DataScience Basics

September 1, 2019

[3]:
```
#Importing iris data from sklearn
from sklearn.datasets import load_iris
iris=load_iris()
type(iris)
#iris is a Buch data type.Bunch data type is a specila data type in sklearn
```

[3]:
```
sklearn.utils.Bunch
```

[4]:
```
#printing features' name of iris(e.g.sepal length(cm),sepal width(cm),petal␣
 ↪length(cm),petal width(cm))
print(iris.feature_names)
```

```
['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width
(cm)']
```

[5]:
```
#printing output(setosa,versicolor,virginica)
print(iris.target_names)
```

```
['setosa' 'versicolor' 'virginica']
```

[8]:
```
#There are three outputs which are represented as numericals(0,1,2)
print(iris.target)
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
```

iris.target is a vector

[9]:
```
#printing the length of feature_names in cm
print(iris.data)
```

```
[[5.1 3.5 1.4 0.2]
 [4.9 3.  1.4 0.2]
 [4.7 3.2 1.3 0.2]
```

```
[4.6 3.1 1.5 0.2]
[5.  3.6 1.4 0.2]
[5.4 3.9 1.7 0.4]
[4.6 3.4 1.4 0.3]
[5.  3.4 1.5 0.2]
[4.4 2.9 1.4 0.2]
[4.9 3.1 1.5 0.1]
[5.4 3.7 1.5 0.2]
[4.8 3.4 1.6 0.2]
[4.8 3.  1.4 0.1]
[4.3 3.  1.1 0.1]
[5.8 4.  1.2 0.2]
[5.7 4.4 1.5 0.4]
[5.4 3.9 1.3 0.4]
[5.1 3.5 1.4 0.3]
[5.7 3.8 1.7 0.3]
[5.1 3.8 1.5 0.3]
[5.4 3.4 1.7 0.2]
[5.1 3.7 1.5 0.4]
[4.6 3.6 1.  0.2]
[5.1 3.3 1.7 0.5]
[4.8 3.4 1.9 0.2]
[5.  3.  1.6 0.2]
[5.  3.4 1.6 0.4]
[5.2 3.5 1.5 0.2]
[5.2 3.4 1.4 0.2]
[4.7 3.2 1.6 0.2]
[4.8 3.1 1.6 0.2]
[5.4 3.4 1.5 0.4]
[5.2 4.1 1.5 0.1]
[5.5 4.2 1.4 0.2]
[4.9 3.1 1.5 0.2]
[5.  3.2 1.2 0.2]
[5.5 3.5 1.3 0.2]
[4.9 3.6 1.4 0.1]
[4.4 3.  1.3 0.2]
[5.1 3.4 1.5 0.2]
[5.  3.5 1.3 0.3]
[4.5 2.3 1.3 0.3]
[4.4 3.2 1.3 0.2]
[5.  3.5 1.6 0.6]
[5.1 3.8 1.9 0.4]
[4.8 3.  1.4 0.3]
[5.1 3.8 1.6 0.2]
[4.6 3.2 1.4 0.2]
[5.3 3.7 1.5 0.2]
[5.  3.3 1.4 0.2]
[7.  3.2 4.7 1.4]
```

```
[6.4 3.2 4.5 1.5]
[6.9 3.1 4.9 1.5]
[5.5 2.3 4.  1.3]
[6.5 2.8 4.6 1.5]
[5.7 2.8 4.5 1.3]
[6.3 3.3 4.7 1.6]
[4.9 2.4 3.3 1. ]
[6.6 2.9 4.6 1.3]
[5.2 2.7 3.9 1.4]
[5.  2.  3.5 1. ]
[5.9 3.  4.2 1.5]
[6.  2.2 4.  1. ]
[6.1 2.9 4.7 1.4]
[5.6 2.9 3.6 1.3]
[6.7 3.1 4.4 1.4]
[5.6 3.  4.5 1.5]
[5.8 2.7 4.1 1. ]
[6.2 2.2 4.5 1.5]
[5.6 2.5 3.9 1.1]
[5.9 3.2 4.8 1.8]
[6.1 2.8 4.  1.3]
[6.3 2.5 4.9 1.5]
[6.1 2.8 4.7 1.2]
[6.4 2.9 4.3 1.3]
[6.6 3.  4.4 1.4]
[6.8 2.8 4.8 1.4]
[6.7 3.  5.  1.7]
[6.  2.9 4.5 1.5]
[5.7 2.6 3.5 1. ]
[5.5 2.4 3.8 1.1]
[5.5 2.4 3.7 1. ]
[5.8 2.7 3.9 1.2]
[6.  2.7 5.1 1.6]
[5.4 3.  4.5 1.5]
[6.  3.4 4.5 1.6]
[6.7 3.1 4.7 1.5]
[6.3 2.3 4.4 1.3]
[5.6 3.  4.1 1.3]
[5.5 2.5 4.  1.3]
[5.5 2.6 4.4 1.2]
[6.1 3.  4.6 1.4]
[5.8 2.6 4.  1.2]
[5.  2.3 3.3 1. ]
[5.6 2.7 4.2 1.3]
[5.7 3.  4.2 1.2]
[5.7 2.9 4.2 1.3]
[6.2 2.9 4.3 1.3]
[5.1 2.5 3.  1.1]
```

```
[5.7 2.8 4.1 1.3]
[6.3 3.3 6.  2.5]
[5.8 2.7 5.1 1.9]
[7.1 3.  5.9 2.1]
[6.3 2.9 5.6 1.8]
[6.5 3.  5.8 2.2]
[7.6 3.  6.6 2.1]
[4.9 2.5 4.5 1.7]
[7.3 2.9 6.3 1.8]
[6.7 2.5 5.8 1.8]
[7.2 3.6 6.1 2.5]
[6.5 3.2 5.1 2. ]
[6.4 2.7 5.3 1.9]
[6.8 3.  5.5 2.1]
[5.7 2.5 5.  2. ]
[5.8 2.8 5.1 2.4]
[6.4 3.2 5.3 2.3]
[6.5 3.  5.5 1.8]
[7.7 3.8 6.7 2.2]
[7.7 2.6 6.9 2.3]
[6.  2.2 5.  1.5]
[6.9 3.2 5.7 2.3]
[5.6 2.8 4.9 2. ]
[7.7 2.8 6.7 2. ]
[6.3 2.7 4.9 1.8]
[6.7 3.3 5.7 2.1]
[7.2 3.2 6.  1.8]
[6.2 2.8 4.8 1.8]
[6.1 3.  4.9 1.8]
[6.4 2.8 5.6 2.1]
[7.2 3.  5.8 1.6]
[7.4 2.8 6.1 1.9]
[7.9 3.8 6.4 2. ]
[6.4 2.8 5.6 2.2]
[6.3 2.8 5.1 1.5]
[6.1 2.6 5.6 1.4]
[7.7 3.  6.1 2.3]
[6.3 3.4 5.6 2.4]
[6.4 3.1 5.5 1.8]
[6.  3.  4.8 1.8]
[6.9 3.1 5.4 2.1]
[6.7 3.1 5.6 2.4]
[6.9 3.1 5.1 2.3]
[5.8 2.7 5.1 1.9]
[6.8 3.2 5.9 2.3]
[6.7 3.3 5.7 2.5]
[6.7 3.  5.2 2.3]
[6.3 2.5 5.  1.9]
```

```
 [6.5 3.  5.2 2. ]
 [6.2 3.4 5.4 2.3]
 [5.9 3.  5.1 1.8]]
```

iris.data is a matrix

[10]:
```
print(type(iris.data))
```

```
<class 'numpy.ndarray'>
```

iris.data is a numpy array,numpy array must be numerical values

[11]:
```
print(type(iris.target))
```

```
<class 'numpy.ndarray'>
```

[12]:
```
#X stores input data
X=iris.data
#y stores output data
y=iris.target
```

X stores input values(feature_names) and it is a matrix, y stores output values(target_names) and it is a vector

Fitting a Machine learning Model(KNN algorithm)

[16]:
```
#importing KNNClassifier from sklearn
from sklearn.neighbors import KNeighborsClassifier
#n_neighbors=1,remaining parameters are default
knn=KNeighborsClassifier(n_neighbors=1)
print(knn)
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=1, p=2,
                     weights='uniform')
```

[17]:
```
#fitting the model
knn.fit(X,y)
```

[17]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=1, p=2,
                     weights='uniform')
```

Predicting values

[20]:
```
#Predicting the value([2,4,3,1])
Prediction=knn.predict([[2,4,3,1]])
print(type(Prediction))
```

```
<class 'numpy.ndarray'>
```

[21]:
```
#Predicted output is 0(setosa) when n_neighbors=1
print(Prediction)
```

```
[0]
```

[23]:
```
print(iris.target_names)
```

```
['setosa' 'versicolor' 'virginica']
```

Here setosa=0,versicolor=1,virginica=2

[25]:
```
Prediction=knn.predict([[2,4,3,1],[4,6,5,3]])
print(Prediction)
```

```
[0 2]
```

Now we are using n_neighbors=5

[27]:
```
knn5=KNeighborsClassifier(n_neighbors=5)
knn5.fit(X,y)
```

[27]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                     weights='uniform')
```

[29]:
```
Prediction=knn5.predict([[2,4,3,1]])
print(Prediction)
```

```
[0]
```

[31]:
```
Prediction=knn5.predict([[2,4,3,1],[4,6,5,3]])
print(Prediction)
```

```
[0 1]
```

[33]:
```
knn8=KNeighborsClassifier(n_neighbors=8)
knn8.fit(X,y)
```

[33]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=8, p=2,
                     weights='uniform')
```

[35]:
```
Prediction=knn8.predict([[2,4,3,1],[4,6,5,3]])
print(Prediction)
```

```
[0 2]
```

[47]:
```
#giving the value of n_neighbors from 1 to 30 and predicting the data
numbers=[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30]
for i in numbers:
    knni=KNeighborsClassifier(n_neighbors=i)
    knni.fit(X,y)
    Prediction=knni.predict([[2,4,3,1]])
    print(Prediction)
```

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

[48]:
```
#giving the value of n_neighbors from 1 to 30 and predicting the data using for␣
 ↪loop
numbers=[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30]
for i in numbers:
    knni=KNeighborsClassifier(n_neighbors=i)
    knni.fit(X,y)
    Prediction=knni.predict([[2,4,3,1],[4,6,5,3]])
    print(Prediction)
```

```
[0 2]
[0 2]
[0 2]
[0 1]
[0 1]
[0 1]
[0 2]
[0 2]
[0 2]
[0 2]
[0 2]
[0 2]
[0 2]
[0 2]
[0 2]
[0 2]
[0 2]
[0 2]
[0 2]
[0 2]
[0 2]
[0 2]
[0 2]
[0 2]
[0 2]
[0 2]
[0 2]
[0 2]
[0 2]
[0 2]
```

[54]:
```
numbers=[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30]
for i in numbers:
    knni=KNeighborsClassifier(n_neighbors=i)
    knni.fit(X,y)
    Prediction=knni.predict([[2,4,3,1],[4,6,5,3]])
    print(Prediction)
```

```
[0 2]
[0 2]
[0 2]
[0 1]
[0 1]
[0 1]
[0 2]
[0 2]
```

```
[0 2]
[0 2]
[0 2]
[0 2]
[0 2]
[0 2]
[0 2]
[0 2]
[0 2]
[0 2]
[0 2]
[0 2]
[0 2]
[0 2]
[0 2]
[0 2]
[0 2]
[0 2]
[0 2]
[0 2]
[0 2]
[0 2]
```

[58]:
```
#giving the value of n_neighbors from 1 to 30 and predicting the data using for␣
 ↪loop
for i in range(1,31):
    knni=KNeighborsClassifier(n_neighbors=i)
    knni.fit(X,y)
    Prediction=knni.predict([[2,4,3,1],[4,6,5,3]])
    print(Prediction)
```

```
[0 2]
[0 2]
[0 2]
[0 1]
[0 1]
[0 1]
[0 2]
[0 2]
[0 2]
[0 2]
[0 2]
[0 2]
[0 2]
[0 2]
[0 2]
[0 2]
```

```
[0 2]
[0 2]
[0 2]
[0 2]
[0 2]
[0 2]
[0 2]
[0 2]
[0 2]
[0 2]
[0 2]
[0 2]
[0 2]
[0 2]
```