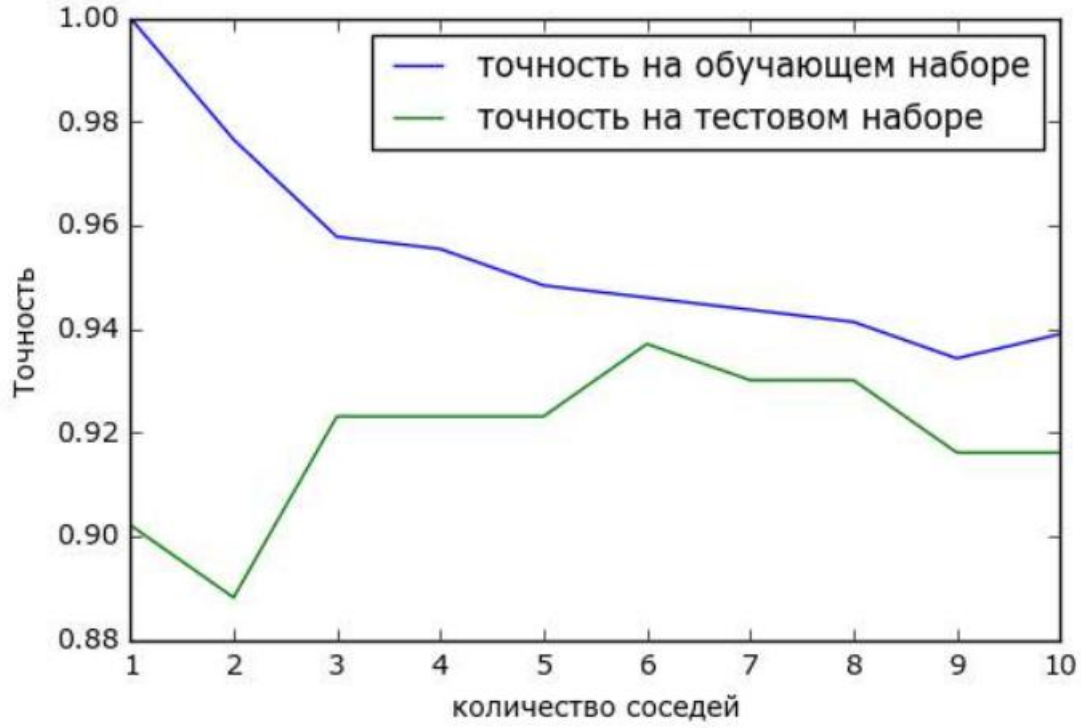
Задание 3. Метод “k ближайших соседей”
(подробней см. в [1, стр. 139–143], [2, стр. 50–59] [3, стр. 271–278], [4])
(документация: https://scikit-learn.org/stable/modules/neighbors.html
sklearn.neighbors.KNeighborsClassifier — scikit-learn 1.1.3 documentation
sklearn.neighbors.KNeighborsRegressor — scikit-learn 1.1.3 documentation)

Метод k ближайших соседей, является одним из самых простых алгоритмов машинного обучения. На этапе обучения (fit) фактически происходит только запоминание обучающего набора данных, который в дальнейшем будет участвовать в предсказании (predict). Для того чтобы сделать прогноз для новой точки данных, алгоритм находит ближайшие к ней точки обучающего набора, то есть находит «ближайших соседей». И выбирает для точки доминирующий среди ее соседей класс. Т.к. в результате обучения не строится никакой закономерности, а происходит просто запоминание набора данных, его еще называют “алгоритмом ленивого обучения”.

Параметр n_neighbors (по умолчанию равен 5) задает количество ближайших соседей, используемых для предсказания. Здесь также как и во всех остальных методах важно разбивать на обучающую и тестовую выборку. Т.к. если мы попытаемся предсказывать на обучающей выборке, то идеальный результат (100%) мы достигнем при n_neighbors = 1 (точка будет предсказывать саму себя). Правильный выбор числа соседей важен для нахождения хорошего баланса между переобучением и недообучением. При малом числе n_neighbors границы области будут слишком сложными и зависеть от отдельных точек, и, наоборот, при сильном увеличении числа n_neighbors может начать доминировать класс с большим числом образцов, и в какой-то момент начать предсказывать везде только этот класс.

Оптимальное количество зависит от конкретного набора данных, и, обычно, пробуют с разным количеством соседей, сравнивают их и выбирают лучшее число для тестовой выборки.



Кроме числа соседей можно задавать еще вес weights отдельных точек. По умолчанию weights = ‘uniform’, то есть вес не задается. Можно выбрать weights = ‘distance’ и тогда для определения класса точки используется не только число соседей, но и их расстояние от нее.

Расстояние (метрика), приведение признаков к одному масштабу. В методе задается функция используемая для измерения расстояния (по умолчанию задано обычное евклидово расстояние: p = 2, metric = 'minkowski'):

$$d\left(x^{(1)}, x^{(2)}\right) = \sqrt[p]{\sum_k \left|x_k^{(1)} - x_k^{(2)}\right|^p}.$$

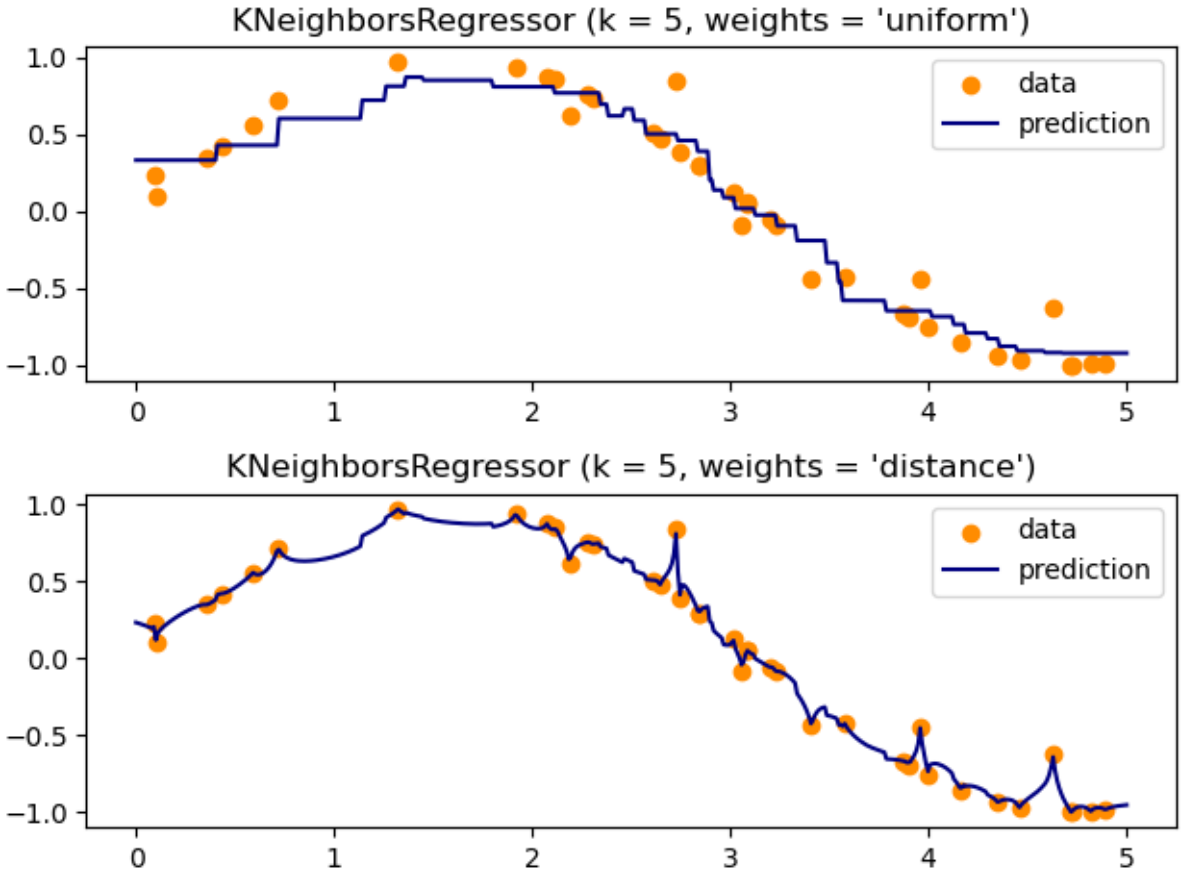
В методах, в которых используется расстояние между точками, фактически все признаки имеющиеся в задачах попадают в одно метрическое пространство. И, например, если один или несколько параметров доминируют в наборе признаков (т.е. их значения больше остальных на порядки), то остальные перестают влиять.

Например, в задаче о выдаче кредита клиенту у него есть атрибуты: наличие собственности, наличие машины, возраст, сбережения, зарплата и т.д. Пусть есть два клиента, у которых зарплата 30000 руб. и 31000 руб., а остальные параметры намного меньше. Тогда расстояние между этими клиентами будет ≈ 1000 независимо от остальных параметров.

В таких задачах применяют приведение признаков к одному масштабу. Это можно сделать с помощью функции sklearn.preprocessing.scale (https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.scale.html).

Есть еще вариант метода (RadiusNeighborsClassifier), в котором задается не число ближайших соседей, а радиус. И класс определяется по соседям в заданном радиусе.

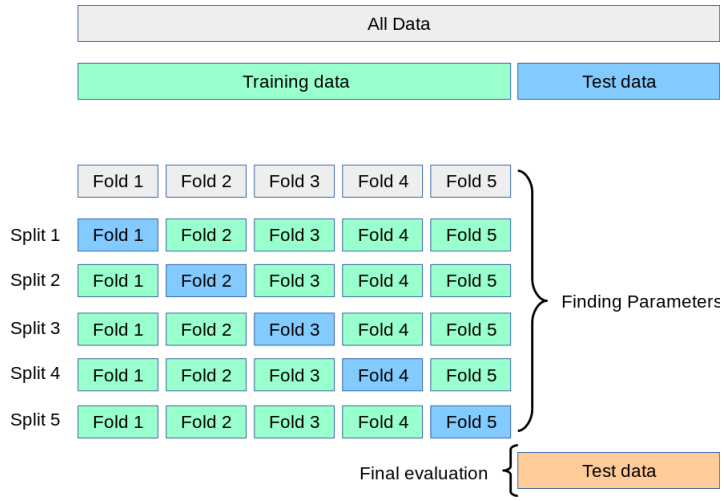
Существует также **регрессионный вариант алгоритма k ближайших соседей**. Значение в точке приближается числом, которое равно среднему значению из k ближайших точек.



Одним из преимуществ метода является то, что его очень легко интерпретировать и, как правило, он дает приемлемое качество без необходимости использования большого количества настроек. Он является хорошим базовым алгоритмом, который нужно попробовать в первую очередь, прежде чем рассматривать более сложные методы. Как правило, построение модели ближайших соседей происходит очень быстро, но, когда ваш обучающий набор очень большой (с точки зрения количества характеристик или количества наблюдений) получение прогнозов может занять некоторое время.

При использовании алгоритма ближайших соседей важно выполнить предварительное масштабирование признаков. Данный метод не так хорошо работает, когда речь идет о наборах данных с большим количеством признаков (сотни и более), и особенно плохо работает в ситуации, когда подавляющее число признаков в большей части наблюдений имеют нулевые значения (так называемые разреженные наборы данных или sparse datasets).

Кросс-валидация (или перекрестная проверка) (см., например, [2, стр. 269] или https://scikit-learn.org/stable/modules/cross_validation.html) используется вместо простого разбиения данных на обучающий и тестовый наборы. Данные разбиваются на несколько частей. И после этого, каждая из этих частей по очереди выступает в роли тестовой, а все остальные части – в качестве обучающих. Качество модели оценивается на основе среднего значения качества для всех таких разбиений. Таким образом, все данные участвуют в обучении и метод запускается несколько раз для разных выборок. Дает более точную оценку качества метода, чем простое разбиение на обучающий и тестовый наборы. Метод KFold (https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html) библиотеки scikit-learn делает разбиение данных на несколько частей (Fold).



Вычислить ошибку на всех разбиениях можно при помощи функции cross_val_score (scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_val_score.html).

Задания

I. Прочитайте пункт “Метод k ближайших соседей” и рекомендованную литературу к нему и ответьте на вопросы:

1) Как определяется класс нового объекта точки в методе k ближайших соседей?
2) Какие параметры можно задавать в методе?
3) Зачем делается приведение признаков к одному масштабу?
4) Какие преимущества и недостатки метода?
5) Что такое кросс-валидация?

II. Выполните задание из в файла "3 statement_neighbours.pdf".

III. Выполните задание из в файла "4 statement-metric-tuning.pdf".

Литература

[1] Рашка Себастьян, Мирджалили Вахид Python и машинное обучение: машинное и глубокое обучение с использованием Python, scikit-learn и TensorFlow 2, 3-е изд.: Пер. с англ. СПб. : ООО "Диалектика", 2020. 848 с.

[2] Андреас Мюллер, Сара Гвидо Введение в машинное обучение с помощью Python. Руководство для специалистов по работе с данными. 393 с.

[3] Крис Элбон Машинное обучение с использованием Python. Сборник рецептов. Пер. с англ. СПб. : БХВ-Петербург, 2019. 384 с.

[4] Машинное обучение (курс лекций, К.В.Воронцов): Метод ближайших соседей (machinelearning.ru), (запись лекции “Метрические методы классификации и регрессии”: https://www.youtube.com/watch?v=GyOxB2itxnc)