Конспект по теме "Сбор данных"

Источники данных

Существует множество источников данных, которые можно использовать для тренировки моделей.

Один из источников — хранилища данных компании. Нужно только извлечь всю нужную информацию и подготовить к анализу. Это оптимистичный сценарий.

Чаще всего данных в компании нет. Если задача распространённая, датасеты можно найти в открытых источниках:

- Платформа для соревнований по Data Science Kaggle;
- Хранилище датасетов Калифорнийского университета в Ирвайне;
- Сборник открытых данных разных госучреждений США;
- Портал открытых данных правительства Москвы.

Для некоторых задач данные можно собрать в интернете. Скачать все страницы нужного портала, а данные извлечь из них специальной программой **«кроулер»** или **«скрейпер»**.

Разметка данных

Возможно, в компании данные есть, но только без значений целевого признака. С такими **неразмеченными данными** тоже можно работать. Чтобы получить из них обучающий набор, проводят **разметку,** или **аннотирование** данных (*data labeling; data annotation*). То есть для каждого снимка определяется правильный ответ (человек или нет). Так данные становятся **размеченными** (*labeled data*). Иногда для разметки не требуется специальное образование. Но если данные касаются здоровья и жизни людей, обращайтесь к экспертам.

Разметкой занимаются специальные сервисы. В них загружают неразмеченные данные и назначают цену за разметку одного объекта.

Любой может зайти на сервис, разметить данные и получить за это деньги.

Вот несколько популярных сервисов, к которым обращаются специалисты по Data Science:

- Amazon Mechanical Turk;
- Яндекс.Толока.

Контроль качества разметки

Повысить качество данных после разметки помогают методы **контроля качества разметки**. Как они работают? Все объекты или их часть размечаются несколько раз, а из них формируется финальный ответ.

Один из способов контроля качества разметки — **голосование по большинству** (*majority vote*). Кто и как «голосует»? Например, каждый объект размечается тремя асессорами. Финальным ответом будет тот, который выбрало большинство.

Утечка целевого признака

После сбора данных нужно проверить, нет ли в них **утечки целевого признака** (*target leakage*). Она происходит, когда в признаки случайно попадает информация о целевом признаке.

Кросс-валидация

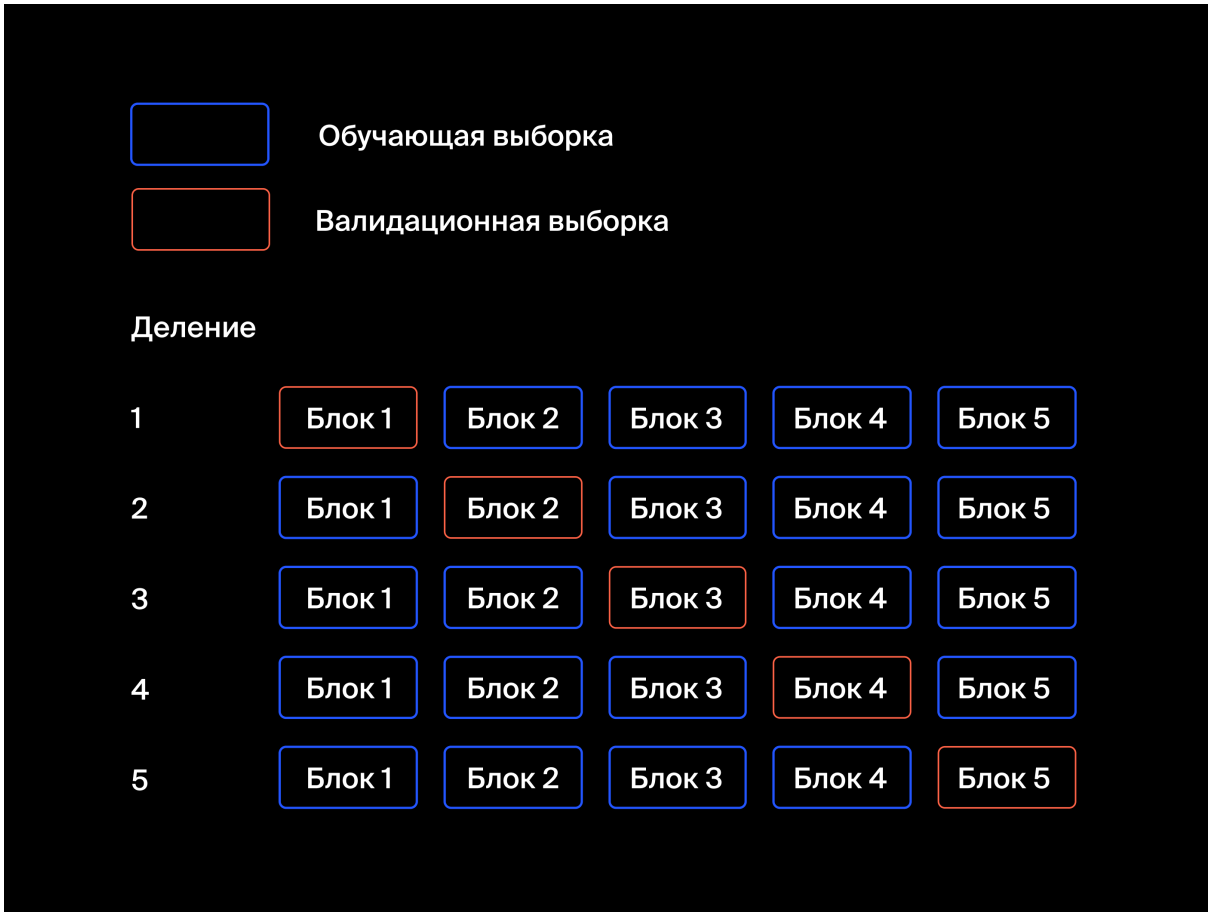
Вы уже умеете делить данные на обучающую, тестовую и валидационную выборки и знаете, что репрезентативность достигается за счёт случайного отбора. Но как гарантировать корректность распределения на большом наборе данных? Нужно взять несколько случайных выборок!

Обучить и протестировать модель на нескольких случайно сформированных выборках поможет **кросс-валидация** (*cross validation*).

Как она работает? Все данные мы поделили на обучающую и тестовую выборки. Тестовую «прячем» для финальной проверки, а обучающую

случайным образом делим на *K* равных частей. Сам метод деления называется *K-Fold*, где *K* — это число блоков.

Допустим, на первом этапе деления «Блок 1» становится валидационной выборкой, а другие блоки — обучающими. На этих выборках модель обучается и проверяется. На втором этапе «Блок 2» — валидационный набор, а другие блоки — обучающие. То же самое повторяется и с остальными: и так всего K раз.



Мы будто «скользим» по данным, каждый раз принимая новый блок за валидационный. А среднее всех оценок, полученных в ходе кросс-валидации, — это и есть оценка качества модели.

Метод кросс-валидации напоминает бутстреп тем, что генерируется несколько выборок. Только в перекрёстной проверке содержимое блоков зафиксировано и не меняется на каждом этапе обучения-проверки. Каждое наблюдение проходит и через обучающий, и валидационный наборы данных.

Кросс-валидация нужна, когда нужно сравнить модели, подобрать гиперпараметры или оценить полезность признаков. Она минимизирует случайность разделения данных и даёт более точный результат. Единственный недостаток кросс-валидации — время на вычисления, особенно при большом количестве наблюдений или блоков. Много времени.

Кросс-валидация в sklearn

Перекрёстную проверку можно выполнить средствами библиотеки *sklearn*. Оценить качество модели кросс-валидацией позволяет функция **cross_val_score** из модуля *sklearn.model_selection*. Функция вызывается так:

```
from sklearn.model_selection import cross_val_score
cross_val_score(model, features, target, cv=3)
```

На вход функция принимает несколько аргументов. Например:

- *model* — модель для кросс-валидации. Она обучается в ходе перекрёстной проверки, поэтому на вход подаётся необученной. Допустим, для дерева решений нужна такая модель:

```
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier()
```

- *features* — признаки;
- *target* — целевой признак;
- *cv* — число блоков для кросс-валидации (по умолчанию их три).

Делить данные на блоки или валидационную и обучающую выборки функция не требует. Все эти процедуры происходят автоматически. На выходе от каждой валидации получаем список оценок качества моделей. Каждая оценка равна *model.score()* на валидацинной выборке. Например, для задачи классификации — это *accuracy*.