

Assignment Report

Vamshi Krishna Edamadaka
Computer Science
University of North Carolina Greensboro
Greensboro, USA
v_edamadaka@uncg.edu

I. ASSIGNMENT 01: RAW DATA TO FEATURE SPACE

II. TASK 1

A. Building your own programming environment!

Installing the IDE on our different PCs is the first step in creating a programming environment. The instructions for installing the Anaconda environment on a PC are as follows.

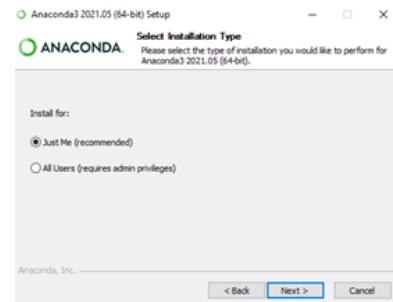
- 1) Go to <https://docs.anaconda.com/anaconda/> for further information. Then, on the left, click the 'Installation' tab.
- 2) Choose the operating system on which Anaconda will be installed. I click the 'Installing on Windows' link to install Anaconda on my local machine.
- 3) Download the Anaconda Installer onto your local environment.



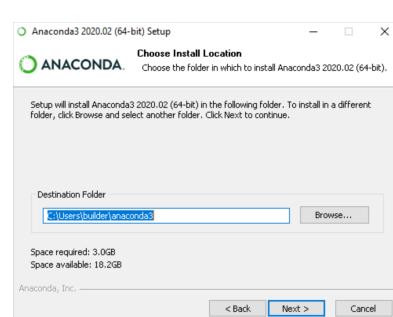
- 4) To run the application, double-click the installer and then select 'Next.'



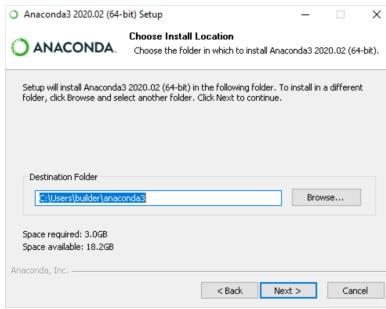
- 5) Read the License Agreement and the click on 'I Agree'.
- 6) Then, for the installation type, select 'Just Me,' as we are simply configuring the environment for the local user. Then press the 'Next' button.



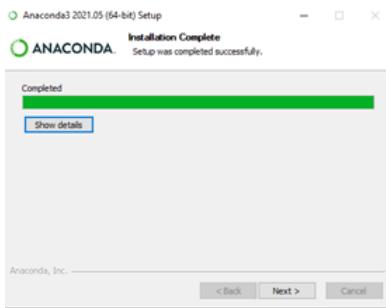
- 7) Select a destination folder in which to install Anaconda.



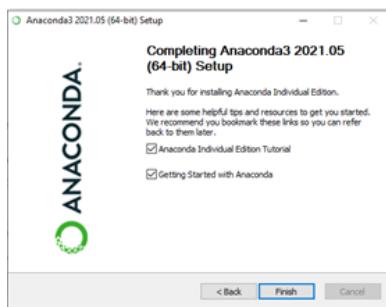
- 8) To avoid any conflicts with other applications installed on your computer, select 'Register Anaconda3 as my default Python 3.8' from the advanced installation menu. Then press the 'Install' button.



- 9) After installation is complete, click on 'Next'.



- 10) The Thank You for installation dialog box appears after successful completion. Here click 'Finish'



Anaconda is now successfully built onto your local environment.

B. Install the Spyder IDE

Spyder[2], or the Scientific Python Development Environment, is a powerful open-source integrated development

environment (IDE) created in Python for Python. Anaconda comes with a free integrated development environment (IDE).

Opening Anaconda and selecting the Spyder IDE by clicking on 'launch' is one approach to start Spyder.

Spyder can be be launched by typing 'spyder' into the Anaconda Prompt.

C. Installing OpenCV

The OpenCV-Python library is a set of Python bindings for solving computer vision issues. Numpy, a highly efficient library for numerical operations with a MATLAB-style syntax, is used by OpenCV-Python. Numpy arrays are translated to and from all OpenCV array forms.

To install the OpenCV package, start a command line and run the following command after navigating to the Anaconda installation folder:

```
pip install opencv-python
```

III. TASK 2

A. Download or generate a fruit and vegetables image datasets!

- 1) The datasets are obtained from the following website[3]: "<https://www.vicos.si/Downloads/FIDS30>"
- 2) The apple, mango, and strawberry were chosen as the three fruits for this task.



Fig. 1. apple



Fig. 2. mango



Fig. 3. strawberry

```

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

```

Fig. 4. Import statements

IV. TASK 3

A. Write a simple code to read your selected images and display them on the environment!

To perform this particular task we import the following libraries:

Apple, mango, and strawberry are the three fruits chosen for this project:

We are now going to plot their R, G, and B channel images.

```

# Read Images (e.g., .jpg, .png, and .tif)
mango_color = cv2.imread("../input_fruits/mango.jpg")
apple_color = cv2.imread("../input_fruits/apple.jpg")
strawberry_color = cv2.imread("../input_fruits/strawberry.jpg")

```

Fig. 5. importing the images

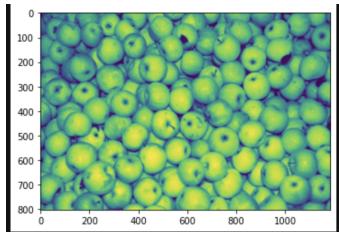


Fig. 6. Apple-RBG

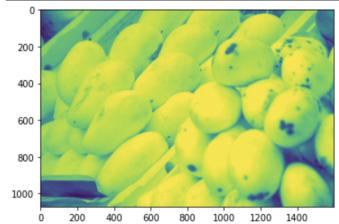


Fig. 7. mango-RBG

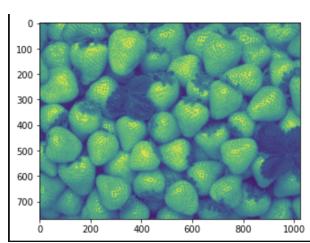


Fig. 8. Strawberry-RBG

B. Convert these color images to grayscale and display them while printing their dimensions

The color photos that have been made can be turned to grayscale and shown while being printed in their original dimensions. When photographs are converted to grey scale, their dimensions are printed.



Fig. 9. apple greyscale with dimensions (-0.5, 1184.5, 801.5, -0.5)



Fig. 10. mango greyscale with Dimensions (-0.5, 1599.5, 1070.5, -0.5)

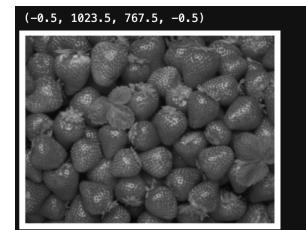


Fig. 11. strawberry with dimensions (-0.5, 1023.5, 767.5, -0.5)

V. TASK 4

1) Resize the images to reduce their dimensions!: We'll downsize the photographs in this work to make them smaller. The picture resizing is done here so that we can extract features from the image faster and utilize them to train our models later. The resized images will produce the following output:

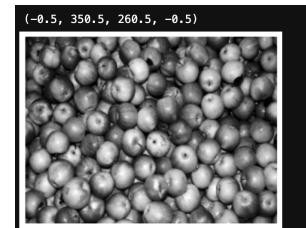


Fig. 12. Apple-Resized to (-0.5, 350.5, 260.5, -0.5)



Fig. 13. Mango-Resized to (-0.5, 314.5, 260.5, -0.5)

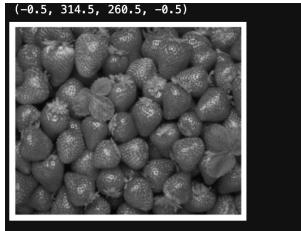


Fig. 14. Strawberry-Resized to (-0.5, 314.5, 260.5, -0.5)

VI. TASK 5

A. Generate block-feature vectors!

Now we'll build code to partition an image into 9x9 pixel sliding blocks, convert them to 81-bit vectors, and label each feature vector with 0, 1, and 2 for the first, second, and third images, respectively. Then, for each image, we create a spreadsheet by saving a feature vector per row in the spreadsheet.

```
mm = round(((heightm)*(widthm)))
flatmm = np.zeros((mm, 82), np.uint8)
k = 0
for i in range(heightm-8):
    for j in range(widthm-8):
        crop_tmp = mango[i:i+9,j:j+9]
        flatmm[k,0:81] = crop_tmp.flatten()
        k = k + 1
fspacemm = pd.DataFrame(flatmm) #panda object
fspacemm.to_csv('../output_fruits/fspacemm.csv', index=False)

aa = round(((heighta)*(widtha)))
flataa = np.zeros((aa, 82), np.uint8)
k = 0
for i in range(heighta-8):
    for j in range(widtha-8):
        crop_tmp = apple[i:i+9,j:j+9]
        flataa[k,0:81] = crop_tmp.flatten()
        k = k + 1
fspaceaa = pd.DataFrame(flataa) #panda object
fspaceaa.to_csv('../output_fruits/fspaceaa.csv', index=False)

ss = round(((heights)*(widths)))
flatss = np.zeros((ss, 82), np.uint8)
k = 0
for i in range(heights-8):
    for j in range(widths-8):
        crop_tmp = strawberry[i:i+9,j:j+9]
        flatss[k,0:81] = crop_tmp.flatten()
        k = k + 1
fspacess = pd.DataFrame(flatss) #panda object
fspacess.to_csv('../output_fruits/fspacess.csv', index=False)
```

Fig. 15. Task 5

VII. TASK 6

A. Generate sliding block-feature vectors!

Now we'll build code to partition an image into 9x9 pixel sliding blocks, convert them to 81-bit vectors, and label each feature vector with 0, 1, and 2 for the first, second, and third images, respectively. Then, for each image, we create a spreadsheet by saving a feature vector per row in the spreadsheet.

```
mm = round(((heightm)*(widthm))/81)
flatm = np.zeros((mm, 82), np.uint8)
k = 0
for i in range(0,heightm,9):
    for j in range(0,widthm,9):
        crop_tmp1 = mango[i:i+9,j:j+9]
        flatm[k,0:81] = crop_tmp1.flatten()
        k = k + 1
fspaceM = pd.DataFrame(flatm) #panda object
fspaceM.to_csv('../output_fruits/fspaceM.csv', index=False)

aa = round(((heighta)*(widtha))/81)
flata = np.ones((aa, 82), np.uint8)
k = 0
for i in range(0,heighta,9):
    for j in range(0,widtha,9):
        crop_tmp2 = apple[i:i+9,j:j+9]
        flata[k,0:81] = crop_tmp2.flatten()
        k = k + 1
fspaceA = pd.DataFrame(flata) #panda object
fspaceA.to_csv('../output_fruits/fspaceA.csv', index=False)

ss = round(((heights)*(widths))/81)
flats = np.full((ss, 82), 2)
k = 0
for i in range(0,heights,9):
    for j in range(0,widths,9):
        crop_tmp3 = strawberry[i:i+9,j:j+9]
        flats[k,0:81] = crop_tmp3.flatten()
        k = k + 1
```

Fig. 16. Task 5

The generated spreadsheet for all three fruits is stored in the 'outputfruits' folder.

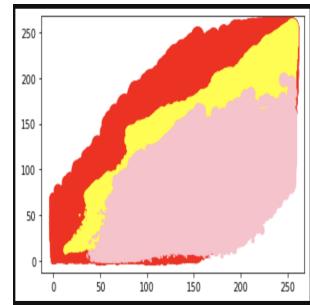
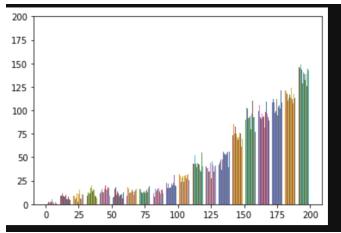
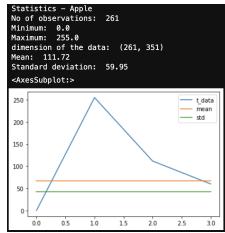
VIII. TASK 7

A. Extract the statistical information from data

We display the statistics like no of observations, minimum, maximum, dimension, mean and standard deviation.

We now get the mean and standard deviation of all the three fruits and represent them visually in the form of a line graph.

Apple Statistics: For Apple its selected feature have min value of 0 and max value of 255 and it mean is 111.72 and standard deviation is 59.95. The graph plotted for the features of Apple shows that _ data increases from 0-1 of x-axis the standard deviation and mean remains constant and as the _ decreases from 1-2 of x-axis the standard deviation and mean still remains constant. The output and graph for min,max,standard deviation,mean of Strawberry are shown in the graph above.

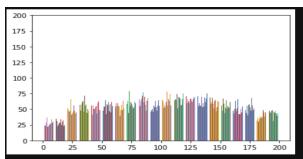
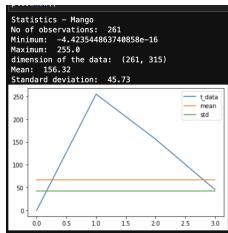


histogram:

Mango Statistics

For Mango its selected feature have min value of 0 and max value of 255 and it mean is 156.32 and standard deviation is 45.73. The graph plotted for the features of mango shows that `_data` increases from 0-1 of x-axis the standard deviation and mean remains contant and as the `_` decreases from 1-2 of x-axis the standard deviation and mean still remains constant. The output and graph for min,max,standard deviation,mean of mango are shown in the graph above.

histogram:

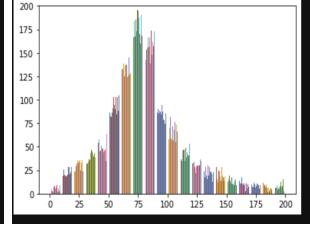
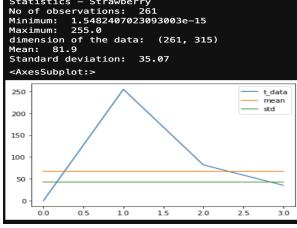


Strawberry Statistics

For Strawberry its selected feature have min value of 0 and max value of 255 and it mean is 81.9 and standard deviation is 35.07. The graph plotted for the features of Strawberry shows that `_data` increases from 0-1 of x-axis the standard deviation and mean remains contant and as the `_` decreases from 1-2 of x-axis the standard deviation and mean still remains constant. The output and graph for min,max,standard deviation,mean of Strawberry are shown in the graph above.

histogram:

We plot the images as scatterplot by its color property as



shown below, apple is shown by red color, mango by yellow and strawberry by pink.

IX. TASK 8

A. Construct a feature space!

To generate a feature space for these images, we first integrate the feature vectors in `image0.csv` and `image1.csv`. To build the suitable feature space for these image classes, each feature and label column aligns vertically. In the output folder, the created feature space is named '`image01.csv`'.

Similarly, we integrate the feature vectors in `image0.csv`, `image1.csv`, and `image2.csv`. To generate the suitable feature space for these three classes, each feature and label column must line vertically. The resulting feature space file is named '`image012.csv`' and saved in the output folder. In the files `image01.csv` and `image012.csv`, we now randomize the placement of the feature vectors. We don't randomize the content of a feature vector here, but the placement (rows) of the csv files.

```
# Join the feature vectors of the classes
frames = [fspaceM, fspaceA]
mged = pd.concat(frames)
mged.to_csv('../output_fruits/image01.csv', index=False)

frames = [fspaceM, fspaceA, fspaceS]
mged = pd.concat(frames)
mged.to_csv('../output_fruits/image012.csv', index=False)

#Randomize
#img01
input_data = pd.read_csv("../output_fruits/image01.csv", header=None)
indx = np.arange(len(input_data))
rndimage01 = np.random.permutation(indx)
rndimage01 = input_data.sample(frac=1).reset_index(drop=True)
rndimage01.to_csv('../output_fruits/randomimage01.csv', index=False)

#img012
input_data = pd.read_csv("../output_fruits/image012.csv", header=None)
indx = np.arange(len(input_data))
rndimage01 = np.random.permutation(indx)
rndimage01 = input_data.sample(frac=1).reset_index(drop=True)
rndimage01.to_csv('../output_fruits/randomimage012.csv', index=False)
```

Finally, we randomize the generated `image01.csv` and `image012.csv` file and generate two new files `randomimage01.csv` and `randomimage012.csv` respectively.

X. TASK 9

A. Display subspaces!

Using the spreadsheets that we prepared, we select two features and plot the two-dimensional feature space, labeling the observations (vectors) of the fruits or vegetables that we selected. To plot a two-dimensional feature space, we use the Apple and mango spreadsheet to choose the fruits.

```

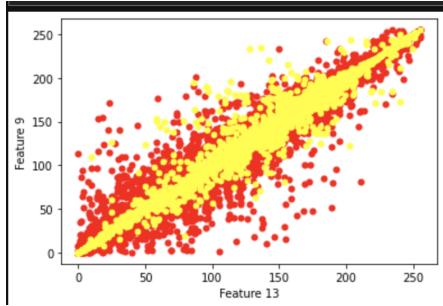
#2D representation
apple1 = pd.read_csv('../output_fruits/fspaceA.csv', index_col = '13')
apple2 = pd.read_csv('../output_fruits/fspaceA.csv', index_col = '9')
plt.xlabel('Feature 26')
plt.ylabel('Feature 17')
plt.scatter(apple1, apple2, c = 'red', linewidths=0)

mango1 = pd.read_csv('../output_fruits/fspaceM.csv', index_col = '13')
mango2 = pd.read_csv('../output_fruits/fspaceM.csv', index_col = '9')
plt.scatter(mango1,mango2, c = 'yellow', linewidths = 0)

#3D representation
apple3 = pd.read_csv('../output_fruits/fspaceA.csv', index_col = '4')
mango3 = pd.read_csv('../output_fruits/fspaceM.csv', index_col = '4')
strawberry1 = pd.read_csv('../output_fruits/fspaceS.csv', index_col = '13')
strawberry2 = pd.read_csv('../output_fruits/fspaceS.csv', index_col = '9')
strawberry3 = pd.read_csv('../output_fruits/fspaceS.csv', index_col = '4')
fig = plt.figure(figsize=(9,9))

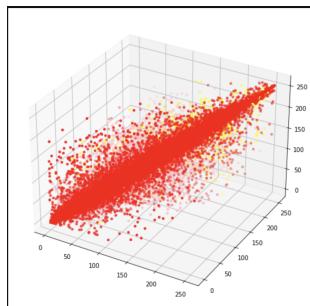
ax = fig.add_subplot(111, projection='3d')
ax.scatter(apple1,apple2,apple3,c = 'red', linewidth = 0)
ax.scatter(mango1,mango2,mango3,c = 'yellow', linewidth = 0)
ax.scatter(strawberry1,strawberry2,strawberry3,c = 'pink', linewidth = 0)
plt.show()

```



From the above graph we see various points in which our features are spread out.

Then, using the spreadsheets we created, we select three features and plot the three-dimensional feature space, labeling the observations (vectors) of the fruits or vegetables we chose. Feature 26, Feature 17, and Feature 4 of our fruits Apple, Mango, and Strawberry are now plotted.



From the graph above, we can see the various distinct plotted points of each feature of all the fruits.

XI. TASK 10 - CONVERTING ALL THE IMAGES IN A FOLDER

Now we'll construct Python code to read any number of images from a folder of many similar images, create feature space, and create spreadsheets for the feature spaces.

In this code we try to read all the images under the

```

for i in range (0,38):
    fruit_color = cv2.imread("../input_fruits/apples/" + str(i) + ".jpg")
    fruitG = cv2.cvtColor(fruit_color, cv2.COLOR_BGR2GRAY)
    heightfg, widthfg, = fruitG.shape
    fruit = cv2.resize(fruitG, dsize=(315,261),interpolation = cv2.INTER_CUBIC)

    mm = round(((heightfg)*(widthfg)))
    flatmm = np.zeros((mm, 82), np.uint8)
    for l in range(heightfg-8):
        for j in range(widthfg-8):
            crop_tmp = mango[l:l+9,j:j+9]
            flatmm[k,0:81] = crop_tmp.flatten()
    fspacemm = pd.DataFrame(flatmm) #panda object
    fspacemm.to_csv('../output_fruits/task10/' + str(i) + '.csv')

```

folder '*inputfruits-> apples*'. The generated spreadsheets for the images under this folder are stored under 'Task 10' in the output folder. As you can see, outputs are generated



under '*outputfruits/task10*' for all the images in the input folder.

XII. TASK 11

We move into higher dimensions when we have a larger block size. Our data becomes increasingly sparse and evenly spread in higher dimensions. As a result, the classifier has a hard time distinguishing between the features and learning from them. As a result, we minimize the block size to assist our classifier in training models more efficiently and quickly.

XIII. ASSIGNMENT 02: FEATURE SPACE TO A CLASSIFIER

XIV. TASK 1

A. Complete and extend the tasks of assignment 1

The four categories of datasets created in the first stage are

- 1) Non overlapping 2 class
- 2) Non overlapping multi class
- 3) Overlapping 2 class
- 4) Overlapping multi class

we are reading all these categories datasets which are saved in the Assignment 1.

```
# importing 4 categories of datasets
non_overlap_2_class = pd.read_csv("../input/image01.csv",header=None)
non_overlap_multi_class = pd.read_csv("../input/image02.csv",header=None)
overlap_2_class = pd.read_csv("../input/randomimage01.csv",header=None)
overlap_multi_class = pd.read_csv("../input/randomimage02.csv",header=None)
```

B. Divide the data domain of the data sets into training and testing sets

All the four categories of data are split into training and testing sets of ratio 78:22 respectively. Below example is the shape of non overlapping 2 class classification after the split.

```
non_overlap_2_class_train.shape
(1674, 82)
non_overlap_2_class_test.shape
(473, 82)
```

C. Selecting features and plotting their Histograms and Scatter Plots

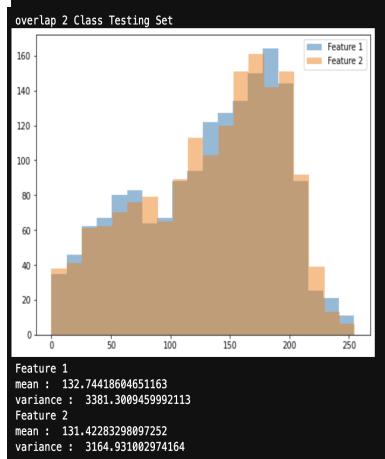
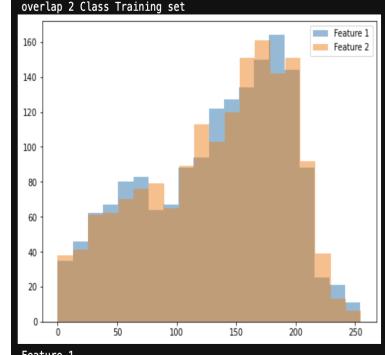
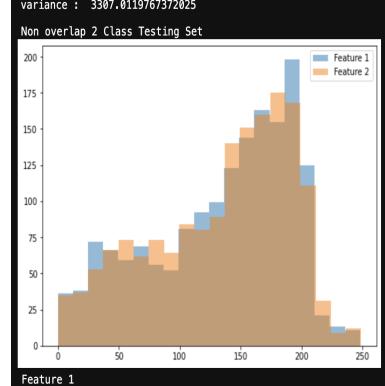
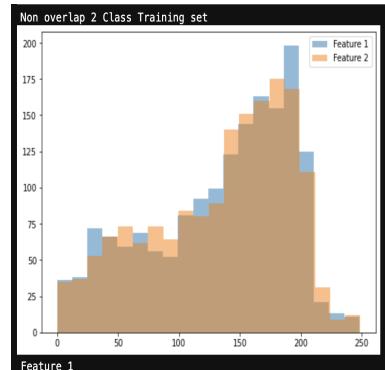
Feature Selection:

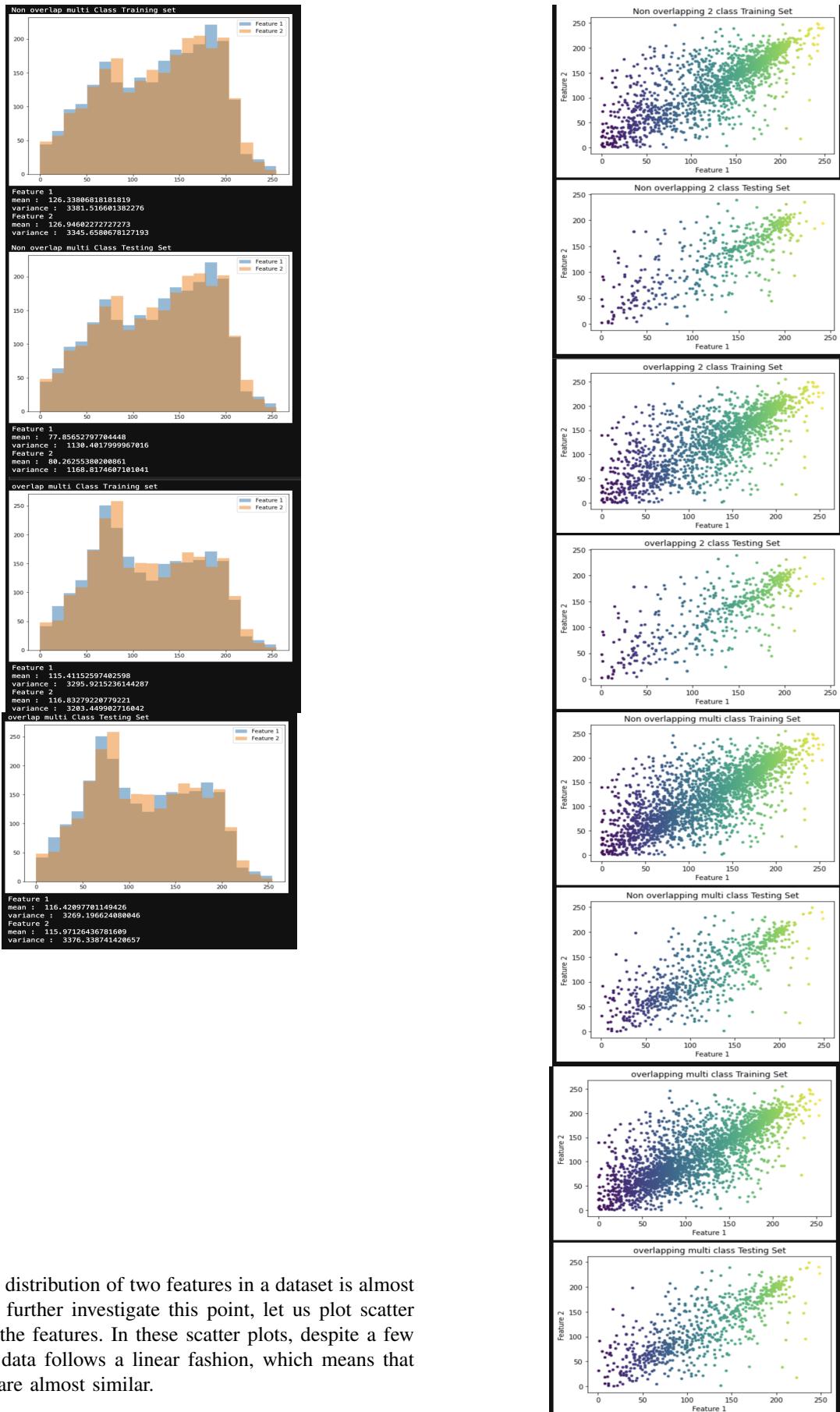
Two features 13 and 26 are selected in training and testing sets of every category as follows. Histograms:

```
: non_overlap_2_class_train_feature_1 = non_overlap_2_class_train[13]
non_overlap_2_class_train_feature_2 = non_overlap_2_class_train[26]
non_overlap_2_class_test_feature_1 = non_overlap_2_class_test[13]
non_overlap_2_class_test_feature_2 = non_overlap_2_class_test[26]
```

Histograms of each category and two datasets in each category are plotted. The histograms of two features are plotted in an overlapping manner to show whether they follow a similar distribution or not.

The following are the histograms and their statistics like mean and variance. In all these histograms, we can observe that the





statistics and distribution of two features in a dataset is almost the same To further investigate this point, let us plot scatter plots for all the features. In these scatter plots, despite a few outliers, the data follows a linear fashion, which means that the features are almost similar.

XV. TASK 2

A. Implementing lasso regression and add predicted label column to respective test dataset

Lasso regression is implemented and used as a two class classifier to classify overlapped and non-overlapped 2 class datasets. Although it is mentioned to use lasso regression as 2 class classifier only in the subtask 1, in the subtask 2 it is written to test the test sets of all the categories of datasets, so I assumed that multi class classifier should also be created. Hence I have used lasso regression to create both two class and multi class classifier and classified all the 4 categories of data.

Training the model and Testing the results:

1) Non overlapping 2 class:

for the non overlapping two class, Training set is split into X_train (containing all the features) and y_train (actual labels) and the model is trained as shown below. In the same way, the model is tested on test set and the

```
Training on non overlapped 2 class

: from sklearn.linear_model import Lasso
: X_train = non_overlap_2_class_train.loc[:,non_overlap_2_class_train.columns!=81]
y_train = non_overlap_2_class_train.loc[:, :-1]

: model = Lasso(alpha=0.1)
model.fit(X_train, y_train)
```

predicted results are added in an additional column as shown below.

non_overlap_2_class_test																																																																																			Predicted column
0	1	2	3	4	5	6	7	8	9	...	73	74	75	76	77	78	79	80	81	Predicted column																																																															
1674	81	77	62	67	88	62	60	62	63	75	...	56	64	100	108	84	83	81	72	1	1																																																														
1675	64	60	47	34	52	85	96	104	107	60	...	13	44	72	47	42	35	44	68	1	1																																																														
1676	113	119	118	124	124	121	125	128	116	105	...	23	77	103	124	147	141	150	147	1	0																																																														
1677	121	117	102	107	99	88	64	52	40	103	...	146	151	146	155	147	145	135	121	1	0																																																														
1678	16	78	141	155	28	7	19	17	4	58	...	104	83	51	6	2	20	15	10	1	1																																																														
...																																																															
2142	35	36	28	28	30	27	29	25	21	25	...	32	47	54	59	60	67	63	55	1	1																																																														
2143	20	18	14	16	18	13	46	73	80	8	...	42	47	73	105	115	131	144	149	1	1																																																														
2144	89	97	100	95	97	91	66	77	58	101	...	150	147	154	158	153	152	151	147	1	1																																																														
2145	43	28	28	27	24	21	20	17	17	92	...	124	101	103	103	84	71	52	47	1	1																																																														
2146	10	4	2	9	20	39	73	85	95	22	...	27	30	56	64	76	85	83	103	1	1																																																														

473 rows x 83 columns

2) Non overlapping multi class:

for the non overlapping multi class, Training set is split into X_train (containing all the features) and y_train (actual labels) and the model is trained as shown below. In the same way, the model is tested on test set and the

```
Training on non overlapped multi class

: X_train = non_overlap_multi_class_train.loc[:,non_overlap_multi_class_train.columns!=81]
y_train = non_overlap_multi_class_train.loc[:, :-1]

: non_overlap_multi_class_train[81].unique()
: array([0, 1, 2])

: model = Lasso(alpha=0.1)
model.fit(X_train, y_train)

: Lasso(alpha=0.1)
```

predicted results are added in an additional column as shown below.

non_overlap_multi_class_test																																																																																			Predicted column
0	1	2	3	4	5	6	7	8	9	...	73	74	75	76	77	78	79	80	81	Predicted column																																																															
2465	135	125	92	106	176	175	156	163	81	183	...	100	119	116	103	182	13	105	100	2	1																																																														
2466	91	94	86	85	84	85	76	75	79	85	...	107	147	79	113	84	79	72	64	2	1																																																														
2467	76	60	28	76	17	19	30	51	64	64	...	57	47	25	26	39	23	48	25	2	1																																																														
2468	58	70	53	64	58	58	39	26	25	32	...	19	22	26	58	63	72	67	72	2	1																																																														
2469	23	61	88	88	81	89	128	104	99	75	...	73	91	163	81	163	35	183	209	2	1																																																														
...																																																															
3157	51	61	50	35	33	42	47	60	45	53	...	45	48	50	53	57	70	58	64	2	1																																																														
3158	48	68	69	68	70	85	69	81	119	63	...	67	53	73	67	122	79	67	91	2	1																																																														
3159	84	75	101	101	92	125	159	173	129	145	...	94	94	151	89	81	100	88	82	2	1																																																														
3160	95	81	58	60	61	61	73	69	94	95	112	...	109	107	76	92	107	147	114	85	2	1																																																													
3161	89	42	42	50	84	34	59	64	69	81	...	57	50	53	28	14	44	54	61	2	1																																																														

697 rows x 83 columns

3) Overlapping 2 class

for the overlapping 2 class, Training set is split into X_train (containing all the features) and y_train (actual labels) and the model is trained as shown below. In

```
Lasso Regression for overlapping features

Training on overlapped 2 Class

X_train = overlap_2_class_train.loc[:,overlap_2_class_train.columns!=81]
y_train = overlap_2_class_train.loc[:, :-1]

model = Lasso(alpha=0.1)
model.fit(X_train, y_train)

Lasso(alpha=0.1)
```

the same way, the model is tested on test set and the predicted results are added in an additional column as shown below.

overlap_2_class_test																																																																																			Predicted column
0	1	2	3	4	5	6	7	8	9	...	73	74	75	76	77	78	79	80	81	Predicted column																																																															
1675	168	181	176	168	165	153	162	158	156	168	...	172	168	166	159	151	140	130	131	0	0																																																														
1676	196	197	198	197	195	193	194	169	149	196	...	195	194	195	188	194	192	190	185	0	0																																																														
1677	192	193	191	197	179	176	181	184	182	192	...	177	177	178	179	179	177	171	173	0	0																																																														
1678	206	204	200	196	193	191	190	188	186	205	...	202	199	196	194	193	190	189	185	0	0																																																														
1679	207	198	204	209	198	198	198	194	191	205	...	195	195	191	190	191	193	185	188	0	0																																																														
...																																																															
2143	149	151	157	152	153	140	129	119	111	158	...	194	195	198	193	208	195	199	193	1	0																																																														
2144	221	221	204	199	191	199	177	204	204	222	...	226	220	221	211	181	165	191	198	202	1																																																														
2145	112	114	111	109	104	100	101	102	116	142	...	186	124	133	145	176	213	230	230	0	0																																																														
2146	205	204	203	200	199	196	189	184	184	204	...	192	190	188	185	184	182	179	178	0	0																																																														
2147	30	31	46	60	59	68	103	109	86	27	...	23	25	31	52	43	73	99	89	1	1																																																														

473 rows x 83 columns

4) Overlapping multi class

or the overlapping multi class, Training set is split into X_train (containing all the features) and y_train (actual labels) and the model is trained as shown below. In

```
Training on overlapped multi Class

X_train = overlap_multi_class_train.loc[:,overlap_multi_class_train.columns!=81]
y_train = overlap_multi_class_train.loc[:, :-1]

model = Lasso(alpha=0.1)
model.fit(X_train, y_train)

Lasso(alpha=0.1)

overlap_multi_class_train[81].unique()
array([1, 0, 2])
```

the same way, the model is tested on test set and the predicted results are added in an additional column as shown below.

overlap_multi_class_test		Predicted column
0	1	2
3	4	5
6	7	8
9	8	9
...
73	74	75
76	77	78
79	80	81
82	83	84
85	86	87
88	89	90
91	92	93
94	95	96
97	98	99
100	101	102
103	104	105
106	107	108
109	110	111
112	113	114
115	116	117
118	119	120
121	122	123
124	125	126
127	128	129
129	130	131
132	133	134
135	136	137
138	139	140
141	142	143
144	145	146
147	148	149
150	151	152
153	154	155
156	157	158
157	158	159
159	160	161
162	163	164
165	166	167
168	169	170
171	172	173
174	175	176
177	178	179
179	180	181
182	183	184
185	186	187
188	189	190
191	192	193
194	195	196
197	198	199
199	200	201
202	203	204
205	206	207
207	208	209
209	210	211
212	213	214
215	216	217
217	218	219
219	220	221
221	222	223
223	224	225
225	226	227
227	228	229
229	230	231
231	232	233
233	234	235
235	236	237
237	238	239
239	240	241
241	242	243
243	244	245
245	246	247
247	248	249
249	250	251
251	252	253
253	254	255
255	256	257
257	258	259
259	260	261
261	262	263
263	264	265
265	266	267
267	268	269
269	270	271
271	272	273
273	274	275
275	276	277
277	278	279
279	280	281
281	282	283
283	284	285
285	286	287
287	288	289
289	290	291
291	292	293
293	294	295
295	296	297
297	298	299
299	300	301
301	302	303
303	304	305
305	306	307
307	308	309
309	310	311
311	312	313
313	314	315
315	316	317
317	318	319
319	320	321
321	322	323
323	324	325
325	326	327
327	328	329
329	330	331
331	332	333
333	334	335
335	336	337
337	338	339
339	340	341
341	342	343
343	344	345
345	346	347
347	348	349
349	350	351
351	352	353
353	354	355
355	356	357
357	358	359
359	360	361
361	362	363
363	364	365
365	366	367
367	368	369
369	370	371
371	372	373
373	374	375
375	376	377
377	378	379
379	380	381
381	382	383
383	384	385
385	386	387
387	388	389
389	390	391
391	392	393
393	394	395
395	396	397
397	398	399
399	400	401
401	402	403
403	404	405
405	406	407
407	408	409
409	410	411
411	412	413
413	414	415
415	416	417
417	418	419
419	420	421
421	422	423
423	424	425
425	426	427
427	428	429
429	430	431
431	432	433
433	434	435
435	436	437
437	438	439
439	440	441
441	442	443
443	444	445
445	446	447
447	448	449
449	450	451
451	452	453
453	454	455
455	456	457
457	458	459
459	460	461
461	462	463
463	464	465
465	466	467
467	468	469
469	470	471
471	472	473
473	474	475
475	476	477
477	478	479
479	480	481
481	482	483
483	484	485
485	486	487
487	488	489
489	490	491
491	492	493
493	494	495
495	496	497
497	498	499
499	500	501
501	502	503
503	504	505
505	506	507
507	508	509
509	510	511
511	512	513
513	514	515
515	516	517
517	518	519
519	520	521
521	522	523
523	524	525
525	526	527
527	528	529
529	530	531
531	532	533
533	534	535
535	536	537
537	538	539
539	540	541
541	542	543
543	544	545
545	546	547
547	548	549
549	550	551
551	552	553
553	554	555
555	556	557
557	558	559
559	560	561
561	562	563
563	564	565
565	566	567
567	568	569
569	570	571
571	572	573
573	574	575
575	576	577
577	578	579
579	580	581
581	582	583
583	584	585
585	586	587
587	588	589
589	590	591
591	592	593
593	594	595
595	596	597
597	598	599
599	600	601
601	602	603
603	604	605
605	606	607
607	608	609
609	610	611
611	612	613
613	614	615
615	616	617
617	618	619
619	620	621
621	622	623
623	624	625
625	626	627
627	628	629
629	630	631
631	632	633
633	634	635
635	636	637
637	638	639
639	640	641
641	642	643
643	644	645
645	646	647
647	648	649
649	650	651
651	652	653
653	654	655
655	656	657
657	658	659
659	660	661
661	662	663
663	664	665
665	666	667
667	668	669
669	670	671
671	672	673
673	674	675
675	676	677
677	678	679
679	680	681
681	682	683
683	684	685
685	686	687
687	688	689
689	690	691
691	692	693
693	694	695
695	696	697
697	698	699
699	700	701
701	702	703
703	704	705
705	706	707
707	708	709
709	710	711
711	712	713
713	714	715
715	716	717
717	718	719
719	720	721
721	722	723
723	724	725
725	726	727
727	728	729
729	730	731
731	732	733
733	734	735
735	736	737
737	738	739
739	740	741
741	742	743
743	744	745
745	746	747
747	748	749
749	750	751
751	752	753
753	754	755
755	756	757
757	758	759
759	760	761
761	762	763
763	764	765
765	766	767
767	768	769
769	770	771
771	772	773
773	774	775
775	776	777
777	778	779
779	780	781
781	782	783
783	784	785
785	786	787
787	788	789
789	790	791
791	792	793
793	794	795
795	796	797
797	798	799
799	800	801
801	802	803
803	804	805
805	806	807
807	808	809
809	810	811
811	812	813
813	814	815
815	816	817
817	818	819
819	820	821
821	822	823
823	824	825
825	826	827
827	828	829
829	830	831
831	832	833
833	834	835
835	836	837
837	838	839
839	840	841
841	842	843
843	844	845
845	846	847
847	848	849
849	850	851
851	852	853
853	854	855
855	856	857
857	858	859
859	860	861
861	862	863
863	864	865
865	866	867
867	868	869
869	870	871
871	872	873
873	874	875
875	876	877
877	878	879
879	880	881
881	882	883
883	884	885
885	886	887
887	888	889
889	890	891
891	892	893
893	894	895
895	896	897
897	898	899
899	900	901
901	902	903

C. Qualitative measures - Confusion Matrix based

Specificity and Sensitivity are chosen as the confusion based qualitative measures for this task.

Specificity is calculated by $TN / (TN + FP)$

Sensitivity is calculated by $TP / (TP + FN)$

But when it comes to multi class classification, both these measures are individually calculated and their average is considered as final quality measure.

All these values are calculated from the confusion matrix and measured. Below are the values of each category.

1) Non overlapping 2 class:

Below are the values

```
Specificity and Sensitivity
Sensitivity = TP / TP + FN
Sensitivity = cf_matrix[[1][1]] / (cf_matrix[[1][1]] + cf_matrix[[0][1]])
print("Sensitivity is : ",Sensitivity)
Sensitivity is : 1.0
Specificity = TN / TN + FP
Specificity = cf_matrix[[0][0]] / (cf_matrix[[0][0]] + cf_matrix[[1][0]])
print("Specificity is : ",Specificity)
Specificity is : 1.0
```

2) Non overlapping multi class:

Below are the values

```
Specificity and Sensitivity
• Sensitivity = TP / TP + FN
• Specificity = TN / TN + FP
• Considering measures of specificity and sensitivity are calculated individually and then their average is considered as final measure.

Sensitivity_0 = cf_matrix[[0][0]] / (cf_matrix[[0][0]] + cf_matrix[[1][0]] + cf_matrix[[2][0]])
Sensitivity_1 = cf_matrix[[1][1]] / (cf_matrix[[1][1]] + cf_matrix[[0][1]] + cf_matrix[[2][1]])
Sensitivity_2 = cf_matrix[[2][2]] / (cf_matrix[[2][2]] + cf_matrix[[0][2]] + cf_matrix[[1][2]])
Total_Sensitivity = (Sensitivity_0 + Sensitivity_1 + Sensitivity_2)/3
Specificity_0 = (cf_matrix[[0][0]]+cf_matrix[[1][0]]+cf_matrix[[2][0]]) - (cf_matrix[[0][1]]+cf_matrix[[0][2]]+cf_matrix[[1][2]])
Specificity_1 = (cf_matrix[[0][0]]+cf_matrix[[1][0]]+cf_matrix[[2][0]]) - (cf_matrix[[0][0]]+cf_matrix[[1][1]]+cf_matrix[[2][1]])
Specificity_2 = (cf_matrix[[0][0]]+cf_matrix[[1][0]]+cf_matrix[[2][0]]) - (cf_matrix[[0][0]]+cf_matrix[[0][2]]+cf_matrix[[1][2]])
Total_Specificity = (Specificity_0 + Specificity_1 + Specificity_2)/3
print("Specificity is : ",Total_Specificity)
print("Sensitivity is : ",Total_Sensitivity)
Sensitivity is : 0.3333333333333333
Specificity is : 0.6666666666666667
```

3) Overlapping 2 class:

Below are the values

```
Specificity and Sensitivity
Sensitivity = cf_matrix[[1][1]] / (cf_matrix[[1][1]] + cf_matrix[[0][1]])
Specificity = cf_matrix[[0][0]] / (cf_matrix[[0][0]] + cf_matrix[[1][0]])
print("Sensitivity is : ",Sensitivity)
print("Specificity is : ",Specificity)
Sensitivity is : 0.652396696214876
Specificity is : 0.6272727272727273
```

4) Overlapping multi class:

Below are the values

```
Sensitivity_0 = cf_matrix[[0][0]] / (cf_matrix[[0][0]] + cf_matrix[[1][0]] + cf_matrix[[2][0]])
Sensitivity_1 = cf_matrix[[1][1]] / (cf_matrix[[1][1]] + cf_matrix[[0][1]] + cf_matrix[[2][1]])
Sensitivity_2 = cf_matrix[[2][2]] / (cf_matrix[[2][2]] + cf_matrix[[0][2]] + cf_matrix[[1][2]])
Total_Sensitivity = (Sensitivity_0 + Sensitivity_1 + Sensitivity_2)/3
Specificity_0 = (cf_matrix[[0][0]]+cf_matrix[[1][0]]+cf_matrix[[2][0]]) - (cf_matrix[[0][1]]+cf_matrix[[0][2]]+cf_matrix[[1][2]])
Specificity_1 = (cf_matrix[[0][0]]+cf_matrix[[1][0]]+cf_matrix[[2][0]]) - (cf_matrix[[0][0]]+cf_matrix[[1][1]]+cf_matrix[[2][1]])
Specificity_2 = (cf_matrix[[0][0]]+cf_matrix[[1][0]]+cf_matrix[[2][0]]) - (cf_matrix[[0][0]]+cf_matrix[[0][2]]+cf_matrix[[1][2]])
Total_Specificity = (Specificity_0 + Specificity_1 + Specificity_2)/3
print("Specificity is : ",Total_Specificity)
print("Sensitivity is : ",Total_Sensitivity)
Specificity is : 0.594135697276772
Sensitivity is : 0.6933495359732051
```

XVI. TASK 3

A. Implementing random forest and add predicted label column to respective test dataset

Random Forest is implemented and used as a two class classifier and multi class classifier to classify all the 4

categories of data.

Training the model and Testing the results:

1) Non overlapping 2 class:

for the non overlapping two class, Training set is split into X_{train} (containing all the features) and y_{train} (actual labels) and the model is trained as shown below. In the same way, the model is tested on test set and the

non overlapping 2 class

```
from sklearn.ensemble import RandomForestRegressor
rf = RandomForestRegressor(n_estimators = 1000, random_state = 42)
X_train = non_overlap_2_class_train.loc[:,non_overlap_2_class_train.columns!=81]
y_train = non_overlap_2_class_train.iloc[:, -1]
rf.fit(X_train, y_train);
```

predicted results are added in an additional column.

2) Non overlapping multi class:

for the non overlapping multi class, Training set is split into X_{train} (containing all the features) and y_{train} (actual labels) and the model is trained as shown below. In the same way, the model is tested on test set and the

non overlapping multi class

```
rf = RandomForestRegressor(n_estimators = 1000, random_state = 42)
X_train = non_overlap_multi_class_train.loc[:,non_overlap_multi_class_train.columns!=81]
y_train = non_overlap_multi_class_train.iloc[:, -1]
rf.fit(X_train, y_train);
```

predicted results are added in an additional column.

3) Overlapping 2 class

for the overlapping 2 class, Training set is split into X_{train} (containing all the features) and y_{train} (actual labels) and the model is trained as shown below. In

overlapping 2 class

```
rf = RandomForestRegressor(n_estimators = 1000, random_state = 42)
X_train = overlap_2_class_train.loc[:,overlap_2_class_train.columns!=81]
y_train = overlap_2_class_train.iloc[:, -1]
rf.fit(X_train, y_train);
```

the same way, the model is tested on test set and the predicted results are added in an additional column.

4) Overlapping multi class

or the overlapping multi class, Training set is split into X_{train} (containing all the features) and y_{train} (actual labels) and the model is trained as shown below. In

overlapping multi class

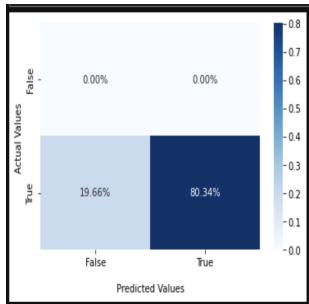
```
rf = RandomForestRegressor(n_estimators = 1000, random_state = 42)
X_train = overlap_multi_class_train.loc[:,overlap_multi_class_train.columns!=81]
y_train = overlap_multi_class_train.iloc[:, -1]
rf.fit(X_train, y_train);
```

the same way, the model is tested on test set and the predicted results are added in an additional column.

B. Construct Confusion matrix for each category

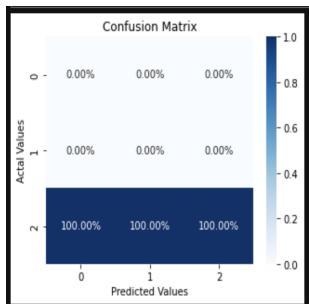
1) Non overlapping 2 class:

Below is the confusion matrix



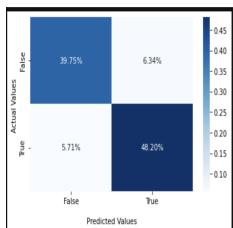
2) Non overlapping multi class:

Below is the confusion matrix



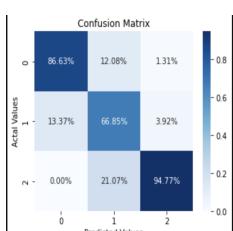
3) overlapping 2 class:

Below is the confusion matrix



4) overlapping multi class:

Below is the confusion matrix



C. Qualitative measures - Confusion Matrix based

Specificity and Sensitivity are chosen as the confusion based qualitative measures for this task.

Specificity is calculated by $TN / (TN + FP)$

Sensitivity is calculated by $TP / (TP + FN)$

But when it comes to multi class classification, both these measures are individually calculated and their average is considered as final quality measure.

All these values are calculated from the confusion matrix and measured. Below are the values of each category.

1) Non overlapping 2 class:

Below are the values

Specificity and Sensitivity

```

Sensitivity = cf_matrix[1][1] / (cf_matrix[1][1] + cf_matrix[0][1])
Specificity = cf_matrix[0][0] / (cf_matrix[1][0] + cf_matrix[0][0] + cf_matrix[1][1])
print("Sensitivity is : ",Sensitivity)
print("Specificity is : ",Specificity)
Sensitivity is : 1.0
Specificity is : 1.0

```

2) Non overlapping multi class:

Below are the values

```

Sensitivity_0 = cf_matrix[0][0] / (cf_matrix[0][0] + cf_matrix[0][1] + cf_matrix[0][2])
Sensitivity_1 = cf_matrix[1][1] / (cf_matrix[0][1] + cf_matrix[1][1] + cf_matrix[1][2])
Sensitivity_2 = cf_matrix[2][2] / (cf_matrix[0][2] + cf_matrix[1][2] + cf_matrix[2][2])
Total_Sensitivity = (Sensitivity_0 + Sensitivity_1 + Sensitivity_2)/3
Specificity_0 = (cf_matrix[1][1] + cf_matrix[2][1] + cf_matrix[1][2] + cf_matrix[2][2]) / (cf_matrix[0][0] + cf_matrix[1][0] + cf_matrix[2][0])
Specificity_1 = (cf_matrix[0][0] + cf_matrix[2][0] + cf_matrix[0][2] + cf_matrix[2][2]) / (cf_matrix[0][0] + cf_matrix[1][0] + cf_matrix[2][0])
Specificity_2 = (cf_matrix[0][0] + cf_matrix[1][0] + cf_matrix[0][1] + cf_matrix[1][1]) / (cf_matrix[0][0] + cf_matrix[1][0] + cf_matrix[0][1] + cf_matrix[1][1])
Total_Specificity = (Specificity_0 + Specificity_1 + Specificity_2)/3
print("Sensitivity is : ", Total_Sensitivity)
print("Specificity is : ", Total_Specificity)
Sensitivity is: 0.3333333333333333
Specificity is: 0.6666666666666666

```

3) overlapping 2 class:

Below are the values

Specificity and Sensitivity

```

Sensitivity = cf_matrix[1][1] / (cf_matrix[1][1] + cf_matrix[0][1])
Specificity = cf_matrix[0][0] / (cf_matrix[1][0] + cf_matrix[0][0] + cf_matrix[1][1])
print("Sensitivity is : ",Sensitivity)
print("Specificity is : ",Specificity)
Sensitivity is : 0.548762238769231
Specificity is : 0.4736842165283576

```

4) overlapping multi class:

Below are the values

```

Sensitivity_0 = cf_matrix[0][0] / (cf_matrix[0][0] + cf_matrix[1][0] + cf_matrix[2][0])
Sensitivity_1 = cf_matrix[1][1] / (cf_matrix[0][1] + cf_matrix[1][1] + cf_matrix[2][1])
Sensitivity_2 = cf_matrix[2][2] / (cf_matrix[0][2] + cf_matrix[1][2] + cf_matrix[2][2])
Total_Sensitivity = (Sensitivity_0 + Sensitivity_1 + Sensitivity_2)/3
Specificity_0 = (cf_matrix[1][1] + cf_matrix[2][1] + cf_matrix[1][2] + cf_matrix[2][2]) / (cf_matrix[0][0] + cf_matrix[1][0] + cf_matrix[2][0])
Specificity_1 = (cf_matrix[0][0] + cf_matrix[2][0] + cf_matrix[0][2] + cf_matrix[2][2]) / (cf_matrix[0][0] + cf_matrix[1][0] + cf_matrix[2][0])
Specificity_2 = (cf_matrix[0][0] + cf_matrix[1][0] + cf_matrix[0][1] + cf_matrix[1][1]) / (cf_matrix[0][0] + cf_matrix[1][0] + cf_matrix[0][1] + cf_matrix[1][1])
Total_Specificity = (Specificity_0 + Specificity_1 + Specificity_2)/3
print("Sensitivity is : ", Total_Sensitivity)
print("Specificity is : ", Total_Specificity)
Sensitivity is: 0.827526348578526
Specificity is: 0.774133840727862

```

XVII. TASK 4

A. Built-in measures

`accuracy_score` and `precision_score` from `scikitlearn metrics` are used as built in qualitative measures in this task.

accuracy and precision are calculated in earlier tasks along with confusion matrix based measures, in this task let us go ahead and print those results.

Accuracy and Precision for Lasso regression.

```
Non overlap 2 Class
Accuracy : 0.4545454545454545
Precision : 1.0
Non overlap multi Class
Accuracy : 0.027259684361549498
Precision : 0.027259684361549498
overlap 2 Class
Accuracy : 0.7674418604651163
Precision : 0.8193832599118943
overlap multi Class
Accuracy : 0.5488505747126436
Precision : 0.5941350697276772
```

Accuracy and Precision for Random Forest.

```
Non overlap 2 Class
Accuracy : 0.8033826638477801
Precision : 1.0
Non overlap multi Class
Accuracy : 0.3328550932568149
Precision : 0.3333333333333333
overlap 2 Class
Accuracy : 0.879492600422833
Precision : 0.8837209302325582
overlap multi Class
Accuracy : 0.7830459770114943
Precision : 0.8275206348570526
```

quantitative differences between confusion matrix based and built in measures:

Confusion matrix methods and built in methods give similar results. Accuracy and Precision within the library calculate them just like how we manually calculate from the confusion matrix.

Among all the results, Random forest and overlap 2 class have the best quality among all the pairs. We have used the accuracy and precision to come to this conclusion

As we can see, overlapped datasets have better accuracy and precision. As we have balanced data in those dataset, it is a more efficient way to train the model. As the model will learn different ways to predict the classes with minimal error

REFERENCES:

- 1)Evaluation measures: <https://classeval.wordpress.com/introduction/basic-evaluation-measures/>
<https://towardsdatascience.com/multi-class-metrics-made-simple-part-i-precision-and-recall-9250280bddc2>
- 2)Scatter Plot: <https://stackoverflow.com/questions/67272705/how-to-color-scatterplot-in-matplotlib-based-on-the-values-of-y-axis>
<https://www.machinelearningplus.com/plots/python-scatter-plot/>