Assignment Report

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

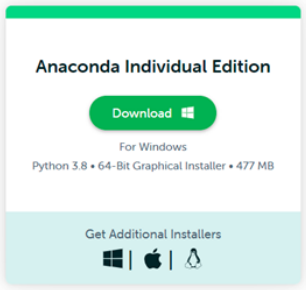
I. Assignment 01: Raw Data to Feature Space

II. Task 1

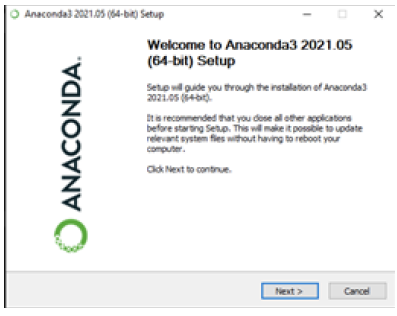
A. Building your own programming environment!

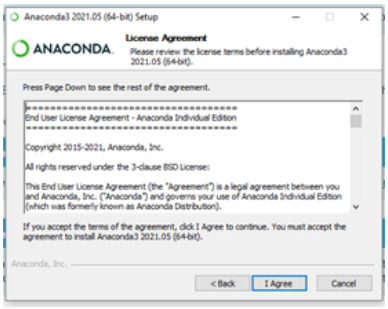
Installing the IDE on our different PCs is the first step in creating a programming environment. The instructions for installing the Anaconda environment on a PC are as follows.

1) Go to https://docs.anaconda.com/anaconda/[1] 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.



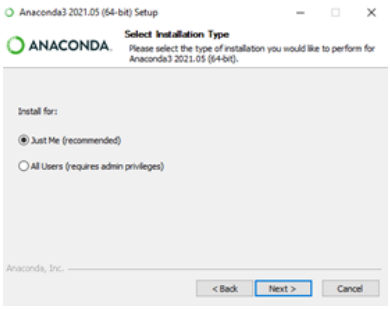
4) To run the application, double-click the installer and then select 'Next.'



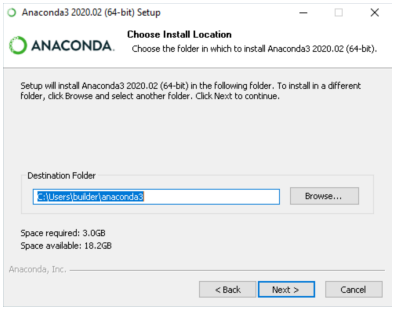


5) Read the License Agreement and the click on 'I Agree'.

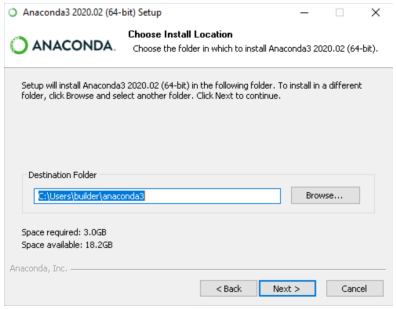
6) Then, for the installation type, select 'Just Me,' as we are simply configuring the environment for the local user. Then press the 'Next' button.



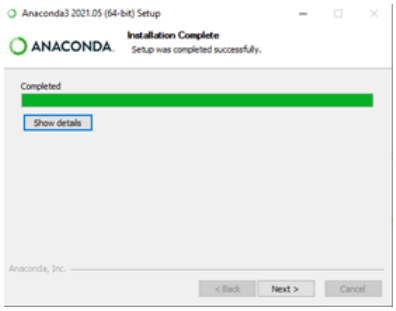
7) Select a destination folder in which to install Anaconda.



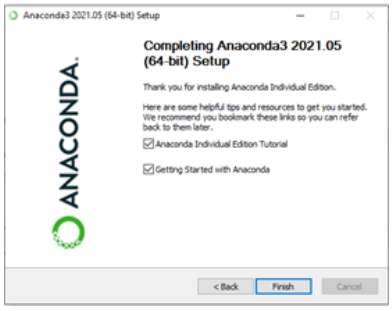
8) To avoid any conflicts with other applications installed on your computer, select 'Register Anaconda3 as my default Python 3.8' from the advanced installation menu. Then press the 'Install' button.



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

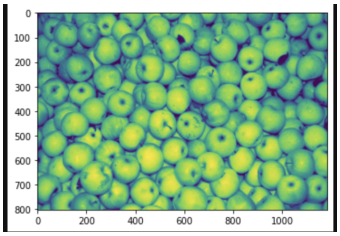


Fig. 6. Apple-RBG

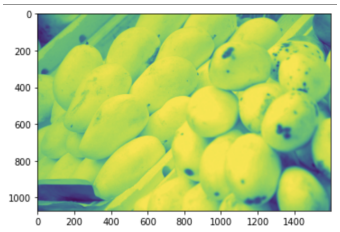


Fig. 7. mango-RBG

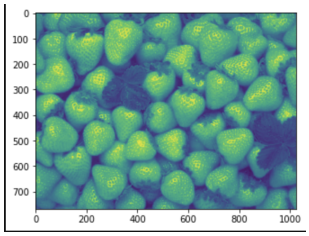


Fig. 8. Strawberry-RBG

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.

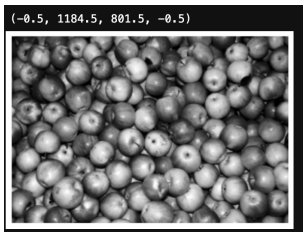


Fig. 9. apple greyscale with dimensions (-0.5, 1184.5, 801.5, -0.5)

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

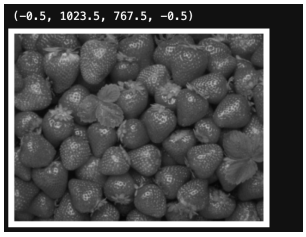


Fig. 11. strawberry with dimensions (-0.5, 1023.5, 767.5, -0.5)

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:

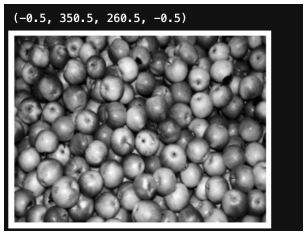


Fig. 12. Apple-Resized to (-0.5, 350.5, 260.5, -0.5)

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

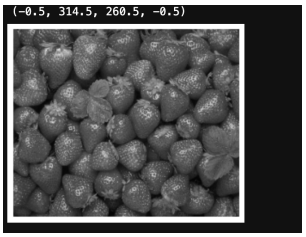


Fig. 14. Strawberry-Resized to (-0.5, 314.5, 260.5, -0.5)

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
fspaceaa = pd.DataFrame(flatss)  #panda object
fspaceaa.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

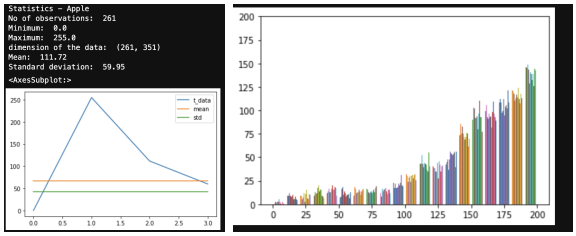
The generated spreadsheet for all three fruits is stored in the 'output$_{f}ruits'folder$.

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 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.

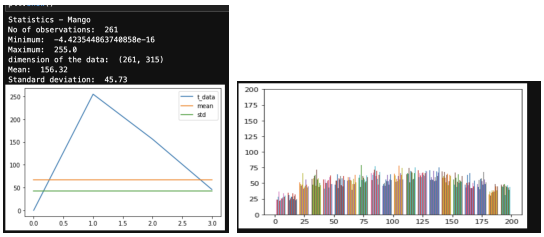


histogram:

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:

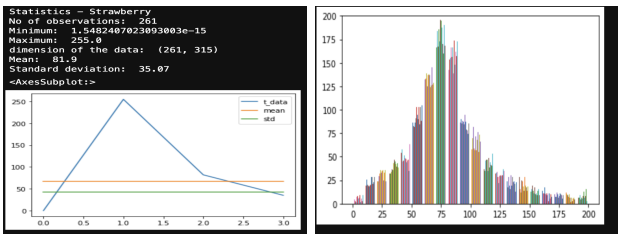


Strawberry Statistics

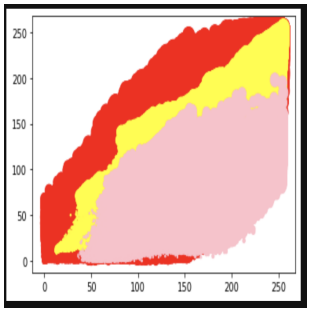
For Strawberry its selected feature have min value of 0 and max value of 255 aand it mean is 81.9 and standard deviation is 35.07. The graph plotted for the features of Strawberry shows that _ data increases from 0-1 of x-axis the standard deviation and mean remains contant and as the _ decreases from 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)
```

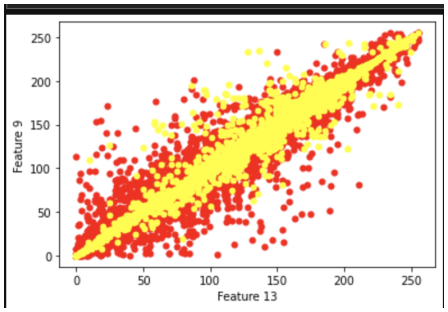
Finally, we randomize the generated image01.csv and image012.csv file and generate two new files randomimage01.csv and randomimage012.csv respectively.

X. Task 9

A. Display subspaces!

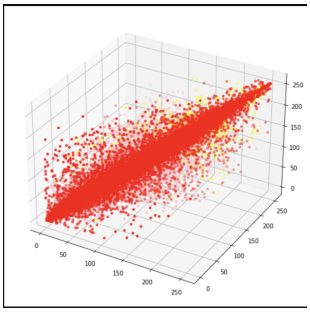
Using the spreadsheets that we prepared, we select two features and plot the two-dimensional feature space, labeling the observations (vectors) of the fruits or vegetables that we selected. To plot a two-dimensional feature space, we use the Apple and mango spreadsheet to choose the fruits.

```
#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.

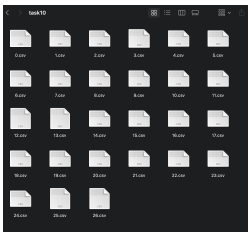
XI. Task 10 - converting all the images in a folder

Now we'll construct Python code to read any number of images from a folder of many similar images, create feature spaces, 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)
    fruitG = cv2.cvtColor(fruit_color, cv2.COLOR_BGR2GRAY)
    heightfg, widthfg, = fruitG.shape
    fruit = cv2.resize(fruitG, dsize=(315,261),interpolation

    mm = round(((heightfg)*(widthfg)))
    flatmm = np.zeros((mm, 82), np.uint8)
    for l in range(heightm-8):
        for j in range(widthm-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) + '.c
```

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/image012.csv",header=None)
overlap_2_class = pd.read_csv("../input/randomimage01.csv",header=None)
overlap_multi_class = pd.read_csv("../input/randomimage012.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)
```

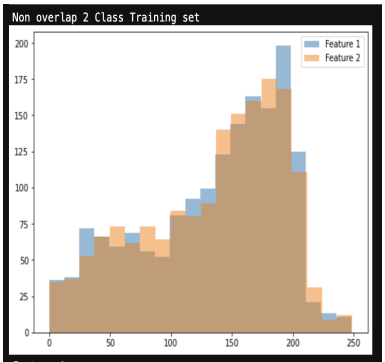
C. Selecting features and plotting their Histograms and Scatter Plots

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]
```

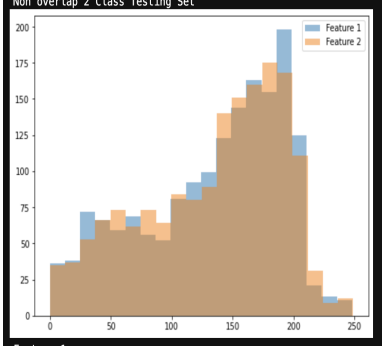
Histograms of each category and two datasets in each category are plotted.The histograms of two features are plotted in an overlapping manner to show whether they follow a similar distribution or not.
The following are the histograms and their statistics like mean and variance. In all these histograms, we can observe that the

Non overlap 2 Class Training set



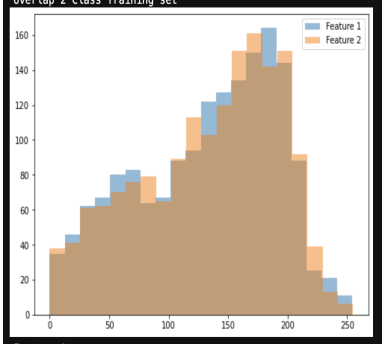

```
Feature 1
mean :  134.18589360430366
variance :  3400.3464037614003
Feature 2
mean :  134.19246861924685
variance :  3307.0119767372025
```

Non overlap 2 Class Testing Set



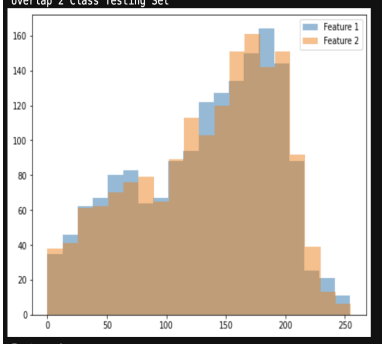

```
Feature 1
mean :  124.75475687103594
variance :  3640.6982029598307
Feature 2
mean :  127.13107822410149
variance :  3599.448883792597
```

overlap 2 Class Training set

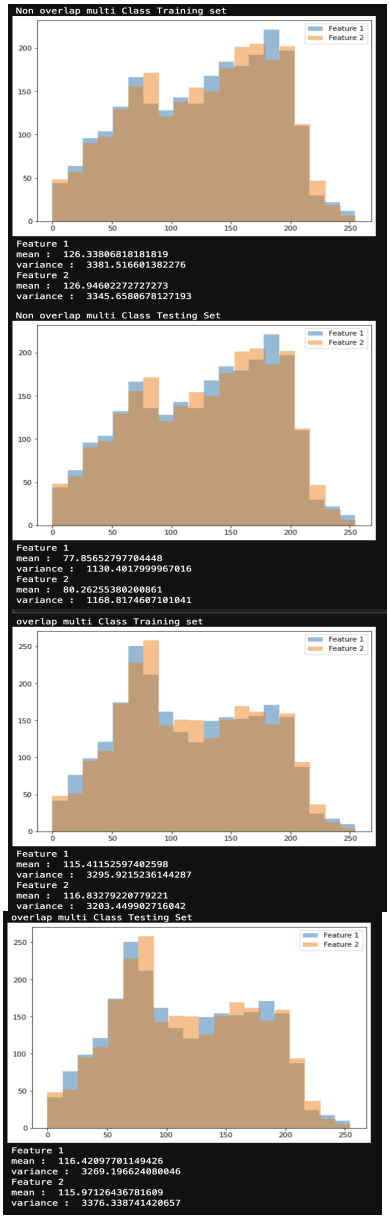


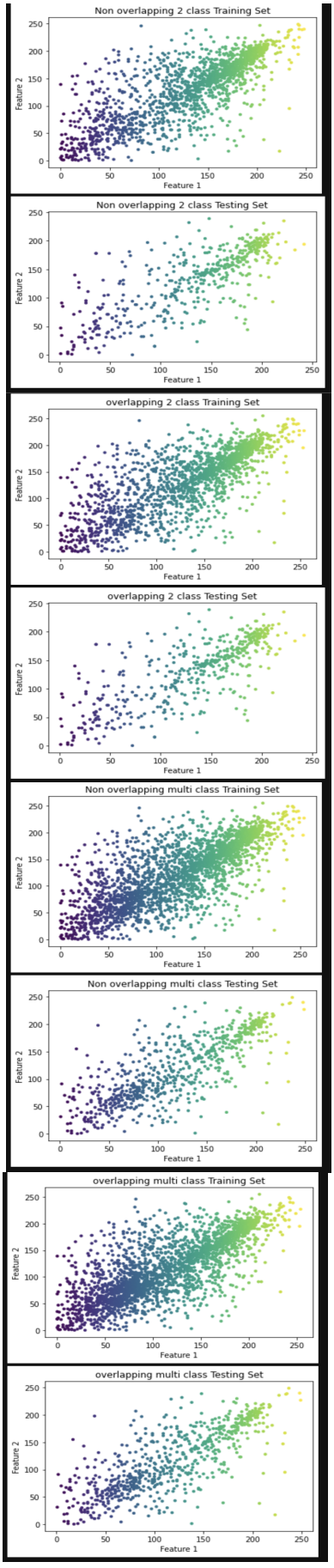

```
Feature 1
mean :  131.94019138755982
variance :  3494.845014445695
Feature 2
mean :  132.99700956937798
variance :  3441.658279502003
```

overlap 2 Class Testing Set




```
Feature 1
mean :  132.74418604651163
variance :  3381.3009459992113
Feature 2
mean :  131.42283298097252
variance :  3164.931002974164
```





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.

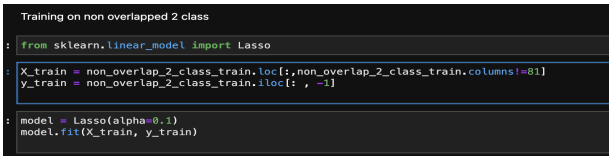
XV. TASK 2

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

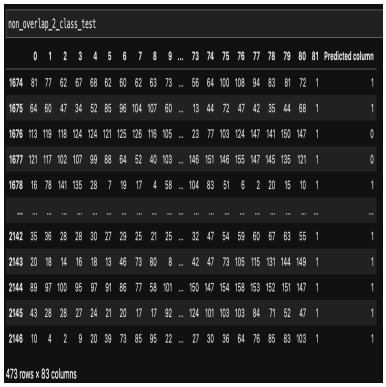
Lasso regression is implemented and used as a two class classifier to classify overlapped and non-overlapped 2 class datasets. Although it is mentioned to use lasso regression as 2 class classifier only in the subtask 1, in the subtask 2 it is written to test the test sets of all the categories of datasets, 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

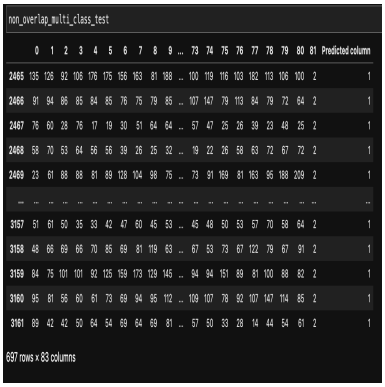


predicted results are added in an additional column as shown below.



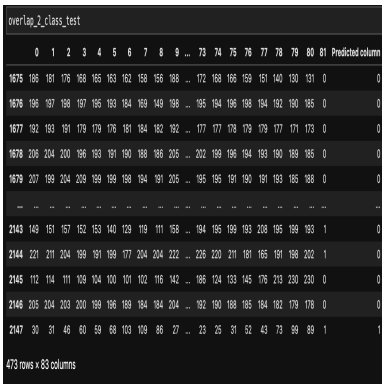
2) Non overlapping multi class:
for the non overlapping multi class, Training set is split into X_train (containing all the features) and y_train (actual labels) and the model is trained as shown below. In the same way, the model is tested on test set and the

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

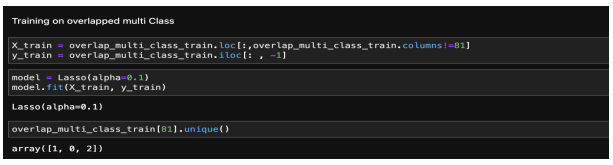


3) Overlapping 2 class
for the overlapping 2 class, Training set is split into X_train (containing all the features) and y_train (actual labels) and the model is trained as shown below. In

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



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



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

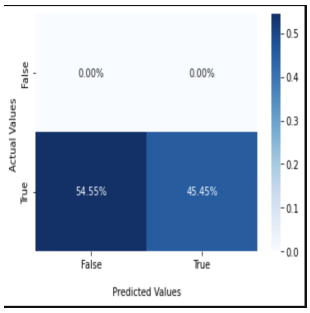
overlap_multi_class_test

	0	1	2	3	4	5	6	7	8	9	...	73	74	75	76	77	78	79	80	81	Predicted column
2466	65	42	28	78	179	193	188	173	166	68	...	87	99	115	130	146	145	150	158	1	1
2467	113	127	147	164	135	136	139	131	135	124	...	149	153	167	176	182	178	183	175	1	1
2468	185	182	180	173	173	169	167	164	163	173	...	159	164	160	155	156	153	161	175	0	0
2469	36	33	41	48	57	47	54	47	48	30	...	135	131	64	50	38	66	128	139	2	1
2470	206	209	208	206	197	188	165	158	155	206	...	158	157	155	154	156	164	155	159	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
3158	197	198	199	204	200	202	198	203	206	197	...	193	204	198	203	204	198	200	200	0	0
3159	230	235	182	168	162	160	157	157	159	234	...	179	182	181	177	164	165	155	157	0	0
3160	179	175	173	168	176	180	171	171	169	179	...	170	168	168	169	166	164	164	163	0	0
3161	202	200	202	202	202	204	204	203	199	202	...	198	197	200	203	204	204	202	200	0	0
3162	63	27	2	36	32	40	48	52	75	62	...	18	11	31	60	63	64	75	96	1	2

696 rows x 83 columns

B. Construct Confusion matrix for each category

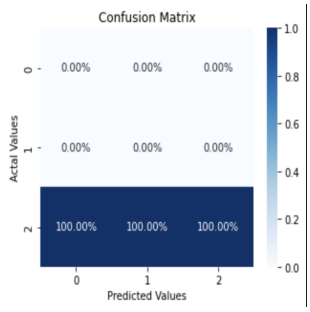
1) Non overlapping 2 class:
   Below is the confusion matrix
   We can see that our model is poorly trained, this



is because the data in the training set is imbalanced, the ratio of the class labels is not equal. This can be understood by looking at the below value counts.

```
non_overlap_2_class_train[81].value_counts()

0    1015
1     658
Name: 81, dtype: int64
```

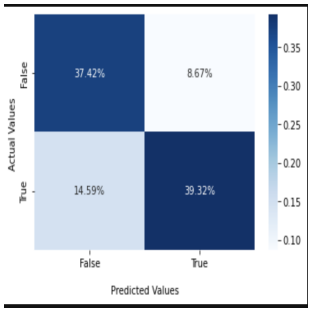
2) Non overlapping multi class:
   Below is the confusion matrix
   We can see that our model is poorly trained, this



is because the data in the training set is imbalanced, the ratio of the class labels is not equal. This can be understood by looking at the below value counts.

```
non_overlap_multi_class_train[81].value_counts()

1    1131
0    1015
2     318
Name: 81, dtype: int64
```

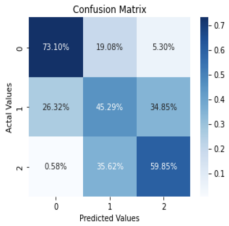
3) overlapping 2 class:
   Below is the confusion matrix
   We can see that our model is properly trained, this



is because the data in the training set is balanced, the ratio of the class labels is almost equal. This can be understood by looking at the below value counts.

```
overlap_2_class_train[81].value_counts()

1    876
0    796
Name: 81, dtype: int64
```

4) overlapping multi class:
   Below is the confusion matrix
   We can see that our model is properly trained, this



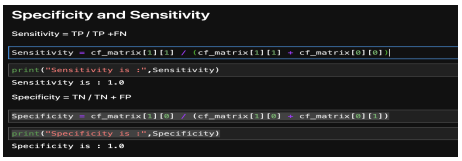
is because the data in the training set is balanced, the ratio of the class labels is almost equal. This can be understood by looking at the below value counts.

```
overlap_multi_class_train[81].value_counts()

1    862
0    807
2    795
Name: 81, dtype: int64
```

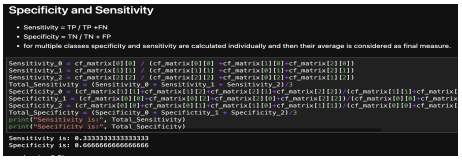
C. Qualitative measures - Confusion Matrix based

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.

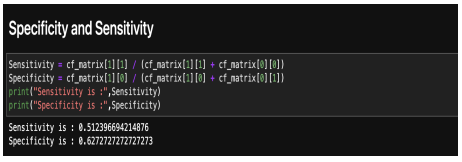
1) Non overlapping 2 class:
   Below are the values



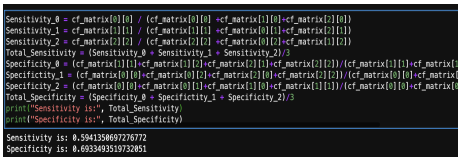
2) Non overlapping multi class:
   Below are the values



3) overlapping 2 class:
   Below are the values



4) overlapping multi class:
   Below are the values



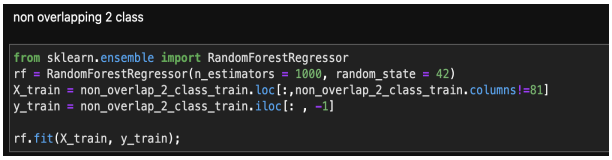
XVI. TASK 3

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

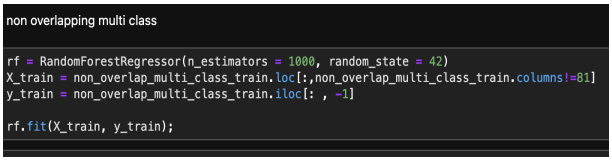
Random Forest is implemented and used as a two class classifier and multi class classifier to classify all the 4 categories of data.

Training the model and Testing the results:

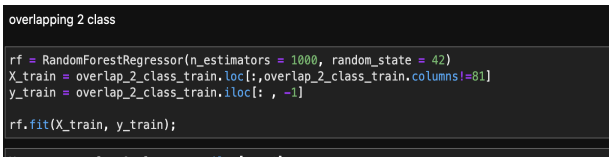
1) Non overlapping 2 class:
   for the non overlapping two class, Training set is split into X_train (containing all the features) and y_train (actual labels) and the model is trained as shown below. In the same way, the model is tested on test set and the



   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



   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

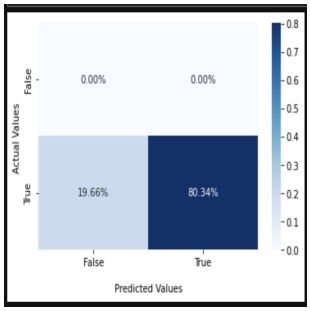


   the same way, the model is tested on test set and the predicted results are added in an additional column.
4) Overlapping multi class
   or the overlapping multi class, Training set is split into X_train (containing all the features) and y_train (actual labels) and the model is trained as shown below. In

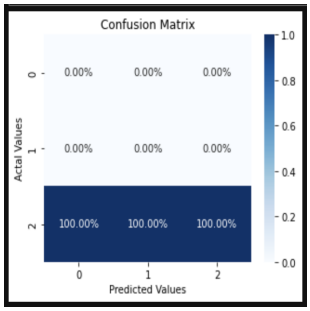
   the same way, the model is tested on test set and the predicted results are added in an additional column.

B. Construct Confusion matrix for each category

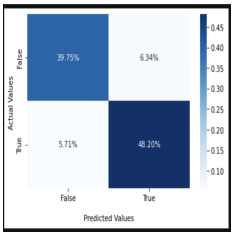
1) Non overlapping 2 class:
Below is the confusion matrix



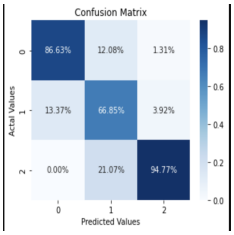
2) Non overlapping multi class:
Below is the confusion matrix



3) overlapping 2 class:
Below is the confusion matrix



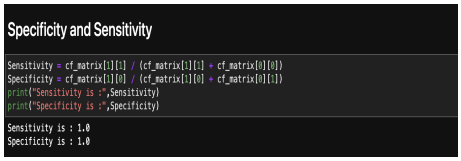
4) overlapping multi class:
Below is the confusion matrix



C. Qualitative measures - Confusion Matrix based

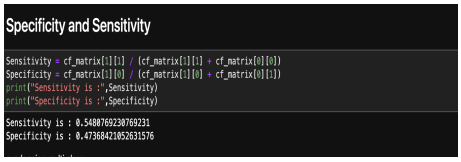
Specificity and Sensitivity are chosen as the confusion based qualitative measures for this task.
Specificity is calculated by TN/TN+FP
Sensitivity is calculated by TP/TP+FN
But when it comes to multi class classification, both these measures are individually calculated and their average is considered 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

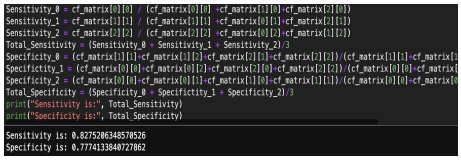


2) Non overlapping multi class:
Below are the values

3) overlapping 2 class:
Below are the values



4) overlapping multi class:
Below are the values



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 : 0.45454545454545453
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: $https : //stackoverflow.com/questions/67272705/how - to - color - scatterplot - in - matplotlib - based - on - the - values - of - y - axis$
$https : //www.machinelearningplus.com/plots/python - scatter - plot/$