Haberman Data Analysis Assignmet 2

In [1]:

```
##Haberman data set analysis
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [2]:

```
##Loading the haber man data set as a dataframe
Haberman = pd.read_csv("haberman.csv")
```

In [3]:

```
##Insights of dataframe
##Knowing the top rows and last rows of dataframe
print("Head of the dataset is \n {}\n".format(Haberman.head()))
print("Tail of the dataset is \n {}\n".format(Haberman.tail()))
print("Shape of dataset is \n{}\n".format(Haberman.shape))
print("Column names of dataset is \n {}\n".format(Haberman.columns))
```

```
Head of the dataset is
    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

Tail of the dataset is
      age  year  nodes  status
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

Shape of dataset is
(306, 4)

Column names of dataset is
 Index(['age', 'year', 'nodes', 'status'], dtype='object')
```

Observations

1. There are 4 columns age,year,nodes,status.
2. There are 306 records of data.
3. Age,Year,Nodes are features to identify the surival period of patient(status).
4. Here nodes are lymph nodes which leads to cancer spread of cancer.
5. Staus is 1 if people survives more than 5 years, if it is less than 5 years it is 2.

In [4]:

```
## Tocheck is there any null values
##Get those rows which have null values
Haberman[Haberman.isnull().any(axis = 1)]
```

Out[4]:

age	year	nodes	status

In [5]:

```
##To drop NaN Values from the data frame
##Drop those rows which have Nan Values
Haberman[~Haberman.dropna().any(axis = 1)]
```

Out[5]:

age	year	nodes	status

In [6]:

```
##Overview of the data set
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]:

```
## Checking the uniformity of dataset
print("status \n {} \n".format(Haberman['status'].value_counts()))
```

```
status
 1    225
2     81
Name: status, dtype: int64
```

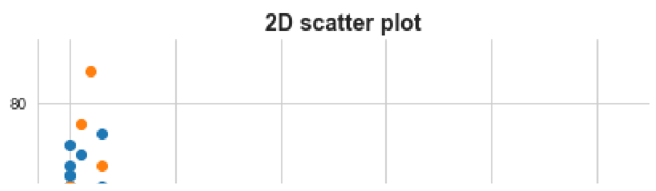
Conclusion

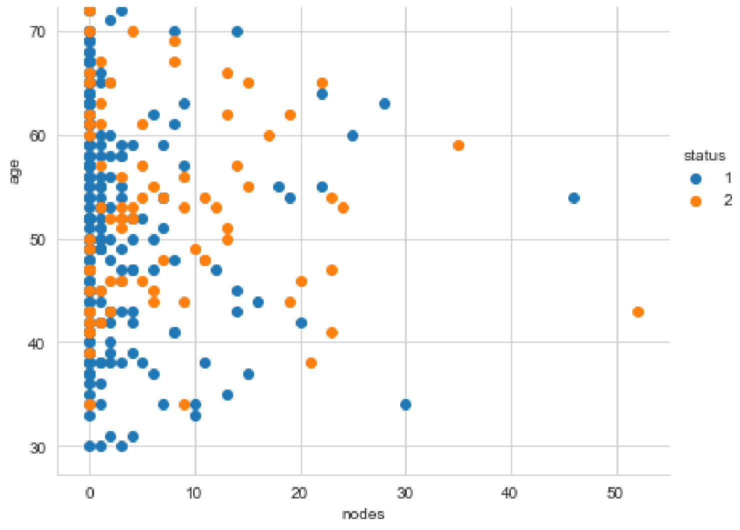
1. 225 people survived more than 5 years after the surgery
2. 81 people survived less than 5 years after the surgery

In [8]:

```
## 2D Scatter plot for all features
sns.set_style('whitegrid')
sns.despine(left=True)
#sns.FacetGrid(Haberman, hue = "status" , size = 6 ).map(plt.scatter,"age","year").add_legend()
sns.FacetGrid(Haberman,hue = "status", height = 6).map(plt.scatter,'nodes','age').add_legend()
plt.title("2D scatter plot",fontsize = 14, fontweight = 'bold')
plt.show()
```

```
<Figure size 432x288 with 0 Axes>
```





Observations

1. Of all the features age vs nodes gives more clarity than other features
2. Majority of people who survivesd more than 5 years are having 0 to 10 nodes
3. But most of the data is widely spread

In [9]:

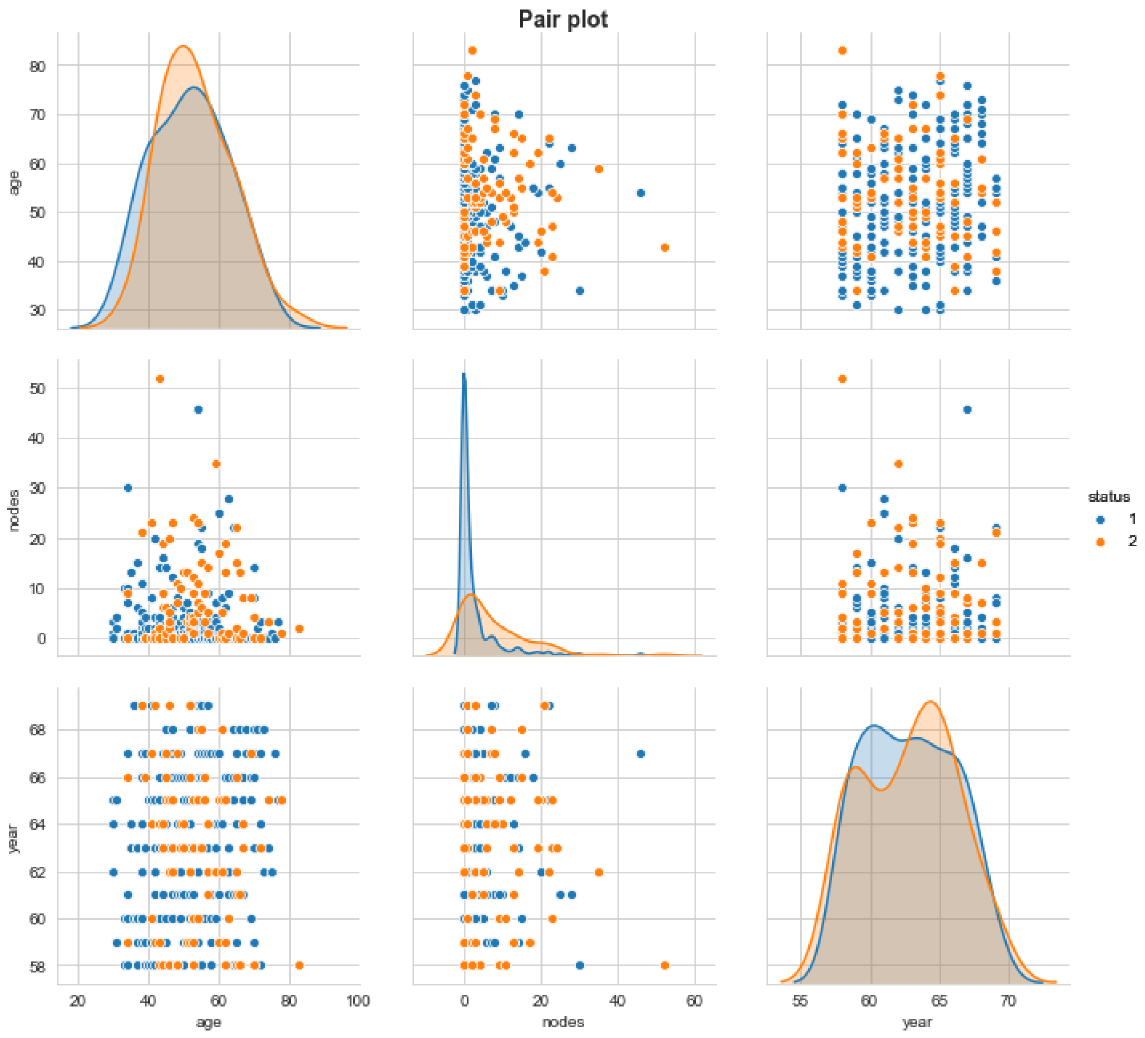
```
import plotly.express as px
#df = px.data.Haberman()
```

In [10]:

```
fig = px.scatter_3d(Haberman, x='age', y='year', z='nodes',
                    color='status', symbol='status')
fig.show()
```

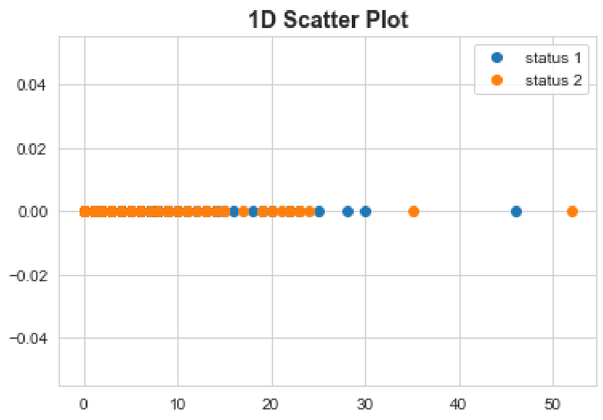
In [11]:

```
## Pair Plots
## Pair Plots
plt.close()
sns.set_style('whitegrid')
sns.pairplot(Haberman,vars = ['age','nodes','year'], hue='status',height = 3).add_legend().\
fig.suptitle("Pair plot",y = 1.0,fontsize = 14, fontweight = 'bold')
plt.show()
```



In [12]:

```
##Plotting the 1-D scatter plot based on status =1 or status =2
Haberman_status1 = Haberman[Haberman['status'] == 1]
Haberman_status2 = Haberman[Haberman['status'] == 2]
plt.plot(Haberman_status1['nodes'],np.zeros_like(Haberman_status1['nodes']),'o',label = 'status 1'
)
plt.plot(Haberman_status2['nodes'],np.zeros_like(Haberman_status2['nodes']),'o',label = 'status 2'
)
plt.legend()
plt.title("1D Scatter Plot",fontsize = 14, fontweight = 'bold')
plt.show()
```

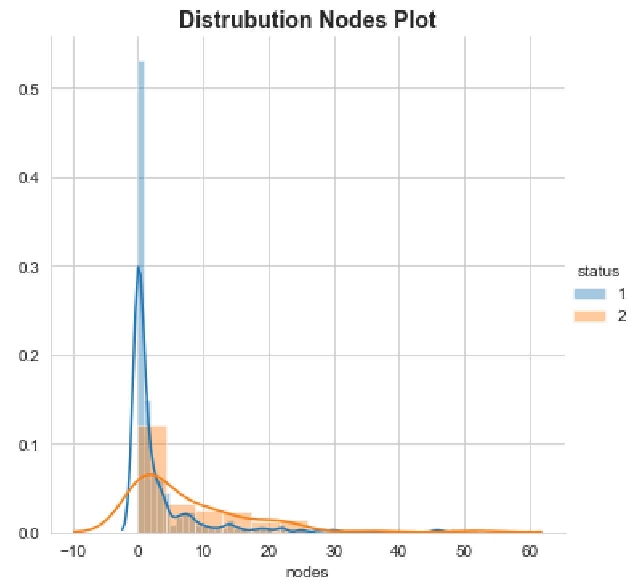


Observations

1. Cannot able to see the status 1 plots

In [14]:

```
###Distrubution plot
sns.FacetGrid(Haberman,hue= 'status',height= 5).map(sns.distplot,'nodes').add_legend()
plt.title("Distrubution Nodes Plot",fontsize =14, fontweight= 'bold')
plt.show()
```

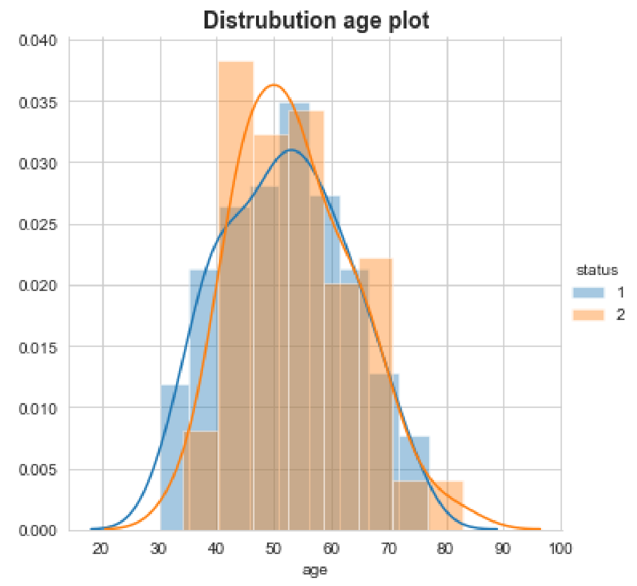


Observations

1. Both the status are overlapping with each other
2. staus 1 ranges from -2 to 10
3. status2 widely ranges from -10 to 25

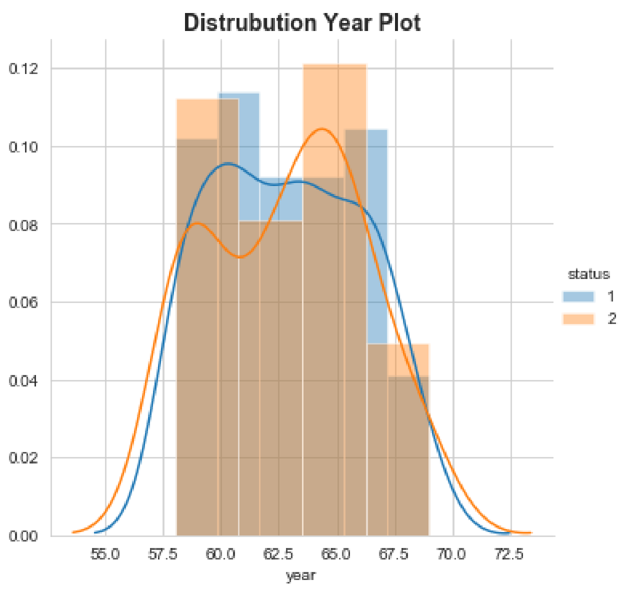
In [15]:

```
sns.FacetGrid(Haberman,hue= 'status',size = 5).map(sns.distplot,'age').add_legend()
plt.title("Distrubution age plot",fontsize =14, fontweight = 'bold')
plt.show()
```



In [16]:

```
sns.FacetGrid(Haberman,hue = 'status', size = 5).map(sns.distplot,'year').add_legend()
plt.title("Distrubution Year Plot", fontsize =14, fontweight = 'bold')
plt.show()
```

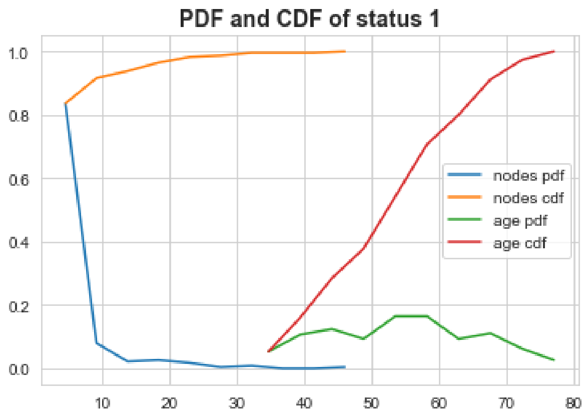


In [17]:

```
##Plotting a histogram

##Cdf and pdf for nodes based
counts,bin_edges = np.histogram(Haberman_status1['nodes'],bins =10 , density = True)
pdf = counts/sum(counts)
cdf = np.cumsum(pdf)
plt.plot(bin_edges[1:],pdf,label = 'nodes pdf')
plt.plot(bin_edges[1:],cdf,label = 'nodes cdf')


##Cdf and pdf for age based
counts,bin_edges = np.histogram(Haberman_status1['age'],bins = 10, density = True)
pdf = counts/sum(counts)
plt.plot(bin_edges[1:],pdf,label = 'age pdf')
cdf = np.cumsum(pdf)
plt.plot(bin_edges[1:],cdf,label = 'age cdf')
plt.legend()
plt.title("PDF and CDF of status 1",fontsize = 14 ,fontweight = 'bold')
```

Out[17]:

```
Text(0.5, 1.0, 'PDF and CDF of status 1')
```



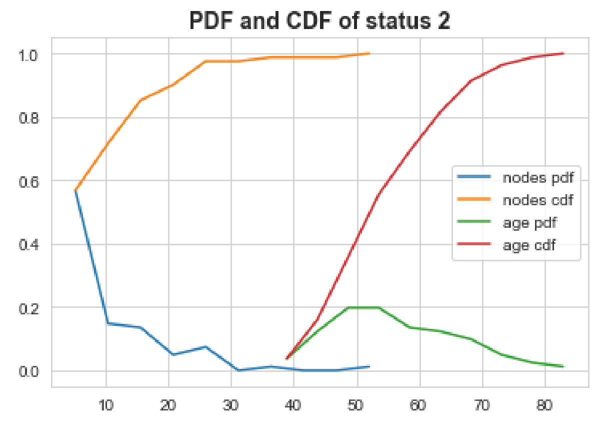
Obeservartions

[illegible]

1. About 90% of people who survived more than 5 years have lymph nodes in range of 4 to 9
2. 70% of the people who survived are in the age group of 40 to 65
3. We cannot come to a conclusion here because around 70% of people having less than 9 nodes are in status2

In [18]:

```
##PDF and CDF for Year based
counts, bin_edges = np.histogram(Haberman_status2['nodes'],bins =10)
pdf = counts/sum(counts)
cdf = np.cumsum(pdf)
plt.plot(bin_edges[1:],pdf,label = 'nodes pdf')
plt.plot(bin_edges[1:],cdf,label = 'nodes cdf')

counts,bin_edges = np.histogram(Haberman_status2['age'],bins =10)
pdf = counts/sum(counts)
cdf = np.cumsum(pdf)
plt.plot(bin_edges[1:],pdf, label = 'age pdf')
plt.plot(bin_edges[1:],cdf,label = 'age cdf')
plt.legend()
plt.title("PDF and CDF of status 2",fontsize = 14 ,fontweight = 'bold')
plt.show()
```



Observations

1. 60% of people are more than 50 years old who did not survive more than 5 years
2. 30% of people are having more than 10 lymph nodes
3. With these two features age and nodes are very helpful in getting the number of people who survived
4. Simple if else model will if age > 50 and nodes > 3 then status 2

In [19]:

```
### Mean , Variance ,standard deviation

print ("\nStatus1")
print ("Mean age: ",np.mean(Haberman_status1['age']))
print ("Mean Lymph nodes: ",np.mean(Haberman_status1['nodes']))
print ("Status2")
print ("Mean age: ",np.mean(Haberman_status2['age']))
print ("Average Lymph nodes: ",np.mean(Haberman_status2['nodes']))

print ("\nVariance: ")

print ("Status1")
print ("Age: ",np.var(Haberman_status1['age']))
print ("Nodes: ",np.var(Haberman_status1['nodes']))
print ("Status2")
print ("Age: ",np.var(Haberman_status2['age']))
print ("Nodes: ",np.var(Haberman_status2['nodes']))

print ("\nStandard Deviation: ")

print ("Status1")
print ("Age: ",np.std(Haberman_status1['age']))
print ("Nodes: ",np.std(Haberman_status1['nodes']))
print ("Status2")
print ("Age: ",np.std(Haberman_status2['age']))
print ("Nodes: ",np.std(Haberman_status2['nodes']))
```

```
Status1
Mean age:  52.017777777777778
Mean Lymph nodes:  2.7911111111111113
Status2
Mean age:  53.67901234567901
Average Lymph nodes:  7.45679012345679

Variance:
Status1
Age:  120.72857283950623
Nodes:  34.30747654320981
Status2
Age:  102.09449778997102
Nodes:  83.3345526596555

Standard Deviation:
Status1
Age:  10.98765547510051
Nodes:  5.857258449412131
Status2
Age:  10.10418219303131
Nodes:  9.128776076761632
```

In [20]:

```
### Median, Percentile, Quartile , IDR , MAD(Median Absolute Deviation )
print ("\nMedian")
print ("Status1")
print ("Age: ",np.median(Haberman_status1['age']))
print ("nodes: ",np.median(Haberman_status1['nodes']))
print ("Status2")
print ("age: ",np.median(Haberman_status2['age']))
print ("nodes: ",np.median(Haberman_status2['nodes']))

print ("\nQuantiles: ")

print ("Status1")
print ("Age: ",np.percentile(Haberman_status1['age'],np.arange(0,101,25)))
print ("Nodes: ",np.percentile(Haberman_status1['nodes'],np.arange(0,101,25)))
print ("Status2")
print ("Age: ",np.percentile(Haberman_status2['age'],np.arange(0,101,25)))
print ("Nodes: ",np.percentile(Haberman_status2['nodes'],np.arange(0,101,25)))

print ("\nPercentiles: ")
print ("Status1")
print ("Age: ",np.percentile(Haberman_status1['age'],90))
print ("Nodes: ",np.percentile(Haberman_status1['nodes'],90))
print ("Status2")
print ("Age: ",np.percentile(Haberman_status2['age'],90))
print ("Nodes: ",np.percentile(Haberman_status2['nodes'],90))

print ("MedianAbsolute deviation")
from statsmodels import robust
print ("Status1")
print ("Age: ",robust.mad(Haberman_status1['age']))
print ("Nodes: ",robust.mad(Haberman_status1['nodes']))
print ("Status2")
print ("Age: ",robust.mad(Haberman_status2['age']))
print ("Nodes: ",robust.mad(Haberman_status2['nodes']))
```

```
Median
Status1
Age:  52.0
nodes:  0.0
Status2
age:  53.0
nodes:  4.0

Quantiles:
Status1
Age:  [30. 43. 52. 60. 77.]
Nodes:  [ 0.  0.  0.  3. 46.]
```

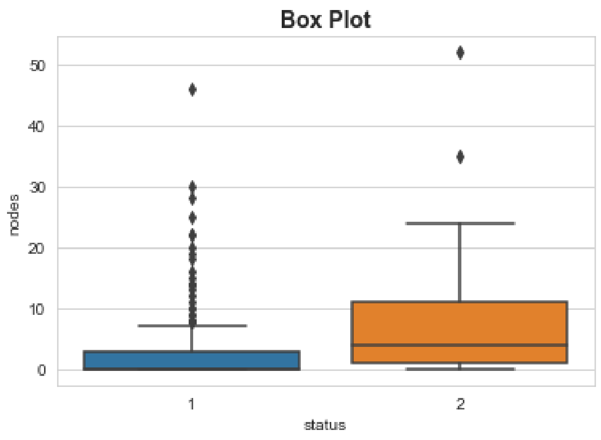
```
Status2
Age:  [34. 46. 53. 61. 83.]
Nodes:  [ 0.  1.  4. 11. 52.]

Percentiles:
Status1
Age:  67.0
Nodes:  8.0
Status2
Age:  67.0
Nodes:  20.0
MedianAbsolute deviation
Status1
Age:  13.343419966550417
Nodes:  0.0
Status2
Age:  11.860817748044816
Nodes:  5.930408874022408
```

Observations

1. Of all the plots this will give clears insight view of data though some data is missing
2. Using the above data we can build a simple if else model which identify status 1 or status 2
3. For status 1 if age < 60 and nodes <= 3 then status 1 else status 2 --> 25% error
4. For status 2 if age >53 and node > 4 then status 2 --> 25% error

In [21]:

```
##Box plot
## By this plot we can identify the Quartiles
sns.boxplot(x='status',y='nodes', data= Haberman)
plt.title("Box Plot",fontsize =14, fontweight ='bold')
plt.show()
```

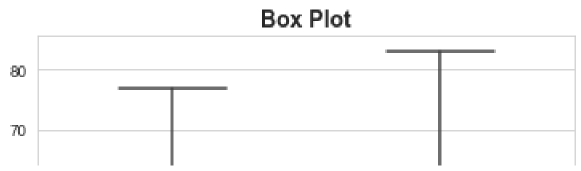


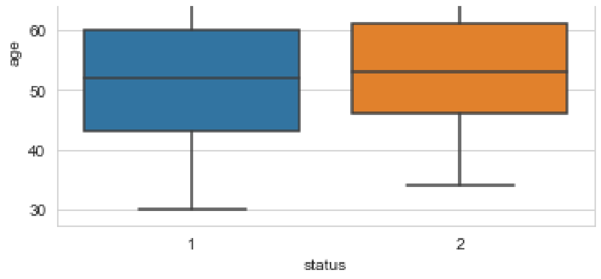
Observations:

1. There are so many outliers in nodes of status1
2. Only two outliers are there in nodes staus 2
3. Using single feature nodes we cannot distinguish status1 and status2

In [22]:

```
sns.boxplot(x= 'status',y= 'age',data = Haberman)
plt.title("Box Plot",fontsize =14, fontweight ='bold')
plt.show()
```



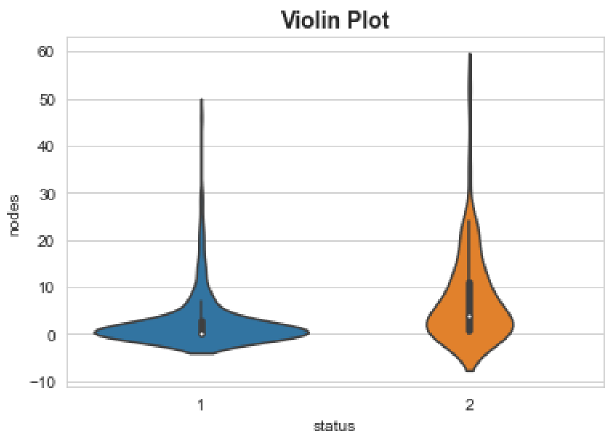


Observations:

1. The quantile of both the plots very similar cannot distinguish them based on age
2. Age and nodes will be crucial in predicting the status

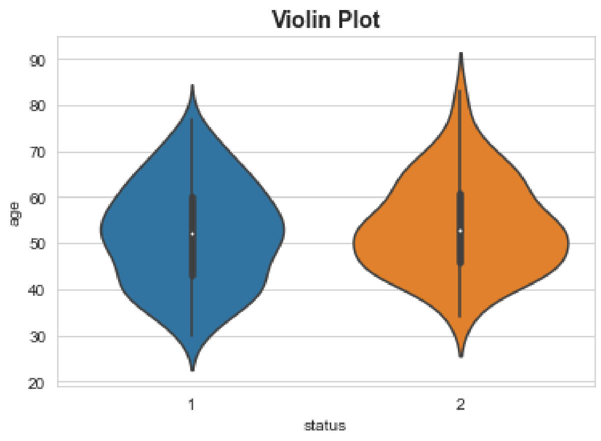
In [23]:

```
##Violin plots
## Mixture of both Box and PDF
sns.violinplot(x="status", y="nodes", data=Haberman, size=8)
plt.title("Violin Plot",fontsize =14, fontweight ='bold')
plt.show()
```



In [24]:

```
sns.violinplot(x= "status",y="age",data= Haberman,size = 8)
plt.title("Violin Plot",fontsize =14, fontweight ='bold')
plt.show()
```



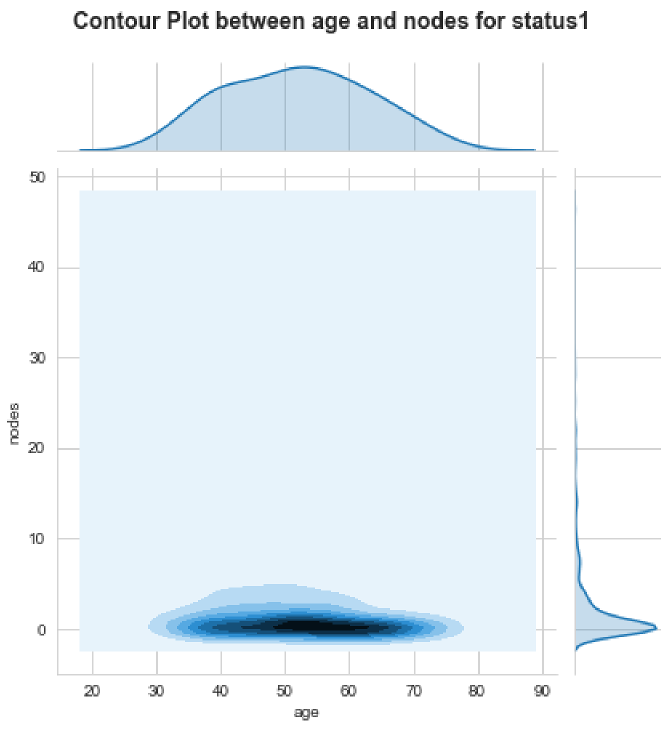
Observations

1. feature age is overlapped with each other
2. Even in nodes most of the status1 nodes are overlapped with statu2 nodes
3. Cannot able to calculate the error as two features are involved

3. Cannot able to calculate the error as two features are involved

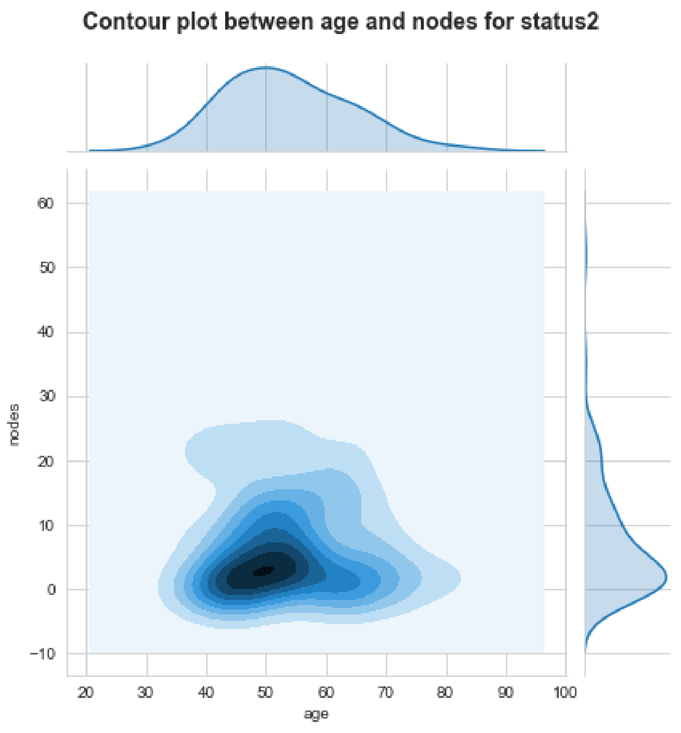
In [25]:

```
## Contour plot
## plotting age vs nodes
sns.jointplot(x="age" ,y = "nodes", data= Haberman_status1,kind = "kde").\
fig.suptitle("Contour Plot between age and nodes for status1",y = 1.05,fontsize = 14, fontweight =
'bold')
plt.show()
```



In [26]:

```
sns.jointplot(x="age",y= "nodes",data= Haberman_status2,kind = "kde").\
fig.suptitle('Contour plot between age and nodes for status2',y=1.05,fontsize = 14, fontweight =
'bold')
plt.show()
```



Conclusion:

1. Overall data is overlapping with each other when tried to predict using one feature.
2. Value-counts helps in identifying number of people in status 1 and status 2 which are 225 and 81 respectively.
3. Distrubution plot helps in checking the overlap of data among various features.
4. Histogram and CDF helps in identifying the age and nodes as crucial factors
5. Finding the quantiles placed a major role in seperating the two data
6. With Box plot we can be able to say that there are many outliers in status 1,whereas in status 2 only 2 outliers are present
7. Contour plot is also useful in finding the important insights
8. We can construct a simple if else model using age vs nodes as features. For status 1 if (age < 60 and nodes <= 3): status 1 else: status 2 For status 2 if (age >53 and node > 4): status 2 else : status 1