In [16]:

```
#The Haberman's survival data set contains cases from a study that was conducted between 1958 and
1970 at the University of Chicago's Billings Hospital on the survival of patients who had undergon
e surgery for breast cancer
#Attribute Information:
#1. Age of patient at time of operation (numerical)
#2. Patient's year of operation (year — 1900, numerical)
#3. Number of positive auxillary nodes detected (numerical)
#4. Survival status (class attribute) 1 = the patient survived 5 years or longer 2 = the patient d
ied within 5 years
#Objective:-
#objective is to predict whether the patient will survive after 5 years or not based upon the pati
ent's age, year of treatment and the number of positive lymph nodes
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
haberman = pd.read_csv("haberman.csv")
haberman.head()
```

Out[16]:

	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 [17]:

```
#Columns Name in our dataset
print(haberman.columns)
print(haberman.info())
```

```
Index(['age', 'year', 'nodes', 'status'], dtype='object')
<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
None
```

In [18]:

```
#Status
haberman['status'] = haberman['status'].map({1:"yes", 2:"no"})
haberman['status'] = haberman['status'].astype('category')
print(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 category
dtypes: category(1), int64(3)
memory usage: 7.6 KB
None
```

In [20]:

```
#After Categorization
haberman.head()
```

Out[20]:

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

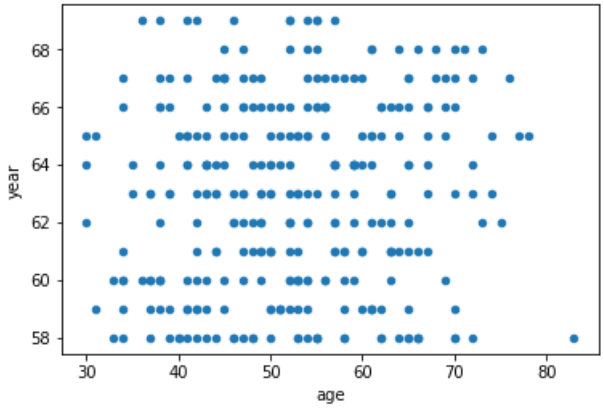
In [53]:

```
#Find Total
print(haberman['status'].value_counts())
'''
observation:-

1) Median is 52
2) 25% of people have 0 nodes
3) 75% people have less than 5 nodes
4) 225/306= 73% people have survival more than 5 years
'''
print(haberman.describe())
```

```
yes    225
no      81
Name: status, dtype: int64
              age        year       nodes
count  306.000000  306.000000  306.000000
mean    52.457516   62.852941    4.026144
std     10.803452    3.249405    7.189654
min     30.000000   58.000000    0.000000
25%     44.000000   60.000000    0.000000
50%     52.000000   63.000000    1.000000
75%     60.750000   65.750000    4.000000
max     83.000000   69.000000   52.000000
```

In [27]:

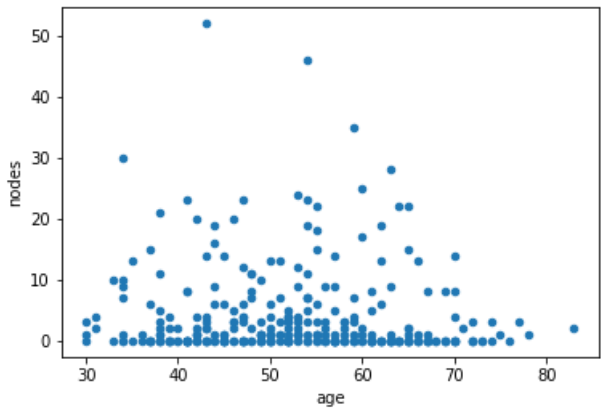
```
haberman.plot(kind='scatter',x='age',y='year');
plt.show()
# With the below plot we can only find the count of people of certain age at certain year which is
not our objective
```



In [28]:

```
haberman.plot(kind='scatter',x='age',y='nodes');
```

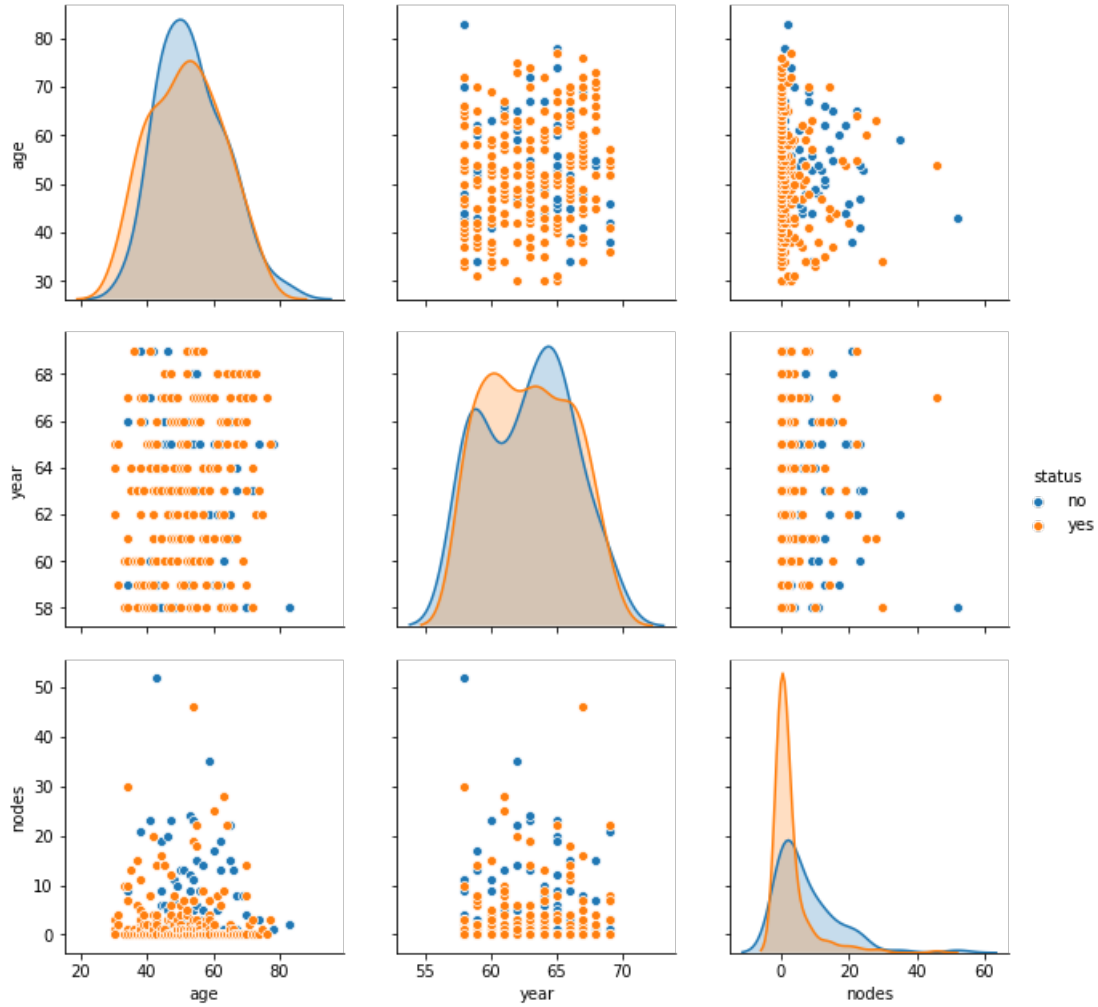
```
plt.show()
#We cannot find our objective with the below plot as we cannot find given points falls in which ca
tegory
```



In [29]:

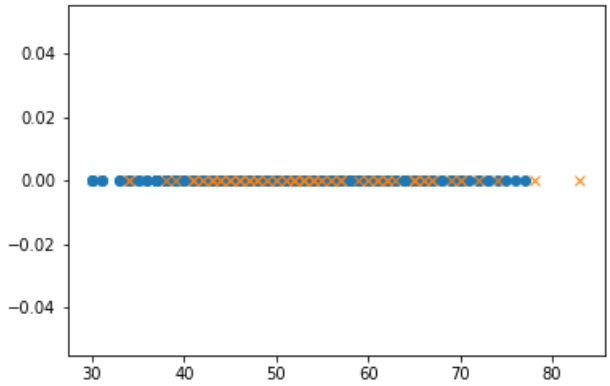
```
sns.pairplot(haberman,hue='status',size=3)
plt.show()
#Pairplot is useful when we dont know which attributes are useful for differentiating data.
#We can find from below plots that plot 3 and plot 7 can differentiate data though little complex
but better than other plots
#we can assume from Plot 3 and 7 that nodes and age is useful attribute to categorize data
```

```
c:\users\nishu\appdata\local\programs\python\python36-32\lib\site-
packages\seaborn\axisgrid.py:2065: UserWarning: The `size` parameter has been renamed to `height`;
pleaes update your code.
  warnings.warn(msg, UserWarning)
```



In [39]:

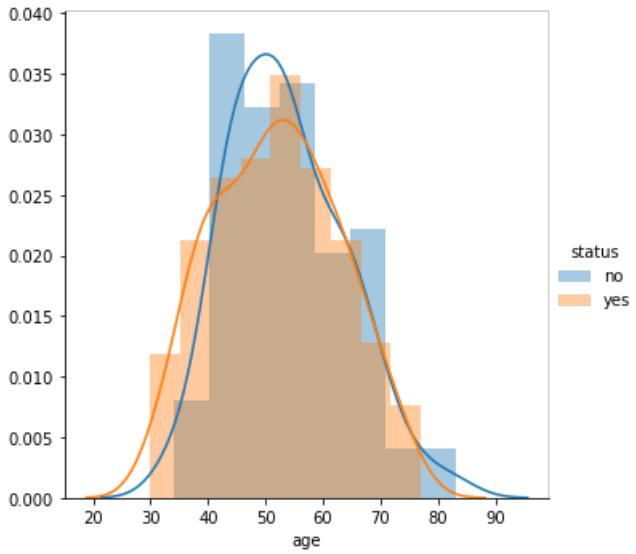
```
#lets check if single attributes are useful for finding our objective
haberman_yes = haberman.loc[haberman['status']=='yes']
haberman_no = haberman.loc[haberman['status']=='no']
#print(haberman_yes)
plt.plot(haberman_yes['age'],np.zeros_like(haberman_yes['age']),'o')
plt.plot(haberman_no['age'],np.zeros_like(haberman_no['age']),'x')
plt.show()
#Below 1 D scatter plot is not useful to differentiate or divide data properly as its overlapping
```



In [46]:

```
#to overcome overlapping will check using histogram and pdf for attribute age
sns.FacetGrid(haberman, hue='status', size=5)\
.map(sns.distplot,'age').add_legend()
plt.show()
'''
Though below plot with age attribute have almost same Probability density
however we can still try to construct model out of it */

if (age > = 30 and age < =75)
        survival is more than 5 Years
    else if(age >=30 and age < =75)
        survival is less than 5 years
Observation :-

1 We can see that model is not correct and it will not give results

'''
```

```
c:\users\nishu\appdata\local\programs\python\python36-32\lib\site-
packages\seaborn\axisgrid.py:230: UserWarning: The `size` paramter has been renamed to `height`; p
lease update your code.
  warnings.warn(msg, UserWarning)
```



Out[46]:

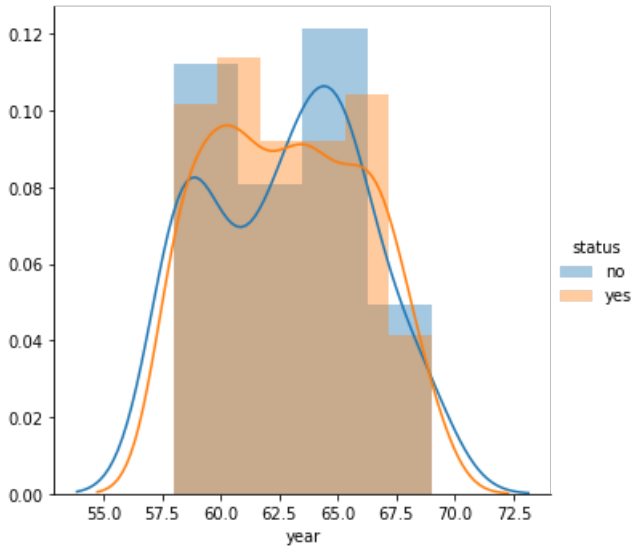
```
"\nThough below plot with age attribute have almost same Probability density \nhowever we can stil
l construct model out of it which wont be accurate*/\n\nif (age > = 30 and age <40) or (age>60 and
```

```
age <65) or (age>70 and age <=75) \n        survival is more than 5 Years \n    else if(age >=40 ar
d age < =60) or( age >=65 and age < =70) or (age >77 and age <= 83)\n        survival is less than
5 years \n        \nHowever above model is not good when age is 40,50,77,70 etc where\n    data poi
nts overlaps but considering the height of the bar '=' is assigned i.e height of bar\n    at 40 is
more for survival less than four years \n"
```

In [49]:

```
#to overcome overlapping will check using histogram  and pdf for attribute  nodes
sns.FacetGrid(haberman, hue='status', size=5)\
.map(sns.distplot,'year').add_legend()
plt.show()

'''


Observation :-
1) still pdf is overlapping and we cannot construct good model out of it

'''
```

```
c:\users\nishu\appdata\local\programs\python\python36-32\lib\site-
packages\seaborn\axisgrid.py:230: UserWarning: The `size` paramter has been renamed to `height`; p
lease update your code.
  warnings.warn(msg, UserWarning)
```

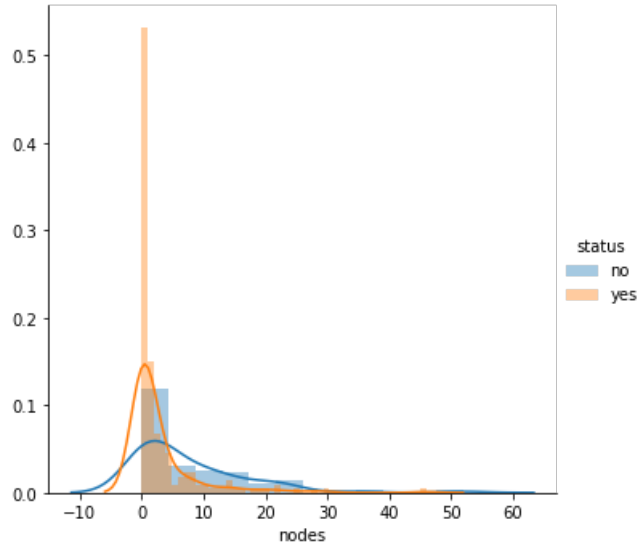


In [48]:

```
#to overcome overlapping will check with histogram for attribute  nodes
sns.FacetGrid(haberman, hue='status', size=5)\
.map(sns.distplot,'nodes').add_legend()
plt.show()

'''
Using PDF we can easily construct model for the node attribute

Model:-
 if nodes < 3
     survival after 5 years
 else if nodes > 3
     survival less than 5 years

Observation:-

1) PLOT looks better than the plot of age and year attributes

2) We can say that nodes are more useful attribute in categorizing the data
nodes> year> age

3) Peak is high for not survival and survival after five years between 0 to 3 but no of counts is
more for survival
after 5 years hence above model is constructed taking this into consideration.So, we cannot say mo
del will give accurate
```

```
results when nodes are between 0 to 3.

'''
```

```
c:\users\nishu\appdata\local\programs\python\python36-32\lib\site-
packages\seaborn\axisgrid.py:230: UserWarning: The `size` paramter has been renamed to `height`; p
lease update your code.
  warnings.warn(msg, UserWarning)
```



Out[48]:

```
'\nModel:-\n if nodes < 2 or (nodes > 26 and nodes < 52) \n     survival after 5 years\n else if (
nodes > 2 and nodes < 26) or (nodes > 52)\n     survival less than 5 years\n    \n\n'
```

In [67]:

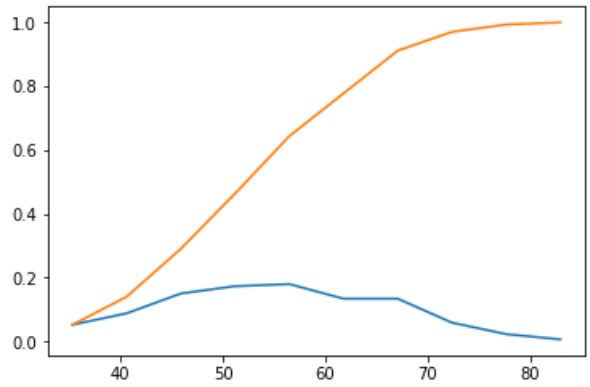
```
age_counts , age_bin_edges = np.histogram(haberman['age'],bins=10,density=True)
age_pdf = age_counts/sum(age_counts)
print("PDF AGE:",age_pdf)
print("AGE_BINS:",age_bin_edges)
print("\n")

year_counts , year_bin_edges = np.histogram(haberman['year'],bins=10,density=True)
year_pdf = year_counts/sum(year_counts)
print("PDF_YEAR:",year_pdf)
print("YEAR_BINS:",year_bin_edges)
print("\n")

nodes_counts , nodes_bin_edges = np.histogram(haberman['nodes'],bins=10,density=True)
nodes_pdf = nodes_counts/sum(nodes_counts)
print("PDF_NODES:",nodes_pdf)
print("NODE_BINS:",nodes_bin_edges)
```

```
PDF AGE: [0.05228758 0.08823529 0.1503268  0.17320261 0.17973856 0.13398693
 0.13398693 0.05882353 0.02287582 0.00653595]
AGE_BINS: [30.  35.3 40.6 45.9 51.2 56.5 61.8 67.1 72.4 77.7 83. ]


PDF_YEAR: [0.20588235 0.09150327 0.08496732 0.0751634  0.09803922 0.10130719
 0.09150327 0.09150327 0.08169935 0.07843137]
YEAR_BINS: [58.  59.1 60.2 61.3 62.4 63.5 64.6 65.7 66.8 67.9 69. ]


PDF_NODES: [0.77124183 0.09803922 0.05882353 0.02614379 0.02941176 0.00653595
 0.00326797 0.         0.00326797 0.00326797]
NODE_BINS: [ 0.   5.2 10.4 15.6 20.8 26.  31.2 36.4 41.6 46.8 52. ]
```

In [70]:

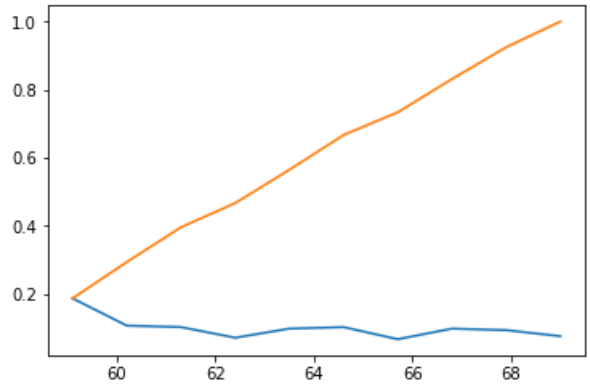
```
#Computing CDF for age
age_cdf = np.cumsum(age_pdf)
plt.plot(age_bin_edges[1:],age_pdf)
plt.plot(age_bin_edges[1:],age_cdf)
```

```
plt.show()
```



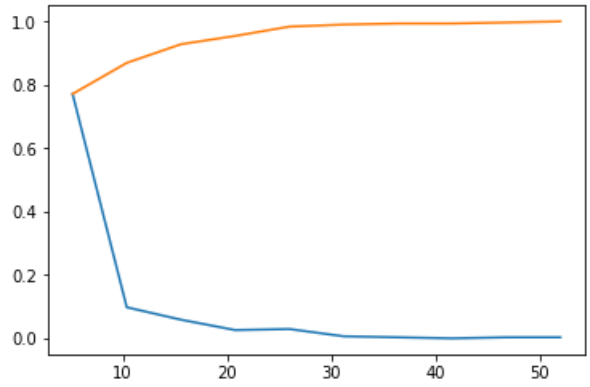
In [76]:

```
#Computing CDF for year
year_cdf = np.cumsum(year_pdf)
plt.plot(year_bin_edges[1:],year_pdf)
plt.plot(year_bin_edges[1:],year_cdf)
plt.show()
```



In [72]:

```
#Computing CDF for nodes
nodes_cdf = np.cumsum(nodes_pdf)
plt.plot(nodes_bin_edges[1:],nodes_pdf)
plt.plot(nodes_bin_edges[1:],nodes_cdf)
plt.show()
```



In [78]:

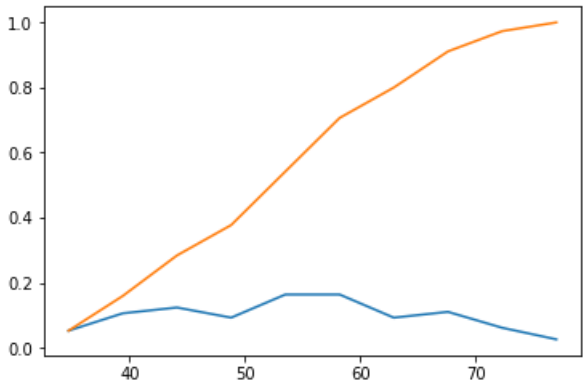
```
age_counts , age_bin_edges = np.histogram(haberman_yes['age'],bins=10,density=True)
age_pdf = age_counts/sum(age_counts)
print("PDF AGE:",age_pdf)
print("AGE_BINS:",age_bin_edges)
print("\n")

year_counts , year_bin_edges = np.histogram(haberman_yes['year'],bins=10,density=True)
year_pdf = year_counts/sum(year_counts)
```

```
year_pdf = year_counts/sum(year_counts)
print("PDF_YEAR:",year_pdf)
print("YEAR_BINS:",year_bin_edges)
print("\n")

nodes_counts , nodes_bin_edges = np.histogram(haberman_yes['nodes'],bins=10,density=True)
nodes_pdf = nodes_counts/sum(nodes_counts)
print("PDF_NODES:",nodes_pdf)
print("NODE_BINS:",nodes_bin_edges)
```

```
PDF AGE: [0.05333333 0.10666667 0.12444444 0.09333333 0.16444444 0.16444444
 0.09333333 0.11111111 0.06222222 0.02666667]
AGE_BINS: [30.  34.7 39.4 44.1 48.8 53.5 58.2 62.9 67.6 72.3 77. ]


PDF_YEAR: [0.18666667 0.10666667 0.10222222 0.07111111 0.09777778 0.10222222
 0.06666667 0.09777778 0.09333333 0.07555556]
YEAR_BINS: [58.  59.1 60.2 61.3 62.4 63.5 64.6 65.7 66.8 67.9 69. ]


PDF_NODES: [0.83555556 0.08       0.02222222 0.02666667 0.01777778 0.00444444
 0.00888889 0.         0.         0.00444444]
NODE_BINS: [ 0.   4.6  9.2 13.8 18.4 23.  27.6 32.2 36.8 41.4 46. ]
```

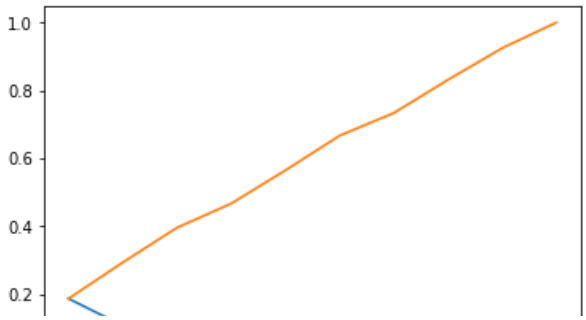
In [75]:

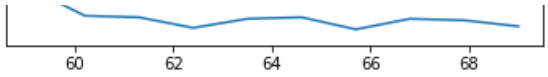
```
#Computing CDF for status=yes for age attribute
age_cdf = np.cumsum(age_pdf)
plt.plot(age_bin_edges[1:],age_pdf)
plt.plot(age_bin_edges[1:],age_cdf)
plt.show()

#Observation
#1)when age is less than 55 then there are 45 % chances of survival after 5 years.
#this doesnt give us good result. Anyway we have to find % for nodes which gave us model to find t
he patient results
```



In [79]:

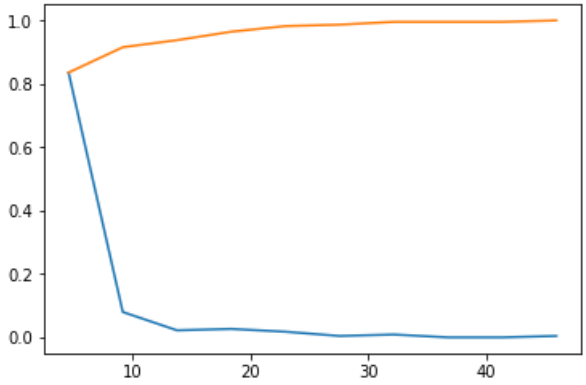
```
#Computing CDF for year
year_cdf = np.cumsum(year_pdf)
plt.plot(year_bin_edges[1:],year_pdf)
plt.plot(year_bin_edges[1:],year_cdf)
plt.show()
#observation
#less than 59 years age have 20% probability of survival. Not good enough to find exact count
```





In [80]:

```
#Computing CDF for nodes
nodes_cdf = np.cumsum(nodes_pdf)
plt.plot(nodes_bin_edges[1:],nodes_pdf)
plt.plot(nodes_bin_edges[1:],nodes_cdf)
plt.show()
#Observation
#when nodes are less than 5 there are 82% chances of survival.
#We have also seen from our pdf model that less than node 3 have survival more than 5  years. The
CDF gave us probability of survival
```



In [93]:

```
#Mean, Variance, Std Deviation
print(np.mean(haberman_yes['nodes']))
print(np.mean(np.append(haberman_yes['nodes'],100)))
print(np.mean(haberman_no['nodes']))
print(np.mean(np.append(haberman_no['nodes'],100)))
#std
print(np.std(haberman_yes['nodes']))
print(np.std(haberman_no['nodes']))

#Observation
#Mean of survival after 5 years is 2.79 while after adding outlier it is 3.2 whch is almost same
#Mean of survival less than 5 years is 7.45 which means spread is more and probability of total su
rvival less than 5 years is more
```

```
2.7911111111111113
3.2212389380530975
7.45679012345679
8.585365853658537
5.857258449412138
9.128776076761635
```

In [97]:

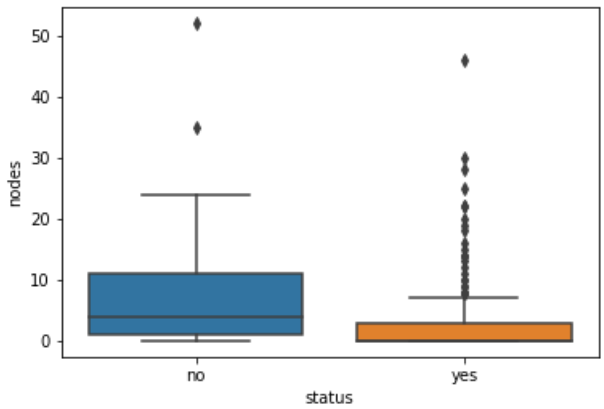
```
#Median, Quantiles and Percentiles
print("Survival Median:",np.median(haberman_yes['nodes']))
print("Survival Mdian with Outlier:",np.median(np.append(haberman_yes['nodes'],100)))

print("Less Survival Median:",np.median(haberman_no['nodes']))
print("Less Survival Median with outlier:",np.median(np.append(haberman_no['nodes'],100)))

#Quantiles
print(np.percentile(haberman_yes['nodes'],np.arange(0,100,25)))
print(np.percentile(haberman_no['nodes'],np.arange(0,100,25)))

#Percentiles
print(np.percentile(haberman_yes['nodes'],90))
print(np.percentile(haberman_no['nodes'],90))

#Observation
#median is 0 for survival more than 5 years case means average nodes left will be 0
#median is 4 for survival less than 5 years case means average nodes left will be 4
```

```
#25th, 50th and 75th percentile is 0  for survival more than 5 years means only  25% patient will
have 3 nodes in survival more than 5 years
#survival less than 5 years have 25th quantile as 0, 50th as 1, 75th as 4 and 100th as 11 means 25
% patient will have 1 nodes,50% will have 4 nodes and 75% will have 11 nodes.
#90th percentile has value 8 means more than 8 nodes will increase the chances of survival more th
an 5 years.
#90th percentile has value 20 means  more than 20 nodes will decrease the chances of survival more
than 5 years
```

```
Survival Median: 0.0
Survival Mdian with Oulier: 0.0
Less Survival Median: 4.0
Less Survival Median with oulier: 4.0
[0. 0. 0. 3.]
[ 0.  1.  4. 11.]
8.0
20.0
```

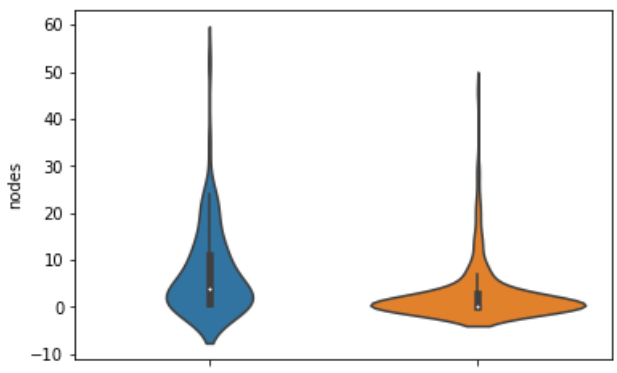
In [98]:

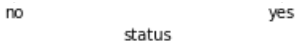
```
#BoxPlot
sns.boxplot(x='status', y='nodes', data=haberman)
plt.show()

#Observation
#between node 0 to 7 is survival more than 5 years
#between node 0 to 25 survival is less than 5 years
#There are error of around 60% between node 0 and 7 as less survival data is also present
#for survival more than 5 years more than 7 nodes are ouliers and in case of less survival more th
an 35 nodes are outlier
#25th and 50th percentile of survival after 5 years is same
#50th percentile of survival less than 5 years is almost same as 75th percentile of survival more
than 5 years
```



In [99]:

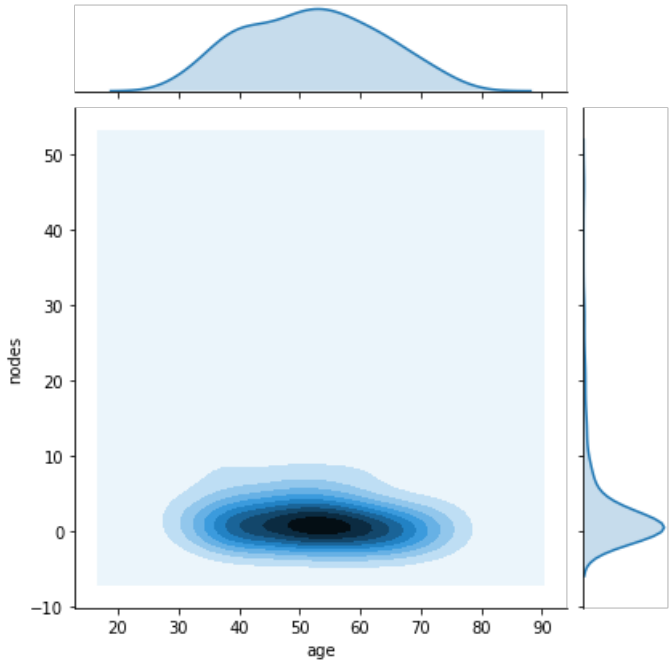
```
#Violin Plot
sns.violinplot(x='status', y='nodes', data=haberman,size=8)
plt.show()
#Observation
#interquantile range is 0-12 nodes for survival less than 5 years and 0-2 nodes for survival more
than 5 years
#
```





In [102]:

```
#ContourPlot
sns.jointplot(x='age',y='nodes',data=haberman_yes,kind='kde')
plt.show()

#Observation
#density is more between age 45 and 62 and node lies between 0 to 3 i.e survival rate is high when
node is between 0 to 3 and age between 45 and 62
```



In [ ]:

```
'''
Conclusion:-

The hiberman survival data has less number of rows. However through pair plots we observed that ag
e and node attributes
separates data and through PDF plot we found that node was useful to create the model to find the
survival chances after
5 years. And CDF helped to find count data  for survival after 5 years for nodes.We observed from
percentile how to increase
the chances of survival based on nodes. percentage of Error in results can be calculated from  mul
tivariate analysis.

'''
```