|  |
| --- |
| 1. EXPERIMENT NO: 8 |
| 2. TITLE: **K-NEAREST NEIGHBOUR** |
| 1. LEARNING OBJECTIVES:    * Make use of Data sets in implementing the machine learning algorithms.    * Implement ML concepts and algorithms in Python |
| 1. AIM:    * Write a program to implement k-Nearest Neighbour algorithm to classify the iris data set. Print both correct and wrong predictions. Java/Python ML library classes can be used for this problem. |
| 1. THEORY:    * K-Nearest Neighbors is one of the most basic yet essential classification algorithms in Machine Learning. It belongs to the supervised learning domain and finds intense application in pattern recognition, data mining and intrusion detection.    * It is widely disposable in real-life scenarios since it is non-parametric, meaning, it does not make any underlying assumptions about the distribution of data.    * Algorithm   Input: Let m be the number of training data samples. Let p be an unknown point. Method:   1. Store the training samples in an array of data points arr[]. This means each element of this array represents a tuple (x, y). 2. for i=0 to m   Calculate Euclidean distance d(arr[i], p).   1. Make set S of K smallest distances obtained. Each of these distances correspond to an already classified data point. 2. Return the majority label among S. |

sklearn.model\_selection.train\_test\_split(*\*arrays*, *test\_size=None*, *train\_size=None*, *random\_state=None*, *shuffle=True*, *stratify=None*)[[source]](https://github.com/scikit-learn/scikit-learn/blob/0d378913b/sklearn/model_selection/_split.py#L2321)

Split arrays or matrices into random train and test subsets

*class*sklearn.neighbors.KNeighborsClassifier(*n\_neighbors=5*, *\**, *weights='uniform'*, *algorithm='auto'*, *leaf\_size=30*, *p=2*, *metric='minkowski'*, *metric\_params=None*, *n\_jobs=None*)[[source]](https://github.com/scikit-learn/scikit-learn/blob/0d378913b/sklearn/neighbors/_classification.py#L22)

Classifier implementing the k-nearest neighbors vote

scikit-learn comes with a few standard datasets, for instance the [iris](https://en.wikipedia.org/wiki/Iris_flower_data_set) and [digits](https://archive.ics.uci.edu/ml/datasets/Pen-Based+Recognition+of+Handwritten+Digits) datasets for classification and the [diabetes dataset](https://www4.stat.ncsu.edu/~boos/var.select/diabetes.html) for regression

sklearn.datasets.load\_iris(*\**, *return\_X\_y=False*, *as\_frame=False*)[[source]](https://github.com/scikit-learn/scikit-learn/blob/0d378913b/sklearn/datasets/_base.py#L520)[¶](https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_iris.html#sklearn.datasets.load_iris)

Load and return the iris dataset (classification).

The iris dataset is a classic and very easy multi-class classification dataset.

|  |  |
| --- | --- |
| Classes | 3 |
| Samples per class | 50 |
| Samples total | 150 |
| Dimensionality | 4 |
| Features | real, positive |

from sklearn.datasets import load\_iris

iris=load\_iris()

for keys **in** iris.keys() :

print(keys)

data

target

target\_names

DESCR

feature\_names

The number of neighbors is the core deciding factor. K is generally an odd number if the number of classes is 2. When K=1, then the algorithm is known as the nearest neighbor algorithm.

score(X\_test,Y\_test) is **measuring the accuracy of the model against the testing data**.

The best **possible score is 1.0** and it can be negative (because the model can be arbitrarily worse).

