# Lab\_Manual\_2\_Exploring the Data

## Overview

This lab is a continuation of the guided labs in lab1. Tasked with improving orthopedic abnormality detection, you'll use machine learning on a dataset containing biomechanical features and patient labels. The dataset involves two classification tasks

## Introducing the business scenario

You work for a healthcare provider, and want to improve the detection of abnormalities in orthopedic patients.

You are tasked with solving this problem by using machine learning (ML). You have access to a dataset that contains six biomechanical features and a target of *normal* or *abnormal*. You can use this dataset to train an ML model to predict if a patient will have an abnormality.

## About this dataset

This biomedical dataset was built by Dr. Henrique da Mota during a medical residence period in the Group of Applied Research in Orthopaedics (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 first task consists in classifying patients as belonging to one of three categories:

*Normal* (100 patients)

*Disk Hernia* (60 patients)

*Spondylolisthesis* (150 patients)

For the second task, the categories *Disk Hernia* and *Spondylolisthesis* were merged into a single category that is labeled as *abnormal*. Thus, the second task consists in classifying patients as

belonging to one of two categories: *Normal* (100 patients) or *Abnormal* (210 patients).

## Attribute information

Each patient is represented in the dataset by six biomechanical attributes that are derived from the shape and orientation of the pelvis and lumbar spine (in this order):

Pelvic incidence Pelvic tilt

Lumbar lordosis angle

Sacral slope Pelvic radius

Grade of spondylolisthesis

The following convention is used for the class labels: DH (Disk Hernia)

Spondylolisthesis (SL) Normal (NO) Abnormal (AB)

For more information about this dataset, see the [Vertebral Column dataset webpage](http://archive.ics.uci.edu/ml/datasets/Vertebral%2BColumn).

## Dataset attributions

This dataset was obtained from: Dua, D. and Graff, C. (2019). UCI Machine Learning Repository (<http://archive.ics.uci.edu/ml>). Irvine, CA: University of California, School of Information and

Computer Science.

# Lab setup

Because this solution is split across several labs in this module, you must run the following cells so that you can load the data:

## Importing the data

In [2]:

**import** warnings**,** requests**,** zipfile**,** io warnings**.**simplefilter('ignore')

**import** pandas **as** pd

**from** scipy.io **import** arff

In [3]:

f\_zip **=** ['http://archive.ics.uci.edu/ml/machine-learning-databases/00212/vertebral\_col](http://archive.ics.uci.edu/ml/machine-learning-databases/00212/vertebral_col) r **=** requests**.**get(f\_zip, stream**=True**)

Vertebral\_zip **=** zipfile**.**ZipFile(io**.**BytesIO(r**.**content)) Vertebral\_zip**.**extractall()

In [4]:

data **=** arff**.**loadarff('column\_2C\_weka.arff') df **=** pd**.**DataFrame(data[0])

# Step 1: Exploring the data

You will start by looking at the data in the dataset.

To get the most out of this lab, carefully read the instructions and code before you run the cells. Take time to experiment!

First, you will use **shape** to examine the number of rows and columns

In [5]:

Out[5]:

In [7]:

Out[7]:

In [8]:

Out[8]:

In [25]:

(310, 7)

df**.**shape

You will now get a list of the columns.

df**.**columns

Index(['pelvic\_incidence', 'pelvic\_tilt', 'lumbar\_lordosis\_angle',

'sacral\_slope', 'pelvic\_radius', 'degree\_spondylolisthesis', 'class'], dtype='object')

You can see the six biomechanical features, and the target column is named *class*. What column types do you have?

df**.**dtypes

pelvic\_incidence float64

pelvic\_tilt float64

lumbar\_lordosis\_angle float64

sacral\_slope float64

pelvic\_radius float64 degree\_spondylolisthesis float64 class object

dtype: object

df

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Out[25]: | **pelvic\_incidence** | **pelvic\_tilt** | **lumbar\_lordosis\_angle** | **sacral\_slope** | **pelvic\_radius** | **degree\_spondylolis** |
|  | **0** 63.027817 | 22.552586 | 39.609117 | 40.475232 | 98.672917 | -0.2 |
|  | **1** 39.056951 | 10.060991 | 25.015378 | 28.995960 | 114.405425 | 4.5 |
|  | **2** 68.832021 | 22.218482 | 50.092194 | 46.613539 | 105.985135 | -3.5 |
|  | **3** 69.297008 | 24.652878 | 44.311238 | 44.644130 | 101.868495 | 11.2 |
|  | **4** 49.712859 | 9.652075 | 28.317406 | 40.060784 | 108.168725 | 7.9 |
|  | **...** ... | ... | ... | ... | ... |  |
|  | **305** 47.903565 | 13.616688 | 36.000000 | 34.286877 | 117.449062 | -4.2 |
|  | **306** 53.936748 | 20.721496 | 29.220534 | 33.215251 | 114.365845 | -0.4 |
|  | **307** 61.446597 | 22.694968 | 46.170347 | 38.751628 | 125.670725 | -2.7 |
|  | **308** 45.252792 | 8.693157 | 41.583126 | 36.559635 | 118.545842 | 0.2 |
|  | **309** 33.841641 | 5.073991 | 36.641233 | 28.767649 | 123.945244 | -0.1 |

310 rows × 7 columns

You have six floats for the biomechanical features, but the target is a class.

To look at the statistics for the first column, you can use the **describe** function.

In [9]:

Out[9]:

In [8]:

df**.**describe()

count 310.000000

df['pelvic\_incidence']**.**describe()

|  |  |
| --- | --- |
| 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 |

**Challenge Task:** Try updating the code in the previous cell to view the statistics of other features. Which features have outliers that you might want to examine?

Because this dataset only has six features, you can display the statistics of each feature by running **describe** on the entire DataFrame.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Out[8]: | **pelvic\_incidence** | **pelvic\_tilt** | **lumbar\_lordosis\_angle** | **sacral\_slope** | **pelvic\_radius** | **degree\_spondyl** |
|  | **count** 310.000000 | 310.000000 | 310.000000 | 310.000000 | 310.000000 | 31 |
|  | **mean** 60.496653 | 17.542822 | 51.930930 | 42.953831 | 117.920655 | 2 |
|  | **std** 17.236520 | 10.008330 | 18.554064 | 13.423102 | 13.317377 | 3 |
|  | **min** 26.147921 | -6.554948 | 14.000000 | 13.366931 | 70.082575 | -1 |
|  | **25%** 46.430294 | 10.667069 | 37.000000 | 33.347122 | 110.709196 |  |
|  | **50%** 58.691038 | 16.357689 | 49.562398 | 42.404912 | 118.268178 | 1 |
|  | **75%** 72.877696 | 22.120395 | 63.000000 | 52.695888 | 125.467674 | 4 |
|  | **max** 129.834041 | 49.431864 | 125.742385 | 121.429566 | 163.071041 | 41 |

50% - is median (of sorted values). 25% - is median of first half of the sorted values. 75% - is median of second half of the sorted values.

**Question:** Are there any features that aren't well-distributed? Are there any features with outliers that you want to look at? Does it look like there might be any correlations between features?

It's not always easy to make observations when you look only at numbers, so you will now plot these values.

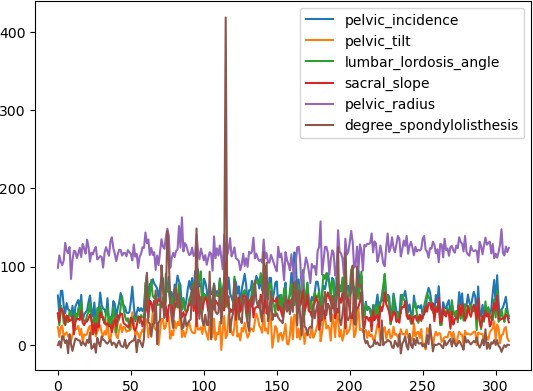
In [17]:

**import** matplotlib.pyplot **as** plt

**%matplotlib** inline df**.**plot()

Out[17]:

<AxesSubplot:>

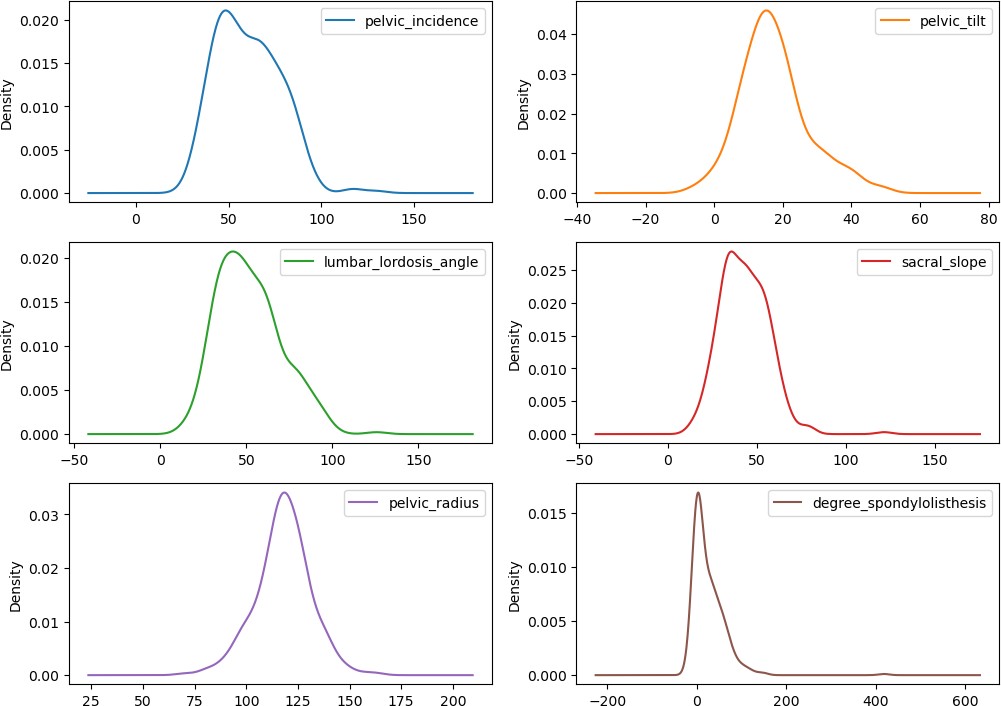


# You should have more detail knowledge on Plotting ## One of the source https://pandas.pydata.org/pandas- docs/version/0.16.2/visualization.html

You will now plot the distribution of the values for each feature by using a *density or kernel density estimate (KDE) plot*.

In [18]:

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



Do any of the visualizations stand out?

### Investigating degree\_spondylolisthesis

You will now investigate **degree\_spondylolisthesis**:

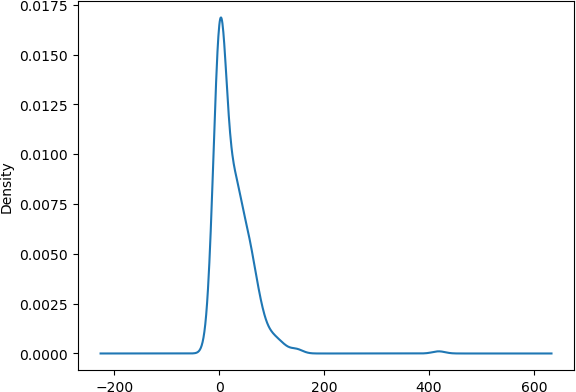
Start with the *density plot*, which if you recall, shows the *distribution of the values*.

In [19]:

Out[19]:

<AxesSubplot:ylabel='Density'>

df['degree\_spondylolisthesis']**.**plot**.**density()



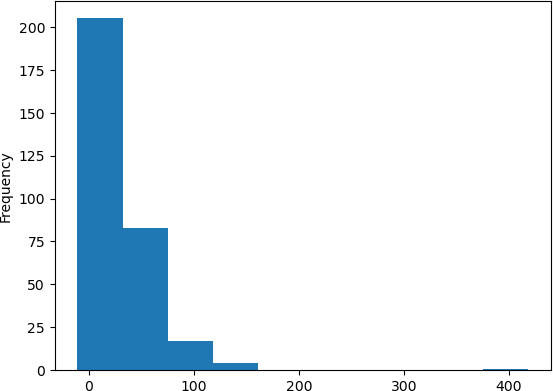
A density plot smooths out the curve. It looks like there might be an increase around **400**. Visualize the data with a *histogram*.

In [20]:

Out[20]:

<AxesSubplot:ylabel='Frequency'>

df['degree\_spondylolisthesis']**.**plot**.**hist()



By using a *box plot*, you can see if there any outliers.

In [21]:

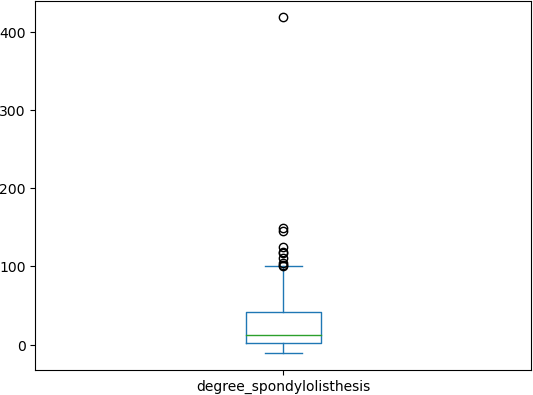
Out[21]:

In [22]:

Out[22]:

<AxesSubplot:>

df['degree\_spondylolisthesis']**.**plot**.**box()



You can see a small increase around **400**. Sometimes, outliers like this can throw off training

models. The only way to find out would be to test the model both with and without the outliers, and compare the models' scores. However, this is a task for a later lab.

You can see from the box plot that there seems to be a cluster above what *looks like* the maximum. Is there a correlation between those data points and the target?

Before you can look for a correlation, you will examine the target more.

### Analyzing the target

First, what kind of distribution do you have?

df['class']**.**value\_counts()

b'Abnormal' 210

b'Normal' 100

Name: class, dtype: int64

It loks like you have about 1/3 *Normal* and 2/3 *Abnormal*. This result should be fine, but if you could get more data, you would want to try and balance the numbers more.

The class values aren't going to work for your ML model, so you will convert this column to a numeric value. You can use a *mapper* for this task.

In [23]:

class\_mapper **=** {b'Abnormal':1,b'Normal':0}

df['class']**=**df['class']**.**replace(class\_mapper)

Now, you can plot the *degree\_spondylolisthesis* against the target.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| In [24]: | df | | | | | | |
| Out[24]: |  | **pelvic\_incidence** | **pelvic\_tilt** | **lumbar\_lordosis\_angle** | **sacral\_slope** | **pelvic\_radius** | **degree\_spondylolis** |
|  | **0** | 63.027817 | 22.552586 | 39.609117 | 40.475232 | 98.672917 | -0.2 |
|  | **1** | 39.056951 | 10.060991 | 25.015378 | 28.995960 | 114.405425 | 4.5 |
|  | **2** | 68.832021 | 22.218482 | 50.092194 | 46.613539 | 105.985135 | -3.5 |
|  | **3** | 69.297008 | 24.652878 | 44.311238 | 44.644130 | 101.868495 | 11.2 |
|  | **4** | 49.712859 | 9.652075 | 28.317406 | 40.060784 | 108.168725 | 7.9 |
|  | **...** | ... | ... | ... | ... | ... |  |
|  | **305** | 47.903565 | 13.616688 | 36.000000 | 34.286877 | 117.449062 | -4.2 |
|  | **306** | 53.936748 | 20.721496 | 29.220534 | 33.215251 | 114.365845 | -0.4 |
|  | **307** | 61.446597 | 22.694968 | 46.170347 | 38.751628 | 125.670725 | -2.7 |
|  | **308** | 45.252792 | 8.693157 | 41.583126 | 36.559635 | 118.545842 | 0.2 |
|  | **309** | 33.841641 | 5.073991 | 36.641233 | 28.767649 | 123.945244 | -0.1 |

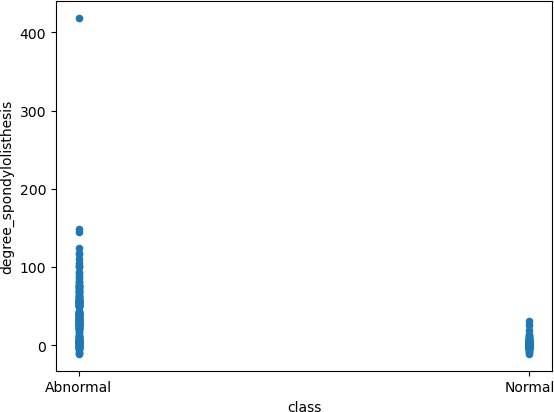
310 rows × 7 columns

In [11]:

Out[11]:

<AxesSubplot:xlabel='class', ylabel='degree\_spondylolisthesis'>

df**.**plot**.**scatter(y**=**'degree\_spondylolisthesis',x**=**'class')

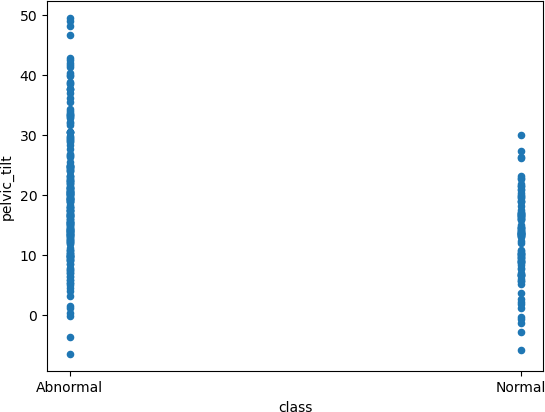


In [12]:

Out[12]:

<AxesSubplot:xlabel='class', ylabel='pelvic\_tilt'>

df**.**plot**.**scatter(y**=**'pelvic\_tilt',x**=**'class')



What do you see?

Though there appears to be a link between the high values and the abnormalities, there are also many values that are in the same range. So, there could be a correlation, but it's worth taking a closer look at the data.

**Challenge Task:** By using the previous cells, determine how the values of other features correspond against the target.

### Visualizing multiple variables

As the previous steps demonstrate, visualizations can be very powerful. Sometimes, you will want to analyze multiple data points. You can do this by using *groupby*.

Plotting out the features for both *Abnormal* and *Normal* values side by side might help you observe any other differences.

In [26]:

Out[26]:

In [27]:

corr\_matrix **=** df**.**corr()

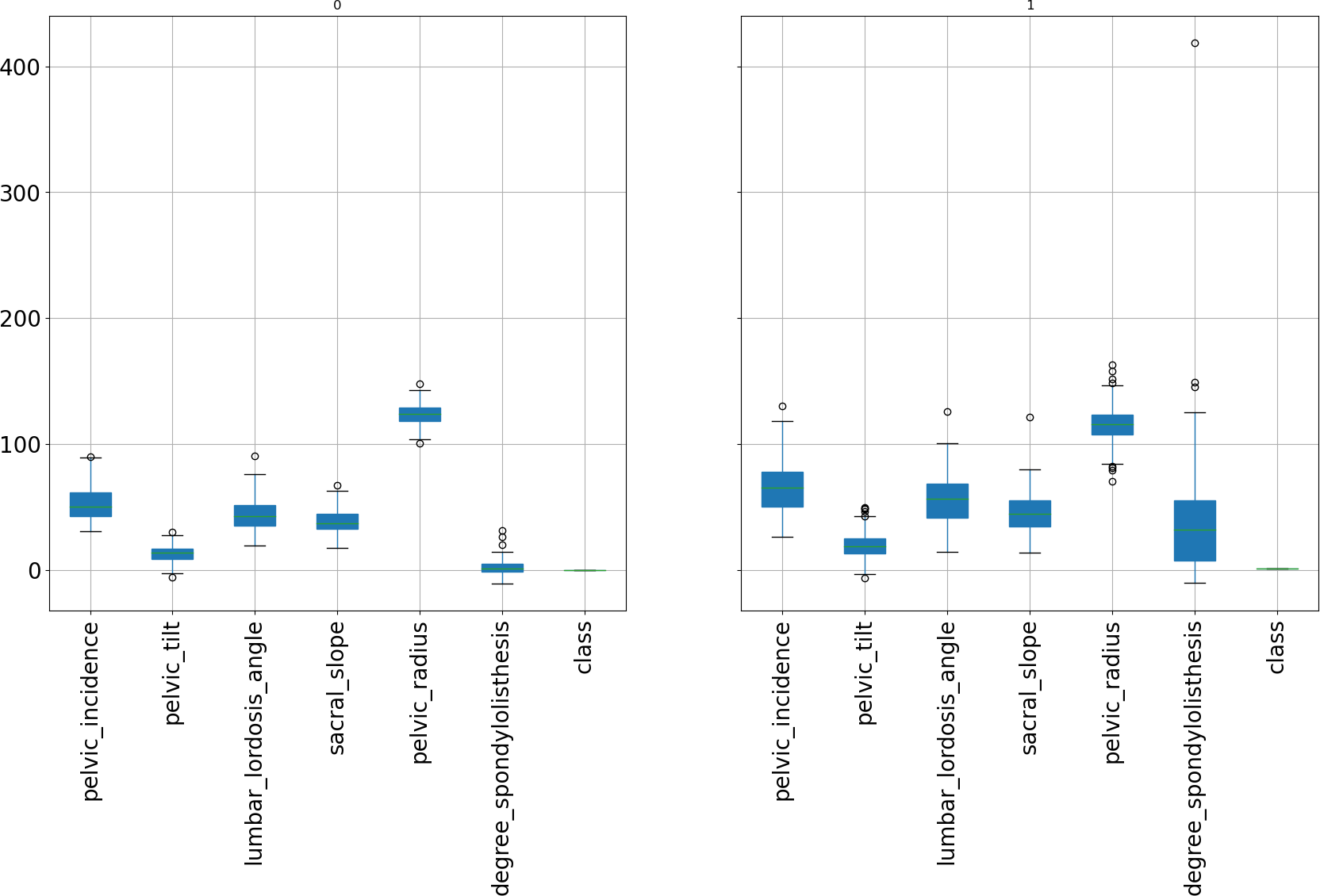
corr\_matrix["class"]**.**sort\_values(ascending**=False**)

0 AxesSubplot(0.1,0.15;0.363636x0.75)

df**.**groupby('class')**.**boxplot(fontsize**=**20,rot**=**90,figsize**=**(20,10),patch\_artist**=True**)

1 AxesSubplot(0.536364,0.15;0.363636x0.75)

dtype: object



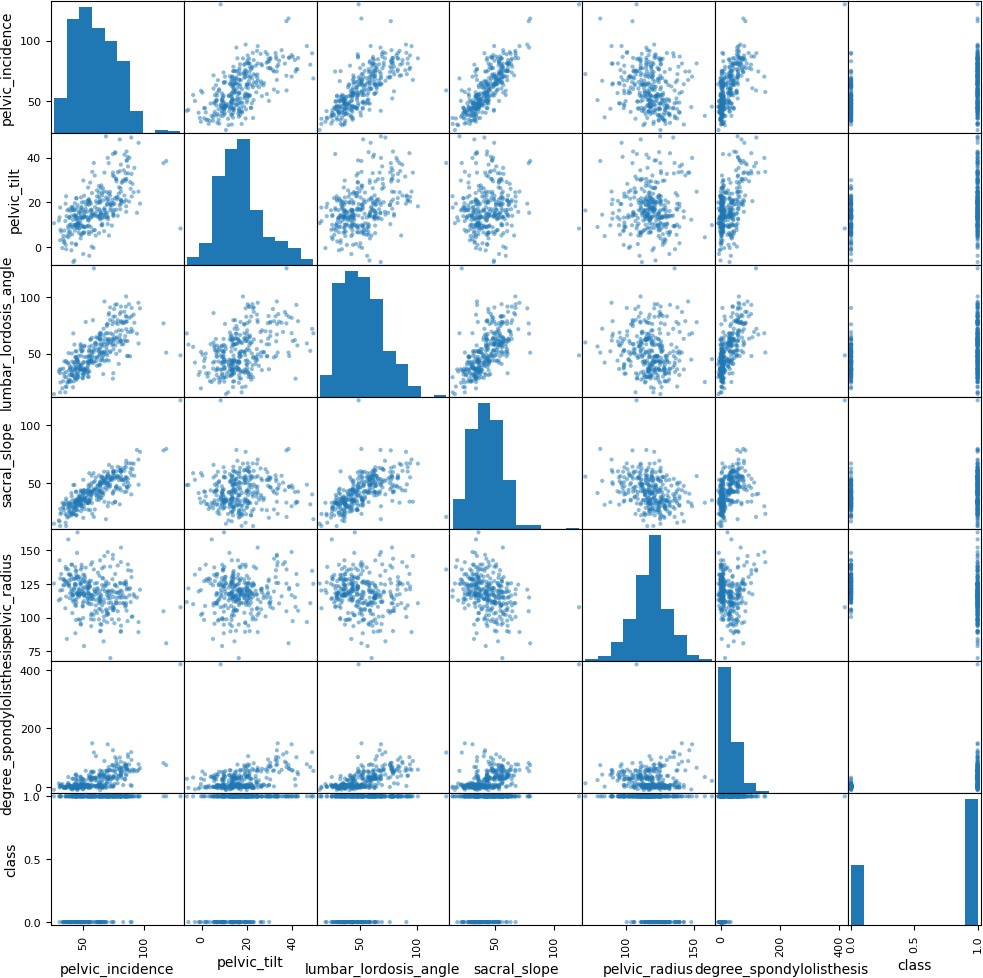
Using the **corr** function, you can create a correlation matrix for the entire dataset.

Out[27]:

|  |  |
| --- | --- |
| 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  Name: class, dtype: float64 | -0.309857 |
| You can also plot out this data. |  |

In [17]:

pd**.**plotting**.**scatter\_matrix(df,figsize**=**(12,12)) plt**.**show()



By using **seaborn**, you can visualize the correlation as a *heatmap*.

In [28]:

**import** seaborn **as** sns

*## Plot figsize*

fig, ax **=** plt**.**subplots(figsize**=**(10, 10))

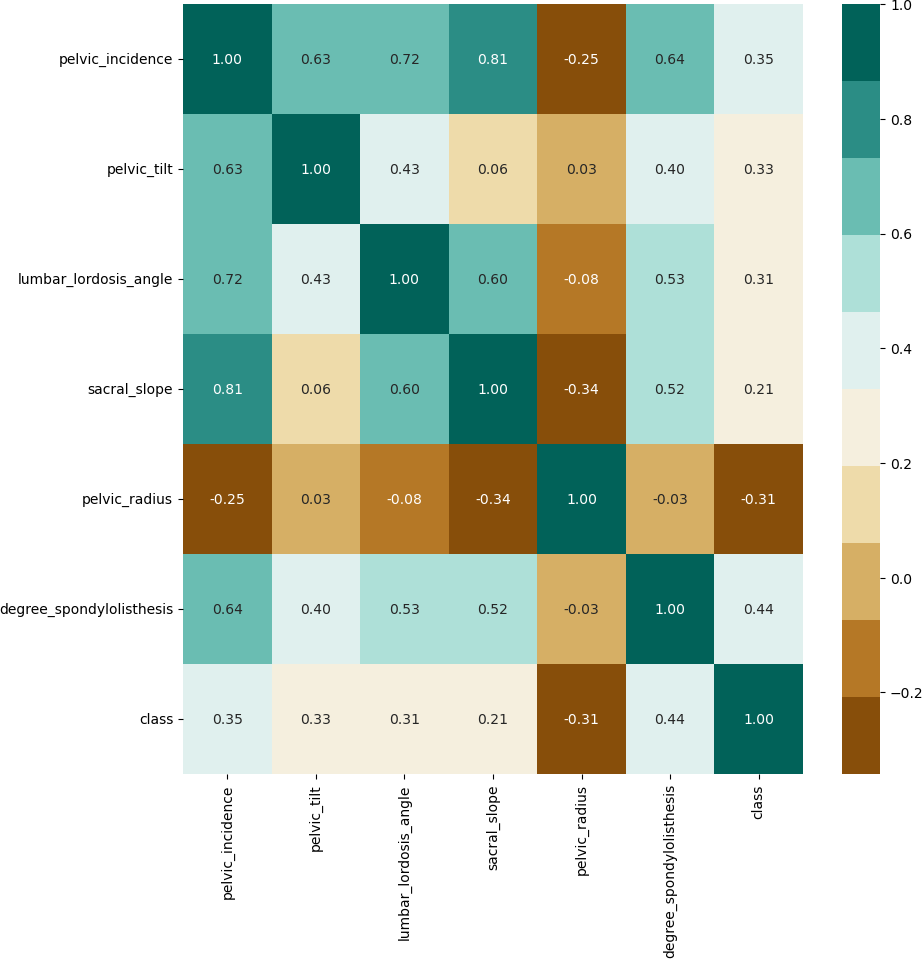
*# Generate Color Map*

*# 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()



**Challenge task:** Find other data from the UCI Machine Learning Repository. Using the previous code for reference, go explore!

# End of the Lab!