**Seaborn**

**Content**

* **Introduction**
* **Installing and Importing**
* **Loading Dataset using seaborn**
* **Plots using Seaborn**
  + Histogram
  + KDE Plot
  + Scatterplot
  + Joint Plot
  + Pair Plot
* **Categorical Plots**
  + Count Plot
  + Box Plot
  + Voilin Plot
* **Finding correlations among attributes**
  + corr()
  + Heat Map
* **Choosing right visualization for a given purpose**
* **Use Case: Visualizing Tips Dataset**
* **Challenge: Visualize Titanic Dataset**

**Introduction**

* **Seaborn** is also a Python **Data Visualisation Library**
* Just **like Matplotlib**
* Infact, it is **built on top of matplotlib**
* It uses **Pandas DataFrame** to work with datasets

**Installing Seaborn**

**You must know this by now**

Using !pip install command

!pip install seaborn

Requirement already satisfied: seaborn in /home/aryanj/anaconda3/envs/tf-gpu/lib/python3.9/site-packages (0.11.2)

Requirement already satisfied: pandas>=0.23 in /home/aryanj/anaconda3/envs/tf-gpu/lib/python3.9/site-packages (from seaborn) (1.3.4)

Requirement already satisfied: numpy>=1.15 in /home/aryanj/anaconda3/envs/tf-gpu/lib/python3.9/site-packages (from seaborn) (1.21.2)

Requirement already satisfied: scipy>=1.0 in /home/aryanj/anaconda3/envs/tf-gpu/lib/python3.9/site-packages (from seaborn) (1.7.1)

Requirement already satisfied: matplotlib>=2.2 in /home/aryanj/anaconda3/envs/tf-gpu/lib/python3.9/site-packages (from seaborn) (3.5.0)

Requirement already satisfied: pillow>=6.2.0 in /home/aryanj/anaconda3/envs/tf-gpu/lib/python3.9/site-packages (from matplotlib>=2.2->seaborn) (8.4.0)

Requirement already satisfied: packaging>=20.0 in /home/aryanj/anaconda3/envs/tf-gpu/lib/python3.9/site-packages (from matplotlib>=2.2->seaborn) (21.3)

Requirement already satisfied: pyparsing>=2.2.1 in /home/aryanj/anaconda3/envs/tf-gpu/lib/python3.9/site-packages (from matplotlib>=2.2->seaborn) (3.0.4)

Requirement already satisfied: cycler>=0.10 in /home/aryanj/anaconda3/envs/tf-gpu/lib/python3.9/site-packages (from matplotlib>=2.2->seaborn) (0.11.0)

Requirement already satisfied: fonttools>=4.22.0 in /home/aryanj/anaconda3/envs/tf-gpu/lib/python3.9/site-packages (from matplotlib>=2.2->seaborn) (4.25.0)

Requirement already satisfied: python-dateutil>=2.7 in /home/aryanj/anaconda3/envs/tf-gpu/lib/python3.9/site-packages (from matplotlib>=2.2->seaborn) (2.8.2)

Requirement already satisfied: kiwisolver>=1.0.1 in /home/aryanj/anaconda3/envs/tf-gpu/lib/python3.9/site-packages (from matplotlib>=2.2->seaborn) (1.3.1)

Requirement already satisfied: pytz>=2017.3 in /home/aryanj/anaconda3/envs/tf-gpu/lib/python3.9/site-packages (from pandas>=0.23->seaborn) (2021.3)

Requirement already satisfied: six>=1.5 in /home/aryanj/anaconda3/envs/tf-gpu/lib/python3.9/site-packages (from python-dateutil>=2.7->matplotlib>=2.2->seaborn) (1.16.0)

**Importing Seaborn**

* You should be able to import Seaborn after installing it
* We’ll import seaborn as its **alias name sns**

import seaborn as sns

import numpy as np

import matplotlib.pyplot as plt

**Loading Dataset using seaborn**

* Seaborn contains some **datasets in-built within the library itself**
* We can use its **load\_dataset()** function to **read the data**
* Works just like Pandas’s read\_csv()

**Have you heard of the Iris Dataset?**

* It’s like the **“Hello World” of datasets**
* Mostly used by **beginners** as a practice
* It’s the dataset about **3 species of Iris flower**
* Feel free to read more about the dataset here:

<https://en.wikipedia.org/wiki/Iris_flower_data_set>

**Let’s load and read Iris Dataset using seaborn**

iris = sns.load\_dataset('iris')

**Now, Let’s check what all is there in the dataset**

type(iris)

pandas.core.frame.DataFrame

iris.head()

|  | **sepal\_length** | **sepal\_width** | **petal\_length** | **petal\_width** | **species** |
| --- | --- | --- | --- | --- | --- |
| **0** | 5.1 | 3.5 | 1.4 | 0.2 | setosa |
| **1** | 4.9 | 3.0 | 1.4 | 0.2 | setosa |
| **2** | 4.7 | 3.2 | 1.3 | 0.2 | setosa |
| **3** | 4.6 | 3.1 | 1.5 | 0.2 | setosa |
| **4** | 5.0 | 3.6 | 1.4 | 0.2 | setosa |

**As you can see:**

* The iris dataset is **loaded as a Pandas DataFrame**
* The iris dataset has **5 columns (attributes)**
  1. sepal\_length
  2. sepal\_width
  3. petal\_length
  4. petal\_width
  5. species

**We’ll explore these 5 variables (features) one by one**

* The last attribute **species** tells us which **species** the flower is **based on those measurements**

**Let’s check it out in a bit more detail**

iris['species'].unique()

array(['setosa', 'versicolor', 'virginica'], dtype=object)

**So, the flowers in dataset can belong to one of the 3 categories**

1. setosa
2. versicolor
3. virginica

* However, we will not go into the details of the dataset

**The purpose of this lecture is to see the functionality of seaborn**

* Feel free to explore the iris dataset more on your own

**Let’s plot our first graph using seaborn for the variable petal\_length**

iris['petal\_length']

0 1.4

1 1.4

2 1.3

3 1.5

4 1.4

...

145 5.2

146 5.0

147 5.2

148 5.4

149 5.1

Name: petal\_length, Length: 150, dtype: float64

**Histogram**

* We want to check the **distribution of this variable petal\_length**
* So, we use Histogram
* For **Scatter Plot**, we need **2 variables** - one on x-axis and one on y-axis
* For **Bar Chart**, we need a **categorical variable**
  + Like if we want to count how many employees in a company have Bachelors, Masters or PhD.

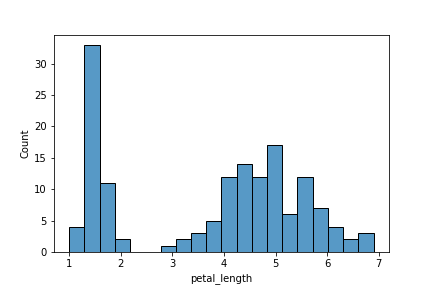
**Remember histogram from the matplotlib lecture?**

* We can use seaborn’s histplot() to plot a histogram
* We can also set the number of bins we want in our plot

**Let’s plot the distribution for the petal\_length column**

sns.histplot(iris['petal\_length'], bins= 20)

plt.show()



**As you can see here**

* We got a **histogram for petal\_length attribute**
* **More than 30 flowers** have petal\_length **between approx 1.25 and 1.5**
* **No flowers** have petal\_length between approx **2.25 and 2.75**
* … and so on. You can make similar other observations

See how neatly each bar is separated from the other

* Like we mentioned seaborn is built on top of matplotlib
* It has **some features** (not all) that are an **enhancement** over matplotlib.pyplot

**Now, let’s check out another plot using seaborn**

**Kernel Density Estimate (KDE) Plot**

* A KDE plot is a method for visualizing the distributions
* Just like histogram
* But instead of bars, KDE represents data using a **continuous probability density curve**
* You can check documentation for more details

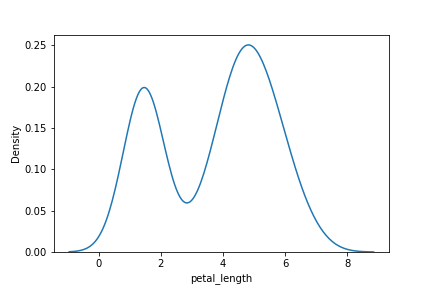
**Now, Why do we even need KDE plots?**

* Compared to histogram, KDE produces a plot which is **less cluttered** and **more interpretable**
* Especially when drawing multiple distributions.

**Let’s plot KDE using seaborn's kdeplot**

sns.kdeplot(iris['petal\_length'])

plt.show()



**As you can see**

* We got a **curve** instead of bars
* **y-axis** has the **probabilities** instead of actual count

**It gives the same information as histogram, but in a smoother way**

* Probability of flowers having petal\_length between approx 1.25 and 1.5 is high (~ 0.2)
* Probability of flowers having petal\_length between approx 2.25 and 2.75 is low (~ 0.05)

**Scatterplot**

**Remember scatterplot from Matplotlib?**

**Scatter Plot is mostly used to visualize relationship b/w 2 variables (Bi-Variate Analysis)**

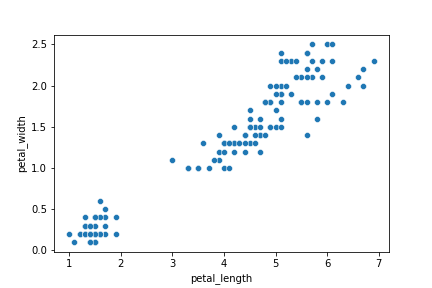
* A scatter plot displays each (x, y) coordinate point separately
* The points are scattered over the graph

**Let’s plot scatterplot now using seaborn**

* We will take **petal\_length as x-coordinates** and **petal\_width as y-coordinates**
* These (x, y) pairs of points will be plotted as a scatter plot

sns.scatterplot(x= iris['petal\_length'], y = iris['petal\_width'])

plt.show()



**Did you notice?**

* We did not have to specifically mention plt.xlabel() or plt.ylabel()
* **Seaborn automatically labelled axes for us**, unlike matplotlib.

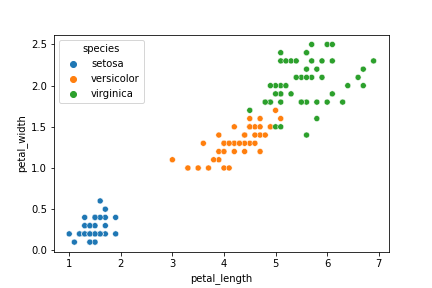
**There’s one more interesting thing we can do with seaborn’s scatterplot**

* We can visualize different species in different colours

**Let’s see how we can do it**

sns.scatterplot(x ='petal\_length', y ='petal\_width' , data= iris, hue='species')

plt.show()



* So, we provide our iris dataset as **input data parameter**
* We **set the hue** to species column

**Each different species is plotted in a different colour**

* The default colours are chosen automatically by the plot
* Check the **top left corner**
  + **Colour Legends** for each category are also shown automatically

**Now, What if we had to draw the same Scatter Plot using Matplotlib?**

* We’d have to separate the datasets for the 3 flowers first
* Then, we’d have to write scatterplot code for each individual dataset

**Let’s try and draw the same Scatter Plot using Matplotlib and compare it with Seaborn**

setosa = iris[iris['species'] == 'setosa']

versicolor = iris[iris['species'] == 'versicolor']

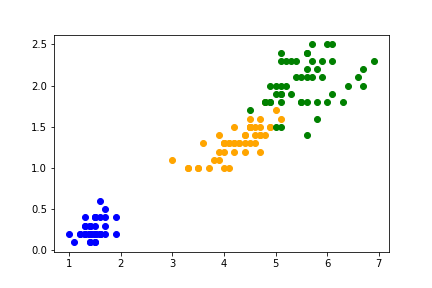
virginica = iris[iris['species'] == 'virginica']

plt.scatter(x=setosa['petal\_length'], y=setosa['petal\_width'], c = 'blue')

plt.scatter(x=versicolor['petal\_length'], y=versicolor['petal\_width'], c = 'orange')

plt.scatter(x=virginica['petal\_length'], y=virginica['petal\_width'], c = 'green')

plt.show()



* As you can see, we got the same plot.
* But with seaborn, the **code is just so much simpler and smaller**
* That’s the **convenience** we have with seaborn

**Let’s see a few more plots that we can visualize using seaborn**

**Joint Plot**

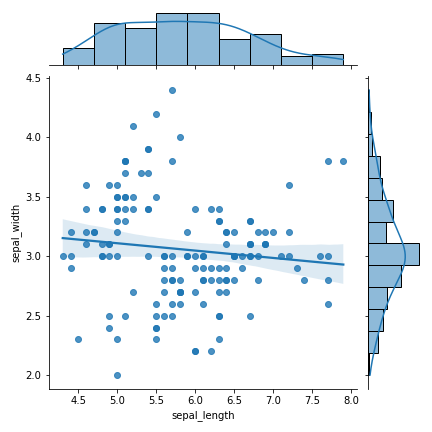
* It draws a plot of two variables
* It shows scatter, histogram and KDE graphs in the same plot.

**Let’s check it out**

* We will take **sepal\_length as x-coordinates** and **sepal\_width as y-coordinates**
* Again, we will pass iris dataset as **input data parameter**
* We can select from different values for **parameter kind** and it **will plot accordingly**
  + “scatter” | “kde” | “hist” | “hex” | “reg” | “resid”
* We will set **parameter kind** to **'reg'** here

sns.jointplot(x= 'sepal\_length', y = 'sepal\_width', data= iris, kind='reg')

plt.show()



**As we can see here:**

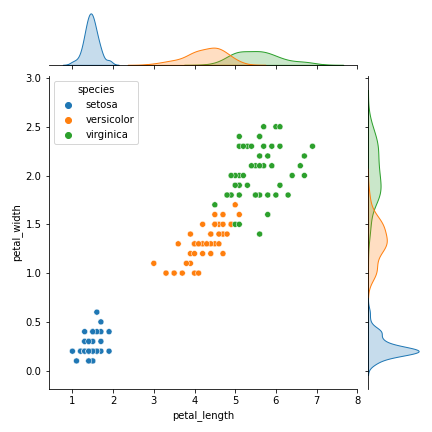
* jointplot plots **scatter, histogram and KDE in the same graph** when we set **kind=reg**
* Scatter shows the **scattering of (sepal\_length, sepal\_width) pairs as (x, y) points**
* Histogram and KDE shows the separate distributions of sepal\_length and sepal\_width in the data

**We can also add hue to Joint Plot**

* Let’s check how the 3 species of flowers are distributed in terms of petal\_length and petal\_width
* We’ll take **petal\_length** as **x-coordinates** and **petal\_width** as **y-coordinates**

sns.jointplot(x= 'petal\_length', y = 'petal\_width', data= iris, hue='species')

plt.show()



**Pair Plot**

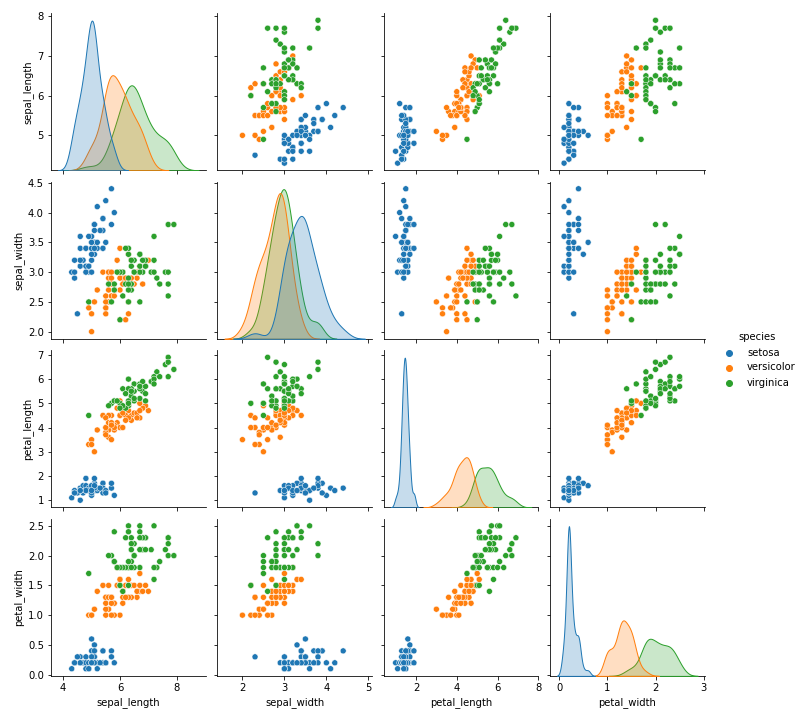
* pairplot() in seaborn creates a **grid of Axes by default**
* Each numeric attribute in data is shared across the y-axes across a single row and the x-axes across a single column.
* It displays a **scatterplot between each pair of attributes in the data** with different **hue** for each category

**Since, the diagonal plots belong to same attribute at both x and y axis, they are treated differently**

* A univariate distribution plot is drawn to show the marginal distribution of the data in each column.

sns.pairplot(data = iris, hue= 'species')

plt.show()



**Notice that:**

* It is **like a scatterplot of iris with hue='species'**
* But the **scatter is plotted between every pair of attributes**
* **Colour Legends** for each species category are given on **right side**
* It shows **relation between each pair of attributes**

**Diagonal plots are different from scatterplots**

* Because x and y axis have same attribute
* Diagonal plots show a univariate curve category-wise for each attribute

**It is also possible to show a subset of variables or plot different variables on the rows and columns**

* Feel free to experiment this on your own

**Categorical Plots**

**Now, we’ll see some Categorical Plots using seaborn**

* Categorical Plots are the **plots based on categories**

**Do we have a categorical variable in our dataset?**

* **species in the case of iris dataset**

**Which plots, in general, do you think are suitable to visualize categorical variables?**

* Bar Plots
* Because we want to see the **frequency/count of data points belonging to each category** of that categorical variable

**Count Plot**

* It’s like a bar plot
* It plots a simple bar graph displaying **count of datapoints (rows) belonging to each category**
* We provide **column label on x-axis**
* **Count of data** (no. of flowers) **belonging to each category** in the column is **on y-axis**

sns.countplot(x = 'species', data = iris)

plt.show()



**What do you notice?**

**How many flowers in dataset belong to each species?**

* **All the 3 species** - setosa, versicolor and virginica have the **same number of flowers**
* There are **50 flowers belonging to each species in the Iris Dataset**

**Box Plot**

* It draws a box plot to show distributions with respect to categories.

**But what exactly is a Box Plot?**

* A box plot or **box-and-whisker plot** shows the **distribution of quantitative data** in a way that **facilitates comparisons between attributes** or **across levels** of a categorical attribute.
* The **box** shows the **quartiles** of the dataset
* While the **whiskers** extend to show the **rest of the distribution**
* Except for points that are determined to be “outliers” using a method that is a function of the **inter-quartile range**.

**Let’s start with understanding what’s a quartile**

* Box plot shows distribution of numerical data and skewness through displaying the **data percentiles**, called **quartiles**

**Box plots show the five-number summary of data:**

1. Minimum score,
2. first (lower) quartile
3. Median
4. Third (upper) quartile
5. maximum score

**Minimum Score**

* It is the **lowest value**, excluding outliers
* It is shown at the **end of bottom whisker**

**Lower Quartile**

* **25% of values fall below the lower quartile value**
* It is also known as the **first quartile**.

**Median**

* Median marks the **mid-point of the data**
* It is shown by the **line that divides the box into two parts**
* It is sometimes known as the **second quartile**.
* **Half the scores are greater than or equal to this value and half are less**.

**Upper Quartile**

* **75% of the values fall below the upper quartile value**
* It is also known as the **third quartile**.
* So, **25% of data are above this value**.

**Maximum Score**

* It is the **highest value**, excluding outliers
* It is shown at the **end of upper whisker**.

**Whiskers**

* The upper and lower whiskers represent **values outside the middle 50%**
* That is, the **lower 25% of values** and the **upper 25% of values**.

**Interquartile Range (or IQR)**

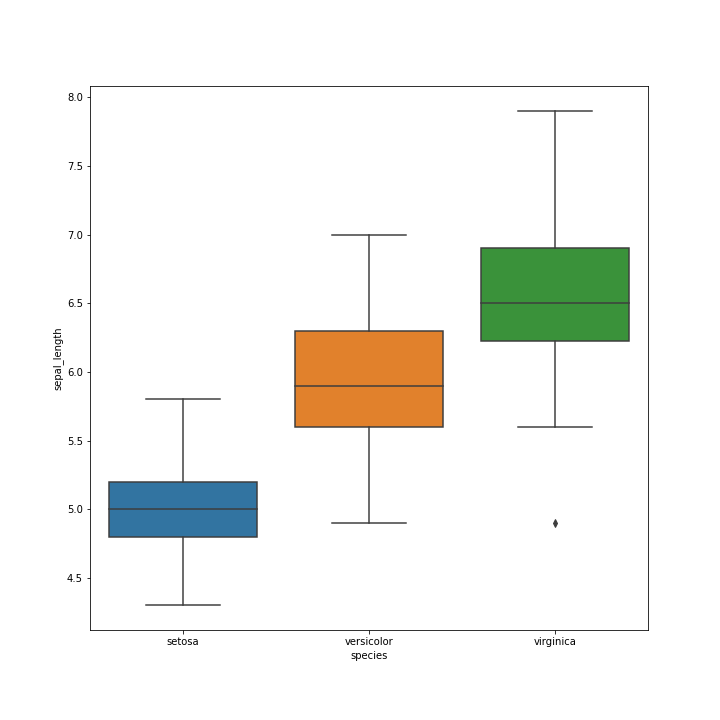
* This is the box plot showing the **middle 50% of scores**
* It is the **range between the 25th and 75th percentile**.

**Now, Let’s plot a box plot to check variation of sepal\_length among the 3 species of iris**

plt.figure(figsize=(10,10))

sns.boxplot(x = 'species', y = 'sepal\_length', data = iris)

plt.show()



**As you can see:**

* **Different species have different range of sepal\_length**

**Each species’ box plot shows:**

* the **lowest sepal\_length** in data for that species
* the **25th percentile (lower quartile) value of sepal\_length** for that species
* the **median sepal\_length** in data for that species
* the **75th percentile (upper quartile) value of sepal\_length** for that species
* the **highest sepal\_length** in data for that species

**Whiskers show the**

* **sepal\_length outside the middle 50% of values**
* The **lower 25% of sepal\_length** and the **upper 25% of sepal\_length**.

**Voilin Plot**

* Its a **combination of Box Plot and Distribution Plot**.
* It works similar to a box and whisker plot.
* It shows the distribution of quantitative data across several levels of categorical attribute such that those distributions can be compared.

**How is it different from box plot?**

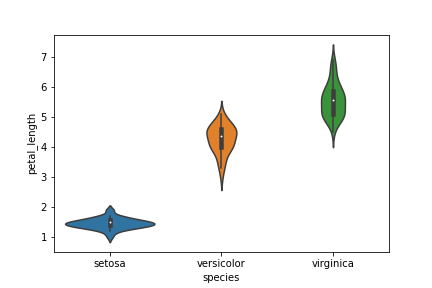
* In a box plot, all of the plot components correspond to actual datapoints
* Whereas, the violin plot features a KDE of the underlying distribution.

**This can be an effective and attractive way to show multiple distributions of data at once**

**Let’s draw a Violin Plot for petal\_length for the 3 species of iris**

sns.violinplot(x = 'species', y = 'petal\_length', data = iris)

plt.show()



**Observe:**

* Inside violin, you will see a boxplot.
* Left and right side represents the **distributions of petal\_length for each species**.

**Finding correlations among attributes**

* We can find the level of correlation b/w different attributes (variables)

**But what exactly is a correlation?**

* Two variables are correlated when **they change in same/opposite direction**

**We can check coefficient of correlation using corr()**

iris.corr()

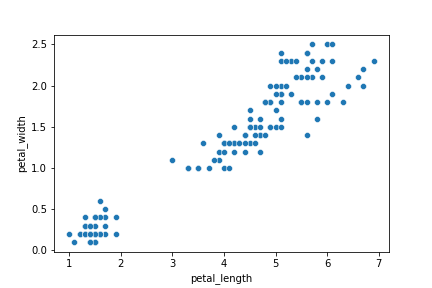
|  | **sepal\_length** | **sepal\_width** | **petal\_length** | **petal\_width** |
| --- | --- | --- | --- | --- |
| **sepal\_length** | 1.000000 | -0.117570 | 0.871754 | 0.817941 |
| **sepal\_width** | -0.117570 | 1.000000 | -0.428440 | -0.366126 |
| **petal\_length** | 0.871754 | -0.428440 | 1.000000 | 0.962865 |
| **petal\_width** | 0.817941 | -0.366126 | 0.962865 | 1.000000 |

* Higher the **MAGNITUDE** of coefficient of correlation, more the variables are **correlated**
* The **sign just determines the direction of change**
  + + means increase in value of one variable causes increase in value of other variable
  + - means increase in value of one variable causes decrease in value of other variable, and vice versa

**As you can see, petal\_length and petal\_width have the highest correlation coeff of 0.96**

sns.scatterplot(x= 'petal\_length', y= 'petal\_width', data = iris)

plt.show()



* When petal\_length increases, petal\_width also increases

**But Remember**

**Correlation does NOT mean Causation**

* We cannot conclude that change in values of a variable is causing change in values of other variable

**Now, Let’s look at a way to visualize correlation among variables**

**Heat Map**

* A heat map plots rectangular data as a color-encoded matrix.
* **Stronger the colour, stronger the correlation b/w the variables**

**Let’s plot a Heat Map using correlation coefficient matrix generated using corr()**

sns.heatmap(iris.corr(), cmap= "Blues", annot=True)

plt.show()



* **annot=True** is for writing correlation coeff inside each cell

**You can change the colours of cells in Heat Map if you like**

* There are a lot of options available!

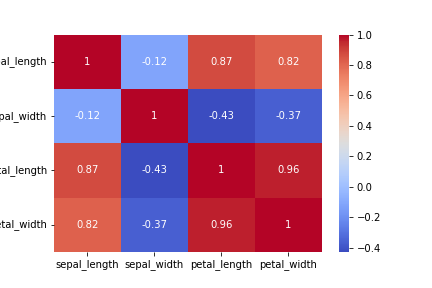
print(plt.colormaps())

['magma', 'inferno', 'plasma', 'viridis', 'cividis', 'twilight', 'twilight\_shifted', 'turbo', 'Blues', 'BrBG', 'BuGn', 'BuPu', 'CMRmap', 'GnBu', 'Greens', 'Greys', 'OrRd', 'Oranges', 'PRGn', 'PiYG', 'PuBu', 'PuBuGn', 'PuOr', 'PuRd', 'Purples', 'RdBu', 'RdGy', 'RdPu', 'RdYlBu', 'RdYlGn', 'Reds', 'Spectral', 'Wistia', 'YlGn', 'YlGnBu', 'YlOrBr', 'YlOrRd', 'afmhot', 'autumn', 'binary', 'bone', 'brg', 'bwr', 'cool', 'coolwarm', 'copper', 'cubehelix', 'flag', 'gist\_earth', 'gist\_gray', 'gist\_heat', 'gist\_ncar', 'gist\_rainbow', 'gist\_stern', 'gist\_yarg', 'gnuplot', 'gnuplot2', 'gray', 'hot', 'hsv', 'jet', 'nipy\_spectral', 'ocean', 'pink', 'prism', 'rainbow', 'seismic', 'spring', 'summer', 'terrain', 'winter', 'Accent', 'Dark2', 'Paired', 'Pastel1', 'Pastel2', 'Set1', 'Set2', 'Set3', 'tab10', 'tab20', 'tab20b', 'tab20c', 'magma\_r', 'inferno\_r', 'plasma\_r', 'viridis\_r', 'cividis\_r', 'twilight\_r', 'twilight\_shifted\_r', 'turbo\_r', 'Blues\_r', 'BrBG\_r', 'BuGn\_r', 'BuPu\_r', 'CMRmap\_r', 'GnBu\_r', 'Greens\_r', 'Greys\_r', 'OrRd\_r', 'Oranges\_r', 'PRGn\_r', 'PiYG\_r', 'PuBu\_r', 'PuBuGn\_r', 'PuOr\_r', 'PuRd\_r', 'Purples\_r', 'RdBu\_r', 'RdGy\_r', 'RdPu\_r', 'RdYlBu\_r', 'RdYlGn\_r', 'Reds\_r', 'Spectral\_r', 'Wistia\_r', 'YlGn\_r', 'YlGnBu\_r', 'YlOrBr\_r', 'YlOrRd\_r', 'afmhot\_r', 'autumn\_r', 'binary\_r', 'bone\_r', 'brg\_r', 'bwr\_r', 'cool\_r', 'coolwarm\_r', 'copper\_r', 'cubehelix\_r', 'flag\_r', 'gist\_earth\_r', 'gist\_gray\_r', 'gist\_heat\_r', 'gist\_ncar\_r', 'gist\_rainbow\_r', 'gist\_stern\_r', 'gist\_yarg\_r', 'gnuplot\_r', 'gnuplot2\_r', 'gray\_r', 'hot\_r', 'hsv\_r', 'jet\_r', 'nipy\_spectral\_r', 'ocean\_r', 'pink\_r', 'prism\_r', 'rainbow\_r', 'seismic\_r', 'spring\_r', 'summer\_r', 'terrain\_r', 'winter\_r', 'Accent\_r', 'Dark2\_r', 'Paired\_r', 'Pastel1\_r', 'Pastel2\_r', 'Set1\_r', 'Set2\_r', 'Set3\_r', 'tab10\_r', 'tab20\_r', 'tab20b\_r', 'tab20c\_r', 'rocket', 'rocket\_r', 'mako', 'mako\_r', 'icefire', 'icefire\_r', 'vlag', 'vlag\_r', 'flare', 'flare\_r', 'crest', 'crest\_r']

sns.heatmap(iris.corr(), cmap= "coolwarm", annot=True)

plt.savefig("plot13")

plt.show()



**Choosing right visualization for a given purpose**

* There’s a whole bunch of charts and plots we’ve seen
  + Bar chart
  + Historgam
  + Box Plot
  + Violin Plot
  + Scatterplot
  + Count Plot
  + Heat Map
  + … and so on
* But we always need to **select the right plot for every purpose**

**What is the right chart to use for a given problem?**

* We need to decide the right chart/plot to use for a dataset at-hand
* We can’t just blindly use any chart for any data that’s available to us
* There are certain **thumb rules** that we need to consider

**Now, What exactly is the process of selecting the right chart?**

* First you need to look at what is the type of variable you’re dealing with

**Let’s divide this step in into:**

1. 1-Dimensional
2. 2-Dimensional
3. Multi-Dimensional

**1-Dimensional Visualization**

**Which one will we choose when we just want to analyze 1 variable?**

* 1-D
* We look at only 1 variable at a time

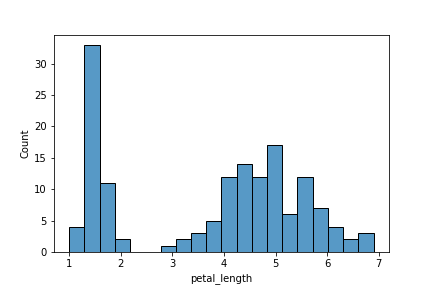
**Now, Which plot to use if the variable is continuous (numeric)?**

* **Histogram** —> To see **distribution** of that continuous variable
* **Box Plot** —> To see the **inter-quartile range of values** of that continuous variable

**For example: What we saw with petal\_length above:**

sns.histplot(iris['petal\_length'], bins= 20)

plt.show()



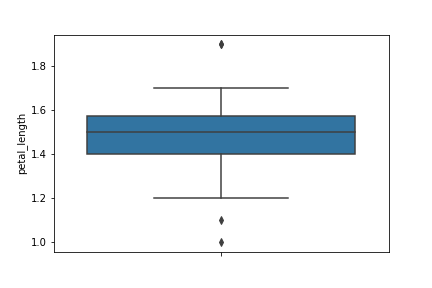
**Let’s say we want to visualize how petal\_length varies for just setosa species**

* We can utilize the **Box Plot** here we saw earlier

setosa = iris[iris['species'] == 'setosa']

sns.boxplot(y = 'petal\_length', data = setosa)

plt.show()



**What if the variable is categorical?**

* Categorical variable will have **discrete unique values**
* **Bar Chart / Count Plot** —> To see **how many datapoints belong to each category**
* **Pie Chart** —> To see **ratio (%age) of datapoints** belonging to each category

**For example: Bar/Count Plot we saw above to check no. of flowers belonging to each species:**

sns.countplot(x = 'species', data = iris)

plt.show()



**We can also check the proportions / %ages of flowers belonging to each species using a Pie Chart:**

* **Seaborn doesn’t have a direct function to create a pie chart**
* So, we’ll use **matplotlib** here

n\_setosa = iris[iris['species'] == 'setosa'].shape[0] # We are getting the number of datapoints in each species

n\_versicolor = iris[iris['species'] == 'versicolor'].shape[0]

n\_virginica = iris[iris['species'] == 'virginica'].shape[0]

data = [n\_setosa, n\_versicolor, n\_virginica]

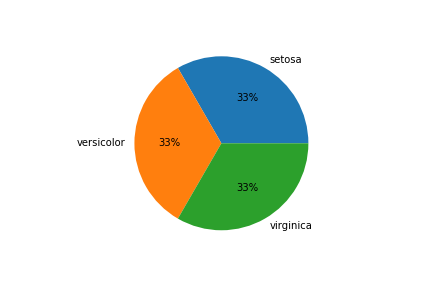
labels=['setosa','versicolor','virginica']

plt.pie(data,

labels=labels,

autopct='%.0f%%') # To show the portions in %ages

plt.show()



**2-Dimensional Visualization**

**Now, What if we want to analyze 2 variables at a time?**

* We usually do this to check the relationship b/w 2 variables
* Its a **Bi-Variate Analysis**

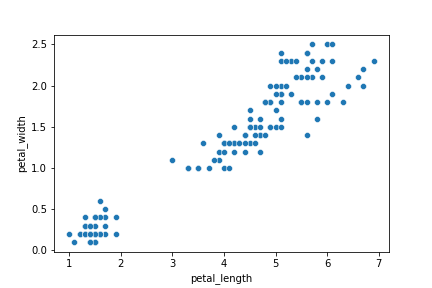
**Now, Which chart will we use if both variables are continuous (numeric)?**

* **Scatter Plot** —> To see how the 2 continuous variables are **dependent on each other or vary with each other**
* **Line Chart** —> To see the **approximate relationship (dependency) b/w the 2 variables** represented by a line

**For example: When we analyzed how petal\_length and petal\_width vary with each other**

sns.scatterplot(x= 'petal\_length', y= 'petal\_width', data = iris)

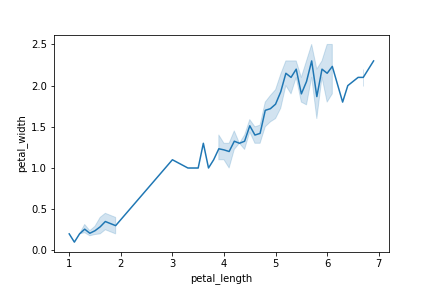
plt.show()



**We can also use Line Chart to get an approximate relationship b/w petal\_length and petal\_width**

sns.lineplot(x= 'petal\_length', y= 'petal\_width', data = iris)

plt.show()



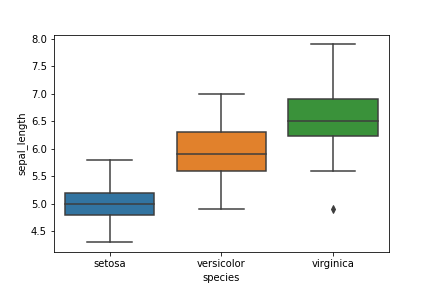
**Now, Which chart to use when we have 1 continuous and 1 discrete variable?**

* You should know this by now!
* **Box Plot** —> To see **distribution of numeric variable across each category of categorical variable**

**For example: When we checked above how sepal\_length varies for each species of iris**

sns.boxplot(x = 'species', y = 'sepal\_length', data = iris)

plt.show()



* It helps us answer questions like **“is there a change/difference in sepal\_length for different species?”**
* Box Plot allows these types of **comparisons of a numerica variable among different categories**

**Now, What if both our variables are categorical?**

* We can use a Bar Chart (Stacked or Dodged)

**For example:**

* Let’s say we have some employees data of a company
* We want to check **how many employees are there in each department - Engg, HR, Operations, Sales**
* Now, we want to check **how many Males and Females are there in each department**
* A **Stacked/Dodged Bar Chart can be appropriately used here**

**Multi-Dimensional Visualization**

**Now, Let’s talk about multi-dimensional visualization**

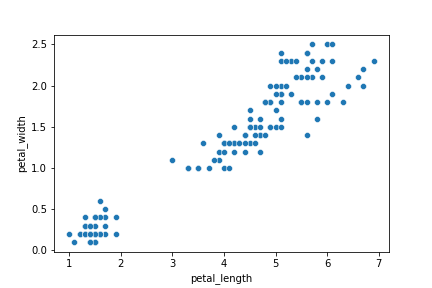
* It’s not very straightforward
* But if plotted in a **constructive way**, a multi-dimensional plot can reveal a lot of useful information

**Let’s start with 2 dimensions, i.e., 2 variables, and then we’ll add more dimensions to the plot**

**Do you remember the Scatter Plot b/w petal\_length and petal\_width?**

sns.scatterplot(x= 'petal\_length', y= 'petal\_width', data = iris)

plt.show()



We can make use of

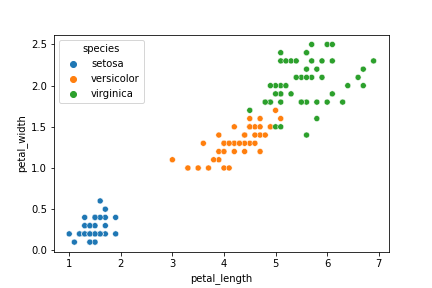
* Colour
* Shape
* Size

to add more dimensions to our Scatter Plot

We have already used “colour” when we plotted Scatter Plot b/w petal\_length and petal\_width and added species as hue to the Scatter Plot

sns.scatterplot(x ='petal\_length', y ='petal\_width' , data= iris, hue='species')

plt.show()



* Different species are now represented by different colours
* We had a 2-D Scatter Plot b/w petal\_length and petal\_width
* The, we added **species as a 3rd dimension** using the **colour**

**Now, It’s a multi-dimensional plot**

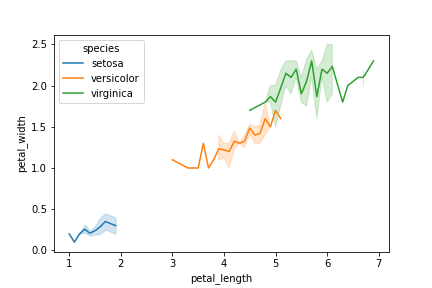
**By looking at it, Can you now tell “which points belong to which species of iris”?**

* You can also tell what’s the **range (clusters) of petal\_length and petal\_width** of each species

**We can also add this dimension of species in our Line Plot we saw earlier:**

sns.lineplot(x= 'petal\_length', y= 'petal\_width', data = iris, hue='species')

plt.show()



* Same way we can add more dimensions to the same plot using **shape** and **size** to both Scatter Plots
* We’ll cover them when we come to **Exploratory Data Analysis (EDA)**
* For now, just remember that we can always **constructively plot a multi-dimensional plot to reveal important information from our data**

**Visualizing Tips Dataset**

* Let’s move on to visualizing another dataset from seaborn library
* **Tips Dataset**
* Its some tipping data where one waiter recorded information about each tip he/she received over a period of a few months working in one restaurant

**Let’s go ahead and load the data**

tips = sns.load\_dataset('tips')

**Let’s see what all the tips data includes**

tips.shape

(244, 7)

tips.head()

|  | **total\_bill** | **tip** | **sex** | **smoker** | **day** | **time** | **size** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 16.99 | 1.01 | Female | No | Sun | Dinner | 2 |
| **1** | 10.34 | 1.66 | Male | No | Sun | Dinner | 3 |
| **2** | 21.01 | 3.50 | Male | No | Sun | Dinner | 3 |
| **3** | 23.68 | 3.31 | Male | No | Sun | Dinner | 2 |
| **4** | 24.59 | 3.61 | Female | No | Sun | Dinner | 4 |

**It has 7 columns**

* How much was the total bill
* How much did the customer tip
* Was the customer male or female
* Was the customer a smoker or not
* What day was it
* What meal time was it
* size represents no. of people in the party

**Now, we’ll answer a few questions related to dataset using visualization through seaborn**

**Can you visualize in how many cases the customer was male and in how many cases the customer was female?**

sns.countplot(x = tips['sex'])

plt.show()



**What all days does the dataset has?**

tips['day'].unique()

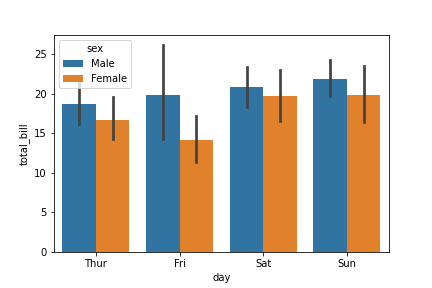
['Sun', 'Sat', 'Thur', 'Fri']

Categories (4, object): ['Thur', 'Fri', 'Sat', 'Sun']

**Can you tell out of all the days which day gets the highest bill amount for both male and female customers?**

sns.barplot(x = 'day', y = 'total\_bill', data=tips, estimator=np.mean, hue = 'sex')

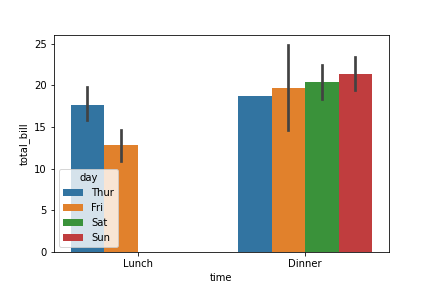
plt.show()



**Can you tell when do people prefer coming to restaurant more - lunch time or dinner time? And on what days?**

sns.barplot(x = 'time', y = 'total\_bill', data = tips, hue = 'day')

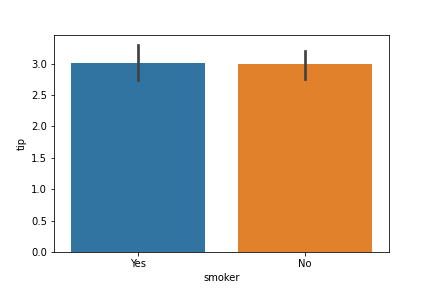
plt.show()



**Can you tell whether people who smoke give more tip or not?**

sns.barplot(x = 'smoker', y = 'tip', data = tips)

plt.show()

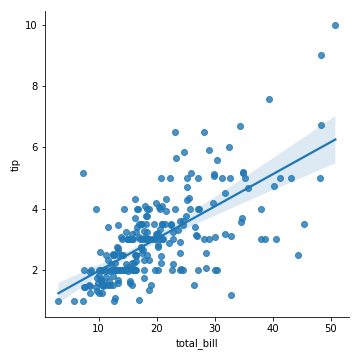


**How are total\_bill and tip related? Is there a correlation b/w them?**

* We can use lmplot()

sns.lmplot(x = 'total\_bill', y= 'tip', data = tips)

plt.show()



**This is all for seaborn for now**

* You can explore the library more by checking out its docs

Last changed by

**Scaler Topics**Scaler Topics

 165

Published on [**HackMD**](https://hackmd.io/)