- Exploratory data analysis (EDA) on Haberman dataset
- Objective:

1. To clearly specify the given data accordingly.
2. To perform the statistical operations like Mean ,Median , Std Deviation, Percentile, etc.
3. To clearly represent the given data in the pictorial plots which will reduce the coplexity of classification and make our analysis easy.

In [16]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
import warnings
warnings.filterwarnings("ignore")
import os

print(os.getcwd())

#data=pd.read_csv("../haberman.csv")

data=pd.read_csv("D:\haberman\haberman.csv")
print(data)
```

```
C:\Users\bskam\Music\AAIC Haberman assignment
     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
..   ...   ...    ...     ...
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]
```

- Imported the data to the data variable and read the data using Pandas.

In [17]:
```
print(data.shape)
```

```
(306, 4)
```

- Minimum and Maximum values of the columns data in the table.

In [18]:
```
data.columns=['age','year','nodes','status']
print('min age =',data.age.min())
print('max age =',data.age.max())
print('min year ',data.year.min())
print('max year=',data.year.max())
print('min nodes = ',data.nodes.min())
print('max nodes=',data.nodes.max())
```

```
min age = 30
max age = 83
min year  58
max year= 69
min nodes =  0
max nodes= 52
```

- Displaying all the data present in table according to nodes.

In [19]:
```
n=data.groupby('nodes')
for n1 in n:
  print(n1)
```

```
(0,      age  year  nodes  status
2     30    65      0       1
6     33    60      0       1
7     34    59      0       2
13    34    60      0       1
15    35    63      0       1
..   ...   ...    ...     ...
295   72    64      0       1
297   73    62      0       1
298   73    68      0       1
300   74    63      0       1
302   76    67      0       1

[136 rows x 4 columns])
(1,      age  year  nodes  status
0     30    64      1       1
10    34    60      1       1
16    36    60      1       1
29    38    64      1       1
32    38    60      1       1
53    42    69      1       2
56    42    60      1       1
61    42    63      1       1
77    44    63      1       1
82    45    67      1       2
88    45    67      1       1
116   49    61      1       1
119   49    60      1       1
120   49    62      1       1
123   49    67      1       1
129   50    63      1       1
130   50    58      1       1
135   50    66      1       1
139   51    59      1       1
141   51    66      1       1
152   52    62      1       1
157   53    65      1       2
162   53    58      1       1
163   53    60      1       1
165   53    61      1       1
178   54    58      1       1
182   55    58      1       1
184   55    58      1       1
189   55    67      1       1
194   56    66      1       1
199   57    64      1       2
210   58    61      1       1
218   59    64      1       1
225   60    61      1       1
231   61    68      1       2
245   63    60      1       2
267   65    67      1       1
271   66    58      1       1
274   67    63      1       2
301   75    62      1       1
304   78    65      1       2)
(2,      age  year  nodes  status
3     31    59      2       1
25    38    59      2       1
38    39    59      2       1
40    40    58      2       1
57    42    59      2       1
63    43    59      2       2
67    43    64      2       1
70    43    63      2       1
89    46    58      2       2
111   48    62      2       1
131   50    59      2       1
143   52    59      2       2
164   53    60      2       1
193   56    66      2       1
214   58    61      2       1
226   60    67      2       1
259   65    61      2       2
265   65    59      2       1
292   71    68      2       1
305   83    58      2       2)
(3,      age  year  nodes  status
1     30    62      3       1
28    38    62      3       1
68    43    64      3       1
90    46    69      3       2
94    46    58      3       1
103   47    58      3       1
121   49    63      3       1
137   51    59      3       2
142   52    69      3       2
144   52    62      3       2
158   53    59      3       2
172   54    60      3       1
187   55    69      3       1
191   56    66      3       2
209   58    60      3       1
213   58    58      3       1
222   59    67      3       1
296   72    67      3       1
299   74    65      3       2
303   77    65      3       1)
(4,      age  year  nodes  status
4     31    65      4       1
39    39    63      4       1
58    42    61      4       1
72    43    66      4       1
104   47    60      4       1
105   47    68      4       1
134   50    65      4       1
145   52    66      4       2
147   52    63      4       1
149   52    60      4       1
156   53    58      4       2
219   59    64      4       1
286   70    58      4       2)
(5,      age  year  nodes  status
33    38    67      5       1
91    46    62      5       2
150   52    60      5       1
169   54    65      5       2
197   57    61      5       2
229   61    62      5       2)
(6,      age  year  nodes  status
21    37    59      6       1
73    44    64      6       2
80    45    65      6       2
100   47    63      6       1
127   50    61      6       1
180   55    63      6       2
241   62    62      6       1)
(7,      age  year  nodes  status
12    34    67      7       1
109   48    67      7       2
138   51    64      7       1
170   54    68      7       2
171   54    59      7       1
176   54    69      7       1
221   59    64      7       1)
(8,      age  year  nodes  status
47    41    59      8       1
50    41    69      8       1
110   48    61      8       1
235   61    65      8       1
273   67    64      8       2
281   69    67      8       2
290   70    59      8       1)
(9,      age  year  nodes  status
8     34    66      9       2
74    44    58      9       2
159   53    60      9       2
190   56    65      9       2
200   57    64      9       1
251   63    61      9       1)
(10,      age  year  nodes  status
5     33    58     10       1
11    34    61     10       1
115   49    64     10       2)
(11,      age  year  nodes  status
31    38    66     11       1
107   48    58     11       2
108   48    58     11       2
167   54    60     11       2)
(12,      age  year  nodes  status
106   47    66     12       1
161   53    65     12       2)
(13,      age  year  nodes  status
14    35    64     13       1
124   50    63     13       2
136   51    59     13       2
238   62    59     13       2
269   66    61     13       2)
(14,      age  year  nodes  status
66    43    63     14       1
85    45    59     14       1
198   57    62     14       2
287   70    66     14       1)
(15,      age  year  nodes  status
22    37    60     15       1
181   55    68     15       2
261   65    66     15       2)
(16,     age  year  nodes  status
79   44    67     16       1)
(17,      age  year  nodes  status
223   60    59     17       2)
(18,      age  year  nodes  status
185   55    66     18       1)
(19,      age  year  nodes  status
75    44    63     19       2
177   54    63     19       1
240   62    65     19       2)
(20,     age  year  nodes  status
59   42    62     20       1
92   46    65     20       2)
(21,     age  year  nodes  status
24   38    69     21       2)
(22,      age  year  nodes  status
188   55    69     22       1
254   64    65     22       1
260   65    62     22       2)
(23,      age  year  nodes  status
43    41    60     23       2
96    47    63     23       2
168   54    65     23       2)
(24,      age  year  nodes  status
160   53    63     24       2)
(25,      age  year  nodes  status
227   60    61     25       1)
(28,      age  year  nodes  status
252   63    61     28       1)
(30,    age  year  nodes  status
9   34    58     30       1)
(35,      age  year  nodes  status
215   59    62     35       2)
(46,      age  year  nodes  status
174   54    67     46       1)
(52,     age  year  nodes  status
62   43    58     52       2)
```

In [20]:
```
print('data where maximum nodes =\n',data[data.nodes==data.nodes.max()])
```

```
data where maximum nodes =
     age  year  nodes  status
62   43    58     52       2
```

In [21]:
```
data.describe()
```

Out[21]:

	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 [22]:
```
print('mean of age =',data.age.mean())
print('mean of year =',data.year.mean())
print('mean of nodes =',data.nodes.mean())
```

```
mean of age = 52.45751633986928
mean of year = 62.85294117647059
mean of nodes = 4.026143790849673
```

In [23]:
```
print('median of age =',data.age.median())
print('median of year =',data.year.median())
print('median of nodes =',data.nodes.median())
```

```
median of age = 52.0
median of year = 63.0
median of nodes = 1.0
```

In [24]:
```
print('standard deviation of age =',data.age.std())
print('standard deviation of year =',data.year.std())
print('standard deviation of nodes =',data.nodes.std())
```

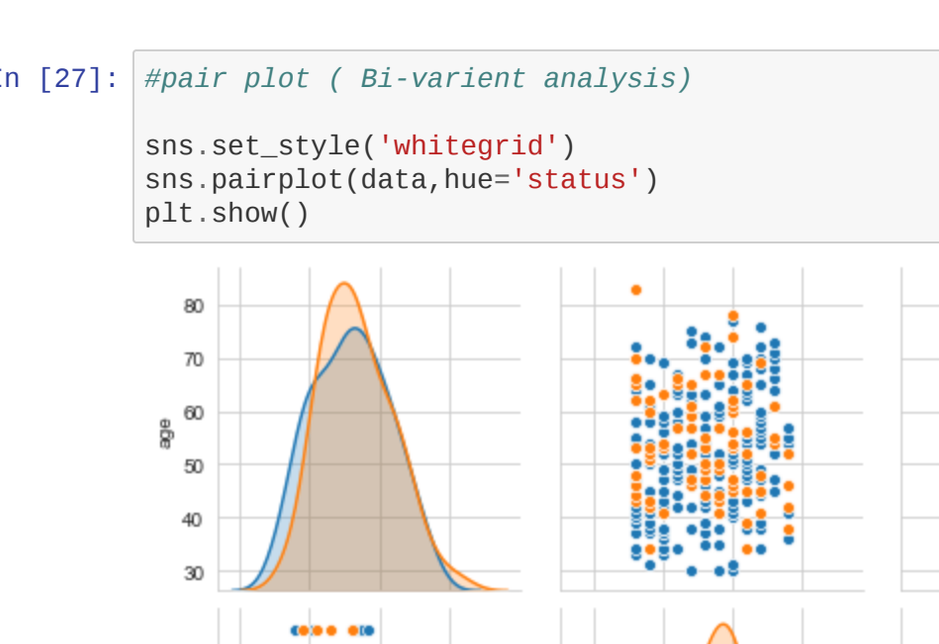
```
standard deviation of age = 10.80345234930328
standard deviation of year = 3.249404663223851
standard deviation of nodes = 7.189653506248565
```

In [25]:
```
print('percentile of age =',np.percentile(data['age'],np.arange(0,100,25)))
print('percentile of year =',np.percentile(data['year'],np.arange(0,100,25)))
print('percentile of nodes =',np.percentile(data['nodes'],np.arange(0,101,25)))
```

```
percentile of age = [30.   44.   52.   60.75]
percentile of year = [58.   60.   63.   65.75]
percentile of nodes = [ 0.  0.  1.  4. 52.]
```

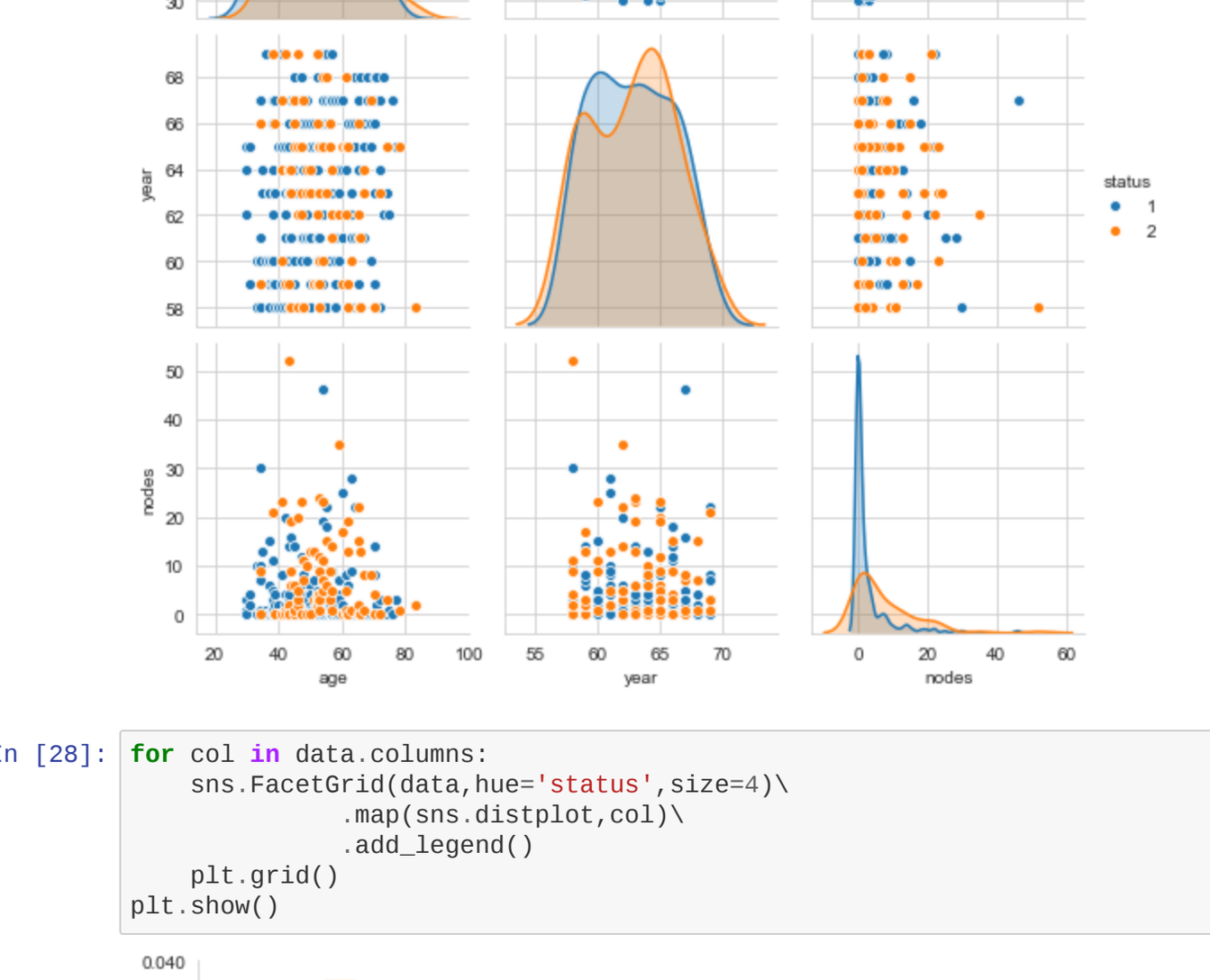
- Univariate analysis

In [26]:
```
sns.FacetGrid(data,hue='status',size=4)\
.map(sns.distplot,'year')\
.add_legend()
plt.grid()
plt.show()
```

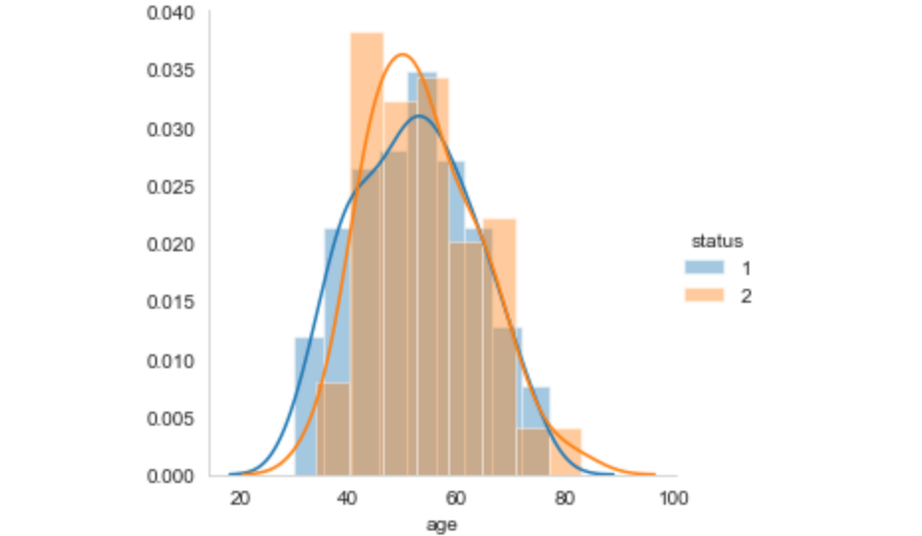


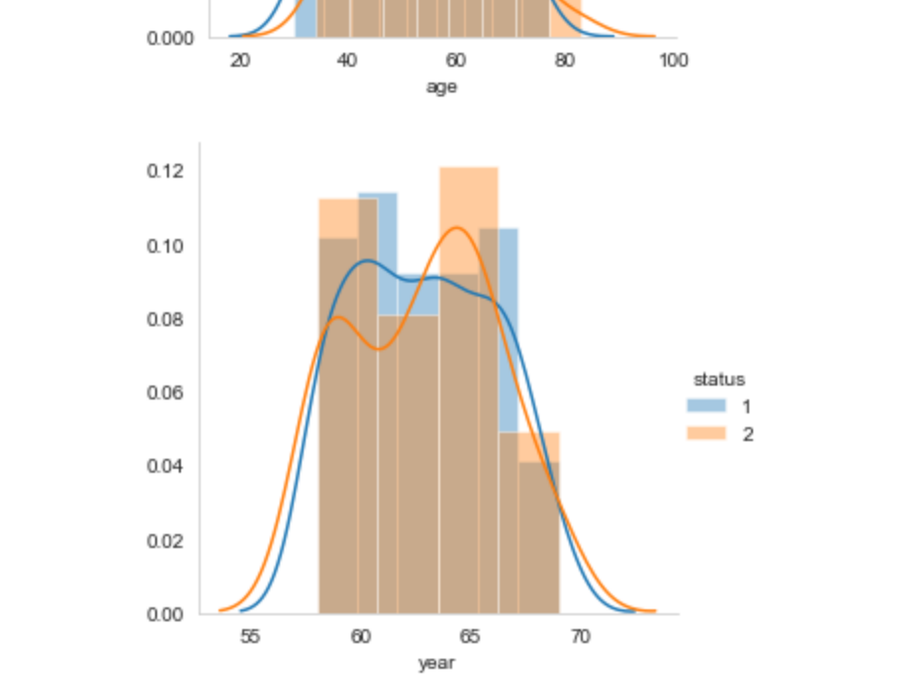
- Pair plot for the given data to study multiple scenarios.
- As shown in the pair plot all the points are very close to each other, hence it is very difficult to analyse the data in below predicted plots.

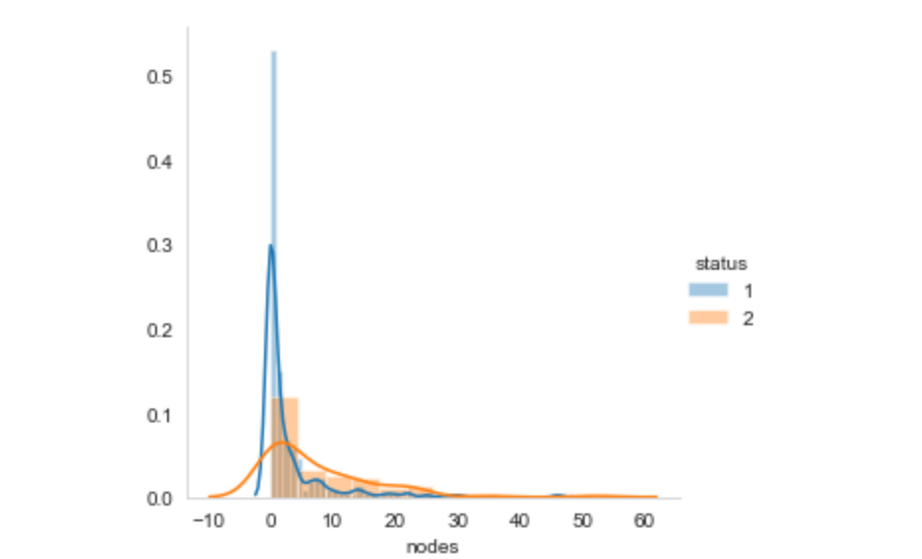
In [27]:
```
#pair plot ( Bi-varient analysis)

sns.set_style('whitegrid')
sns.pairplot(data,hue='status')
plt.show()
```



In [28]:
```
for col in data.columns:
    sns.FacetGrid(data,hue='status',size=4)\
            .map(sns.distplot,col)\
            .add_legend()
    plt.grid()
plt.show()
```





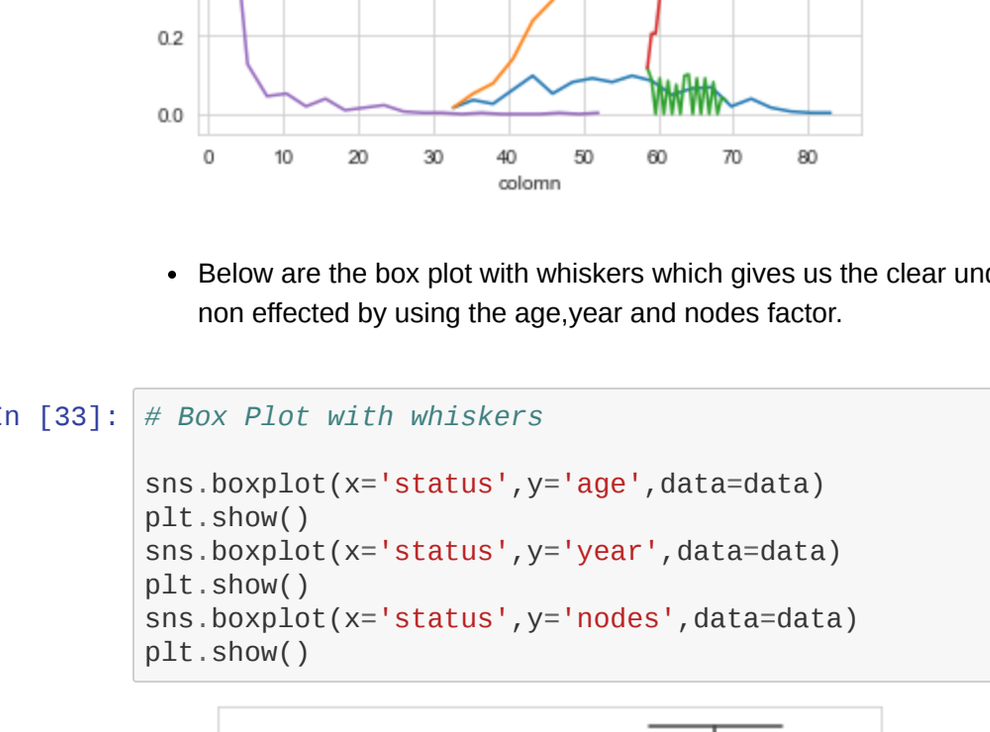


- Cumulative Density function (CDF) with PDF for given data is predicted below who is effected.

In [32]:
```
''' data of AGE '''
counts,bin_edges= np.histogram(data['age'],bins=20,density=True)
pdf=counts/sum(counts)
print("pdf =: ",pdf)
print("bin edges =: ",bin_edges)
cdf=np.cumsum(pdf)
plt.plot(bin_edges[1:],pdf)
plt.plot(bin_edges[1:],cdf)
plt.xlabel('colomn')

''' data of YEAR '''
counts,bin_edges= np.histogram(data['year'],bins=20,density=True)
pdf=counts/sum(counts)
print("pdf =: ",pdf)
print("bin edges =: ",bin_edges)
cdf=np.cumsum(pdf)
plt.plot(bin_edges[1:],pdf)
plt.plot(bin_edges[1:],cdf)
plt.xlabel('colomn')

'''data of NODES '''

counts,bin_edges= np.histogram(data['nodes'],bins=20,density=True)
pdf=counts/sum(counts)
print("pdf =: ",pdf)
print("bin edges =: ",bin_edges)
cdf=np.cumsum(pdf)
plt.plot(bin_edges[1:],pdf)
plt.plot(bin_edges[1:],cdf)
plt.xlabel('colomn')
```

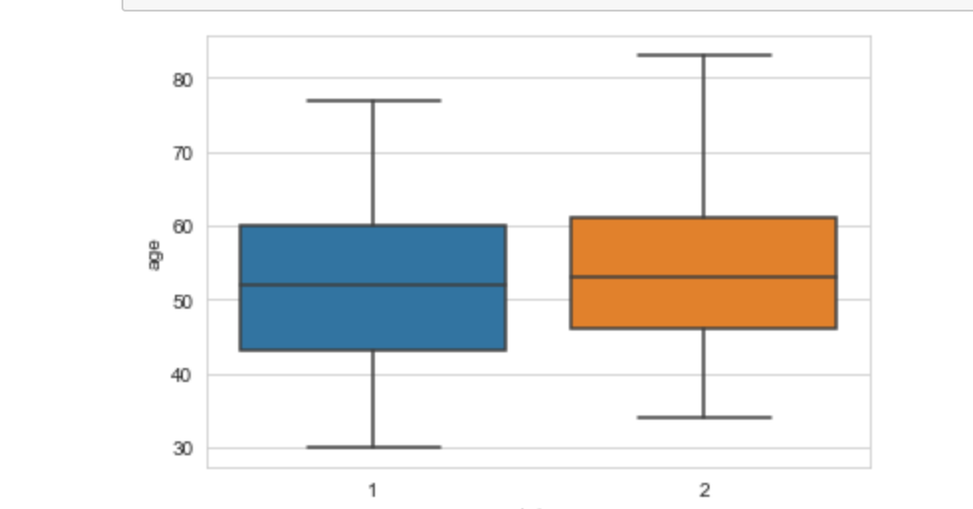
```
pdf =:  [0.01633987 0.03594771 0.02614379 0.0620915  0.09803922 0.05228758
 0.08169935 0.09150327 0.08169935 0.09803922 0.08496732 0.04901961
 0.06535948 0.06862745 0.01960784 0.03921569 0.01633987 0.00653595
 0.00326797 0.00326797]
bin edges =:  [30.   32.65 35.3  37.95 40.6  43.25 45.9  48.55 51.2  53.85 56.5  59.15
 61.8  64.45 67.1  69.75 72.4  75.05 77.7  80.35 83.  ]
pdf =:  [0.11764706 0.08823529 0.         0.09150327 0.         0.08496732
 0.         0.0751634  0.         0.09803922 0.10130719 0.
 0.09150327 0.         0.09150327 0.         0.08169935 0.
 0.04248366 0.03594771]
bin edges =:  [58.   58.55 59.1  59.65 60.2  60.75 61.3  61.85 62.4  62.95 63.5  64.05
 64.6  65.15 65.7  66.25 66.8  67.35 67.9  68.45 69.  ]
pdf =:  [0.64379085 0.12745098 0.04575163 0.05228758 0.01960784 0.03921569
 0.00980392 0.01633987 0.02287582 0.00653595 0.00326797 0.00326797
 0.         0.00326797 0.         0.         0.         0.00326797
 0.         0.00326797]
bin edges =:  [ 0.   2.6  5.2  7.8 10.4 13.  15.6 18.2 20.8 23.4 26.  28.6 31.2 33.8
 36.4 39.  41.6 44.2 46.8 49.4 52. ]
```

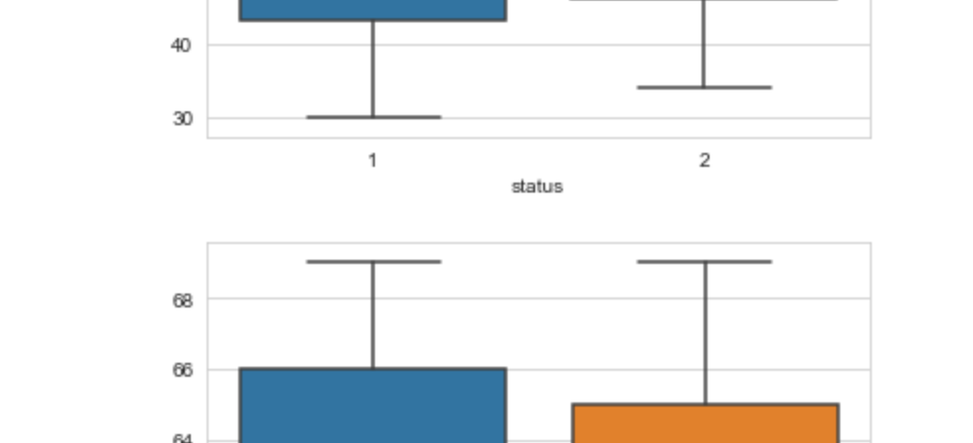
Out[32]:
```
Text(0.5, 0, 'colomn')
```

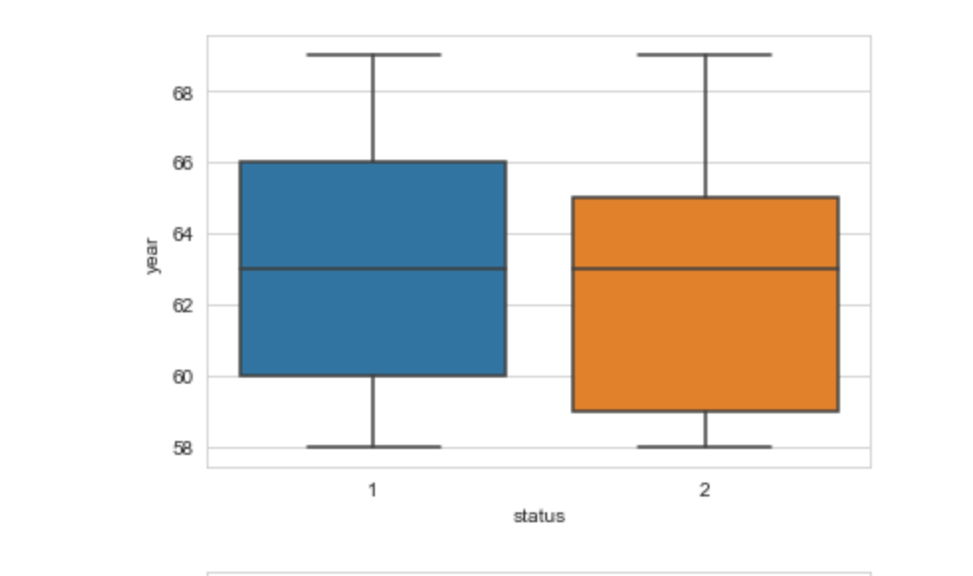


- Below are the box plot with whiskers which gives us the clear understanding where we can differentiate the effected and non effected by using the age,year and nodes factor.

In [33]:
```
# Box Plot with whiskers

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

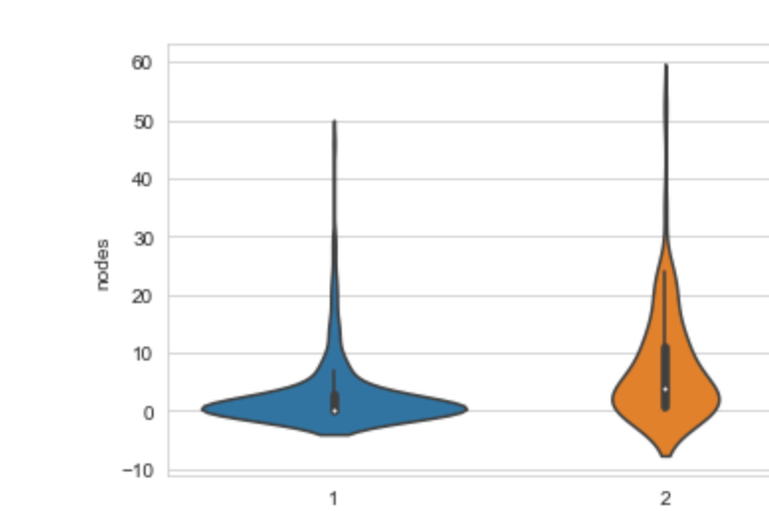


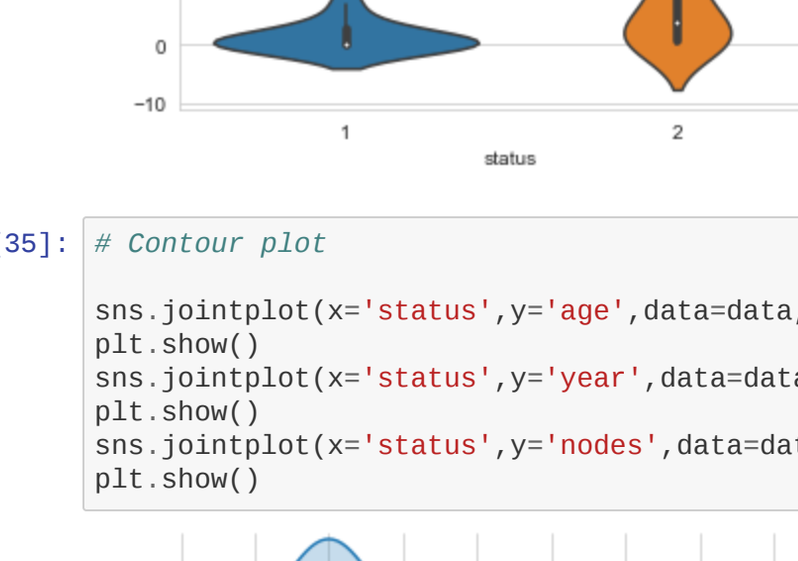




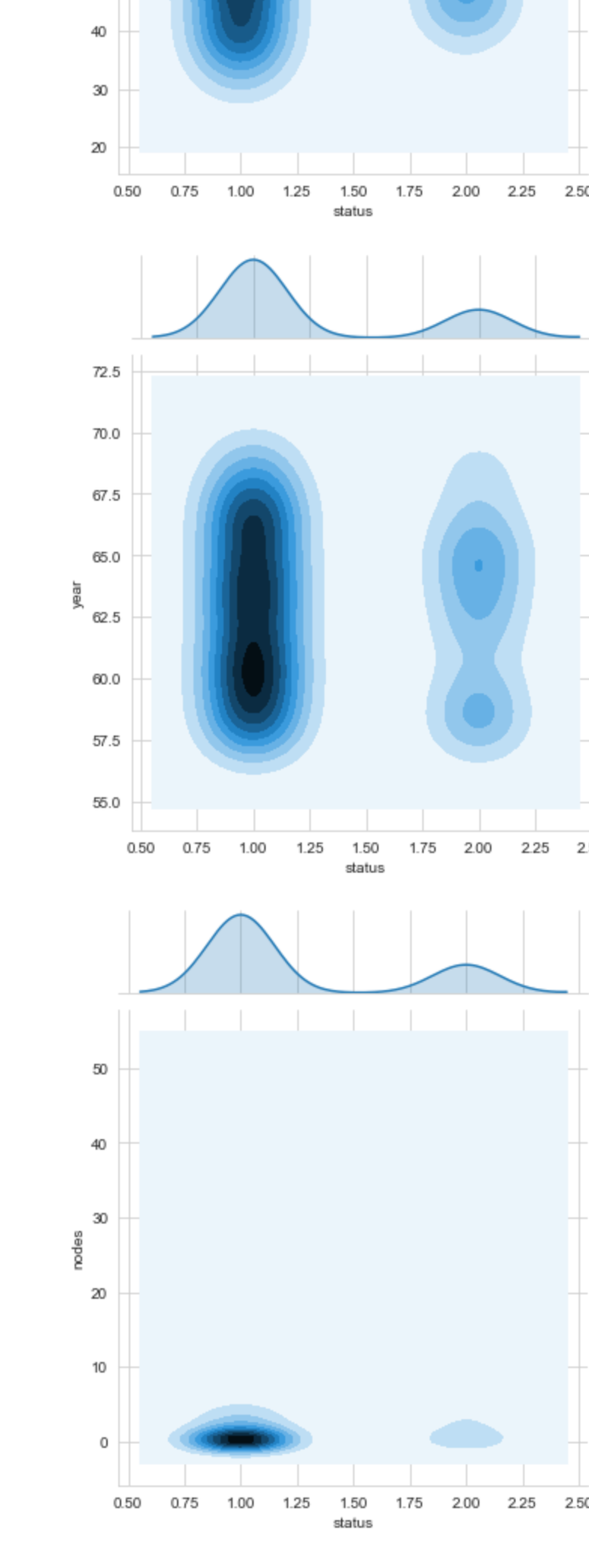
- Below are the Violin plots for the effected people which clearly saperates the effected and non effected than box plot.
- Because Violin plot shows the density of the effected people and non effected people which reduces the error to occur when compared to box plot.

In [34]:
```
# Violin Plot

sns.violinplot(x='status',y='age',data=data)
plt.show()
sns.violinplot(x='status',y='year',data=data)
plt.show()
sns.violinplot(x='status',y='nodes',data=data)
plt.show()
```





In [35]:
```
# Contour plot

sns.jointplot(x='status',y='age',data=data,kind='kde')
plt.show()
sns.jointplot(x='status',y='year',data=data,kind='kde')
plt.show()
sns.jointplot(x='status',y='nodes',data=data,kind='kde')
plt.show()
```



- The above is the Contour plot which clearly shows the density of the effected area clearly than other plots.