EDA - Assessment to be continued

February 28, 2020

1 Exercise:

1. Download Haberman Cancer Survival dataset from Kaggle. You may have to create a Kaggle account to donwload data. (https://www.kaggle.com/gilsousa/habermans-survival-data-set)
2. Perform a similar alanlaysis as above on this dataset with the following sections:

- High level statistics of the dataset: number of points, numer of features, number of classes, data-points per class.
- Explain our objective.
- Perform Univaraite analysis(PDF, CDF, Boxplot, Voilin plots) to understand which features are useful towards classification.
- Perform Bi-variate analysis (scatter plots, pair-plots) to see if combinations of features are useful in classfication.
- Write your observations in english as crisply and unambigously as possible. Always quantify your results.

2 1. Download Haberman Cancer Survival dataset from Kaggle. You may have to create a Kaggle account to donwload data. (https://www.kaggle.com/gilsousa/habermans-survival-data-set)

In [2]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np


haberman = pd.read_csv(r"C:\Users\ASUS\Downloads\Applied AI\Assignments - Applied AI\Da
haberman
```

Out[2]:

	age	year	nodes	status
0	30	64	1	1
1	30	62	3	1
2	30	65	0	1
3	31	59	2	1
4	31	65	4	1

5	33	58	10	1
6	33	60	0	1
7	34	59	0	2
8	34	66	9	2
9	34	58	30	1
10	34	60	1	1
11	34	61	10	1
12	34	67	7	1
13	34	60	0	1
14	35	64	13	1
15	35	63	0	1
16	36	60	1	1
17	36	69	0	1
18	37	60	0	1
19	37	63	0	1
20	37	58	0	1
21	37	59	6	1
22	37	60	15	1
23	37	63	0	1
24	38	69	21	2
25	38	59	2	1
26	38	60	0	1
27	38	60	0	1
28	38	62	3	1
29	38	64	1	1
..	...	...	...	...
276	67	66	0	1
277	67	61	0	1
278	67	65	0	1
279	68	67	0	1
280	68	68	0	1
281	69	67	8	2
282	69	60	0	1
283	69	65	0	1
284	69	66	0	1
285	70	58	0	2
286	70	58	4	2
287	70	66	14	1
288	70	67	0	1
289	70	68	0	1
290	70	59	8	1
291	70	63	0	1
292	71	68	2	1
293	72	63	0	2
294	72	58	0	1
295	72	64	0	1
296	72	67	3	1
297	73	62	0	1

```
298   73    68      0       1
299   74    65      3       2
300   74    63      0       1
301   75    62      1       1
302   76    67      0       1
303   77    65      3       1
304   78    65      1       2
305   83    58      2       2

[306 rows x 4 columns]
```

3 2. Perform a similar alanlaysis as above on this dataset with the following sections:

a) High level statistics of the dataset: number of points, numer of features, number of classes, data-points per class.

```
In [3]: # (Q) how many data-points and features?
        print (haberman.shape)
```

```
(306, 4)
```

```
In [4]: haberman.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 306 entries, 0 to 305
Data columns (total 4 columns):
age       306 non-null int64
year      306 non-null int64
nodes     306 non-null int64
status    306 non-null int64
dtypes: int64(4)
memory usage: 9.6 KB
```

```
In [5]: haberman.head()
```

```
Out[5]:    age  year  nodes  status
        0   30    64      1       1
        1   30    62      3       1
        2   30    65      0       1
        3   31    59      2       1
        4   31    65      4       1
```

```
In [6]: haberman.describe()
```

```
Out[6]:              age        year       nodes      status
        count  306.000000  306.000000  306.000000  306.000000
        mean    52.457516   62.852941    4.026144    1.264706
        std     10.803452    3.249405    7.189654    0.441899
        min     30.000000   58.000000    0.000000    1.000000
        25%     44.000000   60.000000    0.000000    1.000000
        50%     52.000000   63.000000    1.000000    1.000000
        75%     60.750000   65.750000    4.000000    2.000000
        max     83.000000   69.000000   52.000000    2.000000

In [7]: haberman["status"].value_counts() #number of classes and number of datapoints per clas

Out[7]: 1    225
        2     81
        Name: status, dtype: int64

In [8]: haberman.columns

Out[8]: Index(['age', 'year', 'nodes', 'status'], dtype='object')
```

4 2. Perform a similar alanlaysis as above on this dataset with the following sections:

b) Explain our objective.

Obective : classify the persons who are survived after the surgery and who are not.

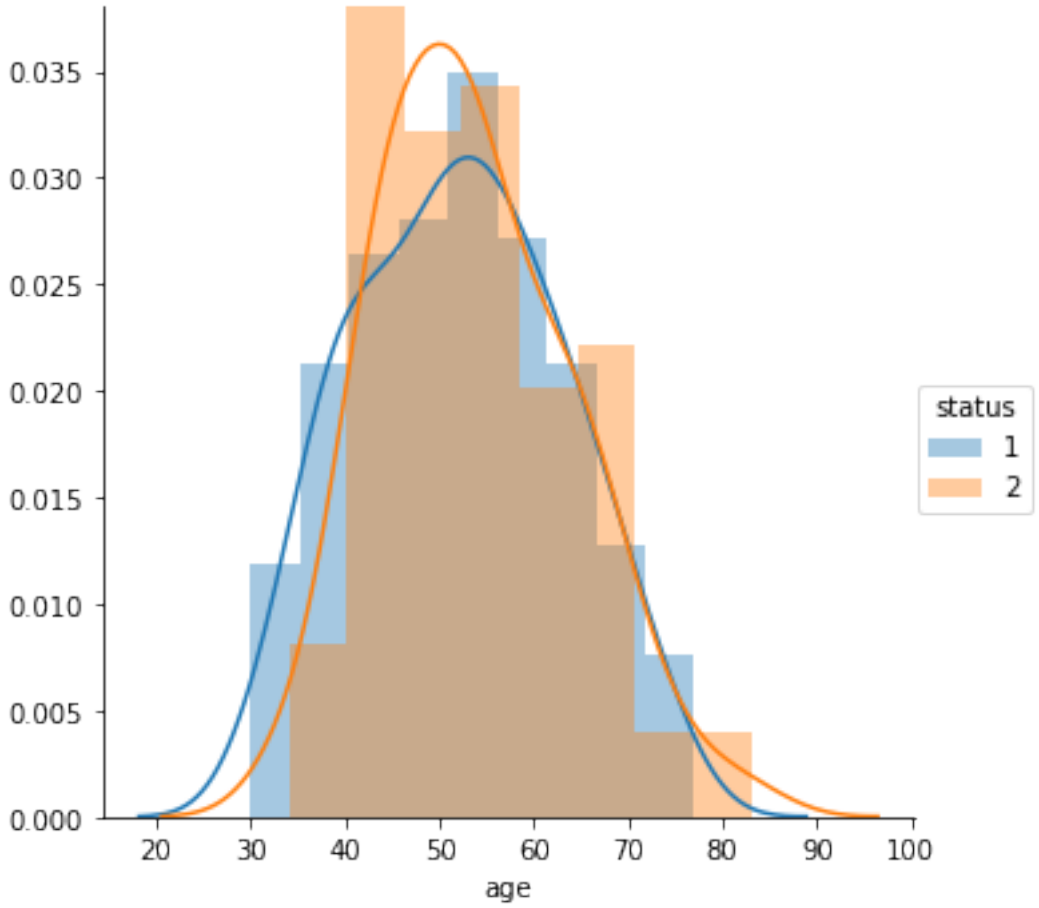
since, I was looking for more insights on the data, there was it poped #https://www.kaggle.com/vj1998/haberman-s-survival-exploratory-data-analysis. I just took the objective from it, rest I had gone through the sample code of EDA and from some other seaborn datasets for graph.

5 2. Perform a similar alanlaysis as above on this dataset with the following sections:

c) Perform Univaraite analysis(PDF, CDF, Boxplot, Voilin plots) to understand which features are useful towards classification.

Histogram

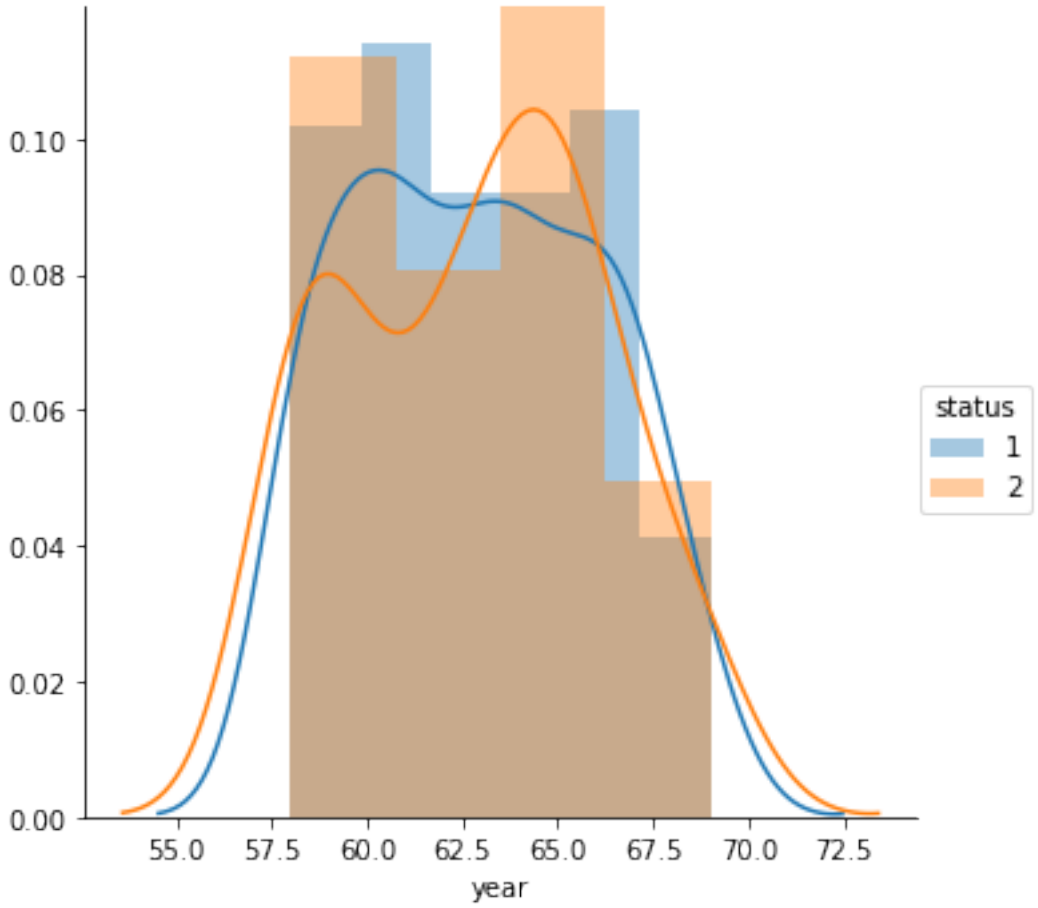
```
In [9]: import warnings # Current version of Seaborn generates a bunch of warnings that we'll
        warnings.filterwarnings("ignore")

In [12]: sns.FacetGrid(haberman, hue="status", size=5) \
             .map(sns.distplot, "age") \
             .add_legend();
         plt.show();
```




```
In [ ]: sns.FacetGrid(iris, hue="status", size=5) \
           .map(sns.distplot, "nodes") \
           .add_legend();
        plt.show();
```

Observation : From this above figure, we can come to an assumption that as the number of nodes are less chances of survival is more

```
In [13]: sns.FacetGrid(haberman, hue="status", size=5) \
            .map(sns.distplot, "year") \
            .add_legend();
         plt.show();
```



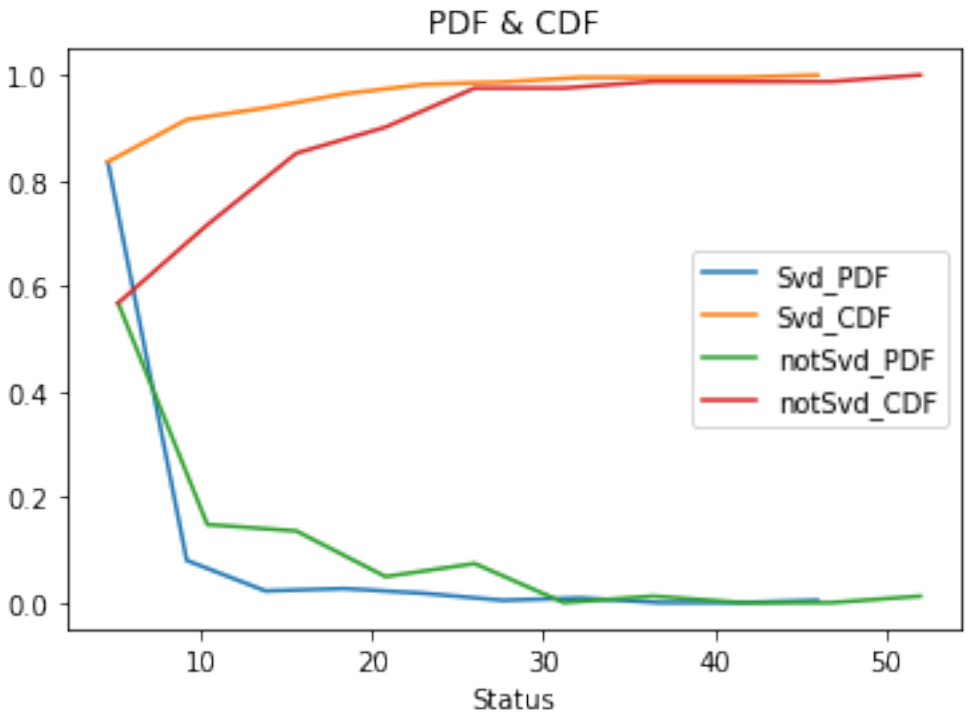
Obervations from Histogram : figure 2 gives us information on the dataset, rest of the figures are quite complex to interpret the information.

PDF & CDF

```
In [14]: import numpy as np
         haberman_svd = haberman.loc[haberman["status"] == 1];
         haberman_notsvd = haberman.loc[haberman["status"] == 2];

         # Survived.

         counts, bin_edges = np.histogram(haberman_svd['nodes'], bins=10,
                                          density = True)
         pdf = counts/(sum(counts))
         print(pdf);
         print(bin_edges)
         cdf = np.cumsum(pdf)
         plt.plot(bin_edges[1:],pdf)
         plt.plot(bin_edges[1:], cdf)
```

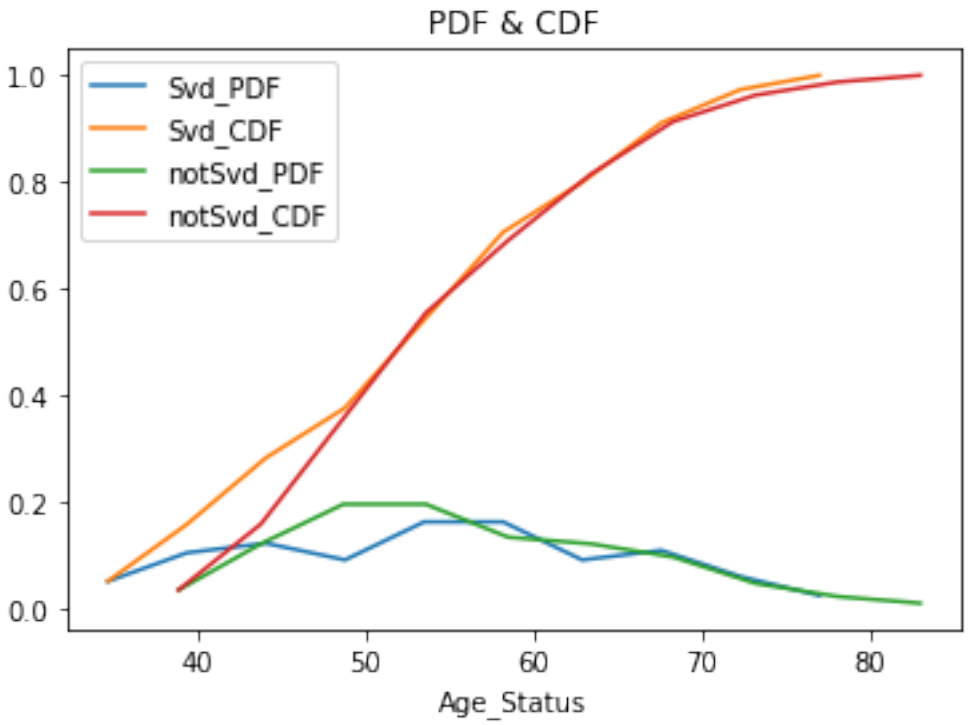
```
        # Not survived
        counts, bin_edges = np.histogram(haberman_notsvd['nodes'], bins=10,
                                         density = True)
        pdf = counts/(sum(counts))
        print(pdf);
        print(bin_edges)
        cdf = np.cumsum(pdf)
        plt.plot(bin_edges[1:],pdf)
        plt.plot(bin_edges[1:], cdf)


        plt.title('PDF & CDF')
        #plt.legend('nodes_status')
        plt.legend(['Svd_PDF', 'Svd_CDF', 'notSvd_PDF', 'notSvd_CDF'])
        plt.xlabel("Status")
        plt.show();
```

```
[0.83555556 0.08       0.02222222 0.02666667 0.01777778 0.00444444
 0.00888889 0.         0.         0.00444444]
[ 0.   4.6  9.2 13.8 18.4 23.  27.6 32.2 36.8 41.4 46. ]
[0.56790123 0.14814815 0.13580247 0.04938272 0.07407407 0.
 0.01234568 0.         0.         0.01234568]
[ 0.   5.2 10.4 15.6 20.8 26.  31.2 36.4 41.6 46.8 52. ]
```



In [15]:
```
import numpy as np
haberman_svd = haberman.loc[haberman["status"] == 1];
haberman_notsvd = haberman.loc[haberman["status"] == 2];

# Survived.

counts, bin_edges = np.histogram(haberman_svd['age'], bins=10,
                                 density = True)
pdf = counts/(sum(counts))
print(pdf);
print(bin_edges)
cdf = np.cumsum(pdf)
plt.plot(bin_edges[1:],pdf)
plt.plot(bin_edges[1:], cdf)


# Not survived
counts, bin_edges = np.histogram(haberman_notsvd['age'], bins=10,
                                 density = True)
pdf = counts/(sum(counts))
print(pdf);
print(bin_edges)
cdf = np.cumsum(pdf)
plt.plot(bin_edges[1:],pdf)
plt.plot(bin_edges[1:], cdf)


plt.title('PDF & CDF')
#plt.legend('status')
plt.legend(['Svd_PDF', 'Svd_CDF', 'notSvd_PDF', 'notSvd_CDF'])
plt.xlabel("Age_Status")
plt.show();
```

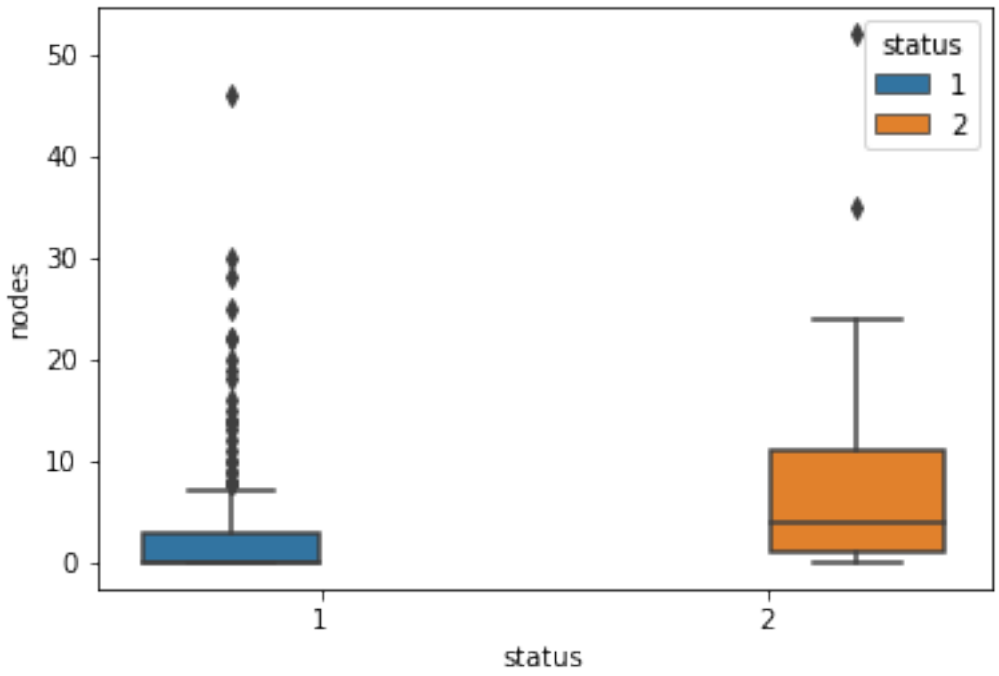
```
[0.05333333 0.10666667 0.12444444 0.09333333 0.16444444 0.16444444
 0.09333333 0.11111111 0.06222222 0.02666667]
[30.  34.7 39.4 44.1 48.8 53.5 58.2 62.9 67.6 72.3 77. ]
[0.03703704 0.12345679 0.19753086 0.19753086 0.13580247 0.12345679
 0.09876543 0.04938272 0.02469136 0.01234568]
[34.  38.9 43.8 48.7 53.6 58.5 63.4 68.3 73.2 78.1 83. ]
```



Observations from PDF & CDF are: Figure 1 - we can come to an assumption that nodes size is small, chances of survival is high figure 2 - chances of survival is nominal if the age is less (quite contradicting)

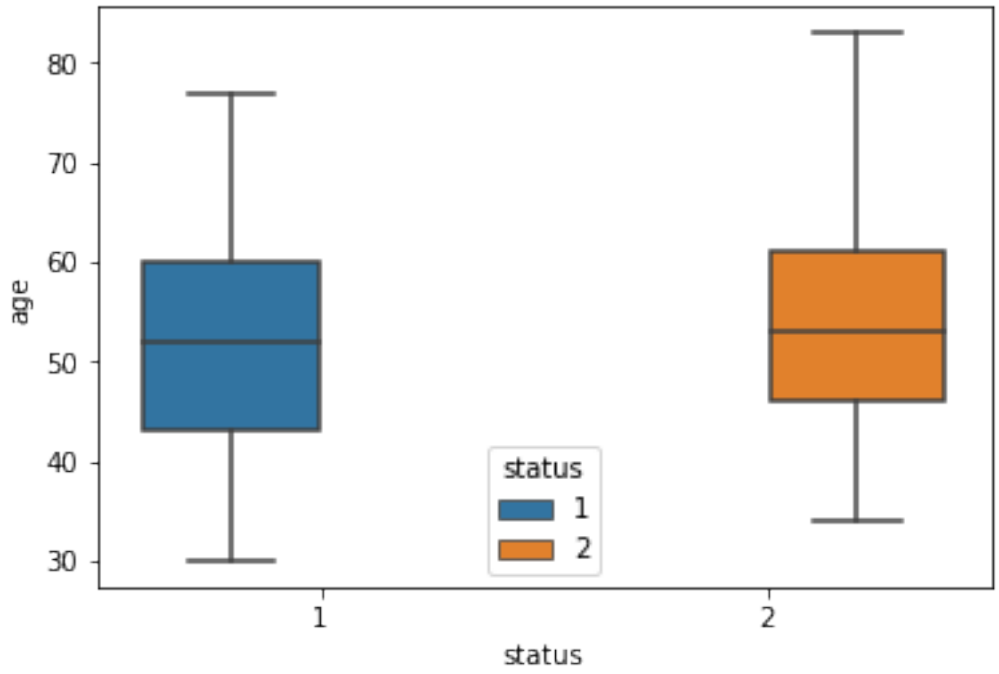
Box plot and Whiskers

```
In [17]: #Box-plot with whiskers: another method of visualizing the  1-D scatter plot more int

         sns.boxplot(x='status',y='nodes', data=haberman, hue='status')
         plt.show()
```



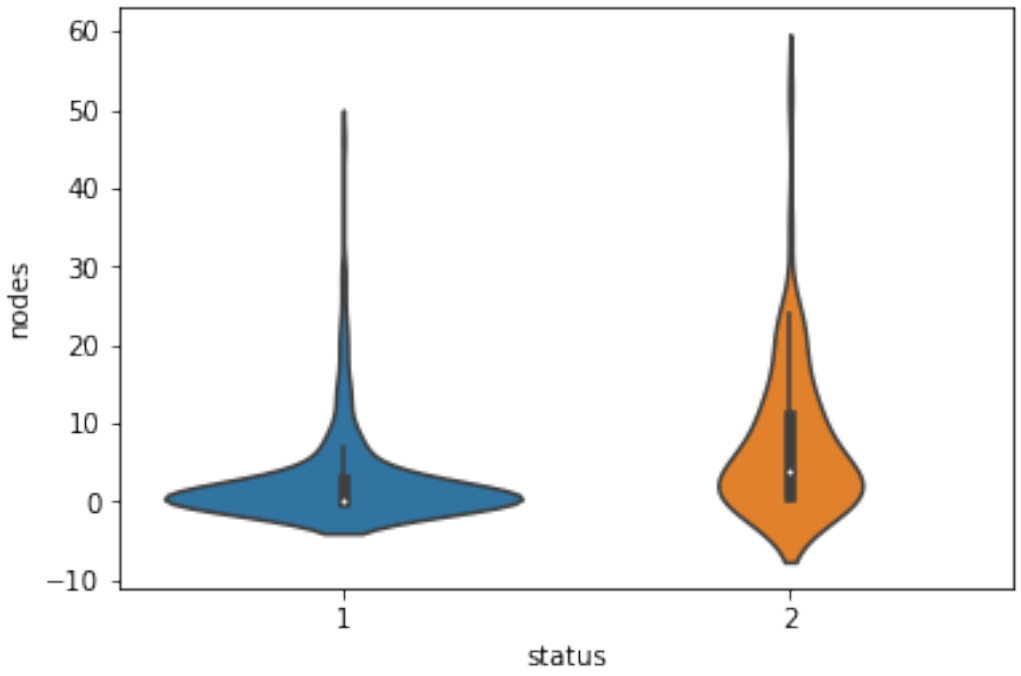
In [18]: *#Box-plot with whiskers: another method of visualizing the 1-D scatter plot more int*

```
sns.boxplot(x='status',y='age', data=haberman, hue='status')
plt.show()
```



Observations: 1) 25th and 50th percentiles are overlapped - skewed to left - nodes are less for more than 50% of data we can say that survival is more 2) 25th, 50th and 75th percentiles are visible.

```
In [21]: # A violin plot combines the benefits of the previous two plots
         #and simplifies them

         # Denser regions of the data are fatter, and sparser ones thinner
         #in a violin plot

         sns.violinplot(x="status", y="nodes", data=haberman, size=8)
         plt.show()
```



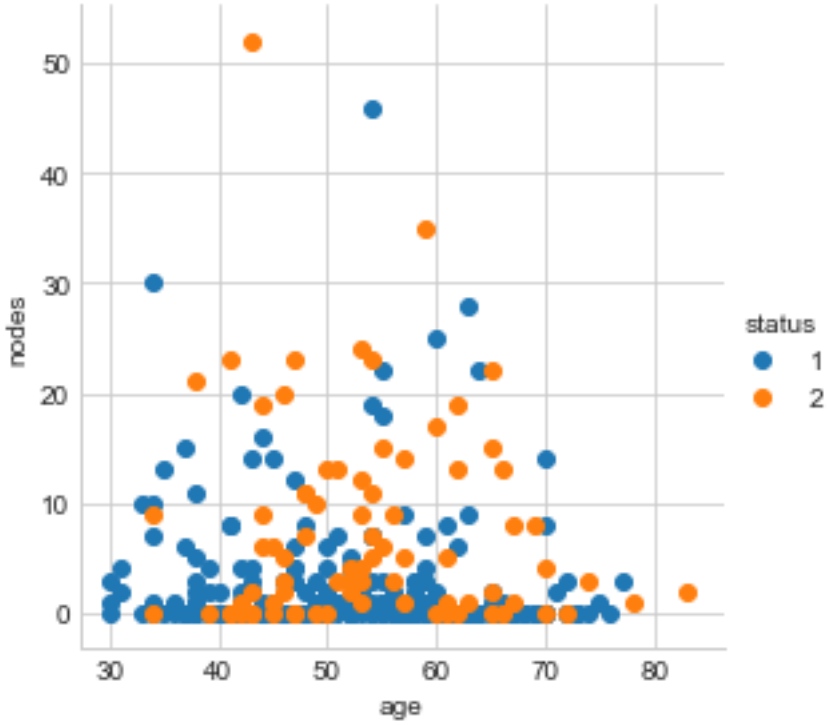
Observations: 50th percentile of survivors have 0 positive nodes, 75th percentie of survivors have less than 3 positive axilary nodes 25th percentile of dead have 1 positive axilary node, 50th percentile of dead have positive axilary nodes below 4,

6 2. Perform a similar alanlaysis as above on this dataset with the following sections:

d) Perform Bi-variate analysis (scatter plots, pair-plots) to see if combinations of features are useful in classfication.

2-D Scatter Plot

In [22]:
```
# 2-D Scatter plot with color-coding for each flower type/class.
# Here 'sns' corresponds to seaborn.
sns.set_style("whitegrid");
sns.FacetGrid(haberman, hue="status", size=4) \
   .map(plt.scatter, "age", "nodes") \
   .add_legend();
plt.show();

# Notice that the blue points can be easily seperated
# from red and green by drawing a line.
# But red and green data points cannot be easily seperated.
# Can we draw multiple 2-D scatter plots for each combination of features?
# How many cobinations exist? 4C2 = 6.
```



Pair-plot

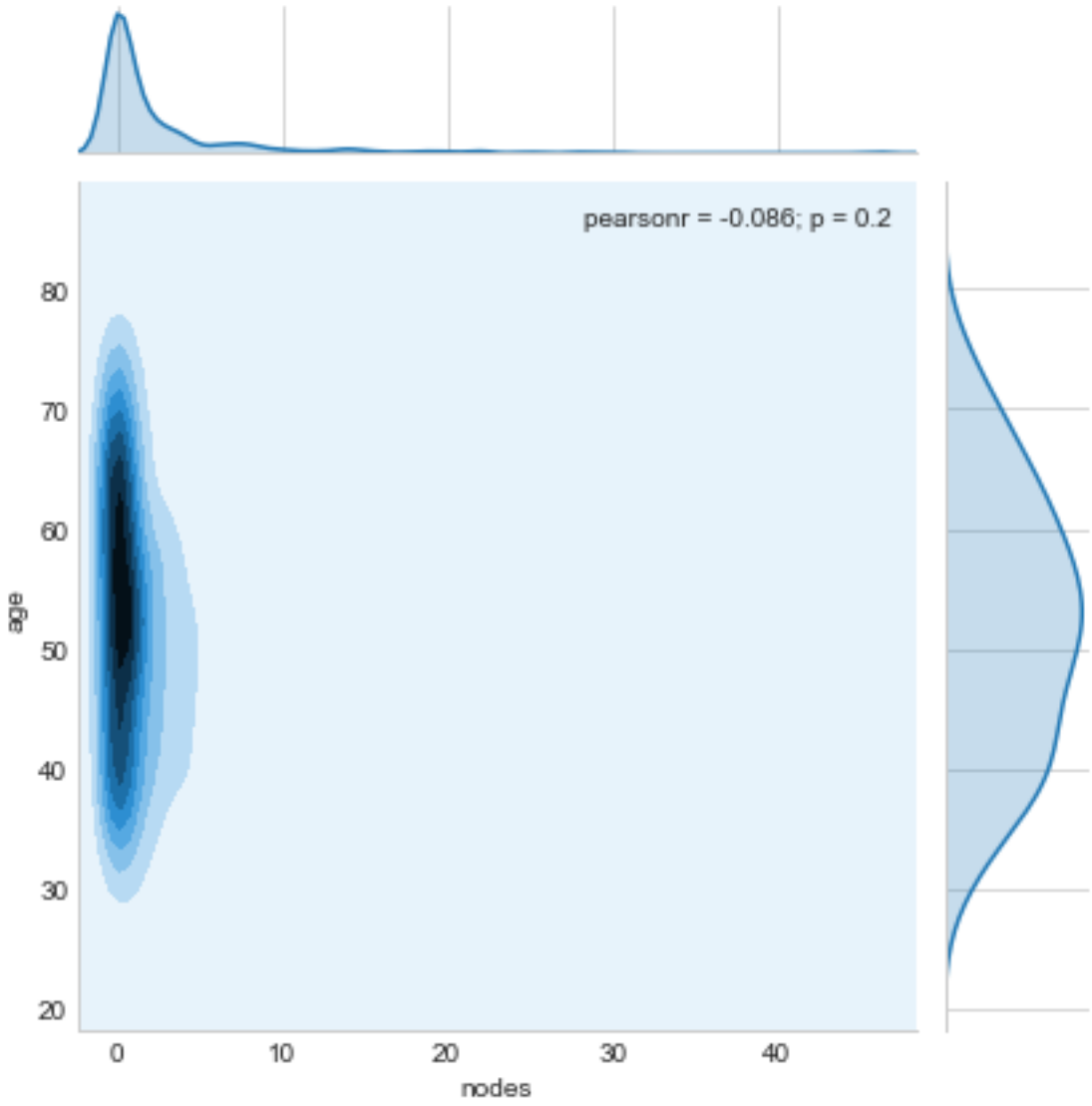
In [23]:
```
# pairwise scatter plot: Pair-Plot
# Dis-advantages:
##Can be used when number of features are high.
##Cannot visualize higher dimensional patterns in 3-D and 4-D.
#Only possible to view 2D patterns.
plt.close();
sns.set_style("whitegrid");
sns.pairplot(haberman, hue="status", size=3, vars=["age", "year", "nodes"], markers=[
```

```
plt.show()
# NOTE: the diagnol elements are PDFs for each feature. PDFs are expalined below.
```

Observations from Pair plot: It is very difficult fr us to interpret on the plots, all the plots are very complex to interpret.

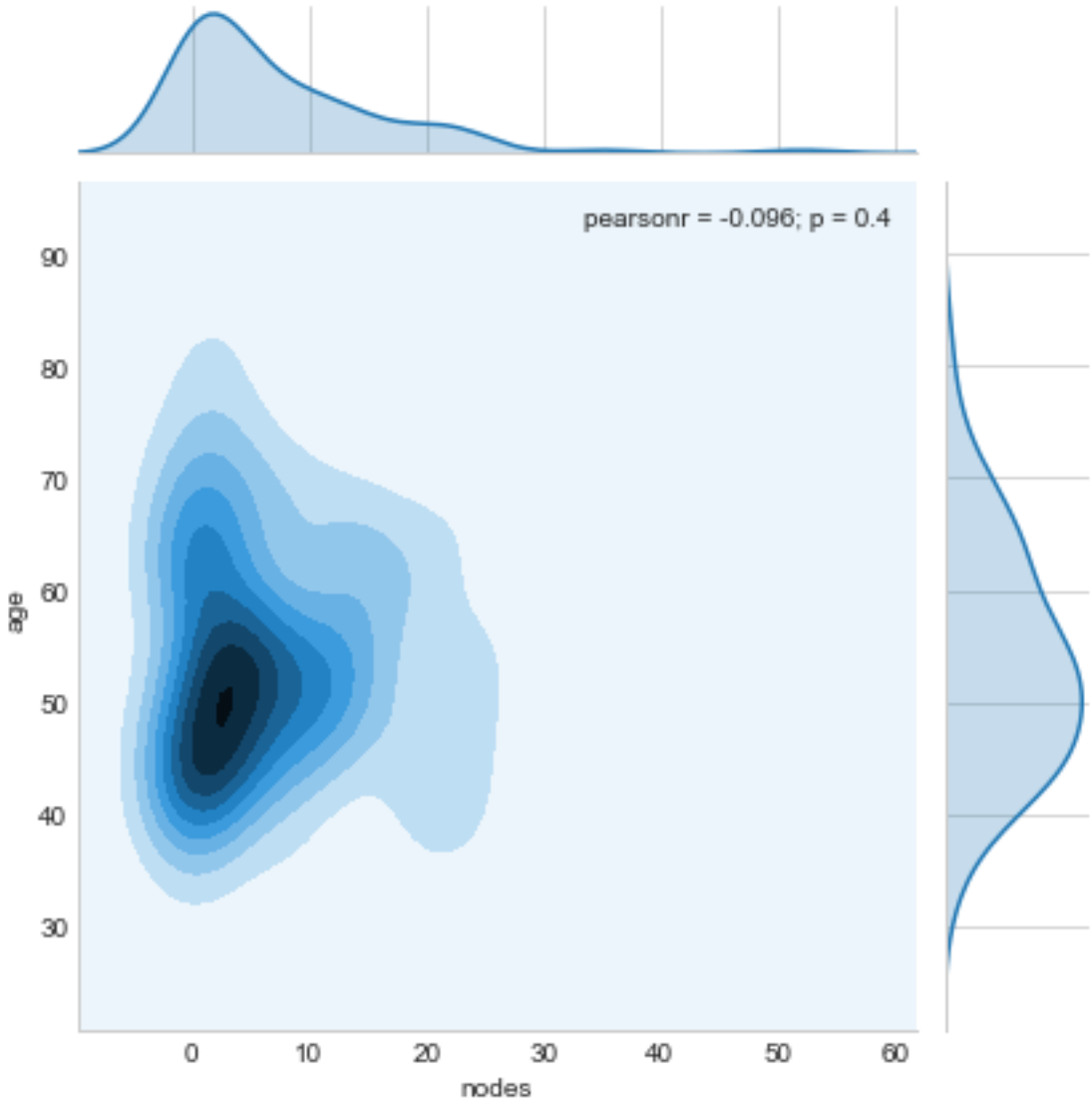
Joint plots

```
In [24]: #2D Density plot, contors-plot
         sns.jointplot(x="nodes", y="age", data=haberman_svd, kind="kde");
         plt.show();
```



Observation : Chances of survival is high if the number of nodes are less and we could conclude that "as number of nodes are less the chances of survival is high"

```
In [25]: #2D Density plot, contors-plot
         sns.jointplot(x="nodes", y="age", data=haberman_notsvd, kind="kde");
         plt.show();
```



In [ ]: Observations:
As nodes increases the chances of survival is less.