 # Метрические методы классификации

1 Методы ближайших соседей

В этой части работы вы:

- научитесь готовить данные к построению модели (предобработка, или препро цессинг данных);
- познакомитесь с методами ближайших соседей для задач классификации и регрессии, реализованными в библиотеке `scikit-learn` ;
- научитесь оценивать качество модели с помощью отложенной выборки.

Указания к выполнению

1. Подключитесь к одному из наборов данных на Kaggle:
   - Вариант 1: Adult income;
   - Вариант 2: Telecom churn;
   - Вариант 3: Bank marketing;
   - Вариант 4: Online news popularity;
   - Вариант 5: Wine quality;
   - Вариант 6: Breast cancer.

   Разберитесь в том, как устроен ваш датасет и какова постановка задачи.
2. Извлеките целевой признак (target). Какая из задач обучения с учителем рассмат ривается –– классификация или регрессия?
3. Каково распределение значений target-переменной? Постройте подходящую ви зуализацию. Прокомментируйте результат.
4. Проведите необходимую **предобработку данных** (preprocessing). Для построения моделей с помощью метрических методов все признаки должны быть закодиро ваны числами. Полезными будут следующие методы библиотеки Pandas:
   - `map()` –– для перекодировки категориальной переменной числовыми мет ками;
   - `get_dummies()` –– для создания нескольких бинарных признаков на осно ве категориального.

Также может потребоваться **масштабирование данных** (scaling). Воспользуйтесь классом `StandardScaler` библиотеки Scikit-learn.

5. Разбейте набор данных на обучающую и валидационную (тестовую) выборки с помощью метода `train_test_split` .
6. Обучите алгоритм классификации `kNeighborsClassifier` или регрессии `KNeighborsRegressor` . Оцените качество каждой модели на валидационной выборке с помощью
   - `accuracy_score` для классификации;
   - `mean_squared_error` для регрессии.

   Сравните результаты и сделайте выводы.

2 Настройка оптимального числа ближайших соседей в методе kNN

В данной части работы вы научитесь настраивать параметр `n_neighbors` алгоритма kNN с помощью перекрёстной проверки (кросс-валидации).

Указания к выполнению

1. Создайте генератор разбиений, который перемешивает выборку перед созданием блоков (`shuffle=True`). Число блоков `n_splits` равно 5. Задайте также параметр `random_state` для воспроизводимости результатов. Например:

   ```
   kf = KFold(n_splits=5, shuffle=True, random_state=42)
   ```

   Найдите показатель качества модели kNN на кросс-валидации. Подумайте, приемлемо ли использование вашей меры (метрики) качества в данной задаче? При необходимости пересчитайте качество с помощью другой метрики из списка.

2. Осуществите кросс-валидацию модели при числе соседей $k \in [1;50]$. Используйте `GridSearchCV` . При каком k качество получилось наилучшим? Чему равна

эта оценка качества? Постройте график значений метрики в зависимости от k (`matplotlib.pyplot.plot()`).

3 Выбор метрики в методе kNN

Главным параметром любого метрического алгоритма является функция расстояния (метрика), используемая для измерения сходства между объектами. Можно использовать стандартный вариант (например, евклидову метрику), но гораздо более эффективным вариантом является подбор метрики под конкретную задачу. Один из подходов –– 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 данной части работы –– выбор метрики из некоторого класса метрик. Мы возьмем за основу метрику Минковского:

$$\hat{a}_p(x,y) = \left(\sum_{j=1}^{n} |x_j - y_j|^p \right)^{1/p}.$$

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

Указания к выполнению

1. Переберите разные варианты значений параметра p по сетке от 1 до 10 с таким шагом, чтобы всего было протестировано 200 вариантов (удобно использовать функцию `numpy.linspace`). Используйте `KNeighborsClassifier` или `KNeighborsRegressor` с оптимальным значением `n_neighbors`, найденным ранее. Задайте опцию `weights='distance'` –– данный параметр добавляет в алгоритм веса, зависящие от расстояния до ближайших соседей. В качестве метрики качества снова используйте `accuracy`. Качество оценивайте с помощью кросс-валидации по 5 блокам.
2. Определите, при каком p качество на кросс-валидации оказалось оптимальным.

Обратите внимание, что `cross_val_score` возвращает массив показателей качества по блокам; необходимо максимизировать среднее этих показателей.

4 Другие метрические методы

Поэкспериментируйте с другими метрическими методами для задач регрессии и клас сификации, представленными в библиотеке Scikit-learn:

- `RadiusNeighborsClassifier` ;
- `RadiusNeighborsRegressor` ;
- `NearestCentroid` .