introdução

Este trabalho busca obter uma rede neural para classificação dores lombares em normais ou anormais baseado em sintomas e características fisiologicas utilizando o modelo de rede neural MLP.

A base de dados foi a lower back pain symptons dataset, retirada da plataforma kaggle. Para rede neural foi utilizada a biblioteca scikit-learn da linguagem python.

Bibliotecas e Dados

A seguir estão as bibliotecas usadas durante todo o projeto.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.base import TransformerMixin
from sklearn.preprocessing import (FunctionTransformer,
StandardScaler)
from sklearn.decomposition import PCA
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
as LDA
from scipy.stats import boxcox
from sklearn.model_selection import (train_test_split, KFold,
StratifiedKFold, cross_val_score, GridSearchCV, learning_curve,
validation_curve)
from sklearn.pipeline import Pipeline
from sklearn.base import BaseEstimator, TransformerMixin
from collections import Counter
import warnings

from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from xgboost import (XGBClassifier, plot_importance)
from sklearn.svm import SVC
from sklearn.ensemble import (RandomForestClassifier,
AdaBoostClassifier, ExtraTreesClassifier, GradientBoostingClassifier)
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from time import time

from sklearn.neural_network import MLPClassifier
from sklearn.neural_network import MLPRegressor

from sklearn.model_selection import train_test_split
```

```python
from sklearn.metrics import mean_squared_error
from math import sqrt
from sklearn.metrics import r2_score
from sklearn.metrics import classification_report,confusion_matrix

from zipfile import ZipFile
import urllib.request
import requests
import shutil
import os

%matplotlib inline
warnings.filterwarnings('ignore')
sns.set_style('whitegrid')
```

carregamos a nossa base de dados como um dataframe.

```python
df = pd.read_csv(r'Dataset_spine.xls')
```

a seguir estão as colunas com os atributos. O atributo que desejamos prever é o 13 que classifica a dor lombar de uma pessoa como normal ou anormal.

```python
print(df)
```

```
          Col1       Col2       Col3       Col4        Col5       Col6  \
0    63.027817  22.552586  39.609117  40.475232   98.672917  -0.254400
1    39.056951  10.060991  25.015378  28.995960  114.405425   4.564259
2    68.832021  22.218482  50.092194  46.613539  105.985135  -3.530317
3    69.297008  24.652878  44.311238  44.644130  101.868495  11.211523
4    49.712859   9.652075  28.317406  40.060784  108.168725   7.918501
..         ...        ...        ...        ...         ...        ...
305  47.903565  13.616688  36.000000  34.286877  117.449062  -4.245395
306  53.936748  20.721496  29.220534  33.215251  114.365845  -0.421010
307  61.446597  22.694968  46.170347  38.751628  125.670725  -2.707880
308  45.252792   8.693157  41.583126  36.559635  118.545842   0.214750
309  33.841641   5.073991  36.641233  28.767649  123.945244  -0.199249

         Col7      Col8      Col9      Col10      Col11     Col12
```

```
Class_att  \
0    0.744503  12.5661  14.5386  15.30468 -28.658501  43.5123
Abnormal
1    0.415186  12.8874  17.5323  16.78486 -25.530607  16.1102
Abnormal
2    0.474889  26.8343  17.4861  16.65897 -29.031888  19.2221
Abnormal
3    0.369345  23.5603  12.7074  11.42447 -30.470246  18.8329
Abnormal
4    0.543360  35.4940  15.9546   8.87237 -16.378376  24.9171
Abnormal
..        ...      ...      ...       ...        ...      ...       ..
.
305  0.129744   7.8433  14.7484   8.51707 -15.728927  11.5472
Normal
306  0.047913  19.1986  18.1972   7.08745   6.013843  43.8693
Normal
307  0.081070  16.2059  13.5565   8.89572   3.564463  18.4151
Normal
308  0.159251  14.7334  16.0928   9.75922   5.767308  33.7192
Normal
309  0.674504  19.3825  17.6963  13.72929   1.783007  40.6049
Normal

                                           Unnamed: 13
0                                                  NaN
1                                                  NaN
2    Prediction is done by using binary classificat...
3                                                  NaN
4                                                  NaN
..                                                 ...
305                                                NaN
306                                                NaN
307                                                NaN
308                                                NaN
309                                                NaN

[310 rows x 14 columns]
```

altereramos o nome das colunas para o nome real dos dados.

```
df.rename(columns = {'Col1':'pelvic_incidence'}, inplace = True)
df.rename(columns = {'Col2':'pelvic tilt'}, inplace = True)
df.rename(columns = {'Col3':'lumbar_lordosis_angle'}, inplace = True)
df.rename(columns = {'Col4':'sacral_slope'}, inplace = True)
df.rename(columns = {'Col5':'pelvic_radius'}, inplace = True)
df.rename(columns = {'Col6':'degree_spondylolisthesis'}, inplace = 
True)
df.rename(columns = {'Col7':'pelvic_slope'}, inplace = True)
```

```
df.rename(columns = {'Col8':'Direct_tilt'}, inplace = True)
df.rename(columns = {'Col9':'thoracic_slope'}, inplace = True)
df.rename(columns = {'Col10':'cervical_tilt'}, inplace = True)
df.rename(columns = {'Col11':'sacrum_angle'}, inplace = True)
df.rename(columns = {'Col12':'scoliosis_slope'}, inplace = True)
```

o atributo 13 é uma string e precisamos muda-lo para um numero real. Escolheremos 1 para representar um diagnóstico anormal e 0 para normal.

```
class_att = list(df['Class_att'])
print(class_att)
```

```
['Abnormal', 'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal',
'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal',
'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal',
'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal',
'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal',
'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal',
'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal',
'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal',
'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal',
'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal',
'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal',
'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal',
'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal',
'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal',
'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal',
'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal',
'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal',
'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal',
'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal',
'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal',
'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal',
'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal',
'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal',
'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal',
'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal',
'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal',
'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal',
'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal',
'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal',
'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal',
'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal',
'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal',
'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal',
'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal',
'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal',
'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal',
'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal',
'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal',
'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal',
```

```
'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal',
'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal',
'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal',
'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal',
'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal', 'Abnormal', 'Normal',
'Normal', 'Normal', 'Normal', 'Normal', 'Normal', 'Normal', 'Normal',
'Normal', 'Normal', 'Normal', 'Normal', 'Normal', 'Normal', 'Normal',
'Normal', 'Normal', 'Normal', 'Normal', 'Normal', 'Normal', 'Normal',
'Normal', 'Normal', 'Normal', 'Normal', 'Normal', 'Normal', 'Normal',
'Normal', 'Normal', 'Normal', 'Normal', 'Normal', 'Normal', 'Normal',
'Normal', 'Normal', 'Normal', 'Normal', 'Normal', 'Normal', 'Normal',
'Normal', 'Normal', 'Normal', 'Normal', 'Normal', 'Normal', 'Normal',
'Normal', 'Normal', 'Normal', 'Normal', 'Normal', 'Normal', 'Normal',
'Normal', 'Normal', 'Normal', 'Normal', 'Normal', 'Normal', 'Normal',
'Normal', 'Normal', 'Normal', 'Normal', 'Normal', 'Normal', 'Normal',
'Normal', 'Normal', 'Normal', 'Normal', 'Normal', 'Normal', 'Normal',
'Normal', 'Normal', 'Normal', 'Normal', 'Normal', 'Normal', 'Normal',
'Normal', 'Normal', 'Normal', 'Normal', 'Normal', 'Normal', 'Normal',
'Normal', 'Normal', 'Normal', 'Normal', 'Normal', 'Normal', 'Normal',
'Normal']
```

```
for i in range(len(class_att)) :
  if class_att[i] == "Normal":
    class_att[i] = 0
  if class_att[i] == "Abnormal":
    class_att[i] = 1

print(class_att)
```

```
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

adicionando a conversão para a o nosso data frame e dropando a antiga 'class_att'

```
df['Att'] = class_att
df.drop(df.columns[[12]], axis=1, inplace=True)
print(df)
```

```
     pelvic_incidence  pelvic tilt  lumbar_lordosis_angle
sacral_slope  \
0           63.027817    22.552586              39.609117
40.475232
1           39.056951    10.060991              25.015378
28.995960
2           68.832021    22.218482              50.092194
46.613539
3           69.297008    24.652878              44.311238
44.644130
4           49.712859     9.652075              28.317406
40.060784
..                ...          ...                    ...            ..
.
305         47.903565    13.616688              36.000000
34.286877
306         53.936748    20.721496              29.220534
33.215251
307         61.446597    22.694968              46.170347
38.751628
308         45.252792     8.693157              41.583126
36.559635
309         33.841641     5.073991              36.641233
28.767649

     pelvic_radius  degree_spondylolisthesis  pelvic_slope
Direct_tilt  \
0        98.672917                 -0.254400      0.744503
12.5661
1       114.405425                  4.564259      0.415186
12.8874
2       105.985135                 -3.530317      0.474889
26.8343
3       101.868495                 11.211523      0.369345
23.5603
4       108.168725                  7.918501      0.543360
35.4940
..             ...                       ...           ...            ..
.
305     117.449062                 -4.245395      0.129744
7.8433
306     114.365845                 -0.421010      0.047913
19.1986
307     125.670725                 -2.707880      0.081070
16.2059
308     118.545842                  0.214750      0.159251
14.7334
309     123.945244                 -0.199249      0.674504
19.3825
```

```
     thoracic_slope  cervical_tilt  sacrum_angle  scoliosis_slope  Att
0           14.5386       15.30468    -28.658501          43.5123    1
1           17.5323       16.78486    -25.530607          16.1102    1
2           17.4861       16.65897    -29.031888          19.2221    1
3           12.7074       11.42447    -30.470246          18.8329    1
4           15.9546        8.87237    -16.378376          24.9171    1
..              ...            ...           ...              ...  ...
305         14.7484        8.51707    -15.728927          11.5472    0
306         18.1972        7.08745      6.013843          43.8693    0
307         13.5565        8.89572      3.564463          18.4151    0
308         16.0928        9.75922      5.767308          33.7192    0
309         17.6963       13.72929      1.783007          40.6049    0

[310 rows x 13 columns]
```

Análise dos Dados

Nesta seção serão apresentadas algumas características e descrição da nossa base de dados.

As informações mais básicas como média, minimo e máximo e quartis:

```
df1 = df[ df.columns.tolist()[0:4] ]
df1.describe()
```

```
       pelvic_incidence  pelvic tilt  lumbar_lordosis_angle
sacral_slope
count        310.000000   310.000000             310.000000
310.000000
mean          60.496653    17.542822              51.930930
42.953831
std           17.236520    10.008330              18.554064
13.423102
min           26.147921    -6.554948              14.000000
13.366931
25%           46.430294    10.667069              37.000000
33.347122
```

50%	58.691038	16.357689	49.562398	42.404912
75%	72.877696	22.120395	63.000000	52.695888
max	129.834041	49.431864	125.742385	121.429566

```python
df1 = df[ df.columns.tolist()[4:8] ]
df1.describe()
```

	pelvic_radius	degree_spondylolisthesis	pelvic_slope	Direct_tilt
count	310.000000	310.000000	310.000000	310.000000
mean	117.920655	26.296694	0.472979	21.321526
std	13.317377	37.559027	0.285787	8.639423
min	70.082575	-11.058179	0.003220	7.027000
25%	110.709196	1.603727	0.224367	13.054400
50%	118.268178	11.767934	0.475989	21.907150
75%	125.467674	41.287352	0.704846	28.954075
max	163.071041	418.543082	0.998827	36.743900

```python
df1 = df[ df.columns.tolist()[8:13] ]
df1.describe()
```

	thoracic_slope	cervical_tilt	sacrum_angle	scoliosis_slope
count	310.000000	310.000000	310.000000	310.000000
mean	13.064511	11.933317	-14.053139	25.645981
std	3.399713	2.893265	12.225582	10.450558
min	7.037800	7.030600	-35.287375	7.007900
25%	10.417800	9.541140	-24.289522	17.189075
50%	12.938450	11.953835	-14.622856	24.931950
75%	15.889525	14.371810	-3.497094	33.979600
max	19.324000	16.821080	6.972071	44.341200

Correlação

Vamos analisar algumas caractéristicas da nossa base de dados. Primeiramente vamos ver a matriz de correlação.

```python
corr = df.corr()
corr.style.background_gradient(cmap='coolwarm')
```

```
<pandas.io.formats.style.Styler at 0x7f5c7cc33c40>
```

Vemos que em geral os atributos apresentão uma baixa correlação. Os mais correlacionados são:

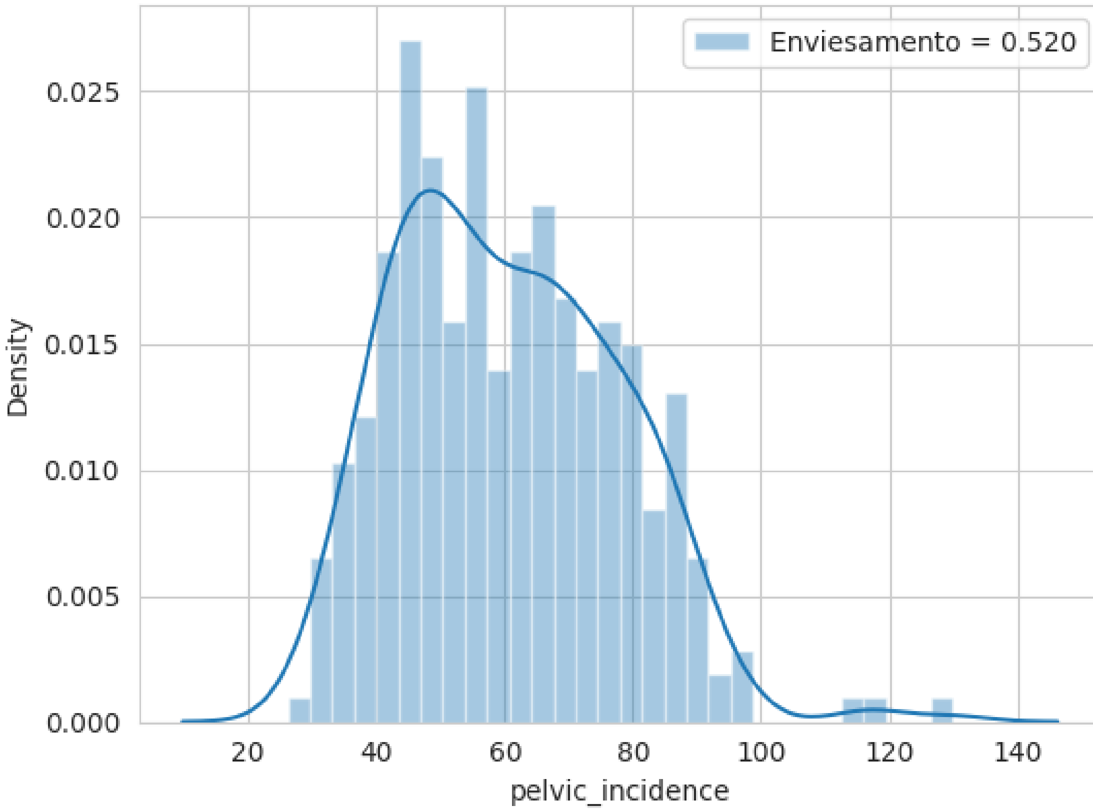
- col1: pelvic_incidence
- Col2: pelvic tilt
- Col3: lumbar_lordosis_angle
- Col4: sacral_slope
- Col5: pelvic_radius
- Col6: degree_spondylolisthesis

Para o nosso atributo alvo vemos que as maiores correlações é entre degree_spondylolisthesis, pelvic_incidence e pelvic_tilt. As maiores anticorrelações são com pelvic_radius, scoliosis_slope e thoracic_slope. As com menor correlação são sacrum angle e direct_tilt.

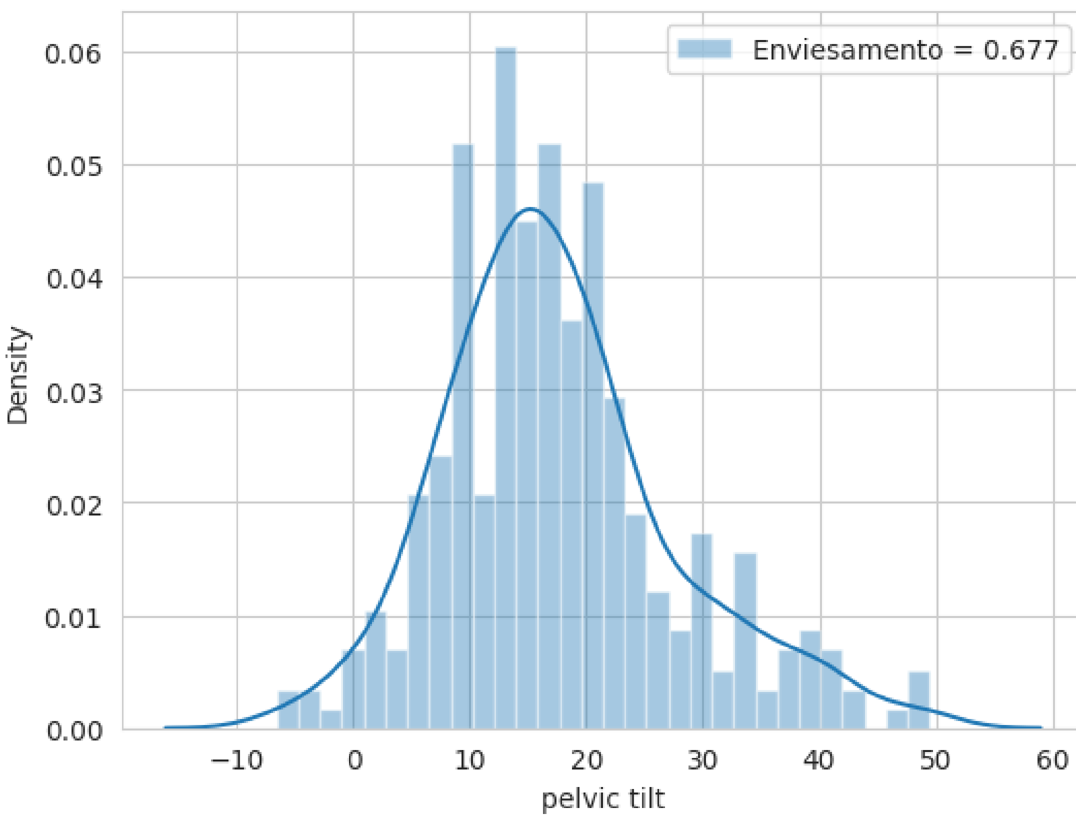
distribuição

A seguir estão a distribuição de todos os atributos.

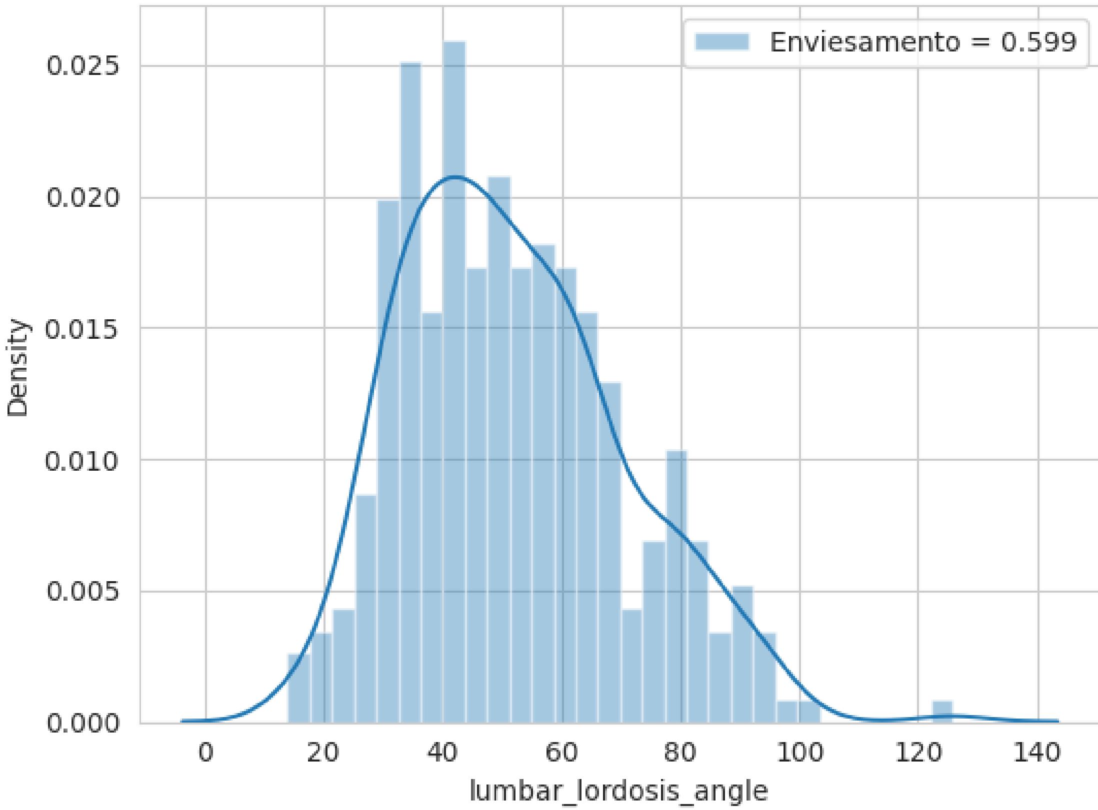
```
for feat in df.columns:
    skew = df[feat].skew()
    sns.distplot(df[feat], kde=True, label='Enviesamento = %.3f' %
(skew), bins=30)
    plt.legend(loc='best')
    plt.show()
```

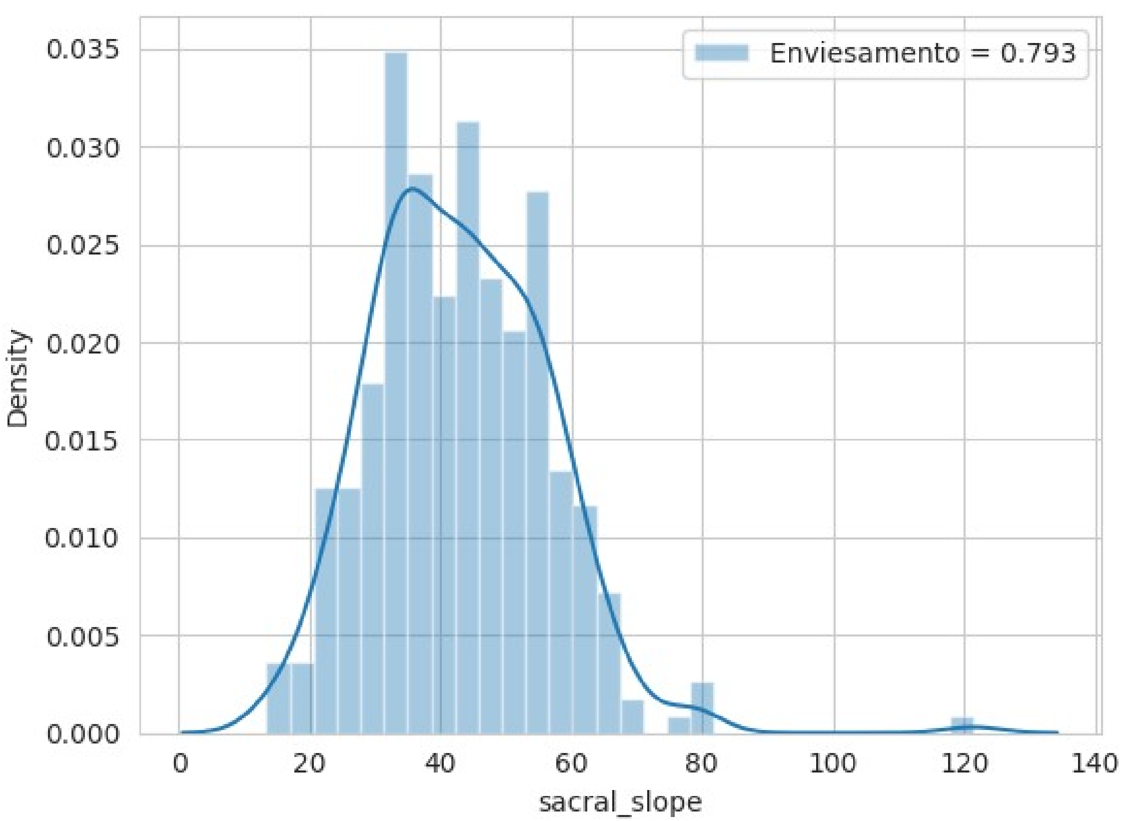


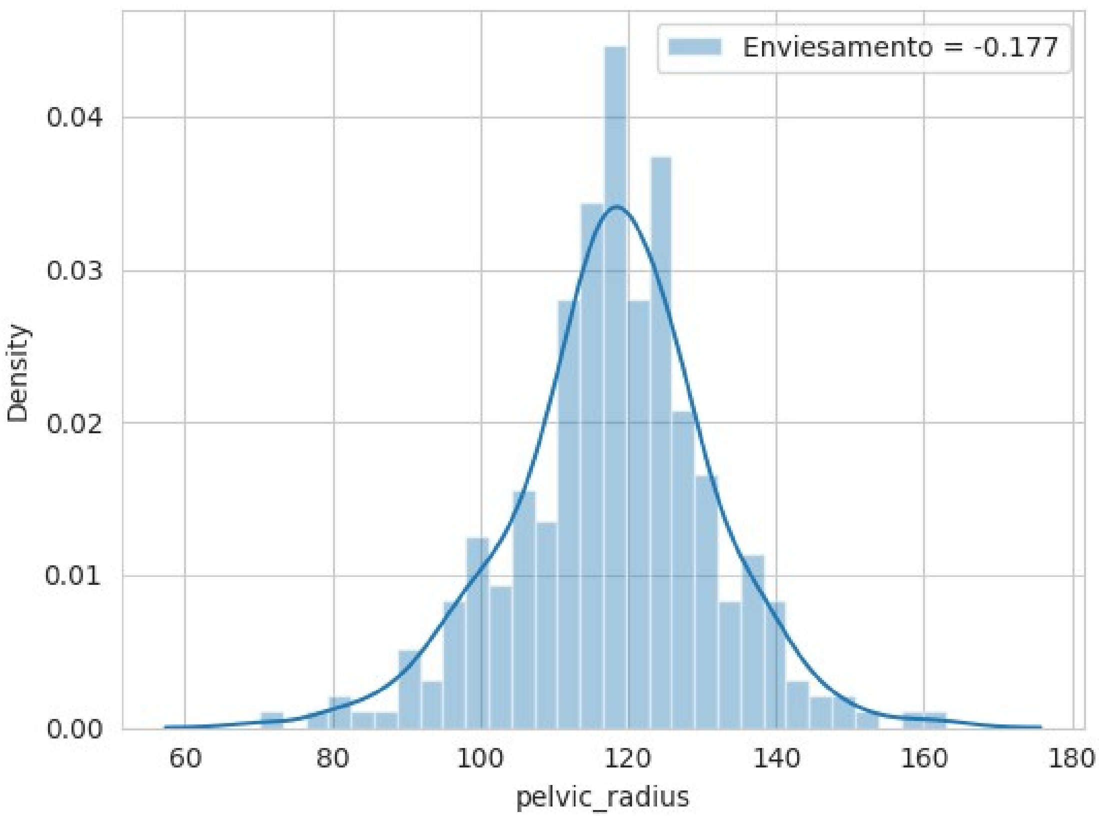
Enviesamento = 0.520
0.025
0.020
0.015
0.010
0.005
0.000
Density
20
40
60
80
100
120
140
pelvic_incidence



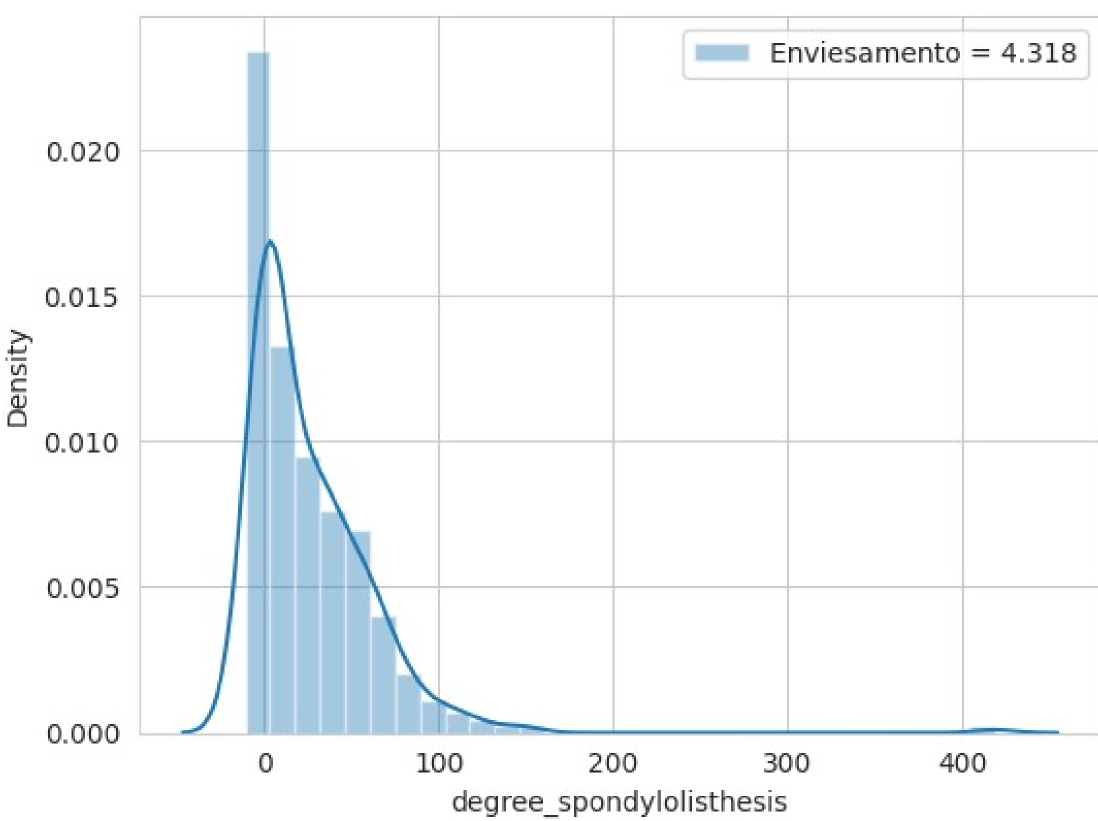
Enviesamento = 0.677
0.06
0.05
0.04
0.03
0.02
0.01
0.00
Density
−10
0
10
20
30
40
50
60
pelvic tilt


Enviesamento = 0.599
Density
0.000
0.005
0.010
0.015
0.020
0.025
0
20
40
60
80
100
120
140
lumbar_lordosis_angle

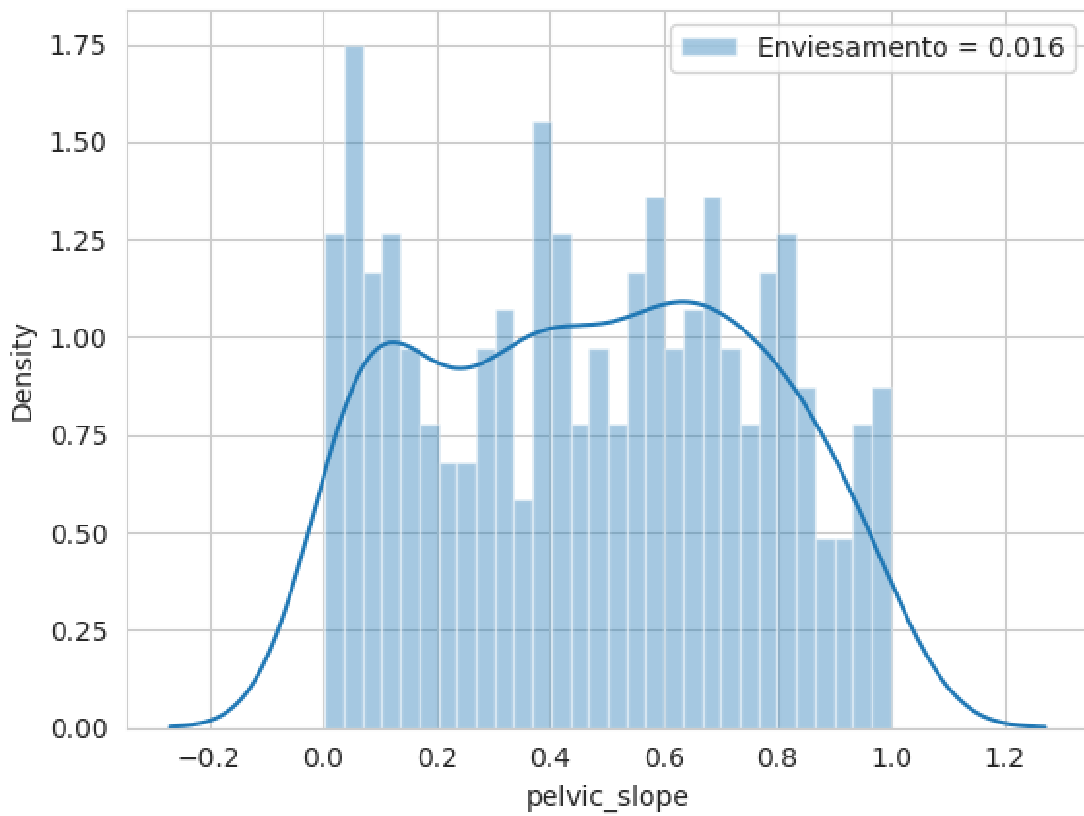

Enviesamento = 0.793
Density
0.000
0.005
0.010
0.015
0.020
0.025
0.030
0.035
0
20
40
60
80
100
120
140
sacral_slope

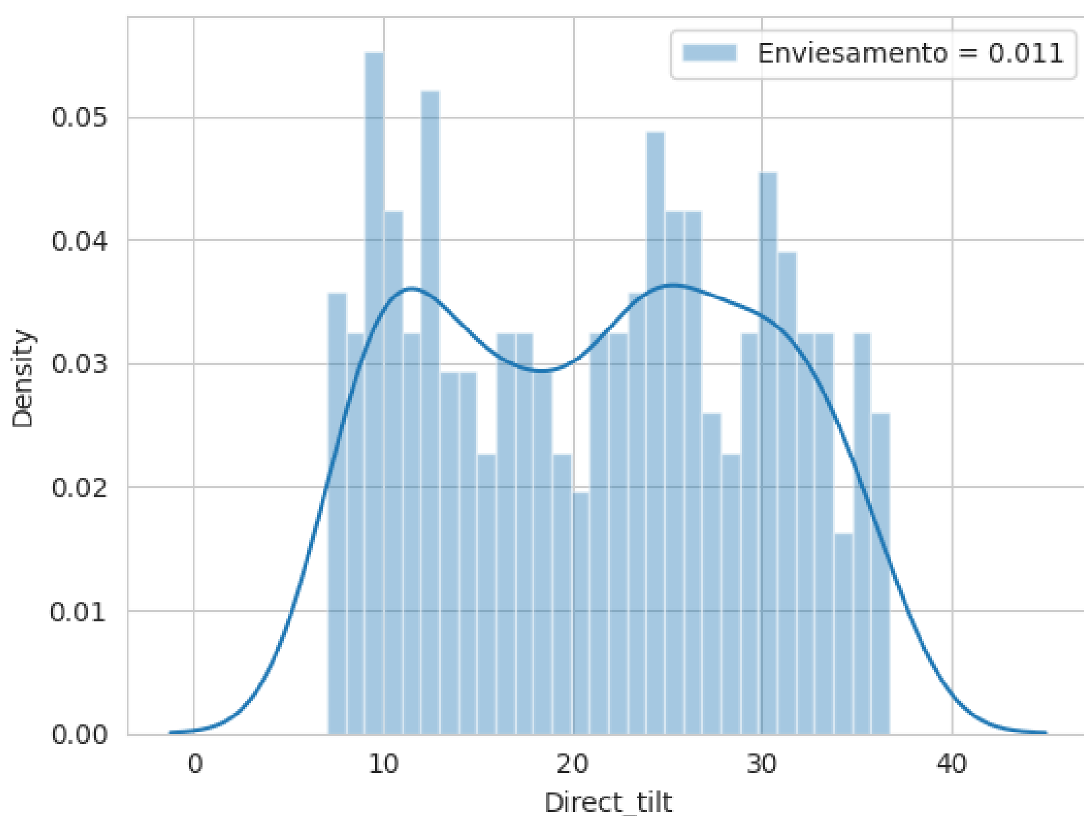


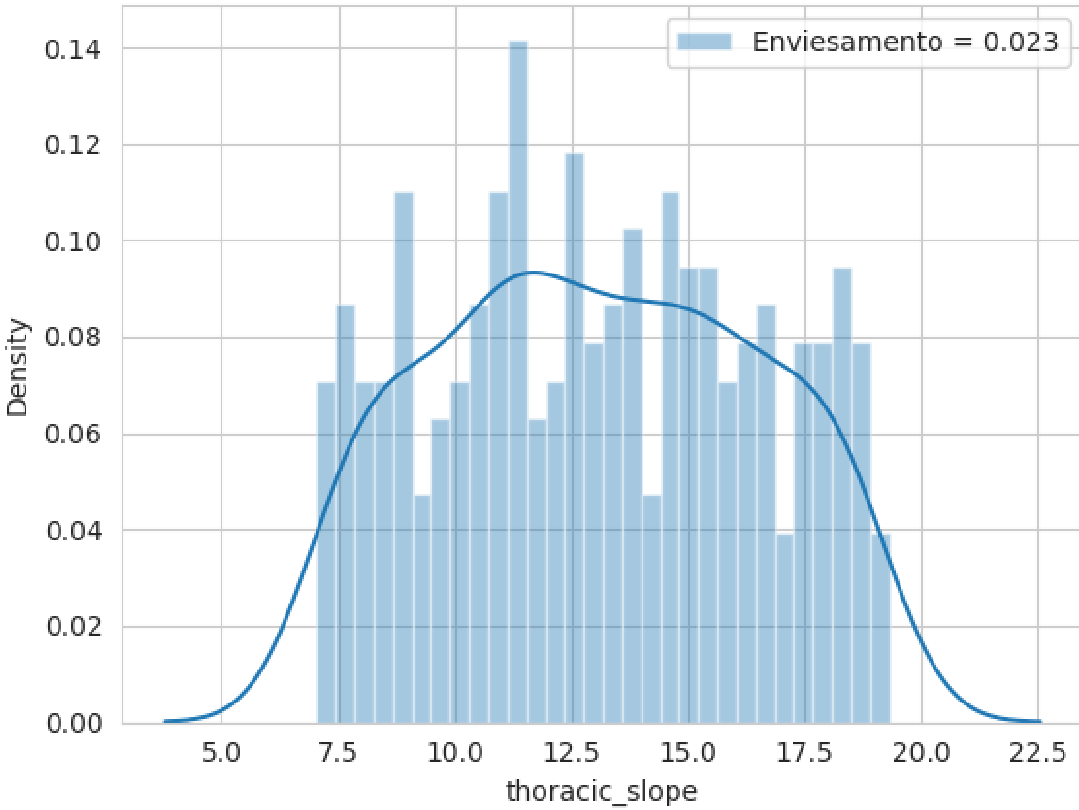
Enviesamento = -0.177
0.04
0.03
0.02
0.01
0.00
Density
60
80
100
120
140
160
180
pelvic_radius

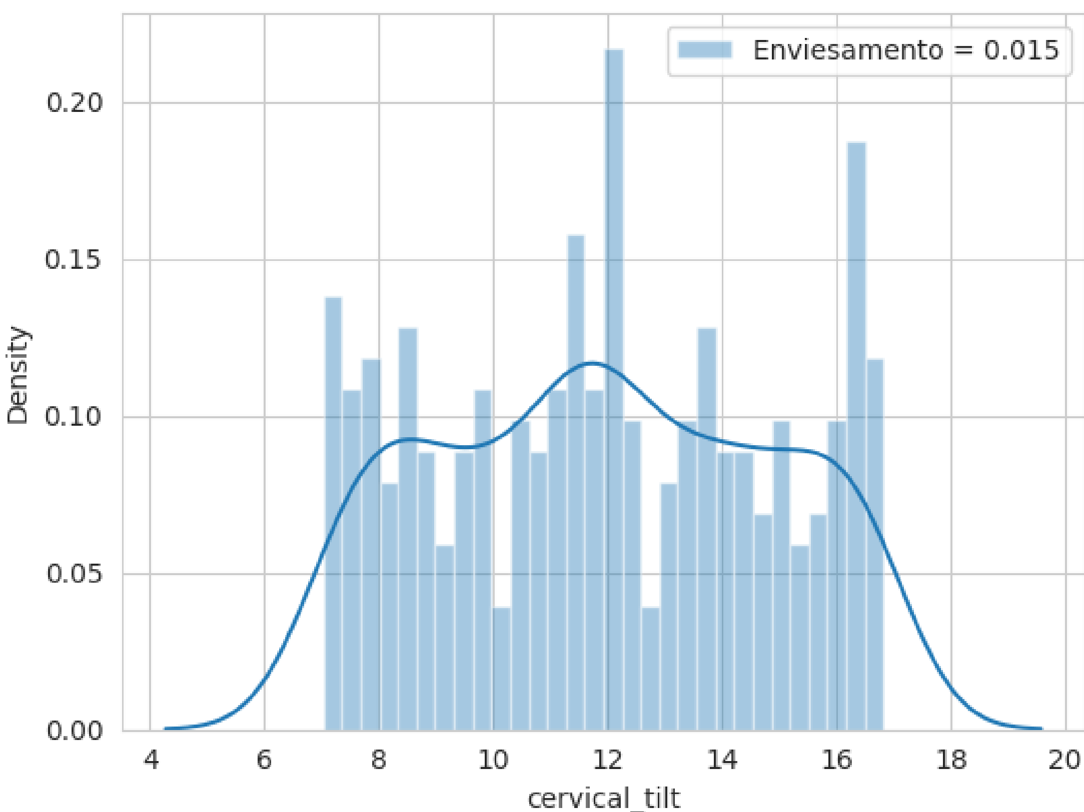


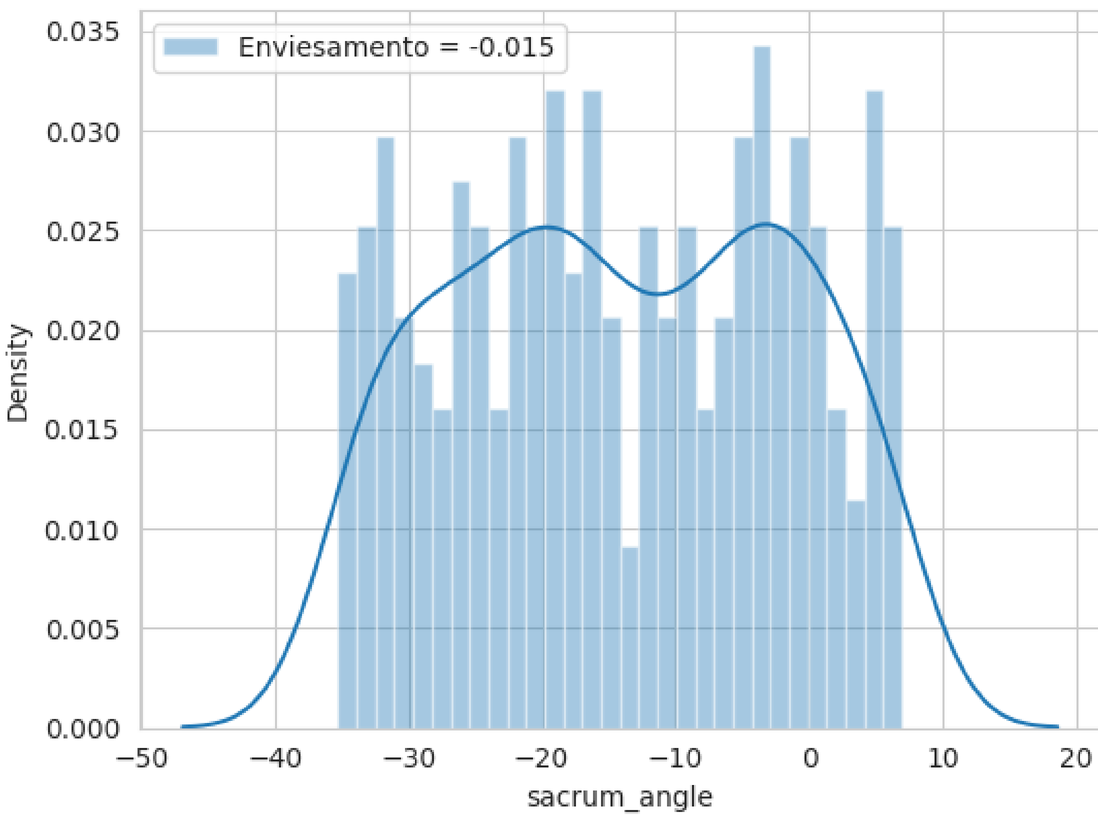
Enviesamento = 4.318
0.020
0.015
0.010
0.005
0.000
Density
0
100
200
300
400
degree_spondylolisthesis


Enviesamento = 0.016
Density
pelvic_slope

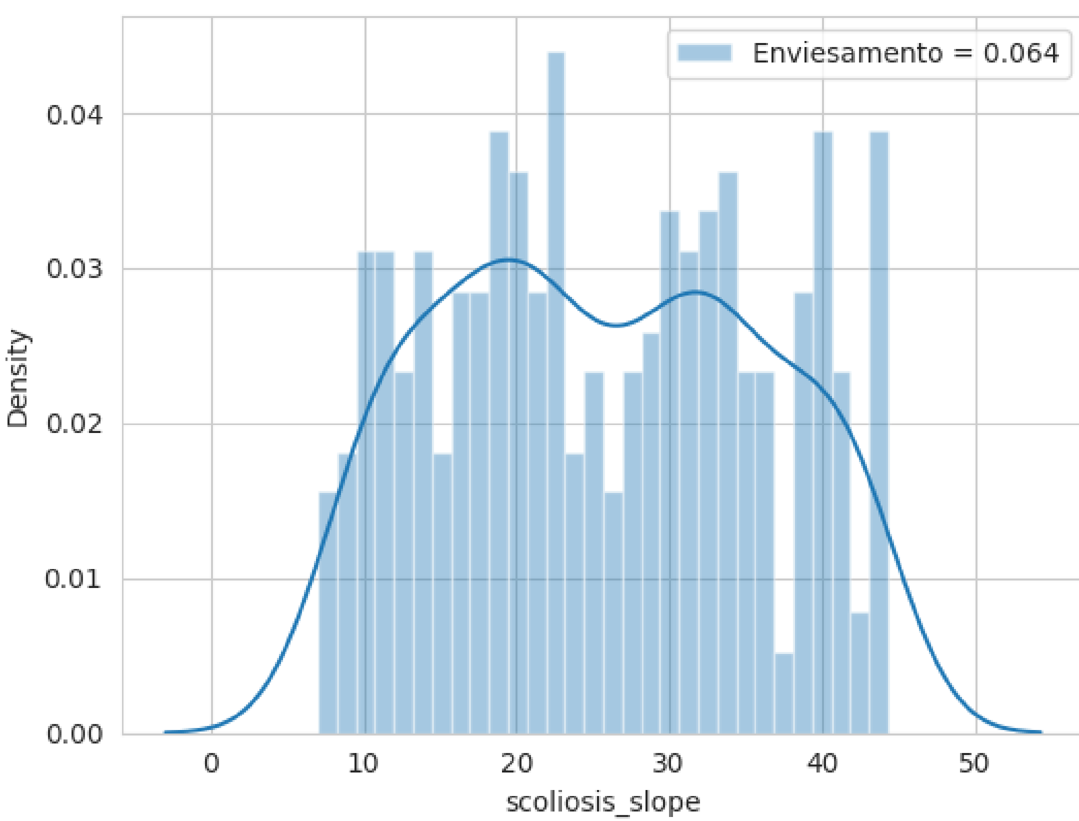

Enviesamento = 0.011
Density
Direct_tilt


Enviesamento = 0.023
Density
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
5.0
7.5
10.0
12.5
15.0
17.5
20.0
22.5
thoracic_slope

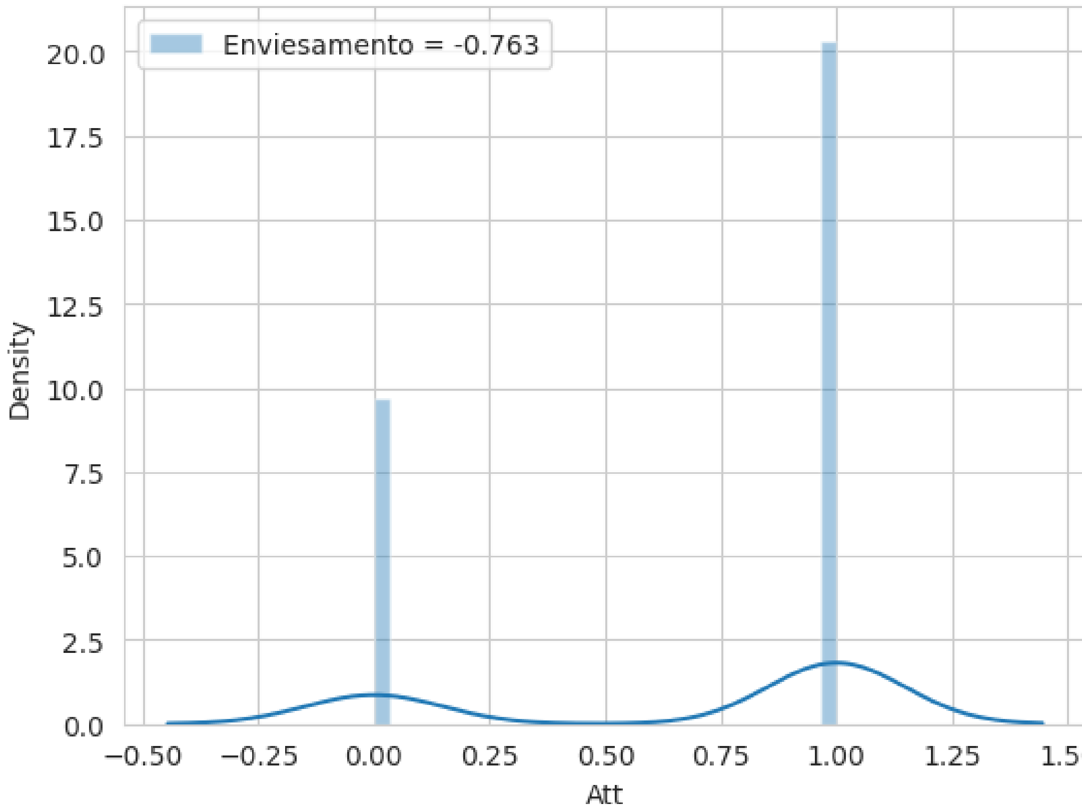

Enviesamento = 0.015
Density
0.00
0.05
0.10
0.15
0.20
4
6
8
10
12
14
16
18
20
cervical_tilt



Enviesamento = -0.015
Density
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
−50
−40
−30
−20
−10
0
10
20
sacrum_angle



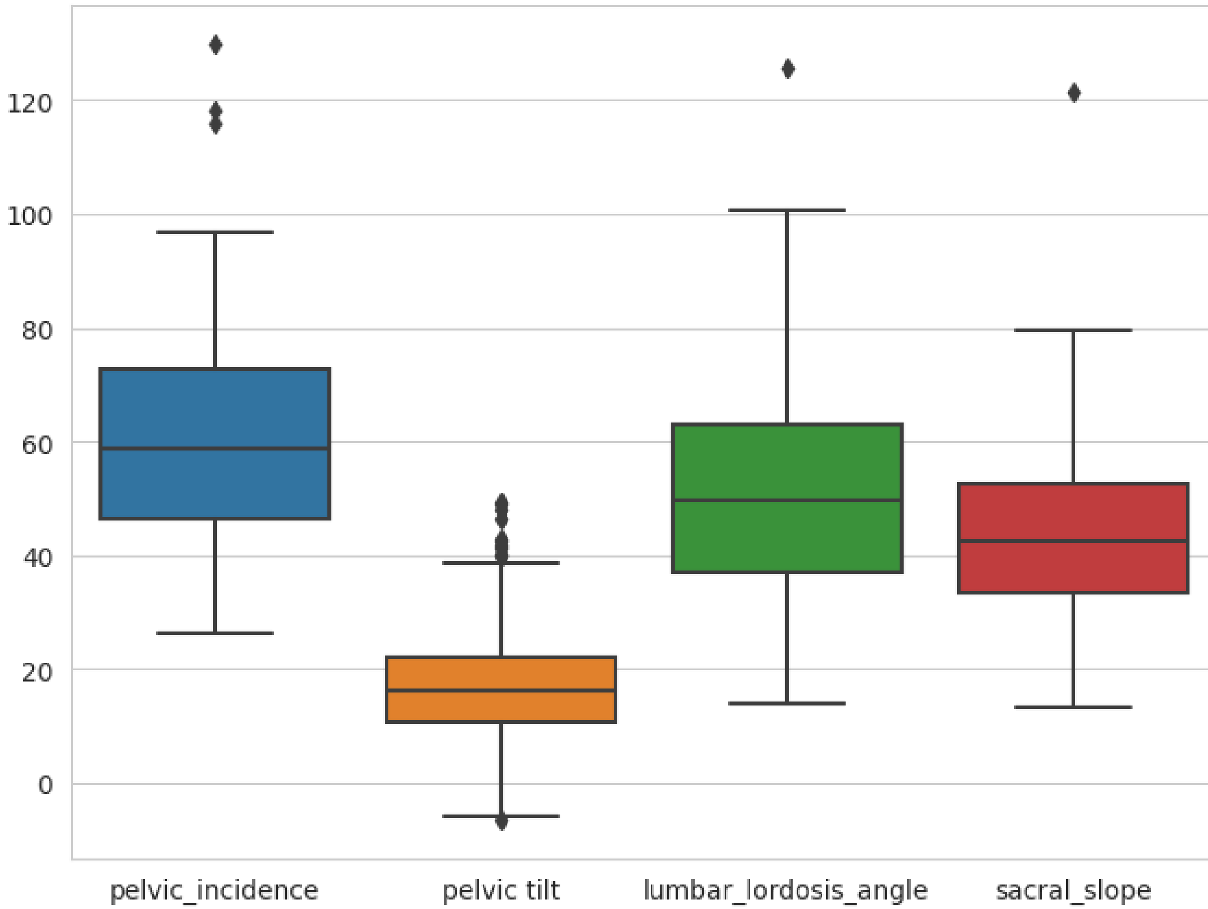
Enviesamento = 0.064
Density
0.04
0.03
0.02
0.01
0.00
0
10
20
30
40
50
scoliosis_slope



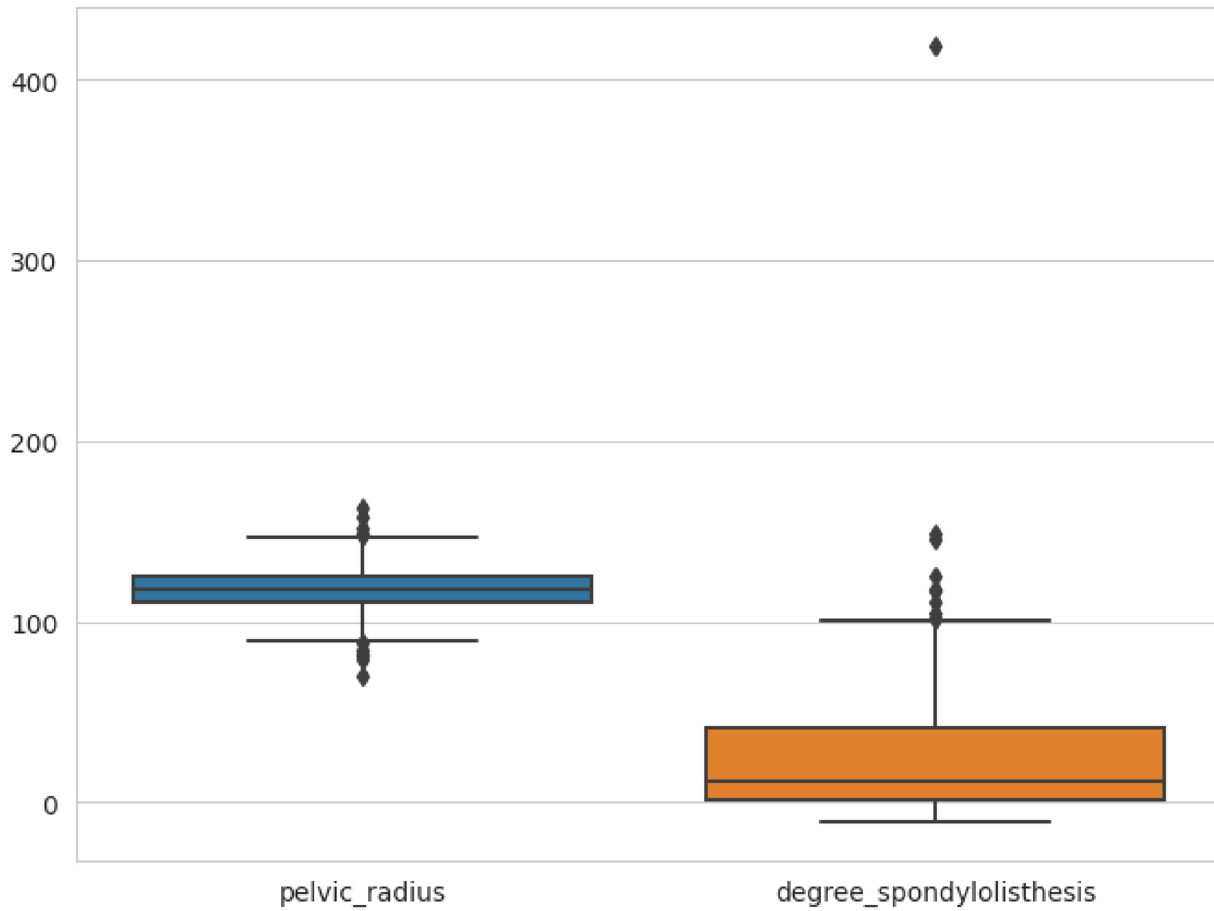
box plots

boxplots dos atributos

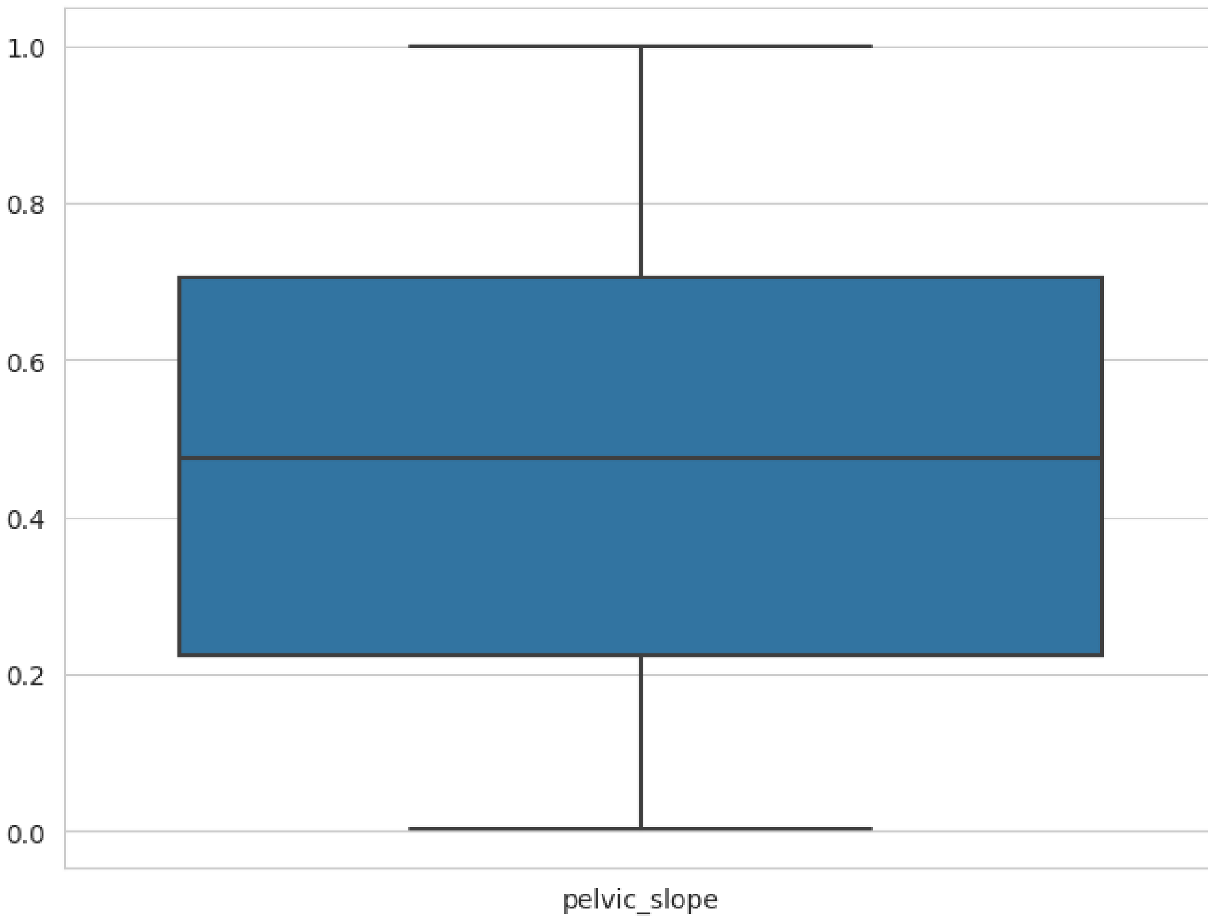
```
att1 = df.columns.tolist()[0:4]
att2 = df.columns.tolist()[4:6]
att3 = df.columns.tolist()[6:7]
att4 = df.columns.tolist()[7:12]

plt.figure(figsize=(8,6))
sns.boxplot(data=df[att1])
plt.show()
```



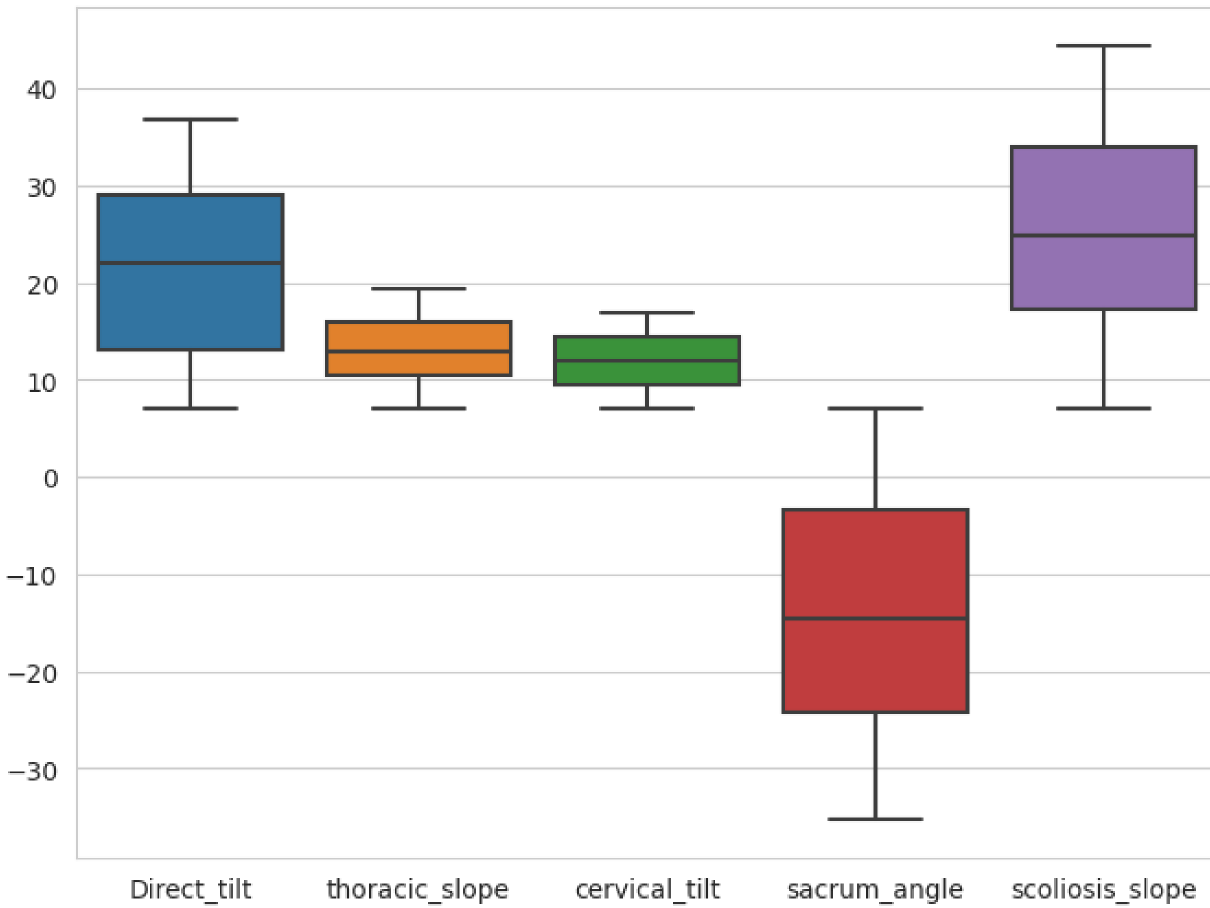

```
plt.figure(figsize=(8,6))
sns.boxplot(data=df[att2])
plt.show()
```




```
plt.figure(figsize=(8,6))
sns.boxplot(data=df[att3])
plt.show()
```




```
plt.figure(figsize=(8,6))
sns.boxplot(data=df[att4])
plt.show()
```



como podemos ver há presença de outliers nos atributos mais correlatos. Podemos remove-los ou deixa-los na nossa base dados. Se removermos podemos obter resultados mais concisos porem estaremos diminuindo nossa base de dados e podemos ter overfiting.

Por ter um coeficiente de correlação relativamente baixo com nosso atributo alvo resolvemos não remover os outliers.

Porem é fato que outliers podem corromper uma base dados. Há tecnicas para lidar com eles como mudar a função do calculo de erro. porem não há como alterar a função de error na MLP da biblioteca scikit-learn.

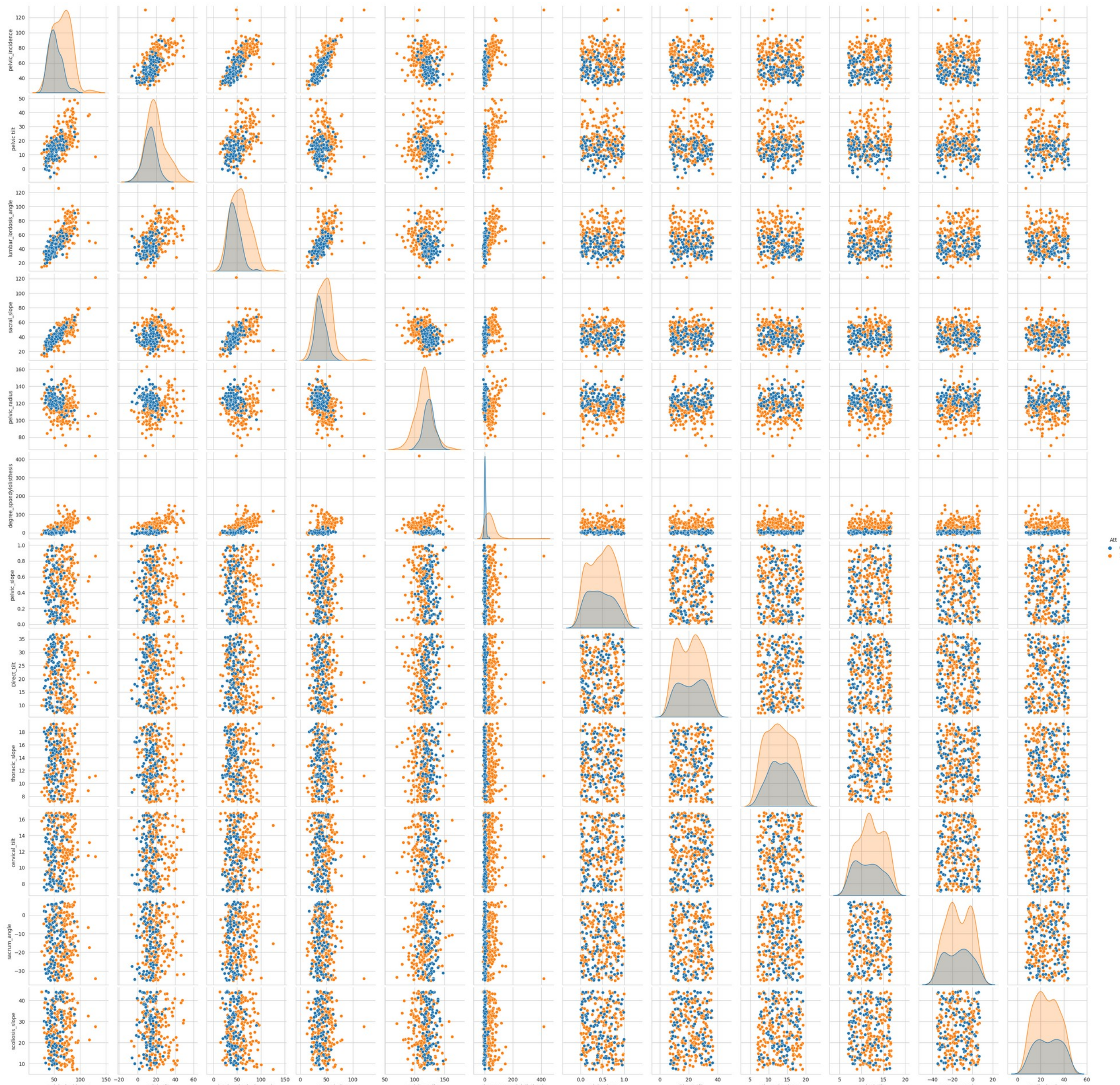
distribuição pair plot

distribuição pair plot serve para visualizar relações entre os diferentes atributos da nossa base de dados. Usando a função pairplot da biblioteca seaborn podemos projetar essas relações. Destacamos o parâmetro Att que é nosso diagnóstico e variável de interesse em destaque.

Em laranja estão os diagnósticos considerados anormais e em azul os normais.

```
plt.figure(figsize=(8,8))
sns.pairplot(df,hue="Att" )
plt.show()
```

```
<Figure size 800x800 with 0 Axes>
```



Rede neural e treinamento

Nesta seção descreveremos a treino, implementação e resultados da nossa rede neural. Seguindo os passos esse tutorial implementaremos nossa rede MLP.

primeiramente definimos o atributo alvo e normalizamos os outros atributos.

```
target_column = ['Att']
predictors = list(set(list(df.columns))-set(target_column))
```

```
df[predictors] = df[predictors]/df[predictors].max()
df.describe().transpose()
```

```
                          count      mean       std       min
25%  \
pelvic_incidence          310.0  0.465954  0.132758  0.201395
0.357613
pelvic tilt               310.0  0.354889  0.202467 -0.132606
0.215793
lumbar_lordosis_angle     310.0  0.412995  0.147556  0.111339
0.294252
sacral_slope              310.0  0.353735  0.110542  0.110080
0.274621
pelvic_radius             310.0  0.723124  0.081666  0.429767
0.678902
degree_spondylolisthesis  310.0  0.062829  0.089738 -0.026421
0.003832
pelvic_slope              310.0  0.473535  0.286122  0.003224
0.224631
Direct_tilt               310.0  0.580274  0.235125  0.191243
0.355281
thoracic_slope            310.0  0.676077  0.175932  0.364200
0.539112
cervical_tilt             310.0  0.709426  0.172002  0.417964
0.567213
sacrum_angle              310.0 -2.015633  1.753508 -5.061247 -
3.483832
scoliosis_slope           310.0  0.578378  0.235685  0.158045
0.387655
Att                       310.0  0.677419  0.468220  0.000000
0.000000

                               50%       75%  max
pelvic_incidence          0.452047  0.561314  1.0
pelvic tilt               0.330914  0.447493  1.0
lumbar_lordosis_angle     0.394158  0.501024  1.0
sacral_slope              0.349214  0.433963  1.0
pelvic_radius             0.725256  0.769405  1.0
degree_spondylolisthesis  0.028116  0.098645  1.0
pelvic_slope              0.476548  0.705674  1.0
Direct_tilt               0.596212  0.787997  1.0
thoracic_slope            0.669553  0.822269  1.0
cervical_tilt             0.710646  0.854393  1.0
sacrum_angle             -2.097347 -0.501586  1.0
scoliosis_slope           0.562275  0.766321  1.0
Att                       1.000000  1.000000  1.0
```

separamos os valores usados para teste e treinamento.

```
X = df[predictors].values
y = df[target_column].values

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.20, random_state=40)

print(X_train.shape); print(X_test.shape)
```

```
(248, 12)
(62, 12)
```

instanciamos um MLP e com três camadas, cada camado com 13 neurônios, função de ativação relu e método de optimização de pesos para adam. Usamos o metodo predict para prever as saidas dos testes e dos treinos.

```
mlp = MLPClassifier(hidden_layer_sizes=(13,13,13), activation='relu',
solver='adam', max_iter=500)

mlp.fit(X_train,y_train)


predict_train = mlp.predict(X_train)
print(predict_train)

predict_test = mlp.predict(X_test)
print(predict_test)
```

```
[1 1 1 0 1 1 1 1 0 1 1 0 1 1 1 1 0 1 0 1 1 1 1 1 1 0 0 0 1 1 0 0 1 1 1
1 1
 0 1 1 1 0 1 1 1 1 1 1 0 0 1 1 1 1 1 0 1 1 1 0 1 1 1 1 1 0 1 0 1 1 1 0
0 1
 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 0 1 1 0 0 1 1 1 1 0 0 0 1 1
1 1
 1 1 0 1 1 1 0 1 1 1 0 0 0 1 1 0 0 0 1 1 0 0 1 1 1 1 1 1 1 0 1 1 0 1 1
1 1
 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 1 0 1 1 0 1 1 1 1 0 1 1 1 1 0 1 1 1 1
1 1
 0 1 1 1 1 1 0 0 1 0 0 1 0 0 1 0 0 1 1 1 0 0 0 1 1 1 0 1 1 1 0 0 1 1 0
0 1
 0 1 0 0 1 1 1 1 0 1 1 1 1 1 1 1 1 1 0 1 0 1 0 1 0 1]
[1 1 1 1 1 1 1 1 0 1 1 0 1 1 1 1 1 0 1 1 1 1 1 1 0 0 1 1 1 0 0 1 1 1 1
1 0
 1 1 1 0 1 0 1 1 1 1 1 1 1 1 0 1 1 1 1 0 0 1 0 1 1]
```

resultados

Esta seção discute os os resultados obtidos pela nossa rede MLP.

abaixo temos a matriz de confusão para as saidas previstas com as saidas reais dos dados de treino.

```python
print(confusion_matrix(y_train,predict_train))
```

```
[[ 61  17]
 [ 23 147]]
```

```python
print(classification_report(y_train,predict_train))
```

```
              precision    recall  f1-score   support

           0       0.80      0.76      0.78        78
           1       0.89      0.91      0.90       170

    accuracy                           0.86       248
   macro avg       0.84      0.83      0.84       248
weighted avg       0.86      0.86      0.86       248
```

abaixo temos a matriz de confusão para as saidas previstas com as saidas reais dos dados de teste.

```python
print(confusion_matrix(y_test,predict_test))
```

```
[[14  8]
 [ 3 37]]
```

```python
print(classification_report(y_test,predict_test))
```

```
              precision    recall  f1-score   support

           0       0.82      0.64      0.72        22
           1       0.82      0.93      0.87        40

    accuracy                           0.82        62
   macro avg       0.82      0.78      0.79        62
weighted avg       0.82      0.82      0.82        62
```

variações

Nesta seção discutimos a variação de algums parâmetros da nossa rede para observar como eles alteram as medidas de precisão e acurácia.

Transformamos nossa rede em uma função que recebe:

- Número de camadas e nodos por camada
- função de ativação
- método de optimazação

```python
hidden_layers = (13, 13, 13)
act = "relu"
solver = 'adam'
```

```
def MLP_net(hidden_layers, act,solver):
  mlp = MLPClassifier(hidden_layer_sizes=hidden_layers, activation=
act, solver='adam', max_iter=500)
  mlp.fit(X_train,y_train)
  predict_train = mlp.predict(X_train)
  predict_test = mlp.predict(X_test)
  print(classification_report(y_test,predict_test))
```

agora vamos fazer variações em alguns parametros para ver como eles influenciam nos resultados.

primeiro vamos alterar a função de ativação e ver como tanh, logistic e identity

```
MLP_net(hidden_layers, "tanh",solver)
```

```
[[15  7]
 [ 2 38]]
              precision    recall  f1-score   support

           0       0.88      0.68      0.77        22
           1       0.84      0.95      0.89        40

    accuracy                           0.85        62
   macro avg       0.86      0.82      0.83        62
weighted avg       0.86      0.85      0.85        62
```

```
MLP_net(hidden_layers, "logistic",solver)
```

```
[[ 0 22]
 [ 0 40]]
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        22
           1       0.65      1.00      0.78        40

    accuracy                           0.65        62
   macro avg       0.32      0.50      0.39        62
weighted avg       0.42      0.65      0.51        62
```

```
MLP_net(hidden_layers, "identity",solver)
```

```
[[13  9]
 [ 4 36]]
              precision    recall  f1-score   support

           0       0.76      0.59      0.67        22
           1       0.80      0.90      0.85        40

    accuracy                           0.79        62
```

```
   macro avg       0.78      0.75      0.76        62
weighted avg       0.79      0.79      0.78        62
```

escolheremos a função de ativação tanh já que ele se provou a mais precisa. Agora vamos comparar as camadas. primeiramente aumentaremos e diminuiremos o numero de nodos em cada camada.

```
hidden_layers = (20,20,20)
MLP_net(hidden_layers, "tanh",solver)

[[15  7]
 [ 4 36]]
              precision    recall  f1-score   support

           0       0.79      0.68      0.73        22
           1       0.84      0.90      0.87        40

    accuracy                           0.82        62
   macro avg       0.81      0.79      0.80        62
weighted avg       0.82      0.82      0.82        62


hidden_layers = (7,7,7)
MLP_net(hidden_layers, "tanh",solver)

[[12 10]
 [ 3 37]]
              precision    recall  f1-score   support

           0       0.80      0.55      0.65        22
           1       0.79      0.93      0.85        40

    accuracy                           0.79        62
   macro avg       0.79      0.74      0.75        62
weighted avg       0.79      0.79      0.78        62
```

a precisão não ficou particularmente melhor ou pior. Nesse caso manteremos o numero de nodos por camadas iguais a 13.

agora aumentaremos e diminuiremos o numero de camadas.

```
hidden_layers = (13,13,13,13,13)
MLP_net(hidden_layers, "tanh",solver)

[[15  7]
 [ 2 38]]
              precision    recall  f1-score   support
```

```
              0       0.88      0.68      0.77        22
              1       0.84      0.95      0.89        40

       accuracy                           0.85        62
      macro avg       0.86      0.82      0.83        62
   weighted avg       0.86      0.85      0.85        62
```

```
hidden_layers = (13,13)
MLP_net(hidden_layers, "tanh",solver)
```

```
[[ 9 13]
 [ 1 39]]
              precision    recall  f1-score   support

              0       0.90      0.41      0.56        22
              1       0.75      0.97      0.85        40

       accuracy                           0.77        62
      macro avg       0.82      0.69      0.71        62
   weighted avg       0.80      0.77      0.75        62
```

vemos que para menos camadas a precição, recall, f1 ficam piores.

```
hidden_layers = (13,13,13,13,13,13,13,13,13,13)
MLP_net(hidden_layers, "tanh",solver)
```

```
[[ 0 22]
 [ 0 40]]
              precision    recall  f1-score   support

              0       0.00      0.00      0.00        22
              1       0.65      1.00      0.78        40

       accuracy                           0.65        62
      macro avg       0.32      0.50      0.39        62
   weighted avg       0.42      0.65      0.51        62
```

agora com dez camadas o desempenho se não se torna melhor.

por fim vamos alterar o metodo de optimazação para os pesos.

pra sgd, sthocastic gradient discent.

```
hidden_layers = (13,13,13,13,13)
MLP_net(hidden_layers, "tanh","sgd")
```

```
[[15  7]
 [ 2 38]]
```

```
              precision    recall  f1-score   support

           0       0.88      0.68      0.77        22
           1       0.84      0.95      0.89        40

    accuracy                           0.85        62
   macro avg       0.86      0.82      0.83        62
weighted avg       0.86      0.85      0.85        62
```

para lbfgs

```
MLP_net(hidden_layers, "tanh","lbfgs")

[[12 10]
 [ 3 37]]
              precision    recall  f1-score   support

           0       0.80      0.55      0.65        22
           1       0.79      0.93      0.85        40

    accuracy                           0.79        62
   macro avg       0.79      0.74      0.75        62
weighted avg       0.79      0.79      0.78        62
```

podemos concluir que para esta base de dados específica os parâmetros para gerar os melhores resultados por precisão e acurácia foram: ativação por tangente hiperbólica, cinco camadas intermediarias com 13 nodos cada e optmização por adam.

agora vamos alterar o tamanho dedicado a testes e treinamento e ver a influência deles na acurácia e precisão.

```
target_column = ['Att']
predictors = list(set(list(df.columns))-set(target_column))
df[predictors] = df[predictors]/df[predictors].max()

X = df[predictors].values
y = df[target_column].values

def MLP_train(test):
  print(test)
  X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=test, random_state=40)
  MLP_net((13,13,13,13,13), "tanh","sgd")


for i in range(19):
  MLP_train(0.5 - i*0.025)
```

```
0.5
              precision    recall  f1-score   support

           0       0.88      0.64      0.74        22
           1       0.83      0.95      0.88        40

    accuracy                           0.84        62
   macro avg       0.85      0.79      0.81        62
weighted avg       0.84      0.84      0.83        62

0.475
              precision    recall  f1-score   support

           0       0.88      0.68      0.77        22
           1       0.84      0.95      0.89        40

    accuracy                           0.85        62
   macro avg       0.86      0.82      0.83        62
weighted avg       0.86      0.85      0.85        62

0.45
              precision    recall  f1-score   support

           0       0.88      0.64      0.74        22
           1       0.83      0.95      0.88        40

    accuracy                           0.84        62
   macro avg       0.85      0.79      0.81        62
weighted avg       0.84      0.84      0.83        62

0.425
              precision    recall  f1-score   support

           0       0.83      0.91      0.87        22
           1       0.95      0.90      0.92        40

    accuracy                           0.90        62
   macro avg       0.89      0.90      0.90        62
weighted avg       0.91      0.90      0.90        62

0.4
              precision    recall  f1-score   support

           0       0.78      0.64      0.70        22
           1       0.82      0.90      0.86        40

    accuracy                           0.81        62
   macro avg       0.80      0.77      0.78        62
weighted avg       0.80      0.81      0.80        62

0.375
```

```
              precision    recall  f1-score   support

           0       0.79      0.68      0.73        22
           1       0.84      0.90      0.87        40

    accuracy                           0.82        62
   macro avg       0.81      0.79      0.80        62
weighted avg       0.82      0.82      0.82        62

0.35
              precision    recall  f1-score   support

           0       0.93      0.59      0.72        22
           1       0.81      0.97      0.89        40

    accuracy                           0.84        62
   macro avg       0.87      0.78      0.80        62
weighted avg       0.85      0.84      0.83        62

0.32499999999999996
              precision    recall  f1-score   support

           0       0.84      0.73      0.78        22
           1       0.86      0.93      0.89        40

    accuracy                           0.85        62
   macro avg       0.85      0.83      0.84        62
weighted avg       0.85      0.85      0.85        62

0.3
              precision    recall  f1-score   support

           0       0.93      0.64      0.76        22
           1       0.83      0.97      0.90        40

    accuracy                           0.85        62
   macro avg       0.88      0.81      0.83        62
weighted avg       0.87      0.85      0.85        62

0.275
              precision    recall  f1-score   support

           0       0.83      0.68      0.75        22
           1       0.84      0.93      0.88        40

    accuracy                           0.84        62
   macro avg       0.84      0.80      0.82        62
weighted avg       0.84      0.84      0.83        62

0.25
              precision    recall  f1-score   support
```

```
           0       0.83      0.68      0.75        22
           1       0.84      0.93      0.88        40

    accuracy                           0.84        62
   macro avg       0.84      0.80      0.82        62
weighted avg       0.84      0.84      0.83        62

0.22499999999999998
              precision    recall  f1-score   support

           0       0.93      0.64      0.76        22
           1       0.83      0.97      0.90        40

    accuracy                           0.85        62
   macro avg       0.88      0.81      0.83        62
weighted avg       0.87      0.85      0.85        62

0.19999999999999996
              precision    recall  f1-score   support

           0       0.88      0.64      0.74        22
           1       0.83      0.95      0.88        40

    accuracy                           0.84        62
   macro avg       0.85      0.79      0.81        62
weighted avg       0.84      0.84      0.83        62

0.175
              precision    recall  f1-score   support

           0       0.88      0.68      0.77        22
           1       0.84      0.95      0.89        40

    accuracy                           0.85        62
   macro avg       0.86      0.82      0.83        62
weighted avg       0.86      0.85      0.85        62

0.14999999999999997
              precision    recall  f1-score   support

           0       0.89      0.73      0.80        22
           1       0.86      0.95      0.90        40

    accuracy                           0.87        62
   macro avg       0.88      0.84      0.85        62
weighted avg       0.87      0.87      0.87        62

0.125
              precision    recall  f1-score   support
```

```
              0       0.88      0.68      0.77        22
              1       0.84      0.95      0.89        40

    accuracy                          0.85        62
   macro avg       0.86      0.82      0.83        62
weighted avg       0.86      0.85      0.85        62

0.09999999999999998
              precision    recall  f1-score   support

              0       0.79      0.68      0.73        22
              1       0.84      0.90      0.87        40

    accuracy                          0.82        62
   macro avg       0.81      0.79      0.80        62
weighted avg       0.82      0.82      0.82        62

0.07499999999999996
              precision    recall  f1-score   support

              0       0.76      0.73      0.74        22
              1       0.85      0.88      0.86        40

    accuracy                          0.82        62
   macro avg       0.81      0.80      0.80        62
weighted avg       0.82      0.82      0.82        62

0.04999999999999999
              precision    recall  f1-score   support

              0       0.81      0.59      0.68        22
              1       0.80      0.93      0.86        40

    accuracy                          0.81        62
   macro avg       0.81      0.76      0.77        62
weighted avg       0.81      0.81      0.80        62
```

para esta configuração as casos de proporção testes/treino mais interessantes foram

% testes	precisão 0	precisão 1	acurácia
47,9%	0.88	0.84	0.85
42,9%	0.83	0.95	0.90
30%	0.93	0.85	0.85
22,4%	0.93	0.83	0.85
17,5%	0.88	0.84	0.85
15%	0.89	0.86	0.87
10,5%	0.79	0.84	0.82

originalmente a proporção para casos de testes era de 20%. Vemos que uma proporção de 42,9% gera resultados tão ou mais precisos que aqueles com proporção menor para testes.

Podemos concluir que ou há algo de errado com nossa base de dados pois treino com menos casos se provam tão ou mais precisos que com mais casos, ou a partir de um certo limiar começa a ocorrer overfitting em nossa rede e ela passa a errar mais do que acertar

conclusão

Usamos uma rede neural MLP para prever a classificação dores lombares. Bem como exploramos os dados e fizemos experimentos para estabelecer os melhores parâmetros da nossa rede.