In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [9]:
```
pwd
```

Out[9]:
```
'C:\\Users\\Sai Preenith'
```

In [15]:
```
haberman=pd.read_csv("haberman.csv")
```

In [13]:
```
print(haberman.shape)
```

```
(306, 4)
```

In [14]:
```
print(haberman.columns)
```

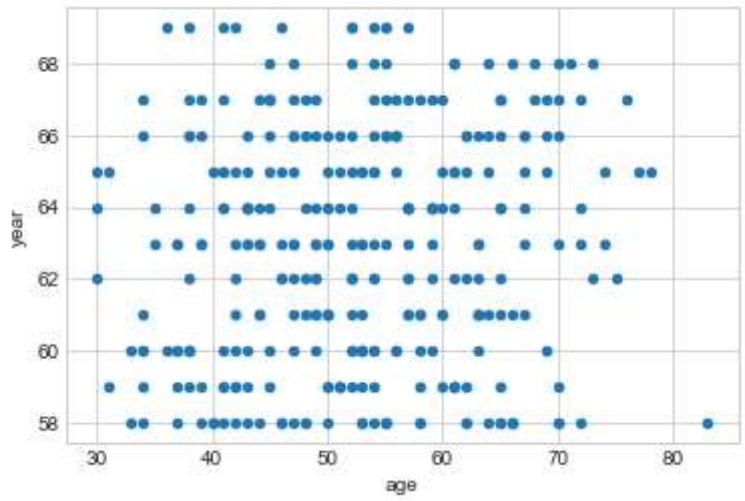
```
Index(['age', 'year', 'nodes', 'status'], dtype='object')
```

In [16]:
```
haberman['status'].value_counts()
```

Out[16]:
```
1    225
2     81
Name: status, dtype: int64
```

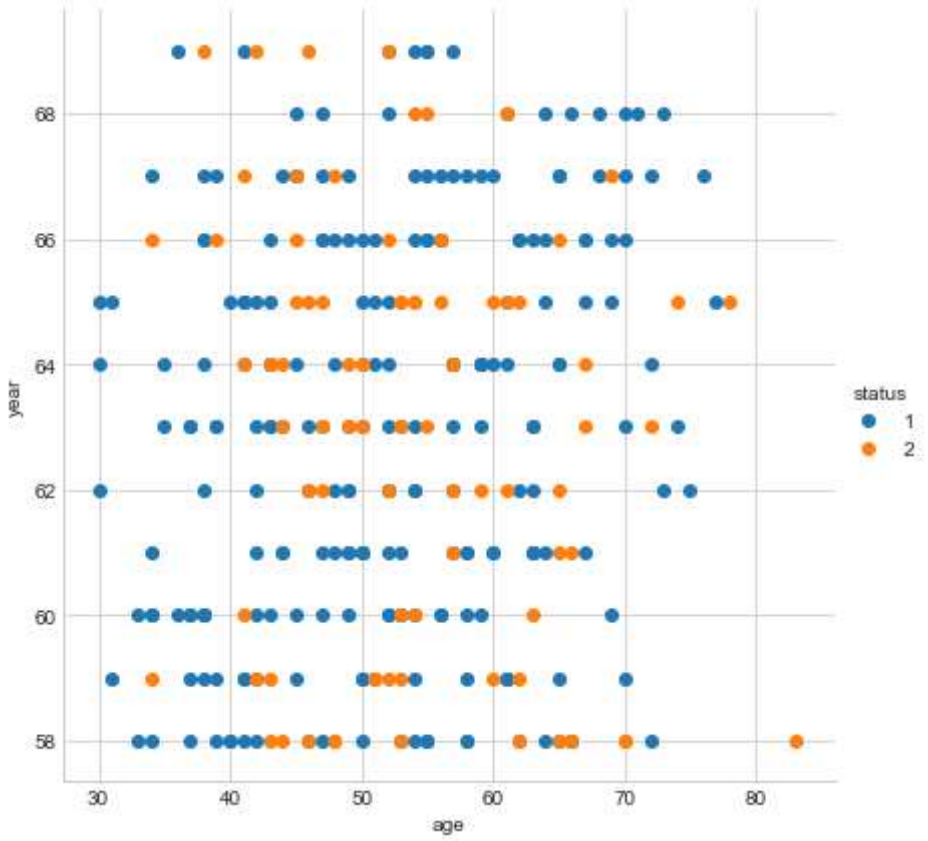
As we can see the data points are not equally distributed hence it is Imbalanced Data set

In [39]:
```
#2D Scatter plot
haberman.plot(kind='scatter',x='age',y='year');
plt.show();
```



Here you can see that it is not clear and we cannot make sense age varies from 30 to 80 year varies from 58 to 68

In [23]:
```
sns.set_style("whitegrid");
sns.FacetGrid(haberman,hue='status',size=6)\
   .map(plt.scatter,'age','year')\
.add_legend();
plt.show()
```



Observations:

1)Here we had performed using (age,year) we didn't distingiush properly

2) Seperating the status 1 and 2 is difficult

In [45]:
```
#Pair Plot
```

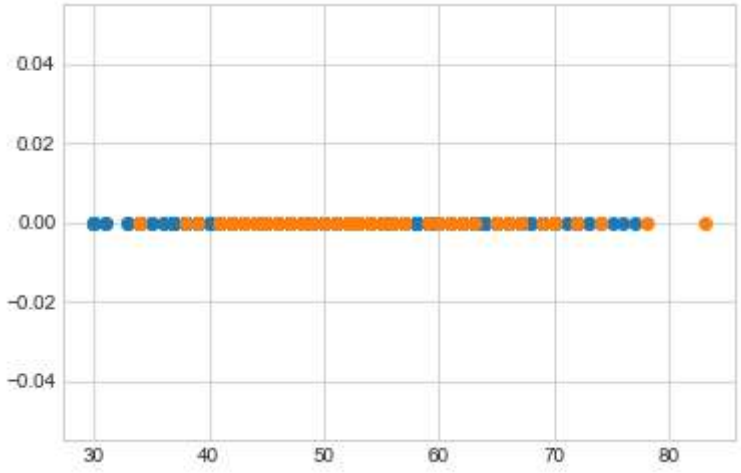
In [44]:
```
g=sns.pairplot(haberman,hue='status',vars=['age','year','nodes'],size=4)
g
```

Out[44]: <seaborn.axisgrid.PairGrid at 0x180f67e1780>

Observations 1) we dont have any feature which can distinguish our class label 2) we have a overlap between satus 1 and 2 3) Therefore we cant find line which can be linearly seperable

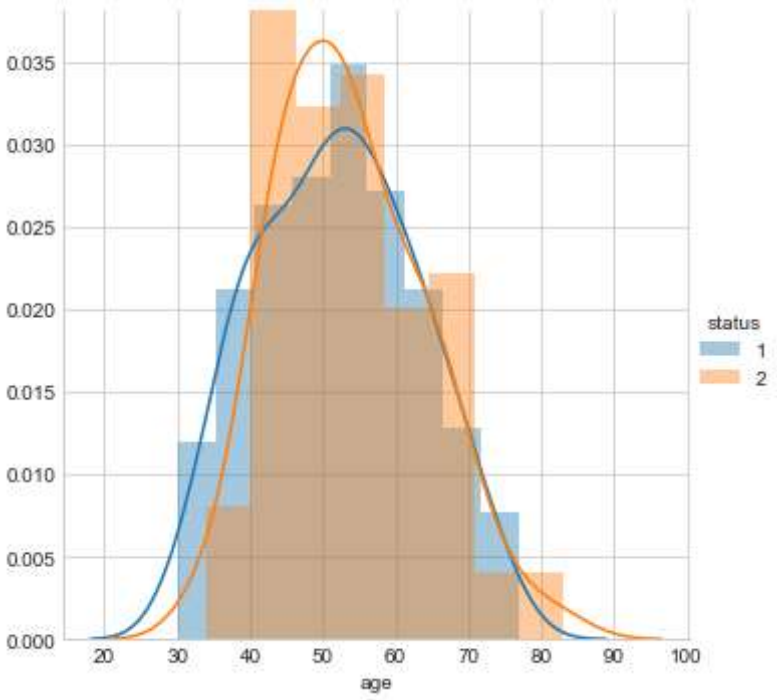
In [52]:
```
#Histogram,PDF,CDf [ 1-D Scatter plot]
import numpy as np
haberman_1=haberman.loc[haberman['status']==1];
haberman_2=haberman.loc[haberman['status']==2];
plt.plot(haberman_1['age'],np.zeros_like(haberman_1['age']),'o')
plt.plot(haberman_2['age'],np.zeros_like(haberman_2['age']),'o')
```

Out[52]:
```
[<matplotlib.lines.Line2D at 0x1808029f6a0>]
```



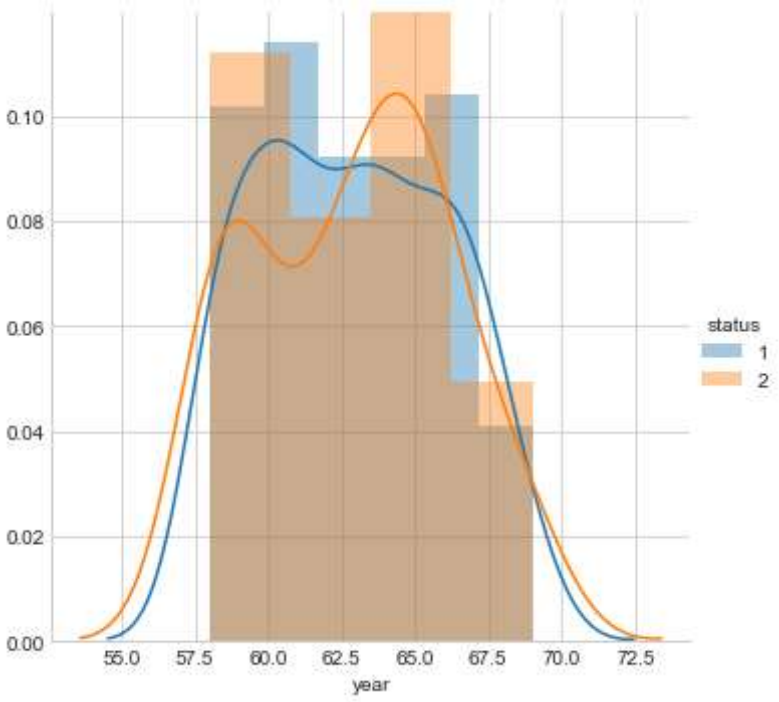
In [53]:
```
sns.FacetGrid(haberman, hue="status", size=5) \
   .map(sns.distplot, "age") \
   .add_legend();
plt.show();
```

```
C:\Users\Sai Preenith\Anaconda3\lib\site-packages\matplotlib\axes\_axes.py:64
62: UserWarning: The 'normed' kwarg is deprecated, and has been replaced by t
he 'density' kwarg.
  warnings.warn("The 'normed' kwarg is deprecated, and has been "
C:\Users\Sai Preenith\Anaconda3\lib\site-packages\matplotlib\axes\_axes.py:64
62: UserWarning: The 'normed' kwarg is deprecated, and has been replaced by t
he 'density' kwarg.
  warnings.warn("The 'normed' kwarg is deprecated, and has been "
```



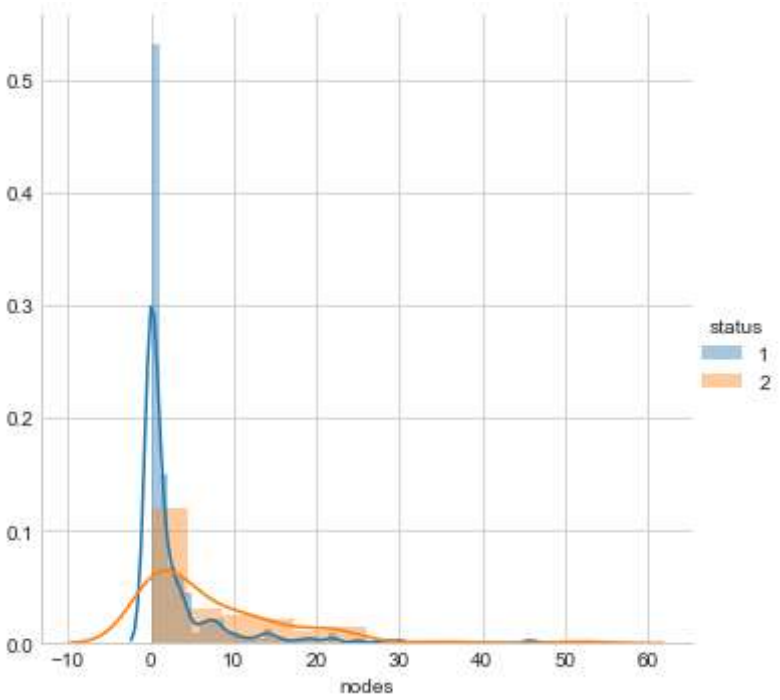
In [54]:
```
sns.FacetGrid(haberman, hue="status", size=5) \
   .map(sns.distplot, "year") \
   .add_legend();
plt.show();
```

```
C:\Users\Sai Preenith\Anaconda3\lib\site-packages\matplotlib\axes\_axes.py:64
62: UserWarning: The 'normed' kwarg is deprecated, and has been replaced by t
he 'density' kwarg.
  warnings.warn("The 'normed' kwarg is deprecated, and has been "
C:\Users\Sai Preenith\Anaconda3\lib\site-packages\matplotlib\axes\_axes.py:64
62: UserWarning: The 'normed' kwarg is deprecated, and has been replaced by t
he 'density' kwarg.
  warnings.warn("The 'normed' kwarg is deprecated, and has been "
```



In [55]:
```
sns.FacetGrid(haberman, hue="status", size=5) \
   .map(sns.distplot, "nodes") \
   .add_legend();
plt.show()
```

```
C:\Users\Sai Preenith\Anaconda3\lib\site-packages\matplotlib\axes\_axes.py:64
62: UserWarning: The 'normed' kwarg is deprecated, and has been replaced by t
he 'density' kwarg.
  warnings.warn("The 'normed' kwarg is deprecated, and has been "
C:\Users\Sai Preenith\Anaconda3\lib\site-packages\matplotlib\axes\_axes.py:64
62: UserWarning: The 'normed' kwarg is deprecated, and has been replaced by t
he 'density' kwarg.
  warnings.warn("The 'normed' kwarg is deprecated, and has been "
```



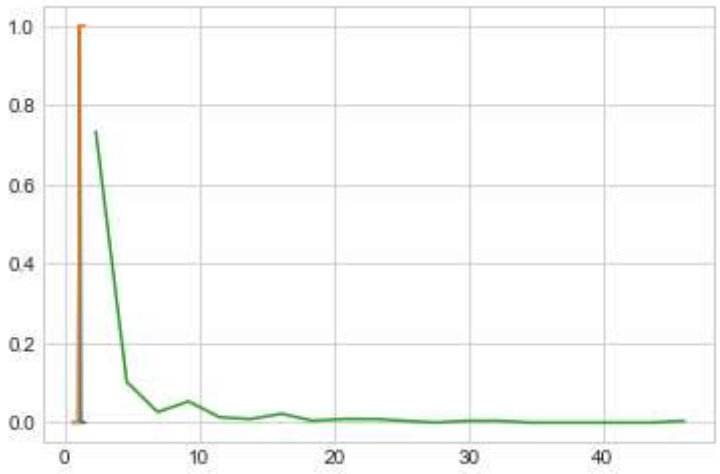
Observations: 1)As you can see age and year has high overlap and nodes has less overlap so nodes>year>age

In [57]:
```
counts, bin_edges = np.histogram(haberman_1['status'], 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)


counts, bin_edges = np.histogram(haberman_1['nodes'], bins=20,
                                 density = True)
pdf = counts/(sum(counts))
plt.plot(bin_edges[1:],pdf);

plt.show();
```

```
[0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
[0.5 0.6 0.7 0.8 0.9 1.  1.1 1.2 1.3 1.4 1.5]
```



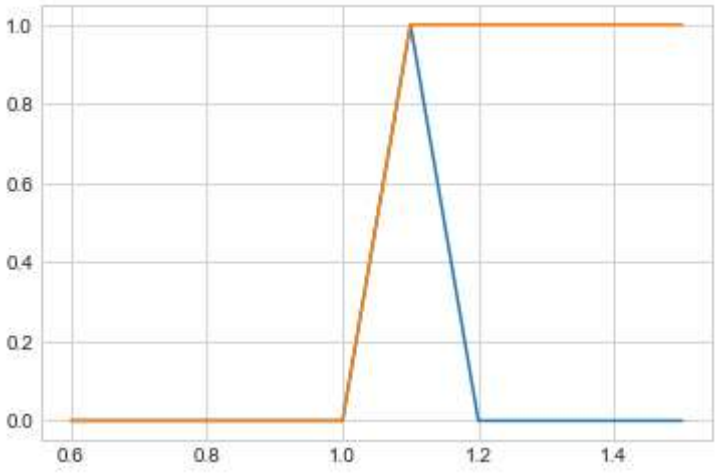
In [58]:
```
counts, bin_edges = np.histogram(haberman_1['status'], bins=10,
                                 density = True)
pdf = counts/(sum(counts))
print(pdf);
print(bin_edges)

#compute CDF
cdf = np.cumsum(pdf)
plt.plot(bin_edges[1:],pdf)
plt.plot(bin_edges[1:], cdf)



plt.show();
```

```
[0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
[0.5 0.6 0.7 0.8 0.9 1.  1.1 1.2 1.3 1.4 1.5]
```

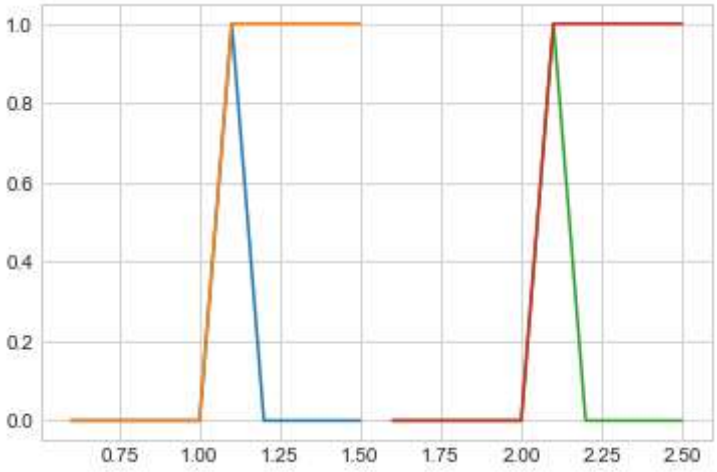


In [59]:
```
#1
counts, bin_edges = np.histogram(haberman_1['status'], bins=10,
                                 density = True)
pdf = counts/(sum(counts))
print(pdf);
print(bin_edges)
cdf = np.cumsum(pdf)
plt.plot(bin_edges[1:],pdf)
plt.plot(bin_edges[1:], cdf)


# 2
counts, bin_edges = np.histogram(haberman_2['status'], 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)
```

```
[0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
[0.5 0.6 0.7 0.8 0.9 1.  1.1 1.2 1.3 1.4 1.5]
[0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
[1.5 1.6 1.7 1.8 1.9 2.  2.1 2.2 2.3 2.4 2.5]
```

Out[59]: [<matplotlib.lines.Line2D at 0x18082171828>]



In [61]:
```
#Mean,Variance and standard deviation
#Median, Quantiles, Percentiles, IQR.
print("\nMedians:")
print(np.median(haberman_1["nodes"]))
#Median with an outlier
print(np.median(np.append(haberman_1["nodes"],50)));
print(np.median(haberman_2["nodes"]))


print("\nQuantiles:")
print(np.percentile(haberman_1["nodes"],np.arange(0, 100, 25)))
print(np.percentile(haberman_2["nodes"],np.arange(0, 100, 25)))


print("\n90th Percentiles:")
print(np.percentile(haberman_1["nodes"],90))
print(np.percentile(haberman_2["nodes"],90))


from statsmodels import robust
print ("\nMedian Absolute Deviation")
print(robust.mad(haberman_1["nodes"]))
print(robust.mad(haberman_2["nodes"]))
```

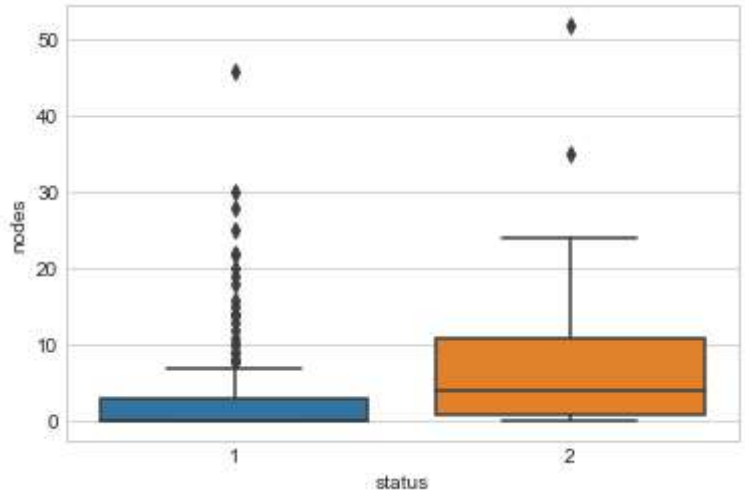
```
Medians:
0.0
0.0
4.0

Quantiles:
[0. 0. 0. 3.]
[ 0.  1.  4. 11.]

90th Percentiles:
8.0
20.0

Median Absolute Deviation
0.0
5.930408874022408
```

In [62]:
```
#Box plot and whiskers
sns.boxplot(x='status',y='nodes', data=haberman)
plt.show()
```



In [63]:
```
#Violin Plot

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