Problem5

February 4, 2020

[36]:
```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns

feature_labels = np.arange(280)
row_labels = np.arange(452)

# Import PatientData.csv with Pandas and show the data
data = pd.read_csv('PatientData.csv', names=feature_labels, index_col=False)

print(data)
print(data.index)
```

```
     0    1    2    3    4    5    6    7    8    9    …  270   271   272  \
0    75    0  190   80   91  193  371  174  121  -16  …  0.0   9.0  -0.9
1    56    1  165   64   81  174  401  149   39   25  …  0.0   8.5   0.0
2    54    0  172   95  138  163  386  185  102   96  …  0.0   9.5  -2.4
3    55    0  175   94  100  202  380  179  143   28  …  0.0  12.2  -2.2
4    75    0  190   80   88  181  360  177  103  -16  …  0.0  13.1  -3.6
..   …  …  …  …  …  …  …  …  …  …  …  …  …   …   …
447  53    1  160   70   80  199  382  154  117  -37  …  0.0   4.3  -5.0
448  37    0  190   85  100  137  361  201   73   86  …  0.0  15.6  -1.6
449  36    0  166   68  108  176  365  194  116  -85  …  0.0  16.3 -28.6
450  32    1  155   55   93  106  386  218   63   54  … -0.4  12.0  -0.7
451  78    1  160   70   79  127  364  138   78   28  …  0.0  10.4  -1.8

     273  274  275  276   277   278  279
0    0.0  0.0  0.9  2.9  23.3  49.4    8
1    0.0  0.0  0.2  2.1  20.4  38.8    6
2    0.0  0.0  0.3  3.4  12.3  49.0   10
3    0.0  0.0  0.4  2.6  34.6  61.6    1
4    0.0  0.0 -0.1  3.9  25.4  62.8    7
..   …  …  …  …   …   …  …
447  0.0  0.0  0.7  0.6  -4.4  -0.5    1
448  0.0  0.0  0.4  2.4  38.0  62.4   10
449  0.0  0.0  1.5  1.0 -44.2 -33.2    2
450  0.0  0.0  0.5  2.4  25.0  46.6    1
```

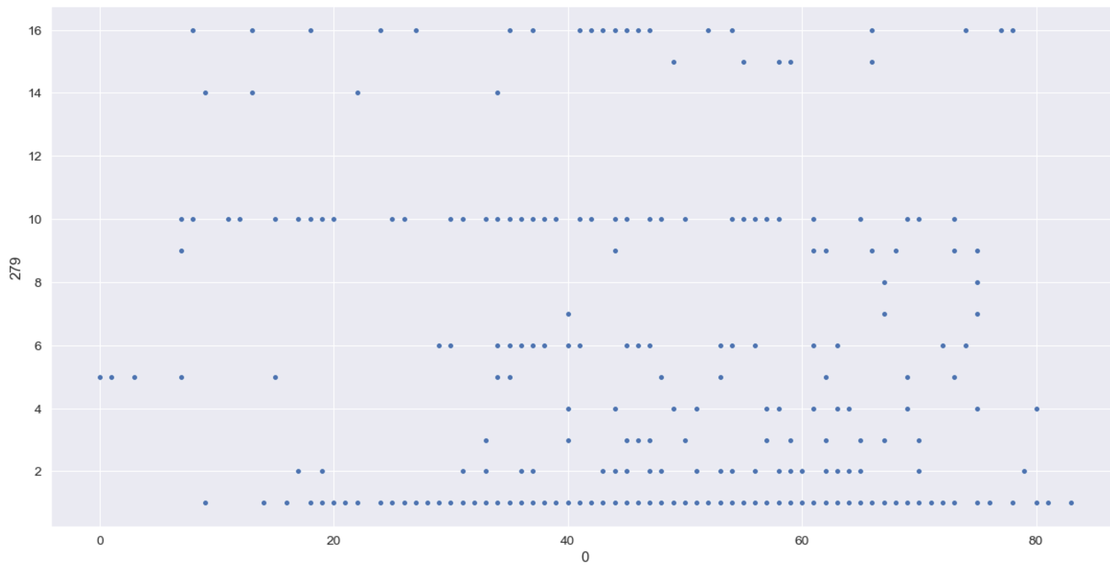
```
451  0.0  0.0  0.5  1.6  21.3  32.8    1

[452 rows x 280 columns]
RangeIndex(start=0, stop=452, step=1)
```

(a) There are 452 patients and 279 features.

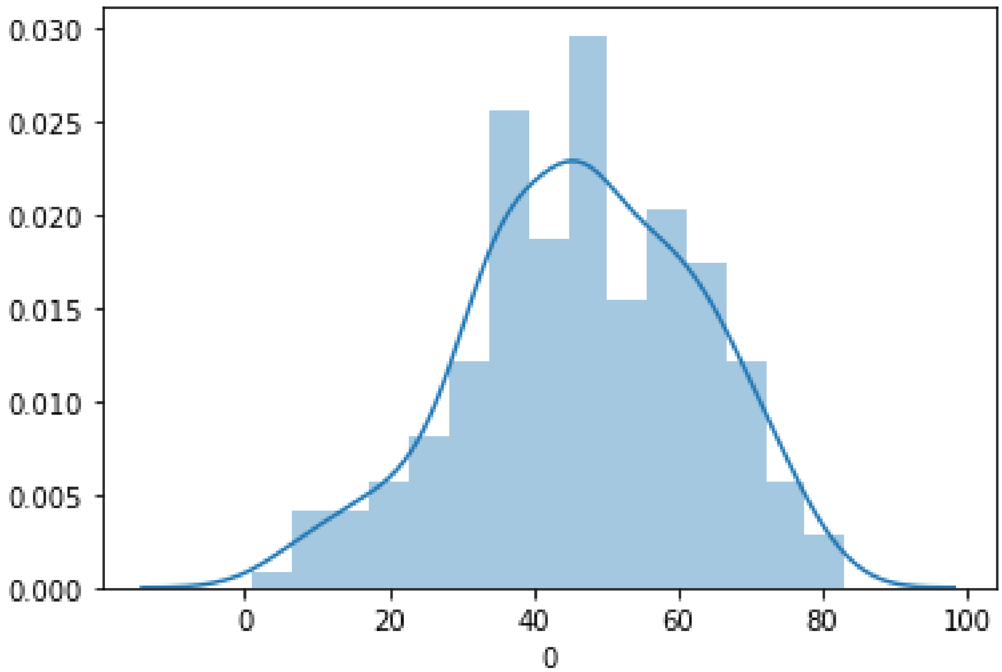
[175]:
```
fig, ax = plt.subplots(figsize=(20,10))
sns.scatterplot(data[0], data[279])
np.mean(data[0])
```

[175]:
```
46.4712389380531
```



[14]:
```
sns.distplot(data[0])
```

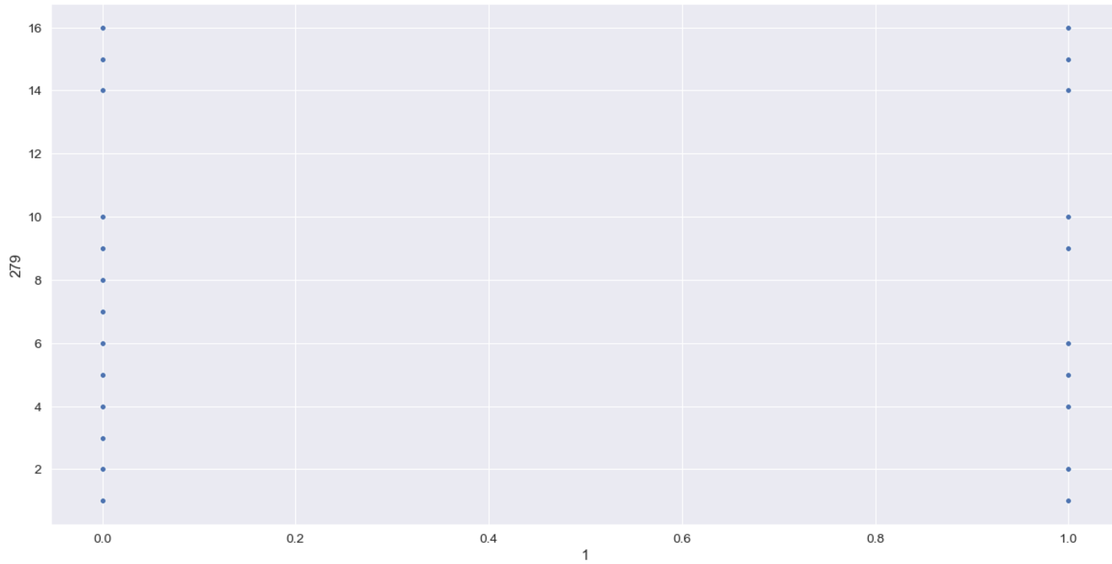
[14]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a26f2bbd0>
```



This is the first feature of the dataframe. Here we can see that this feature has a mean of 46.4. This seems to resemble the average age of patients that are admitted into a health facility and sampled. Also we can see that different conditions have different age distributions, which may be a result of dissimilar diseases/conditions.

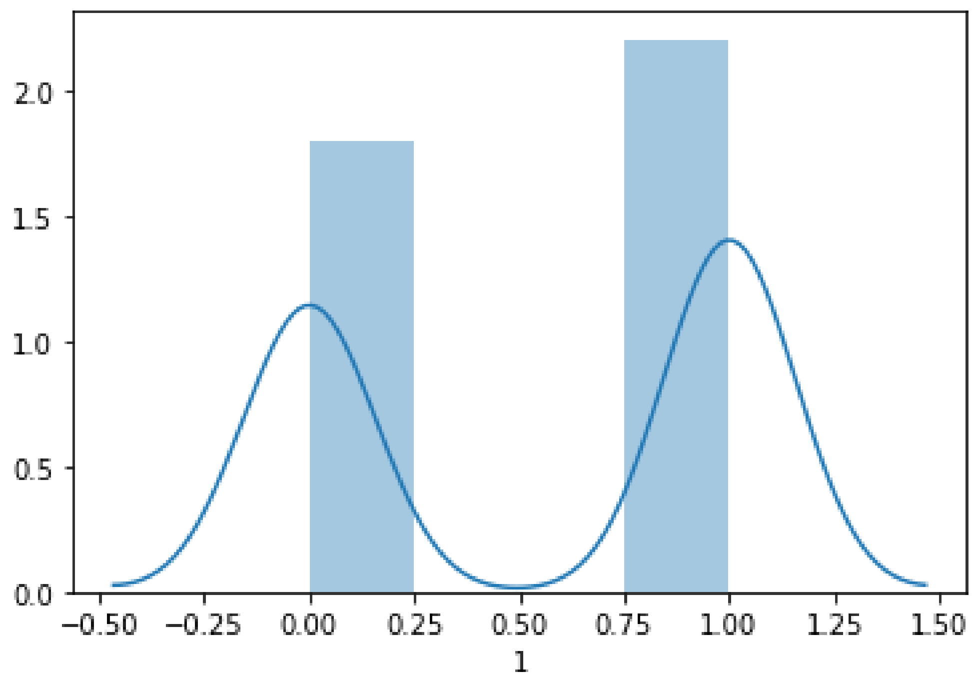
[176]:
```
fig, ax = plt.subplots(figsize=(20,10))
sns.scatterplot(data[1], data[279])
```

[176]: <matplotlib.axes._subplots.AxesSubplot at 0x1a273976d0>



[13]:
```
sns.distplot(data[1])
```

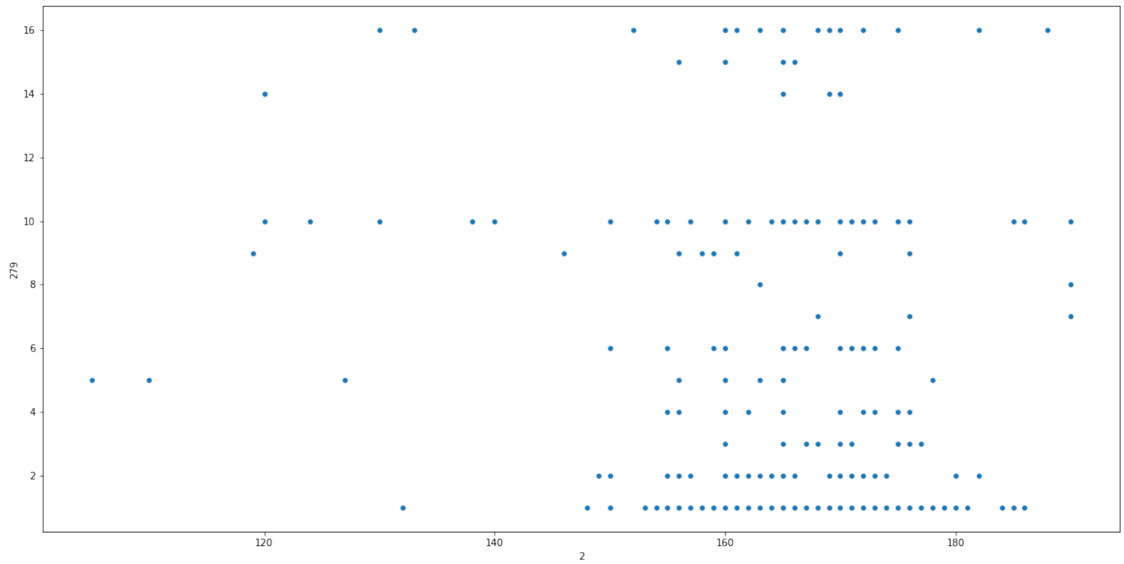
[13]: <matplotlib.axes._subplots.AxesSubplot at 0x1a26ca5710>



For this feature, we see that it is binary. Since this is patient data, it is most likely representative of the patient's sex (probably only binary feature).

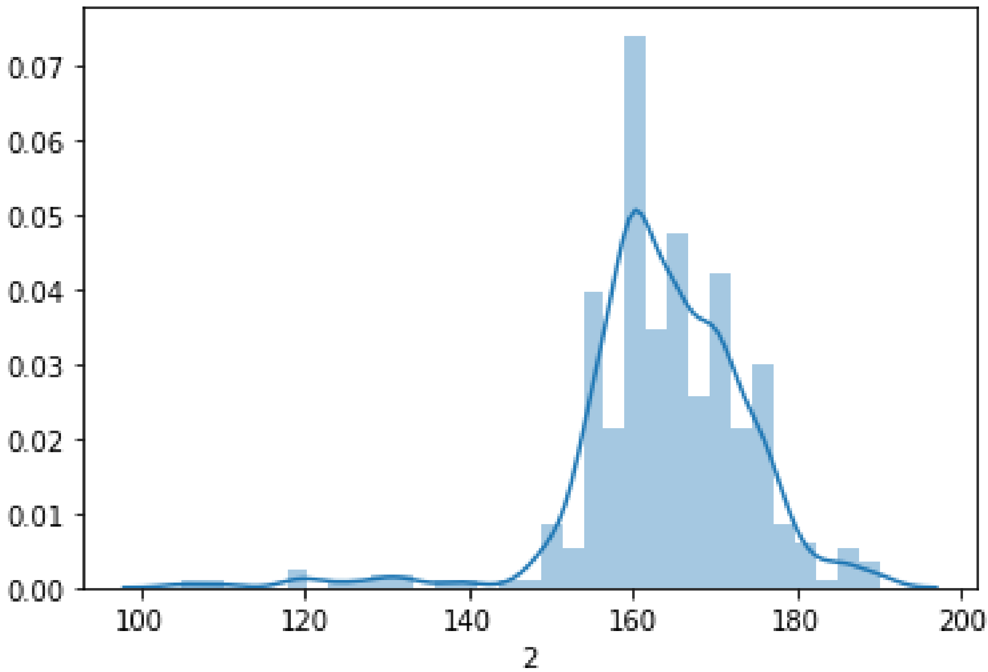
[7]:
```
fig, ax = plt.subplots(figsize=(20,10))
data = data[data[2] < 200]
sns.scatterplot(data[2], data[279])
np.mean(data[2])
```

[7]: 163.84222222222223



[8]:
```
sns.distplot(data[2])
```

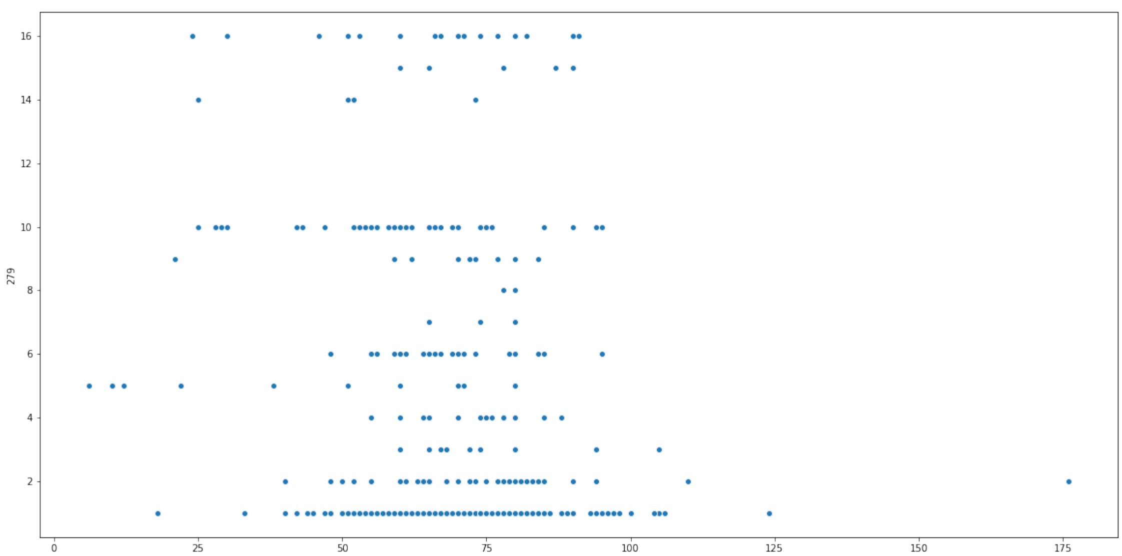
[8]: <matplotlib.axes._subplots.AxesSubplot at 0x1a26bd6b50>



With the third feature, we have come across a mean of 163.7. We believe this is representative of the patient's weight.

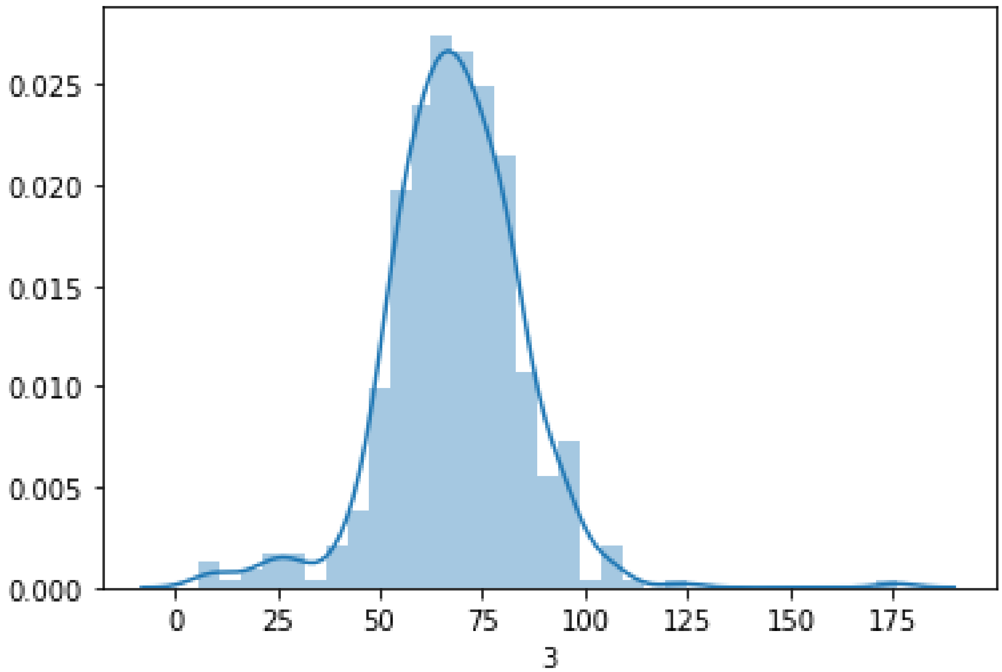
[3]:
```
fig, ax = plt.subplots(figsize=(20,10))
sns.scatterplot(data[3], data[279])
np.mean(data[3])
```

[3]: 68.17035398230088



[5]:
```python
sns.distplot(data[3])
```

[5]: <matplotlib.axes._subplots.AxesSubplot at 0x1a26745510>



In our last feature, we get a mean of 68. This is very similar to the average resting BPM of a human. Therefore, we have reason to believe this represents the resting BPM of the patient.

[37]:
```python
# data.replace('?', np.nan, inplace=True)

for column in data.columns:

    sum1, count = 0.0, 0
    for row in data.index:

        if (data.loc[row,column] != '?'):
            sum1 += float(data.loc[row,column])
            count += 1

    if (count > 0):
        column_mean = sum1/count
    else:
        column_mean = 0
```

```
    print(column_mean)

    for row in data.index:
        if ((data.loc[row,column]) == '?'):
            data.loc[row,column] = column_mean
```

```
46.4712389380531
0.5508849557522124
166.18805309734512
68.17035398230088
88.92035398230088
155.15265486725664
367.2079646017699
169.94911504424778
90.00442477876106
33.676991150442475
36.1509009009009
48.913953488372094
36.71618625277162
-13.592105263157896
74.46341463414635
5.628318584070796
51.6283185840708
20.920353982300885
0.1415929203539823
0.0
30.035398230088497
0.0022123893805309734
0.011061946902654867
0.011061946902654867
0.004424778761061947
0.004424778761061947
0.008849557522123894
5.619469026548672
54.336283185840706
20.5929203539823
0.4336283185840708
0.1504424778761062
31.63716814159292
0.017699115044247787
0.028761061946902654
0.0022123893805309734
0.004424778761061947
0.004424778761061947
0.015486725663716814
16.02654867256637
41.982300884955755
```

20.327433628318584
2.3008849557522124
0.3185840707964602
30.513274336283185
0.0022123893805309734
0.035398230088495575
0.0022123893805309734
0.017699115044247787
0.011061946902654867
0.004424778761061947
45.36283185840708
19.327433628318584
7.79646017699115
2.8230088495575223
0.07079646017699115
31.23008849557522
0.011061946902654867
0.004424778761061947
0.004424778761061947
0.004424778761061947
0.004424778761061947
0.008849557522123894
10.274336283185841
43.575221238938056
19.84070796460177
0.8141592920353983
0.0
27.300884955752213
0.0
0.017699115044247787
0.0022123893805309734
0.0022123893805309734
0.00663716814159292
0.0022123893805309734
7.477876106194691
50.4070796460177
19.79646017699115
0.7699115044247787
0.22123893805309736
29.876106194690266
0.004424778761061947
0.024336283185840708
0.0
0.0022123893805309734
0.0022123893805309734
0.0022123893805309734
12.601769911504425
23.84070796460177

42.123893805309734
3.9911504424778763
0.11504424778761062
18.72566371681416
0.00663716814159292
0.015486725663716814
0.008849557522123894
0.008849557522123894
0.011061946902654867
0.022123893805309734
6.327433628318584
33.610619469026545
43.610619469026545
2.0353982300884956
0.17699115044247787
22.628318584070797
0.008849557522123894
0.008849557522123894
0.01327433628318584
0.008849557522123894
0.008849557522123894
0.017699115044247787
3.814159292035398
42.46017699115044
41.68141592920354
0.5398230088495575
0.13274336283185842
27.734513274336283
0.024336283185840708
0.004424778761061947
0.00663716814159292
0.008849557522123894
0.00663716814159292
0.011061946902654867
3.2389380530973453
46.07964601769911
42.415929203539825
0.5221238938053098
0.12389380530973451
31.04424778761062
0.004424778761061947
0.01327433628318584
0.0
0.0
0.004424778761061947
0.017699115044247787
4.946902654867257
46.91150442477876

39.94690265486726
0.2831858407079646
0.0
31.964601769911503
0.0
0.00663716814159292
0.0
0.0022123893805309734
0.0
0.00663716814159292
6.716814159292035
50.23893805309734
28.309734513274336
0.1504424778761062
0.0
32.16814159292036
0.0022123893805309734
0.0022123893805309734
0.004424778761061947
0.0
0.0
0.011061946902654867
-0.2070796460176991
-0.1929203539823009
6.013053097345132
-1.0256637168141594
0.006858407079646017
0.0
0.6471238938053094
0.9853982300884955
13.84424778761062
20.818362831858412
-0.1212389380530973
-0.2482300884955753
7.169247787610619
-1.3347345132743362
0.019469026548672573
-0.005973451327433629
0.9820796460176985
1.375
16.955530973451307
26.95862831858406
0.0794247787610619
-1.0059734513274334
3.492256637168138
-1.7396017699115043
0.15398230088495574
-0.01327433628318584

0.42765486725663726
0.35287610619469034
2.6707964601769913
5.590044247787613
0.14712389380530966
-5.234070796460175
0.9000000000000006
-1.1469026548672572
0.11283185840707964
-0.0008849557522123895
-0.767256637168142
-1.1438053097345127
-15.630752212389384
-23.49911504424778
-0.1553097345132743
-0.45398230088495567
3.4495575221238948
-1.240929203539823
0.02809734513274336
0.0
0.10265486725663711
0.30154867256637186
5.450221238938053
7.279424778761061
-0.0011061946902654772
-0.36371681415929213
4.86216814159292
-1.3179203539823008
0.05132743362831858
-0.019247787610619467
0.6803097345132747
0.8681415929203541
9.776106194690259
16.01836283185841
0.6592920353982301
-1.4201327433628315
1.6336283185840712
-6.554646017699112
0.31769911504424786
-0.008849557522123894
-0.3305309734513277
0.17610619469026575
-18.738495575221247
-15.881194690265483
0.9639380530973454
-0.9143805309734514
3.9778761061946906
-9.048893805309739

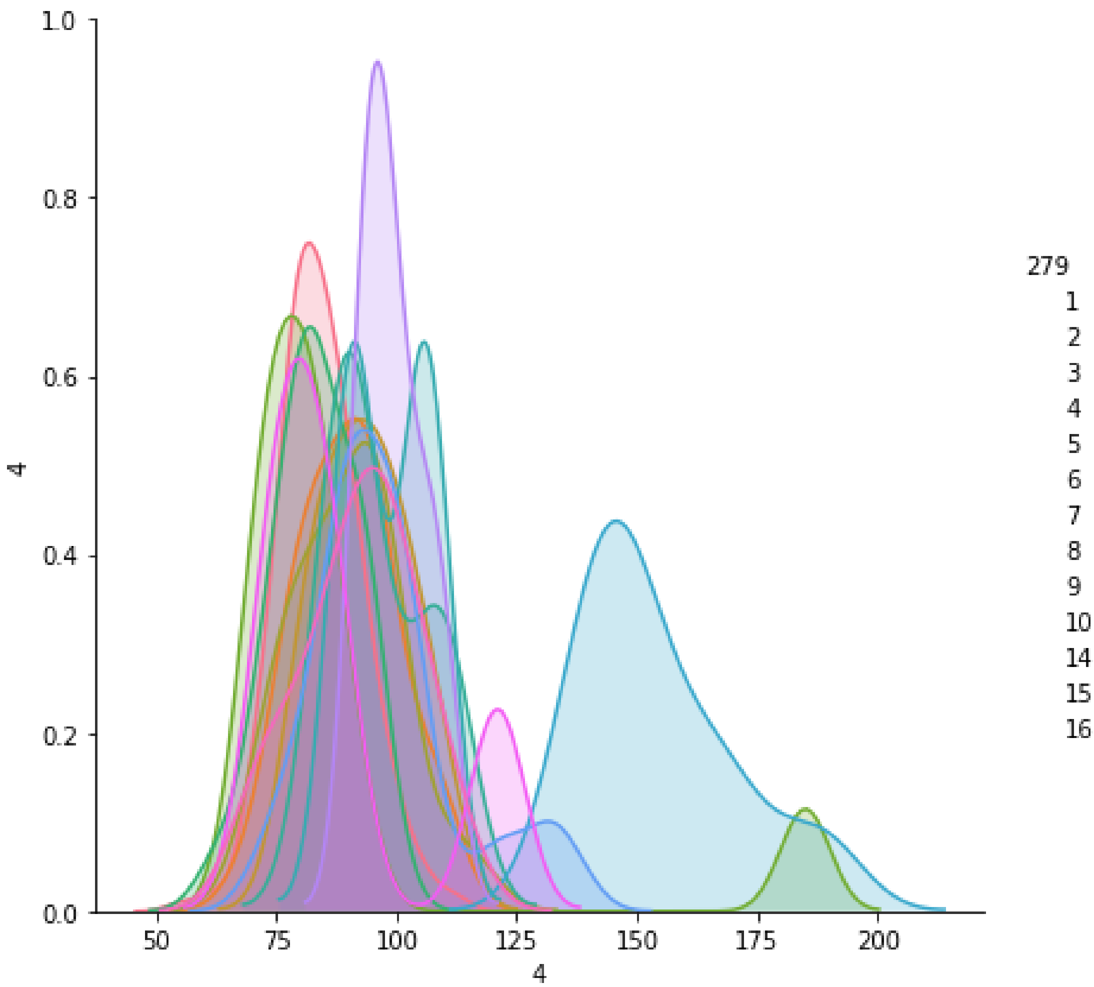
0.18141592920353983
-0.01592920353982301
0.0015486725663716929
2.6179203539822997
-17.982743362831858
10.245796460176983
0.7681415929203542
-0.6535398230088495
8.039601769911506
-10.150663716814176
0.032964601769911506
-0.013495575221238938
0.22676991150442485
3.8946902654867217
-8.269026548672567
32.42278761061948
0.0011061946902654759
-0.297566371681416
11.839380530973449
-7.034513274336288
0.025663716814159295
-0.0028761061946902654
0.5477876106194688
2.535840707964603
10.081194690265495
33.328539823008846
-0.28539823008849574
-0.277212389380531
11.369911504424776
-3.6075221238938058
0.016814159292035398
0.0
0.5466814159292036
1.7221238938053105
17.8400442477876
32.87146017699117
-0.30243362831858467
-0.27898230088495585
9.048008849557515
-1.4573008849557536
0.003982300884955752
0.0
0.5148230088495577
1.2223451327433625
19.32610619469028
29.47323008849559
3.8805309734513274

Here we have replaced all of the missing values with their correspsonding column mean.

In order to determine feature importance, we need to look at how specific features influence the classification of a data entry. Finding correlation here wouldn't really work because this is a classification task. We can plot the relationship as shown below between a few features and conditions. The features that will have the biggest impact will be those that offer clear distinctions between conditions.

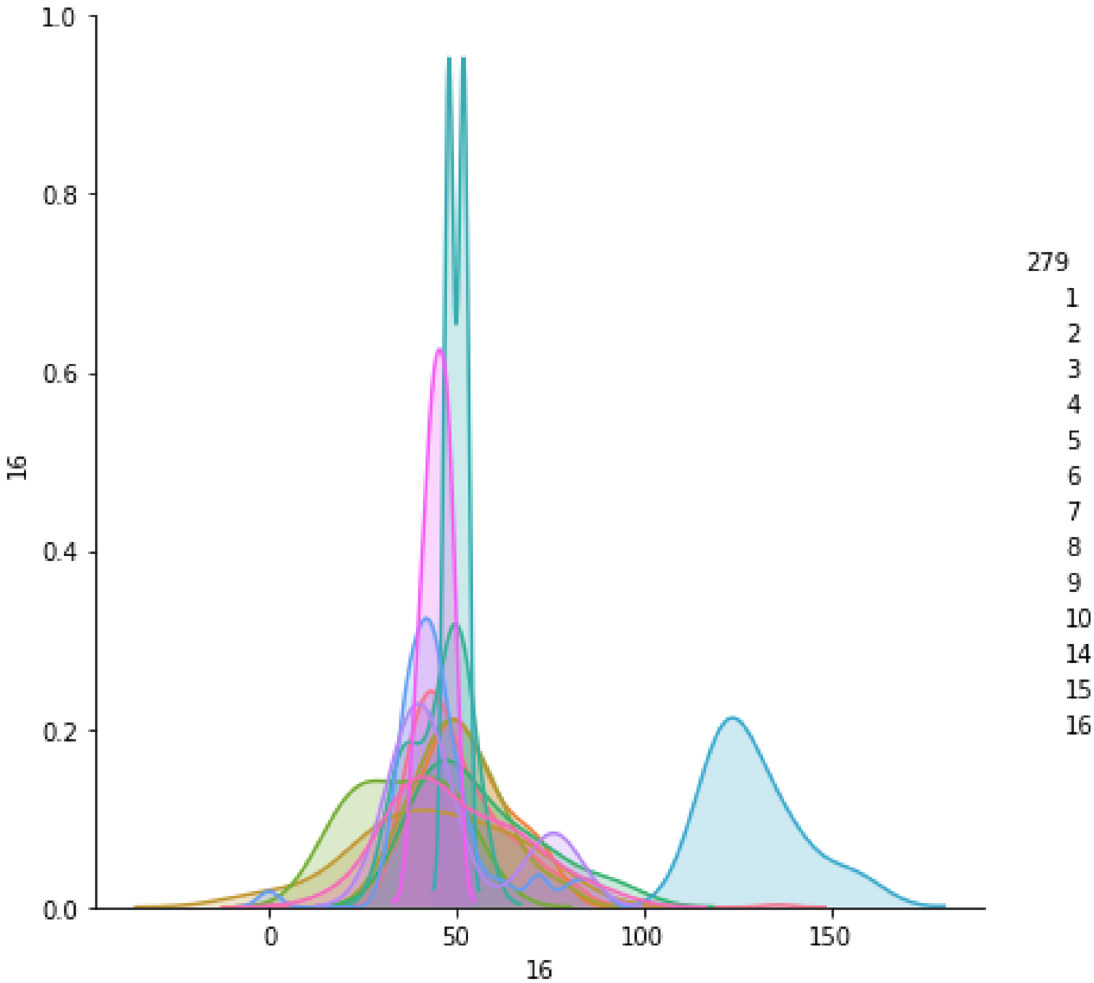
[32]:
```
sns.pairplot(data=data, vars=[4], hue=279, height = 6)
```

[32]: <seaborn.axisgrid.PairGrid at 0x1a284dba50>



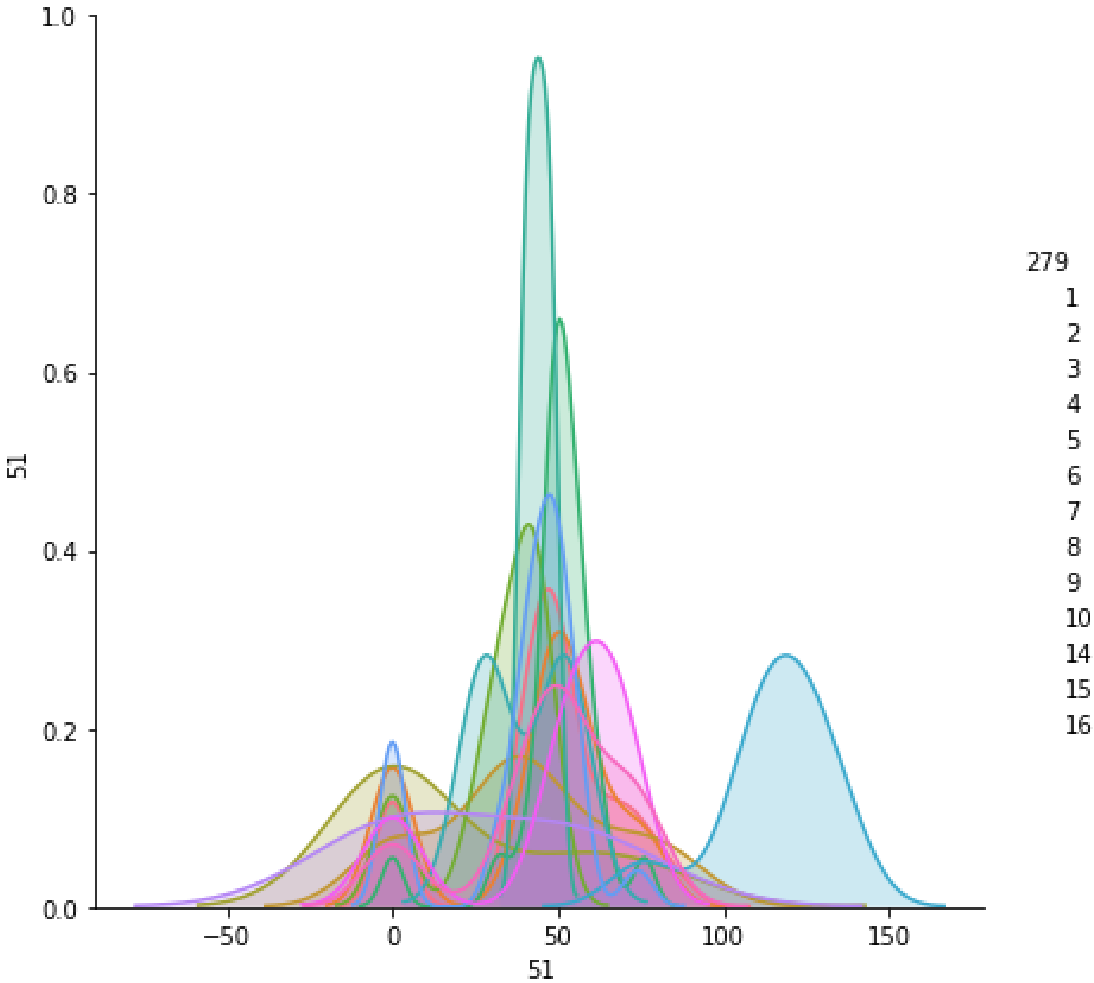
[40]:
```
sns.pairplot(data=data, vars= [16], hue=279, height = 6)
```

[40]: <seaborn.axisgrid.PairGrid at 0x1a39bb0090>



[56]:
```
sns.pairplot(data=data, vars= [51], hue=279, height = 6)
```

[56]: <seaborn.axisgrid.PairGrid at 0x1a5a316e90>



These three features above show some pretty clean delineations between classification classes, which indicate how import the value is to predicting the patient's condition. Therefore, feature 4, 16, 51 were are most important features in our short data exploration.

```
[ ]:
```