Отчёт

Гуляев Сергей Владимирович

группа: 501-И

Языки программирования:

- **Python**

Используемые пакеты:

- **numpy**
  Пакет для научных вычислений
- **pandas**
  Пакет для быстрого и простого хранения, структурирования и анализа данных
- **sklearn**
  Пакет содержащий простые и эффективные инструменты для сбора и анализа данных
- **bokeh**
  Пакет для создания интерактивной визуализации данных, для отображения в современных браузерах.

Оглавление

Метрические алгоритмы классификации

Пусть задано множество объектов X и конечное множество классов Y , при этом существует зависимость $y^*: X \to Y$, известная для некоторых объектов конечного множества $X^l \in X$ где l мощность множества X^l . Также пусть задана некоторая функция расстояния $\rho: X \times X \to \mathbb{R}^+ \cup \{0\}$.Требуется построить алгоритм, как можно более точно аппроксимирующий функцию $y^*(x)$ на всё множество X .

Далее в работе над данным методом в качестве функции расстояния будет использоваться функция $\rho(x, y) = \sum (x_i + y_i)^2$ а алгоритмы классификации взяты из пакета **sklearn**

Метрический алгоритм классификации с обучающей выборкой X^l относит объект x к тому классу $y \in Y$, для которого суммарный вес обучающих объектов $\Gamma_y(x, X^l)$ максимален. $\Gamma_y(x, X^l)$ называется *оценкой близости объекта x к классу y*

$$a(x; X^l) = arg\max_{y \in Y} \Gamma_y(x, X^l); \qquad \Gamma_y(x, X^l) = \sum_{i=1}^{l} [y_x^{(i)} = y] w(i, x);$$

Где функция $w(i, x)$ является коэффициентом важности соседа i для объекта x .

– Метод ближайших соседей

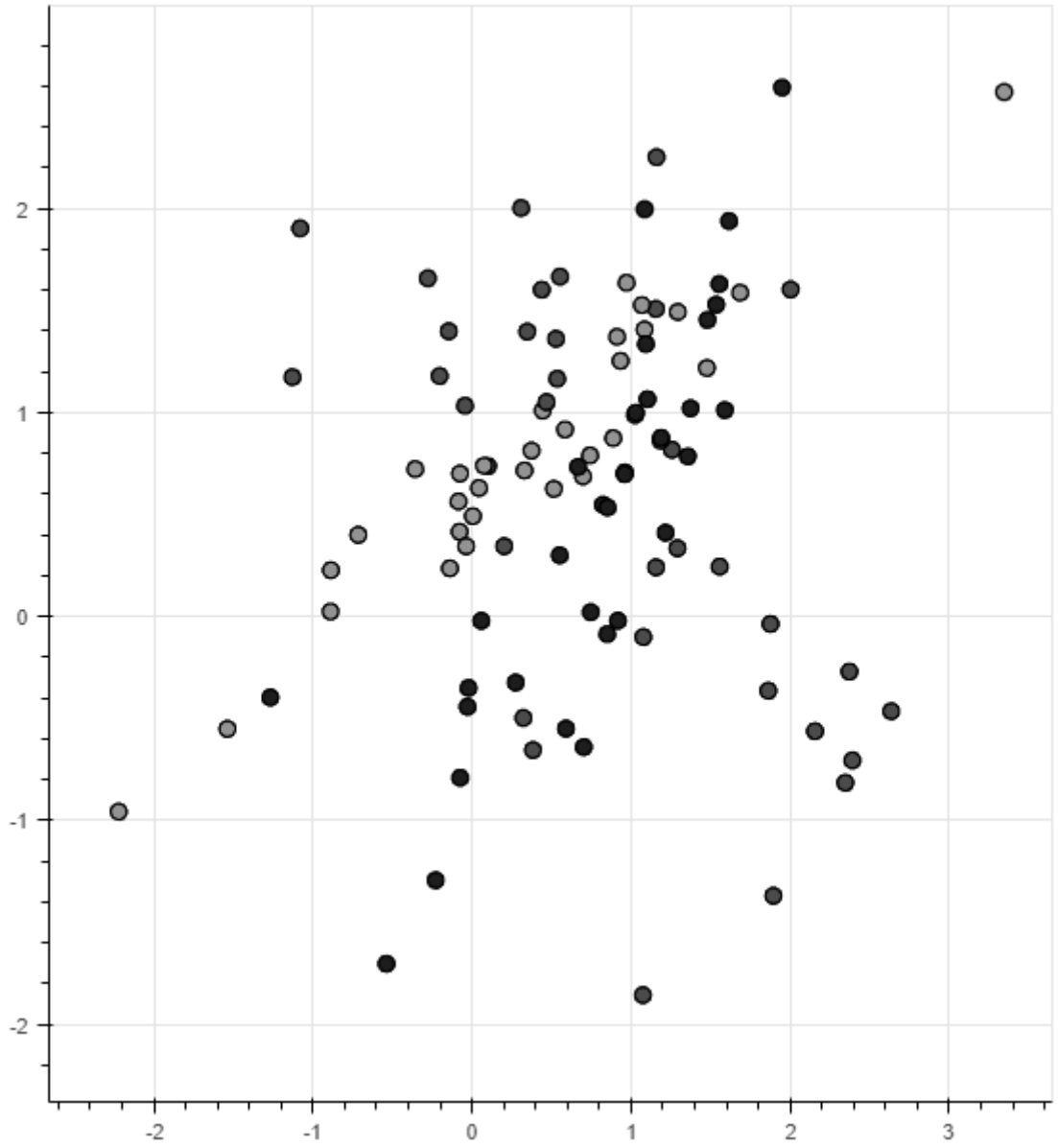
Рассмотрим алгоритм классификации, который относит объект к тому классу, элементов которого окажется больше среди k его ближайших соседей.

В пакете **sklearn** данный классификатор описан в классе **KneighborsClassifier**

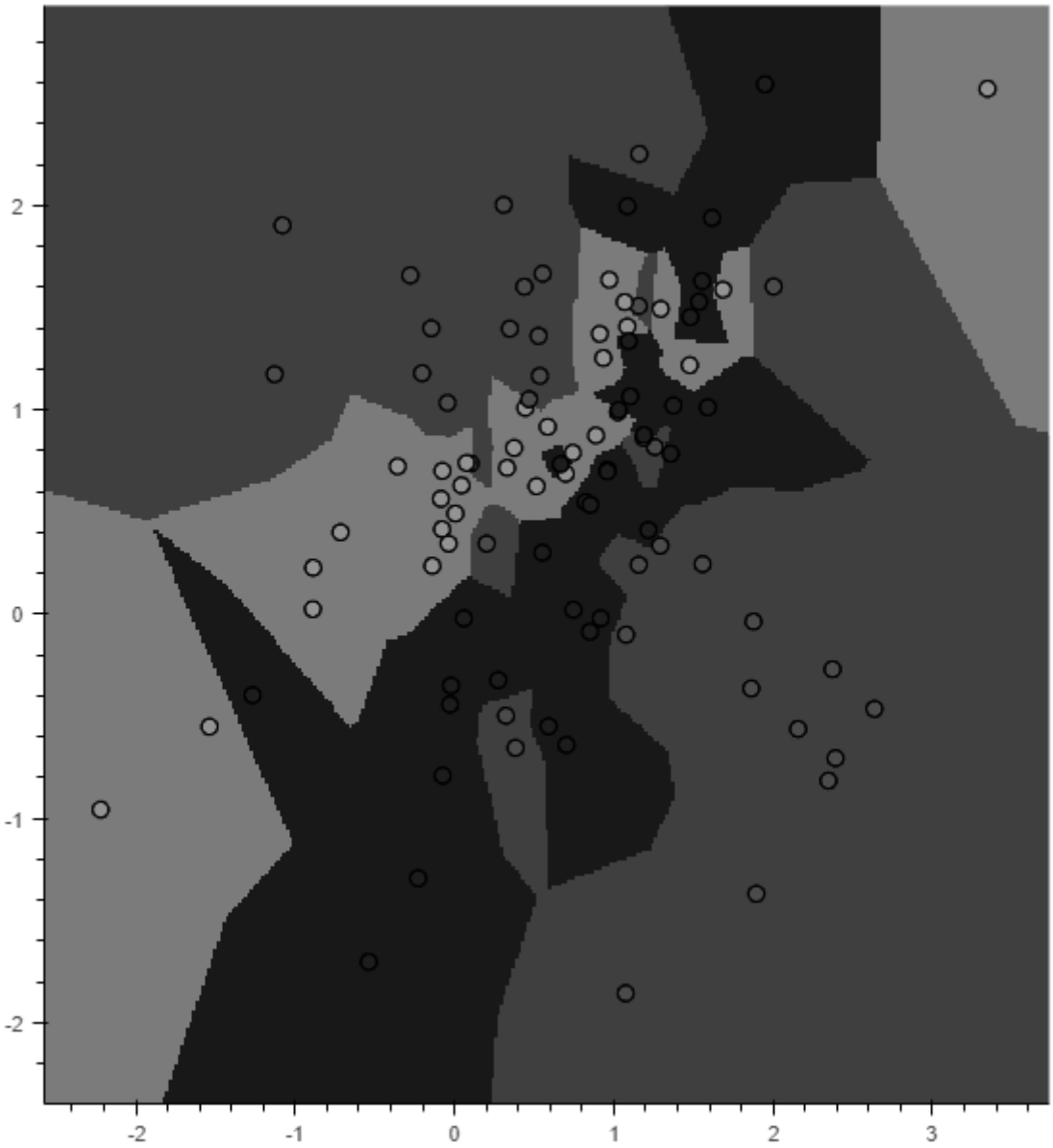
KneighborsClassifier при инициализации позволяет установить не только k **,** но также метрику, алгоритм внутренней обработки данных и функцию весов w .

```
n_neighbors,cross_validation_result = leave_one_out(X,Y)
clf = neighbors.KNeighborsClassifier(n_neighbors, weights='distance')
clf.fit(X, Y)
```

Для классификации будет использоваться данная выборка:

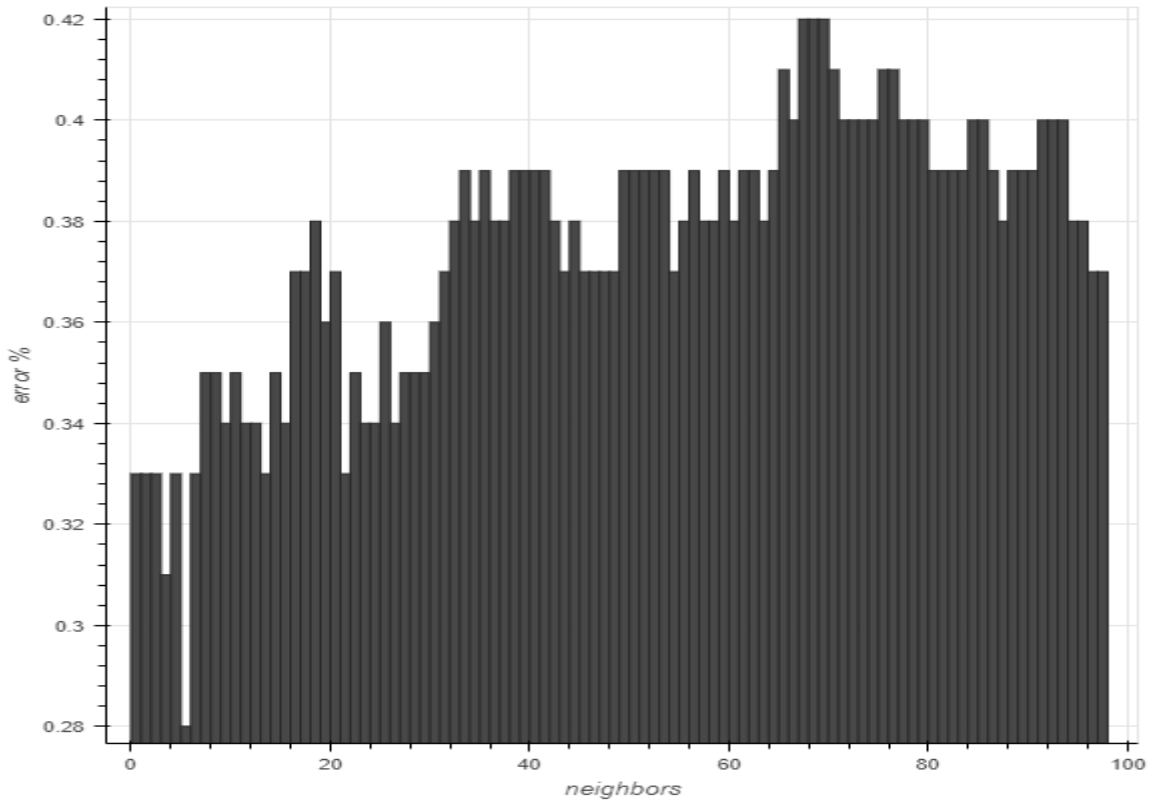


Как можно заметить по следующему рисунку при значении $k=1$, данный метод будет очень чувствителен к выбросам. Поэтому для определения оптимального значения k будет использоваться функция LOO(leave one out):



```
def leave_one_out(X,Y):
    item_amount = len(X)
    error_percentage = []
    minimal_good_neighbors = 0
    minimal_errors = len(X)
    for n_neighbors in range(1,item_amount - 1):
        errors = 0
        for i in range(item_amount):
            item = X[i]
            item_class = Y[i]
            X_t = np.delete(X,i,0)
            Y_t = np.delete(Y,i,0)
            clf = neighbors.KNeighborsClassifier(n_neighbors, weights='distance')
            clf.fit(X_t, Y_t)
            predicted_class = clf.predict(item.reshape(1, -1))
            if(predicted_class != item_class):
                errors = errors + 1
        error_percentage.append(errors/item_amount)
        if(errors<minimal_errors):
            minimal_errors = errors
            minimal_good_neighbors = n_neighbors
    return minimal_good_neighbors,error_percentage
```

После работы функции мы можем увидеть какой процент ошибок классификации на существующей выборке возникал при каждом k .



Также видим , что метод стал менее чувствителен к выбросам.