Iris dataset

In [1]:
```
import pandas as pd
import numpy as np
iris=pd.read_csv("Iris.csv")
iris.head()
```

Out[1]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	3	4.7	NaN	1.3	0.2	Iris-setosa
4	4	4.6	3.1	1.5	0.2	Iris-setosa

In [2]:
```
# setosa-0
#-versicolor-1
#-verginica-2
iris.isna().sum()
```

Out[2]:
```
Id               0
SepalLengthCm    0
SepalWidthCm     1
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

In [3]:
```
iris.dropna(inplace=True)
```

In [4]:
```
iris.isnull().sum()
```

Out[4]:
```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

In [5]:
```
#map()- mapping to a particular value
iris.Species.unique()
```

Out[5]:
```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
```

In [6]:
```
iris['mapped']=iris.Species.map({'Iris-setosa':0, 'Iris-versicolor':1, 'Iris-virginica'
iris.head()
```

Out[6]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species	mapped
0	1	5.1	3.5	1.4	0.2	Iris-setosa	0
1	2	4.9	3.0	1.4	0.2	Iris-setosa	0
2	3	4.7	3.2	1.3	0.2	Iris-setosa	0
4	4	4.6	3.1	1.5	0.2	Iris-setosa	0
5	5	5.0	3.6	1.4	0.2	Iris-setosa	0

In [7]:
```
iris.dtypes
```

Out[7]:
```
Id                int64
SepalLengthCm     float64
SepalWidthCm      float64
PetalLengthCm     float64
PetalWidthCm      float64
Species            object
mapped              int64
dtype: object
```

In [8]:
```
x=iris[['SepalLengthCm','SepalWidthCm','PetalLengthCm','PetalWidthCm']]
x.head()
```

Out[8]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
4	4.6	3.1	1.5	0.2
5	5.0	3.6	1.4	0.2

In [9]:
```
y=iris['mapped']
y.head()
```

Out[9]:
```
0    0
1    0
2    0
4    0
5    0
Name: mapped, dtype: int64
```

In [20]:
```
#- Training and Validation Data
iris.shape
```

Out[20]:
```
(150, 7)
```

In [64]:
```
#-slicing
#-scikit learn class train_test_split
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
```

In [65]:
```
x_train.shape
```

Out[65]:
```
(120, 4)
```

In [66]:
```
from sklearn.linear_model import LinearRegression
```

In [67]:
```
reg=LinearRegression()
```

In [68]:
```
reg.fit(x_train,y_train)
```

Out[68]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)
```

In [69]:
```
accuracy=round(reg.score(x_train,y_train)*100,3)
print(accuracy)
```

```
92.247
```

KNN ALGO (K=5 default)

In [70]:
```
from sklearn.neighbors import KNeighborsClassifier
kmodel=KNeighborsClassifier(n_neighbors=3)
kmodel.fit(x_train,y_train)
#.fit()- to train the model  kmodel.fit(x_train,y_train)
#.predict()- to predict for test model
#.score()- to find accuracy
# .ravel()-used to convert in 1-D array

y_pred=kmodel.predict(x_test)
y_pred
```

Out[70]:
```
array([2, 0, 1, 0, 1, 0, 0, 0, 1, 1, 2, 0, 1, 1, 2, 2, 2, 0, 0, 1, 0, 2,
       2, 2, 0, 2, 0, 2, 2, 0], dtype=int64)
```

In [71]:
```
(y_test==y_pred).sum()/len(x_test)
```

Out[71]:
```
1.0
```

In [72]:
```
kmodel.score(x_test,y_test)
```

Out[72]:
```
1.0
```

In [76]:
```
kmodel.predict([[1.2,1.3,3.2,3.1],[2.1,1.2,4.7,4.4]])
```

Out[76]:
```
array([1, 2], dtype=int64)
```

In [79]:
```
import numpy as np
plabel=np.array(['setosa','versicolor','virginica'])
plabel[kmodel.predict([[1.2,1.3,3.2,3.1],[2.1,1.2,4.7,4.4]])]
```

Out[79]:
```
array(['versicolor', 'virginica'], dtype='<U10')
```

-Cross Validation by doing training & prediction 10 times(cv=10) and calc avg

In [73]:
```
from sklearn.model_selection import cross_val_score
```

In [74]:
```
cv=cross_val_score(kmodel,x_test,y_test,cv=10)
```

```
C:\Applications\lib\site-packages\sklearn\model_selection\_split.py:652: Warning: The
least populated class in y has only 7 members, which is too few. The minimum number of
members in any class cannot be less than n_splits=10.
  % (min_groups, self.n_splits)), Warning)
```

In [75]:
```
np.mean(cv)
```

Out[75]:
```
1.0
```