

## Plotting for Exploratory Data Analysis:-

→ Analyzing data using plotting tools, statistics, linear algebra etc.

→ column to be predicted = class label / dependant variable

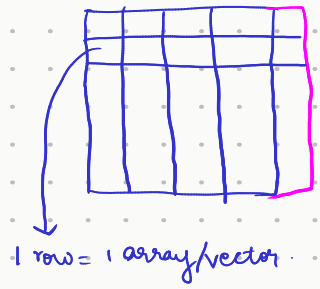
Data already there = data points / vectors

→ n dimensional numerical array.

columns are called features / input variable / independent variable

→ 1D vector = scalar

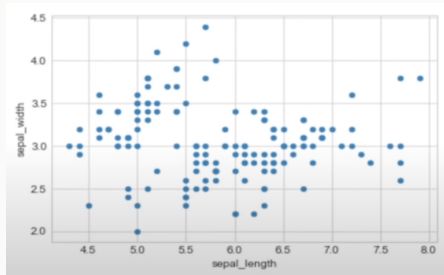
$[a]$ ,  $[1]$ ,  $[8.1]$   
↓     ↓     ↓  
'a'    1     8.1



→ Balanced Dataset → each class has equal number of data points.

→ When reading a plot, always read axis labels & values. It doesn't always start at 0.

→ `iris.plot(kind='scatter', x='sepal_length', y='petal_width');`



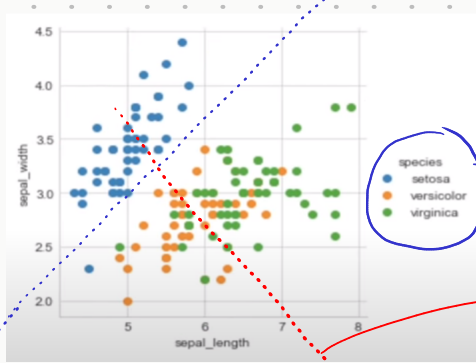
(`sns.scatterplot()` only plots one plot.  
`sns.FacetGrid()` can plot multiple plots.)

→ import Seaborn as sns

`sns.set_style('whitegrid')` → white grid structure

`sns.FacetGrid(iris, hue='species', size=4).map(plt.scatter, 'sepal_length', 'sepal_width').add_legend();`

dataset  
by which  
column should  
the dataset be  
colored



type of plot

x-axis

y-axis

legend

Can't separate as too many outliers.

line could separate setosa from versicolor & virginica (this is called linear separable)

Observations:-  
① S-length & S-width can separate setosa flowers from others.  
② separating versicolor & virginica is harder.

## QUICK SEABORN INTRODUCTION:-

→ simpler way to plot attractive plots.

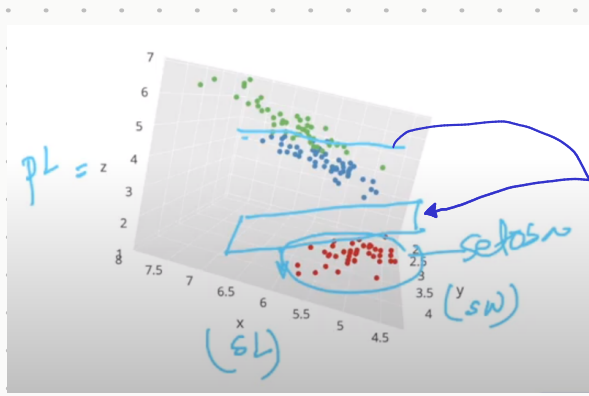
→ high level interface to matplotlib.

→ some features include:

- default aesthetic themes.
- custom color palettes
- ★ • Visualizing info from matrices & dataframes.
- ★ • attractive statistical plots.
- ★ • easily displaying distributions

→ Seaborn is a complement, not a substitute of matplotlib

## 3D Scatter Plot :-



When 3D plot is plotted, we use a plane to separate instead of a line.

Plotting petal length helps notice that plotting a line can separate them which we were not able to do in a 2D plot.

Drawbacks:-  $\rightarrow$  While being interactive is a pro, it can't be viewed on 2D surfaces like paper or in an ipynb.

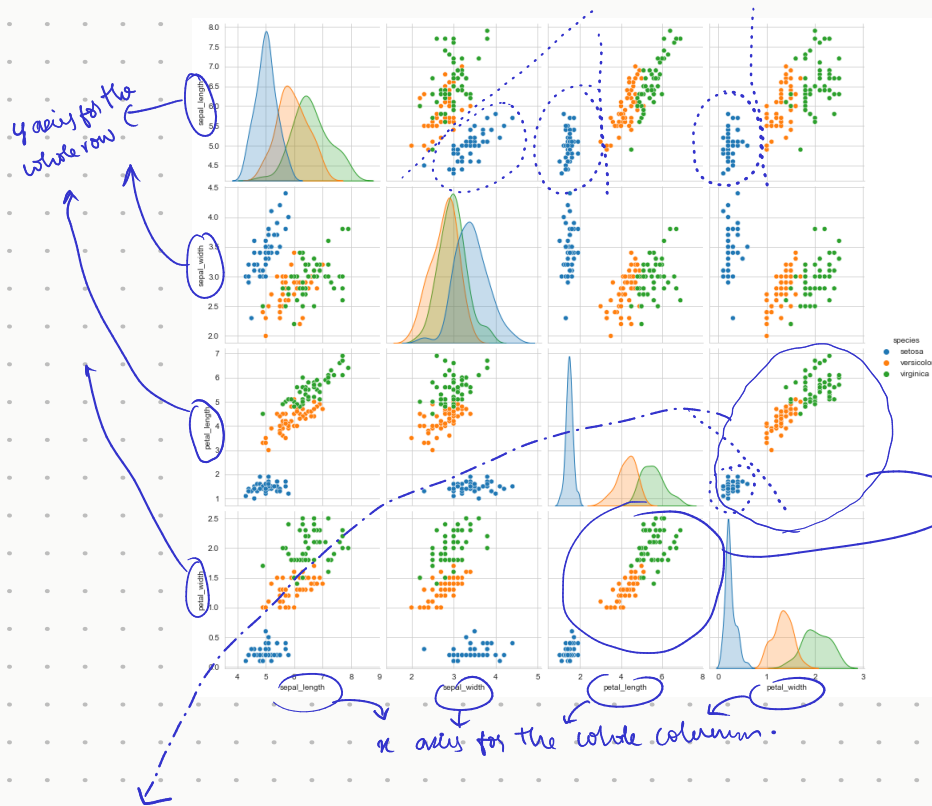
$\rightarrow$  Since we can't visualize 4D datasets, we use pair plots.

there are 4 variable  $\Rightarrow 4C_2$  pairs are possible.

$$6 \rightarrow \begin{cases} (SL, SW), (SW, PL) \\ (SL, PL), (PW, PL) \\ (SL, PW), (SW, PW) \end{cases}$$

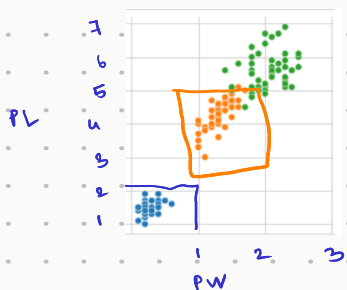
Since we can't visualize 4D, we try to visualize these 6.

`< sns.pairplot(iris, hue="species", size=3)>`



(The diagonal plots are PDFs of each feature)

measure. As both are h/w PL, PW.  
Similarly there are 3 pairs in total.  
So we focus only on 6 plots.



$\Rightarrow$  if  $PL \leq 2$  &  $PW \leq 1$   
then flower type = setosa

if  $(PW \leq 2 \& PW \geq 1) \& (PL \leq 5 \& PL \geq 2.5)$   
then versicolor  
 $\rightarrow$  There will be some error but that's OK.

## Drawback of pair-plots:-

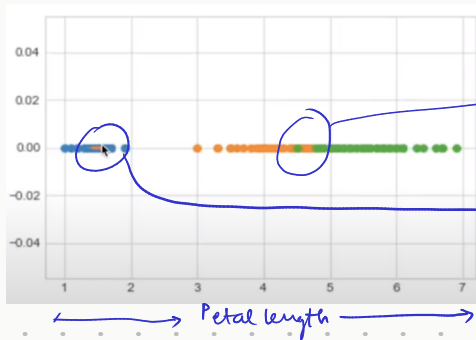
→ when dimensionality is 4, we get  $t_{c_2}$  plots

If the dimensionality is  $n$ , we will get  $n^2$  plots which is not that useful.

So we use techniques like PCA, t-SNE that will help with this.

In real world, we limit pair-plots to only important features & not apply it on all features.

## 1D Scatter Plots:-



→ harder to read.

→ green overlaps orange.

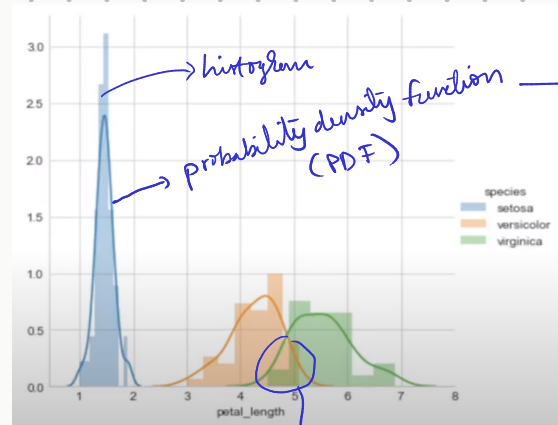
Number of points here is not clear.

To overcome this, instead of using a scatterplot, we can use a histogram.

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

distribution  
plot aka  
histogram aka  
density plot

y-axis:  
count



Smoothed  
histogram.  
Smoothed using  
kernel  
Density  
Estimation.  
(KDE)

if petal length < 2  
then setosa.

if petal length > 2 &  
petal length < 4.7  
then versicolor

if petal length > 4.7  
then virginica.

But  
these  
will  
be  
better