

1. Plotting for Exploratory data analysis (EDA)

(1.1) Haberman Dataset:

- Download Haberman Cancer Survival dataset from Kaggle. You may have to create a Kaggle account to download data. (<https://www.kaggle.com/gilsubas/habermans-survival-dataset>)
- Perform a similar analysis as above on this dataset with the following sections:
 - High level statistics of the dataset: number of points, number of features, number of classes, data-points per class.
 - Explain our objective
 - Perform Univariate analysis(PDF, CDF, Boxplot, Violin 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 classification.
 - Write your observations in english as crisply and unambiguously as possible. Always quantify your results.

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

#Importing numpy,pandas,matplotlib,seaborn packages
#which help us for mathematical operations and plotting the data and exploring the data

In [2]: haberman = pd.read_csv("haberman.csv") #The file in csv(comma separated values) format

In [3]: print(haberman.shape) #It prints how many data-points and features in the dataset
(366, 4)


In [4]: print(haberman.columns) #prints columns
Index(['age', 'year', 'nodes', 'status'], dtype='object')

In [5]: haberman["status"].value_counts()
#prints how many classes there are and prints how many points in each class
#Haberman is imbalanced dataset
#Haberman consist 2 diff-Classes
#(1). status1 :The Person survive more than 5years(5<x)
#(11). status2 : The Person survive less than 5years(5<x)

Out[5]: 1    225
        2     81
        Name: status, dtype: int64
```

(1.2) 2-D Scatter Plot

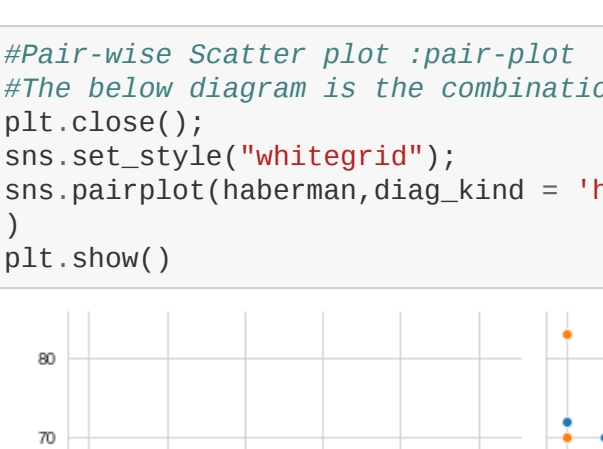
```
In [6]: haberman.plot(kind = "scatter", x = "age", y = "year" )
plt.title("2D Scatter-plot") #make a title for the plot
plt.show() #prints plot by given values
```



Observations :

- From the above plot all the points are overlapping and having the same color.
- We can't classify them.

```
In [7]: #2-D Scatter plot with color coding for each type/class
sns.set_style("whitegrid"); #sns refers to seaborn
sns.FacetGrid(haberman, hue = "status", height = 4) \
    .map(plt.scatter, "age", "year") \
    .add_legend() #adding legends to the plot
plt.title("2D Scatter plot with color coding")
plt.show()
```




Observations :

- From Seaborn we are importing FacetGrid for better classification
- We are using two features like "age" on X-axis and "year" on Y-axis.
- Status = 1(blue) :The person survive more than 5 years ; status = 2(orange) :The person survive less than 5 years
- Status_1 and status_2 points are overlapping.

(1.3) Pair-plot

```
In [8]: #Pair-wise Scatter plot :pair-plot
#The below diagram is the combination of all the plots of given features in the data
plt.close();
sns.set_style("whitegrid");
sns.pairplot(haberman,diag_kind = 'hist', hue='status', vars=['age','year','nodes'],height=4)
plt.show()
```



Observations

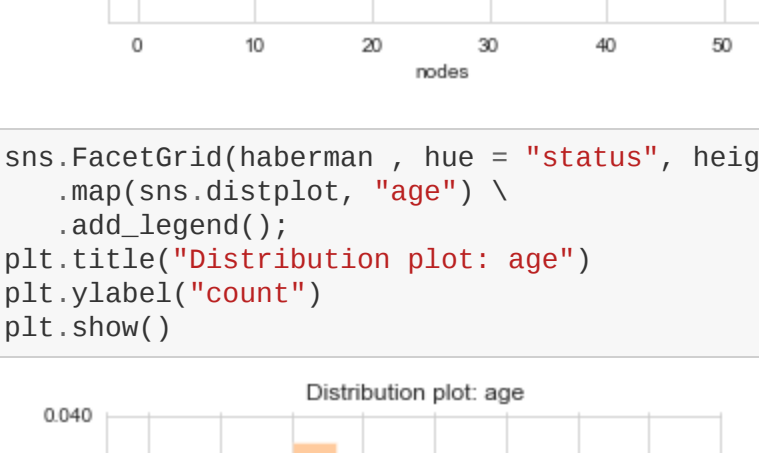
- Age and year are the most useful features to identify various class types.
- From the plot, num 3 and plot, num 7 the points are linearly separable
- plot1, plot5, plot9 are histograms
- We can use features like age and nodes for further data operations

(1.4) Histogram, PDF, CDF

```
In [9]: # 1-D Scatter plot using the Feature "nodes"
# We are using numpy for plotting 1-D scatter plot
import numpy as np
haberman_long_survive = haberman.loc[haberman["status"] == 1]; # Status == 1 belongs to 225/366(Survive more than 5years)
haberman_short_survive = haberman.loc[haberman["status"] == 2]; # status == 2 belongs to 81/366(Survive less than 5years)

plt.plot(haberman_long_survive["nodes"], np.zeros_like(haberman_long_survive["nodes"]), 'o')
plt.plot(haberman_short_survive["nodes"], np.zeros_like(haberman_short_survive["nodes"]), 'o')
plt.title("1D Scatter plot : nodes")
plt.xlabel("nodes")
plt.ylabel("count")
plt.show()

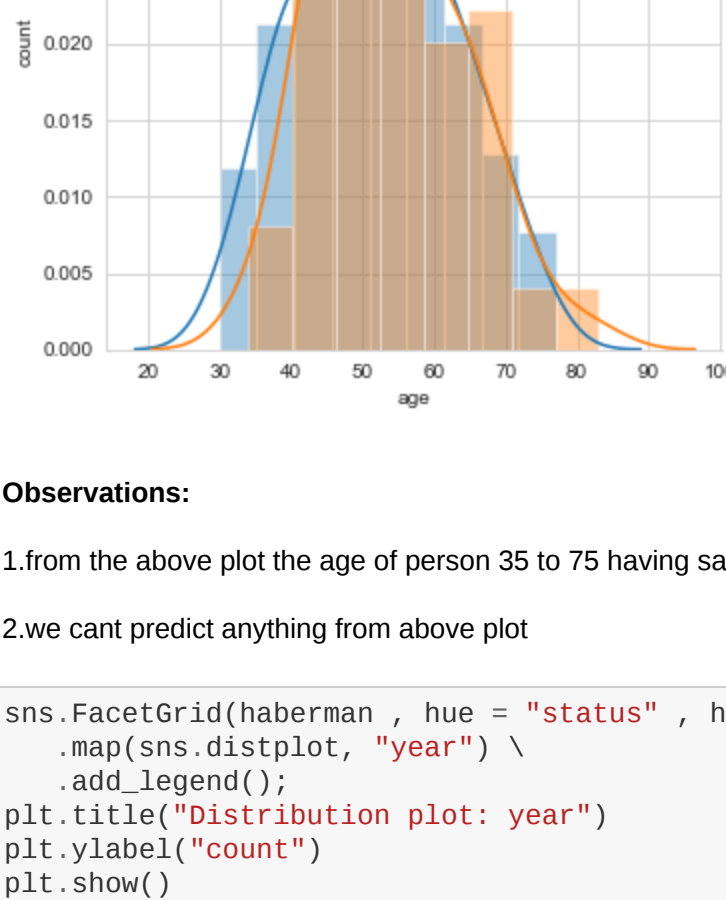
# most of the points are overlapping
# Disadvantage of the plot is we can't make a sense out of it
```



Observations

- From the above Histogram, pdf, cdf diagrams, the person who is having less nodes (Feature) will survive longer compared to other features (Age and Year)
- From the pdf, we can't calculate patient survival status

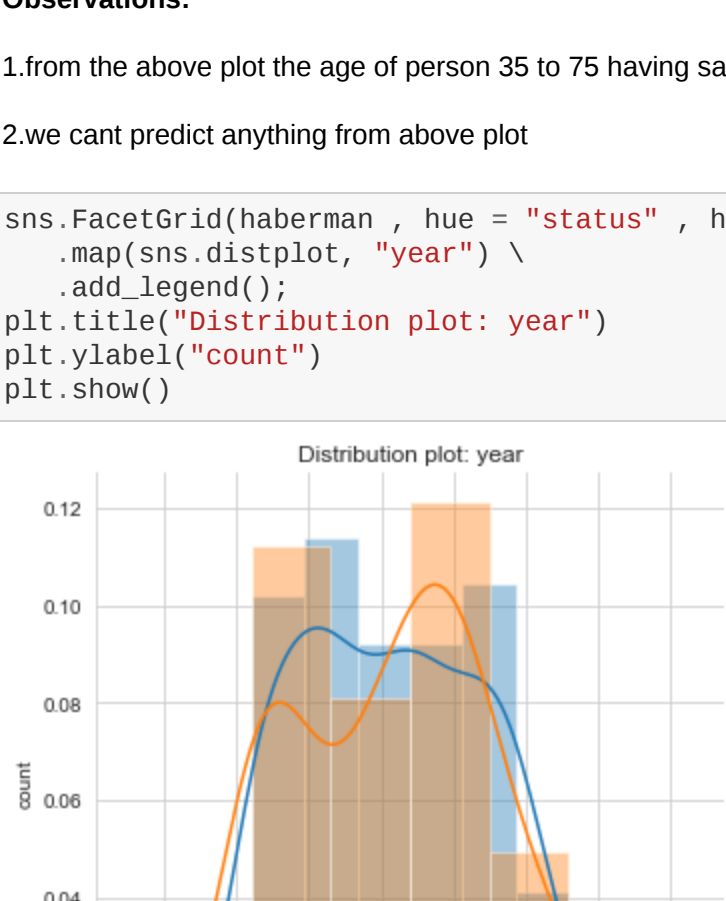
```
In [10]: sns.FacetGrid(haberman, hue = "status", height = 5) \
    .map(sns.distplot, "age") \
    .add_legend();
plt.title("Distribution plot: age")
plt.ylabel("count")
plt.show()
```



Observations:

- From the above plot, the age of person 35 to 75 having the same status for survival and death
- We can't predict anything from the above plot

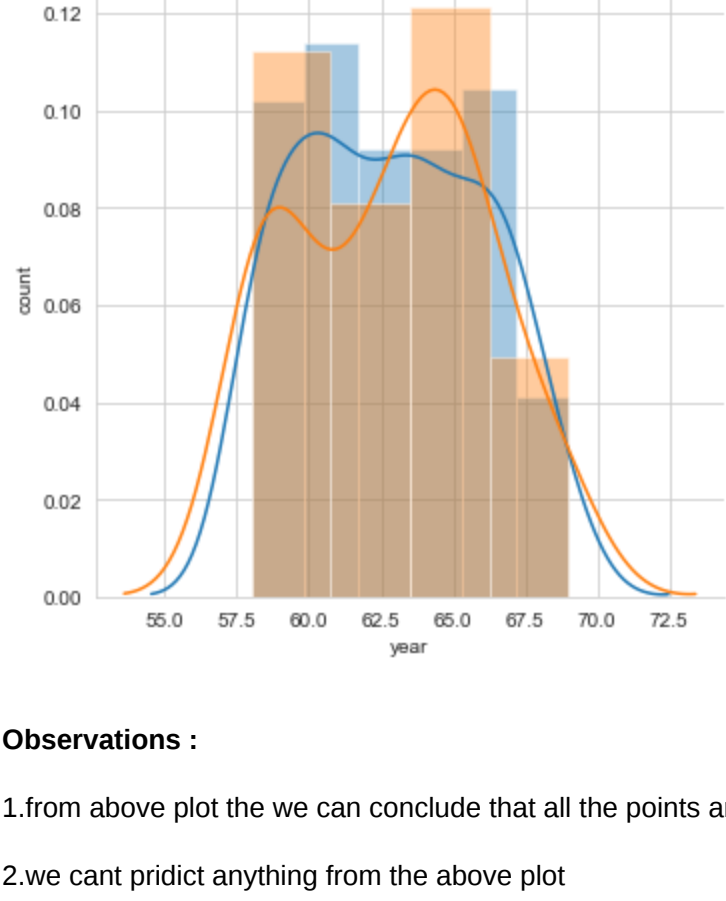
```
In [11]: sns.FacetGrid(haberman, hue = "status", height = 5) \
    .map(sns.distplot, "year") \
    .add_legend();
plt.title("Distribution plot: year")
plt.ylabel("count")
plt.show()
```



Observations :

- From the above plot, we can conclude that all the points are mostly overlapping
- We can't predict anything from the above plot

```
In [12]: sns.FacetGrid(haberman, hue = "status", height = 5) \
    .map(sns.distplot, "nodes") \
    .add_legend();
plt.title("Distribution plot : nodes")
plt.ylabel("count")
plt.show()
```



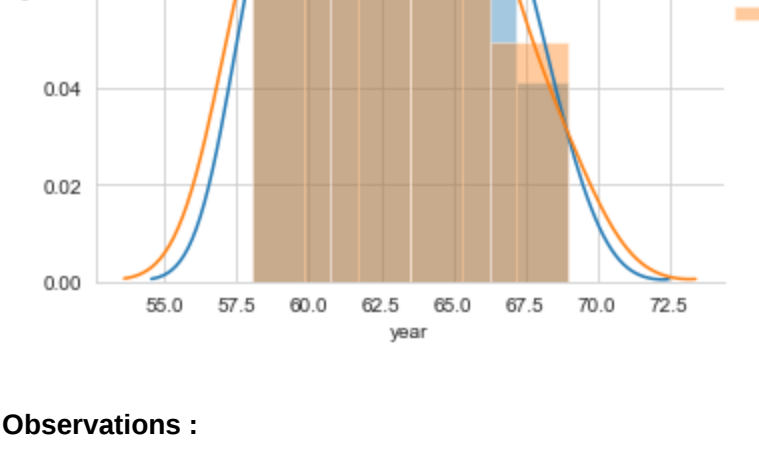
Observations :

- From the above Histogram, pdf, cdf diagrams, the person who is having less nodes (Feature) will survive longer compared to other features (Age and Year)
- From the pdf, we can't calculate patient survival status

```
In [13]: # Plotting the pdf and cdf for Haberman_long_survive data .
# From the pdf we can't find accurate percentage of person who live longer and who live short or
# From the cdf we can calculate survival status of a person

counts, bin_edges = np.histogram(haberman_long_survive["nodes"], bins = 10, density = True)
pdf = counts / (sum(counts)) #Formula for PDF(probability density function)
print(pdf)
print(bin_edges)

cdf = np.cumsum(pdf) #Formula for CDF(cumulative distribution function)
plt.plot(bin_edges[1:], pdf, label = "pdf")
plt.plot(bin_edges[1:], cdf, label = "cdf")
plt.title("pdf and cdf for Haberman_long_survive")
plt.xlabel("nodes")
plt.ylabel("probabilities")
plt.legend()
plt.show()
```



Observation :

- Cdf for haberman_long_survive
- Pdf formula : pdf = counts / (sum(counts))
- Cdf formula : cdf = np.cumsum(pdf) np = numpy, cumsum = Cumulative sum

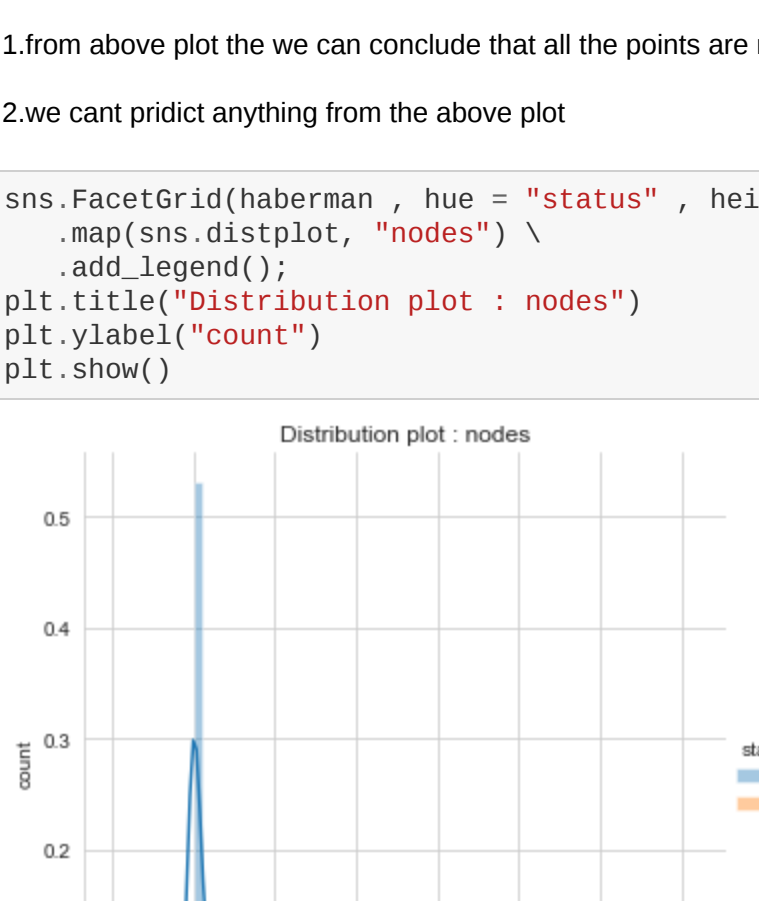
```
In [14]: #We are plotting the both Haberman_long_survive and Haberman_short_survive
#We can easily understand the survival status of the persons

#prints pdf and cdf for Haberman_long_survive
counts, bin_edges = np.histogram(haberman_long_survive["nodes"], bins = 10, density = True)
pdf = counts / (sum(counts))
print(pdf)
print(bin_edges)

cdf = np.cumsum(pdf) #using numpy we can calculate cumulative sum of pdf which equals to cdf value
plt.plot(bin_edges[1:], pdf, label = "pdf_1") #prints pdf's bin_edges from 1
plt.plot(bin_edges[1:], cdf, label = "cdf_1") #prints cdf's bin_edges from 1

#prints pdf and cdf for Haberman_short_survive
counts, bin_edges = np.histogram(haberman_short_survive["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_2")
plt.plot(bin_edges[1:], cdf, label = "cdf_2")
plt.title("pdf and cdf for both haberman_long_survive , haberman_short_survive") #prints title, x_label, y_label
plt.xlabel("nodes")
plt.ylabel("probabilities")
plt.legend()
plt.show()
```



Observations :

- Cdf for both haberman_long_survive and haberman_short_survive
- Blue(pdf) and Orange(cdf) for Haberman_long_survive
- Green(pdf) and Red(cdf) for haberman_short_survive

(1.5) Mean, Variance and Std-dev

```
In [15]: #Finding mean and std_dev for both long_survive and short_survive

print("means:")
print(np.mean(haberman_long_survive["nodes"]))
print(np.mean(np.append(haberman_long_survive["nodes"], 50))) #mean with an outlier
print(np.mean(haberman_short_survive["nodes"]))

print("\nstd_dev :")
print(np.std(haberman_long_survive["nodes"]))
print(np.std(haberman_short_survive["nodes"]))

means :
2.7911111111111113
3.6
7.45679012345679

std_dev :
6.857258449412131
9.128776876761632
```

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

```
In [16]: print("Medians :") #prints median for both haberman_long_survive and haberman_short_survive
6
print(np.median(haberman_long_survive["nodes"]))
print(np.median(np.append(haberman_long_survive["nodes"], 50))) #median with an outlier
print(np.median(haberman_short_survive["nodes"]))

print("\nQuantiles :") #prints quantile for both haberman_long_survive and haberman_short_survive
print(np.percentile(haberman_long_survive["nodes"], np.arange(0,100,25)))
print(np.percentile(haberman_short_survive["nodes"], np.arange(0,100,25)))

print("\nPercentile :") #prints Percentile for both haberman_long_survive and haberman_short_survive
print(np.percentile(haberman_long_survive["nodes"], 90))
print(np.percentile(haberman_short_survive["nodes"], 90))

from statsmodels import robust #importing robust model from statsmodel
print("\nMedian absolute deviation :") #prints Median absolute deviation for both haberman_long_survive and haberman_short_survive
print(robust.mad(haberman_long_survive["nodes"]))
print(robust.mad(haberman_short_survive["nodes"]))

Medians :
0.0
0.0
0.0
4.0

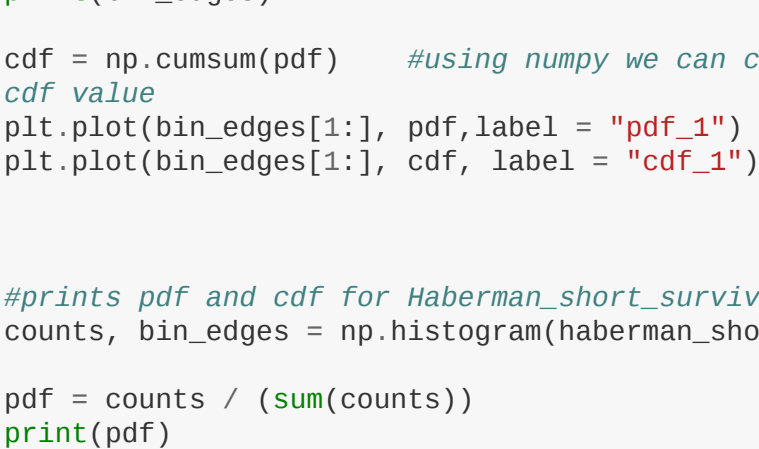
Quantiles :
[0. 0. 0. 3.]
[0. 1. 4. 11.]

Percentile :
8.0
20.0

Median absolute deviation :
0.0
5.930408874022408
```

(1.7) Box plot and Whiskers

```
In [17]: #Box-plots can be represented all the data between 25th percentile and 75th percentile
#Box-plot is another method of visualizing the 1-D scatter plot more intuitively
# How to draw whiskers: [no standard way] Could use min and max or use other complex statistical techniques.
# IQR like ideas
sns.boxplot(x = "status", y = "nodes", data = haberman) # "sns" stands for seaborn
plt.legend()
plt.title("Box plot(status, nodes)")
plt.show()
```

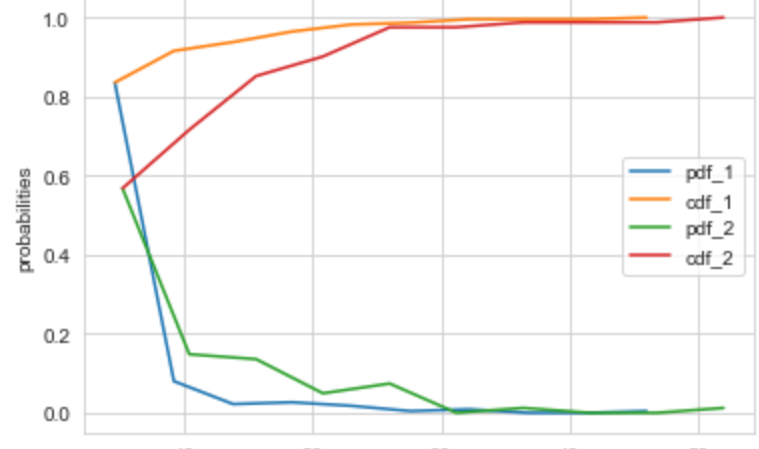


Observations

- From the above plot, we can conclude that status = 1 is Haberman_long_survive having values from 0 to 8 and haberman_short_survive having values from 2 to 24
- Both values are overlapping in the above plot
- Long survival 25% of points is mostly equal to 50% of short survival values
- Short survival having 50% values same as 75% of long survival

(1.8) Violin plots

```
In [18]: #violin plots is combination of both Box-plots and Pdf
sns.jointplot(x = "status", y = "nodes", data = haberman)
plt.title("Violin plot(status, nodes)")
plt.show()
```



Observations

- From the above plot, we can understand the long survival plot is mostly overlapped with the short survival
- Long survival having density value from 0-6
- Short survival having density value from 2-24

(1.9) Multivariate probability density, contour plot.

```
In [19]: #contour plot is like density plots the more points placed at a point the data looks like a peak
#having less points at a place shows light color having more points looks like thicker color
#the plot is kde
sns.jointplot(x = "age", y = "nodes", data = haberman_long_survive, kind = "kde")
plt.show()
```



Observations

- From the above plot, we can understand that the more thick consists of more numbers of points lying in that region
- Less number of points shows lighter shade color
- No data points in more areas get darker color; we can visualize hill-like structure, the top of the hill having high density and slope of a hill have less density

In [] :

In [] :