
print("New Dimension of image divisible by is 9:" , new_height, new_fruit_width)
image_resized = cv2.resize(cantG, dsize = (new_height , new_fruit_width))
cv2.imwrite('C:/Users/Manish/Desktop/bigdata_assignment1/output_new/16_grayresized.jpg' , image_resized)
```

Fig. 14: Function to resize grayscale images divisible by 9

```
"""
Created on Sun Feb 20 23:49:17 2022

@author: Shah Manish
"""
import cv2
#read  the image
cantG = cv2.imread('C:/Users/Manish/Desktop/bigdata_assignment1/output_new/16_gray.jpg')
#print the original dimensions
print("Original Dimensions:" ,cantG.shape )
std_height = 261
height = cantG.shape[0]
width = cantG.shape[1]

#resize function
def newwidth(height , width):
    print("Original Height :", height)
    print("Original Width :", width)
    aspect_ratio = round(height/width)
    new_width = aspect_ratio * 261
    print("My width :" , new_width)
    return round(new_width)

#function to find divisble by 9
def div9(dim):
      remainder = dim % 9
      if remainder == 0:
          dim = dim
      else:
          dim = dim + (9-remainder)
      return dim

#Values when both need to be divisible by 9
new_fruit_width = newwidth(height,width)
new_height = std_height

#Values when both the height and width is to be divisible by 9
print("New Dimension of image divisible by is 9:" , new_height, new_fruit_width)
image_resized = cv2.resize(cantG, dsize = (new_height , new_fruit_width))
cv2.imwrite('C:/Users/Manish/Desktop/bigdata_assignment1/output_new/16_grayresized.jpg' , image_resized)
```

Fig. 15: Find new width of cantG image

IV. Task 4: Resize the images to reduce their dimensions!

4.1 Resize the grayscale images so that dimensions are divisible by 9.

In this I have written a function which reduces the grayscale images in pixel size, such that the output dimensions are divisible by 9. For this a parametric function is created in python, which checks divisibility of the dimensions (width/height) by 9. If not, reduce it, such that it is divisible by 9 by subtracting it from the remainder. The code snippet is shown in the Figure 14.

4.2 Resize the grayscale images to height of 261.

Here the height of grayscale images is fixed as 261. The code is written which maintains the aspect ratio and calculate the new width of the images such that it is divisible by 9. The code is applied on all the three fruit images individually and the new images are stored with a different names. Figure 15 shows code for finding the new width of cantaloupe.

V. Task 5: Generate block-feature vectors!

5.1 Divide image into block of 9*9 pixels and generate non overlapping feature space

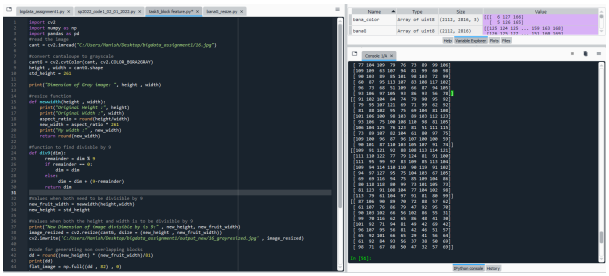


Fig. 16: Generating non overlapping feature vectors

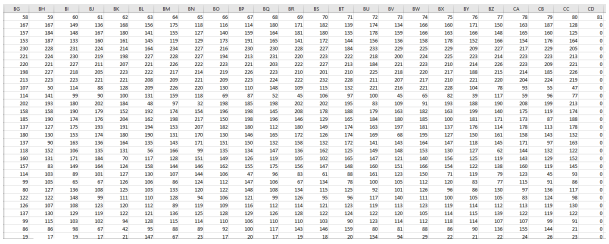


Fig. 17: Non overlapping Feature space for Cantaloupe having label 0

In this part, the resized grayscale images are used. The code for converting images to grayscale and resizing is added with the code of creating non overlapping blocks of 9*9 size to generate feature vectors of size 9*9. The feature vectors are stored in a .csv file for each image. The labels for images of cantaloupe, banana and tomato in the feature space in the .csv file with labels of 0, 1, 2. All the images contains 841 blocks of 9*9 size. In the feature space there is a total of 81 features and 1 column for label and total of 841 observations excluding the top row. Figure 16 shows the code for creating non overlapping feature space and Figure 17 contains .csv file generated for cantaloupe fruit.

VI. Task 6: Generate sliding block-feature vectors!

6.1 Divide image into block of 9*9 pixels and generate overlapping feature vector space

In this part, an overlapping block of size 9*9 is used to generate feature vector space of 81 columns. Images of cantaloupe, banana and tomato are labelled with 0, 1 and 2 respectively. The feature vectors are stored in a .csv file which contains 64010 blocks each. The code contains converting colored image to grayscale and resizing the images to get width divisible by 9 and overlapping code. The code snippet is shown in Figure 18 and the generated csv in Figure 19.

VII. Task 7: Derive statistical descriptors!

7.1 Extract statistical information about number of observations and data dimensions

For non overlapping feature space there are 841 observations with 81 features for cantaloupe, banana and tomato images. In case of overlapping feature space, there are 64010 observations with 81 features for all images.

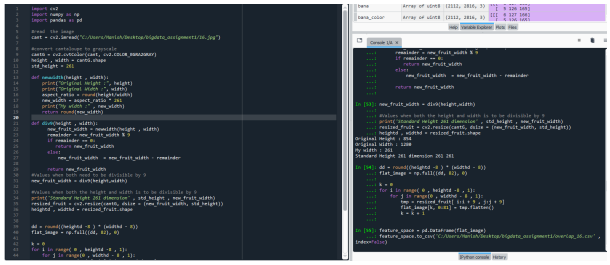


Fig. 18: Generating overlapping feature vectors

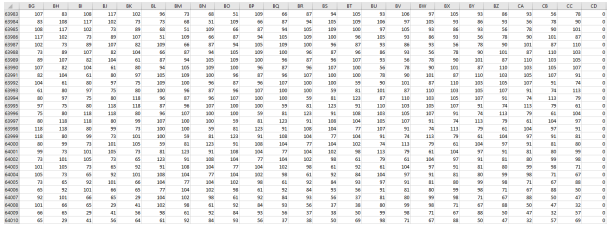


Fig. 19: Overlapping Feature space for Cantaloupe having label 0

7.2 Extract statistical information about mean of each feature

To calculate mean, I took the non overlapping .csv files and find the mean of each feature for each image and plot it which can be seen in Figure 21. The code snippet for finding the mean can be seen in Figure 20. From the two plots we can see that for the overlapping blocks, the difference between mean values of consecutive features is very less as compared to the non overlapping blocks. From mean we can see that banana has higher mean than both, cantaloupe has higher mean than tomato. This tells that the average values in banana is higher, which infers it has brighter image pixels than both. Secondly we can infer cantaloupe image is brighter than tomato image.

7.3 Extract statistical information about standard deviation of each feature

In this I have calculated standard deviation of each feature for every fruit in non overlapping blocks csv files. The code for calculating standard deviation is in the Figure 23 and the

```
import matplotlib.pyplot as plt
import pandas as pd

#read the csv file
cant = pd.read_csv('C:/Users/Manish/Desktop/bigdata_assignment1/16_nonoverlap.csv')
bana = pd.read_csv('C:/Users/Manish/Desktop/bigdata_assignment1/43_nonoverlap.csv')
toma = pd.read_csv('C:/Users/Manish/Desktop/bigdata_assignment1/5_nonoverlap.csv')

#calculate the mean of the CSV files
mean_cant =  round(cant.mean())
mean_bana =  round(bana.mean())
mean_toma =  round(toma.mean())

a =print(mean_cant)
b =print(mean_bana)
c =print(mean_toma)

#set x and y values
plt.title( 'Mean Plot Non-Overlapping Set' )
plt.xlabel('Features')
plt.ylabel('Mean Values')
#Plot the mean values
plt.plot(mean_cant, 'r')
plt.plot(mean_bana, 'b')
plt.plot(mean_toma, 'g')
```

Fig. 20: Code to calculate mean of fruit data

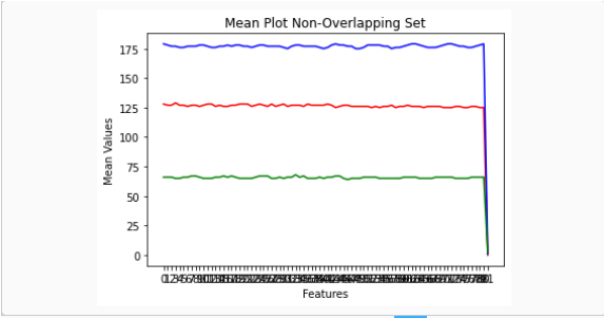


Fig. 21: Mean of non overlapping fruit data

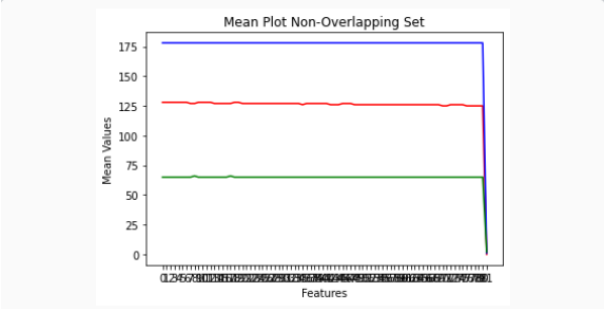


Fig. 22: Mean of overlapping fruit data

plot is shown in Figure 24. From the plot it can be seen the standard deviation is increasing among all fruit data. From the plot it can be seen that banana has higher standard deviation than others. So it is more spread out while other fruit values are more closely cluster around the mean.

7.4 Histogram of fruit images

I have selected one feature for each image and plot histogram for both overlapping and non overlapping feature vectors. For cantaloupe, the features are skewed towards the middle for both overlapping and non overlapping. For banana, the features are skewed towards the right, for both overlapping

```python
import matplotlib.pyplot as plt
import pandas as pd
#import numpy as np

#read the csv file
cant = pd.read_csv('C:/Users/Manish/Desktop/bigdata_assignment1/16_nonoverlap.csv')
bana = pd.read_csv('C:/Users/Manish/Desktop/bigdata_assignment1/43_nonoverlap.csv')
toma = pd.read_csv('C:/Users/Manish/Desktop/bigdata_assignment1/5_nonoverlap.csv')



std_cant =  round(cant.std())
std_bana =  round(bana.std())
std_toma =  round(toma.std())

a =print(std_cant)
b =print(std_bana)
c =print(std_toma)


#Plot the mean values
plt.plot(std_cant, 'r')
plt.plot(std_bana, 'b')
plt.plot(std_toma, 'g')

#Setting the Co-ordinate values
plt.title( 'Standard Deviation for nonoverlapping data' )
plt.xlabel('Features')
plt.ylabel('standard deviation')
```

Fig. 23: Code to find Standard deviation of non overlapping fruit data

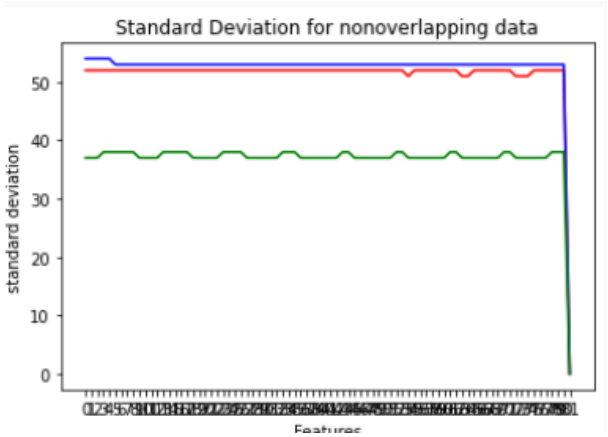


Fig. 24: plot of standard deviation of overlapping fruit data

```python
import matplotlib.pyplot as plt
import pandas as pd


#read the fruit
cant = pd.read_csv('C:/Users/Manish/Desktop/bigdata_assignment1/overlap_16.csv')
bana = pd.read_csv('C:/Users/Manish/Desktop/bigdata_assignment1/overlap_43.csv')
toma = pd.read_csv('C:/Users/Manish/Desktop/bigdata_assignment1/overlap_5.csv')

feature = cant['16']


nbins = 30
plt.hist(feature, nbins )
plt.title("Cantaloupe Histogram for feature 16")
plt.xlabel('Feature 16')
plt.ylabel('value')
plt.show()
```

Fig. 25: Histogram generation code

and non overlapping. For tomato, the features are skewed towards the left for both overlapping and non overlapping. The code is shown in Figure 25 and histograms are shown below that.

7.5 Scatter Plot of fruit images

I have generate scatter plot of all the three fruits for both overlapping and non overlapping feature vectors. I have selected feature 16 and 27 for cantaloupe, features 45 and 64 for banana and features 23 and 68 for tomato. The code can be seen in Figure 32 and the scatter plots are mentioned below

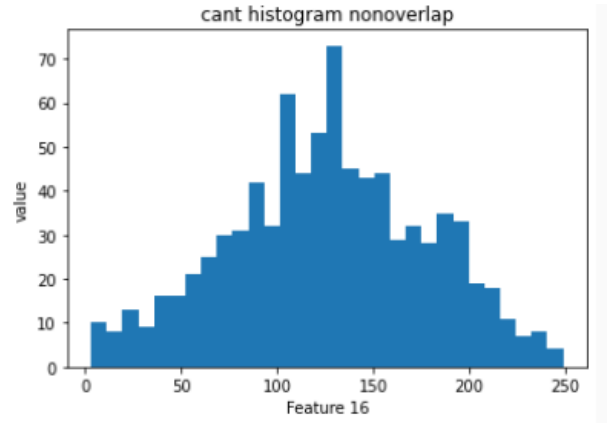


Fig. 26: Histogram of cant non overlap

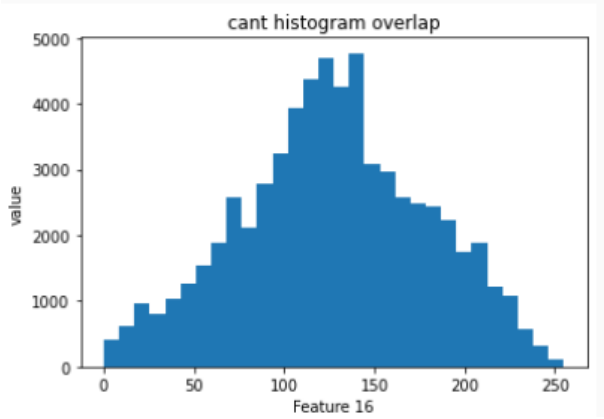


Fig. 27: Histogram of cant overlap

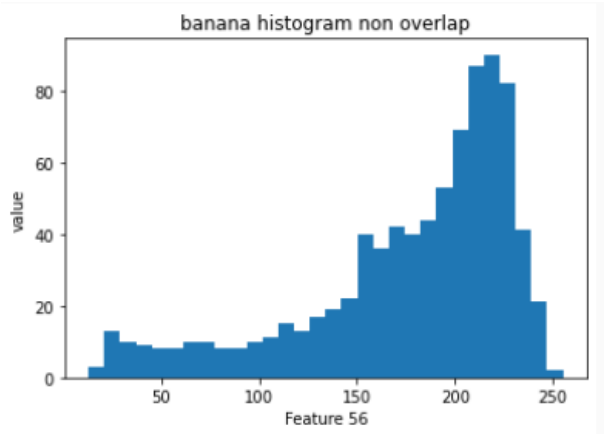


Fig. 28: Histogram of banana non overlap

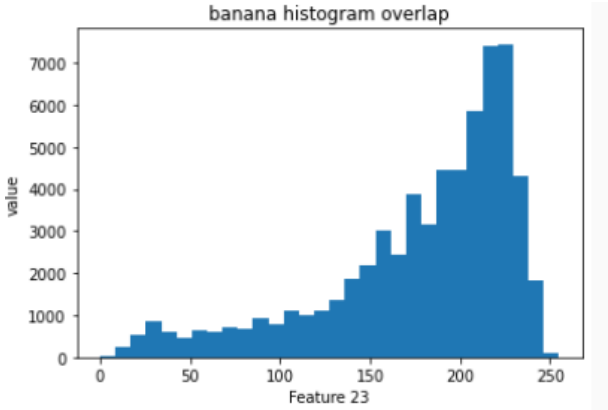


Fig. 29: Histogram of banana overlap

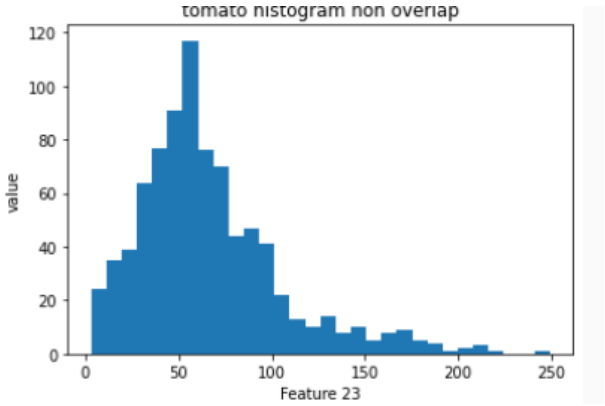


Fig. 30: Histogram of tomato non overlap

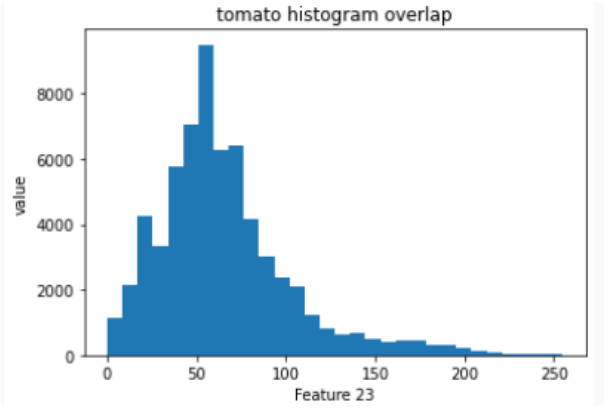


Fig. 31: Histogram of tomato overlap

it.

7.6 Answer the questions!

1 Data characteristics: Imbalanced, Incomplete, Inaccurate.

The data is balanced because the number of observations in all the three csv files are the same for both overlapping and non overlapping. As the number of class labels are the same for all so it is balanced. The data is not incomplete because it has no missing values in any csv files. The data is accurate as was it is generated properly and individually so it cannot be inaccurate.

2 Is it trivial data or possibly Big Data?

```
import matplotlib.pyplot as plt
import pandas as pd


#read the csv file
cant = pd.read_csv('C:/Users/Manish/Desktop/bigdata_assignment1/16_nonoverlap.csv')


f16 = cant['16']
f27 = cant['27']

plt.scatter(f16, f27, color =['green'] , s=1 )
plt.title("Feature Plot Cantaloupe")
plt.xlabel('Feature 16')
plt.ylabel('Feature 27')
plt.show()
```

Fig. 32: Scatter plot generation code

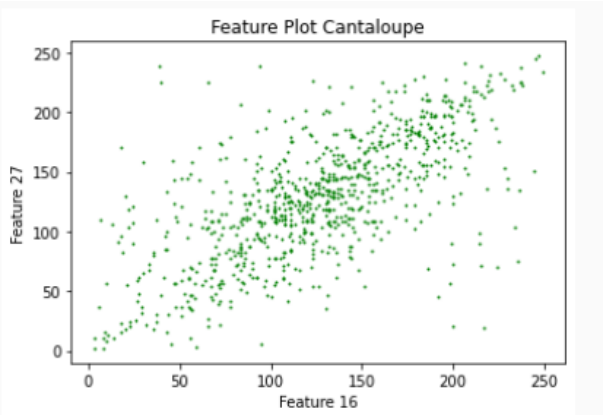


Fig. 33: Scatter plot for cantaloupe non overlap

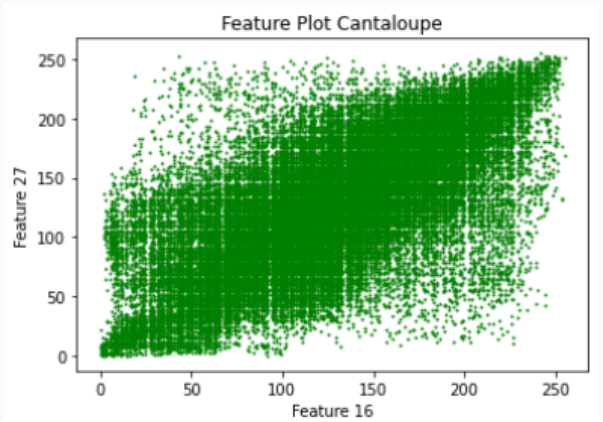


Fig. 34: Scatter plot for cantaloupe overlap

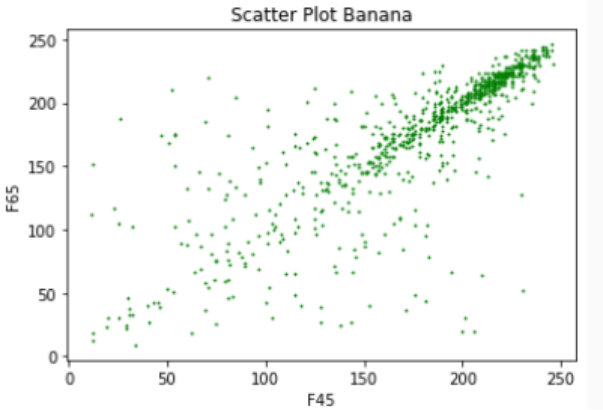


Fig. 35: Scatter plot for banana non overlap

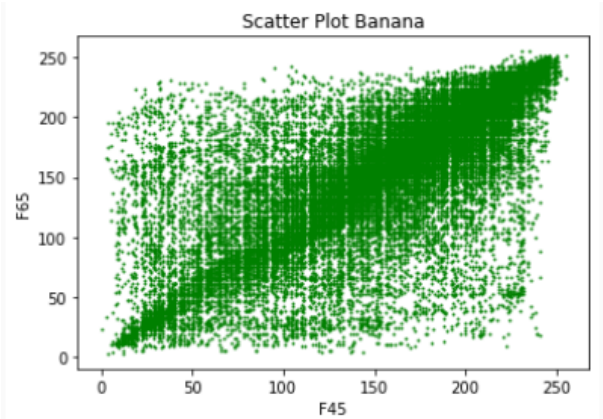


Fig. 36: Scatter plot for banana overlap

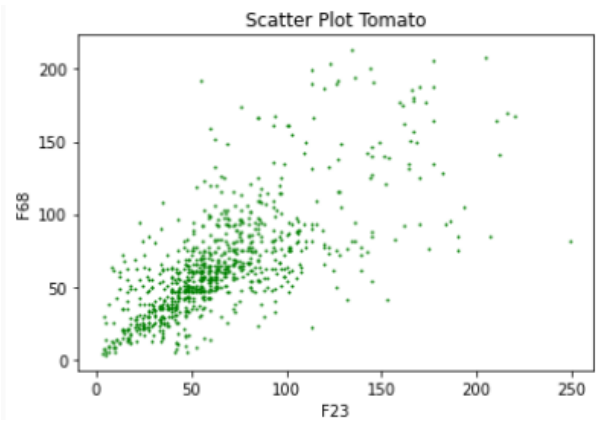


Fig. 37: Scatter plot for tomato non overlap

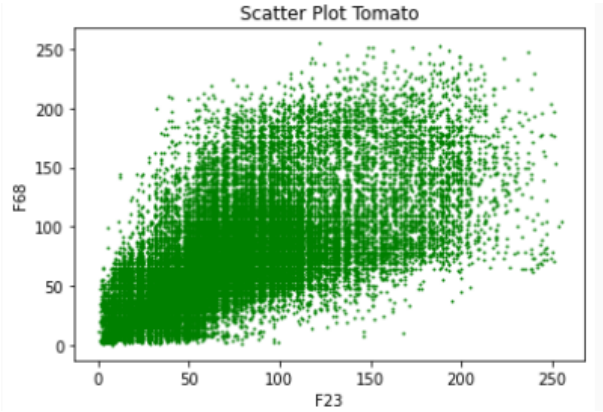


Fig. 38: Scatter plot for tomato overlap

```
import pandas as pd

cant = pd.read_csv('C:/Users/Manish/Desktop/bigdata_assignment1/16_nonoverlap.csv')
bana= pd.read_csv('C:/Users/Manish/Desktop/bigdata_assignment1/43_nonoverlap.csv')
toma= pd.read_csv('C:/Users/Manish/Desktop/bigdata_assignment1/5_nonoverlap.csv')

frames = [cant,bana]
merged = pd.concat(frames)

merged.to_csv('C:/Users/Manish/Desktop/bigdata_assignment1/image01.csv')
```

Fig. 39: Code to merge two files

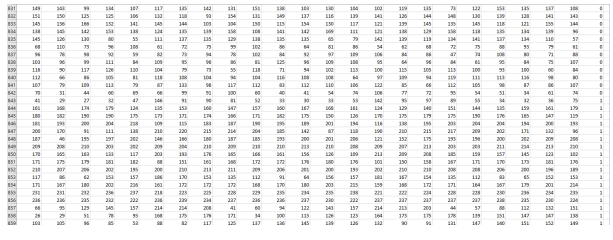


Fig. 40: image01.csv

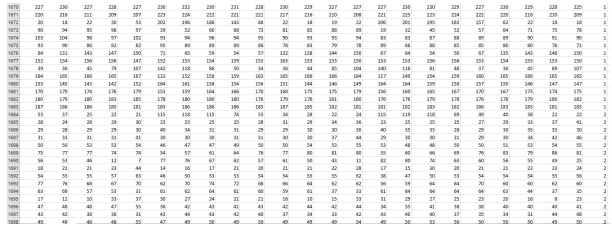


Fig. 41: image012.csv

The dataset generated is a trivial data and not big data as it is not exponentially increasing with time. It is trivial because for big data the number of classes should be many whereas in our case it's just three classes/labels. Secondly, the number of observations is more than the number of features so it is low dimensional data.

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

The data is not high dimensional because the number of observations are more than the number of rows. The data generated is not increasing with time so it doesn't have any scalability problem. As each feature vector has size 9*9 it has no scalability problem. If the number of feature increases then it becomes high dimensional and scalability problem might arise but right now it has no scalability problem.

4 Do you need to standardize? Do you need to normalize? How do they affect the data characteristics?

Yes, the data needs to be normalize and standardize. To have high accuracy all the values of the dataset needs to follow the same curve/data pattern. It is necessary for all the data points to fall in the same range and for that normalization and standardization is required. If the data is not standardize/normalized, then it would create issue while reading the data and will output wrong information. It has no effect on data characteristics.

VIII. Task 8: Construct a feature space!

In this part, I am making use of my files 16_nonoverlap.csv and 43_nonoverlap.csv which represents cantaloupe and banana files and use the concat function to merge the csv files and name the new created file as image01.csv. Secondly I use image01.csv to merge with 5_nonoverlap.csv and created a super merged file and name it as image012.csv. Lastly I use randomize function to randomize the placement of data in the image012.csv and generate a random dataset in it. The code to merge the datasets is shown in Figure 39, randomize code in Figure 42 and the generated dataset files are displayed after that.

IX. Task 9: Display subspaces!

9.1 Plot 2D feature space

For creating 2D scatter plot, I am using files 16_nonoverlap.csv (cantaloupe file), 43_nonoverlap.csv (banana file), 5_nonoverlap.csv (tomato file). I have selected features 45 and 75 for when combining cantaloupe and banana images, features 23 and 57 when combining banana and tomato images and features 14 and 69 when combining cantaloupe and tomato images. I am representing cantaloupe by blue, banana by yellow and tomato by red color in scatter plots. The code for 2D scatter plot is shown in Figure 44 and the scatter plots are displayed after that.

9.2 Plot 3D scatter plot

For plotting 3D scatter plot I have used image012_random.csv file, selected first 500 observations from it and selected features 11, 32, 59. I have represented cantaloupes, banana and tomato based on labels by 3 different colors. Figure 51 shows code for 3d feature space and the plot is shown after that. From Figure 52 it can be seen that the bananas are represented by blue color, cantaloupes by purple and tomatoes by yellow color. The bananas are skewed towards right, tomatoes towards left and cantaloupes are at the center and evenly spread on both sides too. From the plot

```
import pandas as pd
import numpy as np

image01 = pd.read_csv('C:/Users/Manish/Desktop/bigdata_assignment1/image01_overlap.csv')
toma= pd.read_csv('C:/Users/Manish/Desktop/bigdata_assignment1/overlap_5.csv')


df = [image01 , toma]
m1 = pd.concat(df)

indx = np.arange(len(m1))


rndm = np.random.permutation(indx)
rndm= m1.sample(frac=1).reset_index(drop=True)
rndm.to_csv('C:/Users/Manish/Desktop/bigdata_assignment1/image012_overlap_random.csv', index=False)
```

Fig. 42: Code to randomize placement

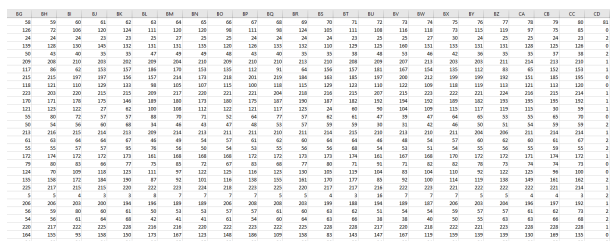


Fig. 43: random_012.csv

```
import matplotlib.pyplot as plt
import pandas as pd


#read the csv file
cant = pd.read_csv('C:/Users/Manish/Desktop/bigdata_assignment1/16_nonoverlap.csv')

bana = pd.read_csv('C:/Users/Manish/Desktop/bigdata_assignment1/43_nonoverlap.csv')


f45 = bana['45']
f75 = bana['75']

j45 = cant['45']
j75 = cant['75']


plt.scatter(f45, f75, color =['yellow'] , s=1 )
plt.scatter(j45, j75, color =['blue'] , s=1 )

plt.title("Scatter Plot Cantaloupe and Banana")
plt.xlabel('Feature 45')
plt.ylabel('Feature 75')
plt.show()
```

Fig. 44: Scatter plot 2D code

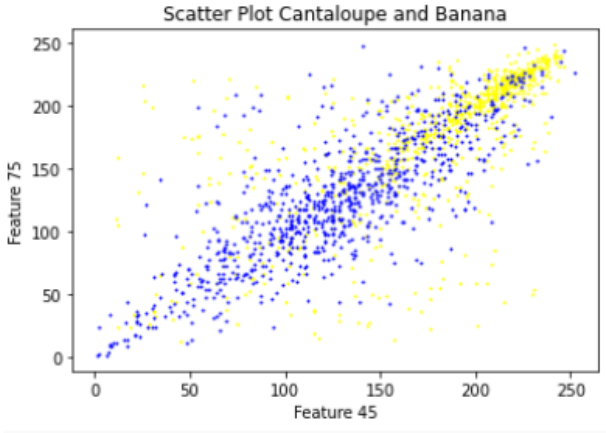


Fig. 45: 2d plot cant/bana

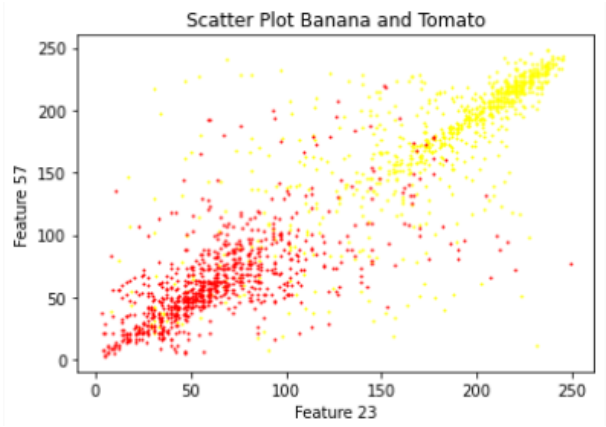


Fig. 46: 2d plot bana/toma

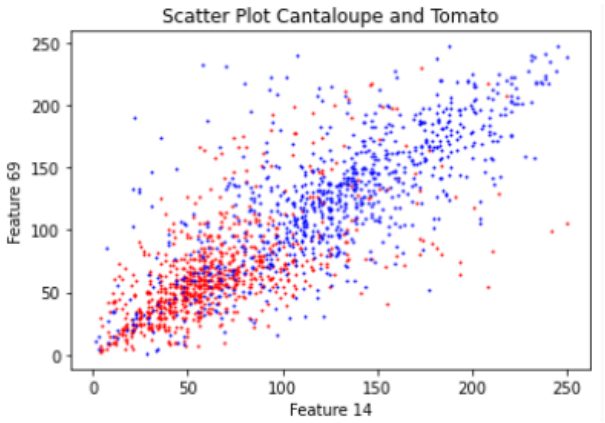


Fig. 47: 2d plot cant/toma

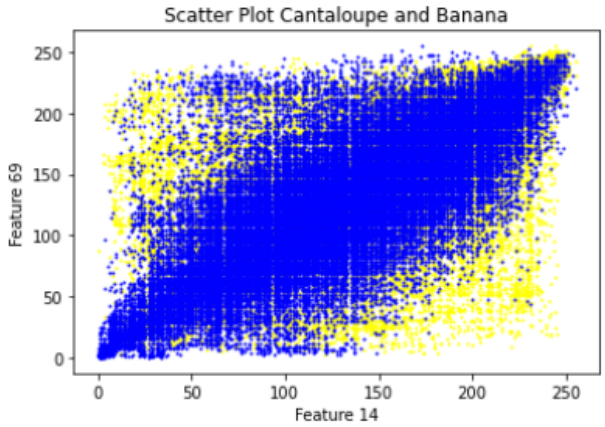


Fig. 48: 2d plot cant/bana overlapping

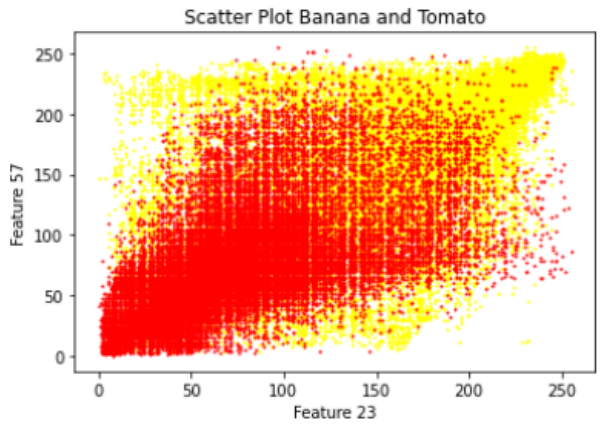


Fig. 49: 2d plot bana/toma overlapping

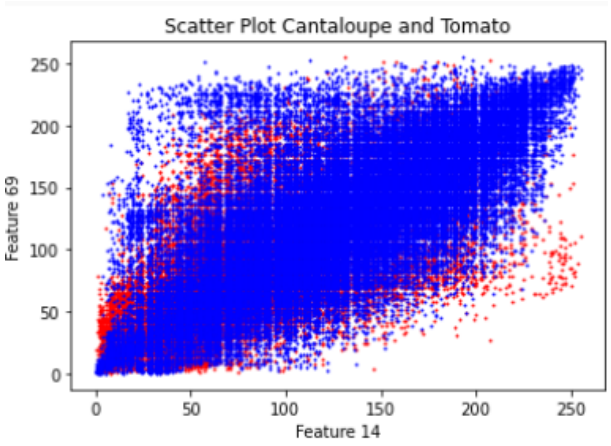


Fig. 50: 2d plot cant/toma overlapping

```
import pandas as pd
import cv2
import matplotlib.pyplot as plt
import numpy as np
from mpl_toolkits.mplot3d import Axes3D
ax = plt.axes(projection='3d')

II = pd.read_csv("C:/Users/Manish/Desktop/bigdata_assignment1/Image012_random.csv",header=None)
I1 = np.array(II)
NN = 500
ax.scatter3D(I1[1:NN,11], I1[1:NN,32], I1[1:NN,59], c=I1[1:NN,81])
plt.show()
```

Fig. 51: Code for 3d scatter plot

it seems the points are overlapping and it becomes difficult to classify them. But from some other angle it can be seen more clearly and maybe points are not overlapping and a plane can pass through them which can classify them.

9.3 Discuss these figures and describe your observations in terms of their separable features

From 2D scatter plot we can see that the bananas are skewed towards right, cantaloupes are spread at the center and tomatoes are skewed towards left. This can also be verify from the histogram. So it can become easy to classify based on their skewness. From 3D scatter plot we can see that points are overlapping over each other from the given angle. But maybe from other angle it can happen that the points are not overlapping over each other but are at different distances from each other, then it will become easy to classify them.

X. Task 10: Read multiple files and generate feature space!

For this task I have written a for loop which iterates through the folder files and apply feature space creation code to it and outputs feature space for all the images in individual csv files. The code can be seen in Figure 53 and the screenshot of output is after that.

XI. Task 11: Effects of block size on feature space dimensions and vector space

The block size that we are selecting in this assignment is 9*9. This results in creating a feature space of 81 dimensions and hence 81 features. If we reduce the block size to 8*8 then the number of vectors/observations increases but the dimension decreases. If we increase the block size to 10*10,

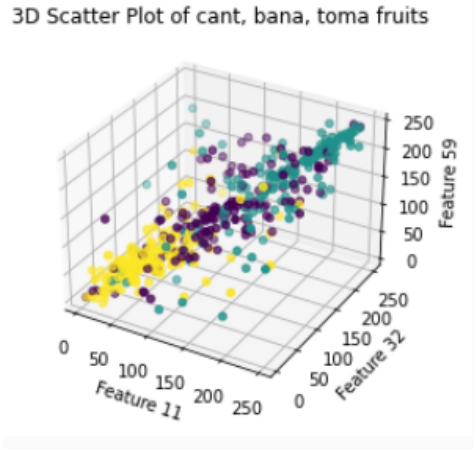


Fig. 52: 3d scatter plot for fruits

```
import cv2
import os
import numpy as np
import pandas as pd

std_height = 261

def new_width(heightcG,widthcG):
    aspect_ratio = widthcG/heightcG
    new_width = aspect_ratio*std_height
    return round(new_width)

def div9(height_fruit,width_fruit):
    new_fruit_width = new_width(height_fruit, width_fruit)
    rem = new_fruit_width % 9
    if rem == 0:
        return new_fruit_width
    elif rem >4:
        new_fruit_width = new_fruit_width + (9-rem)
    else:
        new_fruit_width = new_fruit_width-rem
    return new_fruit_width

def featurespace(img,filename):
    convertG = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    heightcG, widthcG = convertG.shape
    new_fruit_width = div9(heightcG,widthcG)
    resized_fruit = cv2.resize(convertG, dsize=(new_fruit_width, std_height), interpolation=cv2.INTER_CUBIC)
    heightc, widthc = resized_fruit.shape
    cc = round((heightc)*(widthc)/81)
    flatc = np.full((cc, 82), 0)
    k = 0
    for i in range(0,heightc,9):
        for j in range(0,widthc,9):
            crop_tmpi = resized_fruit[i:i+9,j:j+9]
            flatc[k,0:81] = crop_tmpi.flatten()
            k = k + 1
    fspacec = pd.DataFrame(flatc)
    #panda object
    fspacec.to_csv(filename, index=False)


location = 'H:/UNCG SEMS/Sem 2 (Spring 22)/CSCE610 ML/assignment/assignment 1/datasets/assignment 1/FIDS30/apricots'
images = []
for image in os.listdir(location):
    img = cv2.imread(os.path.join(location,image))
    if img is not None:
        images.append(img)
        splitted_filename = image.split('.')
        rename_csvfile = 'Apricots'+splitted_filename[0]+'.csv'
        featurespace(img,rename_csvfile)
```

Fig. 53: Code to read multiple files and generate feature space

then the number of feature increases which can lead to high dimensionality issues.

XII. Assignment 2: Feature Space to a Classifier

XIII. Task 12:Complete and extend the tasks of assignment 1

12.1 Divide the data domain of the datasets into 78:22 and save corresponding files

For this task, I have divided the overlapping and non overlapping files for two class and three class generated in assignment1 into 78% of training and 22% of testing data using the shape function of python. The code for the task is shown is Figure 55.

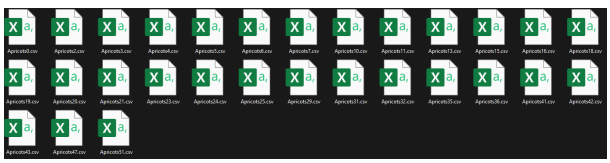
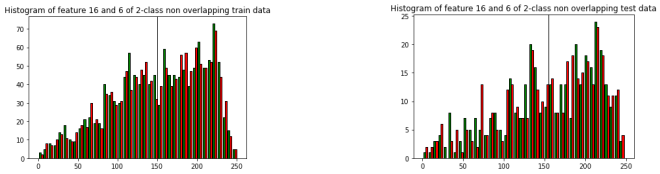


Fig. 54: Creation of multiple feature spaces

```
import pandas as pd
import numpy as np

# Read a feature space
input_data = pd.read_csv("C:/Users/Manish/Desktop/data/overlapping/merge/random/image012_overlap_random.csv" , header

X = input_data
row, col = X.shape

#Split the data into 78% - Training Dataset
TR = round(row*0.78)
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,:]

#savetTraining dataset
data_train = pd.DataFrame(X_train)
data_train.to_csv('C:/Users/Manish/Desktop/splitted data/train/image012_train_overlap.csv', index=False)

#save testing Dataset
data_test = pd.DataFrame(X_test)
data_test.to_csv('C:/Users/Manish/Desktop/splitted data/test/image012_test_overlap.csv', index=False)
```

Fig. 55: Code to split data into 78:22



(a) 01 *train* (b) 01 *test*

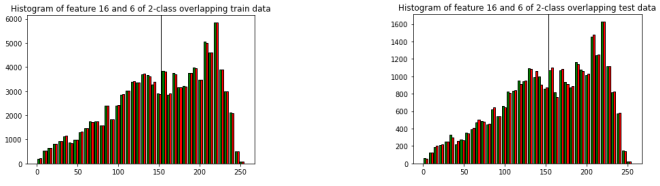
Fig. 56: Histogram of feature 16 & 6 of 2-class non overlap train/test data

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

I have selected two features: feature16 and feature6 and plot histograms for training and test dataset generated in the previous steps. From the histograms in Figure 56 to Figure 59, it is clear that both train and test data follow the same distribution for overlapping feature space. For non overlapping data, train data don't have much difference among features 6 and 16 but for test data there happens to be a considerable difference between the features because of small data size. The information about the mean can also be seen in the histograms shown by the solid dark line.

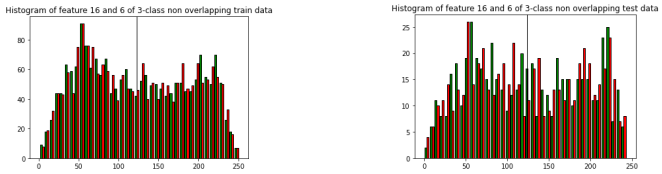
The mean for 2 class non overlapping train data is around 150.53, but the mean of 2 class non overlapping test data is around 154.84. Similarly for the 3 class, non overlapping train data is around 122.93, but the mean of 2 class non overlapping test data is around 123.72. So the test data is bit skew to the left side than the train data.

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



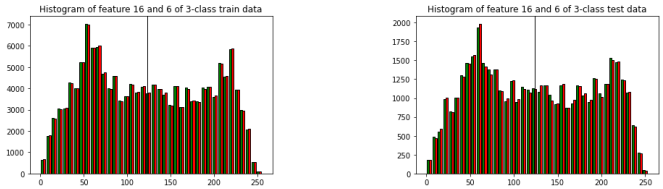
(a) 01 *train* (b) 01 *test*

Fig. 57: Histogram of feature 16 & 6 of 2-class overlap train/test data



(a) 012 *train* (b) 012 *test*

Fig. 58: Histogram of feature 16 & 6 of 3-class non overlap train/test data



(a) 012 *train* (b) 012 *test*

Fig. 59: Histogram of feature 16 & 6 of 3-class overlapping train/test data

The same features: feature16 and feature6 is selected for scatter plots. Figure 60 shows the code and Figure 61 to Figure 68 shows the scatter plots for the same.

XIV. Task 13: Implementing a regression-based model

13.1 Implement and train lasso regression or elastic-net regression as a two-class classifier using the training sets of the overlapping and non-overlapping feature vectors.

I have implemented lasso regression model for this task. I have calculated the parameter 'a' and the model 'y' of the lasso regression using the formula provided in the textbook. I have used lambda as 0.1 for calculation. I trained the model with the training data files for 2 class classification. Then predict the test data on the model. The code is shown in the Figure 69.

13.2 Add the predicted labels next to their actual labels in the corresponding spreadsheets.

In this step, the predicted values are merged with the test data actual label values as 83rd column in a new csv file. Figure 70 shows the added column.

```
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
df = pd.read_csv('C:/Users/Manish/Desktop/splitted data/train/image012_train_overlap.csv')
df = df.head(2000)
f6 = df['6']
f16 = df['16']
label = df['81']


fig = plt.figure(figsize = (20 , 20))
ax1 = fig.add_subplot(221)
ax1.set_title("image012_train dataset overlapping")
ax1.set_xlabel("Feature 6")
ax1.set_ylabel("Feature 16")
sns.scatterplot(x = f6 , y= f16 , hue= label)
plt.show()
```

Fig. 60: Code for scatter plot for train/test data

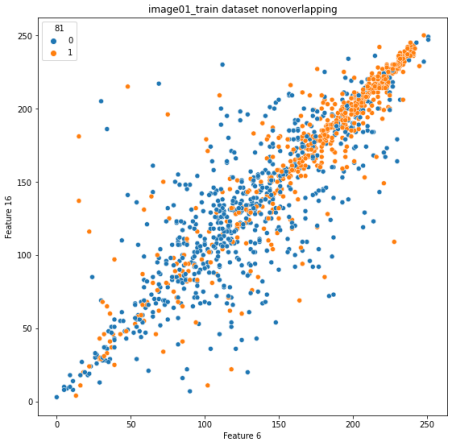


Fig. 61: Scatter plot 2-class train data non overlap

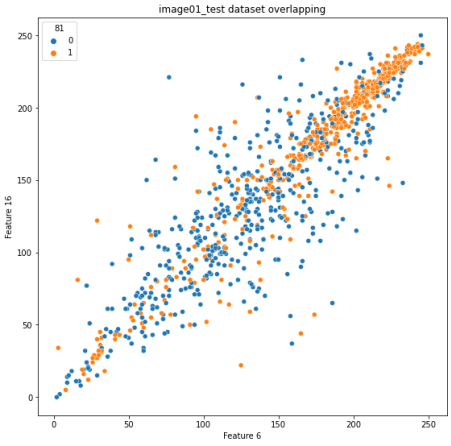


Fig. 64: Scatter plot 2-class test data overlap

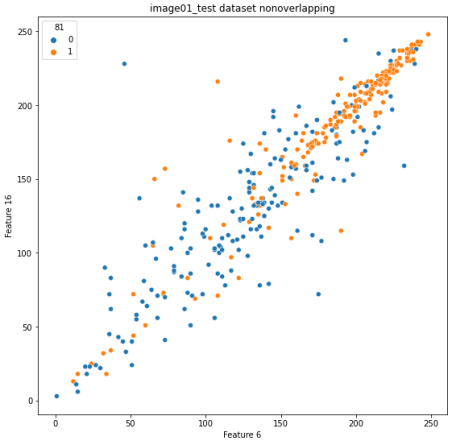


Fig. 62: Scatter plot 2-class test data non overlap

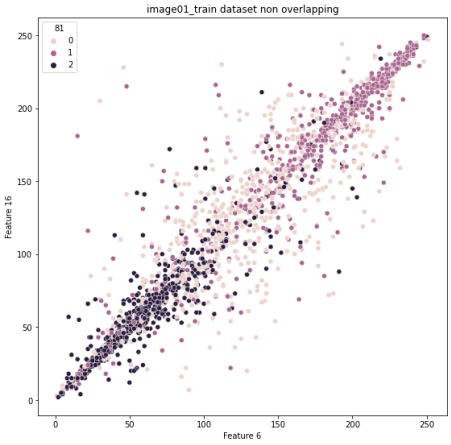


Fig. 65: Scatter plot 3-class train data non overlap

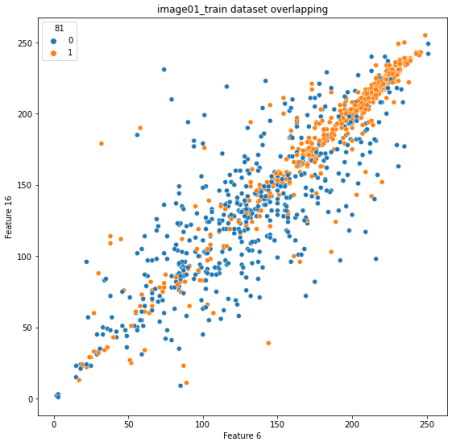


Fig. 63: Scatter plot 2-class train data overlap

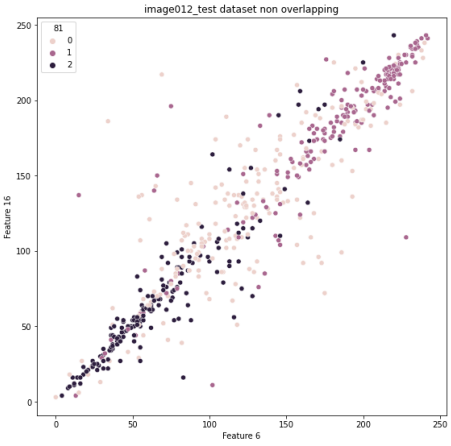


Fig. 66: Scatter plot 3-class test data non overlap

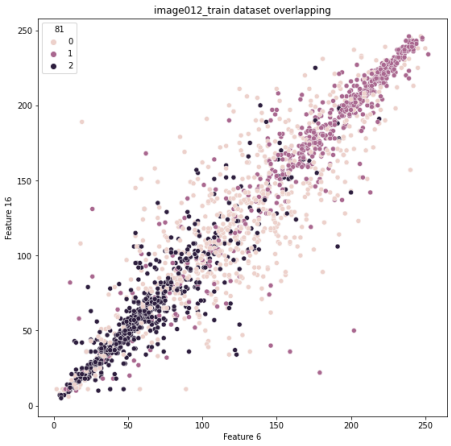


Fig. 67: Scatter plot 3-class train data overlap

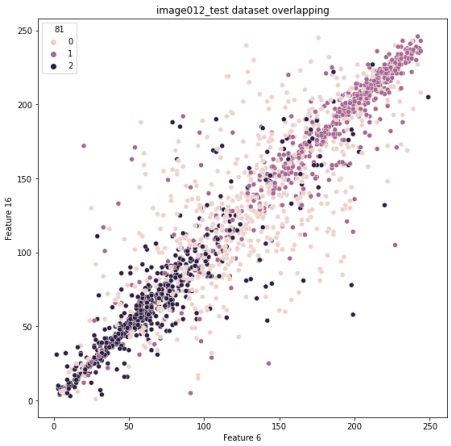


Fig. 68: Scatter plot 3-class test data overlap

```python
import pandas as pd
import numpy as np
from numpy.linalg import inv
from sklearn.metrics import confusion_matrix

#read the training and the testing dataset
input_data_train = pd.read_csv('C:/Users/Manish/Desktop/splitted data/train/image01_train_overlap.csv' , header =
input_data_test = pd.read_csv('C:/Users/Manish/Desktop/splitted data/test/image01_test_overlap.csv', header=None)


#extract the labels column
y = input_data_train[81]
Y = np.array(y)

#extract festures data
input_data_train.drop(81,axis=1,inplace=True)
X = input_data_train
X1 = np.array(X)

#set lambda as 0.1
lamda = 0.1

# Calculate X transpose
X2 = X1.transpose()

#Perform xx'
XX = np.matmul( X2 , X1)

#Calculating inverse
IX = inv(XX)

#Calculate yx'
YX = np.matmul(Y,X1)
```

Fig. 69: Code for Lasso Regression

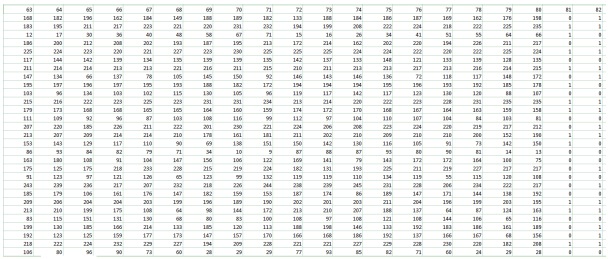


Fig. 70: Added predicted label next to actual label in test data

```python
#confusion matrix building
CC_test = confusion_matrix(test_y, y2)

TN = CC_test[0,0]
FP = CC_test[0,1]
FN = CC_test[1,0]
TP = CC_test[1,1]
FPFN = FP+FN
TPTN = TP+TN

Accuracy = 1/(1+(FPFN/TPTN))
print("Our_Accuracy_Score:",Accuracy)

Precision = 1/(1+(FP/TP))
print("Our_Precision_Score:",Precision)

Sensitivity = 1/(1+(FN/TP))
print("Our_Sensitivity_Score:",Sensitivity)

Specificity = 1/(1+(FP/TN))
print("Our_Specificity_Score:",Specificity)
```

Fig. 71: Confusion matrix creation

13.2 Construct confusion matrices using the actual and predicted labels in the 82nd and 83rd columns of the testing datasets.

The code for constructing the confusion matrix using the actual and predicted labels can be seen in the Figure 71. The result of confusion matrix can be seen in Figure 72 - Figure 73.

13.3 Select two qualitative measures that use the concept of confusion matrix to quantify the performance of a machine learning model.

The qualitative measures accuracy, precision, specificity,

```
In [92]: CC_test
Out[92]:
array([[ 9140,  4984],
       [ 2744, 11296]], dtype=int64)
```

Fig. 72: Output of confusion matrix for overlapping data

```
In [100]: CC_test
Out[100]:
array([[116,  64],
       [ 56, 134]], dtype=int64)
```

Fig. 73: Output of confusion matrix for non overlapping data

```
Our_Accuracy_Score: 0.7256071580741372
Our_Precision_Score: 0.6938574938574938
Our_Sensitivity_Score: 0.8045584045584045
Our_Specificity_Score: 0.6471254602095724
```

Fig. 74: Output of confusion matrix for image01 overlapping dataset

```
import pandas as pd
import numpy as np
from sklearn.metrics import confusion_matrix,plot_confusion_matrix
from sklearn.metrics import accuracy_score,precision_score,recall_score
from sklearn.ensemble import RandomForestClassifier
from sklearn import tree
import matplotlib.pyplot as plt

input_data_train = pd.read_csv('C:/Users/Manish/Desktop/splitted data/train/image01_train.csv' , header = None)
input_data_test = pd.read_csv('C:/Users/Manish/Desktop/splitted data/test/image01_test.csv' , header = None)

#spliting the labels
y = input_data_train[81]
Y = np.array(y)
#splitting the features
input_data_train.drop(81,axis=1,inplace=True)
X = input_data_train
X1 = np.array(X)

# Train the model
rF = RandomForestClassifier(random_state=0,n_estimators=100,oob_score=True, n_jobs=1)
rF = rF.fit(X1, Y)

#saving the test dataset to peek
peek = pd.DataFrame(input_data_test)

#Testing the model using Trained results0
test_y = input_data_test[81]
Y_test = np.array(test_y)

#splitting the features from the test dataset to test the model
input_data_test.drop(81,axis=1,inplace=True)
input_data_test_np = np.array(input_data_test)
y_pred = rF.predict(input_data_test_np)
yhat_saved = pd.DataFrame(y_pred)

peek['82'] = yhat_saved
peek.to_csv('C:/Users/Manish/Desktop/RF_Test_result_image01_nonoverlap.csv', index=False)
```

Fig. 75: Random forest implementation for overlapping and non overlapping dataset

sensitivity are there for confusion matrix. I am considering accuracy and precision for this task. Figure 74 shows the qualitative measures result.

From the figure it can be seen that image01 overlapping data have accuracy of 72.5% and precision of 69.38%. While for non overlapping data, accuracy is 67.5% and precision is 67.6%. So, in our case overlapping data performed better than non overlapping data because more data points are available for it to train.

XV. Task 14:Implementing random forest or deep learning

14.1 Implement the random forest or the sequential (simple deep learning) technique as two-class and three-class classifiers using the training sets of the overlapping and non-overlapping feature vectors that you generated.

I have implemented random forest classifier for this task. It is built as a 2-class and a 3-class classifier. For this, I have first split the training dataset into features and labels and gave it as an input to the classifier to train. After training, I split the test set features and gave it to the model to predict the labels. The code for Random forest can be seen in Figure 75.

14.2 Apply the trained model to the test sets of all the categories of datasets and add the predicted labels next to their actual labels in the corresponding spreadsheets.

After predicting the labels using the step discussed above, the labels are stored into a new csv file by converting into a data frame and saving the data frame as csv file. The output

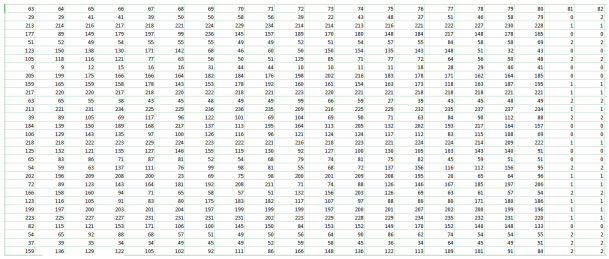


Fig. 76: Predicted labels added in 83rd column for overlapping dataset

```
#creating confusion matrix
CC_test = confusion_matrix(test_y, y_pred)
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

print(CC_test)
plot_confusion_matrix(rF, input_data_test_np, test_y)
plt.show()

#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)
```

Fig. 77: Confusion matrix for Random Forest

of the csv file is as shown in the Figure 76. The output of the random forest classifier is saved in a pdf file.

14.3 Construct confusion matrices using the actual and predicted label in the 82nd and 83rd columns in the testing datasets for all the four categories of datasets.

Creation of confusion matrix is as shown in Figure 77. The output of the confusion matrix for 2class and 3class of overlapping and non overlapping data is shown from Figure 78 to Figure 81. From the output of the confusion matrix it is clear that the random forest is predicting good results because the diagonal elements have higher values compared to other elements of the matrix.

Compared to non overlapping, overlapping dataset gives better results. It is because the model has more points to train before testing.

14.4 Adapt the same two measures used in Task 2 to quantify the performance of the models.

After predicting the labels and giving it as input to confusion matrix, I have selected the accuracy and precision as my qualitative measures to quantify the performance of model.

From Figure 82 and Figure 83 it is clear that 3-class have better accuracy and precision as compared to 2-class. For 2-class non overlapping, accuracy is 89% and precision is 89.26%. For 3-class non overlapping, accuracy is 90.71% and precision is 91.55%.

For 2-class overlapping, accuracy is 98.75% and precision is 98.32%. For 3-class overlapping, accuracy is 98.40% and

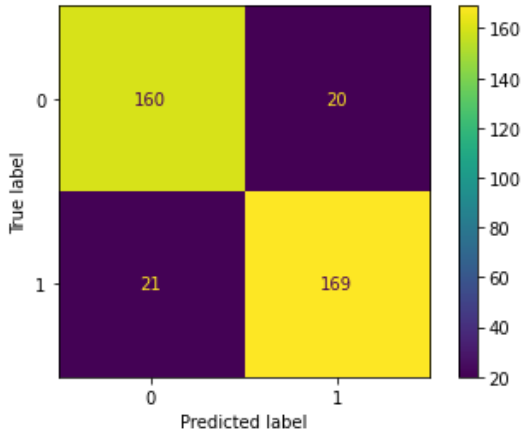


Fig. 78: Confusion matrix for 2class non overlap

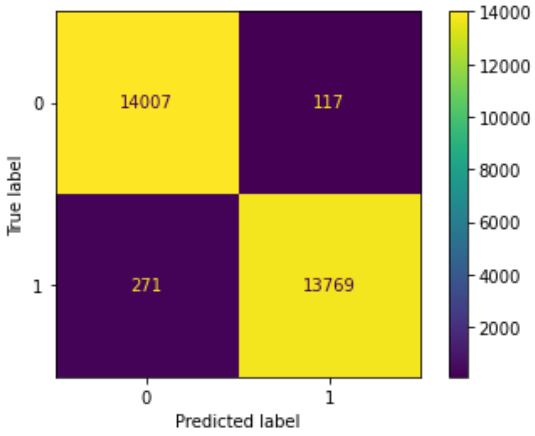


Fig. 79: Confusion matrix for 2class overlap

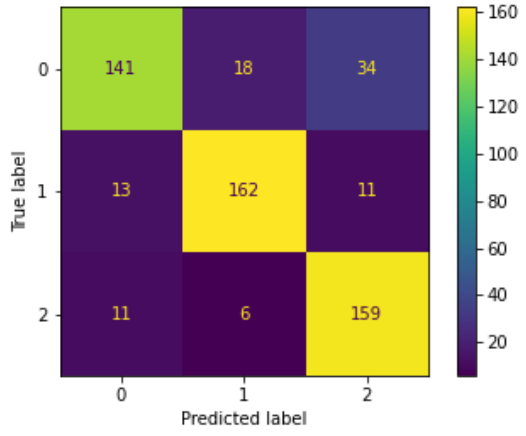


Fig. 80: Confusion matrix for 3class non overlap

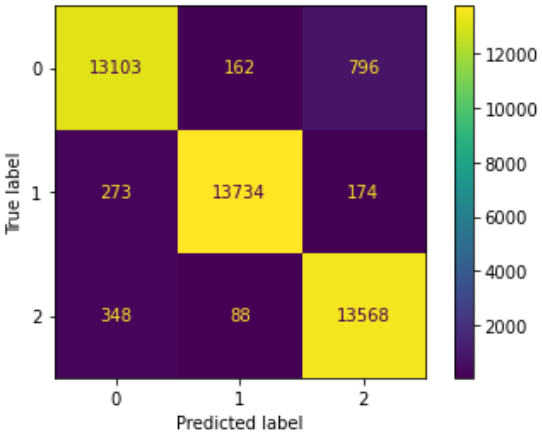


Fig. 81: Confusion matrix for 3class overlap

```
Test_Accuracy_Score: 0.8891891891891891
Test_Precision_Score: 0.8926553672316384
Test_Sensitivity_Score: 0.8777777777777778
Test_Specificity_Score: 0.8999999999999999
```

Fig. 82: Random Forest Measure of 2 class non overlapping dataset

precision is 98%. So it seems clear that overlapping data performs better with testing as compared to non overlapping data.

XVI. Task 15:Evaluation of the learning models

15.1 Find some built-in measures that are available in software libraries like metrics.accuracy_score in Python's sklearn environment. Use such two measures to compare the performance of the models.

For this task, I have used sklearn.metrics for Accuracy, Precision, Sensitivity as accuracy_score, precision_score. These are built in measures for calculation. Figure 86 shows inbuilt measure performance for lasso regression for both overlapping and non overlapping dataset and Figure 87 - Figure 87 shows Random Forest qualitative measures.

```
Test_Accuracy_Score: 0.907185628742515
Test_Precision_Score: 0.9155844155844156
Test_Sensitivity_Score: 0.8867924528301887
Test_Specificity_Score: 0.9257142857142858
```

Fig. 83: Random Forest Measure of 3 class non overlapping dataset

```
Test_Accuracy_Score: 0.9875372816361312
Test_Precision_Score: 0.9832292470703811
Test_Sensitivity_Score: 0.9920702350608893
Test_Specificity_Score: 0.9829772079772079
```

Fig. 84: Random Forest Measure of 2 class overlapping dataset

```
Test_Accuracy_Score: 0.984049574655324
Test_Precision_Score: 0.9795903110047848
Test_Sensitivity_Score: 0.9877874104787033
Test_Specificity_Score: 0.9805097451274363
```

Fig. 85: Random Forest Measure of 3 class overlapping dataset

```
Overlapping measures Lasso Regression
BuiltIn_Accuracy: 0.7341996875443829
BuiltIn_Precision: 0.7224107506447672
BuiltIn_Sensitivity (recall): 0.7581196581196581

In [128]: print("Nonoverlapping measures Lasso Regr
     ...: from sklearn import metrics
     ...: print("BuiltIn_Accuracy:",metrics.accurac
     ...: print("BuiltIn_Precision:",metrics.preci
     ...: print("BuiltIn_Sensitivity (recall):",met
Nonoverlapping measures Lasso Regression
BuiltIn_Accuracy: 0.6756756756756757
BuiltIn_Precision: 0.6767676767676768
BuiltIn_Sensitivity (recall): 0.7052631578947368
```

Fig. 86: Lasso Regression builtin measures for overlapping and non overlapping dataset

15.2 Describe the quantitative differences between the built-in measures and the confusion-matrix-based measures that you used in the analysis to compare the models with the four categories of datasets.

For this task, I have created a table as shown in Figure 89 with values for built-in measures and measures derived from confusion matrix. From the table it is clear that there is not much difference between the values. While confusion matrix measures are slightly higher than inbuilt measures in case of 3-class classification than the built-in measures.

15.3 Compare all the results (qualitative measures) and determine which pair (a model and a category of feature vectors) is superior among the ones that you considered. Use your data and findings to support this result.

```
Random Forest 2 class overlap output
sklearn.metrics Accuracy 0.9875372816361312
sklearn.metrics precision 0.9919499748436714
sklearn.metrics sensitivity 0.982977207977208
Random Forest 2 class non overlap output
sklearn.metrics Accuracy 0.8891891891891892
sklearn.metrics precision 0.8860103626943006
sklearn.metrics sensitivity 0.9
```

Fig. 87: Random Forest builtin measures for 2-class overlapping and non overlapping data

```
Random Forest 3 class non overlap output
sklearn.metrics Accuracy 0.841726618705036
sklearn.metrics precision 0.882520205490658
sklearn.metrics sensitivity 0.8832902187511079
Random Forest 3 class overlap output
sklearn.metrics Accuracy 0.9587426326129665
sklearn.metrics precision 0.9692900112056182
sklearn.metrics sensitivity 0.9690416080012648
```

Fig. 88: Random Forest builtin measures for 3-class overlapping and non overlapping data

		RANDOM FOREST			
		Buit-in measures		Confusion matrix	
		Accuracy	Precision	Accuracy	Precision
012 overlap		95.68	96.78	98.48	98.05
012 nonoverlap		84.53	88.52	90.53	90.5
01 overlap		98.62	98.62	98.62	98.1
01 non overlap		88.91	88.93	88.91	89.26
		Logistic Regression			
		Buit-in measures		Confusion matrix	
		Accuracy	Precision	Accuracy	Precision
012 overlap		45.88	23.13	68.63	63.29
012 nonoverlap		48.2	24.46	70.71	65.43
01 overlap		73.41	72.24	73.41	72.24
01 non overlap		67.56	67.67	67.56	67.67

Fig. 89: Output of builtin measures and confusion matrix based measures for RF and LR

In my opinion Random Forest performs better than Logistic Regression for all category of feature vectors. Random Forest performed better in overlapping feature space then non overlapping feature space. For 2-class overlapping dataset, it gives 98.62% accuracy while for 3-class overlapping dataset, it gives 98.48 % of accuracy. So random forest for overlapping dataset is the model-category pair which is superior among all. The table in Figure 89 is evident of that.

XVII. Assignment 3: Classifiers on a Conceptualized Big Data System

XVIII. Task 16: Making sure all the tasks of assignment 1 and assignment 2 are completed

16.1 Complete all the tasks stated in assignment 1 towards developing a set of suitable feature spaces

I have completed all the tasks assigned in assignment 1. I have generated feature spaces for overlapping and non overlapping feature vector for 2-class and 3-class classification.

16.2 Complete all the tasks stated in assignment 2 that perform the application of Random Forest and Deep Learning algorithm using the feature spaces

For this task, I have make use of Random Forest and Deep learning method on the feature spaces generated in assignment 1. I have used these algorithms on both overlapping and non overlapping feature spaces and calculate the time and accuracy, suggest which works best on 3-class and 2-class classification for both overlapping and non overlapping blocks.

16.3 Running dataset on conceptualized big data system and a local system, find the differences

For this part, I am planning to generate the feature spaces that are large enough to execute on local machine but can be managed on big data system. I am planning to generate a feature space with around 1 million data points. After executing, I will be calculating the time and accuracy of the algorithm implemented in both the systems.

XIX. Task 17: Understanding and the 10-fold validation technique presented in Chapter 8 of the textbook

17.1 Read chapter 8 and explain the concept of n-fold validation in your own words. You must clearly explain the effect of different values of n on seen-data and unseen-data problems.

n-fold validation technique is a block based technique, where the dataset is divided into n equal parts. Each part is disjoint from each other. Let's try with n = 10; which is 10 fold cross validation technique.

In 10 fold cross validation technique, initially the model is fed with all the blocks except the 10^{th} block. The 10^{th} block is used for testing. In the next step, all the blocks are used for training except the 9^{th} block. It is used for testing the model.

As we can see, the 9^{th} block was seen in the first step, but the trained model in the second step didn't see 9^{th} block. As a result the testing is done on both seen and unseen data. This is known as cross validation. This process is repeated for all other blocks and accuracy is calculated, take the average of it and we get the final accuracy. Based on accuracy, one final model is selected or retraining is done.

With different values of n, the accuracy and error in each step will change. For large value of n, we will get better accuracy because our data will get more points to train. But optimal data size for training is also to be determined to avoid over fitting problem. Along with the value of number of blocks, the size of blocks also matters. If we divide a small data into many blocks, each block contains very few data points. If we train and validate it on such blocks, it won't be accurate. So size of block also matters. We need to define an appropriate values of the size of blocks.

In our case, we will be dividing the data into unequal block sizes and performing training and testing on it.

17.2 Now connect your data set to the 10-fold validation concept that is explained in Figure 8.6 by identifying the number of observations in each fold and the statistical differences between each fold.

I have generated my dataset using 2 types of fruit images: apples and bananas. I have selected 5 images of each fruit type, so in total 10 images. I then divided my data into 10 blocks and apply the 10 fold validation technique that we discussed above on local machine and on big data system.

For dividing the data, I have plotted the histogram of a feature to randomly divide the data into blocks. The detailed information is explained in further steps.

17.3 Make sure to draw your conceptualized big data system and write down the data characteristics of a subdomain that is delivered to a node in the conceptualized cluster.

Our big data system contains 9 nodes. Each node is feed with training and test data. Training data contains 8 blocks while test data contains 1 block. The size of each block is unique.

Each block contains data points within a certain range. This is the data characteristic of each block that is feed. The

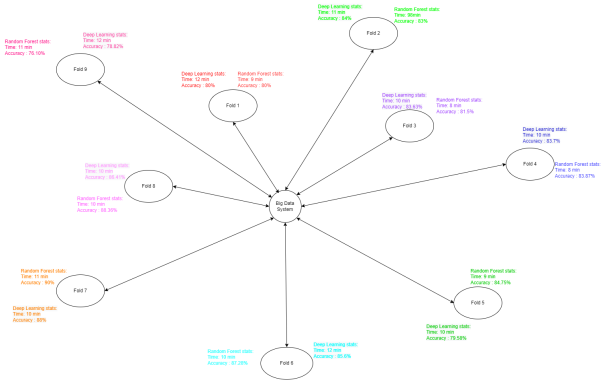


Fig. 90: Conceptualized Big Data System

first block contains data points within the range of (0,25.5), second block contains in range of (25.5,51). The third block contains in the range of (51,76.5), fourth block in the range of (76.5,102), fifth block in the range of (102,127.5), sixth block in the range of (127.5,153). The seventh block in contains in the range of (153,178.5), eighth block in the range of (178.5,204), ninth block within the range of (204,255).

The diagram of our conceptualized big data system can be seen in Figure 90. Our big data system has one main system and 9 sub nodes. These sub nodes are basically the machines on which we will run each folds simultaneously. Figure 90 shows the big data system when run for Random Forest and Deep Learning model. The accuracy and time taken for each model is mentioned on the nodes. The final time taken and accuracy using each technique is calculated by taking mean of time and mean of accuracies. It is an example of Map-Reduce where data is divided among the nine clusters so processing of big data becomes efficient. The key value pair for map reduce is mentioned as: key is the mean of the range of each bin and the value is the data points contained in that bin of histogram.

The key value pair for a block is defined as: the key contains the mean of range of bin. For example, out first key will be (0+25.5)/2 = 12.75. For second block it will be (25.5 + 51)/2 = 38.25. Similarly is the method for all other blocks/splits. For the value contained within the blocks, it contains all the rows that fall within that range. For example, block_1 contains the points whose feature23 values lie within the range of 0 to 25.5. Such is the key-value pair for all other blocks/splits.

17.3 Then create a data frame (or a feature space) for each node and save it as a spreadsheet (.csv file). Do the same for every node and save the spreadsheets for performing Task 3.

For this task, I have merged the data frames to create a big csv files with block-1 to block-8 in it and block-9 is in the testing. I did the same for all other blocks, keeping one block in the testing and combining all the remaining blocks. Then I fed the nodes with that train and test data.

First node is fed with data that contains block-9 as testing and remaining blocks in the training. Second node is fed with

```
min_point = min(input_data['23'])
max_point = max(input_data['23'])
print(f"The min value of feature 23 is {min_point} and max value of feature 23 is {max_point}")

plt.hist(f, nbins,ec='white')
plt.title("Total data")
plt.xlabel('Feature 23')
plt.ylabel('frequency')
plt.show()

first_block=pd.DataFrame()
first_block = input_data[f<=25.5]
first_block.to_csv('C:/Users/Manish/Desktop/csvs/10 fold/first_block.csv', index= False)

second_block=pd.DataFrame()
second_block = input_data[f.between(25.5,51)]
second_block.to_csv('C:/Users/Manish/Desktop/csvs/10 fold/second_block.csv', index= False)

third_block=pd.DataFrame()
third_block = input_data[f.between(51,76.5)]
third_block.to_csv('C:/Users/Manish/Desktop/csvs/10 fold/third_block.csv', index= False)

fourth_block=pd.DataFrame()
fourth_block = input_data[f.between(76.5,102)]
fourth_block.to_csv('C:/Users/Manish/Desktop/csvs/10 fold/fourth_block.csv', index= False)

fifth_block=pd.DataFrame()
fifth_block = input_data[f.between(102,127.5)]
fifth_block.to_csv('C:/Users/Manish/Desktop/csvs/10 fold/fifth_block.csv', index= False)

sixth_block=pd.DataFrame()
sixth_block = input_data[f.between(127.5,153)]
sixth_block.to_csv('C:/Users/Manish/Desktop/csvs/10 fold/sixth_block.csv', index= False)

seventh_block=pd.DataFrame()
seventh_block = input_data[f.between(153,178.5)]
seventh_block.to_csv('C:/Users/Manish/Desktop/csvs/10 fold/seventh_block.csv', index= False)

eigth_block=pd.DataFrame()
eigth_block = input_data[f.between(178.5,204)]
eigth_block.to_csv('C:/Users/Manish/Desktop/csvs/10 fold/eigth_block.csv', index= False)
```

Fig. 91: 10 folds data split

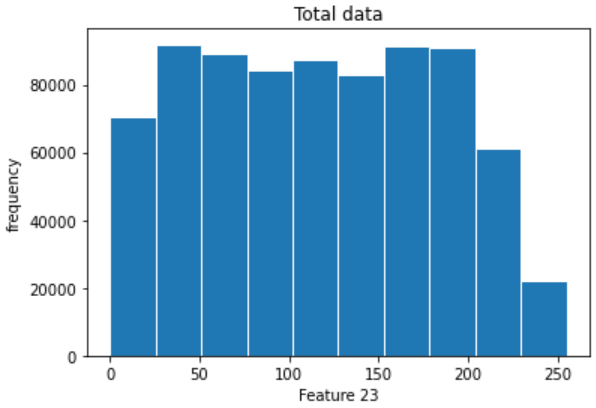


Fig. 92: Histogram of feature 23

data that contains block-8 as testing and remaining blocks in the training. Third node is fed with data that contains block-7 as testing and remaining blocks in the training. Fourth node is fed with data that contains block-6 as testing and remaining blocks in the training.

Fifth node is fed with data that contains block-5 as testing and remaining blocks in the training. Sixth node is fed with data that contains block-4 as testing and remaining blocks in the training. Seventh node is fed with data that contains block-3 as testing and remaining blocks in the training. Eighth node is fed with data that contains block-2 as testing and remaining blocks in the training. Ninth node is fed with data that contains block-1 as testing and remaining blocks in the training. All these nodes are fed with data at the same time and performing parallelization. The accuracy and time duration is calculated. The average of accuracy is done and we get the final accuracy.

There are 9 files for training and nine files for testing. It can be seen in the data folder.

XX. Task 18: Machine Learning Tasks

18.1 Prepare a dataset with 2-class classification with non overlapping feature space. Then divide the data to perform 10 fold cross validation.

For this step, I have used 10 images in total: 5 images of apples and 5 images of banana. I have generated the data using overlapping feature vectors and have generated the combined data with more than 700,000 data points.

Further to generate 10 blocks of data, I have selected one feature: Feature 23 and calculated the min() and max() of the feature 23 to get the range of it. After I have got the range I have plotted the histogram of it with bins = 10. By doing this, I get 10 blocks of data. But the tenth block was of size much smaller than other blocks. So combined the block-9 and block-10. The code for this is shown in Figure 91. The histogram can be seen in the Figure 92. From the histogram it is clear that the size of block-10 is much smaller. So I appended block-10 with block-9 and finally get 9 blocks of data with comparative size of blocks.

18.2 Now apply random forest and deep learning to the entire data frame and calculate the accuracies and measure the computational time. Then do the same to each fold and calculate the time durations.

I have applied Random Forest and Deep Learning to entire dataset. Figure 93 shows the accuracies of random forest and deep learning. From the figure it can be seen that random forest took around 14 minutes to train the 80% of data that has 700,000 points with accuracy of 92.43%. While, deep learning method took 33 minutes to train 80% of the data with accuracy of 86%.

The deep learning method was trained on 10 epochs with batch size of 20. Provided more batch size and epochs, system would have performed better.

For 10 fold data, random forest took 10 minutes to train with average accuracy of 75.48%. The value of n_estimators for this task is 30. This 75% accuracy can be improved by increasing the value of n_estimators.

Deep learning method took 10 minutes with the accuracy of 87% while running on 5 epochs and batch size of 50. Provided more epoch and optimal batch size, accuracy would have increased. The details about accuracy and time can be seen in Figure 94 and Figure 95

XXI. Task 19: Comparing the results on a local system versus a big data system

19.1 Compile all the results that you produced from the experiment that implemented the random forest and deep learning techniques on the conceptualized local system using your data set/s.

The results of implementing random forest on local system using merged data can be seen in the Figure 96 and Figure 97. For deep learning results can be seen in Figure 98. From the figure it can be seen that random forest has 94% accuracy while deep learning method has 86% accuracy. The reason for lower accuracy in deep learning could be low number of epochs and batch size and no chance to perform parallelization.

Random Forest (Complete Data)	
Time	Accuracy
14 min	92.40%
Deep Learning(Complete Data)	
Time	Accuracy
33 min	86%

Fig. 93: Total Data accuracies

Random Forest			
Train	Test	Time	Accuracy
except 9	9 fold	9 min	80%
except 8	8 fold	8 min	83%
except 7	7 fold	8 min	81.50%
except 6	6 fold	8 min	83.87%
except 5	5 fold	8 min	84.75%
except 4	4 fold	10 min	87.28%
except 3	3 fold	11 min	90%
except 2	2 fold	10 min	88.36%
except 1	1 fold	11 min	76.10%

Fig. 94: 10 folds data accuracies of Random Forest

Deep Learning			
Train	Test	Time	Accuracy
except 9	9 fold	12 min	80%
except 8	8 fold	11 min	84%
except 7	7 fold	10 min	83.63%
except 6	6 fold	10 min	83.70%
except 5	5 fold	10 min	79.58%
except 4	4 fold	12 min	85.60%
except 3	3 fold	10 min	88%
except 2	2 fold	10 min	86.41%
except 1	1 fold	12 min	78.82%

Fig. 95: 10 folds data accuracies of Deep Learning

```
Our_Accuracy_Score: 0.9240307969254877
Our_Precision_Score: 0.9202618013817295
Our_Sensitivity_Score: 0.9526603635719506
Our_Specificity_Score: 0.8836908197488496

In [51]: from sklearn import metrics
    ...: print("BuiltIn_Accuracy:",metrics.accuracy_score(y_test, yhat_test))
    ...: print("BuiltIn_Precision:",metrics.precision_score(y_test, yhat_test))
    ...: print("BuiltIn_Sensitivity (recall):",metrics.recall_score(y_test, yhat_test))
BuiltIn_Accuracy: 0.9240307969254877
BuiltIn_Precision: 0.9298153467377923
BuiltIn_Sensitivity (recall): 0.8836908197488496
```

Fig. 96: Random Forest on merged Data

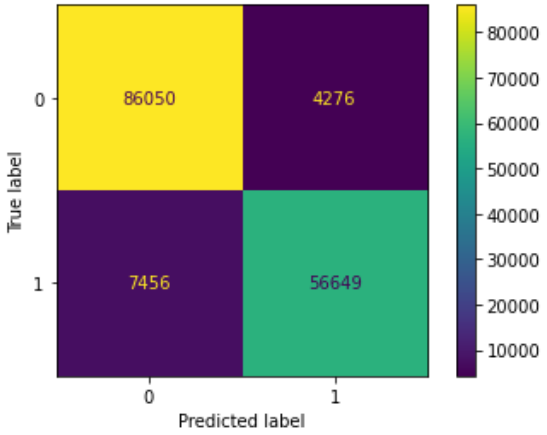


Fig. 97: Confusion matrix of Random Forest on merged data

19.2 Compile all the results that you produced from the experiment that implemented the random forest and deep learning techniques on the conceptualized big system using the 10-fold data sets.

With conceptualized big data system and 10 folds, random forest takes 10 minutes to train and validate all 10 blocks with average accuracy of 75.48%. For this task I have chosen n_estimators=50,This accuracy can be improved by increasing the number of n_estimators to 100 or 200 but it will take more time to train the data.

With deep learning method and parallelization concept, ten folds took 10 minutes with 87% accuracy on average. This accuracy can be further increased by increasing the number of epoch and taking more batch size in each epoch. For this task, I have used 5 epochs with batch size of 50. But it can be increased much more.

I will include the screenshots of each fold accuracy for random forest and deep learning in the screenshot folder. The accuracy for random forest and deep learning on 1st fold is shown in Figure 99 and Figure 100.

19.3 Compare all the results (qualitative measures) and de-

```
Our_Accuracy_Score: 0.8675913514773589
Our_Precision_Score: 0.8847060118916539
Our_Sensitivity_Score: 0.889544538671036
Our_Specificity_Score: 0.836658606972935
BuiltIn_Accuracy: 0.8675913514773588
BuiltIn_Precision: 0.8431560579145116
BuiltIn_Sensitivity (recall): 0.836658606972935
```

Fig. 98: Deep learning on merged data

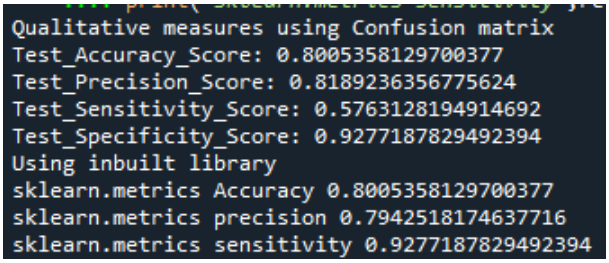


Fig. 99: Random forest accuracy on split_1

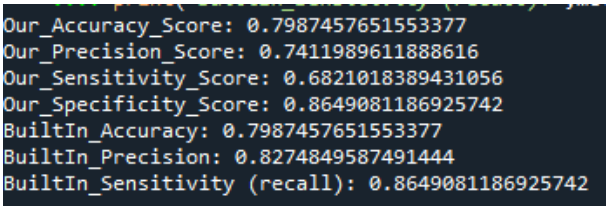


Fig. 100: Deep learning accuracy on split_1

termine if the conceptualized big data system was able to run the application that was failed to be completed on the conceptualized local system.

From the tasks 19.1 and 19.2 we can see that Deep Learning works well on a big data system with Ten Folds. As you can see its accuracy increased from 86% in local system to 87% on big data system.

As for random forest, it took 8.3 minutes to run on big data system because each fold is run simultaneously to each node of the big data system. It gives the total accuracy of 75.48%. Wile on local machine random forest is feed with the full merged data at a time so it took 14 minutes with the accuracy of 92.40%.

So from the results it is clear that big data system helps in running the application that failed to be completed on conceptualized local system.