Конспект по теме "Первая обученная модель"

Обучающий набор данных

Обучение программ происходит точно также, как учатся эксперты. Они собирают базу знаний, классифицируют её, выявляют взаимосвязи и набираются опыта. Для решения задачи машинного обучения, нужно накопить базу, на которой будет учиться компьютер —**обучающий (тренировочным) набор данных,** или **обучающую выборкой.**

В анализе данных строки назывались наблюдениями, а столбцы — переменными. В машинном обучении в строках записываем **объекты**, а в столбцах — **признаки.** Признак, который нужно предсказать, — **целевой**.

Обучение с учителем

В вашем распоряжении обучающий набор данных и целевой признак, который нужно предсказать по остальным признакам. Такие задачи относятся к классу **«обучение с учителем»**. «Учитель» ставит **вопросы** (признаки) и указывает **ответы** (целевой признак). И ничего не объясняет. Обучение с учителем подходит для решения многих бизнес-задач.

Также существуют и другие типы задач:

- **обучение без учителя** — без целевого признака;
- **частичное обучение** — целевой признак известен только для части обучающих данных;
- **рекомендации** — вместо признаков и объектов есть пользователи и товары.

Выясним, какие подзадачи есть у обучения с учителем.

Все переменные и признаки (в том числе, целевой признак) бывают двух типов: категориальные и количественные.

Если целевой признак категориальный, то решается задача **классификации**. Когда категорий всего две, речь идёт о **бинарной (двоичной) классификации**.

Если целевой признак количественный, то решается задача **регрессии** — по данным восстанавливается зависимость между переменными.

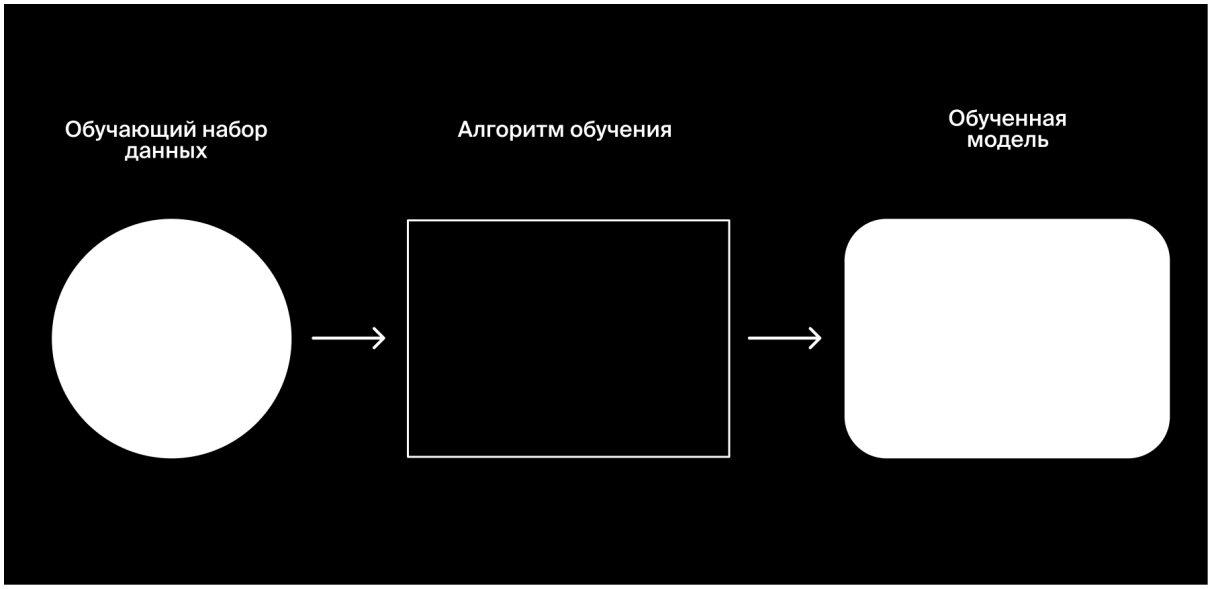
Модели и алгоритмы

Чтобы делать предсказания, нужно понимать взаимосвязи признаков и ответов. Специалист по Data Science выдвигает предположение, как устроены эти взаимосвязи, а на его основании делает предсказания. Если они совпадают с реальностью, значит, предположение верное. Этот подход называется **моделированием**, а сами предположения и способы предсказания — **моделями машинного обучения.**

Рассмотрим одну популярную модель — **решающее дерево**. Оно может описывать процесс принятия решения почти в любой ситуации. На основе значений признаков формулируются ответы, а затем выстраивается дерево с ответами «Да»/«Нет» и различными вариантами действий.

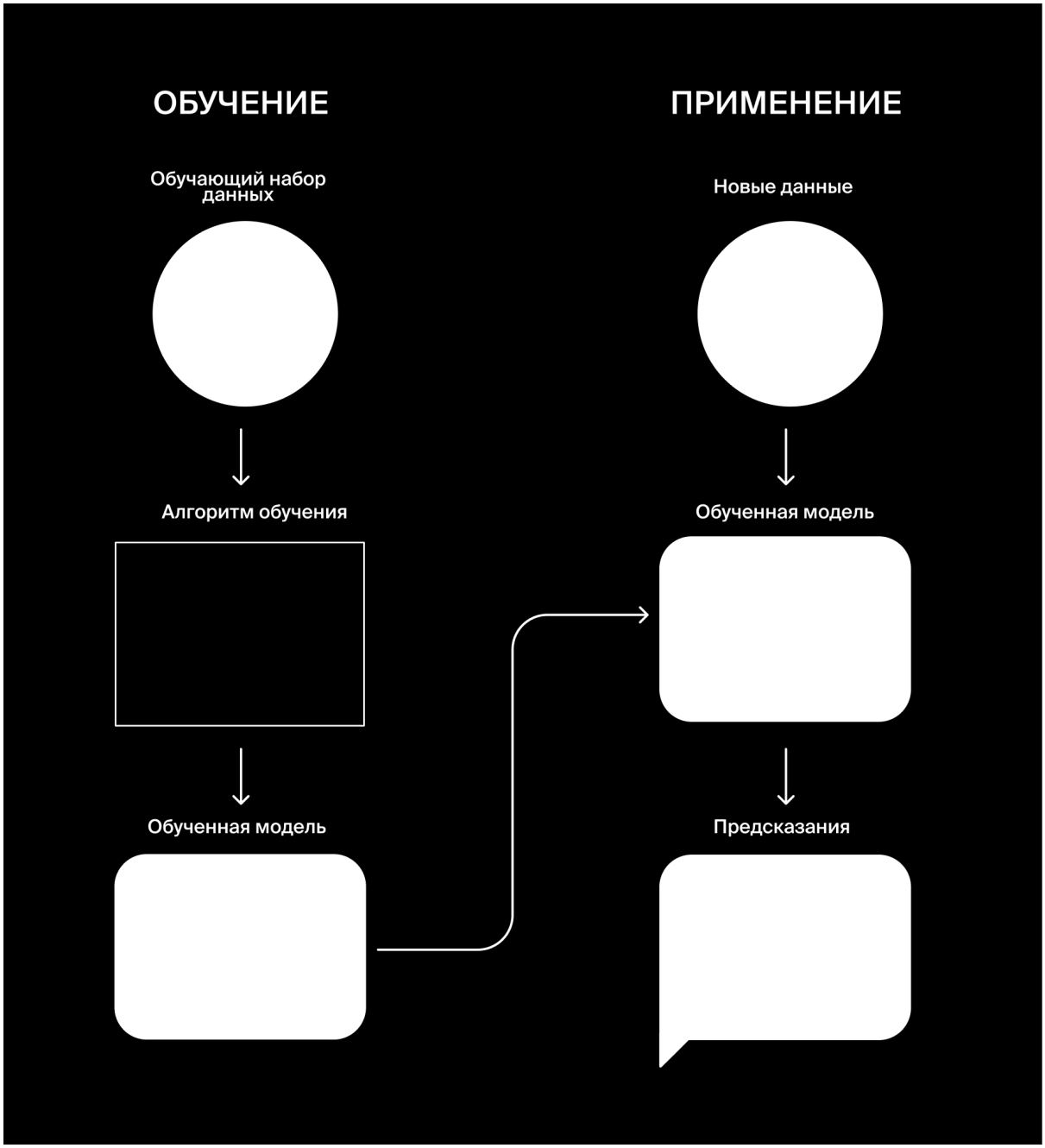
Какое именно из всего множества деревьев? Чтобы найти то самое, нужно **обучить модель:** подобрать решающее дерево, которое больше всего подойдёт нашей обучающей выборке. За это отвечают **алгоритмы**

обучения, а их результат работы называется **обученной моделью** (либо просто **моделью**).



После обучения модель готова **предсказывать**: получать на вход новые объекты (признаки) и выдавать ответы (целевой признак).

Процесс машинного обучения делится на два этапа — **обучение модели** и **работа этой модели**.



Библиотека scikit-learn

Алгоритмы обучения чаще устроены сложнее, чем сама модель. Поэтому представим их **чёрными ящиками**. Пока неважно, что происходит внутри них. Главное понять, что именно положить в ящик и как работать с тем, что он выдаст.

В библиотеках Python уже доступны многие алгоритмы. Одна из библиотек — **scikit-learn**, или **sklearn**. В *sklearn* много инструментов

работы с данными и моделей, поэтому они разложены по подразделам. В модуле **tree** находится решающее дерево.

Каждой модели в *sklearn* соответствует отдельная структура данных. **DecisionTreeClassifier** — это структура данных для классификации деревом решений. Импортируем её из библиотеки:

```
from sklearn.tree import DecisionTreeClassifier
```

Затем создаём объект этой структуры данных.

```
model = DecisionTreeClassifier()
```

В переменной *model* будет храниться модель. Наша модель пока не умеет предсказывать. Чтобы она научилась, нужно запустить алгоритм обучения.

Для того, чтобы запустить обучение, нужно вызвать метод **fit()** и передать ему данные.

```
model.fit(features, target)
```

После обучения в переменной *model* содержится полноценная модель. Чтобы предсказать ответы, нужно вызвать метод **predict()** и передать ему таблицу с признаками новых объектов.

```
answer = model.predict(new_features)
```