**Постановка задачи**: использовать метод k ближайших соседей для реализации классификации точек в двумерном пространстве

**Исходные данные**:

* ***x*** – список с координатами точек на плоскости (пример: [[1, 1], [1, 2], [2, 1], [2, 2]])
* ***y*** – список с метками классов соответствующие точкам (пример: [0, 0, 1, 1])
* ***k*** – количество просматриваемых ближайших соседей (по умолчанию: 3)
* ***classesCount*** – количество классов (по умолчанию: 2)
* ***pointsCount1, pointsCount2*** – количество точек первого и второго классов (по умолчанию 50 для обоих переменных)

Этапы разработки программы:

1. **Инициализация исходных данных**

Для каждого класса задать минимальные и максимальные значения (границы) по осям x и y:

* ***xMin1*** – минимальное значение по оси х для первого класса
* ***xMax1*** – максимальное значение по оси х для первого класса
* ***yMin1*** – минимальное значение по оси y для первого класса
* ***yMax1*** – максимальное значение по оси y для первого класса
* ***xMin2*** – минимальное значение по оси х для второго класса
* ***xMax2*** – максимальное значение по оси х для второго класса
* ***yMin2*** – минимальное значение по оси y для второго класса
* ***yMax2*** – максимальное значение по оси y для второго класса

Эти значения будут означать границы, в пределах которых будут генерироваться значения классов. Визуально их можно отобразить в виде прямоугольника

Изображение выглядит как текст, диаграмма, линия, График

Контент, сгенерированный ИИ, может содержать ошибки.

Значения для данных переменных можно задать самостоятельно, но в пределах разумного 🙃

Ваша задача для каждого класса сгенерировать pointsCount1 и pointsCount2 значений координат точек в диапазоне [xMin1, xMax1] и [yMin1, yMax1] для первого класса и в диапазоне [xMin2, xMax2], [yMin2, yMax2] для второго класса. Притом координаты должны иметь тип данных float.

Для генерации случайного нецелого числа в заданном диапазоне можно использовать функцию uniform из библиотеки random. Например, чтобы получить случайное нецелое число в диапазоне от 1 до 2, можно написать следующие строчки кода:

import random

print(random.uniform(1, 2))

# 1.1164544158573022

1. ***Разбитие исходных данных на обучающую и тестовую выборки***

После того, как были инициализированы переменные ***x*** и ***y***, необходимо разбить данные на обучающую и тестовую выборки. То есть элементы списка х необходимо разделить на 2 списка: ***x\_train*** (данные для обучения) и ***x\_test*** (данные для тестирования). Аналогичным образом необходимо поступить и для переменной ***у***: ***y\_train*** и ***y\_test***.

Переменная **p** определяет, какая часть данных из списков ***х*** и ***у*** попадёт в списки ***x\_train*** и ***y\_train*** соответственно. Например, если ***p***=0.8, то 80% случайных данных из ***x*** и ***y*** попадёт в списки ***x\_train*** и ***y\_train*** соответственно. Оставшаяся часть данных попадут в списки ***x\_test*** и ***y\_test***

Параметр ***p*** по умолчанию равен 0.8.

Ваша задача – реализовать разделение 2 исходных списков (***x*** и ***у***) на 4 (***x\_train***, ***x\_test***, ***y\_train***, ***y\_test***). Можно даже написать свою функцию для этого, назвав её, например, ***train\_test\_split***. В таком случае на входе данная функция будет принимать 2 переменных: х и у, а возвращать 4 переменных: x\_train, x\_test, y\_train, y\_test

1. ***Реализация метода k ближайших соседей***

Реализовать функцию, которая будет принимать 3 параметра:

* ***x\_train***
* ***y\_train***
* ***x\_test***

Эта функция будет возвращать

* ***y\_predict*** – рассчитанные с помощью алгоритма k ближайших соседей списки меток классов для точек из списка ***x\_test***

Рекомендуется назвать данную функцию ***fit***.

1. **Реализация функции для расчёта метрики *accuracy* для оценки точности работы метода *k* ближайших соседей**

Реализовать функцию, которая будет принимать на вход 2 параметра:

* ***y\_test***
* ***y\_predict***

Эта функция будет возвращать

* ***accuracy*** – число, обозначающее значение метрики accuracy

Accuracy рассчитывается по следующей формуле:

где ***КС*** – это количество совпадающих элементов с одинаковыми индексами в списках ***y\_test*** и ***y\_predict***;

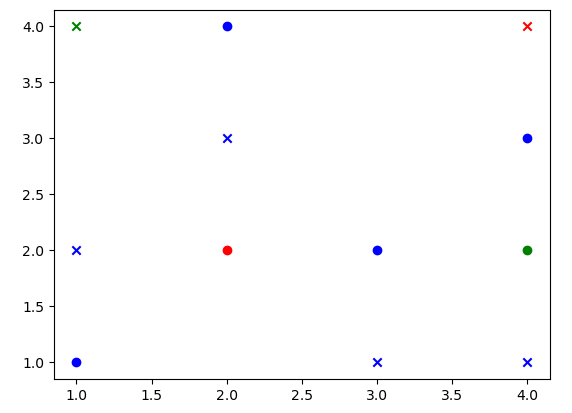
***N*** – длина списка ***y\_test*** (длина списка ***y\_predict*** тоже ***N***)

Функцию для расчёта метрики accuracy рекомендуется называть ***computeAccuracy***.

1. **Визуализировать результат работы алгоритма**

С помощью библиотеки matplotlib визуализировать результат работы метода k ближайших соседей.

Пример вывода точек на экран представлен на рисунке ниже:



Здесь можно выделить следующие виды точек:

* Синие точки-кружочки. Это точки из переменной x\_train, относящиеся к нулевому классу
* Синие точки-крестики. Это точки из переменной x\_train, относящиеся к первому классу
* Красные точки-кружочки. Это точки из переменной x\_test, относящиеся к нулевому классу, которые наш классификатор классифицировал неверно
* Красные точки-крестики. Это точки из переменной x\_test, относящиеся к первому классу, которые наш классификатор классифицировал неверно
* Зелёные точки-кружочки. Это точки из переменной x\_text, относящиеся к нулевому классу, которые наш классификатор классифицировал верно
* Зелёные точки-крестики. Это точки из переменной x\_test, относящиеся к первому классу, которые наш классификатор классифицировал верно