*class* sklearn.neighbors.**KNeighborsClassifier**(*n\_neighbors=5*, *weights=’uniform’*, *algorithm=’auto’*, *leaf\_size=30*, *p=2*, *metric=’minkowski’*, *metric\_params=None*, *n\_jobs=1*, *\*\*kwargs*)

Классификатор реализующий выбор к-ближайших соседей:

Параметры:

n\_neighbors(n - соседей) :целое число, опциональный параметр (по умолчанию значение = 5)

Количество соседей использ. по умолчанию для к-соседних запросов

weights(веса) - вызываемый , опциональный параметр (по умолчанию ‘uniform’ - равнозначный)

Весовая функция используется для прогнозирования.Возможные значения:

uniform - все точки из каждого класса весят одинаково

distance(расстояние) - взвешивает точки путем инвертирования их дистанций. Таким образом ближайший сосед запрашиваемой точки оказывает большее влияние чем сосед который расположен дальше.

[callable]: (вызываемый) - пользовательски определенная функция которая принимает массив расстояний и возвращает массив такого же типа который уже содержит веса.

**algorithm** : {‘auto’, ‘ball\_tree’, ‘kd\_tree’, ‘brute’}, optional

“авто” - подбирает наиболее подходящий алгоритм исходя из входящих параметров , метрическое дерево” , “k - мерное дерево” - алгоритм который позволяет разбивать к- мерное метрическое пространство на меньшие части путем его рассечения гиперплоскостями, ‘brute” - брут - форс перебирает все точки метрического пространства.

Примечание: разряженные входящие данные переопределят сеттинг вышеперечисленных параметров , так что в этом случае используйте - брут-форс.

**leaf\_size(размер листа)**  : int, optional (default = 30) (целочисленный, опциональный ( по умолчанию - 30)

Размер листа относится к параметру algorithm , а конкретно к “ метрическому дереву” и “к мерное метрическое дерево” . Данный параметр может ускорить данный алгоритм - структуру данных, все зависит от памяти которая требуется для хранения этой самой структуры данных.Оптимально значение зависит от происхождения задачи.

**p** : integer, optional (default = 2) (целочисленный , опциональный (по умолчанию 2)

Параметр изменение которого приведет к изменению метрики Минковского

l1 - когда p = 1, l1 = норме пространства l1 то есть сумма модуля разности координат 2-х точек.

l2 - когда p = 2, l2 - Эвклидово пространство

и lp когда p > 2

…

**metric** : string or callable, default ‘minkowski’ (метрика, строка либо вызываемый параметр, по умолчанию “минковский”)

Метрика расстояния используется для дерева.По умолчанию метрика Минковского при p = 2 - l2.

**n\_jobs** : int, optional (default = 1) (целочисленный , опциональный параметр по умолчанию = 1)

Число параллельных вычислений для поиска соседей. Если данный параметр равен -1 то число вычислений устанавливается по количеству ядер процессора.

**Пример:**

**>>> X = [[0], [1], [2], [3]]  
 >>> y = [0, 0, 1, 1]  
 >>> from sklearn.neighbors import KNeighborsClassifier  
 >>> neigh = KNeighborsClassifier(n\_neighbors=3)  
 >>> neigh.fit(X, y)   
 KNeighborsClassifier(...)  
 >>> print(neigh.predict([[1.1]]))  
 [0]  
 >>> print(neigh.predict\_proba([[0.9]]))  
 [[ 0.66666667 0.33333333]]**

**Методы:**

[**fit**](http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html#sklearn.neighbors.KNeighborsClassifier.fit)**(X, y) - подстраивает модель, использую Х как обучающую выборку , а y как целевые значения.**

**get\_params([deep]) - считывает параметры для оценки.**

**kneighbors([X, n\_neighbors, return\_distance]) - осущ. поиск к - соседей запрашиваемой точки.**

**kneighbors\_graph([X, n\_neighbors, mode]) - просчитывает взвешенный граф для К - соседей точки Х.**

**predict(X) - прогнозирует для входящих данных принадлежность к классам.**

[**predict\_proba**](http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html#sklearn.neighbors.KNeighborsClassifier.predict_proba)**(X) - возвращает возможность оценить тестовую выборку Х.**

**score(X, y[, sample\_weight]) - возвращает среднюю точность определения меток принадлежности к классам для тестовой выборки.**

**set\_params(\*\*params) - устанавливает параметры для оценки.**

**\_\_init\_\_(*n\_neighbors=5*, *weights=’uniform’*, *algorithm=’auto’*, *leaf\_size=30*, *p=2*, *metric=’minkowski’*, *metric\_params=None*, *n\_jobs=1*, *\*\*kwargs*)**

**fit(*X*, *y*)**

**Подстраивает модель используя Х как обучающую выборку и у как целевые значения.**

**Параметры: X : {array-like, sparse matrix, BallTree, KDTree}**

**Training data. If array or matrix, shape [n\_samples, n\_features], or [n\_samples, n\_samples] if metric=’precomputed’.**

**y : {array-like, sparse matrix}**

**Target values of shape = [n\_samples] or [n\_samples, n\_outputs]**

**get\_params(*deep=True*)**

**Параметры: deep: булевый , опциональный. Если истина то вернет параметры для этой оценки и подобъекты…**

**Тип возвращаемого значеия: params: сопоставление строки - параметры сопоставляются со своими значениями**

**kneighbors(*X=None*, *n\_neighbors=None*, *return\_distance=True*)**

**Ищет к - соседей точки , Возвращает индексы и расстояния до соседей каждой точки.**

**kneighbors\_graph(X=None, n\_neighbors=None, mode=’connectivity’)**

**Просчитывает взвешенный граф к-соседей до нужной точки.**

**Параметры: X : array-like, shape (n\_query, n\_features), or (n\_query, n\_indexed) if metric == ‘precomputed’**

**Запрашиваемая точка или точки. Если отсутствует то находятся соседи для кажлй проиндексированной точки. В этом случае точка запроса не считается своим соседом.**

**n\_neighbors : int**