

## ▼ 3. Plotting for Exploratory data analysis (EDA)

Double-click (or enter) to edit

### ▼ (3.1) Basic Terminology

- What is EDA?
- Data-point/[vector/Observation](#)
- Data-set.
- Feature/[Variable/Input-variable/Dependent-varibale](#)
- Label/[Independent-variable/Output-variable/Class/Class-label/Response](#) label
- Vector: 2-D, 3-D, 4-D,... n-D

Q. What is a 1-D vector: Scalar

### ▼ Iris Flower dataset

Toy Dataset: Iris Dataset: [https://en.wikipedia.org/wiki/Iris\\_flower\\_data\\_set](https://en.wikipedia.org/wiki/Iris_flower_data_set)

- A simple dataset to learn the basics.
- 3 flowers of Iris species. [see images on wikipedia link above]
- 1936 by Ronald Fisher.
- Petal and Sepal: [http://terpconnect.umd.edu/~petersd/666/html/iris\\_with\\_labels.jpg](http://terpconnect.umd.edu/~petersd/666/html/iris_with_labels.jpg)
- Objective: Classify a new flower as belonging to one of the 3 classes given the 4 features.
- Importance of domain knowledge.
- Why use petal and sepal dimensions as features?
- Why do we not use 'color' as a feature?

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
```

```
'''downlaod iris.csv from https://raw.githubusercontent.com/uiuc-cse/data-fa14/gh-pages/da
#Load Iris.csv into a pandas DataFrame.
iris = pd.read_csv("iris.csv")
```

```
# (Q) how many data-points and features?
```

```
print (iris.shape)
```

```
(150, 5)
```

```
#(Q) What are the column names in our dataset?
```

```
print (iris.columns)
```

```
Index(['sepal_length', 'sepal_width', 'petal_length', 'petal_width',  
      'species'],  
      dtype='object')
```

```
#(Q) How many data points for each class are present?
```

```
#(or) How many flowers for each species are present?
```

```
iris["species"].value_counts()
```

```
# balanced-dataset vs imbalanced datasets
```

```
#Iris is a balanced dataset as the number of data points for every class is 50.
```

```
virginica      50  
setosa         50  
versicolor    50  
Name: species, dtype: int64
```

## ▼ (3.2) 2-D Scatter Plot

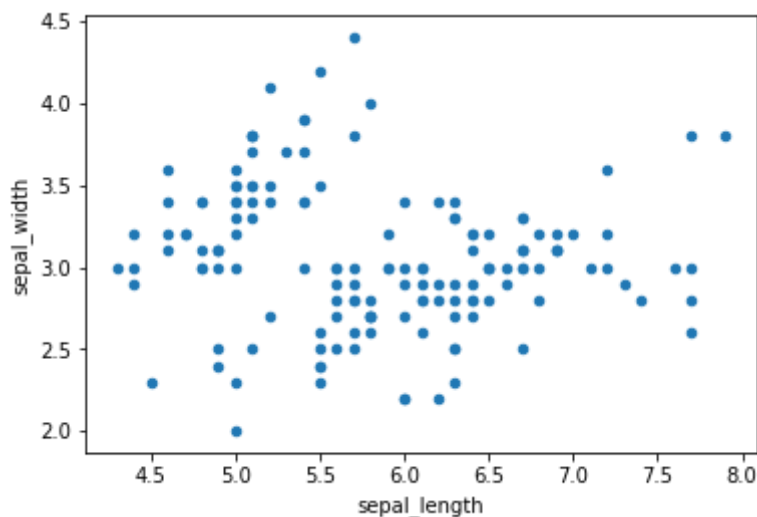
```
#2-D scatter plot:
```

```
#ALWAYS understand the axis: labels and scale.
```

```
iris.plot(kind='scatter', x='sepal_length', y='sepal_width') ;  
plt.show()
```

```
#cannot make much sense out it.
```

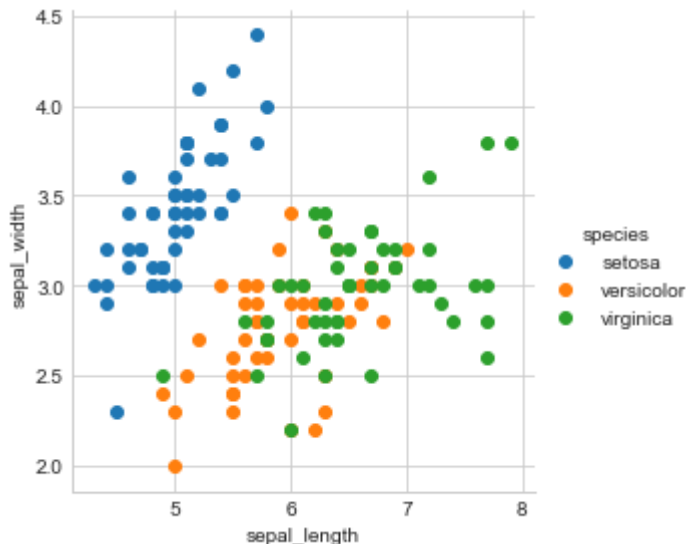
```
#What if we color the points by thier class-label/flower-type.
```



```
# 2-D Scatter plot with color-coding for each flower type/class.
```

```
# Here 'sns' corresponds to seaborn.
sns.set_style("whitegrid");
sns.FacetGrid(iris, hue="species", size=4) \
    .map(plt.scatter, "sepal_length", "sepal_width") \
    .add_legend();
plt.show();

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



### Observation(s):

1. Using sepal\_length and sepal\_width features, we can distinguish Setosa flowers from others.
2. Separating Versicolor from Virginica is much harder as they have considerable overlap.

## 3D Scatter plot

<https://plot.ly/pandas/3d-scatter-plots/>

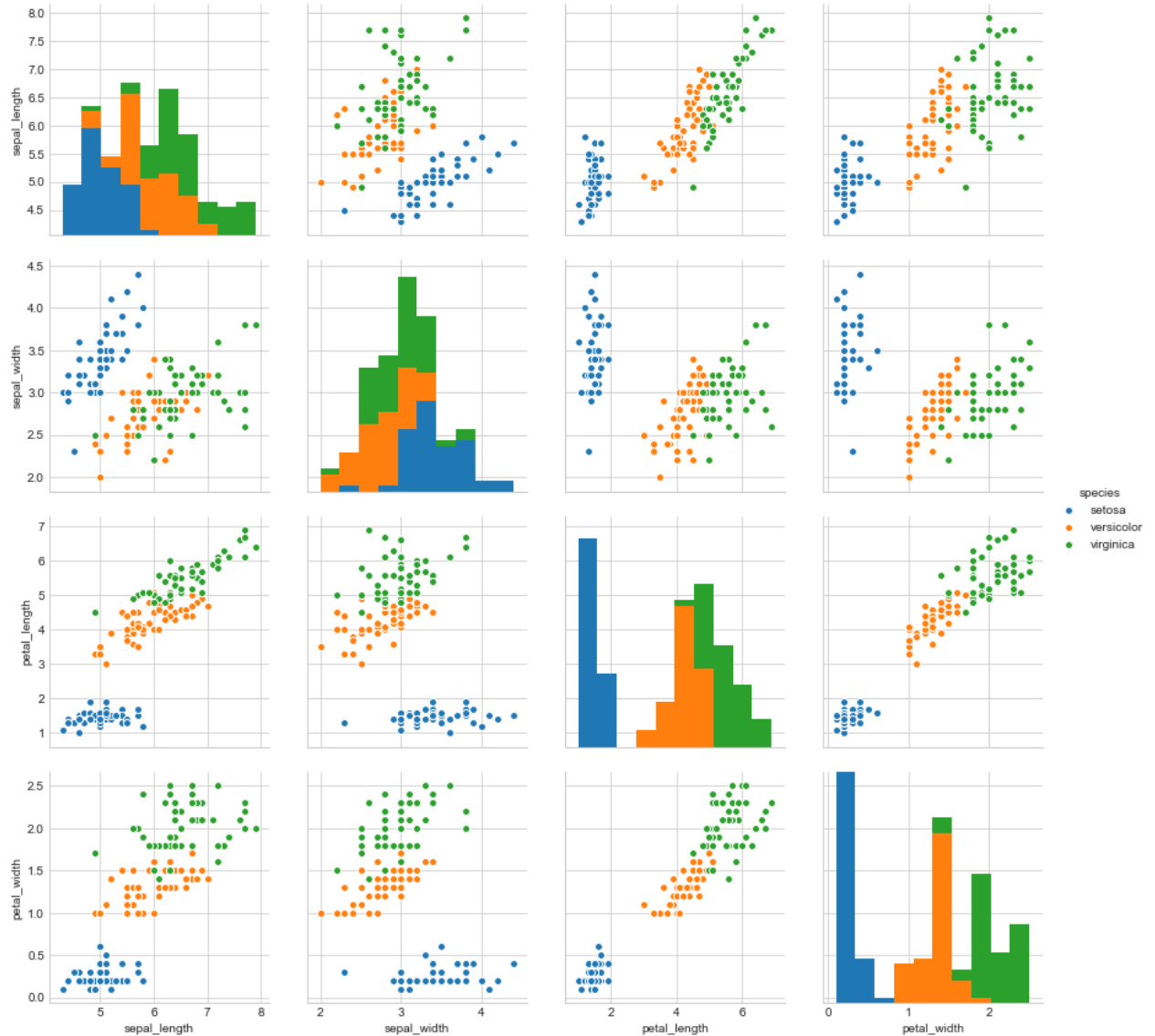
Needs a lot to mouse interaction to interpret data.

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

## ▼ (3.3) Pair-plot

```
# 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(iris, hue="species", size=3);
plt.show()
# NOTE: the diagonol elements are PDFs for each feature. PDFs are expalined below.
```



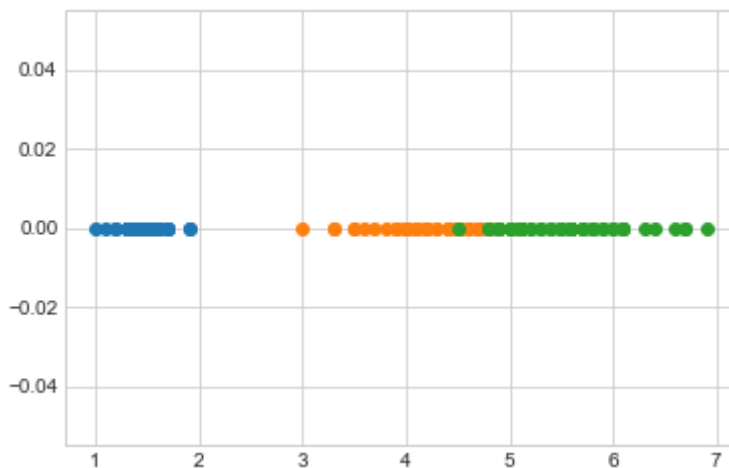
## Observations

1. petal\_length and petal\_width are the most useful features to identify various flower types.
2. While Setosa can be easily identified (linearly separable), Virginica and Versicolor have some overlap (almost linearly separable).
3. We can find "lines" and "if-else" conditions to build a simple model to classify the flower types.

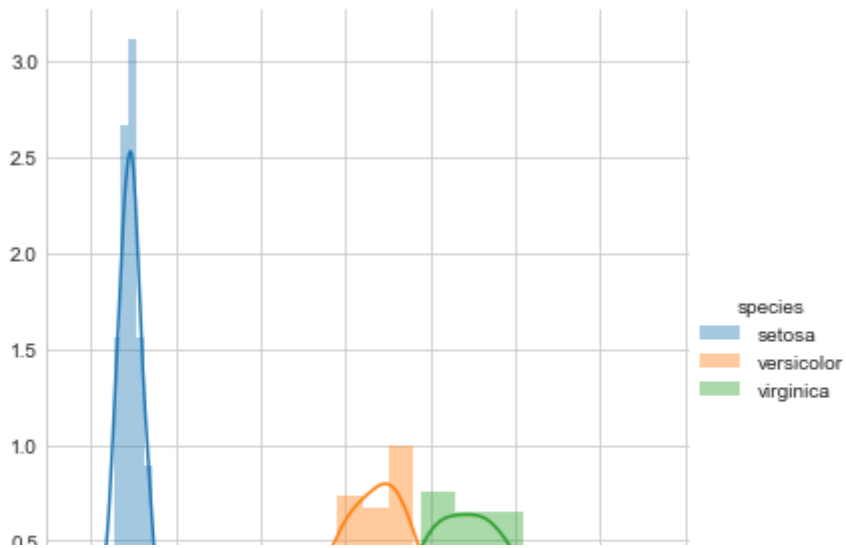
## ▼ (3.4) Histogram, PDF, CDF

```
# What about 1-D scatter plot using just one feature?
#1-D scatter plot of petal-length
import numpy as np
iris_setosa = iris.loc[iris["species"] == "setosa"];
iris_virginica = iris.loc[iris["species"] == "virginica"];
iris_versicolor = iris.loc[iris["species"] == "versicolor"];
#print(iris_setosa["petal_length"])
plt.plot(iris_setosa["petal_length"], np.zeros_like(iris_setosa['petal_length']), 'o')
plt.plot(iris_versicolor["petal_length"], np.zeros_like(iris_versicolor['petal_length']), 'o')
plt.plot(iris_virginica["petal_length"], np.zeros_like(iris_virginica['petal_length']), 'o')

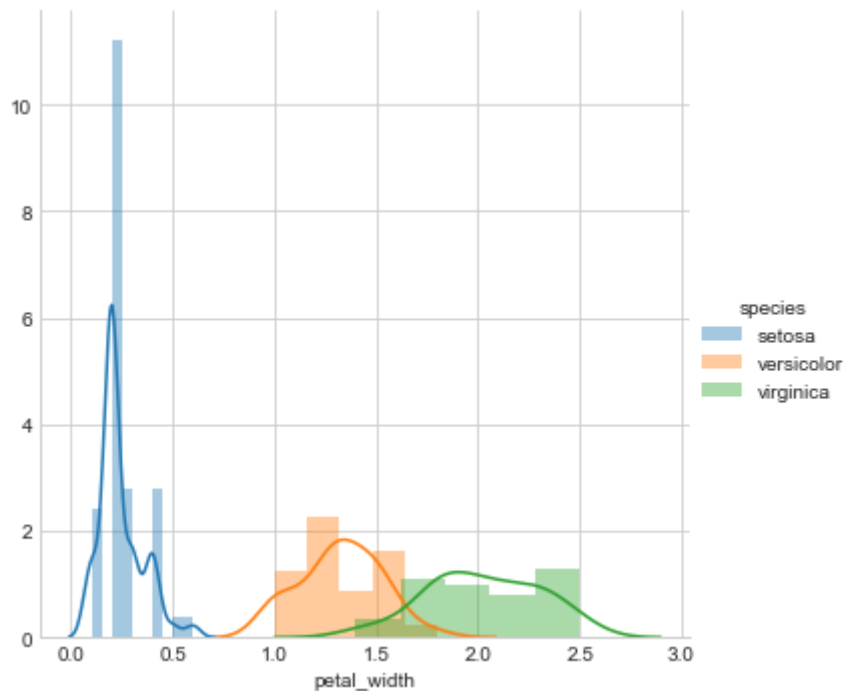
plt.show()
#Disadvantages of 1-D scatter plot: Very hard to make sense as points
#are overlapping a lot.
#Are there better ways of visualizing 1-D scatter plots?
```



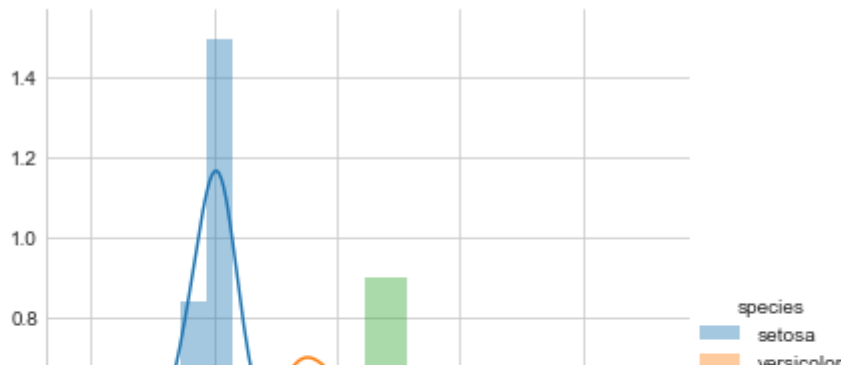
```
sns.FacetGrid(iris, hue="species", size=5) \
    .map(sns.distplot, "petal_length") \
    .add_legend();
plt.show();
```



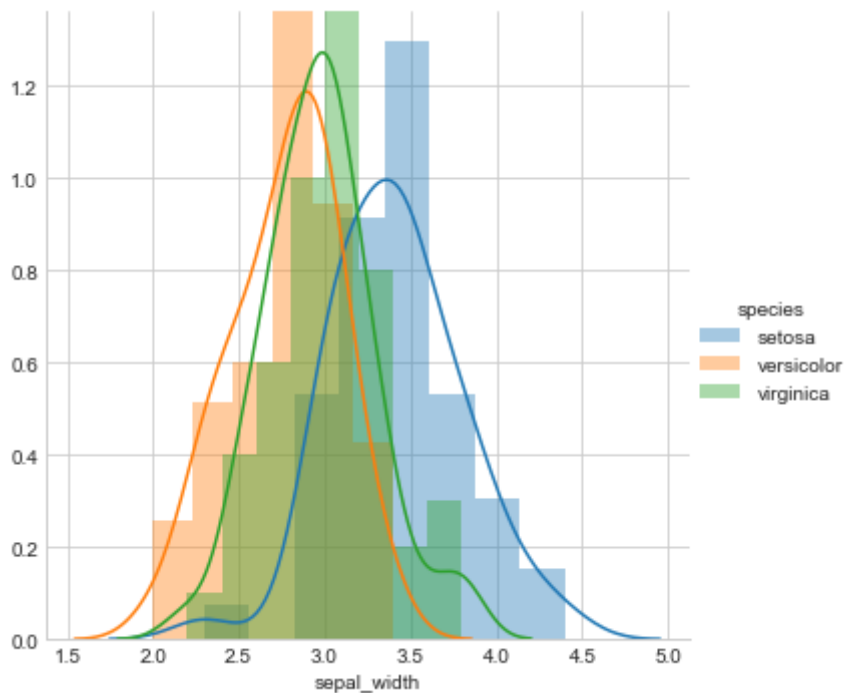
```
sns.FacetGrid(iris, hue="species", size=5) \
    .map(sns.distplot, "petal_width") \
    .add_legend();
plt.show();
```



```
sns.FacetGrid(iris, hue="species", size=5) \
    .map(sns.distplot, "sepal_length") \
    .add_legend();
plt.show();
```



```
sns.FacetGrid(iris, hue="species", size=5) \
    .map(sns.distplot, "sepal_width") \
    .add_legend();
plt.show();
```



```
# Histograms and Probability Density Functions (PDF) using KDE
# How to compute PDFs using counts/frequencies of data points in each window.
# How window width effects the PDF plot.
```

```
# Interpreting a PDF:
## why is it called a density plot?
## Why is it called a probability plot?
## for each value of petal_length, what does the value on y-axis mean?
# Notice that we can write a simple if..else condition as if(petal_length) < 2.5 then flow
# Using just one feature, we can build a simple "model" suing if..else... statements.
```

```
# Disadv of PDF: Can we say what percentage of versicolor points have a petal_length of 1e
```

```
# Do some of these plots look like a bell-curve you studied in under-grad?
# Gaussian/Normal distribution.
# What is "normal" about normal distribution?
# e.g: Hieghts of male students in a class.
# One of the most frequent distributions in nature.
```

```

# Need for Cumulative Distribution Function (CDF)
# We can visually see what percentage of versicolor flowers have a
# petal_length of less than 5?
# How to construct a CDF?
# How to read a CDF?

#Plot CDF of petal_length

counts, bin_edges = np.histogram(iris_setosa['petal_length'], 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(iris_setosa['petal_length'], bins=20,
                                  density = True)

pdf = counts/(sum(counts))
plt.plot(bin_edges[1:],pdf);

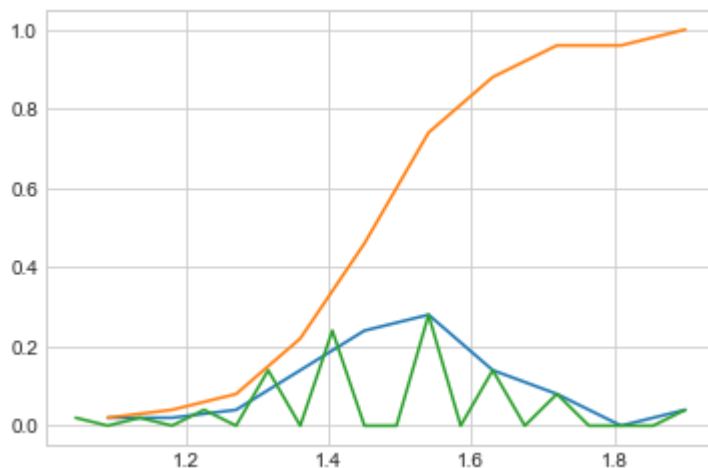
plt.show();

```

```

[ 0.02  0.02  0.04  0.14  0.24  0.28  0.14  0.08  0.    0.04]
[ 1.    1.09  1.18  1.27  1.36  1.45  1.54  1.63  1.72  1.81  1.9 ]

```



```

# Need for Cumulative Distribution Function (CDF)
# We can visually see what percentage of versicolor flowers have a
# petal_length of less than 1.6?
# How to construct a CDF?
# How to read a CDF?

#Plot CDF of petal_length

```

```

counts, bin_edges = np.histogram(iris_setosa['petal_length'], bins=10,
                                  density = True)

```

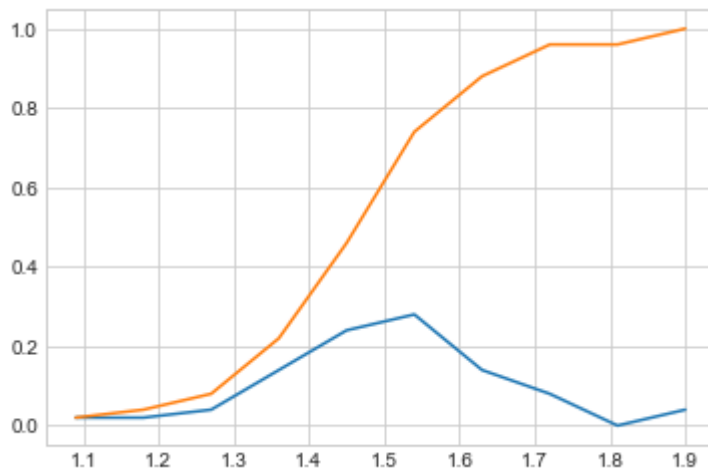


```
pdf = counts/(sum(counts))
print(pdf);
print(bin_edges)
```

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

```
plt.show();
```

```
[ 0.02  0.02  0.04  0.14  0.24  0.28  0.14  0.08  0.    0.04]
[ 1.    1.09  1.18  1.27  1.36  1.45  1.54  1.63  1.72  1.81  1.9 ]
```



```
# Plots of CDF of petal_length for various types of flowers.
```

```
# Misclassification error if you use petal_length only.
```

```
counts, bin_edges = np.histogram(iris_setosa['petal_length'], 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)
```

```
# virginica
```

```
counts, bin_edges = np.histogram(iris_virginica['petal_length'], bins=10,
                                density = True)
```

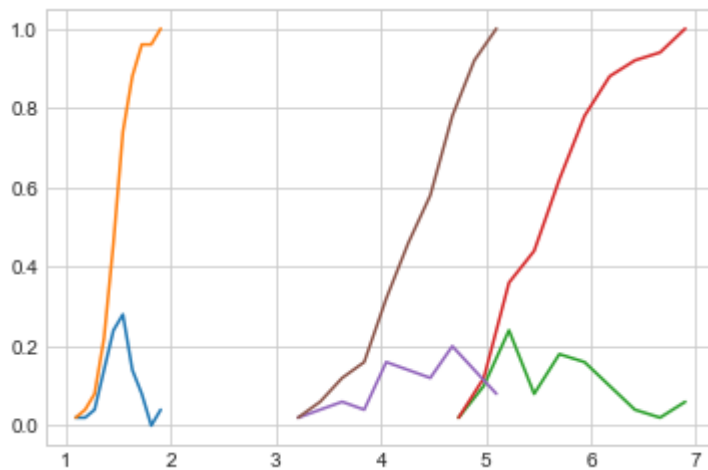
```
pdf = counts/(sum(counts))
print(pdf);
print(bin_edges)
cdf = np.cumsum(pdf)
plt.plot(bin_edges[1:],pdf)
plt.plot(bin_edges[1:], cdf)
```

```
#versicolor
counts, bin_edges = np.histogram(iris_versicolor['petal_length'], 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();
```

```
[ 0.02  0.02  0.04  0.14  0.24  0.28  0.14  0.08  0.    0.04]
[ 1.    1.09  1.18  1.27  1.36  1.45  1.54  1.63  1.72  1.81  1.9 ]
[ 0.02  0.1   0.24  0.08  0.18  0.16  0.1   0.04  0.02  0.06]
[ 4.5   4.74  4.98  5.22  5.46  5.7   5.94  6.18  6.42  6.66  6.9 ]
[ 0.02  0.04  0.06  0.04  0.16  0.14  0.12  0.2   0.14  0.08]
[ 3.    3.21  3.42  3.63  3.84  4.05  4.26  4.47  4.68  4.89  5.1 ]
```



## ▼ (3.5) Mean, Variance and Std-dev

```
#Mean, Variance, Std-deviation,
print("Means:")
print(np.mean(iris_setosa["petal_length"]))
#Mean with an outlier.
print(np.mean(np.append(iris_setosa["petal_length"],50)));
print(np.mean(iris_virginica["petal_length"]))
print(np.mean(iris_versicolor["petal_length"]))

print("\nStd-dev:");
print(np.std(iris_setosa["petal_length"]))
print(np.std(iris_virginica["petal_length"]))
print(np.std(iris_versicolor["petal_length"]))
```

Means :

```
1.464
2.41568627451
5.552
4.26
```

```
Std-dev:
0.171767284429
0.546347874527
0.465188133985
```

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

```
#Median, Quantiles, Percentiles, IQR.
print("\nMedians:")
print(np.median(iris_setosa["petal_length"]))
#Median with an outlier
print(np.median(np.append(iris_setosa["petal_length"],50)));
print(np.median(iris_virginica["petal_length"]))
print(np.median(iris_versicolor["petal_length"]))

print("\nQuantiles:")
print(np.percentile(iris_setosa["petal_length"],np.arange(0, 100, 25)))
print(np.percentile(iris_virginica["petal_length"],np.arange(0, 100, 25)))
print(np.percentile(iris_versicolor["petal_length"], np.arange(0, 100, 25)))

print("\n90th Percentiles:")
print(np.percentile(iris_setosa["petal_length"],90))
print(np.percentile(iris_virginica["petal_length"],90))
print(np.percentile(iris_versicolor["petal_length"], 90))

from statsmodels import robust
print ("\nMedian Absolute Deviation")
print(robust.mad(iris_setosa["petal_length"]))
print(robust.mad(iris_virginica["petal_length"]))
print(robust.mad(iris_versicolor["petal_length"]))
```

Medians:

```
1.5
1.5
5.55
4.35
```

Quantiles:

```
[ 1.    1.4    1.5    1.575]
[ 4.5    5.1    5.55   5.875]
[ 3.     4.     4.35   4.6 ]
```

90th Percentiles:

```
1.7
6.31
4.8
```

```
Median Absolute Deviation
0.148260221851
0.667170998328
0.518910776477
```

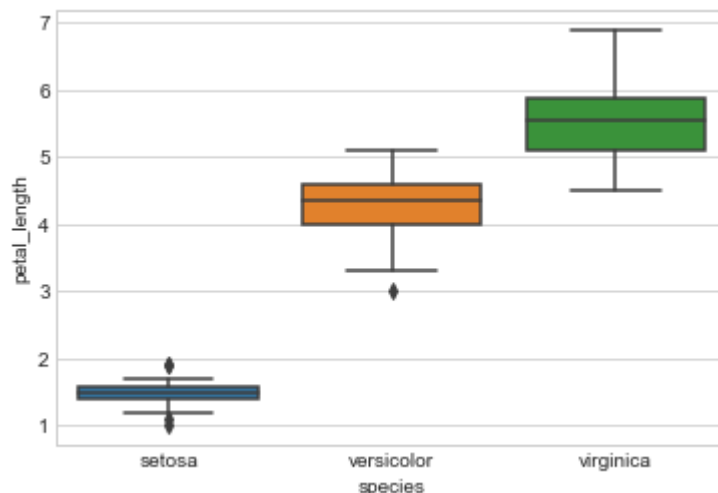
## ▼ (3.7) Box plot and Whiskers

```
#Box-plot with whiskers: another method of visualizing the 1-D scatter plot more intuitiv
# 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 complex stati
# IQR like idea.
```

```
#NOTE: IN the plot below, a technique call inter-quartile range is used in plotting the wh
#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='species',y='petal_length', data=iris)
plt.show()
```

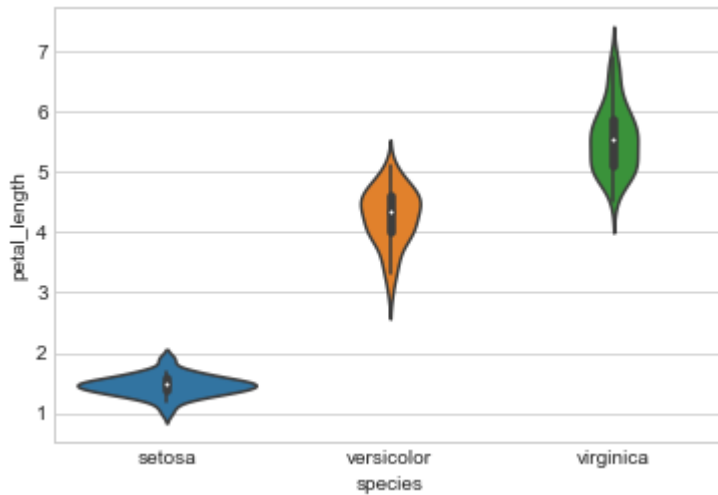


## ▼ (3.8) Violin plots

```
# 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="species", y="petal_length", data=iris, size=8)
plt.show()
```



### ▼ (3.9) Summarizing plots in english

- Explain your findings/conclusions in plain english
- Never forget your objective (the problem you are solving) . Perform all of your EDA aligned with your objectives.

### (3.10) Univariate, bivariate and multivariate analysis.

# Def: Univariate, Bivariate and Multivariate analysis.

### ▼ (3.11) Multivariate probability density, contour plot.

```
#2D Density plot, contours-plot
sns.jointplot(x="petal_length", y="petal_width", data=iris_setosa, kind="kde");
plt.show();
```

### ▼ (3.12) Exercise:

1. Download Haberman Cancer Survival dataset from Kaggle. You may have to create a Kaggle account to download data. (<https://www.kaggle.com/gilsousa/habermans-survival-data-set>)
2. 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.

## Exercise Solutions:

1) High level statistics of the dataset: number of points, number of features, number of classes, data-points per class.

### 1.1) Loading Haberman Data

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np

haberman = pd.read_csv("haberman.csv")
```

### 1.2) Finding number of points and number of features

We can find the number of points and number of features by using the word shape

```
print(haberman.shape)

(306, 4)
```

### 1.3) Finding number of classes using Column

```
print(haberman.columns)

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

### 1.4) Data-points per class

```
haberman["status"].value_counts()

1    225
2     81
```

Name: status, dtype: int64

## 2) Objectives:

1. We have to perform data analysis on Haberman Datasets.
2. From this datasets we come to know that we have for classes that is already shown in number of classes.
3. Throuh the data points we come to know that there are 225 people alive and 81 people died.
4. Now we are going to analysis the data through the various kind of data analysis.

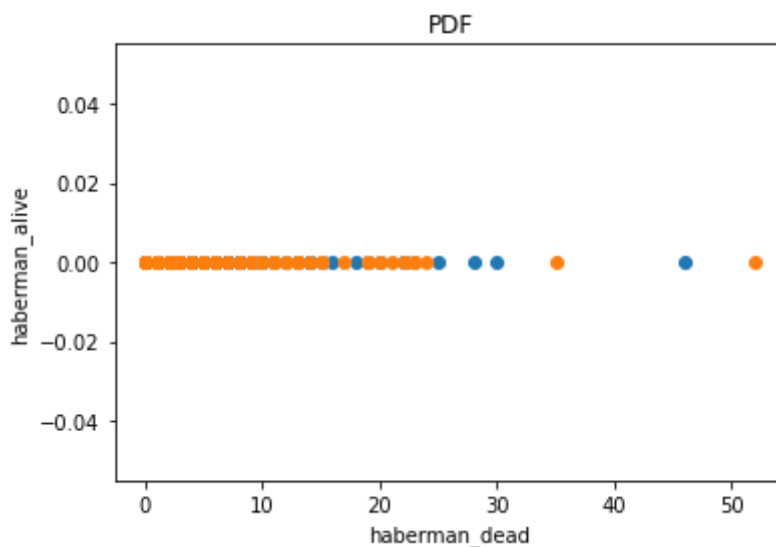
## 3) Perform Univarait analysis(PDF, CDF, Boxplot, Voilin plots)

### 3.1) PDF and CDF

```
import numpy as np
haberman_alive = haberman.loc[haberman["status"] == 1];
haberman_dead = haberman.loc[haberman["status"] == 2];

plt.plot(haberman_alive["nodes"], np.zeros_like(haberman_alive['nodes']), 'o')
plt.plot(haberman_dead["nodes"], np.zeros_like(haberman_dead['nodes']), 'o')
plt.ylabel("haberman_alive")
plt.xlabel("haberman_dead")
plt.title("PDF")

plt.show()
```

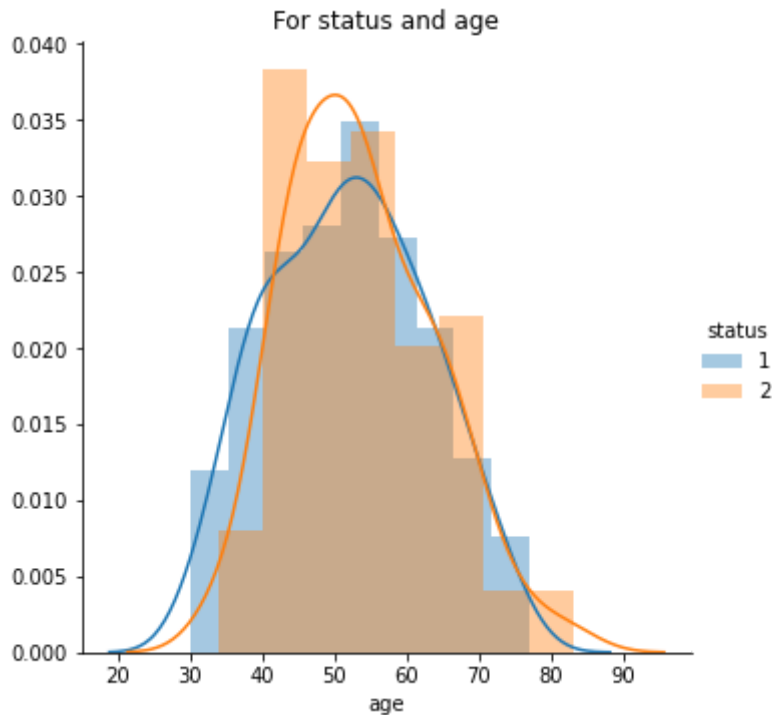


```
import warnings
```

```
warnings.filterwarnings("ignore")
```

### 3.2) For status and age

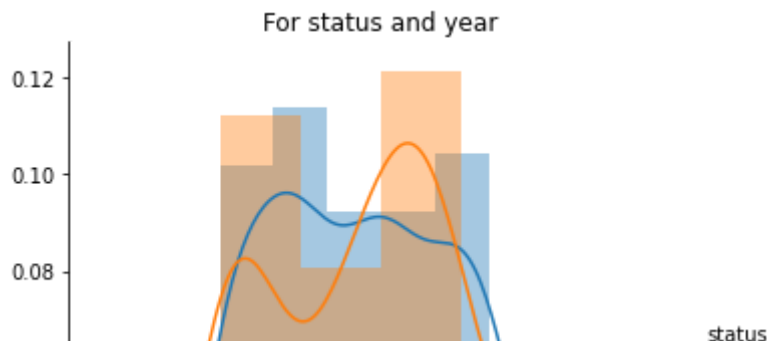
```
sns.FacetGrid(haberman, hue="status", size=5) \
    .map(sns.distplot, "age") \
    .add_legend();
plt.title("For status and age")
plt.show();
```



### 3.3) For status and year

```
sns.FacetGrid(haberman, hue="status", size=5) \
    .map(sns.distplot, "year") \
    .add_legend();
plt.title("For status and year")
plt.show();
```

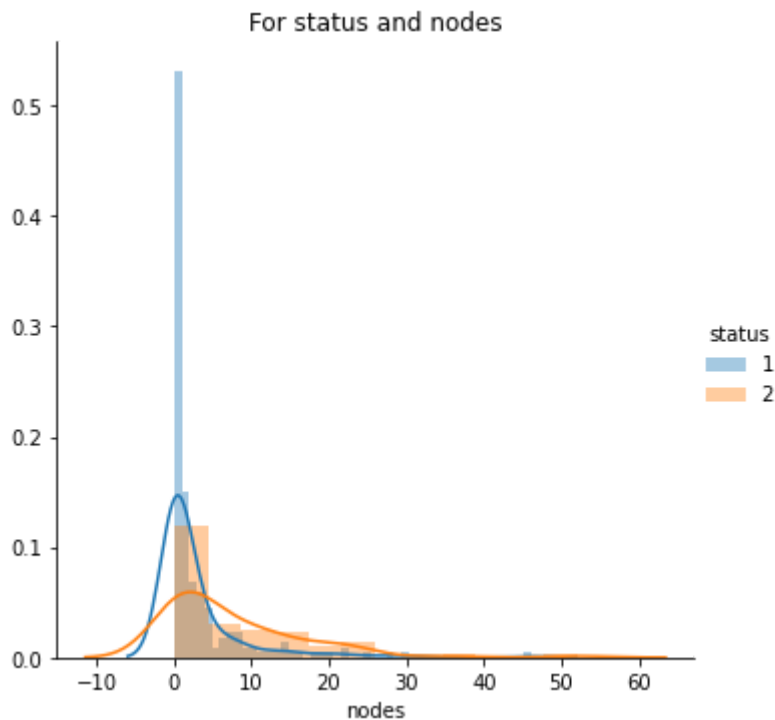




### 3.4) For status and nodes



```
sns.FacetGrid(haberman, hue="status", size=5) \
    .map(sns.distplot, "nodes") \
    .add_legend();
plt.title("For status and nodes")
plt.show();
```



### 3.5) Plot CDF of age

```
counts, bin_edges = np.histogram(haberman_alive['age'], bins=10,
                                  density = True)

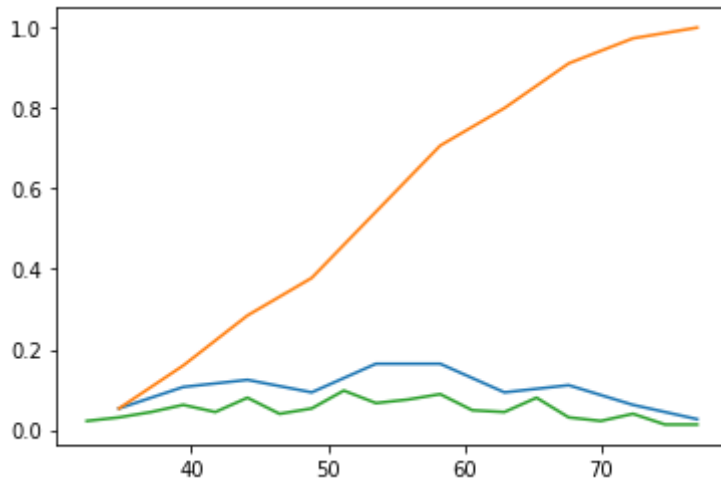
pdf = counts/(sum(counts))
print(pdf);
print(bin_edges);
cdf = np.cumsum(pdf)
plt.plot(bin_edges[1:],pdf);
plt.plot(bin_edges[1:], cdf)
```

```
counts, bin_edges = np.histogram(haberman_alive['age'], bins=20,
                                  density = True)

pdf = counts/(sum(counts))
plt.plot(bin_edges[1:],pdf);

plt.show();
```

```
[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. ]
```



```
counts, bin_edges = np.histogram(haberman_alive['age'], 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();
```

```
[0.05333333 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.  ]
```

### 3.6) Plots of CDF of age for two types of status.

```
0.8 |
    |
counts, bin_edges = np.histogram(haberman_alive['age'], bins=10,
                                density = True)

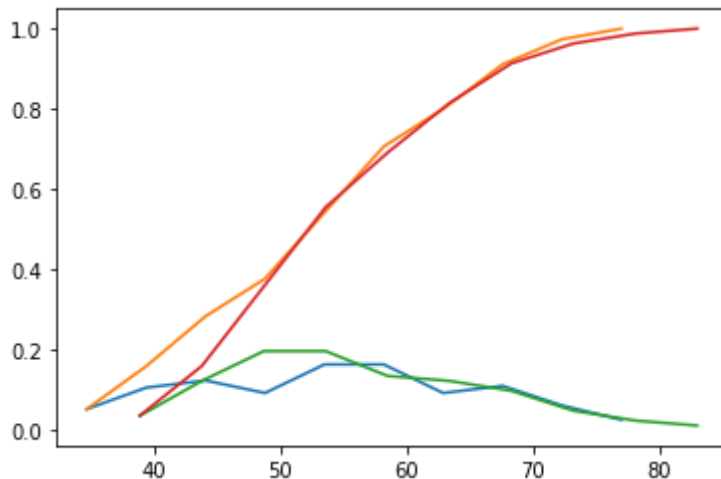
pdf = counts/(sum(counts))
print(pdf);
print(bin_edges)
cdf = np.cumsum(pdf)
plt.plot(bin_edges[1:],pdf)
plt.plot(bin_edges[1:], cdf)
```

```
counts, bin_edges = np.histogram(haberman_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)
plt.plot(bin_edges[1:], cdf)
```

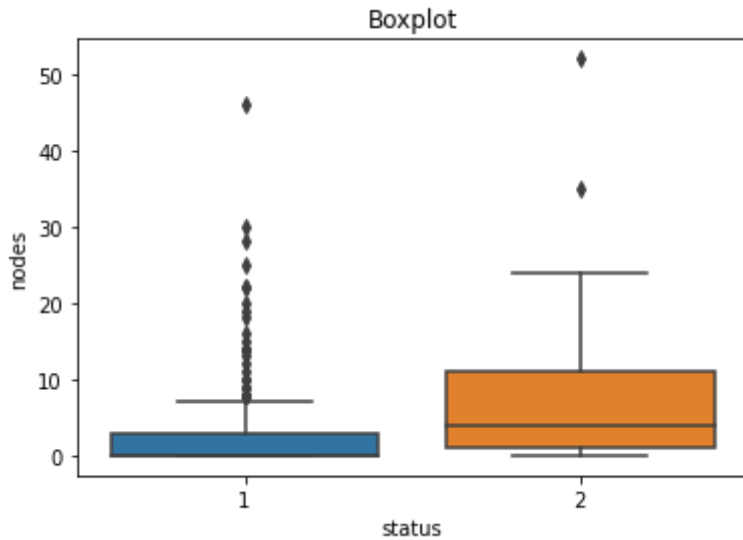
```
plt.show();
```

```
[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.  ]
```



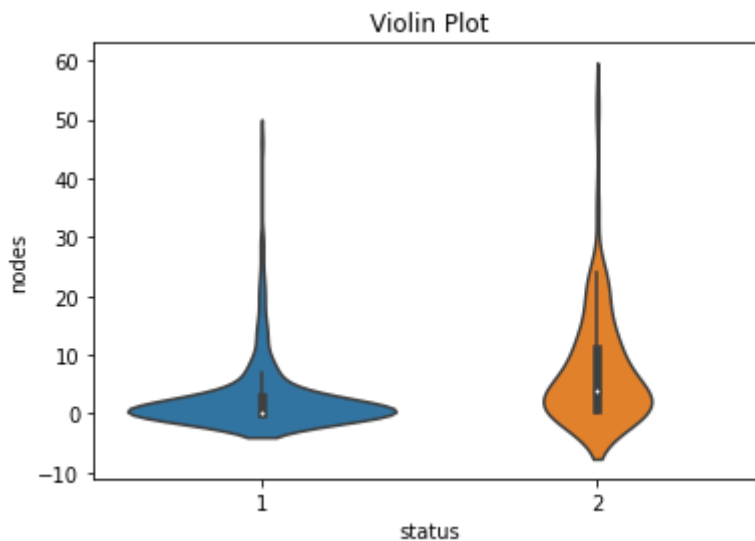
### 3.7) Boxplot

```
sns.boxplot(x='status', y='nodes', data=haberman)
plt.title("Boxplot")
plt.show()
```



### 3.8) Violin Plot

```
sns.violinplot(x="status", y="nodes", data=haberman, size=8)
plt.title("Violin Plot")
plt.show()
```



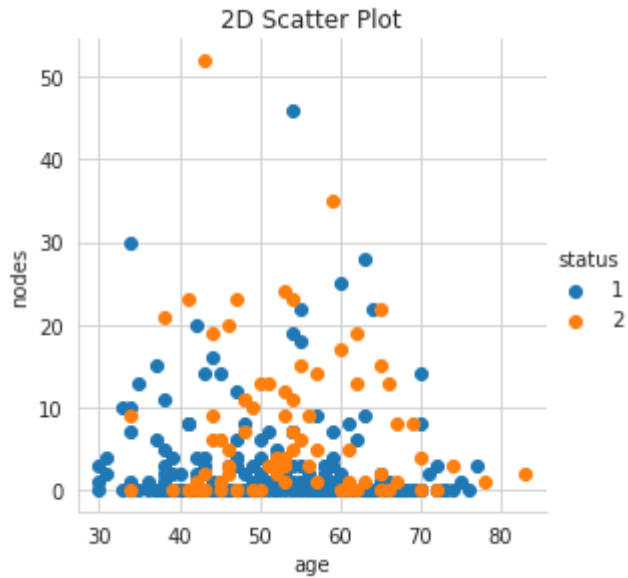
## 3) Perform Bi-variate analysis (scatter plots, pair-plots)

### 3.1) 2-D scatter plot:

```
sns.set_style("whitegrid"):
```

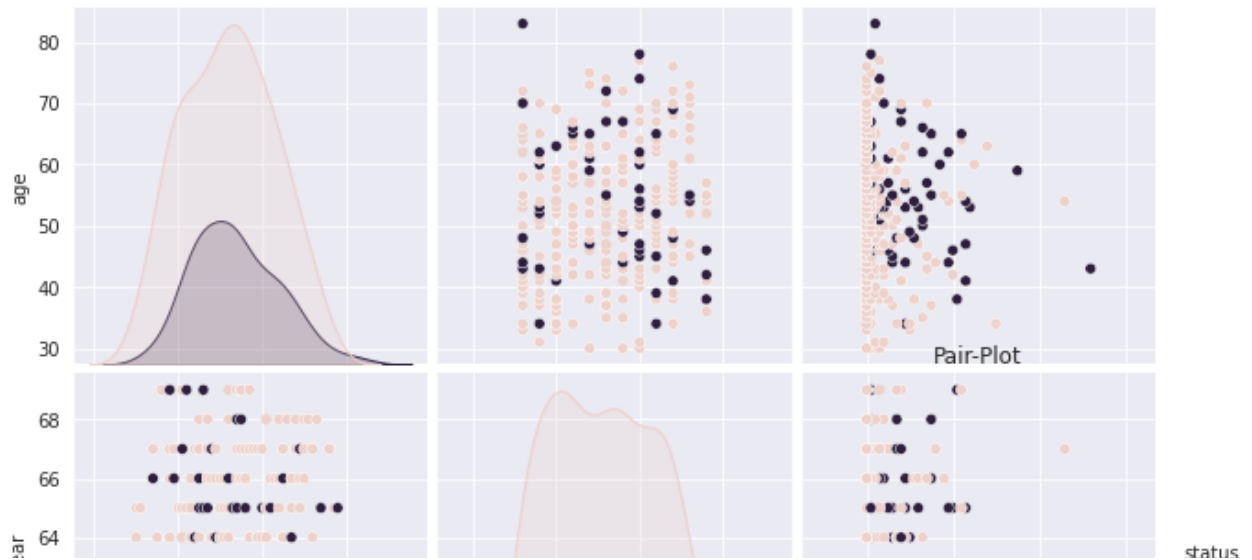
```
sns.FacetGrid(haberman, hue="status", size=4) \
    .map(plt.scatter, "age", "nodes") \
    .add_legend();

plt.title("2D Scatter Plot")
plt.show();
```



### 3.2) Pairwise scatter plot: Pair-Plot

```
plt.close();
sns.set_style("darkgrid");
sns.pairplot(haberman, hue="status", size=3);
plt.title("Pair-Plot")
plt.show()
```



## 5) CONCLUSION:

After analysing the Haberman Dataset I came up to the following conclusion:

- 1) I learned to load datasets, finding number of points, class, features and data points per class.
- 2) I analysed the Probability Distribution Function(PDF) and Cumulative Distribution Function(CDF) where I worked on the graph on status with age, years and nodes.
- 3) I also learned about Boxplot and Violin Plot
- 4) Afterwards I worked on 2-D Scatter Plot and Pair Plot.

