INDIAN INSTITUTE OF TECHNOLOGY MADRAS • सिद्धिर्भवति कर्मजा •

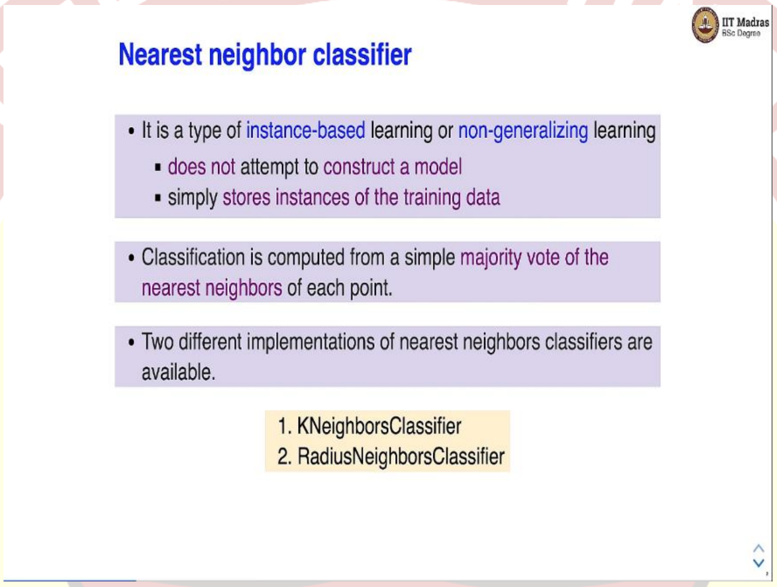
IIT Madras

ONLINE DEGREE

Machine Learning Program
Professor. Dr. Ashish Tendulkar
Indian Institute of Technology, Madras
K - Nearest Neighbors

Namaste! welcome to the next video of Machine Learning Practice Course. In this video, we will discuss how to implement K-Nearest Neighbor with sklearn. We will mostly be doing this discussion in the context of classification.

(Refer Slide Time: 00:26)



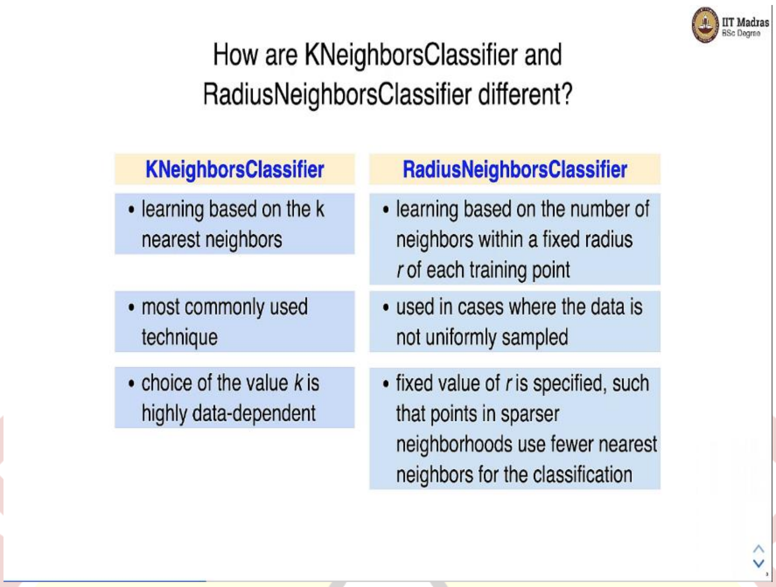
Nearest neighbor classifier

- It is a type of instance-based learning or non-generalizing learning
  - does not attempt to construct a model
  - simply stores instances of the training data
- Classification is computed from a simple majority vote of the nearest neighbors of each point.
- Two different implementations of nearest neighbors classifiers are available.

1. KNeighborsClassifier
2. RadiusNeighborsClassifier

So, as you know nearest neighbor classifier is a type of instance-based learning or non-generalizing learning. It does not attempt to construct a model, but it simply stores instances of the training data. Classification is computed from a simple majority vote of the nearest neighbor of each data point. There are two different implementations of nearest neighbor classifiers in sklearn. One is KNeighborsClassifier and second is RadiusNeighborsClassifier.

(Refer Slide Time: 01:00)



IIT Madras
BSc Degree

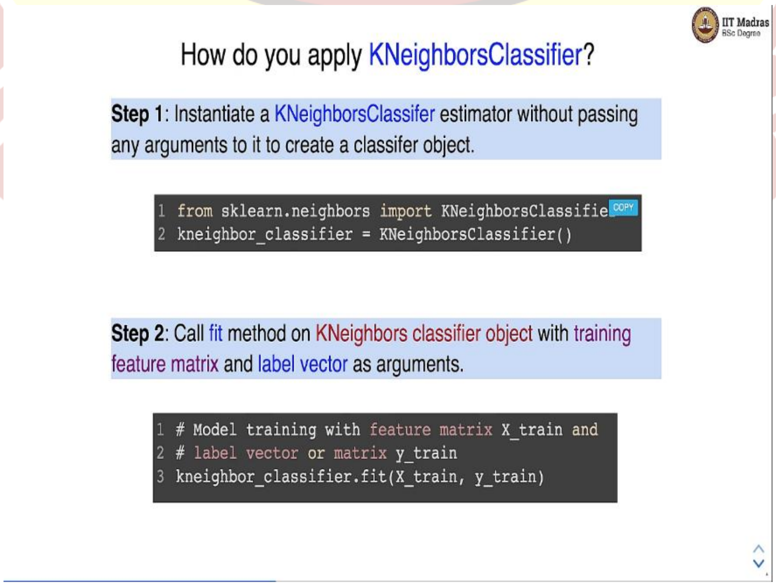
How are KNeighborsClassifier and RadiusNeighborsClassifier different?

KNeighborsClassifier	RadiusNeighborsClassifier
• learning based on the k nearest neighbors	• learning based on the number of neighbors within a fixed radius *r* of each training point
• most commonly used technique	• used in cases where the data is not uniformly sampled
• choice of the value *k* is highly data-dependent	• fixed value of *r* is specified, such that points in sparser neighborhoods use fewer nearest neighbors for the classification

So, let us see, how KNeighborsClassifier and RadiusNeighborsClassifiers differ. So, KNeighborsClassifier learns based on k-nearest neighbors, whereas, the RadiusNeighborsClassifier learn based on the number of neighbors within a fixed radius r of each training point. KNeighborsClassifier is most commonly used techniques, whereas, RadiusNeighborsClassifier is used in cases where the data is not uniformly sampled.

In KNeighborsClassifier, the choice of the value of k is highly data-dependent, whereas, in RadiusNeighborsClassifier, we used fixed value of r, such that the points in sparser neighborhood use fewer nearest neighbors for the classification.

(Refer Slide Time: 01:45)



IIT Madras
BSc Degree

How do you apply KNeighborsClassifier?

Step 1: Instantiate a KNeighborsClassifer estimator without passing any arguments to it to create a classifer object.

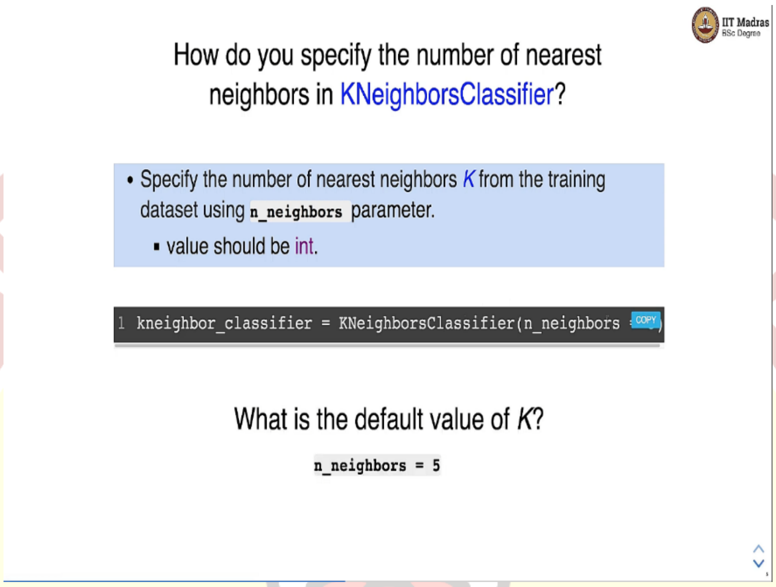
```
from sklearn.neighbors import KNeighborsClassifie
kneighbor_classifier = KNeighborsClassifier()
```

Step 2: Call fit method on KNeighbors classifier object with training feature matrix and label vector as arguments.

```
# Model training with feature matrix X_train and
# label vector or matrix y_train
kneighbor_classifier.fit(X_train, y_train)
```

Let us see, how to apply KNeighborsClassifier. We instantiate a KNeighborsClassifier without passing any argument to create a classifier object. So, we import KNeighborsClassifier from sklearn.neighbors module. And in the second step, we call fit method on the classifier object with training feature matrix and label vector as arguments.

(Refer Slide Time: 02:09)



How do you specify the number of nearest neighbors in KNeighborsClassifier?

- Specify the number of nearest neighbors *K* from the training dataset using `n_neighbors` parameter.
  - value should be int.

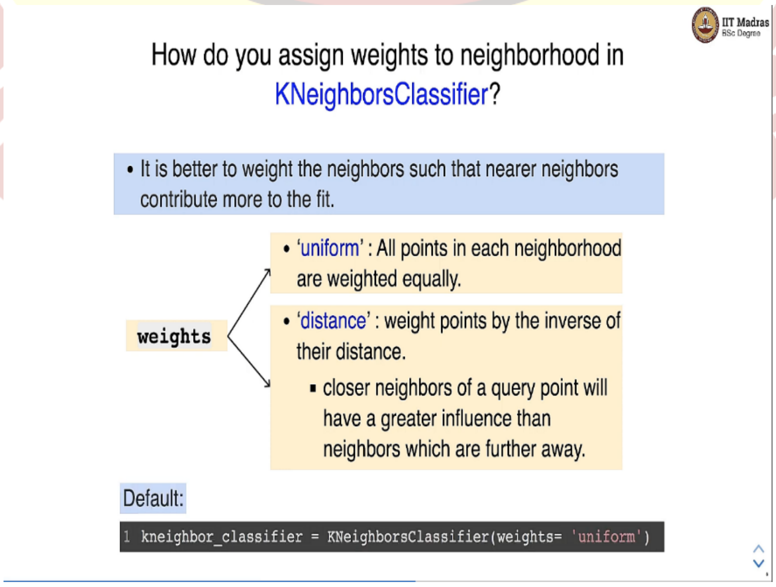
```
kneighbor_classifier = KNeighborsClassifier(n_neighbors
```

What is the default value of *K*?

`n_neighbors = 5`

Let us say, how to specify the number of nearest neighbor in KNeighborsClassifier. So, there is a parameter called n_neighbors that we need to basically set. So, here k while instantiating object of KNeighborsClassifier, we set n neighbors to the desired value and the default value of nearest neighbor is 5.

(Refer Slide Time: 02:34)



How do you assign weights to neighborhood in KNeighborsClassifier?

- It is better to weight the neighbors such that nearer neighbors contribute more to the fit.

`weights`

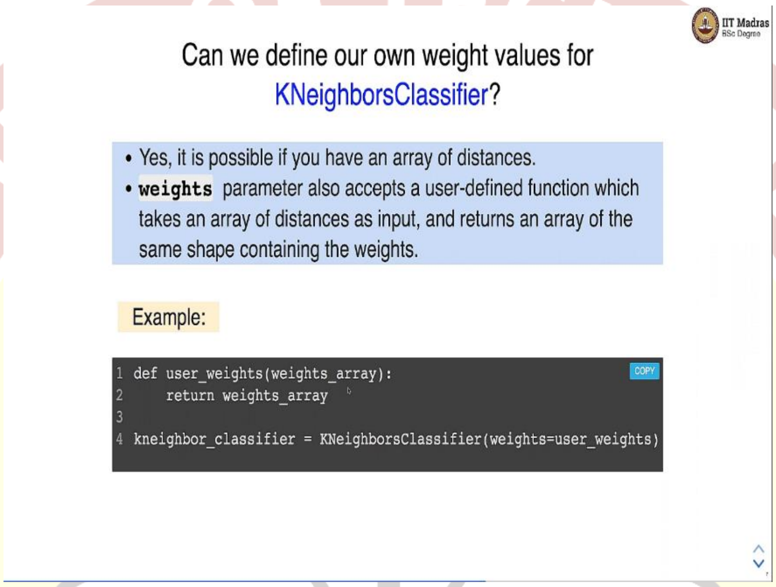
- 'uniform' : All points in each neighborhood are weighted equally.
- 'distance' : weight points by the inverse of their distance.
  - closer neighbors of a query point will have a greater influence than neighbors which are further away.

Default:

```
kneighbor_classifier = KNeighborsClassifier(weights= 'uniform')
```

So, there is a possibility that we can assign weights to the neighborhood. So, the idea here is that the higher weights are assigned to the nearest neighbor , so that they contribute more towards the labelling of the new point. The weights can be uniform, where all points in the neighbor hood are weighted equally or weight can be based on the distance. The points that are closer will have greater influence and the points that are further away. By default, KNeighborsClassifier uses uniform weight.

(Refer Slide Time: 03:09)



Can we define our own weight values for KNeighborsClassifier?

- Yes, it is possible if you have an array of distances.
- `weights` parameter also accepts a user-defined function which takes an array of distances as input, and returns an array of the same shape containing the weights.

Example:

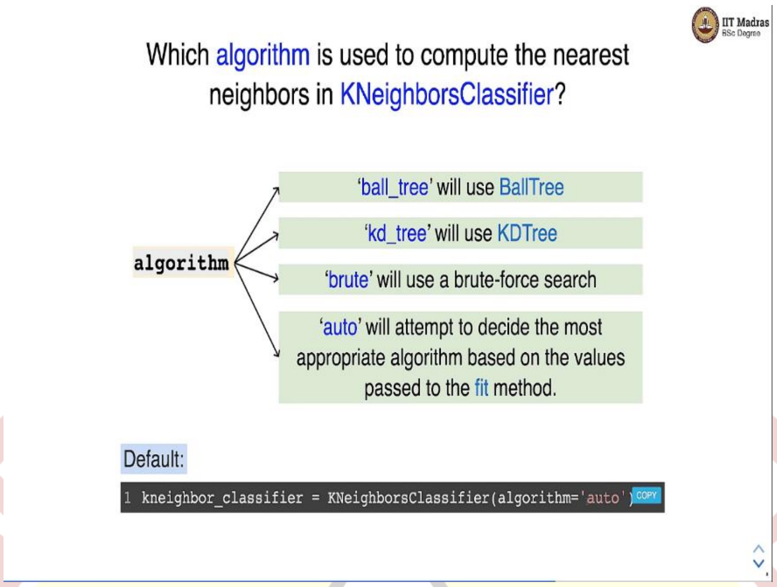
```
def user_weights(weights_array):
    return weights_array

kneighbor_classifier = KNeighborsClassifier(weights=user_weights)
```

So, can we define our own weight? Yes, it is quite possible that we can define our own weights as a function of the distance. So, the weight parameter also accepts a user-defined function, which takes an array of distance as input, and returns an array of the same shape containing the weights.

So, here we have shown an example of a user-defined function. So, here in weight, we are specifying a user-defined function, which is user _weights, which we have defined over here. It takes an input, which is array of weights. As an example, we are just returning the weight array that we receive in the argument, but it is possible that you can perform some computation over here and return an array of the same shape as the argument.

(Refer Slide Time: 03:59)



IIT Madras
BSc Degree

Which algorithm is used to compute the nearest neighbors in KNeighborsClassifier?

`algorithm`
- 'ball_tree' will use BallTree
- 'kd_tree' will use KDTree
- 'brute' will use a brute-force search
- 'auto' will attempt to decide the most appropriate algorithm based on the values passed to the fit method.

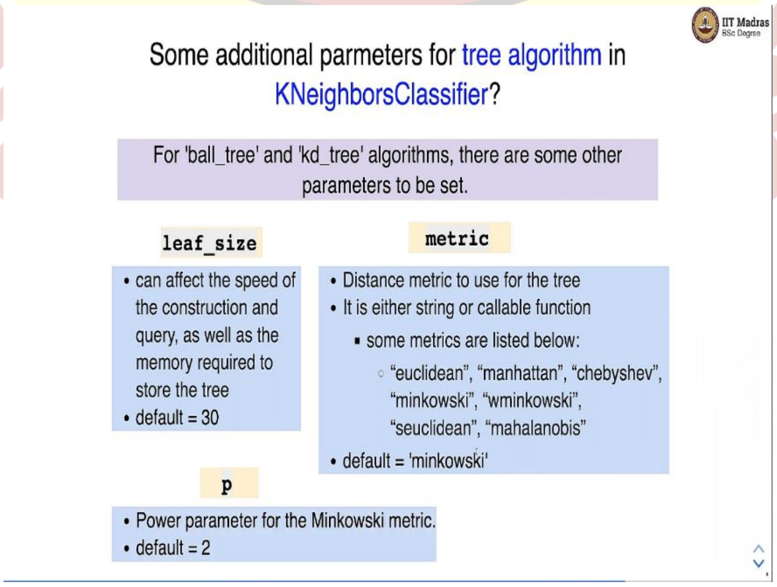
Default:

```
kneighbor_classifier = KNeighborsClassifier(algorithm='auto')
```

So, let us see, which algorithms are available in sklearn to compute the nearest neighbor. Because nearest neighbor computation is the most expensive operation in the KNeighborsClassifier. We can use algorithm like ball tree or kd tree or we can use a brute-force search.

So, ball tree is specified with ball _tree, kd tree specified with kd _tree and brute-force is specified with brute in the parameter algorithm. There is also an additional value that algorithm parameter takes which is auto that decides the most appropriate algorithm based on the values passed in the fit method. So, by default, the algorithm parameter is set to auto.

(Refer Slide Time: 04:46)



IIT Madras
BSc Degree

Some additional parmeters for tree algorithm in KNeighborsClassifier?

For 'ball_tree' and 'kd_tree' algorithms, there are some other parameters to be set.

`leaf_size`
- can affect the speed of the construction and query, as well as the memory required to store the tree
- default = 30

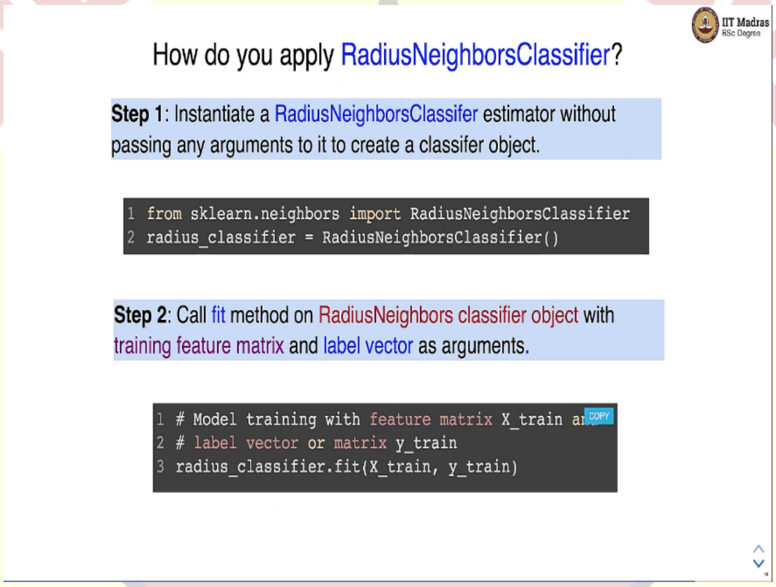
`metric`
- Distance metric to use for the tree
- It is either string or callable function
  - some metrics are listed below:
    - "euclidean", "manhattan", "chebyshev", "minkowski", "wminkowski", "seuclidean", "mahalanobis"
- default = 'minkowski'

`p`
- Power parameter for the Minkowski metric.
- default = 2

There are some additional parameters for the tree algorithm. For ball_tree and kd_tree algorithms, there are some parameters like leaf _size that can affect the speed of the construction and querying the tree as well as the memory required to store the tree. The default leaf size is 30.

We also need to specify the distance metric for the tree. So, it is either string or a callable function. There are few options that are available here, which are Euclidean, Manhattan, Chebyshev, Minkowski, Mahalanobis, etc. The default one is Minkowski with the power parameter set to 2.

(Refer Slide Time: 5:33)



How do you apply RadiusNeighborsClassifier?

Step 1: Instantiate a RadiusNeighborsClassifer estimator without passing any arguments to it to create a classifer object.

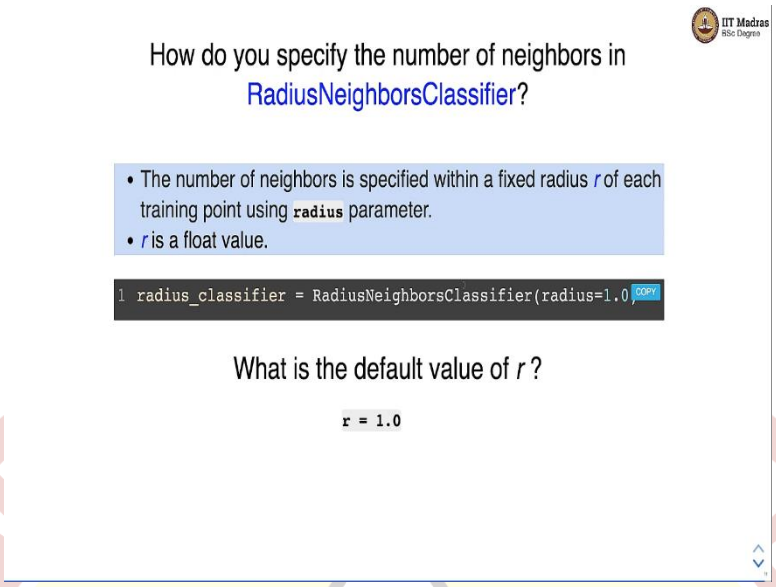
```
from sklearn.neighbors import RadiusNeighborsClassifier
radius_classifier = RadiusNeighborsClassifier()
```

Step 2: Call fit method on RadiusNeighbors classifier object with training feature matrix and label vector as arguments.

```
# Model training with feature matrix X_train an
# label vector or matrix y_train
radius_classifier.fit(X_train, y_train)
```

So, that was about KNeighborsClassifier. Let us look at the other classifier, which is RadiusNeighborsClassifier. Here, if you want to use RadiusNeighborsClassifier in the first step, you need to instantiate radius neighbors estimator and RadiusNeighborsClassifier is implemented in sklearn**.**neighbors module. In the second step, we call the fit method on the classifier object with feature matrix and label vector as arguments.

(Refer Slide Time: 06:05)



How do you specify the number of neighbors in RadiusNeighborsClassifier?

- The number of neighbors is specified within a fixed radius *r* of each training point using `radius` parameter.
- *r* is a float value.

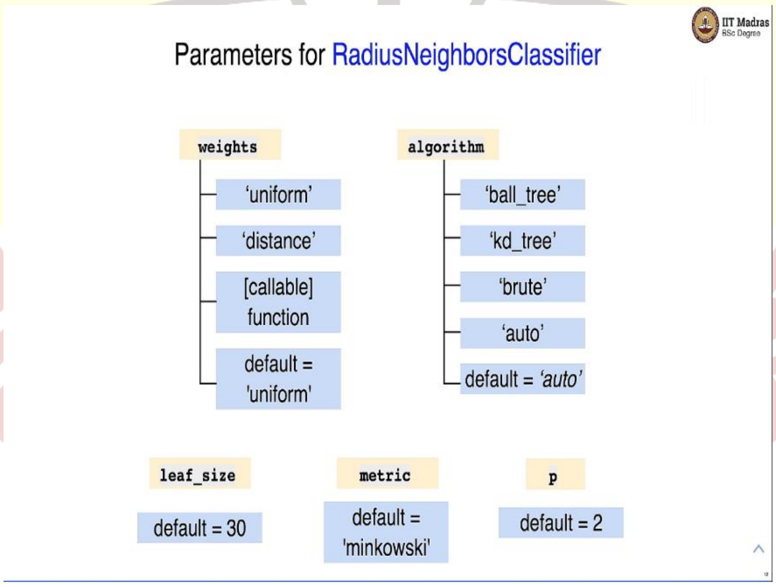
```
radius_classifier = RadiusNeighborsClassifier(radius=1.0)
```

What is the default value of *r* ?

`r = 1.0`

Let us see, how to specify the number of neighbors in RadiusNeighborsClassifier. So, as we have seen earlier, we specify instead of specifying the neighbors, we specify the fixed radius r which is a float value. And that radius is set in the parameter radius in the constructor of RadiusNeighborsClassifier. The default value of r is 1.

(Refer Slide Time: 06:33)



Parameters for RadiusNeighborsClassifier

`weights`
- 'uniform'
- 'distance'
- [callable] function
- default = 'uniform'

`algorithm`
- 'ball_tree'
- 'kd_tree'
- 'brute'
- 'auto'
- default = *'auto'*

`leaf_size`: default = 30

`metric`: default = 'minkowski'

`p`: default = 2

There are some parameters for RadiusNeighborsClassifier like weights. Again, it could be uniform, distance or it could be callable function. The default weight is uniform. And again just like KNeighborsClassifier, we can use algorithms like ball_tree, kd_tree, brute-force search and auto. Auto decides the appropriate algorithm based on the values that are passed in the fit function. The default leaf size is 30 and the default matrix is Minkowski with the value of p set to 2. So, in this video, we discussed couple of nearest neighbor classifiers implemented in sklearn.