3. Plotting for Exploratory data analysis (EDA)

(3.1) Basic Terminology

- What is EDA?

  Exploratory Data Analysis, which means the detailed analytical view of an dataset. Plotting histogram,pdf,cdf and all the statistics work.
- Data-point/vector/Observation Data -point -> say data set has 1000 rows, each row is called data point. vector -> anything which represents more than 1 D called vector.
- Data-set.-> which contains data about the project. rows and columns like table.
- Feature/Variable/Input-variable/Dependent-varibale: Feature- column Variable-xi's,yi's input variable: all xi's(rows and columns) Dependent variable-> which depends input variable for analysis.
- Label/Indepdendent-variable/Output-varible/Class/Class-label/Response label Label- output variable. (yi's)
- Vector: 2-D, 3-D, 4-D,.... n-D

Q. What is a 1-D vector: Scalar

Haberman dataset

Data Description The Haberman's survival 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.

Attribute Information:

Age of patient at time of operation (numerical) Patient's year of operation (year - 1900, numerical) Number of positive axillary nodes detected (numerical) Survival status (class attribute) 1 = the patient survived 5 years or longer 2 = the patient died within 5 years

- Objective: Classify a Survival status
- Importance of domain knowledge.

In [2]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np



#Load haberman.csv into a pandas dataFrame.
df = pd.read_csv("haberman.csv")
```

In [3]:
```
# (Q) how many data-points and features?
print (df.shape)
```

```
(306, 4)
```

In [4]:
```
#(Q) What are the column names in our dataset?
print (df.columns)
```

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

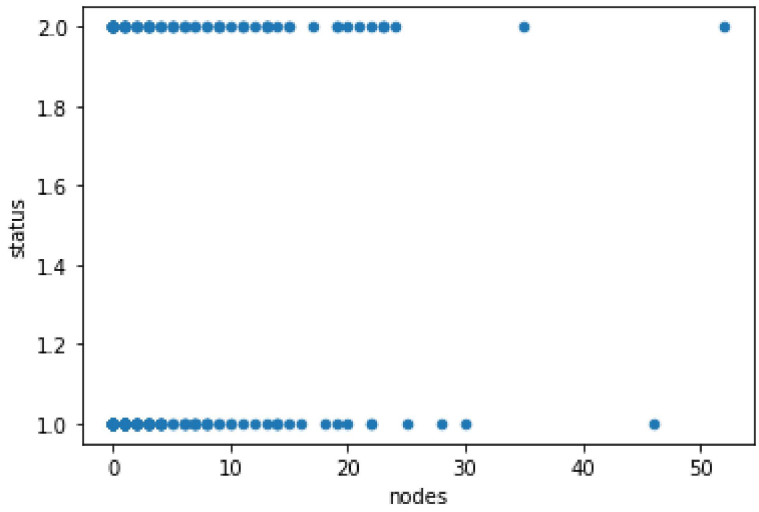
In [6]:
```
#(Q) How many data points for each class are present?
#(or) How many status are present?
print(df["status"].value_counts())
print(df["age"].value_counts())
print(df["nodes"].value_counts())
# balanced-dataset vs imbalanced datasets
#haberman is a not balanced dataset as the number of data points for survived and
```

```
1    225
2     81
Name: status, dtype: int64
52    14
54    13
50    12
47    11
53    11
43    11
57    11
55    10
65    10
49    10
38    10
41    10
61     9
45     9
42     9
63     8
59     8
62     7
44     7
58     7
56     7
46     7
70     7
34     7
48     7
37     6
67     6
60     6
51     6
39     6
66     5
64     5
72     4
69     4
40     3
30     3
68     2
73     2
74     2
36     2
35     2
33     2
31     2
78     1
71     1
```

```
75     1
76     1
77     1
83     1
Name: age, dtype: int64
0     136
1      41
2      20
3      20
4      13
6       7
7       7
8       7
5       6
9       6
13      5
14      4
11      4
10      3
15      3
19      3
22      3
23      3
12      2
20      2
46      1
16      1
17      1
18      1
21      1
24      1
25      1
28      1
30      1
35      1
52      1
Name: nodes, dtype: int64
```

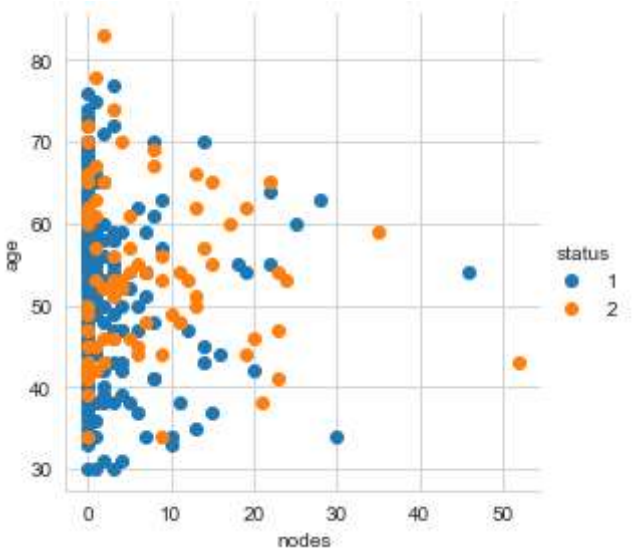
(3.2) 2-D Scatter Plot

In [7]:
```
#2-D scatter plot:
#ALWAYS understand the axis: labels and scale.

df.plot(kind='scatter', x='nodes', y='status') ;
plt.show()

#cannot make much sense out it.
#What if we color the points by thier class-label/flower-type.
```



In [ ]:
```
**Observations:
    with this 2d-scatter plot , we are not able to diffrentiate
    which feature has more importance to determine status.
```

In [25]:
```
# 2-D Scatter plot with color-coding for each flower type/class.
# Here 'sns' corresponds to seaborn.
sns.set_style("whitegrid");
sns.FacetGrid(df, hue="status", height=4) \
   .map(plt.scatter, "nodes", "age") \
   .add_legend();
plt.show();

# Notice that the blue points can be easily seperated
# from red and green by drawing a line.
# But red and green data points cannot be easily seperated.
# Can we draw multiple 2-D scatter plots for each combination of features?
# How many cobinations exist? 4C2 = 6.
```



Observation(s):
1.with age and nodes , we are not able to differentiate survival status,it has overlap of both data points.
2.there is max 0-10 nodes and more data points above age 40.

3D Scatter plot

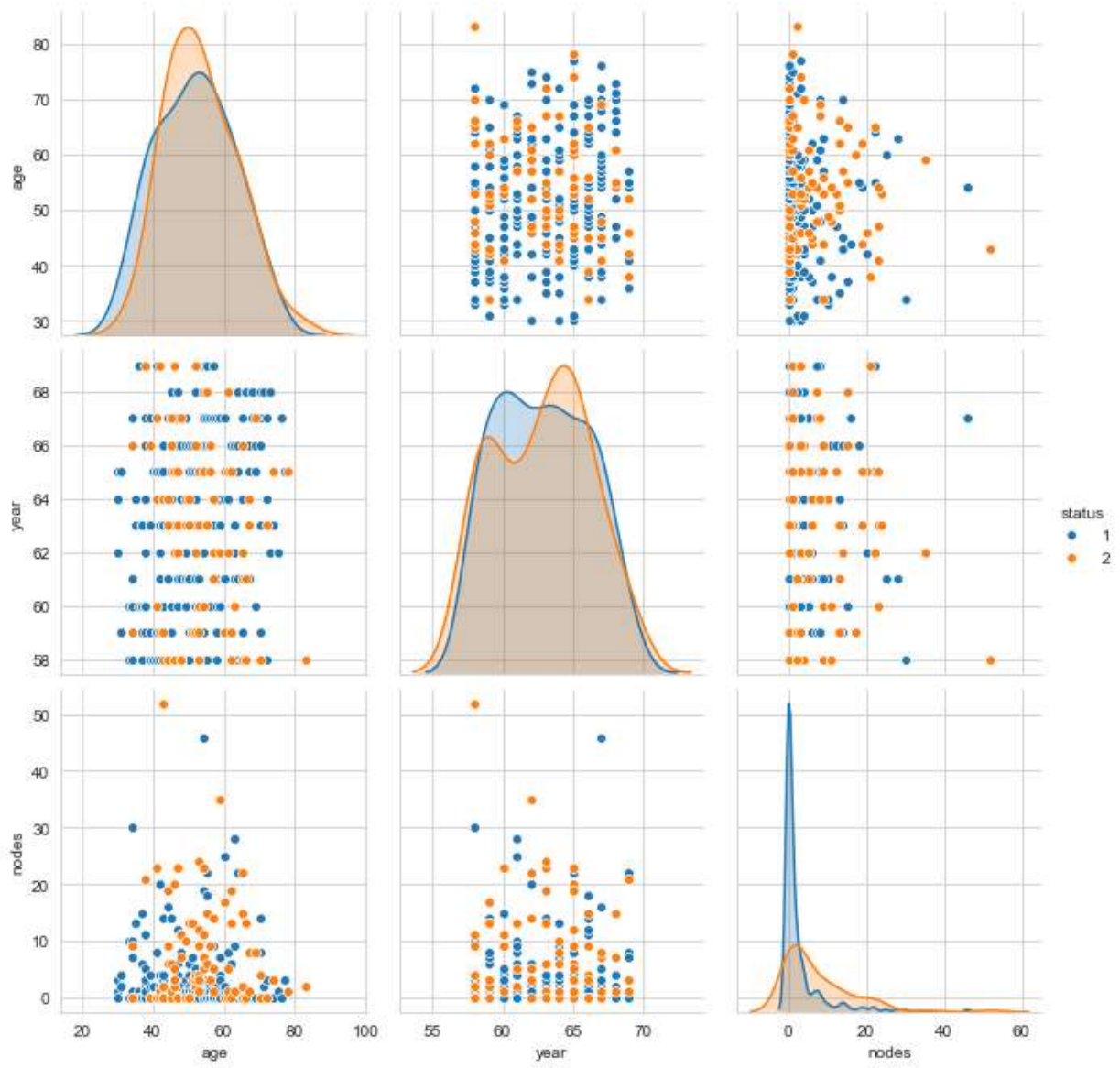
https://plot.ly/pandas/3d-scatter-plots/ (https://plot.ly/pandas/3d-scatter-plots/)

Needs a lot to mouse interaction to interpret data.

What about 4-D, 5-D or n-D scatter plot?

(3.3) Pair-plot

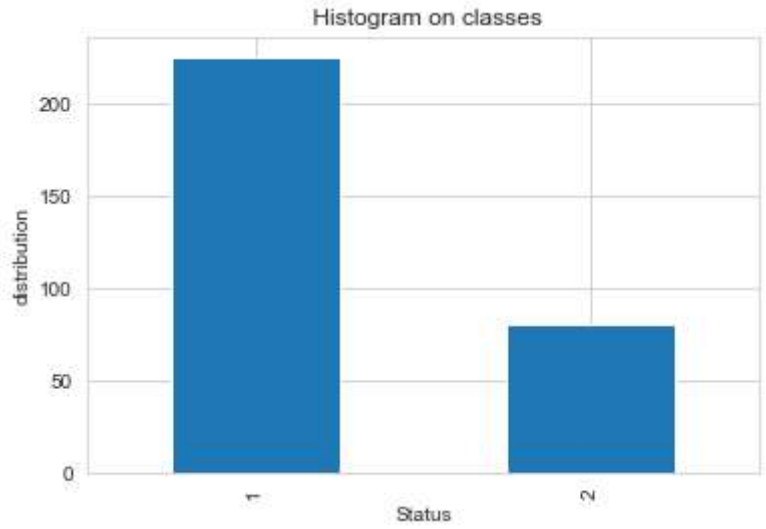
In [9]:
```
# pairwise scatter plot: Pair-Plot
# Dis-advantages:
##Can be used when number of features are high.
##Cannot visualize higher dimensional patterns in 3-D and 4-D.
#Only possible to view 2D patterns.
plt.close();
sns.set_style("whitegrid");
sns.pairplot(df, hue="status", height=3);
plt.show()
# NOTE: the diagnol elements are PDFs for each feature. PDFs are expalined below.
```



In [ ]:
```
**Observations**
1.there is strong relationship between nodes and status. if the nodes between 0 -
2. more than that shows not survival.
3.with year,age we cant distinguish anything because both has overlapping .
```

(3.4) Histogram, PDF, CDF

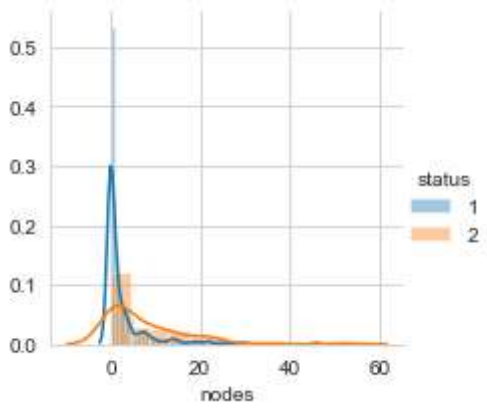
In [10]:
```
df_status = pd.value_counts(df["status"])
df_status.plot(kind = 'bar')
plt.title(" Histogram on classes")
plt.xlabel("Status")
plt.ylabel("distribution")
plt.show()
```



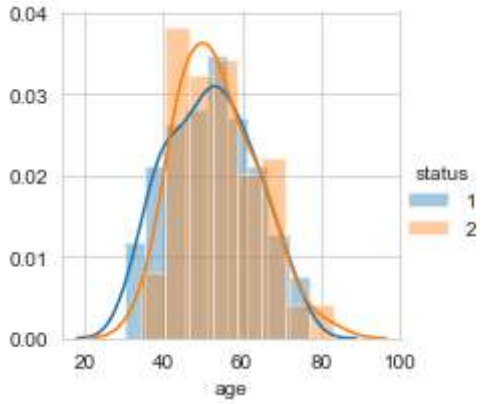
In [11]:
```
df_status.values
```

Out[11]:
```
array([225,  81], dtype=int64)
```

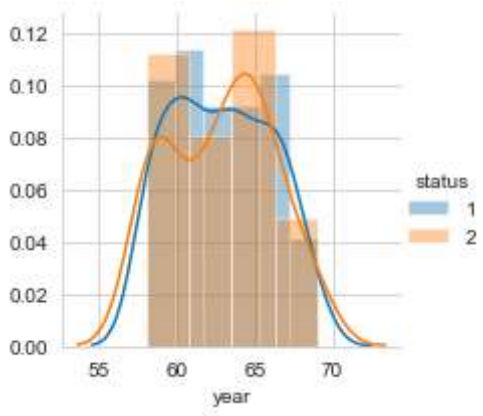
In [13]:
```
sns.FacetGrid(df, hue="status", height=3) \
   .map(sns.distplot, "nodes") \
   .add_legend();
plt.show();
```



In [23]:
```
sns.FacetGrid(df, hue="status", height=3) \
   .map(sns.distplot, "age") \
   .add_legend();
plt.show();
```



In [24]:
```
sns.FacetGrid(df, hue="status", height=3) \
   .map(sns.distplot, "year") \
   .add_legend();
plt.show();
```



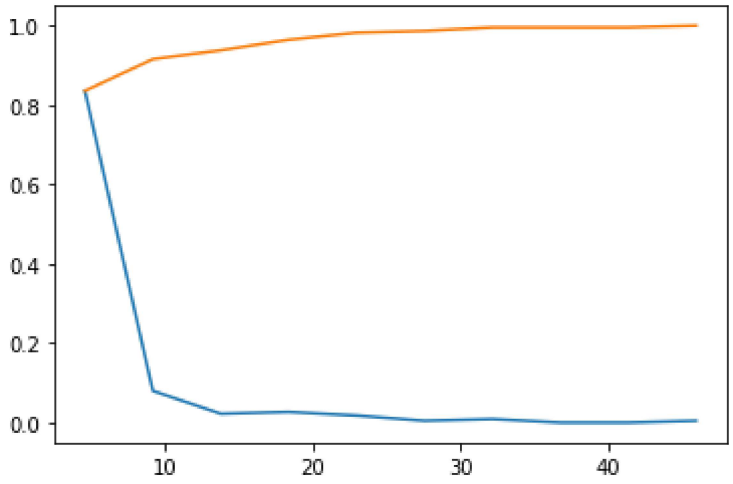
In [ ]:
```
**Observations**
    Histogram shows that how many columns have status 1 and 2.
    Distribution plot shows that PDF on each feature. we can see that with age ar
    year we cant differentiate status as it has too much
    overlapping, only with nodes we can conclude that if less nodes
    more survival and more nodes less survival.
```

In [16]:
```
status_1 = df.loc[df["status"] == 1];
status_2 = df.loc[df["status"] == 2];
print(status_1)
```

```
     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
..   ...   ...    ...     ...
298   73    68      0       1
300   74    63      0       1
301   75    62      1       1
302   76    67      0       1
303   77    65      3       1

[225 rows x 4 columns]
```

In [ ]:
```
# Need for Cumulative Distribution Function (CDF)
# How to construct a CDF?
# How to read a CDF?

#Plot CDF of petal_length

counts, bin_edges = np.histogram(status_1['nodes'], bins=10,
                                 density = True)
pdf = counts/(sum(counts))
print(pdf);
print(bin_edges)

#compute CDF
cdf = np.cumsum(pdf)
plt.plot(bin_edges[1:],pdf)
plt.plot(bin_edges[1:], cdf)



plt.show();
```

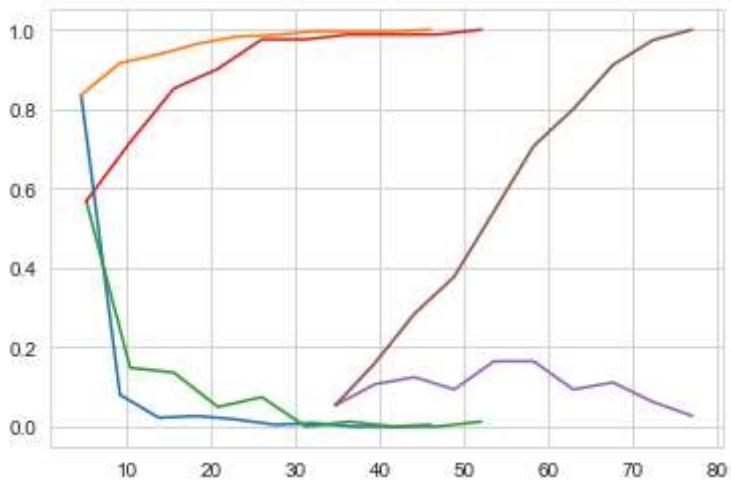
```
[0.83555556 0.08       0.02222222 0.02666667 0.01777778 0.00444444
 0.00888889 0.         0.         0.00444444]
[ 0.   4.6  9.2 13.8 18.4 23.  27.6 32.2 36.8 41.4 46. ]
```



In [19]:
```
CDF is the probability that a random variable, X, will take a value less than or
CDF always lies between 0 and 1
there is a 80% chance of long survival if number of axillary nodes detected are <
Also pdf shows that  nodes increases survival chances also reduces.
CDF will shows in detail view of PDF.
on the nodes feature for survived how many , not survived how many. what percenta
so the above diagram tells that if nodes more than 10 means very less chance of b
```

In [20]:
```
counts, bin_edges = np.histogram(status_1['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(status_2['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(status_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)


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. ]
```



(3.5) Mean, Variance and Std-dev

In [21]:
```
# #Mean, Variance, Std-deviation,
# print("Means:")
# print(np.mean(status_1['nodes']))
# #Mean with an outlier.
# print(np.mean(np.append(status_1['nodes'],50)));
# print(np.mean(status_2["nodes"]))

# print("\nStd-dev:");
# print(np.std(status_1['nodes']))
# print(np.std(status_2['nodes']))

print("Means:")
print("survived")
print(np.mean(status_1['age']))
print(np.mean(status_1['year']))
print(np.mean(status_1['nodes']))

print()

print("not survived")
print(np.mean(status_2['age']))
print(np.mean(status_2['year']))
print(np.mean(status_2['nodes']))

print()

print("std:")
print(np.std(df['age']))
print(np.std(df['year']))
print(np.std(df['nodes']))
```

```
Means:
survived
52.01777777777778
62.86222222222222
2.7911111111111113

not survived
53.67901234567901
62.82716049382716
7.45679012345679

std:
10.78578520363183
3.244090833563246
7.177896092811152
```

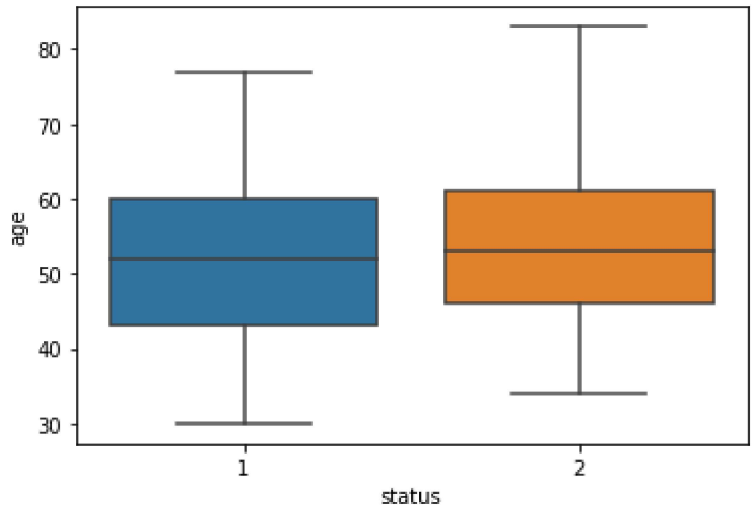
In [22]:
```
print("\nMedians:")
print(np.median(df["age"]))
print(np.median(df["nodes"]))
print(np.median(df["year"]))


print("\nQuantiles:")
print(np.percentile(df["age"],np.arange(0, 100, 25)))
print(np.percentile(df["nodes"],np.arange(0, 100, 25)))
print(np.percentile(df["year"], np.arange(0, 100, 25)))

print("\n90th Percentiles:")
print(np.percentile(df["age"],90))
print(np.percentile(df["nodes"],90))
print(np.percentile(df["year"], 90))

from statsmodels import robust
print ("\nMedian Absolute Deviation")
print(robust.mad(df["age"]))
print(robust.mad(df["nodes"]))
print(robust.mad(df["year"]))
```

```
Medians:
52.0
1.0
63.0

Quantiles:
[30.   44.   52.   60.75]
[0. 0. 1. 4.]
[58.   60.   63.   65.75]

90th Percentiles:
67.0
13.0
67.0

Median Absolute Deviation
11.860817748044816
1.482602218505602
4.447806655516806
```

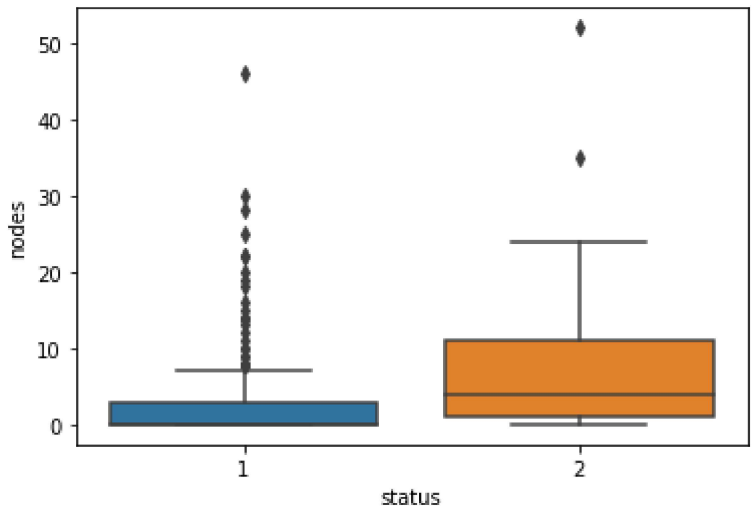
(3.6) Median, Percentile, Quantile, IQR, MAD

(3.7) Box plot and Whiskers

In [ ]:
```
#Box-plot with whiskers: another method of visualizing the  1-D scatter plot more
# The Concept of median, percentile, quantile.
# How to draw the box in the box-plot?
# How to draw whiskers: [no standard way] Could use min and max or use other comp
# IQR like idea.

#NOTE: IN the plot below, a technique call inter-quartile range is used in plotti
#Whiskers in the plot below donot correposnd to the min and max values.

#Box-plot can be visualized as a PDF on the side-ways.

sns.boxplot(x='status',y='age', data=df)
plt.show()
```



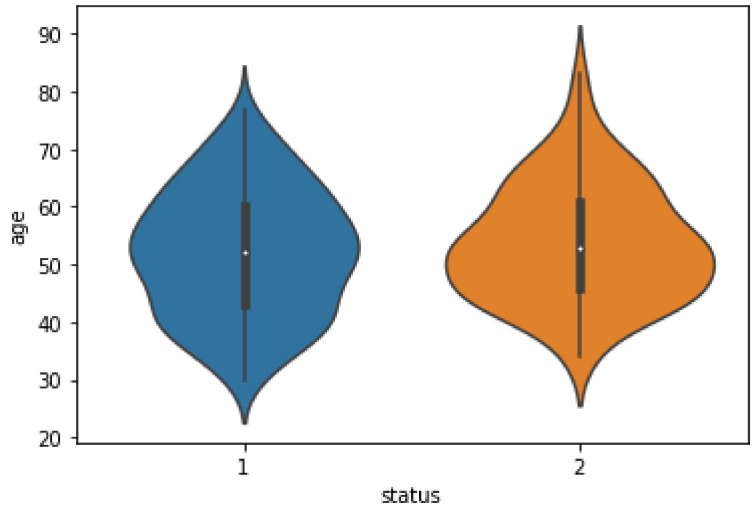
In [ ]:
```
sns.boxplot(x='status',y='nodes', data=df)
plt.show()
```



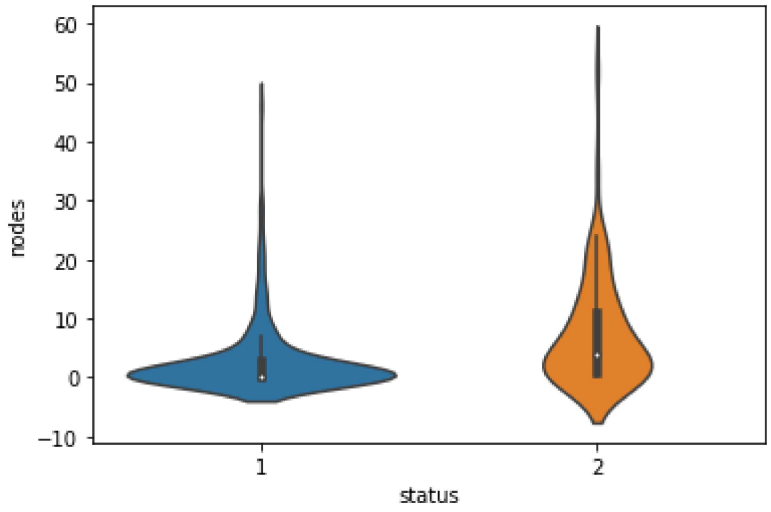
In [ ]:
```
**Observations**
this box plot clearly tells the survived(1) has more spread when nodes less than
orange plot shows not survived if more nodes present .
using this we can see what is median of nodes which has survival status
1 or 2 and all the percentiles(IQR)
```

(3.8) Violin plots

In [ ]:
```
# A violin plot combines the benefits of the previous two plots
#and simplifies them

# Denser regions of the data are fatter, and sparser ones thinner
#in a violin plot

sns.violinplot(x="status", y="age", data=df, size=8)
plt.show()
```



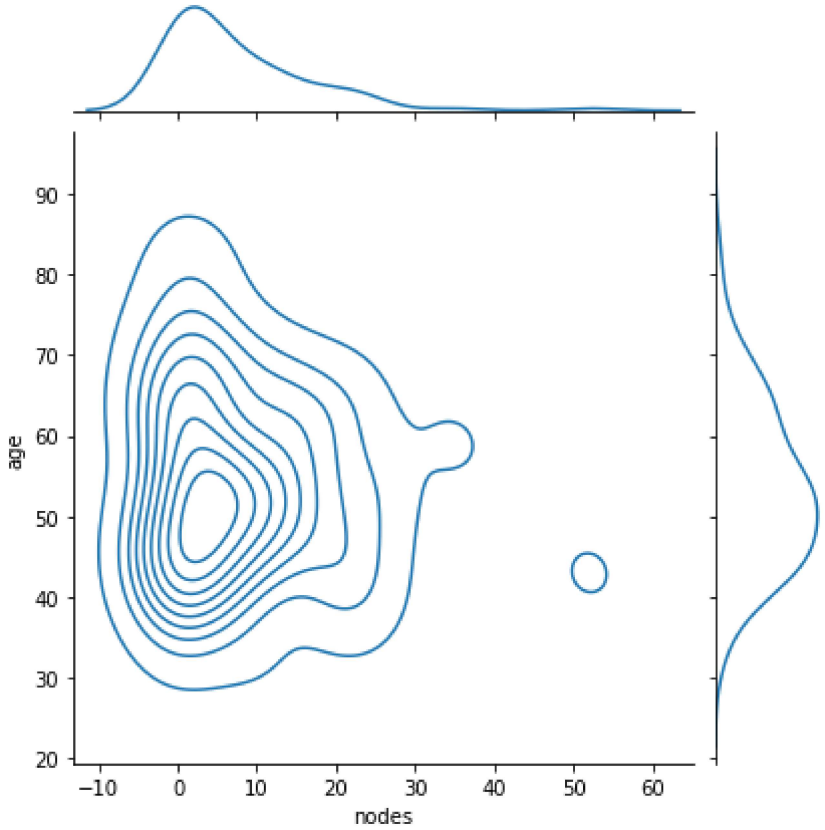
In [ ]:
```
sns.violinplot(x="status", y="nodes", data=df, size=8)
plt.show()
```



In [ ]:
```
**observations
1.more density status 1 on 0-5 nodes
2.another diagram shows that more density when nodes increases.
```

(3.11) Multivariate probability density, contour plot.

In [ ]:
```
#2D Density plot, contors-plot
sns.jointplot(x="nodes", y="age", data=status_2, kind="kde");
plt.show();
```



(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 (https://www.kaggle.com/gilsousa/habermans-survival-data-set))
2. Perform a similar alanlaysis as above on this dataset with the following sections:
3. High level statistics of the dataset: number of points, numer of features, number of classes, data-points per class.
4. Explain our objective.
5. Perform Univaraite analysis(PDF, CDF, Boxplot, Voilin plots) to understand which features are useful towards classification.

6. Perform Bi-variate analysis (scatter plots, pair-plots) to see if combinations of features are useful in classfication.
7. Write your observations in english as crisply and unambigously as possible. Always quantify your results.

In [ ]: