Assignment 1

Vathana Him

September 12, 2021

1 Abstract

The purpose of this assignment was to begin the introduction to data preprocessing as a preparation step for machine learning and deep learning. This assignment utilized images from UCI respository as sample datasets that will be used to train a machine learning model. Images that were processed represented three fruits spanish pear, fuji apple, watermelon.

The data for these images were processed using the OpenCV python library. The main goals for this assignment were to process the data for each respective images into their appropriate feature space and feature vector, analyze thess datasets to see if there were any imbalance, explore if any standardization was needed, and export these data into appropriate .csv format to be used as training datasets for machine learning.

2 Setting Up

The developing environment was installed by going to the anaconda website and downloading anaconda. Then anaconda was launched. Once it was launched, navigate to the environment tab, then the create button was pressed in order to create the new working environment. The enviroment was named as MSIA. Within the environment, python 3.7 was installed. Subsequently, pip was used to install a list of python libraries. These libraries included Opencv, Pandas, Plotly, Matplotlib, and Numpy.

2.1 Section 1 Figures

Figure 1: environment

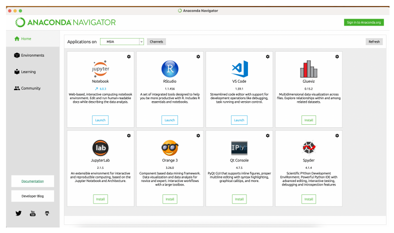


Figure 2: Pip Command

```
Ex.
pip install pandas
pip install opencv
```

3 Choosen Images

The choosen images were gathered from the UCI data respository. The

[1]https://www.ic.unicamp.br/ rocha/pub/downloads/

link for these images can be found here [1]. This dataset contains a variety of fruits and vegatables but only three images were choosen for this assignment. These image were spanish pear, fuji apple, and watermelon. Each of these images respresent a fuji apple, spanish pear, and watermelon respectively. These images were chosen due to the nature of their color as apple was a bright red color, spanish pear was a bright yellow, and watermelon was green. Intuitively, the human eyes was able to distinguish the differences betweent these colors easily. In correlation, computer vision should be able to distinguish the same differences due to its difference in RGB value. Subsequently, these images also have consistent dimensions which makes scaling and resizing of images more consistent.

3.1 Section 3 Figures

Figure 3: Image Folders

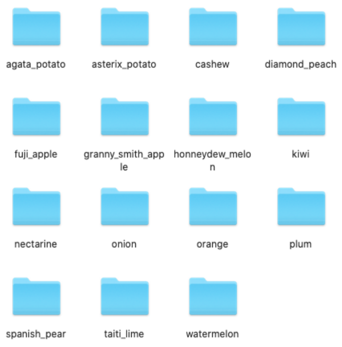


Figure 4: Spanish Pear

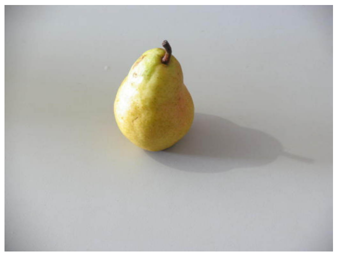


Figure 5: Fuji Apple

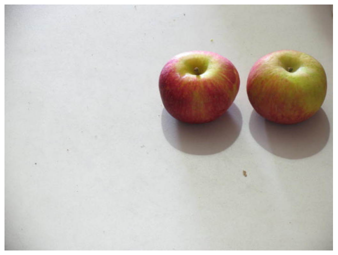
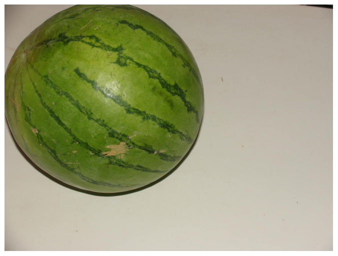


Figure 6: Watermelon



4 Task 3

The code to display the three chosen images using python and opencv library is displayed below.

4.1 Task 3 Results

Figure 7: Apple RGB

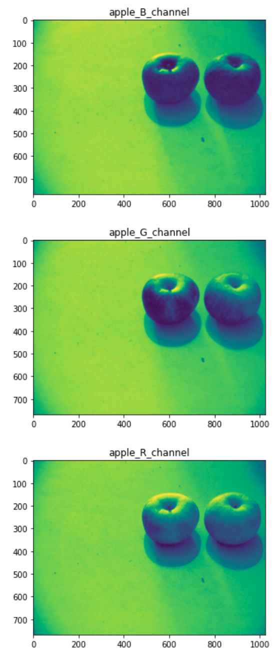


Figure 8: Pear RGB

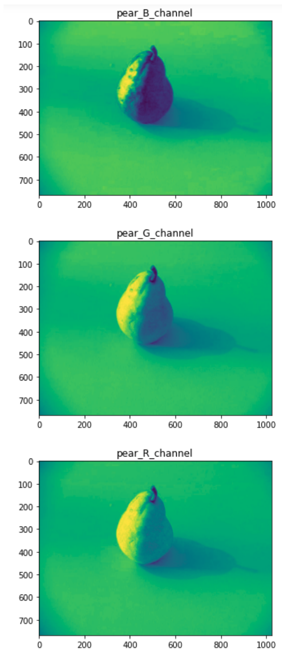


Figure 9: Watermelon RGB

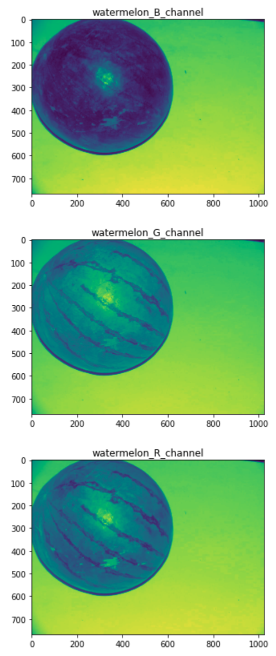
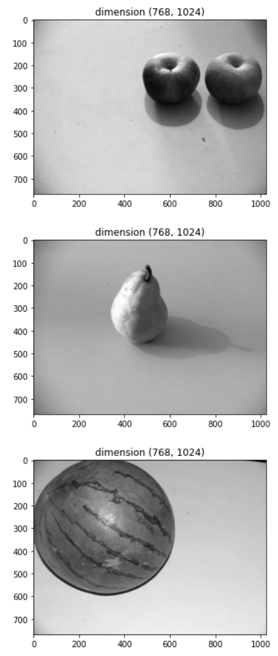


Figure 10: Gray scale



4.2 Code

```
import cv2 as cv
import sys
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import numpy as np
import pandas as pd
img_apple = cv.imread("../data/fuji_apple_001.jpg")
img_pear = cv.imread("../data/spanish_pear_001.jpg")
img_watermelon = cv.imread("..data/watermelon_001.jpg")

img_list = [img_apple, img_pear, img_watermelon]

title = ["apple","pear","watermelon"]

color_channel = ["B_channel", "G_channel", "R_channel"]

num = 0
for i in img_list:
n = 0
#split color channel of each image by B, G, R
for j in cv.split(i):
plt.figure()
plt.imshow(j)
plt.title(title[num]+"_"+color_channel[n])
n = n+1
num = num+1

img_list_gray = []
for i in img_list:
img = cv.cvtColor(i, cv.COLOR_BGR2GRAY)
img_list_gray.append(img)
plt.figure()
plt.imshow(img, cmap='gray')
plt.title("dimension_{}".format(img.shape))
```

5 Task 4

The grayscale images were then resized and their new dimensions were displayed through matplotlib. There was a function that was used to help create this gray scale image so it can be generalized for future use.

5.1 Task 4 Result

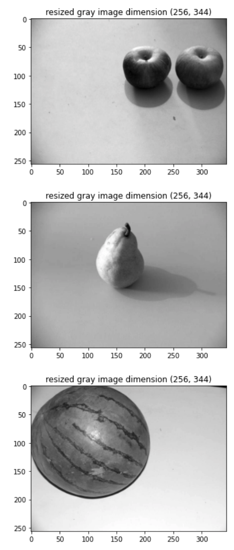

resized gray image dimension (256, 344)
resized gray image dimension (256, 344)
resized gray image dimension (256, 344)

Figure 11: Gray scale

5.2 Code

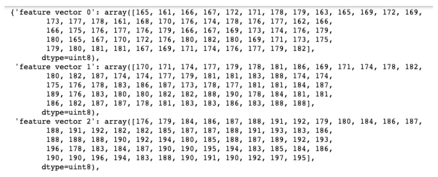
```
gray_imgresize = []

for i in img_list_gray:
temp_img = img_resize(344, 256, i)
gray_imgresize.append(temp_img)
for i in gray_imgresize:
plt.figure()
plt.imshow(i, cmap='gray')
plt.title("resized gray image dimension {}".format(i.shape))
```

6 Task 5

The grayscale images were then divided into blocks of 8X8 pixels image. Each feature was then assigned a label respectively to identify them.

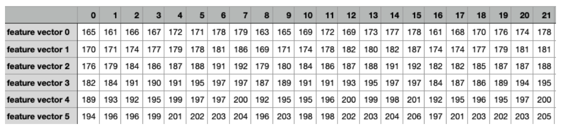
6.1 Task 5 Results



```
{'feature vector 0': array([165, 161, 166, 167, 172, 171, 178, 179, 163, 165, 169, 172, 169,
        173, 177, 178, 161, 168, 170, 176, 174, 178, 176, 177, 162, 166,
        166, 175, 176, 177, 176, 179, 166, 167, 169, 173, 174, 176, 179,
        180, 165, 167, 170, 172, 176, 180, 182, 180, 169, 171, 173, 175,
        179, 180, 181, 181, 167, 169, 171, 174, 176, 177, 179, 182],
       dtype=uint8),
 'feature vector 1': array([170, 171, 174, 177, 179, 178, 181, 186, 169, 171, 174, 178, 182,
        180, 182, 187, 174, 174, 177, 179, 181, 181, 183, 188, 174, 174,
        175, 176, 178, 183, 186, 187, 173, 178, 177, 181, 181, 184, 187,
        189, 176, 183, 180, 180, 182, 182, 188, 190, 178, 184, 181, 181,
        186, 182, 187, 187, 178, 181, 183, 183, 186, 183, 188, 188],
       dtype=uint8),
 'feature vector 2': array([176, 179, 184, 186, 187, 188, 191, 192, 179, 180, 184, 186, 187,
        188, 191, 192, 182, 182, 185, 187, 187, 188, 191, 193, 183, 186,
        188, 188, 188, 190, 192, 194, 180, 185, 188, 187, 189, 192, 193,
        196, 178, 183, 184, 187, 190, 190, 195, 194, 183, 185, 184, 186,
        190, 190, 196, 194, 183, 188, 190, 191, 190, 192, 197, 195],
       dtype=uint8),
```

Figure 12: Feature Vector Output

Figure 13: CSV Output



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
feature vector 0	165	161	166	167	172	171	178	179	163	165	169	172	169	173	177	178	161	168	170	176	174	178
feature vector 1	170	171	174	177	179	178	181	186	169	171	174	178	182	180	182	187	174	174	177	179	181	181
feature vector 2	176	179	184	186	187	188	191	192	179	180	184	186	187	188	191	192	182	182	185	187	187	188
feature vector 3	182	184	191	190	191	195	197	197	187	189	191	191	193	195	197	197	184	187	186	189	194	195
feature vector 4	189	193	192	195	199	197	197	200	192	195	195	196	200	199	198	201	192	195	196	195	197	200
feature vector 5	194	196	196	199	201	202	203	204	196	203	198	198	202	203	204	206	197	201	203	202	203	205

6.2 Code

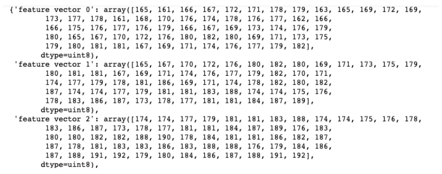
```
#convert each image into blocks of 8X8 and store them in a list
img_blocks = []
for i in gray_imgresize:
img_blocks.append(con_to_img_blocks(i))
#convert to dictionary of feature value for each image and store all ima
img_feature_list = []
for i in img_blocks:
img_feature_list.append(label_feature(i))
#convert each feature of size 64 vectirs into csv
for i in range(len(img_feature_list)):
if (i==0):
s='img_0'
elif (i==1):
s="img_1"
else:
s="img_2"
temp_df = to_dataframe(img_feature_list[i])
temp_df.to_csv("img_csvs/"+"{}.csv".format(s))
```

7 Task 6

After being divided into 8X8 pixel blocks, the grayscale images were then divided into sliding block of 8X8 pixels. Each feature was then assigned a label respectively to identify them. Two helper functions were used for this task, sliding blocks feature function which converts an image into sliding image blocks given an image object, and label feature function which create feature labels for each image given a list that contains the array of sliding images.

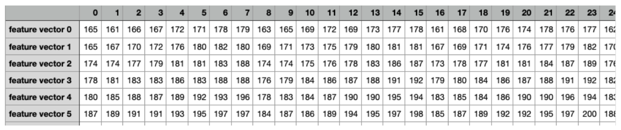
7.1 Task 6 Results

Figure 14: Feature Vector Output



```
{'feature vector 0': array([165, 161, 166, 167, 172, 171, 178, 179, 163, 165, 169, 172, 169,
        173, 177, 178, 161, 168, 170, 176, 174, 178, 176, 177, 162, 166,
        166, 175, 176, 177, 176, 179, 166, 167, 169, 173, 174, 176, 179,
        180, 165, 167, 170, 172, 176, 180, 182, 180, 169, 171, 173, 175,
        179, 180, 181, 181, 167, 169, 171, 174, 176, 177, 179, 182],
       dtype=uint8),
 'feature vector 1': array([165, 167, 170, 172, 176, 180, 182, 180, 169, 171, 173, 175, 179,
        180, 181, 181, 167, 169, 171, 174, 176, 177, 179, 182, 170, 171,
        174, 177, 179, 178, 181, 186, 169, 171, 174, 178, 182, 180, 182,
        187, 174, 174, 177, 179, 181, 181, 183, 188, 174, 174, 175, 176,
        178, 183, 186, 187, 173, 178, 177, 181, 181, 184, 187, 189],
       dtype=uint8),
 'feature vector 2': array([174, 174, 177, 179, 181, 181, 183, 188, 174, 174, 175, 176, 178,
        183, 186, 187, 173, 178, 177, 181, 181, 184, 187, 189, 176, 183,
        180, 180, 182, 182, 188, 190, 178, 184, 181, 181, 186, 182, 187,
        187, 178, 181, 183, 183, 186, 183, 188, 188, 176, 179, 184, 186,
        187, 188, 191, 192, 179, 180, 184, 186, 187, 188, 191, 192],
       dtype=uint8),
```

Figure 15: CSV Output



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	2
feature vector 0	165	161	166	167	172	171	178	179	163	165	169	172	169	173	177	178	161	168	170	176	174	178	176	177	16
feature vector 1	165	167	170	172	176	180	182	180	169	171	173	175	179	180	181	181	167	169	171	174	176	177	179	182	17
feature vector 2	174	174	177	179	181	181	183	188	174	174	175	176	178	183	186	187	173	178	177	181	181	184	187	189	17
feature vector 3	178	181	183	183	186	183	188	188	176	179	184	186	187	188	191	192	179	180	184	186	187	188	191	192	18
feature vector 4	180	185	188	187	189	192	193	196	178	183	184	187	190	190	195	194	183	185	184	186	190	190	196	194	18
feature vector 5	187	189	191	191	193	195	197	197	184	187	186	189	194	195	197	198	185	187	189	192	192	195	197	200	18

7.2 Code

```
#convert each image into sliding blocks of 8X8 and store them in a list each ind
sliding_block_list = [sliding_blocks(i)for i in gray_imgresize]


#convert to dictionary of feature value for each image in sliding block and stor
sliding_img_feature_list = [label_feature(i) for i in sliding_block_list]
```

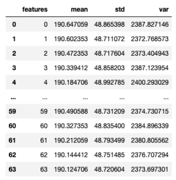
8 Task 7

The dimension of the gray images were resized to height of 256 and width of 344. The purpose for these choosen dimensions was to keep its aspect ratio. Additionally, the resized height and width must also be divible by eight since this project divided the targeted image into sliding blocks of 8 by 8 height and width. The feature vector that was constructed from these images created 3400 feature vectors and each feature vector lies a 8 by 8 pixels who's value lies between 0-255 of the gray image scale. The features of each feature vector was then flatten to 64 features for each respective feature vector.

The mean of each feature lies between 189-190 for img0 apple, img1 177-178 pear, and img2 148-151 watermelon. This data does not need normalization as its mean feature values seem to correlate to the color of each fruit that belong to their respective image. For example, both the color for apple and pear are brighter than that of the color for watermelon, therefore, it should intuitively be true that the mean of each feature in the watermelon image should be lower than that of the brighter colored images. Subsequently, the standard deviation value of each image was normally distributed according to the histogram. The mean and variance of img1 and img2 were relatively similar, this indicates that normalization may not be needed. However, normalization may be needed for img0 because its histogram seemed to have a two tail distribution. Its mean and variance values also deviated from the subsequent two images. For img1 and img2 normalization was not needed since their values were normally distributed. This suggests that the data of feature value across all feature vectors was consistent and balanced, thus data transformation may not be needed.

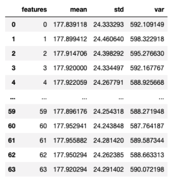
8.1 Task 7 Results

Figure 16: img0 summary stats



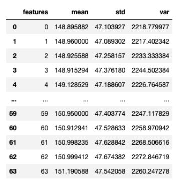
	features	mean	std	var
0	0	190.647059	48.865398	2387.827146
1	1	190.602353	48.711072	2372.768573
2	2	190.472353	48.717604	2373.404943
3	3	190.339412	48.858203	2387.123954
4	4	190.184706	48.992785	2400.293029
...	...	...	...	...
59	59	190.490588	48.731209	2374.730715
60	60	190.327353	48.835400	2384.896339
61	61	190.212059	48.793499	2380.805562
62	62	190.144412	48.751485	2376.707294
63	63	190.124706	48.720604	2373.697301

Figure 17: img1 summary stats



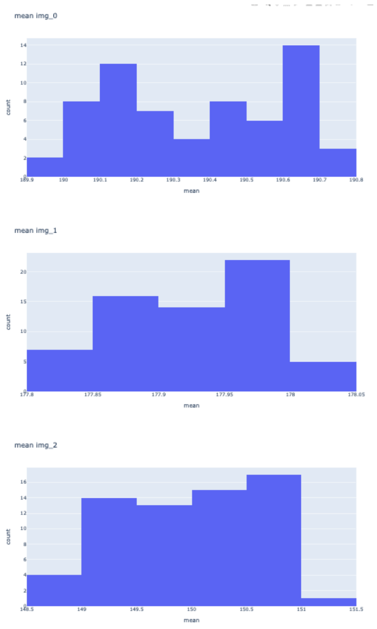
	features	mean	std	var
0	0	177.839118	24.333293	592.109149
1	1	177.899412	24.460640	598.322918
2	2	177.914706	24.398292	595.276630
3	3	177.920000	24.334497	592.167767
4	4	177.922059	24.267791	588.925668
...	...	...	...	...
59	59	177.896176	24.254318	588.271948
60	60	177.952941	24.243848	587.764187
61	61	177.955882	24.281420	589.587344
62	62	177.950294	24.262385	588.663313
63	63	177.920294	24.291402	590.072198

Figure 18: img2 summary stats



	features	mean	std	var
0	0	148.895882	47.103927	2218.779977
1	1	148.960000	47.089302	2217.402342
2	2	148.925588	47.258157	2233.333384
3	3	148.915294	47.376180	2244.502384
4	4	149.128529	47.188607	2226.764587
...	...	...	...	...
59	59	150.950000	47.403774	2247.117829
60	60	150.912941	47.528633	2258.970942
61	61	150.998235	47.628842	2268.506616
62	62	150.999412	47.674382	2272.846719
63	63	151.190588	47.542058	2260.247278

Figure 19: Histogram



8.2 Code

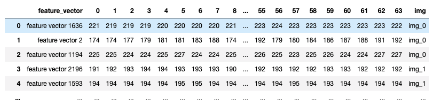
```
import matplotlib.pyplot as plt
import plotly.express as px
for i in sliding_block_list:
print(i.shape)
#function to calculate mean, var, std of each feature for each immage, convert i
def get_stats_img(feature_list):
stat_dic_img = {}
for i in range(len(feature_list)):
temp_df = to_dataframe(feature_list[i])
temp_df_mean = temp_df.mean().to_frame().reset_index()
temp_df_std = temp_df.std().to_frame().iloc[:,0].values
temp_df_var = temp_df.var().to_frame().iloc[:,0].values
temp_df_mean['std'],temp_df_mean['var'] = [temp_df_std, temp_df_var]
temp_df_mean.columns = ['features', 'mean', 'std', 'var']
stat_dic_img['img_{}'.format(i)] = temp_df_mean
return stat_dic_img
stat_dic_img = get_stats_img(sliding_img_feature_list)
for i in stat_dic_img.keys():
fig = px.histogram(stat_dic_img[i], x=stat_dic_img[i]['mean'], title="mean_{}".f
fig.show()
```

9 Task 8

The csvs of image0 and image1 were merged to create a feature space of these images. Similary, a feature space of image0, image1, and image2 was also created by merging their cvs together. Each of these feature spaces were then randomized so it could be used to train a machine learning model for the next assginment.

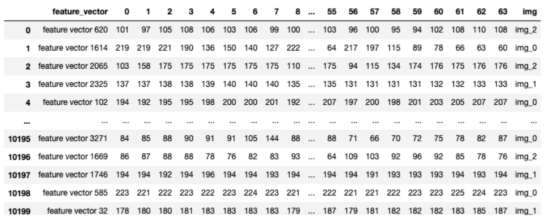
9.1 Task 8 Results

Figure 20: Merge img01



	feature_vector	0	1	2	3	4	5	6	7	8	...	55	56	57	58	59	60	61	62	63	img
0	feature vector 1636	221	219	219	219	220	220	220	220	221	...	223	224	223	223	223	223	223	223	222	img_0
1	feature vector 2	174	174	177	179	181	181	183	188	174	...	192	179	180	184	186	187	188	191	192	img_0
2	feature vector 1194	225	225	224	224	225	227	224	224	225	...	226	225	223	225	226	224	224	227	227	img_0
3	feature vector 2196	191	192	193	194	194	193	193	193	190	...	192	193	192	192	193	193	192	192	192	img_1
4	feature vector 1593	194	194	194	194	194	195	195	194	194	...	194	194	195	194	193	194	194	194	194	img_1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

Figure 21: Merge img012



	feature_vector	0	1	2	3	4	5	6	7	8	...	55	56	57	58	59	60	61	62	63	img
0	feature vector 620	101	97	105	108	106	103	106	99	100	...	103	96	100	95	94	102	108	110	108	img_2
1	feature vector 1614	219	219	221	190	136	150	140	127	222	...	64	217	197	115	89	78	66	63	60	img_0
2	feature vector 2065	103	158	175	175	175	175	175	175	110	...	175	94	115	134	174	176	175	176	176	img_2
3	feature vector 2325	137	137	138	138	139	140	140	140	135	...	135	131	131	131	131	132	132	133	133	img_1
4	feature vector 102	194	192	195	195	198	200	200	201	192	...	207	197	200	198	201	203	205	207	207	img_0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
10195	feature vector 3271	84	85	88	90	91	91	105	144	88	...	88	71	66	70	72	75	78	82	87	img_0
10196	feature vector 1669	86	87	88	88	78	76	82	83	93	...	64	109	103	92	96	92	85	78	76	img_2
10197	feature vector 1746	194	194	192	194	196	194	194	193	194	...	194	194	191	193	193	193	194	193	194	img_1
10198	feature vector 585	223	221	222	223	222	223	224	223	221	...	222	221	221	222	223	223	225	224	223	img_0
10199	feature vector 32	178	180	180	181	183	183	183	183	179	...	187	179	181	182	182	182	183	185	187	img_1

9.2 Code

```
img_0 = pd.read_csv('../data/sliding_img_csvs/sliding_img_0.csv',
index_col=0)
img_1 = pd.read_csv('../data/sliding_img_csvs/sliding_img_1.csv',
index_col= 0)
img_2 = pd.read_csv('../data/sliding_img_csvs/sliding_img_2.csv',
index_col=0)
img_0['img'] = "img_0"
img_1['img'] = "img_1"
img_2['img'] = "img_2"
image01 = pd.concat([img_0, img_1]).sample(frac=1)
image012 = pd.concat([img_0,img_1,img_2]).sample(frac=1)
image01.to_csv('../data/combined_img_csvs/image01.csv')
image012.to_csv('../data/combined_img_csvs/image012.csv')
```

10 Task 9

The 2-d feature space plot that showed observations for image0 and image1 indicated that each of their respective feature value was similar. One explaination for this could be that the intensity of their gray color channels were alike. Additionally, there were also some observations that fell out of the mean cluster of the plot. It could be interpreted that these observations represent the color of the different fruits, whereas the observations in the mean of the cluster represent the white spaces of the images. Similarly, when three features of image0 and image1 were selected, there were some observations that fell out the mean cluster, while most observations were within. This relationship supported the earlier analysis that was stated.

In order to further understand this relationship, a feature space of three class label was plotted. The 2-d feature space plot for the three class labels of their three respective images showed a similar relationship to the two class label plots. The majority of observations within each class lied within the mean cluster as some observations lied outside the mean. Futhermore, this behavior was also observed in the 3-d three features plot where most observations lied within the mean cluster and some observations were outside the mean cluster. This indicated that the majority of the observations between each image had similar color on the gray color scale as the outliers could represent the color on the gray color scale that were indicative to the actual color of each fruits. Thus, further supporting the initial analysis. Finally, it can be concluded that a feature selection method may be needed in order to eliminate features of covariance.

10.1 Task 9 Results

Figure 22: 2-D Feature Space img01

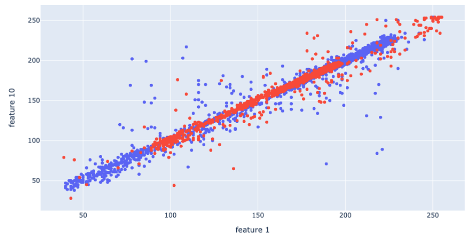


Figure 23: 3-D Feature Space img01

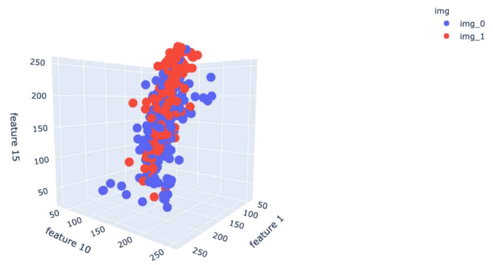


Figure 24: 2-D Feature Space img012

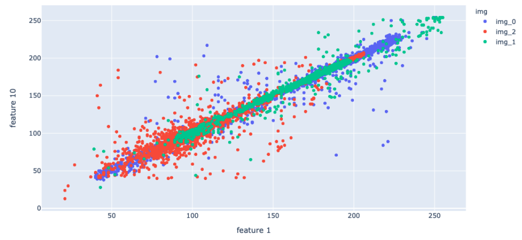
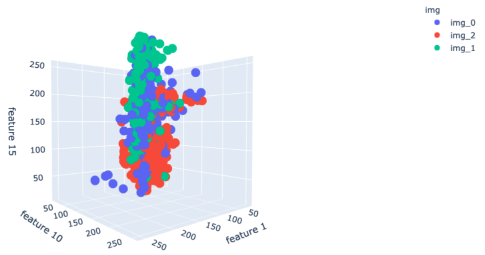


Figure 25: 3-D Feature Space img012



10.2 Code

```
fig = px.scatter(df_image01, x='0', y='10', color='img', labels={'10':'feature␣1
fig.show()
fig = px.scatter_3d(df_image01, x='0', y='10', z='15', color='img',
labels={'10':'feature␣10', '0':'feature␣1', '15':'feature␣15'})
fig.show()
fig = px.scatter(df_image012, x='0', y='10', color='img',
labels={'10':'feature␣10', '0':'feature␣1'})
fig.show()
```

```
fig = px.scatter_3d(df_image012, x='0', y='10', z='15', color='img',
labels={'10':'feature_10', '0':'feature_1', '15':'feature_15'})
fig.show()
```

11 Task 10

Below lists a set of helper functions that can read any number of images from a folder, resize image, convert each image to blocks of 8X8, convert each image to sliding blocks of 8X8, create a dataframe for the feature vectores of each image, and generate the appropriate cvs. These functions was created so that it can be generally used by passing in a python object and it would generate the desired result.

11.1 Code

```
#helper function

#def get image from given directory
def get_img(path:str):
img_paths = glob.glob('../data/*.jpg')
return img_paths
#read image given a path
def read_img(path:str):
img = cv.imread(path)
return img
#resize image given image object and its width and height
def img_resize(width, height, img):
dim = (width, height)
resized = cv.resize(img, dim, interpolation = cv.INTER_AREA)
return resized

#divide image into 8 by 8 blocks
def con_to_img_blocks(img):
blocks = np.array([img[i:i+8, j:j+8] for j in range(0,344,8) for i in range(0,25
return blocks

#divide image into sliding blocks
def sliding_blocks(img):
step = 5
(w_height, w_width) = (8,8)
slidingblocks = np.array([img[i:i+w_height, j:j+w_width] for j in range(0, img.s
return slidingblocks

#convert each feature of an image into a list of 64 values and add it dictionary
#return a dictionary of an image with each key as a feature of that image
def label_feature(img):
```

```
block_dic = {}
for i in range(len(img)):
block_dic["feature_vector_{}".format(i)] = img[i].ravel()
return block_dic

#convert a dictionary of an image into pandas dataframe
def to_dataframe(img_dic):
df = pd.DataFrame(img_dic).T
return df

#generate csv given list of dataframe
def to_csv(df_list):
for i in range(len(df_list)):
if (i==0):
s='img_0'
elif (i==1):
s="img_1"
else:
s="img_2"
temp_df = to_dataframe(df_list[i])
temp_df.to_csv("sliding_img_csvs/"+"{}.csv".format(s))
```

12 Task 11

The dimension of original gray images were 256H by 344W. Once divided into sliding blocks, their dimensions were reduced to 8H by 8W of 3400 feature vectors. The pixels representing these blocks were then converted into a one-dimensional array that contains 64 features. Therefore, each feature vectors contain 64 features. These new dimensions were also labeled to represent their images respectively. The conversion of these dimensions to a feature space represented by 64 features would allow machine learning models to distinguish unique features among these images. Classification methods used would allow machine learning modes to understand the differences between the color scale of each pixels within each feature in relation to each feature space and the feature space that belong to their respective image. It would allow machine learning models to understand the differences between each image based on their feature and feature space. Therefore, enabling it to classify future images based on commonality of characteristics from each unique features.