

ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ

Занятие 1

Преподаватель: Борзун Анна
Вадимовна

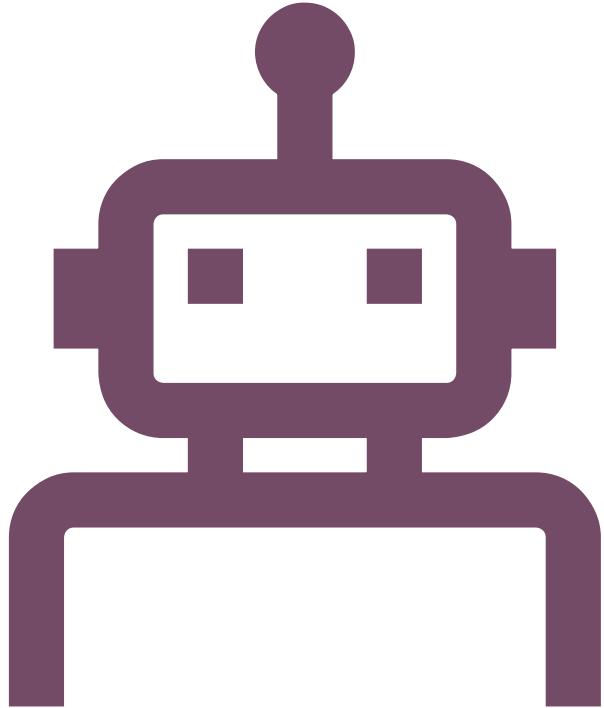
ПРЕДУСЛОВИЕ. НАМ ПОНАДОБЯТСЯ

- Python 3: [The Python Tutorial — Python 3.13.5 documentation](#)
 - NumPy: <https://numpy.org/doc/>
 - Pandas: <https://pandas.pydata.org/docs/>
 - scikit-learn (sklearn): <https://scikit-learn.org/stable/documentation.html>
 - Matplotlib: <https://matplotlib.org/stable/contents.html>
-

-
- **Искусственный интеллект** — это научная область, которая занимается созданием программ и устройств, имитирующих интеллектуальные функции человека. ИИ считается таковым, даже если умеет хорошо имитировать только одну функцию, например чтение.

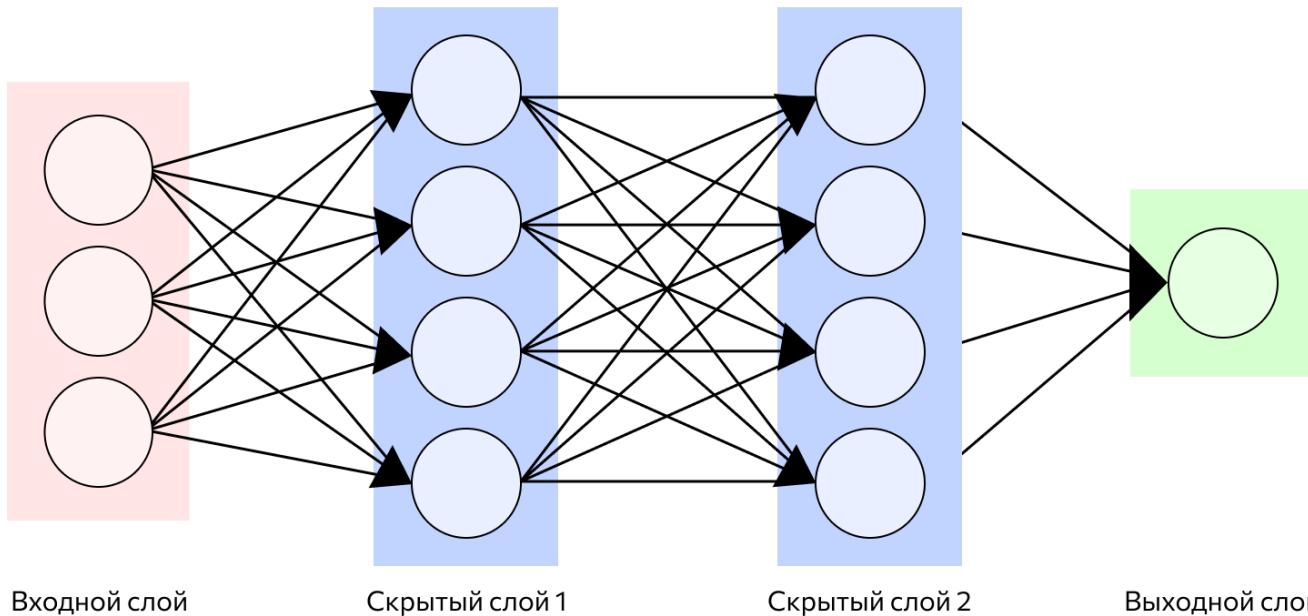


ТЕСТ ТЬЮРИНГА — ЭМПИРИЧЕСКИЙ ТЕСТ, ИДЕЯ КОТОРОГО БЫЛА ПРЕДЛОЖЕНА АЛАНОМ ТЬЮРИНГОМ В СТАТЬЕ «ВЫЧИСЛИТЕЛЬНЫЕ МАШИНЫ И РАЗУМ» (АНГЛ. COMPUTING MACHINERY AND INTELLIGENCE), ОПУБЛИКОВАННОЙ В 1950 ГОДУ В ФИЛОСОФСКОМ ЖУРНАЛЕ «MIND». ТЬЮРИНГ ЗАДАЛСЯ ЦЕЛЬЮ ОПРЕДЕЛИТЬ, МОЖЕТ ЛИ МАШИНА МЫСЛИТЬ.



«Если компьютер может работать так, что человек не в состоянии определить, с кем он общается — с другим человеком или с машиной, — считается, что он прошел тест Тьюринга»

ГЛУБОКИЕ НЕЙРОСЕТИ



МАШИННОЕ ОБУЧЕНИЕ

- — класс методов искусственного интеллекта, решающих задачу, строя алгоритм на основе размеченных данных (но иногда и неразмеченных).
- Приведите примеры применения МО для повседневных задач.

ПРИМЕРЫ ИСПОЛЬЗОВАНИЯ МАШИННОГО ОБУЧЕНИЯ ДЛЯ ПОВСЕДНЕВНЫХ ЗАДАЧ

Предсказание
стоимости акций;

Предсказание
температуры
воздуха в
ближайшем
будущем;

Рекомендательные
системы в
маркетплейсах и
сервисах по типу
СберЗвук.

Беспилотные
автомобили;

Генерация новых
изображений по
описанию;

Предсказание
третичной
структуры белков
по их молекулам;

	Кол-во ядер	RAM (ГБ)	Объем жесткого диска (ГБ)	Диагональ/разрешение	Работа от аккумулятора	Цена (руб.)
1		4	16	1000 (HDD)+ 128 (SSD)	17"/1920x1080 пикс.	до 6 часов
2		2	4	500 (HDD)	15"/1920x1080 пикс.	до 5 часов
3		4	8	256 (SSD)	14"/1920x1080 пикс.	до 12 часов
4		4	16	1000 (HDD)	17"/1920x1080 пикс.	до 3 часов
5		8	16	1000 (HDD) + 256 (SSD)	17"/1920x1080 пикс.	до 11 часов

Какой признак наиболее важный?

ПРЕДСКАЖЕМ СТОИМОСТЬ?

КАК МЫ ПРИШЛИ К РЕШЕНИЮ?

Признаки

	Кол-во ядер	RAM (Гб)	Объем жесткого диска (ГБ)	Диагональ/разрешение	Работа от аккумулятора	Цена (руб.)	
1		4	16	1000 (HDD)+ 128 (SSD)	17"/1920x1080 пикс.	до 6 часов	?
2		2	4	500 (HDD)	15"/1920x1080 пикс.	до 5 часов	31 490
3		4	8	256 (SSD)	14"/1920x1080 пикс.	до 12 часов	60 990
4		4	16	1000 (HDD)	17"/1920x1080 пикс.	до 3 часов	65 990
5		8	16	1000 (HDD) + 256 (SSD)	17"/1920x1080 пикс.	до 11 часов	109 990

КАК МЫ ПРИШЛИ К РЕШЕНИЮ?

	Кол-во ядер	RAM (ГБ)	Объем жесткого диска (ГБ)	Диагональ/разрешение	Работа от аккумулятора	Цена (руб.)
1		4	16	1000 (HDD)+ 128 (SSD) Датасет (обучающая выборка)	часов	?
2		2	4	500 (HDD)	15"/1920x1080 пикс.	до 5 часов
3		4	8	256 (SSD)	14"/1920x1080 пикс.	до 12 часов
4		4	16	1000 (HDD)	17"/1920x1080 пикс.	до 3 часов
5		8	16	1000 (HDD) + 256 (SSD)	17"/1920x1080 пикс.	до 11 часов

КАК МЫ ПРИШЛИ К РЕШЕНИЮ?

	Кол-во ядер	RAM (ГБ)	Объем жесткого диска (ГБ)	Диагональ/разрешение	Работа от аккумулятора	Цена (руб.)	
1		4	16	1000 (HDD)+ 128 (SSD)	17"/1920x1080 пикс.	до 6 часов	Таргеты (ответы)
2		2	4	500 (HDD)	15"/1920x1080 пикс.	до 5 часов	31 490
3		4	8	256 (SSD)	14"/1920x1080 пикс.	до 12 часов	60 990
4		4	16	1000 (HDD)	17"/1920x1080 пикс.	до 3 часов	65 990
5		8	16	1000 (HDD) + 256 (SSD)	17"/1920x1080 пикс.	до 11 часов	109 990

КАК МЫ ПРИШЛИ К РЕШЕНИЮ?

	Кол-во ядер	RAM (ГБ)	Объем жесткого диска (ГБ)	Диагональ/разрешение	Работа от аккумулятора	Цена (руб.)	
1		4	16	1000 (HDD)+ 128 (SSD)	17"/1920x1080 пикс.	до 6 часов	?
2		2	4	500 (HDD)	15"/1920x1080 пикс.	до 5 часов	31 490
3		4	8	256 (SSD)	14"/1920x1080 пикс.	до 12 часов	60 990
4		4	16	1000 (HDD)	17"/1920x1080 пикс.	до 3 часов	65 990
5		8	16	1000 (HDD) + 256 (SSD)	17"/1920x1080 пикс.	до 11 часов	109 990

Матрица объектов-признаков

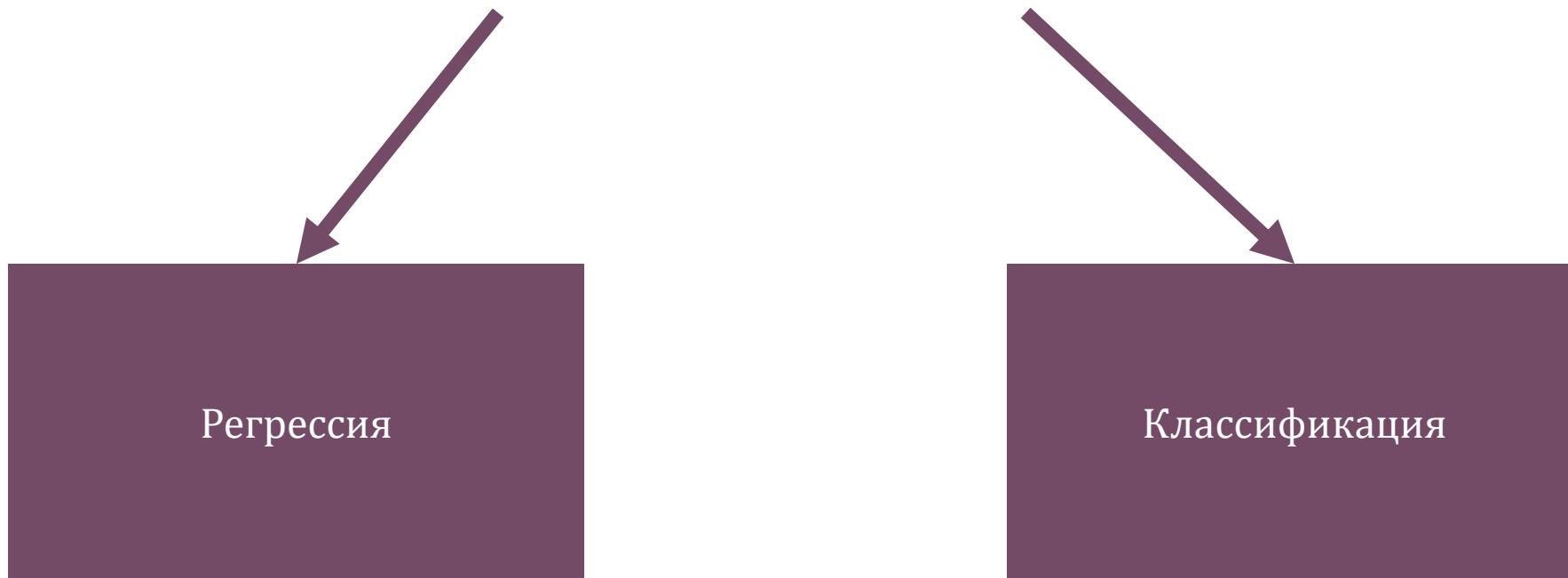
Матрица ответов

Задача машинного обучения. Нам дано множество объектов X и множество ответов Y , а также неизвестная закономерность $f : X \rightarrow Y$.

Помимо этого у нас есть обучающая выборка — подмножество X , для которого известно значение f . Чтобы обучить модель, мы отправляем обучающую выборку в алгоритм машинного обучения, который строит приближение истинной закономерности f .

ОБУЧЕНИЕ С УЧИТЕЛЕМ

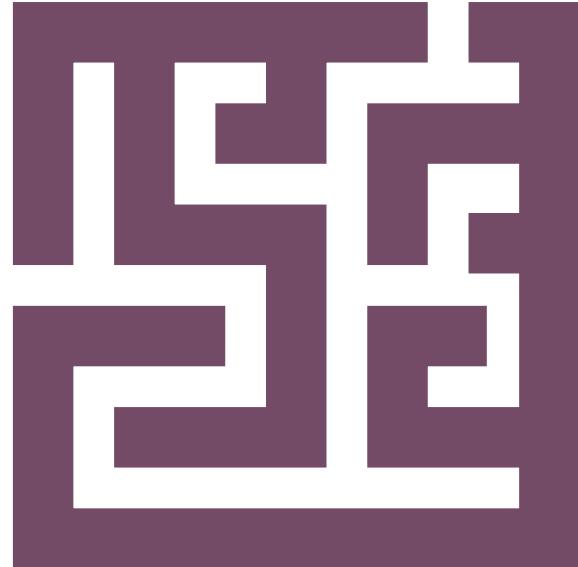
(ПРАВИЛЬНЫЕ ОТВЕТЫ ДЛЯ КАЖДОГО ОБЪЕКТА ОБУЧАЮЩЕЙ ВЫБОРКИ ЗАРАНЕЕ ИЗВЕСТНЫ)



+ задачи о порождении новых
объектов на основе существующих

КЛАССИФИКАЦИЯ

- Это задача машинного обучения, где модель предсказывает категориальную метку, то есть принадлежность к одному или нескольким классам.



ЩЕНОК ИЛИ БУЛКА?

Щенок



Булка



РЕГРЕССИЯ

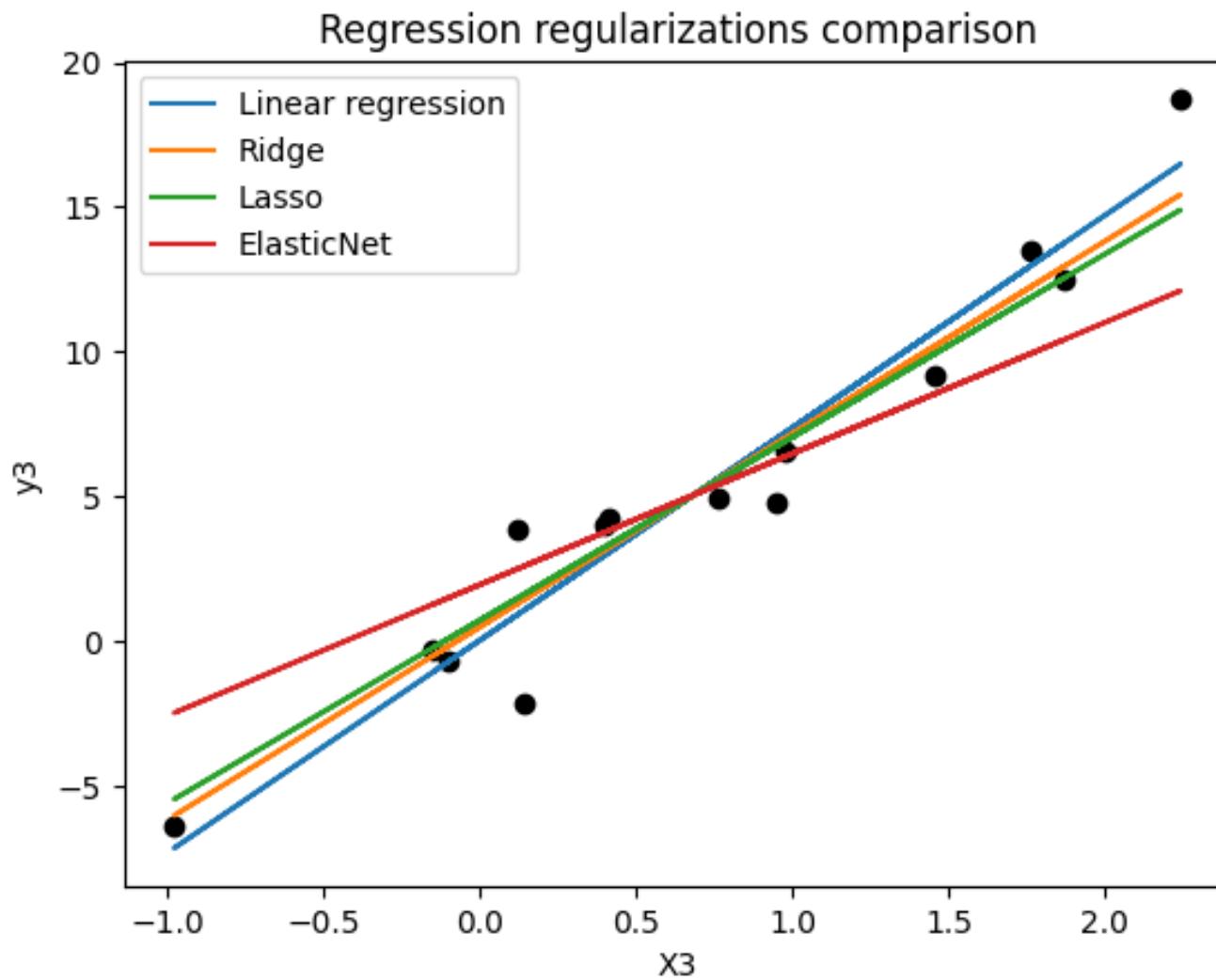
- Это задача машинного обучения, при которой модель предсказывает количественное или непрерывное значение на основе входных данных.



ЧТО ТАКОЕ РЕГРЕССИЯ (ЛИНЕЙНАЯ)?

Температура (x, °C)	Продано лимонада (y, стаканов)
20	38
22	46
23	50
27	52
35	60

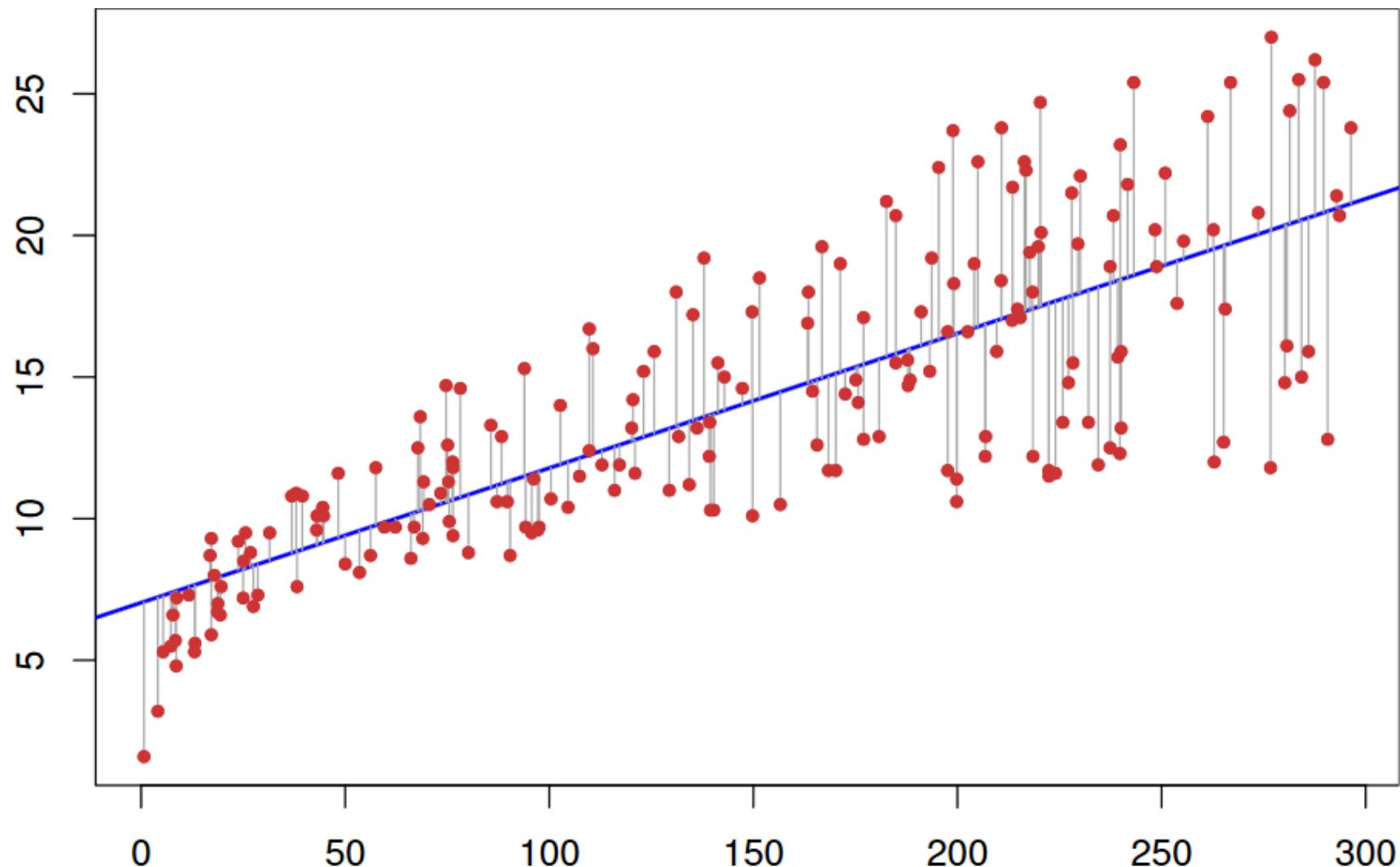
- Представим, что мы владельцы магазина, где продаётся лимонад. Мы заметили, что чем жарче на улице, тем больше продаётся стаканов лимонада. Мы решил собрать данные и начать предугадывать количество проданных стаканов в зависимости от температуры воздуха.
- Вот собранные данные за 5 дней:



ЗАДАЧА ЛИНЕЙНОЙ РЕГРЕССИИ – НАЙТИ ПРЯМУЮ, КОТОРАЯ НАИЛУЧШИМ ОБРАЗОМ (В СРЕДНЕМ) ПРИБЛИЖАЕТ ВСЕ ТОЧКИ НА ГРАФИКЕ.

РЕШИМ КАК ВЗРОСЛЫЕ?

- Метод наименьших квадратов — это способ **проводить прямую линию через точки на графике так, чтобы она максимально точно описывала все точки**.
Мы ищем такую прямую, которая **наименее ошибается в среднем**.
-



ОТКЛОНЕНИЕ ТОЧКИ ОТ ЛИНИИ — ОШИБКА

ЕСЛИ ТЫ НАРИСУЕШЬ ЛИНИЮ И ПОСМОТРИШЬ НА КАЖДУЮ ТОЧКУ, ТО РАЗНИЦА МЕЖДУ РЕАЛЬНЫМ ЗНАЧЕНИЕМ y И ЗНАЧЕНИЕМ, КОТОРОЕ ДАЁТ ПРЯМАЯ — ЭТО ОШИБКА (ОСТАТОК):

$$\text{ошибка}_i = y_i - (b + ax_i)$$

Например:

- реальное значение $y_i = 50$
- модель предсказала $\hat{y}_i = 52$
- ошибка = $50 - 52 = -2$

ПОЧЕМУ «НАИМЕНЬШИХ КВАДРАТОВ»?

Мы хотим, чтобы все ошибки в совокупности были как можно меньше.

Но если просто сложить ошибки, положительные и отрицательные будут взаимно уничтожаться.

Поэтому:

- ◆ Мы **возводим каждую ошибку в квадрат**, чтобы получить только положительные значения:

$$(y_i - (b + ax_i))^2$$

- ◆ Потом **суммируем** их все:

$$\text{Сумма квадратов ошибок} = \sum_{i=1}^n (y_i - (b + ax_i))^2$$

- ◆ Метод **наименьших квадратов** — это процесс **нахождения таких a и b , которые минимизируют** эту сумму.

ФОРМУЛЫ: ОТКУДА БЕРУТСЯ «А» И «В»?

- Формулы для коэффициентов получены из математики (минимизации функции), но мы будем их просто использовать:

Формула для угла наклона (a):

$$a = \frac{n \sum(xy) - \sum x \sum y}{n \sum(x^2) - (\sum x)^2}$$

- Эта формула показывает, **насколько сильно** и в каком направлении влияет x на y .
- Если $a > 0$, прямая "вверх" — рост x вызывает рост y .

ФОРМУЛЫ: ОТКУДА БЕРУТСЯ «A» И «B»?

- n — это количество точек (наблюдений) в наших данных.

Формула для сдвига (b):

$$b = \frac{\sum y - a \sum x}{n}$$

- Это точка, где прямая **пересекает ось y** , когда $x = 0$.
 - Показывает **базовое значение y** , если бы x был нулём.
-

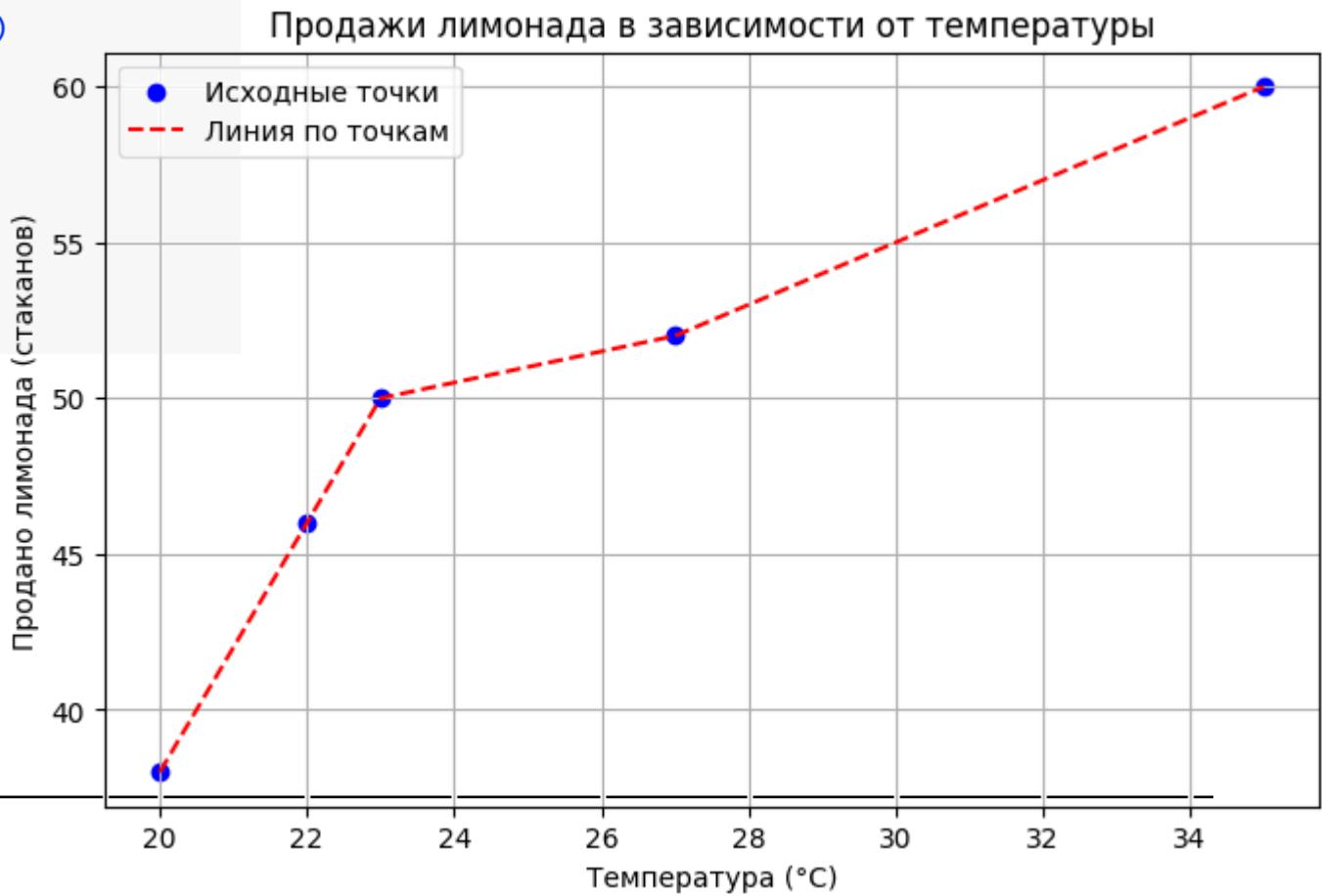
```
import matplotlib.pyplot as plt

x = [20, 22, 23, 27, 35]
y = [38, 46, 50, 52, 60]

plt.figure(figsize=(8, 5))
plt.scatter(x, y, color='blue', label='Исходные точки')
plt.plot(x, y, color='red', linestyle='--', label='Линия по точкам')

plt.title("Продажи лимонада в зависимости от температуры")
plt.xlabel("Температура (°C)")
plt.ylabel("Продано лимонада (стаканов)")
plt.grid(True)
plt.legend()

plt.show()
```



ЗАДАЧУ РЕШИМ С ПОМОЩЬЮ PYTHON

```
# Наши наблюдения. X -
x = [20, 22, 23, 27, 35]
y = [38, 46, 50, 52, 60]

# Количество точек
n = len(x)

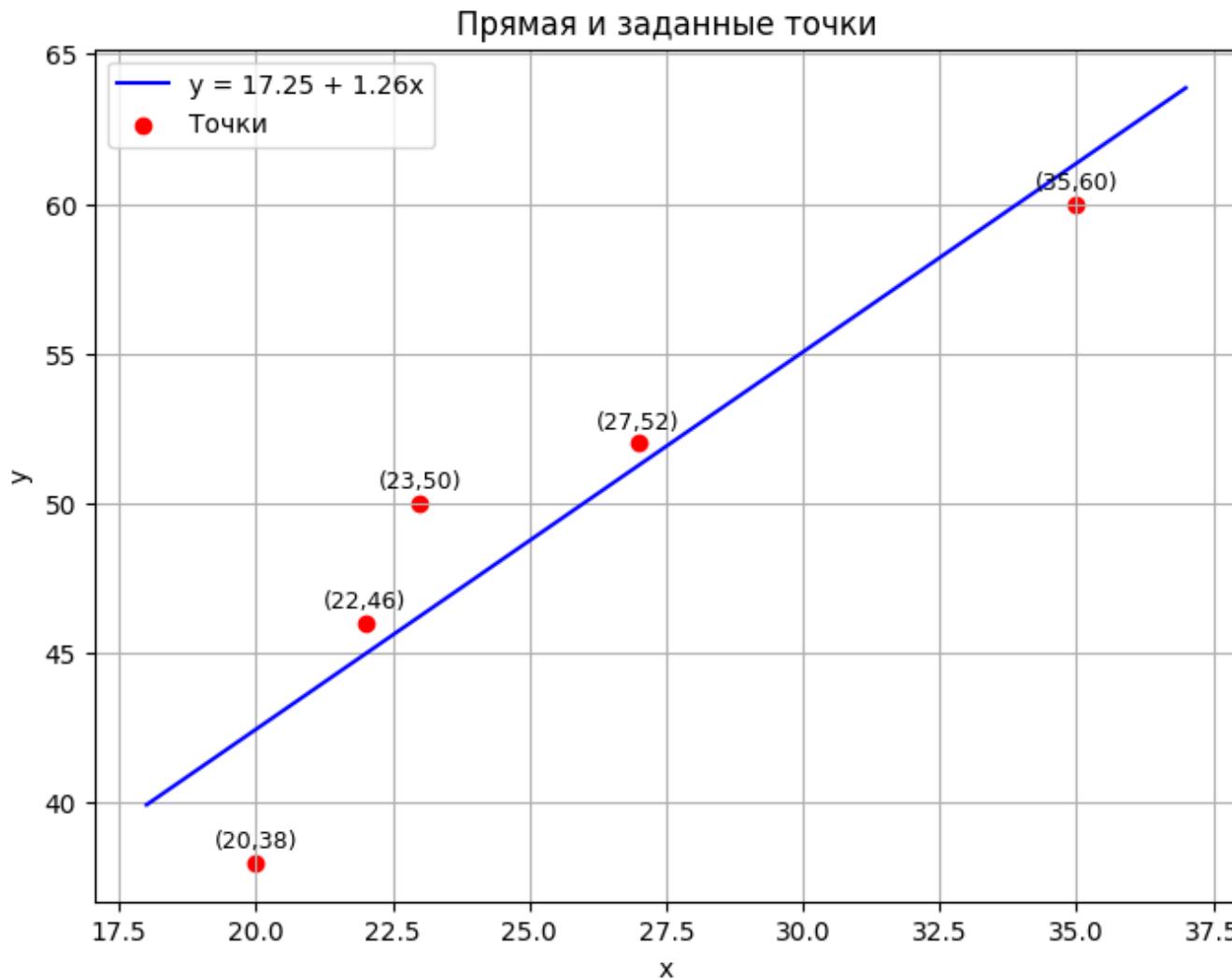
# Расчёт необходимых сумм
sum_x = sum(x)
sum_y = sum(y)
sum_xy = sum(xi * yi for xi, yi in zip(x, y))
sum_x2 = sum(xi**2 for xi in x)

# Формулы линейной регрессии
a = (n * sum_xy - sum_x * sum_y) / (n * sum_x2 - sum_x**2)
b = (sum_y - a * sum_x) / n

# Результат
print(f"Коэффициент наклона (a): {a}")
print(f"Свободный член (b): {b}")
print(f"Уравнение линейной регрессии: y = {b:.2f} + {a:.2f}x")
```

Коэффициент наклона (a): 2.0
Свободный член (b): 0.0
Уравнение линейной регрессии: y = 0.00 + 2.00x

РЕШЕНИЕ ЗАДАЧИ НА ГРАФИКЕ



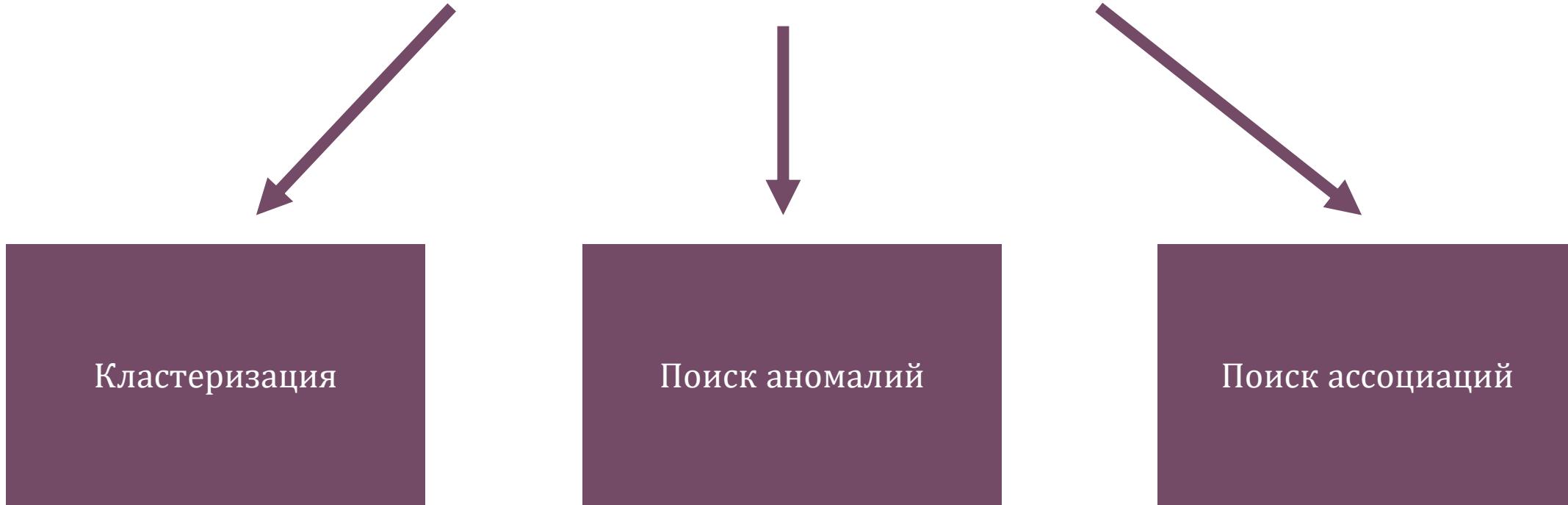
А КАКОЙ ТИП ЗАДАЧА ПРО НОУТБУКИ?

	Кол-во ядер	RAM (Гб)	Объем жесткого диска (Гб)	Диагональ/разрешение	Работа от аккумулятора	Цена (руб.)
1		4	16	1000 (HDD)+ 128 (SSD)	17"/1920x1080 пикс.	до 6 часов
2		2	4	500 (HDD)	15"/1920x1080 пикс.	до 5 часов
3		4	8	256 (SSD)	14"/1920x1080 пикс.	до 12 часов
4		4	16	1000 (HDD)	17"/1920x1080 пикс.	до 3 часов
5		8	16	1000 (HDD) + 256 (SSD)	17"/1920x1080 пикс.	до 11 часов
						?

Эта задача – задача многомерной регрессии.

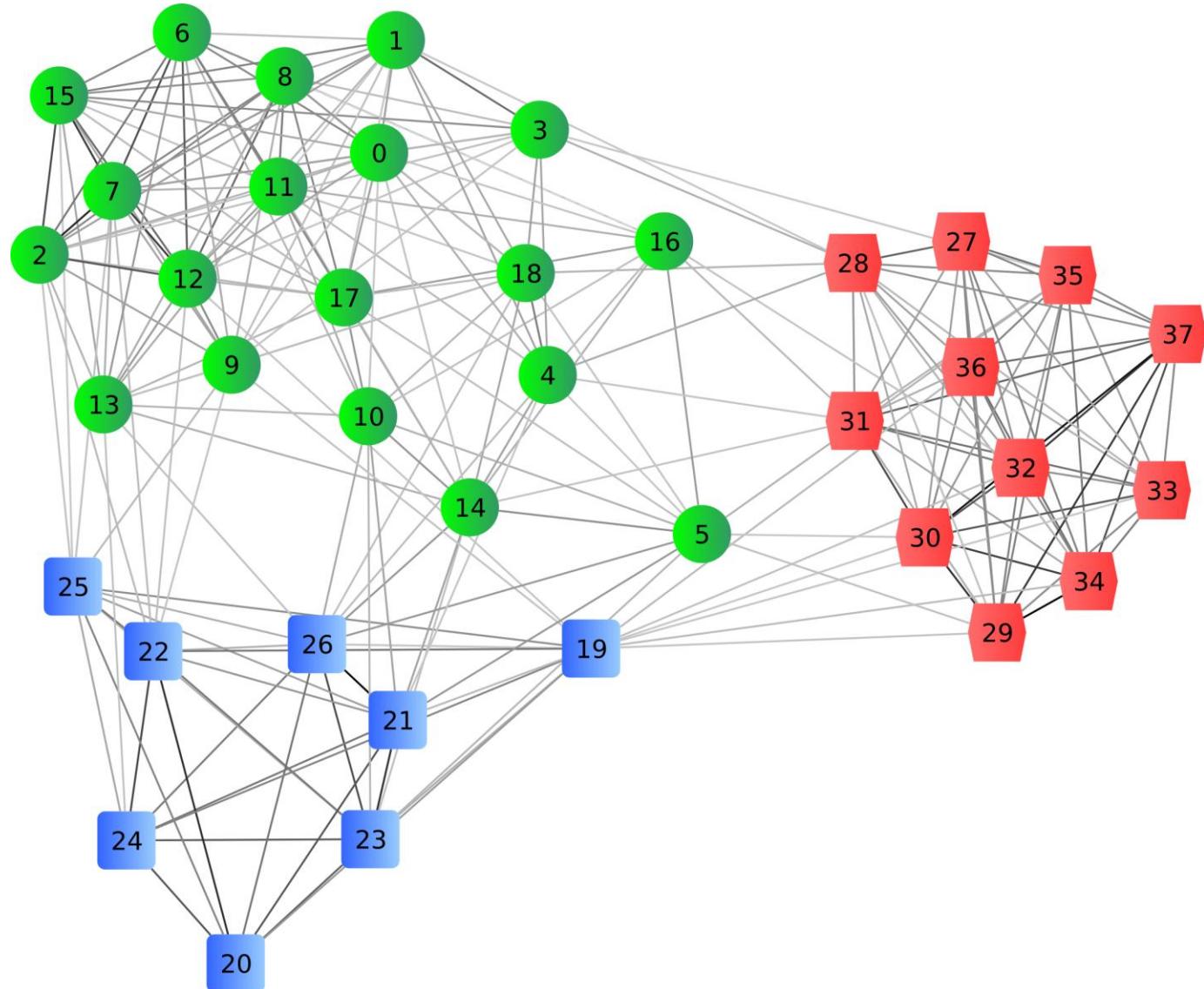
Решением этой задачи будет нахождение всего массива коэффициентов.

ОБУЧЕНИЕ БЕЗ УЧИТЕЛЯ



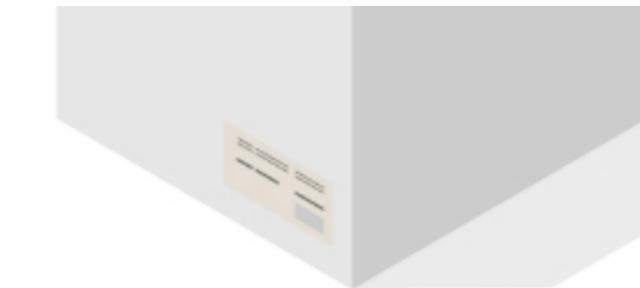
КЛАСТЕРИЗАЦИЯ

Это задача разделения данных на кластеры по общим признакам.





Честно говоря, я вообще **не знаю**, что там лежит, но объедини мне их по общим признакам 🤷‍♂️🤷‍♂️🤷‍♂️



Я тебе дал фотографии кошек, собак, коров, коз, ослов, медведей, лис, бобров, ящериц, змей, антилоп, снежных барсов, бабочек...

Рассортируй их, пожалуйста 🙏🙏🙏

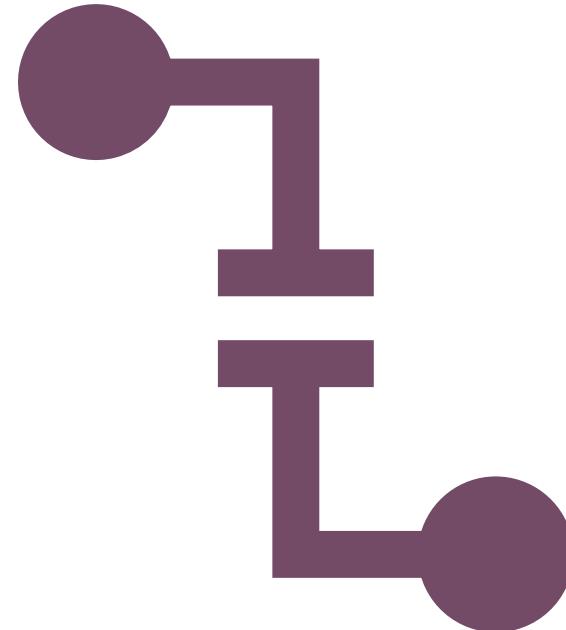
КЛАСТЕРИЗАЦИЯ VS КЛАССИФИКАЦИЯ

*Во время решения задач по **классификации** мы **заведомо знаем группы**, по которым будем распределять данные.*

*Во время решения задач по **кластеризации** мы **ничего не знаем о данных и пытаемся объединить их по общим признакам.***

ПОИСК АССОЦИАЦИЙ

- В этой задаче модель ищет взаимосвязи между существующими объектами.
- Классический пример — взаимосвязь продуктов в корзине покупателя. Например, к молоку и яйцам рекомендательная система магазина может посоветовать взять муку. Чтобы получить такой «совет», нужно проанализировать, какие комбинации продуктов встречаются чаще всего.



ПОИСК АНОМАЛИЙ

Примеры:

- Мошеннические транзакции
- Повышение температуры в доме (система умный дом)
- Слишком высокое/низкое давление (умные часы)
- Вход в аккаунт с другого устройства в другой стране, подозрительный IP



ПОЧЕМУ «ПОИСК АНОМАЛИЙ» НЕ ОБУЧЕНИЕ С УЧИТЕЛЕМ?

- Поиск аномалий требует длительного анализа ситуации. При обучении с учителем не получится показать модели аномалию сразу, потому что мы не знаем, как она выглядит. И, соответственно, у алгоритма этого знания тоже не будет. Поэтому когда модель встретится с аномалией, то, скорее всего, либо не сможет его задетектировать, либо правильно интерпретировать.
 - При обучении без учителя модель может определять выбросы и аномалии. Аномалия — это принципиально новые данные, которых ещё не было в выборке, они возникают в реальном времени. Выброс — значение, которое редко встречается в наших данных.
-

НА ПОДУМАТЬ. ОПРЕДЕЛИТЕ ТИП СЛЕДУЮЩИХ ЗАДАЧ

Предсказание курса евро к доллару на следующий день.

Стилизация текста. Например, перевод на бюрократический язык: «Пиппина и Мерри похитили!» ↠ «Граждане Тук, Перегрин Паладинович, 2990 года рождения, и Брендибак, Мериадок Сарадокович, 2982 года рождения, были похищены неустановленными лицами».

Детектирование котиков на изображении.

Обучение робокота запрыгивать на стол из произвольной позы.

Поиск наборов товаров, которые посетители супермаркета часто покупают вместе.

ПЕРЕОБУЧЕНИЕ

- Качество работы модели машинного обучения на обучающих и тестовых данных может очень сильно различаться. Для этого рассмотрим задачу регрессии, в которой у каждого объекта всего один признак. Попытаемся восстановить закономерность на обучающих данных, приближая точки многочленами степеней 1,2,3,5.

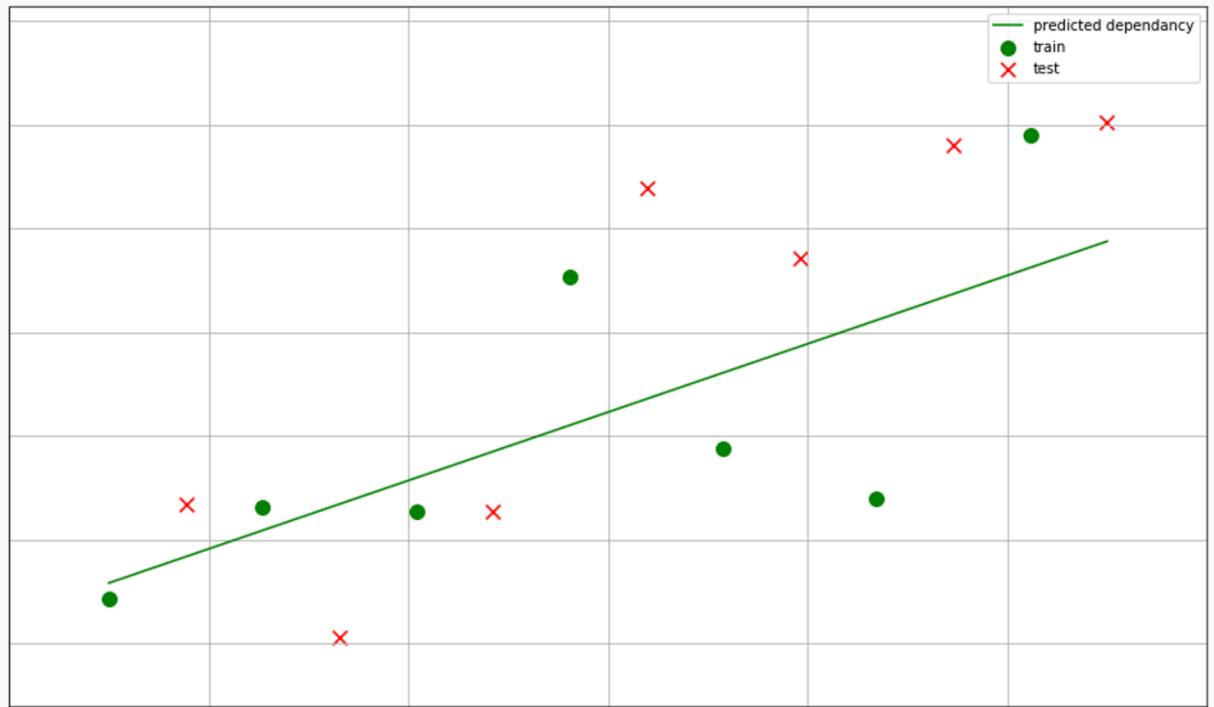


Рис. Приближение многочленом степени 1.

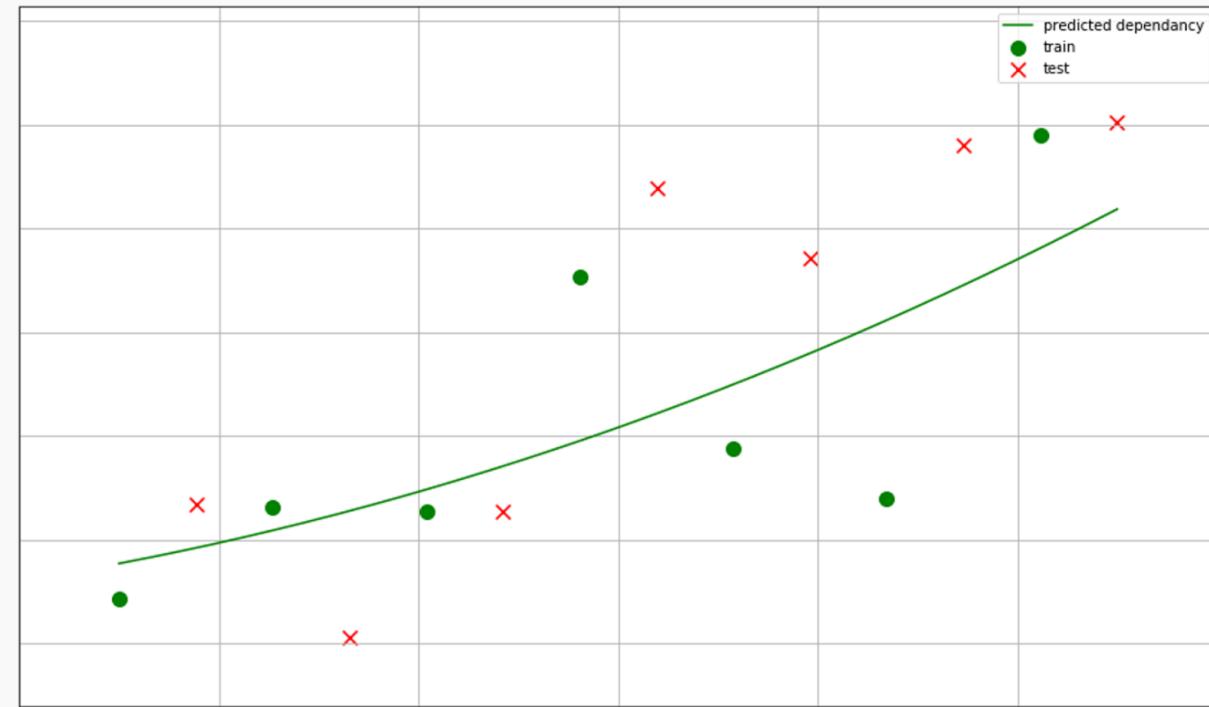


Рис. Приближение многочленом степени 2.

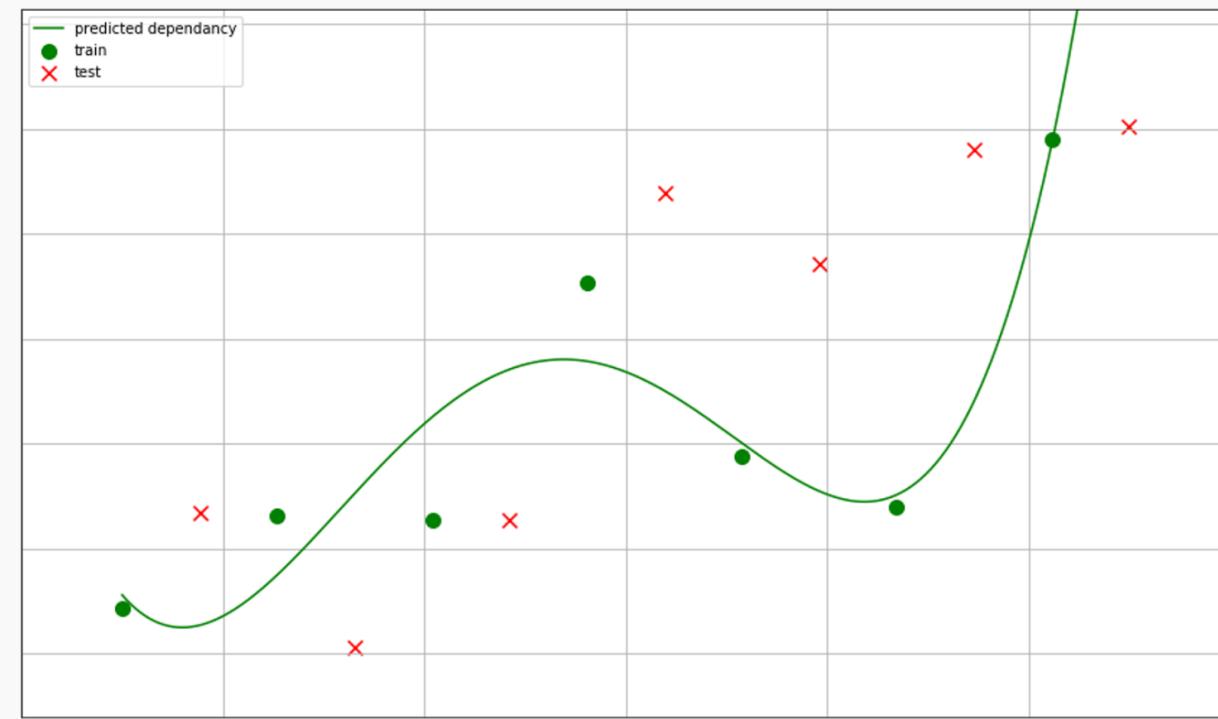


Рис. Приближение многочленом степени 3.

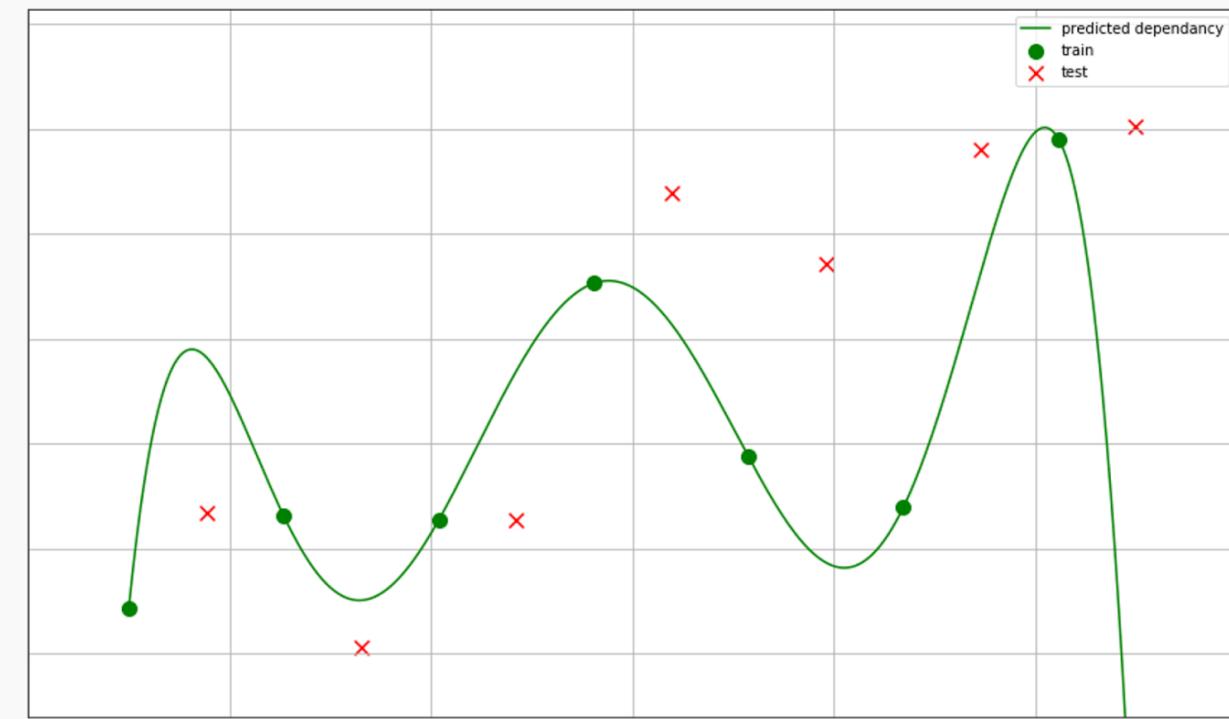
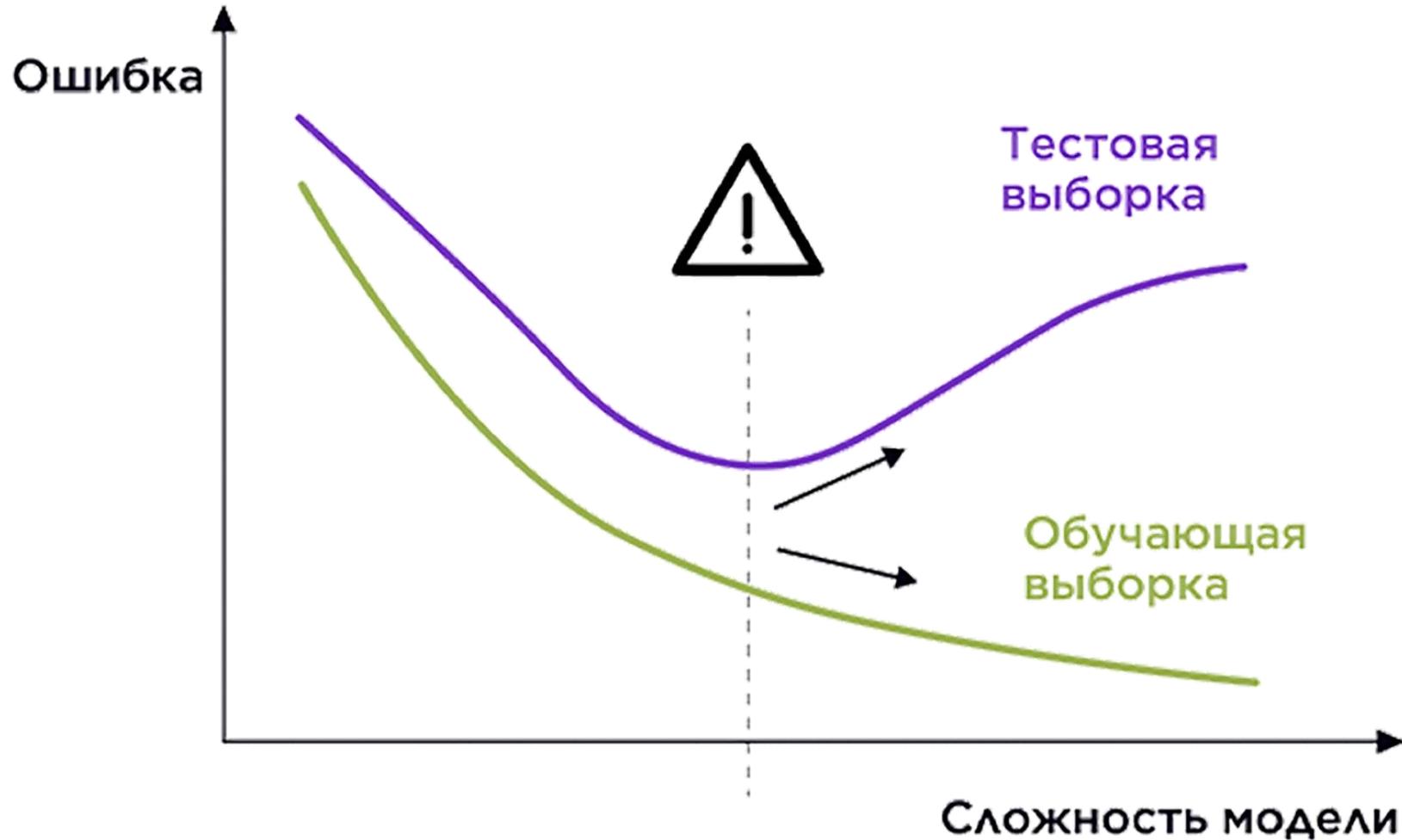


Рис. Приближение многочленом степени 5.

- Видно, что с ростом степени многочлена качество на обучающем множестве увеличивается, а на тестовом драматически ухудшается.



ЭФФЕКТ ПЕРЕОБУЧЕНИЯ

А КАК ОЦЕНИТЬ НАШУ МОДЕЛЬ?

- На практике невозможно посмотреть на графики и выбрать лучшую модель. Необходимо выражать качество наших моделей в численном виде. Для этого существуют функционалы качества (метрики, функции потерь). Для разных задач и моделей используются разные метрики. Остановимся на стандартных метриках в задачах классификации и регрессии.
- Для задачи классификации в качестве метрики часто используют процент верно угаданных ответов. Эту метрику называют accuracy. Например, если речь идёт о предсказании выживших пассажиров «Титаника», то вычисляется по формуле количество пассажиров, для которых модель верно предугадала исход разделить на общее количество пассажиров.

Для задачи регрессии используют две основные функции потерь.

! Определение

Среднеквадратичная ошибка (Mean Squared Error, MSE):

$$MSE = \frac{1}{l} \sum_{i=1}^l (\hat{y}_i - y_i)^2$$

! Определение

Средняя абсолютная ошибка (Mean Absolute Error, MAE):

$$MAE = \frac{1}{l} \sum_{i=1}^l |\hat{y}_i - y_i|$$

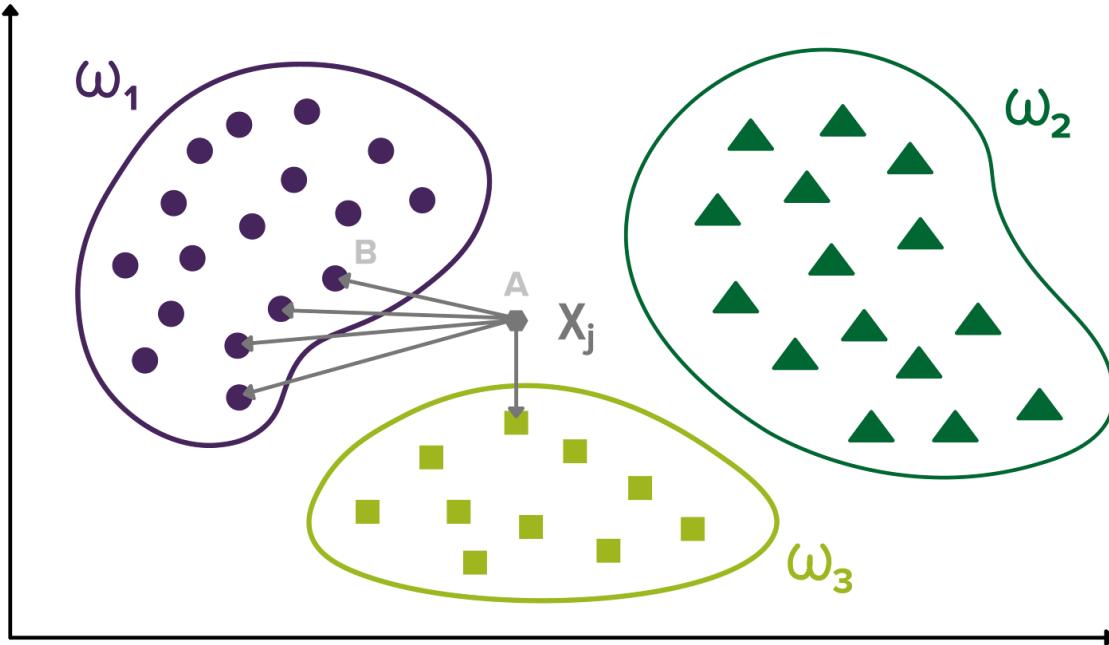
$$F = G \frac{m_1 m_2}{d^2}$$

$$\phi(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{(x-\mu)^2}{2\sigma^2}}$$

НАШ ПЕРВЫЙ
АЛГОРИТМ
МАШИННОГО
ОБУЧЕНИЯ

$$\frac{df}{dt} = \lim_{h \rightarrow 0} \frac{f(t+h) - f(t)}{h}$$

МЕТОД К-БЛИЖАЙШИХ СОСЕДЕЙ



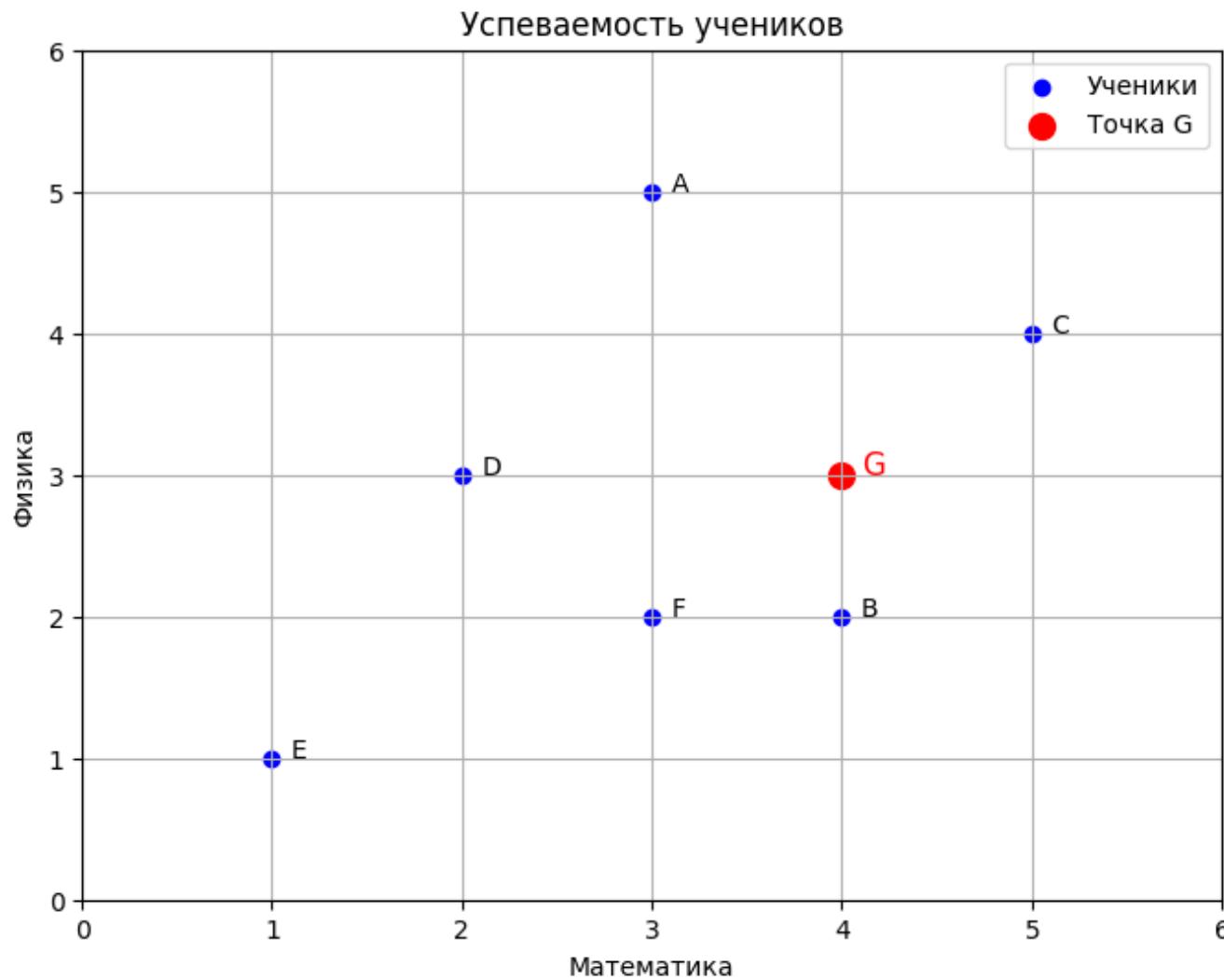
- Сформулируем, как работает этот алгоритм, на примере задачи классификации на три класса: фиолетовый, салатовый и зеленый. Объекты изображены фигурами на картинке.

Ученик	Математика (x)	Физика (y)
A	3	5
B	4	2
C	5	4
D	2	3
E	1	1
F	3	2
G	4	3

РЕШИМ НАГЛЯДНО.

ЗАДАЧА 1

- В классе учится 6 учеников, и их результаты на тестах по математики и физике представлены в виде координат на плоскости. Каждая координата соответствует оценке ученика по двум предметам: математике и физике.
- Предположим, что новый ученик G имеет оценки: математика = 4, физика = 3.
- Учитель рассаживает детей в классе по успеваемости. Ваша задача – определить, к какому из учеников A, B, C, D, E или F ближе всего посадят ученика G.



ЗАДАЧА 1. РЕШЕНИЕ

Расположим точки на графике.

ЗАДАЧА 1. РЕШЕНИЕ

- Вычислить расстояние между учеником G и каждым из других учеников (A, B, C, D, E и F) можно используя формулу Евклидова расстояния: $d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$

ЗАДАЧА 1. РЕШЕНИЕ

```
import math

students = {
    'A': (3, 5),
    'B': (4, 2),
    'C': (5, 4),
    'D': (2, 3),
    'E': (1, 1),
    'F': (3, 2)
}

G = (4, 3)

def euclidean_distance(p1, p2):
    return math.sqrt((p2[0] - p1[0]) ** 2 + (p2[1] - p1[1]) ** 2)

distances = {}
for name, coords in students.items():
    distance = euclidean_distance(G, coords)
    distances[name] = distance
    print(f"Расстояние от G до {name}: {distance:.2f}")

nearest_student = min(distances, key=distances.get)
print(f"\nБлижайший к ученику G – ученик {nearest_student}, расстояние = {distances[nearest_student]:.2f}")
```

Расстояние от G до A: 2.24
Расстояние от G до B: 1.00
Расстояние от G до C: 1.41
Расстояние от G до D: 2.00
Расстояние от G до E: 3.61
Расстояние от G до F: 1.41

Ближайший к ученику G – ученик B, расстояние = 1.00

ЗАДАЧА 2. УСЛОЖНИМ ДО K=3

- В классе учится 6 учеников, и их результаты на тестах по математики и физике представлены в видео координат на плоскости. Каждая координата соответствует оценке ученика по двум предметам: математике и физике.
- Предположим, что новый ученик G имеет оценки: математика = 3, физика = 4.
- Учитель рассаживает детей в классе по успеваемости. Ваша задача – определить, к каким **трём ученикам** из A, B, C, D, E или F ближе всего посадят ученика G.

Ученик	Математика (x)	Физика (y)
A	3	5
B	4	2
C	5	4
D	2	3
E	1	1
F	3	2
G	3	4

ЗАДАЧА 2. РЕШЕНИЕ

- Главное отличие от предыдущей задачи, что в данном случае мы должны ранжировать расстояния. Первые три студента, находящихся ближе всего к искомому – и есть наш ответ.

```
distances.sort(key=lambda x: x[1])  
  
top_3 = distances[:3]  
  
print("\nТри ближайших ученика к G:")  
for name, dist in top_3:  
    print(f"{name} (расстояние = {dist:.2f})")
```

Расстояние от G до A: 2.24
Расстояние от G до B: 1.00
Расстояние от G до C: 1.41
Расстояние от G до D: 2.00
Расстояние от G до E: 3.61
Расстояние от G до F: 1.41

Три ближайших ученика к G:
B (расстояние = 1.00)
C (расстояние = 1.41)
F (расстояние = 1.41)

ЗАДАЧА 3. ДОБАВИМ КЛАССИФИКАЦИИ

- В классе учатся 8 учеников, и их результаты на тесте по математике и физике представлены в виде координат на плоскости. Каждая координата соответствует оценке ученика по двум предметам: математике и физике. Учеников можно классифицировать на две группы: «Гуманитарии» и «Технари».
 - Предположим, что новый ученик I имеет оценки: Математика = 4, Физика = 4.
 - Ваша задача — определить, к какой группе относится ученик I, используя метод k-ближайших соседей.
-

ЗАДАЧА 3. ДОБАВИМ КЛАССИФИКАЦИИ

Ученик	Математика (x)	Физика (y)	Группа
A	5	7	Гуманитарии
B	2	3	Гуманитарии
C	9	6	Технари
D	4	5	Гуманитарии
E	1	1	Гуманитарии
F	6	2	Технари
G	7	8	Технари
H	3	4	Гуманитарии
I	4	4	???

ЗАДАЧА 3. РЕШЕНИЕ

Ученик I классифицирован как: Гуманитарии

Ближайшие соседи:

Группа: Гуманитарии, расстояние: 1.00

Группа: Гуманитарии, расстояние: 1.00

Группа: Гуманитарии, расстояние: 2.24

```
import math
from collections import Counter

students = {
    'A': (5, 7, 'Гуманитарии'),
    'B': (2, 3, 'Гуманитарии'),
    'C': (9, 6, 'Технари'),
    'D': (4, 5, 'Гуманитарии'),
    'E': (1, 1, 'Гуманитарии'),
    'F': (6, 2, 'Технари'),
    'G': (7, 8, 'Технари'),
    'H': (3, 4, 'Гуманитарии')
}

I = (4, 4)

def euclidean(p1, p2):
    return math.sqrt((p2[0] - p1[0]) ** 2 + (p2[1] - p1[1]) ** 2)

def classify_knn(data, point, k=3):
    distances = []
    for name, (x, y, group) in data.items():
        dist = euclidean(point, (x, y))
        distances.append((dist, group))

    distances.sort(key=lambda x: x[0])

    k_nearest = distances[:k]

    groups = [group for _, group in k_nearest]
    vote = Counter(groups).most_common(1)[0][0]

    return vote, k_nearest

result_group, neighbors = classify_knn(students, I, k=3)

print(f"Ученик I классифицирован как: {result_group}")
print("\nБлижайшие соседи:")
for dist, group in neighbors:
    print(f"Группа: {group}, расстояние: {dist:.2f}")
```

КАК ПОДОБРАТЬ НАИЛУЧШИЙ «K»?

Нечётное k предпочтительно при бинарной классификации (2 класса), чтобы не было ничьей.

Обычно пробуют разные значения k (например, от 1 до \sqrt{n}) и:

Маленький k (например, 1) может "шуметь" (чувствительно к выбросам).

Большой k сглаживает, но может терять локальные особенности.

Проверяют точность на отложенной выборке (валидация).

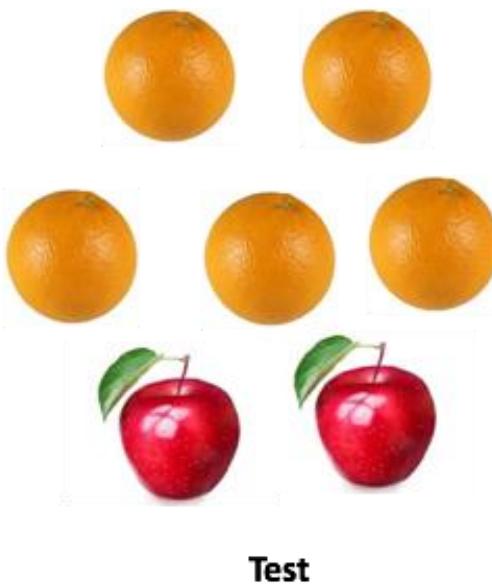
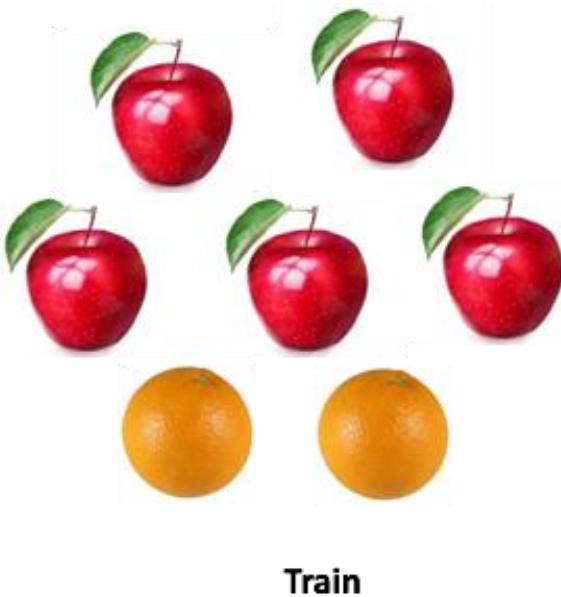
Используют кросс-валидацию, чтобы найти k, дающий наименьшую ошибку.

ЗАДАЧА 3. ДОБАВИМ КРОСС-ВАЛИДАЦИЮ



- **Кросс-валидация** — это метод оценки качества модели машинного обучения (в т.ч. k-ближайших соседей), при котором данные **многократно разбиваются** на обучающую и тестовую части, чтобы **надежно проверить, как будет работать модель на новых данных**.

ОБУЧАЮЩАЯ И ТЕСТОВАЯ ВЫБОРКИ



- Обучающая выборка используется для обучения модели, а тестовая выборка – для оценки ее производительности. Обычно данные делятся в пропорции 70-80% на обучение и 20-30% на тестирование.
- Процесс разделения может быть случайным или стратифицированным. Случайное разделение означает, что каждый объект данных имеет одинаковый шанс попасть в обучающую или тестовую выборку. Стратифицированное разделение гарантирует, что в каждой из выборок будет представлено одинаковое соотношение классов.

КАК РАБОТАЕТ (НА ПРИМЕРЕ K-FOLD CROSS-VALIDATION)

Разбиваем все данные на **k** равных частей (например, на 5 частей — 5-fold).

Повторяем 5 раз:

- каждый раз берём 1 часть как **тест**, остальные 4 — как **обучающая выборка**.

Обучаем и тестируем модель на каждом разбиении.

Считаем **среднюю точность** по всем 5 итерациям — это и будет надёжная оценка качества.

ПРИМЕР 5-FOLD КРОСС-ВАЛИДАЦИИ:

Разбиение	Тестовая часть	Обучающая часть
Fold 1	1-я часть	2-я + 3-я + 4-я + 5-я
Fold 2	2-я часть	1-я + 3-я + 4-я + 5-я
Fold 3	3-я часть	1-я + 2-я + 4-я + 5-я
Fold 4	4-я часть	1-я + 2-я + 3-я + 5-я
Fold 5	5-я часть	1-я + 2-я + 3-я + 4-я



ПРИМЕНЕНИЕ К K-NN

Псевдокод:

для k в диапазоне от 1 до 10:

делаем кросс-валидацию

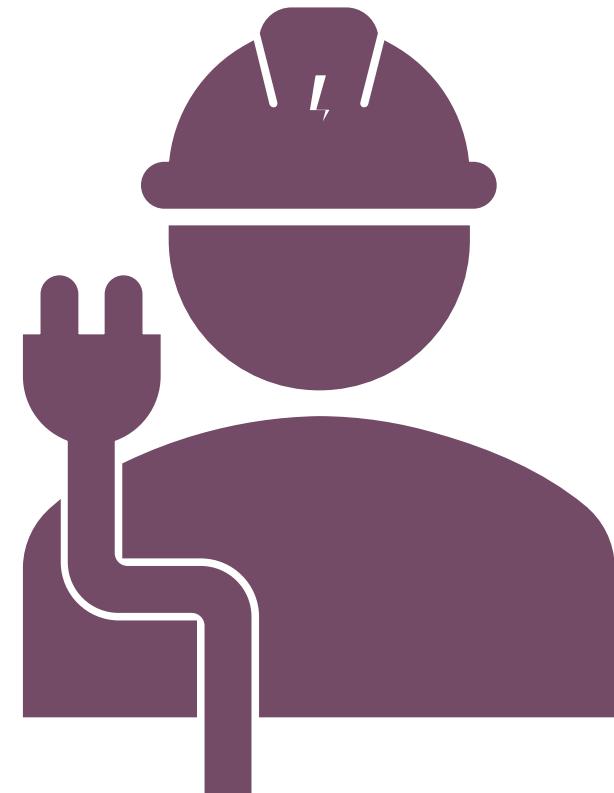
измеряем среднюю точность

выбираем k с лучшей точностью

- Тестируем разные значения k (количество соседей) внутри кросс-валидации, и выбираем то, при котором классификация будет наиболее точной.

ЗАДАЧА 3. ДОБАВЛЯЕМ LOOCV

- Есть 8 учеников с координатами и группой (гуманитарии/технари).
- Мы будем использовать leave-one-out кросс-валидацию (LOOCV) — это частный случай k-fold, где каждый элемент по очереди становится тестовым, а остальные — обучающими.
- Для каждого k (кол-во соседей) мы оценим точность классификации.



ЗАДАЧА 3. РЕШЕНИЕ С LOOCV

Для каждого k от 1 до 7:

- По очереди исключаем по одному ученику из данных (LOOCV),
- Классифицируем его с помощью k -NN на остальных,
- Сравниваем предсказание с реальной группой.
- Считаем точность (доля правильных предсказаний) при каждом k .
- Выводим лучший k .

```
def euclidean(p1, p2):
    return math.sqrt((p2[0] - p1[0])**2 + (p2[1] - p1[1])**2)

def knn(train_data, point, k):
    distances = []
    for x, y, group in train_data:
        dist = euclidean(point, (x, y))
        distances.append((dist, group))
    distances.sort(key=lambda x: x[0])
    k_nearest = distances[:k]
    votes = [group for _, group in k_nearest]
    return Counter(votes).most_common(1)[0][0]

def loocv(data, max_k):
    results = {}
    keys = list(data.keys())

    for k in range(1, max_k + 1):
        correct = 0
        for i in range(len(keys)):
            test_key = keys[i]
            test_point = data[test_key]
            train = [data[keys[j]] for j in range(len(keys)) if j != i]
            predicted = knn(train, test_point[:2], k)
            actual = test_point[2]
            if predicted == actual:
                correct += 1
        accuracy = correct / len(data)
        results[k] = accuracy
    return results

accuracies = loocv(students, max_k=7)

print("Точность при разных значениях k:")
for k, acc in accuracies.items():
    print(f"k = {k}: Точность = {acc:.2f}")

# Оптимальный k
best_k = max(accuracies, key=accuracies.get)
print(f"\nЛучшее значение k: {best_k} (точность = {accuracies[best_k]:.2f})")
```

ЗАДАЧА 3.

РЕШЕНИЕ С LOOCV

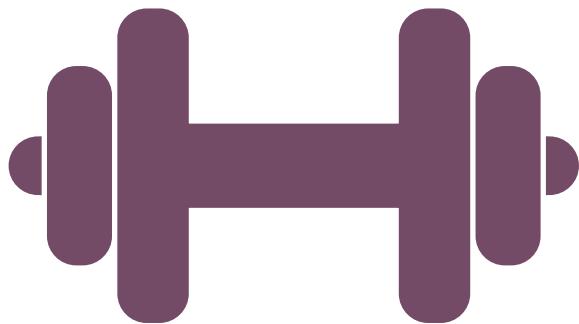
Точность при разных значениях k:

k = 1: точность = 0.75
k = 2: точность = 0.75
k = 3: точность = 0.75
k = 4: точность = 0.75
k = 5: точность = 0.62
k = 6: точность = 0.62
k = 7: точность = 0.62

🔍 Лучшее значение k: 1 (точность = 0.75)



ЗАДАЧА 4. БОЛЬШЕ КЛАССИФИКАЦИЙ, БОЛЬШЕ ПРИЗНАКОВ

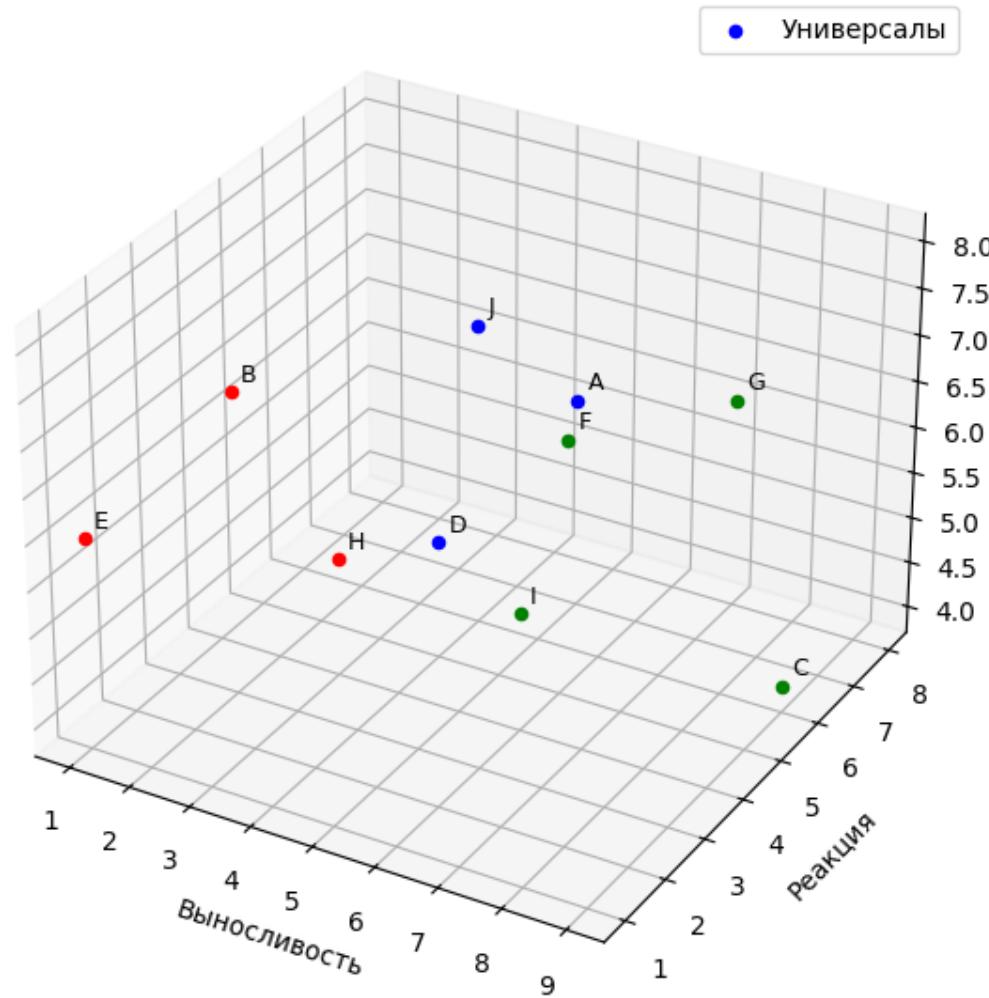


- В спортивной академии оценивают спортсменов по трём параметрам:
 -  **Выносливость**
 -  **Реакция**
 -  **Ловкость**
- Каждому спортсмену присваивается категория:
 - **Спринтеры** — если $\text{Выносливость} < 4$
 - **Универсалы** — если $\text{Выносливость от } 4 \text{ до } 7$
- Спортсмен **K** имеет характеристики:
 - **Выносливость = 4**
 - **Реакция = 4**
 - **Ловкость = 6**
- **Вопрос:** К какой категории отнести K?

ЗАДАЧА 4. ДАТАСЕТ

Спортсмен	Выносливость	Реакция	Ловкость	Категория
A	5	7	6	Универсалы
B	2	3	7	Спринтеры
C	9	6	4	Сталевики
D	4	5	5	Универсалы
E	1	1	6	Спринтеры
F	8	2	8	Сталевики
G	7	8	6	Сталевики
H	3	4	5	Спринтеры
I	6	4	5	Сталевики
J	4	6	7	Универсалы

Визуализация спортсменов в 3D-пространстве



ЗАДАЧА 4. РЕШЕНИЕ

Дано:

Множество тренировочных объектов:

$$D = \{(x_1^{(i)}, x_2^{(i)}, x_3^{(i)}, y^{(i)})\}_{i=1}^N$$

где:

- $x_j^{(i)}$ — значение j-го признака у i-го спортсмена,
- $y^{(i)} \in \{\text{Спринтеры, Сталевики, Универсалы}\}$ — метка класса.

ЗАДАЧА 4. РЕШЕНИЕ

Выносливость (Endurance) — x_1

Реакция (Reaction) — x_2

Ловкость (Agility) — x_3

Метод k-NN:

Для нового объекта $x = (x_1, x_2, x_3)$:

1. Вычислить расстояние от него до всех объектов обучающей выборки:

$$d(x, x^{(i)}) = \sqrt{(x_1 - x_1^{(i)})^2 + (x_2 - x_2^{(i)})^2 + (x_3 - x_3^{(i)})^2}$$

2. Выбрать k ближайших по расстоянию объектов.
 3. Классифицировать по большинству среди этих k объектов.
-

ЗАДАЧА 4. РЕШЕНИЕ

ЗАДАЧА 4. РЕШЕНИЕ

```
import math
from collections import Counter
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

# Данные спортсменов
athletes = {
    'A': (5, 7, 6, 'Универсалы'),
    'B': (2, 3, 7, 'Спринтеры'),
    'C': (9, 6, 4, 'Сталевики'),
    'D': (4, 5, 5, 'Универсалы'),
    'E': (1, 1, 6, 'Спринтеры'),
    'F': (8, 2, 8, 'Сталевики'),
    'G': (7, 8, 6, 'Сталевики'),
    'H': (3, 4, 5, 'Спринтеры'),
    'I': (6, 4, 5, 'Сталевики'),
    'J': (4, 6, 7, 'Универсалы')
}

# Евклидово расстояние в 3D
def euclidean_3d(p1, p2):
    return math.sqrt(sum((a - b) ** 2 for a, b in zip(p1, p2)))
```

```
# k-NN алгоритм
def knn_classify(train_data, new_point, k):
    distances = []
    for val in train_data.values():
        coords = val[:3]
        label = val[3]
        dist = euclidean_3d(new_point, coords)
        distances.append((dist, label))

    distances.sort(key=lambda x: x[0])
    k_nearest = distances[:k]
    votes = [label for _, label in k_nearest]

    most_common = Counter(votes).most_common(1)[0][0]
    return most_common

# LOOCV и матрица ошибок
def evaluate_loocv_confusion(data, k):
    true_labels = []
    pred_labels = []
    keys = list(data.keys())

    for i in range(len(keys)):
        test_key = keys[i]
        test_sample = data[test_key]
        new_point = test_sample[:3]
        true_label = test_sample[3]

        train_data = {k: v for j, (k, v) in enumerate(data.items()) if j != i}
        pred = knn_classify(train_data, new_point, k)

        true_labels.append(true_label)
        pred_labels.append(pred)

    accuracy = sum(1 for t, p in zip(true_labels, pred_labels) if t == p) / len(true_labels)
    return accuracy, true_labels, pred_labels
```

ЗАДАЧА 4. РЕШЕНИЕ

```
# Поиск лучшего k
def search_best_k(data, max_k=10):
    accuracies = {}
    for k in range(1, max_k + 1):
        acc, _, _ = evaluate_loocv_confusion(data, k)
        accuracies[k] = acc
        print(f"k = {k}: Accuracy = {acc:.2f}")
    return accuracies
```

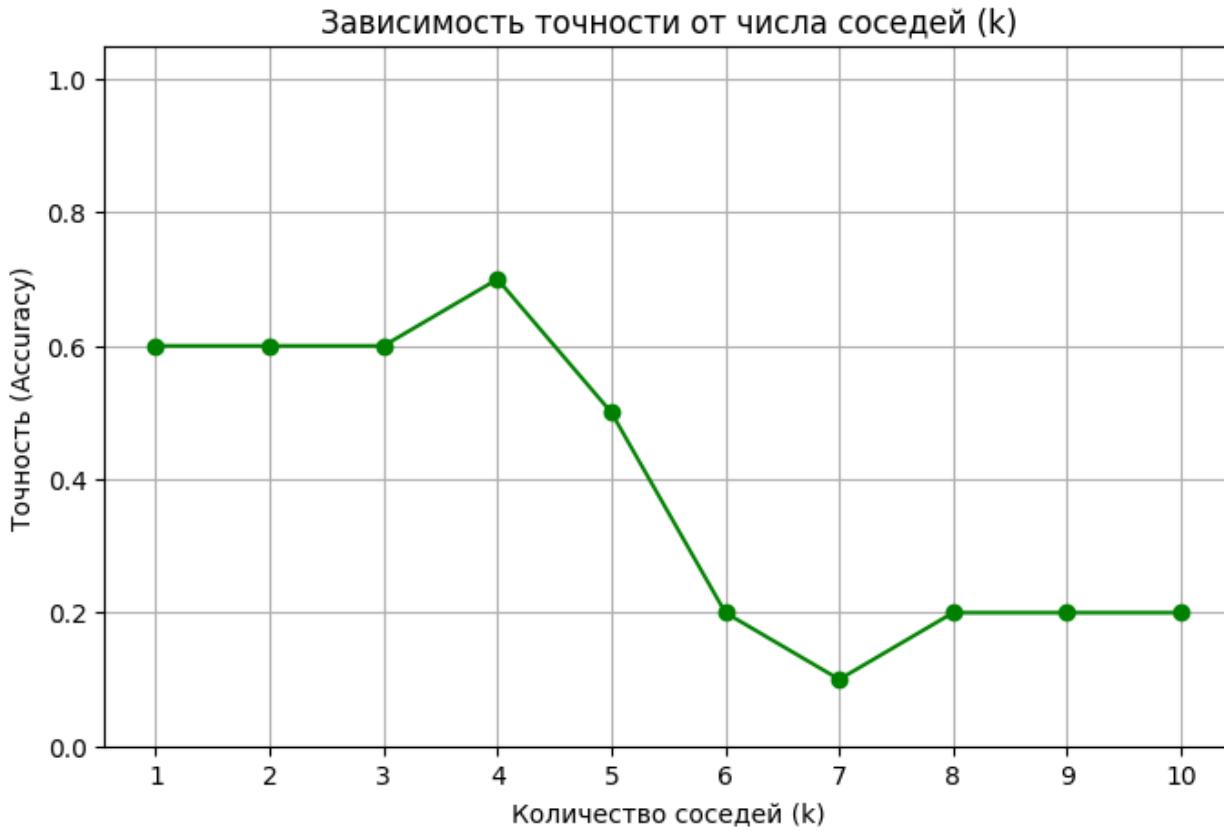
```
accuracies = search_best_k(athletes, max_k=10)
plot_accuracies(accuracies)
```

```
#💡 Лучшее значение k:
best_k = max(accuracies, key=accuracies.get)
print(f"\n💡 Лучший k = {best_k} с точностью {accuracies[best_k]:.2f}")

new_point = (4, 4, 6)
new_label = knn_classify(athletes, new_point, best_k)
print(f"\n⭐ Новая точка {new_point} относится к классу: '{new_label}'")
```

💡 Лучший k = 4 с точностью 0.70

⭐ Новая точка (4, 4, 6) относится к классу: 'Универсалы'



k = 1: Accuracy = 0.60
k = 2: Accuracy = 0.60
k = 3: Accuracy = 0.60
k = 4: Accuracy = 0.70
k = 5: Accuracy = 0.50
k = 6: Accuracy = 0.20
k = 7: Accuracy = 0.10
k = 8: Accuracy = 0.20
k = 9: Accuracy = 0.20
k = 10: Accuracy = 0.20

ЗАДАЧА 4. РЕШЕНИЕ

Время (с)	Скорость (км/ч)
0	0
1	10
2	22
3	33
4	41
5	53
6	63
7	70
8	82
9	90
10	101
11	110
12	120
13	128
14	137

ЗАДАЧА 5. ЛИНЕЙНАЯ РЕГРЕССИЯ С ВЫДЕЛЕНИЕМ СВОБОДНОГО ЧЛЕНА РАВНОГО 0 (ЧЕРЕЗ НАЧАЛО КООРДИНАТ)

- Машина ускоряется, и скорость (км/ч) зависит от времени (сек). Предскажите скорость через 15 секунд.

```
x4 = np.array([0,1,2,3,4,5,6,7,8,9,10,11,12,13,14])
y4 = np.array([0,10,22,33,41,53,63,70,82,90,101,110,120,128,137])

a4 = np.sum(x4*y4) / np.sum(x4**2)

print(f"Задача 4:")
print(f"Уравнение: Скорость = {a4:.2f} * Время (без свободного члена)")

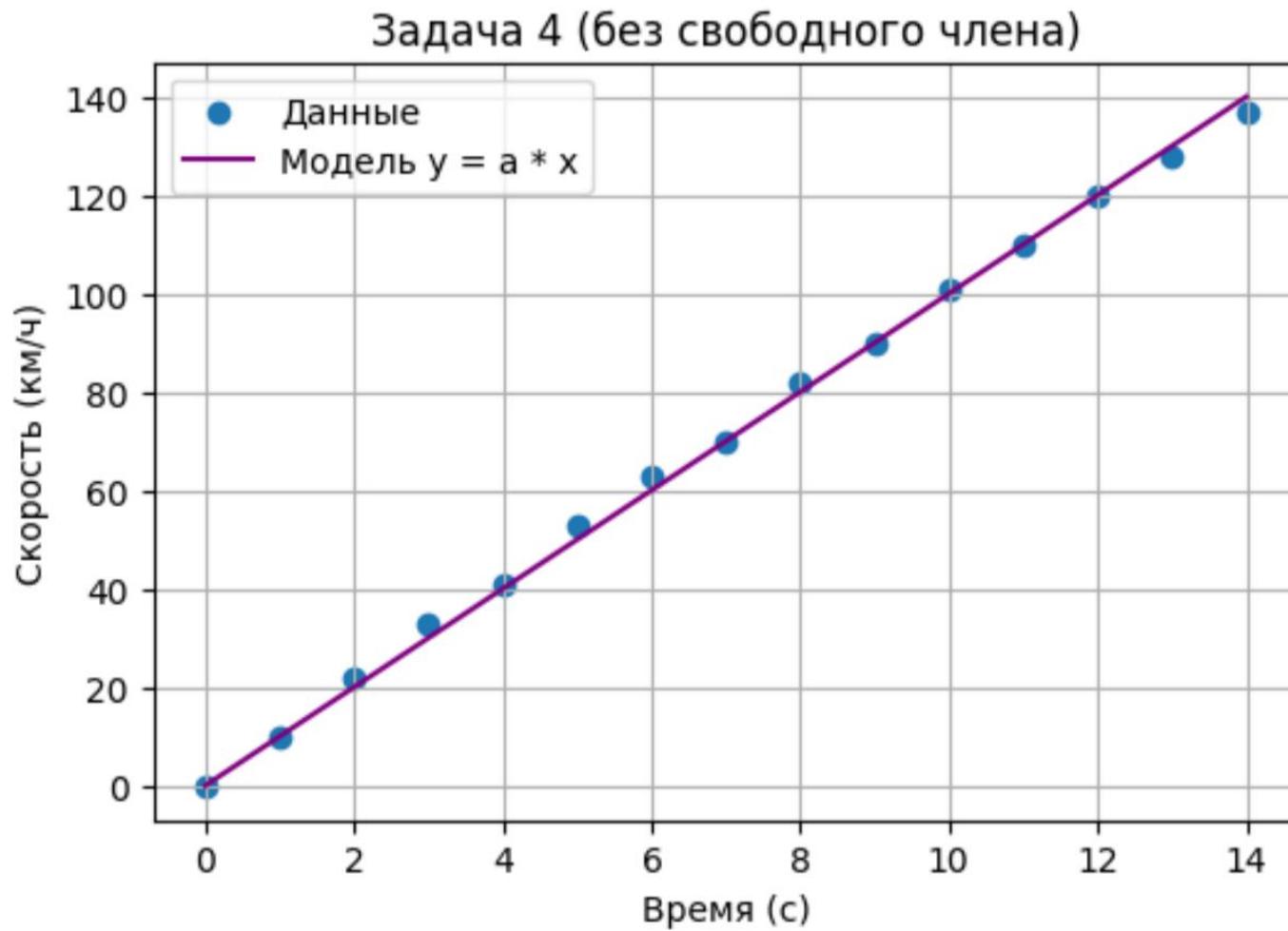
# Предсказание для времени 15 секунд
x4_test = 15
y4_pred = a4 * x4_test
print(f"Предсказание скорости при времени {x4_test} с: {y4_pred:.2f} км/ч\n")

plt.figure(figsize=(6,4))
plt.scatter(x4, y4, label='Данные')
plt.plot(x4, a4*x4, color='purple', label='Модель y = a * x')
plt.title('Задача 4 (без свободного члена)')
plt.xlabel('Время (с)')
plt.ylabel('Скорость (км/ч)')
plt.legend()
plt.grid(True)
plt.show()
```

ЗАДАЧА 5. РЕШЕНИЕ

Уравнение: Скорость = 10.01 * Время (без свободного члена)

Предсказание скорости при времени 15 с: 150.12 км/ч



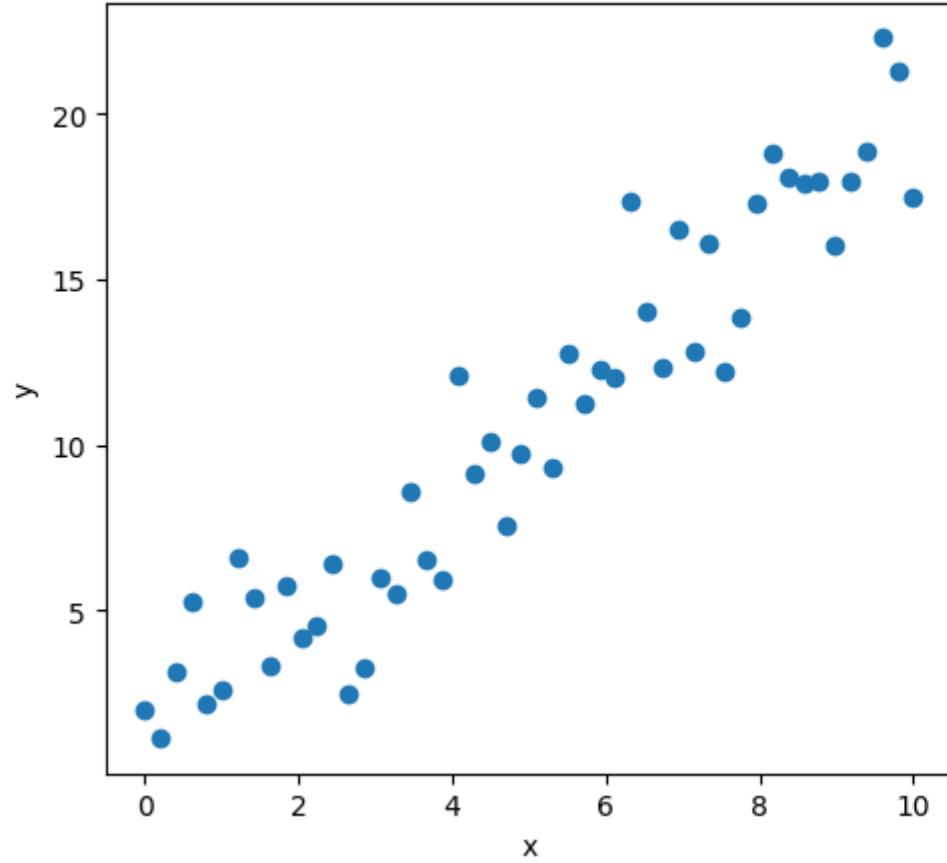
ЗАДАЧА 5. РЕШЕНИЕ

ЗАДАЧА 6. ЛИНЕЙНАЯ РЕГРЕССИЯ С ОШИБКАМИ (ШУМ В ДАННЫХ)

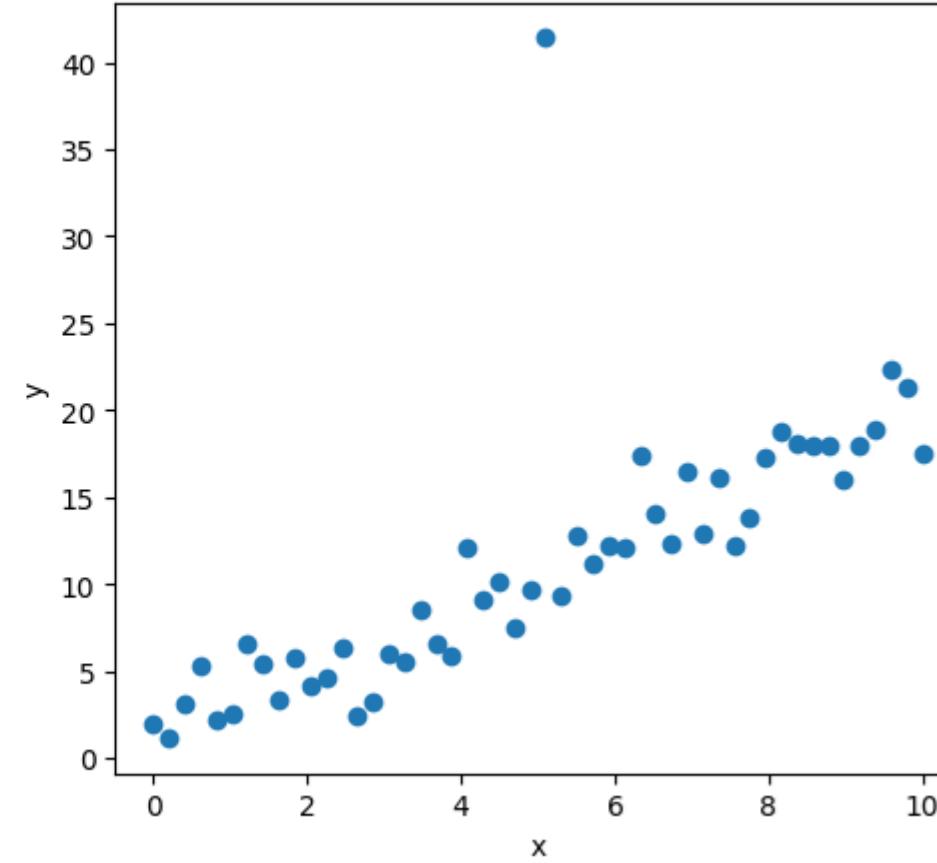
- Время выполнения домашнего задания (мин) зависит от количества страниц, но периодически при выполнении возникают ошибки, которые нужно обязательно исправлять. Предскажите время выполнения домашнего задания при заданных 16 страницах.

Страницы	Время (мин)
1	15
2	25
3	38
4	40
5	55
6	60
7	70
8	78
9	85
10	95
11	105
12	110
13	120
14	128
15	135

Данные с шумом



Данные с выбросом



ЧТО ТАКОЕ ШУМ В ДАННЫХ?

Шум, в отличие от выброса, не меняет общий тренд.

ЗАДАЧА 6.

РЕШЕНИЕ

```
x3 = np.arange(1,16)
y3 = np.array([15,25,38,40,55,60,70,78,85,95,105,110,120,128,135])

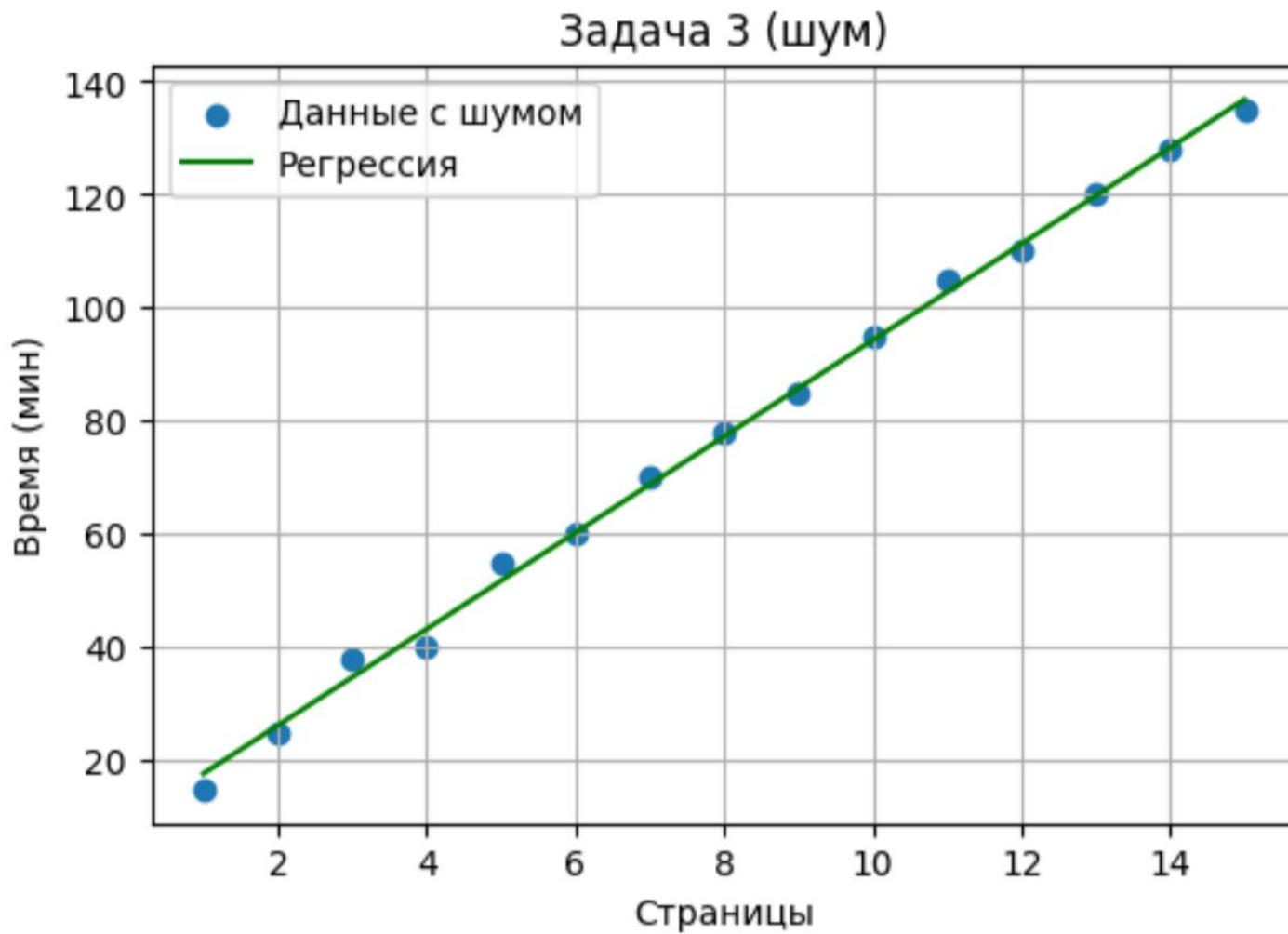
a3 = np.cov(x3, y3, bias=True)[0,1] / np.var(x3)
b3 = np.mean(y3) - a3 * np.mean(x3)

print(f"Задача 3:")
print(f"Уравнение: Время = {a3:.2f} * Страницы + {b3:.2f}")

# Предсказание для 16 страниц
x3_test = 16
y3_pred = a3 * x3_test + b3
print(f"Предсказание для {x3_test} страниц: {y3_pred:.2f} минут\n")

plt.figure(figsize=(6,4))
plt.scatter(x3, y3, label='Данные с шумом')
plt.plot(x3, a3*x3 + b3, color='green', label='Регрессия')
plt.title('Задача 3 (шум)')
plt.xlabel('Страницы')
plt.ylabel('Время (мин)')
plt.legend()
plt.grid(True)
plt.show()
```

Уравнение: Время = 8.51 * Страницы + 9.18
Предсказание для 16 страниц: 145.35 минут



ЗАДАЧА 6. РЕШЕНИЕ

ЗАДАЧА 7. МНОЖЕСТВЕННАЯ ЛИНЕЙНАЯ РЕГРЕССИЯ (ДВА ПРИЗНАКА)

Температура (°C)	Посетители	Продано мороженого (штук)
20	50	200
22	55	220
25	60	250
23	52	230

- Количество проданного мороженого зависит от температуры воздуха и количества посетителей кафе. Предскажите количество мороженого, если температура $33\text{ }^{\circ}\text{C}$, посетителей 95.

Формула линейной регрессии с двумя признаками x_1 и x_2 выглядит так:

$$y = w_0 + w_1x_1 + w_2x_2$$

где

- y — предсказанное значение,
 - x_1, x_2 — признаки (входные переменные),
 - w_0 — свободный член (константа),
 - w_1, w_2 — коэффициенты, показывающие влияние каждого признака на результат.
-

ЗАДАЧА 7. РЕШЕНИЕ

В задаче множественной линейной регрессии у нас есть:

$$\mathbf{y} = X\mathbf{w} + \varepsilon$$

где:

- \mathbf{y} — вектор целевых значений (размер $n \times 1$),
- X — матрица признаков размером $n \times m$ (число строк — количество объектов, число столбцов — число признаков плюс 1 для свободного члена),
- \mathbf{w} — вектор коэффициентов (размер $m \times 1$),
- ε — шум.

$$(X^T X)\mathbf{w} = X^T \mathbf{y}$$

— это **нормальное уравнение** для задачи линейной регрессии.

ЗАДАЧА 7. РЕШЕНИЕ

ЗАДАЧА 7. РЕШЕНИЕ

Данные:

$$X = \begin{bmatrix} 1 & 20 & 50 \\ 1 & 22 & 55 \\ 1 & 25 & 60 \\ 1 & 23 & 52 \end{bmatrix}, \quad y = \begin{bmatrix} 200 \\ 220 \\ 250 \\ 230 \end{bmatrix}$$

Итого

$$X^T X = \begin{bmatrix} 4 & 90 & 217 \\ 90 & 2038 & 4906 \\ 217 & 4906 & 11829 \end{bmatrix}$$

Шаг 1. Найдём матрицу $X^T X$:

$$X^T X = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 20 & 22 & 25 & 23 \\ 50 & 55 & 60 & 52 \end{bmatrix} \cdot \begin{bmatrix} 1 & 20 & 50 \\ 1 & 22 & 55 \\ 1 & 25 & 60 \\ 1 & 23 & 52 \end{bmatrix}$$

ЗАДАЧА 7. РЕШЕНИЕ

Шаг 2. Найдём вектор $X^T y$:

$$X^T y = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 20 & 22 & 25 & 23 \\ 50 & 55 & 60 & 52 \end{bmatrix} \cdot \begin{bmatrix} 200 \\ 220 \\ 250 \\ 230 \end{bmatrix}$$

Шаг 3. Решаем уравнение

$$(X^T X)w = X^T y$$

$$w = \begin{bmatrix} c \\ a \\ b \end{bmatrix}$$

где c — свободный член, a, b — коэффициенты при x_1 и x_2 .

Шаг 4. Находим w :

$$w = (X^T X)^{-1} X^T y$$

ЗАДАЧА 7. РЕШЕНИЕ

```
# Задача 5: Множественная регрессия (4 точки)
X5 = np.array([
    [1, 20, 50],
    [1, 22, 55],
    [1, 25, 60],
    [1, 23, 52]
])
y5 = np.array([200, 220, 250, 230])

Xt = X5.T
w = np.linalg.inv(Xt @ X5) @ Xt @ y5

print(f"Задача 5:")
print(f"Уравнение: Продажи = {w[0]:.2f} + {w[1]:.2f} * Температура + {w[2]:.2f} * Посетители")

# Предсказание для Температуры=33 и Посетителей=95
temp_test = 22
visitors_test = 53
y5_pred = w[0] + w[1]*temp_test + w[2]*visitors_test
print(f"Предсказание продаж при температуре {temp_test}°C и посетителях {visitors_test}")

# 3D график для задачи 5
temp_vals = np.linspace(19, 26, 30)
visitors_vals = np.linspace(48, 62, 30)
temp_grid, visitors_grid = np.meshgrid(temp_vals, visitors_vals)
y_pred_grid = w[0] + w[1]*temp_grid + w[2]*visitors_grid

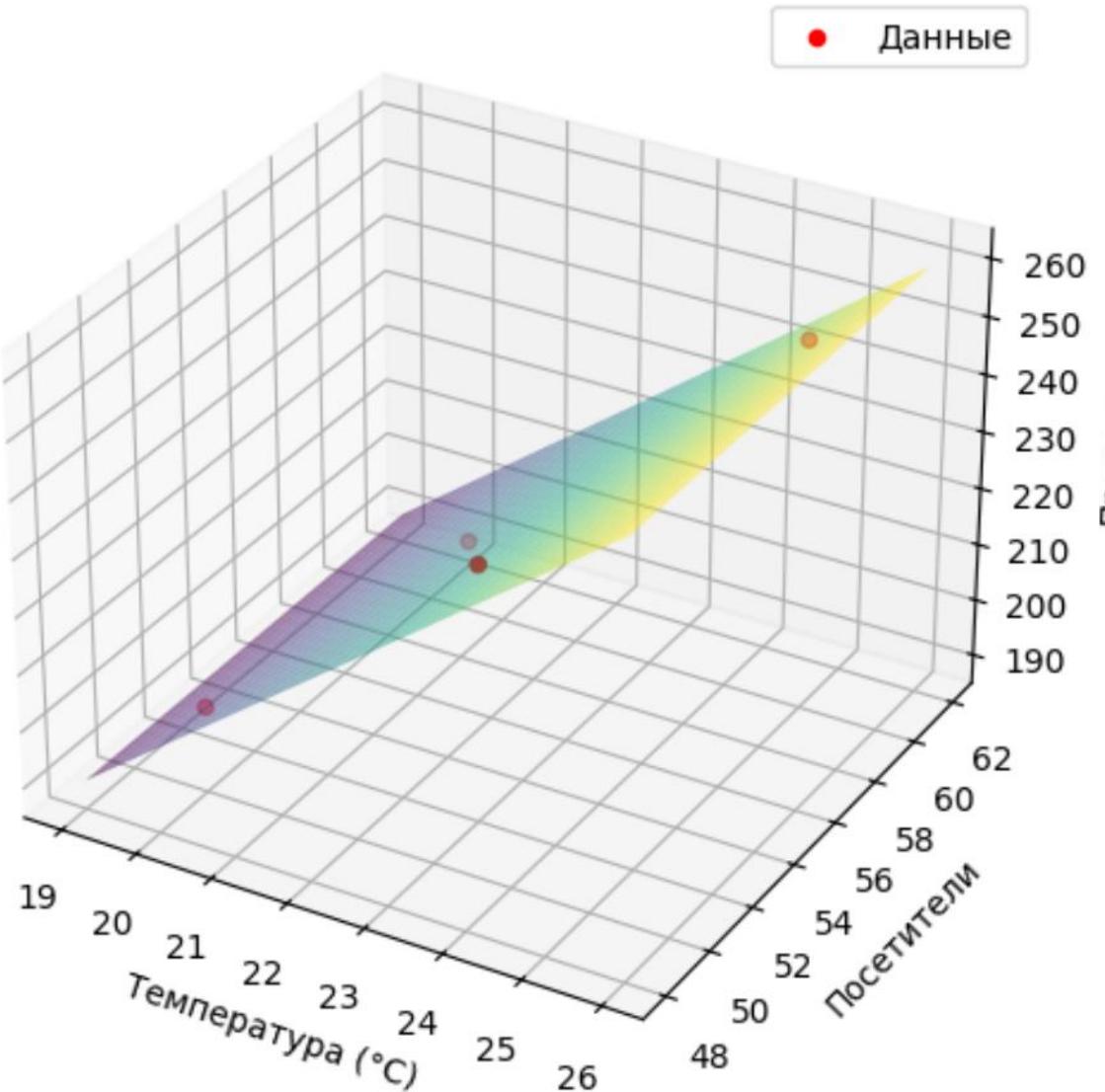
fig = plt.figure(figsize=(8,6))
ax = fig.add_subplot(111, projection='3d')

ax.scatter(X5[:,1], X5[:,2], y5, color='red', label='Данные')
ax.plot_surface(temp_grid, visitors_grid, y_pred_grid, alpha=0.5, cmap='viridis')

ax.set_xlabel('Температура (°C)')
ax.set_ylabel('Посетители')
ax.set_zlabel('Продажи')
ax.set_title('Задача 5: Множественная линейная регрессия (3D)')
ax.legend()
plt.grid(True)
plt.show()
```

Уравнение: Продажи = -0.00 + 10.00 * Температура + 0.00 * Посетители
Предсказание продаж при температуре 33°C и посетителях 95: 330.00

Задача 5: Множественная линейная регрессия (3D)



ЗАДАЧА 7. РЕШЕНИЕ

ЗАДАЧА 8. ЛИНЕЙНАЯ РЕГРЕССИЯ С ГРАДИЕНТНЫМ СПУСКОМ

- Стоимость билета в автобус зависит от расстояния в километрах. Известны данные по 15 поездкам. Предскажите стоимость билета на 80 км.

Расстояние (км)	Стоимость (руб.)
5	50
10	90
15	130
20	170
25	210
30	250
35	290
40	330
45	370
50	410
55	450
60	490
65	530
70	570
75	610

ГРАДИЕНТНЫЙ СПУСК?

- **Ключевые аспекты алгоритма градиентного спуска.** Градиент указывает направление наискорейшего возрастания функции, поэтому движение в противоположном направлении позволяет находить минимумы.
 - **Шаг градиентного спуска** ($\text{learning rate}, \alpha$) определяет размер корректировки весов на каждой итерации. От его значения зависит скорость сходимости алгоритма и вероятность "перепрыгивания" через минимум.
 - **Критерии остановки** определяют, когда алгоритм достиг приемлемого решения. Это может быть достижение максимального числа итераций, минимального изменения функции потерь между шагами или наблюдение за лоссом на валидационном наборе данных.
-

ШАГ ГРАДИЕНТНОГО СПУСКА

Шаг градиента (learning rate) — это как размер шага, который мы делаем, чтобы дойти до «лучшего» решения.

Представьте, что мы идём по холмистой местности вниз к самой низкой точке (минимуму).

Шаг градиента — это насколько далеко мы делаем шаг за один раз.

Если шаг слишком большой — можно перепрыгнуть минимальную точку и уйти дальше.

Если шаг слишком маленький — идти будешь очень медленно.

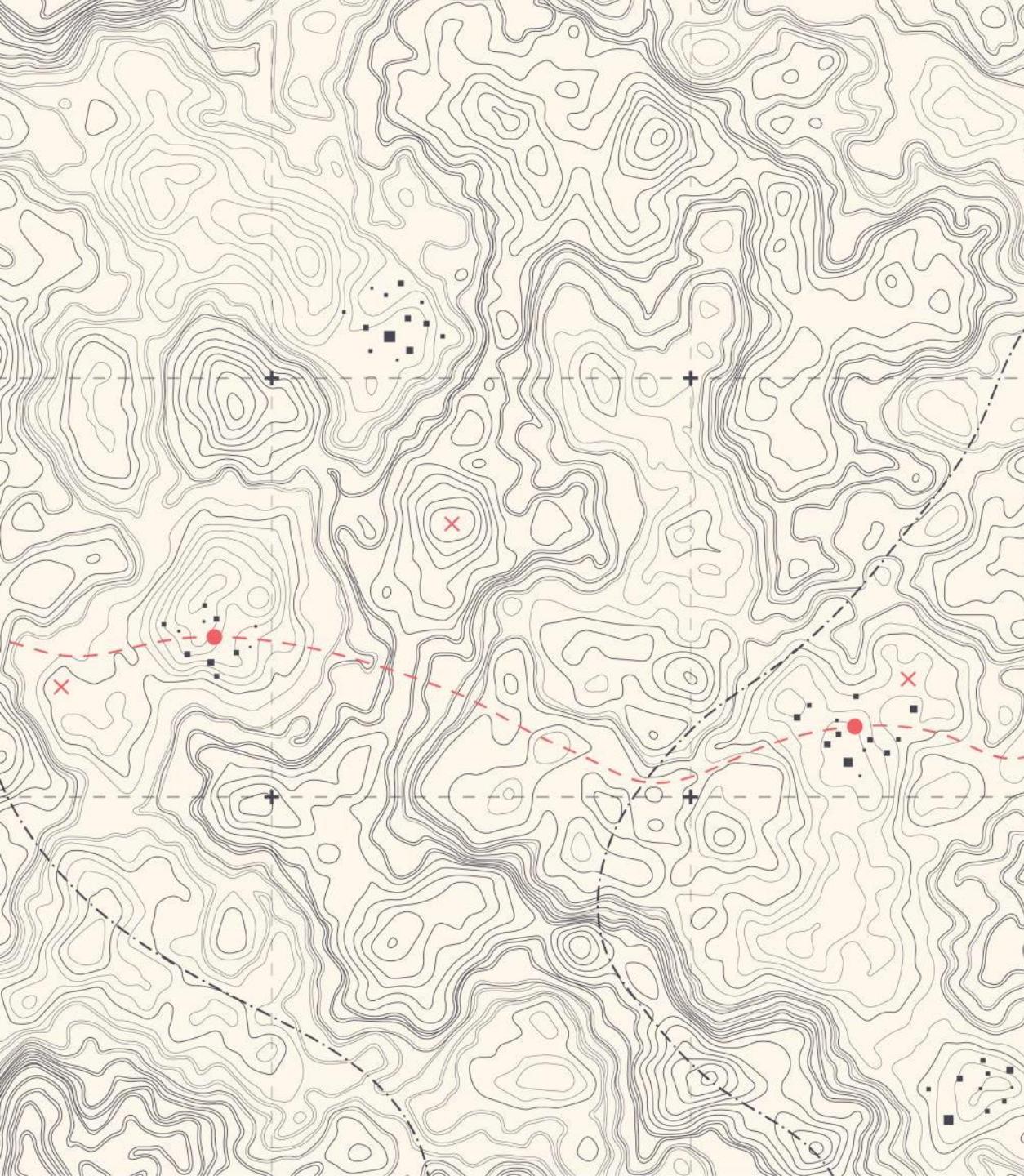
КРИТЕРИЙ ОСТАНОВКИ

- **Критерий остановки** — это правило, по которому мы решаем, когда уже достаточно близко подошли к ответу и можно остановиться.
 - Например:
 - Мы можем остановиться, когда изменение ошибки стало очень маленьким (почти не меняется).
 - Или когда мы сделали максимально допустимое количество шагов.
 - Или когда достигли нужной точности.
-

ГРАДИЕНТНЫЙ СПУСК VS МЕТОД НАИМЕНЬШИХ КВАДРАТОВ

- Если данные небольшие и модель простая — удобнее использовать МНК.
- Если данные большие, сложные, или модель сложнее (например, нейросети) — выбирают градиентный спуск.

<i>Критерий</i>	<i>Метод наименьших квадратов (МНК)</i>	<i>Градиентный спуск (ГС)</i>
Как работает	Решает уравнение напрямую (формула)	Идёт шаг за шагом, постепенно приближаясь к ответу
Сложность вычислений	Обычно решается за один шаг, быстро	Требует много шагов (итераций), может быть медленнее
Точность	Даёт точное решение, если данные «хорошие»	Может приблизиться к точному, зависит от параметров
Применимость	Хорош для небольших и средних задач с 1-2 переменными	Отлично подходит для больших наборов данных и сложных моделей
Масштабируемость	Плохо масштабируется при очень больших данных или множественных переменных	Лучше работает при больших данных и сложных функциях
Настройки	Практически не требует настройки	Нужно выбирать параметры: шаг градиента, критерий остановки и др.
Обучение	Быстрое, без многоократных повторов	Итеративное, требует много повторений



ПРОСТАЯ АНАЛОГИЯ

- **Градиентный спуск:**
Ты спускаешься с горы с завязанными глазами.
Шаг градиента — это длина твоего шага.
Критерий остановки — когда понимаешь,
что уже внизу (или шаги стали слишком
маленькие, чтобы дальше идти).
 - **Метод наименьших квадратов:**
Это как если бы у тебя была карта с
точным указанием, куда идти — и ты сразу
идёшь туда без промедления.
-

Модель:

$$\hat{y}_i = ax_i + b$$

Функция ошибки (MSE):

$$J(a, b) = \frac{1}{2n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 = \frac{1}{2n} \sum (y_i - (ax_i + b))^2$$

где $n = 15$.

ЗАДАЧА 8. РЕШЕНИЕ

Для градиентного спуска нужны производные J по параметрам a, b :

$$\frac{\partial J}{\partial a} = -\frac{1}{n} \sum x_i(y_i - (ax_i + b))$$

$$\frac{\partial J}{\partial b} = -\frac{1}{n} \sum (y_i - (ax_i + b))$$

ЗАДАЧА 8. РЕШЕНИЕ

3. Алгоритм

- Инициализируем $a = 0, b = 0$
- Выбираем скорость обучения α (например, 0.0001)
- На каждой итерации считаем градиенты и обновляем:

$$a := a - \alpha \frac{\partial J}{\partial a}$$

$$b := b - \alpha \frac{\partial J}{\partial b}$$

ЗАДАЧА 8. РЕШЕНИЕ

ЗАДАЧА 8. ПЕРВЫЕ ИТЕРАЦИИ

Итерация 0

При $a = 0, b = 0,$

$$\hat{y}_i = 0, \quad e_i = y_i$$

Подсчитаем

$$\frac{\partial J}{\partial a} = -\frac{1}{15} \sum x_i y_i = -16866.67$$

$$\frac{\partial J}{\partial b} = -\frac{1}{15} \sum y_i = -330$$

Обновляем параметры:

$$a = 0 + 0.0001 \times 16866.67 = 1.6867$$

$$b = 0 + 0.0001 \times 330 = 0.033$$

Итерации продолжаются, пока не будет выполнен критерий остановки.

ЗАДАЧА 8.

РЕШЕНИЕ

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D # для 3D графика

# Задача 1: Градиентный спуск
def gradient_descent(x, y, alpha=0.0001, iters=1000):
    a, b = 0, 0
    n = len(x)
    for _ in range(iters):
        y_pred = a * x + b
        da = (-2/n) * np.sum(x * (y - y_pred))
        db = (-2/n) * np.sum(y - y_pred)
        a -= alpha * da
        b -= alpha * db
    return a, b

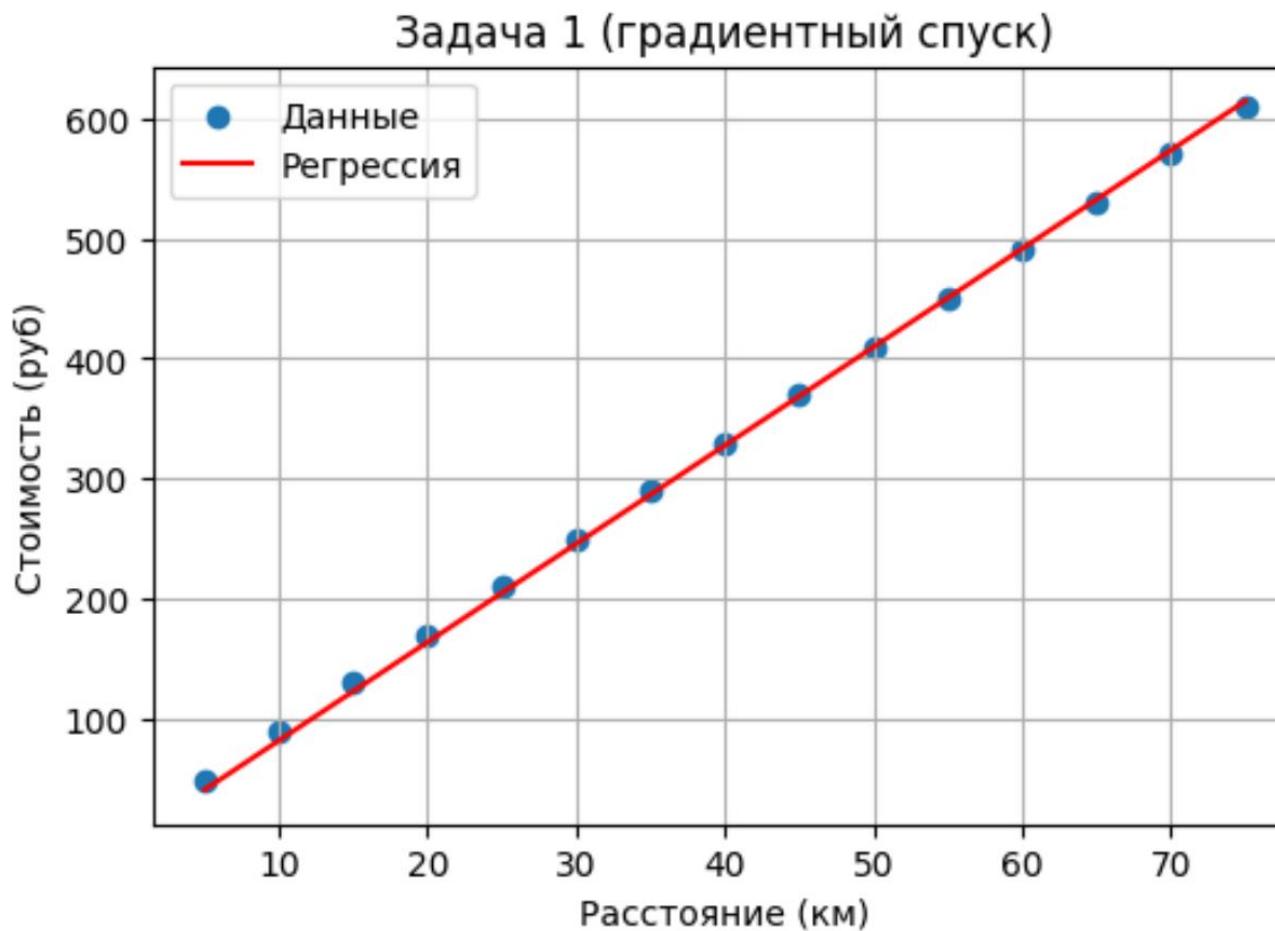
x1 = np.array([5,10,15,20,25,30,35,40,45,50,55,60,65,70,75])
y1 = np.array([50,90,130,170,210,250,290,330,370,410,450,490,530,570,610])
a1, b1 = gradient_descent(x1, y1)

print("Задача 1:")
print("Уравнение: Стоимость = {a1:.2f} * Расстояние + {b1:.2f}")

x1_test = 80
y1_pred = a1 * x1_test + b1
print("Предсказание для расстояния {x1_test} км: {y1_pred:.2f} руб.\n")
```

Уравнение: Стоимость = 8.18 * Расстояние + 0.59

Предсказание для расстояния 80 км: 655.16 руб.



ЗАДАЧА 8. РЕШЕНИЕ