Advanced Machine Learning

CIS550

Spring ‘24

Lab Homework 2

Submitted by :

Name : Manas Vishal (01971464)

Email : mvishal@umassd.edu

Title : Homework on exploring datasets and its analysis

Exploring the data

The goal is to explore the dataset and its classification. We also aim to study the data by their correlation values. The data was recorded by Dr. Henrique da Mota during a medical residence period in the Group of Applied Research in Orthopedics (GARO) of the Centre Médico-Chirurgical de Réadaptation des Massues, Lyon, France. The data has been organized in two different, but related, classification tasks.

The features of the data are:

- Pelvic incidence
- Pelvic tilt
- Lumbar lordosis angle
- Sacral slope
- Pelvic radius
- Grade of spondylolisthesis

Importing the data

In this part we first import the data and the modules needed to analyze the various features of it.

```
import warnings, requests, zipfile, io
warnings.simplefilter('ignore')
import pandas as pd
from scipy.io import arff
```

[2] ✓ 1.2s Python

```
f_zip = 'https://archive.ics.uci.edu/static/public/212/vertebral+column.zip'
r = requests.get(f_zip, stream=True)
Vertebral_zip = zipfile.ZipFile(io.BytesIO(r.content))
Vertebral_zip.extractall()
```

[8] ✓ 0.6s Python

```
data = arff.loadarff('column_2C_weka.arff')
df = pd.DataFrame(data[0])
```

[9] ✓ 0.0s Python

Fig1. Importing the data

Exploring the data

We explore the data in this section. We first load the data in arff and then see the shape and print out the column indentifiers.

```
[9]
1 data = arff.loadarff('column_2C_weka.arff')
2 df = pd.DataFrame(data[0])
✓ 0.0s Python
```

```
[11]
1 df.shape
✓ 0.0s Python
```

```
(310, 7)
```

```
[12]
1 df.columns
✓ 0.0s Python
```

```
Index(['pelvic_incidence', 'pelvic_tilt', 'lumbar_lordosis_angle',
       'sacral_slope', 'pelvic_radius', 'degree_spondylolisthesis', 'class'],
      dtype='object')
```

Fig 2. Exploring the data

```
[13]
1 df.dtypes
✓ 0.0s Python
```

```
pelvic_incidence            float64
pelvic_tilt                 float64
lumbar_lordosis_angle       float64
sacral_slope                float64
pelvic_radius               float64
degree_spondylolisthesis    float64
class                        object
dtype: object
```

+ Code + Markdown

Fig 3. Datatypes

We see the data types of the associated column field.

```
[14]  1  df
      ✓ 0.0s                                                    Python
```

	pelvic_incidence	pelvic_tilt	lumbar_lordosis_angle	sacral_slope	pelvic_radius	degree_s
0	63.027817	22.552586	39.609117	40.475232	98.672917	
1	39.056951	10.060991	25.015378	28.995960	114.405425	
2	68.832021	22.218482	50.092194	46.613539	105.985135	
3	69.297008	24.652878	44.311238	44.644130	101.868495	
4	49.712859	9.652075	28.317406	40.060784	108.168725	
...	...	...	...	...	...	
305	47.903565	13.616688	36.000000	34.286877	117.449062	
306	53.936748	20.721496	29.220534	33.215251	114.365845	
307	61.446597	22.694968	46.170347	38.751628	125.670725	
308	45.252792	8.693157	41.583126	36.559635	118.545842	
309	33.841641	5.073991	36.641233	28.767649	123.945244	

310 rows × 7 columns

Fig. 4 The dataset

We print the dataset to screen.

We can use the describe function to get a quick statistic on the full data or the subset of it.

```
[15]  1  df['pelvic_incidence'].describe()
      ✓ 0.0s                                                    Python
```

```
count    310.000000
mean      60.496653
std       17.236520
min       26.147921
25%       46.430294
50%       58.691038
75%       72.877696
max      129.834041
Name: pelvic_incidence, dtype: float64
```

Fig. 5 Describe function on a Column

```
1 df.describe()
```

[16] ✓ 0.0s Python

	pelvic_incidence	pelvic_tilt	lumbar_lordosis_angle	sacral_slope	pelvic_radius	degre
count	310.000000	310.000000	310.000000	310.000000	310.000000	
mean	60.496653	17.542822	51.930930	42.953831	117.920655	
std	17.236520	10.008330	18.554064	13.423102	13.317377	
min	26.147921	-6.554948	14.000000	13.366931	70.082575	
25%	46.430294	10.667069	37.000000	33.347122	110.709196	
50%	58.691038	16.357689	49.562398	42.404912	118.268178	
75%	72.877696	22.120395	63.000000	52.695888	125.467674	
max	129.834041	49.431864	125.742385	121.429566	163.071041	

Fig. 6 Describe function on full data

Since we don’t see any anomaly by eye hence we now rely on the metrics and plots.

```
1  import matplotlib.pyplot as plt
2  %matplotlib inline
3  df.plot()
4
```

[18] ✓ 0.5s Python

```
<Axes: >
```

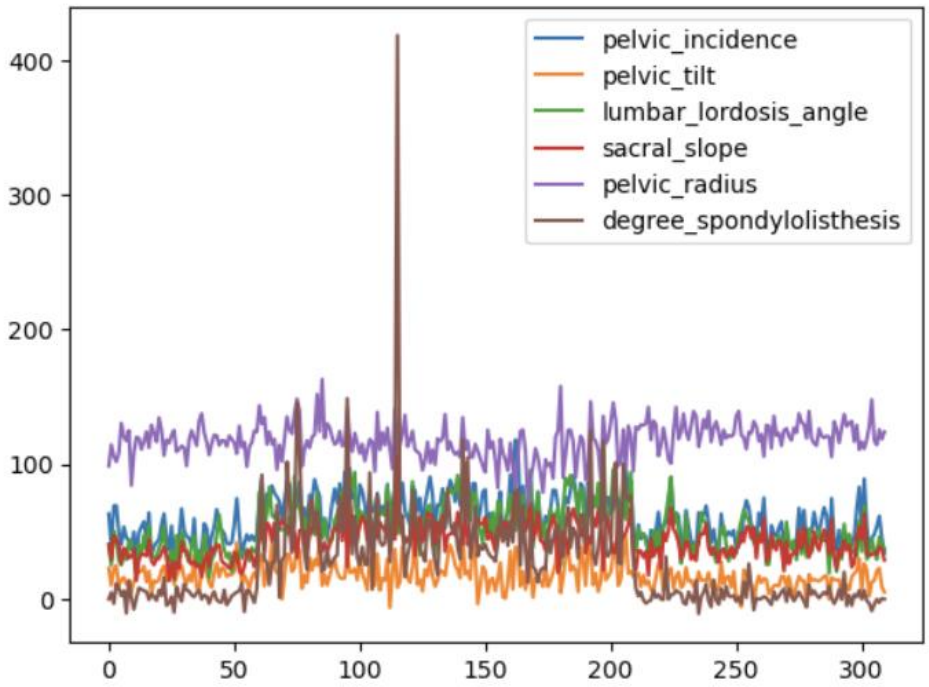


Fig. 7 Plot of data

We can also use the density plot to visualize the distribution of each column/field/feature.

```
1  df.plot(kind='density',subplots=True,layout=(4,2),figsize=(12,12),sharex=False)
2  plt.show()
```

[19] ✓ 1.3s Python

...

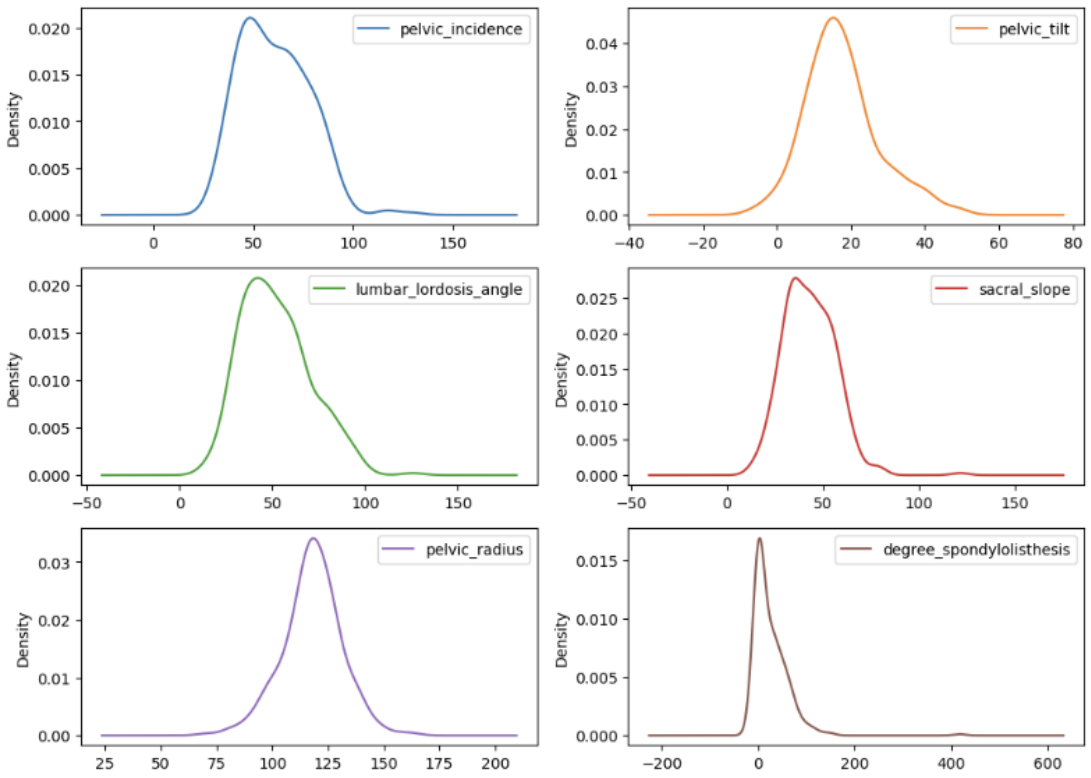


Fig. 8 Density plot of data

```
[20]  1  df['degree_spondylolisthesis'].plot.density()
      ✓ 0.2s                                                    Python
```

```
<Axes: ylabel='Density'>
```

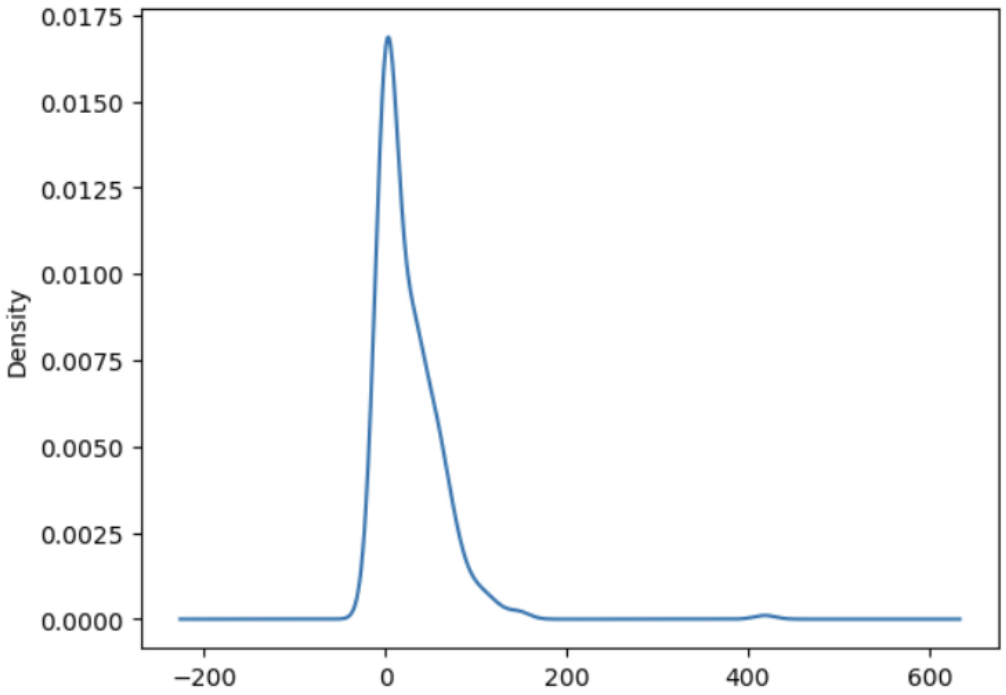


Fig. 9 Density plot of the Spondylolisthesis feature

We also plot histogram to see the skewness of the distribution

```
1  df['degree_spondylolisthesis'].plot.hist()
```

0.1s Python

```
<Axes: ylabel='Frequency'>
```

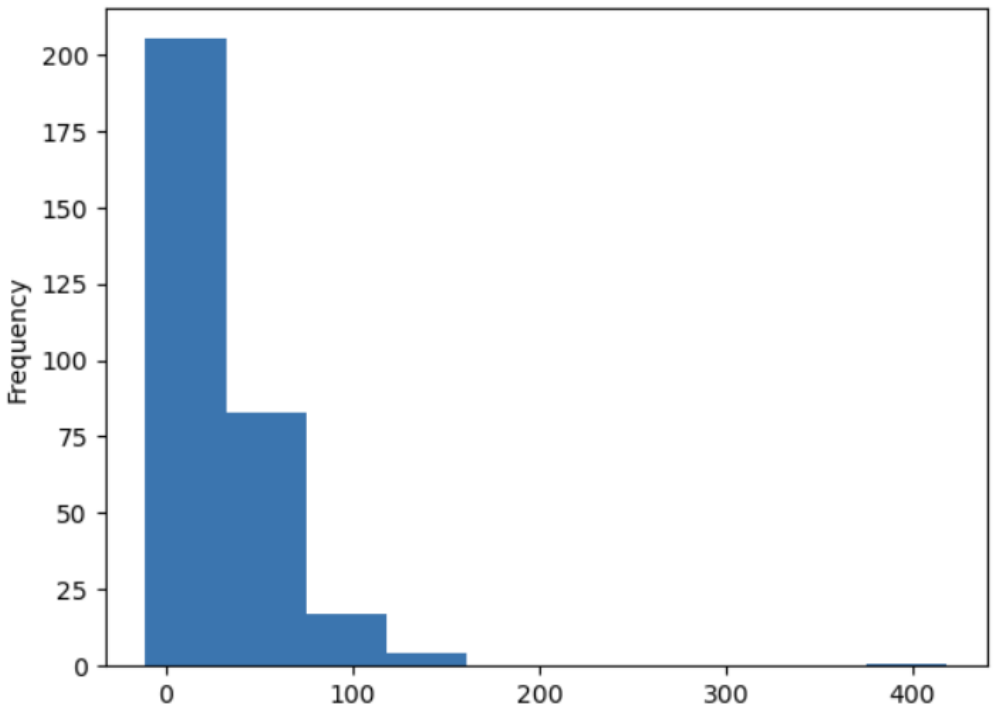


Fig. 10 Histogram plot of the feature

We then use the whisker box plot to see any outlier(s). From the box plot we can see that there seems to be quite a big value near 400.

```
[22]  1  df['degree_spondylolisthesis'].plot.box()
      ✓ 0.1s                                                    Python
```

```
<Axes: >
```

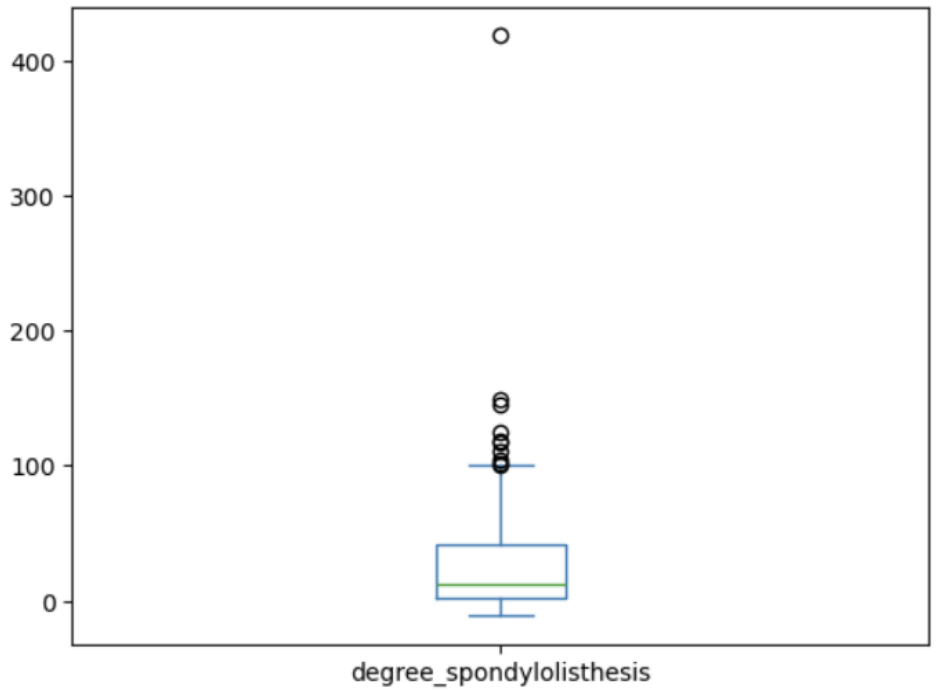


Fig. 11 Box plot of data

Analyzing the target

We check the number of abnormal and normal values. We cannot use strings for analysis, hence we convert abnormal to 1 and normal to 0.

```
[23]  df['class'].value_counts()
      ✓ 0.0s
```

```
class
b'Abnormal'    210
b'Normal'      100
Name: count, dtype: int64
```

```
[24]  class_mapper = {b'Abnormal':1,b'Normal':0}
      df['class']=df['class'].replace(class_mapper)

      ✓ 0.0s
```

```
[25]  df
      ✓ 0.0s
```

	pelvic_incidence	pelvic_tilt	lumbar_lordosis_angle	sacral_slope	pelvic_radius	degree_spondylolisthesis	class
0	63.027817	22.552586	39.609117	40.475232	98.672917	-0.254400	1
1	39.056951	10.060991	25.015378	28.995960	114.405425	4.564259	1
2	68.832021	22.218482	50.092194	46.613539	105.985135	-3.530317	1
3	69.297008	24.652878	44.311238	44.644130	101.868495	11.211523	1
4	49.712859	9.652075	28.317406	40.060784	108.168725	7.918501	1
...	...	...	...	...	...	...	...
305	47.903565	13.616688	36.000000	34.286877	117.449062	-4.245395	0
306	53.936748	20.721496	29.220534	33.215251	114.365845	-0.421010	0
307	61.446597	22.694968	46.170347	38.751628	125.670725	-2.707880	0
308	45.252792	8.693157	41.583126	36.559635	118.545842	0.214750	0
309	33.841641	5.073991	36.641233	28.767649	123.945244	-0.199249	0

310 rows × 7 columns

Fig. 12 Updated class values according to number

We then make a scatter plot of the data according to class map we did earlier

```
1  df.plot.scatter(y='degree_spondylolisthesis',x='class')
```

26] ✓ 0.3s

```
<Axes: xlabel='class', ylabel='degree_spondylolisthesis'>
```

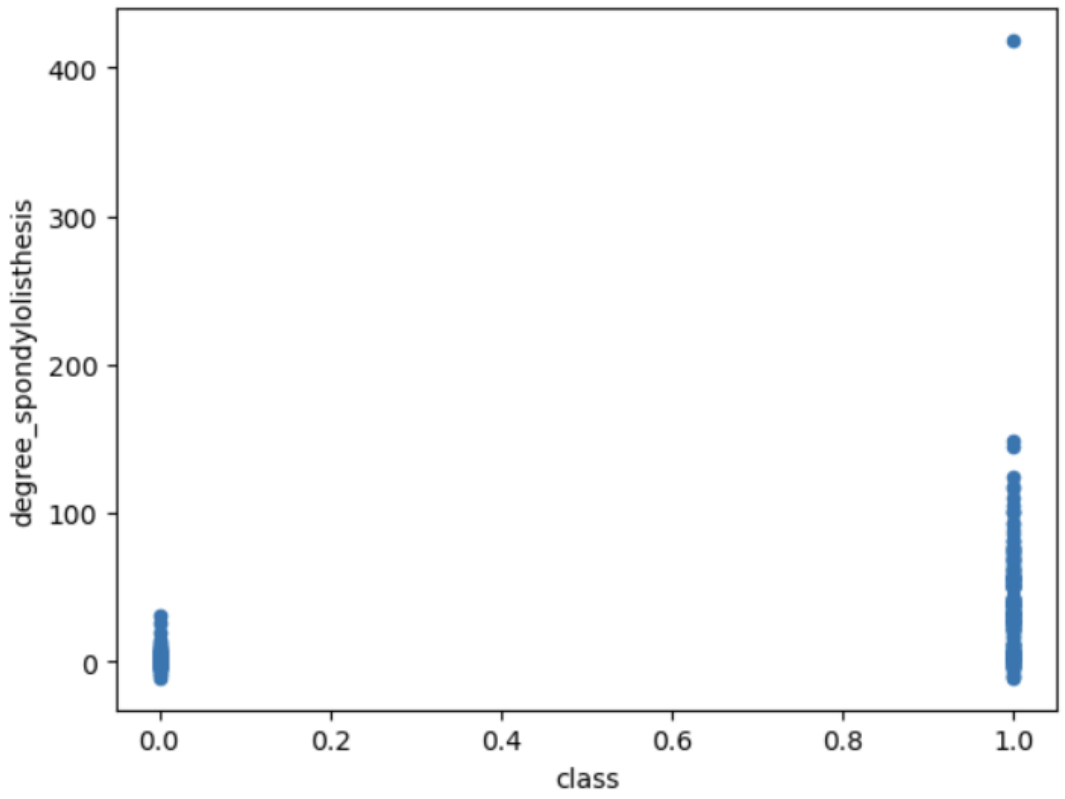


Fig. 13 Scatter Plot of abnormal and normal class

(Challenge Task)

For other classes we do the same by just changing the string

```
[38]  1 df.plot.scatter(y='pelvic_incidence',x='class')
      ✓ 0.1s
```

```
<Axes: xlabel='class', ylabel='pelvic_incidence'>
```

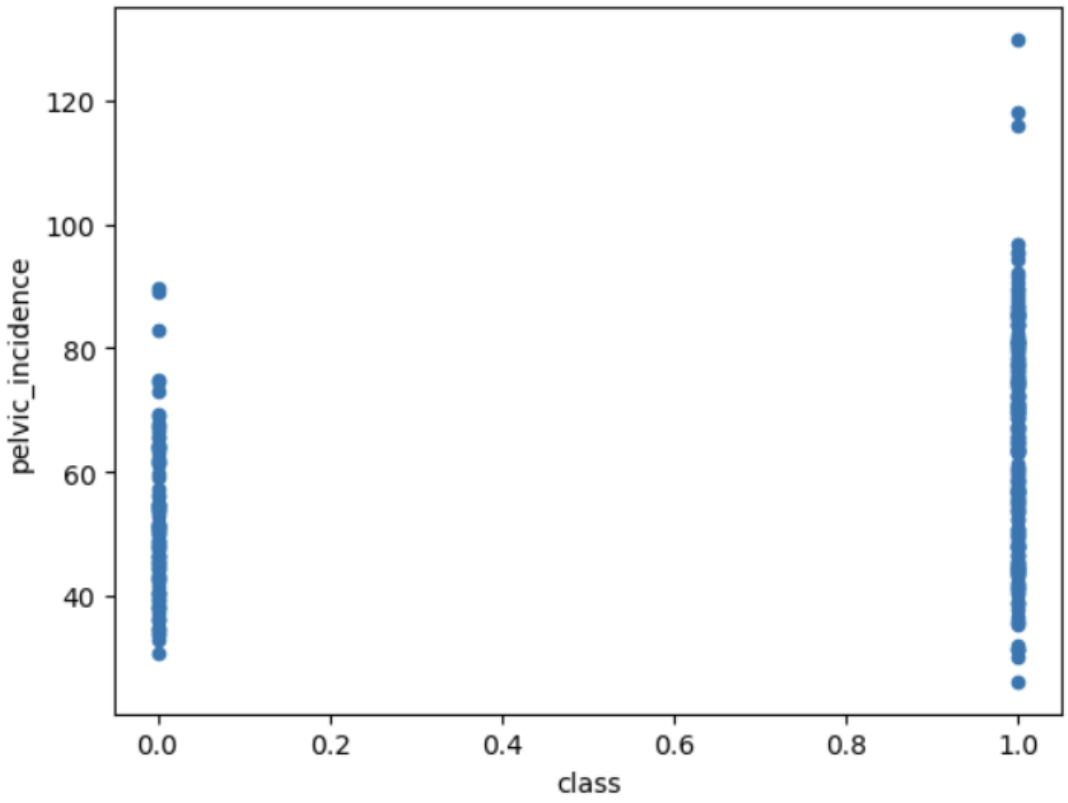


Fig. 14 Scatter plot for different class

To analyze multiple features at once we use the boxplot

```
1  df.groupby('class').boxplot(fontsize=20,rot=90,figsize=(20,10),patch_artist=True
```

[39] ✓ 1.8s Python

```
0         Axes(0.1,0.15;0.363636x0.75)
1    Axes(0.536364,0.15;0.363636x0.75)
dtype: object
```

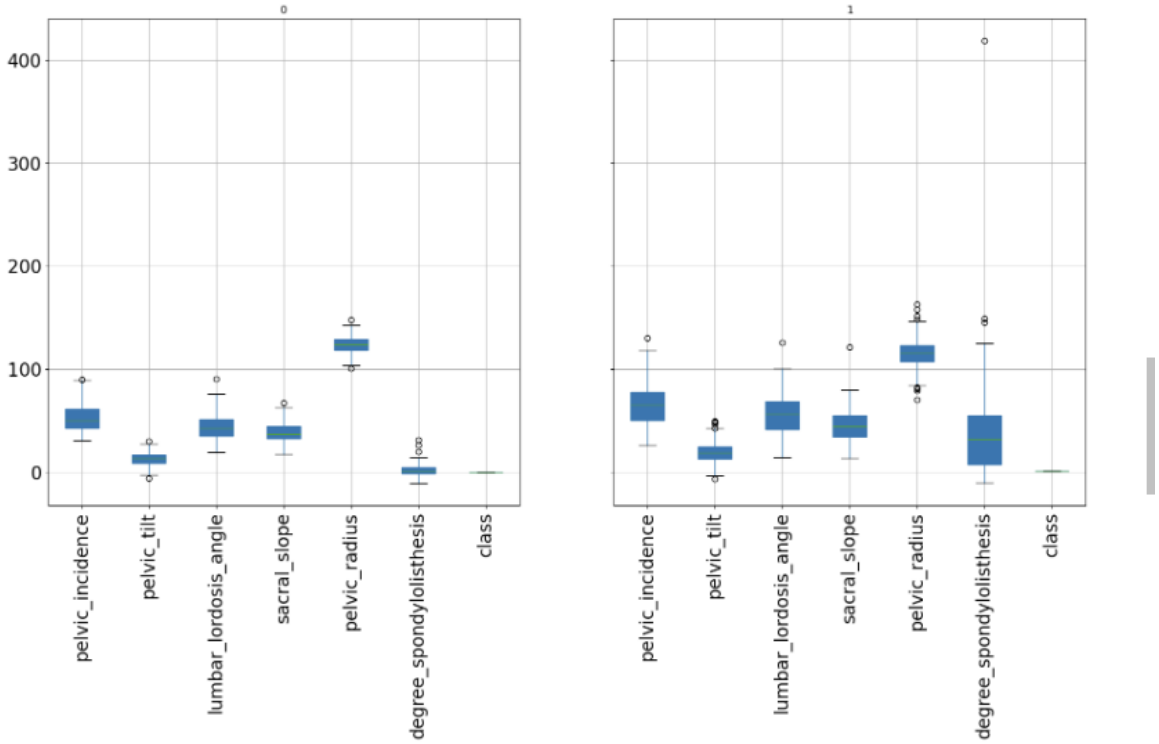


Fig. 15 Boxplot for all variable

.corr function gives us the correlation value beetween each features

```
1  corr_matrix = df.corr()
2  corr_matrix["class"].sort_values(ascending=False)
3
```

[29] ✓ 0.0s Python

```
class                      1.000000
degree_spondylolisthesis   0.443687
pelvic_incidence           0.353336
pelvic_tilt                0.326063
lumbar_lordosis_angle      0.312484
sacral_slope               0.210602
pelvic_radius             -0.309857
Name: class, dtype: float64
```

Fig. 16 Correlation matrix and values

We make a scatter plot to visualize the correlation matrix and then followup with heatmap of correlation matrix.

```
1  pd.plotting.scatter_matrix(df,figsize=(12,12))
2  plt.show()
```

[31] ✓ 2.4s Python

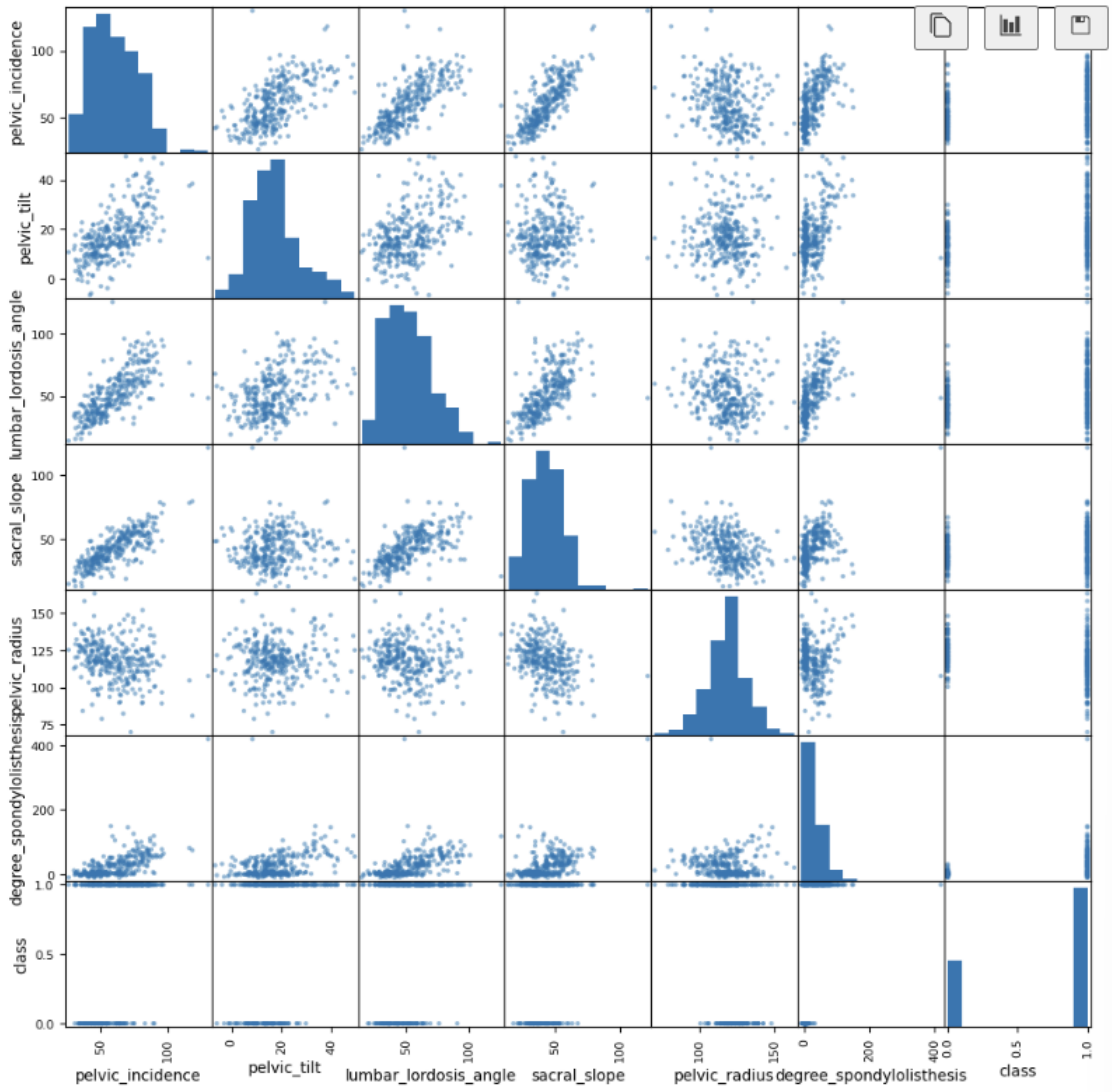


Fig. 17 Scatterplot for correlation matrix

My laptop did not have seaborn installed so I install seaborn in my conda environment with the conda install command.

Fig. 18 Seaborn installation

```python
import seaborn as sns
## Plot figsize
fig, ax = plt.subplots(figsize=(10, 10))
# Generate Color Map
# colormap = sns.diverging_palette(220, 10, as_cmap=True)
colormap = sns.color_palette("BrBG", 10)
# Generate Heat Map, allow annotations and place floats in map
sns.heatmap(corr_matrix, cmap=colormap, annot=True, fmt=".2f") #ax.set_yticklabe
plt.show()

```

[36] ✓ 0.3s Python

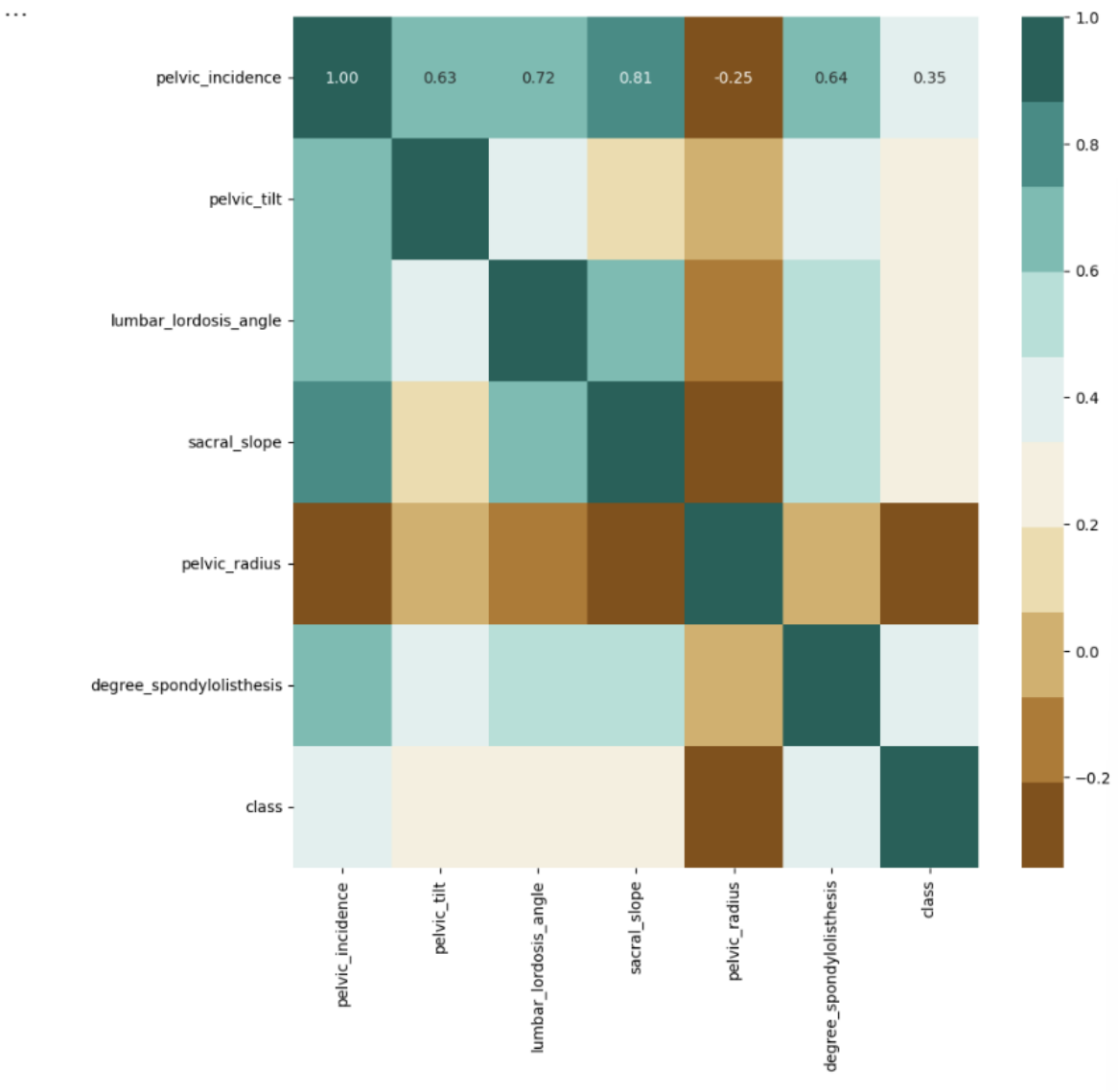


Fig. 19 Heatmap of correlation matrix

(Challenge Task)

Working on a different dataset.

I chose the Rice dataset [Rice (Cammeo and Osmancik)] from
Rice (Cammeo and Osmancik) - UCI Machine Learning Repository

```
[21]
import warnings, requests, zipfile, io
warnings.simplefilter('ignore')
import pandas as pd
from scipy.io import arff
```
✓ 0.0s Python

```
[22]
f_zip = 'https://archive.ics.uci.edu/static/public/545/rice+cammeo+and+osmancik.zip'
r = requests.get(f_zip, stream=True)
Vertebral_zip = zipfile.ZipFile(io.BytesIO(r.content))
Vertebral_zip.extractall()
```
✓ 0.8s Python

```
[31]
data = arff.loadarff('Rice_Cammeo_Osmancik.arff')
df = pd.DataFrame(data[0])
```
✓ 0.0s Python

```
[33]
df.columns
```
✓ 0.0s Python

```
Index(['Area', 'Perimeter', 'Major_Axis_Length', 'Minor_Axis_Length',
       'Eccentricity', 'Convex_Area', 'Extent', 'Class'],
      dtype='object')
```

```
[34]
df.dtypes
```
✓ 0.0s Python

```
Area                float64
Perimeter           float64
Major_Axis_Length   float64
Minor_Axis_Length   float64
Eccentricity        float64
Convex_Area         float64
Extent              float64
Class                object
dtype: object
```

Fig. 20 Importing New Dataset

```
1 df
```

[35] ✓ 0.0s Python

	Area	Perimeter	Major_Axis_Length	Minor_Axis_Length	Eccentricity	Convex_Area	Extent	Class
0	15231.0	525.578979	229.749878	85.093788	0.928882	15617.0	0.572896	b'Cammeo'
1	14656.0	494.311005	206.020065	91.730972	0.895405	15072.0	0.615436	b'Cammeo'
2	14634.0	501.122009	214.106781	87.768288	0.912118	14954.0	0.693259	b'Cammeo'
3	13176.0	458.342987	193.337387	87.448395	0.891861	13368.0	0.640669	b'Cammeo'
4	14688.0	507.166992	211.743378	89.312454	0.906691	15262.0	0.646024	b'Cammeo'
...	...	...	...	...	...	...	...	...
3805	11441.0	415.858002	170.486771	85.756592	0.864280	11628.0	0.681012	b'Osmancik'
3806	11625.0	421.390015	167.714798	89.462570	0.845850	11904.0	0.694279	b'Osmancik'
3807	12437.0	442.498993	183.572922	86.801979	0.881144	12645.0	0.626739	b'Osmancik'
3808	9882.0	392.296997	161.193985	78.210480	0.874406	10097.0	0.659064	b'Osmancik'
3809	11434.0	404.709991	161.079269	90.868195	0.825692	11591.0	0.802949	b'Osmancik'

3810 rows × 8 columns

```
1 df['Extent'].describe()
```

[36] ✓ 0.0s Python

```
count    3810.000000
mean        0.661934
std         0.077239
min         0.497413
25%         0.598862
50%         0.645361
75%         0.726562
max         0.861050
Name: Extent, dtype: float64
```

Fig. 21 Printing the dataset on screen

+ Code + Markdown

```
1 df.describe()
```

38] ✓ 0.0s Python

	Area	Perimeter	Major_Axis_Length	Minor_Axis_Length	Eccentricity	Convex_Area	Extent
count	3810.000000	3810.000000	3810.000000	3810.000000	3810.000000	3810.000000	3810.000000
mean	12667.727559	454.239180	188.776222	86.313750	0.886871	12952.496850	0.661934
std	1732.367706	35.597081	17.448679	5.729817	0.020818	1776.972042	0.077239
min	7551.000000	359.100006	145.264465	59.532406	0.777233	7723.000000	0.497413
25%	11370.500000	426.144753	174.353855	82.731695	0.872402	11626.250000	0.598862
50%	12421.500000	448.852493	185.810059	86.434647	0.889050	12706.500000	0.645361
75%	13950.000000	483.683746	203.550438	90.143677	0.902588	14284.000000	0.726562
max	18913.000000	548.445984	239.010498	107.542450	0.948007	19099.000000	0.861050

Fig. 22 Find a simple statistic of the data

We now visualize the dataset by plotting the density plot

```
1  df.plot(kind='density',subplots=True,layout=(3,3),figsize=(12,12),sharex=False)
2  plt.show()
```

[43] ✓ 1.3s Python

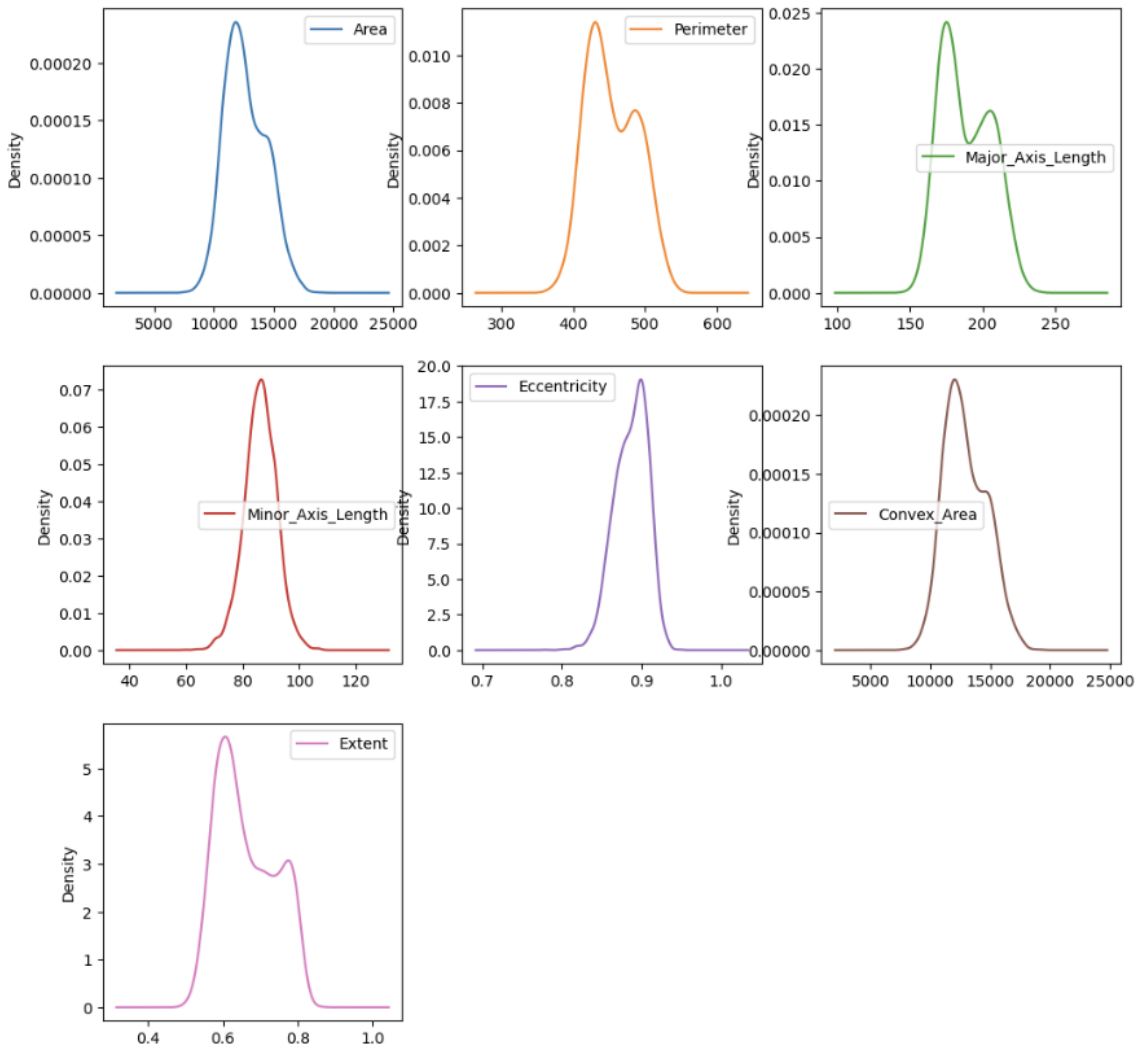
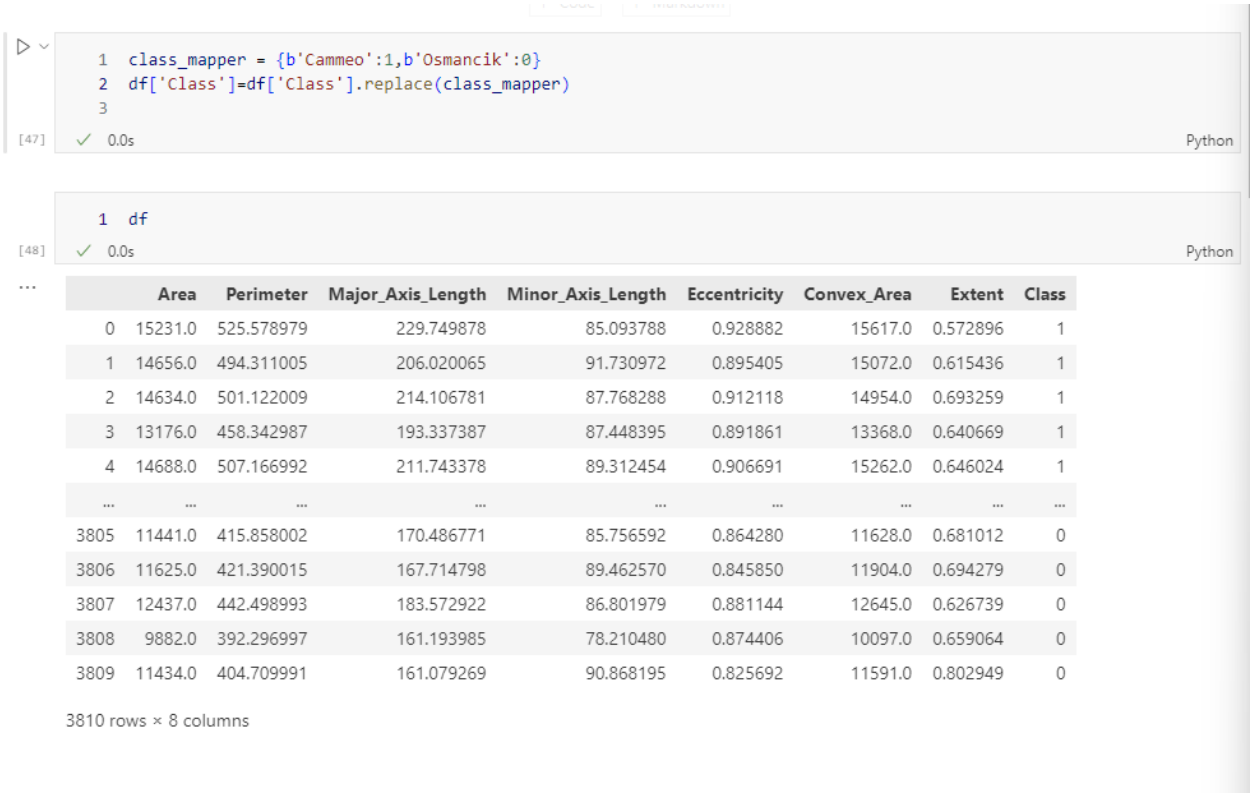


Fig. 23 Density plot of the data

I chose class by using Cammeo as 1 Osmancik as 0



```
1 class_mapper = {b'Cammeo':1,b'Osmancik':0}
2 df['Class']=df['Class'].replace(class_mapper)
3
```

[47] ✓ 0.0s Python

```
1 df
```

[48] ✓ 0.0s Python

	Area	Perimeter	Major_Axis_Length	Minor_Axis_Length	Eccentricity	Convex_Area	Extent	Class
0	15231.0	525.578979	229.749878	85.093788	0.928882	15617.0	0.572896	1
1	14656.0	494.311005	206.020065	91.730972	0.895405	15072.0	0.615436	1
2	14634.0	501.122009	214.106781	87.768288	0.912118	14954.0	0.693259	1
3	13176.0	458.342987	193.337387	87.448395	0.891861	13368.0	0.640669	1
4	14688.0	507.166992	211.743378	89.312454	0.906691	15262.0	0.646024	1
...	...	...	...	...	...	...	...	...
3805	11441.0	415.858002	170.486771	85.756592	0.864280	11628.0	0.681012	0
3806	11625.0	421.390015	167.714798	89.462570	0.845850	11904.0	0.694279	0
3807	12437.0	442.498993	183.572922	86.801979	0.881144	12645.0	0.626739	0
3808	9882.0	392.296997	161.193985	78.210480	0.874406	10097.0	0.659064	0
3809	11434.0	404.709991	161.079269	90.868195	0.825692	11591.0	0.802949	0

3810 rows × 8 columns

Fig. 24 Replacing with target class

```
1  df.groupby('Class').boxplot(fontsize=20,rot=90,figsize=(20,10),patch_artist=True)
```

[51] ✓ 0.4s Python

```
0         Axes(0.1,0.15;0.363636x0.75)
1    Axes(0.536364,0.15;0.363636x0.75)
dtype: object
```

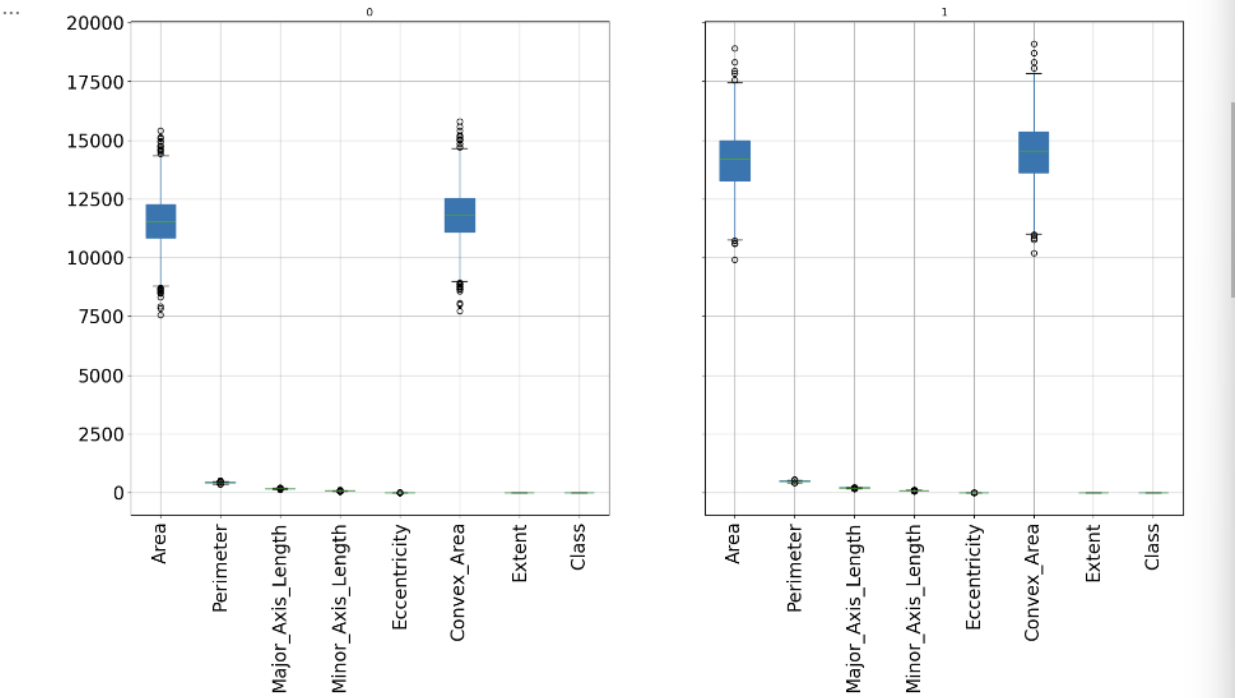


Fig. 25 Boxplots sorted by class

```
1  pd.plotting.scatter_matrix(df,figsize=(12,12))
2  plt.show()
```

53] ✓ 3.0s Pyt

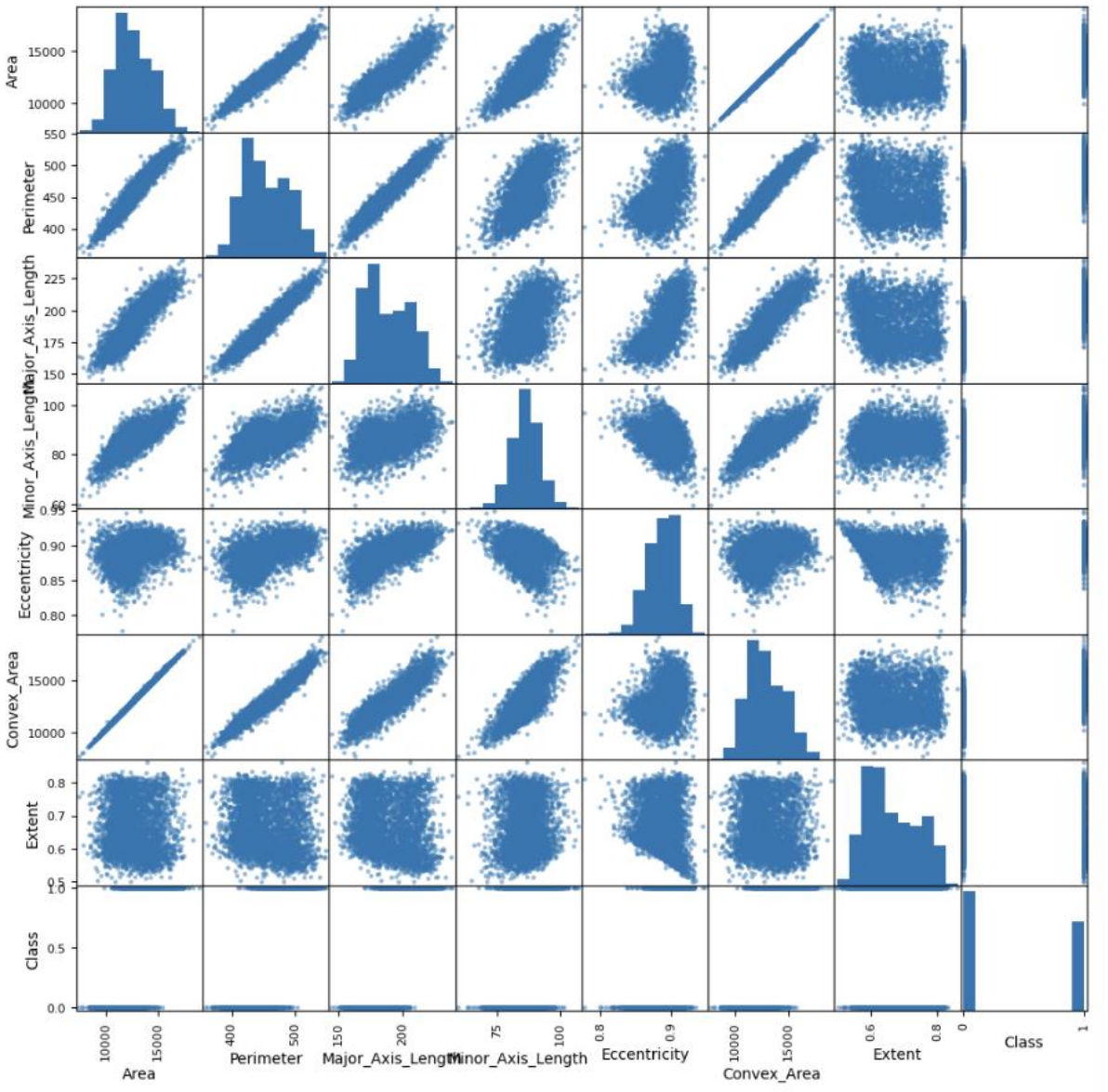


Fig. 26 Scatter Matrix of the new data set

```python
import seaborn as sns
## Plot figsize
fig, ax = plt.subplots(figsize=(10, 10))
# Generate Color Map
# colormap = sns.diverging_palette(220, 10, as_cmap=True)
colormap = sns.color_palette("BrBG", 10)
# Generate Heat Map, allow annotations and place floats in map
sns.heatmap(corr_matrix, cmap=colormap, annot=True, fmt=".2f") #ax.set_yticklabels(column_names);
plt.show()

```

56] ✓ 0.2s Python

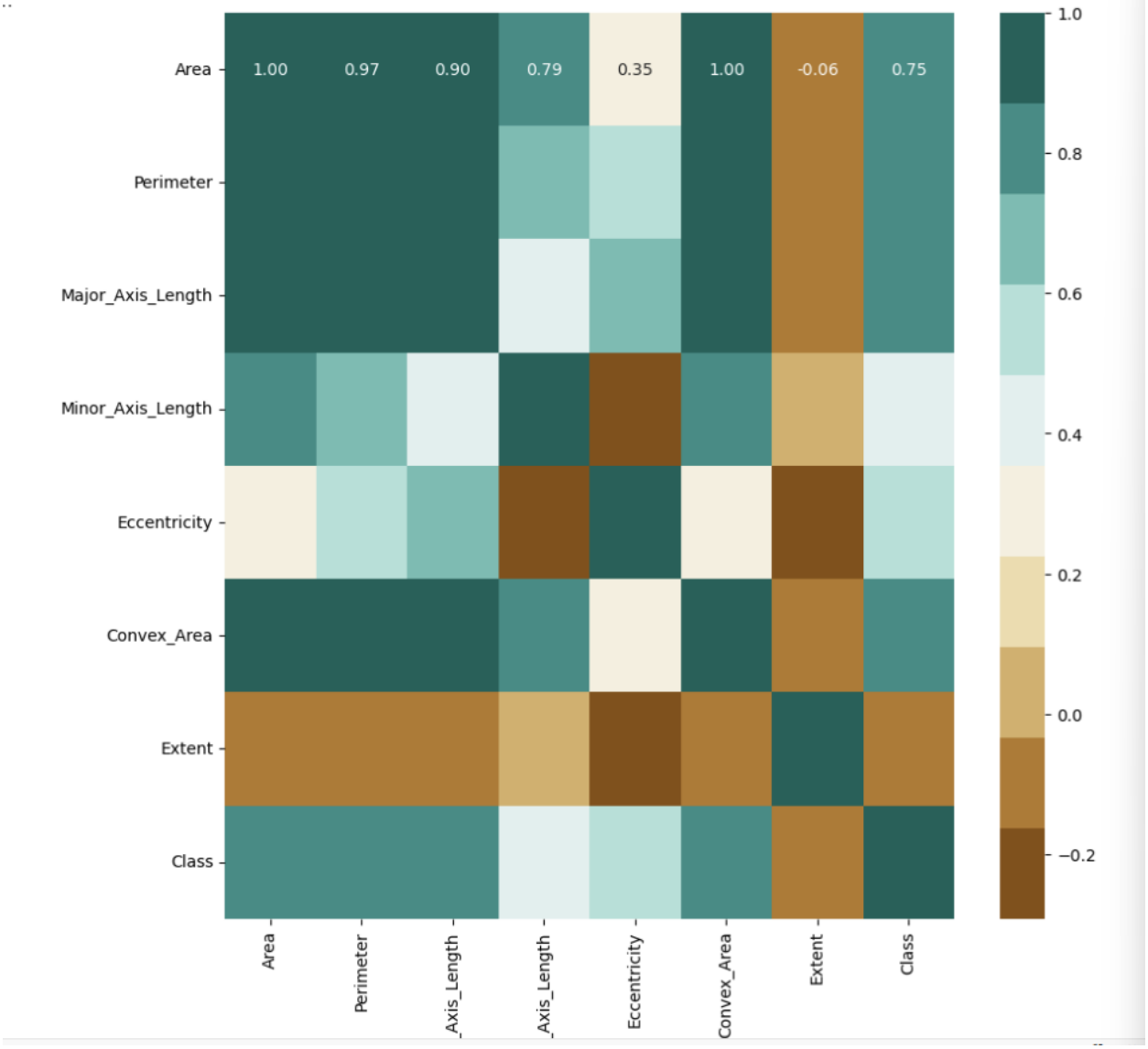


Fig. 27 Heat Map of the new correlation matrix

Conclusion

From this homework, we learn a lot of new techniques in data analysis and visualization. To start with importing and exploring the dataset, we first learnt how to download data in python using the pandas module. We learnt how to visualize the several features by using various statistical tools and techniques. Going forward in this machine learning class, these tools are going to be very helpful in determining the model chosen for the dataset. I look forward to next lectures and assignments.