Data science

Лекция 6. Задачи анализа данных

2025/2026 учебный год

Доцент кафедры МО&МО, Махно В.В.

Классификация

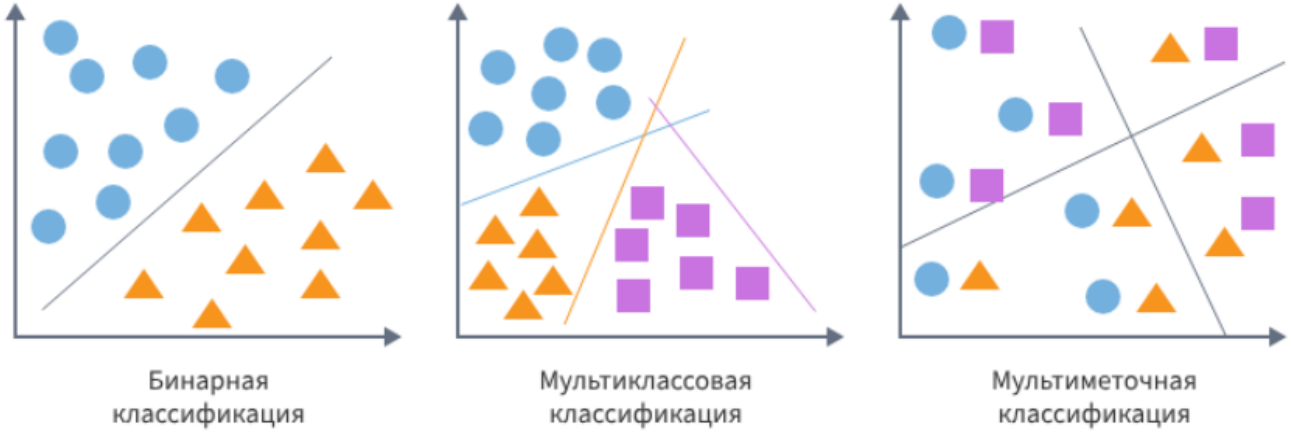
Классификация — это задача, в которой модель **предсказывает, к какому классу относится объект.**

Не «Сколько?», а **«Что именно?»**

Примеры задач:

- Спам или не спам?
- Мужчина или женщина?
- Солнечно, облачно, дождь?
- Купит клиент товар или нет?

Классификация



🔢 Типы классификации

Тип задачи	Пример	Классы
Бинарная	Спам или не спам	0 / 1
Мультиклассовая	Вид животного	Кошка / Собака / Лиса
Многоклассовая + мульти-ответ	Темы статьи	[Технологии, Наука]

Цель модели

Пример

Возраст	Доход	Купил (target)
22	30k	Нет (0)
45	90k	Да (1)
35	70k	Да (1)

Популярные алгоритмы классификации

Модель	Особенности
LogisticRegression	Простая, быстрая, интерпретируемая
KNN	Смотрит на "похожих" соседей
DecisionTree	Принимает решения по признакам, как вопросы
RandomForest	Несколько деревьев голосуют
SVM	Строит границу между классами
NaiveBayes	Часто используется в текстах (спам-фильтры)

Где используется классификация?

Область	Пример
Маркетинг	Классификация клиентов: уйдёт / останется
Финансы	Одобрит ли банк кредит
Медицина	Заболел / не заболел
HR	Подходит ли кандидат
E-commerce	Рекомендации и персонализация

Переход от линейной к логистической регрессии

Линейная регрессия предсказывает **любое число** (от $-\infty$ до $+\infty$).
Пример: цена, рост, доход.

Логистическая регрессия используется, когда нужно предсказывать **класс (0 или 1)**.
Пример: купит / не купит, спам / не спам.

В чём суть перехода?

Сначала считается линейная комбинация признаков:

$z=w \cdot x+b$

Затем это значение пропускается через **сигмоиду**, чтобы получить вероятность:

$$P = \frac{1}{1+e^{-z}}$$

Если вероятность > 0.5 → класс 1, иначе — 0.

Качество классификации

1) Базовая метрика — Accuracy (точность)

$$\text{Accuracy} = \frac{\text{Количество правильных предсказаний}}{\text{Общее количество примеров}}$$

Пример:
Модель предсказала верно 90 из 100 → Accuracy = 90%

Хорошо работает, если классы сбалансированы.

Плохо, если один класс сильно преобладает.

Подробный разбор — по матрице ошибок (confusion matrix):

- **TP (True Positive)** — правильно предсказал класс 1
- **FP (False Positive)** — ошибочно предсказал 1, но это был 0
- **FN (False Negative)** — ошибочно предсказал 0, но это был 1
- **TN (True Negative)** — правильно предсказал 0

	Факт: 1	**Факт: 0**
Предсказал: 1	TP	FP
Предсказал: 0	FN	TN

Качество классификации

2) **Precision — точность положительного класса**

$$\text{Precision} = \frac{TP}{TP+FP}$$

Вопрос: из всех, кого модель посчитала положительными, **сколько были действительно положительными?**

Важно, если **ошибки FP критичны**
(например, диагностика болезни — нельзя ошибочно поставить диагноз)

	Факт: 1	**Факт: 0**
Предсказал: 1	TP	FP
Предсказал: 0	FN	TN

Качество классификации

	Факт: 1	Факт: 0
Предсказал: 1	TP	FP
Предсказал: 0	FN	TN

3) Recall — полнота

$$\text{Recall} = \frac{TP}{TP+FN}$$

Вопрос: из всех реальных положительных случаев, **сколько модель нашла?**

Важно, если **нельзя пропустить важное** (например, обнаружение мошенничества, болезнь)

4) F1-score — баланс между precision и recall

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

Удобно использовать, если классы несбалансированы и важно учитывать обе метрики.

Сравнение метрик

Метрика	Что измеряет	Когда использовать
Accuracy	Доля правильных ответов	Когда классы сбалансированы
Precision	Сколько предсказанных 1 были верны	Когда FP опасны (ошибки "ложного да")
Recall	Сколько реальных 1 было найдено	Когда FN опасны (пропуски)
F1-score	Баланс точности и полноты	При дисбалансе классов

Машинное обучение

Почему базовых моделей недостаточно?

Базовые модели (линейная, логистическая регрессия, KNN) хороши для старта.

Но **в реальных задачах**:

- данные часто **нелинейные**, сложные;
- признаки взаимодействуют между собой;
- много **пропусков**, **шумов**, **категориальных переменных**.

Значит, нужны более гибкие, устойчивые и точные модели.

Decision Tree — дерево решений

Модель **разбивает данные по признакам**, задавая простые вопросы.

Если возраст < 30: если доход > 50k → класс = 1 иначе → класс = 0 иначе: → класс = 1

Плюсы:

- Интуитивно понятно, легко визуализировать
- Умеет работать с числовыми и категориальными признаками
- Не требует масштабирования

Минусы:

- Переобучается (особенно без ограничения глубины)
- Нестабильна: небольшие изменения в данных → совсем другое дерево

Где используется:

- Бизнес-правила
- Кредитные скоринги
- Прогнозы продаж

Random Forest — случайный лес

Это **множество деревьев решений**, обученных на случайных подвыборках. Каждое дерево голосует, и побеждает **большинство**.

Плюсы:

- Очень устойчив к переобучению
- Работает хорошо «из коробки»
- Умеет определять **важность признаков**

Минусы:

- Менее интерпретируемый, чем одно дерево
- Медленнее при больших объёмах

Где используется:

- Банковский скоринг
- Прогнозирование спроса
- Табличные данные в бизнесе

Gradient Boosting (XGBoost, LightGBM, CatBoost)

Алгоритм **обучает деревья по очереди**, каждое исправляет ошибки предыдущего.
Сначала строим первую модель. Затем учим вторую — чтобы она "улучшила" первую. И так далее.

Популярные реализации:

- **XGBoost** — классика, гибкая и точная
- **LightGBM** — очень быстрая и эффективная
- **CatBoost** — хорошо работает с категориальными данными

Плюсы:

- Высокая точность
- Гибкие настройки
- Хорошо работает на соревнованиях (например, Kaggle)

Минусы:

- Требует настройки (бустинг ≠ магия)
- Может переобучиться без регуляризации
- Медленнее, чем случайный лес

Где используется:

- Финансовое моделирование
- Маркетинг и рекомендации
- Предсказание оттока, конверсий, лояльности

KNN — k ближайших соседей

Модель **не учится**, а при предсказании **сравнивает объект с "похожими"**.

→ Находит k ближайших точек и **голосует**.

Плюсы:

- Очень прост в реализации
- Нет этапа обучения
- Интуитивно понятен

Минусы:

- **Медленный при большом датасете**
- Не работает хорошо в задачах с большим числом признаков
- Чувствителен к масштабу признаков

Где используется:

- Рекомендательные системы
- Сегментация
- Простейшие классификаторы

Нейросети (Neural Networks)

Модели, вдохновлённые работой мозга.
Состоят из **слоёв нейронов**, которые преобразуют вход в результат.

Плюсы:

- Могут находить **очень сложные зависимости**
- Работают с **текстом, изображениями, звуком**
- Подходят для задач "глубокого обучения"

Минусы:

- Сложные и требовательные
- Трудны в интерпретации
- Требуют большого объёма данных и вычислений

Где используются:

- Компьютерное зрение
- Обработка естественного языка (NLP)
- ChatGPT, голосовые помощники, генерация изображений

Кластеризация

Группировка объектов без меток

Кластеризация — это задача, в которой модель **сама находит группы (кластеры)** внутри данных.

В отличие от классификации:

Нет целевой переменной (y)

Модель работает **без учителя** (unsupervised learning)

Примеры:

Ситуация	Что группируем?	Кластеры могут быть
E-commerce	Поведение клиентов	Новички / Активные / Спящие
Бизнес	Товары по продажам	Хиты / Сезонные / Неактивные
География	Координаты точек	Города, кластеры по регионам

K-Means

Цель алгоритма - разделить объекты **на группы**, чтобы:

- Внутри каждой группы элементы были **похожи друг на друга**
- Между группами была **разница**

Основные алгоритмы кластеризации

1) K-Means

Принцип работы:

- Задаём количество кластеров k
- Алгоритм находит k центров и относит каждую точку к ближайшему

Плюсы:

- Простой и быстрый
- Подходит для числовых данных

Минусы:

- Нужно заранее выбрать k
- Плохо работает при сложной форме кластеров

```
from sklearn.cluster import KMeans
model = KMeans(n_clusters=3)
model.fit(X)
labels = model.labels_
```

DBSCAN

2) **DBSCAN**

Принцип работы:

- Объединяет точки, находящиеся **достаточно близко друг к другу**
- Не требует заранее k

Плюсы:

- Ищет кластеры **произвольной формы**
- Умеет находить **выбросы (noise)**

Минусы:

- Требует настройки радиуса и минимального числа точек

Иерархическая кластеризация

3) **Иерархическая кластеризация**

Принцип работы:

- Создаёт древовидную структуру из объектов
- Можно строить **дендрограмму**

Плюсы:

- Удобно для визуального анализа
- Не требует k на старте (можно обрезать дерево на нужном уровне)

Применение кластеризации

Область	Применение
Маркетинг	Сегментация клиентов
Ритейл	Группировка товаров
Аналитика	Поиск аномалий, схожих пользователей
Биология	Группировка генов, белков

Пример визуализации кластеров

```
import matplotlib.pyplot as plt
import seaborn as sns

sns.scatterplot(x=X[:,0], y=X[:,1], hue=labels, palette='viridis')
plt.title("Результаты кластеризации")
plt.show()
```

Как понять, хорошо ли получилось?

Если у нас **нет правильных ответов**, то используем **внутренние метрики**:

Метрика	Что показывает
Silhouette score	Насколько хорошо объекты "вписываются" в свой кластер
Davies–Bouldin index	Чем меньше, тем лучше разделение
Inertia (для KMeans)	Суммарное расстояние до центров кластеров

```
from sklearn.metrics import silhouette_score

score = silhouette_score(X, labels)
print("Silhouette:", score)
```

Благодарю за внимание!