Haberman's Survival Data

Dataset:https://www.kaggle.com/gilsousa/habermans-survival-data-set/version/1

- A Dataset taken to perform Assignment
- Survival Status of patients who had undergone surgery for breast cancer.
- On 1999 by Tjen-Sien Lim
- To understand Axillary Nodes please go through the website : [https://jamanetwork.com/journals/jama/fullarticle/1750133]
- Objective:To explore the Data and perform various Plots based on Survival Status

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

#downlaod haberman.csv from https://www.kaggle.com/gilsousa/habermans-s
urvival-data-set/version/1
#Loading haberman.csv into a pandas dataFrame.
Haberman = pd.read_csv("haberman.csv")
```

In [2]:
```
import warnings
warnings.filterwarnings("ignore")
```

In [3]:
```
#count of (Rows and Columns) or(Data-points & Features)
print(Haberman.shape)
```

```
(306, 4)
```

Observation-1:

Dataset contains 306 Data points and 4 features.

In [4]:
```
#Printing first 5 Rows and the column names
print (Haberman.columns)
Haberman.head()
```

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

Out[4]:

	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

Observation-2: The column names are not defined in a good user understanding way.

In [5]:
```
#Renaming the Column names:
Haberman.columns = ['Age','Op_year','Axil_nodes','Survival_status']
#Printing first 5 Rows and the column names
print (Haberman.columns)
Haberman.head()
```

```
Index(['Age', 'Op_year', 'Axil_nodes', 'Survival_status'], dtype='objec
t')
```

Out[5]:

	Age	Op_year	Axil_nodes	Survival_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

Observation-3:

Age: Age of patient at time of operation (numerical)

Op_year: Patient's year of operation (year - 1900, numerical)

Axil_nodes: Number of positive axillary nodes detected (numerical)

Survival_status =1 If Patient survived 5 years or longer. Survival_status =2 If Patient died within 5 year

In [6]:
```
#Survival count

Haberman["Survival_status"].value_counts()
```

Out[6]:
```
1    225
2     81
Name: Survival_status, dtype: int64
```

Observation-4:It is a Unbalanced datset with

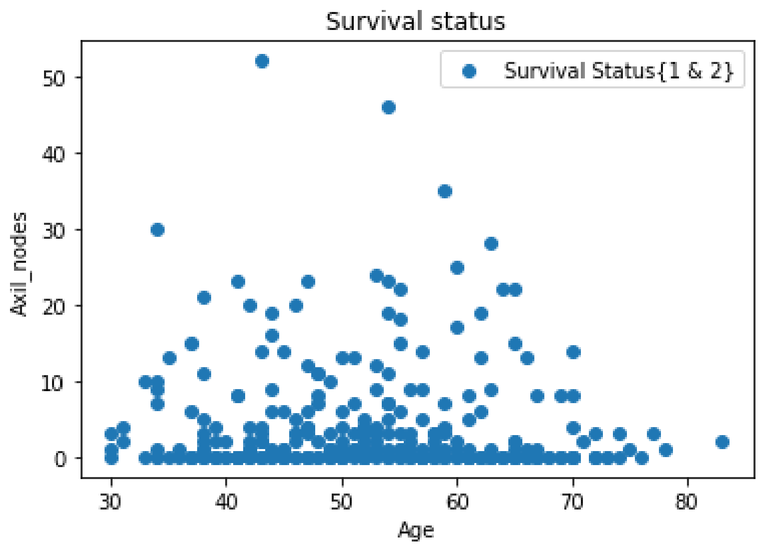
224 people survived more than 5 years

81 people couldn't survive more than 5 years.

2-D Plot Scatter Plot

In [7]:
```
plt.title("Survival status")
plt.scatter(Haberman.Age, Haberman.Axil_nodes, label='Survival Status{1
 & 2}')
plt.legend()
plt.xlabel("Age")
plt.ylabel("Axil_nodes")
```

```
#Haberman.plot(kind='scatter', x='Age', y='Axil_nodes');
plt.show()

#Plotting the Survival status with respective to A.nodes count
```

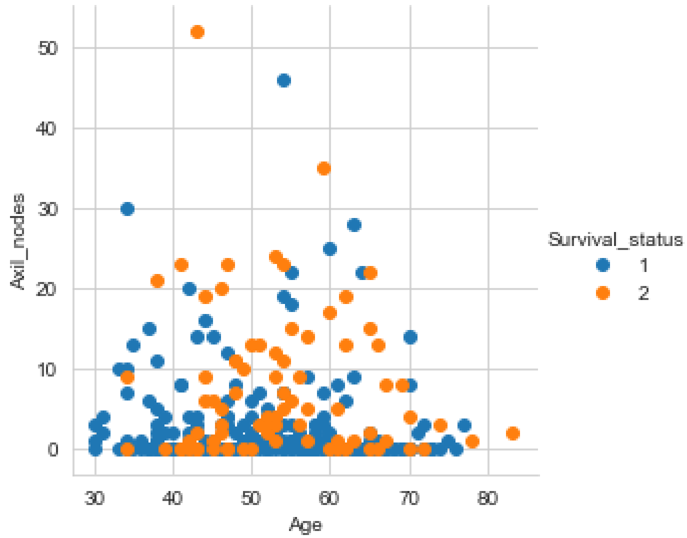


Observation-5:

Most of the people have Zero nodes or less than 5 nodes

2-D Plot Scatter Plot with Colour-coding

In [8]:
```
sns.set_style("whitegrid");
sns.FacetGrid(Haberman, hue="Survival_status", size=4) \
   .map(plt.scatter,"Age", "Axil_nodes") \
   .add_legend();
plt.show();
```



Observation:

People with zero node cannot be distinguished w.r.t survival_status

Pair-Plot

In [9]:
```
plt.close();
sns.set_style("whitegrid");
sns.pairplot(Haberman, hue='Survival_status' ,vars = ['Age', 'Op_year',
 'Axil_nodes'], height = 3)
plt.show()
```

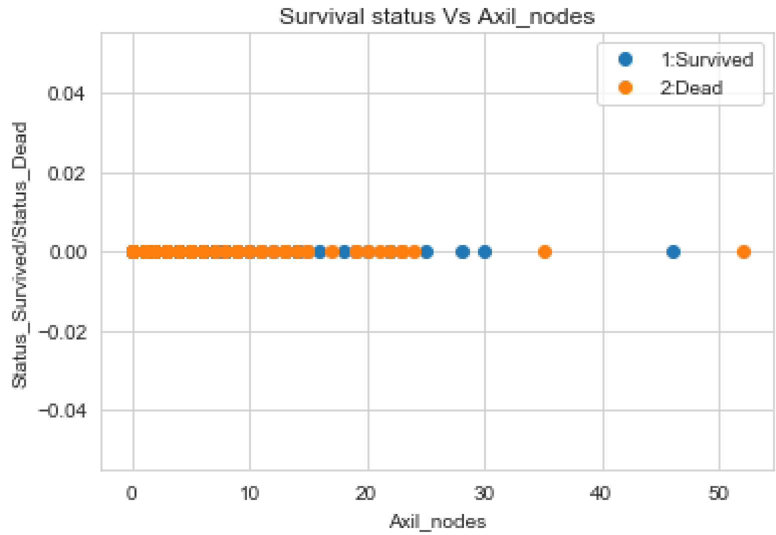
Histogram, PDF, CDF

Histogram:

- A histogram is a plot that lets you discover, and show, the underlying frequency distribution of a set of continuous data.
- To construct a histogram from a continuous variable you first need to split the data into intervals, called bins
- The width of each bin is calculated as = ((max - min) / bins)
- There is no right or wrong answer as to how wide a bin should be, but there are rules of thumb. You need to make sure that the bins are not too small or too large
- To find the counts we calcuate how many number of points fall into each bin
- List item

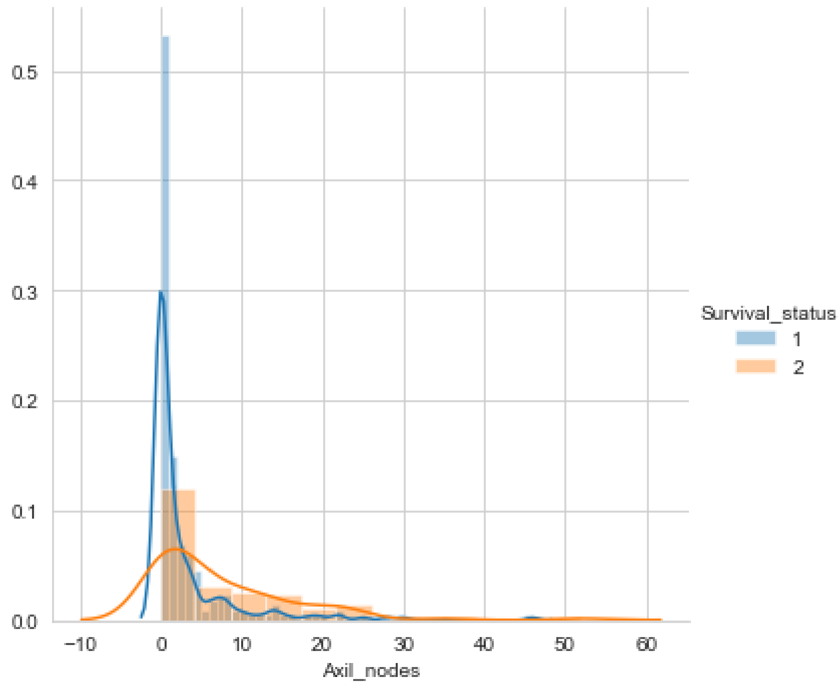
In [10]:

```
#plotting 1D plot
#Survival_status vs nodes
Status_Survive = Haberman.loc[Haberman["Survival_status"] == 1];
Status_Dead = Haberman.loc[Haberman["Survival_status"] == 2];
plt.plot(Status_Survive["Axil_nodes"],np.zeros_like(Status_Survive["Axi
l_nodes"]), 'o',label='1:Survived')
plt.plot(Status_Dead["Axil_nodes"],np.zeros_like(Status_Dead["Axil_node
s"]), 'o',label='2:Dead')
plt.xlabel('Axil_nodes')
plt.ylabel('Status_Survived/Status_Dead')
plt.title("Survival status Vs Axil_nodes")
plt.legend()
plt.show()
```



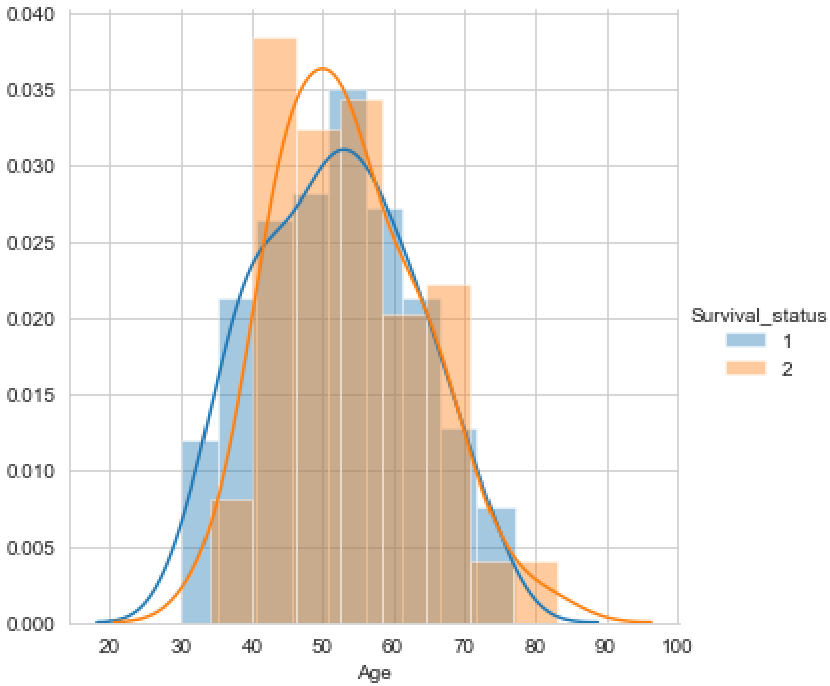
In [11]:

```
#Plotting Survival_status ns Axil_nodes
#Histogram with default bin size value using sns.distplot

sns.FacetGrid(Haberman, hue="Survival_status", height=5) \
   .map(sns.distplot, "Axil_nodes") \
   .add_legend();
plt.show();
```

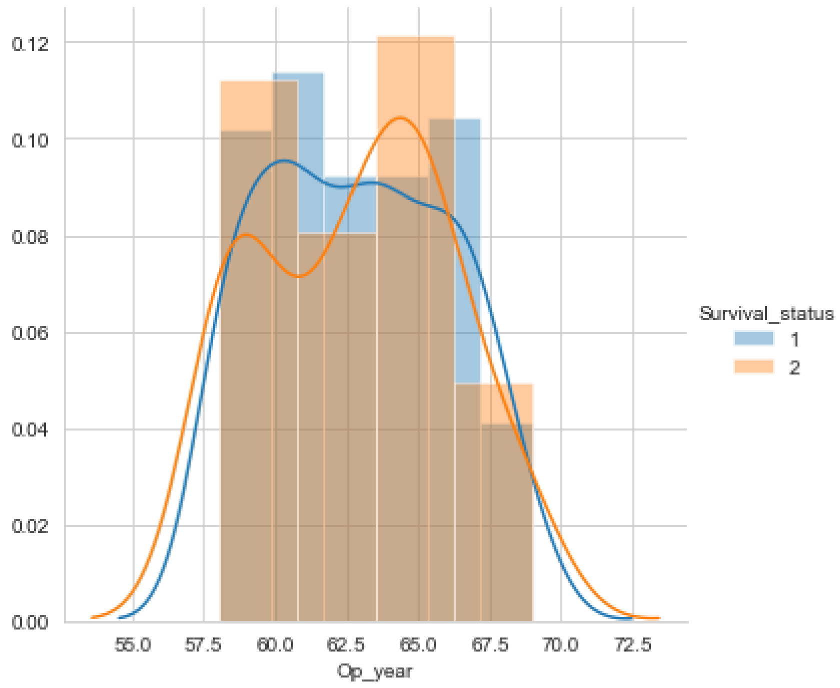


In [ ]:

In [12]:
```
#Plotting Survival_status vs age
#Histogram with default bin size value using sns.distplot
sns.FacetGrid(Haberman, hue="Survival_status", height=5) \
   .map(sns.distplot, "Age") \
   .add_legend();
plt.show();
```



In [13]:
```
#Survival_status vs year
#Histogram with default bin size value using sns.distplot
sns.FacetGrid(Haberman, hue="Survival_status", height=5) \
   .map(sns.distplot, "Op_year") \
   .add_legend();
plt.show();
```



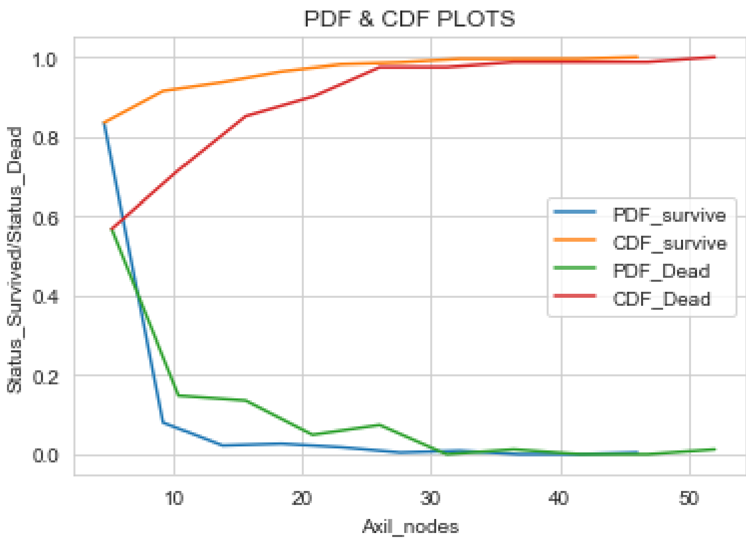
PDF:

In [14]:
```
#Computing Histogram values with bin size value =10
#computing and plotting PDF
#computing and plotting CDF

#Status_Survive vs Axil_nodes
counts, bin_edges = np.histogram(Status_Survive['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,label='PDF_survive')
plt.plot(bin_edges[1:], cdf,label='CDF_survive')
plt.xlabel('Axil_nodes')
```

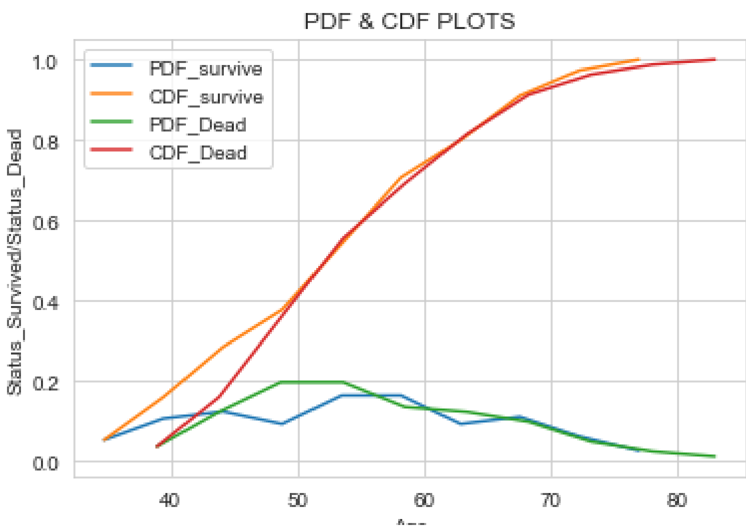
```
plt.ylabel('Status_Survived/Status_Dead')

#Status_Dead vs Axil_nodes
counts, bin_edges = np.histogram(Status_Dead['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,label='PDF_Dead')
plt.plot(bin_edges[1:], cdf,label='CDF_Dead')
plt.legend()
plt.title('PDF & CDF PLOTS')
plt.show();

counts, bin_edges = np.histogram(Status_Survive['Age'], bins=10,
                                 density = True)
pdf = counts/(sum(counts))
print(pdf);
print(bin_edges)
cdf = np.cumsum(pdf)
plt.plot(bin_edges[1:],pdf,label='PDF_survive')
plt.plot(bin_edges[1:], cdf,label='CDF_survive')
plt.xlabel('Age')
plt.ylabel('Status_Survived/Status_Dead')

#Status_Dead vs Axil_nodes
counts, bin_edges = np.histogram(Status_Dead['Age'], bins=10,
                                 density = True)
pdf = counts/(sum(counts))
print(pdf);
print(bin_edges)
cdf = np.cumsum(pdf)
plt.plot(bin_edges[1:],pdf,label='PDF_Dead')
plt.plot(bin_edges[1:], cdf,label='CDF_Dead')
plt.legend()
plt.title('PDF & CDF PLOTS')
plt.show();
```

```python
counts, bin_edges = np.histogram(Status_Survive['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,label='PDF_survive')
plt.plot(bin_edges[1:], cdf,label='CDF_survive')
plt.xlabel('Op_Year')
plt.ylabel('Status_Survived/Status_Dead')

#Status_Dead vs Axil_nodes
counts, bin_edges = np.histogram(Status_Dead['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,label='PDF_Dead')
plt.plot(bin_edges[1:], cdf,label='CDF_Dead')
plt.legend()
plt.title('PDF & CDF PLOTS')
plt.show();
```

```
[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. ]
```

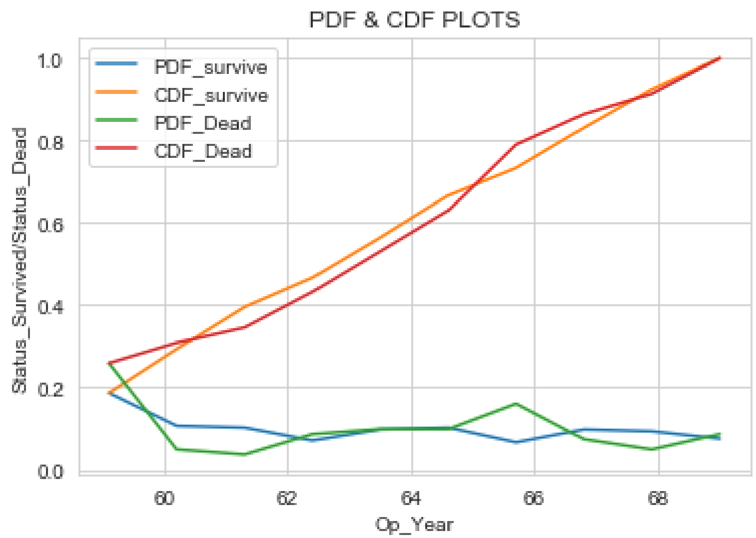



```
[0.05333333 0.10666667 0.12444444 0.09333333 0.16444444 0.16444444
 0.09333333 0.11111111 0.06222222 0.02666667]
[30.  34.7 39.4 44.1 48.8 53.5 58.2 62.9 67.6 72.3 77. ]
[0.03703704 0.12345679 0.19753086 0.19753086 0.13580247 0.12345679
 0.09876543 0.04938272 0.02469136 0.01234568]
[34.  38.9 43.8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



Age

```
[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. ]
```



Mean, Variance and Std-dev

In [15]:
```
print("Means:")
print(np.mean(Status_Survive['Axil_nodes']))
#Mean with an outlier.
print(np.mean(np.append(Status_Survive['Axil_nodes'],50)));
print(np.mean(Status_Dead['Axil_nodes']))

print("\nStd-dev:");
print(np.std(Status_Survive['Axil_nodes']))
print(np.std(Status_Dead['Axil_nodes']))
```

```
Means:
2.7911111111111113
3.0
7.45679012345679

Std-dev:
5.857258449412131
9.128776076761632
```

Median, Percentile, Quantile, IQR, MAD

In [16]:
```
#Median, Quantiles, Percentiles, IQR.
print("\nMedians:")
print(np.median(Status_Survive['Axil_nodes']))
#Median with an outlier
print(np.median(np.append(Status_Survive['Axil_nodes'],50)));
print(np.median(Status_Dead['Axil_nodes']))


print("\nQuantiles:")
print(np.percentile(Status_Survive['Axil_nodes'],np.arange(0, 100, 25
)))
```

```
print(np.percentile(Status_Dead['Axil_nodes'],np.arange(0, 100, 25)))

print("\n90th Percentiles:")
print(np.percentile(Status_Survive['Axil_nodes'],90))
print(np.percentile(Status_Dead['Axil_nodes'],90))

from statsmodels import robust
print ("\nMedian Absolute Deviation")
print(robust.mad(Status_Survive['Axil_nodes']))
print(robust.mad(Status_Dead['Axil_nodes']))
```

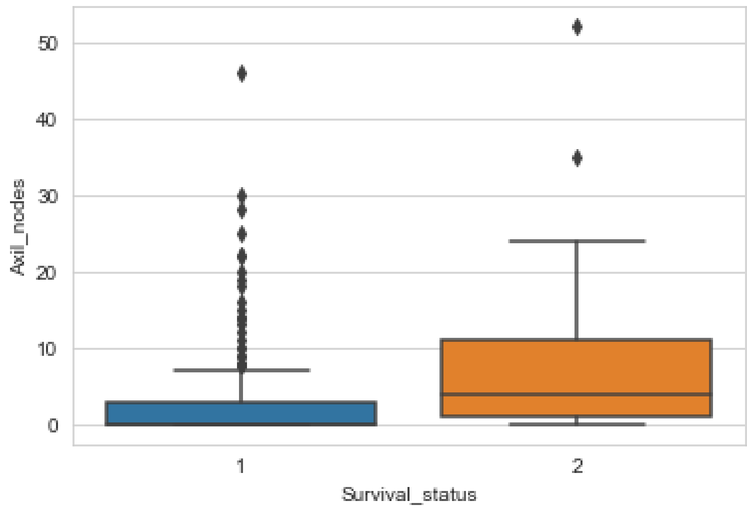
```
Medians:
0.0
0.0
4.0

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

90th Percentiles:
8.0
20.0

Median Absolute Deviation
0.0
5.930408874022408
```

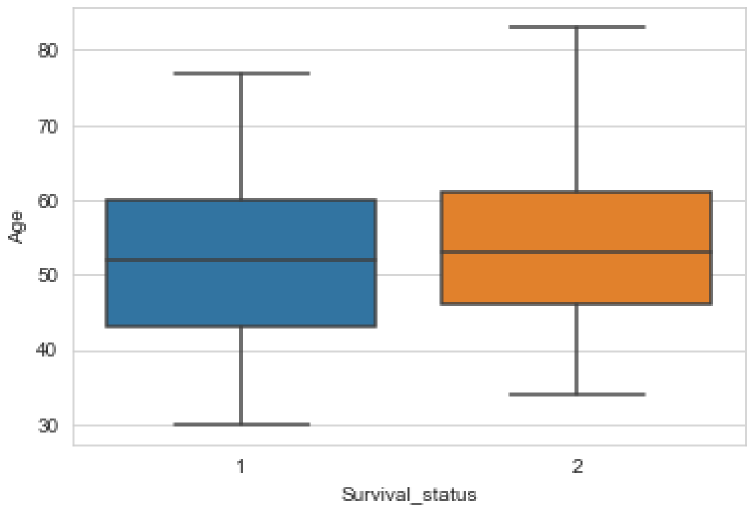
Box plot and Whiskers

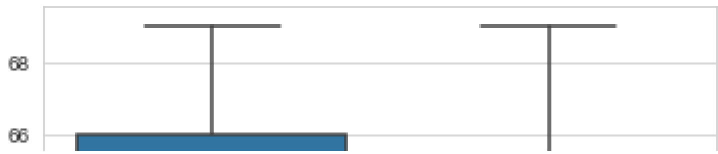
In [17]:

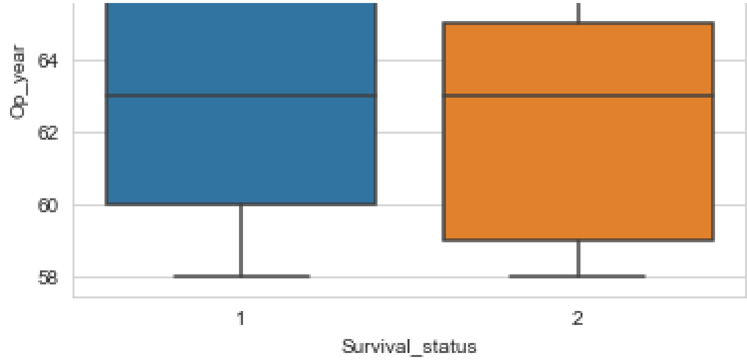
```
sns.boxplot(x='Survival_status',y='Axil_nodes', data=Haberman)
plt.show()

sns.boxplot(x='Survival_status',y='Age', data=Haberman)
plt.show()
```

```
sns.boxplot(x='Survival_status',y='Op_year', data=Haberman)
plt.show()
```



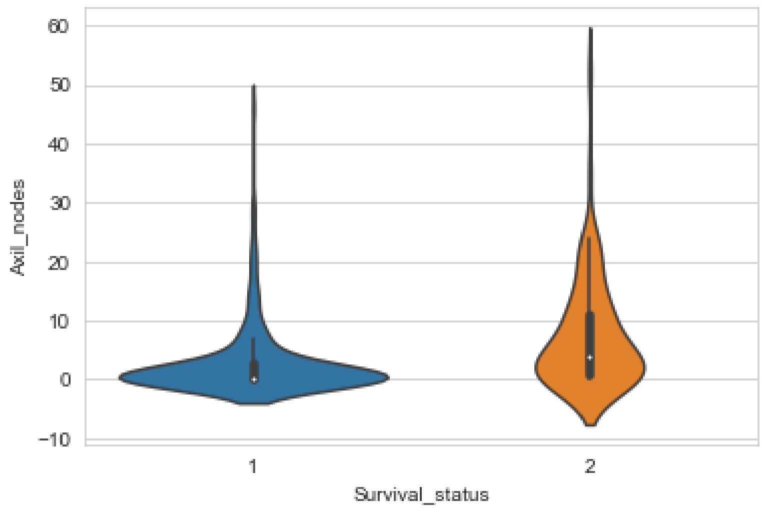


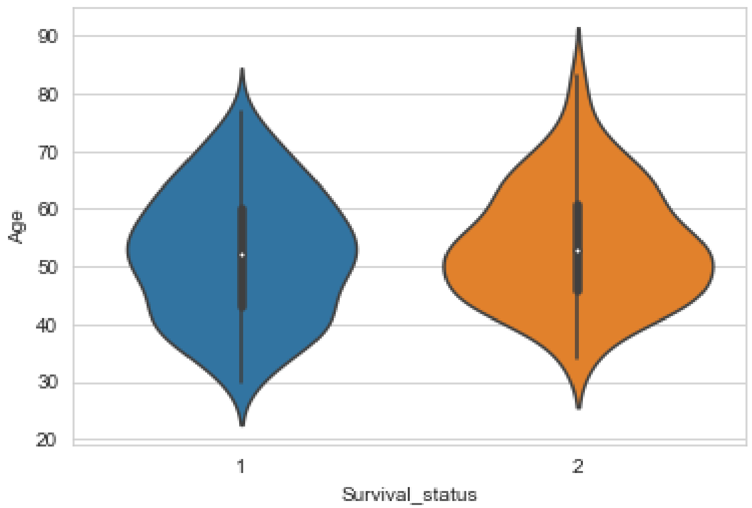


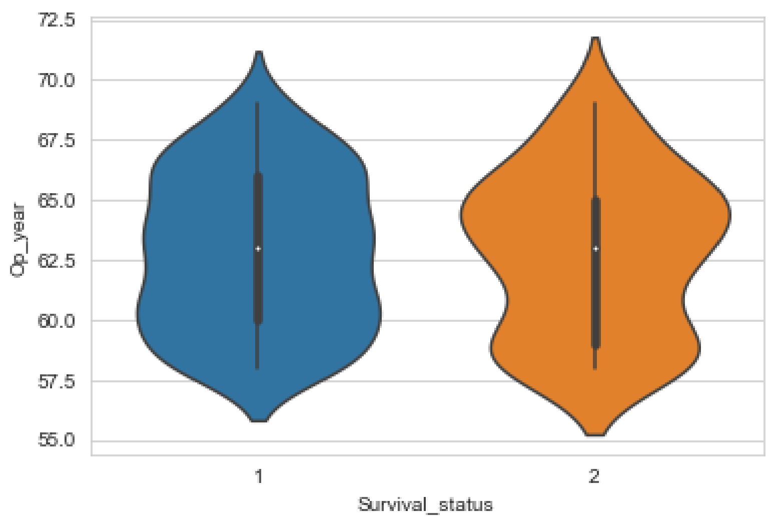


Violin plots

In [18]:
```
sns.violinplot(x='Survival_status',y='Axil_nodes', data=Haberman, size=
8)
plt.show()

sns.violinplot(x='Survival_status',y='Age', data=Haberman, size=8)
plt.show()

sns.violinplot(x='Survival_status',y='Op_year', data=Haberman, size=8)
plt.show()
```


60
50
40
30
20
10
0
−10
Axil_nodes
1
2
Survival_status

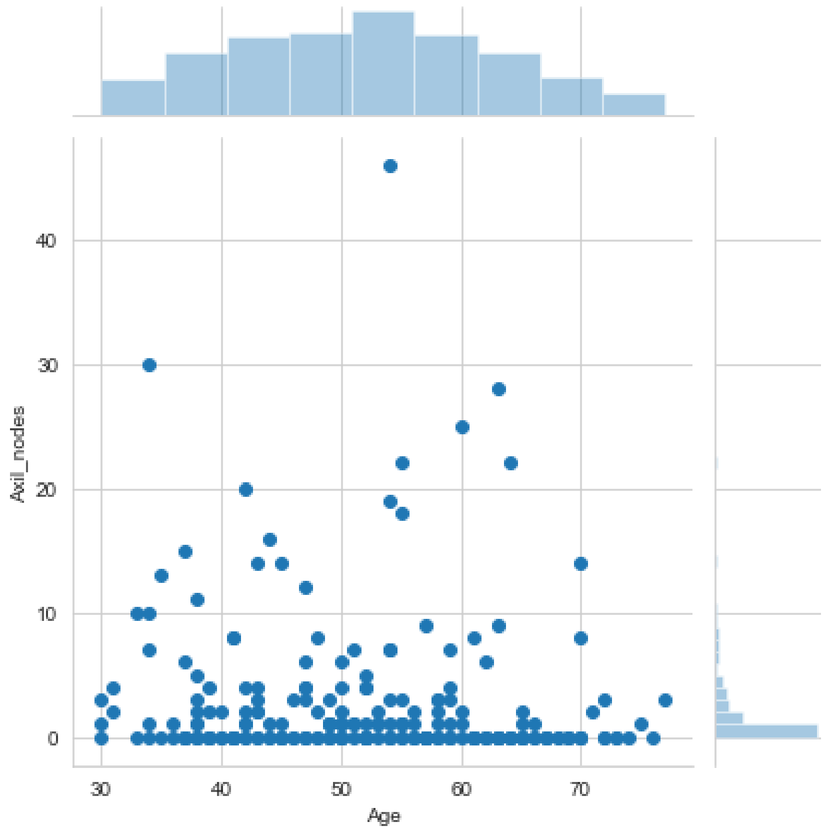

90
80
70
60
50
40
30
20
Age
1
2
Survival_status

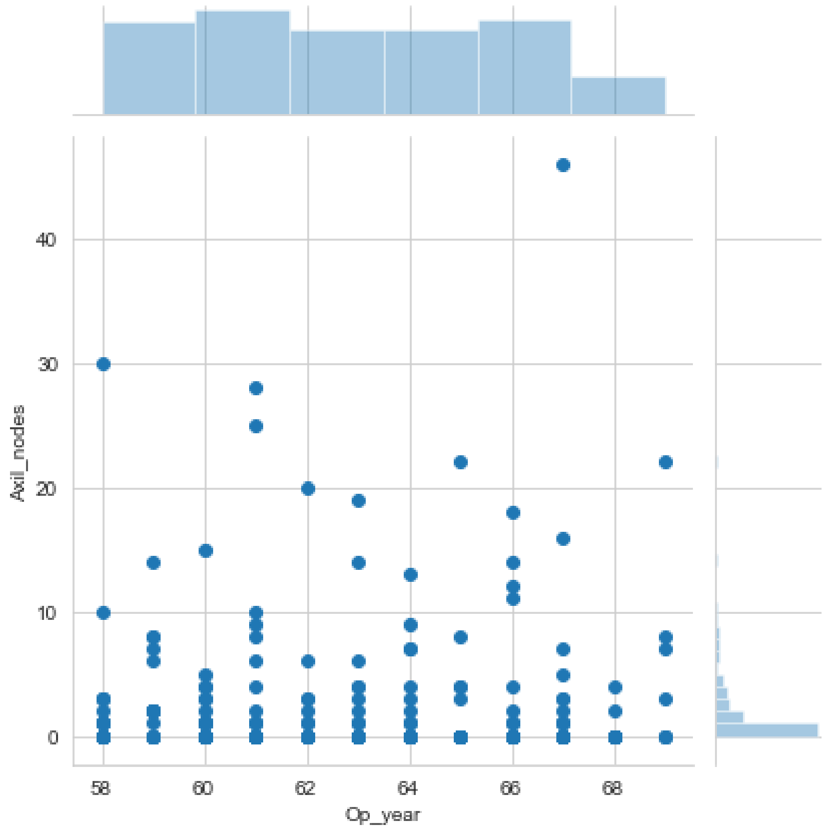


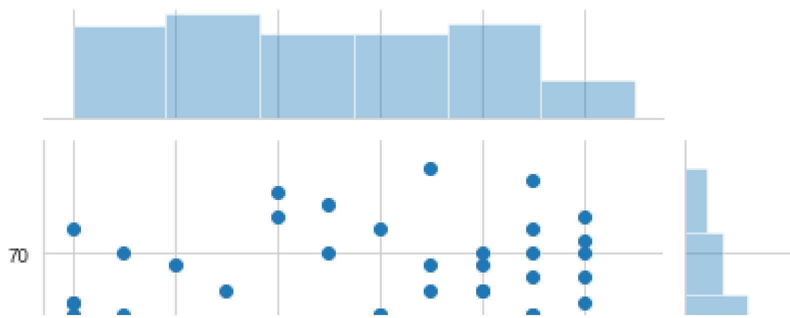
Multivariate probability density, contour plot

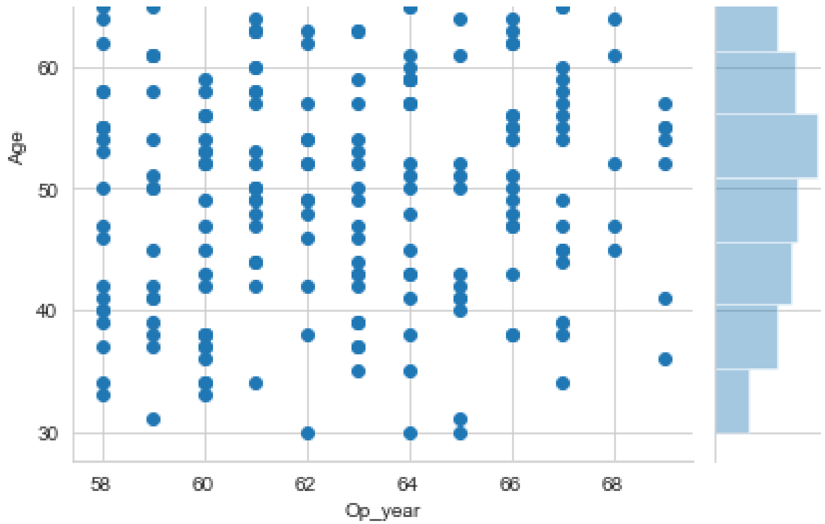
In [23]:
```
sns.jointplot(x='Age',y='Axil_nodes', data=Status_Survive);
plt.show();

sns.jointplot(x='Op_year',y='Axil_nodes', data=Status_Survive);
plt.show();

sns.jointplot(x='Op_year',y='Age', data=Status_Survive);
plt.show();


sns.jointplot(x='Age',y='Axil_nodes', data=Status_Dead);
plt.show();

sns.jointplot(x='Op_year',y='Axil_nodes', data=Status_Dead);
```

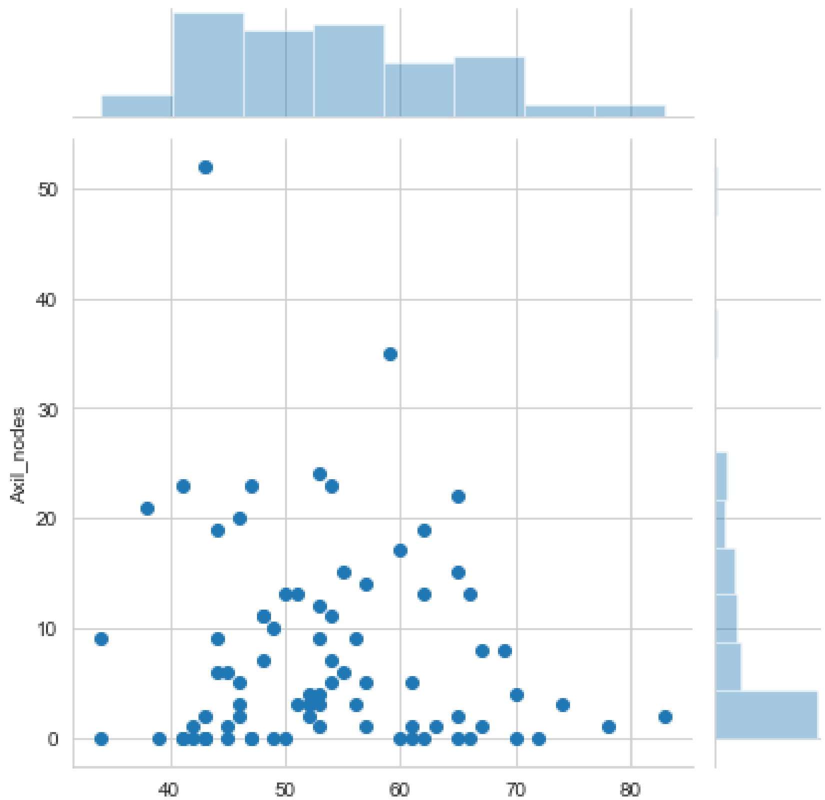
```
plt.show();

sns.jointplot(x='Op_year',y='Age', data=Status_Dead);
plt.show();
```



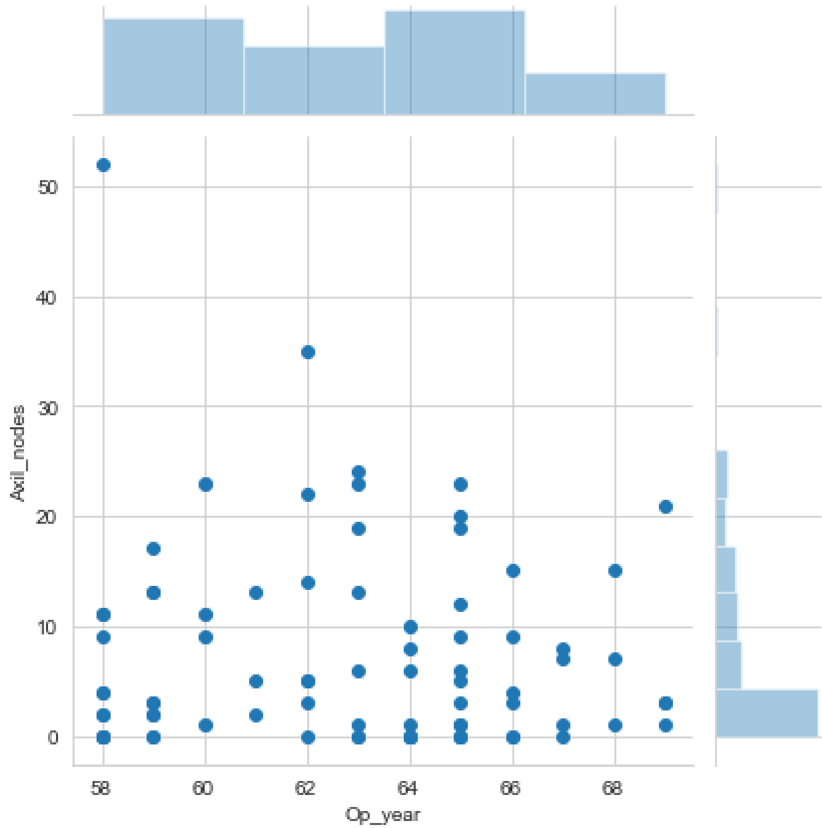

40
30
20
10
0
Axil_nodes
58
60
62
64
66
68
Op_year

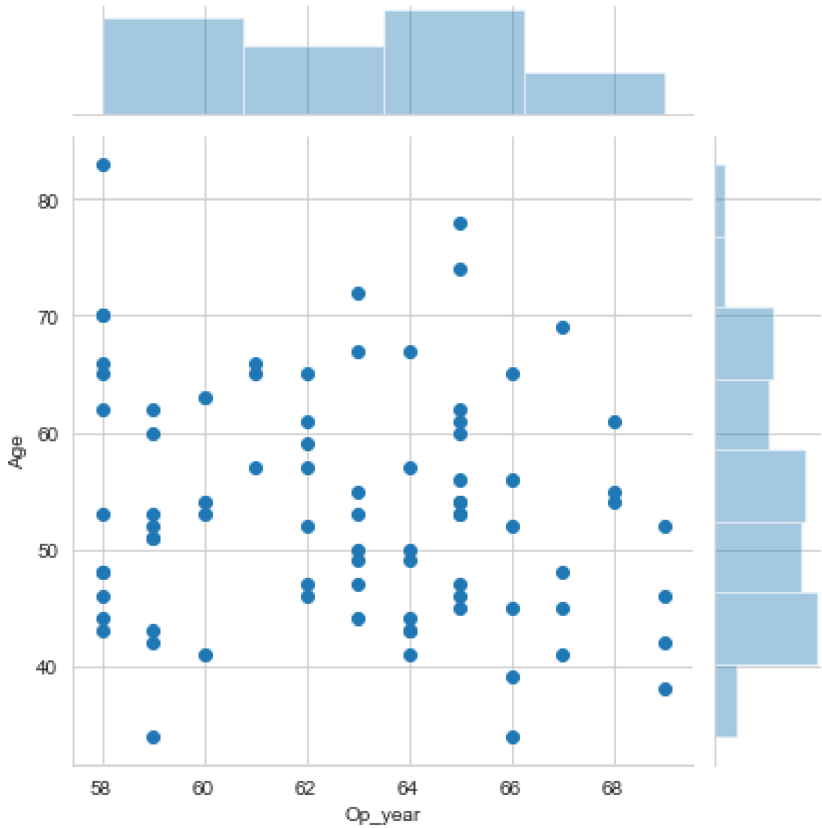

70


Age
60
50
40
30
58
60
62
64
66
68
Op_year


Axil_nodes
50
40
30
20
10
0
40
50
60
70
80

Age





In [ ]: