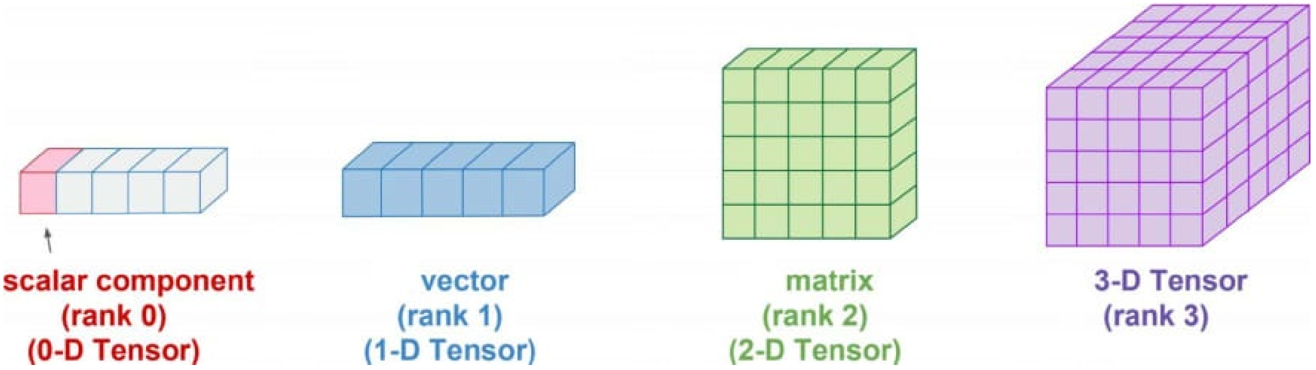
Tensor is like data structure for Machine learning. OR it is multidimensional NumPy array.

0D Tensor, 1D Tensor, 2D Tensor



In [1]:
```
import numpy as np
a=np.array(1)
```

In [2]:
```
type(a)
```

Out[2]:
```
numpy.ndarray
```

In [3]:
```
a.ndim
```

Out[3]:
```
0
```

In [4]:
```
#This is 0D tensor or 0 dimensional numpy array.
```

In [5]:
```
b=np.array([1,2,3])
type(b)
b.ndim
```

Out[5]:
```
1
```

In [6]:
```
#This is 1 D tensor or 1 Dimensional NumPy array.
```

NOTE: 1D array is also known as vector, but if someone asked what is dimension of b vector then there is 3 number so vector dimesion will be 3.

In [7]:
```
#2D Tensor or Matrix
```

In [8]:
```
c=np.array([[1,2,3],[4,5,6]])
c.ndim
```

Out[8]:
```
2
```

In [9]:
```
print(c)
```

```
[[1 2 3]
 [4 5 6]]
```

What is rank of c?

- Two: row and column

In [10]:
```
## 3D tensor - its a combination of 2D
```

In [11]:
```
import numpy as np
d=np.array([[[1,2,3],[4,5,6],[7,8,9]]])
```

In [12]:
```
d.ndim
```

Out[12]:
```
3
```

In [13]:
```
d
```

Out[13]:
```
array([[[1, 2, 3],
        [4, 5, 6],
        [7, 8, 9]]])
```

In [14]:
```
len(d)          #as we have created nested array so in length, len only counts outermost array.
```

Out[14]:
```
1
```

In [15]:
```
d.size
```

In [15]:

```
d.size
```

Out[15]:

```
9
```

PROJECT

In [16]:

```
import numpy as np
import pandas as pd
```

In [17]:

```
df=pd.read_csv(r"C:\Users\USER\Downloads\placement.csv")
```

In [18]:

```
df.head()
```

Out[18]:

	Unnamed: 0	cgpa	iq	placement
0	0	6.8	123.0	1
1	1	5.9	106.0	0
2	2	5.3	121.0	0
3	3	7.4	132.0	1
4	4	5.8	142.0	0

In [19]:

```
#Now we are seeing last three column is important but front 1 is unnecassry so lets use slicing.
```

In [20]:

```
df=df.iloc[:,1:]   #all rows, needed column from 1 to all column
```

In [21]:

```
df.head()
```

Out[21]:

	cgpa	iq	placement
0	6.8	123.0	1
1	5.9	106.0	0
2	5.3	121.0	0
3	7.4	132.0	1
4	5.8	142.0	0

In [22]:

```
# Now lets check if there is any missing values or not
df.isnull().sum()
```

Out[22]:

```
cgpa         0
iq           0
placement    0
dtype: int64
```

In [23]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 3 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   cgpa       100 non-null    float64
 1   iq         100 non-null    float64
 2   placement  100 non-null    int64  
dtypes: float64(2), int64(1)
memory usage: 2.5 KB
```

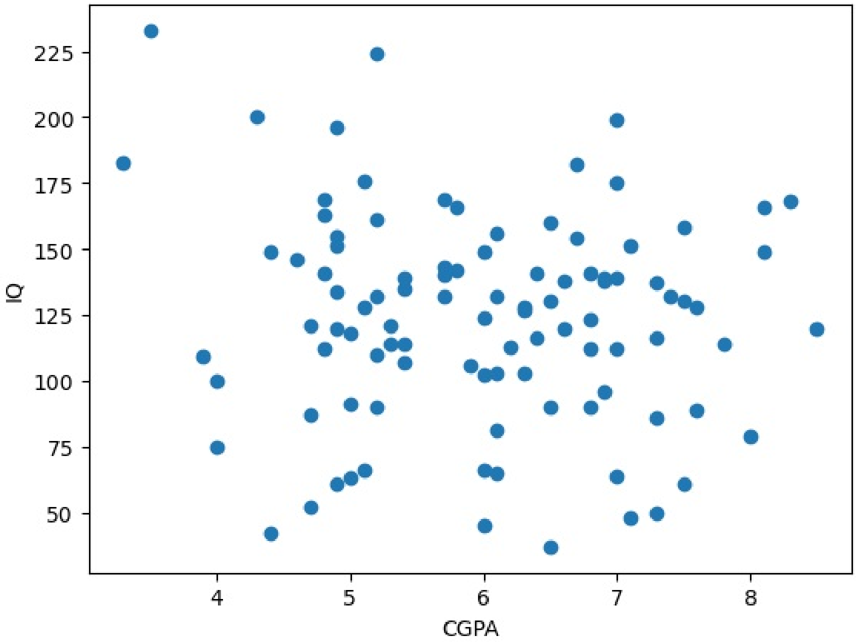
NOTE: WE are just looking very basic ML lifecycle : We just did preprocessing above, now for EDA, here we do not have anything so for now lets just see Scatter plot with the help of matplotlib

In [24]:

```
import matplotlib.pyplot as plt
```

In [25]:

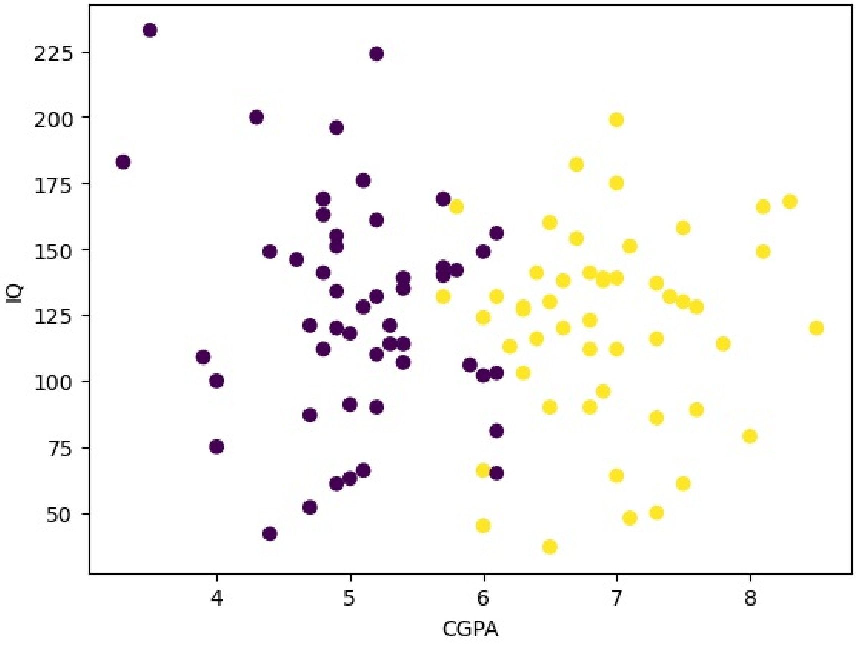
```
plt.scatter(df["cgpa"],df["iq"])   #seeing relationship between cgpa and iq
plt.xlabel("CGPA")
plt.ylabel("IQ")
plt.show()
```



Now for now among these student, if we want to see placements and non placements students, just add few code:

In [26]:

```
plt.scatter(df["cgpa"],df["iq"], c=df["placement"])   #seeing relationship between cgpa and iq
plt.xlabel("CGPA")
plt.ylabel("IQ")
plt.show()
```



In [27]:

```
#yellow color one is placement
```

Feature selection: For now, we are not doing this, but this is another step in ML lifecycle after EDA.

Extracting input and output column for independent & dependent variable:

In [28]:

```
x=df.iloc[:,0:2]
print(x)
```

```
    cgpa     iq
0    6.8  123.0
1    5.9  106.0
2    5.3  121.0
3    7.4  132.0
4    5.8  142.0
..   ...    ...
95   4.3  200.0
96   4.4   42.0
97   6.7  182.0
98   6.3  103.0
99   6.2  113.0

[100 rows x 2 columns]
```

In [29]:

```
y=df.iloc[:,-1]
```

In [29]:

```
y=df.iloc[:,-1]
print(y)
```

```
0     1
1     0
2     0
3     1
4     0
     ..
95    0
96    0
97    1
98    1
99    1
Name: placement, Length: 100, dtype: int64
```

Train -test split:

In [30]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.1)   #adding values in variable
```

Train the model:

In [31]:

```
from sklearn.linear_model import LogisticRegression
model=LogisticRegression()
```

In [32]:

```
model.fit(x_train,y_train)
```

Out[32]:

```
▾ LogisticRegression
LogisticRegression()
```

Evlauting Model : Finding accuracy, how is the model ?

In [34]:

```
x_test=np.array(x_test)
```

In [35]:

```
#But first lets simple predict
y_predict=model.predict(x_test)         # we train model using x_train but testing model using x_test
print(y_predict)
```

```
[0 0 0 0 0 1 1 0 0 1]
```

```
C:\Users\USER\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarning: X does not have valid feature names
, but LogisticRegression was fitted with feature names
  warnings.warn(
```

In [36]:

```
print(y_test)
```

```
92    0
84    0
86    0
23    0
15    0
90    1
74    1
19    0
59    0
38    1
Name: placement, dtype: int64
```

Its look like our model is predicting with 100% accuracy

In [38]:

```
from sklearn.metrics import accuracy_score
```

In [40]:

```
accuracy_score(y_test,y_predict)
```

Out[40]:

```
1.0
```

In [48]:

```
#lets say now, one student come with cgpa 4.3
cgpa1=[[4.3,0]]
y_predict=model.predict(cgpa1)
```

```
C:\Users\USER\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarning: X does not have valid feature names
, but LogisticRegression was fitted with feature names
  warnings.warn(
```

In [49]:

```
print(y_predict)
```

```
[0]
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js