

## ЗАНЯТИЕ 2.1

# ДЕРЕВЬЯ РЕШЕНИЙ.

# КЛАССИФИКАЦИЯ



Артур  
Сапрыкин

---

# ЦЕЛИ ЗАНЯТИЯ

---

## В КОНЦЕ ЗАНЯТИЯ ВЫ НАУЧИТЕСЬ:

- применять **алгоритм** классификации, принятие решений которого **можно проинтерпретировать**
- **измерять качество** решений в задачах классификации
- **оценивать важность** фичей
- понимать **основу продвинутых алгоритмов**, таких как RandomForest, XGBoost, LGBM, etc..

---

О ЧЁМ ПОГОВОРИМ И ЧТО  
СДЕЛАЕМ

- 
1. Задача классификации: постановка и примеры
  2. Дерево решений: как его построить?  
Обзор `sklearn.tree.DecisionTreeClassifier`
  3. Достоинства и недостатки деревьев решений.
  4. Визуализируем принятие решений и предсказания алгоритма; примем участие в соревновании Kaggle
  5. Метрики качества в задачах классификации
  6. Оценим решение Kaggle; классифицируем статьи Ведомостей: политика или финансы?

---

# 1. ЗАДАЧА КЛАССИФИКАЦИИ

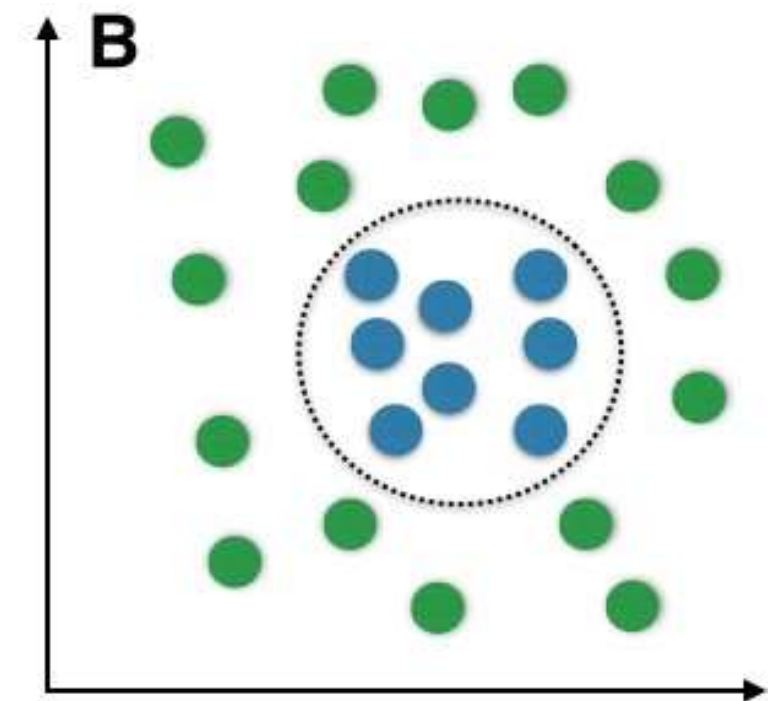
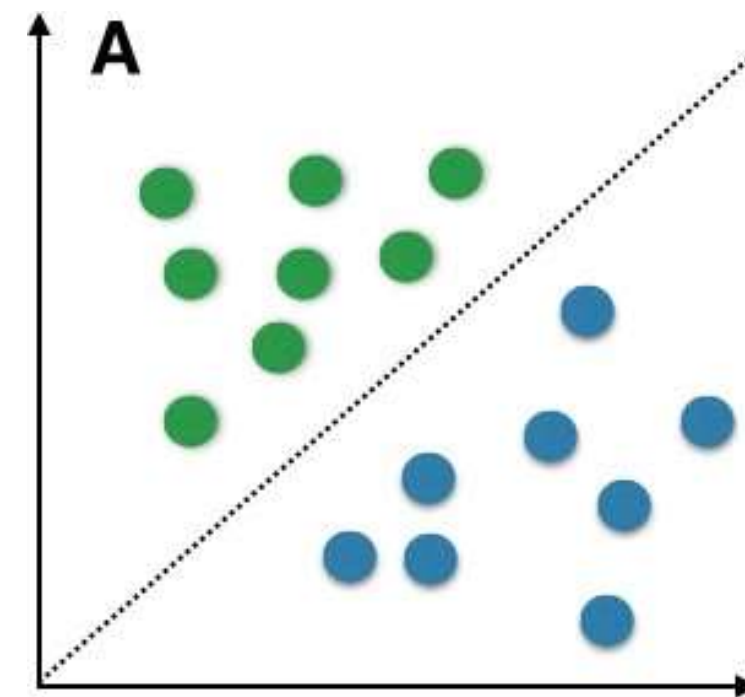
# ТИПЫ ЗАДАЧ

\* **классификация**

\* ранжирование

\* регрессия

\* кластеризация





## ПРИМЕРЫ ЗАДАЧ КЛАССИФИКАЦИИ

**Скоринг.** Вернёт ли клиент кредит? (banking, insurance)

**Отток.** Перестанет ли пользоваться клиент услугами компании?  
Перестанет ли, если дать ему бонус? (marketing)

**Intent recognition.** О чём говорит пользователь в своём обращении?  
(может быть несколько intent'ов, может быть древовидная структура)  
(API.AI)

**Image recognition.** Что на картинке? (Google, FindFace)

# ПОСТАНОВКА ЗАДАЧИ

Задача восстановления зависимости  $y: X \rightarrow Y$ ,  $|Y| < \infty$   
по точкам *обучающей выборки*  $(x_i, y_i)$ ,  $i = 1, \dots, \ell$ .

**Дано:** векторы  $x_i = (x_i^1, \dots, x_i^n)$  — объекты обучающей выборки,  
 $y_i = y(x_i)$  — классификации, ответы учителя,  $i = 1, \dots, \ell$ :

$$\begin{pmatrix} x_1^1 & \dots & x_1^n \\ \dots & \dots & \dots \\ x_\ell^1 & \dots & x_\ell^n \end{pmatrix} \xrightarrow{y^*} \begin{pmatrix} y_1 \\ \dots \\ y_\ell \end{pmatrix}$$

**Найти:** функцию  $a(x)$ , способную классифицировать объекты  
произвольной *тестовой выборки*  $\tilde{x}_i = (\tilde{x}_i^1, \dots, \tilde{x}_i^n)$ ,  $i = 1, \dots, k$ :

$$\begin{pmatrix} \tilde{x}_1^1 & \dots & \tilde{x}_1^n \\ \dots & \dots & \dots \\ \tilde{x}_k^1 & \dots & \tilde{x}_k^n \end{pmatrix} \xrightarrow{a?} \begin{pmatrix} a(\tilde{x}_1) \\ \dots \\ a(\tilde{x}_k) \end{pmatrix}$$

---

## 2. ПОСТРОЕНИЕ ДЕРЕВА РЕШЕНИЙ



## ЦВЕТКИ ИРИСА: ЗАДАЧА



Ирис щетинистый  
(*Iris setosa*)



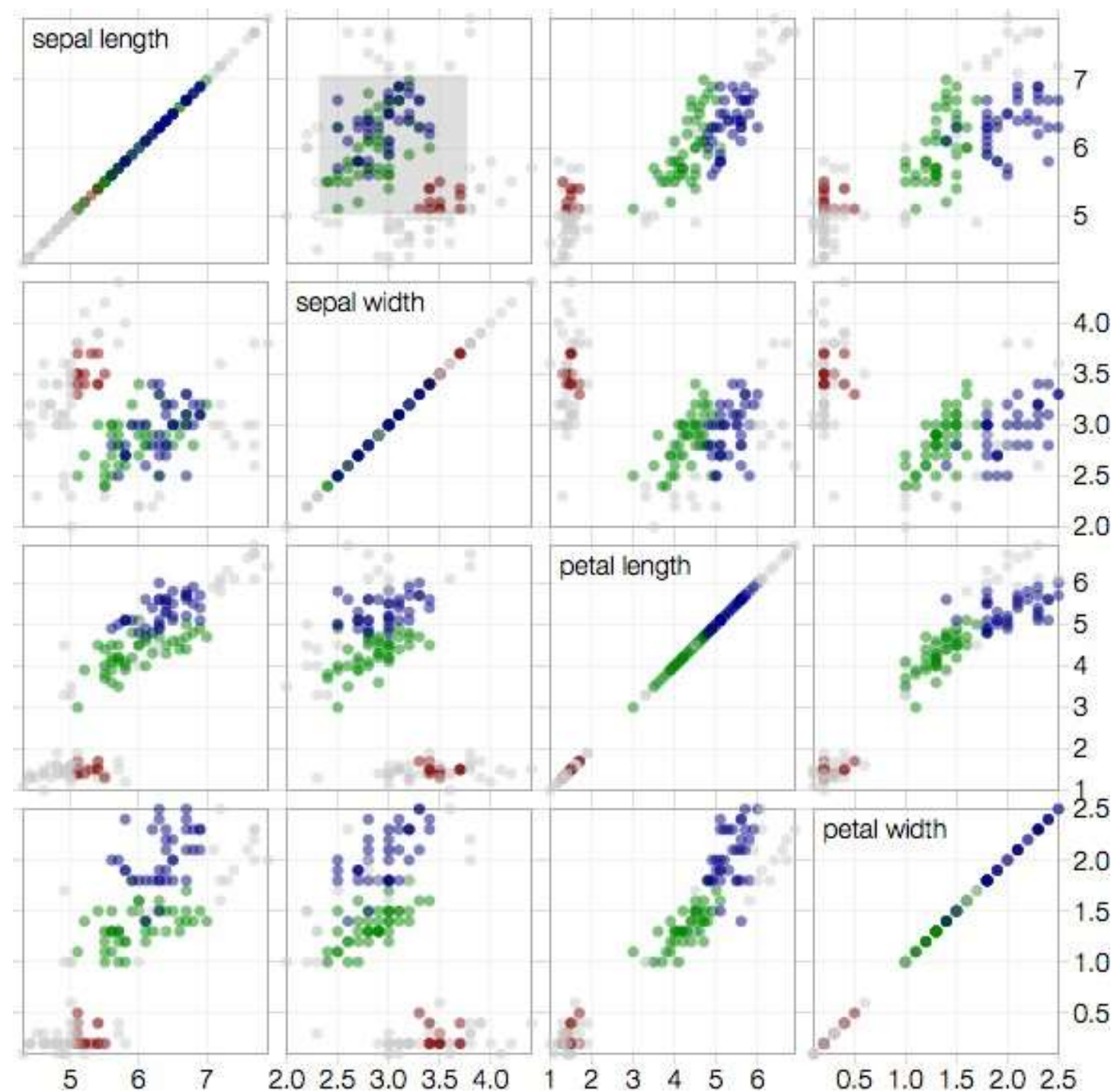
Ирис разноцветный  
(*Iris versicolor*)



Ирис виргинский  
(*Iris virginica*)



# ЦВЕТКИ ИРИСА: ДАННЫЕ



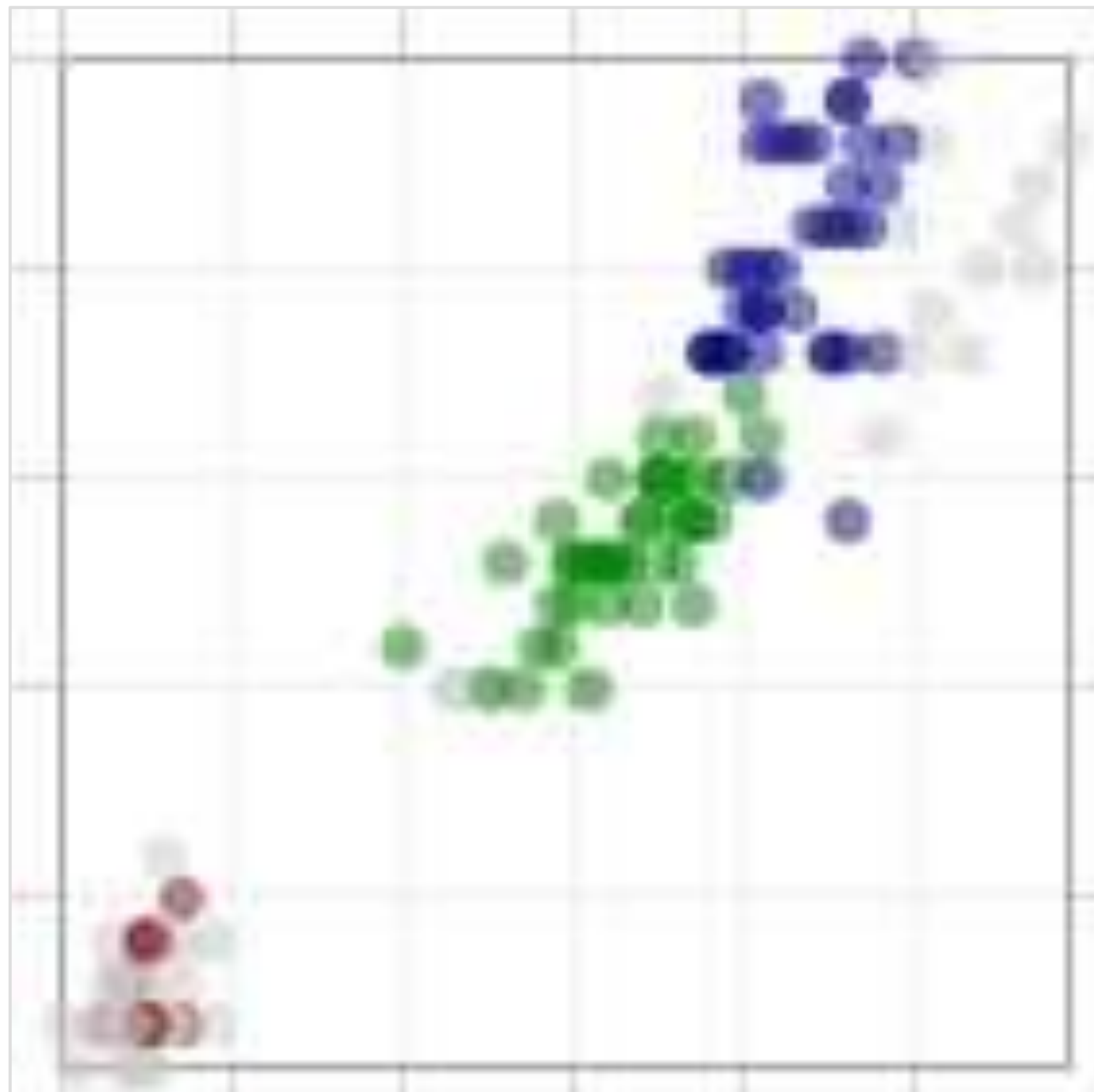
## Дано:

- 3 вида цветков ириса
- 4 параметра: 2 длины и 2 ширины листа
- по 50 наборов значений на каждый вид

## Найти:

- тип цветка по 4 параметрам

## ЦВЕТКИ ИРИСА: ДАННЫЕ



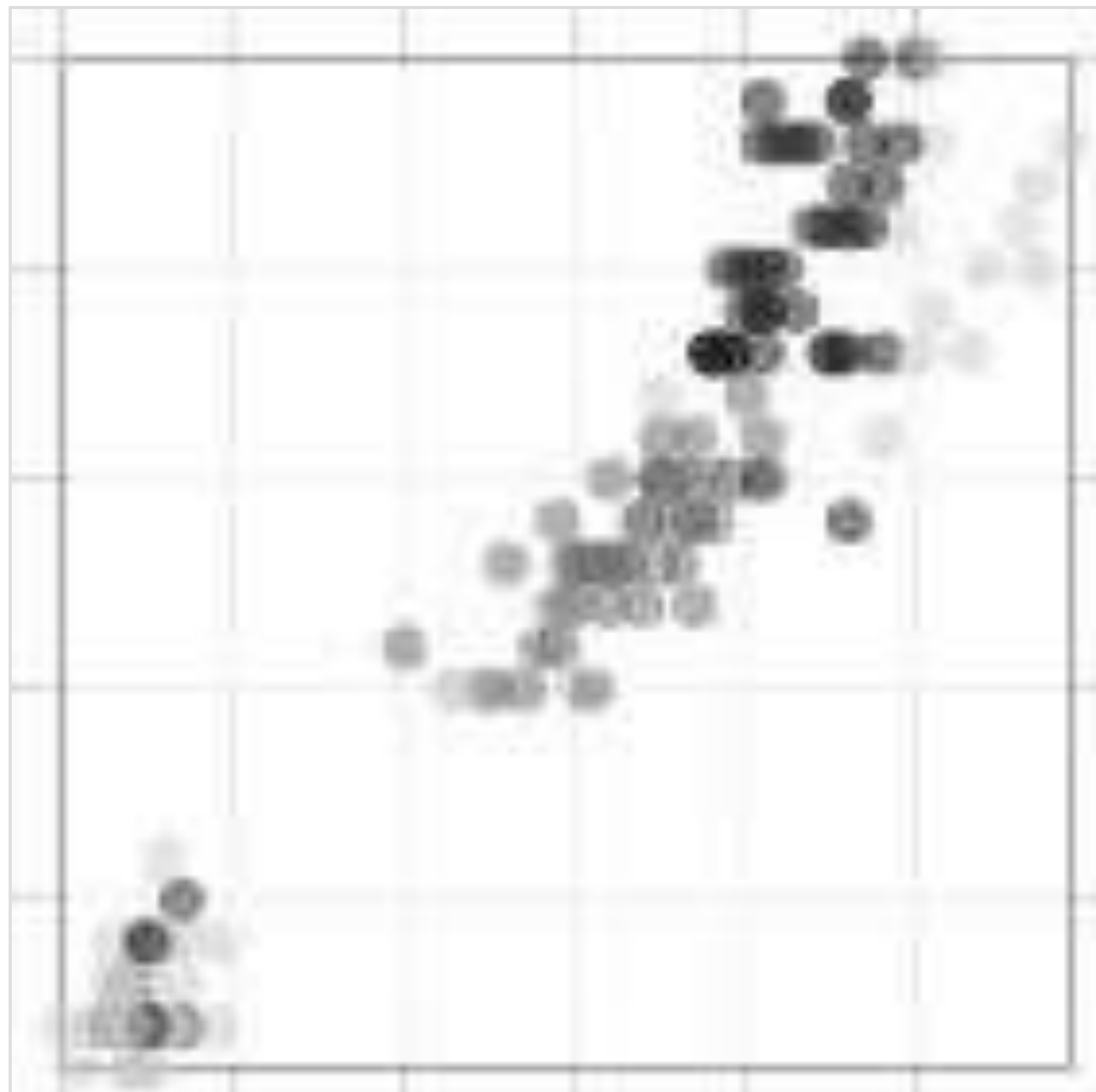
### Дано:

- 3 вида цветков ириса
- 4 параметра: 2 длины и 2 ширины листа
- по 50 наборов значений на каждый вид

### Найти:

- тип цветка по 4 параметрам

## ЦВЕТКИ ИРИСА: ДАННЫЕ



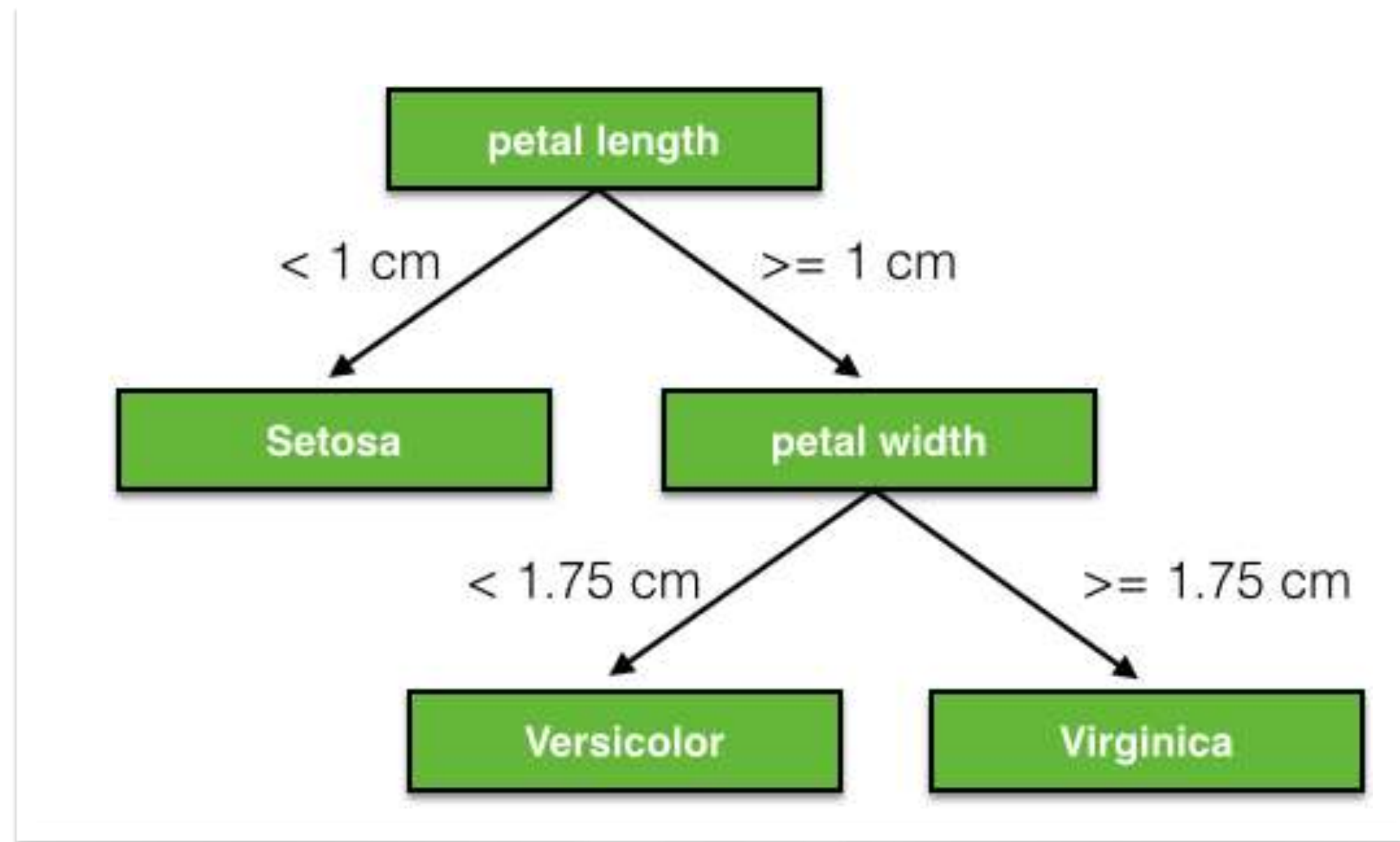
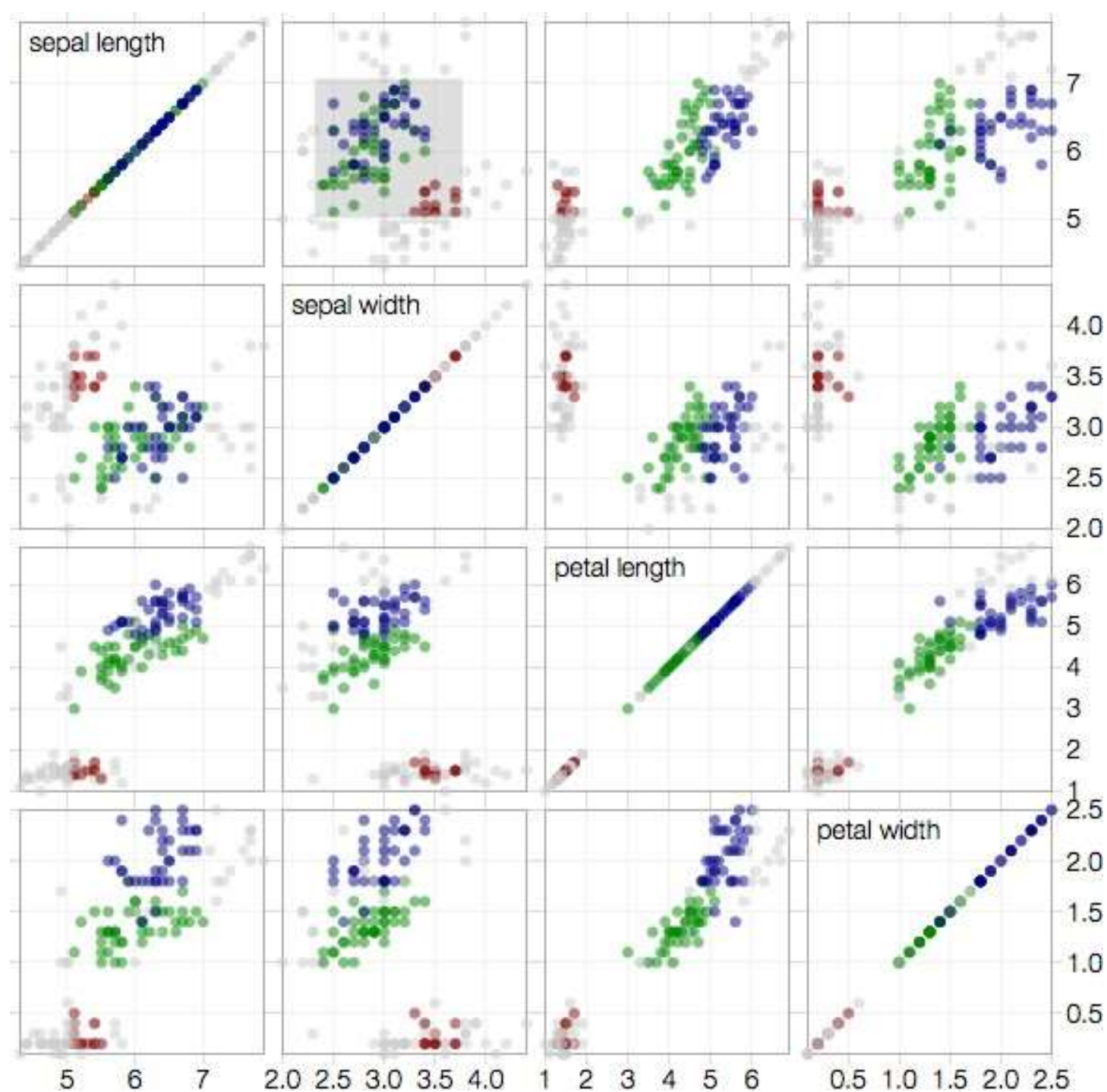
### Дано:

- 3 вида цветков ириса
- 4 параметра: 2 длины и 2 ширины листа
- по 50 наборов значений на каждый вид

### Найти:

- тип цветка по 4 параметрам

# ЦВЕТКИ ИРИСА: РЕШАЮЩЕЕ ДЕРЕВО

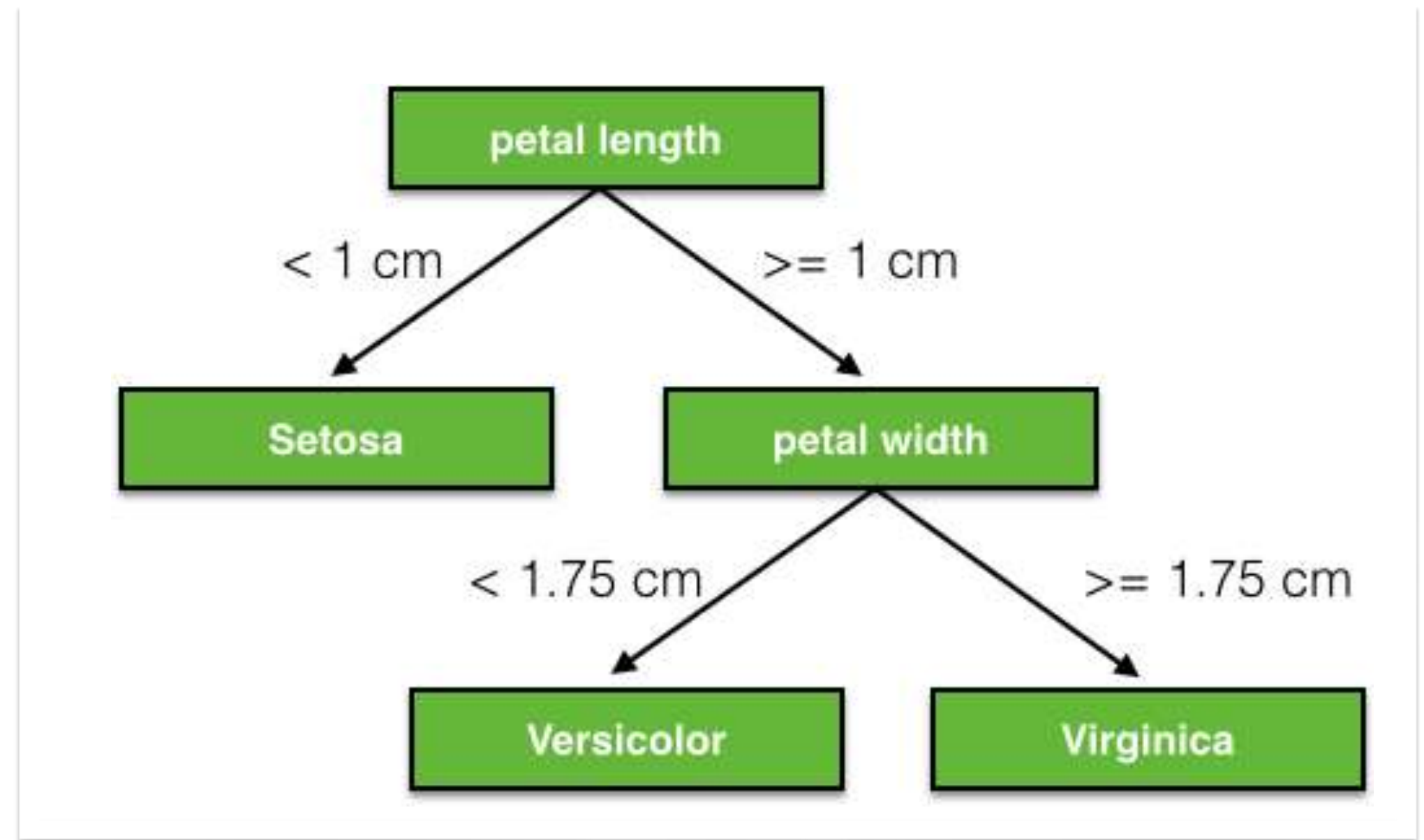




# КАК ПОСТРОИТЬ ДЕРЕВО?

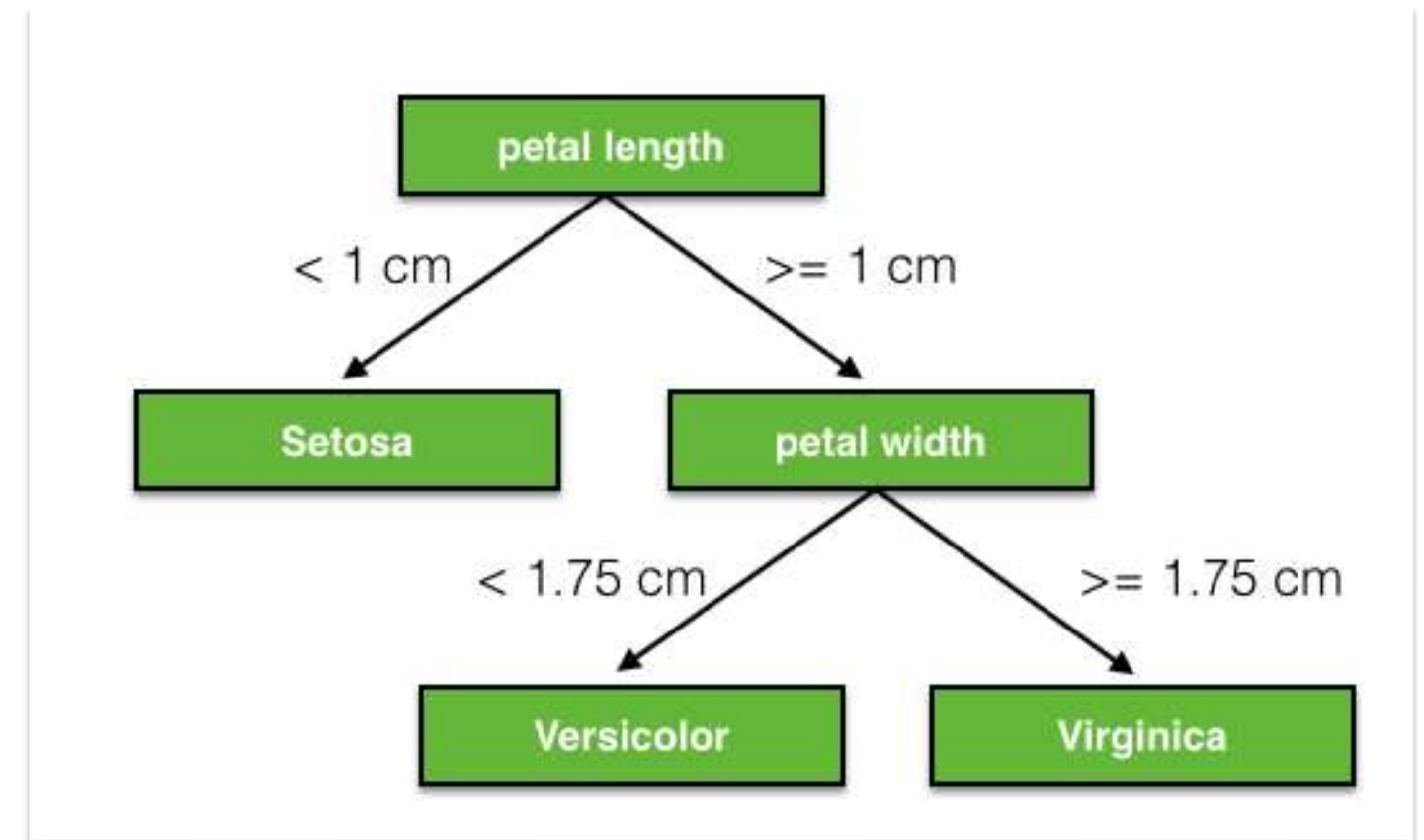
## Определить:

- вид правила разбиения
- критерий информативности разбиения
- критерий останова
- метод стрижки
- обработка пропусков



## ВИД ПРАВИЛА ДЛЯ РАЗБИЕНИЯ

- **одномерное:**  
сравнивается значение одной фичи вектора  $x$
- **линейное:**  
сравнивается линейная комбинация фичей  $x$
- **метрическое:**  
расстояние до точки признакового пространства



*здесь используется одномерный предикат: сравнение идёт лишь по одной фиче из вектора признаков*

# ФУНКЦИОНАЛ КАЧЕСТВА РАЗБИЕНИЯ

## **Идея:**

- взять признак
- отсортировать его по возрастанию
- в зависимости от целевой переменной установить порог разделения выборки на две, максимально снижая численно выражаемый разброс внутри каждой из 2 групп
- подобрать лучшее с точки зрения улучшения разбиение

**Вопрос:** а как измерить улучшение?

## ИЗМЕРЕНИЕ ПОЭТАПНОГО УЛУЧШЕНИЯ

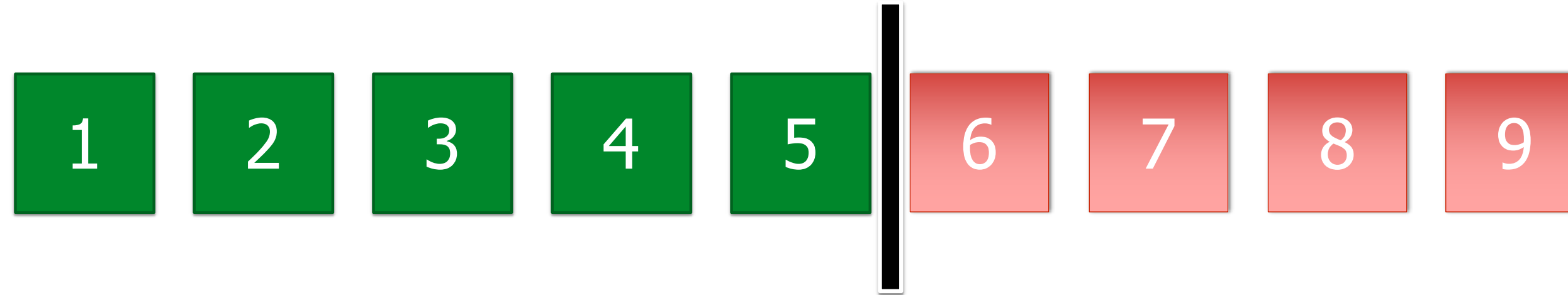


Есть 1 группа, в ней 2 класса

Пусть  $H(R)$  - «критерии информативности» группы,  
больше разнообразия - больше  $H(R)$  - хуже для классификатора

Будем измерять улучшение разбиения по функционалу вида:  $IG(R) = H(R) - q_{\text{left}} * H(R_{\text{left}}) - q_{\text{right}} * H(R_{\text{right}})$ , где  $q_{\text{left}}$  и  $q_{\text{right}}$  - доли объектов, попавших в левый или правый класс соответственно

## ИЗМЕРЕНИЕ ПОЭТАПНОГО УЛУЧШЕНИЯ



$$IG(R) = H(R) - q_{\text{left}} * H(R_{\text{left}}) - q_{\text{right}} * H(R_{\text{right}})$$

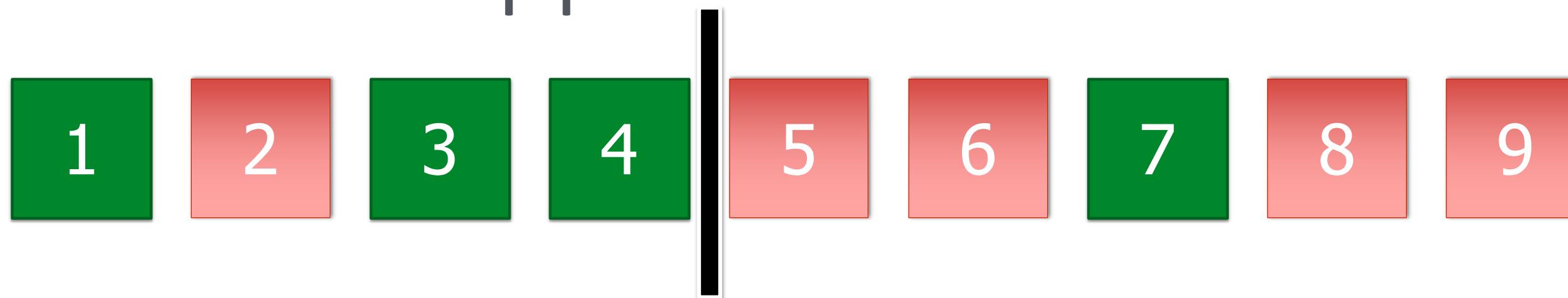
$$H(R) = x > 0$$

$$H(R_{\text{left}}) = 0$$

$$H(R_{\text{right}}) = 0$$

$$IG(R) = x - 5/9 * 0 - 4/9 * 0 = x > 0$$

## КРИТЕРИЙ ДЖИНИ



$$H(R) = \sum_{k=1}^K p_k (1 - p_k)$$

$K$  - количество классов  
 $p_k$  - доля класса в выборке

$$IG(R) = H(R) - q_{\text{left}} * H(R_{\text{left}}) - q_{\text{right}} * H(R_{\text{right}})$$

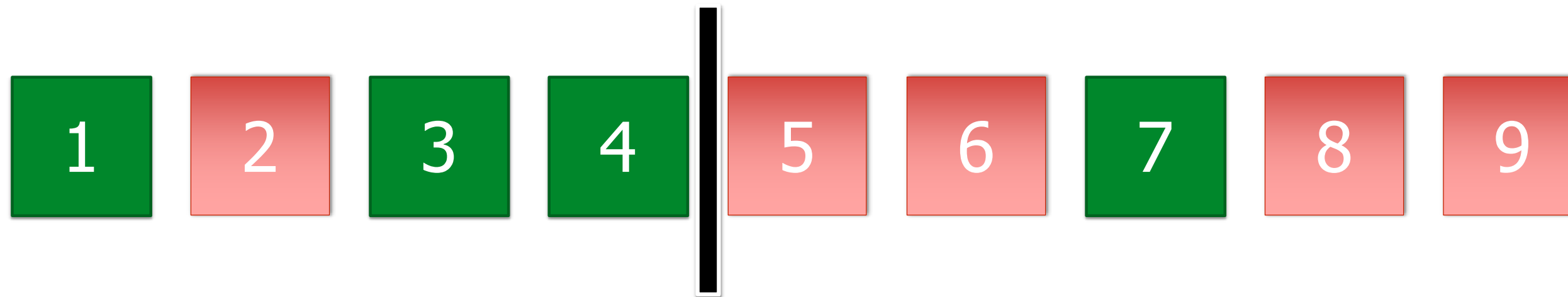
$$H(R) = 4/9 * (1 - 4/9) + 5/9 * (1 - 5/9) = 0.494$$

$$H(R_{\text{left}}) = 3/4 * (1 - 3/4) + 1/4 * (1 - 1/4) = 0.375$$

$$H(R_{\text{right}}) = 1/5 * (1 - 1/5) + 4/5 * (1 - 4/5) = 0.32$$

$$IG(R) = 0.494 - 4/9 * 0.375 - 5/9 * 0.32 = \mathbf{0.15}$$

## ЭНТРОПИЙНЫЙ КРИТЕРИЙ



$$H(R) = - \sum_{k=1}^K p_k \log p_k$$

$K$  - количество классов  
 $p_k$  - доля класса в выборке

$$IG(R) = H(R) - q_{\text{left}} * H(R_{\text{left}}) - q_{\text{right}} * H(R_{\text{right}})$$

$$H(R) = -4/9 * \log_2(4/9) - 5/9 * \log_2(5/9) = 0.991$$

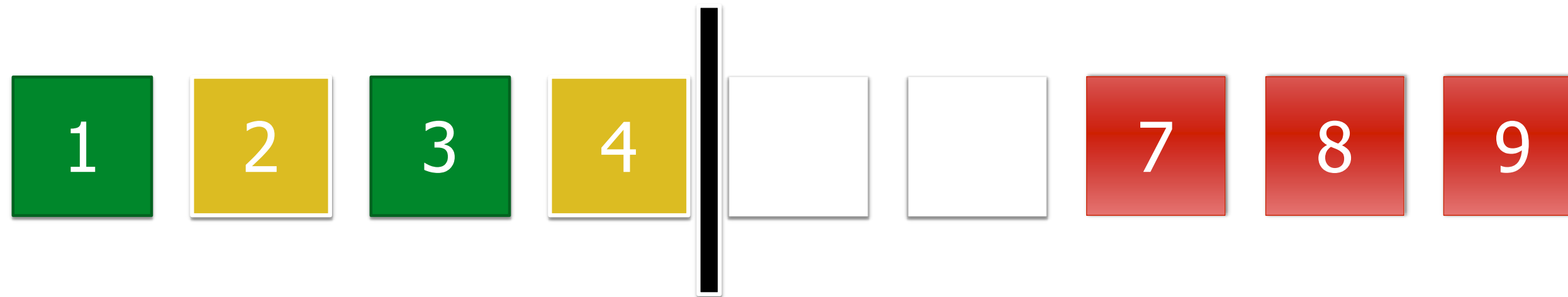
$$H(R_{\text{left}}) = -3/4 * \log_2(3/4) - 1/4 * \log_2(1/4) = 0.811$$

$$H(R_{\text{right}}) = -1/5 * \log_2(1/5) - 4/5 * \log_2(4/5) = 0.722$$

$$IG(R) = 0.991 - 4/9 * 0.811 - 5/9 * 0.722 = \mathbf{0.229}$$

## КРИТЕРИЙ ДЖИНИ

$$IG(R) = ?$$





## КРИТЕРИИ ОСТАНОВА

- Останов, когда в каждом листе объекты только одного класса
- Ограничение  $\max$  глубины дерева
- Ограничение  $\min$  число объектов в листьях
- Требование улучшения функционала качества при дроблении не менее, чем  $x$  или на  $x\%$

## ПРОБЛЕМА ПРОПУСКОВ

- Выкинуть объекты с пропусками из обучающей (что на тестовой?)
- Замена на значения вне средние, медианные..
- Заменить на значения вне области значений фич
- Модифицировать алгоритм построения и работы дерева: включать элементы с пропусками в обе ветки дерева, но взвешивать качество разбиения по объёму пропусков

## СТРИЖКА ДЕРЕВЬЕВ (PRUNING)

Стрижка из полностью построенного дерева убирает наименее информативные листья

Стрижка работает лучше раннего останова

Редко используется, т.к. деревья не используются самостоятельно, а в ансамблях она излишняя (там либо нужно переобучение, либо используется ограничение глубины)

В основе идея регуляризации: в функционале качестве поддеревья линейно штрафуются количество листьев

## ПОПУЛЯРНЫЕ МЕТОДЫ ПОСТРОЕНИЯ

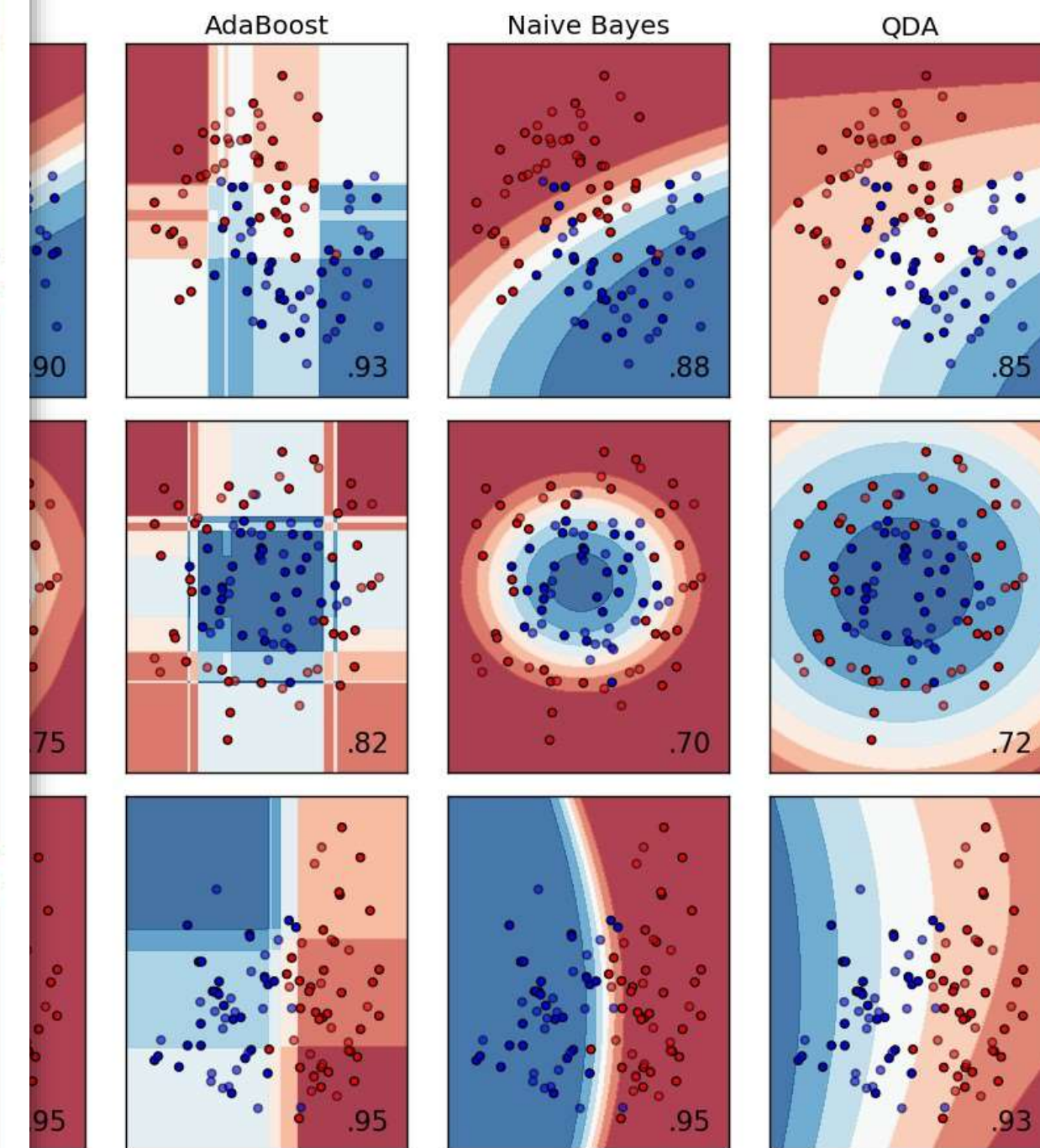
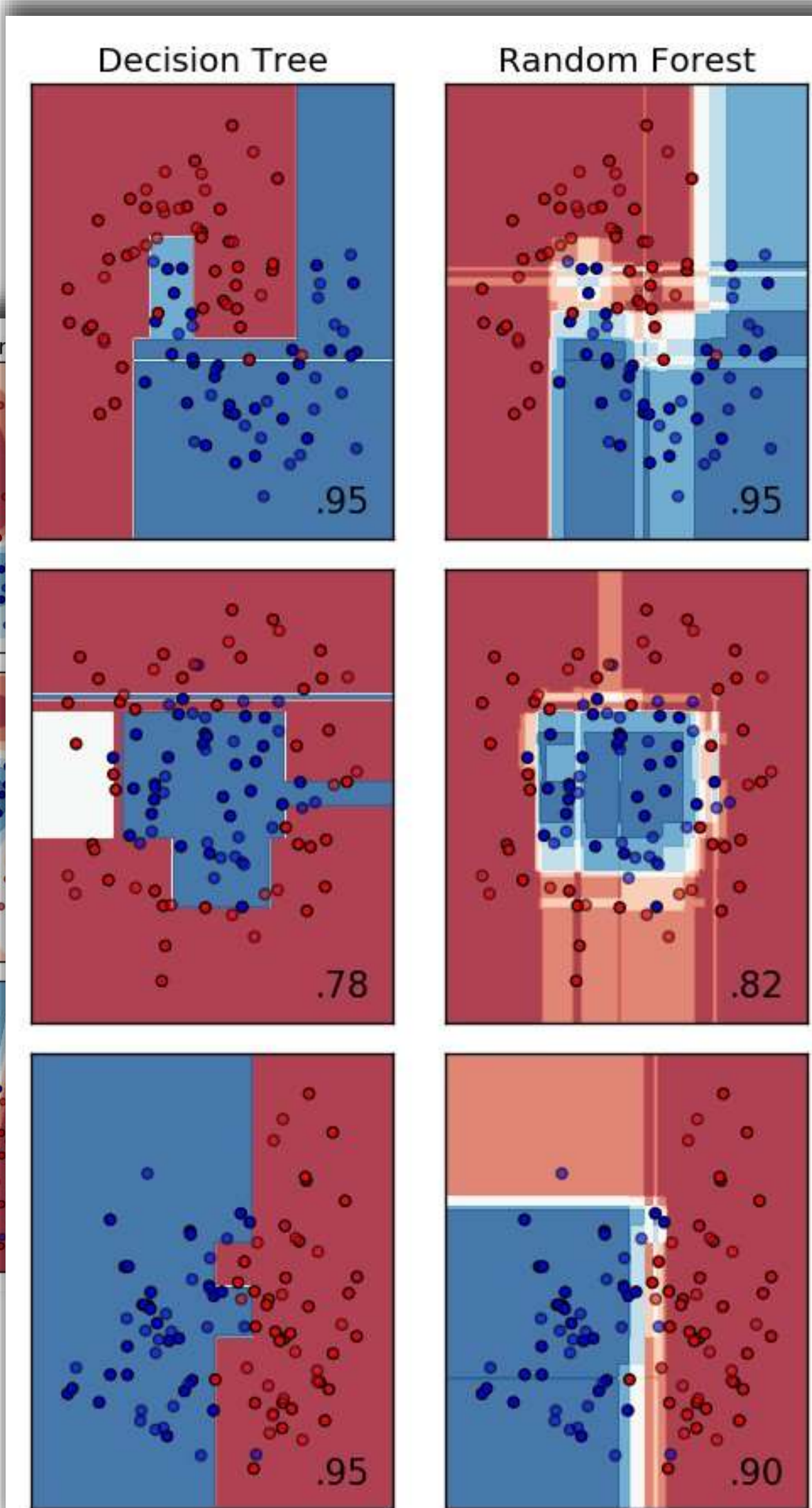
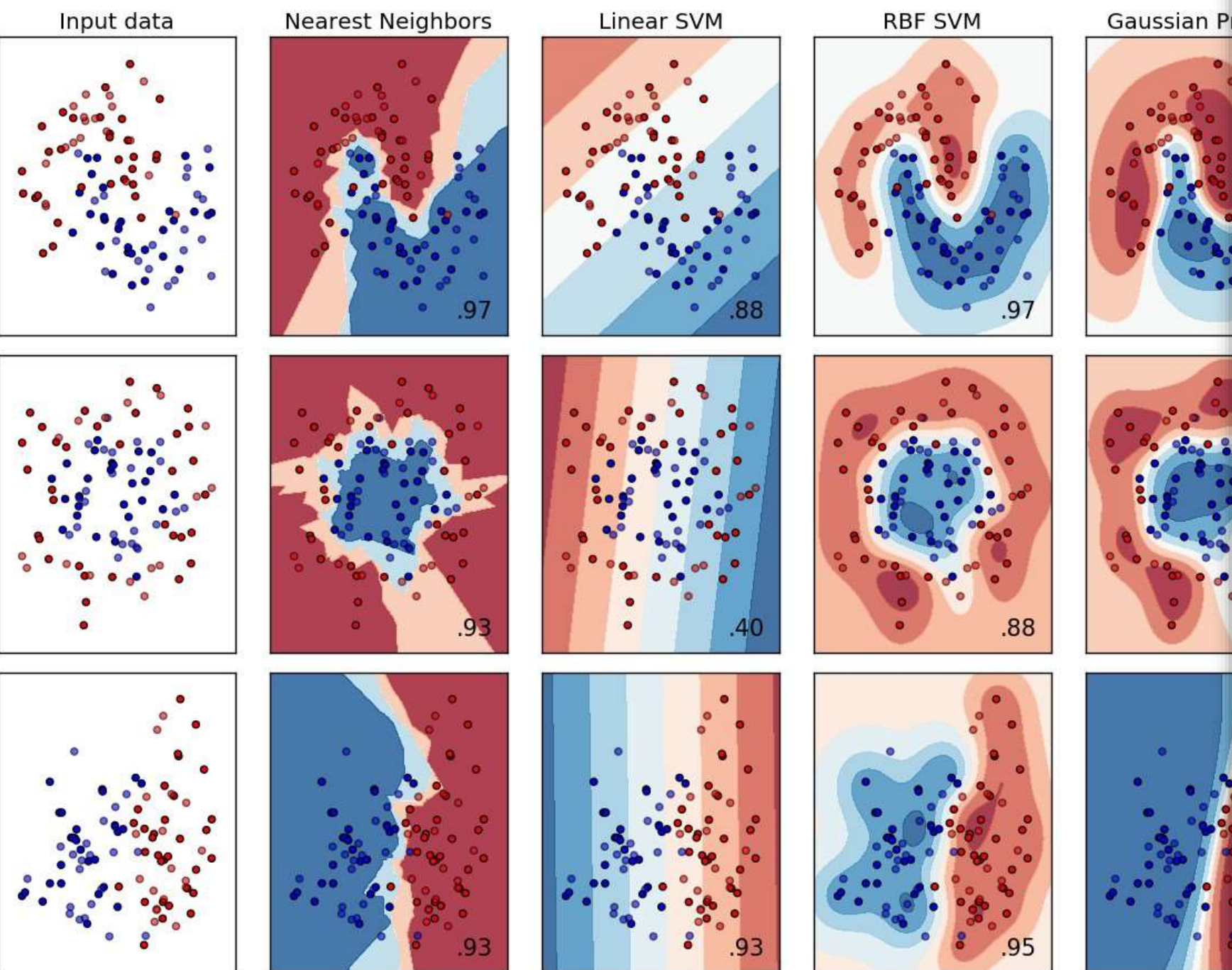
Деревья в силу дискретности не сводятся к оптимизации в аналитическом виде, поэтому все методы их построения являются эвристическими и жадными

Популярные методы отличаются ранее рассмотренными параметрами построения дерева

- ID3: энтропийный критерий, максимально жадный, требуется стрижка (1986)
- C4.5, C5.0: нормированный энтропийный критерий
- CART: критерий Джини - используется в sklearn (optimized)



# ПОСТРОЕНИЕ ДЕРЕВА РЕШЕНИЙ



\* *sklearn, сравнение классификаторов*



## РЕАЛИЗАЦИЯ В SKLEARN

### [sklearn.tree.DecisionTreeClassifier](#)

- \* criterion='gini'
- \* splitter='best'
- \* max\_depth=None
- \* min\_samples\_split=2
- \* min\_samples\_leaf=1
- \* min\_weight\_fraction\_leaf=0.0
- \* max\_features=None
- \* random\_state=None
- \* max\_leaf\_nodes=None
- \* min\_impurity\_split=1e-07
- \* class\_weight=None
- \* presort=False

#### **Основные характеристики**

- \* 12 параметров
- \* Функционал качества: Джини / энтропия
- \* Реализованы различные простые критерии останова: кол-во объектов, улучшение качества..
- \* Не реализована стрижка дерева

#### **Основные методы**

- \* fit
- \* predict, predict\_proba

# Деревья могут оценивать важность фичей



\* наличие в обращении «Mrs» / «Miss»

---

## 3. ДОСТОИНСТВА И НЕДОСТАТКИ ДЕРЕВЬЕВ РЕШЕНИЙ



## ДОСТОИНСТВА

- Легко интерпретировать, визуализировать, «белый ящик»
- Простота подготовки данных: не требуется нормализация, dummy переменные, возможны пропуски
- Скорость работы

## НЕДОСТАТКИ

- Острая проблема переобучения
- Неустойчивость
- Не учитывает нелинейные зависимости или даже простые линейные, которые идут не по осям координат (f.e., представьте дерево для классификатора вида  $y > x$ )
- Чувствителен к несбалансированным классам
- Хорошо интерполирует, плохо экстраполирует

---

# ПРАКТИЧЕСКОЕ ЗАДАНИЕ 2

## ЗАДАНИЕ №2. KAGGLE. SHELTER

Принять участие в соревновании Kaggle:

<https://www.kaggle.com/c/shelter-animal-outcomes/>

Привести изначальные признаки в числовой вид

Использовать кросс-валидацию для обучения дерева

Сделать предсказание

Засабмитить на Kaggle

---

## 4. ОЦЕНКА КАЧЕСТВА КЛАССИФИКАЦИИ

## МАТРИЦА ОШИБОК

confusion matrix	$y = 1$	$y = 0$
$a = 1$	True Positive	False Positive
$a = 0$	False Negative	True Negative

На тестовой выборке имеем:

- \*  $y$  - вектор истинных значений
- \*  $a$  - вектор предсказаний классификатора

Будем раскладывать все пары (предсказание, истина) по ячейкам матрицы ошибок

# ACCURACY

confusion matrix	$y = 1$	$y = 0$
$a = 1$	0	0
$a = 0$	2	998

Accuracy, Доля верных ответов  
(в просторечии точность, но не путать с  
точностью из ML!)

Простая метрика, но абсолютно непоказательна  
в задачах с несбалансированными классами

**Пример:**

определение качества теста на рак. Тест  
постоянно предсказывает отсутствие рака. Доля  
верных ответов: 99.8%

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

# PRECISION

confusion matrix	$y = 1$	$y = 0$
$a = 1$	170	100
$a = 0$	30	700

## Precision, Точность

отсутствие ложных срабатываний

### Пример:

правильное распознавание намерения  
пользователя: лучше переспросить  
пользователя, чем сделать не то, что нужно

Точность =  $170 / (170 + 100) = 0.629$

$$precision = \frac{TP}{TP + FP}$$



# RECALL

confusion matrix	$y = 1$	$y = 0$
$a = 1$	170	100
$a = 0$	30	700

Recall, Полнота  
отсутствие ложных пропусков

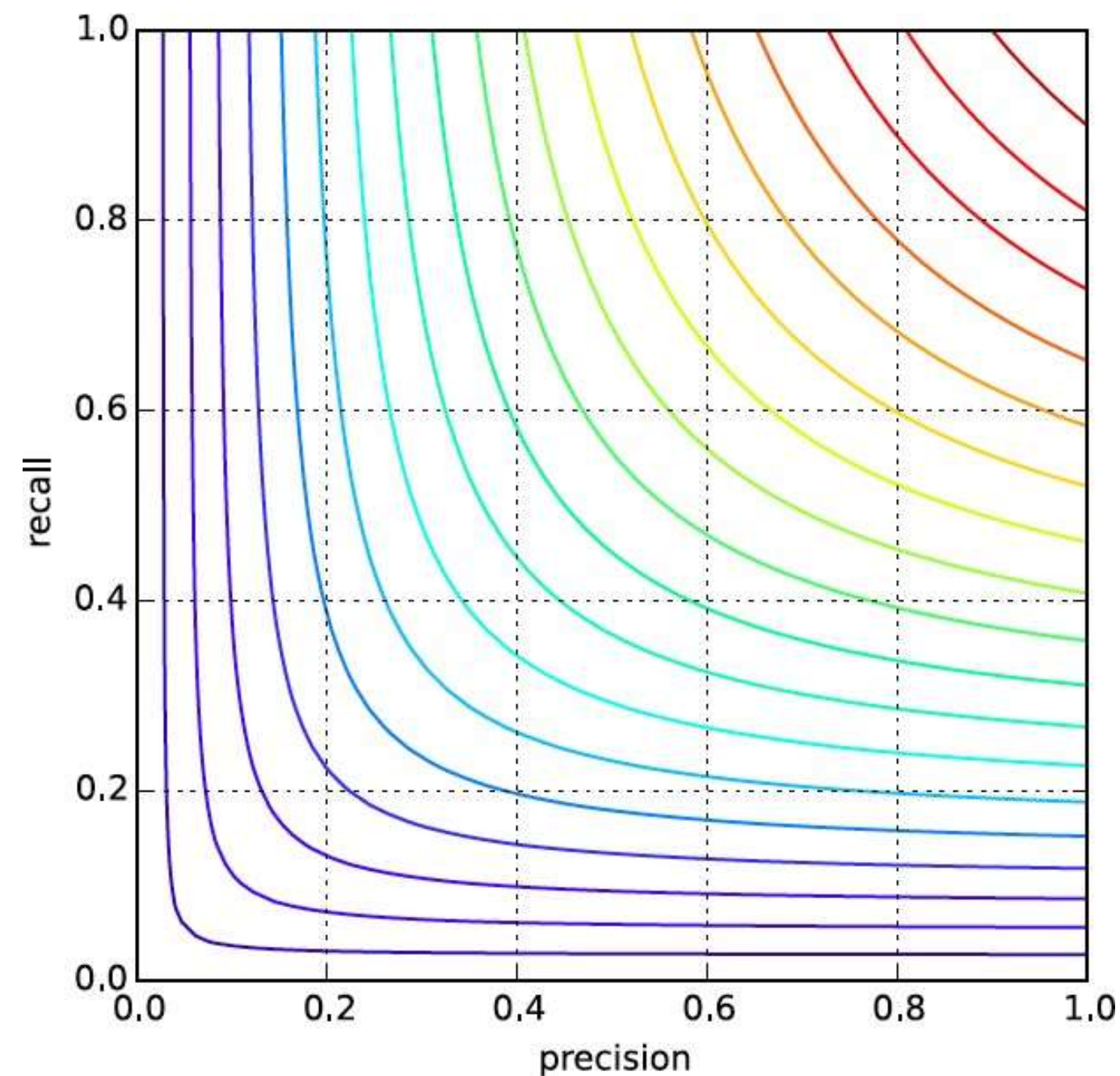
**Пример:**

определение мошеннических действий в банке:  
лучше найти лишнее и проверить, чем не найти

Точность =  $170 / (170+30) = 0.85$

$$recall = \frac{TP}{TP + FN}$$

# F1 - МЕРА



## F1-мера

комбинация точности и полноты в одну метрику

### Пример:

правильное распознавание намерения пользователя. Насколько мы уверены в том, что правильно поняли? Надо ли уточнить?

$$F1 = 2 * 0.629 * 0.85 / (0.629 + 0.85) = 0.723$$

$$F = \frac{2 * \text{precision} * \text{recall}}{\text{precision} + \text{recall}}$$

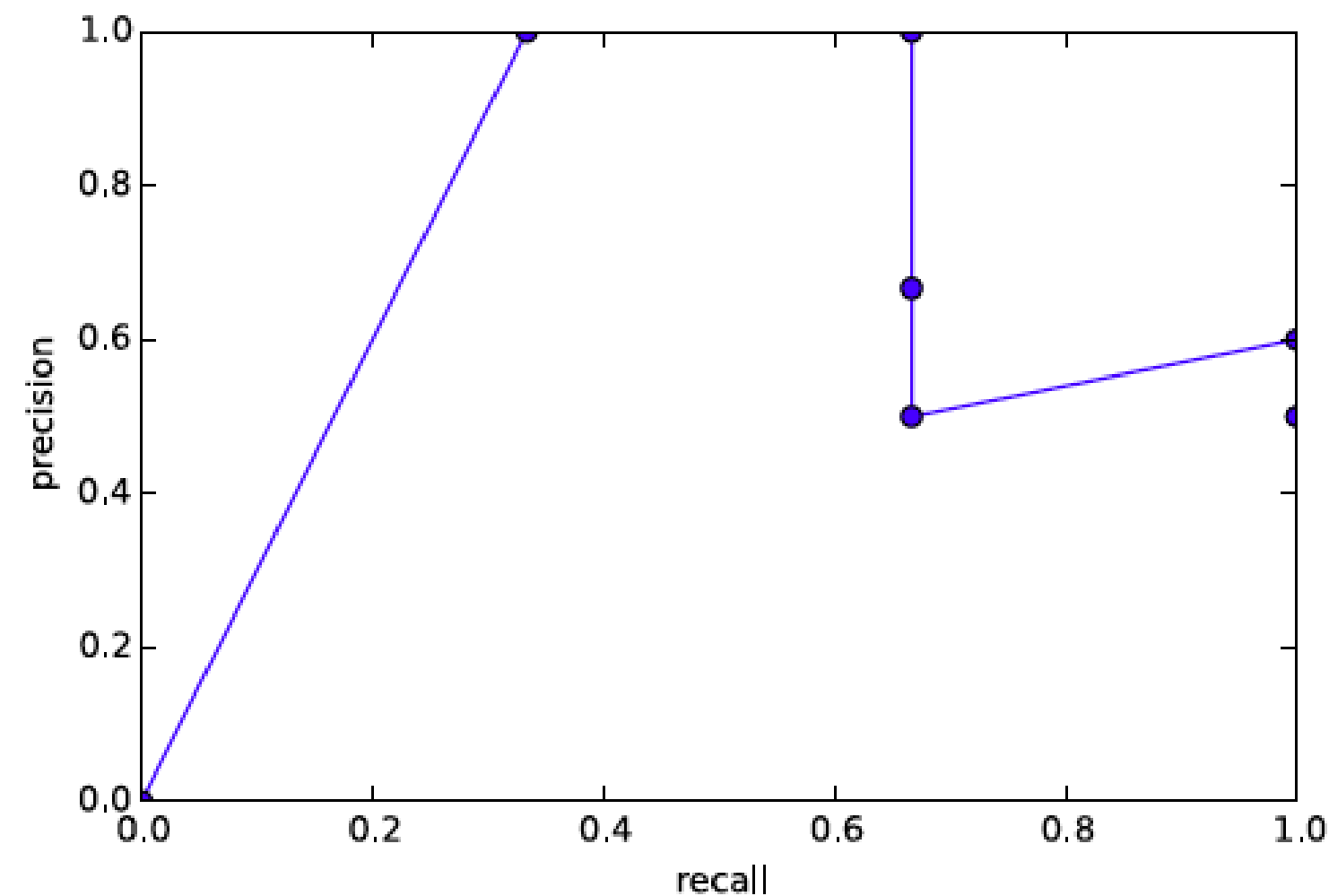
## GOING DEEPER

Классификаторы обычно работают в 2 этапа:

- \* оценка вероятности принадлежности к классам  $a(x)$
- \* выбор порога отсечения, при котором идёт распределение в тот или иной класс

Это 2 отдельные задачи, после получения оценки вероятности можно отсортировать объекты и в различные периоды времени использовать разные пороги

# AUC-PRC



\* из материалов к курсу Воронцова на Coursera

## Precision - Recall (PR curve)

мера качества разделения на классы

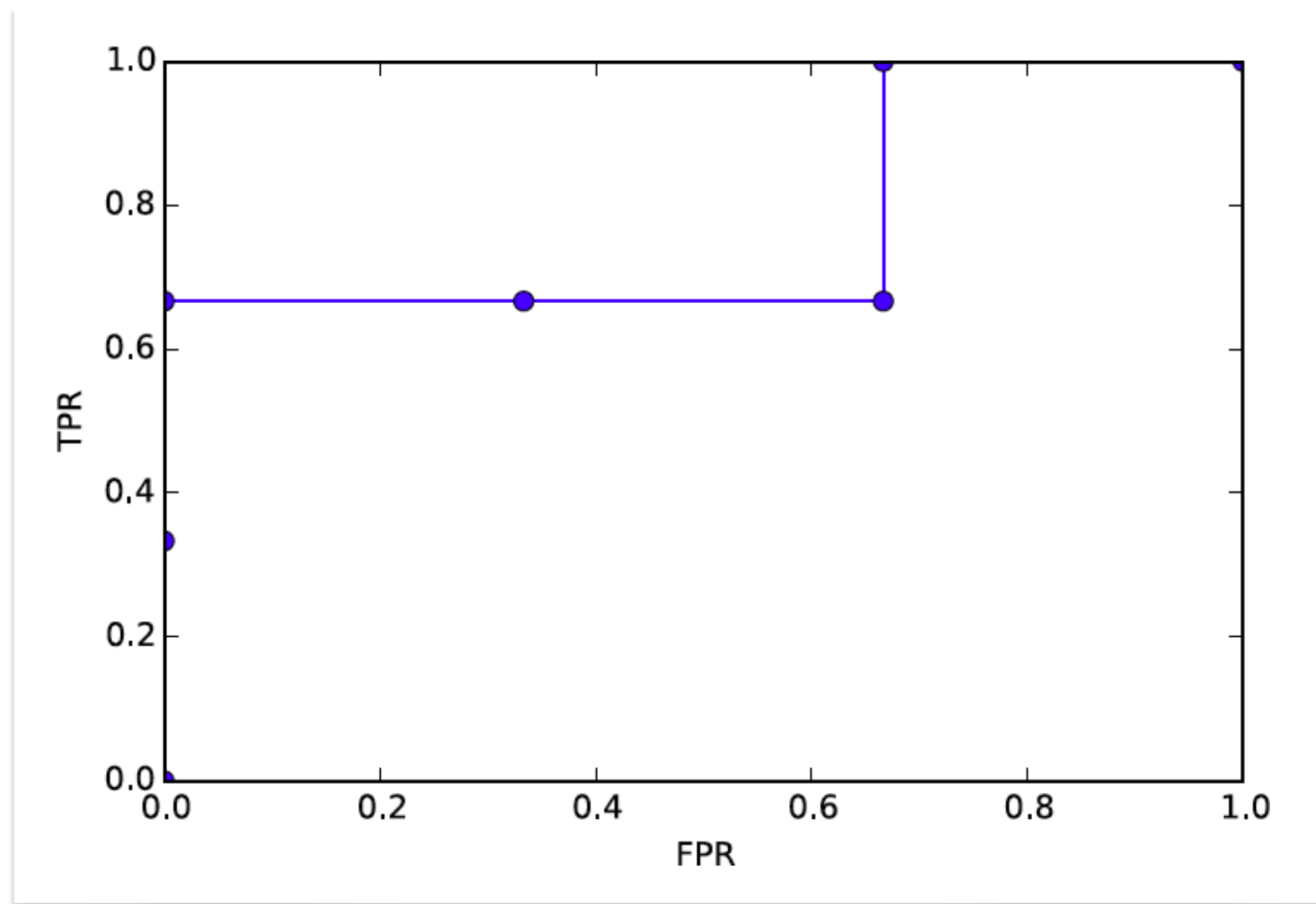
### Построение:

1. Считаем вероятности принадлежности классу 1
2. Сортируем объекты по вероятности
3. Для каждого порога отсечения между объектами считаем precision и recall и последовательно наносим на график
4. Считаем площадь под кривой

**AUC - PRC** (area under curve: precision recall curve)

итоговая метрика. Больше - лучше

# AUC-ROC



\* из материалов к курсу Воронцова на Coursera

Receiver Operating Characteristic curve  
мера качества разделения на классы

Построение:  
аналогично PR-кривой, но наносятся точки

$$\text{FPR} = \text{FP} / (\text{FP} + \text{TN})$$

1-FPR - *специфичность алгоритма*

$$\text{TPR} = \text{TP} / (\text{TP} + \text{FN})$$

TPR - *чувствительность алгоритма*

## AUC - ROC

итоговая метрика. Больше - лучше

## AUC-PRC VS AUC-ROC

AUC-PRC лучше использовать в несбалансированных классах

AUC-ROC можно использовать, когда алгоритм будет оцениваться на одних данных, но с разным соотношением классов

# МНОГОКЛАССОВАЯ КЛАССИФИКАЦИЯ

Задача:  $a(x) \in \{1, \dots, K\}$

Сводится к  $K$  задачам отделения класса  $N$  от остальных

Как усреднить качество  $K$  задач?

# МНОГОКЛАССОВАЯ КЛАССИФИКАЦИЯ

## **micro-averaging:**

- \* вычислим confusion matrix для каждой задачи
- \* усредним по задачам
- \* вычислим итоговую метрику

*классы делают вклад, пропорциональный размеру*

## **macro-averaging:**

- \* вычислим итоговую метрику для каждой задачи
- \* усредним по задачам

*все классы делают равный вклад*



# РЕАЛИЗАЦИЯ В SKLEARN

## sklearn.metrics

Some of these are restricted to the binary classification case:

<code>matthews_corrcoef</code> ( <code>y_true</code> , <code>y_pred</code> [, ...])	Compute the Matthews correlation coefficient (MCC) for binary classes
<code>precision_recall_curve</code> ( <code>y_true</code> , <code>probas_pred</code> )	Compute precision-recall pairs for different probability thresholds
<code>roc_curve</code> ( <code>y_true</code> , <code>y_score</code> [, <code>pos_label</code> , ...])	Compute Receiver operating characteristic (ROC)

Others also work in the multiclass case:

<code>cohen_kappa_score</code> ( <code>y1</code> , <code>y2</code> [, <code>labels</code> , <code>weights</code> ])	Cohen's kappa: a statistic that measures inter-annotator agreement.
<code>confusion_matrix</code> ( <code>y_true</code> , <code>y_pred</code> [, <code>labels</code> , ...])	Compute confusion matrix to evaluate the accuracy of a classification
<code>hinge_loss</code> ( <code>y_true</code> , <code>pred_decision</code> [, <code>labels</code> , ...])	Average hinge loss (non-regularized)

Some also work in the multilabel case:

<code>accuracy_score</code> ( <code>y_true</code> , <code>y_pred</code> [, <code>normalize</code> , ...])	Accuracy classification score.
<code>classification_report</code> ( <code>y_true</code> , <code>y_pred</code> [, ...])	Build a text report showing the main classification metrics
<code>f1_score</code> ( <code>y_true</code> , <code>y_pred</code> [, <code>labels</code> , ...])	Compute the F1 score, also known as balanced F-score or F-measure
<code>fbeta_score</code> ( <code>y_true</code> , <code>y_pred</code> , <code>beta</code> [, <code>labels</code> , ...])	Compute the F-beta score
<code>hamming_loss</code> ( <code>y_true</code> , <code>y_pred</code> [, <code>labels</code> , ...])	Compute the average Hamming loss.
<code>jaccard_similarity_score</code> ( <code>y_true</code> , <code>y_pred</code> [, ...])	Jaccard similarity coefficient score
<code>log_loss</code> ( <code>y_true</code> , <code>y_pred</code> [, <code>eps</code> , <code>normalize</code> , ...])	Log loss, aka logistic loss or cross-entropy loss.
<code>precision_recall_fscore_support</code> ( <code>y_true</code> , <code>y_pred</code> )	Compute precision, recall, F-measure and support for each class
<code>precision_score</code> ( <code>y_true</code> , <code>y_pred</code> [, <code>labels</code> , ...])	Compute the precision
<code>recall_score</code> ( <code>y_true</code> , <code>y_pred</code> [, <code>labels</code> , ...])	Compute the recall
<code>zero_one_loss</code> ( <code>y_true</code> , <code>y_pred</code> [, <code>normalize</code> , ...])	Zero-one classification loss.

And some work with binary and multilabel (but not multiclass) problems:

<code>average_precision_score</code> ( <code>y_true</code> , <code>y_score</code> [, ...])	Compute average precision (AP) from prediction scores
<code>roc_auc_score</code> ( <code>y_true</code> , <code>y_score</code> [, <code>average</code> , ...])	Compute Area Under the Curve (AUC) from prediction scores

## Основные характеристики

- \* 19 функций
- \* Схожий интерфейс: функции от `y`, `y_pred`

## Пример использования `f1_score`

```
>>> from sklearn.metrics import f1_score
>>> y_true = [0, 1, 2, 0, 1, 2]
>>> y_pred = [0, 2, 1, 0, 0, 1]
>>> f1_score(y_true, y_pred, average='macro')
0.26...
>>> f1_score(y_true, y_pred, average='micro')
0.33...
>>> f1_score(y_true, y_pred, average='weighted')
0.26...
>>> f1_score(y_true, y_pred, average=None)
array([ 0.8, 0. , 0. ])
```

---

# ПРАКТИЧЕСКОЕ ЗАДАНИЕ 3

## ЗАДАНИЕ №3. HANDMADE METRICS

Из данных из последней задачи выделить 2 класса: взятие в семью и эвтаназия; обучить классификатор

Для следующих метрик с нуля построить функции и сравнить с аналогами из `sklearn.metrics`:

- \* Accuracy
- \* Precision
- \* Recall
- \* F1

---

# ПРАКТИЧЕСКОЕ ЗАДАНИЕ 4

## ЗАДАНИЕ №4. NEWS CLASSIFICATION

Используя подготовленный датасет из архива Ведомостей самостоятельно поставить задачу классификации темы статьи:

***это финансы или это политика?***

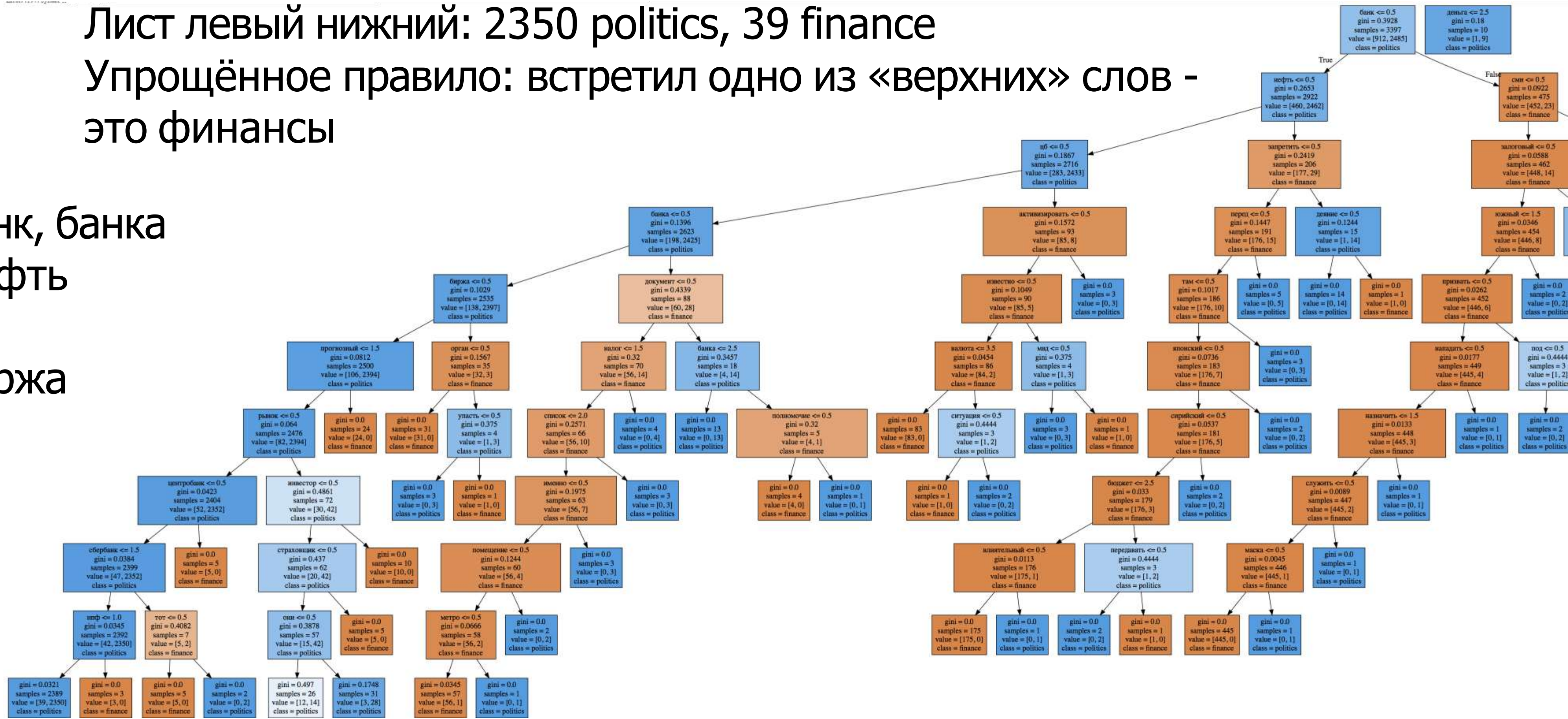
- \* выделить из датасета только финансы и политику
- \* превратить тексты в наборы слов
- \* разбить на обучающую и тестовую выборки
- \* обучить дерево, не забыв про кросс-валидацию
- \* оценить качество по тестовой
- \* вывести визуальное отображение дерева



\_\_\_\_\_

**Y**

■ ■ ■



---

ЧТО МЫ СЕГОДНЯ УЗНАЛИ

1. Деревья решений, объединённые в «лес», составляют одни из наиболее сильных алгоритмов. По одиночке же они являются слабыми, зато очень легко интерпретируемыми и визуализируемыми алгоритмами
2. Деревья позволяют оценивать важность признаков
3. Метрик качества много, они разные по смыслу, для своих задач надо выбирать подходящую



---

ПОЛЕЗНЫЕ МАТЕРИАЛЫ

1. [Документация sklearn по деревьям](#)
2. [Open Data Science, habrahabr: Классификация, дерево решений и метод ближайших соседей](#)
3. Лекция Евгения Соколова на ФКН ВШЭ по деревьям  
[Конспект; іруnb тетрадка](#)
4. [Метрики sklearn](#)
5. [Метрики kaggle](#)
6. [Объяснение метрик на курсе Coursera от Соколова & Воронцова](#)



НЕТОЛОГИЯ  
групп

Спасибо за внимание!

Артур Сапрыкин