Выбор метрики

Данное задание основано на лекциях по метрическим методам и посвящено выбору наилучшей метрики для конкретной задачи.

Вы научитесь:

- подбирать конкретную метрику Минковского для задачи

Введение

Главным параметром любого метрического алгоритма является функция расстояния (или метрика), используемая для измерения сходства между объектами. Можно использовать стандартный вариант (например, евклидову метрику), но гораздо более эффективным подходом является подбор метрики под конкретную задачу. Одним из подходов является использование той же евклидовой метрики, но с весами: каждой координате ставится в соответствие определенный коэффициент; чем он больше, тем выше вклад признака в итоговое расстояние. Веса настраиваются с целью оптимизации качества на отложенной выборке. Другой подход, о котором и пойдет речь в данном задании — выбор метрики из некоторого класса метрик. Мы возьмем за основу метрику Минковского:

$$\rho_p(x, z) = (\sum_{j=1}^{n} |x_j - z_j|^p)^{\frac{1}{p}}$$

Параметром метрики Минковского является число p, которое мы и будем настраивать.

Реализация в sklearn

Нам понадобится решать задачу регрессии с помощью метода k ближайших соседей — воспользуйтесь для этого классом sklearn.neighbors.KNeighborsRegressor. Метрика задается с помощью параметра metric, нас будет интересовать значение 'minkowski'. Параметр метрики Минковского задается с помощью параметра p данного класса.

Инструкция по выполнению

Мы будем использовать в данном задании набор данных Boston, где нужно предсказать стоимость жилья на основе различных характеристик расположения (загрязненность воздуха, близость к дорогам и т.д.). Подробнее о признаках можно почитать по адресу
https://archive.ics.uci.edu/ml/datasets/Housing

1. Загрузите выборку Boston с помощью функции sklearn.datasets.load_boston(). Результатом вызова данной функции является объект, у которого признаки записаны в поле data, а целевой вектор — в поле target.

2. Приведите признаки в выборке к одному масштабу при помощи функции sklearn.preprocessing.scale.

3. Переберите разные варианты параметра метрики p по сетке от 1 до 10 с таким шагом, чтобы всего было протестировано 200 вариантов (используйте функцию numpy.linspace). Используйте KNeighborsRegressor с n_neighbors=5 и weights='distance' — данный параметр добавляет в алгоритм веса, зависящие от расстояния до ближайших соседей. В качестве метрики качества используйте среднеквадратичную ошибку (параметр scoring='mean_squared_error' у cross_val_score; при использовании библиотеки scikit-learn версии 18.0.1 и выше необходимо указывать scoring='neg_mean_squared_error'). Качество оценивайте, как и в предыдущем задании, с помощью кросс-валидации по 5 блокам с random_state = 42, не забудьте включить перемешивание выборки (shuffle=True).

4. Определите, при каком p качество на кросс-валидации оказалось оптимальным. Обратите внимание, что cross_val_score возвраща-

ет массив показателей качества по блокам; необходимо максимизировать среднее этих показателей. Это значение параметра и будет ответом на задачу.

Если ответом является нецелое число, то целую и дробную часть необходимо разграничивать точкой, например, 0.4. При необходимости округляйте дробную часть до одного знака.

Ответ на каждое задание — текстовый файл, содержащий ответ в первой строчке. Обратите внимание, что отправляемые файлы не должны содержать перевод строки в конце.