ASSIGNMENT 1: Raw Data to Feature Space

Venkata Vijay Krishna Gabbula
University of North Carolina Greensboro
Email: V_GABBULA@UNCG.EDU

Task 1: Build your programming environment!

1.1 Install Anaconda Prompt(a Python Environment)

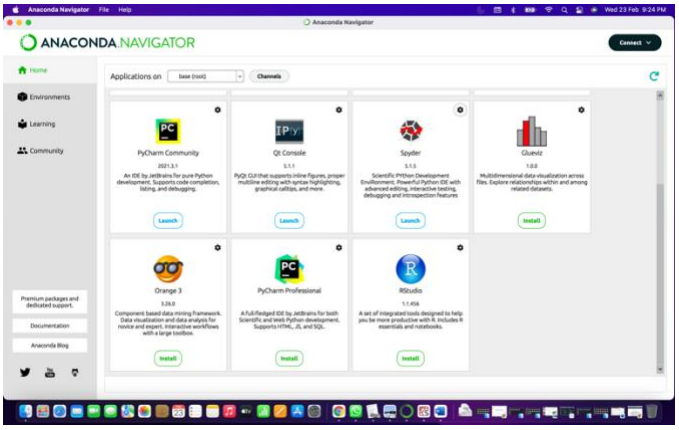


Figure 1: Installed Anaconda

1.2 Install the Spyder IDE :

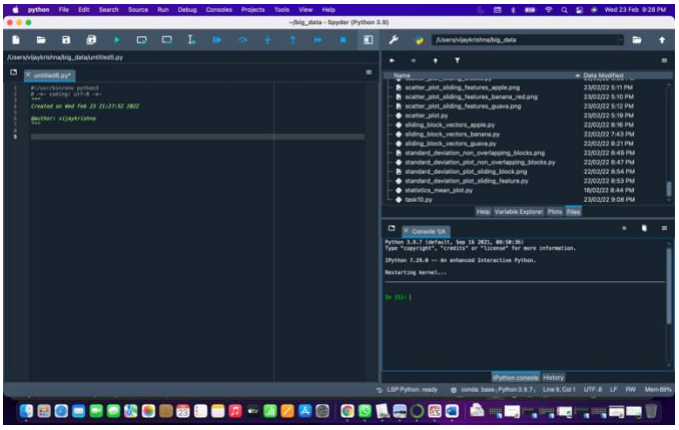


Figure 2: Installed Spyder IDE

1.3 Install Jupyter Notebook

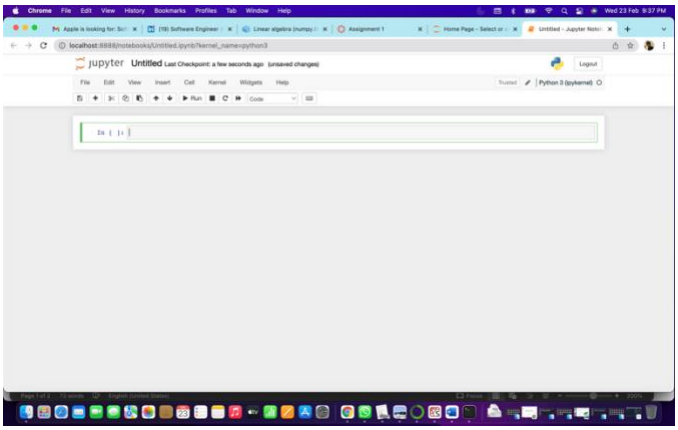


Figure 3: Installed Jupyter Notebook

1.4 Install Open CV

Installed OpenCV with the command

%pip install opencv-python

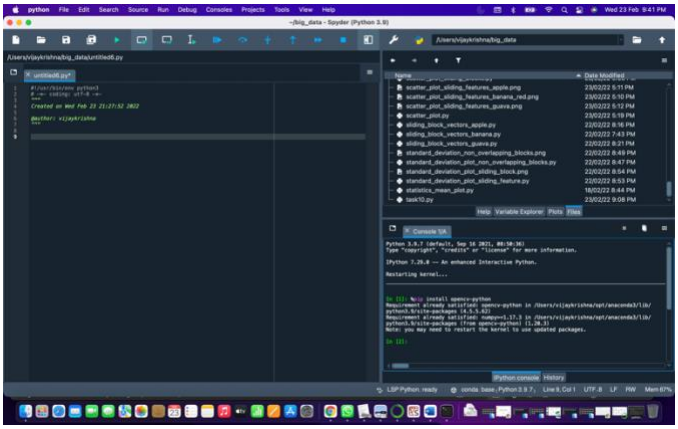


Figure 4: Download and install OpenCV

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

2.1 downloaded the tropical-fruits dataset from http://www.ic.unicamp.br/rocha/pub/downloads/tropical- fruits-DB-1024x768.tar.gz. Figure5 shows the data set stored in my workspace.

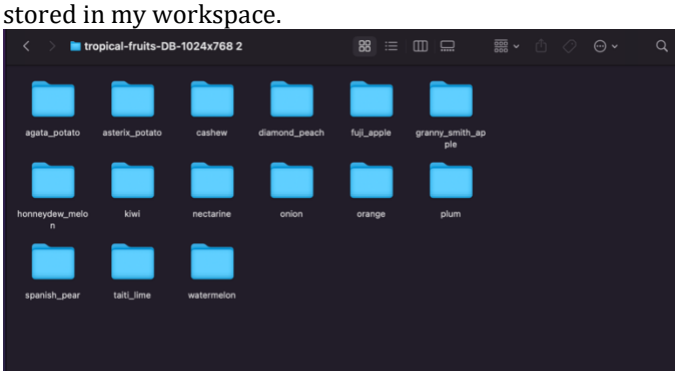


Figure5: screenshot of the dataset in my workspace https://www.vicos.si/Downloads/FIDS30 [FIDS30 Dataset] The below figure is the screenshot.

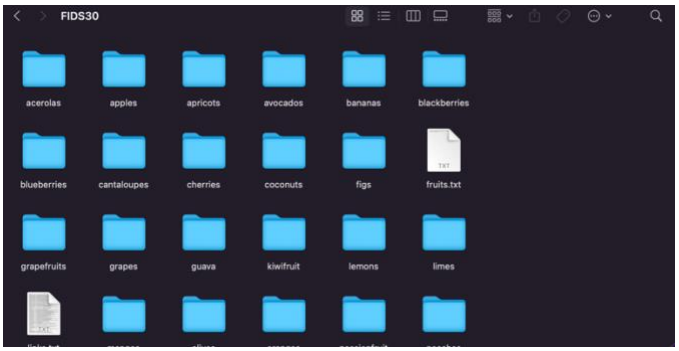


Figure 6: screenshot of the FIDS30 Dataset on my workspace(The source of the dataset is given above).

2.2 Selecting three fruits from dataset

The fruits I am choosing are Banana , Apple and Guava. I choose the images as to have a low background noise and the images are filled without any other colors as possible. I have assigned banana as banana1.jpg , apple as apple1.jpg and Guava as guava1.jpg. Figures 7, 8, 9 show the images of the fruits that I will be working on for this assignment.

Figure7 : banana1.jpg

Figure8 : apple1.jpg

Figure9 : guava1.jpg

TASK 3 **Read your selected images and display them on the environment**

3.1 write a code to read color images and display their R, G, B channels

I wrote a python code and imported cv2 package to read the images and display the R, G, B values for banana1.jpg , apple1.jpg, guava1.jpg. I used the CV2 library and matplotlib.pyplot to read and get the dimensions of the images and to display the images. Figure 10 below shows the code for the same.

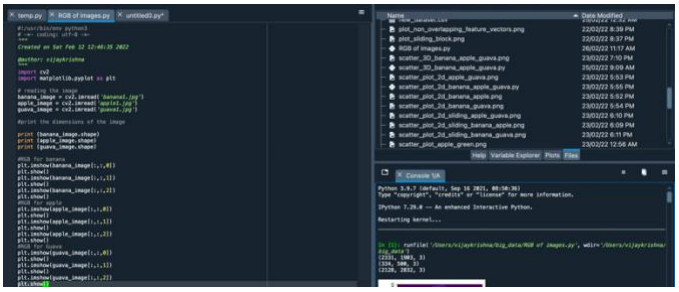


Figure10: RGB of images (banana, apple, guava) along with dimensions in the result

3.2 Converting the color images of fruits to grayscale and display them (also printing their dimensions):

I wrote a python code using CV2 . Firstly, the original images were converted to grayscale by using the function which is given as (cv2.cvtColor(image, cv2.COLOR.BGRA2GRAY)).Then I derived the dimensions of the converted grey image by using the function image.shape. Figure 11 shows the code for all the fruits converted to grey scale. Figure 12 - 14 are the grey images of the chosen fruits.

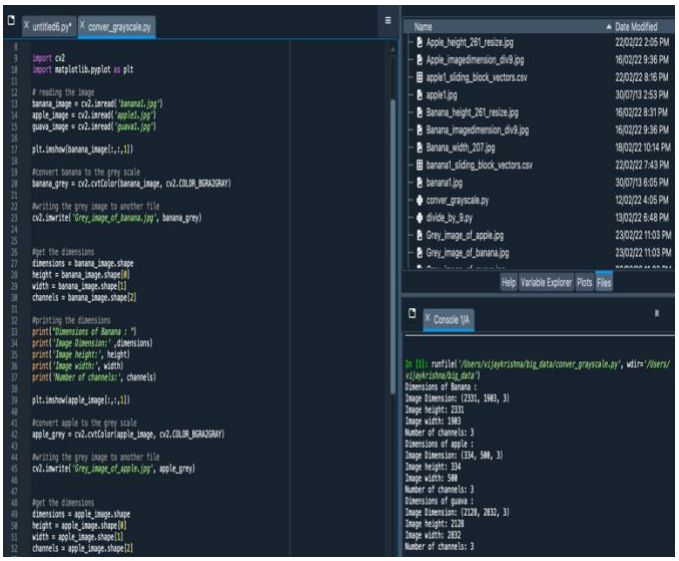


Figure11 : code for converting the images into greyscale

Figure12: Grey_image_of_banana.jpg

Figure13:Grey_image_of_apple.jpg

Figure14 : Grey_image_of_guava.jpg

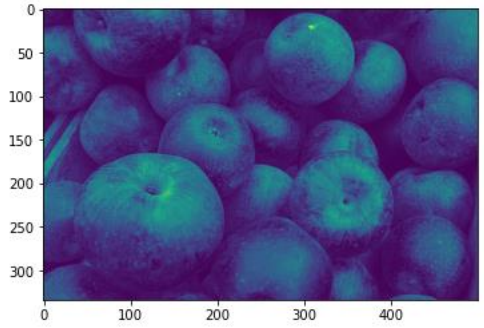


Figure15 : Red channel of apple1.jpg

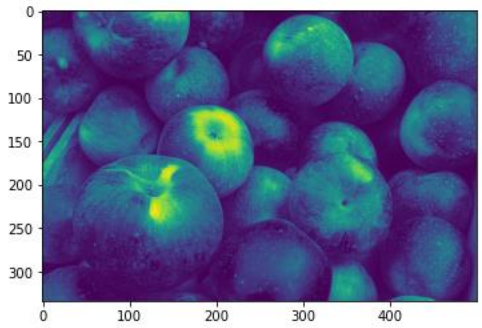


Figure16 : blue channel image of apple1.jpg

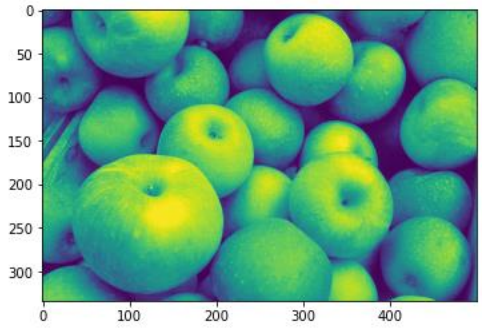


Figure17 : green channel image of apple1.jpg

Figures 15, 16, 17 displays the RGB images of apple1.jpg (apple fruit image considered in this project).

4.Resizing the images to reduce the dimensions :

4.1 Writing a function to resize the Grey_Scale images and setting the size of the image to be divisible by 9

I created a function which takes dimension as a parameter. The code checks if the passed argument is divisible by 9. I wrote an if statement to check if the remainder is equal to zero. If the remainder is zero then it is divisible by 9. If the dimension is not divisible by 9, I adjusted the values such that it is a multiple of 9.(We roundoff the values to be exact multiples of 9) The code is given below Figure 18 shows the python code for resizing the images.

Figure 18: Code for setting the dimensions divisible by 9

4.2 Convert the image to standard height 261

Figure 19 shows the code to convert the image to a standard height of 261. I set the standard height to 261 as per the assignment requirement 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 9. I performed this task for Grey_image_of_banana.jpg , Grey_image_of_apple.jpg and Grey_image_of_guava.jpg and saved the new dimension images in a different file with Banana_height_261_resize.jpg as (261*261), Apple_height_261_resize.jpg as (261*261), Guava_height_261_resize.jpg as (261 * 261). Refer Figure 19 for the code. Here we were asked to maintain the aspect ratio for the width as appropriate and I rounded off the aspect ratio of the width to 1:1 for height and width and so we got height, width as 261 pixels.

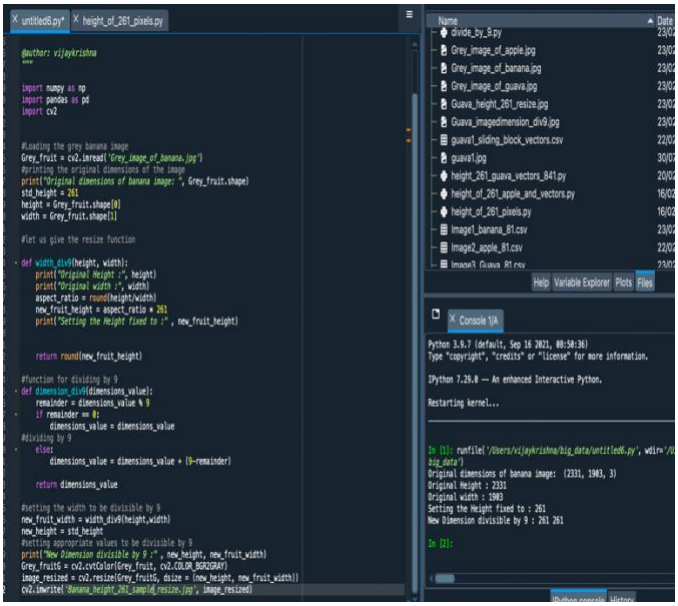


Figure 19 : setting the height to 261
(height set to 261 and width set to appropriate values as per aspect ratio)

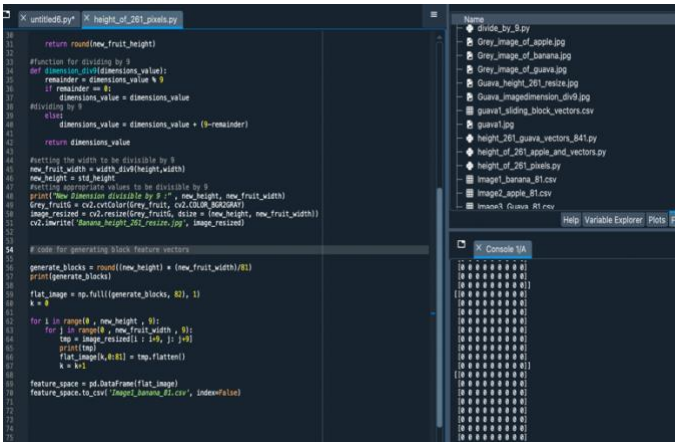
Figure :20 Banana image with a height set to 261 pixels

5 Generate block-feature vectors!

5.1 Create block feature vector of 9*9 size

I reused the code of Task 4 to convert the image to grayscale and make its width divisible by 9 with a standard height of 261. Figure 21 shows the code to generate feature vectors of 9*9 size. I created a CSV file to store the pixel values and labelling each file as Image1_banana_81.csv for Grey_image_of_banana , Image2_apple_81.csv for Grey_image_of_apple. Image3_guava_81.csv for Grey_image_of_guava. Images have 81 features and we have 841 blocks [csv file has 841 rows and 81 columns] [(261*261)/81] = 841. Figure 22-24 show the CSV files generated.

I have given the code only for banana image but I have included the spreadsheet for all the three fruits. I have not included the resized greyscale images as the height and width of the images are set to 261 pixels(approximated to aspect ratio). Here we set the actual label as '0' for banana '1' for apple and '2' for guava (0, 1, 2 for three fruits).



```
    return round(new_fruit_height)

#function for dividing by 9
def dimension_div9(dimensions_value):
    remainder = dimensions_value % 9
    if remainder == 0:
        dimensions_value = dimensions_value
#dividing by 9
    else:
        dimensions_value = dimensions_value + (9-remainder)

    return dimensions_value

#setting the width to be divisible by 9
new_fruit_width = width_div9(height,width)
new_height = std_height
#setting appropriate values to be divisible by 9
print("New Dimension divisible by 9 :", new_height, new_fruit_width)
Grey_fruitG = cv2.cvtColor(Grey_fruit, cv2.COLOR_BGR2GRAY)
image_resized = cv2.resize(Grey_fruitG, dsize = (new_height, new_fruit_width))
cv2.imwrite('Banana_height_261_resize.jpg', image_resized)

# code for generating block feature vectors

generate_blocks = round((new_height) * (new_fruit_width)/81)
print(generate_blocks)

flat_image = np.full((generate_blocks, 82), 1)
k = 0

for i in range(0 , new_height , 9):
    for j in range(0 , new_fruit_width , 9):
        tmp = image_resized[i : i+9, j: j+9]
        print(tmp)
        flat_image[k,0:81] = tmp.flatten()
        k = k+1

feature_space = pd.DataFrame(flat_image)
feature_space.to_csv('Image2_banana_81.csv', index=False)
```

Figure 21 : code to generate block -feature vectors

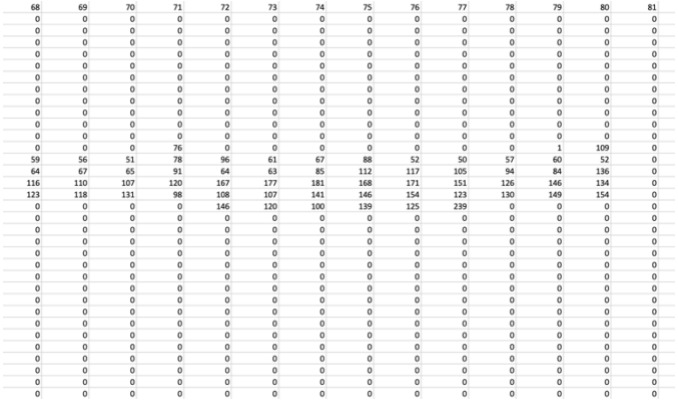


Figure 22: generating block feature vectors for banana image Image1_banana_81.csv(82nd column is assigned as '0') (each csv file generated consists of 841 feature vectors and 81 features)

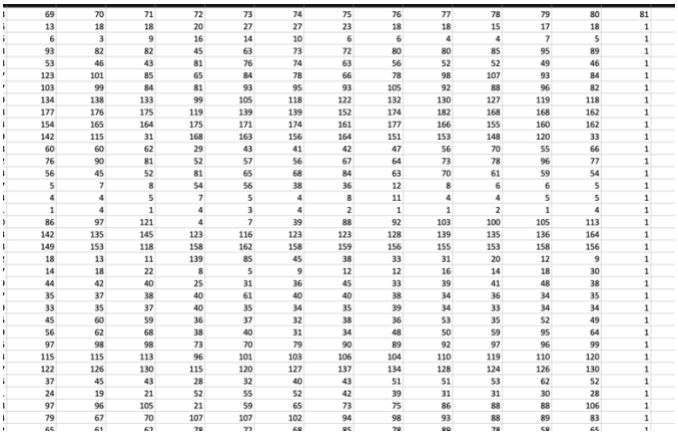


Figure23 : generating block feature vectors for apple image Image2_apple_81.csv (last column is initialized to '1').

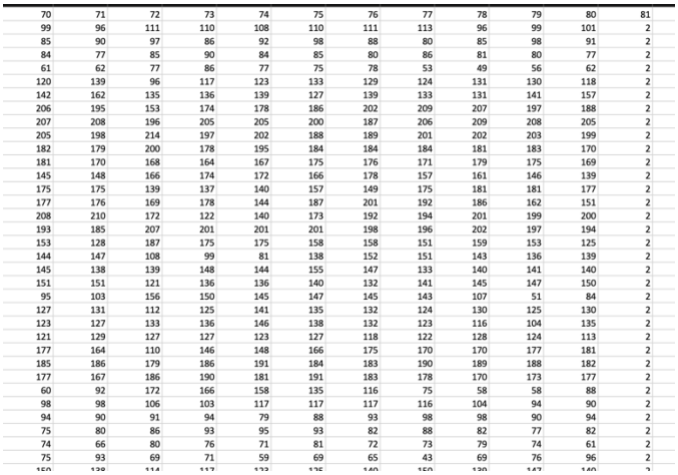


Figure24 : generating block feature vectors for guava image Image3_guava_81.csv (82nd column is initializes with label '2')

6.Generate sliding block-feature vectors! (9 × 9)

In generating sliding block feature, an image is converted to a gray scale and set a standard height of 261 pixels and resized the width appropriately. We take the image of the fruit banana (banana1.jpg), We take the image of the fruit apple (apple1.jpg), We

take the image of the fruit guava (guava1.jpg). Each Image CSV files are generated by the code given in the below figure 25 [banana1_sliding_block_vectors.csv for banana1.jpg image] [apple1_sliding_block_vectors.csv for apple1.jpg image] [guava1_sliding_block_vectors.csv for guava1.jpg image]. Figure 25 shows generating sliding blocks for banana image.

In the below code height is set to 261 pixels and width is appropriated to 207(as per aspect ratio), Then we generate sub-blocks of 54027 (261*207) and 667 blocks of size 81 (54027/81). So we have 81 features with 667 blocks. Each block has 81 cells of 9*9 size.

For the apple image the height is set to 261 and the width is 387 (as per the aspect ratio) [(261*387)/81] = 1247. So the apple image has 1247 blocks and 81 features

For the guava image the height is set to 261 and the width is 342 (as per the aspect ratio) [(261*342)/81] = 1102. We have sub- blocks = 89,262.So the guava image has 1102 blocks and 81 features. I have shared the code screenshot only for banana image (261*207) but I have given the screenshots of three fruits for sliding block vectors. Figures 26-28 shows the sliding blocks generated for three fruits. Even for the sliding block feature vectors we implement actu

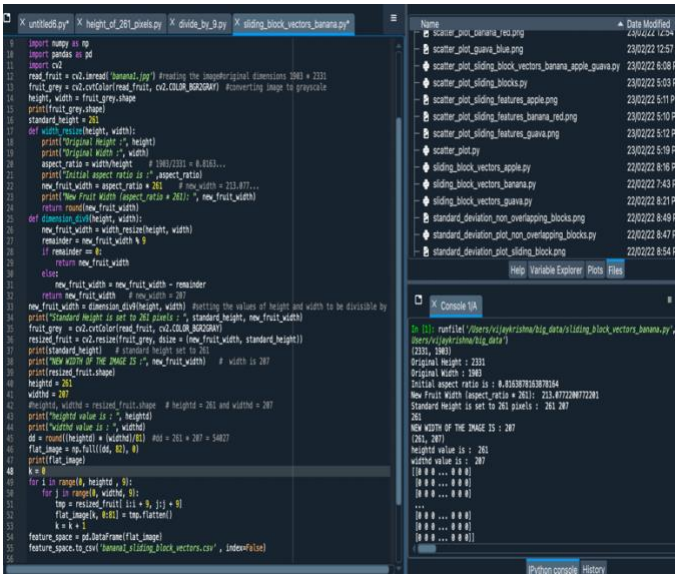


Figure25 : Generating sliding block vectors for the banana image

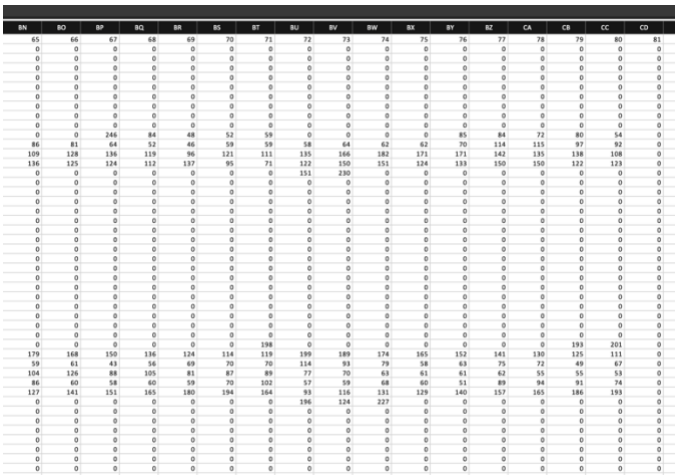


Figure 26: sliding block features of banana image

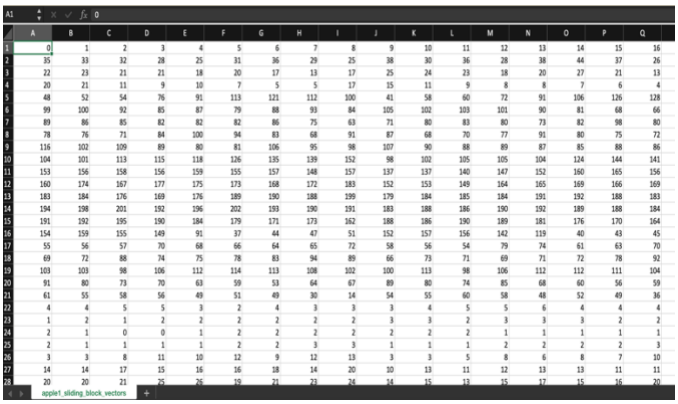


Figure 27: sliding block features of apple image

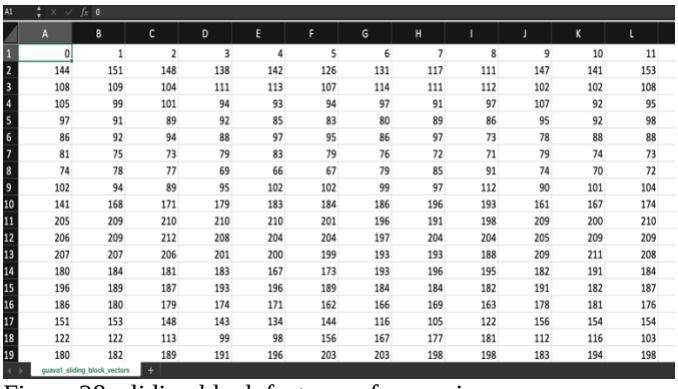


Figure 28: sliding block features of guava image

In figures 26-28 I assigned label ‘0’ for banana, ‘1’ for apple, ‘2’ for guava respectively. (I have done the same for block feature vectors as well) datasheets in figures 22-24.

Task 7: Derive statistical descriptors!

Extract statistical information (e.g. number of observations, dimension of the data, mean of each feature, etc.) from these datasets. Also present visual representations (e.g. histogram, scatter plot, etc.) of the data.

7.1 Observations :

Banana1.jpg(1st fruit image) has 841 observations[(261*261)/81 = 841] and 81 features. Similarly, Apple and Guava images also have the same observations as the blocks generated for resizes images with adjusted height to 261 pixels and width adjusted appropriate to that of height.

Similarly, for sliding block feature vectors width is equal to aspect ratio multiplied by 261. (width = aspect_ratio*261) and as always for this project height is set to 261.

So, sliding block vectors have block size of 667(banana image) 1247(apple image) 1102(guava image) and all have 81 features.

7.2 Mean Plot

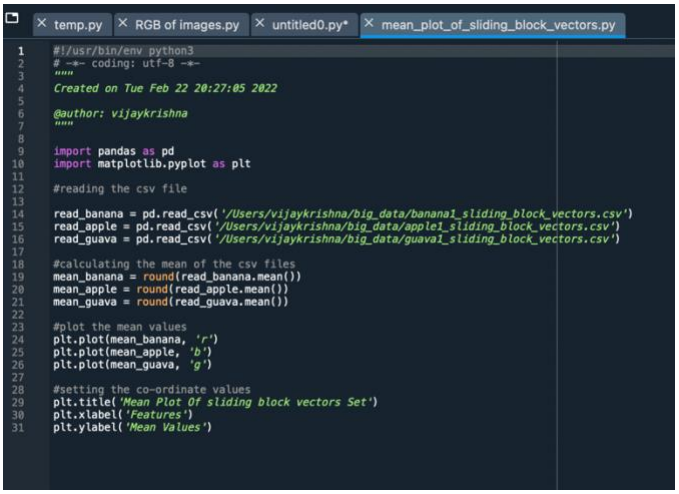


Figure 29: code for plotting mean values (I have taken the plot for sliding block vectors and not including the mean plot of block feature vectors).

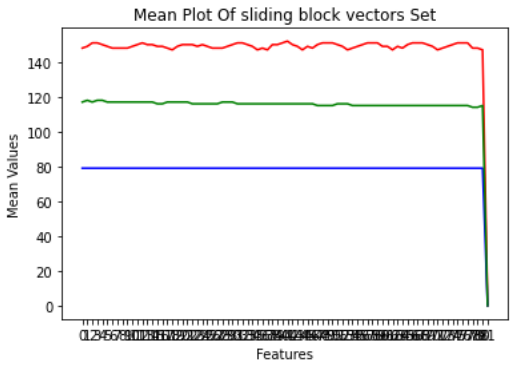


Figure 30: Plotting the mean values of images of sliding block vectors. (banana-red, apple-blue, guava-green).

7.3 standard deviation

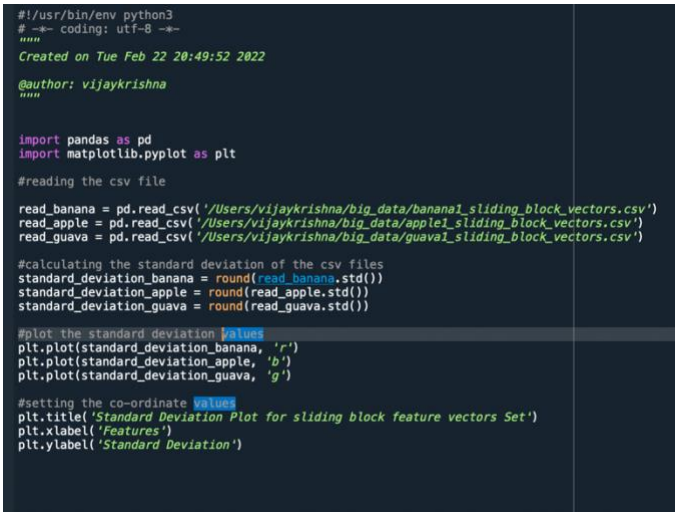


Figure 31: code for standard deviation plot for sliding block feature vectors

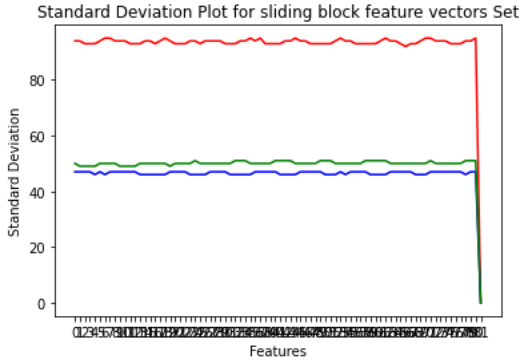


Figure32: standard deviation plot of sliding block vectors (red-banana, apple-blue, guava – green).

I took the standard deviation of the images of fruits and plotted a graph against them. Banana Image ranges around 100 , apple image ranges around 50 and Guava ranges around 50 89 These are the values for sliding block feature vectors. Figure 31 is the code and figure 32 is the figure plot. I have not included scatter plot for block size vectors.

7.4 scatter plot :

The below figures are the scatter plots of sliding block features (red- banana), (blue-apple), (green-guava). Fig33 for banana fig34 for apple and fig36 for guava.

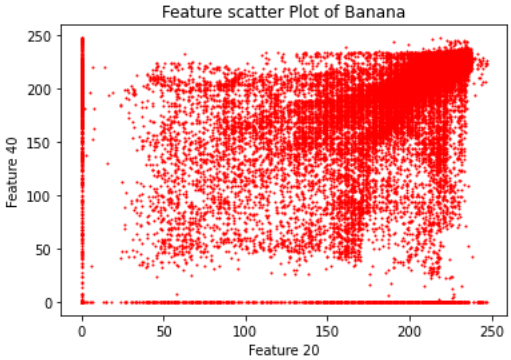


Figure 33: scatter plot of banana image(sliding block features)

Figure34: scatter plot for apple image

```
banana_overlap = pd.read_csv('Image_banana_Overlap.csv')
guava_overlap = pd.read_csv('Image_guava_Overlap.csv')
apple_overlap = pd.read_csv('Image_apple_Overlap.csv')



b20 = banana_overlap['20']
b40 = banana_overlap['40']

plt.scatter(b20, b40, color = ['red'], s=1)


plt.title("Feature scatter Plot of Banana")
plt.xlabel('Feature 20')
plt.ylabel('Feature 40')
plt.show()


a20 = apple_overlap['20']
a40 = apple_overlap['40']

plt.scatter(a20, a40, color = ['blue'], s=1)


plt.title("Feature scatter Plot of apple")
plt.xlabel('Feature 20')
plt.ylabel('Feature 40')
plt.show()


g20 = guava_overlap['20']
g40 = guava_overlap['40']

plt.scatter(g20, g40, color = ['green'], s=1)


plt.title("Feature scatter Plot of guava")
plt.xlabel('Feature 20')
plt.ylabel('Feature 40')
plt.show()
```

Figure35: code for plotting the scatter plot of the sliding block feature vectors (banana, apple and guava).

I have also plotted the histograms for block feature vectors and sliding block feature vectors that are given below.

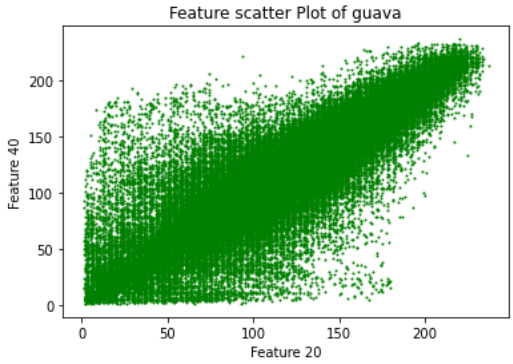


Figure36: scatter plot of guava image

We select any two features of the obtained 81 features for plotting the datapoints in the scatter plot. From code in figure 35 we can see the features selected for plotting.
Figures 33, 34, 36 shows the scatter plots for the sliding block feature vectors generated in step-6.

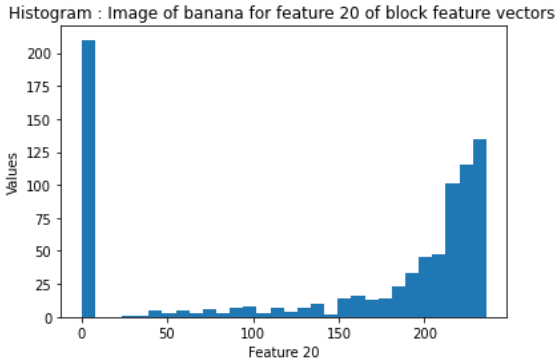


Figure: Histogram plot for banana image for block feature vectors with feature 20.

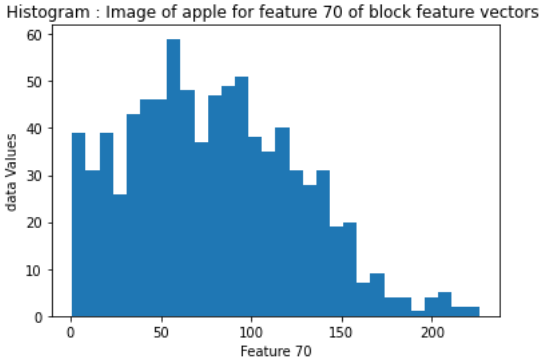


Figure: Histogram plot for apple image for block feature vectors with feature 70.

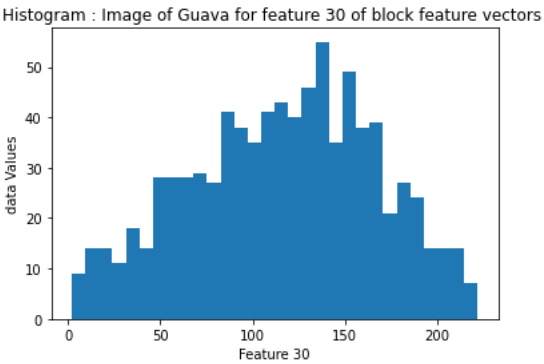


Figure: Histogram plot for guava image for block feature vectors with feature 30.

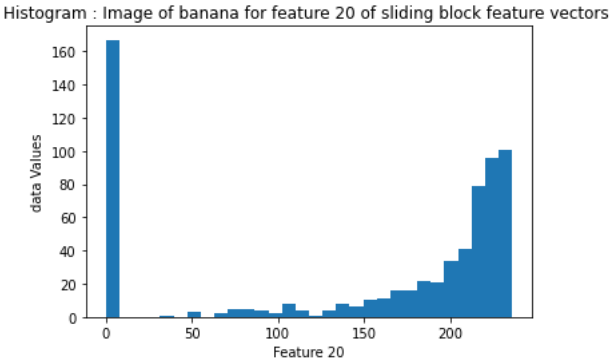


Figure: Histogram plot for banana image for sliding block feature vectors with feature 20.

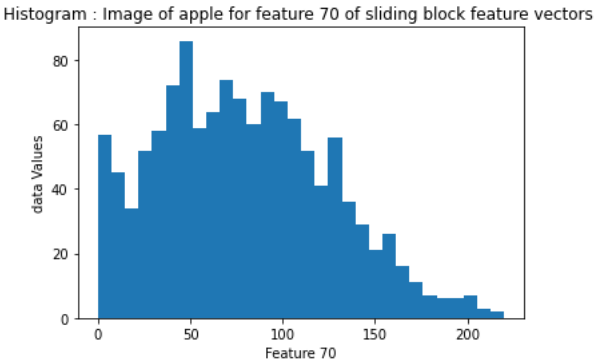


Figure: Histogram plot for apple image for sliding block feature vectors with feature 70.

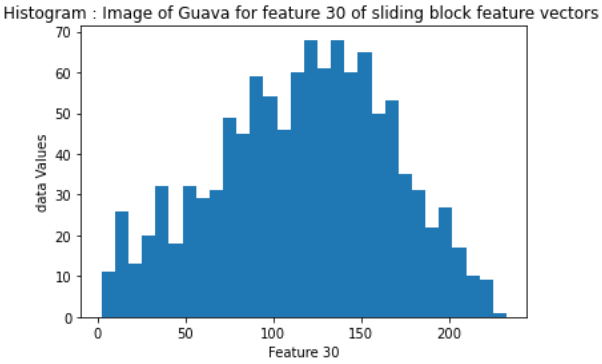


Figure: Histogram plot for guava image for sliding block feature vectors with feature 30.

The above six figures shows the histograms of fruit images on a single feature(for both block feature vectors and sliding block vectors).

Task 7.6. Answer the following questions -- Is the dataset imbalanced, inaccurate or incomplete?

The no of observations in all the three fruit images are same for the feature vectors which were generated in task5. Thus the data set is balanced. But for sliding block feature vectors , the number of observations in Image2_apple is more than Image1_banana and Image3_guava and so data is imbalanced . The data generated in the CSV files has no missing data and so the data wasn't incomplete. We were able to generate all features in the CSV files and hence the data is accurate.

Is it a trivial data or possibly a big data?

The total images that we are considering for analysis in this project is only 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 its observations and features with time .

Does it have scalability problem? Are they high dimensional?

We have the feature size as 9*9 for each feature. So if the features increases the scalability also increases. In this project we are considering three fruit images and plotting so I don't see any high dimensionality or scalability problem.

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

Yes it need to be standardized and normalized. For measuring the accuracy and estimating the distribution of the data we need to standardize and normalize. Consider a example with a variable that ranges between 0 and 100 will definitely out weight a variable that ranges between 0 and 1. Using such variables for standardization/normalization will give the variable with the larger range weight of 100 in the analysis. So it is necessary that all the variables fall in the same normal distribution curve to receive good accuracy.

The data characteristics which are namely volume, velocity and variety seems to have no effect on the dataset with normalization and standardization.

TASK 8 :

Construct a feature space!

I merged the Image1_banana.csv and Image2_apple.csv to produce a concatenated file i.e. Image12_merged.csv. Figure 37 shows the screenshot of the code which merges and uses random function to distribute the data in the csv file randomly. The data is randomized row-wise to jumble the data. Similarly the Image123_merged.csv was obtained by concatenating Image12_merged.csv with Image3_guava.csv. I have done the same for the datasets created for the sliding block feature vectors.

```
Created on Wed Feb 23 00:12:04 2022

@author: vijaykrishna
"""

import pandas as pd
import numpy as np

image1_banana = pd.read_csv('Image1_banana_81.csv')
image2_apple = pd.read_csv('Image2_apple_81.csv')

frames = [image1_banana, image2_apple]

merged = pd.concat(frames)

index = np.arange(len(merged))
merge = np.random.permutation(index)
merge = merged.sample(frac=1).reset_index(drop=True)
merge.to_csv('Image12_merged.csv', index=False)
```

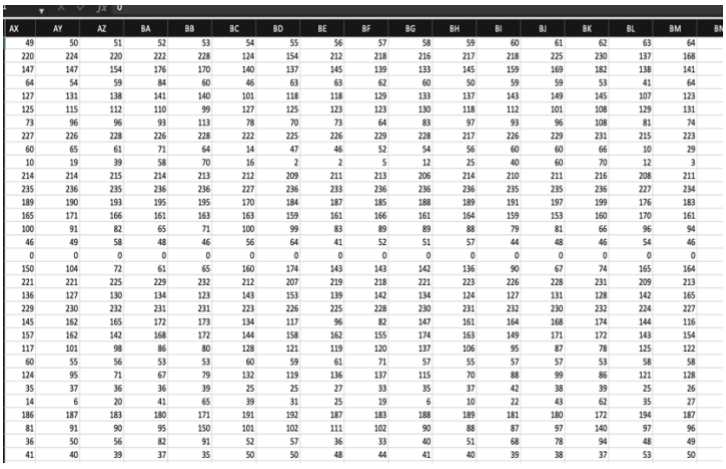
Figure37: code for merging banana and apple datasets of block feature vectors

```
Created on Wed Feb 23 00:40:31 2022

@author: vijaykrishna
"""

import pandas as pd
import numpy as np

image1_banana = pd.read_csv('Image12_merged.csv')
image2_apple = pd.read_csv('Image3_Guava_81.csv')

frames = [image1_banana, image2_apple]

merged = pd.concat(frames)

index = np.arange(len(merged))
merge = np.random.permutation(index)
merge = merged.sample(frac=1).reset_index(drop=True)
merge.to_csv('Image123_merged.csv', index=False)
```

Figure38: code for merging all three datasets of block feature vectors.

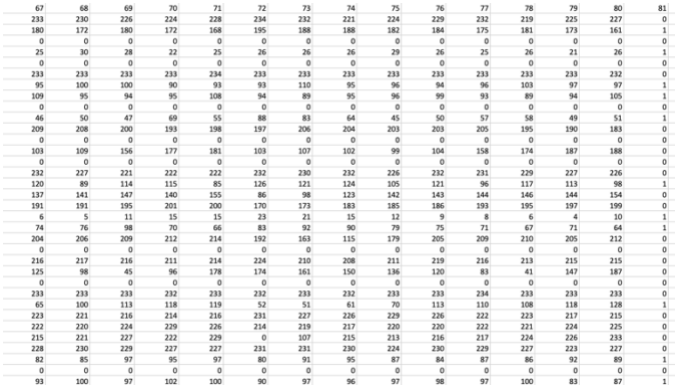
In the block feature vectors we created in task 5 we have 841 observations and 81 features. So in the merged dataset(after merging datasets of apple, guava and banana) we will have a total [841*3 = 2523 observations with 81 features]. The below figure 39 shows a screenshot of the dataset obtained after merging the three datasets. We randomize the rows in the merged file by implementing a random function(only rows are randomly printed but data will be the same. I have only showed you the merged datasets for block feature vector in this presentation. Figure 40 shows the screenshot of dataset obtained from merging the banana and apple datasets. I merged the sliding block features datasets as well.

I used the pd.concat for concatenating the dataframes and implemented np.arrange to randomly select rows while merging. I merged the datasets of all the three fruits of block feature vectors and also for the sliding block feature vectors as well.



AX	AY	AZ	BA	BB	BC	BD	BE	BF	BG	BH	BI	BJ	BK	BL	BM
49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64
220	224	220	222	228	124	154	212	218	216	217	218	225	230	137	168
147	147	154	176	170	140	137	145	139	133	145	159	169	182	138	141
64	54	59	84	60	46	63	63	62	60	50	59	59	53	41	64
127	131	138	141	140	101	118	118	129	133	137	143	149	145	107	123
125	115	112	110	99	127	125	123	123	130	118	112	101	108	129	131
73	96	96	93	113	78	70	73	64	83	97	93	96	108	81	74
227	226	228	226	228	222	225	226	229	228	217	226	229	231	215	223
60	65	61	71	64	14	47	46	52	54	56	60	60	66	10	29
10	19	39	58	70	16	2	2	5	12	25	40	60	70	12	3
214	214	215	214	213	212	209	211	213	206	214	210	211	216	208	211
235	236	235	236	236	227	236	233	236	236	236	235	235	236	227	234
189	190	193	195	195	170	184	187	185	188	189	191	197	199	176	183
165	171	166	161	163	163	159	161	166	161	164	159	153	160	170	161
100	91	82	65	71	100	99	83	89	89	88	79	81	66	96	94
46	49	58	48	46	56	64	41	52	51	57	44	48	46	54	46
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
150	104	72	61	65	160	174	143	143	142	136	90	67	74	165	164
221	221	225	229	232	212	207	219	218	221	223	226	228	231	209	213
136	127	130	134	123	143	153	139	142	134	124	127	131	128	142	165
229	230	232	231	231	223	226	225	228	230	231	232	230	232	224	227
145	162	165	172	173	134	117	96	82	147	161	164	168	174	144	116
157	162	142	168	172	144	158	162	155	174	163	149	171	172	143	154
117	101	98	86	80	128	121	119	120	137	106	95	87	78	125	122
60	55	56	53	53	60	59	61	71	57	55	57	57	53	58	58
124	95	71	67	79	132	119	136	137	115	70	88	99	86	121	128
35	37	36	36	39	25	25	27	33	35	37	42	38	39	25	26
14	6	20	41	65	39	31	25	19	6	10	22	43	62	35	27
186	187	183	180	171	191	192	187	183	188	189	181	180	172	194	187
81	91	90	95	150	101	102	111	102	90	88	87	97	140	97	96
36	50	56	82	91	52	57	36	33	40	51	68	78	94	48	49
41	40	39	37	35	50	50	48	44	41	40	39	38	37	53	50

Figure39: Dataset for the merged datasets of three fruits. Banana-0, apple-1, guava-2 for the last column randomly printed).



67	68	69	70	71	72	73	74	75	76	77	78	79	80	81
233	230	226	224	228	234	232	221	224	229	232	219	225	227	0
180	172	180	172	168	195	188	188	182	184	175	181	173	161	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
25	30	28	22	25	26	26	26	29	26	25	26	21	26	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
233	233	233	233	234	233	233	233	233	233	233	233	233	232	0
95	100	100	90	93	93	110	95	96	94	96	103	97	97	1
109	95	94	95	108	94	89	95	96	99	93	89	94	105	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
46	50	47	69	55	88	83	64	45	50	57	58	49	51	1
209	208	200	193	198	197	206	204	203	203	205	195	190	183	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
103	109	156	177	181	103	107	102	99	104	158	174	187	188	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
232	227	221	222	222	232	230	232	226	232	231	229	227	226	0
120	89	114	115	85	126	121	124	105	121	96	117	113	98	1
137	141	147	140	155	86	98	123	142	143	144	146	144	154	0
191	191	195	201	200	170	173	183	185	186	193	195	197	199	0
6	5	11	15	15	23	21	15	12	9	8	6	4	10	1
74	76	98	70	66	83	92	90	79	75	71	67	71	64	1
204	206	209	212	214	192	163	115	179	205	209	210	205	212	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
216	217	216	211	214	224	210	208	211	219	216	213	215	215	0
125	98	45	96	178	174	161	150	136	120	83	41	147	187	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
233	233	233	232	233	232	233	232	233	233	234	233	233	233	0
65	100	113	118	119	52	51	61	70	113	110	108	118	128	1
223	221	216	214	216	231	227	226	229	226	222	223	217	215	0
222	220	224	229	226	214	219	217	220	220	222	221	224	225	0
215	221	227	222	229	0	107	215	213	216	217	224	226	233	0
228	230	229	227	227	231	231	230	224	230	229	227	223	227	0
82	85	97	95	97	80	91	95	87	84	87	86	92	89	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
93	100	97	102	100	90	97	96	97	98	97	100	83	87	1

Figure 40: Dataset of merged datasets of banana and apple(banana-0 and apple-1 in last column)rows randomized in printing.

Task 9: Display subspaces!

9.1 Select two features and plot the two-dimensional feature space with labelling the observations (vectors) of the fruits or vegetables that you selected by using the spreadsheets that you generated

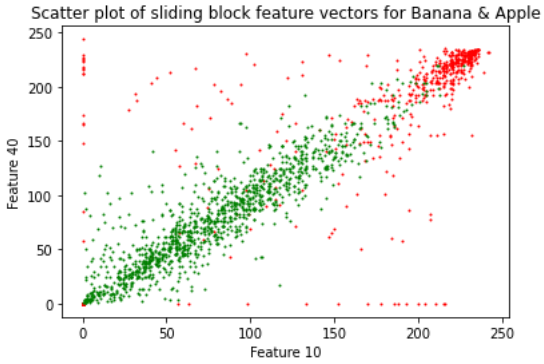


Figure 41: 2D scatter plot for banana(red) and apple(green). (sliding block feature vectors).

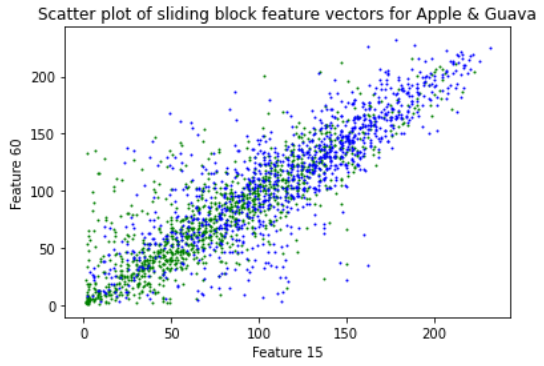


Figure 42: 2D Scatter plot for apple(green) and guava(blue) (sliding block feature vectors).

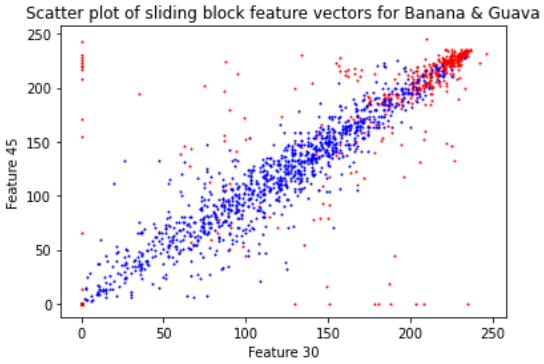


Figure 43: 2D scatter plot for banana(red) and guava(blue) (sliding block feature vectors).

Figures 41-43 shows the 2D scatter plots for two image features for sliding block feature vectors. We select the same features for both the fruits and plot the datapoints on the graph. Figure 44 gives the code used for plotting the graphs of fruits in 2D.

```
#scatter plot to print 2D images of sliding block feature vectors
import pandas as pd
import matplotlib.pyplot as plt

# read the csv file
read_apple = pd.read_csv('apple1_sliding_block_vectors.csv')
read_banana = pd.read_csv('banana1_sliding_block_vectors.csv')
read_guava = pd.read_csv('guava1_sliding_block_vectors.csv')

b10 = read_banana['10']
b40 = read_banana['40']
a10 = read_apple['10']
a40 = read_apple['40']

plt.scatter(b10, b40, color = 'red', s=1)
plt.scatter(a10, a40, color = 'green', s=1)

plt.title("Scatter plot of sliding block feature vectors for Banana & Apple")
plt.xlabel('Feature 10')
plt.ylabel('Feature 40')
plt.show()

a20 = read_apple['20']
a60 = read_apple['60']
g20 = read_guava['20']
g60 = read_guava['60']

plt.scatter(a20, a60, color = 'green', s=1)
plt.scatter(g20, g60, color = 'blue', s=1)

plt.title("Scatter plot of sliding block feature vectors for Apple & Guava")
plt.xlabel('Feature 15')
plt.ylabel('Feature 60')
plt.show()

b30 = read_banana['30']
b50 = read_banana['50']
g30 = read_guava['30']
g50 = read_guava['50']

plt.scatter(b30, b50, color = 'red', s=1)
plt.scatter(g30, g50, color = 'blue', s=1)

plt.title("Scatter plot of sliding block feature vectors for Banana & Guava")
plt.xlabel('Feature 30')
plt.ylabel('Feature 45')
plt.show()
```

Figure 44: scatter plot code for two fruits

9.2 Select three features and plot the three-dimensional feature space with labelling the observations (vectors) of the fruits or vegetables that you selected by using the spreadsheets that you generated.

```
"""
Created on Wed Feb 23 18:37:31 2022

@author: vijaykrishna
"""

#scatter plot to printing 3D image of non-overlapping feature vectors
import pandas as pd
import matplotlib.pyplot as plt

# read the csv file

read_banana = pd.read_csv('Image1_banana_81.csv')
read_apple = pd.read_csv('Image2_apple_81.csv')
read_guava = pd.read_csv('Image3_Guava_81.csv')

b20 = read_banana['20']
b40 = read_banana['40']
b60 = read_banana['60']

a20 = read_apple['20']
a40 = read_apple['40']
a60 = read_apple['60']

g20 = read_guava['20']
g40 = read_guava['40']
g60 = read_guava['60']

# create figure

fig = plt.figure(figsize = (20, 40))
ax = plt.axes(projection = "3d")

# creating plot
ax.scatter3D(b20, b40, b60, color = "green")
ax.scatter3D(a20, a40, a60, color = "blue")
ax.scatter3D(g20, g40, g60, color = "red")

plt.title("3D scatter plot of Banana & Apple & Guava")
plt.xlabel('Feature 20')
plt.ylabel('Feature 40')
ax.set_zlabel('Feature 60')
plt.show()
```

Figure 45: code for 3D scatter plot for three fruits.

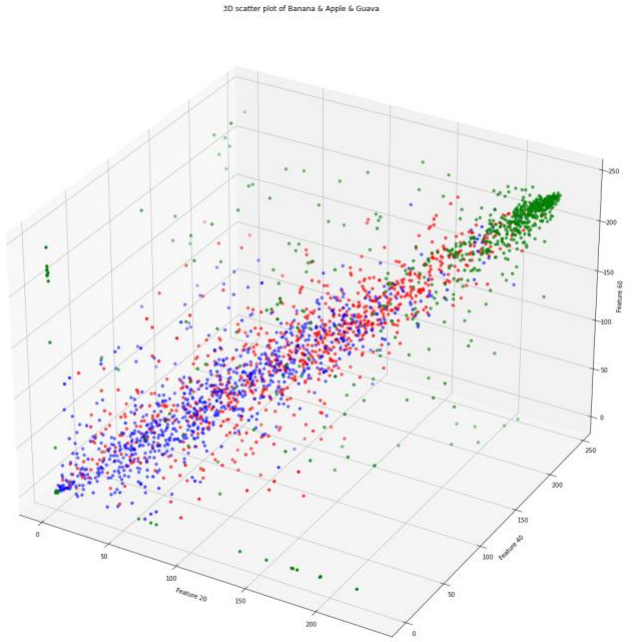


Figure46: 3D plotting of all the three fruits (feature20 x-axis) (feature40 y-axis) (feature60 z-axis) fruits(banana-green, apple-blue, guava-red).

Figures 45 and 46 shows the code and the scatter plots for all the three fruits considered in the project. I have provided separate 2d to show the two class labels for fruits plot.

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

Figure 41 above is a imbalanced data as the plot for banana is higher and only concentrated on one side but the plot for apple(green) is almost

uniformly distributed. Figure 42 is a balanced data as all the datapoints of two fruits are almost uniformly distributed in the graph. It will be more complex to classify between the datapoints of two images for figure 42 Again the figure 43 is imbalanced data as the scatter plot for banana(red) is only concentrated higher in the graph but data is uniformly distributed for guava(blue).

Task10. Read multiple files from a folder

I reused the code from the task 6 to generate the datasets and also wrote a function which will read all the images and create their respective CSV files. Figure 47-48 show the code and the CSV files generated by reading all of them.

I have taken the grapes Dataset from FIDS 30 to read and generate all the datasets of every image present in the dataset.

```
std_height = 261

def width_div9(height, width):
    print("Original Height :", height)
    print("Original width :", width)
    aspect_ratio = round(height/width)
    new_fruit_width = aspect_ratio * 261
    print("width :" , new_fruit_width)
    return round(new_fruit_width)

def read_multiple_files( image , filename):
    fruit_gray = cv2. cvtColor(image, cv2.COLOR_BGRA2GRAY)
    height , width = fruit_gray.shape
    print("Dimension of Gray image:", height , width)
    new_fruit_width = width_div9(height,width)

    #Values when both the height and width is to be divisible by 9
    print("New Dimension divisible by 9 :" , 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)/81)
    print (dd)
    flat_image = np.full((dd , 82) , 1)
    k=0
    for i in range(0 , heightd, 9):
        for j in range(0, widthd ,9):
            tmp = image_resized[i : i+9 , j: j+9]
        print (tmp)
        flat_image[k, 0:81] = tmp. flatten()
        k = k+ 1

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

    return feature_space

folder_path = '/Users/vijaykrishna/big_data_datasets/FIDS30/grapes'

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='GrapesDataset'+splitted_filename[0]+'.csv'
        read_multiple_files(img, rename_csvfile)
```

Figure 47: code to generate multiple files from a folder

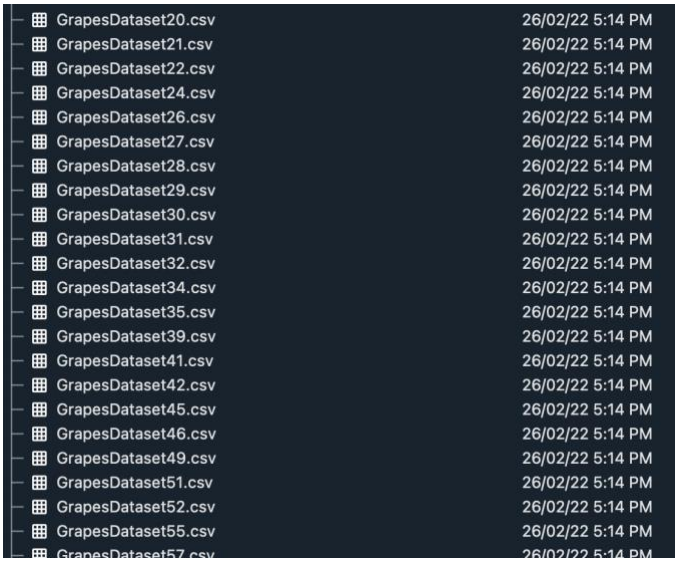


Figure 48: shows multiple csv files generated

Task 11: Describe the effects of block size on the dimensionality of the feature space and the number of vectors in the domain. Also, describe how these effect may influence the classifier that divides the domain.

The images that we generated are of size 9 * 9 in this project. Thus the feature vector will be of size 81. Suppose let us say we decrease the size to 4*4 then the number of observations will be more to that of the features. If we increase size by 16 * 16 the features will increase and the number of observations might decrease.

Hence I conclude that this would lead to high dimension or low dimension issue and so it may effect the accuracy and so we might get inaccurate results.

Assignment 2: Feature Space to a Classifier

Task 1:Dividing the data domain of the datasets into 78:22 (testing and training respectively).

```
import numpy as np

#reading the feature space of block feature vectors of apple and banana
file=pd.read_csv('Image12_merged.csv')
row,col=file.shape
# splitting the dataset into 78:22 for training and testing respectively

training=round(row*0.78)
file1=np.array(file)

# setting the testing values to 22%
testing=row-training

# creating the training and testing datasets
file_training=file1[0:training,:]
file_testing=file1[training:row,:]

#saving the testing dataset
training_data=pd.DataFrame(file_training)
training_data.to_csv('train_Image12_merged.csv')

#saving the testing dataset
testing_data=pd.DataFrame(file_testing)
testing_data.to_csv('testing_Image12_merged.csv')

#reading the feature space of block feature vectors of apple and banana and guava

file_123=pd.read_csv('Image123_merged.csv')

row,col=file_123.shape

# splitting the dataset into 78:22 for training and testing respectively

training123=round(row*0.78)
file_merged=np.array(file_123)

# setting the testing values to 22%
testing123=row-training123
# creating the training and testing datasets

file_training123=file_merged[0:training123,:]
file_testing123=file_merged[training123:row,:]

#saving the testing dataset
training_data123=pd.DataFrame(file_training123)
training_data123.to_csv('train_Image123_merged.csv')

#saving the testing dataset
testing_data123=pd.DataFrame(file_testing123)
testing_data123.to_csv('testing_Image123_merged.csv')
```

Figure 49(a): code for dividing the data domain 78:22 (block feature vectors for merged sets of two and three images respectively).

```
#code for generating training and testing datasets for sliding block feature vectors

import pandas as pd
import numpy as np

#reading the feature space of sliding block feature vectors of apple and banana
file=pd.read_csv('Image12_sliding_banana_apple_merged.csv')

row,col=file.shape

# splitting the dataset into 78:22 for training and testing respectively

TR=round(row*0.78)

file1=np.array(file)

# setting the testing values to 22 percent
TT=row-TR

# creating the training and testing datasets

file_TR=file1[0:TR,:]
file_TT=file1[TR:row,:]

#saving the testing dataset
TR_data=pd.DataFrame(file_TR)
TR_data.to_csv('train_Image12_sliding_merged.csv')

#saving the testing dataset
TT_data=pd.DataFrame(file_TT)
TT_data.to_csv('testing_Image12_sliding_merged.csv')
```

Figure 49(b): Code for dividing sets 78:22 for sliding block feature vectors of two fruits.

Selecting two features in each of training and testing datasets, (I choose features ‘76’ and ‘77’ and plotting their histograms.

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

file=pd.read_csv('train_Image123_merged.csv')

f76=file['76']
f77=file['77']

nbins=60

plt.title('Histogram of block feature vectors f76 training dataset for Image123')
plt.hist(f76, nbins, color='r', edgecolor='k')
plt.axvline(f76.mean(), color='g', linestyle='dotted', linewidth=1)
plt.text(f76.mean(), 261,r'mean')
plt.show()



plt.title('Histogram of block feature vectors f77 training dataset for Image 123')
plt.hist(f77, nbins, color='r', edgecolor='k')
plt.axvline(f77.mean(), color='g', linestyle='dashdot', linewidth=1)
plt.text(f77.mean(), 261,r'mean')
plt.show()
```

Figure 50: Code for Histogram plot of training set of block feature vectors (dataset is the merged dataset of three fruits).

I have plotted the histograms for block feature vectors and for the sliding block feature vectors (for both training and testing datasets). I also plotted the histograms for the merged datasets of two fruits as well.

I got the same gaussian curve trend and then I plotted the mean values of the two features and showed that in the plot. All the histogram plots are for the same features ‘76’ and ‘77’.

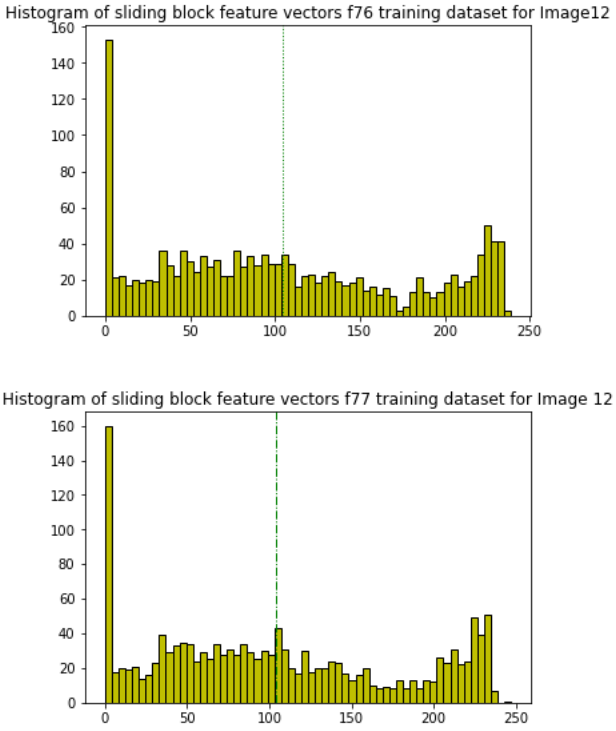


Figure: Histogram plots for the features ‘76’ and ‘77’ for training dataset of sliding block feature vectors of two fruits merged dataset (merged dataset of apple and banana).

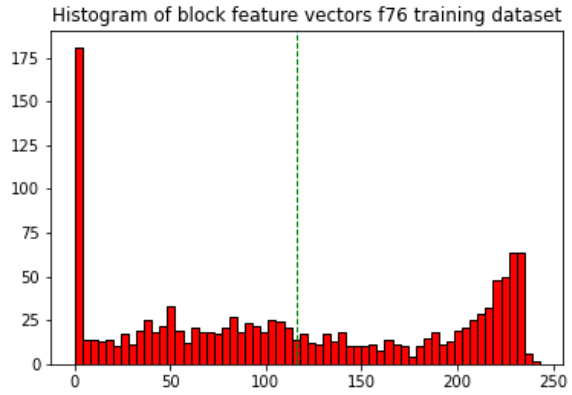


Figure51: histogram plot for block feature vectors of training dataset for the feature '76'.

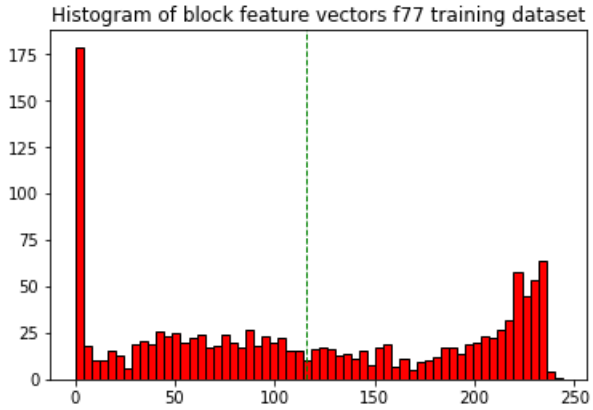


Figure52: histogram plot for block feature vectors of training dataset for the feature '77'.

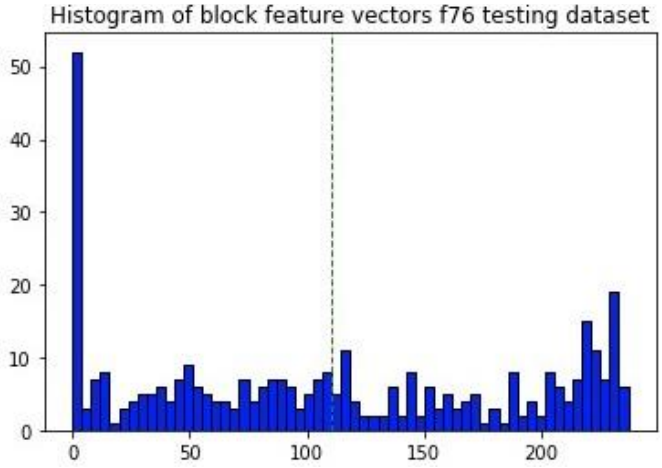


Figure53: histogram plot for block feature vectors of testing dataset for the feature '76'.

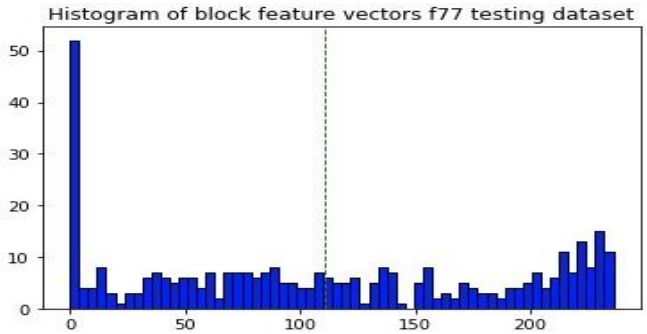


Figure53: histogram plot for block feature vectors of testing dataset for the feature '77'.

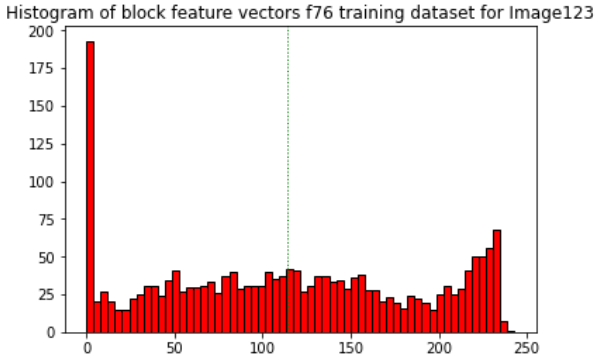


Figure54: histogram plot for block feature vectors of training dataset for the feature '76' and for three fruits merged dataset.

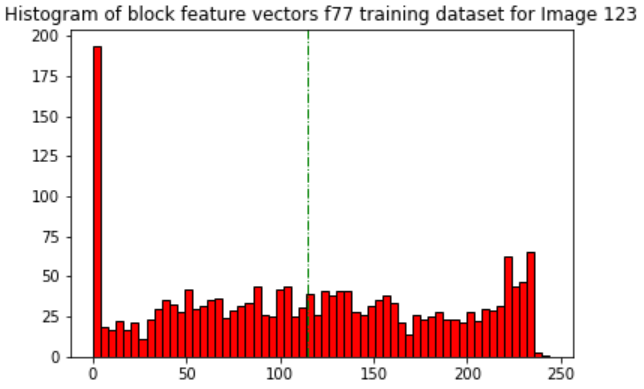


Figure55: histogram plot for block feature vectors of training dataset for the feature '77' and for three fruits merged dataset.

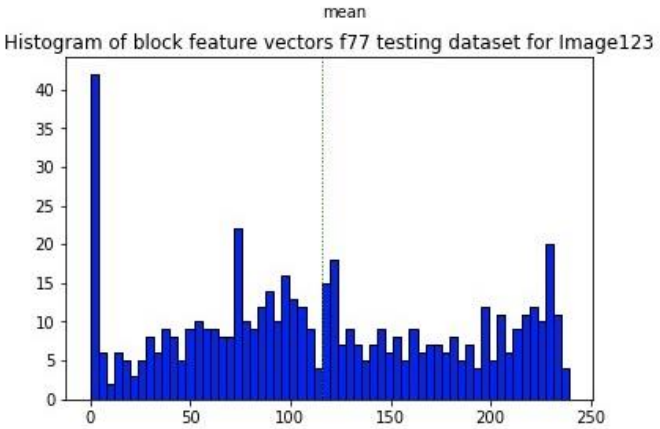


Figure56: histogram plot for block feature vectors of testing dataset for the feature '77' and for three fruits merged dataset.

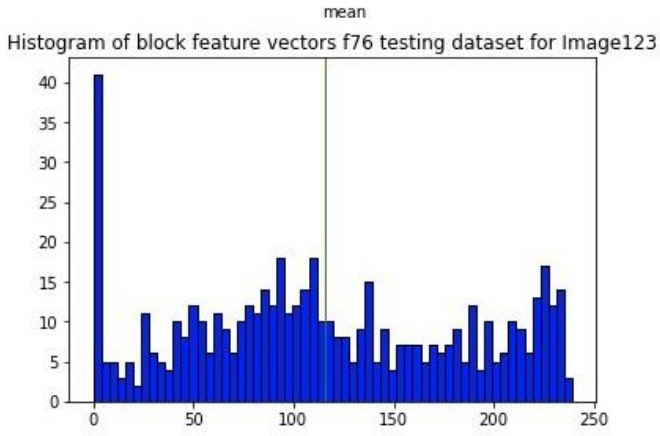


Figure57: histogram plot for block feature vectors of testing dataset for the feature '77' and for three fruits merged dataset.

Similarly I plotted the histograms for sliding block features dataset (for merged datasets of two fruits and also for three fruits merged dataset) (All the histogram plots are for features ‘76’ and ‘77’).

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

file=pd.read_csv('train_Image123_merged.csv')

f76=file['76']
f77=file['77']

nbins=60

plt.title('Histogram of block feature vectors f76 training dataset for Image123_block_feature_vectors')
plt.hist(f76, nbins, color='r', edgecolor='k')
plt.axvline(f76.var(), color='r', linestyle='dotted', linewidth=5)

#plt.text(f76.mean(), 261,r'mean')
plt.text(f76.var(), 222,r'varience')

plt.show()
print('Varience value of feature 76 of merged dataset for block feature vector is :',f76.var())
print('Varience value of feature 77 of merged dataset for block feature vector is :',f77.var())
```

```
Varience value of feature 76 of merged dataset for block feature vector is :
5430.534411282076
Varience value of feature 77 of merged dataset for block feature vector is :
5394.139711489578
```

Figure: plotting the varience for feature ‘76’ for merged dataset of block feature vectors.

The above figure and code is for plotting the varience for feature ‘76’ of the training dataset of block feature vectors of three fruits. Varience for the feature ‘76’ is 5430 and for feature ‘77’ is 5394. Varience is given in the red dotted lines in the above plot where histogram values are in black.

Generating Scatter Plots:

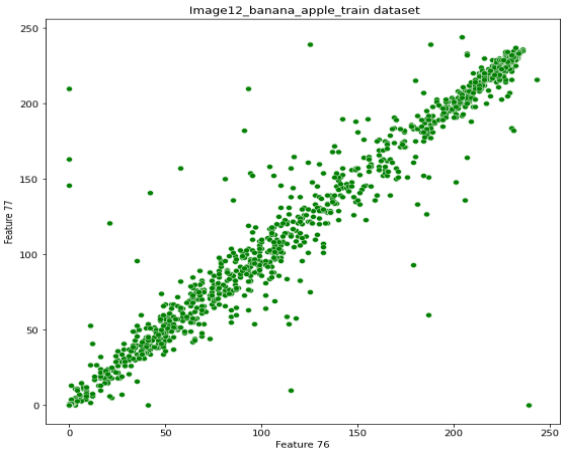


Figure58: Scatter plot for training dataset of block feature vectors for two fruits (banana and apple) for features ‘76’ and ‘77’.

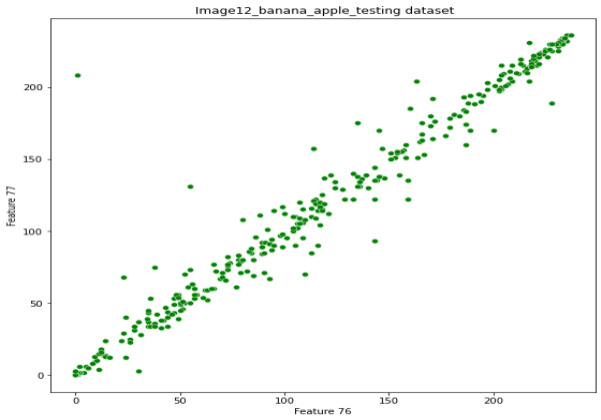


Figure59: Scatter plot for testing dataset of block feature vectors for two fruits (banana and apple) for features ‘76’ and ‘77’.

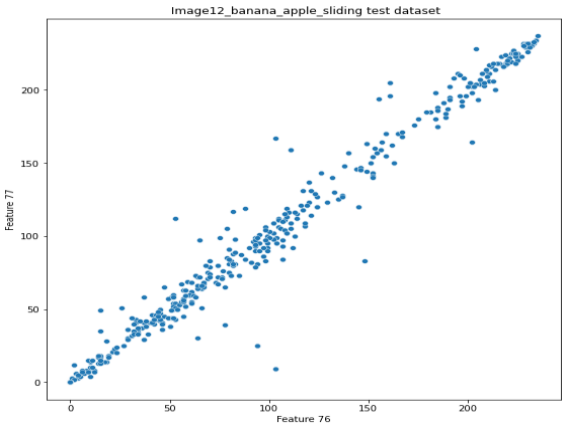


Figure60: Scatter plot for testing dataset of sliding block feature vectors for two fruits (banana and apple) for features ‘76’ and ‘77’.

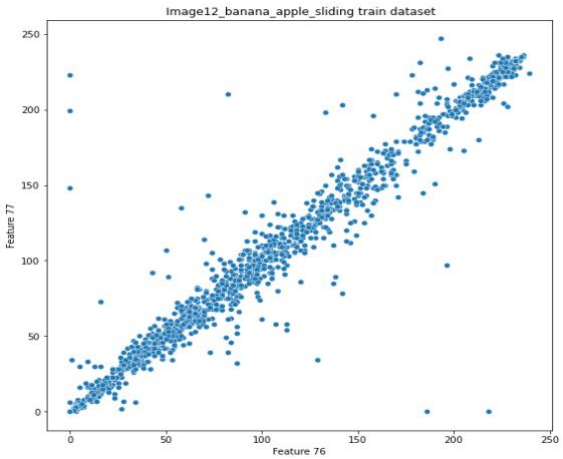


Figure61: Scatter plot for training dataset of sliding block feature vectors for two fruits (banana and apple) for features ‘76’ and ‘77’.

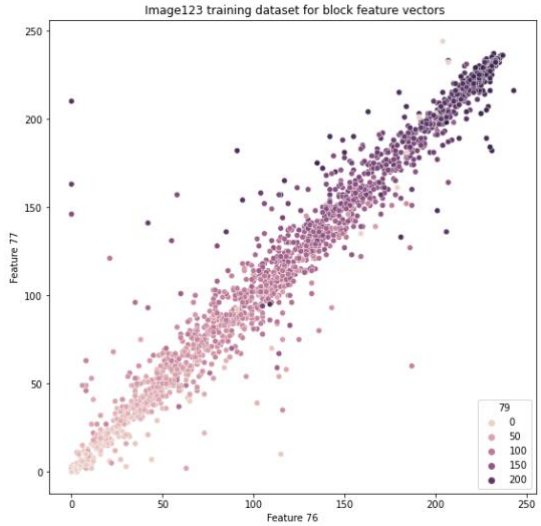


Figure62: Scatter plot for training dataset of block feature vectors for three fruits (banana and apple and guava) for features ‘76’ and ‘77’.

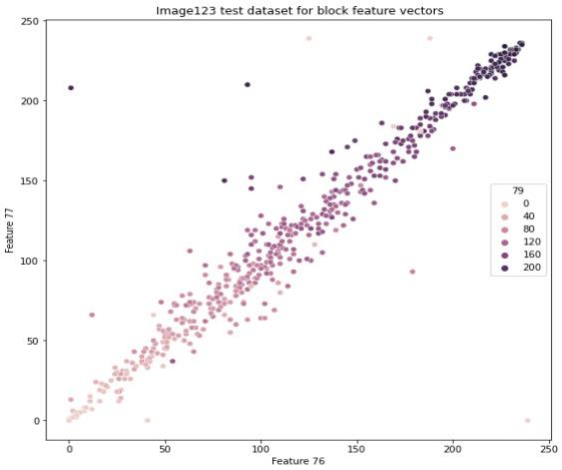


Figure63: Scatter plot for testing dataset of block feature vectors for three fruits (banana and apple and guava) for features ‘76’ and ‘77’.

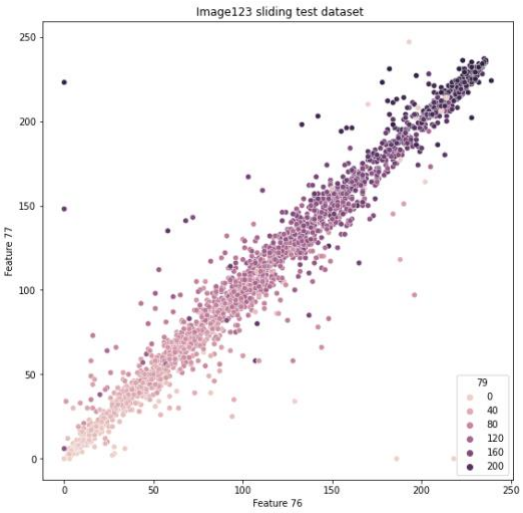


Figure64: Scatter plot for training dataset of sliding block feature vectors for three fruits (banana and apple and guava) for features ‘76’ and ‘77’.

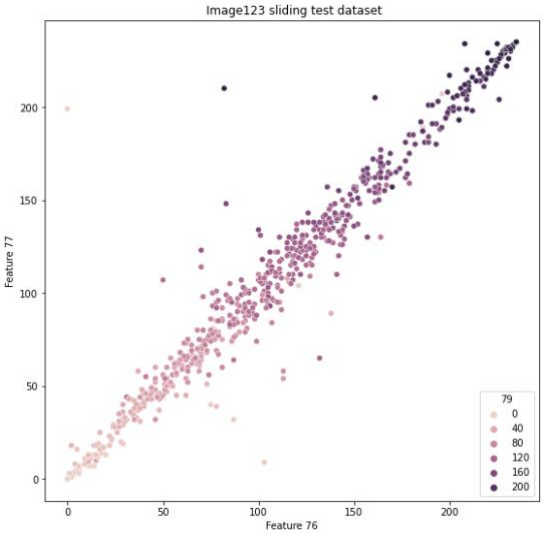


Figure65: Scatter plot for testing dataset of sliding block feature vectors for three fruits (banana and apple and guava) for features ‘76’ and ‘77’.

Figures 58-65 shows the scatter plots for block feature vectors and sliding block feature vectors for features ‘76’ and ‘77’ for two fruits and also for three fruits.

```
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
df=pd.read_csv('train_Image123_merged.csv')
f61 = df['76']
f62 = df['77']
label = df['79']


fig = plt.figure(figsize = (20 , 20))
ax1 = fig.add_subplot(221)
ax1.set_title(" Image123 training dataset for block feature vectors")
ax1.set_xlabel("Feature 76")
ax1.set_ylabel("Feature 77")
sns.scatterplot( x=f61 , y= f62 , hue= label, color='green')
plt.show()
```

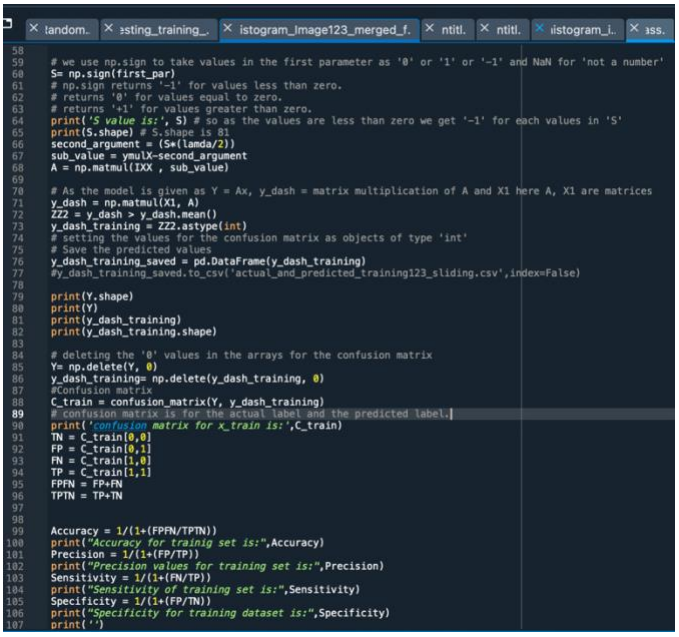
Figure 66: code for plotting scatter plot for two features.

Task 2:

Step I: Implementing a regression-based model

I implemented Lasso Regression

```
#read the training and the testing dataset
x_train = pd.read_csv('train_Image123_sliding_merged.csv' , header = None)
print('initial shape of x_train is :',x_train.shape)
x_train = x_train.drop(0, axis=1)
print('shape of x_train after dropping unwanted 0th column :',x_train.shape)
print(x_train)

# train_Image123_sliding_merged.csv has 82 columns(0-81) including the indexing column
x_test = pd.read_csv('testing_Image123_sliding_merged.csv',header=None)
print('shape of x_test is :',x_test.shape)
x_test = x_test.drop(0, axis=1)
print('shape of x_test after dropping the index is :',x_test.shape)

#read the label and store in y
y = x_train[82] # last but one column is 81
Y = np.array(y)
print('array of Y is:',Y) #Y is array of last column
print('shape of Y is:', Y.shape)

#drop the label column
x_train.drop(82,axis=1,inplace=True)
print('shape of x_train is :',x_train.shape)
X = x_train
X1 = np.array(X)
#lambda value is set to 0.01
lamda = 0.01
print('x_train shape is :',x_train.shape)
# X' is the transpose of X
X2 = X1.transpose()
print('shape of X2 is:', X2.shape)
#XX transpose is by multiplying X1 and X2
XX_transpose = np.matmul( X2 , X1)
#shape of XX after multiplying X1 and X2 is (81,81)
print('shape of XX_transpose is :',XX_transpose.shape)
# [XX']inverse is IXX
IXX = inv(XX_transpose)
print('shape of inverse of XX_transpose is:',IXX.shape)
#Ydash = Y.transpose()
ymulX = np.matmul(X2, Y)
print(ymulX.shape)
# XX_transpose is 81 x 81, ymulx is 81
first_par = np.matmul(ymulX,IXX)
print('first parameter value is :', first_par)
```



```
# we use np.sign to take values in the values for parameter as '0' or '1' or '-1' and NaN for 'not a number'
S= np.sign(first_par)
# np.sign returns '-1' for values less than zero.
# returns '0' for values equal to zero.
# returns '+1' for values greater than zero.
print('S value is:', S) # so as the values are less than zero we get '-1' for each values in 'S'
print(S.shape) # S.shape is 81
second_argument = (S*(lamda/2))
sub_value = ymulX-second_argument
A = np.matmul(IXX , sub_value)

# As the model is given as Y = Ax, y_dash = matrix multiplication of A and X1 here A, X1 are matrices
y_dash = np.matmul(X1, A)
ZZ2 = y_dash > y_dash.mean()
y_dash_training = ZZ2.astype(int)
# setting the values for the confusion matrix as objects of type 'int'
# Save the predicted values
y_dash_training_saved = pd.DataFrame(y_dash_training)
#y_dash_training_saved.to_csv('actual_and_predicted_training123_sliding.csv',index=False)

print(Y.shape)
print(Y)
print(y_dash_training)
print(y_dash_training.shape)

# deleting the '0' values in the arrays for the confusion matrix
Y= np.delete(Y, 0)
y_dash_training= np.delete(y_dash_training, 0)
#Confusion matrix
C_train = confusion_matrix(Y, y_dash_training)
# confusion matrix is for the actual label and the predicted label.
print('confusion matrix for x_train is:',C_train)
TN = C_train[0,0]
FP = C_train[0,1]
FN = C_train[1,0]
TP = C_train[1,1]
FPFN = FP+FN
TPTN = TP+TN


Accuracy = 1/(1+(FPFN/TPTN))
print("Accuracy for trainig set is:",Accuracy)
Precision = 1/(1+(FP/TP))
print("Precision values for training set is:",Precision)
Sensitivity = 1/(1+(FN/TP))
print("Sensitivity of training set is:",Sensitivity)
Specificity = 1/(1+(FP/TN))
print("Specificity for training dataset is:",Specificity)
print('')
```

Figure 67 : code for Lasso regression

I used the formula of lasso regression from the text book.

$$A = \left(y\mathbf{x}' - \frac{\lambda}{2}\mathbf{s} \right) (\mathbf{x}\mathbf{x}')^{-1}$$

Step II :

Applying the predicted label generated from lasso regression to the training and testing datasets of overlapping and non-overlapping block feature vectors. Figures 68 and 69 shows the Datasheets with included 83rd column which is the predicted label.

	BM	BN	BO	BP	BQ	BR	BS	BT	BU	BV	BW	BX	BY	BZ	CA	CB	CC	CD	CE	CF
1	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83
2	203	196	195	188	193	185	188	188	191	203	206	204	190	187	182	187	185	181	2	0
3	205	196	197	193	191	186	182	173	171	201	200	197	194	187	181	169	174	170	0	1
4	128	133	140	141	142	132	114	121	118	123	132	126	122	132	119	122	123	114	2	1
5	68	73	74	76	77	73	75	68	71	71	77	80	81	77	76	77	73	68	1	1
6	77	66	65	82	92	104	115	135	146	81	70	52	72	90	122	107	117	136	2	0
7	172	171	167	161	129	141	165	166	181	172	170	167	151	164	167	167	169	169	2	1
8	195	149	104	199	208	208	211	215	220	191	165	110	196	206	210	205	216	218	0	1
9	75	75	85	86	92	84	79	90	93	74	78	83	82	84	84	82	87	89	1	1
10	196	196	188	195	203	204	205	215	218	195	198	202	201	205	209	210	214	214	0	0
11	169	171	161	161	160	162	155	159	160	162	167	160	166	160	155	147	158	151	2	1
12	55	55	56	59	49	59	64	50	54	52	47	47	51	49	52	49	46	50	1	1
13	72	75	72	87	95	99	82	96	98	74	74	74	67	85	84	83	87	88	2	0
14	13	1	3	3	3	3	3	5	4	20	5	3	3	6	5	3	4	6	2	1
15	105	110	109	102	102	97	94	91	80	105	111	110	105	101	95	95	93	85	1	0
16	100	106	102	115	122	104	122	115	120	111	116	117	121	123	120	116	116	126	1	0
17	57	55	56	51	49	45	45	39	39	54	53	54	51	48	50	43	41	37	1	1
18	95	82	80	108	108	114	121	119	128	113	69	89	98	104	110	123	128	143	2	0
19	90	91	84	92	91	90	93	98	86	86	92	86	89	94	92	94	93	82	2	1
20	223	225	220	224	222	224	232	229	215	226	226	228	226	226	230	227	228	229	0	0
21	79	76	73	70	65	64	63	63	59	58	65	61	52	55	59	64	65	58	1	1
22	121	121	119	122	128	131	134	134	131	120	128	124	119	139	148	145	145	138	1	0
23	68	62	65	33	16	24	21	20	20	69	69	58	51	22	28	26	26	27	1	1
24	107	107	105	105	107	106	103	107	120	101	106	104	113	112	106	111	115	114	1	0
25	134	130	124	117	109	104	100	93	93	140	137	132	121	114	103	109	102	92	1	1
26	127	129	129	126	113	111	113	121	114	136	130	126	125	111	109	111	122	113	1	1
27	60	41	13	11	12	10	7	7	1	55	56	32	11	10	9	7	2	2	1	0
28	45	59	66	63	61	62	70	74	63	43	42	48	55	58	61	66	65	68	1	0
29	73	75	76	82	80	79	62	56	63	74	75	75	77	72	68	59	53	64	1	0
30	91	97	103	102	105	111	116	117	119	89	95	98	99	109	111	108	106	99	1	0
31	150	145	141	147	167	162	151	150	154	155	162	152	157	175	163	147	158	153	2	0
32	176	176	165	170	166	166	157	146	133	171	173	179	169	173	164	159	145	134	2	1
33	160	171	170	167	162	163	145	136	140	158	173	177	173	171	171	151	143	157	1	1
34	209	214	212	219	216	220	221	225	226	210	209	212	213	215	218	218	222	223	0	1

Figure 68: Datasheet of sliding block feature vectors(training set) of three fruit images, Columns 82 and 83 are for actual and predicted labels respectively

69	70	71	72	73	74	75	76	77	78	79	80	81	82	83						
122	134	130	130	102	120	109	113	110	104	129	144	130	1	0						
85	84	74	86	81	83	81	88	88	77	78	79	73	2	1						
40	42	44	45	42	42	40	42	43	43	44	43	43	2	0						
61	61	61	67	86	82	66	53	46	60	51	49	60	2	0						
93	92	92	91	102	105	106	96	98	87	93	92	103	2	0						
79	69	69	86	81	70	66	71	85	84	97	100	86	2	0						
132	121	130	122	40	27	85	115	123	118	122	116	124	2	0						
1	1	1	1	5	3	2	1	0	0	2	0	0	1	1						
220	225	226	227	221	220	222	224	223	220	222	225	226	0	0						
30	29	21	24	97	74	73	70	73	51	24	26	24	2	1						
229	229	227	228	225	228	229	229	229	229	228	229	228	0	0						
187	179	161	165	168	164	158	165	174	167	174	173	176	2	1						
119	113	107	111	99	106	105	105	106	114	112	110	114	1	1						
40	39	37	38	46	46	43	42	40	41	44	47	46	1	1						
139	129	124	133	148	156	137	122	123	129	128	133	137	1	0						
155	157	157	152	131	137	132	138	134	134	150	150	151	1	0						
41	45	37	47	33	35	48	47	46	52	42	44	59	1	1						
144	136	146	148	114	141	153	159	163	171	162	173	158	1	0						
105	97	72	29	131	114	122	107	116	106	101	67	31	2	0						
202	204	0	0	215	215	208	197	196	207	0	0	0	0	0						
42	49	36	43	34	40	39	33	37	37	46	39	39	1	1						
191	191	195	192	0	0	0	176	185	192	191	193	178	0	0						
144	137	143	190	170	163	151	149	152	137	149	142	164	0	0						
76	83	81	84	76	77	85	78	77	80	80	82	82	1	1						
38	29	27	24	48	40	35	38	38	31	27	27	25	1	0						
118	114	115	122	81	73	71	106	109	109	108	114	120	2	0						
94	84	86	85	73	78	95	112	98	86	84	87	87	1	1						
196	191	225	0	222	221	215	213	205	193	194	227	0	0	0						
165	152	112	108	190	182	179	174	167	158	146	105	103	0	0						
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1						
204	200	200	201	207	203	209	207	209	206	208	209	202	1	0						
214	214	213	213	220	212	213	216	210	213	216	212	217	0	1						
217	215	212	214	212	215	215	219	218	216	215	214	211	0	1						

Figure 69: Datasheet of sliding block feature vectors(testing set) of three fruit images, Columns 82 and 83 are for actual and predicted labels respectively.
Index start from '1' in the above dataset.

Similarly I have added the predicted values column to the training and testing datasets of the block feature vectors i.e. non- overlapping features(not included in this presentation).

Step III and Step IV: Confusion Matrix and Quantitative Measures :

```
#Confusion matrix
C_train = confusion_matrix(Y, y_dash_training)
TN = C_train[0,0]
FP = C_train[0,1]
FN = C_train[1,0]
TP = C_train[1,1]
FPFN = FP+FN
TPTN = TP+TN


Accuracy = 1/(1+(FPFN/TPTN))
print("Accuracy for trainig set is:",Accuracy)
Precision = 1/(1+(FP/TP))
print("Precision values for training set is:",Precision)
Sensitivity = 1/(1+(FN/TP))
print("Sensitivity of training set is:",Sensitivity)
Specificity = 1/(1+(FP/TN))
print("Specificity for training dataset is:",Specificity)
print('')
print('')
```

Figure70: Confusion Matrix for actual label and predicted label.(82nd and 83rd column). Output of confusion matrix is given in the figure 73.

```
Accuracy for trainig set is: 0.2821881254169446
Precision values for training set is: 0.400974025974026
Sensitivity of training set is: 0.2589098532494759
Specificity for training dataset is: 0.3229357798165138


(665, 1)
Accuracy for testing set is: 0.34216867469879514
Precision values for testing set is: 0.5581395348837209
Sensitivity of testing set is: 0.32764505119453924
Specificity for testing dataset is: 0.3770491803278688
```

Figure71: Output accuracy values of train and testing datasets for sliding block feature vectors of three fruit images along with Quantitative measures- sensitivity, specificity and precision.

```
Accuracy for trainig set is: 0.28365019011406845
Precision values for training set is: 0.23857868020304568
Sensitivity of training set is: 0.22274881516587677
Specificity for training dataset is: 0.34017595307917886


(556, 1)
Accuracy for testing set is: 0.2343324250681199
Precision values for testing set is: 0.27607361963190186
Sensitivity of testing set is: 0.21634615384615385
Specificity for testing dataset is: 0.2578616352201258
```

Figure72: Output accuracy values of train and testing datasets for block feature vectors of three fruit images along with Quantitative measures- sensitivity, specificity and precision. Figure76 gives the values for Overlapping feature vectors.

Figure 70 gives the code for constructing the confusion matrix for the actual and predicted labels. Figures 71 and 72 gives the Qualitative measures which use the confusion matrix for estimating the performance of the machine learning model.

We can compare the performance of all types of models with these Quantitative measures. From figure 70, we can say that the Overall performance of the test dataset is more than that of the train dataset for the overlapping feature vectors, Whereas for the non-Overlapping features dataset the accuracy for training dataset is more than that of the testing Dataset.

```
In [12]: C_test
Out[12]:
array([[ 58, 121],
       [138,  53]])

In [13]: C_train
Out[13]:
array([[230, 432],
       [441, 209]])
```

Figure73: Confusion Matrix values for block feature vectors for merged dataset of two fruits (banana and apple). We have two labels 0 and 1.

```
Accuracy for trainig set is: 0.3346036585365854
Precision values for training set is: 0.32605304212168484
Sensitivity of training set is: 0.32153846153846155
Specificity for training dataset is: 0.34743202416918434


(370, 1)
Confusion Matrix is : [[ 58 121]
 [138  53]]
Accuracy for testing set is: 0.3
Precision values for testing set is: 0.30459770114942525
Sensitivity of testing set is: 0.2774869109947644
Specificity for testing dataset is: 0.3240223463687151
```

Figure 74: Quantitative measures for 2 fruits merged dataset for non-overlapping features (banana and apple)

```
In [22]: C_train
Out[22]:
array([[176, 369,   0],
       [707, 247,   0],
       [335, 518,   0]])
```

Figure 75: Confusion matrix of training dataset for 3 fruits(we have three labels 0, 1, 2) merged dataset for overlapping feature vectors. (Not included the matrix for test dataset).

Confusion Matrix is for actual label and the predicted label(predicted label is from the lasso regression). I have included the Confusion Matrix output for Random Forest Classifier later in this document.

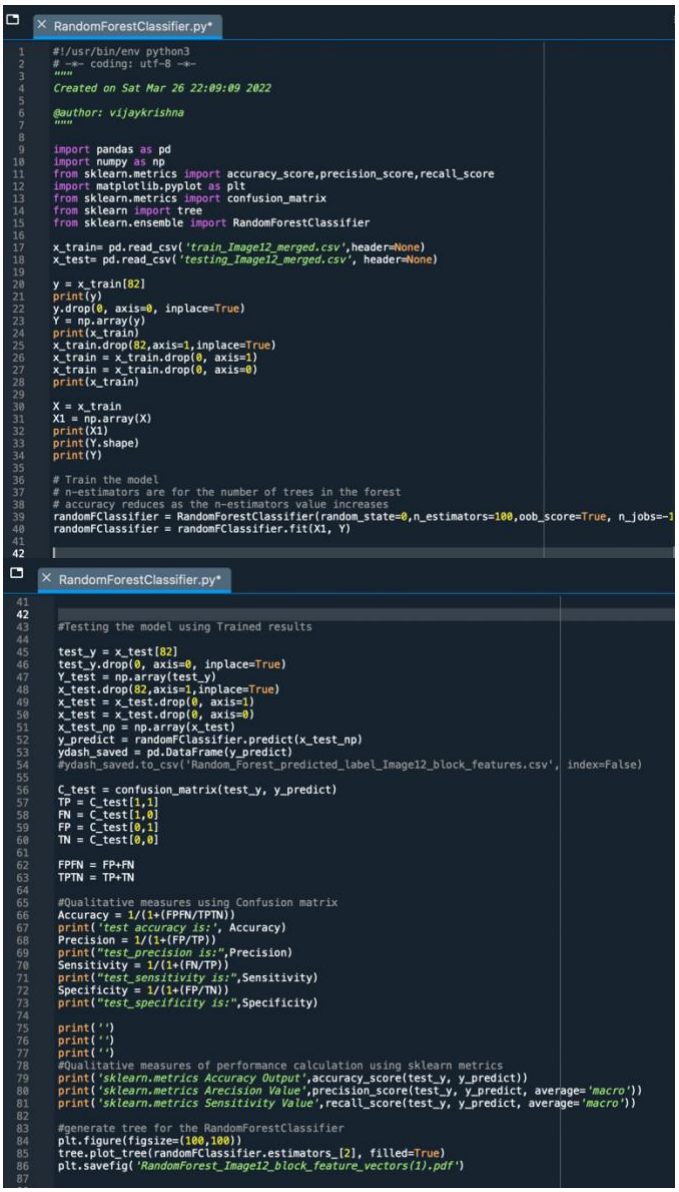
```
Accuracy for trainig set is: 0.2821881254169446
Precision values for training set is: 0.400974025974026
Sensitivity of training set is: 0.2589098532494759
Specificity for training dataset is: 0.3229357798165138


Accuracy for testing set is: 0.34216867469879514
Precision values for testing set is: 0.5581395348837209
Sensitivity of testing set is: 0.32764505119453924
Specificity for testing dataset is: 0.3770491803278688
```

Figure76:The Qualitative measures for three fruits merged dataset(Overlapping block feature vectors)

Task 3:

Implementing random forest or deep learning:

Step I : I Implemented The Random Forest Classifier



```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Sat Mar 26 22:09:09 2022

@author: vijaykrishna
"""

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

x_train= pd.read_csv('train_Image12_merged.csv',header=None)
x_test= pd.read_csv('testing_Image12_merged.csv', header=None)

y = x_train[82]
print(y)
y.drop(0, axis=0, inplace=True)
Y = np.array(y)
print(x_train)
x_train.drop(82,axis=1,inplace=True)
x_train = x_train.drop(0, axis=1)
x_train = x_train.drop(0, axis=0)
print(x_train)

X = x_train
X1 = np.array(X)
print(X1)
print(Y.shape)
print(Y)

# Train the model
# n-estimators are for the number of trees in the forest
# accuracy reduces as the n-estimators value increases
randomFClassifier = RandomForestClassifier(random_state=0,n_estimators=100,oob_score=True, n_jobs=-1
randomFClassifier = randomFClassifier.fit(X1, Y)
```

RandomForestClassifier.py*

```
#Testing the model using Trained results

test_y = x_test[82]
test_y.drop(0, axis=0, inplace=True)
Y_test = np.array(test_y)
x_test.drop(82,axis=1,inplace=True)
x_test = x_test.drop(0, axis=1)
x_test = x_test.drop(0, axis=0)
x_test_np = np.array(x_test)
y_predict = randomFClassifier.predict(x_test_np)
ydash_saved = pd.DataFrame(y_predict)
#ydash_saved.to_csv('Random_Forest_predicted_label_Image12_block_features.csv', index=False)

C_test = confusion_matrix(test_y, y_predict)
TP = C_test[1,1]
FN = C_test[1,0]
FP = C_test[0,1]
TN = C_test[0,0]

FPFN = FP+FN
TPTN = TP+TN

#Qualitative measures using Confusion matrix
Accuracy = 1/(1+(FPFN/TPTN))
print('test accuracy is:', Accuracy)
Precision = 1/(1+(FP/TP))
print("test_precision is:",Precision)
Sensitivity = 1/(1+(FN/TP))
print("test_sensitivity is:",Sensitivity)
Specificity = 1/(1+(FP/TN))
print("test_specificity is:",Specificity)

print('')
print('')
print('')
#Qualitative measures of performance calculation using sklearn metrics
print('sklearn.metrics Accuracy Output',accuracy_score(test_y, y_predict))
print('sklearn.metrics Precision Value',precision_score(test_y, y_predict, average='macro'))
print('sklearn.metrics Sensitivity Value',recall_score(test_y, y_predict, average='macro'))

#generate tree for the RandomForestClassifier
plt.figure(figsize=(100,100))
tree.plot_tree(randomFClassifier.estimators_[2], filled=True)
plt.savefig('RandomForest_Image12_block_feature_vectors(1).pdf')
```

Figure77: Code for Implementing Random Forest (Above code is for a two class classifier of non-overlapping datasets).

Step II , III and IV:

Figure 82 is for the Datasheet with predicted labels in the 83rd column. Figure 78 is construction of confusion matrix and Figure 79 is the Qualitative Measures.

```
In [39]: C_test
Out[39]:
array([[167,  12],
       [  9, 182]])
```

Figure80: Confusion Matrix for block feature vectors dataset for two images (banana, apple).

```
C_test = confusion_matrix(test_y, y_predict)
TP = C_test[1,1]
FN = C_test[1,0]
FP = C_test[0,1]
TN = C_test[0,0]

FPFN = FP+FN
TPTN = TP+TN

#Qualitative measures using Confusion matrix
Accuracy = 1/(1+(FPFN/TPTN))
print('test accuracy is:', Accuracy)
Precision = 1/(1+(FP/TP))
print("test_precision is:",Precision)
Sensitivity = 1/(1+(FN/TP))
print("test_sensitivity is:",Sensitivity)
Specificity = 1/(1+(FP/TN))
print("test_specificity is:",Specificity)
```

Figure78: Construction of Confusion Matrix for actual and predicted Label

```
Accuracy is: 0.9432432432432432
Test_Precision_Score: 0.9488636363636362
Test_Sensitivity_Score: 0.9329608938547487
Test_Specificity_Score: 0.9528795811518325
```

Figure79: Qualitative measures by implementing Confusion Matrix for a two class dataset of non-overlapping feature vectors.

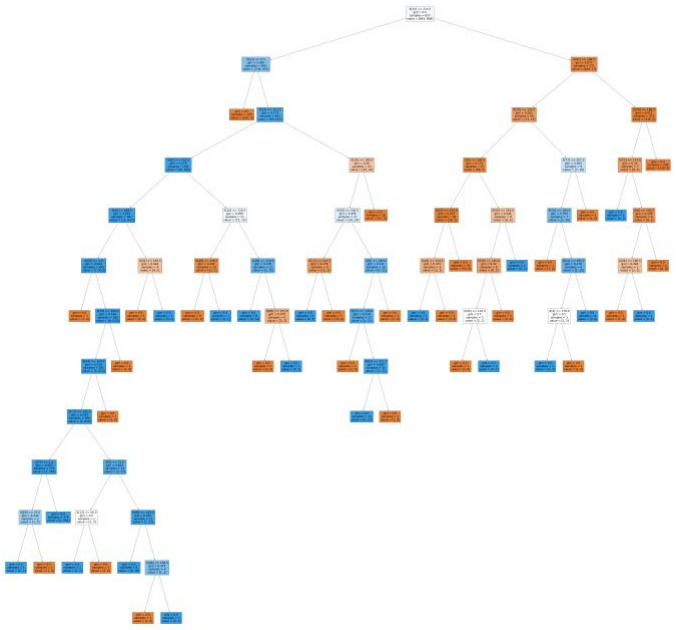


Figure 81(a): Random Forest tree for block feature vectors merged dataset of two images (banana, apple). i.e. two class classifier, n-estimators=50.

Accuracy of the dataset with RandomForestClassifier will be slightly reduced if the Random forest tree size increases. I used the function tree.plot_tree to show the tree plot.

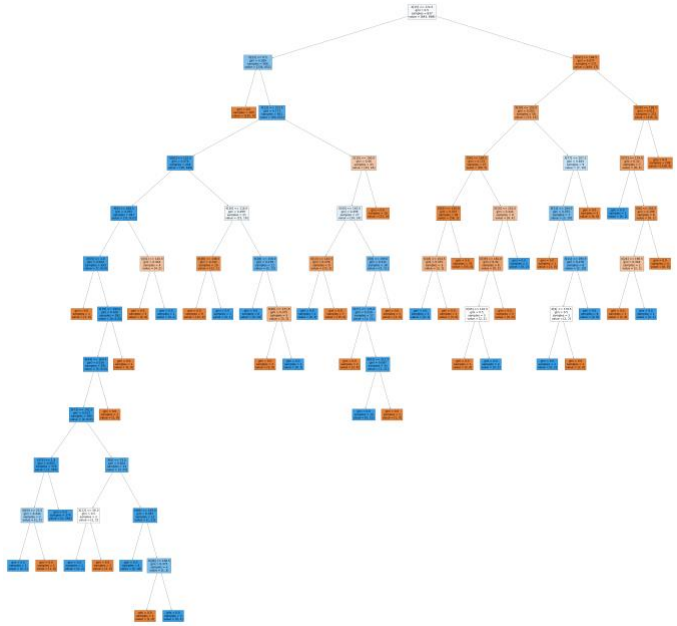


Figure 81(b): Random Forest tree for block feature vectors merged dataset of two images (banana, apple). i.e. two class classifier, n-estimators=100.

BT	BU	BV	BW	BX	BY	BZ	CA	CB	CC	CD	CE	CF
70	71	72	73	74	75	76	77	78	79	80	81	82	83
82	80	81	83	80	82	82	84	85	84	89	108	1	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0
74	171	169	187	180	185	180	181	181	180	169	162	0	0
00	102	111	113	114	132	125	116	114	108	101	101	1	1
28	229	231	229	233	232	231	228	189	154	220	232	0	0
32	229	220	226	228	222	228	230	230	227	227	229	0	0
49	163	156	128	142	154	127	153	151	135	156	152	1	0
26	131	116	204	187	183	163	155	139	132	129	118	1	0
96	203	204	0	151	149	145	170	180	183	193	198	0	0
13	216	211	218	216	215	216	215	211	212	211	209	0	0
33	232	233	232	233	232	233	233	234	233	233	233	0	0
45	46	49	32	36	35	34	38	75	43	45	55	1	1
13	118	119	52	51	61	70	113	110	108	118	128	1	1
10	10	11	7	10	14	7	10	10	14	9	10	1	1
37	38	38	36	39	42	39	44	44	35	47	37	1	1
35	235	234	228	231	236	234	236	236	235	236	235	0	0
82	107	119	189	152	60	54	52	70	105	112	122	0	1
33	110	134	131	152	150	138	133	140	109	113	103	1	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0
72	80	93	86	82	80	70	67	72	69	67	82	1	1
0	3	2	27	21	16	7	3	2	3	2	3	1	1
28	226	228	216	218	216	216	222	221	223	227	229	0	0
31	228	228	232	230	229	230	227	230	230	230	223	0	0
48	148	141	161	158	157	158	154	155	146	143	135	1	1
47	155	159	178	170	167	156	158	160	157	150	152	1	1
31	37	35	31	32	27	35	38	36	33	36	37	1	1
47	140	155	86	98	123	142	143	144	146	144	154	0	1
78	93	99	45	39	35	45	55	73	87	99	85	1	1
52	135	110	215	214	214	211	200	170	149	131	116	0	0
3	19	17	2	2	3	1	2	6	9	23	18	1	1
32	232	232	232	233	232	233	233	233	233	232	233	0	0
47	56	54	36	38	42	46	49	56	58	49	54	1	1
07	117	81	152	134	155	137	133	122	103	80	81	1	1
65	60	58	102	94	85	63	62	59	57	65	64	1	1

Figure82: Datasheet of block feature vectors of two images with 82nd column as actual label and 83rd label is the predicted label obtained by the RandomForestClassifier. We do the same and generate the datasets for block feature vectors and sliding block feature vectors of two and three classes.(I have not included all the datasets in this documentation).

```
test accuracy is: 0.976510067114094
test_precision is: 0.9862068965517242
test_sensitivity is: 0.9662162162162162
test_specificity is: 0.9866666666666666

In [4]: C_test
Out[4]:
array([[148,   2,   9],
       [  5, 143,  60],
       [ 21,  59, 108]])
```

Figure83: Confusion Matrix and Qualitative measures for test Dataset of three labels(0, 1, 2) of three fruits, So we got a matrix of 3*3. The predicted label is from the Random Forest Classifier method.

Task 4: Evaluation of the learning models

I compared the Qualitative values obtained by confusion matrix and the sklearn.metrics functions. Images 84-89 shows the values of Quantitative measures and sklearn metrics.

```
test accuracy is: 0.9432432432432432
test_precision is: 0.9381443298969072
test_sensitivity is: 0.9528795811518325
test_specificity is: 0.9329608938547487


sklearn.metrics Accuracy Output 0.9432432432432433
sklearn.metrics Arecision Value 0.9381443298969072
sklearn.metrics Sensitivity Value 0.9528795811518325
```

Figure84:Random Forest Qualitative measures for two labels dataset non-overlapping feature vectors.

```
Accuracy for testing set is: 0.3
Precision values for testing set is: 0.30459770114942525
Sensitivity of testing set is: 0.2774869109947644
Specificity for testing dataset is: 0.3240223463687151


sklearn.metrics Accuracy 0.3
sklearn.metrics precision 0.30025803424818204
sklearn.metrics sensitivity 0.30075462868173974
```

Figure85:Qualitative measures for Lasso regression for non-Overlapping dataset of two fruit classification (apple, banana). Above given values are for the testing Dataset.

```
Accuracy for trainig set is: 0.3346036585365854
Precision values for training set is: 0.32605304212168484
Sensitivity of training set is: 0.32153846153846155
Specificity for training dataset is: 0.34743202416918434


sklearn.metrics Accuracy 0.33460365853658536
sklearn.metrics precision 0.3344125121189646
sklearn.metrics sensitivity 0.3344852428538229
(370, 1)
```

Figure86:Qualitative measures for Lasso regression for Overlapping dataset of two fruit classification (apple, banana). Above given values are for the training Dataset.

```
Accuracy for trainig set is: 0.34762223710649703
Precision values for training set is: 0.49418604651162795
Sensitivity of training set is: 0.35196687370600416
Specificity for training dataset is: 0.3396584440227704


sklearn.metrics Accuracy 0.347622237106497
sklearn.metrics precision 0.3582731474794164
sklearn.metrics sensitivity 0.3458126588643873
```

```
Accuracy for testing set is: 0.30403800475059384
Precision values for testing set is: 0.4680851063829787
Sensitivity of testing set is: 0.31316725978647686
Specificity for testing dataset is: 0.2857142857142857


sklearn.metrics Accuracy 0.30403800475059384
sklearn.metrics precision 0.319879463062734
sklearn.metrics sensitivity 0.2994407727503813
```

Figure87:Qualitative measures for Lasso regression for Overlapping dataset of two fruit classification (apple, banana). Above given values are for the training Dataset.

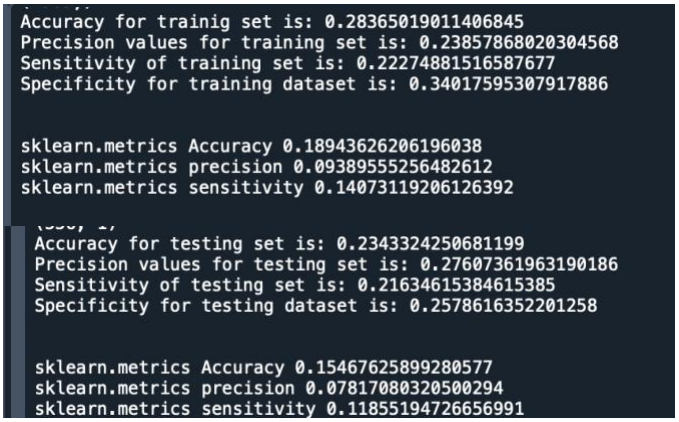


Figure88:Qualitative measures(by confusion matrix construction) and sklearn metrics for three class classifier(3 fruits- apple, banana and guava) for non-overlapping dataset.

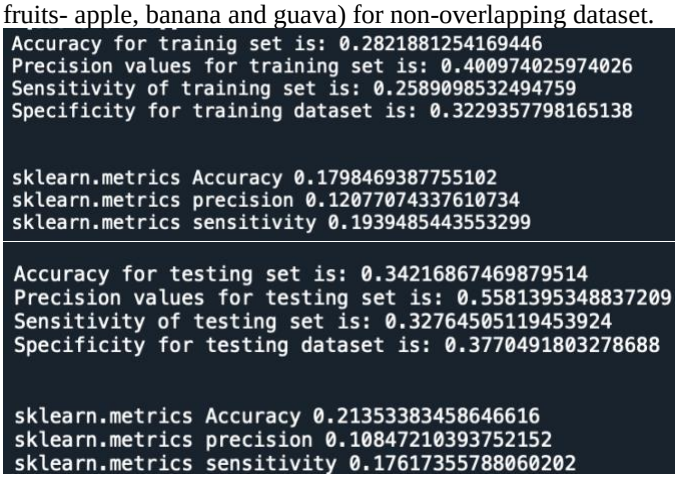


Figure89:Qualitative measures(by confusion matrix construction) and sklearn metrics for three class classifier(3 fruits- apple, banana and guava) for Overlapping dataset.

Figures 85-87 are for the lasso regression of two class classifier(banana and apple) for both the Overlapping and Overlapping datasets. Figures 88-89 are the lasso regression performance metrics for three class classifiers(banana , apple, guava) for non-Overlapping and Overlapping respectively.

For the two class classifier (banana and apple) the performance metrics and Qualitative measures obtained from confusion matrix are almost similar, But for the three class classifier the sklearn metrics values greatly reduces in comparision to the values obtained from the confusion matrix.

As we have Three classes (0, 1, 2) in the actual labels and we compare the actual labels with only two labels (0, 1) in the predicted label, the direct sklearn metrics values are greatly reduced in accuracy, precision and specificity, So we have highly reduced values for the sklearn.metrics for three class classifier.

For the two class classifier we have 0 and 1 in the actual label and also 0 and 1 in the predicted label we get similar results of accuracy, specificity and precision for both confusion matrix measured values and sklearn metric values.

If we compare the values which we got from Random Forest classifier output for predicted labels and the actual labels we get almost similar results for the quantitative measures(confusion matrix measures) and as well as for the sklearn metrics.

I am not including all the outputs of the random forest classifier as we are getting similar Qualitative measures and sklearn metrics results for all the two and three label classification. For the Random Forest Classification always the Accuracy, Precision, and sensitivity is for more better than that of the Lasso regression.

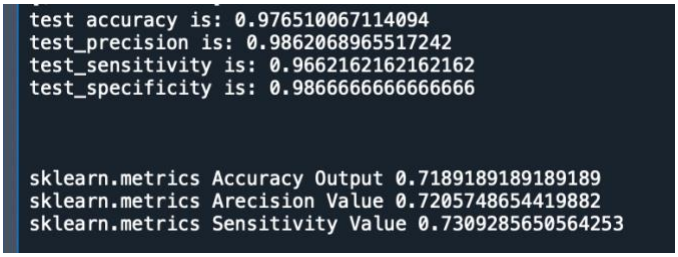


Figure 90:Random forest classifier for training dataset of 3 class classifier of non-overlapping block feature vectors.

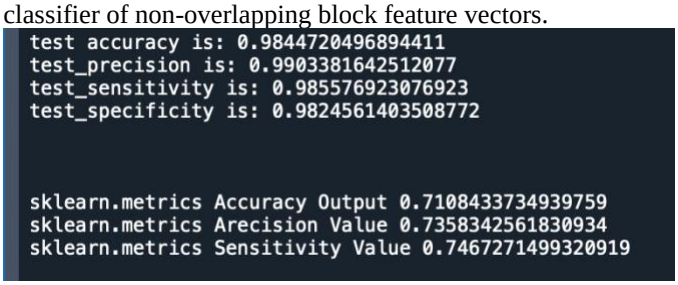


Figure91:Random forest classifier for training dataset of 3 class classifier of Overlapping block feature vectors.

Model	sklearn	Confusion matrix based	sklearn	Confusion matrix based	sklearn	Confusion matrix based
	Accuracy	Accuracy	Precision	Precision	Sensitivity	Sensitivity
Lasso (2) overlap	0.3346	0.3346	0.3344	0.3260	0.3344	0.3215
Lasso (2) non-overlap	0.3	0.3	0.3045	0.3002	0.300	0.277
Lasso (3) overlap	0.1798	0.2821	0.1207	0.4009	0.1939	0.2589
Lasso (3) non-overlap	0.1546	0.2343	0.0781	0.2760	0.118	0.2578
RF (2) overlap	0.952	0.952	0.9407	0.9466	0.955	0.964
RF (2) non-overlap	0.9432	0.9432	0.9435	0.9528	0.9429	0.9326
RF (3) overlap	0.7108	0.9844	0.735	0.9903	0.7467	0.9824
RF (3) non-overlap	0.7189	0.9765	0.7205	0.9862	0.7309	0.9662

Figure92: Table comparing the values of sklearn metrics and values of confusion matrix metrics. I have only tabulated the values for testing datasets of Lasso regression and Random Forest Classifier.

In the above figure I tried to tabulate the measures of Accuracy, Precision, Sensitivity for both the sklearn metrics and as well as the outputs by using the confusion matrix.

We have done the lasso regression for a two class classifier and so the predicted label is designed as either '0' or '1'. So for the two labels dataset the values of Accuracy, Precision and Sensitivity range around 30 percent[Both for metrics based as well as the confusion matrix based approaches]. As we do the lasso regression for the three label dataset with two label predicted dataset the metrics values greatly reduces which I tabulated above.

For the RandomForestClassifier the Qualitative Measures are really good for a two class classifier (for both sliding and block feature vectors). The values are above 90 percent for both Qualitative measures based or for the Confusion Matrix based. But again the values greatly reduces for a three class classifier as the predicted label we generate is only a for two class classifier.

According to me RandomForestClassifier has better Accuracy, Precision and Sensitivity both for three class or two class classifier. For the three class classifier in the random forest we get a high difference in values for confusion matrix based and relatively low values when used the sklearn measures. But when it comes to Lasso regression the overlapping features dataset of two class classifier has better outputs than compared to all the other measures which were considered in this project.

Assignment 3: Classifiers on a Conceptualized Big Data System

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

Completed all the tasks of assignment 1 and 2. Using the code from assignment 1, I generated datasets for 10 images and created block feature vectors for all the Images selected.
All the datasets generated are saved in .csv files
Each image has 81 features and 841 blocks in its dataset (all images resized to 261 x 261). So created a Bigdata file merging all the 10 csv files and randomised the rows.
I have taken 5 images of banana and 5 images for apple (again banana datasets are labelled as '0' and apple datasets are labelled as '1'). The Bigdata csv file has the rows randomised while merging all the 10 files.

Completed the Random Forest Algorithm on the Bigdata created with '10' fruit images(as mentioned above).
In this RandomForestClassifier, I split the whole data into Training and testing just like that we did in the assignment 2 (78% for training and 22% for testing).
I implemented the same code which I used in assignment 2 for the random forest classifier

```
test accuracy is: 0.7048648648648648
test_precision is: 0.7973421926910299
test_sensitivity is: 0.5309734513274337
test_specificity is: 0.8710359408033826



sklearn.metrics Accuracy Output 0.7048648648648649
sklearn.metrics Precision Value 0.7287993014737201
sklearn.metrics Sensitivity Value 0.7010046960654082

In [16]: C_test
Out[16]:
array([[824, 122],
       [424, 480]])
```

Figure 93: output from the random forest classifier performed on a big data. test-train split →78% and 22%
(used 10 images of fruits).
[used the n-estimators = 1000]

Task 2:

Validation of a model can be defined as testing the model on multiple combinations of training and test datasets. The cross-validation approach can be implemented on the unseen data for the testing of the models. N-fold cross validation can be called as a block- based circular shift algorithm. I have taken '10' fruit images for this assignment and the dataset of each image can be considered as fold i.e. a subset of big data

In the first approach entire dataset is divided into 10 equal sub dataframes (In the project we merge 10 datasets to make it a Bigdata and I directly considered the initial datasets for cross validation). As the data in the csv file is already randomized I splitted the data into 10 Dataframes. So each dataset can be called as folds (fold 1…..fold10). First step is to use the first 9 folds as train and 10th fold as test. In the second step 9th fold is used for testing and remaining folds are in the training. 9th fold was seen in the first step and in the second step it is unseen. Hence the test set consists of both seen and unseen data. This process is continued for all the 10 combinations of training and testing data.
This technique is similar to Leave-One-Out cross validation technique as of 8.2.2 in page 193 of the textbook. So, in this assignment I created '10' train and '10' test datasets which I later use them to perform RandomForest Classification and print out the time taken to run it and accuracies along with confusion matrix.
In our assignment as I take a 10-fold cross validation(each time 90% unseen and 10% seen data) so after 10-folds all the unseen data is changed to seen data.
If the 'n' value is selected a '5' then for 5-folds we might have a considerable effect of unseen data problem. As I split the Bigdata into 10 splits, I use 10-fold cross validation.

Taken the Bigdata I split the dataset into '10' equal parts/splits and then used the 10-fold cross validation technique as mentioned above. Created '10' training and '10' testing datasets. (here all the datasets are of 2 labels and the rows/vectors randomised)

Created 10 folds of datasets by the concept of figure 8.6 and done the 10 fold cross validation. I plotted the histogram, mean and variance for each testing and training datasets generated from cross validation. In total I generated 10 training and 10 testing datasets.

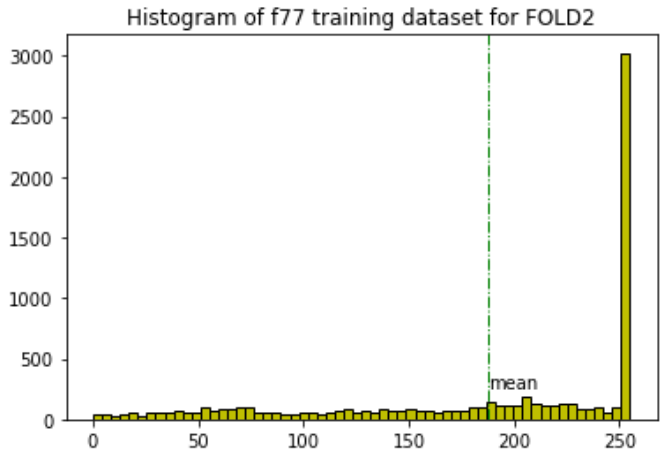


Figure94: Histogram of f77 for training dataset for fold2

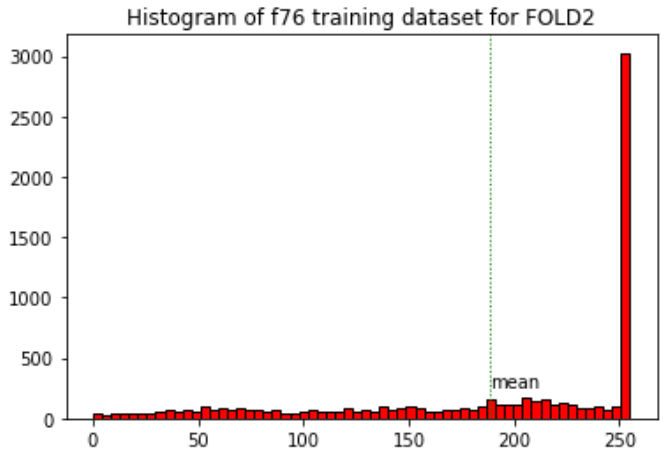


Figure95: Histogram of f76 for training dataset for fold2

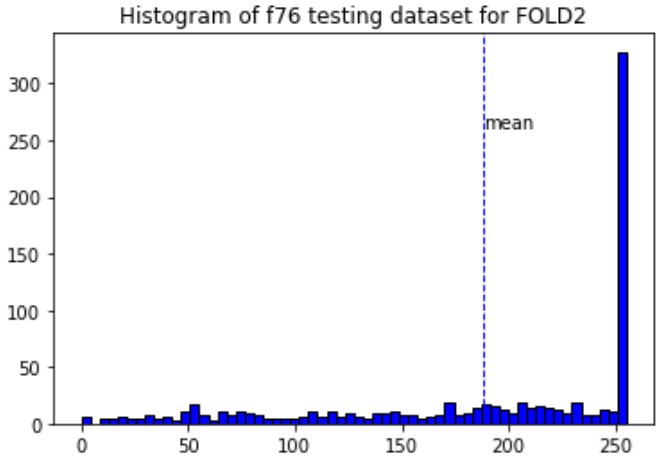


Figure96: Histogram of f76 for testing dataset for fold2

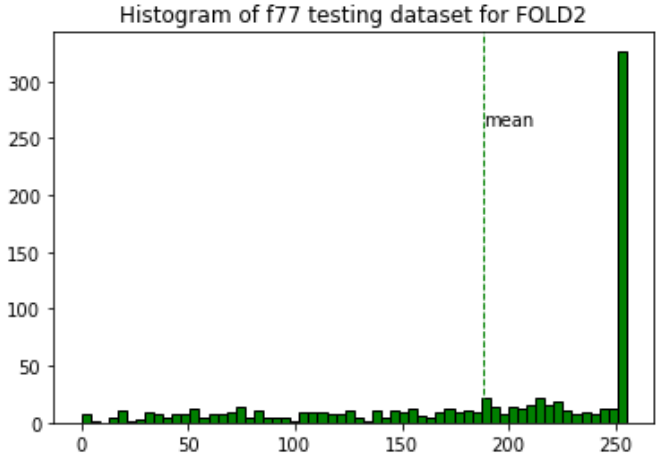


Figure97: Histogram of f77 for testing dataset for fold2

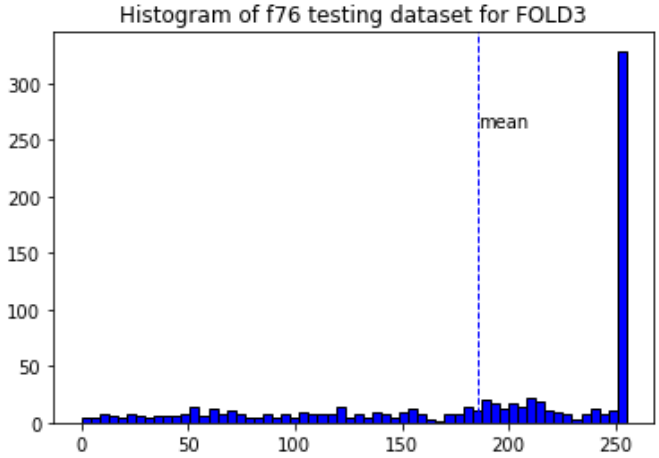


Figure98: Histogram of f76 for testing dataset for fold3

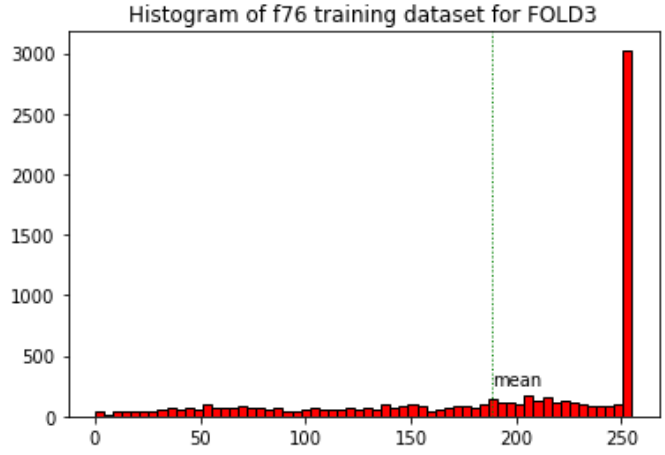


Figure99: Histogram of f76 for training dataset for fold3

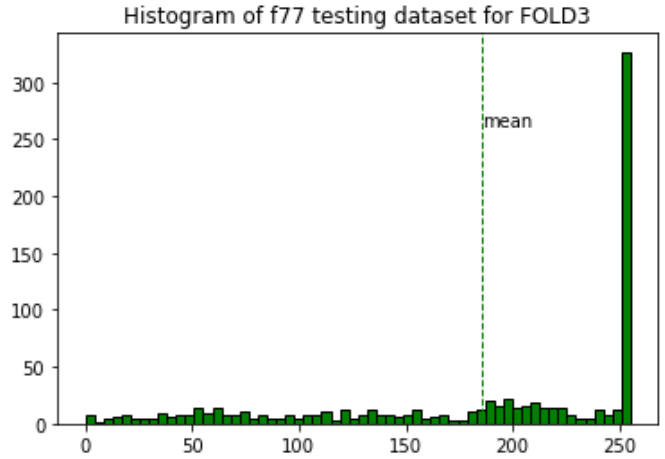


Figure100: Histogram of f77 for testing dataset for fold3

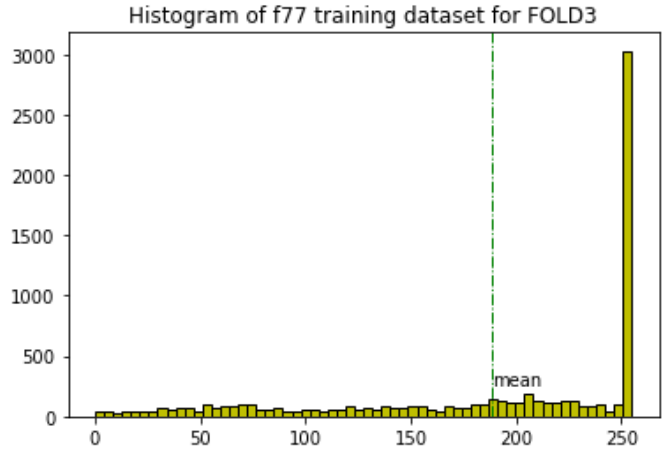


Figure101: Histogram of f77 for training dataset for fold3

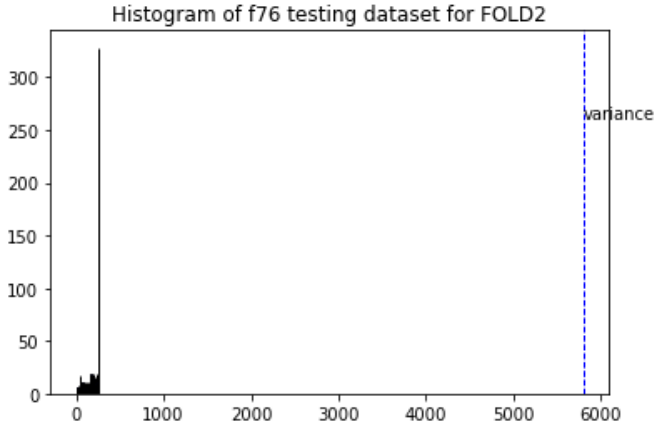


Figure102: Variance of f76 for testing dataset for fold2

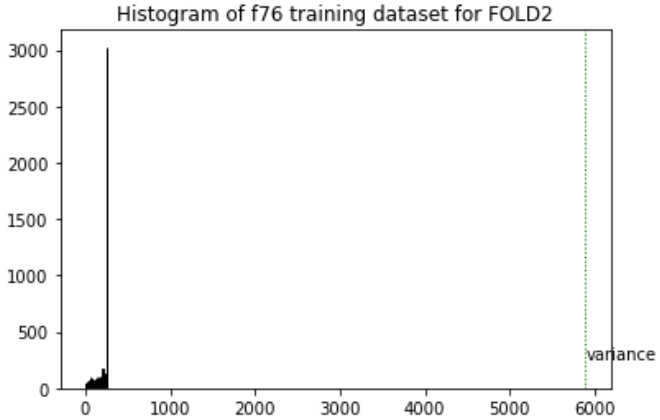


Figure103: Variance of f76 for training dataset for fold2

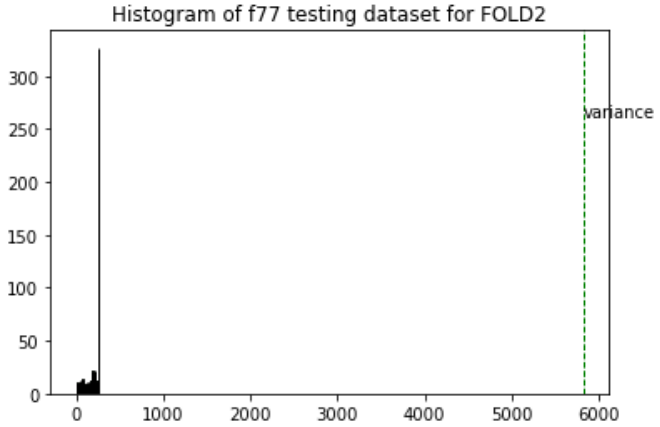


Figure104: Variance of f77 for testing dataset for fold2

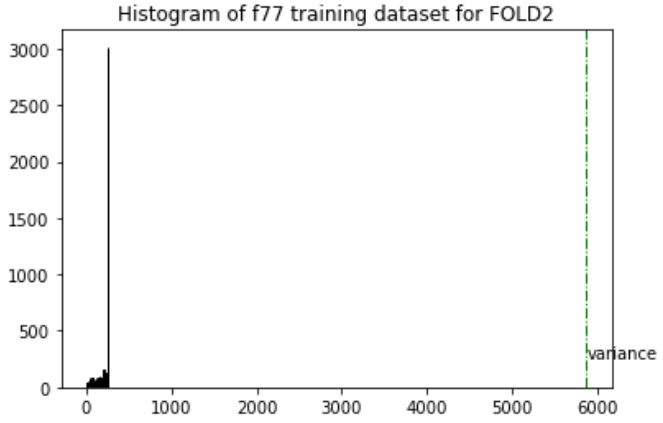


Figure105: Variance of f77 for training dataset for fold2

Created the data frame for each node and saved it as a spreadsheet(csv file) for all the nodes. I have attached all the csv files in the data folder.

Task 3 : Machine Learning tasks

I have taken two fruits (banana and apple). Banana label='0' and apple label = '1'. I have taken five images each and created the non-overlapping block feature vectors, merged all the spread sheets and randomised the data in the Bigdata dataset. Divided the domain into 10 sub domains and created 10-folds for validation (As a result created '10' training and '10' testing datasets)

All the datasets are included in the Data folder. The dataset which I created has 8410 rows x 82 columns. So all the test and train datasets generated from the cross validation have (7569 x 82) in training model and (841 x 82) in the testing.

```
time taken to run the RandomForest is (in seconds) 5.730526687999998

Confusion matrix is
 [[365  57]
 [181 238]]

test accuracy is: 0.7170035671819263
test_precision is: 0.8067796610169491
test_sensitivity is: 0.568019093078759
test_specificity is: 0.8649289099526067



sklearn.metrics Accuracy Output 0.7170035671819263
sklearn.metrics Precision Value 0.7376389147575588
sklearn.metrics Sensitivity Value 0.7164740015156827
For FOLD 1
```

Fold1

```
time taken to run the RandomForest is (in seconds) 5.643439367000042

Confusion matrix is
 [[397  45]
 [170 229]]

test accuracy is: 0.7443519619500594
test_precision is: 0.8357664233576643
test_sensitivity is: 0.5739348370927319
test_specificity is: 0.8981900452488688



sklearn.metrics Accuracy Output 0.7443519619500595
sklearn.metrics Precision Value 0.7679713951003488
sklearn.metrics Sensitivity Value 0.7360624411708003
For FOLD 2
```

Fold2

```
time taken to run the RandomForest is (in seconds) 5.42342114500002

Confusion matrix is
 [[361  60]
 [190 230]]

test accuracy is: 0.7027348394768134
test_precision is: 0.793103448275862
test_sensitivity is: 0.5476190476190476
test_specificity is: 0.8574821852731591



sklearn.metrics Accuracy Output 0.7027348394768134
sklearn.metrics Precision Value 0.7241379310344828
sklearn.metrics Sensitivity Value 0.7025506164461034
For FOLD 3
```

Fold3

```
time taken to run the RandomForest is (in seconds) 5.470550658000036

Confusion matrix is
 [[383  47]
 [183 228]]

test accuracy is: 0.7265160523186682
test_precision is: 0.829090909090909
test_sensitivity is: 0.5547445255474452
test_specificity is: 0.8906976744186046



sklearn.metrics Accuracy Output 0.7265160523186682
sklearn.metrics Precision Value 0.7528846771602955
sklearn.metrics Sensitivity Value 0.7227210999830249
For FOLD 4
```

Fold4

```
time taken to run the RandomForest is (in seconds) 5.7371804780000275

Confusion matrix is
 [[359  46]
 [194 242]]

test accuracy is: 0.7146254458977407
test_precision is: 0.8402777777777778
test_sensitivity is: 0.555045871559633
test_specificity is: 0.8864197530864197



sklearn.metrics Accuracy Output 0.7146254458977408
sklearn.metrics Precision Value 0.7447320172794856
sklearn.metrics Sensitivity Value 0.7207328123230263
For FOLD 5
```

Fold5

```
time taken to run the RandomForest is (in seconds) 5.773470750999991

Confusion matrix is
 [[351  62]
 [180 248]]

test accuracy is: 0.7122473246135553
test_precision is: 0.8
test_sensitivity is: 0.5794392523364486
test_specificity is: 0.8498789346246973



sklearn.metrics Accuracy Output 0.7122473246135553
sklearn.metrics Precision Value 0.7305084745762712
sklearn.metrics Sensitivity Value 0.714659093480573
For FOLD 6
```

Fold6

```
time taken to run the RandomForest is (in seconds) 5.450844899999993

Confusion matrix is
 [[328  71]
 [198 244]]

test accuracy is: 0.6801426872770512
test_precision is: 0.7746031746031746
test_sensitivity is: 0.5520361990950226
test_specificity is: 0.8220551378446115



sklearn.metrics Accuracy Output 0.6801426872770512
sklearn.metrics Precision Value 0.6990886595449333
sklearn.metrics Sensitivity Value 0.6870456684698171
For FOLD 7
```

Fold7

```
time taken to run the RandomForest is (in seconds) 5.874500657999988

Confusion matrix is
 [[366  47]
 [195 233]]

test accuracy is: 0.7122473246135553
test_precision is: 0.8321428571428571
test_sensitivity is: 0.544392523364486
test_specificity is: 0.8861985472154963



sklearn.metrics Accuracy Output 0.7122473246135553
sklearn.metrics Precision Value 0.7422746371275784
sklearn.metrics Sensitivity Value 0.7152955352899912
For FOLD 8
```

Fold8

```
time taken to run the RandomForest is (in seconds) 5.599392928999919

Confusion matrix is
 [[372  61]
 [186 222]]

test accuracy is: 0.7063020214030915
test_precision is: 0.7844522968197879
test_sensitivity is: 0.5441176470588236
test_specificity is: 0.859122401847575



sklearn.metrics Accuracy Output 0.7063020214030915
sklearn.metrics Precision Value 0.7255594817432274
sklearn.metrics Sensitivity Value 0.7016200244531993
For FOLD 9
```

Fold9

```
time taken to run the RandomForest is (in seconds) 5.757712367999943

Confusion matrix is
 [[370  57]
 [200 214]]

test accuracy is: 0.6944114149821641
test_precision is: 0.7896678966789669
test_sensitivity is: 0.5169082125603864
test_specificity is: 0.8665105386416861



sklearn.metrics Accuracy Output 0.6944114149821641
sklearn.metrics Precision Value 0.7193953518482553
sklearn.metrics Sensitivity Value 0.6917093756010363
For FOLD 10
```

Fold10

Figure 106: screenshots of RF classifier for all 10 test and train datasets.

The above screenshots are the outputs of random forest (also calculated the accuracies, computational times, confusion matrix for all the datasets generated from 10 fold cross validation. All the dataset contain two labels 0 and 1.

For block feature vectors of 10 images when applied a deep learning model it tool around 20 seconds to fit the model and then when I take the data points size of three million then the deep learning model runs for around 1000 seconds (I added more images datasheets to make the dataset to 3 million points).

Including all the datasets and code files in the zip folder where I have taken more images and created datasets using the same codes from assignment 1

```
test accuracy is: 0.7048648648648648
test_precision is: 0.7973421926910299
test_sensitivity is: 0.5309734513274337
test_specificity is: 0.8710359408033826



sklearn.metrics Accuracy Output 0.7048648648648649
sklearn.metrics Precision Value 0.7287993014737201
sklearn.metrics Sensitivity Value 0.7010046960654082

In [16]: C_test
Out[16]:
array([[824, 122],
       [424, 480]])
```

Figure 107: screenshots of outputs from deep learning model I have included the code and datasets in the folder.

Task 4: Comparing the results on a local system versus a big data system

I have taken a dataset with around more than 6 million datapoints (csv file is around 110MB) It throws an error message as of figure 108 and when I do the same in a big data system then the run time is around 813.33 seconds(used DataBricks Community edition).

Similarly, for the deep learning model with 20 epochs it took around 2500 seconds and when I used the big data platform the run time are 2100 seconds.
When the size of the dataset is low the time elapsed is around 20 seconds(when the datapoints are around 600k).

```
  File "/Users/vijaykrishna/opt/anaconda3/lib/python3.9/site-packages/joblib/parallel.py", line 1050, i
    self.retrieve()

  File "/Users/vijaykrishna/opt/anaconda3/lib/python3.9/site-packages/joblib/parallel.py", line 935, in
    self._output.extend(job.get(timeout=self.timeout))

  File "/Users/vijaykrishna/opt/anaconda3/lib/python3.9/multiprocessing/pool.py", line 765, in get
    self.wait(timeout)

  File "/Users/vijaykrishna/opt/anaconda3/lib/python3.9/multiprocessing/pool.py", line 762, in wait
    self._event.wait(timeout)

  File "/Users/vijaykrishna/opt/anaconda3/lib/python3.9/threading.py", line 574, in wait
    signaled = self._cond.wait(timeout)

  File "/Users/vijaykrishna/opt/anaconda3/lib/python3.9/threading.py", line 312, in wait
    waiter.acquire()
```

Figure 108 : error message while running the BigData datasets

```
[531344 rows x 81 columns]
[[1 1 1 ... 1 1 1]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [1 1 1 ... 1 1 1]
 [1 1 1 ... 1 1 1]
 [1 1 1 ... 1 1 1]]
(531344,)
[1 0 0 ... 1 1 1]
time taken to run the RandomForest is (in seconds) 813.3394742990001
```

Figure 109: runtime for random forest in bigdata platform.

When we run the bigdata in bigdata system the accuracies for the RF classifier is 95 percent and for the small datasets on our local machine it is around 70 percent. But for the deep learning model the accuracies are not as good as the random forest classifier It gives the same results when we run that in the local machine or on the bigdata system. Figure 107 is the accuracies for deep learning model when run in the local system.