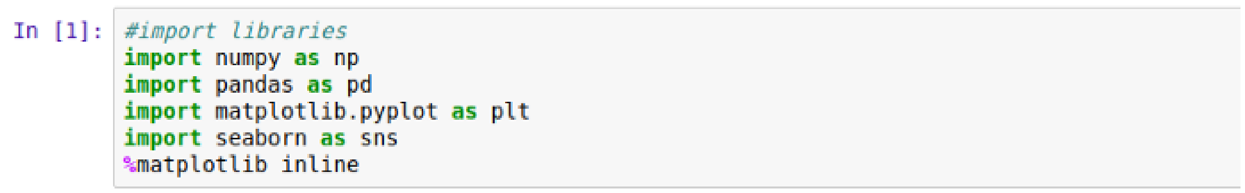
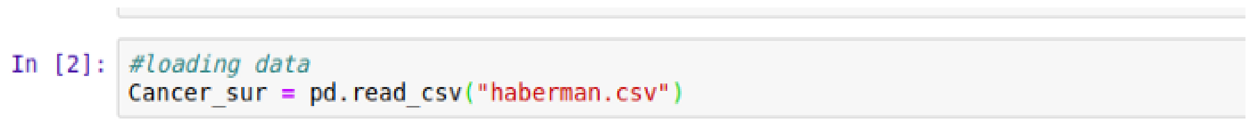
.importing libraries:



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

%matplotlib inline in above code snippet allows us to view our graphs in jupyter notebook itself.

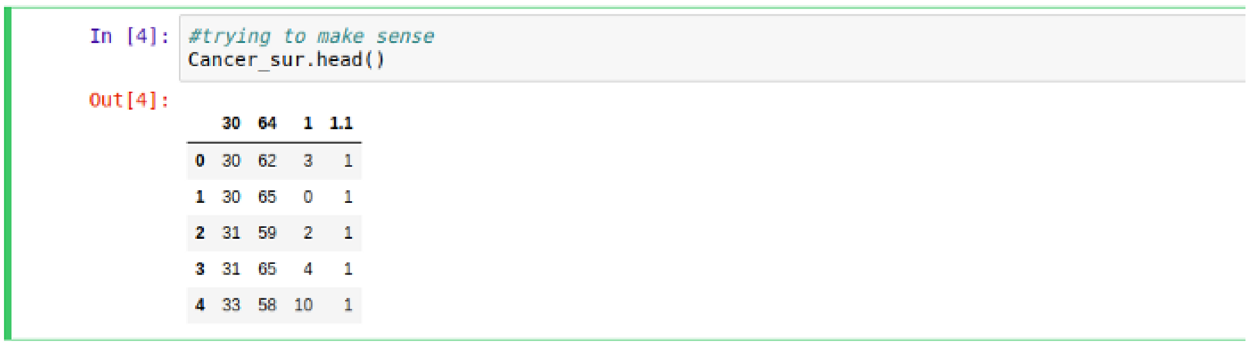
Load dataset: import the dataset to a variable of your own convention. i am going with *Cancer_sur* by using pandas function *pd.read_csv()*.



```
In [2]: #loading data
        Cancer_sur = pd.read_csv("haberman.csv")
```

loaind data set to a varaible

you can see the top 5 lines of data by using *Cancer_sur.head()*.



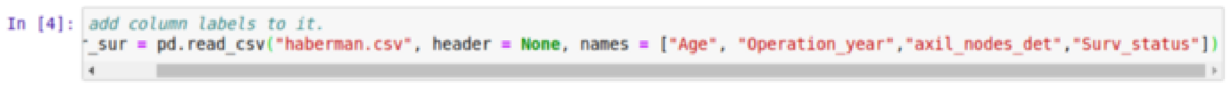
```
In [4]: #trying to make sense
        Cancer_sur.head()
Out[4]:
```

	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 column labels to it.

checking top5 rows of data

if you look at it, you can see top 5 rows, but not able to make sense, because there are no column labels to it. let's add columns to it.



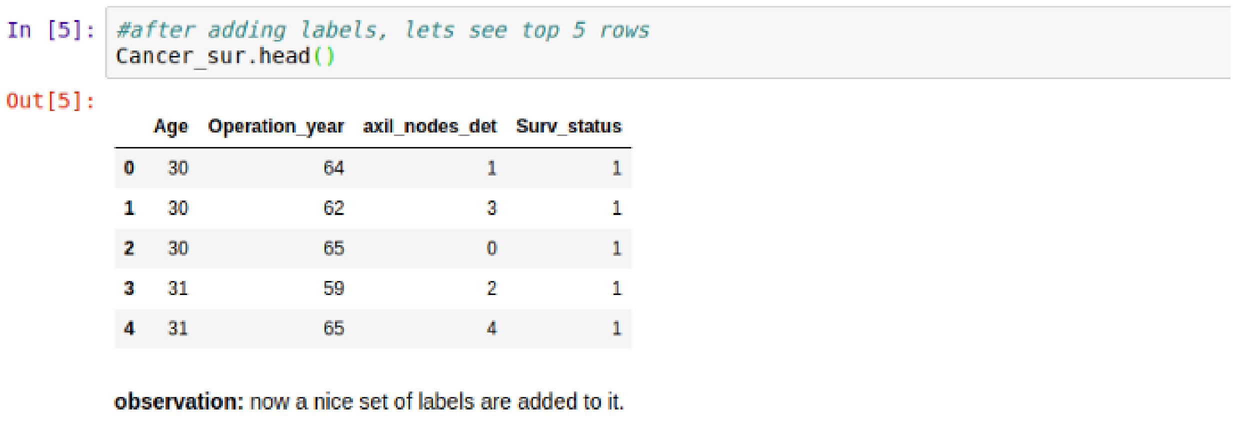
```
In [4]: add column labels to it.
        _sur = pd.read_csv("haberman.csv", header = None, names = ["Age", "Operation_year","axil_nodes_det","Surv_status"])
```

adding column labels and loading dataset again to Cancer_sur variable.

in the above snippet, *header* = *None* removes its headers, *names =[]* adds column names to the dataset as *“Age”, “Operation_year, “axil_nodes_det”, “Surv_status”.*

2.Some Basic analysis:

lets see top 5 rows after updating labels using *Cancer_sur.head().*



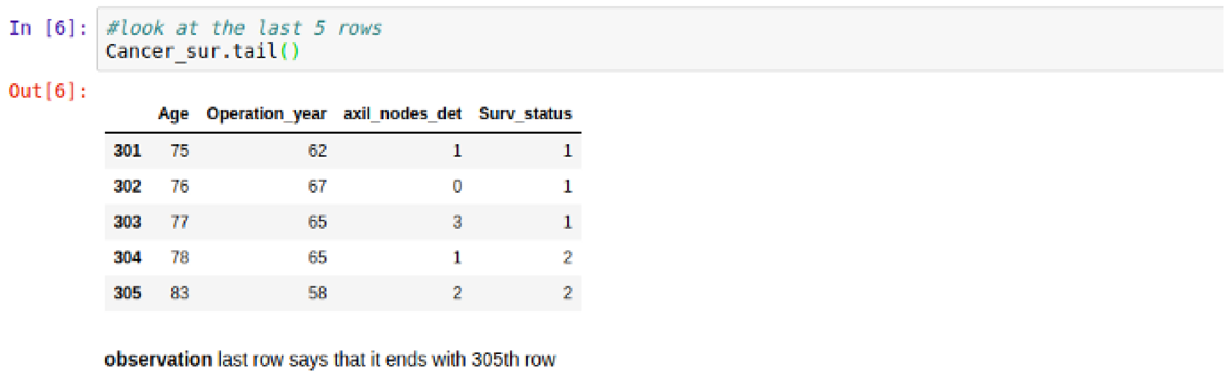
```
In [5]: #after adding labels, lets see top 5 rows
        Cancer_sur.head()
Out[5]:
```

	Age	Operation_year	axil_nodes_det	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

observation: now a nice set of labels are added to it.

image after labelling

lets see last 5 rows using *Cancer_sur.tail()*.



```
In [6]: #look at the last 5 rows
        Cancer_sur.tail()
Out[6]:
```

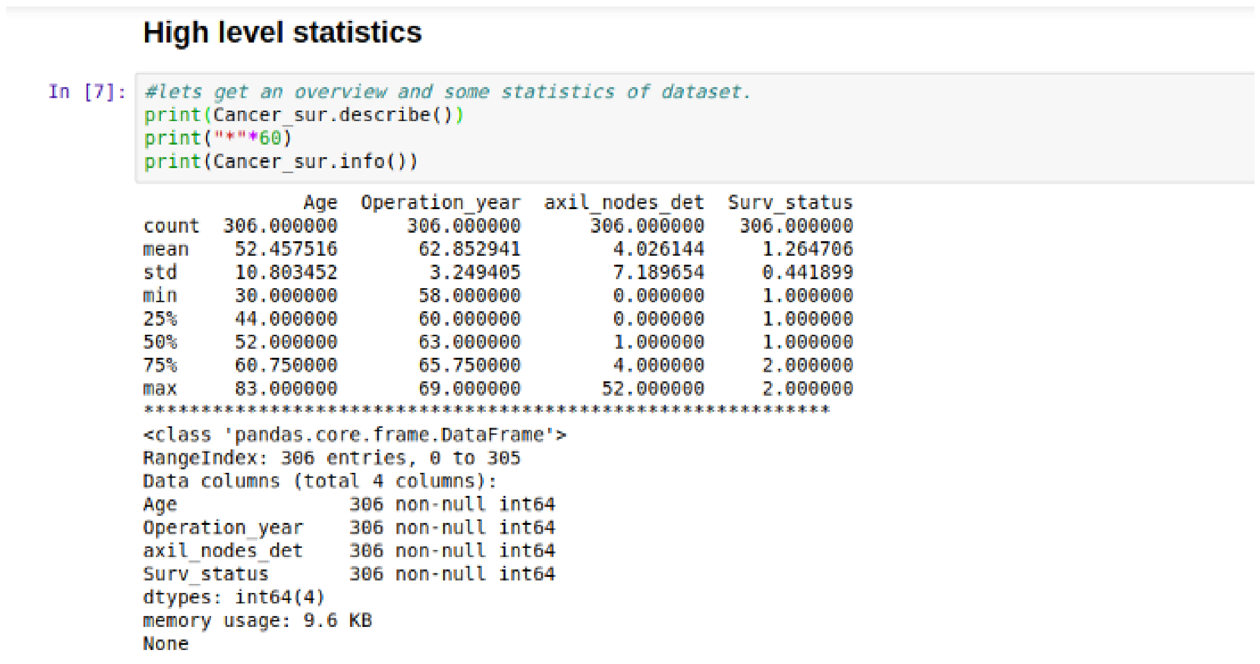
	Age	Operation_year	axil_nodes_det	Surv_status
301	75	62	1	1
302	76	67	0	1
303	77	65	3	1
304	78	65	1	2
305	83	58	2	2

observation last row says that it ends with 305th row

last 5 rows

3.High level statistics:

u can see count(gives total rows),Mean(average),std(standard deviation from one point to another),min,max and total coulmns of dataset and its rows, its data types by using *.describe()* and *.info()*.

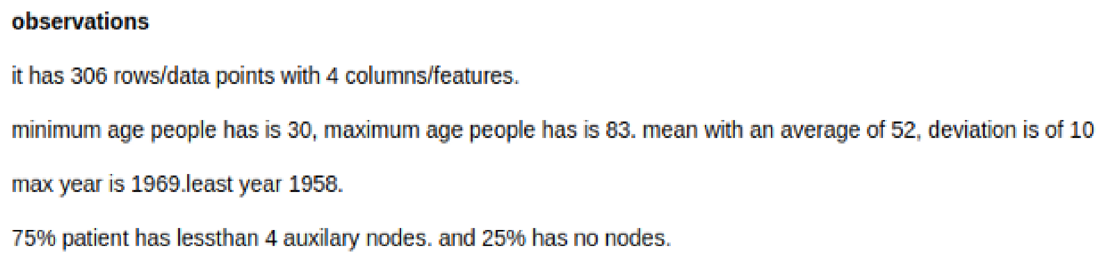


High level statistics

```
In [7]: #lets get an overview and some statistics of dataset.
        print(Cancer_sur.describe())
        print("*"*60)
        print(Cancer_sur.info())
```

```
              Age  Operation_year  axil_nodes_det  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
************************************************************
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 306 entries, 0 to 305
Data columns (total 4 columns):
Age                 306 non-null int64
Operation_year      306 non-null int64
axil_nodes_det      306 non-null int64
Surv_status         306 non-null int64
dtypes: int64(4)
memory usage: 9.6 KB
None
```

observations:



observations

it has 306 rows/data points with 4 columns/features.

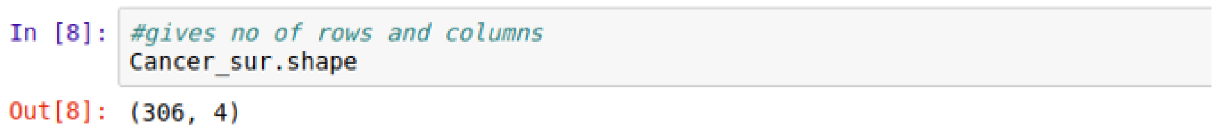
minimum age people has is 30, maximum age people has is 83. mean with an average of 52, deviation is of 10

max year is 1969.least year 1958.

75% patient has lessthan 4 auxilary nodes. and 25% has no nodes.

observations.

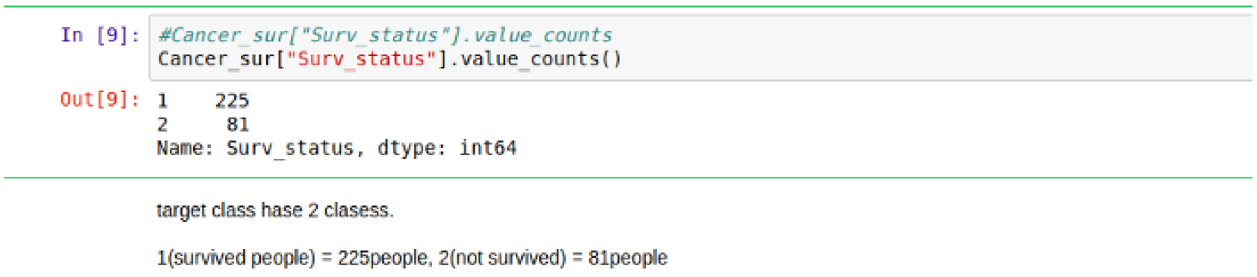
.*shape* gives no of rows and columns.



```
In [8]: #gives no of rows and columns
        Cancer_sur.shape
Out[8]: (306, 4)
```

total rows and columns of dataset

Surv_status is a target column where it gives 2 values 1(means survived) and 2(not survived). let's see them. in the entire dataset, we have 225 rows(people) with value 1(survived) and 81rows(people) with value 2(not survived).



```
In [9]: #Cancer_sur["Surv_status"].value_counts
        Cancer_sur["Surv_status"].value_counts()
Out[9]: 1    225
        2     81
        Name: Surv_status, dtype: int64
```

target class hase 2 clasess.

1(survived people) = 225people, 2(not survived) = 81people

now let's see some univariate analysis.

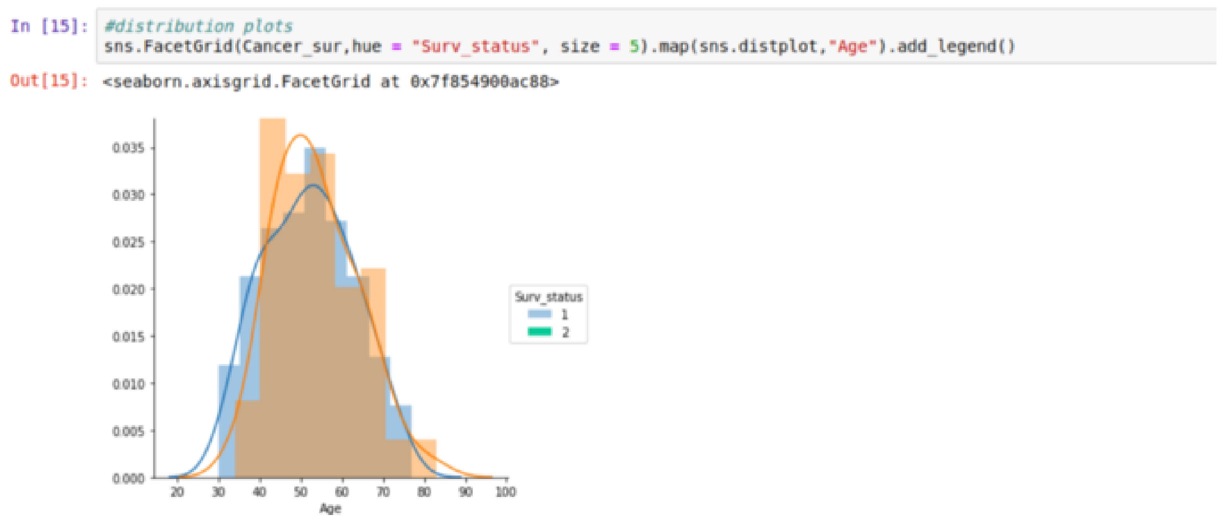
Univariate analysis(PDF, CDF, Boxplot, Violin plots,Distribution plots):-

Analysis done based only on one variable. we are not going to the math behind these concepts, for now, let's see what these are in graphs. (*please have some basic idea on these concepts if you don't get them by seeing graphs*).

Distribution plot:

people follow their own ways of coding that gives similar results. The distribution plot gives the density of distributions from point to point in general terms.

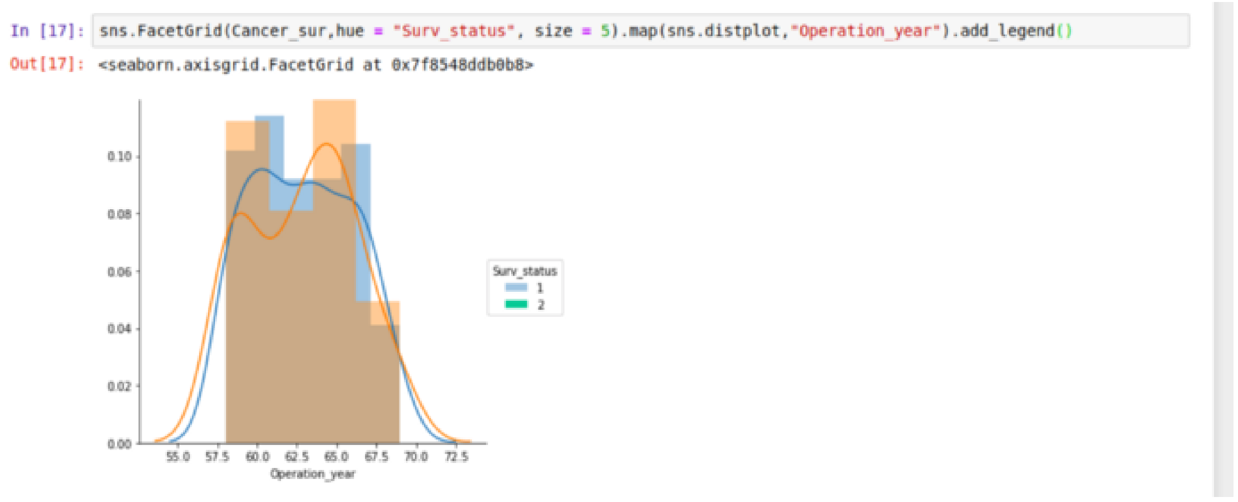
we draw this using seaborn as sns, Facetgrid gives grid layout, Cancer_sur is a variable that we loaded data into. Hue colours the value/column name that you give to it. Size is graph size and mapping all these to sns.distplot on “Age” column.



```
In [15]: #distribution plots
         sns.FacetGrid(Cancer_sur,hue = "Surv_status", size = 5).map(sns.distplot,"Age").add_legend()
Out[15]: <seaborn.axisgrid.FacetGrid at 0x7f854900ac88>
```

from the above graph, you can observe that people age 50 to 60 have more survival rate.

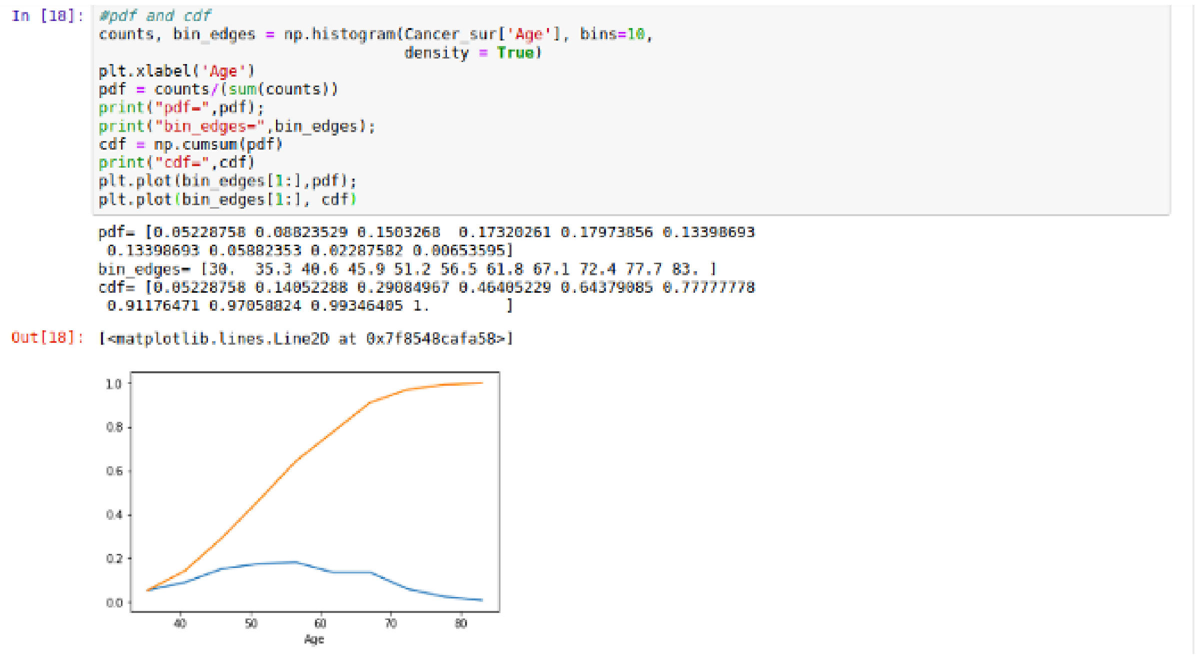
now let's draw the same with another column “Operation_year”



```
In [17]: sns.FacetGrid(Cancer_sur,hue = "Surv_status", size = 5).map(sns.distplot,"Operation_year").add_legend()
Out[17]: <seaborn.axisgrid.FacetGrid at 0x7f8548ddb0b8>
```

from the above graph, we can say that people who had an operation in the year from 58 to 66 had more survival rate.

CDF(cumulative distributive function), PDF(probability density function):



```
In [18]: #pdf and cdf
         counts, bin_edges = np.histogram(Cancer_sur['Age'], bins=10,
                                          density = True)
         plt.xlabel('Age')
         pdf = counts/(sum(counts))
         print("pdf=",pdf);
         print("bin_edges=",bin_edges);
         cdf = np.cumsum(pdf)
         print("cdf=",cdf)
         plt.plot(bin_edges[1:],pdf);
         plt.plot(bin_edges[1:], cdf)

         pdf= [0.05228758 0.08823529 0.1503268  0.17320261 0.17973856 0.13398693
          0.13398693 0.05882353 0.02287582 0.00653595]
         bin_edges= [30.  35.3 40.6 45.9 51.2 56.5 61.8 67.1 72.4 77.7 83. ]
         cdf= [0.05228758 0.14052288 0.29084967 0.46405229 0.64379085 0.77777778
          0.91176471 0.97058824 0.99346405 1.        ]

Out[18]: [<matplotlib.lines.Line2D at 0x7f8548cafa58>]
```

we draw this using univariable “age” and drawn cumulative distribution function and probability density function.

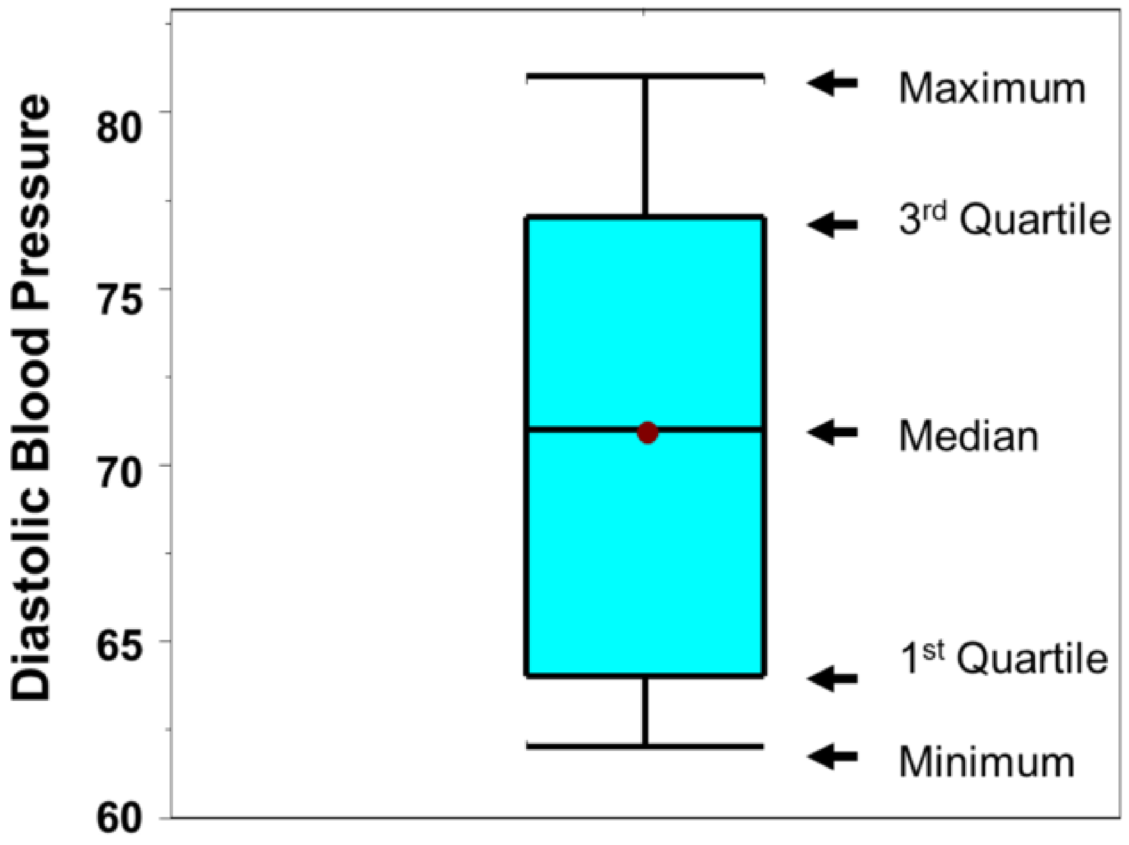
if we draw a straight line from Age value at 70, then it intersects the curve Cumulative distribution function(yellow) at a value approximately equal to 0.8 i.e there are 80% people from a cumulative sum of 30 to 70 age.

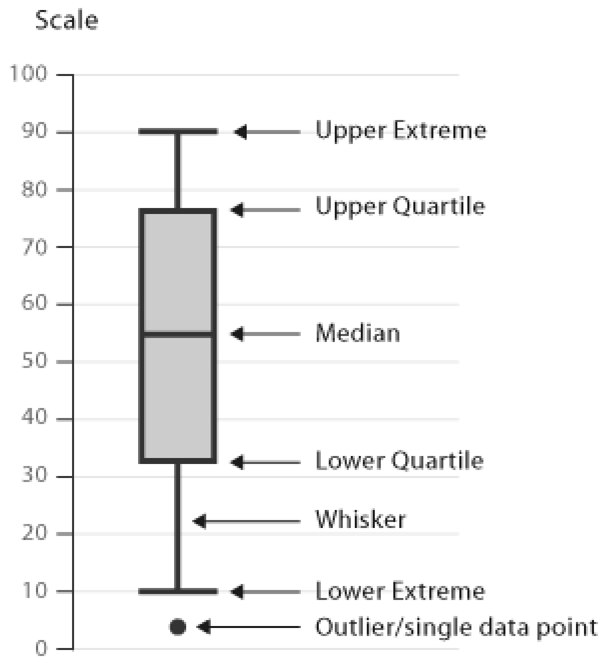
Bivariate analysis:

this gives the relationship between the two variables, hence its called bivariate analysis.

Box plot:

Box plot is a nice way of viewing some statical values along with relationship between two values.


Diastolic Blood Pressure
80
75
70
65
60
Maximum
3rd Quartile
Median
1st Quartile
Minimum

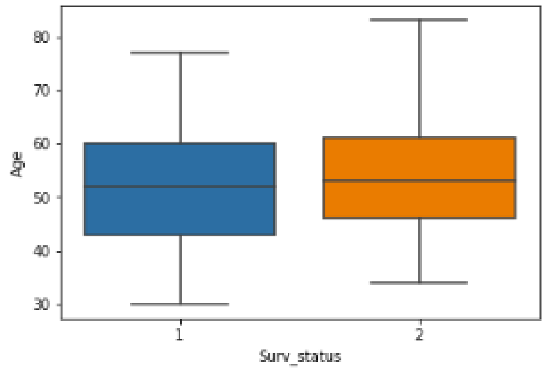


note: from the next coming to all graphs, graphs that are in blue shows value1(survived) and yellow with value2(not survived).

it uses seaborn as sns to visualise boxplot between X =' Surv_status', y= "'Age", data= Cancer_surv, because it is where all our dataset loaded to

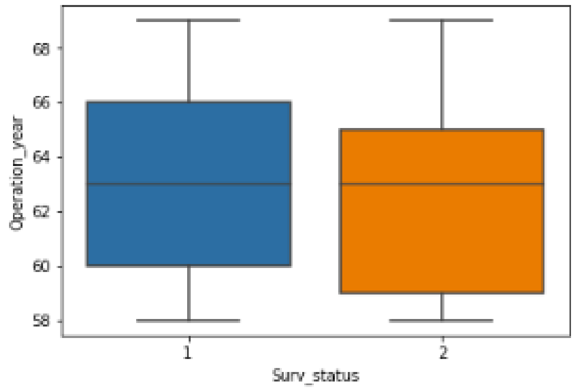
Box plot

```
In [22]: #boxplot
         sns.boxplot(x='Surv_status',y='Age', data=Cancer_sur)
         plt.show()
```



Now, let's draw the Box plot between *Surv_Status* and *Operation year*.

```
In [23]: sns.boxplot(x='Surv_status',y='Operation_year', data=Cancer_sur)
         plt.show()
```

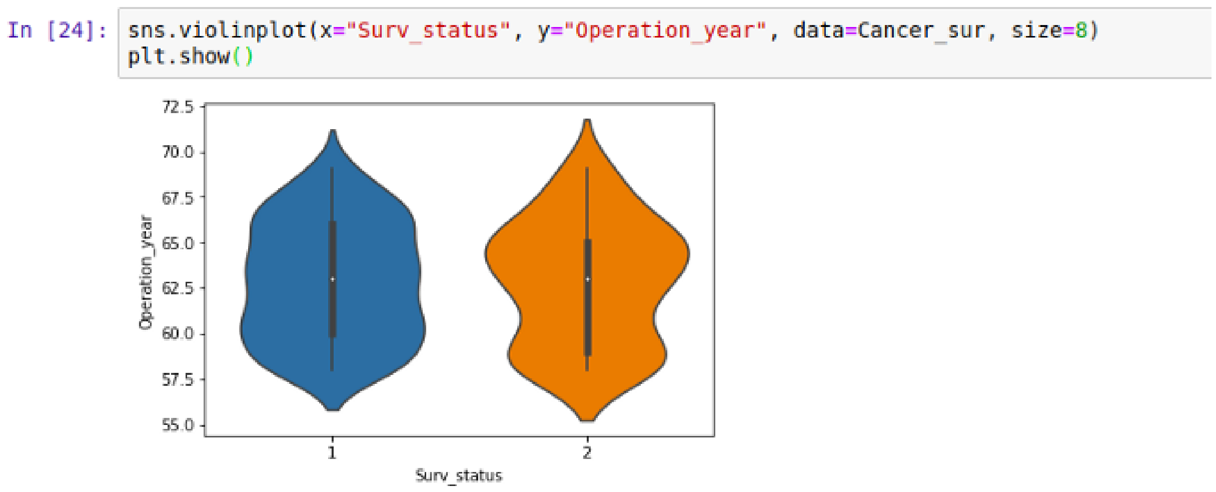


people who has operation in year 1958 to 1966 survived more.

it is observed that people that had the operation in the year 1958 to 1966 survived.

Violin plot:

violin plots also like box plots, but these give pdf along with box plots in it. they look a violin, so named to .please see this **image.**



```
In [24]: sns.violinplot(x="Surv_status", y="Operation_year", data=Cancer_sur, size=8)
         plt.show()
```

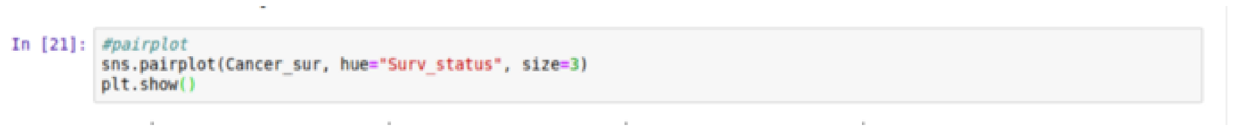
this is the same as the above box plot, but here we used the violin plot to look more pretty and to get the pdf at the same time.

you can observe that people with operation year from 58 to 66 survived more as after that the graphs decreased as you can see in the above figure.

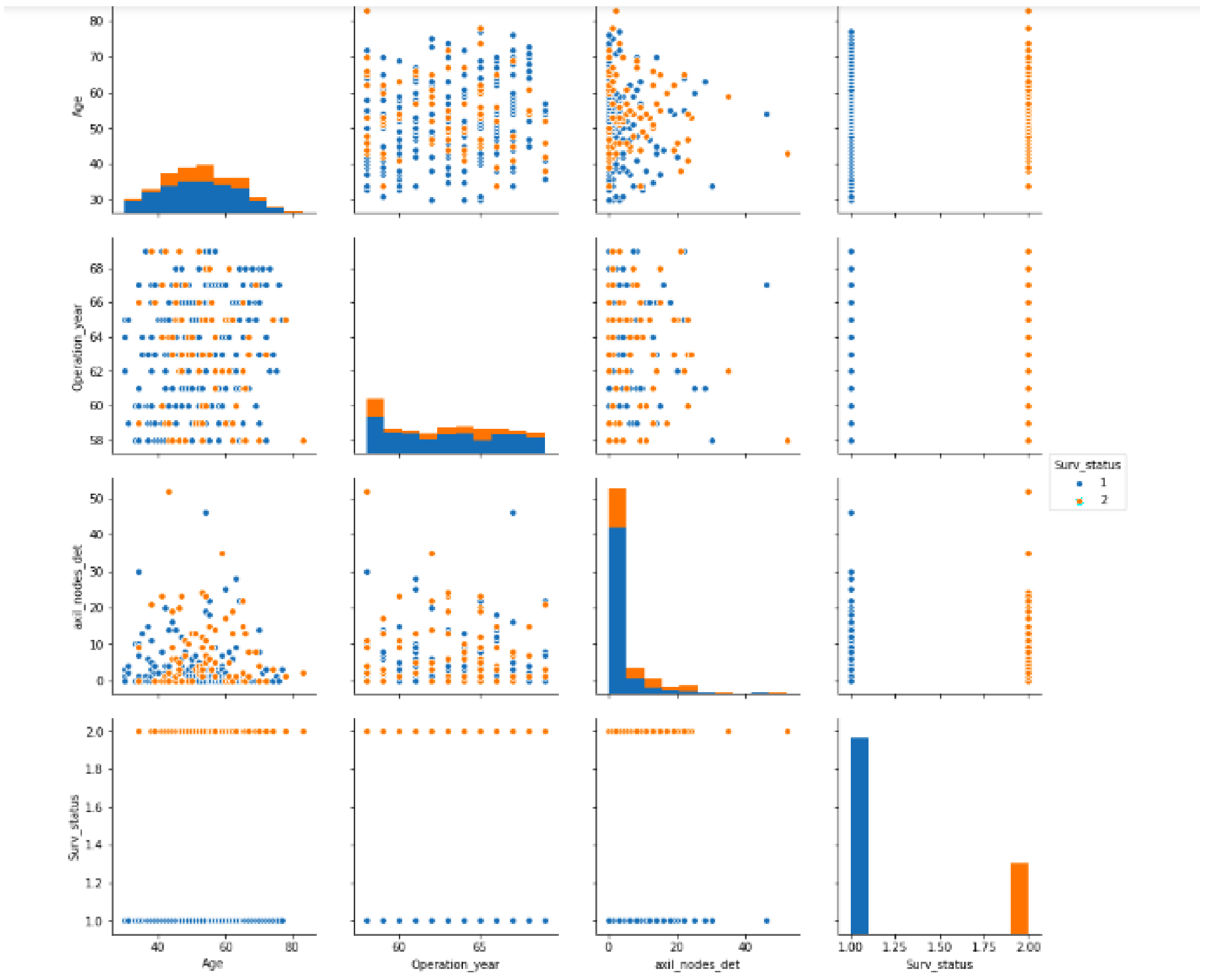
Multivariate Analysis:

Pair plot:

pair plot shows a clear and nice view of all variables and their relation ship with all other variables.



```
In [21]: #pairplot
         sns.pairplot(Cancer_sur, hue="Surv_status", size=3)
         plt.show()
```



it uses seaborn as sns to draw a pair plot with dataset variable

Cancer_sur and colours the graph using Surv_status with size = 3