(3.12) Exercise:

1. **Download Haberman Cancer Survival dataset from Kaggle. You may have to create a Kaggle account to donwload data. (https://www.kaggle.com/gilsousa/habermans-survival-data-set)**
2. **Perform a similar alanlaysis as above on this dataset with the following sections:**

- **High level statistics of the dataset: number of points, numer of features, number of classes, data-points per class.**
- **Explain our objective.**
- **Perform Univaraite analysis(PDF, CDF, Boxplot, Voilin plots) to understand which features are useful towards classification.**
- **Perform Bi-variate analysis (scatter plots, pair-plots) to see if combinations of features are useful in classfication.**
- **Write your observations in english as crisply and unambigously as possible. Always quantify your results.**

In [3]:

```
"""
Importing the file from the local system
"""
from google.colab import files
files=files.upload()
```

Choose File **No file selected**

Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving haberman.csv to haberman.csv
```

In [4]:

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

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

Out[4]:

	Age	Op_Year	axil_nodes	Surv_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 × 4 columns

In [6]:

```
"""
Prints the number of points
"""
print(len(haberman))
```

```
306
```

In [5]:

```
"""
Prints the features of the dataset
"""
print(haberman.columns)
```

```
Index(['Age', 'Op_Year', 'axil_nodes', 'Surv_status'], dtype='object')
```

In [6]:

```
haberman.describe()
```

Out[6]:

	Age	Op_Year	axil_nodes	Surv_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]:

```
"""
Prints the classes and number of datapoints per class
"""
print(haberman["Surv_status"].value_counts())
```

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

The above dataset has been taken from Chicago Hospital Billings desk and conatins the following features:

1. **Age- The age of the patient at the time of the operation**
2. **Op_Year- The year in which the patient underwent operation for breast cancer**
3. **axil_nodes- Number of axilliary nodes**
4. **Surv_status- i. 1- The patient survived for more than 5 years after the operation ii. 2- The patient died within 5 years of the operation**

Our objective in this task is to determine, given a data point with the values of Age, Op_Year and auxilliary nodes, in which class will it fall i.e., they will die after 5 years of the operation(haberman_1) or within 5 years of the operation(haberman_2)

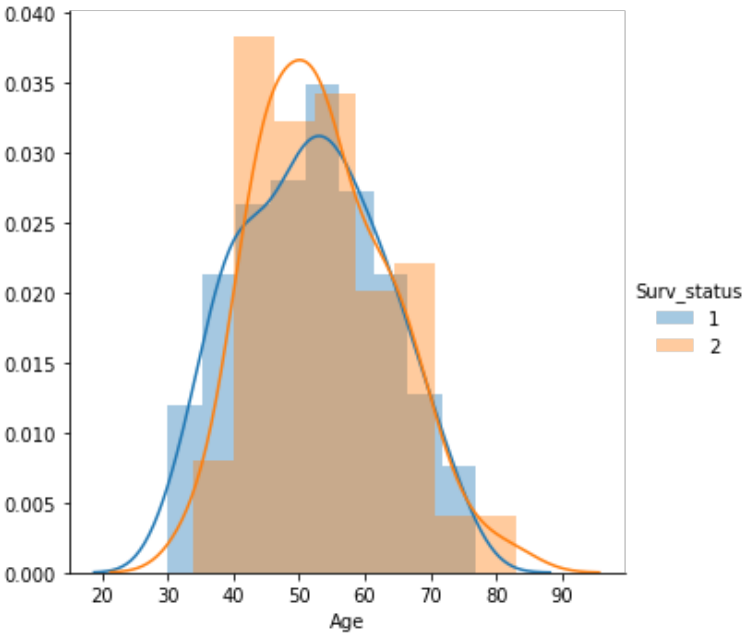
Univariate Analysis

In [8]:

```
"""
Histogram plot of Surv_status with respect to Age
```

```
"""
sns.FacetGrid(haberman, hue="Surv_status", height=5) \
   .map(sns.distplot, "Age") \
   .add_legend()
plt.show()
```

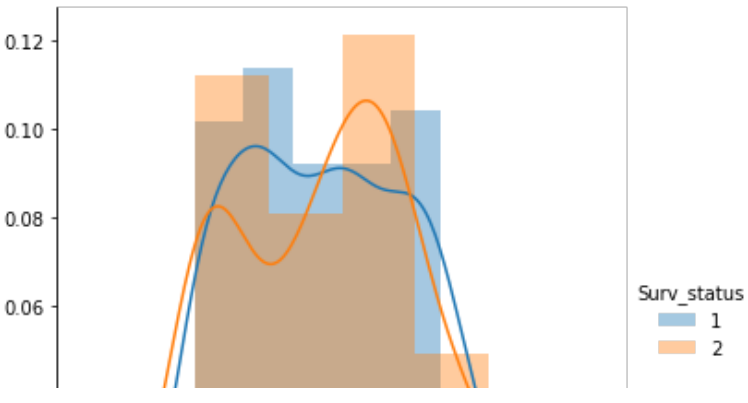
```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2557: FutureWarning: `dis
tplot` is a deprecated function and will be removed in a future version. Please adapt you
r code to use either `displot` (a figure-level function with similar flexibility) or `his
tplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2557: FutureWarning: `dis
tplot` is a deprecated function and will be removed in a future version. Please adapt you
r code to use either `displot` (a figure-level function with similar flexibility) or `his
tplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

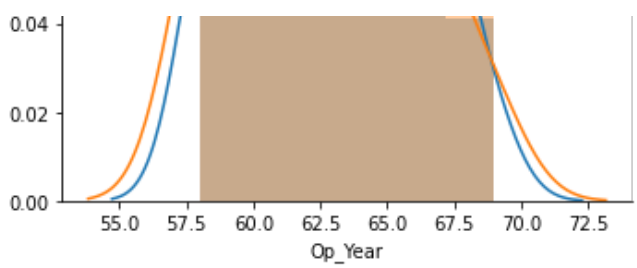


In [9]:

```
"""
Histogram plot of Surv_status with respect to Op_Year
"""
sns.FacetGrid(haberman, hue="Surv_status", height=5) \
   .map(sns.distplot, "Op_Year") \
   .add_legend()
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2557: FutureWarning: `dis
tplot` is a deprecated function and will be removed in a future version. Please adapt you
r code to use either `displot` (a figure-level function with similar flexibility) or `his
tplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2557: FutureWarning: `dis
tplot` is a deprecated function and will be removed in a future version. Please adapt you
r code to use either `displot` (a figure-level function with similar flexibility) or `his
tplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

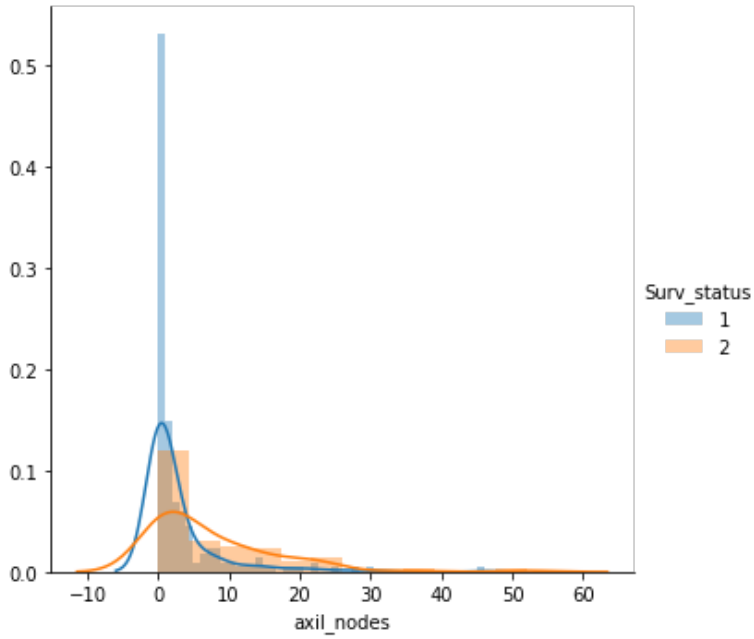




In [10]:

```
"""
Histogram plot of Surv_status with respect to axil_nodes
"""
sns.FacetGrid(haberman, hue="Surv_status", height=5) \
   .map(sns.distplot, "axil_nodes") \
   .add_legend()
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2557: FutureWarning: `dis
tplot` is a deprecated function and will be removed in a future version. Please adapt you
r code to use either `displot` (a figure-level function with similar flexibility) or `his
tplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2557: FutureWarning: `dis
tplot` is a deprecated function and will be removed in a future version. Please adapt you
r code to use either `displot` (a figure-level function with similar flexibility) or `his
tplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



The above three histogram plots are with respect to the variables age,operation year and number of auxilliary nodes respectively. Here, our primary aim is to isolate the pdf and cdf of haberman_1 and haberman_2 as much as possible with respect to any three of the above mentioned variables. Now, as we can see that in case of Age and Op_Year the histogram plots are overlapping significantly, so they do not provide reliable results, but, in the case of auxilliary nodes, the plot provides the best possible isolation because the maximum number of haberman_1 plots(more than 50%) occur in the approximate range of 0-2 but it is also not very reliable beacuse small fragments of data keep occurring in the subsequent bars.

In [12]:

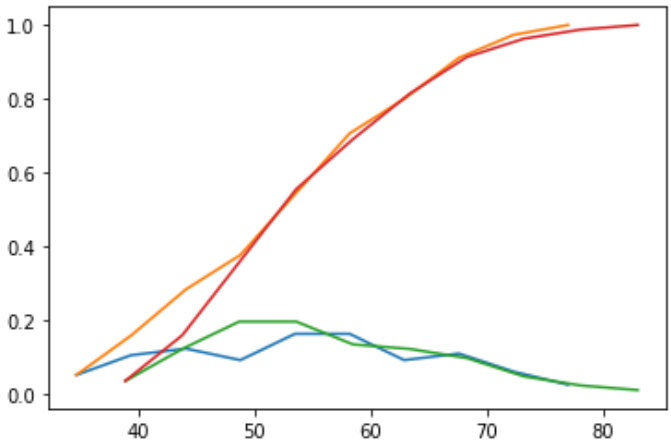
```
"""
Pdf and Cdf of haberman_1 and haberman_2 with respect to Age
"""
haberman_1=haberman.loc[haberman["Surv_status"]==1]
haberman_2=haberman.loc[haberman["Surv_status"]==2]
counts, bin_edges = np.histogram(haberman_1["Age"], bins=10,
                                 density = True)
```

```
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_2["Age"], 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)
```

```
[0.05333333 0.10666667 0.12444444 0.09333333 0.16444444 0.16444444
 0.09333333 0.11111111 0.06222222 0.02666667]
[30.  34.7 39.4 44.1 48.8 53.5 58.2 62.9 67.6 72.3 77. ]
[0.03703704 0.12345679 0.19753086 0.19753086 0.13580247 0.12345679
 0.09876543 0.04938272 0.02469136 0.01234568]
[34.  38.9 43.8 48.7 53.6 58.5 63.4 68.3 73.2 78.1 83. ]
```

Out[12]:

```
[<matplotlib.lines.Line2D at 0x7f77a7f39cd0>]
```



In [13]:

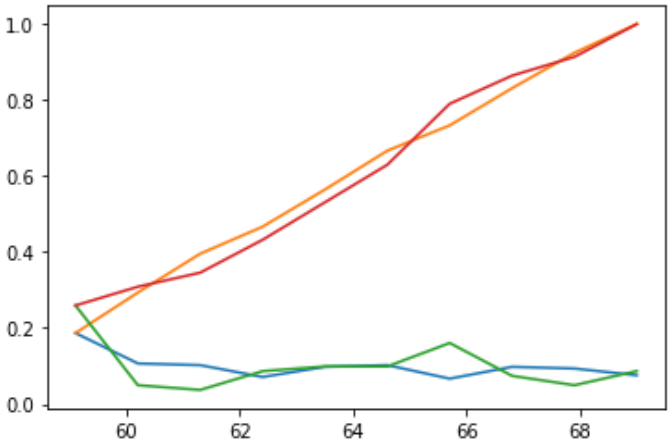
```
"""
Pdf and Cdf of haberman_1 and haberman_2 with respect to Op_Year
"""
counts, bin_edges = np.histogram(haberman_1["Op_Year"], bins=10,
                                 density = True)
pdf = counts/(sum(counts))
print(pdf);
print(bin_edges)
cdf = np.cumsum(pdf)
plt.plot(bin_edges[1:],pdf)
plt.plot(bin_edges[1:], cdf)


counts, bin_edges = np.histogram(haberman_2["Op_Year"], 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)
```

```
[0.18666667 0.10666667 0.10222222 0.07111111 0.09777778 0.10222222
 0.06666667 0.09777778 0.09333333 0.07555556]
[58.  59.1 60.2 61.3 62.4 63.5 64.6 65.7 66.8 67.9 69. ]
[0.25925926 0.04938272 0.03703704 0.08641975 0.09876543 0.09876543
 0.16049383 0.07407407 0.04938272 0.08641975]
```

```
[58.  59.1 60.2 61.3 62.4 63.5 64.6 65.7 66.8 67.9 69. ]
```

Out[13]:

```
[<matplotlib.lines.Line2D at 0x7f77a7eaee10>]
```



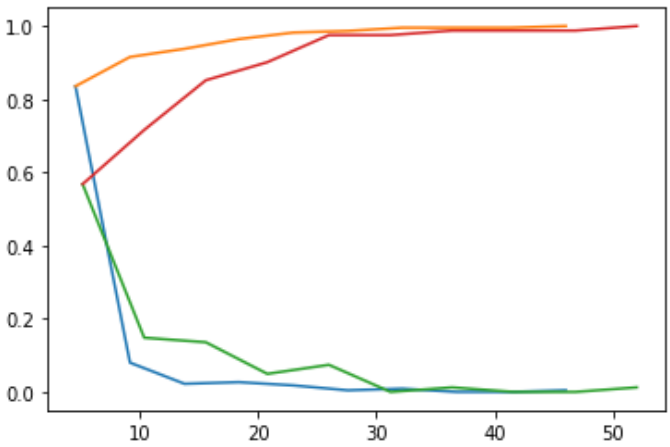
In [14]:

```
"""
Pdf and Cdf of haberman_1 and haberman_2 with respect to auxil_nodes
"""
counts, bin_edges = np.histogram(haberman_1["axil_nodes"], 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_2["axil_nodes"], 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)
```

```
[0.83555556 0.08       0.02222222 0.02666667 0.01777778 0.00444444
 0.00888889 0.         0.         0.00444444]
[ 0.   4.6  9.2 13.8 18.4 23.  27.6 32.2 36.8 41.4 46. ]
[0.56790123 0.14814815 0.13580247 0.04938272 0.07407407 0.
 0.01234568 0.         0.         0.01234568]
[ 0.   5.2 10.4 15.6 20.8 26.  31.2 36.4 41.6 46.8 52. ]
```

Out[14]:

```
[<matplotlib.lines.Line2D at 0x7f77a7e22310>]
```



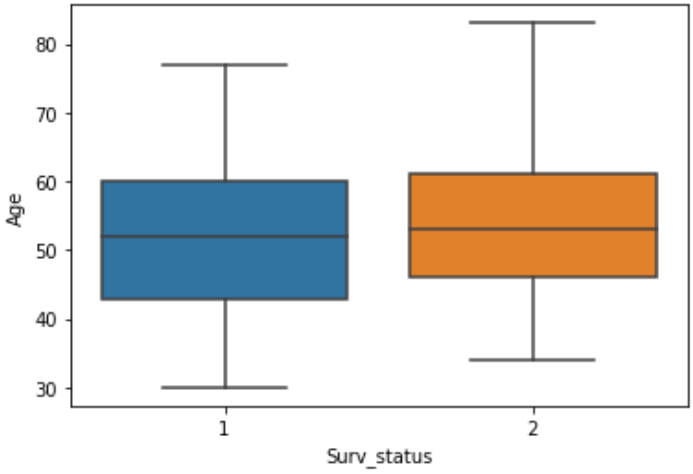
In the above Cdf graph for auxilliary nodes approximately 92% haberman_1 data comes in the 0-10 range

In the above CDF graph for auxiliary nodes approximately 92% haberman_1 data comes in the 0-10 range whereas while reaching 20 it touches around 98% of its entire dataset while in case of haberman_2 the 0-10 range accounts for approximately 75% of the data points.

Box Plot

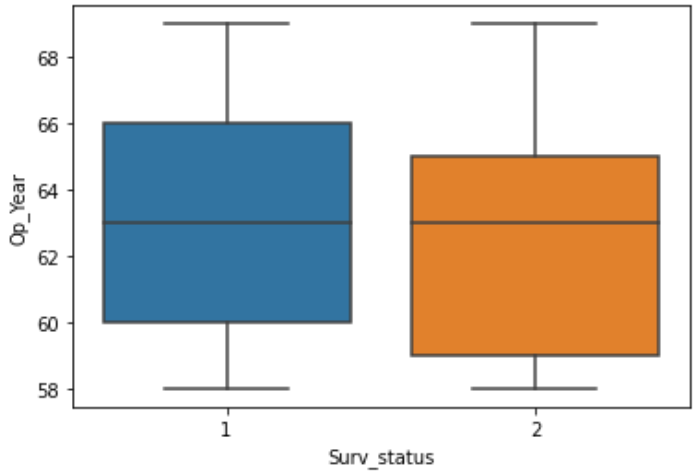
In [15]:

```
sns.boxplot(x="Surv_status",y="Age", data=haberman)
plt.show()
```



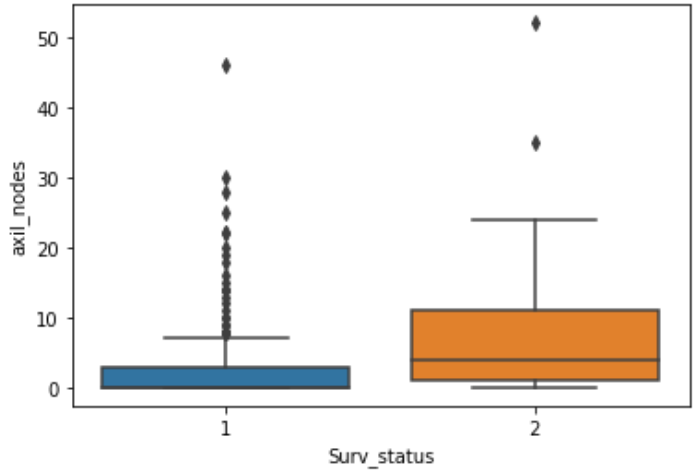
In [16]:

```
sns.boxplot(x="Surv_status",y="Op_Year", data=haberman)
plt.show()
```



In [17]:

```
sns.boxplot(x="Surv_status",y="axil_nodes", data=haberman)
plt.show()
```

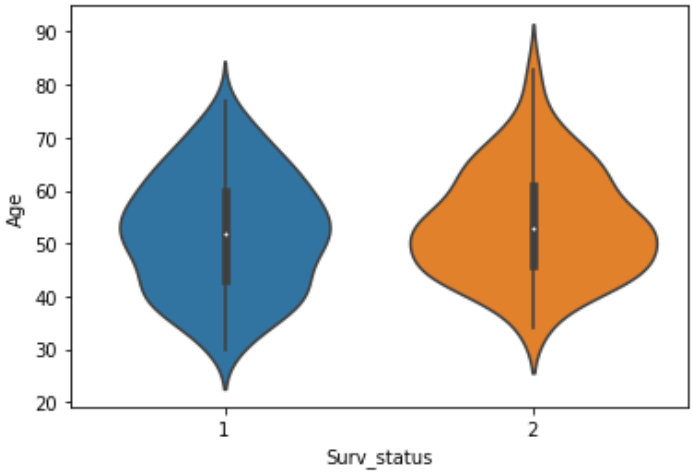


In case of box-plot of auxilliary_nodes, the maximum data-points of haberman_1(75%) is located near 0 value whereas in case of haberman_2 it goes as high as 12 to reach it's share of 75% of the entire dataset.

Violin Plot

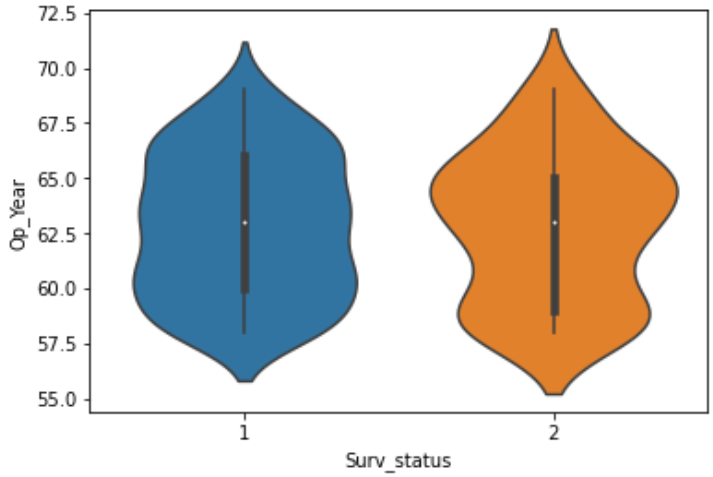
In [18]:

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



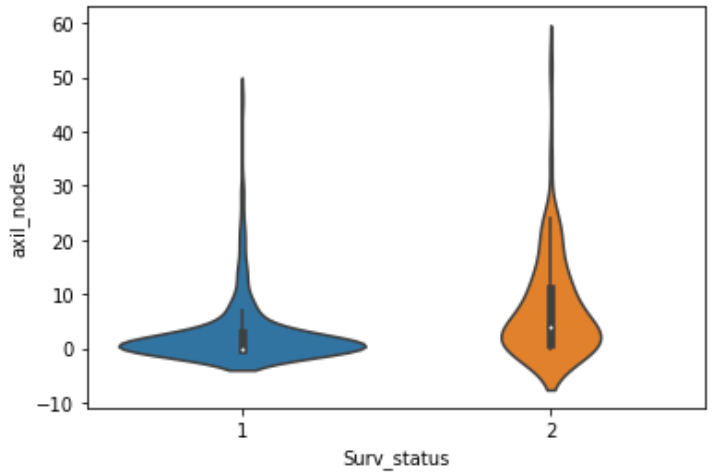
In [19]:

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



In [20]:

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



Bi-variate analysis

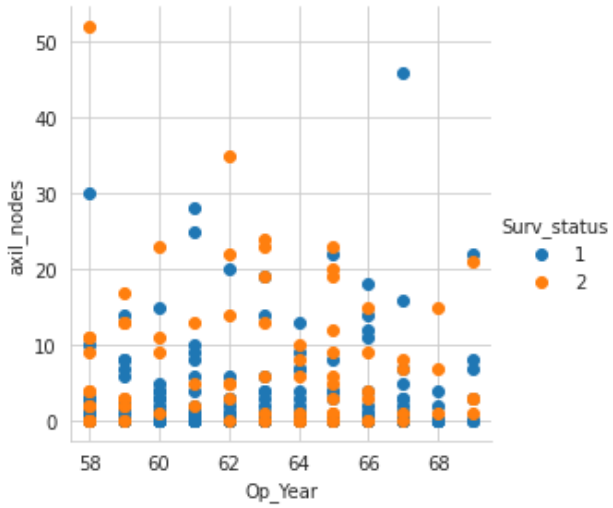
Bi-variate analysis

In [21]:

```
"""
2-D scatter plot for Op_Year vs Age
"""
sns.set_style("whitegrid")
sns.FacetGrid(haberman,hue="Surv_status",height=4)\
    .map(plt.scatter,"Age","Op_Year")\
    .add_legend()
plt.show()
```

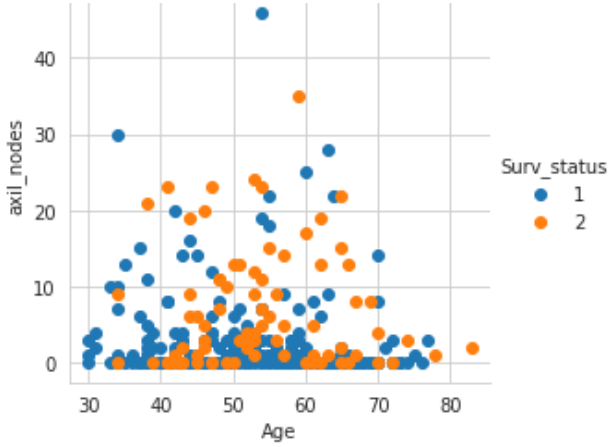
In [22]:

```
"""
2-D scatter plot for axil_nodes vs Op_Year
"""
sns.set_style("whitegrid")
sns.FacetGrid(haberman,hue="Surv_status",height=4)\
    .map(plt.scatter,"Op_Year","axil_nodes")\
    .add_legend()
plt.show()
```



In [23]:

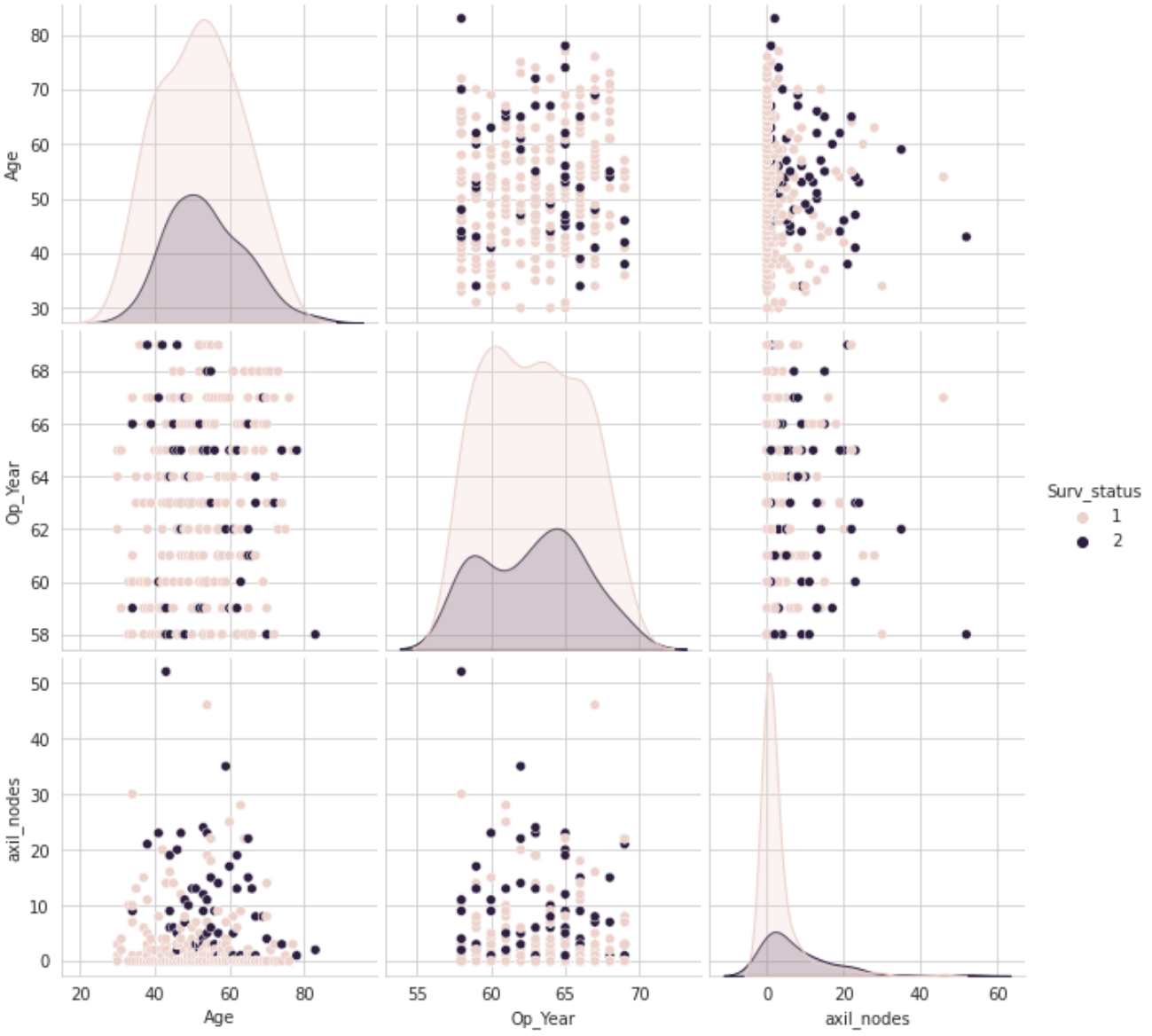
```
"""
2-D scatter plot for axil_nodes vs Age
"""
sns.set_style("whitegrid")
sns.FacetGrid(haberman,hue="Surv_status",height=4)\
    .map(plt.scatter,"Age","axil_nodes")\
    .add_legend()
plt.show()
```



Pair-Plot

In [24]:

```
plt.close()
sns.set_style("whitegrid")
sns.pairplot(haberman,hue="Surv_status",height=3)
plt.show()
```



Findings: In the graph of Age vs auxil_nodes, we can say that approximately 90% of haberman_1 data points are occurring 0-2 value range, so, we can vaguely say that, if 0=<axil_nodes<=2 Surv_status=1 else Surv_status=2 with reference to Age vs axil_nodes graph

But, nonetheless this equation will be prone to errors.

As a whole, we can say that the auxilliary nodes value for haberman_2 is more scattered as compared to that of

As a whole, we can say that the auxiliary nodes value for haberman_2 is more scattered as compared to that of haberman_1.

In [ ]: