

Анализ данных в век ИИ

Урок 12 · часть I

Соседи и деревья

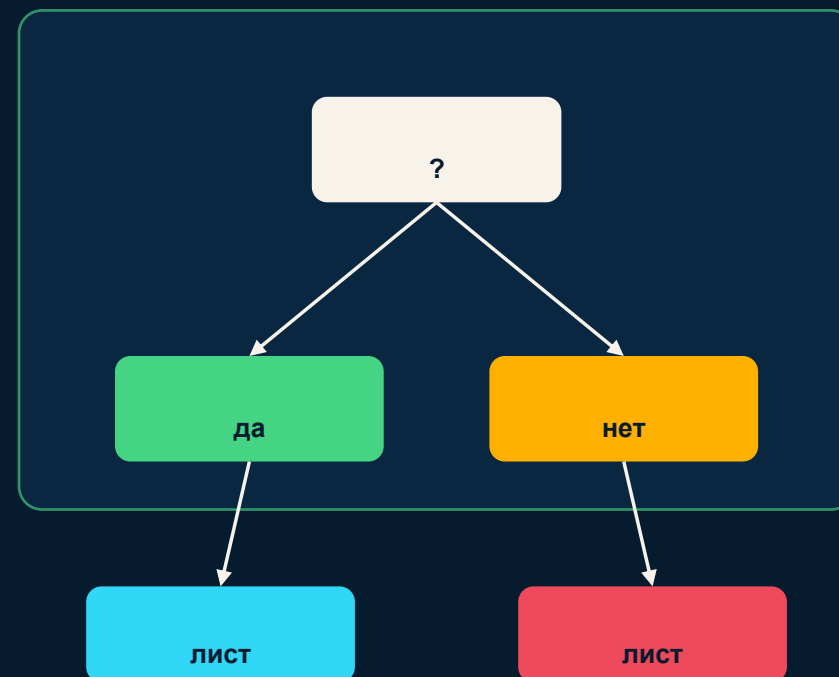
Метод ближайших соседей · Дерево решений



Сегодня у нас три героя

В первой части — соседи и деревья. Временные ряды оставим на вторую часть.

**Соседи смотрят на похожих.
Дерево задаёт вопросы.
Время смотрит назад.**



Как бы вы предсказали цену квартиры?

До алгоритмов — обычная человеческая интуиция.

Новая квартира получает прогноз по ценам похожих квартир.

признак 2



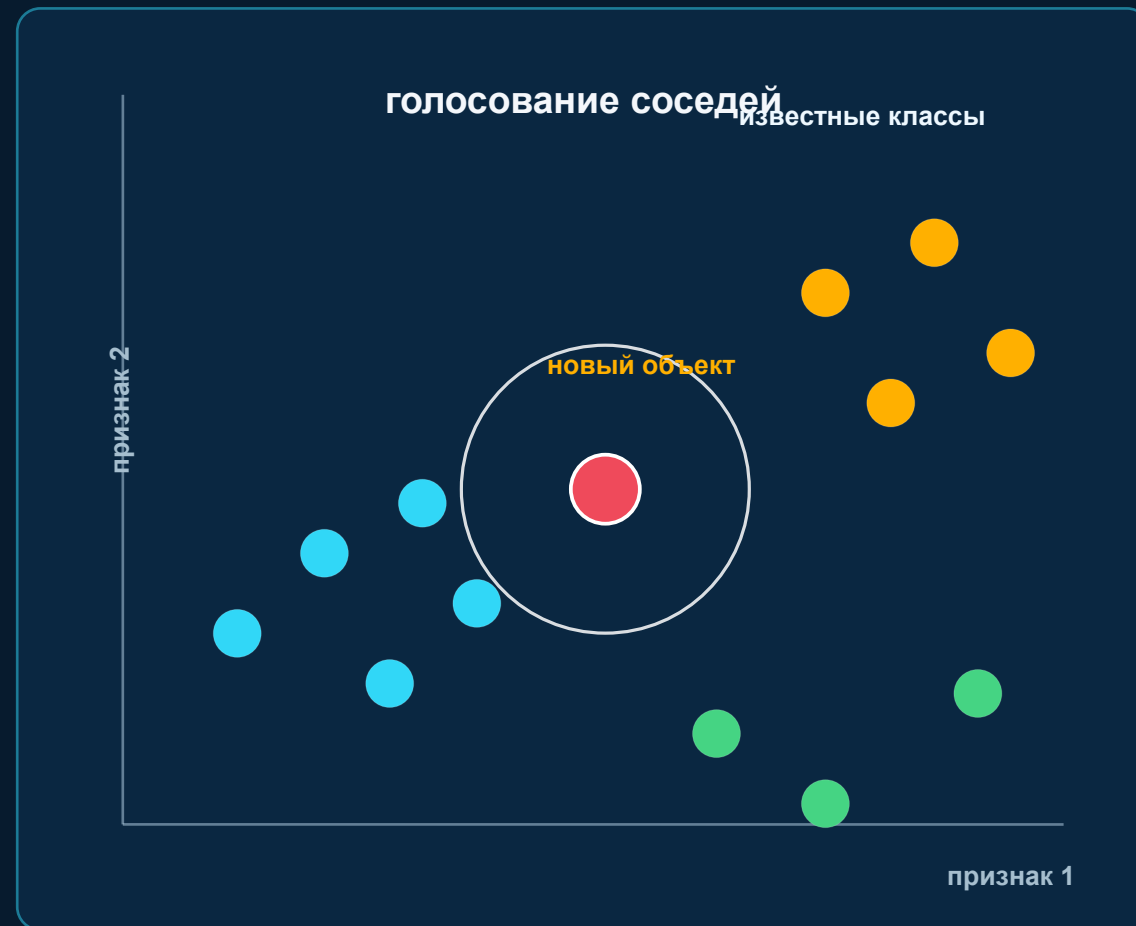
признак 1

Похожие квартиры помогают оценить цену новой.

А если надо определить класс?

Класс — это не число, а категория: да/нет, красный/синий, спам/не спам.

- Новый объект пока без класса.
- Известные точки уже имеют классы.
- Ближайшие соседи голосуют за ответ.



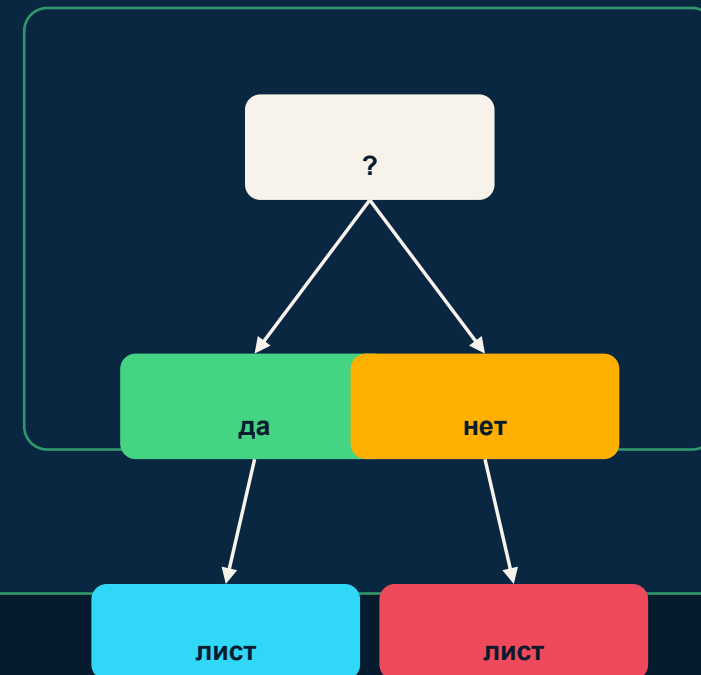
Две очень человеческие стратегии

Сначала спросим похожие объекты, а затем научимся принимать решения через понятные вопросы.

Спросить похожих



Задать вопросы



Метод ближайших соседей: идея за 30 секунд

Новый объект получает ответ от ближайших известных объектов.

Новый объект спрашивает: «Кто здесь на меня похож, и какой ответ был у них?»

Пословица kNN: «Скажи мне, кто твои соседи, и я скажу, кто ты».



Классификация: соседи голосуют

Если большинство ближайших соседей синего класса — новый объект тоже считаем синим.

- Мы выбираем k ближайших точек.
- Каждый сосед отдаёт голос.
- Побеждает большинство.
- На скольких соседей смотреть?



Вопрос

А если взять только одного ближайшего соседа?

Что получится, если $k = 1$?



?

Ответ

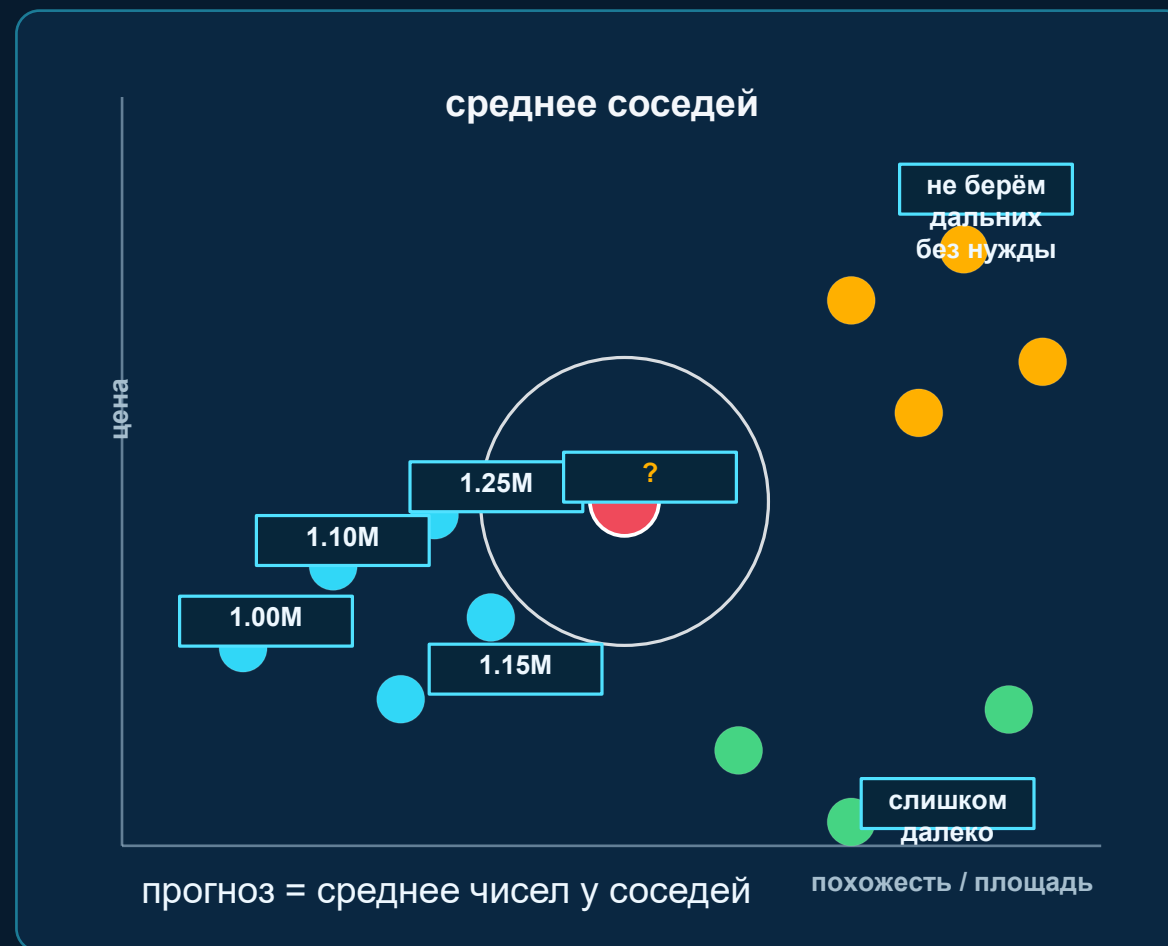
Получится 1-NN: быстро, понятно, но очень чувствительно к случайностям.

**Один сосед может быть гением.
А может быть просто шумной точкой.**

Регрессия: соседи не голосуют, а считают

Для числового прогноза берём среднее значение у ближайших соседей.

- У соседей уже есть числовые ответы.
- Мы усредняем их значения.
- Более близких соседей можно учитывать сильнее.



Одна идея — две задачи

Один и тот же принцип работает и для классов, и для чисел.

Классификация

- соседи голосуют
- побеждает большинство
- можно получить вероятность

Регрессия

- соседи дают числа
- считаем среднее
- близких можно взвесить сильнее

Ключевая мысль

kNN не строит сложную формулу заранее.

Он откладывает решение
до появления нового объекта.

В лекции это удобно помнить так: kNN — ленивый, но наблюдательный алгоритм.

Что значит “рядом”?

Алгоритм должен измерить расстояние между объектами.

- Строка таблицы становится точкой.
- Признак становится координатой.
- Похожесть превращается в расстояние.



Расстояние Евклида — на пальцах

Вправо 3, вверх 4 — по диагонали 5.



Расстояние — это
“насколько далеко”
точки друг от друга в
пространстве признаков.

признак X

Но данные редко живут в двух измерениях

Возраст, доход, площадь, район, комнаты, расстояние до центра...

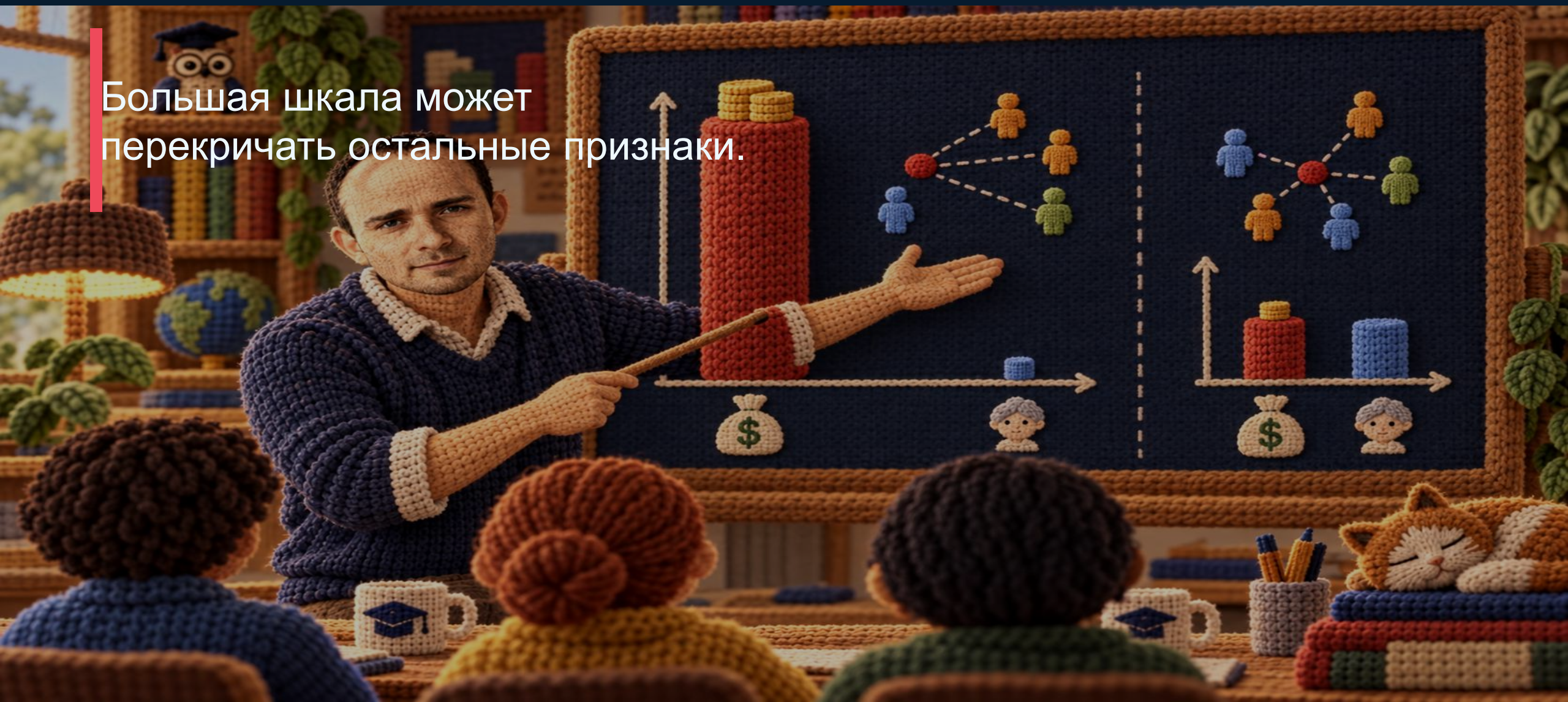
Одна строка таблицы может быть точкой в пространстве из многих признаков.

- возраст
- доход
- площадь
- комнаты
- район
- сезон

Ловушка: одна шкала может съесть остальные

Если один признак измеряется тысячами, он может доминировать в расстоянии.

Большая шкала может
перекричать остальные признаки.



Вопрос

Кто ближе: похожий по возрасту или похожий по доходу?

Без нормализации ответ часто задают единицы измерения, а не смысл данных.



?

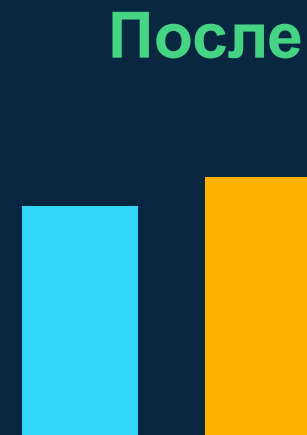
Ответ

Зависит от масштаба признаков.

Если шкалы несопоставимы, “близость” может стать иллюзией.

Нормализация

Приводим признаки к сопоставимым шкалам перед расчётом расстояний.



признаки сравнимы по масштабу

Совет от Игоря

Если модель считает расстояние — сначала проверьте масштаб.

Для kNN нормализация часто важнее, чем красивый выбор цвета точек.

Сколько соседей спрашивать?

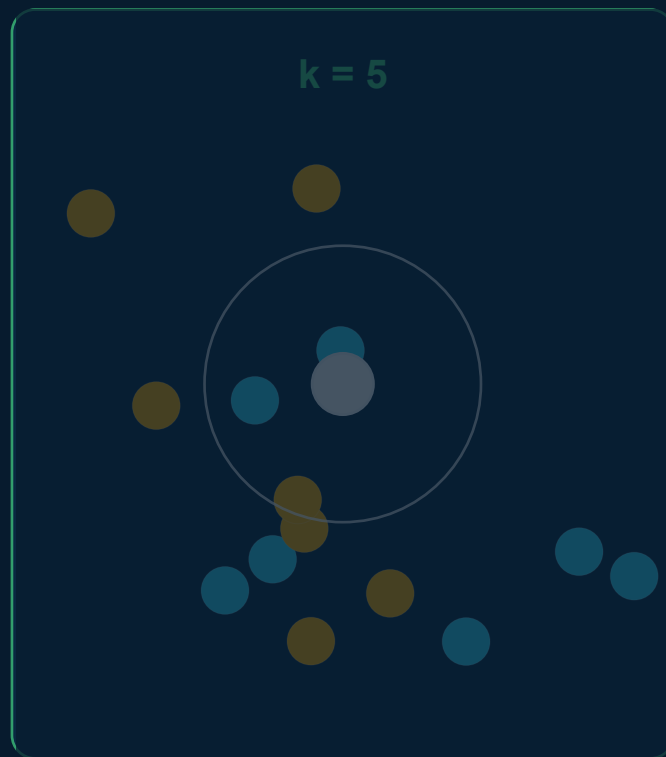
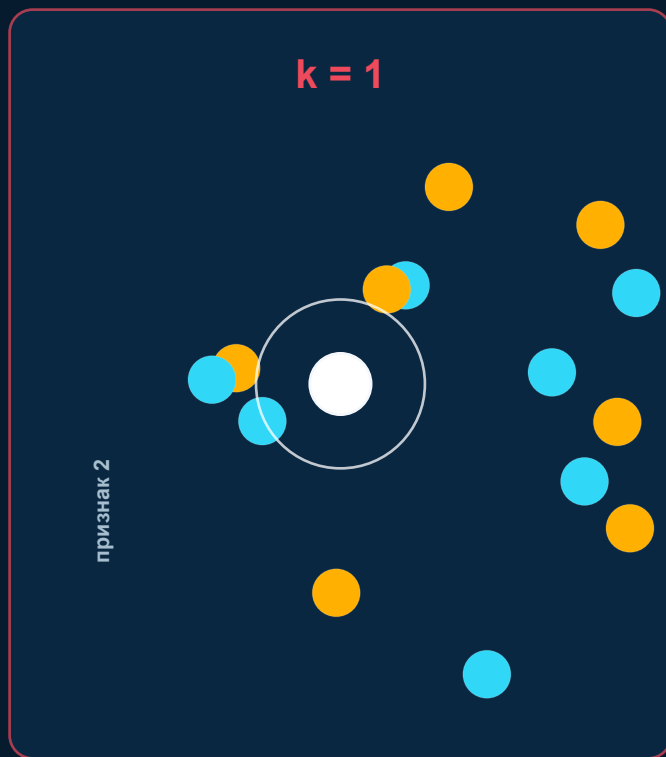
k — главный параметр метода ближайших соседей.



Малое k слышит детали.
Большое k слышит толпу.

$k = 1$

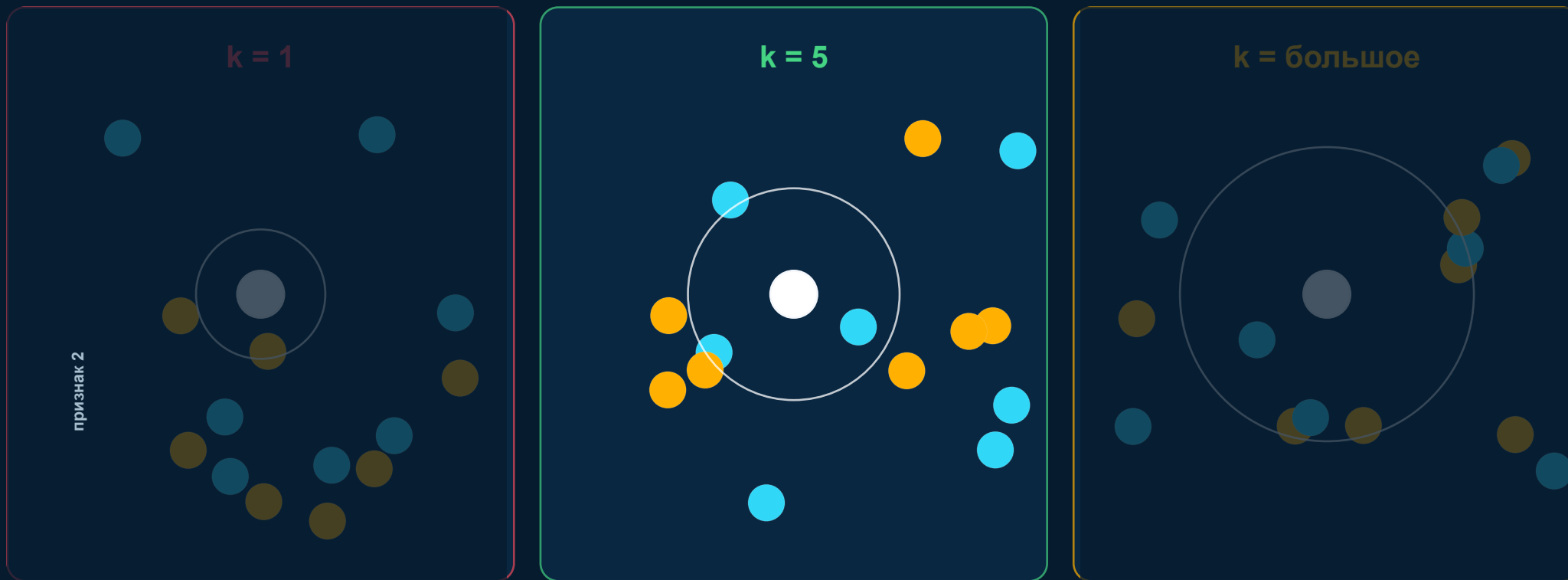
Очень гибко, но очень чувствительно к одной точке.



признак 1
Фокус на первом варианте: решение может резко меняться из-за одной точки.

$k = 5$

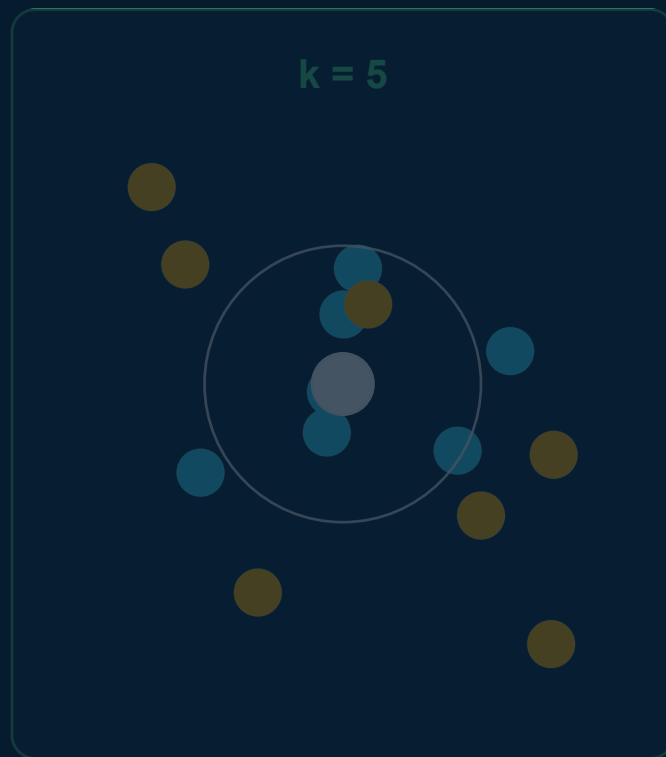
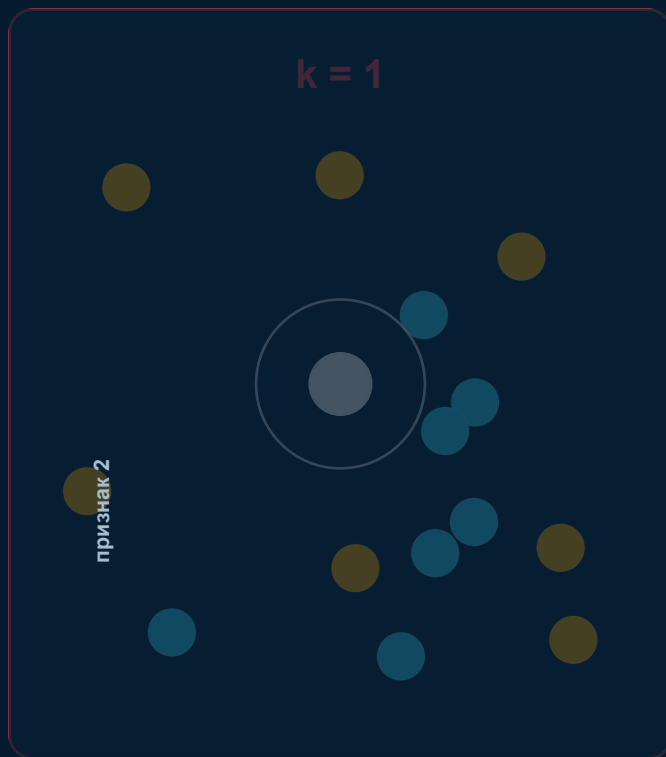
Голосует небольшая компания соседей.



признак 1
Небольшая компания соседей часто стабильнее одного голоса.

к слишком большое

Локальная структура растворяется в общем среднем.



признак 1
Слишком много соседей — и локальная структура исчезает.

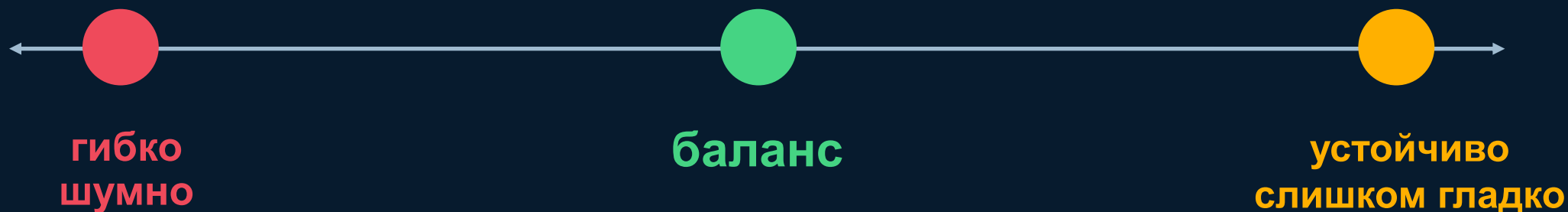
Три поведения одного алгоритма

Количество соседей меняет не только результат, но и характер модели.



Количество соседей меняет характер модели

Малое k даёт гибкость, но ловит шум. Большое k сглаживает, но теряет детали.



Как выбрать k?

Не гадать. Проверять качество на validation или cross-validation.

Выбор k проверяем экспериментом, а не угадываем.

Orange

Test & Score

Сравнить несколько значений k через validation или cross-validation. Смотрим качество, а не интуицию.

сюда вставим актуальный скрин

Для любопытных: простой, но серьёзный

При росте данных kNN может приближаться к оптимальному правилу — если k растёт правильно.

Если данных становится много,
а k растёт медленно:

$$k \rightarrow \infty, \text{ но } k / n \rightarrow 0$$

тогда kNN может приближаться к лучшему возможному правилу
при стандартных условиях.

Важно: это не означает, что фиксированный 1-NN всегда оптимален.

Плюсы kNN

Почему такой простой метод до сих пор живёт.

очень понятная идея

классификация и регрессия

нелинейные границы

почти нет обучения заранее

Минусы kNN

Где простота начинает дорогого стоить.

чувствителен к масштабу

страдает от лишних признаков

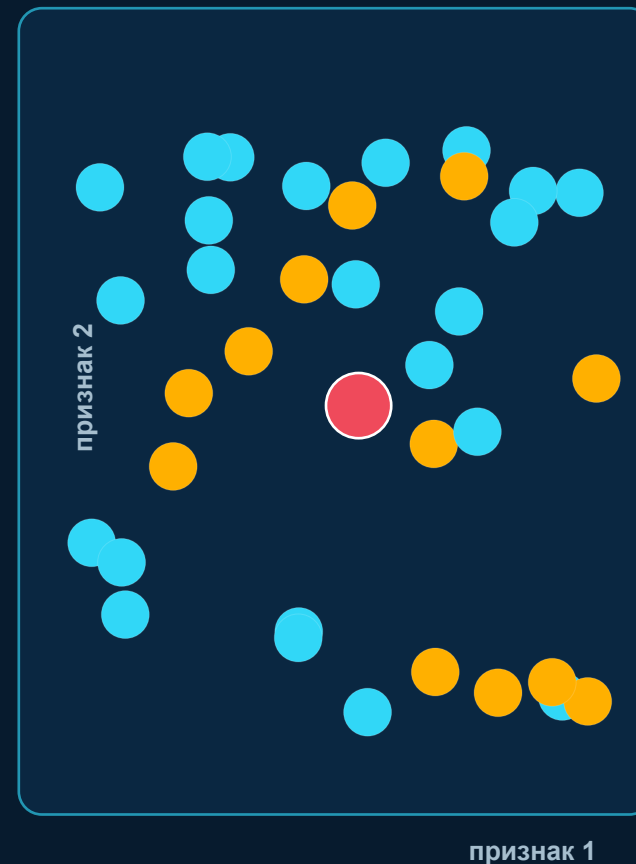
медленный на больших данных

**плохо чувствует “близость” в высокой
размерности**

Проклятие размерности — очень коротко

В большой размерности “близкие” точки становятся не такими уж близкими.

Чем больше измерений, тем труднее быть по-настоящему близким соседом.

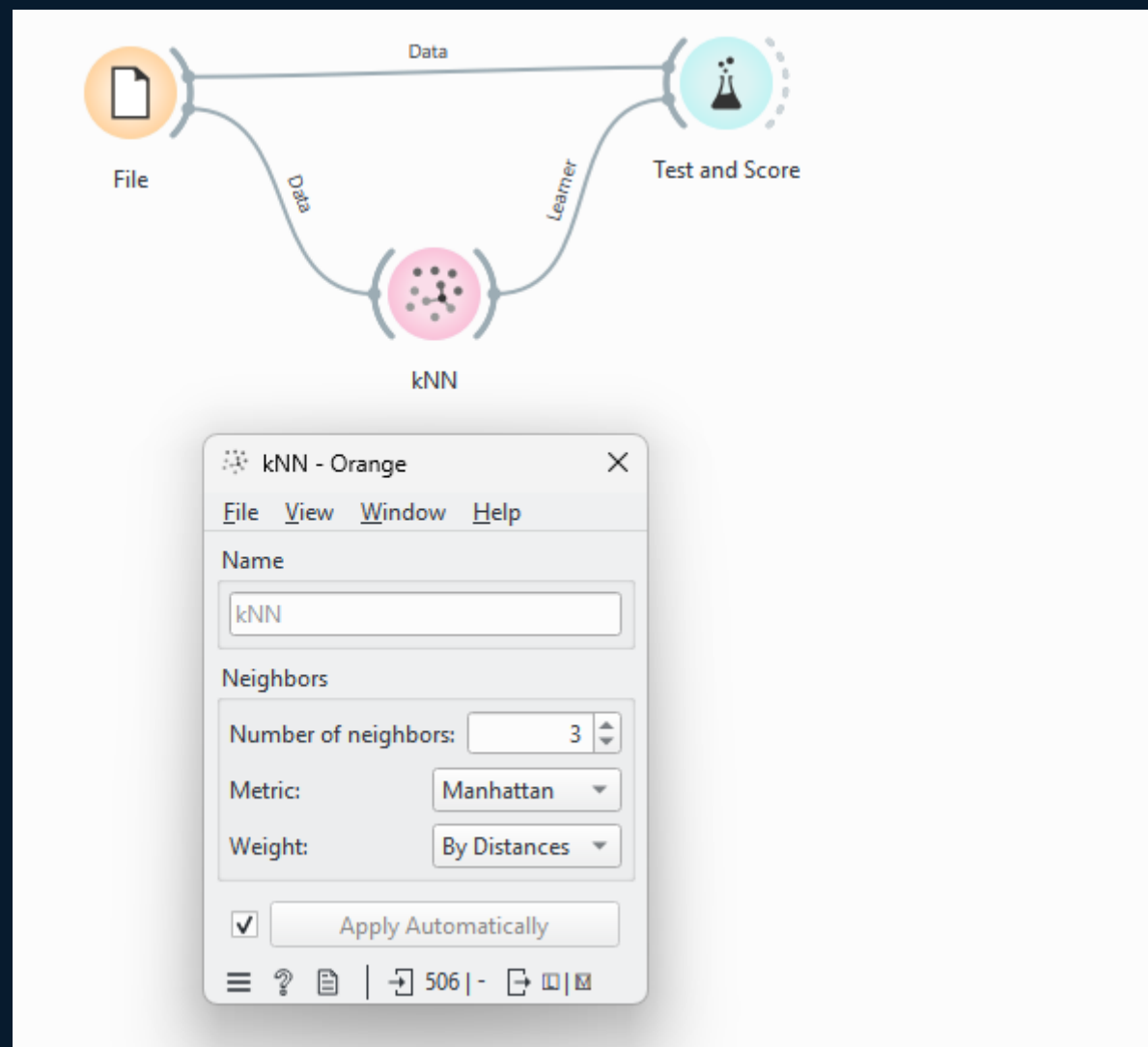


kNN в Orange

Минимальный workflow: данные → нормализация → Neighbors → Test & Score.

В Orange важно собрать честную цепочку: данные, нормализация, модель, проверка.

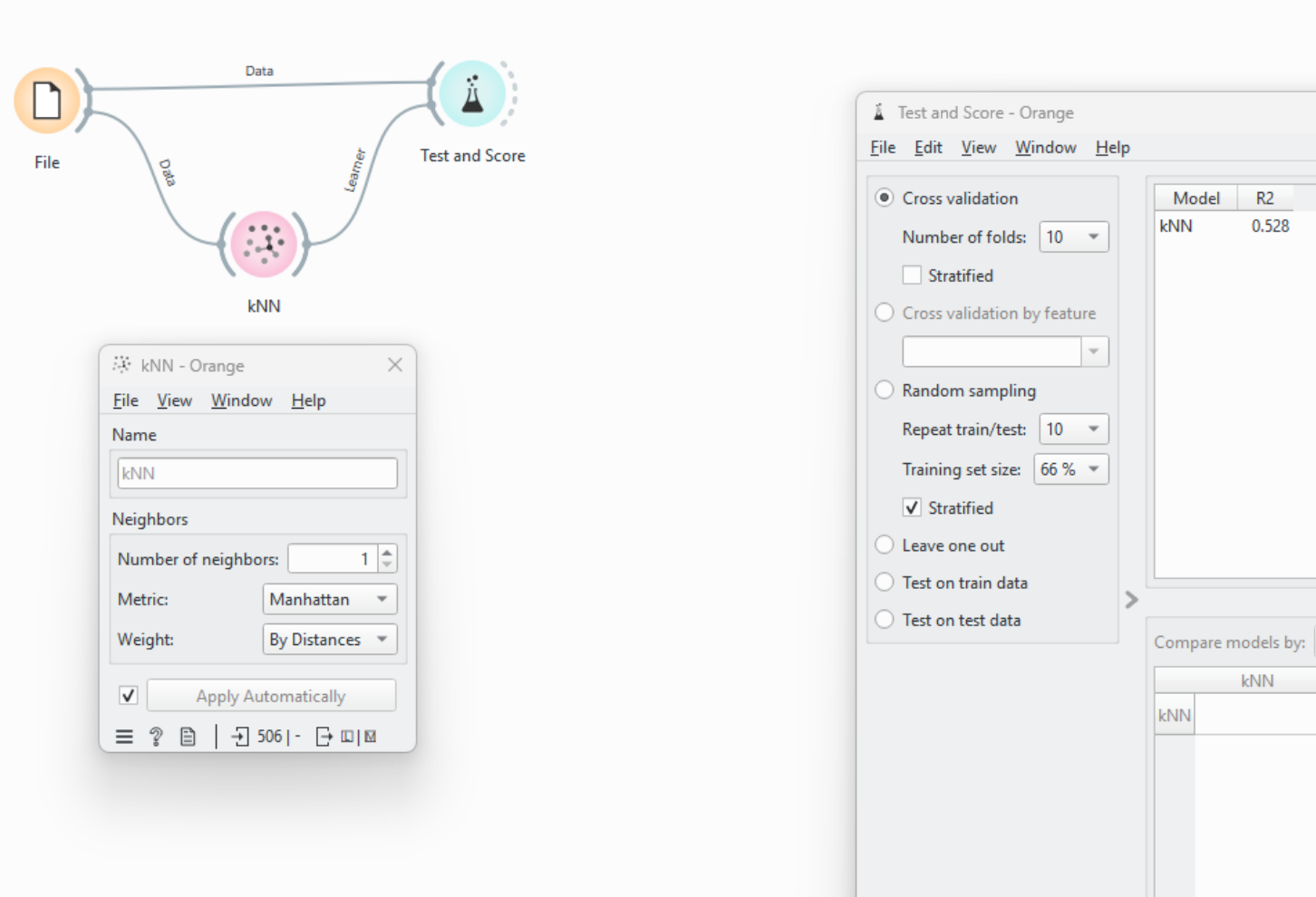
Нормализация в KNN определена по умолчанию



Где меняется k

В настройках Neighbors ищем число соседей.

- Меняем количество соседей.
- Сравниваем качество.
- Не доверяем одной случайной попытке.



The image shows an Orange3 workflow and the settings for the kNN widget.

Workflow: A 'File' widget connects to a 'Data' widget. The 'Data' widget connects to a 'kNN' widget. The 'kNN' widget connects to a 'Test and Score' widget.

kNN - Orange settings:

- Name: kNN
- Neighbors:
 - Number of neighbors: 1
 - Metric: Manhattan
 - Weight: By Distances
- ☒ Apply Automatically

Test and Score - Orange settings:

- ☒ Cross validation
 - Number of folds: 10
 - ☐ Stratified
- ☐ Cross validation by feature
- ☐ Random sampling
 - Repeat train/test: 10
 - Training set size: 66 %
 - ☒ Stratified
- ☐ Leave one out
- ☐ Test on train data
- ☐ Test on test data

Compare models by:

Model	R2
kNN	0.528

Где меняется k

В настройках Neighbors ищем число соседей.

- Меняем количество соседей.
- Сравниваем качество.
- Не доверяем одной случайной попытке.

The workflow diagram shows a 'File' widget (orange circle with a document icon) connected to a 'kNN' widget (pink circle with a network icon) via a 'Data' link. The 'kNN' widget is then connected to a 'Test and Score' widget (blue circle with a flask icon) via a 'Learner' link. The 'kNN' widget is highlighted with a pink circle.

The 'kNN - Orange' settings window is open, showing the following configuration:

- Name: kNN
- Neighbors:
 - Number of neighbors: 5
 - Metric: Manhattan
 - Weight: By Distances
- ☒ Apply Automatically

The 'Test and Score - Orange' window shows the results of the model evaluation:

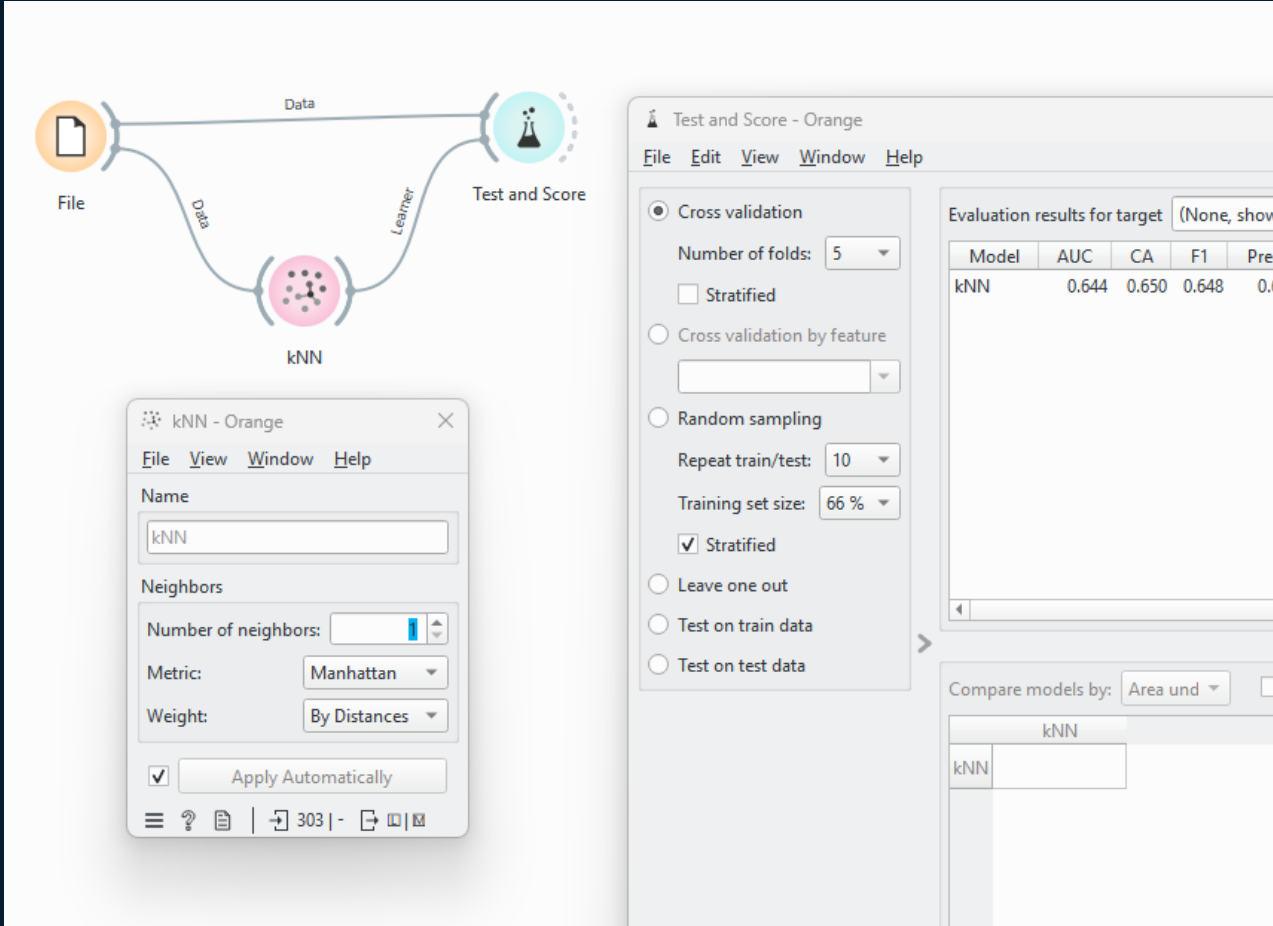
Model	R2
kNN	0.656

The 'Compare models by' dropdown is set to 'Mean sqi' and the 'Negligible dif' checkbox is unchecked.

Эксперимент 1: классификация

Один датасет, несколько значений k , сравниваем качество.

Одна и та же идея при разных k превращается в разные модели.



The image shows an Orange3 workflow and the configuration for a k-Nearest Neighbors (kNN) model.

Workflow: A 'File' widget connects to a 'Data' widget. The 'Data' widget connects to a 'kNN' widget (labeled 'Learner'). The 'kNN' widget connects to a 'Test and Score' widget.

kNN - Orange settings:

- Name: kNN
- Neighbors:
 - Number of neighbors: 1
 - Metric: Manhattan
 - Weight: By Distances
- ☒ Apply Automatically

Test and Score - Orange settings:

- ☒ Cross validation
 - Number of folds: 5
 - ☐ Stratified
- ☐ Cross validation by feature
- ☐ Random sampling
 - Repeat train/test: 10
 - Training set size: 66 %
 - ☒ Stratified
- ☐ Leave one out
- ☐ Test on train data
- ☐ Test on test data

Evaluation results for target (None, show):

Model	AUC	CA	F1	Prec
kNN	0.644	0.650	0.648	0.6

Compare models by: Area und

Эксперимент 1: классификация

Один датасет, несколько значений k , сравниваем качество.

Одна и та же идея при разных k превращается в разные модели.

The screenshot displays the Orange3 data mining software interface. At the top, a workflow is shown with three widgets: 'File', 'kNN', and 'Test and Score'. The 'File' widget is connected to the 'kNN' widget via a 'Data' link, and the 'kNN' widget is connected to the 'Test and Score' widget via a 'Learner' link. Below the workflow, the 'kNN - Orange' widget settings are visible. The settings include a 'Name' field with the value 'kNN', a 'Neighbors' section with 'Number of neighbors' set to 7, 'Metric' set to 'Manhattan', and 'Weight' set to 'By Distances'. There is also an 'Apply Automatically' checkbox which is checked. To the right, the 'Test and Score - Orange' widget settings are shown. The 'Cross validation' option is selected, with 'Number of folds' set to 5. The 'Stratified' checkbox is also checked. The 'Evaluation results for target' table is visible, showing the following data:

Model	AUC	CA	F1
kNN	0.748	0.673	0.673

Below the table, the 'Compare models by' section is visible, with 'Area under the curve' selected.

Эксперимент 2: до и после нормализации

Та же модель может вести себя иначе после Preprocess.

Мы меняем не модель, а линейку, которой она измеряет расстояния.

В Orange KNN делает нормализацию автоматически

Orange

Вопрос-проверка kNN

Один признак измеряется тысячами, остальные — единицами.

Что, скорее всего, случится без нормализации?

А. Все признаки будут равны по влиянию

В. Большой по масштабу признак захватит расстояние

С. kNN автоматически всё исправит

Вопрос-проверка kNN

Один признак измеряется тысячами, остальные — единицами.

Что, скорее всего, случится без нормализации?

A. Все признаки будут равны по влиянию

B. Большой по масштабу признак захватит расстояние

C. kNN автоматически всё исправит

Ответ + мини-summary

kNN держится на четырёх словах.



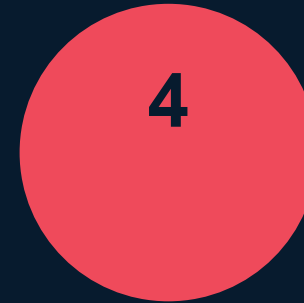
соседи



расстояние



количество k



нормализация

kNN: “Покажите мне похожих”

Он почти не объясняет решение — он показывает ближайшие примеры.

kNN объясняет решение через ближайшие примеры, а не через правило.

признак 2



серьёзную, носящуюся с ней, борьбу, чтобы закрепить её.



Задача

Классификация

Регрессия

Набор данных

Кластеры



Точек

15

Классов

2

Шум

0.25

Новая выборка

Очистить поле

Модель

Число соседей K

1

Веса соседей

Равные

По расстоянию

Метрика

Евклидова

Манхэттенская

Чебышёва

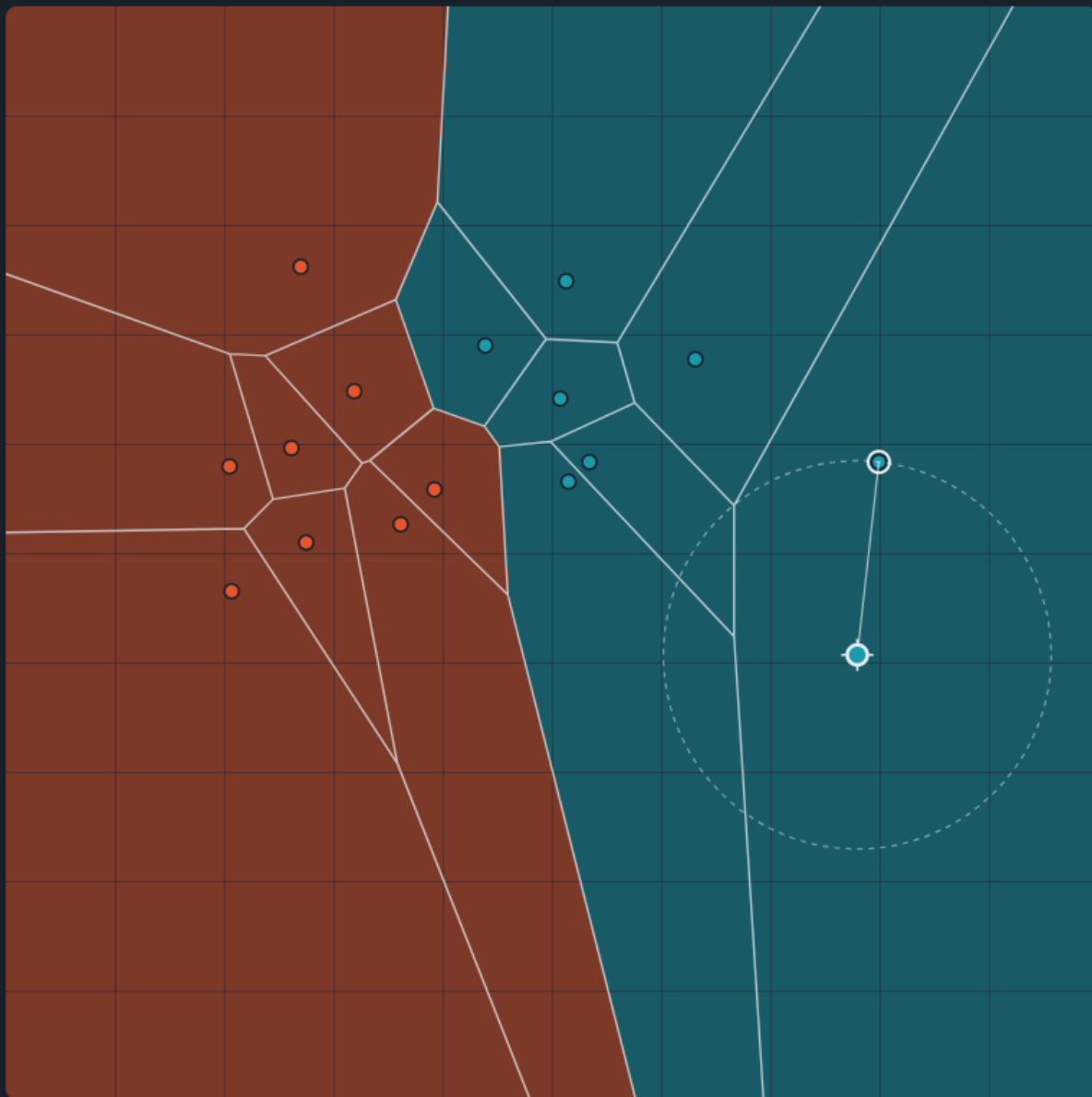
Показать

☒ Карту решений

☒ Линейки Вороного

KNN и диаграмма Вороного

Как число соседей K, веса и метрика меняют границы решений. Наведите курсор на поле, чтобы увидеть ближайших соседей точки запроса; нажмите, чтобы закрепить её.



Задача

Классификация

Регрессия

Набор данных

Кластеры



Точек

15

Классов

2

Шум

0.25

Новая выборка

Очистить поле

Модель

Число соседей K

1

Веса соседей

Равные

По расстоянию

Метрика

Евклидова

Манхэттенская

Чебышёва

Показать

☒ Карту решений

☒ Ячейки Вороного

KNN и диаграмма Вороного

Как число соседей K, веса и метрика меняют границы решений. Наведите курсор на поле, чтобы увидеть ближайших соседей точки запроса; нажмите, чтобы закрепить её.



● Класс 1 ● Класс 2

x = 0,06, y = 0,20

Задача

Классификация

Регрессия

Набор данных

Кластеры



Точек

15

Классов

2

Шум

0,65

Новая выборка

Очистить поле

Модель

Число соседей K

7

Веса соседей

Равные

По расстоянию

Метрика

Евклидова

Манхэттенская

Чебышёва

Показать



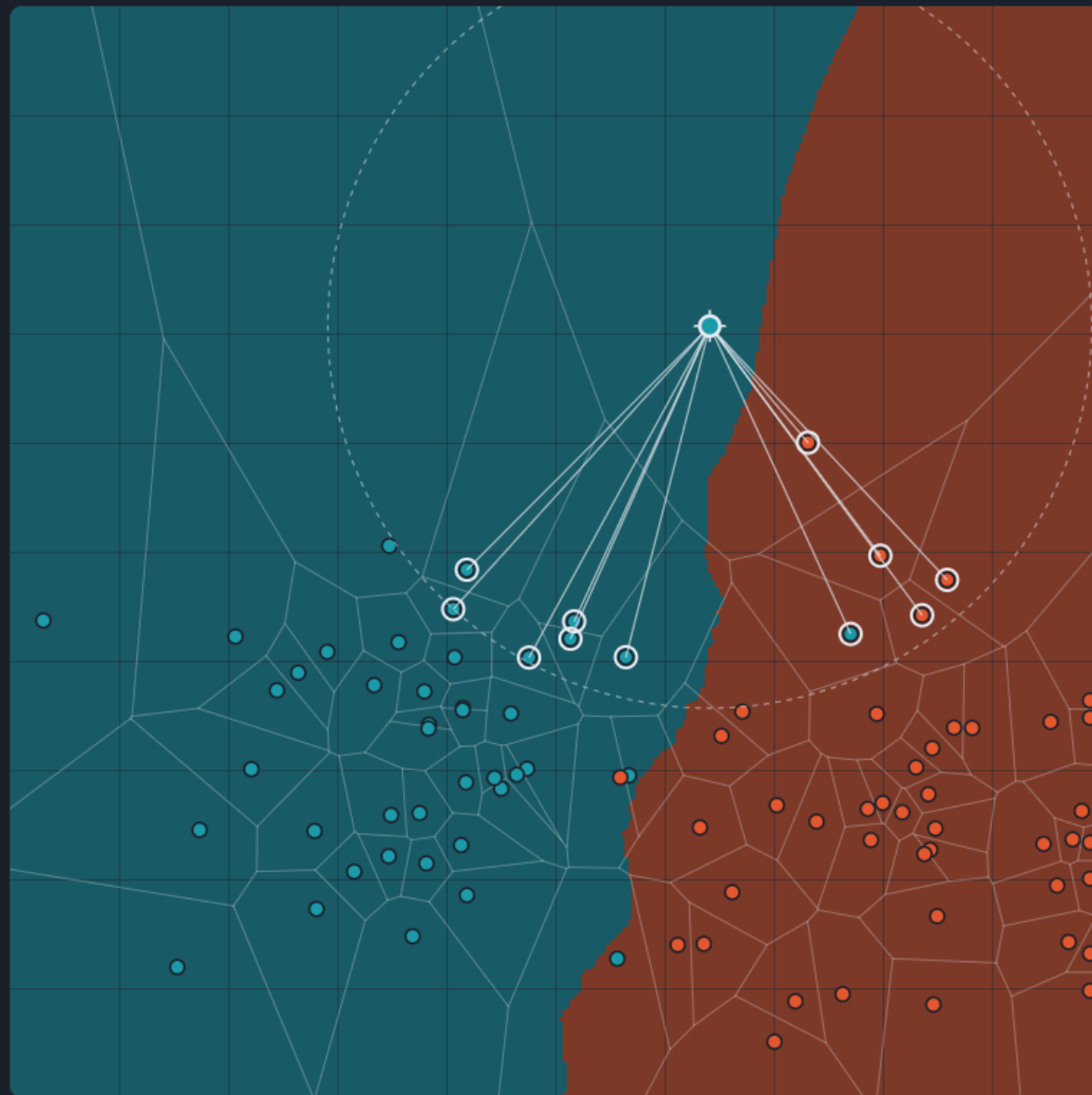
Карту решений



Ячейки Вороного

KNN и диаграмма Вороного

Как число соседей K, веса и метрика меняют границы решений. Наведите курсор на поле, чтобы увидеть ближайших соседей точки запроса; нажмите, чтобы закрепить её.



● Класс 1 ● Класс 2

$x = 0,64, y = 0,29$

Задача

Классификация

Регрессия

Набор данных

Кластеры



Точек

85

Классов

2

Шум

0,65

Новая выборка

Очистить поле

Модель

Число соседей K

11

Веса соседей

Равные

По расстоянию

Метрика

Евклидова

Манхэттенская

Чебышёва

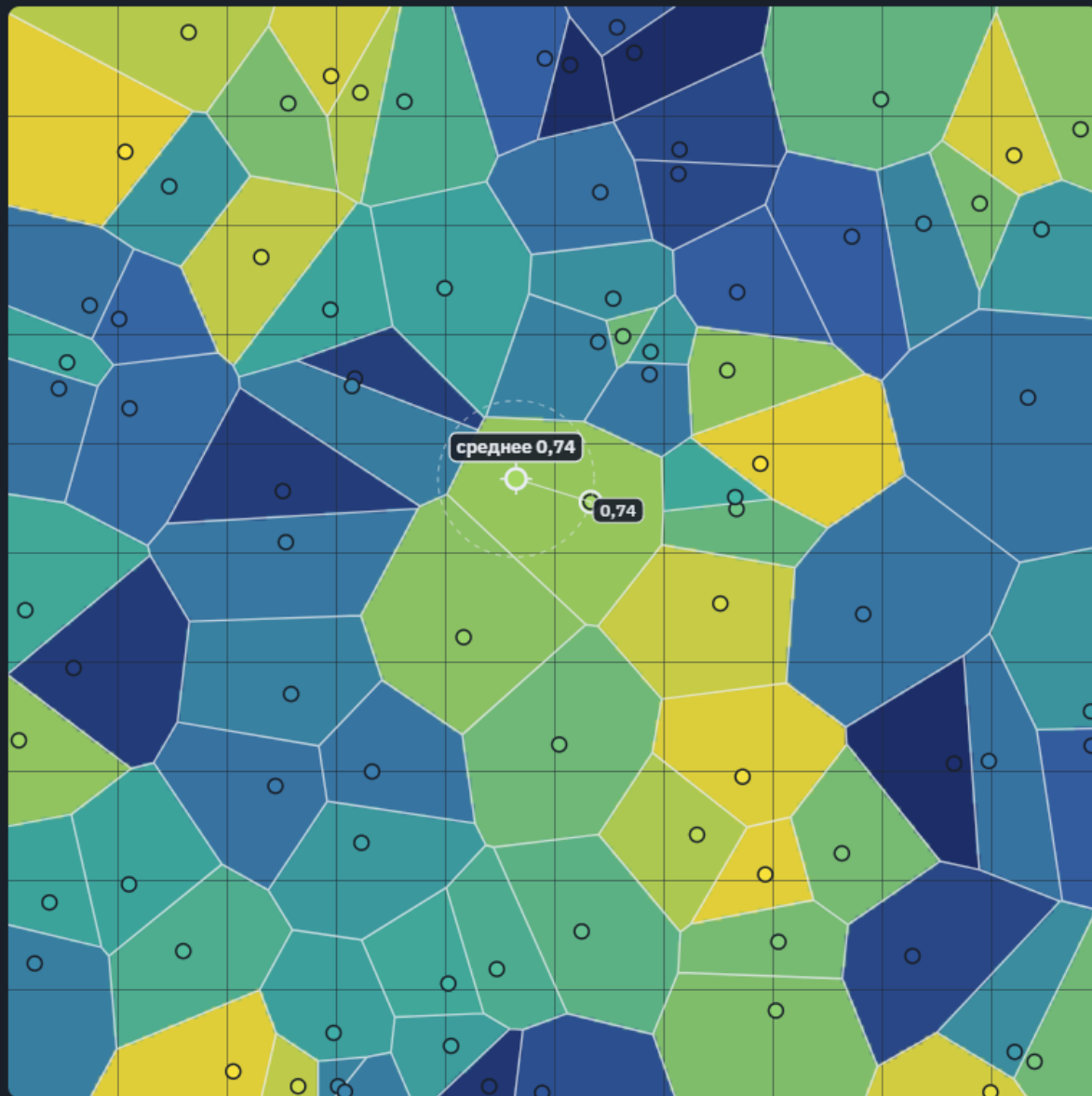
Показать

☒ Карту решений

☒ Ячейки Вороного

KNN и диаграмма Вороного

Как число соседей K, веса и метрика меняют границы решений. Наведите курсор на поле, чтобы увидеть ближайших соседей точки запроса; нажмите, чтобы закрепить её.



0 — значение цели

x = 0,47, y = 0,43

Задача

Классификация

Регрессия

Набор данных

Волны

Точек

85

Шум

0,65

Новая выборка

Очистить поле

Модель

Число соседей K

1

Веса соседей

Равные

По расстоянию

Метрика

Евклидова

Манхэттенская

Чебышёва

Показать

☒ Карту решений

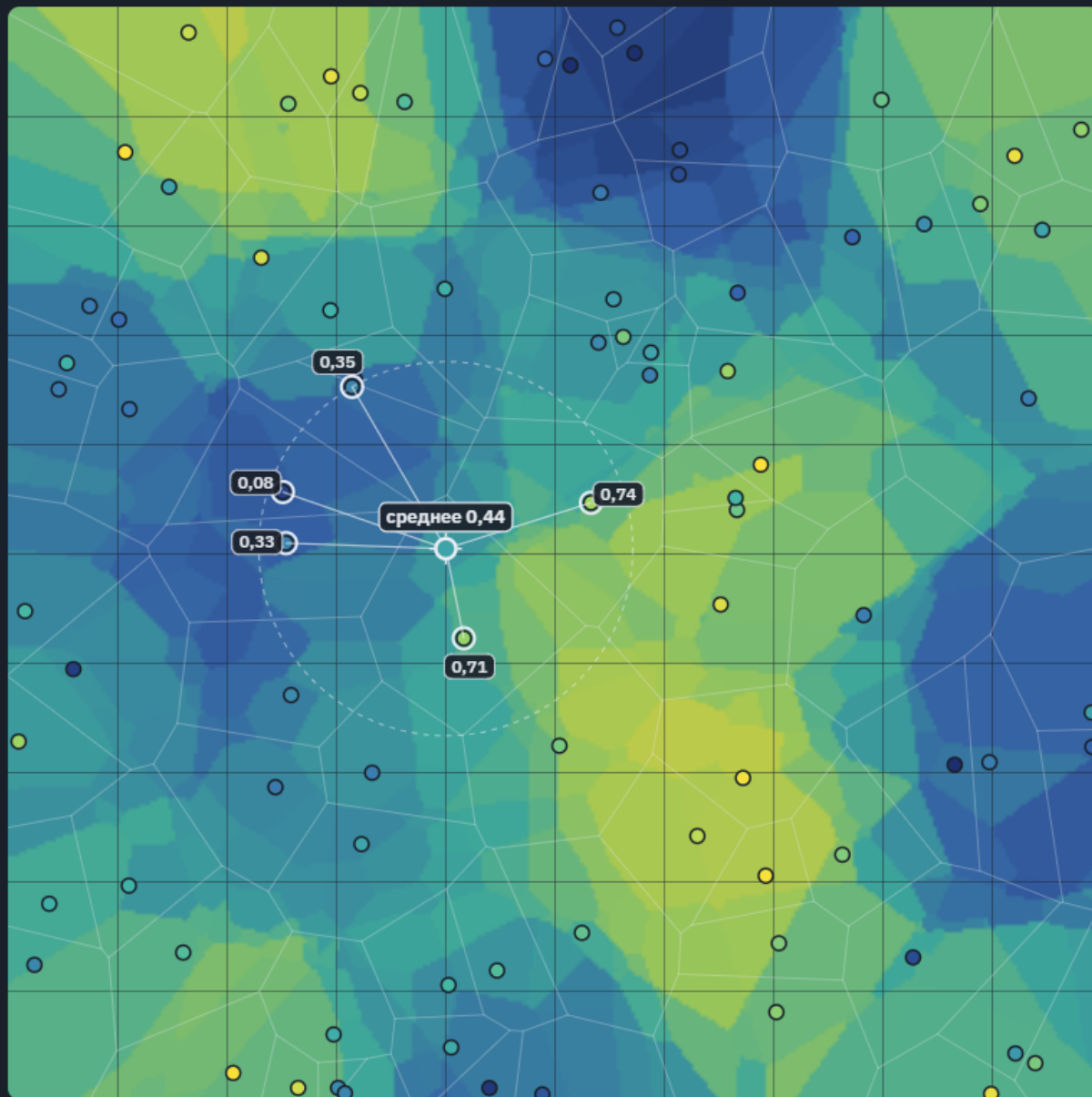
☒ Ячейки Вороного

☒ Точки выборки

Что на карте

KNN и диаграмма Вороного

Как число соседей K, веса и метрика меняют границы решений. Наведите курсор на поле, чтобы увидеть ближайших соседей точки запроса; нажмите, чтобы закрепить её.



Задача

Классификация

Регрессия

Набор данных

Волны

Точек

85

Шум

0,65

Новая выборка

Очистить поле

Модель

Число соседей K

5

Веса соседей

Равные

По расстоянию

Метрика

Евклидова

Манхэттенская

Чебышёва

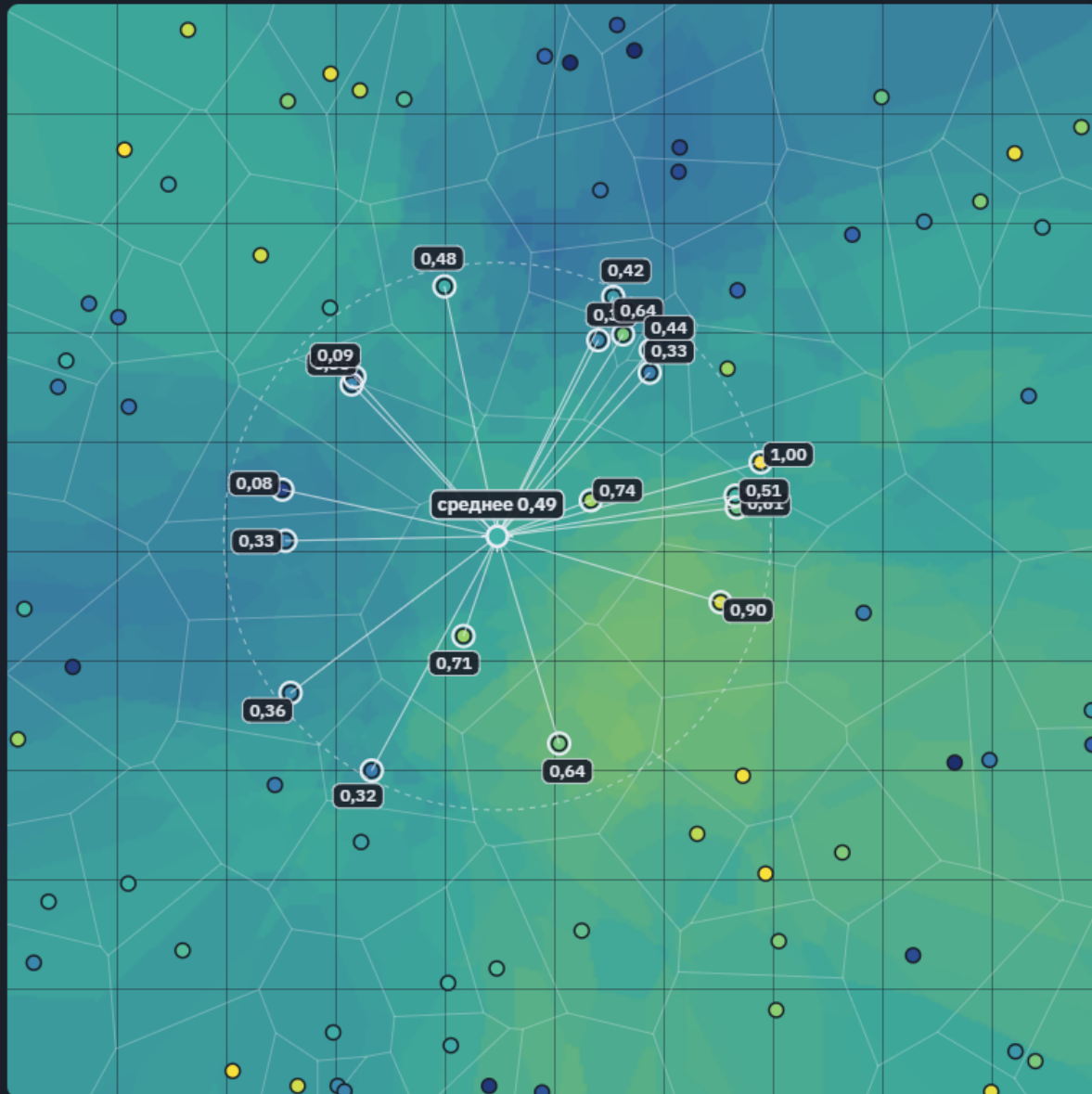
Показать

- ☒ Карту решений
- ☒ Ячейки Вороного
- ☒ Точки выборки

Что на карте

KNN и диаграмма Вороного

Как число соседей K, веса и метрика меняют границы решений. Наведите курсор на поле, чтобы увидеть ближайших соседей точки запроса; нажмите, чтобы закрепить её.



0 — значение цели 1 — значение цели

x = 0,45, y = 0,49

Задача

Классификация

Регрессия

Набор данных

Волны

Точек

85

Шум

0,65

Новая выборка

Очистить поле

Модель

Число соседей K

19

Веса соседей

Равные

По расстоянию

Метрика

Евклидова

Манхэттенская

Чебышёва

Показать

☒ Карту решений

☒ Ячейки Вороного

☒ Точки выборки

Что на карте

КАК kNN ИЗМЕРЯЕТ БЛИЗОСТЬ?

Метрика расстояния решает, кто станет "соседом"

Данные
сближают
идеи ♥

БОЛЬШЕ
ЧЕМ ДАННЫЕ
БОЛЬШЕ
ВОЗМОЖНОСТЕЙ

Разные метрики
— разные соседи
— разные прогнозы

ОДНИ И ТЕ ЖЕ ДАННЫЕ.
РАЗНОЕ ПОНЯТИЕ БЛИЗОСТИ.
РАЗНЫЕ СОСЕДИ.
РАЗНЫЕ РЕШЕНИЯ.

НОВЫЙ
ОБЪЕКТ

ЕВКЛИДОВА

Прямая линия между точками.

$$d = \sqrt{\sum (x_i - y_i)^2}$$



ПЛЮСЫ

- ✓ интуитивная
- ✓ быстрая

МИНУСЫ

- ✗ чувствительна к масштабу
- ✗ выбросы влияют сильнее

МАНХЭТТЕН

Путь по кварталам.

$$d = \sum |x_i - y_i|$$



ПЛЮСЫ

- ✓ проще переживает выбросы
- ✓ часто хороша в многомерных данных

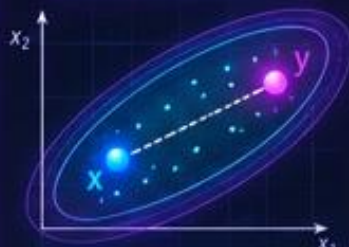
МИНУСЫ

- ✗ зависит от масштаба
- ✗ движение только по осям

МАХАЛАНОВИСА

Расстояние с учётом масштаба и корреляций признаков.

$$d^2 = \Delta^T S^{-1} \Delta$$



ПЛЮСЫ

- ✓ учитывает корреляции
- ✓ видит "форму" данных

МИНУСЫ

- ✗ нужно оценивать ковариацию
- ✗ может быть нестабильна при малой выборке/сильной мультиколлинеарности

ХЭММИНГА

Считаем, сколько позиций не совпало.

101101 vs 100111
2 различия

x	1	0	1	1	0	1
y	1	0	0	1	1	1

несовпадения

ПЛЮСЫ

- ✓ идеальна для бинарных/категориальных признаков
- ✓ очень понятная

МИНУСЫ

- ✗ не видит величину различия
- ✗ важно корректное кодирование категорий

КОСИНУСНАЯ

Сравнивает направление, а не длину.

$$d = 1 - \cos \theta$$



ПЛЮСЫ

- ✓ полезна для текстов и разреженных данных
- ✓ масштаб вектора почти не важен

МИНУСЫ

- ✗ игнорирует абсолютную величину
- ✗ плохо работает с нулевыми векторами



МАЛЕНЬКАЯ
МЕТРИКА —
БОЛЬШАЯ РАЗНИЦА
В РЕШЕНИЯХ



ГЛАВНОЕ ПРАВИЛО >>

Сменили метрику → сменились соседи →
может измениться прогноз kNN.



ПРАКТИЧЕСКАЯ ПОДСКАЗКА

Масштаб признаков важен.
В Orange kNN по умолчанию стандартизует
числовые признаки; в виджете Distances доступны
Euclidean, Manhattan, Mahalanobis, Hamming и Cosine.

Правильная
метрика —
ближе к истине ♥

ЗНАНИЯ
СЕГОДНЯ —
РЕШЕНИЯ ЗАВТРА

Метрики расстояния

Как разные способы измерить «близость» двух объектов меняют форму окрестности и ответ модели. Перетаскивайте точки A и B на поле, а в примере с Хэммингом — переключайте биты.

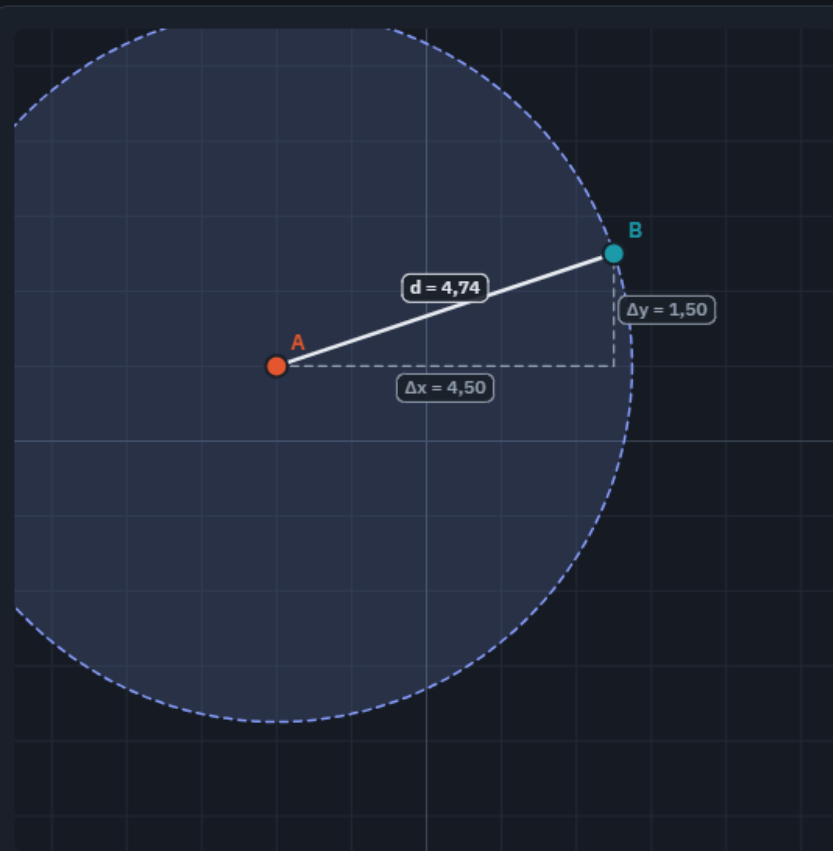
Евклидова

Манхэттенская

Косинусная

Хэмминга

Махаланобиса



● A = (-2,0; 1,0) ● B = (2,5; 2,5)

Для этих же точек: евклидово **4,74**, манхэттенское **6,00**, косинусное **1,32**, Махаланобиса **2,30** (при текущей Σ).

Евклидово расстояние (L_2)

Длина отрезка между точками — «по прямой», как измерили бы линейкой. Это теорема Пифагора, обобщённая на любое число признаков.

Формула

$$d(a,b) = \sqrt{\sum_i (a_i - b_i)^2}$$

Расчёт для текущего примера

$\Delta x = 4,50$, $\Delta y = 1,50$

$d = \sqrt{(4,50^2 + 1,50^2)} = \sqrt{(20,25 + 2,25)} = 4,743$

Плюсы

- Интуитивна и совпадает с обычным геометрическим расстоянием
- Не меняется при повороте системы координат
- Быстро считается, гладкая (удобна для оптимизации)
- Единственная метрика, при которой среднее точек — ближайшая к ним точка (k-means)

Минусы

- Сильно зависит от масштаба признаков: признак в километрах «перевесит» признак в метрах — нужна нормализация
- Квадраты усиливают влияние выбросов
- В высоких размерностях расстояния «слипаются»: все точки почти одинаково далеки
- Игнорирует корреляции между признаками

Когда использовать

Числовые признаки в сопоставимых единицах, невысокая размерность: KNN, k-means, физические координаты.

Метрики расстояния

Как разные способы измерить «близость» двух объектов меняют форму окрестности и ответ модели. Перетаскивайте точки A и B на поле, а в примере с Хэммингом — переключайте биты.

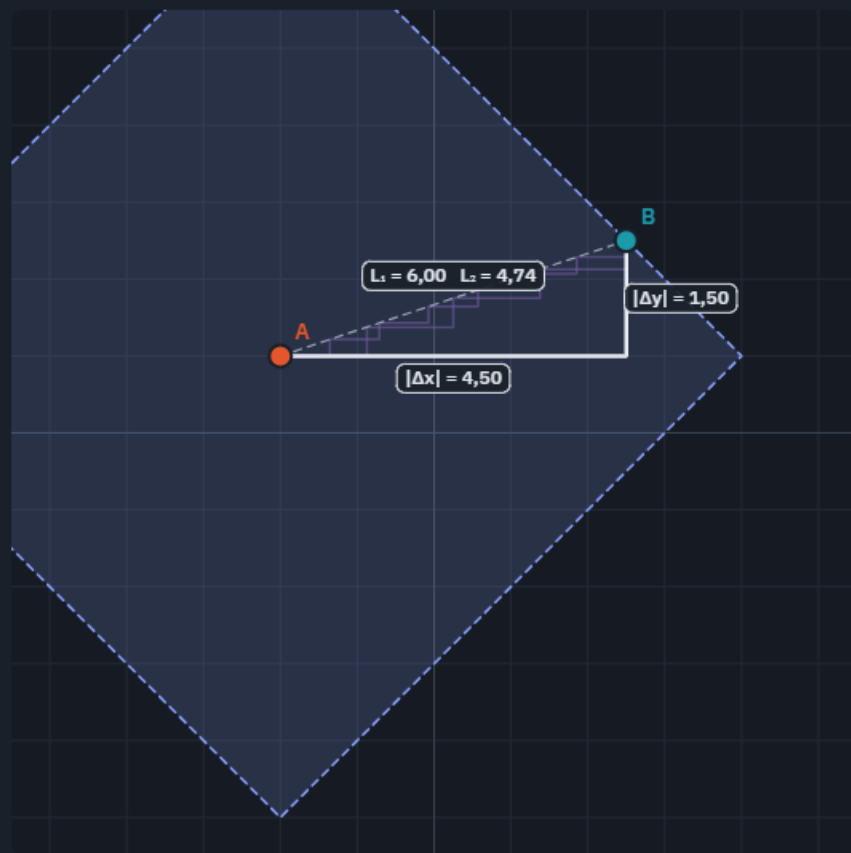
Евклидова

Манхэттенская

Косинусная

Хэмминга

Махаланобиса



● A = (-2, 0; 1, 0) ● B = (2, 5; 2, 5)

Для этих же точек: евклидово **4,74**, манхэттенское **6,00**, косинусное **1,32**, Махаланобиса **2,30** (при текущей Σ).

Манхэттенское расстояние (L_1)

Сумма модулей разностей по каждой координате — путь по «городским кварталам», где двигаться можно только вдоль улиц. Все ступенчатые пути одинаковой длины.

Формула

$$d(a, b) = \sum_i |a_i - b_i|$$

Расчёт для текущего примера

$$d = |4,50| + |1,50| = 4,50 + 1,50 = 6,00$$

Для сравнения евклидово: 4,743 — всегда не больше манхэттенского.

Плюсы

- Устойчивее к выбросам: разность входит линейно, без квадратов
- В высоких размерностях лучше различает точки, чем евклидова
- Естественна для счётчиков, разреженных признаков, сеточных задач
- Каждый признак вносит вклад «как есть», без взаимного усиления

Минусы

- Зависит от выбора осей: при повороте координат расстояние меняется
- Тоже требует нормализации масштаба
- Негладкая в нуле — сложнее для градиентных методов
- Множество равноудалённых точек — ромб, что не всегда соответствует смыслу данных

Когда использовать

Много признаков, есть выбросы, признаки-счётчики; L1-регуляризация (Lasso); маршруты по сетке.

Метрики расстояния

Как разные способы измерить «близость» двух объектов меняют форму окрестности и ответ модели. Перетаскивайте точки А и В на поле, а в примере с Хэммингом — переключайте биты.

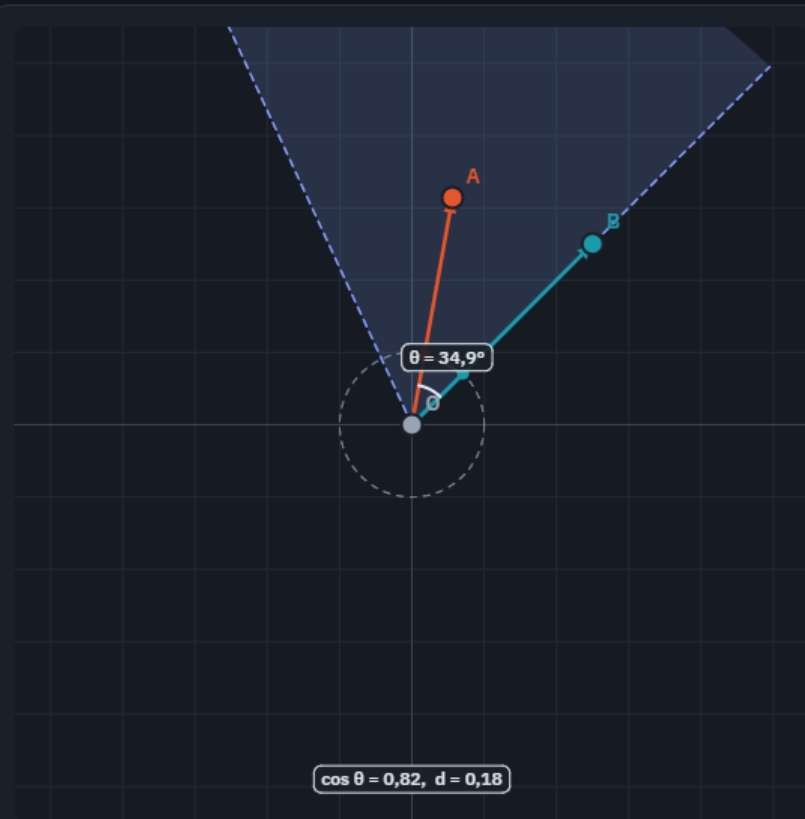
Евклидова

Манхэттенская

Косинусная

Хэмминга

Махаланобиса



● A = (0,6; 3,1) ● B = (2,5; 2,5)

Растянуть вектор В

×1,0

Длина вектора меняется, а угол — нет: косинусное расстояние остаётся прежним, евклидово растёт.

Для этих же точек: евклидово **2,04**, манхэттенское **2,58**, косинусное **0,18**, Махаланобиса **2,55** (при текущей Σ).

Косинусное расстояние

Сравнивает направления векторов, а не их длину: расстояние равно **1 – косинус** угла между ними. Векторы, смотрящие в одну сторону, «одинаковы», как бы ни отличались их длины.

Формула

$$d(a,b) = 1 - \cos \theta = 1 - \frac{a \cdot b}{|a| \cdot |b|}$$

Расчёт для текущего примера

$$a \cdot b = 0,56 \cdot 2,50 + 3,14 \cdot 2,50 = 9,25$$

$$|a| = 3,19, |b| = 3,54$$

$$\cos \theta = 9,25 / (3,19 \cdot 3,54) = 0,821, \theta = 34,9^\circ$$

$$d = 1 - 0,821 = \mathbf{0,179} \text{ (евклидово: 2,041)}$$

Плюсы

- Не зависит от длины вектора: документ и его копия, повторённая дважды, совпадают
- Стандарт для текстов (TF-IDF), эмбеддингов, рекомендаций
- Хорошо работает на разреженных данных высокой размерности
- Значения ограничены: от 0 до 2 (для неотрицательных признаков — от 0 до 1)

Минусы

- Не метрика в строгом смысле: нарушает неравенство треугольника
- Полностью теряет информацию о величине (интенсивности) признаков
- Не определена для нулевого вектора, нестабильна вблизи начала координат
- Зависит от того, где стоит «ноль» — центрирование признаков меняет ответ

Метрики расстояния

Как разные способы измерить «близость» двух объектов меняют форму окрестности и ответ модели. Перетаскивайте точки А и В на поле, а в примере с Хэммингом — переключайте биты.

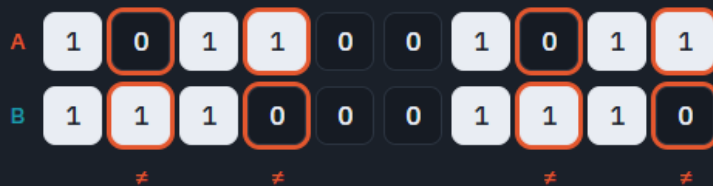
Евклидова

Манхэттенская

Косинусная

Хэмминга

Махаланобиса



Нажимайте на биты, чтобы переключить их. Обведены позиции, в которых строки различаются.

A карета

B ракета

к а р е т а → р а к е т а

Различаются в 2 позициях — $d = 2$. Заметьте: «карета» и «ракета» близки, а сдвиг на одну букву («ракета» → «аракет») даёт большое расстояние.

Расстояние Хэмминга

Число позиций, в которых две последовательности одинаковой длины различаются. Для битовых строк — количество единиц в $a \text{ XOR } b$.

Формула

$$d(a,b) = \sum_i [a_i \neq b_i]$$

Расчёт для текущего примера

Позиции с различием: 2, 4, 8, 10

$d(A,B) = 4$ из 10 (нормированное: 0,40)

Плюсы

- Очень просто и быстро: XOR и подсчёт единиц
- Естественна для бинарных, категориальных и one-hot признаков
- Целочисленная и легко интерпретируется: «сколько признаков не совпало»
- Основа кодов, исправляющих ошибки: код с минимальным расстоянием 3 исправляет одну ошибку

Минусы

- Только для последовательностей одинаковой длины
- Не учитывает степень различия: $1 \neq 2$ «стоит» столько же, сколько $1 \neq 100$
- Сдвиг на одну позицию делает похожие строки совсем разными («карета» и «ккарет»)
- Для текстов обычно лучше расстояние Левенштейна (учитывает вставки и удаления)

Когда использовать

Бинарные векторы, категориальные признаки, хеши, ДНК-последовательности, коды и контрольные суммы.

Метрики расстояния

Как разные способы измерить «близость» двух объектов меняют форму окрестности и ответ модели. Перетаскивайте точки A и B на поле, а в примере с Хэммингом — переключайте биты.

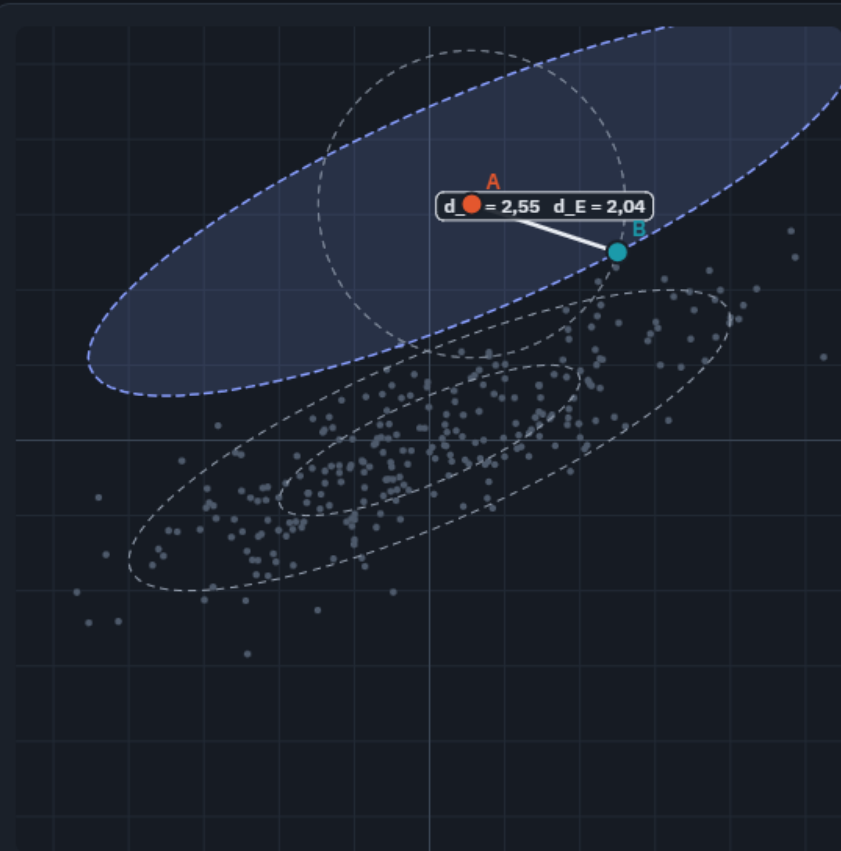
Евклидова

Манхэттенская

Косинусная

Хэмминга

Махаланобиса



● A = (0,6; 3,1) ● B = (2,5; 2,5)

Корреляция признаков ρ

0,80

Разброс по x (σ_x)

2,0

Разброс по y (σ_y)

1,0

Расстояние Махаланобиса

Евклидово расстояние в координатах, где облако данных «выровнено»: оси растянуты по разбросу и повернуты по корреляции. Шаг «вдоль» облака короче, чем такой же шаг «поперёк».

Формула

$$d(a,b) = \sqrt{(a-b)^T \Sigma^{-1} (a-b)}, \quad \Sigma — \text{ковариационная матрица данных}$$

Расчёт для текущего примера

$$\Sigma = \begin{bmatrix} 4,00 & 1,60 \\ 1,60 & 1,00 \end{bmatrix}$$

$$\Delta = (1,94, -0,64)$$

$$d_M = \sqrt{\Delta^T \Sigma^{-1} \Delta} = 2,550 \quad (\text{евклидово: } 2,041)$$

Попробуйте: передвиньте B вдоль облака и поперёк него на одно и то же евклид

Плюсы

- Не зависит от масштаба признаков и их единиц измерения
- Учитывает корреляции: не считает дважды один и тот же «сигнал»
- При $\Sigma = I$ совпадает с евклидовой, то есть обобщает её
- Идеальна для поиска выбросов: «насколько точка нетипична для облака»

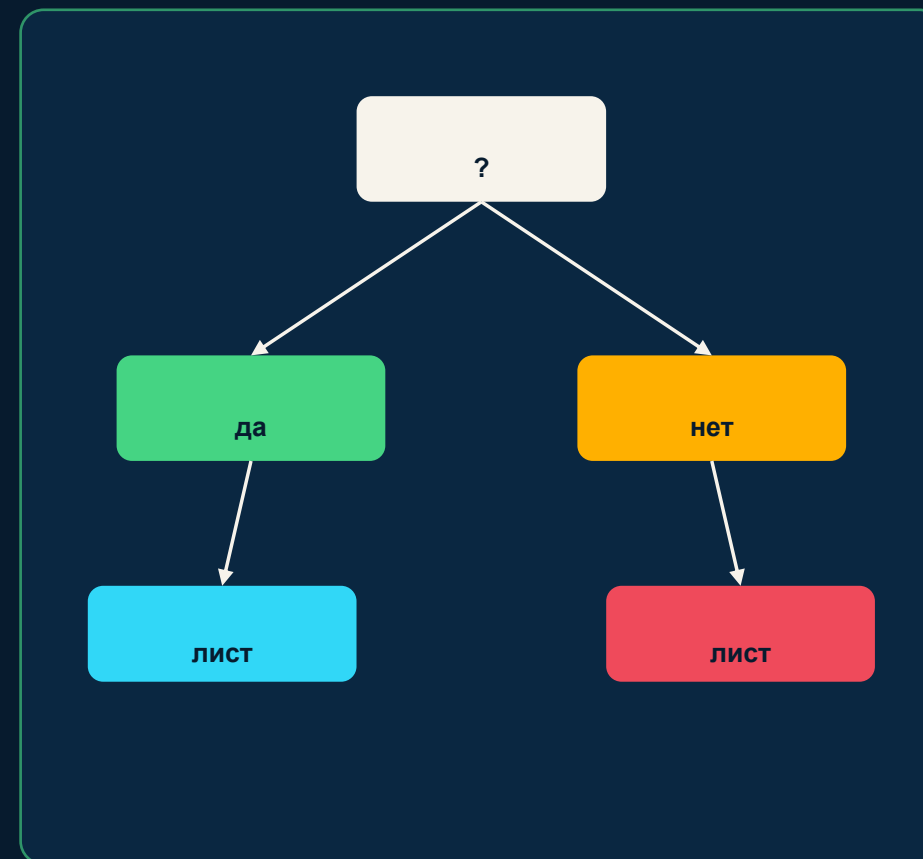
Минусы

- Нужно оценить Σ : при n признаках это n^2 параметров — на малых выборках оценка ненадёжна
- Σ может быть вырожденной (признаков больше, чем объектов) — обратной матрицы нет
- Дороже считать: обращение матрицы, произведения
- Предполагает одно «эллиптическое» облако; для многомодальных данных вводит в заблуждение

А можно получать решение через понятные правила?

Переходим от соседей к вопросам.

Теперь хотим модель, которая объясняет решение понятными правилами.



Урок 12 часть 2 · часть про деревья решений

Как алгоритм превращает вопросы в предсказания

Анализ данных в век ИИ · доктор Игорь Клейнер

Зачем нам деревья решений?

Это один из самых полезных способов строить понятные модели.

Что делает дерево?

Оно разбивает данные серией вопросов “да / нет”, пока не приходит к решению.

Где используется?

В задачах классификации и регрессии, когда нам важны не только прогнозы, но и объяснение.

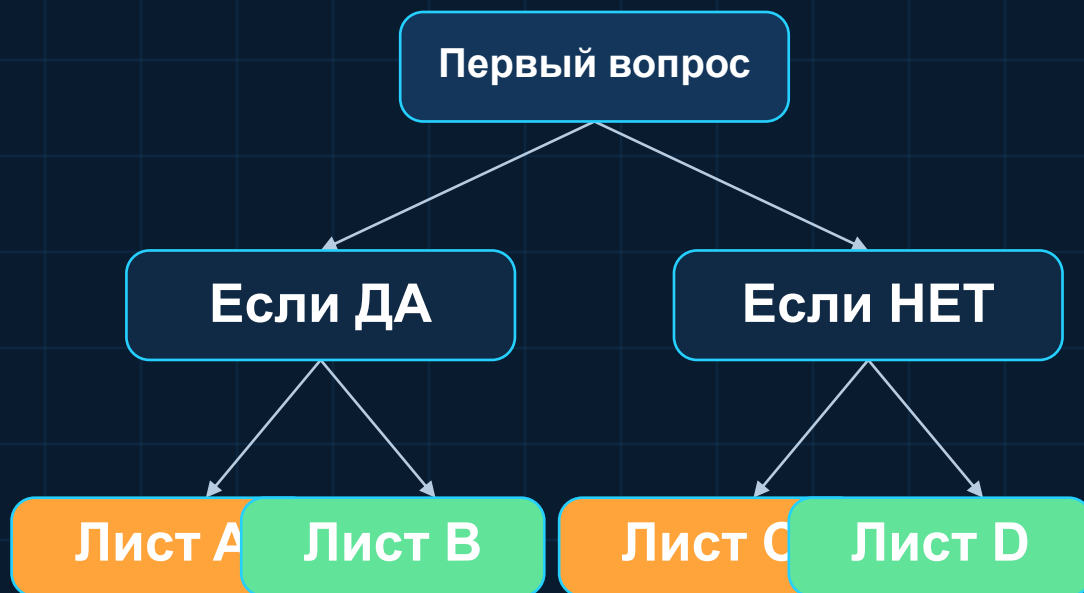
Почему удобно?

Человек может пройти по веткам и увидеть, почему модель сделала именно такой вывод.

Сегодня мы поймём дерево через игру, а потом перенесём идею в машинное обучение.

Дерево решений — это стратегия вопросов

Вместо “посмотрим на похожих” дерево говорит: “давайте зададим хороший вопрос”.



Идея

Каждый вопрос делит данные на две или несколько групп.

Цель

После вопроса группы должны стать более понятными и однородными.

Решение

В конце пути мы попадаем в лист и делаем прогноз.

Начнём с игры “Угадай кто?”

Эта игра почти идеально показывает интуицию дерева решений.



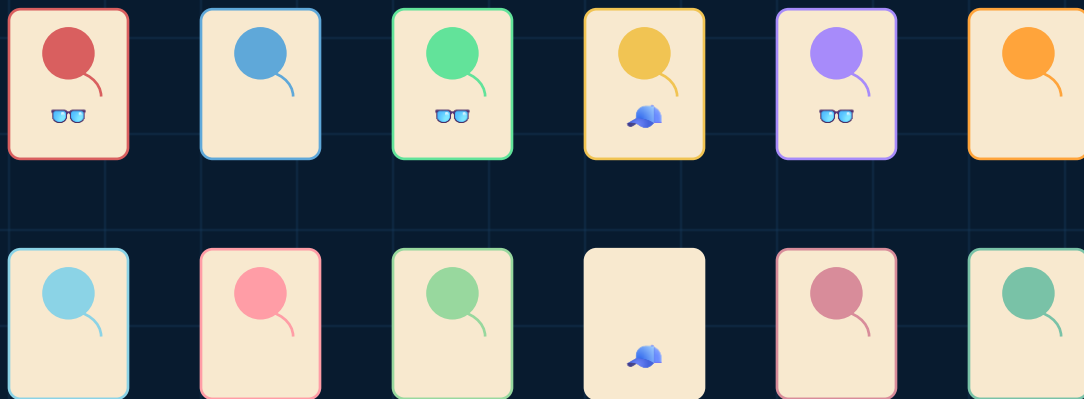
Правила простые:

1. Я загадываю одного персонажа.
2. Вы можете задавать только вопросы с ответом “да” или “нет”.
3. Хорошая стратегия должна найти персонажа за как можно меньше вопросов – зачем много вопросов если достаточно мало?

Сейчас мы не просто играем: мы учимся выбирать полезные вопросы.

У нас есть 12 возможных персонажей

Один из них загадан, но мы не знаем какой.



Задача не в том, чтобы угадывать наугад. Задача — выбирать вопросы, которые быстро убирают много лишних вариантов.

Первый ход

Выбор первого вопроса очень важен: от него зависит весь дальнейший путь.



Задавай
вопросы
Открывай
персонажей
Угадывай!



Хорошие
Вопросы
делают
мир
ярче! ♥

Стандартная колода игры

персонажи для игры в вопросы и догадки



ИГРА
РАЗВИВАЕТ
ЛОГИКУ
И ВНИМАНИЕ

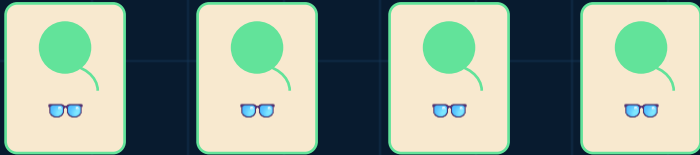


ИГРЫ
ДРУЖБА
СЕМЬЯ
ЗНАНИЯ
УСПЕХ

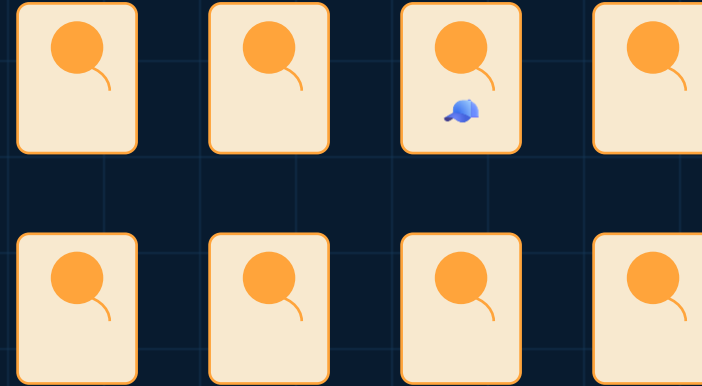
Первый вопрос: “Персонаж носит очки?”

Вопрос делит всех кандидатов на две группы.

ДА



НЕТ



Хороший вопрос не даёт ответ сразу, но резко уменьшает неопределённость. (плохой вопрос – имя начнется на А)

Какой вопрос лучше задать первым?

Мы выбираем не “интересный” вопрос, а вопрос, который помогает быстрее найти ответ.

ВОПРОС

Вариант А

“Персонаж носит очки?”
Этот вопрос делит доску
примерно на две большие
группы.

Вариант В

“У персонажа красная
шапка?”
Этот вопрос может оставить
почти всех на месте.

Перед деревом всегда стоит выбор: какой вопрос действительно уменьшает неопределённость?

Хороший вопрос делит данные полезно

Плохой вопрос почти ничего не меняет.



Хороший split

После вопроса обе группы стали заметно меньше и проще.

Плохой split

Один персонаж ушёл, а почти вся путаница осталась.

Дерево ищет не красивый вопрос, а вопрос, который делает группы чище.

Теперь заменим персонажей на строки таблицы

Игра превращается в алгоритм машинного обучения.

В игре

Персонажи отличаются признаками: очки, шапка, волосы, возраст.

В данных

Объекты отличаются признаками: площадь, доход, возраст, температура.

В модели

Вопросы по признакам ведут к прогнозу: класс или число.

Дерево решений — это “Угадай кто?”, где персонажи заменены строками данных.

Что такое дерево решений?

Это модель, которая принимает решение через последовательность вопросов.

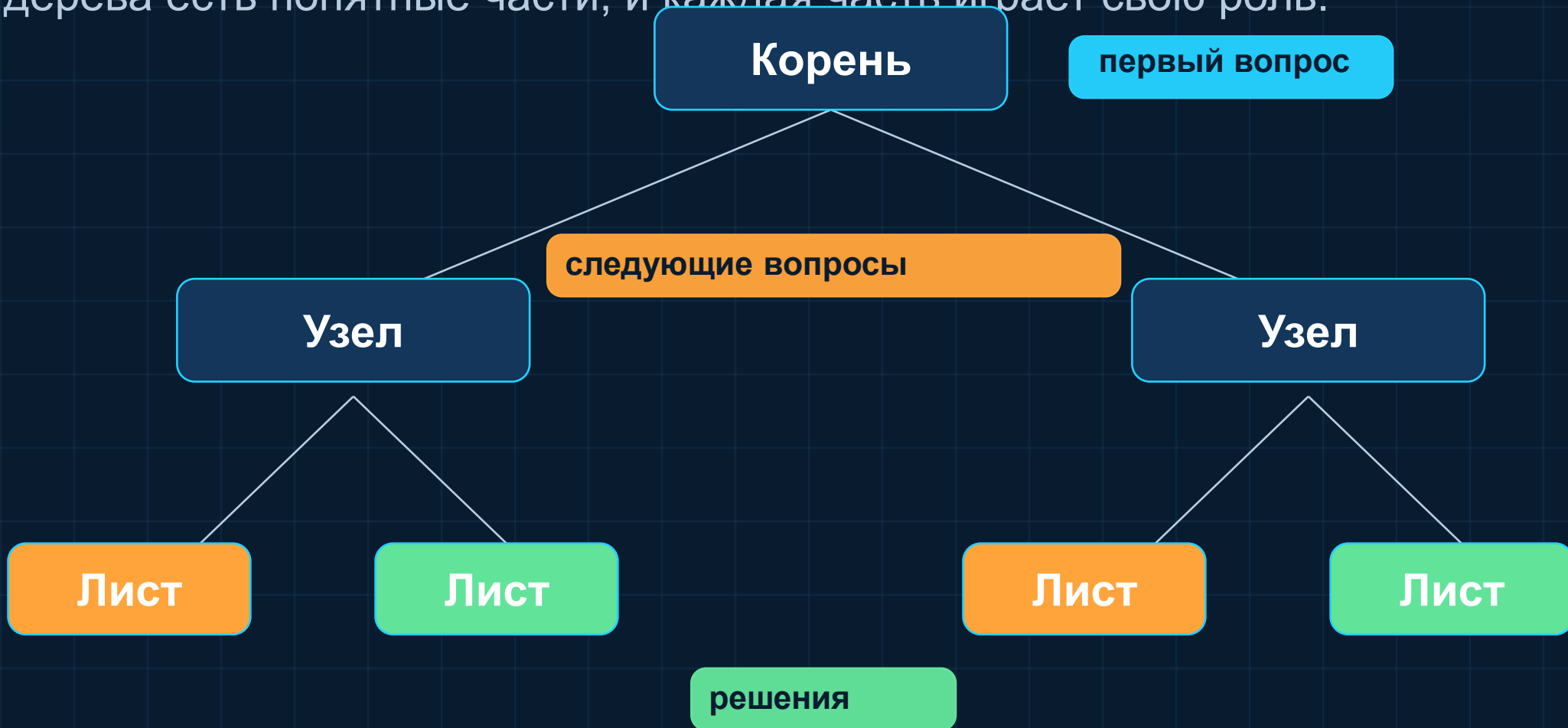


Каждый внутренний узел содержит вопрос о признаке. Каждая ветвь — это возможный ответ. Каждый лист — это итоговое решение модели.

В дереве путь сверху вниз — это объяснение решения.

Анатомия дерева: корень, ветви, листья

У дерева есть понятные части, и каждая часть играет свою роль.



Один объект проходит по дереву как по маршруту

Для нового объекта дерево задаёт вопросы в строгом порядке.

Площадь квартиры $> 70 \text{ м}^2$?

Район близко к центру?

Этаж выше второго?

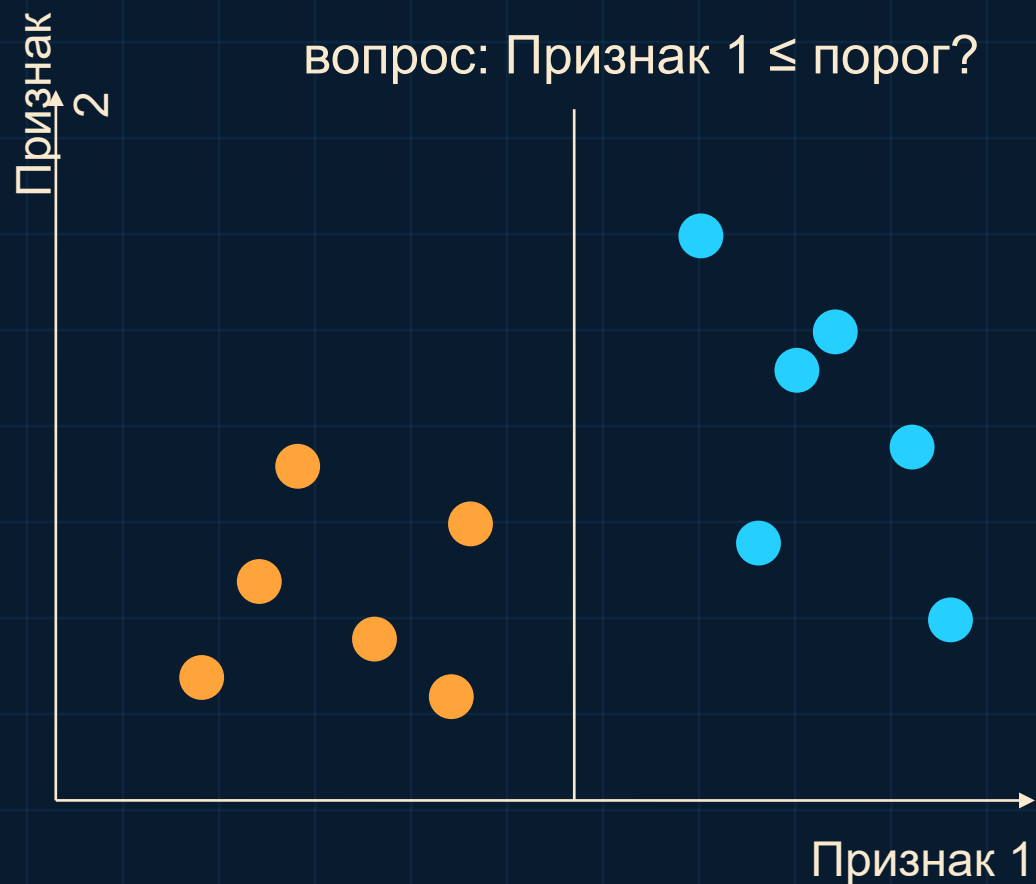
Прогноз: дорогая квартира

Маршрут решения можно прочесть как обычное объяснение: “потому что площадь большая, район центральный, этаж подходит”.

Сильная сторона дерева — оно умеет объяснять решение цепочкой условий.

Дерево для классификации

В листе дерево выбирает класс или вероятность класса.



Классификация

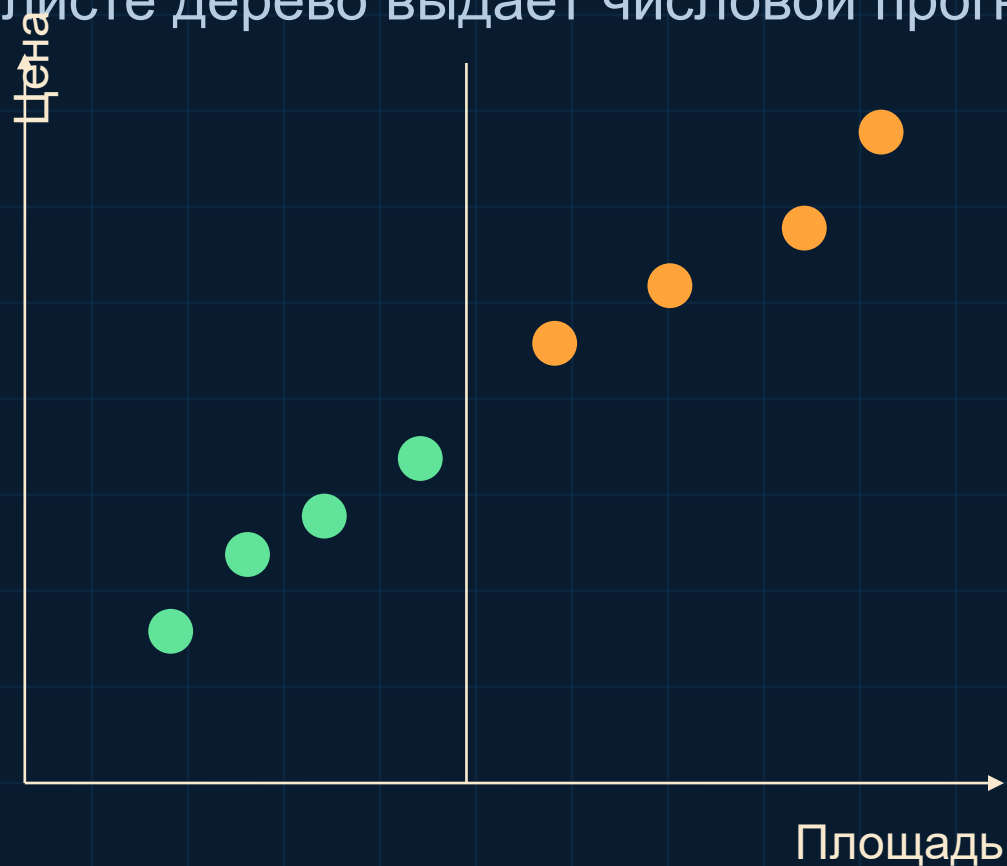
Лист содержит классы, например “покупатель ответит да / нет”.

Важная деталь

Дерево может вернуть не только класс, но и долю объектов каждого класса в листе.

Дерево для регрессии

В листе дерево выдаёт числовой прогноз.



Если площадь $\leq 70 \text{ м}^2$

**Прогноз: средняя цена
в левом листе**

Если площадь $> 70 \text{ м}^2$

**Прогноз: средняя цена
в правом листе**

Как дерево “учится”?

Оно перебирает возможные вопросы и выбирает наиболее полезный.

Шаг 1

Взять признак и возможный порог:
например
“площадь ≤ 70 ”.

Шаг 2

Посмотреть,
насколько хорошо
этот вопрос
разделяет
данные.

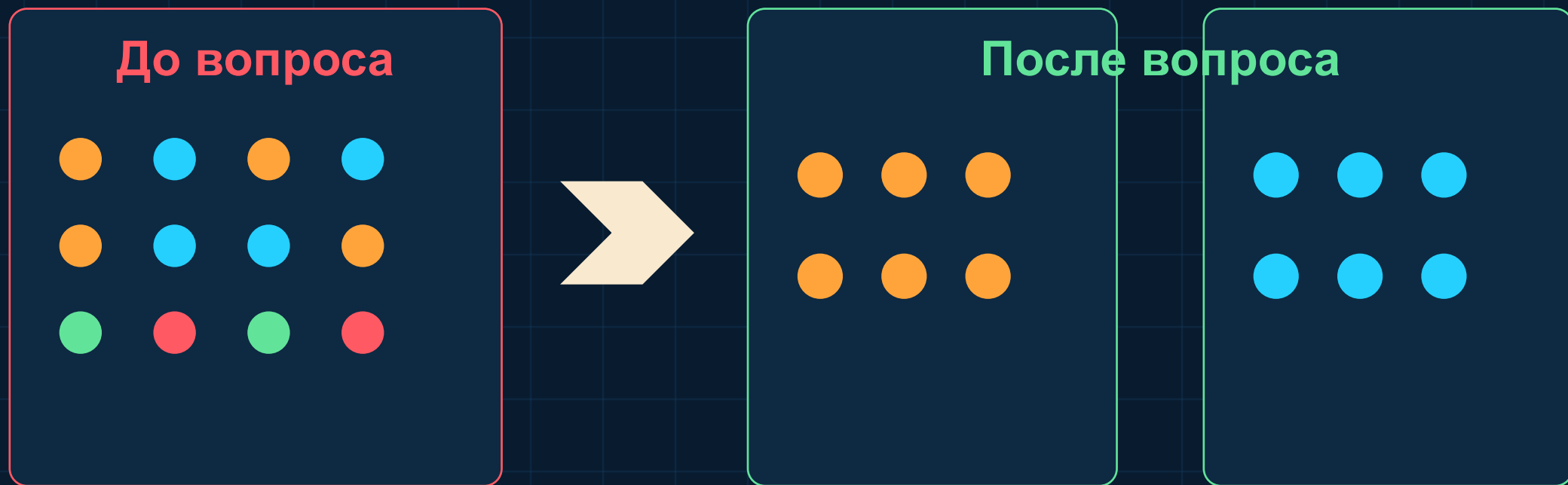
Шаг 3

Выбрать лучший
вопрос и
повторить
процедуру внутри
новых групп.

Дерево строится жадно: на каждом шаге ищет хороший локальный вопрос.

Что значит “хорошо разделяет”?

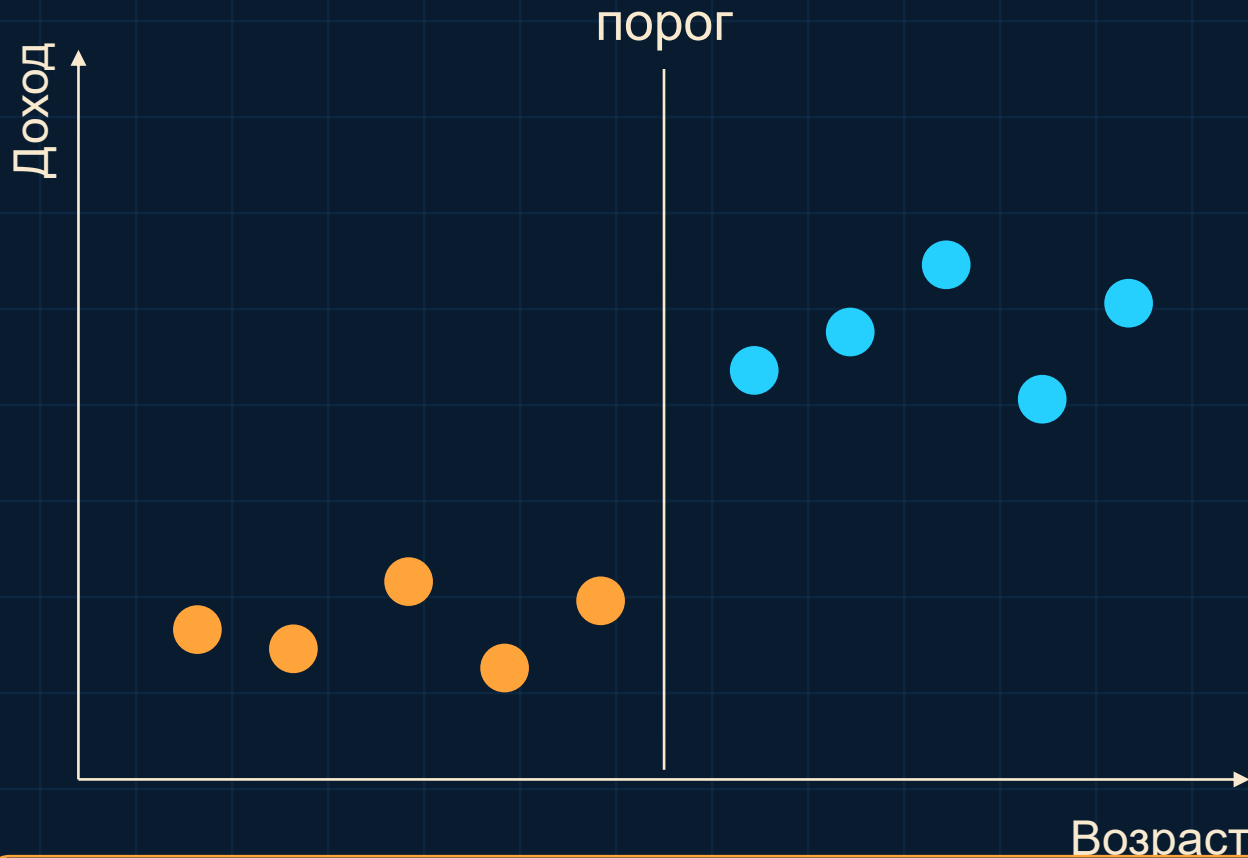
После вопроса группы должны стать более чистыми.



Gini и entropy — это способы измерить, насколько группы стали чище.

Числовой признак: дерево само ищет порог

Например, вопрос может звучать так: “возраст ≤ 35 ?”



Дерево проверяет разные пороги и выбирает тот, после которого данные разделяются лучше всего.

Вопрос в дереве — это не обязательно “да / нет” про текстовый признак. Это может быть порог для числа.

Как дерево делит пространство признаков?

Каждый вопрос добавляет новый разрез.



В двумерной картинке дерево строит области, а в каждой области принимает своё решение.

Почему дереву обычно не нужна нормализация?

Дерево сравнивает значения с порогами, а не измеряет расстояния.

kNN

Сравнивает расстояния.
Поэтому масштаб признаков
может полностью изменить
соседей.

Дерево

Сравнивает порядок
значений и пороги. Если
шкалу умножить, вопрос
просто перепишется.

**Пример: вопрос “ $x \leq 7$ ” после умножения на 1000 станет
“ $1000x \leq 7000$ ”. Логика разбиения не меняется.**

Слишком маленькое дерево недообучается

Если вопросов слишком мало, модель видит только грубую картину.



**Такое дерево легко
объяснить, но оно может
быть слишком грубым.**

**Недообучение: модель слишком проста, чтобы уловить
закономерность.**

Разумное дерево задаёт несколько хороших вопросов

Оно достаточно гибкое, но ещё читаемое человеком.



Баланс

Дерево должно быть достаточно сложным, чтобы ловить структуру, но не настолько большим, чтобы запоминать шум.

Интерпретация

Если человек может пройти по веткам и понять решение, дерево сохраняет своё главное преимущество.

Слишком большое дерево может запомнить данные

На обучающей выборке оно выглядит героем, а на новых данных может провалиться.



Проблема

Дерево может продолжать задавать вопросы, пока почти каждый пример не окажется в своём отдельном листе.

Цена

На train всё прекрасно, но новая ситуация уже не похожа на запомненный маршрут.

Переобучение: дерево выучило детали прошлого, но потеряло общую закономерность.

Как укрощают дерево?

Мы заранее ограничиваем, насколько подробно оно может расспрашивать данные.

Максимальная глубина

Сколько уровней вопросов разрешено построить.

Минимум объектов в листе

Не позволяем дереву делать лист ради одного случайного примера.

Обрезка дерева

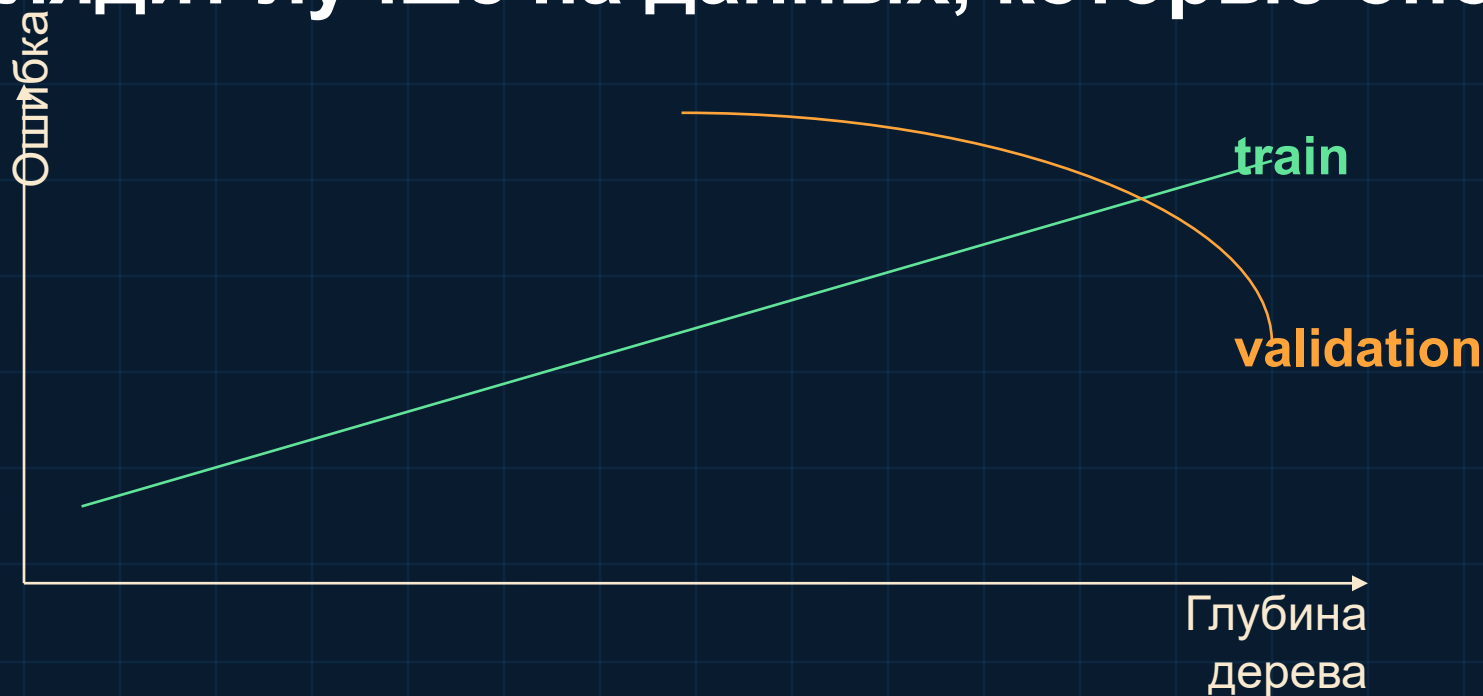
Удаляем ветки, которые дают красивую сложность, но не улучшают обобщение.

Настройка дерева — это борьба с соблазном задать слишком много вопросов.

Как выбрать сложность дерева?

Проверяем качество не только на обучающих данных.

Train может обмануть: глубокое дерево почти всегда выглядит лучше на данных, которые оно уже видело.



Идея

Ищем не самое глубокое дерево, а дерево с лучшим качеством на проверке.

Дерево решений в Orange

Практическая цепочка проста: данные → дерево → проверка → просмотр дерева.



В Orange дерево удобно не только обучать, но и читать через Tree Viewer.

Настройки Tree в Orange

The screenshot shows the Orange3 workflow editor with a 'File' widget connected to a 'Tree' widget, which is then connected to a 'Test and Score' widget. The 'Tree' widget's settings window is open, showing the following parameters:

- Name:** Tree
- Parameters:**
 - ☐ Induce binary tree
 - ☒ Min. number of instances in leaves: 5
 - ☒ Do not split subsets smaller than: 5
 - ☒ Limit the maximal tree depth to: 2
- Classification:**
 - ☐ Stop when majority reaches [%]: 95
 - ☒ Apply Automatically

The 'Test and Score' widget's settings window is also open, showing the following options:

- Cross validation:**
 - Number of folds: 5
 - ☐ Stratified
- Cross validation by feature:** (empty dropdown)
- Random sampling:**
 - Repeat train/test: 10
 - Training set size: 66 %
 - ☒ Stratified
- Leave one out**
- Test on train data**
- Test on test data**

The 'Evaluation results for target' table shows:

Model	AUC
Tree	0.741

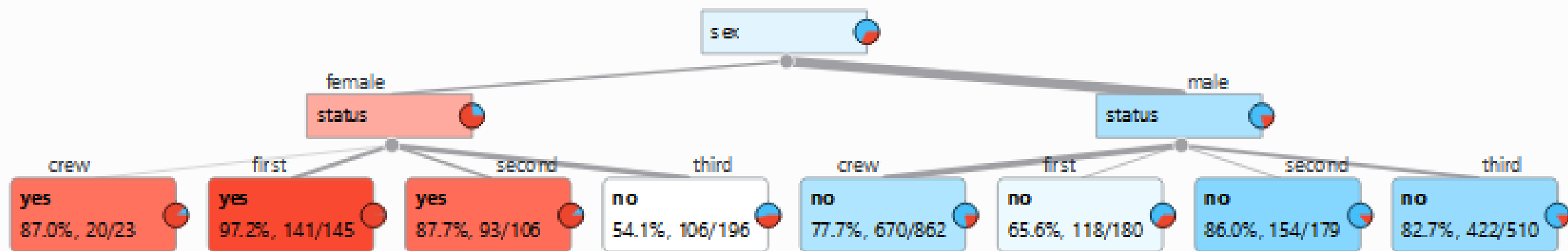
The 'Compare models by' dropdown is set to 'Area und'.

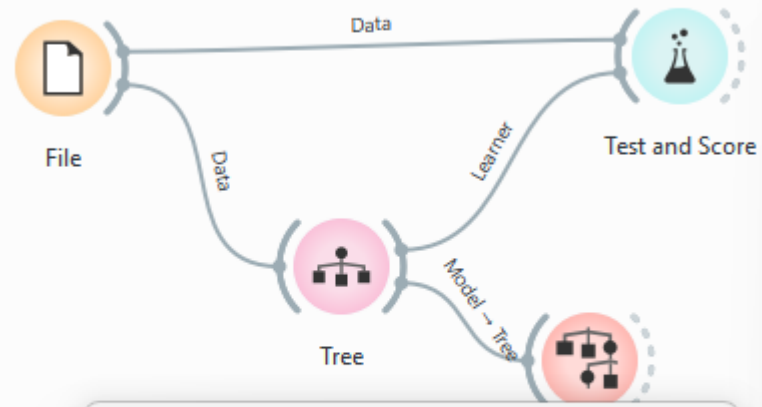
Что подсветить

depth, minimum instances in leaves
и сам вид дерева.

Что сказать

Мы ограничиваем дерево, чтобы оно не
запоминало данные слишком подробно.





Tree - Orange

File View Window Help

Name

Tree

Parameters

☐ Induce binary tree

☒ Min. number of instances in leaves: 5

☒ Do not split subsets smaller than: 5

☒ Limit the maximal tree depth to: 3

Classification

☐ Stop when majority reaches [%]: 95

☒ Apply Automatically

2201 | -

Test and Score - Orange

File Edit View Window Help

☒ Cross validation

Number of folds: 5

☐ Stratified

☐ Cross validation by feature

☐ Random sampling

Repeat train/test: 10

Training set size: 66 %

☒ Stratified

☐ Leave one out

☐ Test on train data

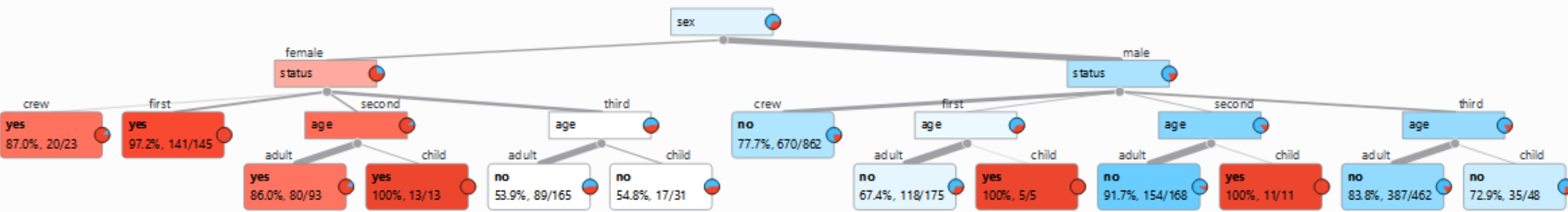
☐ Test on test data

Evaluation results for target

Model	AUC
Tree	0.757

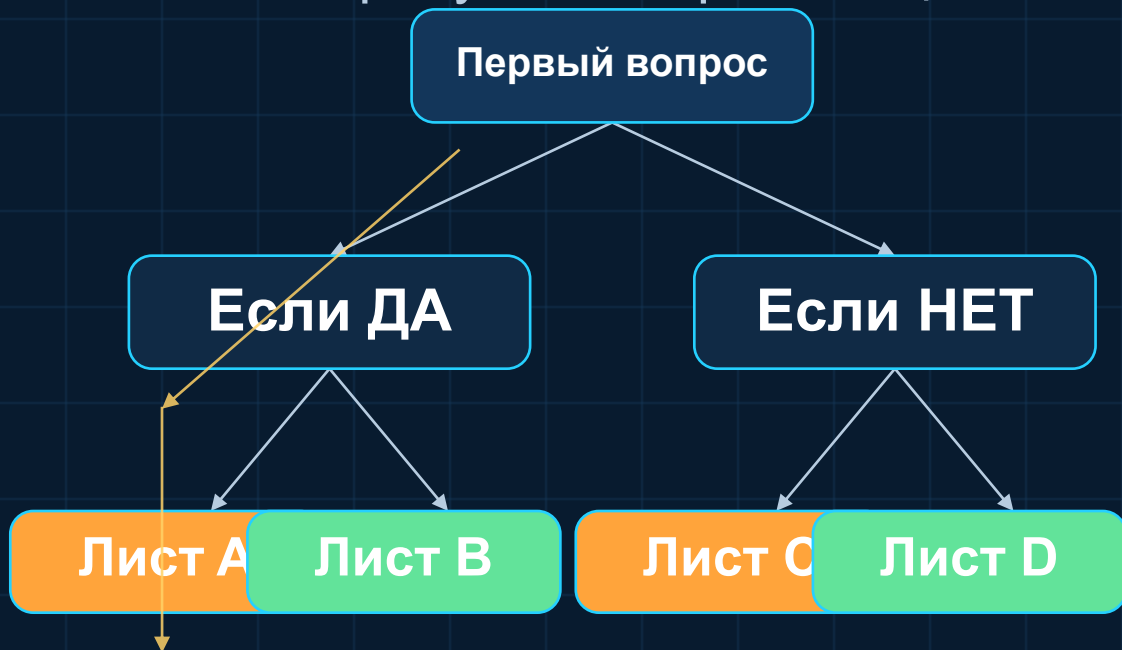
Compare models by: Area u

Tree	
Tree	



Как читать дерево в Tree Viewer

Нас интересует не картинка целиком, а путь от корня к листу.



Чтение дерева

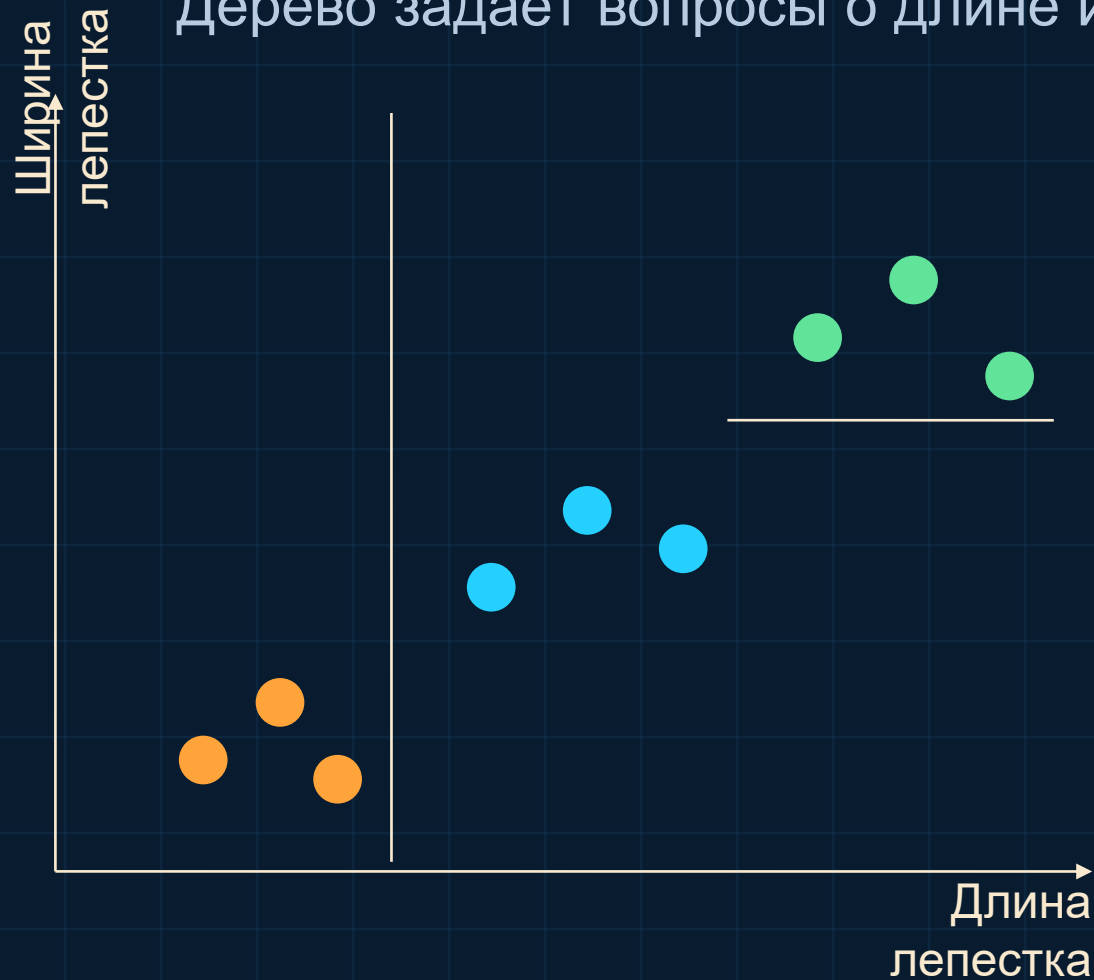
Начинаем сверху, отвечаем на вопрос, идём по ветке и продолжаем до листа.

Объяснение

Именно этот путь можно превратить в понятную фразу для человека.

Мини-пример: ирисы

Дерево задаёт вопросы о длине и ширине лепестков.



пример вопросов:
“petal length $\leq$...?”
“petal width $\leq$...?”

Смысл

Дерево ищет простые пороги, которые хорошо отделяют классы.

Практическое задание в Orange

Проверяем дерево как модель, а не любимся картинкой.

1. Построить

Загрузите данные и подключите Tree к Test & Score.

2. Настроить

Сравните маленькое дерево и более глубокое дерево.

3. Объяснить

Откройте Tree Viewer и прочитайте один путь до листа.

Хороший результат — это не только метрика, но и понятное объяснение решения.

Проверка понимания

Какой первый вопрос лучше для дерева?

ВОПРОС

**У нас 100 объектов: 50 красных и 50 синих.
Какой вопрос выглядит более полезным?**

A

Вопрос, после которого получается 49/1 и 1/49.

B

Вопрос, после которого почти всё остаётся перемешанным.

Ответ

Полезный вопрос делает группы более чистыми и понятными.

ОТВЕТ

Лучше вариант А: после такого вопроса каждая группа почти “знает”, какой класс в ней доминирует.

Именно это дерево пытается делать на каждом шаге: уменьшать неопределённость.

Дерево строит не красивую схему, а последовательность полезных разделений.

Совет от Игоря

Дерево хорошо тем, что его можно объяснить — но только пока оно читаемое.



Если дерево занимает весь экран и требует лупу, это уже не объяснимая модель, а ботанический сад с сюрпризами.

Интерпретируемость — это не название модели. Это способность человека реально прочитывать решение.

Итог: что нужно помнить о деревьях решений?

Если эта карта понятна, тема собрана.

1. Идея

Дерево принимает решение через последовательность вопросов.

2. Обучение

Дерево ищет вопросы, которые делают группы более чистыми.

3. Риск

Слишком глубокое дерево может запомнить данные и переобучиться.

Главная фраза: дерево — это модель вопросов, а не просто красивая схема с ветками.

Игра превращается в дерево

Корень — первый вопрос. Ветви — ответы. Лист — решение.



Дерево — это маршрут по ответам к решению.

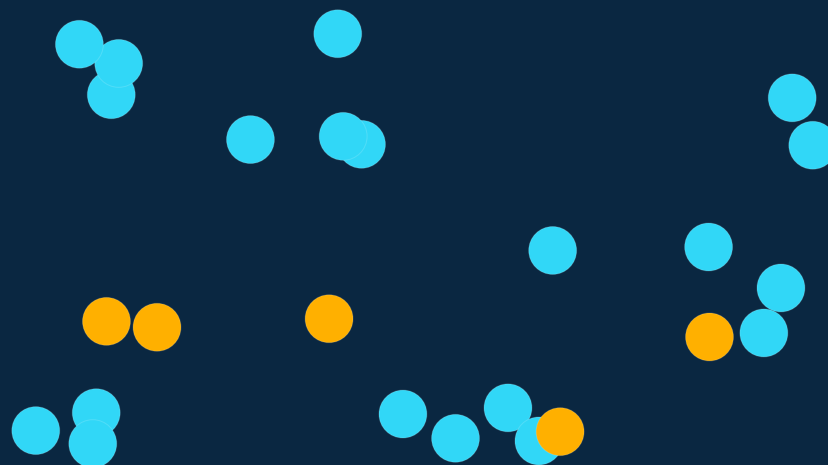
Gini / entropy — только смысл

Меньше перемешанности = меньше неопределённости.

перемешано



чище



Дерево-пень

Слишком мало вопросов — модель слишком простая.



Слишком простая модель не видит структуру данных.

Разумное дерево

Достаточно вопросов, чтобы поймать структуру, но не запомнить шум.

Разумная сложность ловит
закономерность, но не запоминает шум.



Дерево-монстр

Слишком глубокое дерево может запомнить обучающие данные.

Слишком сложная модель может запомнить шум вместо закономерности.



Как укрощают дерево

Ограничивают глубину и размер листьев, а параметры проверяют на validation.

Дерево подрезают, чтобы оно объясняло данные, а не запоминало лес.

- max depth
- min samples leaf
- min samples split
- validation / CV

Совет от Игоря

Интерпретируемое дерево должно помещаться в голову.

**Если дерево имеет 147 уровней,
это уже не объяснение, а ботанический лес.**

Совет от Игоря

Вопрос → ответ

Что сильнее зависит от нормализации: kNN или дерево?

kNN ищет похожих

Tree задаёт вопросы

Вопрос → ответ

Что сильнее зависит от нормализации: kNN или дерево?

Ответ: kNN сильнее зависит от нормализации.

kNN ищет похожих

Tree задаёт вопросы

Во второй части время начнёт иметь значение.

Ловушки при работе с деревьями

Дерево может выглядеть умным и понятным, но всё равно ошибаться. Чаще всего проблемы начинаются в трёх местах.



Поэтому дерево почти всегда надо проверять на новых данных.

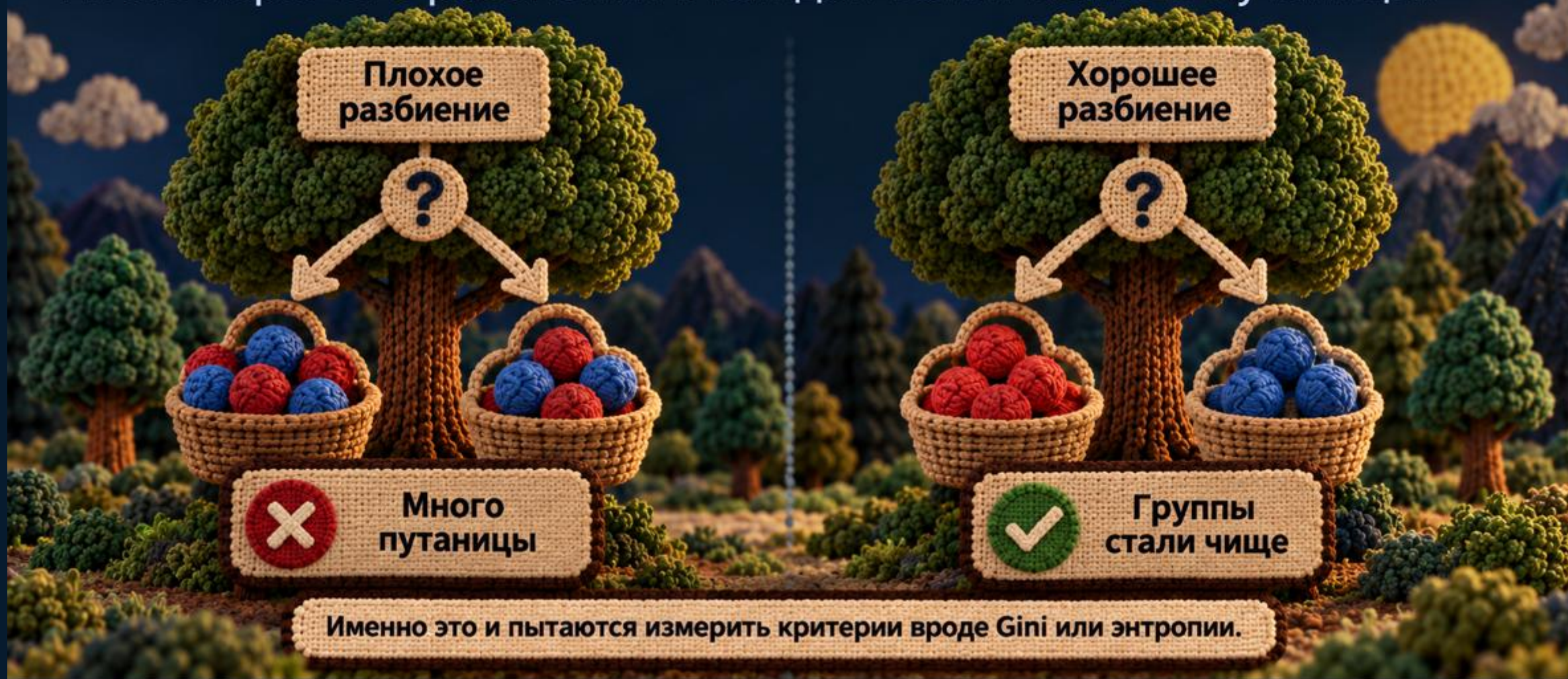
Плюсы и минусы дерева решений

Дерево удобно тем, что его легко объяснять. Но за простоту приходится платить нестабильностью и риском переобучения.



Что значит: группы стали чище?

Хороший признак быстро отделяет разные ответы друг от друга.
После хорошего разбиения в каждой ветви меньше путаницы.



От одного дерева к лесу

Одно дерево может быть понятным, но нестабильным. Если объединить много деревьев, модель обычно становится надёжнее.

1

Одно дерево принимает решение по последовательности вопросов.

2

Несколько деревьев смотрят на задачу по-разному.

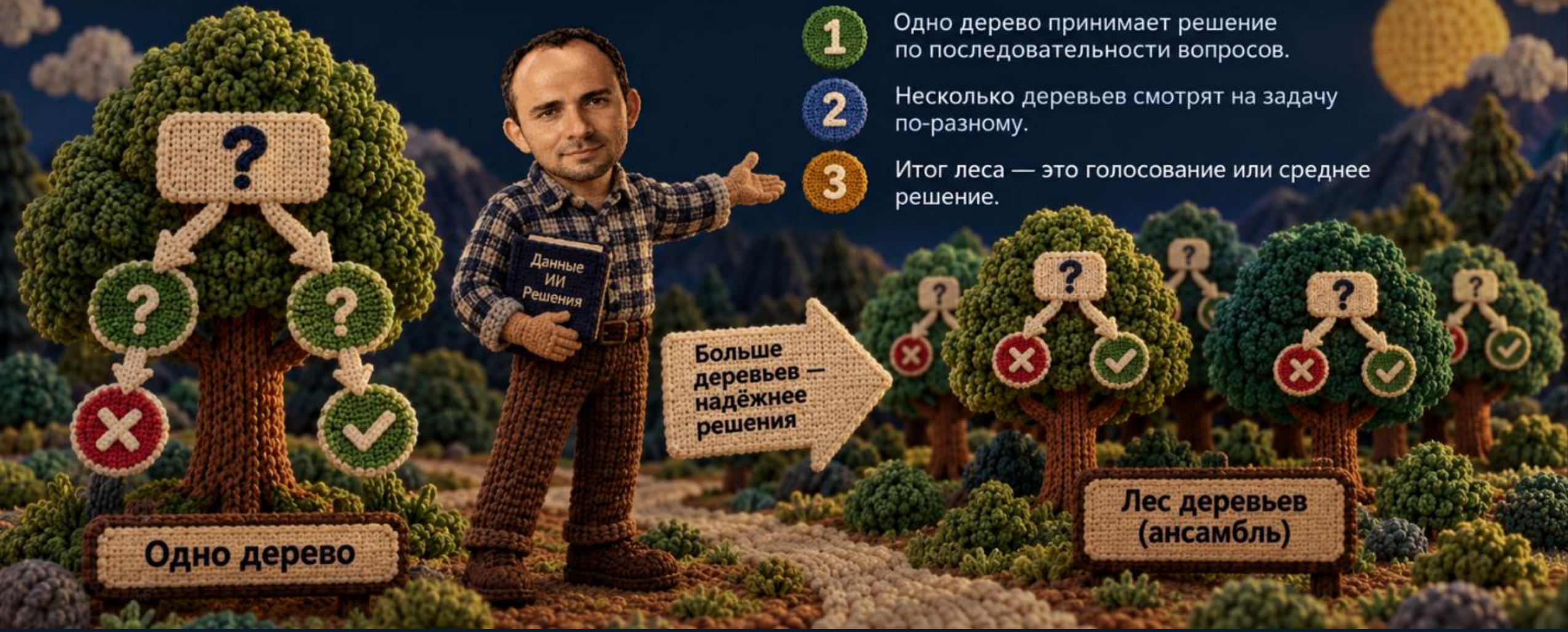
3

Итог леса — это голосование или среднее решение.

Больше
деревьев —
надёжнее
решения

Одно дерево

Лес деревьев
(ансамбль)



ПАНОРАМА ЗНАНИЙ: kNN в Orange

Как метод k ближайших соседей принимает решение

Данные
+ правильные инструменты
= большие возможности! ♥

Открывай
закономерности
вместе с Orange! ♥

ДАННЫЕ

МЕТОДЫ

ПОНИМАНИЕ

ПРАКТИКА

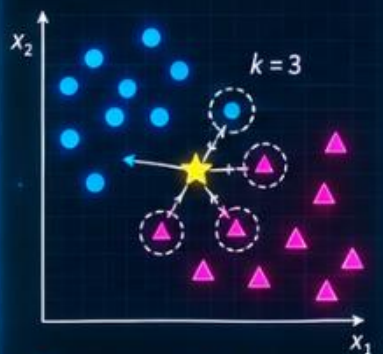
РЕЗУЛЬТАТЫ

1. ИДЕЯ МЕТОДА

Новый объект смотрит
на k ближайших соседей.

Классификация:
берём большинство
классов среди соседей.

Регрессия:
усредняем значения
соседей.



● Класс А ▲ Класс В
★ Новый объект ○ Ближайшие соседи

2. ГЛАВНЫЕ НАСТРОЙКИ В ORANGE

k — число соседей
Сколько ближайших соседей учитывать

5

Метрика расстояния
Euclidean, Manhattan, Mahalanobis, Hamming, Cosine

Euclidean

Взвешивание соседей
Все соседи равны или более близкие имеют больший вес (distance weighting)

Равные веса (uniform)

Нормализация признаков
Стандартизировать признаки (важно для корректных расстояний)

Да

Тип задачи
Классификация Регрессия

Большинство классов Среднее значение

3. КАК ВЛИЯЕТ k?

Малое k (например, k = 1)

- более гибкая граница
- может учитывать шум
- высокая чувствительность



Граница сложная, «подстраивается» под данные (может быть шумной)

Большое k (например, k = 15)

- более стабильная граница
- менее чувствительна к шуму
- может «сглаживать» структуру



Граница более гладкая (может быть слишком простой)

4. СИЛЬНЫЕ СТОРОНЫ

Простой и интуитивно понятный метод

Не требует сложного обучения

Хорошо работает с нелинейными границами

Часто даёт сильный baseline

Отличный выбор для быстрого старта и сравнения с другими методами!

5. СЛАБЫЕ СТОРОНЫ

Чувствителен к масштабу признаков

Медленный на больших наборах данных

Страдает от нерелевантных признаков и высокой размерности

Чувствителен к шуму и выбросам

Важно правильно готовить данные и подбирать настройки!

6. ПРАКТИЧЕСКИЕ СОВЕТЫ

Стандартизируйте числовые признаки (Normalize)

Выбирайте k с помощью кросс-валидации (виджет Test & Score)

Следите за несбалансированностью классов

Убирайте нерелевантные признаки (отбор признаков)

Хорошие данные + проверка настроек = надёжные результаты!

7. ТИПИЧНЫЙ WORKFLOW В ORANGE



8. ГЛАВНАЯ МЫСЛЬ

Изменение метрики расстояния или числа соседей k может изменить, кто попадёт в ближайшие соседи, а значит — и предсказание.

Небольшое изменение — большое влияние! ♥



ПАНОРАМА ЗНАНИЙ: Decision Tree в Orange

Как дерево решений шаг за шагом делит данные

Понятные
модели
реальные
результаты

ДАННЫЕ
АЛГОРИТМЫ
ИНТЕРПРЕТАЦИЯ
ВОЗМОЖНОСТИ

Большие идеи
начинаются
с простых вопросов

Одно дерево —
много инсайтов

ДАННЫЕ → ВОПРОСЫ → РЕШЕНИЯ

ИДЕЯ МЕТОДА

Модель задаёт последовательность простых вопросов и делит данные на ветви, пока не дойдёт до листа (класса).



В каждом узле — простой признак и порог. В каждом листе — предсказанный класс.

КАК ПРИНИМАЕТ РЕШЕНИЕ

На каждом шаге дерево ищет разбиение, которое лучше всего разделяет классы (уменьшает "неоднородности"). Путь от корня до листа даёт итоговое предсказание.



Итог: набор вопросов → лист → класс

ГЛАВНЫЕ НАСТРОЙКИ В ORANGE

Виджет Tree позволяет управлять сложностью и качеством дерева.

	Максимальная глубина Ограничивает, насколько глубоким может быть дерево.	max depth 5
	Минимум примеров в узле Минимальное число объектов для разбиения узла.	min samples 5
	Минимум примеров в листе Минимальное число объектов в конечном узле.	min leaf 2
	Критерий качества разбиения Оценка "чистоты" разбиения (Gini, Entropy, Gain Ratio).	criterion Gini
	Обрезка / остановка роста Уменьшает переобучение, делает дерево проще.	☒ prune
	Бинарные разбиения Делить на две ветви (да/нет) или использовать многовариантные разбиения (для категориальных).	split type Binary

СИЛЬНЫЕ СТОРОНЫ

- Легко интерпретировать и объяснять
- Видимые и понятные правила
- Работает с числовыми и категориальными признаками
- Обычно не требует масштабирования данных

СЛАБЫЕ СТОРОНЫ

- Может переобучаться (слишком сложное дерево)
- Нестабилен при небольших изменениях в данных
- Жадные локальные решения (не гарантирует глобальный оптимум)
- Одно дерево может обобщать хуже, чем ансамбли (например, Random Forest)

ПРАКТИЧЕСКИЕ СОВЕТЫ

- ✓ Ограничьте глубину дерева.
- ✓ Проверяйте качество на Test & Score (кросс-валидация).
- ✓ Используйте обрезку или упрощение дерева.
- ✓ Обязательно смотрите на дерево в Tree Viewer.
- ✓ Сравнивайте с другими моделями (kNN, SVM, Random Forest).

Простое
дерево часто
работает
очень хорошо!

ТИПИЧНЫЙ WORKFLOW В ORANGE

Классический процесс анализа с деревом решений.



От данных к пониманию

ГЛАВНАЯ МЫСЛЬ

Каждое разбиение → это вопрос, а всё дерево → карта решений.

Простые
вопросы.
Большие
возможности.

Учиться
значит
видеть
больше