Домашнее задание №1: «Соседи и вино»

Дедлайн 1 (20 баллов): 26 февраля, 23:59
Дедлайн 2 (10 баллов): 5 марта, 23:59

Домашнее задание нужно написать на языке Python и сдать в виде одного файла. Правило именования файла: `name_surname_1.py`. Например, если вас зовут Иван Петров, то имя файла должно быть: `ivan_petrov_1.py`.

В этом задании предлагается опровергнуть миф про то, что все вина на вкус одинаковые, и заодно разобраться с классификатором k-ближайших соседей. По ссылке[1] находятся данные, описывающие химический состав трёх вин из некоторого региона Италии. Первая колонка каждой строки — идентификатор вина, который может быть равен 1, 2 или 3. Значения остальных колонок указаны в заголовке файла.

1 Прежде чем приступить к написанию классификатора, реализуйте функцию разбиения выборки на обучающую и тестовую. Функция должна принимать прочитанные из файла данные: матрицу признаков X, вектор меток класса y и соотношение, в котором нужно разбить выборку.

На Python функцию можно записать так:

```
def train_test_split(X, y, ratio):
    # ...
    return X_train, y_train, X_test, y_test
```

Результат функции должен удовлетворять условию

```
len(X_train) / (len(X_test) + len(X_train)) == ratio
len(y_train) / (len(y_test) + len(y_train)) == ratio
```

2 Классификатор k-ближайших соседей не предполагает отдельной процедуры обучения, поэтому сразу перейдём к функции, предсказывающей метки класса по известным примерам. Функция `knn` должна принимать:

- обучающую выборку `X_train`, `y_train`,
- выборку, которую нужно классифицировать, `X_test`,
- количество соседей `k` и
- функцию расстояния `dist`, например, Евклидово расстояние.

Выходом функции является вектор `y_test`, в котором для каждого элемента `X_test` хранится соответствующий ему класс.

[1] `https://gist.github.com/ktisha/25e6d5a79628d19cd4b0908cd0fd2459`

3 Самое время оценить качество получившегося классификатора в терминах точности (*precision*) и полноты (*recall*). Формально эти метрики определяются следующим образом:

$$\text{Precision} = \frac{TP}{TP + FP} \qquad \text{Recall} = \frac{TP}{TP + FN}$$

Здесь

- TP — это количество элементов, которые классификатор верно отнёс к классу c,
- FP — количество элементов, которые классификатор неверно отнёс к классу c,
- FN — количество элементов, которые классификатор неверно отнёс к классу, отличному от c.

Реализуйте функцию, вычисляющую для каждого класса точность и полноту по полученным от `knn` предсказаниям. На Python функцию можно записать так:

```
# Обозначим за y_pred результат работы k-ближайших соседей на тестовой
# выборке X_test.
y_pred = knn(X_train, y_train, X_test, k=..., dist=...)

def print_precision_recall(y_pred, y_test):
    # Подсказка: значение n_classes можно вычислить по y_test
    #     n_classes = len(set(y_test))
    # или
    #     import numpy as np
    #     n_classes = len(np.unique(y_test))
    for c in range(n_classes):
        # ...
        print(class, precision, recall)
```

4 Реализуйте функцию для выбора оптимального значения k по методу LOO (leave one out) кросс-валидации. Функция должна принимать обучающую выборку и функцию расстояния:

```
def loocv(X_train, y_train, dist):
    # ...
    return opt_k
```

5 Оцените точность и полноту предсказаний классификатора с оптимальным k и двумя любыми функциями расстояния. Правда ли, что все вина одинаковые?