Seaborn

- Seaborn is visualization library for statistical plotting
- It is designed to work with data frame objects in pandas
- It contains default attractive styles

In [1]:
```
import seaborn as sns
```

In [2]:
```
sns.__version__
```

Out[2]: '0.12.2'

color_palette()

- It is used to generate few colours in seaborn
- sns.color_palette()

In [3]:
```
sns.color_palette()
```

...

In [4]:
```
help(sns.color_palette())
```

...

In [5]:
```
sns.palplot(sns.color_palette())
```

...

In [6]:
```
sns.palplot(sns.color_palette("muted"))
```

...

In [7]:
```
sns.palplot(sns.color_palette("deep"))
```

...

In [10]:
```
sns.palplot(sns.light_palette("blue"))
```

...

In [11]:
```
sns.palplot(sns.dark_palette("blue"))
```

...

In [17]:
```
1 sns.get_dataset_names()
```

...

In [18]:
```
1 iris=sns.load_dataset('iris')
```

In [20]:
```
1 iris.shape
```

Out[20]: (150, 5)

In [21]:
```
1 iris.head()
```

...

In [22]:
```
1 iris.tail()
```

...

Categorical plot

- By default it gives scatter plot
- sns.catplot()

In [25]:
```
1 sns.catplot(x='species',y='sepal_length',data=iris)
```

...

Categorical distribution of data

- boxplot

In [26]:
```
1 help(sns.catplot)
```

...

In [ ]:
```
1 :func:`stripplot` (with `kind="strip"`; the default)
2     - :func:`swarmplot` (with `kind="swarm"`)
3
4     Categorical distribution plots:
5
6     - :func:`boxplot` (with `kind="box"`)
7     - :func:`violinplot` (with `kind="violin"`)
8     - :func:`boxenplot` (with `kind="boxen"`)
```

In [29]:
```
1 sns.catplot(x="sepal_length",y='sepal_width',data=iris,
2             kind="box")
```

...

In [30]:
```
1 sns.catplot(data=iris,kind='box')
```
...

In [31]:
```
1 sns.catplot(data=iris,kind='box',orient='h')
```
...

In [32]:
```
1 sns.catplot(x='species',y='sepal_width',data=iris,
2             kind='violin')
```
...

In [33]:
```
1 sns.catplot(x='species',y='sepal_width',data=iris,
2             kind='bar')
```
...

In [34]:
```
1 sns.catplot(x='species',y='sepal_length',data=iris,
2             kind='boxen')
```
...

In [35]:
```
1 sns.catplot(x='species',y='petal_length',data=iris,
2             kind='swarm')
```
...

In [36]:
```
1 sns.catplot(x='species',y='petal_width',data=iris,
2             kind='strip')
```
...

Heatmaps

- It is used to find correlation between two columns in a data frame
- sns.heatmap()

In [52]:
```
1 sns.get_dataset_names()
2 t=sns.load_dataset('tips')
```

In [53]:
```
1 t
```
...

In [54]:
```
1 t.head()
```
...

In [55]:
```
1 t.corr()
```

...

In [56]:
```
1 sns.heatmap(t.corr())
```

...

Scikit learn (sklearn)

- Scikit-learn is one of the popular framework for Data science
- scikit-learn contains tools for data preprocessing and data mining
- data preprocessing is a technique to convert raw data into a clean dataset

In [57]:
```
1 from sklearn.impute import SimpleImputer
```

In [58]:
```
1 import numpy as np
2 import pandas as pd
3 d={"a":pd.Series([12,78,90,np.nan,85],index=[1,2,3,4,5]),
4   "b":pd.Series([87,677,55,90],index=[1,3,4,5]),
5   "c":pd.Series([12,89,908,54],index=[1,2,3,4])}
6 df=pd.DataFrame(d)
```

In [59]:
```
1 df
```

...

In [60]:
```
1 si=SimpleImputer(strategy='median')
2 si.fit_transform(df)
```

...

In [61]:
```
1 df.median()
```

...

In [62]:
```
1 si=SimpleImputer(strategy='mean')
2 si.fit_transform(df)
```

...

In [63]:
```
1 df.mean()
```

...

In [65]:
```
1 si=SimpleImputer(strategy='most_frequent')
2 si.fit_transform(df)
```

...

In [ ]:
```
[100 gms , 10 tons , 1 kg]  # weights

100+10+1.....  # incorrect result
```

Scaling techniques

- Standardizing data
- standard data means 0 mean and unit variance

In [66]:
```
://Users//1pssdc//OneDrive//Desktop//DATA ANALYSIS WORKSHOP//Advertising.csv')
```

In [67]:
```
adv
```

...

In [72]:
```
sns.kdeplot(adv["TV"])   # kernal density plot
sns.kdeplot(adv["radio"])
sns.kdeplot(adv["newspaper"])
```

...

In [ ]:
```
# standard format x-mean(x)/std(x)
```

In [73]:
```
adv["TV"][0]
```

Out[73]: 230.1

In [74]:
```
# Standardizing data
(adv["TV"][0]-adv["TV"].mean())/adv["TV"].std()
```

Out[74]: 0.9674245973763037

In [79]:
```
from sklearn.preprocessing import scale
```

In [80]:
```
scl=scale(adv)
scl
```

...

In [81]:
```
scl_data=pd.DataFrame(scl,columns=adv.columns)
scl_data
```

...

In [84]:
```
sns.kdeplot(scl_data["TV"])
sns.kdeplot(scl_data["radio"])
sns.kdeplot(scl_data["newspaper"])
```

...

In [85]:
```
adv.mean()
```

...

In [89]:
```
scl_data.std().round(3)
```

...

Data Range

- It is done column wise or label wise
- Scaling is done by compressing data into a fixed range
- in most use cases [0,1]
- MinMaxScaler()

In [90]:
```
from sklearn.preprocessing import MinMaxScaler
```

In [91]:
```
mnsl=MinMaxScaler()
```

In [92]:
```
mnscale=mnsl.fit_transform(adv)
```

In [93]:
```
adv.mean()
```

...

In [95]:
```
mnscale.mean().round(4)
```

...

In [96]:
```
mnscale.std().round(3)
```

...

Normalizing data

- scaling is done based on individual rows
- When clustering data , we need to apply L2 normalization
- L2 norm is squareroot of sum of squared values of each row

In [104]:
```
1 from sklearn.preprocessing import Normalizer
```

In [106]:
```
1 home=pd.read_csv('C://Users//apssdc//OneDrive//Desktop//DATA ANALYSIS WORK
```

In [107]:
```
1 home
```

...

In [108]:
```
1 nor=Normalizer()
```

In [109]:
```
1 nor_data=nor.fit_transform(home)
```

...

In [110]:
```
1 nor_data
```

...

Robust scaling

- Deals with outliers
- Robustly scales the data, avoid being affected by outliers

In [111]:
```
1 from sklearn.preprocessing import RobustScaler
```

In [112]:
```
1 rscl=RobustScaler().fit_transform(adv)
```

In [115]:
```
1 rscl_data=pd.DataFrame(rscl,columns=adv.columns)
2 rscl_data
```

...

In [119]:
```
1 sns.kdeplot(rscl_data["TV"])
2 sns.kdeplot(rscl_data["radio"])
3 sns.kdeplot(rscl_data["newspaper"])
```

...

In [ ]:
```
1
```