Nicolas Tan \ Spring 2022, *Bioengineering & Data Science* \ Project DS2001 @Northeastern University \ RISE 2023 Applicant: YOLOV4 To Be Implemented (Summer 2022)

NICOLAS' SPONDYLILOTHESIS PHYSIOLOGY PROJECT

Finding Level of Isthmic Spondylilothesis for Spinal Health of Athletes and Predicting Post Surgery Survivability within the elderly demographics

Figure 1. Lateral Radiographs of spondylotic lumbosacral spines. Transdical Screws are the highly attenuated structures attached to the L3-L4 vertebrae.

Abstract: Spondylolysis is one of the most common causes of lower back pain in regular exercisers and for the elderly. It remains asymptomatic in the majority of people who have it. This is detrimental as spondylolysis can progress to spondylolisthesis, which is defined as anterior displacement of the vertebral body in reference to the bordering vertebral bodies. When you injure your back, many tend to dismiss the asymptomatic sign in belief that you perhaps just got a muscle cramp or deadlifted a little too heavy but that in the end run will heal from it just like any other muscle recovery program for miniscular injuries. What people fail to realize is that there are an uncountable amount of variables that can form spondylolysis and often it is extremely hard to know if you ever have it. At abnormally high grade level for spondylilothesis, you can be very likely to have your lower back discs forever dislodged if not pursuing surgery and this can be life changing in a traumatically negative way -- leaving you in a wheelchair or even to some rare occasions dead. Mortality rate is not zero when talking about no treatment after having this for a while and even with surgery it can be dangerous for as to what side effects will result as this is still a niche field.

Knowing one's level of dorsal health is crucial for not only general surgery-inquiries or general health self-evaluation, but also with knowledge of one's lumbar erosive-eseque features, as you can then determine what your solution procedure should be to get rid of whatever your pain may be. Realizing the potential harm in not knowing if one potenitial has isthmic spondylolisthesis and to what level of concern, I felt the relevancy in pursuing a project on this for both patients in the fitness industry to mitigate putting one's back at further risk and to see if there is a way to predict if a middle-aged person has this asymptomatic condition.

Classifying someone's stage of isthmic spondylolisthesis severity based on clincal measurements, can we determine its clincal significance (whether the parcticipant is worth needing surgery treatment or not, mitigating financiala and hierarchical loss) and predict prior to surgery -- or even a doctor's visit one's recovery time, pain pinpoint in numerical value form, and more? Do non-anatomical variables result by cause of anatomical posture (which showcases spondolysi stage for

frequent deadlifters, or is there even a correlation at all to begin with? If so, can we use such understanding to learn how to fix one's posture? This is the first of 2 portions to my project.This is the first of 2 portions to my project. The second portion of the project will focus more on symptoms and pre-operative factors that might influence the 'danger' people get scared of when thinking of surgery, whether this is for the elderly or people with some underlying physioglical issues but who get injured from physcial activity and are thinking of going to get surgery.

In [ ]:

Updates:

- 3/29/22: For this assignment, we have to begin our project and plot out the first graphs.
- 4/14/22: Completed my official final of my project. Executed ML library knowledge to get strong predictors that will impact the healthcare industry.

Chapters:

- PART ONE: 'Quantiative'
  - *Areas of focus revolve around Anatomical Scrutinizing (slip disk), Correlations, Predicting, Classifications*
- PART TWO: 'Categorical'
  - *Areas of focus revolve around Elderly, Correlations, Predicting, Classifications, Discovery*
- PART THREE: 'Prediction'
  - _Areas of focus revolve Machine Learning, Train-Test Split, K-NN Algorithmn, Classification Reports, Accuracy Score, Confusion Matrix
- PART FOUR: (Coming Soon) 'Computer Vision'
  - *Areas of focus revolve around Fitness, Elderly, Gesture Control, Computer Vision, API/GUI, Posture Correctness*

In this project, I will be using two UCI derived repositories. The first dataset is anatomically related and showcasing quantitative biomechanic metrics, while the second set is a rendered symptomatic survey of categorical data displaying pre- and post-surgery questionnaire answers.

Both experiments occured in Warsaw and I personally discovered such data from Kaggle's database. The spinal dataset is going to be for the purpose of determining if we can stratify clumps of parcticipant' data into grades of spondylolithesis, as well as to determine association of slip grade to pelvic tilt. The thoracic surgery dataset will be centered around lower-grade spondylolithesis participants, discovering relations (if applicable) to developement of scoliosis as well as seeing if I can find a pattern to predict if a middle-aged person will not only need to attend surgery, but if they would survive long after the procedure considering their pre-operative symptoms. Sometimes, there are risks to take but other times it is not worth the sacrifice. I hope this project will serve well for readers of any age.

~ For readers new to Jupyter Notebook, please press _ctrl+enter_ to view each cell's results. Happy scrolling! ~

Dataset Sources:

- Siddhartha, Manu. “Thoracic Surgery Dataset.” Kaggle, 29 July 2019, \ https://www.kaggle.com/sid321axn/thoraric-surgery
- Hussain, Ali. “Lower Back Pain Symptoms Dataset(Labeled).” Kaggle, 5 Dec. 2017, \ https://www.kaggle.com/alihussain1993/lower-back-pain-symptoms-datasetlabelled

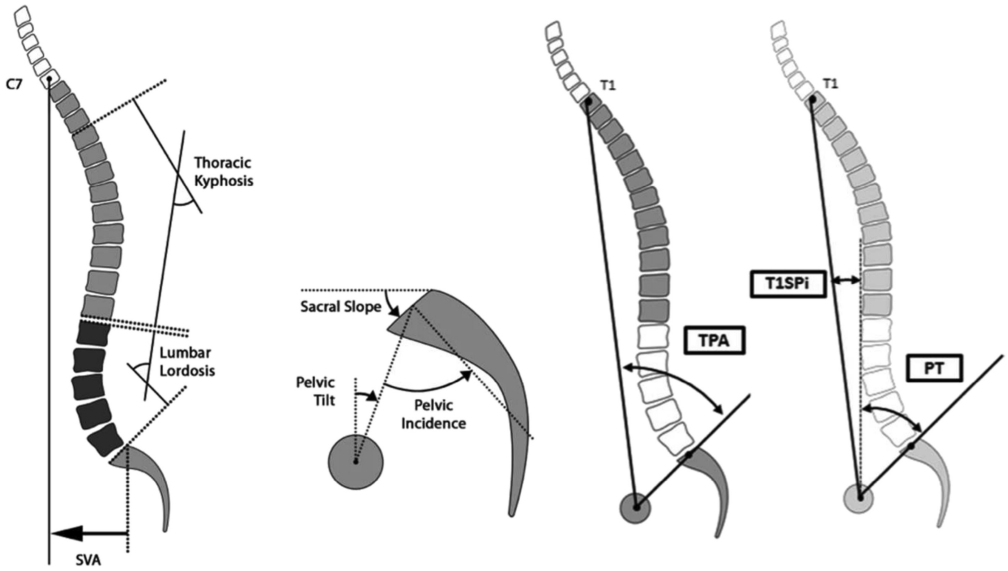
---

Research Background

Spinal measurements are clinically significant for a spinal surgeon before suggesting or shortlisting suitable surgical intervention procedure. Traditionally, the spinal surgeon evaluates the condition of the parcticipant prior to a surgical procedure, so as to verify the usefulness and effectiveness of the adopted procedure. In the case of spinal fusion procedures for your L3 vertebrae, where the lower back region prone to disk slips, will the fusion procedure be able to restore the spinal balance is a question for which the answered is obtained through making relevant spinal measurements, including lumbar lordotic curve angle, both segmental and for whole lumbar spine, lumbosacral angle, spinal heights, dimensions of vertebral bodies etc.

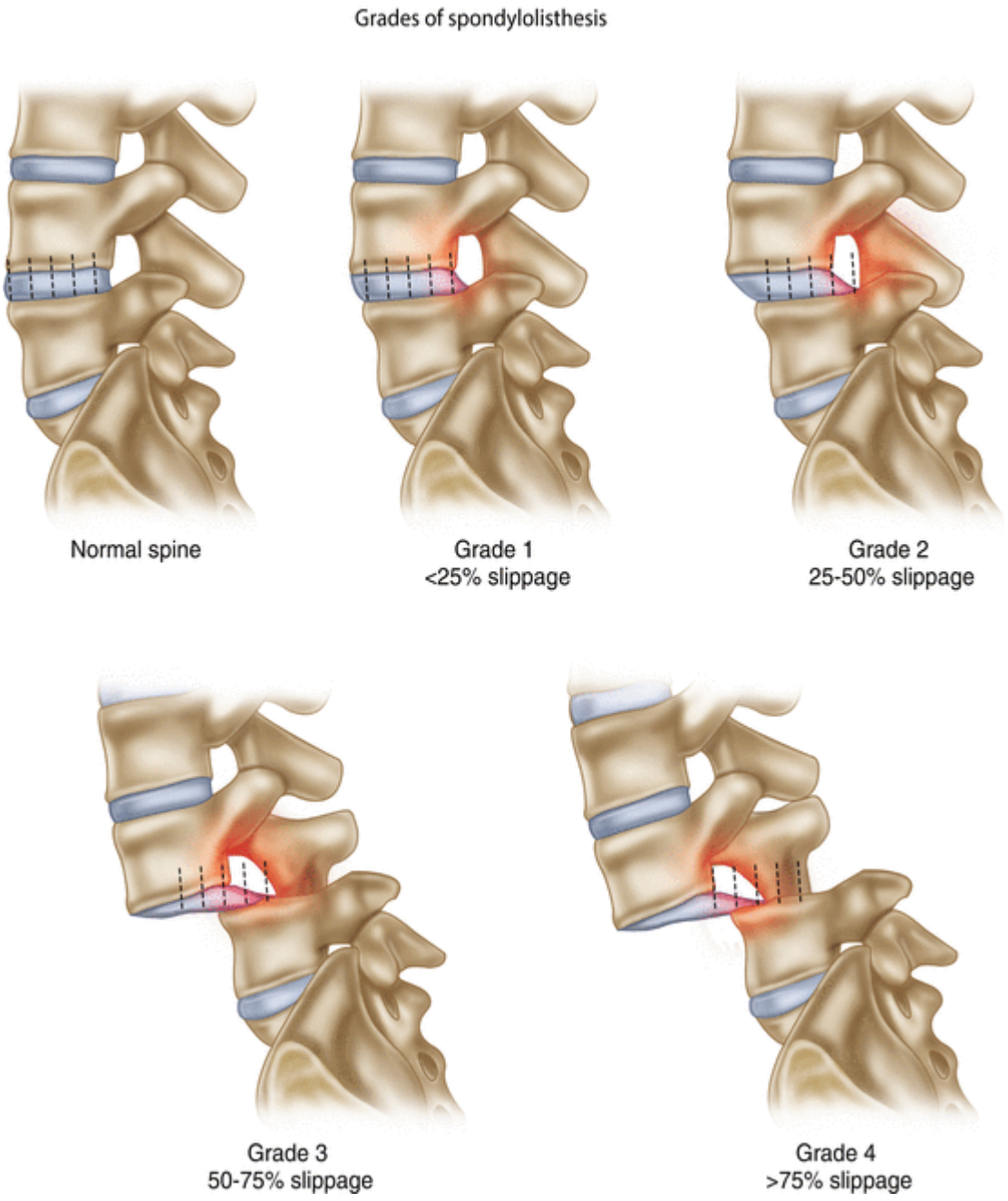
Spondylolysis is defined as an anatomical defect or fracture of the pars interarticularis of the vertebral arch. The pars interarticularis is an isthmus of bone connecting the superior and inferior facet surfaces in the spine at a given level. Spondylolysis occurs at the L5 vertebrae between 85 and 95% of the time and occurs at the L4 vertebrae 5–15% of the time. The two lowest sections on the spinal cord is the L4 followed underneath it with the L5 (Sports Medicine, 3).

While Degenerative spondylolisthesis occurs mostly at the L4-L5 (fourth and fifth lumbar vertebrae) level as opposed to its isthmic counterpart, which occurs most often at the lumbosacral level (L5-S1) . When isthmic spondylolisthesis results in the irritation or impingement of a nerve root, it can result in sciatica pain. Spondololisthesis may vary in terms of its causuality, but often it is either through excessive external loading (eg. weightlifting) or through age, where disk herniations begin as result of bone erosion. Sciatica refers to pain that radiates along the path of the sciatic nerve, which branches from your lower back through your hips and buttocks and down each leg. It is a condition in which compression of one of its these roots or branches occurs due to excessive loading typically. A sharp shooting pain in the leg, usually on one side, is a very common symptom to look out for.



Pelvic incidence, sacral slope and slip percentage have been shown to be important predicting factors for assessing the risk of progression of low and high grade spondylolisthesis. To determine when someone has IS (Isthmic Spondylolisthesis), one can verify through Spino-Pelvic Paramters (PI (Pelvic Incidence), Slip Percentage, and PT (Pelvic Tilt), SS (Sacral slope), LSA (Lumbosacral Angle)) in close scrutiny with the progression of IS -- assesing the differences of parameters of those undergoing thoracic surgery with someone who has IS to better understand the influence IS has on pre & most importantly post-thoracic surgery (Chung et al, 2).

Parcticipants undergo lumbar pedicle screw fixation and fusion for degenerative spondylolisthesis. In aged yet remaninigly accurate study by Duval-Beaupère, the SS, PT, and PI remain the top three most cruicial variables to idetnify key patterns to more easily dtermine the state one's spondylolithesis stage is at. There are 4 grades in spondylolithesis, each representing a level of severity and need for surgery before death: Stage I-IV. Grade 1 is when the degree of pelvic tilt is <25% slippage angle. Grade 2 is 25-50% slippage. Grade 3 is 50-75% slippage. Grade 4 is >75% slippage. As degrees of slippage is a percentage, just convert decimal to percentage if given in decimal format. GRADE 3 onwards need surgery.



In [ ]:

While we can be using technically Matplotlib as our only library and go out of our way to find methods to manuever around not having correlation features, I decided to use the much more efficient EDA approach. Self-taught, I used Pandas, Numpy and Seaborn alongside the standard Matplotlib library for accuracy purposes.

In [3]:

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import seaborn as sns
import math
```

Spinal CSV Cleaning and Extrapolation

In [2]:
```
#Let's Run The File

raw_data_1 = "Dataset_spine.csv"

"""
While we can technically start off using 'with open(raw_data_1, 'r') as file' and proce
I will be using as much Pandas as this library makes organization and viewing more fine
"""

raw1 = pd.read_csv(raw_data_1)
raw1
```

Out[2]:

	pelvic_incidence	pelvic_tilt	lumbar_lordosis_angle	sacral_slope	pelvic_radius	degree_spondylolisth
0	63.027817	22.552586	39.609117	40.475232	98.672917	-0.254
1	39.056951	10.060991	25.015378	28.995960	114.405425	4.564
2	68.832021	22.218482	50.092194	46.613539	105.985135	-3.530
3	69.297008	24.652878	44.311238	44.644130	101.868495	11.211
4	49.712859	9.652075	28.317406	40.060784	108.168725	7.918
...	...	...	...	...	...	
305	47.903565	13.616688	36.000000	34.286877	117.449062	-4.245
306	53.936748	20.721496	29.220534	33.215251	114.365845	-0.421
307	61.446597	22.694968	46.170347	38.751628	125.670725	-2.707
308	45.252792	8.693157	41.583126	36.559635	118.545842	0.214
309	33.841641	5.073991	36.641233	28.767649	123.945244	-0.199

310 rows × 13 columns

In [3]:
```
# Although nothing is wrong with the above iterable made, to conduct _pd_ commands, we
# Let's create a Pandas DataFrame instead of a scalar (Series or numpy list).

df = pd.DataFrame(raw1)
df.head()
```

Out[3]:

	pelvic_incidence	pelvic_tilt	lumbar_lordosis_angle	sacral_slope	pelvic_radius	degree_spondylolisthesi
0	63.027817	22.552586	39.609117	40.475232	98.672917	-0.25440
1	39.056951	10.060991	25.015378	28.995960	114.405425	4.56425
2	68.832021	22.218482	50.092194	46.613539	105.985135	-3.53031
3	69.297008	24.652878	44.311238	44.644130	101.868495	11.21152
4	49.712859	9.652075	28.317406	40.060784	108.168725	7.91850

We want to filter out unneeded columns now to properly data munge & clean.

In [638…

```
# Use .loc to choose all rows and specific columns to look at. Parameters: row, list of
# For this case study, we are focusing heavily on pelvic metrics.
# Variables: PI PT, LLA, SS, DS, Class Attr

filtered = df.loc[:, ['pelvic_incidence', 'pelvic_tilt', 'lumbar_lordosis_angle','scoli
filtered
```

Out[638…

	pelvic_incidence	pelvic_tilt	lumbar_lordosis_angle	scoliosis_slope	degree_spondylolisthesis	Class_
0	63.027817	22.552586	39.609117	43.5123	-0.254400	Abnorn
1	39.056951	10.060991	25.015378	16.1102	4.564259	Abnorn
2	68.832021	22.218482	50.092194	19.2221	-3.530317	Abnorn
3	69.297008	24.652878	44.311238	18.8329	11.211523	Abnorn
4	49.712859	9.652075	28.317406	24.9171	7.918501	Abnorn
…	…	…	…	…	…	
305	47.903565	13.616688	36.000000	11.5472	-4.245395	Norn
306	53.936748	20.721496	29.220534	43.8693	-0.421010	Norn
307	61.446597	22.694968	46.170347	18.4151	-2.707880	Norn
308	45.252792	8.693157	41.583126	33.7192	0.214750	Norn
309	33.841641	5.073991	36.641233	40.6049	-0.199249	Norn

310 rows × 6 columns

In [ ]:

"To dive into correlation of spondylolisthesis grade, represented indirectly here through our 5th column (degree of spondylolithesis per parcticipant), we must section off the two groups of study -- our experimental and control group." For the first set, the goal was to extract only the columns desired (data munging) & look at stratified groups, sectioned by two Normal vs. Abnormal parcticipants

In [ ]:

Normal & Abnormal: Sectioning

In [639…

```
norm_raw = filtered[filtered.loc[:,'Class_att'] == "Normal"]
#this line is saying 'In dataframe df, locate all rows where column Class_att is Normal
#you are only going to output a boolean of each row telling if that Class_att row is in
#column data, you want to add another df.loc to basically say 'display the rows that ar
norm_raw
```

Out[639...

	pelvic_incidence	pelvic_tilt	lumbar_lordosis_angle	scoliosis_slope	degree_spondylolisthesis	Class_a
210	38.505273	16.964297	35.112814	18.0442	7.986683	Norm
211	54.920858	18.968430	51.601455	9.8711	2.001642	Norm
212	44.362490	8.945435	46.902096	33.2503	4.994195	Norm
213	48.318931	17.452121	48.000000	10.3379	-0.910941	Norm
214	45.701789	10.659859	42.577846	34.2846	-3.388910	Norm
...	...	...	...	...	...	
305	47.903565	13.616688	36.000000	11.5472	-4.245395	Norm
306	53.936748	20.721496	29.220534	43.8693	-0.421010	Norm
307	61.446597	22.694968	46.170347	18.4151	-2.707880	Norm
308	45.252792	8.693157	41.583126	33.7192	0.214750	Norm
309	33.841641	5.073991	36.641233	40.6049	-0.199249	Norm

100 rows × 6 columns

We see that majority of the degree of spondylolithesis in Normals are around -7 to +7 on average

In [640...

```
abnorm_raw = filtered.loc[filtered.loc[:,'Class_att'] == "Abnormal"]
abnorm_raw
```

Out[640...

	pelvic_incidence	pelvic_tilt	lumbar_lordosis_angle	scoliosis_slope	degree_spondylolisthesis	Class_a
0	63.027817	22.552586	39.609117	43.5123	-0.254400	Abnorn
1	39.056951	10.060991	25.015378	16.1102	4.564259	Abnorn
2	68.832021	22.218482	50.092194	19.2221	-3.530317	Abnorn
3	69.297008	24.652878	44.311238	18.8329	11.211523	Abnorn
4	49.712859	9.652075	28.317406	24.9171	7.918501	Abnorn
...	...	...	...	...	...	
205	80.111572	33.942432	85.101608	36.0749	100.292107	Abnorn
206	95.480229	46.550053	59.000000	38.7849	77.283072	Abnorn
207	74.094731	18.823727	76.032156	11.6844	73.388216	Abnorn
208	87.679087	20.365613	93.822416	28.6308	76.730629	Abnorn
209	48.259920	16.417462	36.329137	21.6135	28.343799	Abnorn

210 rows × 6 columns

We see that majority of the degree of spondylolithesis in Normals are around mid 10 to even 100 degrees. Very drastic in degrees gap

Why we section:

Theoretically, it would be aesthetic to have only that one column (Class Attr) to be the output when executing a df.loc command. This does not makes sense logically however because you would be only seeing the same value ('abnormal'/'normal') on repeat. There is not usefulness in having this. You want to use data from other columns in association to that filter of 'people who have/lack spondylolithesis' to compare results and in goal recognize a pattern to extrapolate potential enablers.

Clean once again but this time, remove the class_att column now for 'all numerical' vals. This will be useful for graphs that does not allow for Categorial data, such as that of a Seaborn plot -- especially Heatmaps.

In [641…

```
#use abnorm DF but remove the class_att column now for 'all numerical' vals to make a s
#.drop removes columns or rows. The first parameter tells what you want removed and the
# axis = 1 means the first param is a column. axis = 2 means the first param is a row.
abnorm = abnorm_raw.drop('Class_att', axis=1)
abnorm
```

Out[641…

	pelvic_incidence	pelvic_tilt	lumbar_lordosis_angle	scoliosis_slope	degree_spondylolisthesis
0	63.027817	22.552586	39.609117	43.5123	-0.254400
1	39.056951	10.060991	25.015378	16.1102	4.564259
2	68.832021	22.218482	50.092194	19.2221	-3.530317
3	69.297008	24.652878	44.311238	18.8329	11.211523
4	49.712859	9.652075	28.317406	24.9171	7.918501
...	...	...	...	...	...
205	80.111572	33.942432	85.101608	36.0749	100.292107
206	95.480229	46.550053	59.000000	38.7849	77.283072
207	74.094731	18.823727	76.032156	11.6844	73.388216
208	87.679087	20.365613	93.822416	28.6308	76.730629
209	48.259920	16.417462	36.329137	21.6135	28.343799

210 rows × 5 columns

In [642…

```
#repeat this step but for 'norm'
norm = norm_raw.drop('Class_att', axis=1)
norm
```

Out[642…

	pelvic_incidence	pelvic_tilt	lumbar_lordosis_angle	scoliosis_slope	degree_spondylolisthesis
210	38.505273	16.964297	35.112814	18.0442	7.986683
211	54.920858	18.968430	51.601455	9.8711	2.001642
212	44.362490	8.945435	46.902096	33.2503	4.994195

	pelvic_incidence	pelvic_tilt	lumbar_lordosis_angle	scoliosis_slope	degree_spondylolisthesis
213	48.318931	17.452121	48.000000	10.3379	-0.910941
214	45.701789	10.659859	42.577846	34.2846	-3.388910
...	...	...	...	...	...
305	47.903565	13.616688	36.000000	11.5472	-4.245395
306	53.936748	20.721496	29.220534	43.8693	-0.421010
307	61.446597	22.694968	46.170347	18.4151	-2.707880
308	45.252792	8.693157	41.583126	33.7192	0.214750
309	33.841641	5.073991	36.641233	40.6049	-0.199249

100 rows × 5 columns

Now, we can find the total size per the two groups & bar chart plot them.

In [ ]:

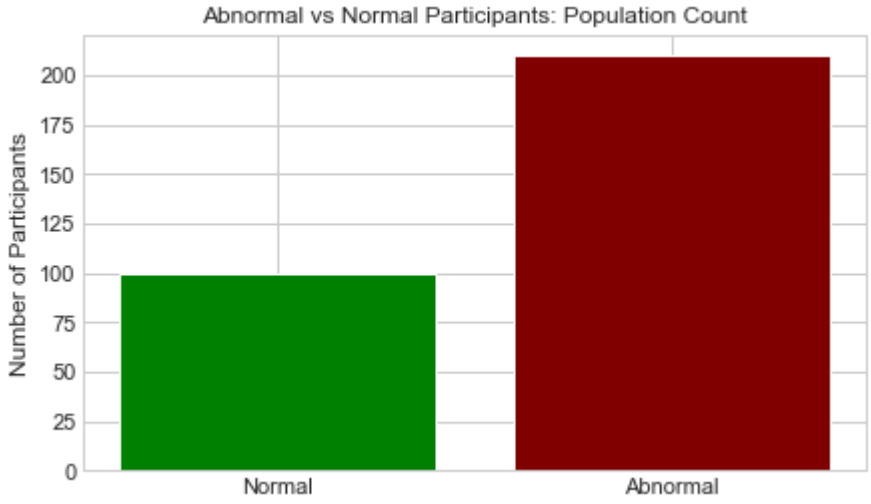
To understand why UCI chose to make certain cell rows abnormal and others normal:

In [370…

```
#y = Amount of patients considered (ab)normal, x= categorical name (abnormal/normal)
plt.bar(['Normal'], len(norm), color = 'green')
plt.bar(['Abnormal'], len(abnorm), color = 'maroon')
plt.title('Abnormal vs Normal Participants: Population Count')
plt.ylabel('Number of Participants')
```

Out[370…

```
Text(0, 0.5, 'Number of Participants')
```

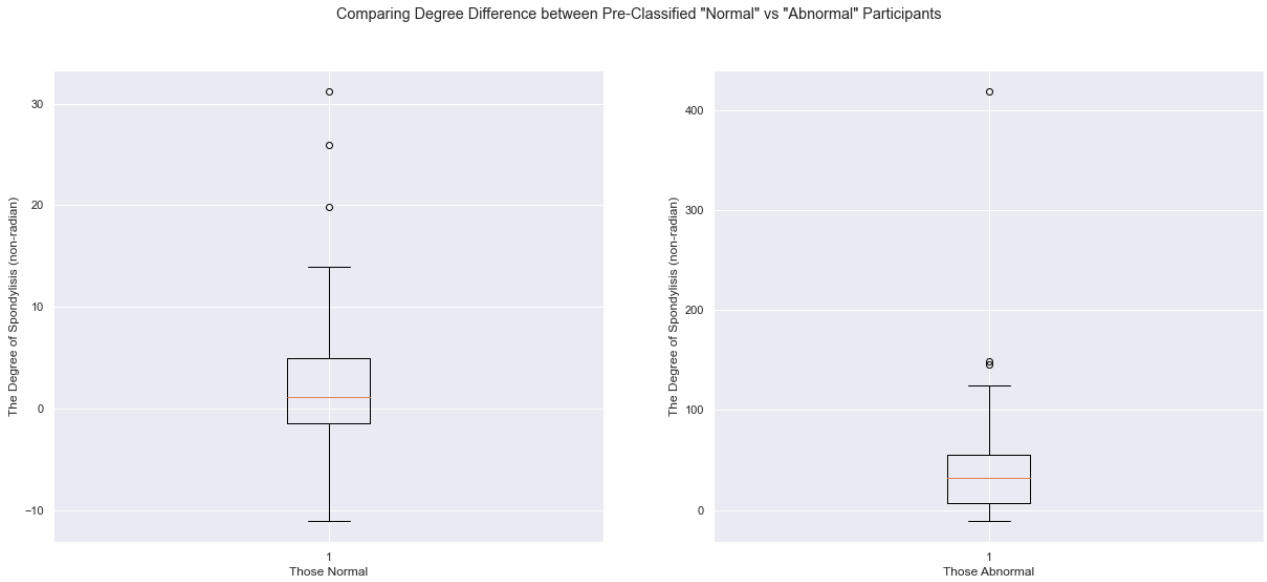


In [674…

```
#With whiskers:
fig, (ax1, ax2) = plt.subplots(1, 2)
fig.suptitle('Comparing Degree Difference between Pre-Classified "Normal" vs "Abnormal"
ax1.boxplot(norm['degree_spondylolisthesis'])
ax2.boxplot(abnorm['degree_spondylolisthesis'])
```

```
ax1.set_xlabel('Those Normal')
ax2.set_xlabel('Those Abnormal')

ax1.set_ylabel('The Degree of Spondylisis (non-radian)')
ax2.set_ylabel('The Degree of Spondylisis (non-radian)')
```

Out[674…
```
Text(0, 0.5, 'The Degree of Spondylisis (non-radian)')
```



In [372…
```
print('The mean degree for the normal patients is {}'.format(np.mean(norm['degree_spond
print('The mean degree for the abnormal patients is {}'.format(np.mean(abnorm['degree_s
```

```
The mean degree for the normal patients is 2.1865720613099997
The mean degree for the abnormal patients is 37.77770509337141
```

As can seen through analysis, we can realize that the normal ones have been classified as normal because while not neccasarily all 0 degrees (grade 0), they are all under 3 degrees. By experimenting tolerance, I discovered that for Big Data, such plus-minus acceptance of tolerance includes around +-5 degrees of freedom. Thus, this is why the normal participants are not all 0 degrees. For abnormal, it makes sense that majority are 25 degrees or more (Grade 1+).

In [ ]:
```

```

Let's head into the main section. To see if correlated, associated, or none of the above with the given participant data, I decided to try a heatmap for each parameter in respect to degrees of spondolilothesis.

About Heatmaps:

We can choose to do a mutlviaraite log-binomial regression or for easier understanding on a visual standpoint -- a heatmap regression. This is a very common choice for human pose esimtation methods as well, which will be discussed in Part Three for Computer Vision.

In heatmaps, the ground-truth heatmaps are usually constructed via covering all skeletal keypoints by 2D gaussian kernels. The standard deviations of these kernels are fixed. The '1.0' is a diagonal line

because they are comparing a variable with itself but other than that, we look at those below that triangular shape to see which has the closest relations and what we can learn from that.

A heatmap contains values representing various shades of the same colour for each value to be plotted. Usually the darker shades of the chart represent higher values than the lighter shade. For a very different value a completely different colour can also be used. For simplicity at a cost of more lines, you can have an overlay with a text displaying the Pearson coefficient on each cell. *This can be done by creating a* ***df.corr()*** *keyword add-in, rounding to 2 if desired.* This will be the df param in your sns heatmap command, essentially overlapping each cell with the Pearon.

The best heatmaps come from the Seaborn library, so I will use that. The syntax for creating a heatmap (to readers new to the data science libraries in Python) is: **sns.heatmap**(clean_df)

Let's check which parameters are correlated to each other (have importance). The areas of focus that have potential correlation (and even conntributors of causation) towards how many degrees a participant's angle of spinal disc is are: _pelvic_incidence, pelvic_tilt, lumbar_lordosis_angle, sacral_slope, pelvic_radius & degree_spondylolisthesis_

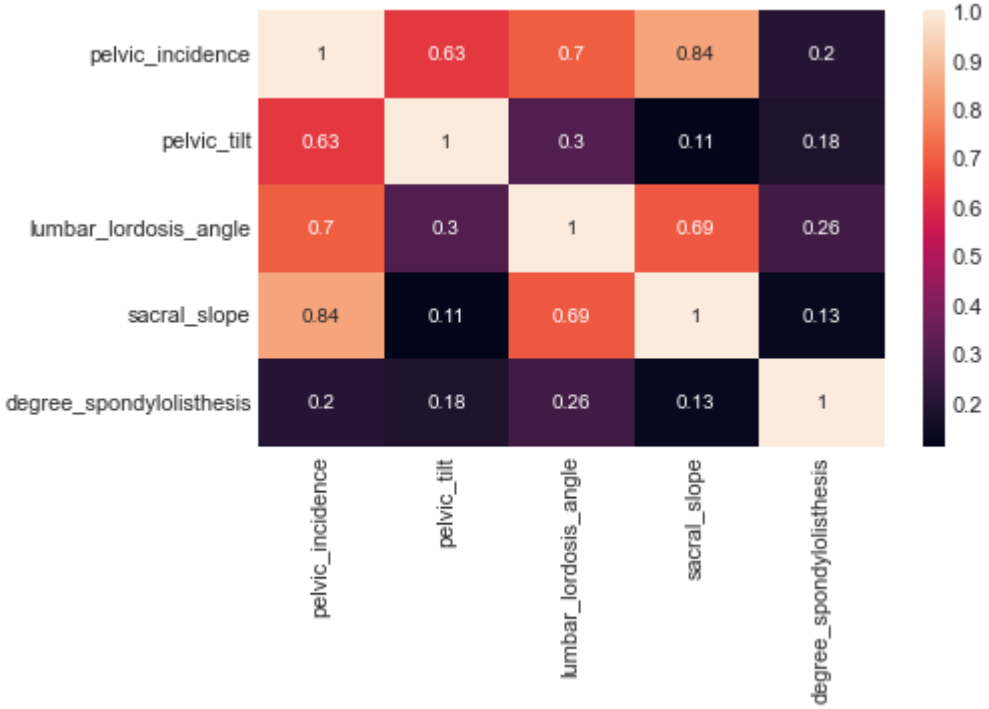
In [ ]:

Identifying Trends & Relations: Pelvic Parameters

NORMAL Grade 0 Spondolilothesis Participants | HEATMAP:

In [373…

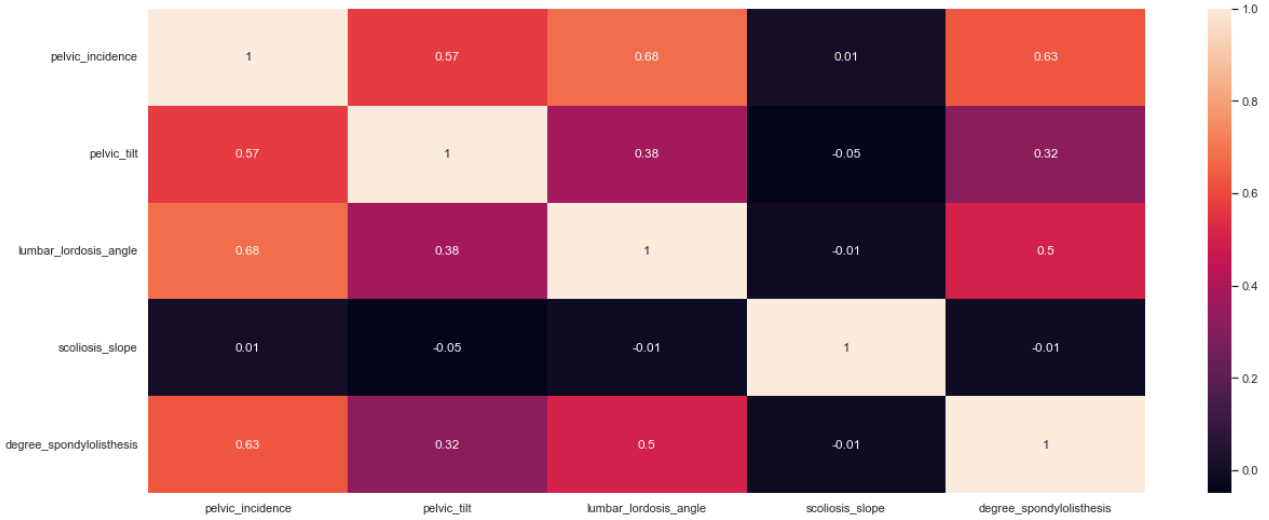
```
#Heatmap (enter cell twice for expanded view)
matrix = norm.corr().round(2)
sns.heatmap(matrix, annot=True)
sns.set(rc={'figure.figsize':(20.7,8.27)})
plt.show()
```



ABNORMAL Grade 1-4 Spondolilothesis Participants | HEATMAP:

In [644…

```
#Heatmap
matrix = abnorm.corr().round(2)
sns.heatmap(matrix, annot=True)
sns.set(rc={'figure.figsize':(20.7,8.27)})
plt.show()
```



Side Note: The 'rc' param is a parameter mapping that overrides the values in the preset seaborn style dictionaries. It is needed to modify fig size.

In [ ]:

Let's say we want a more narrow, 1-to-1 analysis. We can additionally perform linear regressions on Scatter plots & General Mixed Relations.

For both groups, I want to focus on our pelvic parameters in respect to degrees of spondylolisthesis ('DS'). I want to see if each grade of spondylolisthesis is influenced by the following variables: PI, PT, LLA, SSS

To visualize this, we can stratificate our 'DS' into four parts, one per grade level. We can parse this by doing something similar to: \ "_sns.boxplot(x = 'PI', y = 'another*var', hue = 'DS', data = df)*", where hue is the legend but also the stratifier that parses this example up based on degree if already parsed, say, in a list. Here, I just want to focus on 2 variables at a time, so a standard scatterplot with a linear regression embedding is solid enough. We need to section the 'degree_spondolisthesis' variable apart.

Reminder of slippages: \ *Negative Degrees are treated the same -- based on the absolute of the numerical value.*

* Grade 0 = 0% (eg. 0)

* Grade 1 = 0.01 - 24.99% (eg. 0.001-0.25)

* Grade 2 = 25 - 50% (eg. 0.25-0.5)

* Grade 3 = 50-75% (eg. 0.5-0.75)

* Grade 4 = 75+% (eg. 0.75+)

_In Panda's df.loc() command, where df is the variable name of your data frame, you can have multiple condittionels to filter for specific rows in a specific column. You can do this by having df.loc[df[desired_col] = desired_row*val]. The input arg can be having more args via a simple combinatorial symbo like "&" (treated as an 'and' keyword) or "|" (treated as an 'or' keyword).*

Here, we use the given knowledge of our ranges per grade into these 4 grade variables governed by such data structures' format.

In [ ]:

Parsing into Slippage Grade Levels: Normal Patients

In [331…

```
#grade 0
G0 = norm.loc[(norm['degree_spondylolisthesis'] == 0)]
G0
#= none
```

Out[331…

	pelvic_incidence	pelvic_tilt	lumbar_lordosis_angle	sacral_slope	pelvic_radius	degree_spondylolisthesis

In [332…

```
#grade 1
G1 = norm.loc[(norm['degree_spondylolisthesis'] > 0) & (norm['degree_spondylolisthesis'
G1
#Majority
```

Out[332…

	pelvic_incidence	pelvic_tilt	lumbar_lordosis_angle	sacral_slope	pelvic_radius	degree_spondylolisth
210	38.505273	16.964297	35.112814	21.540976	127.632875	7.986
211	54.920858	18.968430	51.601455	35.952428	125.846646	2.001
212	44.362490	8.945435	46.902096	35.417055	129.220682	4.994
217	38.126589	6.557617	50.445075	31.568971	132.114805	6.338
218	51.624672	15.969344	35.000000	35.655328	129.385308	1.009
219	64.311867	26.328369	50.958964	37.983498	106.177751	3.118
224	89.834676	22.639217	90.563461	67.195460	100.501192	3.040
225	59.726140	7.724873	55.343485	52.001268	125.174221	3.235
226	63.959522	16.060945	63.123736	47.898577	142.360125	6.298
227	61.540599	19.676957	52.892229	41.863642	118.686268	4.815

	pelvic_incidence	pelvic_tilt	lumbar_lordosis_angle	sacral_slope	pelvic_radius	degree_spondylolisth
228	38.046551	8.301669	26.236830	29.744881	123.803413	3.885
231	53.911054	12.939318	39.000000	40.971736	118.193035	5.074
232	43.117951	13.815744	40.347388	29.302207	128.517722	0.970
234	37.731992	9.386298	42.000000	28.345694	135.740926	13.683
236	61.821627	13.597105	64.000000	48.224523	121.779803	1.296
237	62.140805	13.960975	58.000000	48.179830	133.281834	4.955
238	69.004913	13.291790	55.570143	55.713123	126.611621	10.832
241	51.529358	13.517847	35.000000	38.011510	126.718516	13.928
244	63.026300	27.336240	51.605017	35.690060	114.506608	7.439
246	46.637864	15.853717	40.000000	30.784147	119.377603	9.064
247	49.828135	16.736435	28.000000	33.091700	121.435558	1.913
248	47.319648	8.573680	35.560252	38.745967	120.576972	1.630
249	50.753290	20.235060	37.000000	30.518231	122.343516	2.288
251	40.746996	1.835524	50.000000	38.911472	139.247150	0.668
253	63.792425	21.345323	66.000000	42.447102	119.550391	12.382
254	72.955644	19.576971	61.007071	53.378673	111.234047	0.813
256	54.752520	9.752520	48.000000	45.000000	123.037999	8.235
258	40.349296	10.194748	37.967747	30.154548	128.009927	0.458
260	54.142408	11.935110	43.000000	42.207298	122.209083	0.153
261	74.976021	14.921705	53.730072	60.054317	105.645400	1.594
262	42.517272	14.375671	25.323565	28.141601	128.905689	0.757
265	48.170746	9.594217	39.710920	38.576530	135.623310	5.360
267	52.862214	9.410372	46.988052	43.451842	123.091240	1.856
268	57.145851	16.489091	42.842148	40.656760	113.806178	5.015
269	37.140150	16.481240	24.000000	20.658910	125.014361	7.366
271	42.515610	16.541216	42.000000	25.974394	120.631941	7.876
272	39.358705	7.011262	37.000000	32.347443	117.818760	1.904
274	43.191915	9.976664	28.938149	33.215251	123.467400	1.741
275	67.289712	16.717514	51.000000	50.572198	137.591778	4.960
276	51.325464	13.631223	33.258578	37.694240	131.306122	1.788
279	48.801909	18.017762	52.000000	30.784147	139.150407	10.442
280	50.086153	13.430044	34.457541	36.656108	119.134622	3.089
281	64.261507	14.497866	43.902504	49.763642	115.388268	5.951

	pelvic_incidence	pelvic_tilt	lumbar_lordosis_angle	sacral_slope	pelvic_radius	degree_spondylolisth
282	53.683380	13.447022	41.584297	40.236358	113.913703	2.737
283	48.995958	13.113820	51.873520	35.882137	126.398188	0.535
284	59.167612	14.562749	43.199158	44.604863	121.035642	2.830
285	67.804694	16.550662	43.256802	51.254033	119.685645	4.867
286	61.734875	17.114312	46.900000	44.620563	120.920200	3.087
287	33.041688	-0.324678	19.071075	33.366366	120.388611	9.354
288	74.565015	15.724320	58.618582	58.840695	105.417304	0.599
290	36.422485	13.879424	20.242562	22.543061	126.076861	0.179
291	51.079833	14.209935	35.951229	36.869898	115.803711	6.905
293	48.902904	5.587589	55.500000	43.315316	137.108289	19.854
295	46.426366	6.620795	48.100000	39.805571	130.350096	2.449
297	45.575482	18.759135	33.774143	26.816347	116.797007	3.131
298	66.507179	20.897672	31.727471	45.609507	128.902905	1.517
299	82.905351	29.894119	58.250542	53.011232	110.708958	6.079
301	89.014875	26.075981	69.021259	62.938894	111.481075	6.061
308	45.252792	8.693157	41.583126	36.559635	118.545842	0.214

In [333…

```
#grade 1's average degrees in PT, PI, and LL are
    #use numpy library commands 'df.mean(desired_col)' to do mathematical operations on
g1_norm_avgPT = np.mean(G1['pelvic_tilt'])
g1_norm_avgPI = np.mean(G1['pelvic_incidence'])
g1_norm_avgLLA =  np.mean(G1['lumbar_lordosis_angle'])
print('The averages of healthy Grade 1 Patients\' pelvic tilt, pelvic incidence, and lu
```

```
The averages of healthy Grade 1 Patients' pelvic tilt, pelvic incidence, and lumbar lord
osis angle are: 14.371583387050842, 54.22449460220339, 44.814289113050854
```

In [335…

```
#grade 2
G2 = norm.loc[(norm['degree_spondylolisthesis'] > 25) & (norm['degree_spondylolisthesis
G2
# = 2 Rows
```

Out[335…

	pelvic_incidence	pelvic_tilt	lumbar_lordosis_angle	sacral_slope	pelvic_radius	degree_spondylolisth
222	56.103774	13.106307	62.637020	42.997467	116.228503	31.172
255	67.538182	14.655042	58.001429	52.883139	123.632260	25.970

In [336…

```
g2_norm_avgPT = np.mean(G2['pelvic_tilt'])
g2_norm_avgPI = np.mean(G2['pelvic_incidence'])
```

```
g2_norm_avgLLA =  np.mean(G2['lumbar_lordosis_angle'])
print('The averages of healthy Grade 2 Patients\' pelvic tilt, pelvic incidence, and lu
```

```
The averages of healthy Grade 2 Patients' pelvic tilt, pelvic incidence, and lumbar lord
osis angle are: 13.880674435, 61.82097752999999, 60.3192243
```

In [337…

```
#grade 3
G3 = norm.loc[(norm['degree_spondylolisthesis'] > 0.5) & (norm['degree_spondylolisthesi
G3
#3 rows
```

Out[337…

	pelvic_incidence	pelvic_tilt	lumbar_lordosis_angle	sacral_slope	pelvic_radius	degree_spondylolisth
251	40.746996	1.835524	50.000000	38.911472	139.247150	0.668
283	48.995958	13.113820	51.873520	35.882137	126.398188	0.535
288	74.565015	15.724320	58.618582	58.840695	105.417304	0.599

In [728…

```
g3_norm_avgPT = np.mean(G3['pelvic_tilt'])
g3_norm_avgPI = np.mean(G3['pelvic_incidence'])
g3_norm_avgLLA =  np.mean(G3['lumbar_lordosis_angle'])
print('The averages of healthy Grade 3 Patients\' pelvic tilt, pelvic incidence, and lu
```

```
The averages of healthy Grade 3 Patients' pelvic tilt, pelvic incidence, and lumbar lord
osis angle are: 10.224554893666667, 54.76932308666667, 53.49736746666667
```

In [729…

```
#grade 4
G4 = norm.loc[(norm['degree_spondylolisthesis'] > 75)]
G4
#none
```

Out[729…

	pelvic_incidence	pelvic_tilt	lumbar_lordosis_angle	scoliosis_slope	degree_spondylolisthesis

To view these filtered tables in a collaborative visual, we use scatterplots. Having overlay features can help us annotate to see the trends/patterns firsthand in our participant database.

In [ ]:

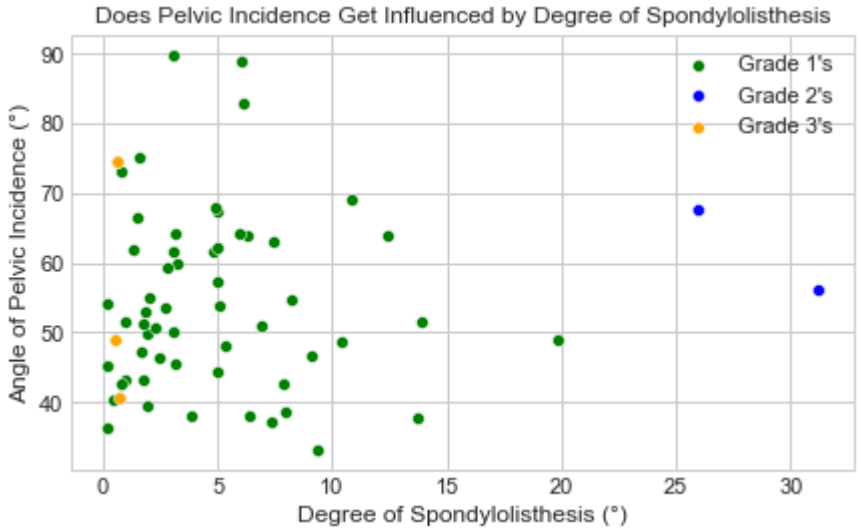
NORMAL Grade 1-4 Spondolilothesis Participants | SCATTERPLOT:

(1.1) Scoping Pelvic Parameters: Pelvic Incidence

In [735…

```
import warnings
warnings.filterwarnings('ignore')

""" We only need the non-empty grades to be compared"""

sns.scatterplot('degree_spondylolisthesis','pelvic_incidence', data = G1, color = 'gree
sns.scatterplot('degree_spondylolisthesis','pelvic_incidence', data = G2, color = 'blue
sns.scatterplot('degree_spondylolisthesis','pelvic_incidence', data = G3, color = 'oran
```

```
plt.title('Does Pelvic Incidence Get Influenced by Degree of Spondylolisthesis')
plt.xlabel('Degree of Spondylolisthesis (°)')
plt.ylabel('Angle of Pelvic Incidence (°)')
#Reminder: G0 and G4 participants are non-existing here for Normal Participants
plt.legend(['Grade 1\'s','Grade 2\'s','Grade 3\'s'], facecolor = 'lightgrey' , edgecolo
plt.show()
```



What more can be done: correlation finding of two vars can be better achieved via:

- an actual coeff of regression
    - This can be achieved via numpy's Pearson correlation coefficient OR through training with Scikit Learn. For simplicity, I have chosen to use numpy's.
    - An example would be "*Degree of Spondylilothesis vs rest: stacked horizontal bar graph*" ==> basically a #pie chart from highest to lowest correlation:
        - Example 1: numpy.cocoref(my_rho = np.corrcoef(var1, var2))
        - Example 2: df.corr(method ='pearson')
          `* Optional method param initializations are: "kendall": Kendall Tau correlation coefficient or "spearman": Spearman rank correlation`
        - Example 3: df['A'].corr(df['B'])
          `* We will use this in our case, as we want to extrapolate from a pandas dataframe, which yields different results than if using numpy.`
          `* In addition though, we want to not find the correlation of every cell but only two specific columns.`

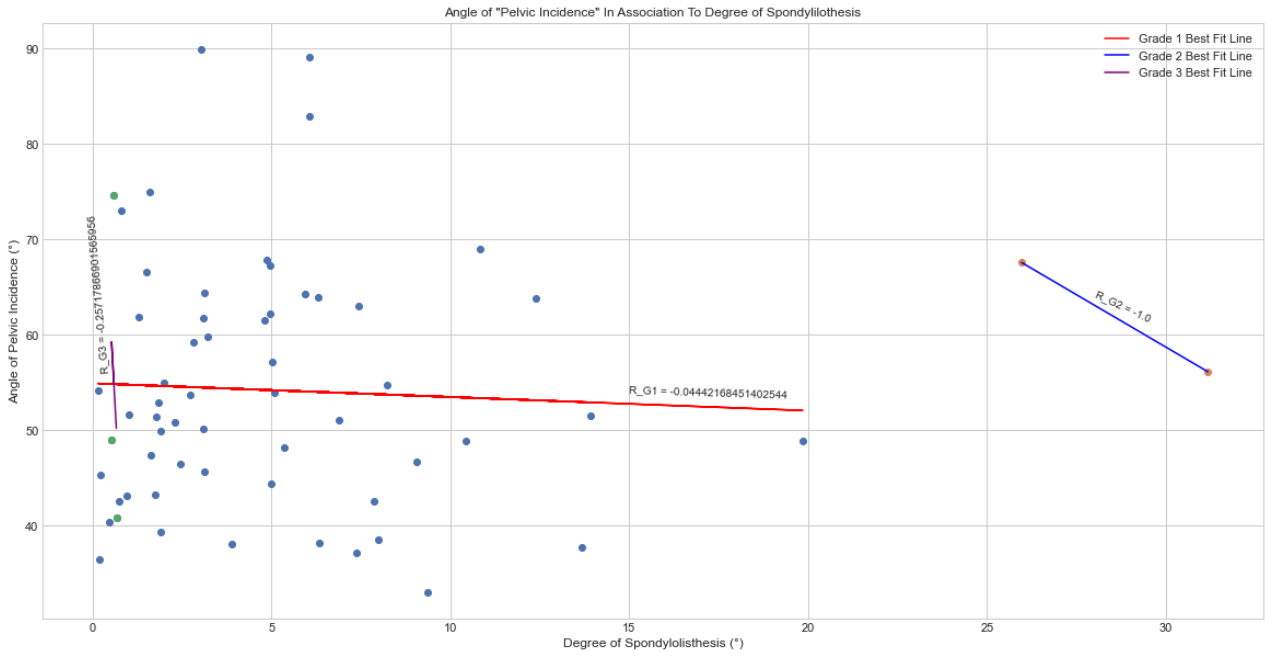
Let's Add Extra Parameters To Enhance The Associations Data Points Have On a Visual Scale

...

In [795…

```
plt.figure(figsize = [20,10])

#grade 1: BEST FIT LINES
x1 = G1['degree_spondylolisthesis']
y1 =  G1['pelvic_incidence']

a, b = np.polyfit(x1,y1, 1)
plt.scatter(x1, y1)
```

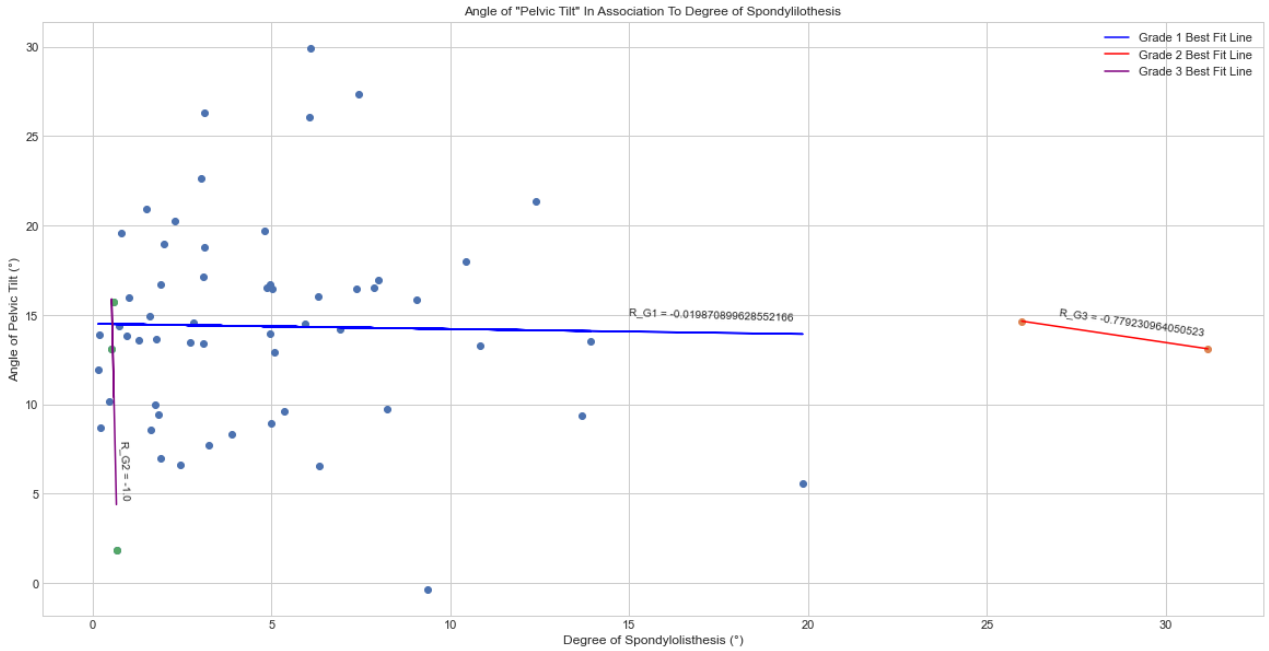
```
plt.plot(x1, a*x+b, color = 'red')

#grade 2: BEST FIT LINES
x2= G2['degree_spondylolisthesis']
y2 =  G2['pelvic_incidence']

a2, b2 = np.polyfit(x2,y2, 1)
plt.scatter(x2, y2)

plt.plot(x2, a2*x2+b2, color = 'blue')

#grade 3: BEST FIT LINES
x3 = G3['degree_spondylolisthesis']
y3 =  G3['pelvic_incidence']

a3, b3 = np.polyfit(x3,y3, 1)
plt.scatter(x3, y3)

plt.plot(x3, a3*x3+b3, color = 'purple')

plt.title('Angle of "Pelvic Incidence" In Association To Degree of Spondylilothesis')
plt.xlabel('Degree of Spondylolisthesis (°)')
plt.ylabel('Angle of Pelvic Incidence (°)')
plt.legend(['Grade 1 Best Fit Line','Grade 2 Best Fit Line','Grade 3 Best Fit Line'], f

#Adding correlation numbers on the best fit lines THROUGH 'TEXT ADDON' Within Plotly Gr
"""
By order: the correfs displayed are for G1, G2, G3,
positioned above each line via rotation diagonal (to align with each plotted line)
"""

plt.text(15,54, 'R_G1 = {}'.format(x1.corr(y1)), rotation=-1.8, rotation_mode='anchor',
plt.text(28,64, 'R_G2 = {}'.format(x2.corr(y2)), rotation=-25, rotation_mode='anchor',
plt.text(0.45,56, 'R_G3 = {}'.format(x3.corr(y3)), rotation=95, rotation_mode='anchor',

#Display Final Result
plt.show()
```



(1.2) Scoping Pelvic Parameters: Pelvic Tilt

In [811…

```
plt.figure(figsize = [20,10])

#grade 1: BEST FIT LINES
x1 = G1['degree_spondylolisthesis']
y1 =  G1['pelvic_tilt']

a, b = np.polyfit(x1,y1, 1)
plt.scatter(x1, y1)

plt.plot(x1, a*x+b, color = 'blue')

#grade 2: BEST FIT LINES
x2= G2['degree_spondylolisthesis']
y2 =  G2['pelvic_tilt']

a2, b2 = np.polyfit(x2,y2, 1)
plt.scatter(x2, y2)

plt.plot(x2, a2*x2+b2, color = 'red')

#grade 3: BEST FIT LINES
x3 = G3['degree_spondylolisthesis']
y3 =  G3['pelvic_tilt']

a3, b3 = np.polyfit(x3,y3, 1)
plt.scatter(x3, y3)

plt.plot(x3, a3*x3+b3, color = 'purple')

plt.title('Angle of "Pelvic Tilt" In Association To Degree of Spondylilothesis')
plt.xlabel('Degree of Spondylolisthesis (°)')
plt.ylabel('Angle of Pelvic Tilt (°)')
plt.legend(['Grade 1 Best Fit Line','Grade 2 Best Fit Line','Grade 3 Best Fit Line'], f

#Adding correlation numbers on the best fit lines THROUGH 'TEXT ADDON' Within Plotly Gr
"""
By order: the correfs displayed are for G1, G2, G3,
positioned above each line via rotation diagonal (to align with each plotted line)
"""

plt.text(15,15, 'R_G1 = {}'.format(x1.corr(y1)), rotation=-1.8, rotation_mode='anchor',
plt.text(0.8,8, 'R_G2 = {}'.format(x2.corr(y2)), rotation=-88, rotation_mode='anchor',
plt.text(27,15, 'R_G3 = {}'.format(x3.corr(y3)), rotation=-8, rotation_mode='anchor', f

#Display Final Result
plt.show()
```

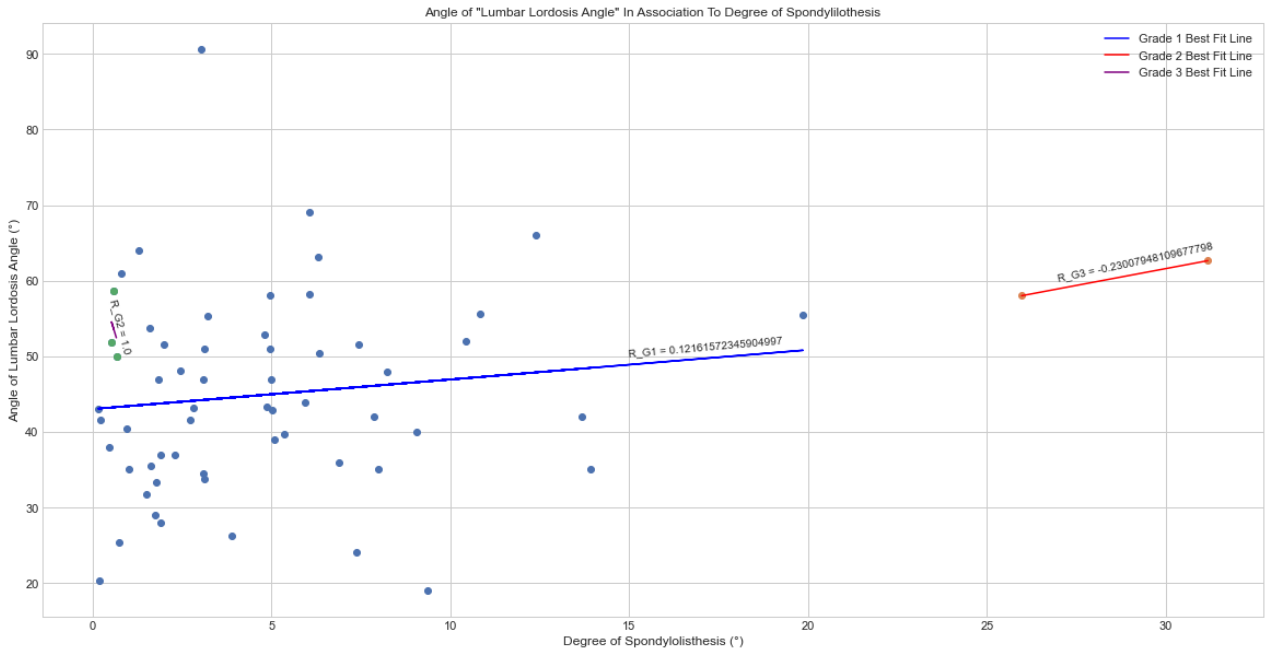


**(1.2) Scoping Pelvic Parameters: Lumbar Lordosis **

In [801…

```
plt.figure(figsize = [20,10])

#grade 1: BEST FIT LINES
x1 = G1['degree_spondylolisthesis']
y1 =  G1['lumbar_lordosis_angle']

a, b = np.polyfit(x1,y1, 1)
plt.scatter(x1, y1)

plt.plot(x1, a*x1+b, color = 'blue')

#grade 2: BEST FIT LINES
x2= G2['degree_spondylolisthesis']
y2 =  G2['lumbar_lordosis_angle']

a2, b2 = np.polyfit(x2,y2, 1)
plt.scatter(x2, y2)

plt.plot(x2, a2*x2+b2, color = 'red')

#grade 3: BEST FIT LINES
x3 = G3['degree_spondylolisthesis']
y3 =  G3['lumbar_lordosis_angle']

a3, b3 = np.polyfit(x3,y3, 1)
plt.scatter(x3, y3)

plt.plot(x3, a3*x3+b3, color = 'purple')

plt.title('Angle of "Lumbar Lordosis Angle" In Association To Degree of Spondylilothesi
plt.xlabel('Degree of Spondylolisthesis (°)')
plt.ylabel('Angle of Lumbar Lordosis Angle (°)')
plt.legend(['Grade 1 Best Fit Line','Grade 2 Best Fit Line','Grade 3 Best Fit Line'], f

#Adding correlation numbers on the best fit lines THROUGH 'TEXT ADDON' Within Plotly Gr
"""
```

```
By order: the correfs displayed are for G1, G2, G3,
positioned above each line via rotation diagonal (to align with each plotted line)
"""

plt.text(15,50, 'R_G1 = {}'.format(x1.corr(y1)), rotation=5, rotation_mode='anchor', fo
plt.text(0.5,57.5, 'R_G2 = {}'.format(x2.corr(y2)), rotation=-75, rotation_mode='anchor
plt.text(27,60, 'R_G3 = {}'.format(x3.corr(y3)), rotation= 12, rotation_mode='anchor',

#Display Final Result
plt.show()
```



In [ ]:

We will repeat this for the abnorm dataframe...

In [ ]:

Parsing into Slippage Grade Levels: Patients with Known Spondylolithesis

In [27]:
```
#grade 0 for abnormal patients (= with spondylolithesis)
AB_G0 = abnorm.loc[(abnorm['degree_spondylolisthesis'] == 0)]
AB_G0
```

Out[27]:

	pelvic_incidence	pelvic_tilt	lumbar_lordosis_angle	sacral_slope	pelvic_radius	degree_spondylolisthesis

In [28]:
```
#grade 1 for abnormal patients (= with spondylolithesis)
AB_G1 = abnorm.loc[(abnorm['degree_spondylolisthesis'] > 0) & (abnorm['degree_spondylol
AB_G1
```

Out[28]:

	pelvic_incidence	pelvic_tilt	lumbar_lordosis_angle	sacral_slope	pelvic_radius	degree_spondylolisth
1	39.056951	10.060991	25.015378	28.995960	114.405425	4.564
3	69.297008	24.652878	44.311238	44.644130	101.868495	11.211
4	49.712859	9.652075	28.317406	40.060784	108.168725	7.918
5	40.250200	13.921907	25.124950	26.328293	130.327871	2.230
6	53.432928	15.864336	37.165934	37.568592	120.567523	5.988
8	43.790190	13.533753	42.690814	30.256437	125.002893	13.289
9	36.686353	5.010884	41.948751	31.675469	84.241415	0.664
11	31.232387	17.715819	15.500000	13.516568	120.055399	0.499
12	48.915551	19.964556	40.263794	28.950995	119.321358	8.028
13	53.572170	20.460828	33.100000	33.111342	110.966698	7.044
14	57.300227	24.188885	47.000000	33.111342	116.806587	5.766
15	44.318907	12.537992	36.098763	31.780915	124.115836	5.415
17	31.276012	3.144669	32.562996	28.131342	129.011418	3.623
18	38.697912	13.444749	31.000000	25.253163	123.159251	1.429
20	43.922840	14.177959	37.832547	29.744881	134.461016	6.451
21	54.919443	21.062332	42.200000	33.857110	125.212716	2.432
22	63.073611	24.413803	54.000000	38.659808	106.424329	15.779
25	54.124920	26.650489	35.329747	27.474432	121.447011	1.571
27	43.580964	16.508884	47.000000	27.072080	109.271634	8.992
28	44.551012	21.931147	26.785916	22.619865	111.072920	2.652
30	50.819268	15.402213	42.528939	35.417055	112.192804	10.869
31	46.390260	11.079047	32.136553	35.311213	98.774546	6.386
32	44.936675	17.443838	27.780576	27.492837	117.980324	5.569
33	38.663257	12.986441	40.000000	25.676816	124.914118	2.703
34	59.595540	31.998244	46.560252	27.597296	119.330354	1.474
35	31.484218	7.826221	24.284818	23.657997	113.833145	4.393
36	32.090987	6.989378	35.998198	25.101609	132.264735	6.413
38	55.843286	28.847448	47.690543	26.995838	123.311845	2.812
39	52.419385	19.011561	35.872660	33.407825	116.559771	1.694
41	46.442078	8.395036	29.037230	38.047043	115.481405	2.045
42	53.854798	19.230643	32.779060	34.624155	121.670915	5.329
46	48.332638	22.227784	36.181993	26.104854	117.384625	6.481
49	41.767732	17.899402	20.030886	23.868330	118.363389	2.062

	pelvic_incidence	pelvic_tilt	lumbar_lordosis_angle	sacral_slope	pelvic_radius	degree_spondylolisth
50	55.285852	20.440118	34.000000	34.845733	115.877017	3.558
51	74.433593	41.557331	27.700000	32.876262	107.949304	5.000
52	50.209670	29.760122	36.104007	20.449548	128.292515	5.740
53	30.149936	11.917445	34.000000	18.232491	112.684141	11.463
55	47.657730	13.277385	36.679985	34.380345	98.249781	6.273
56	43.349606	7.467469	28.065483	35.882137	112.776187	5.753
57	46.855781	15.351514	38.000000	31.504267	116.250917	1.662
59	48.109236	14.930725	35.564683	33.178512	124.056452	7.947
64	76.147212	21.936186	82.961502	54.211027	123.932010	10.431
68	72.076278	18.946176	51.000000	53.130102	114.213013	1.010
79	47.744679	12.089351	39.000000	35.655328	117.512004	21.682
85	45.443750	9.906072	45.000000	35.537678	163.071041	20.315
86	59.785265	17.879323	59.206461	41.905942	119.319111	22.123
88	56.605771	16.800200	42.000000	39.805571	127.294522	24.018
105	65.007964	27.602608	50.947519	37.405357	116.581109	7.015
118	65.536003	24.157487	45.775170	41.378515	136.440302	16.378
131	69.781006	13.777465	58.000000	56.003541	118.930666	17.914
167	72.343594	16.420790	59.869012	55.922805	70.082575	12.072
169	44.253476	1.101087	38.000000	43.152390	98.274107	23.910
170	64.809541	15.174078	58.839994	49.635463	111.679961	21.407
171	78.401254	14.042260	79.694263	64.358994	104.731234	12.392
172	56.668293	13.458203	43.769710	43.210089	93.692209	21.108
173	50.825029	9.064729	56.300000	41.760300	78.999454	23.041
174	61.411737	25.384364	39.096869	36.027373	103.404597	21.843

In [29]:

```
g1_abnorm_avgPT = np.mean(AB_G1['pelvic_tilt'])
g1_abnorm_avgPI = np.mean(AB_G1['pelvic_incidence'])
g1_abnorm_avgLLA =  np.mean(AB_G1['lumbar_lordosis_angle'])
print('The averages of healthy Grade 3 Patients\' pelvic tilt, pelvic incidence, and lu
```

```
The averages of healthy Grade 3 Patients' pelvic tilt, pelvic incidence, and lumbar lord
osis angle are: 16.85401188992983, 51.355172357719304, 40.24043157912281
```

In [30]:

```
#grade 2 for abnormal patients (= with spondylolithesis)
AB_G2 = abnorm.loc[(abnorm['degree_spondylolisthesis'] > 25) & (abnorm['degree_spondylo
AB_G2
```

Out[30]:

	pelvic_incidence	pelvic_tilt	lumbar_lordosis_angle	sacral_slope	pelvic_radius	degree_spondylolisth
62	44.529051	9.433234	52.000000	35.095817	134.711772	29.106
63	77.690577	21.380645	64.429442	56.309932	114.818751	26.931
65	83.933009	41.286305	62.000000	42.646703	115.012334	26.588
66	78.491730	22.181798	60.000000	56.309932	118.530327	27.383
69	58.599529	-0.261499	51.500000	58.861028	102.042812	28.059
...	...	...	...	...	...	
199	75.298478	16.671484	61.296204	58.626995	118.883388	31.575
200	63.364339	20.024621	67.498705	43.339718	130.999258	37.556
203	73.635962	9.711318	63.000000	63.924644	98.727930	26.975
204	56.535051	14.377189	44.991547	42.157862	101.723334	25.773
209	48.259920	16.417462	36.329137	31.842457	94.882336	28.343

66 rows × 12 columns

In [31]:

```
g2_abnorm_avgPT = np.mean(AB_G2['pelvic_tilt'])
g2_abnorm_avgPI = np.mean(AB_G2['pelvic_incidence'])
g2_abnorm_avgLLA =  np.mean(AB_G2['lumbar_lordosis_angle'])
print('The averages of healthy Grade 3 Patients\' pelvic tilt, pelvic incidence, and lu
```

```
The averages of healthy Grade 3 Patients' pelvic tilt, pelvic incidence, and lumbar lord
osis angle are: 14.742573635969698, 65.38548599257575, 58.51165458500001
```

In [32]:

```
#grade 3 for abnormal patients (= with spondylolithesis)
AB_G3 = abnorm.loc[(abnorm['degree_spondylolisthesis'] > 50) & (abnorm['degree_spondylo
AB_G3
```

Out[32]:

	pelvic_incidence	pelvic_tilt	lumbar_lordosis_angle	sacral_slope	pelvic_radius	degree_spondylolisth
60	74.377678	32.053104	78.772013	42.324573	143.560690	56.125
67	75.649731	19.339799	64.148685	56.309932	95.903629	69.551
72	84.974132	33.021175	60.859873	51.952957	125.659534	74.333
74	72.222334	23.077711	91.000000	49.144624	137.736655	56.804
77	58.782548	7.667044	53.338941	51.115504	98.501157	51.584
80	77.106571	30.469994	69.480628	46.636577	112.151600	70.759
81	74.005541	21.122402	57.379502	52.883139	120.205963	74.555
82	88.623908	29.089453	47.564262	59.534455	121.764780	51.805
83	81.104100	24.794168	77.887020	56.309932	151.839857	65.214
84	76.326002	42.396204	57.200000	33.929797	124.267007	50.127
93	58.101935	14.837639	79.649838	43.264295	113.587655	50.237

	pelvic_incidence	pelvic_tilt	lumbar_lordosis_angle	sacral_slope	pelvic_radius	degree_spondylolisth
94	94.174822	15.380770	67.705721	78.794052	114.890113	53.255
96	96.657315	19.461581	90.211498	77.195734	120.673041	64.080
98	77.655119	22.432950	93.892779	55.222169	123.055707	61.211
111	84.998956	29.610098	83.352194	55.388858	126.912990	71.321
113	69.756665	19.279297	48.500000	50.477369	96.491370	51.169
116	70.484104	12.489488	62.417142	57.994617	114.190049	56.902
117	86.041280	38.750670	47.871405	47.290610	122.092954	61.988
122	80.074914	48.069531	52.403439	32.005383	110.709912	67.727
123	65.665347	10.540675	56.489135	55.124672	109.162777	53.932
127	80.433428	16.998479	66.536018	63.434949	116.438981	57.781
128	90.513961	28.272501	69.813942	62.241459	100.892160	58.823
132	69.626283	21.122751	52.766595	48.503532	116.803091	54.816
133	81.754419	20.123466	70.560440	61.630954	119.425086	55.506
134	52.204693	17.212673	78.094969	34.992020	136.972517	54.939
136	88.024499	39.844669	81.774473	48.179830	116.601538	56.766
138	72.054034	24.700737	79.874016	47.353297	107.172358	56.426
142	85.290173	18.278890	100.744220	67.011283	110.660701	58.884
146	85.581710	30.457039	78.231379	55.124672	114.866049	68.376
156	79.476978	26.732268	70.650982	52.744711	118.588669	61.700
158	57.035097	0.345728	49.198003	56.689369	103.048698	52.165
160	92.026308	35.392674	77.416963	56.633634	115.723530	58.057
162	118.144655	38.449501	50.838520	79.695154	81.024541	74.043
165	83.703177	20.268229	77.110598	63.434949	125.480174	69.279
168	95.382596	24.822631	95.157633	70.559965	89.307547	57.660
175	56.563824	8.961262	52.577846	47.602562	98.777115	50.701
178	80.654320	26.344379	60.898118	54.309940	120.103493	52.467
179	68.721910	49.431864	68.056012	19.290046	125.018517	54.691
182	75.437748	31.539454	89.600000	43.898294	106.829590	54.965
183	71.001941	37.515772	84.537093	33.486169	125.164232	67.771
185	91.468741	24.508177	84.620272	66.960564	117.307897	52.623
188	85.680950	38.650035	82.680977	47.030914	120.840707	61.959
189	82.406524	29.276422	77.054565	53.130102	117.042244	62.765
191	86.472905	40.303766	61.141012	46.169139	97.404189	55.752

	pelvic_incidence	pelvic_tilt	lumbar_lordosis_angle	sacral_slope	pelvic_radius	degree_spondylolisth
196	63.772391	12.763385	65.360524	51.009006	89.822741	55.995
207	74.094731	18.823727	76.032156	55.271004	128.405731	73.388

In [70]:
```
g3_abnorm_avgPT = np.mean(AB_G3['pelvic_tilt'])
g3_abnorm_avgPI = np.mean(AB_G3['pelvic_incidence'])
g3_abnorm_avgLLA =  np.mean(AB_G3['lumbar_lordosis_angle'])
print('The averages of healthy Grade 3 Patients\' pelvic tilt, pelvic incidence, and lu
```

```
The averages of healthy Grade 3 Patients' pelvic tilt, pelvic incidence, and lumbar lord
osis angle are: 25.544005013630443, 78.5719782395652, 70.46633484173913
```

In [344…
```
#grade 4 for abnormal patients (= with spondylolithesis)
AB_G4 = abnorm.loc[(abnorm['degree_spondylolisthesis'] > 75)]
AB_G4
```

Out[344…

	pelvic_incidence	pelvic_tilt	lumbar_lordosis_angle	sacral_slope	pelvic_radius	degree_spondylolisth
61	89.680567	32.704435	83.130732	56.976132	129.955476	92.027
71	86.900794	32.928168	47.794347	53.972627	135.075364	101.719
75	70.221452	39.822724	68.118403	30.398728	148.525562	145.378
76	86.753609	36.043016	69.221045	50.710593	139.414504	110.860
92	85.352315	15.844910	71.668660	69.507405	124.419787	76.020
95	57.522356	33.647075	50.909858	23.875281	140.981712	148.753
104	77.409333	29.396545	63.232302	48.012788	118.450731	93.563
107	78.425951	33.425951	76.277439	45.000000	138.554111	77.155
114	80.988074	36.843172	86.960602	44.144903	141.088149	85.872
115	129.834041	8.404475	48.384057	121.429566	107.690466	418.543
121	83.879941	23.077427	87.141512	60.802514	124.646072	80.555
135	77.121344	30.349874	77.481083	46.771470	110.611148	82.093
141	89.504947	48.903653	72.003423	40.601295	134.634291	118.353
143	60.626217	20.595958	64.535262	40.030259	117.225554	104.859
163	115.923261	37.515436	76.800000	78.407825	104.698603	81.198
192	74.469082	33.283157	66.942101	41.185925	146.466001	124.984
197	58.828379	37.577873	125.742385	21.250506	135.629418	117.314
201	67.513053	33.275590	96.283062	34.237463	145.601033	88.301
202	76.314028	41.933683	93.284863	34.380345	132.267286	101.218
205	80.111572	33.942432	85.101608	46.169139	125.593624	100.292
206	95.480229	46.550053	59.000000	48.930176	96.683903	77.283

	pelvic_incidence	pelvic_tilt	lumbar_lordosis_angle	sacral_slope	pelvic_radius	degree_spondylolisth
208	87.679087	20.365613	93.822416	67.313473	120.944829	76.730

In [35]:
```
g4_abnorm_avgPT = np.mean(AB_G4['pelvic_tilt'])
g4_abnorm_avgPI = np.mean(AB_G4['pelvic_incidence'])
g4_abnorm_avgLLA =  np.mean(AB_G4['lumbar_lordosis_angle'])
print('The averages of healthy Grade 3 Patients\' pelvic tilt, pelvic incidence, and lu
```

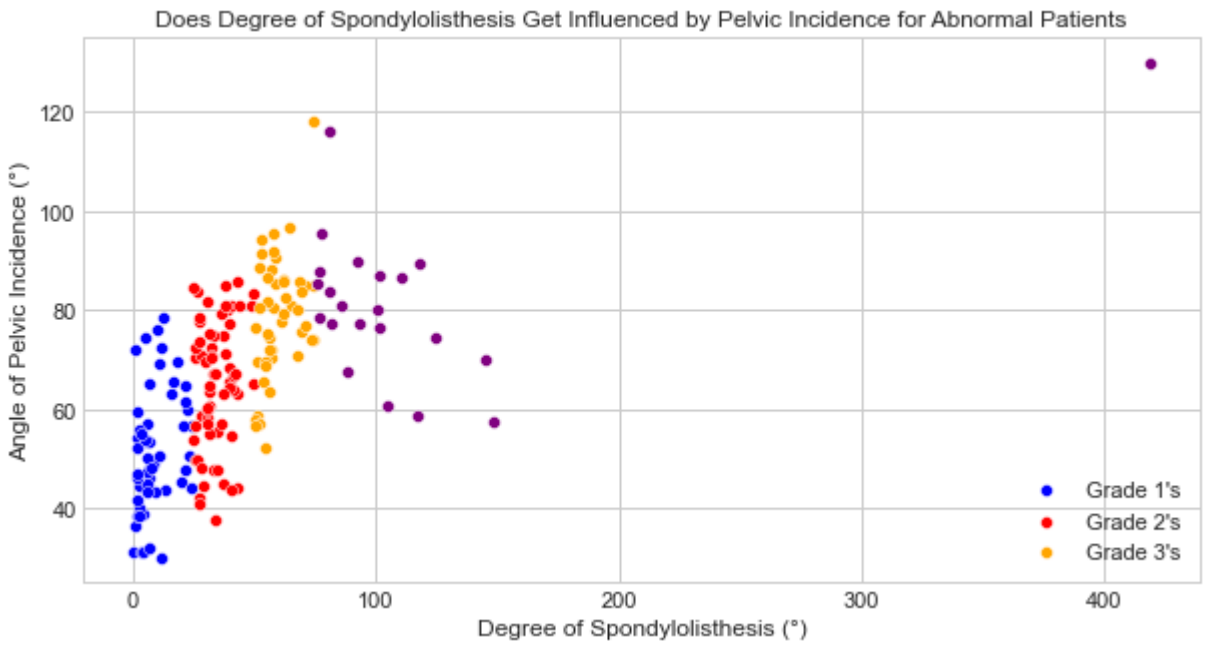
```
The averages of healthy Grade 3 Patients' pelvic tilt, pelvic incidence, and lumbar lord
osis angle are: 32.11051005840909, 82.29725597954545, 75.62887089818183
```

ABNORMAL Grade 1-4 Spondolilothesis Participants | SCATTERPLOT:

(2.1) Scoping Pelvic Parameters: Pelvic Incidence

In [792…
```
""" We only need the non-empty grades to be compared"""
plt.figure(figsize = [10,5])

sns.scatterplot('degree_spondylolisthesis','pelvic_incidence', data = AB_G0, color = 'g
sns.scatterplot('degree_spondylolisthesis','pelvic_incidence', data = AB_G1, color = 'b
sns.scatterplot('degree_spondylolisthesis','pelvic_incidence', data = AB_G2, color = 'r
sns.scatterplot('degree_spondylolisthesis','pelvic_incidence', data = AB_G3, color = 'o
sns.scatterplot('degree_spondylolisthesis','pelvic_incidence', data = AB_G4, color = 'p

plt.title('Does Degree of Spondylolisthesis Get Influenced by Pelvic Incidence for Abno
plt.xlabel('Degree of Spondylolisthesis (°)')
plt.ylabel('Angle of Pelvic Incidence (°)')
plt.legend(['Grade 1\'s','Grade 2\'s','Grade 3\'s'], facecolor = 'lightgrey' , edgecolo
```

Out[792…
```
<matplotlib.legend.Legend at 0x293a8e89370>
```

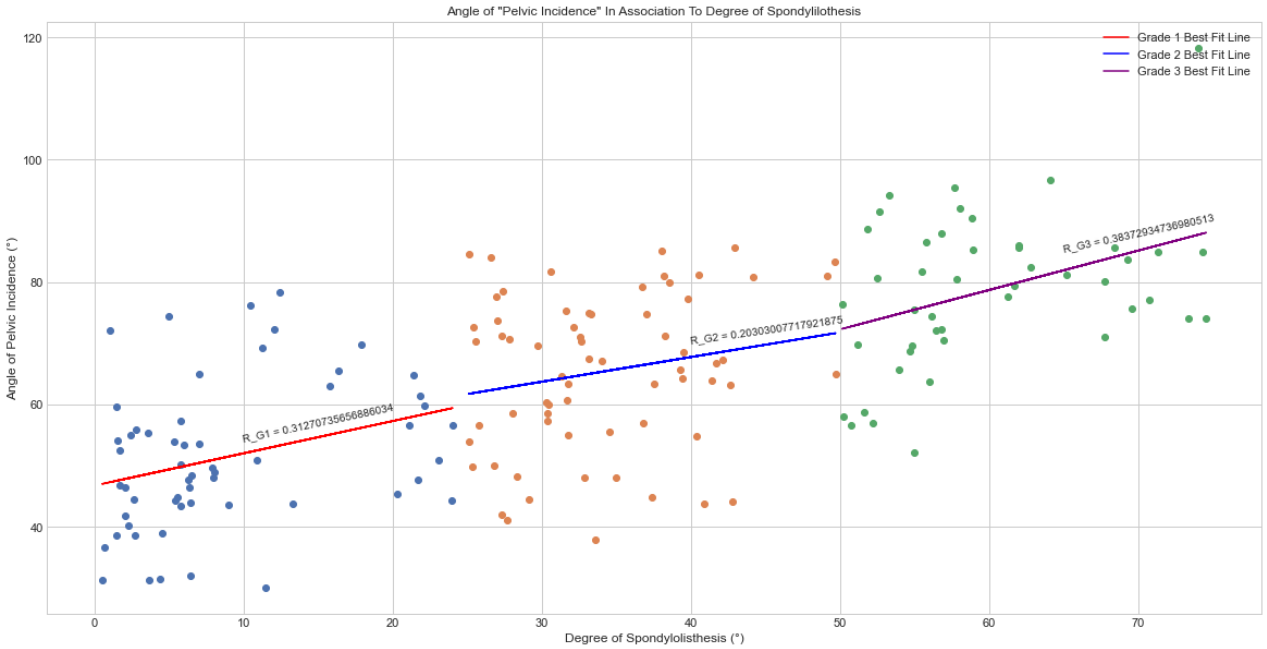


Let's Add Extra Parameters To Enhance The Associations Data Points Have On a Visual Scale

...

In [789…

```
plt.figure(figsize = [20,10])

#grade 1: BEST FIT LINES
x1 = AB_G1['degree_spondylolisthesis']
y1 = AB_G1['pelvic_incidence']

a, b = np.polyfit(x1,y1, 1)
plt.scatter(x1, y1)

plt.plot(x1, a*x1+b, color = 'red')

#grade 2: BEST FIT LINES
x2= AB_G2['degree_spondylolisthesis']
y2 =  AB_G2['pelvic_incidence']

a2, b2 = np.polyfit(x2,y2, 1)
plt.scatter(x2, y2)

plt.plot(x2, a2*x2+b2, color = 'blue')

#grade 3: BEST FIT LINES
x3 = AB_G3['degree_spondylolisthesis']
y3 =  AB_G3['pelvic_incidence']

a3, b3 = np.polyfit(x3,y3, 1)
plt.scatter(x3, y3)

plt.plot(x3, a3*x3+b3, color = 'purple')

plt.title('Angle of "Pelvic Incidence" In Association To Degree of Spondylilothesis')
plt.xlabel('Degree of Spondylolisthesis (°)')
plt.ylabel('Angle of Pelvic Incidence (°)')
plt.legend(['Grade 1 Best Fit Line','Grade 2 Best Fit Line','Grade 3 Best Fit Line'], f

#Adding correlation numbers on the best fit lines THROUGH 'TEXT ADDON' Within Plotly Gr
"""
By order: the correfs displayed are for G1, G2, G3,
positioned above each line via rotation diagonal (to align with each plotted line)
"""

plt.text(10,54, 'R_G1 = {}'.format(x1.corr(y1)), rotation=12, rotation_mode='anchor', f
plt.text(40,70, 'R_G2 = {}'.format(x2.corr(y2)), rotation=8, rotation_mode='anchor', fo
plt.text(65,85, 'R_G3 = {}'.format(x3.corr(y3)), rotation=12, rotation_mode='anchor', f


#Display Final Result
plt.show()
```



What I find so fascinating about this chart is how, compared to NORM's, the three level grades of filled data has been beautifully sparsed in an organic form, clustered to such a tight space that one could even mistake it as a graphical form from a K-Means Clustering -- a powerful unsupervised machine learning method of vector quantization that aims at partitioning *n* observations into *k* clusters. In Python, this is done with the Sci-Kit Learn library (*eg. sklearn.cluster.KMeans*).

As far as strength in the association of Angle of Pelvic Incidence compared to Degree of Spondylolithesis, we see that over time, the angle gets progressively worse and as result increases degree of spondylilothesis as well. While the regression that can be seen is linear, so as to view how these two variables affect each other ('*causation*'), we cannot say the same based on the fluctuating Pearson values from grade 1 to 3.

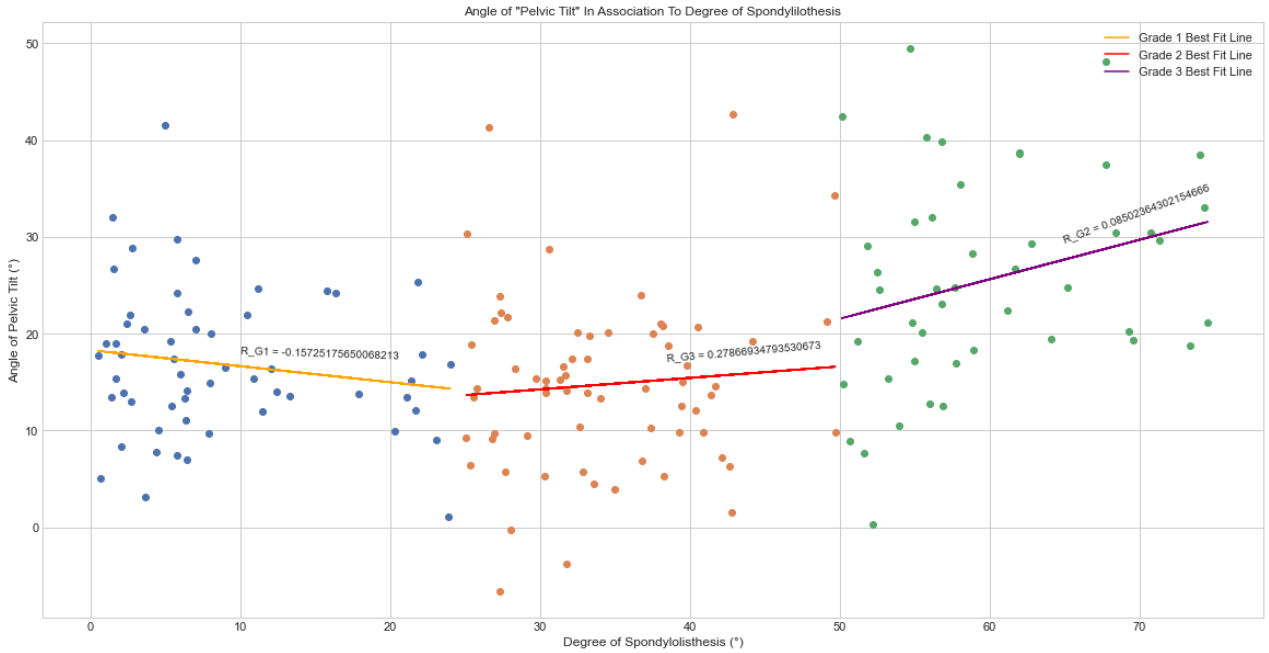
In [ ]:

(2.2) Scoping Pelvic Parameters: Pelvic Tilt

In [787…

```
plt.figure(figsize = [20,10])

#grade 1: BEST FIT LINES
x1 = AB_G1['degree_spondylolisthesis']
y1 = AB_G1['pelvic_tilt']

a, b = np.polyfit(x1,y1, 1)
plt.scatter(x1, y1)

plt.plot(x1, a*x1+b, color = 'orange')

#grade 2: BEST FIT LINES
x2= AB_G2['degree_spondylolisthesis']
y2 =  AB_G2['pelvic_tilt']

a2, b2 = np.polyfit(x2,y2, 1)
plt.scatter(x2, y2)
```

```
plt.plot(x2, a2*x2+b2, color = 'red')

#grade 3: BEST FIT LINES
x3 = AB_G3['degree_spondylolisthesis']
y3 =  AB_G3['pelvic_tilt']

a3, b3 = np.polyfit(x3,y3, 1)
plt.scatter(x3, y3)

plt.plot(x3, a3*x3+b3, color = 'purple')

plt.title('Angle of "Pelvic Tilt" In Association To Degree of Spondylilothesis')
plt.xlabel('Degree of Spondylolisthesis (°)')
plt.ylabel('Angle of Pelvic Tilt (°)')
plt.legend(['Grade 1 Best Fit Line','Grade 2 Best Fit Line','Grade 3 Best Fit Line'], f

#Adding correlation numbers on the best fit lines THROUGH 'TEXT ADDON' Within Plotly Gr
"""
By order: the correfs displayed are for G1, G2, G3,
positioned above each line via rotation diagonal (to align with each plotted line)
"""

plt.text(10,18, 'R_G1 = {}'.format(x1.corr(y1)), rotation=-1.8, rotation_mode='anchor',
plt.text(65,29.5, 'R_G2 = {}'.format(x2.corr(y2)), rotation=20, rotation_mode='anchor',
plt.text(38.5,17.2, 'R_G3 = {}'.format(x3.corr(y3)), rotation=5, rotation_mode='anchor'

#Display Final Result
plt.show()
```

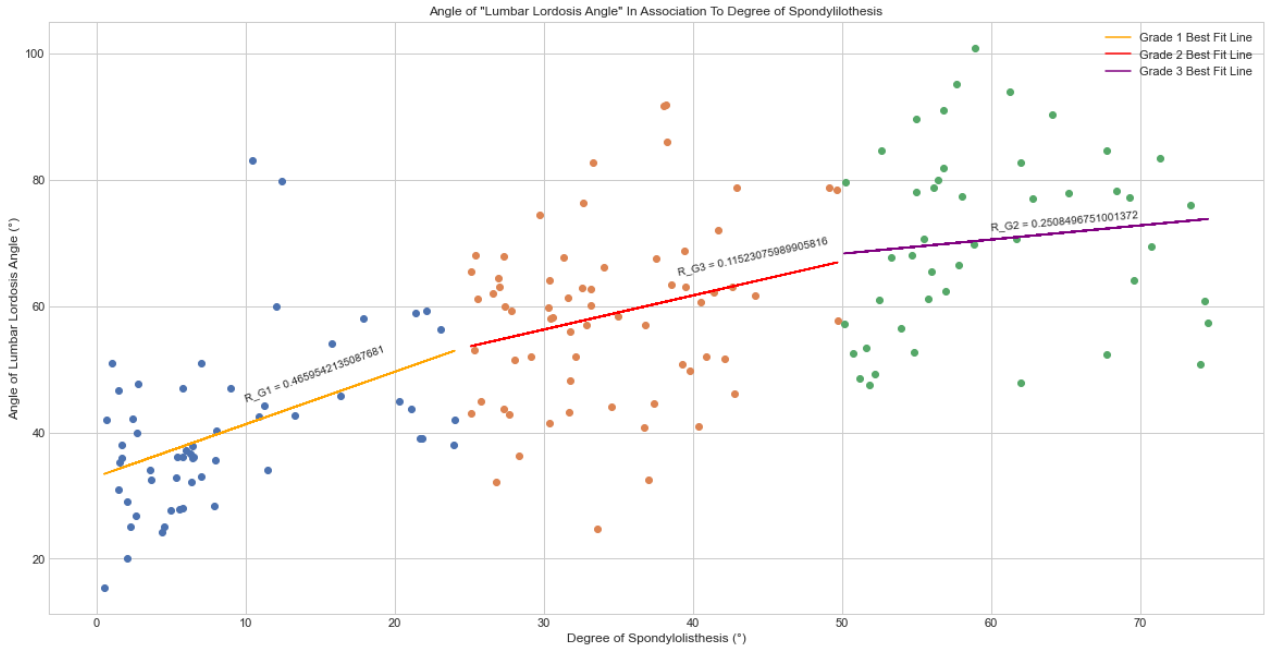


(2.3) Scoping Pelvic Parameters: Lumbarosis

In [770…

```
plt.figure(figsize = [20,10])

#grade 1: BEST FIT LINES
x1 = AB_G1['degree_spondylolisthesis']
y1 = AB_G1['lumbar_lordosis_angle']
```

```
a, b = np.polyfit(x1,y1, 1)
plt.scatter(x1, y1)

plt.plot(x1, a*x1+b, color = 'orange')

#grade 2: BEST FIT LINES
x2= AB_G2['degree_spondylolisthesis']
y2 =  AB_G2['lumbar_lordosis_angle']

a2, b2 = np.polyfit(x2,y2, 1)
plt.scatter(x2, y2)

plt.plot(x2, a2*x2+b2, color = 'red')

#grade 3: BEST FIT LINES
x3 = AB_G3['degree_spondylolisthesis']
y3 =  AB_G3['lumbar_lordosis_angle']

a3, b3 = np.polyfit(x3,y3, 1)
plt.scatter(x3, y3)

plt.plot(x3, a3*x3+b3, color = 'purple')

plt.title('Angle of "Lumbar Lordosis Angle" In Association To Degree of Spondylilothesi
plt.xlabel('Degree of Spondylolisthesis (°)')
plt.ylabel('Angle of Lumbar Lordosis Angle (°)')
plt.legend(['Grade 1 Best Fit Line','Grade 2 Best Fit Line','Grade 3 Best Fit Line'], f

#Adding correlation numbers on the best fit lines THROUGH 'TEXT ADDON' Within Plotly Gr
"""
By order: the correfs displayed are for G1, G2, G3,
positioned above each line via rotation diagonal (to align with each plotted line)
"""

plt.text(10,45, 'R_G1 = {}'.format(x1.corr(y1)), rotation=20, rotation_mode='anchor', f
plt.text(60,72, 'R_G2 = {}'.format(x2.corr(y2)), rotation=5, rotation_mode='anchor', fo
plt.text(39,65, 'R_G3 = {}'.format(x3.corr(y3)), rotation=12, rotation_mode='anchor', f

#Display Final Result
plt.show()
```



In [ ]:

In respect to these 6 visualizations, we should add the last correlation-centered variable that can seal the deal with our pattern findings from visual queues: **covariance** matrices' values. Covariance helps in statistical analysis as it is used to characterize relationships betweeen 2+ variables. They are derived from Gaussian distribution and are written as 'np.cov(one, two)' in Python to see the connection they share. Covariances tend to be displayed through heatmaps/scatterplots so in this case we are going in reverse -- as technically we are here to see the visuals as we already found our Pearson values.

Correlation focuses on the concept of "is there associaiton with such variables and how strong is the bonding." so Causality, also known as regression type, is moreso about how one variable affects/changes the other. Correlations are best when displayed in a best fit line with a Pearson value with +1 meaning the targeted variales are incredbly closes to each other. Regression types are portrayed by linkage through a more more mathetmatical stance, through equations. Regressions are to be discussed later on.

In [ ]:

(3) Retreiving Covariance from both parties

In [342…

```
#numpy has a command for this. While we can do 'sample' and 'population' through adding
#as our scope is every single row in the data labelled (as all are categorized as Norma

"""
eg. np.cov(norm['degree_spondylolisthesis'], norm['lumbar_lordosis_angle'], bias = Fals
    np.cov(norm['degree_spondylolisthesis'], norm['lumbar_lordosis_angle'], bias = True
"""
covariances_norm = []

covariances_norm.append(np.cov(norm['degree_spondylolisthesis'], norm['lumbar_lordosis_
```

```
covariances_norm.append(np.cov(norm['degree_spondylolisthesis'], norm['pelvic_incidence
covariances_norm.append(np.cov(norm['degree_spondylolisthesis'], norm['pelvic_tilt']))

print(covariances_norm)

covariances_abnorm = []

covariances_abnorm.append(np.cov(abnorm['degree_spondylolisthesis'], abnorm['lumbar_lor
covariances_abnorm.append(np.cov(abnorm['degree_spondylolisthesis'], abnorm['pelvic_inc
covariances_abnorm.append(np.cov(abnorm['degree_spondylolisthesis'], abnorm['pelvic_til

print(covariances_abnorm)
```

```
[array([[ 39.78433925,  20.33360547],
       [ 20.33360547, 152.80390785]]), array([[ 39.78433925,  15.62180039],
       [ 15.62180039, 152.97141855]]), array([[39.78433925,  7.7735962 ],
       [ 7.7735962 , 45.94810444]])]
[array([[1656.22468961,  397.36090176],
       [ 397.36090176,  386.88810537]]), array([[1656.22468961,  453.91566974],
       [ 453.91566974,  311.95081095]]), array([[1656.22468961,  137.03211728],
       [ 137.03211728,  110.58355017]])]
```

Looking at this array, we see that there are 2x2's. In a covariance matrix output, the off-diagonal elements contain the covariances of each pair of variables. The diagonal elements of the covariance matrix contain the variances of each variable. The variance measures how much the data are scattered about the mean. Covariances measures the ***direction of the relationship*** *between two variables. A positive covariance (>0) measures that both variabales tend to be high/low at the same time. A large variance indicates that numbers in the set are far from the mean and as whole far from each other. As reminder, variance in probability is a measure of how much values in the distbrtion vary on average with respect to the mean, computed via average squared differnce of each value from the 'expected' value. Often it is denoted as "cov(x,y) = E[(x-E[x]) . (y - E[y])]".*

On anlaysis for this array, we first check if indeed this is an accurate covariance matrix. Seeing how the top right and its diagonal counterpart on the bottom left are the same value, indeed it is symettrical and thus a proper covariance/ The off-diagonal values are the covariance values.

So how would you read these arrays? \ View the non-itereating value (1x) in that matrix

By order, the shared values that means it is the covariance will be:

- LLA Cov (Norm) = 20.33
- LLA Cov (AbNorm) = 397.36/100 --> 39.7
- PI Cov (Norm) = 15.62
- PI Cov (Abnorm) = 453.92/100 --> 45.4
- PT Cov (Norm) = 7.77
- PT Cov (Abnorm) = 137.03/100 --> 13.7

While we know now that there are extremely strong relations, as there are no zeroes and that they share same directional paths, analyzing the differnces, we learn that Pelvic Incidence has the greatest gap, meaning they share the largest range of differnece and as whole have the strongest covariance in respect to how it affects their spondylilothesis danger level.

In [ ]:

Issues with trying to interpret covariance results :

- One downside with having such huge numbers like what we have is that when comparing variances over datasets with different scales, like pound and inches, a 'weak' covariance in one dataset may be stronger in a differnet dataset with differenet scales. **Luckily for us, 'angle' in degrees (not radians!) is the shared scale we compare to for this pelvic parameter focus and thus we can safely omit such concern out of question.**

In addition, we learn another thing...the wide range of value is determined by the simple fact that the larger the X and Y values, the larger the covariance. A value of large numerical value tells us that the variables are correlated but, unlike the corelation coefficients thse two entities might share, not exactly how 'strong' that relationship is. This problem can be fixed by **dividing the covariance by the standard deviation** to get the correlation coeffficient. This would be useful if you would want to find correlation not from the pandas' .corr() but through mathematical equation usage after finding covariance from numpy's command: "corr(X,Y) = Cov(X,Y) / std_x*std_y".

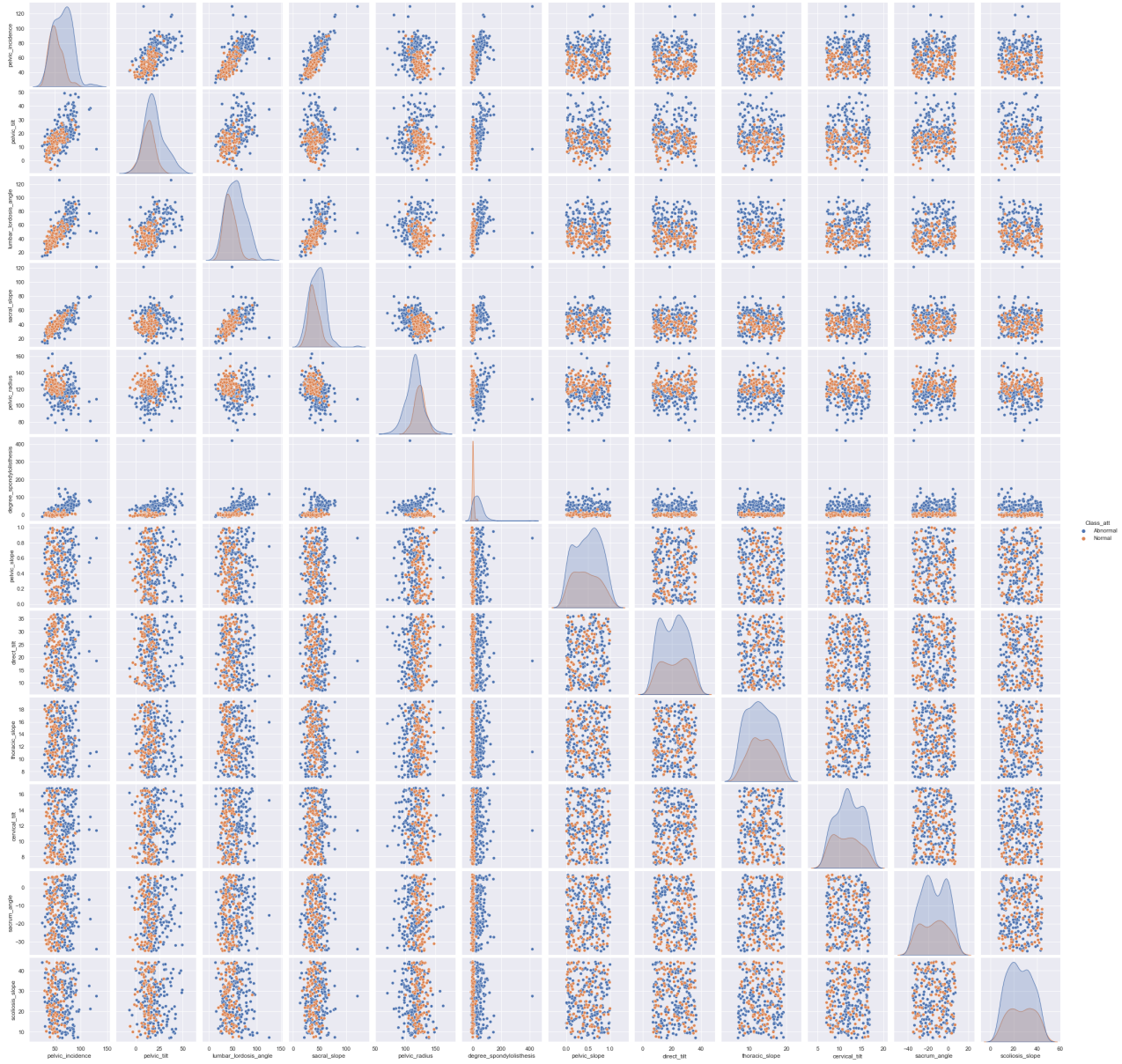
(4) Scoping Pelvic Parameters: Using Pairplots All Together

Pairplot is a graph that is used for conglomerate graphing. Let's add this in for the sake of it. This can help us gauge the average discrepenacies both parties have in contrast with each other.

Pair plots are a great method to identify trends for follow-up analysis and, fortunately, are easily implemented in Python. It helps to form some simple classification models by drawing some simple lines or make linear separation in our data-set. It is especially powerful to quickly see differences and similarities between multiple variables.

In [53]:
```
sns.pairplot(df, hue = 'Class_att')
```

Out[53]:
```
<seaborn.axisgrid.PairGrid at 0x293f614d580>
```



Here, we see a group of the variables all played out to graph the datapoints linked to each bivariate model. For sake of simplicity, I did not go in depth with only having the models for norm and abnorm dataframe. While a more technically efficient manuever would be to select norm and abnorm matrices for our arg in sns command executing, I choose to stick with the original dataframe because it is much easier to visually seperate the abnormal vs normal when a column conains both values -- something abnorm and norm does not as it used a filter to even be created to begin with. You cannot select 'Class_att' if I did the "technically more appropiate" method. The blue shades and data points represents the abnormal group while the other color is for the normal group's.

Spinal Discussion: Learning

Part One Reflection: Now what can we learn from this? The greater the slip grade, the greater/smaller the value of what variables? Positive or Negative Correlation?

Recap: *Classifying someone's stage of isthmic spondylolisthesis severity based on clincal measurements, can we determine its clincal significance (whether the parcticipant is worth needing*

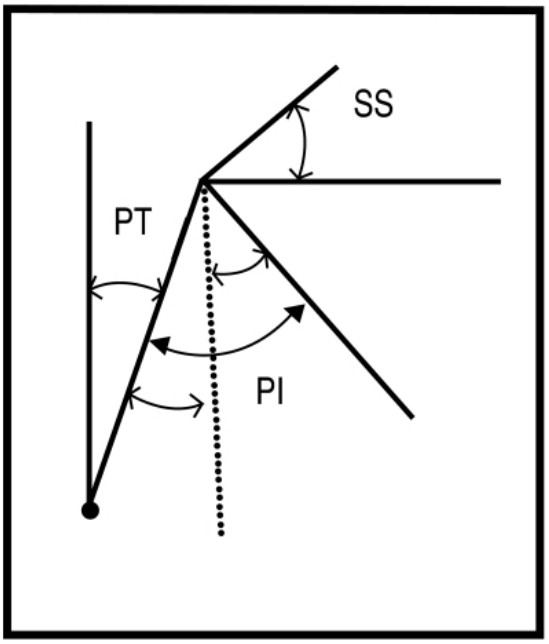
surgery treatment or not, mitigating financiala and hierarchical loss) and predict prior to surgery -- or even a doctor's visit one's recovery time, pain pinpoint in numerical value form, and more?

Analytical Summary Reflection:

Let's take a look at our past findings. What questions can be answered to clarify the influence all the pelvic parameters have on degree of spondylilothesis? We know that **finding the correlations (or even casuality) involved will aid us in understanding our goal of "finding the severity of one's lower back pain."**

In discovery of the pairplots, we can see that abnormal particpants tend to overtake normal particpant data points in pelvic incidence but other than that, they tend to be lower than normal particpants. Scoliosis is insane for how much they drastically differ. As can be seen on the sloped curves for all scoliosis comparisms, the abnormal ones inevitably always have a >45 degrees. Our main focus of interest I believe was very unique to see was degree of spondylilothesis and how the two groups (control and experimental) were at for their pelvic paramters. Looking at each stage, pelvic tilt and lumbar lordosis were averaged at Grade 2 with an 85% of R value on the abnormal section. A pattern can be seen in which the pelvic incidence in degrees is clumped very tightly on the lower end of the scale compared to the normal group for if looking at in comparism to pelvic radius. This can imply that the larger the radius, up to a specific dimensional value, pelvic tilt shifts accordingly in a radial action. We can derive that during lumbopelvic movement , some other non-diagnosed factors that probably influence these variables that showcase the tightness of one's lower back discs could be gussed as physiolgiical based. I would make the assumption, and then dive deeper into it on Part 2 for the symtompic anlaysis, the following vairables as potential contributors: age, hamstring muscle tightness, feet position, muscle fatigue, movement speed and external loading as well phase of motion can affect various aspects of lumbopelvic rhythm. Lumbopelvic movement is important because it is used to identify 'Pelvic cross syndrome' in gymnasts and competitive weightlifters. It is used to identify weakness, inflexibilities or muscles imbalance which contribute to low back pain. As the lumbo-pelvic-hip core complex consists of musculoskeletal structures that stabilize the spine and pelvis, fatigue may affect muscle recruitment, active muscle stiffness and trunk kinematics, compromising trunk stability. Thus, we can learn from the high pearson values and best plot fit lines graphed, alongside the fact that on the heatmaps pelvic incinde and lumbar lordosis angles get heavily proportional in value changing to degree of spondylilothesis, it is a fair assumption that grade level 2 spondylilothesis participant majorities are that most prone to variability and that in correlation to slip percentage, the ones that seem to have had the strongest linearity with back problem stemming was Lumbar Lordosis Angle of grade 1 abnormal participants with a whopping 0.4+ R value. What could this mean? Of course we can make generalized assumptions of what we can assume from the given statistics, but I feel it is safe to say that the greater the slip grade, the greater the degrees in all other pelvic parameters by graphs as well based on almost all having positive correlative linear lines of regressive states. Through this A/B testing determination, I have learnt a lot about how linked everything is to each other and what you can learn from even one varibale. High pelvic incidence can be a factor of L3 & L4 spondylolysis and it may have an influence on the slip progression in patients with L4 isthmic spondylolisthesis, but not on the slip progression in patients with L3 IS. Yet other factors seem to have an influence on the slip progression in patients with L4 isthmic spondylolisthesis. As the PI is increased, the PT and SS

also increased. Generally, the normal range of the PT is very narrow from 10 to 15 degrees, so if the PI increased, the SS will be increased relatively much more than the PT and it will be the cause of increasing lordosis -- the golden variable. If the lordosis is increased, there will be a large load to the posterior complex that will produce a spondylolysis and possible progression of spondylolisthesis. Progression of spondylolisthesis displaces the center of gravity of the pelvis anteriorly, which increases the PT and decreases the SS to compensate.



In [ ]:

Thoracic CSV Cleaning and Extrapolation

For this portion, our focus is centered around the following inquiry to be solved:

"For the elderly in specific, can we determine from pattern-analysis if one would survive their surgery, given one's age, weakness level before surgery , and performance status ? What are other symptoms that occur in associatio to those who died? Myoinfarction for example?"

In [ ]:

Variables to be looked at are:

1. Scoliosis Slope, Degree of Spondylilothesis;

2. Age at Surgery, Weakness Pre-Surgery (PRE11), Performance Activity Status (PRE6), Risk1Y (survival afer surgery? pattern?);

3. MI (myocardial infarction after lumbar spine surgery), Pain Before Surgery (PRE7), PAD (pre25), Smoking (Pre30), PRE17: Type 2 DM - diabetes mellitus (T,F), PRE25: PAD - peripheral arterial diseases, Volume Exhaled (shortness of breath / obesity) (T,F)

Reason of Interest:

I chose these variables in specifics because:

- for Part 1 of this dataset, we are comparing scoliosis development with lower grades of spondylilothesis via 'Scoliosis Slope' (SS) to Grades 0 and 1. Both of these variables are actually from the first dataset, yet performed in this section as the inquiry of interest differs slightly from the purpose of the first project section's purpose of analysis. As scoliosis delveops over time, it becomes a curiosity to see if this relates to spondolthessis due to poor back
- for Part 2, it is because as degenerative diseases of the spine is known as aging spine and has a likelihood of elderly getting spondylilothesis naturally due to erosion on the disks causing a gap in the spine, there are little evaluation tools offered for benefits or cases where elderlgy going throug surger actually make it out alive and for more than a year in thoracic surgery. There are huge dangers to doing surgey when aged. Especially considering weakness level prior to surgery, a potential clincal customer who got injured recently in their lumbar would probably want to undergo a reknown, quick predictor to see if it is safe enough to go to surgery considering their current physiological and health status. You want to know how bad is the sponthesiss.
- Lastly for Part 3, there are no exploratory data analyses (EDA) open-sourced online at all that genuinely covers both general spondylothesis analysis in respect to causes stemmed from both age AND from working out too much improperly. What if we see if complications for spondylthesis is influenced by any shared variable? A good variable to see is Myoinfarction (MI). With surgeries never stopping due to aging still being a thing that leads to spondylilothesis in addition to younger generational need from working out, MI can be one complication that is associate to surgeries. This can relate to mortality and other risk factors of which could tie into working out too. Since MI occurs from surgery and even if young, after surgery if you workout, it IS possible you can get a heart attack out of nowhere. You can get a heart attack from back surgery due to pre-operative factors however so it's interesting to see if danger from surgeyr is because of the surgery itself for young people or from other variables that makes surgery not the issue neccsariyly but more as an enabler/contributor. We can also see how maybe age isn't the only cause of danger surgery holds in its name for the elderlgy as well. As example for analysis, I would look at MI statistics for up to 6 months and see if it was a common/majority-proven variable occuring in almost all age ranges and seeing if bad MI value prior to deaths correlated or was maybe even a casuality to the morality rate. **I am curious to se how risk factors like smoking, diabetes-having, hypertension form stress or more cardial related symptoms (like peripheral arterial disease) affects the influence of MI which is known to lead to death? Does MI relate to sponylilothesis?**

In [ ]:

In [5]:
```
#Let's Run The File

raw_data_2 = "ThoracicSurgery.csv"
```

```
raw2 = pd.read_csv(raw_data_2)
raw2
```

Out[5]:

	id	DGN	PRE4	PRE5	PRE6	PRE7	PRE8	PRE9	PRE10	PRE11	PRE14	PRE17	PRE19	PRE25
0	1	DGN2	2.88	2.16	PRZ1	F	F	F	T	T	OC14	F	F	F
1	2	DGN3	3.40	1.88	PRZ0	F	F	F	F	F	OC12	F	F	F
2	3	DGN3	2.76	2.08	PRZ1	F	F	F	T	F	OC11	F	F	F
3	4	DGN3	3.68	3.04	PRZ0	F	F	F	F	F	OC11	F	F	F
4	5	DGN3	2.44	0.96	PRZ2	F	T	F	T	T	OC11	F	F	F
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
465	466	DGN2	3.88	2.12	PRZ1	F	F	F	T	F	OC13	F	F	F
466	467	DGN3	3.76	3.12	PRZ0	F	F	F	F	F	OC11	F	F	F
467	468	DGN3	3.04	2.08	PRZ1	F	F	F	T	F	OC13	F	F	F
468	469	DGN3	1.96	1.68	PRZ1	F	F	F	T	T	OC12	F	F	F
469	470	DGN3	4.72	3.56	PRZ0	F	F	F	F	F	OC12	F	F	F

470 rows × 18 columns

This time, we wil be heading to a more categorical based datasheet. The abbreviations are of the following:

* PRE5: Volume that has been exhaled at the end of the first second of forced expiration - FEV1 (numeric)

* PRE6: Performance Status - Zubrod scale (PRZ2,PRZ1,PRZ0) (=Hierarchy: Z0 - Asymptomiatic fully active. Z1 = Symptomatic but completely ambulatory (fine), Z2- sympomatic / not feeling well)

* PRE7: Pain before surgery (T,F)

* PRE11: Weakness before surgery (T,F)

* AGE: Age at surgery (numeric)

* Risk1Y: 1 year survival period - True value if died (T,F)

* PRE30: Smoking (T/F)

* PRE30: Myocardial Infarction Post-Surgery Status (T/F)

* PRE17: Type 2 DM - diabetes mellitus (T,F)

* PRE19: MI up to 6 months (T,F)

* PRE25: PAD - peripheral arterial diseases (T,F)

In [6]:
```
#change column names to make it more readable (using df.replace({orig:new_colname, orig
raw2.rename({'PRE5':'Volume_Exhaled', 'PRE6':'Performance_Status','PRE7':'Pain_PreSurg'

#drop the undesired columns with .drop('col_or_row', axis = 0/1) and declare it because
raw2 = raw2.drop(['DGN','PRE4','PRE8','PRE9','PRE10','PRE14','PRE32'],axis=1)
raw2

#(*If error is given, just re-run the raw2 data load line above and try to run this cel
```

Out[6]:

	id	Volume_Exhaled	Performance_Status	Pain_PreSurg	Weakness_PreSurg	Diabetes Mellitus	MI	PAD	S
0	1	2.16	PRZ1	F	T	F	F	F	
1	2	1.88	PRZ0	F	F	F	F	F	
2	3	2.08	PRZ1	F	F	F	F	F	
3	4	3.04	PRZ0	F	F	F	F	F	
4	5	0.96	PRZ2	F	T	F	F	F	
...	...	...	...	...	...	...	...	...	
465	466	2.12	PRZ1	F	F	F	F	F	
466	467	3.12	PRZ0	F	F	F	F	F	
467	468	2.08	PRZ1	F	F	F	F	F	
468	469	1.68	PRZ1	F	T	F	F	F	
469	470	3.56	PRZ0	F	F	F	F	F	

470 rows × 11 columns

In [8]:
```
#rename the df to make it more appropiate
surg = raw2.copy()
surg
```

Out[8]:

	id	Volume_Exhaled	Performance_Status	Pain_PreSurg	Weakness_PreSurg	Diabetes Mellitus	MI	PAD	S
0	1	2.16	PRZ1	F	T	F	F	F	
1	2	1.88	PRZ0	F	F	F	F	F	
2	3	2.08	PRZ1	F	F	F	F	F	
3	4	3.04	PRZ0	F	F	F	F	F	
4	5	0.96	PRZ2	F	T	F	F	F	
...	...	...	...	...	...	...	...	...	
465	466	2.12	PRZ1	F	F	F	F	F	
466	467	3.12	PRZ0	F	F	F	F	F	
467	468	2.08	PRZ1	F	F	F	F	F	

	id	Volume_Exhaled	Performance_Status	Pain_PreSurg	Weakness_PreSurg	Diabetes Mellitus	MI	PAD	S
468	469	1.68	PRZ1	F	T	F	F	F	
469	470	3.56	PRZ0	F	F	F	F	F	

470 rows × 11 columns

In [ ]:

For each age range from infant to 100 years old, ideally I want to do comparative functions to run. To ease this process however, creating class would allow a more organized and efficient manner. Thus, there will be a class called AgeGroup that has class methods for each function I want to run to best get correlation results as way of understanding how associated scoliosis and other semi-related variables work with generation of spondililothesis (or vice versa)! The plan is to execute instances and showcase a conjoint stacked bar chart, combinatorial scatter chart, and a df.corr() like in the last section. The difference is less so of the methods done to achieve analytical responses but moreso of what we are looking at now.

In [ ]:

Identifying Trends & Relations

Things to watch for:

- Counfounding variables (variable biases)

Graphs in a class for age group simplicity

- overlay stacked bar plot (2 variables: unique each time (per stack))
  - Can be done using .plot's keyword arg (also known as a *kwarg*) "zorder"
  - Alternatively can be done with a matrix scatter plot or a profile plot (better solution as matrix scatter is boringly similar to pairplots)
- _ *str* _ printing a statement
- regression fit (with 'LinearRegressio().fit(x,y))
  - Needs 'from sklearn.linear_model import LinearRegression' (a good showcase of various ways one can find linear regression parameters)
  - Slope can be from "str(model.coef_)"
  - R can be from "str(model.score(x,y))"
- chance of scoliosis

Reflection Analysis:

- Correlation? (use "original hypothesis vs actual findings differences in value?"
  - *Reminder: even if a positive correlation but if small R, then no corr can be accurately reported*

In [ ]:

Modify *Surg* column values

Purpose: from categorical to numerical for classifying

Machine Learning models behave best if quantiative data is the input. Let's alleviate difficulty. This also helps for graphs as it can be tough sometimes trying to correlate variables when one is categorical while the other(s) have more quantiative data.

In [9]:
```
surg['Pain_PreSurg'] = surg['Pain_PreSurg'].replace(['T'], '1')
surg['Pain_PreSurg'] = surg['Pain_PreSurg'].replace(['F'], '0')

surg['Weakness_PreSurg'] = surg['Weakness_PreSurg'].replace(['T'], '1')
surg['Weakness_PreSurg'] = surg['Weakness_PreSurg'].replace(['F'], '0')

surg['Weakness_PreSurg'] = surg['Weakness_PreSurg'].replace(['T'], '1')
surg['Weakness_PreSurg'] = surg['Weakness_PreSurg'].replace(['F'], '0')

surg['Diabetes Mellitus'] = surg['Diabetes Mellitus'].replace(['T'], '1')
surg['Diabetes Mellitus'] = surg['Diabetes Mellitus'].replace(['F'], '0')

surg['PAD'] = surg['PAD'].replace(['T'], '1')
surg['PAD'] = surg['PAD'].replace(['F'], '0')

surg['Smoking'] = surg['Smoking'].replace(['T'], '1')
surg['Smoking'] = surg['Smoking'].replace(['F'], '0')


surg['Performance_Status'] = surg['Performance_Status'].replace(['PRZ0'], '0')
surg['Performance_Status'] = surg['Performance_Status'].replace(['PRZ1'], '1')
surg['Performance_Status'] = surg['Performance_Status'].replace(['PRZ2'], '2')

surg
```

Out[9]:

	id	Volume_Exhaled	Performance_Status	Pain_PreSurg	Weakness_PreSurg	Diabetes Mellitus	MI	PAD	S
0	1	2.16	1	0	1	0	F	0	
1	2	1.88	0	0	0	0	F	0	
2	3	2.08	1	0	0	0	F	0	
3	4	3.04	0	0	0	0	F	0	
4	5	0.96	2	0	1	0	F	0	
...	...	...	...	...	...	...	...	...	
465	466	2.12	1	0	0	0	F	0	
466	467	3.12	0	0	0	0	F	0	
467	468	2.08	1	0	0	0	F	0	
468	469	1.68	1	0	1	0	F	0	

	id	Volume_Exhaled	Performance_Status	Pain_PreSurg	Weakness_PreSurg	Diabetes Mellitus	MI	PAD	S
469	470	3.56	0	0	0	0	F	0	

470 rows × 11 columns

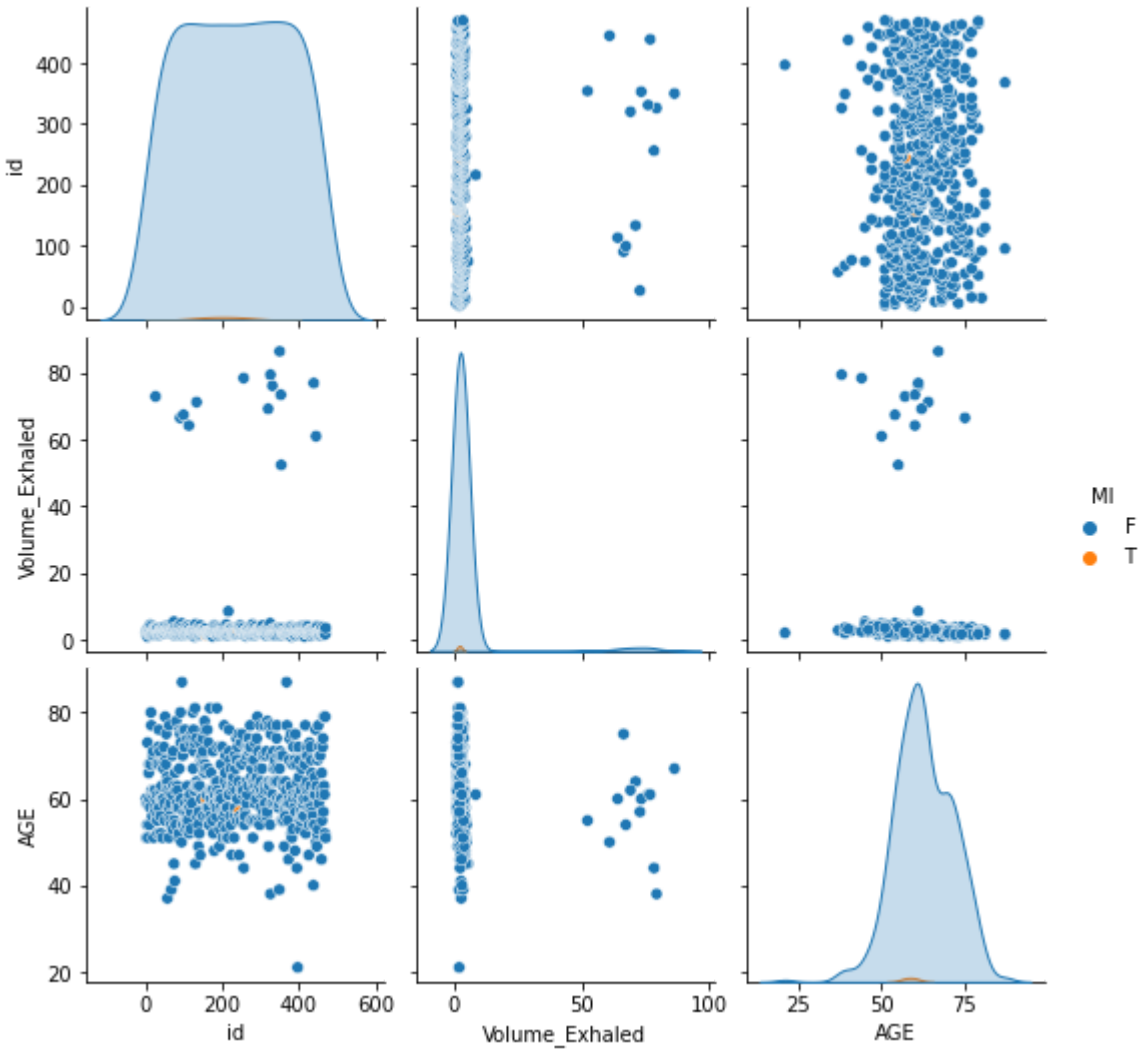
In [ ]:

Identifying Relations Graphically

In [10]:

```
from sklearn.linear_model import LinearRegression

class AgeGroup:
    #For this section, our target age range goes from 20 to 90, stratified in ranges of

    def __init__(self, range_list):
        self.listt = range_list
        self.min = range_list[0]
        self.max = range_list[1]

    def __str__(self):
        return f'The age range for this group is {self.min} to {self.max}'

    def regression(self, x,y):
        """ Create sklearn regression models NOT like before. This time we will use a v
            using an ML library to execute commands that will find the coefficient (Pea
            the slope of the best fit line carved out.

            eg. regression(col1, col2)

            I would use this if I need to. An example would be the R-value with the slo
            As we are going to see regression
            anyways on ML predictor section, I do not need to call it,
            however if desired, the user can always use this class method.
        """
        model = LinearRegression().fit(x,y)
        coef = str(model.coef_)
        slope = str(model.score(x,y))
        print(f'The Pearson (R) Coefficient is {coef} with a slope of {slope}')


    def plot_myocardial_infarction_relation (self):
        """ Calculate the likelihood someone gaining myocardial_infarction over the cou
            pre-opereative variables and death frequency to see if any variable relates
            all of them out as multivariate uniquely colored line graphs.

            Variables: Age, Weakness, Performance ratio (type), morbid rate (died or su
                Contributors: PAD, Smoking

            Target: MI (got it or not)

            For overlap: you can stack with keyword 'hue' on histplots
        """
        sns.pairplot(surg, hue = 'MI')
```

In [13]:
```
l = [50,60]
a = AgeGroup(l)
a.__str__()
a.plot_myocardial_infarction_relation()
```



In [ ]:

Difficulties with Pandas:

- A downside I have found during this project is that when extrapolating and exporting a csv file (or any data entry file to begin with), sometimes the values do not come the most correctly. For example, after my 13th look at my datasets to see if my corrrelations per dataset was accurate -- mirroring the actual xlsx file, I noticed some columns were not the most numerically correct. As example, the PRE7 (Pain_PreSurg) often had some 'True' values here or there yet never was able to have it revealed as the frequency of succh value was so small that it was lost in graphs. While not wrong, I do indeed find it very unfortunate and saddening to see that perhaps this is very commmon. If so, it is sad to see that important values can most often be dismissed due to

the minisucular regularity, resulting in potential significant graphing to know correct pearson values visually. While R may be correct in cases that uses lines telling the system to go over the whole dataset, it fails to account some numbers because they are so rare, sometimes the system even treats it as 0 or non-existint.

In [ ]:

Symptoms Discussion: Learning

Recap: Does lower-grade spondolthesis affect development of idiopathic/uknown-causing scoliosis due to poor lower back strength leading to possible weak spine and as result flimsy posture? For elederly in specific, can we determine (from pattern-analysis on this dataset) IF one would gains myocardial infarction, based on one's given age , weakness level before surgery, and performance status (bedridden or not --- PRZ0 >> PRZ1 >> PRZ2 is bad) ? What are other symptoms that occur in associatio to those who died?

I realized the lack of awareness people, including myself, tend to skip and feel they get confused on being so stuck on during my time making this project. You cannot try to find trends of relations on variables that are not in the same dataset. As close as it may be in terms of topic of interest and more, if not from the same dataset, this is considered 'reaching' a hypothesis to be 'tested' and as whole not credible for whatever result it may output. As such, I decided to opt out on finding if there is casuality in scoliosis and rather focus on the more physiollogically-related and same-dataset-found variable of Myocardial Infarction. Here, we learnt that not only did my hypothesis of all relating in proprtion to be correct, but that interestingly risk ratios were at an all time high with over a whopping perentage being patients to gain MI within 30 days of surgery. This was a bit suprising to an extent, as I did not know how closely connected MI and surgery operation was.

Is it casuality? For casuation, even after all these findings, it cannot be found numerically in statistics. To genuinely find causation, this is more of a 'why is it hapenning' rather than 'seeing' how it happens. So the only way to prove if MI is a casuality variable to mortality rate or even degree of spondylilothesis is solely through semantics: saying 'it is thus pretty likely to be a cause, based on what was found from correlations seen above' -- that is it.

Analytical Summary Reflection:

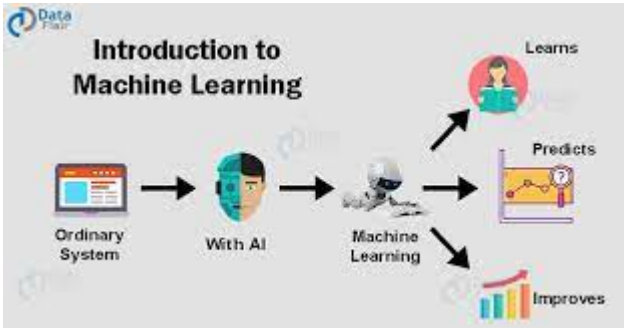
Spinal surgery for fixation and fusion in patients older than 75 years old actually represents a feasible therapeutical option, granting good results in terms of quality of life. Reduction of surgical time represents a crucial point to reach better results and to reduce the co-morbidity due to anaesthesia and blood loss. However, due to general morbidity as well as the rate of fusion, the choice of a surgical treatment for spinal fixation in elderly patients must depend on the balance of risks and benefits in the treatment itself.

You can get a heart attack from back surgery due to pre-operative factors however so it's interesting to see if the concept of 'danger from surgery' is because of the surgery itself for young people or from other variables that makes surgery not the issue neccasarily -- but more as an

enabler/contributor. We can also see how maybe age isn't the only cause of danger surgery holds in its name for the elderly as well bsaed on these correlations. What would be cool in the future would be seeing for fitness goers, how such symptom column variables looked at relates to them and in what way could some algorithmn assit in reducing incorrect exercises that could injure a gym goer. That is where this project's Part 3 would be on -- computer vision embedding (less of an EDA purpose but more of an action suggestion motif, inspired by the findings of trends found here in Parts 1 & 2 that are physiological based).

For my last portion of the project, I will be using machine learning algorithmns to generate a model for predicting key variables from my two parts. Correlation is one thing but to know truly how influential a variable is would be through classification grouping through autonomous methods like through AI. Machine Learning is the pinnacle classic of what data science represents. Thus, let's finish off this project with peak analysis using K-NN -- a simple yet powerful type. The main areas of focus will be on severity of spondolilothesis (from normal to abnormal) and chances of mortality post-surgery (from most likely will die to not going to die).

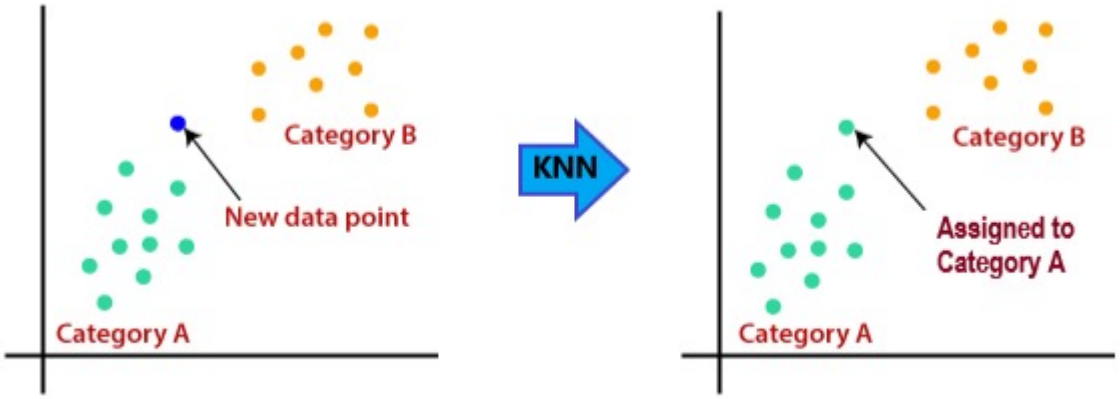
In [ ]:



Embedding Machine Learning Models: What is the K-NN Algorithmn?

The last step however is through what we call **Cross Validation**. Cross-Validation is a statistical method of evaluating and comparing learning algorithms by dividing data into two segments: one used to learn or train a model and the other used to validate the model. Extremely common for predicting, it's a technique used to protect against overfitting in a predictive model, particularly in a case where the amount of data may be limited. In cross-validation, you make a fixed number of folds (or partitions) of the data, run the analysis on each fold, and then average the overall error estimate.

This is most commonly done through implementation of Machine Learning. Luckily, Python is the perfect language for this. We can go for numerous model types, but the easiest to learn and embed is K-NN, also known as K-Nearest Neighbours for prediction.



The k-nearest neighbors (KNN) algorithm is a simple, supervised machine learning algorithm that can be used to solve both classification and regression problems. It can achieve high accuracy in a wide variety of prediction problems. Essentially, the steps are fairly simple.

1. Choosing a numerical value ('k') will help determine your dimensions. This will be done through initalizing two main iterables. 'X' will be your "trained" dataset and "y" as your "test" data.

In machine learning, this is golden rule you follow by. You essentially want to use trained data to find analyses and to use test data, randomly extrapolated rows from your original data, to practice preidction on it to print output of all results found in a 'classification report.'

To perform actions inbetween the steps of train to test data displaying, you would undergo a specific set of actions determined by your model type.

For the K-NN algorithmns, there is a "majority-vote" technique . **Essentially a datapoint is to check 'k' amount of closeby points to see which side has moere points near it. It 'groups' up to that category and gets counted to the 'percentage of datapoints in category. This will be within a 'train' set of datapoints, randomly pulled in from your original database.** Your trained vs test dataset is split into a "testsize" number via a library command called ".train_test_split" which must be done in order to perform any ML algorithmn.

For example, if you have a testsize = 0.33, this means you will seperate your dataset to be 70% randomly chosen as datapoints for your training set and the remaining 30% for test data. More information on the documentation of this command can be found here: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

Training the model is performing arithmetic or statistical operations and test accuracy is how well the model does with test data, once all trained data is completed. A tutorial is found here: https://www.youtube.com/watch?v=s-9Qqpv2hTY.

For K-NN in specifics, you calculate distances between one point and its *k* nearest (= Cartesian coordinatedly closest) neighbors. You would then apply that to majority and repeat. You'd set this in a "knn specific classifier" object as an instnace of class KNN.

https://www.mdpi.com/mathematics/mathematics-08-00286/article_deploy/html/images/mathematics-08-00286-g001-550.jpg

In [ ]:

K-Nearest Neighbours (K-NN) Predictor: Determine Spondylilothesis Severity

*Jumping Into **Machine Learning Prediction** :*

For this project, I want to predict two things. For this dataset, my goal is to **predict the severity of one's lower back health determined by pelvic parameters --** Abnormal meaning a participant is in a detrimentally high level of danger (proneness to require surgery) while being safe will be categorized as Normal.

Signficance of Prediction: As a reminder, if you do not care about the asymptomatic signs while at an abnormal grade level for spondylilothesis, you can be very likely to have your back forever injured if not pursuing surgery and this can be life changing in a negative way -- leaving you in a wheelchair or even to some rare occasions dead. Normality of dorsal health is crucial for not only general surgery-inquiries or general health self-evaluation, but also with knowledge of one's lumbar erosive-eseque features, as you can then determine what your solution procedure should be to get rid of whatever your pain may be.

In [694…

```
import scipy
"""for more optimization and added significant power for manipualting
#and visualizing data"""

from pylab import rcParams
#for adjusting plot parameters

import urllib
#using urls
import sklearn
#use this library to perform all model actions

from sklearn import neighbors
#we will be using KNN from sklearn's neighbour category

from sklearn.neighbors import KNeighborsClassifier
from sklearn import preprocessing

from sklearn.model_selection import train_test_split
#model_selection is just like sklearn.cross_validation (the main command)

from sklearn import metrics
```

Set our plotting parameters for Jupyter Notebook

In [695…

```
np.set_printoptions(precision = 4, suppress = True)
%matplotlib inline
rcParams['figure.figsize'] = 7,4
plt.style.use('seaborn-whitegrid')
```

In [933…

```
#Adjust to make last categorical colulmn into quantiative
surg['MI'] = surg['MI'].replace(['T'], '1')
surg['MI'] = surg['MI'].replace(['F'], '0')
```

Step 1: Splitting raw data into test and training datasets

In [697…

```
""" We want to use the filtered data that only looks at the variables we want -- aka th
    We are going to build a machine learning model that predicts disk slippage angle se
"""
#distinguishable subgroups of which we want analyzed (we must restate what each pelvic
desired_cols = ['pelvic_incidence', 'pelvic_tilt', 'lumbar_lordosis_angle', 'scoliosis_

#SUBSET X_prime will be an indexed selection of columns we know will influence the clas
#The values are the cell values under each of the aforementionned columns.
X_prime = filtered.loc[:, desired_cols].values

# Our SUBSET 'target variable' Y would be class_att (the T/F) determinant
y = filtered.iloc[:,-1].values

# An alternative would have been 'y= filtered.loc[:,['Class_att']].values'
# Both versions of 'y' target variable (what we want to see outputed as result from a p
```

With this information, we now will create a scaled dataset. Using scikit learn's preprocessing tools we will use a scale function, proceeded by throwing/passing in our x-prime object. This is the columns we will say we will use and we want to scale our input variables. This will be needed to split into 2 sets.

We must do this to perform K-NN algorithmns.

In [698…

```
X = preprocessing.scale(X_prime)
```

As stated, now we then split data into test and training sets. We use training sets to train the model & test sets to test, while waiting, the model's performance. To do this, must use sckit seleciton tools using a "train-set-split function" which breaks datasets into train-test splits. Both X and y input and ouputs should be sparsed.

STEP 2. Verify Shape and Split the Data Into Train-Test Subsets

2a) Verify two tests are the same in shape. Having non-similar shapes will make models untrainable as there needs to be a shared amount of matchmaking val-to-val connection when trying to have a model 'learn' what each variable relates to.

In [699…

```
"""To ensure train_test_split does not run into problems (like input vars having inconc
must reshape.

This would be done via REMOVING extra dimension/list inside X variable
and TRANSPOSING it to get equal number of samples in X & y.

You must have same row number because you want to be able to have the model
capable of linking/associating one element from one subset to the other at the same ind
```

```
You can only do this if the row of Y is matching the same length as
the column of X.
"""

#The shape should end up with y being the same row amount as X
shape_X = X.shape
shape_y = y.shape

print(shape_X, shape_y)
```

```
(310, 5) (310,)
```

*As can be seen, the two input data iterables match in shape without need to have to transpose and reshape! Great!

The output should display here specifically: (310,5) (310,)

2b. Now we will split the data into train and test datasets

In [700…

```
#Must have this seed to be passed in because this funciton splits data randomly
"""
params:

X and y = our datasets that are to be split, which is determined
by the percentage of how much we want trained and how much is used
for testing if our model is succesful in prediction.

test_size = Setting 'test_size' is important because it tells what percentage
of your data is being split into the training and rest in test phase.
This is important because you want to put in a lot of training data
so that it can learn how to identify unknown new data -- the
test data.

Ideally 33% is pretty solid and by inputting 0.33 for this arg is common
as you get 77% of the input data as your training data for the machine to learn.
This is more than enough and thus the remaining 33% of data should be a viable
amount of data for the machine to practice on.

random_state = since the function splits data randomly, we must
pass this argument in

"""
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size = 0.33, random_state
```

Definition: Train_test_split splits arrays or matrices into random train and test subsets, returning lists of length = "2*len(arrays)."

STEP 3: Building and training your model with training data

We can now do the actual training for our model. It's like having a dog. You train it to do actions and treat it with treats to encourage positive behavior. Once you feel the dog has mastered its tricks, you put it in the real world by performing calls to have it try to do such actions. The calling method is similar to how your model will use your test data, which is based on after your model has fully 'learnt' your training data well enough to make such actions/predictions.

3a. Here, let's create a K-NN object. We will make a classifier instance of KNN and will fit it through

our trained data.

In [713…

```
#call a clf and set that equal to a KNN classifier
#passing in a training data, we will use this for our train data
clf = neighbors.KNeighborsClassifier()
clf.fit(X_train, y_train)
print(clf)
```

```
KNeighborsClassifier()
```

STEP 4: Evaluating model's predictions against the test dataset

For the last section, we want to do the actual performance of what our model was for -- predicting.

We can rename our y_test with our 'expected outcomes' (y_expect). We wil then create a y_prediction of what our model will predict, passing in our test dataset to be used for practice/actual performance.

Using a scki-kit classication model from 'metrics.', we will print out our expeted and prediciton model results -- our purposed predictor. This is the report that will determine everything (~ a very exagerated tone used here*).

Understanding Classification Reports

The reported averages include macro average (averaging the unweighted mean per label), weighted average (averaging the support-weighted mean per label), and sample average (only for multilabel classification). Micro average (averaging the total true positives, false negatives and false positives) is only shown for multi-label or multi-class with a subset of classes, because it corresponds to accuracy otherwise and would be the same for all metrics.

An output of a ML algorithmn can be mapped by TP/TN/FP/FN. These stand for True Positive, True Negative, False Positive, False Negative . We always desire for TP and TN but due to misclassifications, we sometimes end up in FP and FN. This is why we use a confusion matrix prior to a classification report. From the (y_pred)predicted label and the true outcome (y_expected), we know "Accuracy = TN + TP / TN + FP + TP + FN." With accuracy representing the number of correctly classified data instances over the total number of data instances, we learn that accuracy may not always be the best measure of a dataset is not balanced. Thus, this is why we have precision to be found in a classification report. "Precision = TP/TP+FP." Precision should be ideally high for a good classifier which means you can use accuracy as a good measure of knowing how accurate your results are. Recall is more like for sensitivity or true positive rate, defined as "Recall = TP / TP + FN." Recall should also be 1 to be a good classifier.

In the end result, F1 should be what we look at. It is better than accuracy because it is a harmonic mean of precision and recall and only becomes 1 when both of its dependant variables are of value of 1. As a reminder, you want accuracy but in a balanced set.

PARAMS:

Recall is a measure of your model's completeness. "*What percent of the positive cases did you catch?*"

"High precision with corresponding low recall" means that few results were found/returned, but of those, many of the label predictions returned were very correct. Essentially 'high accuracy, but low completion' .

Precision is the ability of a classifier not to label an instance positive that is actually negative. For each class, it is the ratio of true positives to the sum of a true positive and false positive (TP/(TP+FP)). In summary, "*what percent of your predictions were correct?*"

F1 score is "*what percent of POSITIVE predictions were correct?*". The weighted harmonic mean of precision and recall such that "best score = 1.0" and "worst score = 0.0." The weighted average of F1 should be used usually to *compare* classifier models but not global accuracy. The mathematical formula is "F1 Score = 2*(Recall* Precision)/(Recall + Precision)."

Support is the number of actual occurences of the class in the specified dataset. 'Imbalanced' support in the training data may indicate structural weakness in the repeated scores of the classifier. Support doesn't change between models but instead 'diagnoses the evaluation process.'

*Support honestly is the least useful of the four variables given**

My Classification Report

In [714…

```
y_expect = y_test
y_pred = clf.predict(X_test)

print(metrics.classification_report(y_expect, y_pred))
```

```
              precision    recall  f1-score   support

           F       0.84      0.99      0.91       130
           T       0.67      0.08      0.14        26

    accuracy                           0.84       156
   macro avg       0.75      0.53      0.52       156
weighted avg       0.81      0.84      0.78       156
```

Sklearn.mtrics **"Classification Report builds a text report showing the main classification metrics.** The report is either a string or dictionary (or json), of which is a text summary of the precision, recall, F1 score for each class. A dictionary is returned if output_dict is True. In dictionary format, our output would look like

"{'Abnormal': {'precision':0.79, 'recall':0.88, 'f1-score':0.83, 'support':67}, 'Normal': {...}}"

Reading our model classification report:

We are particularly interested in the 'Normal' row. Recall is how many of the observed cases were predicted positive. It turns out for the "Normal" classification from the predictor being only 56%, this means it is not great. **We see that Normal has a 63% F1-Score meaning the chances of an invidual in this dataset to be 'normal' or to be healthy and not have to go to surgery be not as likely compared to its counterpart 83% category 'Abrnomal.'**

In [ ]:

Contributor Metrics: Confusion Matrix & Accuracy Scoring

In [718…
```
from sklearn.metrics import accuracy_score
print('Accuracy score: ', accuracy_score(y_test, y_pred))
```

```
Accuracy score:  0.8397435897435898
```

In [715…
```
from sklearn.metrics import confusion_matrix
print(y_test.shape, y_pred.shape)
confusion_matrix(y_test, y_pred)
```

Out[715…
```
(156,) (156,)
array([[129,   1],
       [ 24,   2]], dtype=int64)
```

A confusion matrix is good for encapsulating data. It is an N x N matrix used for evaluating the performance of a classification model, where N is the number of target classes. The matrix compares the actual target values with those predicted by the machine learning model. The observed values become the predicted values. Observed 0's are top left, observed 1's are bottom left. That is where we got our 0.77 acccuracy score. Top right tells us #_of_false_positives and bottom left is #_of_false_negatives.

Out of our 36 observed positive cases (*added via sum of bottom row), this matrix tells us we only identified correctly 20 cases.

Out of the 28 predicted-to-be-positive cases (*added via rightmost column), only 20 were actually positives.

It is important to learn confusion matrix in order to comprehend other classification metrics such as precision and recall. Confusion matrix goes deeper than classification accuracy by showing the correct and incorrect (i.e. true or false) predictions on each class.

Why do we need confusion matrices rather than relying on just an accuracy score? Accuracy and error rate are the de facto standard metrics for summarizing the performance of classification models. Classification accuracy fails on classification problems with a skewed class distribution because of the intuitions developed by practitioners on datasets with an equal class distribution.

In the end run however, I still was curious at the accuracy because companies use machine learning models to make practical business decisions, and more accurate model outcomes result in better decisions. The cost of errors can be huge, but optimizing model accuracy mitigates that cost. While different purposed here, the motto is the same. Accuracy represents the number of correctly classified data instances over the total number of data instances and the benefit is that you can know already that this variable alone is not going to determine your knowledge of how precise the model was BUT you can use this *along with other variables* to find the true accuracy.

In [ ]:

K-Nearest Neighbours (K-NN) Predictor: Determine Post-Surgery Mortality Possibility Based On Various Features

For this project, I want to predict two things. For this dataset, my goal is to **predict if any random invidual of a dummy age can be classified as probably going to die after undergoing surgery.**

Significance to Predict: By knowing this, a patient can know if they are willing to go the extra mile and sacrifice chances they have of survival for a better life -- if successful even (which can be countering the odds through viewing this kind of prediction algorithmn). This would help patietns understand the amount of risk that they would be putting themselves into but also help clinicans in knowing if operation should actually occur for such invidual or not -- considering the odds and the symptoms found via correlations from prior.

As all steps for ML algorithmns for K-NN Classifiers has already been explained in the previous part, we will just jump into the code.

In [446…

```
import scipy
"""for more optimization and added significant power for manipualting
#and visualizing data"""

from pylab import rcParams
#for adjusting plot parameters

import urllib
#using urls
import sklearn
#use this library to perform all model actions

from sklearn import neighbors
#we will be using KNN from sklearn's neighbour category

from sklearn.neighbors import KNeighborsClassifier
from sklearn import preprocessing

from sklearn.model_selection import train_test_split
#model_selection is just like sklearn.cross_validation (the main command)

from sklearn import metrics
```

In [447…

```
np.set_printoptions(precision = 4, suppress = True)
%matplotlib inline
rcParams['figure.figsize'] = 7,4
plt.style.use('seaborn-whitegrid')
```

In [706…

```
"""
This section, we are analuzing RISK of DEATH after one year of surgery.
That would be our target variable of interest. All the other columns are
correlated (already found prior to this ML predictor step) and are the treated
X-variable indicating these variables influence Y.

We need to convert T/F for any non-Y columns to be 1/0 values.
Change up for "Performance_Status" to be 0/1/2 for PRZ0/PRZ1/PRZ2,
Pain_PreSurg, Weakness_PreSurg, Diabetes Mellitus, MI, and PAD
```

```
"""

#distinguishable subgroups of which we want analyzed (we must restate what each pelvic
desired_cols = ['Volume_Exhaled','Performance_Status','Pain_PreSurg','Weakness_PreSurg'

#SUBSET X_prime will be an indexed selection of columns we know will influence the clas
#The values are the cell values under each of the aforementionned columns.
X_prime = surg.loc[:, desired_cols].values

#our SUBSET 'target variable' Y would be class_att (the T/F) determinant
#use iloc to find filtered columns through index finding rather than actual column name
y = surg.iloc[:,-1].values
```

In [707…
```
X = preprocessing.scale(X_prime)
```

In [708…
```
#check shape for if can be trained and tested
print(X.shape, y.shape)
```

```
(470, 8) (470,)
```

Sweet, they are same in size! Remember, if not the same size, you cannot execute a train_test bidataset split. You have to be able to match 1 cell value with another from the second iterable.

In [709…
```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size = 0.33, random_state
```

In [710…
```
clf = neighbors.KNeighborsClassifier()
clf.fit(X_train, y_train)
print(clf)
```

```
KNeighborsClassifier()
```

In [711…
```
y_expect = y_test
y_pred = clf.predict(X_test)

print(metrics.classification_report(y_expect, y_pred))
```

```
              precision    recall  f1-score   support

           F       0.84      0.99      0.91       130
           T       0.67      0.08      0.14        26

    accuracy                           0.84       156
   macro avg       0.75      0.53      0.52       156
weighted avg       0.81      0.84      0.78       156
```

In [712…
```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test, y_pred)
```

Out[712…
```
array([[129,   1],
       [ 24,   2]], dtype=int64)
```

In [472…
```
from sklearn.metrics import accuracy_score
print('Accuracy score: ', accuracy_score(y_test, y_pred))
```

```
Accuracy score:  0.8397435897435898
```

In [473…
```
y_expect = y_test
y_pred = clf.predict(X_test)

print(metrics.classification_report(y_expect, y_pred))
```

```
              precision    recall  f1-score   support

           F       0.84      0.99      0.91       130
           T       0.67      0.08      0.14        26

    accuracy                           0.84       156
   macro avg       0.75      0.53      0.52       156
weighted avg       0.81      0.84      0.78       156
```

The F1-Score determines that likelihood of an invidual of any age will most likely be safe and will not die post-surgery with a 91% F1-Score as backup. The accuracy is 84% so that is pretty high and significant enough to be deemed trustworthy of a predictor. As can be observed, we learn that an invidual of any age will most likekly not die after surgery (post 1 year tracking). False represents the system's machine learning model's linkage of data points leading to "surviving surgery" with True being vice versa. This is very ideal actually, as it suggests that the risk is not as high as one thinks and that if having severe/abnormal spondylilothesis (as can be found through a predictor from the first section), it is reccomended to pursue surgery without too much fear. Of course, clinicians will be analyzing to see if further physiological and physcological parameters a patient may have would hinder it -- but at the very least it seems that even with symptoms, the overall results showcase a leaning inclination to indeed take the risk and go for the surgery. Yes of course you may still be at risk of failure, as the 14% F1-Score shows, but sacrifices must happen if you want to not end up in a wheelchair or deceased -- which would be a gunatreed evntuality if not going for surgery.

I hope this project was enjoyable and insightful to read and will be updating this annually as I become more knowledable with ML/AI practice! Thank you for reading through this and please feel free to email me if you any inquiries: nicolasztan@gmail.com

In [ ]:
```
```

REFERENCES

Eltoukhy, Moataz, et al. “Examination of a Lumbar Spine Biomechanical Model for Assessing Axial Compression, Shear, and Bending Moment Using Selected Olympic Lifts.” Journal of Orthopedics, Elsevier, 18 May 2015, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4925752/.

Fares, Mohamad Y, et al. “Low Back Pain among Weightlifting Adolescents and Young Adults.” Cureus, Cureus, 11 July 2020, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7417116/.

Spondylolyisis and Spondylolisthesis. https://www.sportsmedreview.com/blog/spondylolyisis-and-spondylolisthesis/

Correlation of Pelvic Parameters with Isthmic Spondylolisthesis. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2852036/

https://pbs.twimg.com/media/ElvQ-aiWwAABNFh.jpg

Predicting Complications in Elderly Patients Undergoing Lumbar Decompression. https://journals.lww.com/corr/Abstract/2001/03000/Predicting_Complications_in_Elderly_Patients.14.asp

Relationship between the spino-pelvic parameters and the slip grade in isthmic spondylolisthesis. https://pubmed.ncbi.nlm.nih.gov/26652872/

Sciatica Terminology Explained. https://www.mayoclinic.org/diseases-conditions/sciatica/symptoms-causes/syc-2037743

Instrumented fusion surgery in elderly patients (over 75 years old): clinical and radiological results in a series of 53 patients. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3830041/

The incidence of myocardial infarction after lumbar spine surgery. https://pubmed.ncbi.nlm.nih.gov/31325049/

Fischer, Samuel C., et al. “Effect of an Exercise Program That Includes Deadlifts on Low Back Pain.” Human Kinetics, Human Kinetics, 24 Feb. 2021, https://journals.humankinetics.com/view/journals/jsr/30/4/article-p672.xml.

Hussain, Ali. “Lower Back Pain Symptoms Dataset(Labeled).” Kaggle, 5 Dec. 2017, #1https://www.kaggle.com/alihussain1993/lower-back-pain-symptoms-datasetlabelled. Original Dataset was Cited and Generated at UCI: https://archive.ics.uci.edu/ml/datasets/Vertebral+Column (Vertebral Column Dataset)

Vetebrae Angling Dataset: https://www.kaggle.com/caesarlupum/vertebralcolumndataset

(PDF) Pose Trainer: Correcting Exercise Posture Using Pose ... https://www.researchgate.net/publication/324759769_Pose_Trainer_Correcting_Exercise_Posture_using_P

Siddhartha, Manu. “Thoracic Surgery Dataset.” Kaggle, 29 July 2019, https://www.kaggle.com/sid321axn/thoraric-surgery.

University of Kentucky Uknowledge. https://uknowledge.uky.edu/cgi/viewcontent.cgi?article=1091&context=khp_etds.

(External: Presentation Pitch Resources)

My Organizer Google Doc for this project
https://docs.google.com/document/d/1BJrrtXFppDQEznRvCuQoYgYIOWVKn3wX0i5XRiplKMY/edit

https://rapidapi.com/stefan.skliarov/api/HumanAPI/

https://tonygentilcore.com/2017/06/deadlifts-are-only-dangerous-because-youre-unable-coach/

https://www.hss.edu/condition-list_spondylolysis-spondylolisthesis.asp#:~:text=Spondylolysis%20.

https://towardsdatascience.com/an-exploratory-data-analysis-on-lower-back-pain-6283d0b012

Incredibly Useful ML Resources

What is the test-train split https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

How to train an instance K-NN model https://www.youtube.com/watch?v=s-9Qqpv2hTY

In [ ]: