DATA VISUALIZATION WITH HABERMAN DATASET

Relevant Information

The dataset 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 undergone surgery for breast cancer

It contains :-

306 datapoints/rows 4 fearures includeing class label

Attributes/Features Information

There are 4 features including class label/dependent variable.

<> 30 - It represents age of patient at the time of operation(numerical).

<> 64 - It represents year of operation(numerical)

<> 1 - It tells no of +ve auxillry node detected(numerical).

<> 1.1 - Survival status 1 = the patient survived 5 years or longer 2 = the patient died within 5 year

Importing libraries And Reading The Data

In [7]:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np

haberman=pd.read_csv("haberman.csv")
```

In [8]:

```
print(haberman.shape)
```

```
(306, 4)
```

In [9]:

```
print(haberman.columns)
```

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

In [10]:

```
## It gives you the top 5 rows(data-points).
haberman.head()
```

Out[10]:

	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 [11]:

```
# To look at the last 5 rows(data-points), we can also specify that how many data-points we want to see.
haberman.tail()
```

Out[11]:

	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

In [12]:

```
# To know about data summary
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: 9.7 KB
```

In [13]:

```
# To know statistical summary of data which is very important
haberman.describe()
```

Out[13]:

	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 [14]:

```
# To know number of data-points for each class.
# As it is not balanced dataset, it is imbalanced dataset because the number of data-points for both of the class are significantly different.
haberman.status.value_counts()
```

Out[14]:

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

Bivariate Analysis

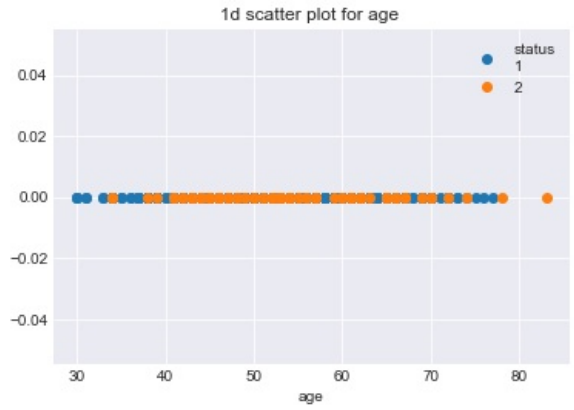
Bivariate analysis is one of the simplest forms of quantitative (statistical) analysis. It involves the analysis of two variables (often denoted as X, Y), for the purpose of determining the empirical relationship between them

Scatter Plot

A scatter plot is a useful visual representation of the relationship between two numerical variables (attributes) and is usually drawn before working out a linear correlation or fitting a regression line. The resulting pattern indicates the type (linear or non-linear) and strength of the relationship between two variables.

In [22]:

```
# 1-d scatter plot

one = haberman.loc[haberman["status"] == 1]
two = haberman.loc[haberman["status"] == 2]
plt.plot(one["age"], np.zeros_like(one["age"]), 'o', label = "status\n" "1")
plt.plot(two["age"], np.zeros_like(two["age"]), 'o', label = "2")
plt.title("1d scatter plot for age ")
plt.xlabel("age")

plt.legend()
plt.show()
```



observation

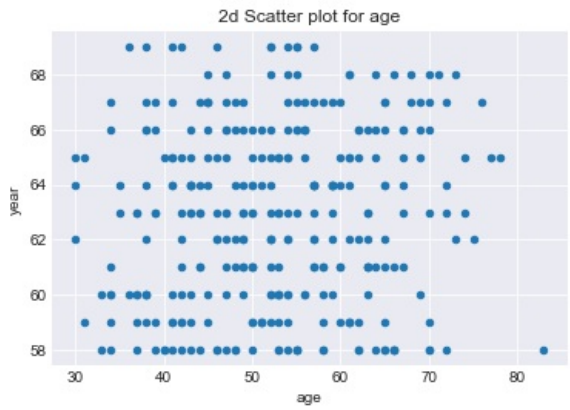
1.After looking at this plot we can easily count number of points that are there in age range who survived or not.

2.Many person died whose age was between 41-70.

2-D SCATTER PLOT

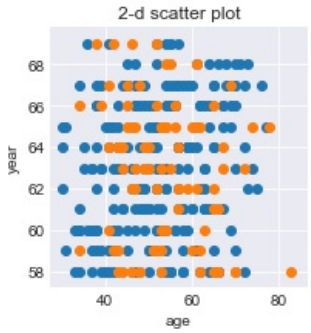
In [23]:

```
haberman.plot(kind = "scatter", x = "age", y = "year")
plt.title("2d Scatter plot for age")
plt.show()
```



In [31]:

```
sns.set_style('darkgrid')
sns.FacetGrid(haberman, hue = "status", size = 3).map(plt.scatter, "age", "year").add_legend
plt.title('2-d scatter plot')
plt.show()
```



In [32]:

```
sns.set_style("darkgrid")
sns.FacetGrid(haberman, hue = "status", size = 3).map(plt.scatter, "age", "nodes").add_legend()
plt.title("2d scatter plot for age and  node")
plt.show()
```

observation

1.In the above 2d scatter plot class label(i.e. a person died or survived) is not linearly seprable

PAIR PLOT

In [26]:

```
plt.close();
sns.set_style('darkgrid');
sns.pairplot(haberman,hue='status',size=3);
plt.show()
```

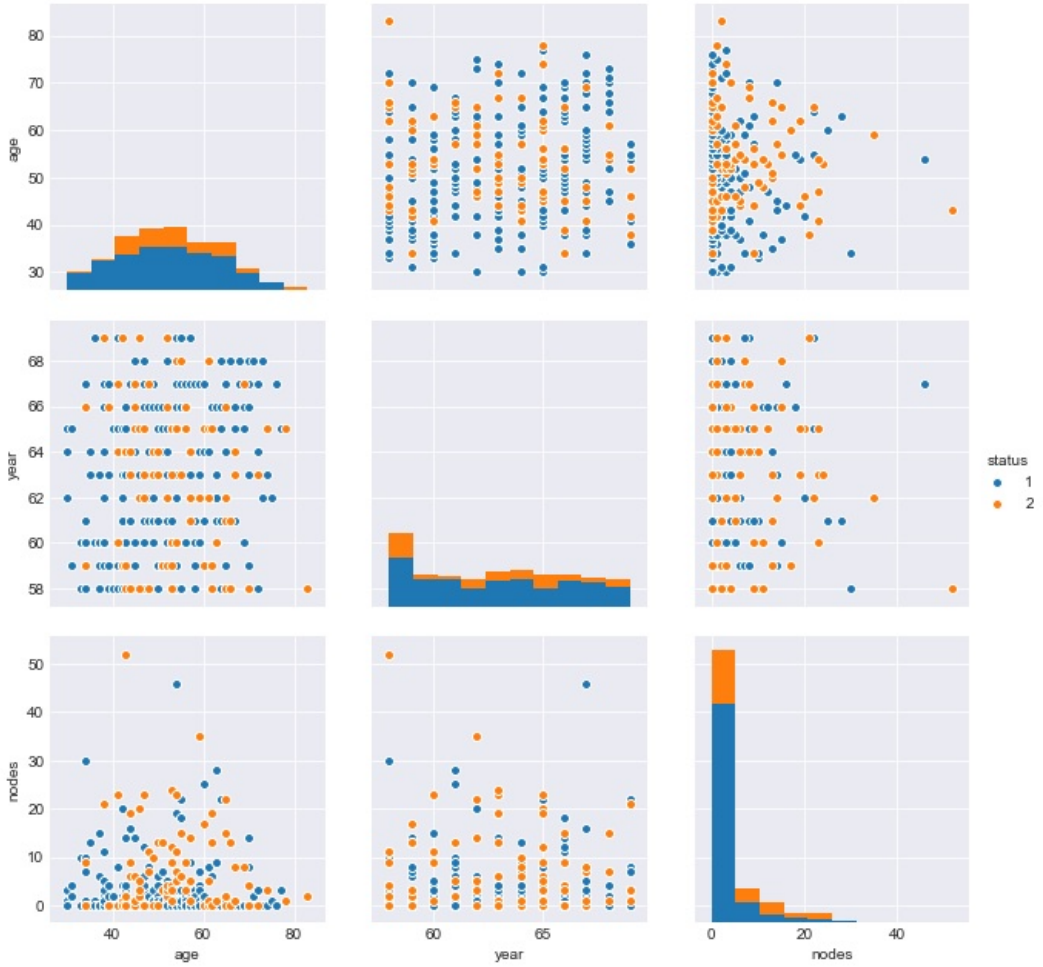
observation

1.As we are unable to classify which is the most useful feature because of too much overlapping. But, Somehow we can say, In year, 60-65 more person died who has less than 6 nodes.

2.And hence, this plot is not much informative in this case.

In [29]:

```
plt.close()
sns.set_style("darkgrid");
sns.pairplot(haberman, hue="status",vars = ['age', 'year', 'nodes'], size = 3)
plt.show()
```



<> Plot between nodes and age is very understanding

Univariate Analysis(pdf, cdf, boxplot and violin plot)

HISTOGRAM, PDF, CDF

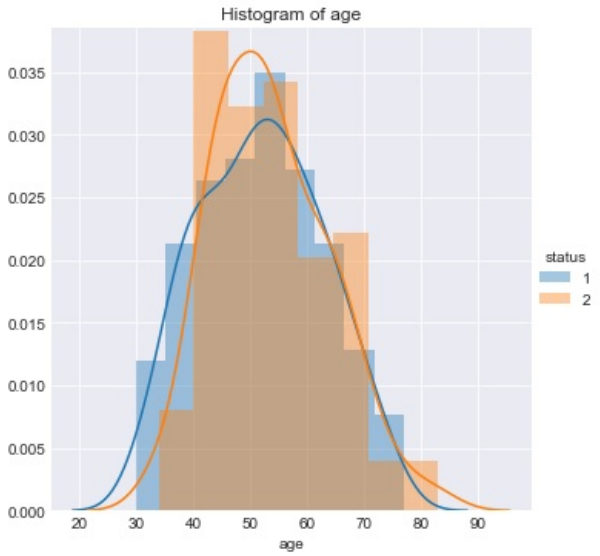
Univariate analysis is the simplest form of analyzing data. “Uni” means “one”, "variate" means "variable or numeric variable" so, in other words your data has only one variable. It doesn't deal with causes or relationships (unlike regression) and it's major purpose is to describe; it takes data, summarizes that data and finds patterns in the data.

PDF(Probability Density Function)

In [45]:

```
sns.FacetGrid(haberman, hue='status',size=5) .map(sns.distplot, "age").add_legend()
plt.title('Histogram of age')
plt.show();
```

```
C:\Users\user\Anaconda3\lib\site-packages\seaborn\distributions.py:218: MatplotlibDeprecationWarning
:
The 'normed' kwarg was deprecated in Matplotlib 2.1 and will be removed in 3.1. Use 'density' instea
d.
  color=hist_color, **hist_kws)
C:\Users\user\Anaconda3\lib\site-packages\seaborn\distributions.py:218: MatplotlibDeprecationWarning
:
The 'normed' kwarg was deprecated in Matplotlib 2.1 and will be removed in 3.1. Use 'density' instea
d.
  color=hist_color, **hist_kws)
```



Observations;

1. The histogram is plotted based on single variable
2. It is also called univariate analysis

In [48]:

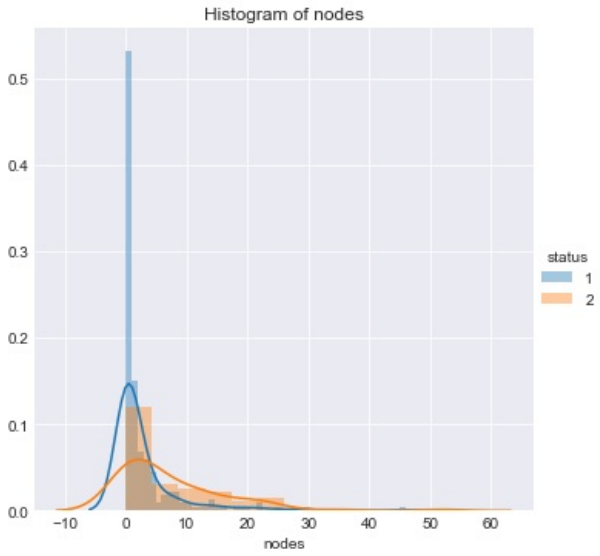
```
sns.FacetGrid(haberman,hue='status',size=5).map(sns.distplot,'year').add_legend()
plt.title('Histogram of year')
plt.show()
```

```
C:\Users\user\Anaconda3\lib\site-packages\seaborn\distributions.py:218: MatplotlibDeprecationWarning
:
The 'normed' kwarg was deprecated in Matplotlib 2.1 and will be removed in 3.1. Use 'density' instea
d.
  color=hist_color, **hist_kws)
C:\Users\user\Anaconda3\lib\site-packages\seaborn\distributions.py:218: MatplotlibDeprecationWarning
:
The 'normed' kwarg was deprecated in Matplotlib 2.1 and will be removed in 3.1. Use 'density' instea
d.
  color=hist_color, **hist_kws)
```

In [49]:

```
sns.FacetGrid(haberman,hue='status',size=5).map(sns.distplot,'nodes').add_legend()
plt.title('Histogram of nodes')
plt.show()
```

```
C:\Users\user\Anaconda3\lib\site-packages\seaborn\distributions.py:218: MatplotlibDeprecationWarning
:
The 'normed' kwarg was deprecated in Matplotlib 2.1 and will be removed in 3.1. Use 'density' instea
d.
  color=hist_color, **hist_kws)
C:\Users\user\Anaconda3\lib\site-packages\seaborn\distributions.py:218: MatplotlibDeprecationWarning
:
The 'normed' kwarg was deprecated in Matplotlib 2.1 and will be removed in 3.1. Use 'density' instea
d.
  color=hist_color, **hist_kws)
```



observation

1.In all the plots the features are overlapping each other massively. But somehow we can say

2.probabily 58% people survived who had 0-5 nodes and 12% died as well.

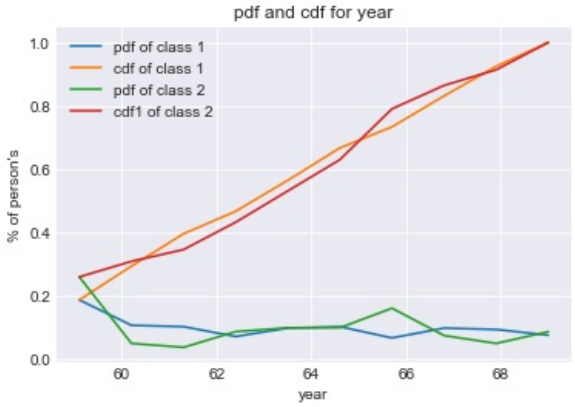
CDF(Cummulative Distributed Function)

In [67]:

```
# cdf gives you cumulative probability associated with a function.
# Cumulative sum of area under curve upto any point for which you are looking for gives you cdf
# class 1 means survived
# class 2 means not survived


one = haberman.loc[haberman["status"] == 1]
two = haberman.loc[haberman["status"] == 2]
label = ["pdf of class 1", "cdf of class 1", "pdf of class 2", "cdf of class 2"]

counts, bin_edges = np.histogram(one["age"], bins=10, density = True)
pdf = counts/(sum(counts))
cdf = np.cumsum(pdf)
plt.title("pdf and cdf for age")
plt.xlabel("age")
plt.ylabel("% of persons")
plt.plot(bin_edges[1:], pdf)
plt.plot(bin_edges[1:], cdf)


counts, bin_edges = np.histogram(two["age"], bins=10, density = True)
pdf = counts/(sum(counts))
cdf = np.cumsum(pdf)
plt.plot(bin_edges[1:], pdf)
plt.plot(bin_edges[1:], cdf)
plt.legend(label)


plt.show()
```

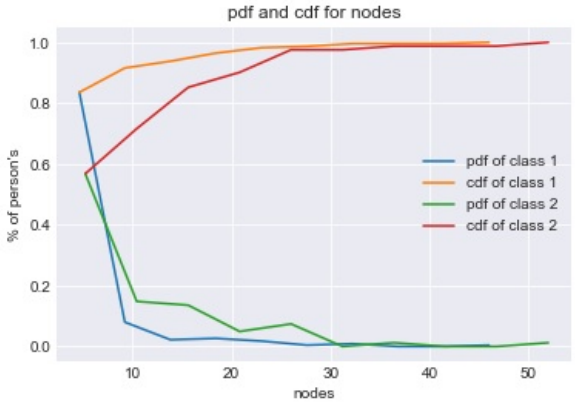
In [57]:

```
label = ["pdf of class 1", "cdf of class 1", "pdf of class 2", "cdf1 of class 2"]
counts, bin_edges = np.histogram(one["year"], bins=10, density = True)
pdf = counts/(sum(counts))
cdf = np.cumsum(pdf)
plt.title("pdf and cdf for year")
plt.xlabel("year")
plt.ylabel("% of person's")
plt.plot(bin_edges[1:], pdf)
plt.plot(bin_edges[1:], cdf)
# plt.legend(class1)



counts, bin_edges = np.histogram(two["year"], bins=10, density = True)
pdf = counts/(sum(counts))
cdf = np.cumsum(pdf)
plt.plot(bin_edges[1:], pdf)
plt.plot(bin_edges[1:], cdf)
plt.legend(label)
plt.show()
```



In [58]:

```
label = ["pdf of class 1", "cdf of class 1", "pdf of class 2", "cdf of class 2"]
counts, bin_edges = np.histogram(one['nodes'], bins=10, density = True)
pdf = counts/(sum(counts))
cdf = np.cumsum(pdf)
plt.plot(bin_edges[1:], pdf)
plt.plot(bin_edges[1:], cdf)
plt.title("pdf and cdf for nodes")
plt.ylabel("% of person's")
plt.xlabel("nodes")


counts, bin_edges = np.histogram(two["nodes"], bins=10, density = True)
pdf = counts/(sum(counts))
cdf = np.cumsum(pdf)
plt.plot(bin_edges[1:], pdf)
plt.plot(bin_edges[1:], cdf)
plt.legend(label)

plt.show();
```



observation

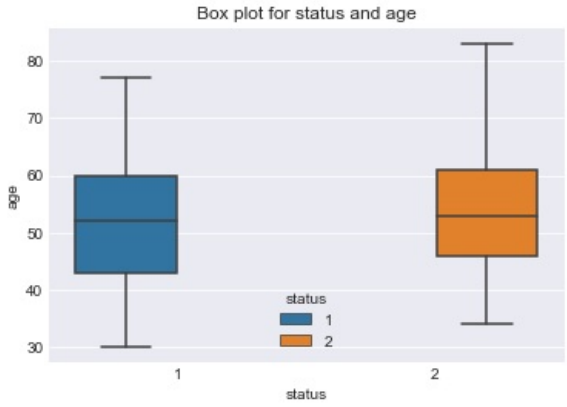
1.15% of the person's have less than or equal to age 37 who survived.

2.perosons' who has more than 46 nodes not survived

Box plot And Whiskers

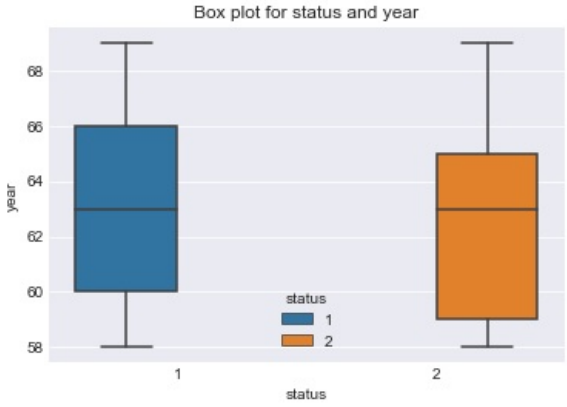
In [60]:

```
# boxplot gives you the statistical summery of data
# Rectangle(box) represent the 2nd and 3rd quartile (horizontal line either side of the rectangle)
# The horizontal line inside rectangle(box) represents median

sns.boxplot(x = "status", y = "age", hue = "status", data = haberman).set_title("Box plot for status and age")
plt.show()
```



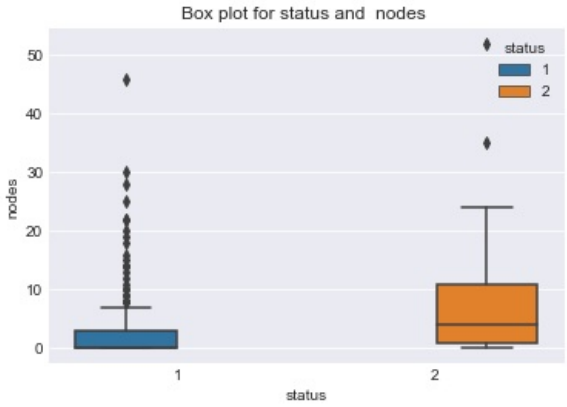
In [62]:

```
# plt.title("Box plot for status and year")
sns.boxplot(x = "status", y = "year", hue = "status", data = haberman).set_title("Box plot for status and year")
plt.show()
```



In [63]:

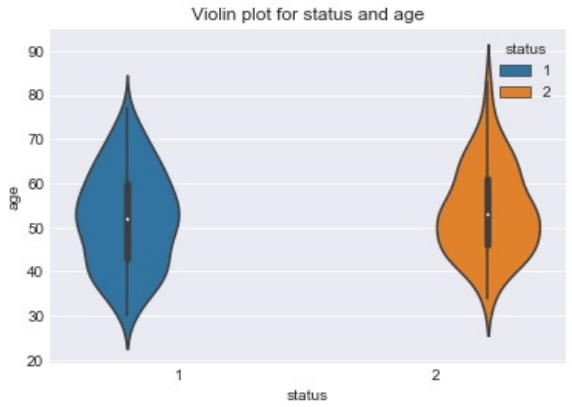
```
#plt.title("Box plot for status and  nodes")
sns.boxplot(x = "status", y = "nodes", hue="status", data = haberman).set_title("Box plot for status and  nodes "
)
plt.show()
```



Violin plots

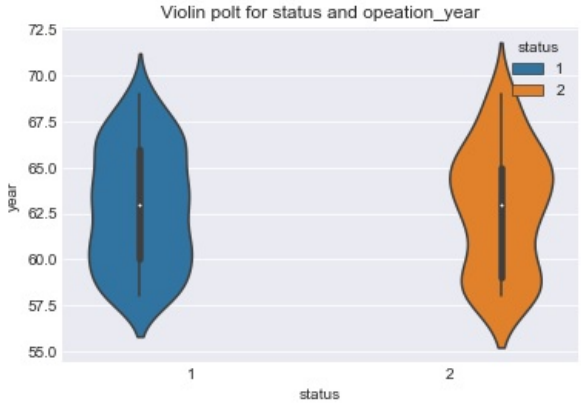
In [64]:

```
# The violin plot shows the full distribution of the data.
# It is combination of box plot and histogram
# central dot represents median
plt.title("Violin plot for status and age")
sns.violinplot(x = "status", y = "age", hue = "status", data = haberman)
plt.show()
```



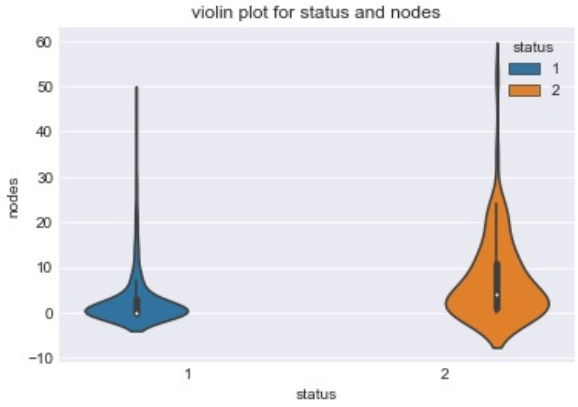
In [65]:

```
plt.title("Violin polt for status and opeation_year")
sns.violinplot(x = "status", y = "year", hue = "status", data = haberman)
plt.show()
```



In [66]:

```
plt.title("violin plot for status and nodes")
sns.violinplot(x = "status", y = "nodes", hue = "status", data = haberman)
plt.show()
```



Conclusions

<>The given dataset is imbalenced as it does not contains euqal number of data-points for each class.

<>The given dataset is not linearly seprable form each class. There are too much overlapping in the data-points and hence it is very diffucult to classify.

<>somehow nodes is giving some intution in the dataset

<>we can not build simple model using only if else condition we need to have some more complex technique to handle this dataset.

In [ ]: