hb

November 21, 2018

1 Data Analysis of Haberman

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

```
In [2]: hb=pd.read_csv('haberman.csv')
        hb.head()
```

```
Out[2]:    30  64   1  1.1
        0  30  62   3    1
        1  30  65   0    1
        2  31  59   2    1
        3  31  65   4    1
        4  33  58  10    1
```

No of data points and features:-

```
In [3]: print(hb.shape)
```

```
(305, 4)
```

This Data has 4 feature
1)Age
2)Operation Year
3)Axil Nodes
4)Survival Status
From above 3 data we have to predict Survival status of patient
Setting columns name Age,oper_year(operation year),axil_nodes,surv_status

```
In [4]: hb.columns=['Age','oper_year','axil_nodes','surv_status']
        hb.head()
```

```
Out[4]:    Age  oper_year  axil_nodes  surv_status
        0   30         62           3            1
        1   30         65           0            1
        2   31         59           2            1
        3   31         65           4            1
        4   33         58          10            1
```

if Survival status (class attribute) 1 = the patient survived 5 years or longer else the patient died within 5 year

```
In [5]: hb['surv_status'].value_counts()
```

```
Out[5]: 1    224
        2     81
        Name: surv_status, dtype: int64
```

2 Objective:-

Our Objective is to perform different operations on haberman dataset from which we could easily differentiate whether it belongs to survival status of type 1 or type 2

First we create two DataFrame h1 and h2 for survival status 1 and 2 respectively.

```
In [6]: h1=hb.loc[hb['surv_status']==1]
        h1.head()
        #first dataframe
```

Out[6]:

	Age	oper_year	axil_nodes	surv_status
0	30	62	3	1
1	30	65	0	1
2	31	59	2	1
3	31	65	4	1
4	33	58	10	1

```
In [7]: h2=hb.loc[hb['surv_status']==2]
        h2.head()
        #second Dataframe
```

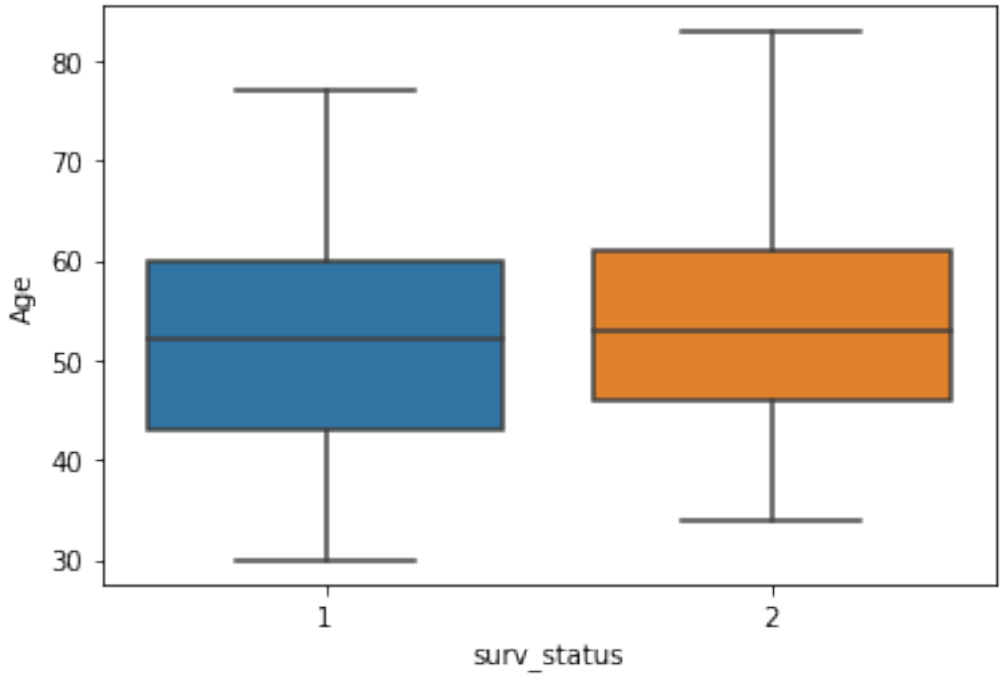
Out[7]:

	Age	oper_year	axil_nodes	surv_status
6	34	59	0	2
7	34	66	9	2
23	38	69	21	2
33	39	66	0	2
42	41	60	23	2

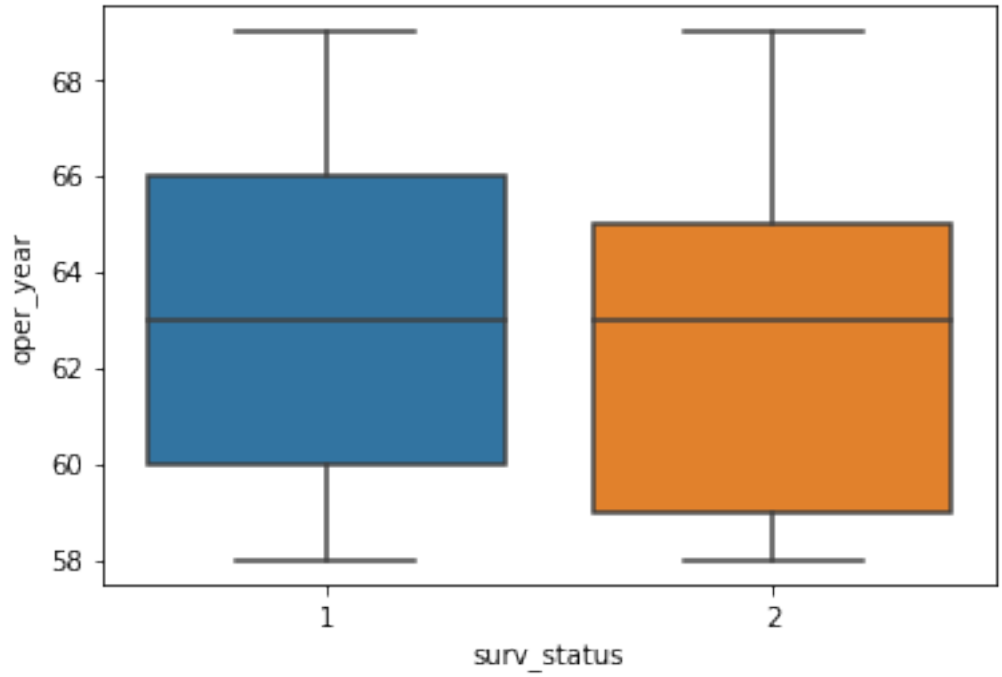
2.1 UNIVARIATE :-

2.1.1 1) Box Plot:-

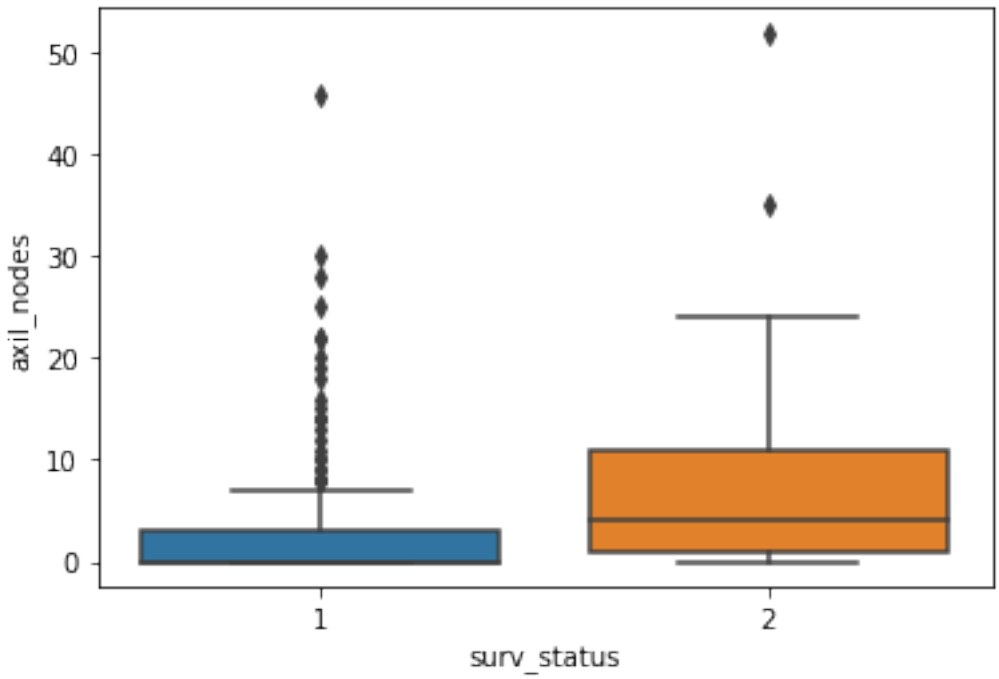
```
In [8]: #Box plot on the basis of Age
        sb.boxplot(x='surv_status',y='Age',data=hb)
        plt.show()
```



In [9]:
```
#Box plot on the basis of operation year
sb.boxplot(x='surv_status',y='oper_year',data=hb)
plt.show()
```



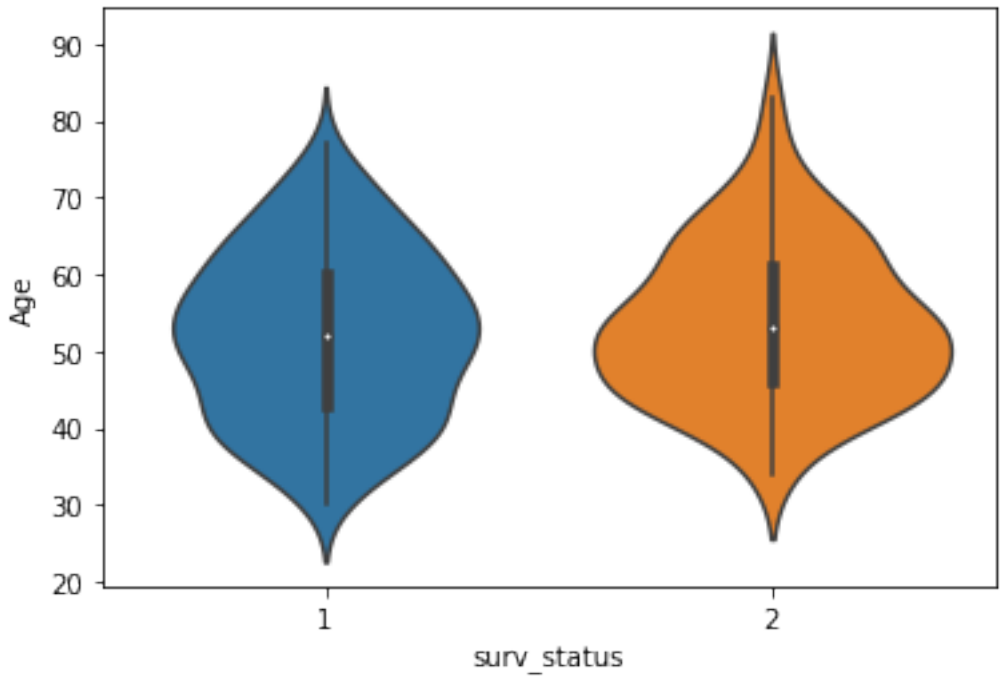
In [10]:
```
#Box plot on the basis of axil_nodes
sb.boxplot(x='surv_status',y='axil_nodes',data=hb)
plt.show()
```



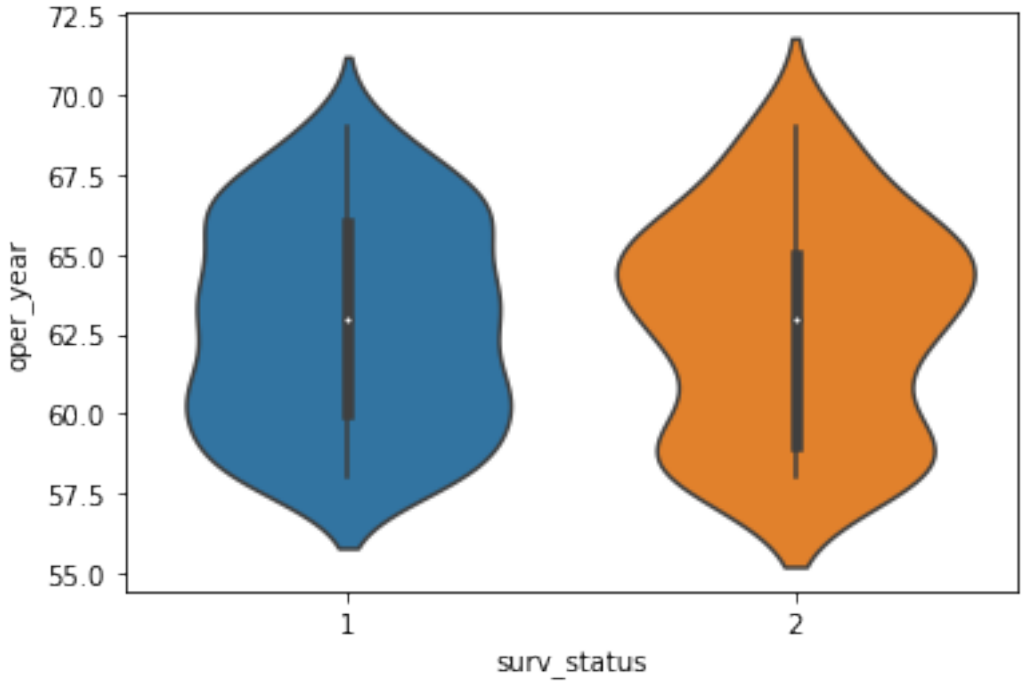
2.1.2 2) Violin Plot : -

In [11]:
```
#Violin Plot on the basis of Age
sb.violinplot(x='surv_status',y='Age',data=hb)
plt.show()
```

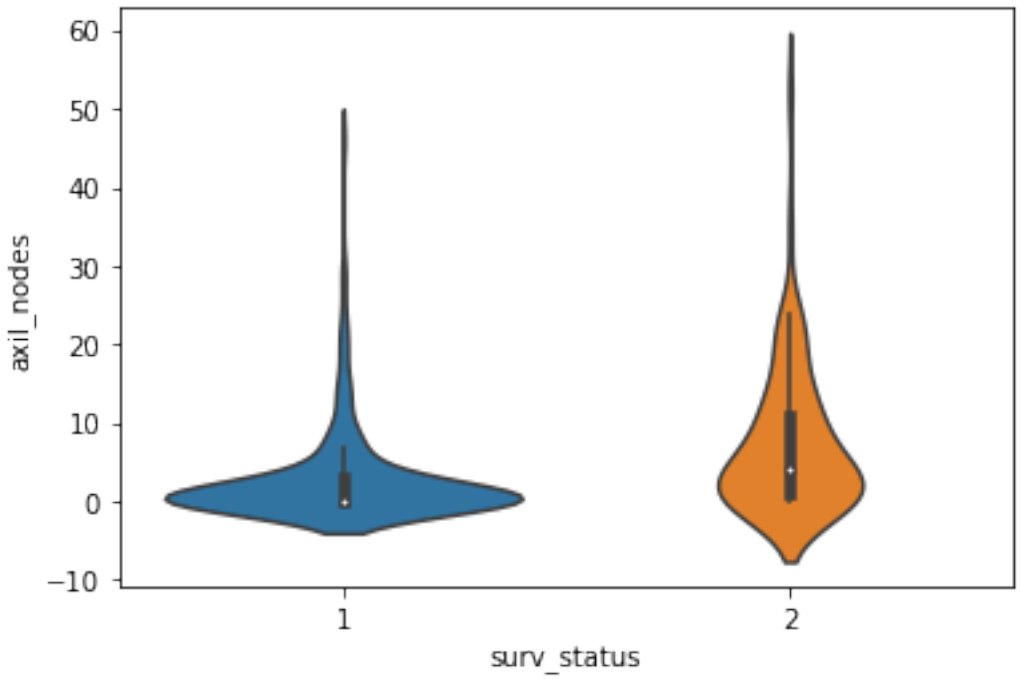
```
/home/piyush/.local/lib/python3.6/site-packages/scipy/stats/stats.py:1713: FutureWarning: Using
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```



In [12]:
```
#Violin Plot on the basis of operation year
sb.violinplot(x='surv_status',y='oper_year',data=hb)
plt.show()
```



In [13]:
```
#Violin Plot on the basis of axil nodes
sb.violinplot(x='surv_status',y='axil_nodes',data=hb)
plt.show()
```



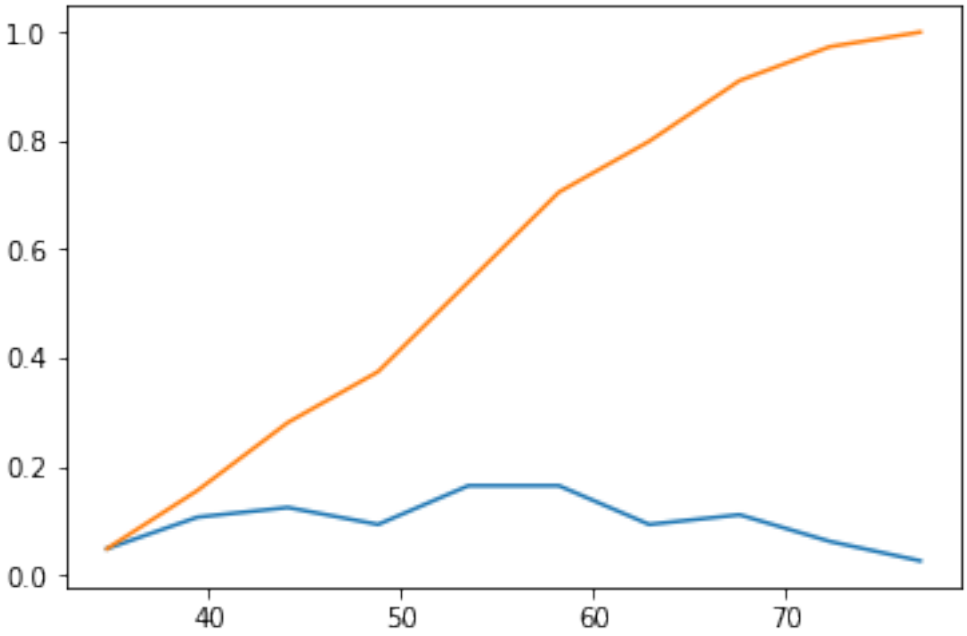
2.1.3 3) PDF and CDF :-

PDF :- a function of a continuous random variable, whose integral across an interval gives the probability that the value of the variable lies within the same interval.

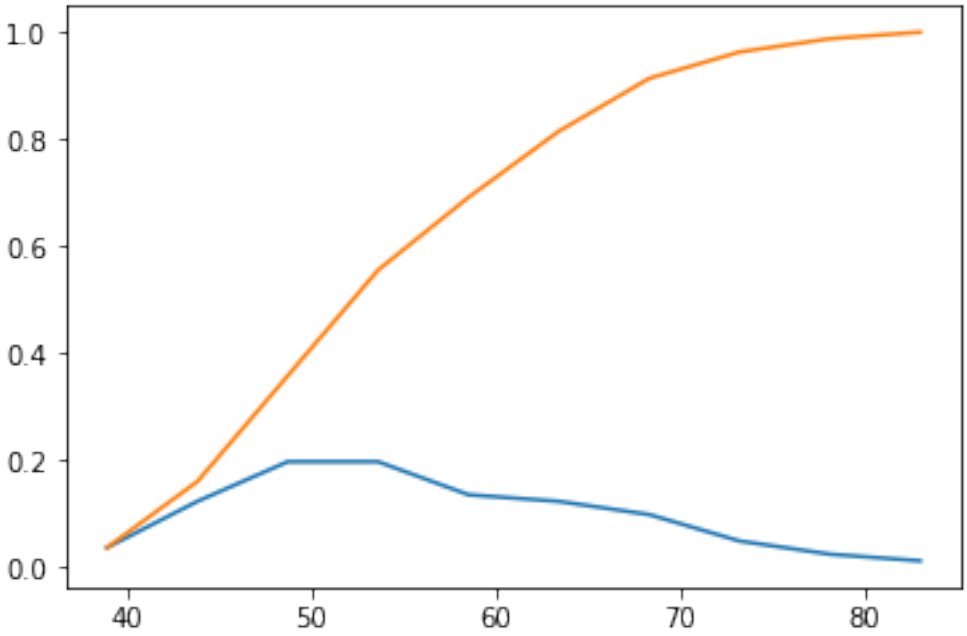
CDF:- cumulative distribution function (CDF) or cumulative frequency function, describes the probability that a variate takes on a value less than or equal to a number .

In [14]:
```
#PDF and CDF on Age (First DataFrame)

counts,bin_edges=np.histogram(h1['Age'],bins=10,density=True)
pdf=counts/sum(counts)
cdf=np.cumsum(pdf) #cummulative sum of pdf
plt.plot(bin_edges[1:],pdf)
plt.plot(bin_edges[1:],cdf)
```

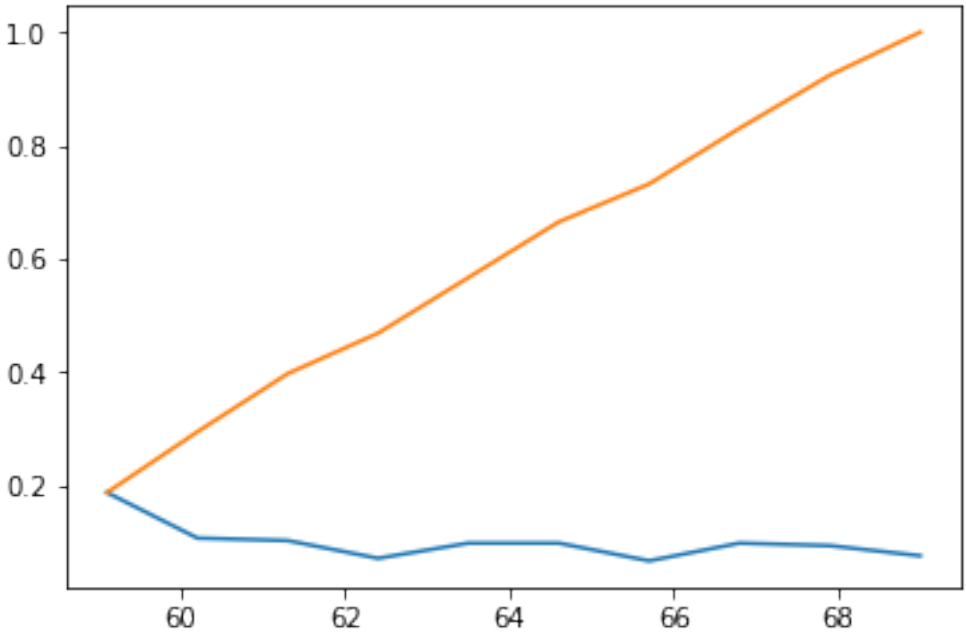
Out[14]:
```
[<matplotlib.lines.Line2D at 0x7f1cc09fe748>]
```



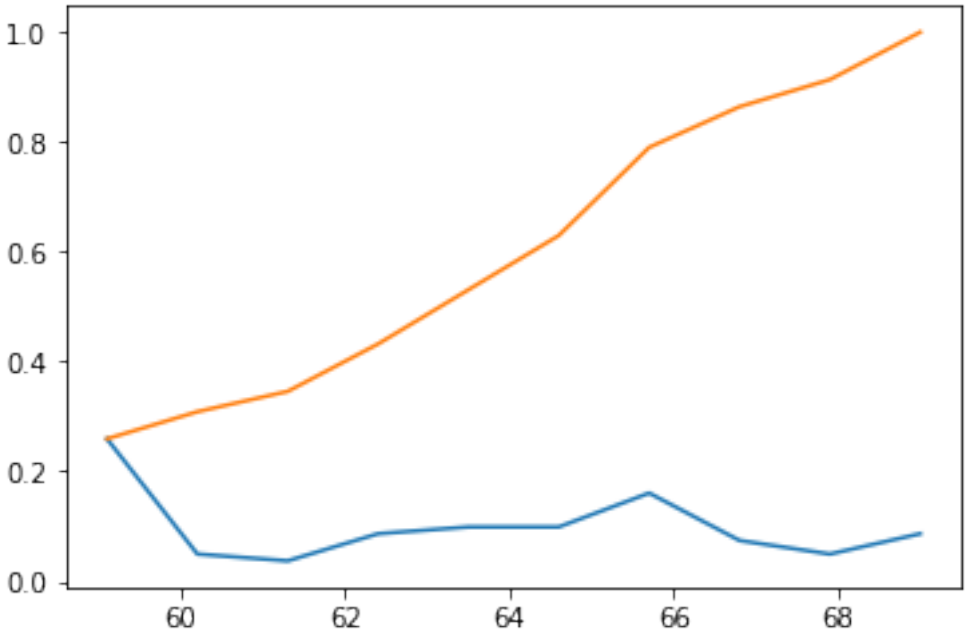
In [15]:
```
#PDF and CDF on Age (Second DataFrame)

counts,bin_edges=np.histogram(h2['Age'],bins=10,density=True)
pdf=counts/sum(counts)
cdf=np.cumsum(pdf) #cummulative sum of pdf
plt.plot(bin_edges[1:],pdf)
plt.plot(bin_edges[1:],cdf)
plt.show()
```



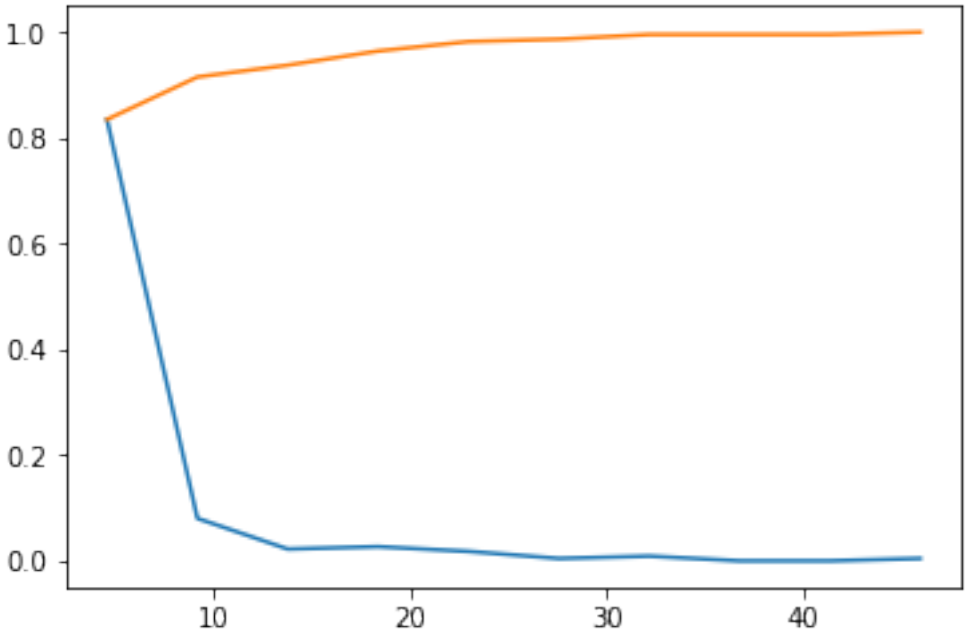
In [16]:
```
#PDF and CDF on OperationYear (First DataFrame)

counts,bin_edges=np.histogram(h1['oper_year'],bins=10,density=True)
pdf=counts/sum(counts)
cdf=np.cumsum(pdf) #cummulative sum of pdf
plt.plot(bin_edges[1:],pdf)
plt.plot(bin_edges[1:],cdf)
plt.show()
```



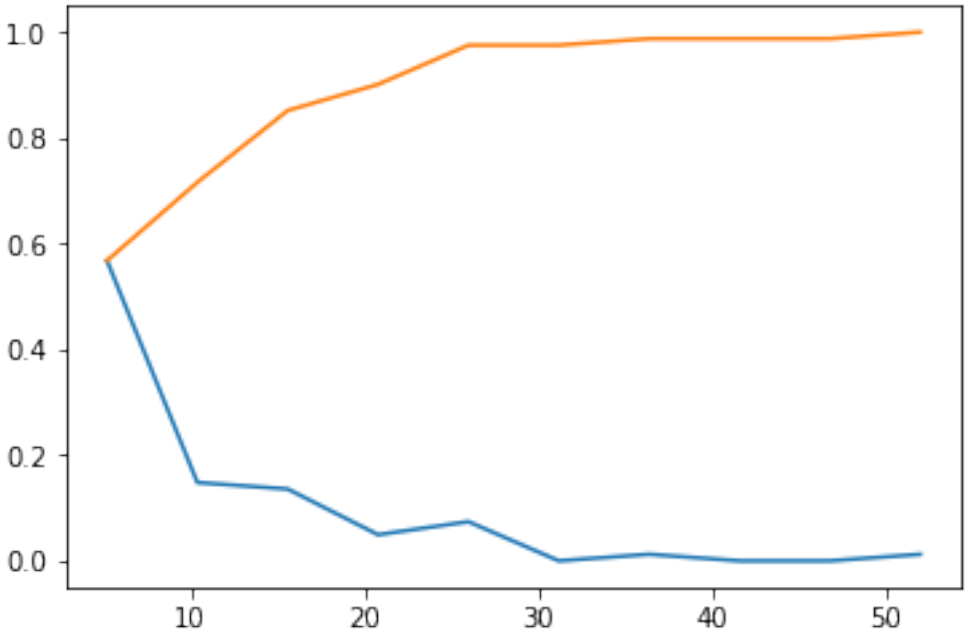
In [17]:
```
#PDF and CDF on OperationYear (Second DataFrame)

counts,bin_edges=np.histogram(h2['oper_year'],bins=10,density=True)
pdf=counts/sum(counts)
cdf=np.cumsum(pdf) #cummulative sum of pdf
plt.plot(bin_edges[1:],pdf)
plt.plot(bin_edges[1:],cdf)
plt.show()
```



In [18]:
```
#PDF and CDF on Axil Nodes (First DataFrame)

counts,bin_edges=np.histogram(h1['axil_nodes'],bins=10,density=True)
pdf=counts/sum(counts)
cdf=np.cumsum(pdf) #cummulative sum of pdf
plt.plot(bin_edges[1:],pdf)
plt.plot(bin_edges[1:],cdf)
plt.show()
```



In [19]:
```
#PDF and CDF on Axil Nodes (second DataFrame)

counts,bin_edges=np.histogram(h2['axil_nodes'],bins=10,density=True)
pdf=counts/sum(counts)
cdf=np.cumsum(pdf) #cummulative sum of pdf
plt.plot(bin_edges[1:],pdf)
plt.plot(bin_edges[1:],cdf)
plt.show()
```

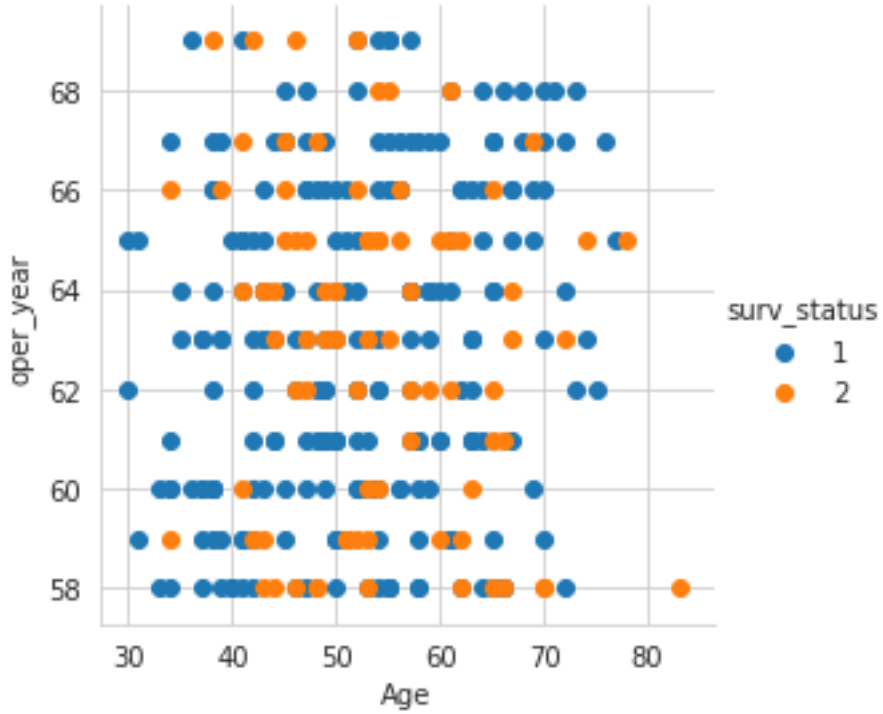


2.2 BIVARIATE :-

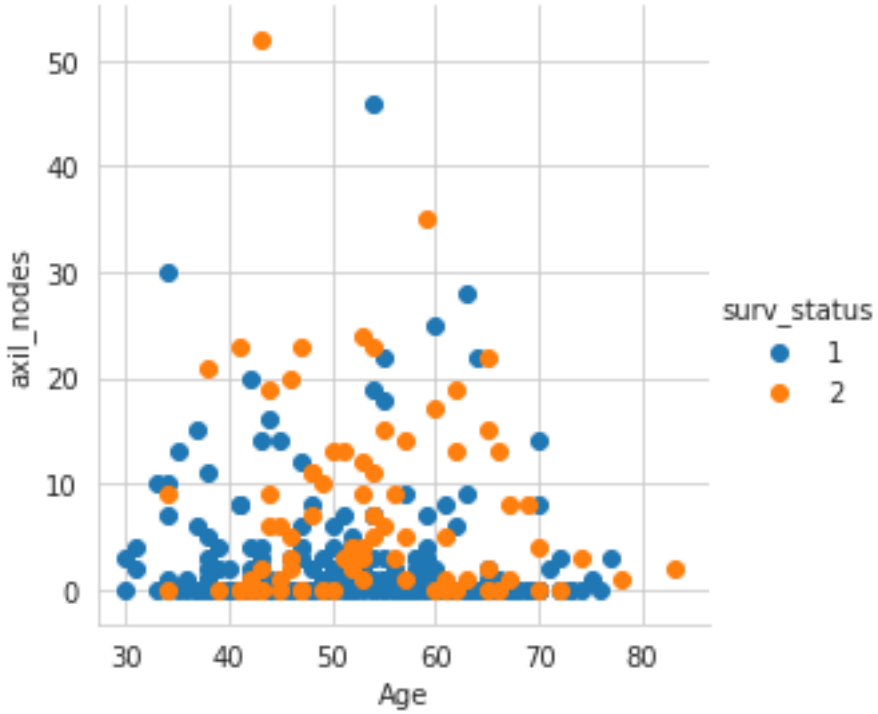
2.2.1 1) Scatter Plots:-

A graph in which the values of two variables are plotted along two axes, the pattern of the resulting points revealing any correlation present.

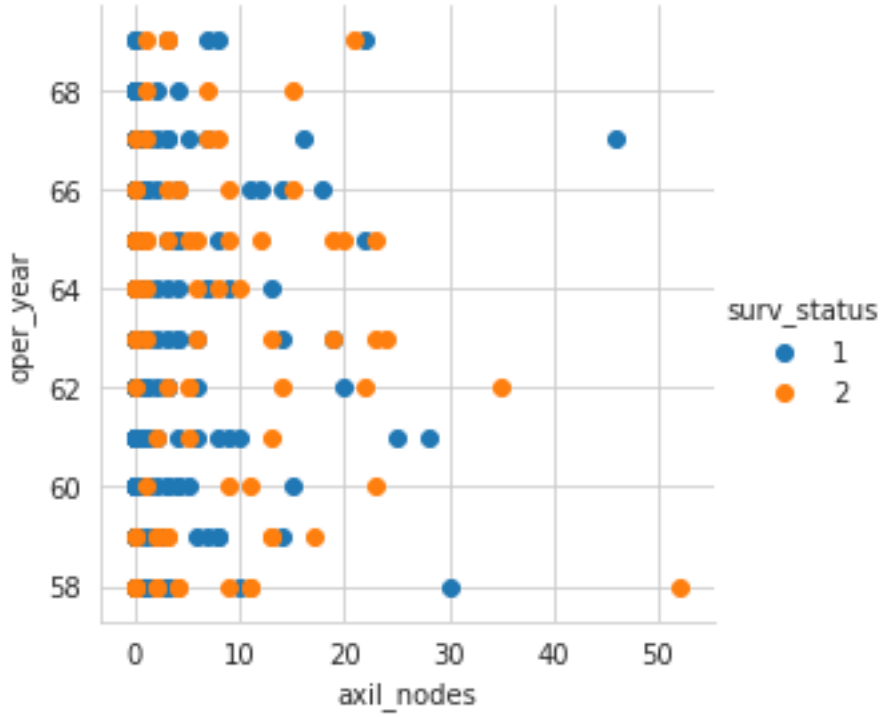
```
In [20]: #Scatter Plot of Age vs operation year

         sb.set_style('whitegrid')
         sb.FacetGrid(hb,hue='surv_status',height=4).map(plt.scatter,'Age','oper_year').add_leg
         plt.show()
```




```
In [21]: #Scatter Plot of Age vs Axil nodes
         sb.set_style('whitegrid')
         sb.FacetGrid(hb,hue='surv_status',height=4).map(plt.scatter,'Age','axil_nodes').add_l
         plt.show()
```



In [22]:
```
#Scatter Plot of Operation year vs Axil nodes
sb.set_style('whitegrid')
sb.FacetGrid(hb,hue='surv_status',height=4).map(plt.scatter,'axil_nodes','oper_year')
plt.show()
```



2.2.2 Pair Plots :-

seaborn.pairplot(data, hue=None, hue_order=None, palette=None, vars=None, x_vars=None, y_vars=None, kind='scatter', diag_kind='auto', markers=None, height=2.5, aspect=1, dropna=True, plot_kws=None, diag_kws=None, grid_kws=None, size=None)
https://seaborn.pydata.org/generated/seaborn.pairplot.html

```
In [23]: sb.set_style('whitegrid')
         sb.pairplot(hb,hue='surv_status',vars=['Age','oper_year','axil_nodes'],height=5)
         plt.show()
```

Conclusion :- Very difficult to differentiate due to overlapping

In [ ]: