

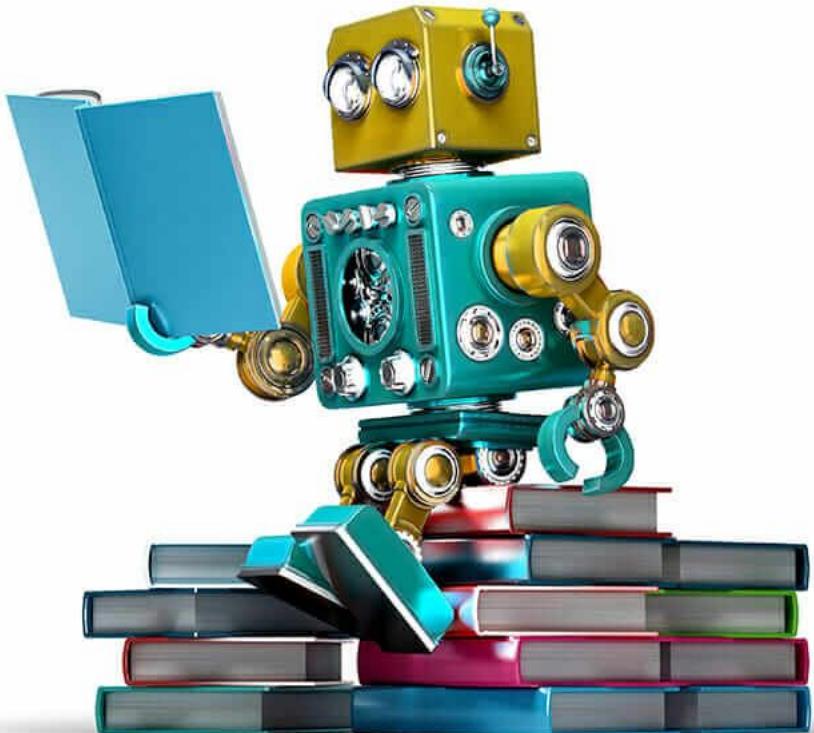
# Lecture 3

---

Bulat Ibragimov  
*Yandex Research*

# План

1. Метрики качества
2. Naive Bayes
3. Логические алгоритмы
4. Композиции алгоритмов
  - a. Bootstrap, Bagging
  - b. Blending
  - c. Stacking
  - d. Random Forest
  - e. Gradient Boosting



# Metrics

---

100  
010

# Confusion matrix

		Actual Value (as confirmed by experiment)	
		positives	negatives
Predicted Value (predicted by the test)	positives	<b>TP</b> True Positive	<b>FP</b> False Positive
	negatives	<b>FN</b> False Negative	<b>TN</b> True Negative

`sklearn.metrics.confusion_matrix`

# Accuracy

Интуитивная, понятная и почти неиспользуемая, так как абсолютно бесполезна для работы с несбалансированными классами

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

`sklearn.metrics.accuracy_score`

# Пример: spam prediction

<https://habr.com/company/ods/blog/328372/>

Допустим, мы хотим оценить работу спам-фильтра почты. У нас есть 100 не-спам писем, 90 из которых наш классификатор определил верно (True Negative = 90, False Positive = 10), и 10 спам-писем, 5 из которых классификатор также определил верно (True Positive = 5, False Negative = 5).

Тогда accuracy:

$$accuracy = \frac{5 + 90}{5 + 90 + 10 + 5} = 86,4$$

Однако если мы просто будем предсказывать все письма как не-спам, то получим более высокую accuracy:

$$accuracy = \frac{0 + 100}{0 + 100 + 0 + 10} = 90,9$$

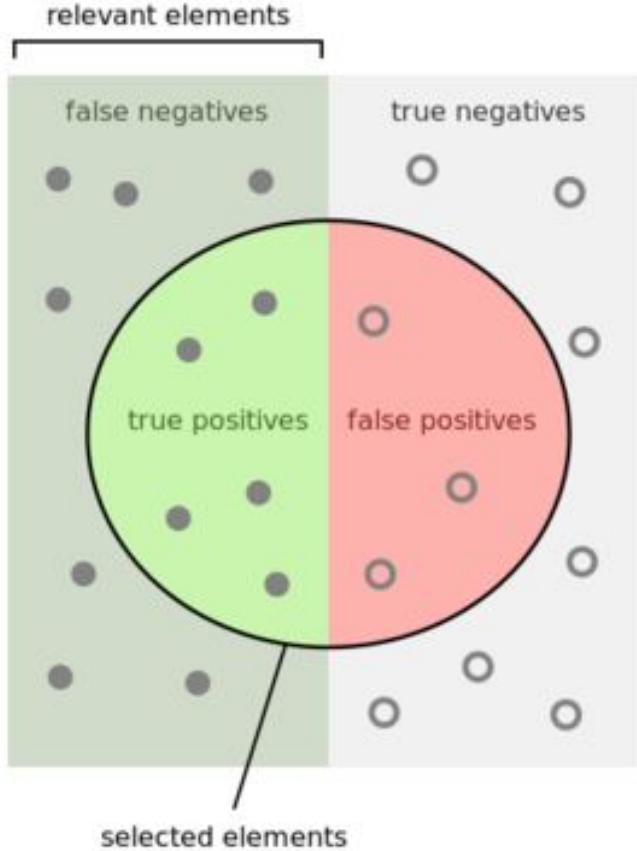
# Precision, Recall, F-мера

Для оценки качества работы алгоритма на каждом из классов по отдельности введем метрики precision (точность) и recall (полнота).

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

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

Precision можно интерпретировать как долю объектов, названных классификатором положительными и при этом действительно являющимися положительными, а recall показывает, какую долю объектов положительного класса из всех объектов положительного класса нашел алгоритм.



How many selected items are relevant?

$$\text{Precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

How many relevant items are selected?

$$\text{Recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

`sklearn.metrics.precision_score`

`sklearn.metrics.recall_score`

# F-мера

Существует несколько различных способов объединить precision и recall в агрегированный критерий качества. F-мера (в общем случае  $F_\beta$ ) — среднее гармоническое precision и recall :

$$F_\beta = (1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}}$$

при  $\beta = 0$  получаем точность

при  $\beta = 1$  — непараметрическую F-меру, (сбалансированная)

при  $\beta = \infty$  — полноту.

`sklearn.metrics.f1_score`

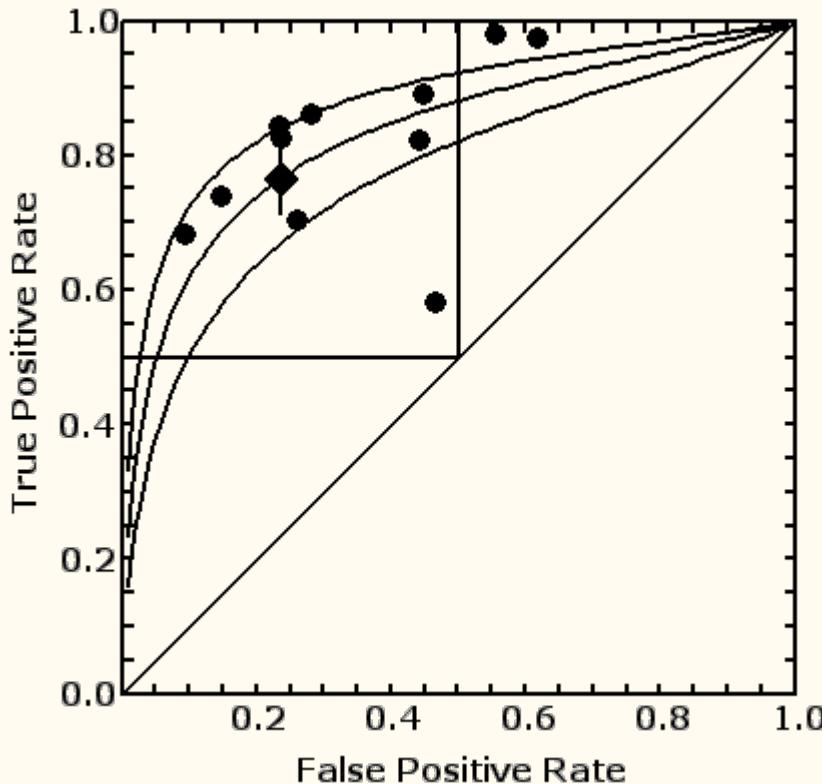
# ROC-AUC

При конвертации вещественного ответа алгоритма в бинарную метку, мы должны выбрать какой-либо порог, при котором 0 становится 1. Естественным и близким кажется порог, равный 0.5, но он не всегда оказывается оптимальным, например, при отсутствии баланса классов.

Одним из способов оценить модель в целом, не привязываясь к конкретному порогу, является AUC-ROC — площадь (Area Under Curve) под кривой ошибок (Receiver Operating Characteristic curve ).

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

$$FPR = \frac{FP}{FP + TN}$$



# ROC-AUC

```
from sklearn import metrics
```

```
fpr, tpr, thresholds = metrics.roc_curve(y, scores, pos_label=2)
```

```
y_true = [0, 2, 2, 1, 1, 0]
```



```
y_true = [0, 1, 1, 0, 0, 0]
```

```
y_pred = [  
    [0.5, 0.4, 0.1],  
    [0.2, 0.3, 0.5],  
    [0.1, 0.1, 0.8],  
    [0.2, 0.4, 0.4],  
    [0.4, 0.2, 0.4],  
    [0.6, 0.1, 0.3],  
]
```



```
y_pred = [  
    0.1,  
    0.5,  
    0.8,  
    0.4,  
    0.4,  
    0.3  
]
```

# ROC-AUC

```
from sklearn import metrics
```

```
fpr, tpr, thresholds = metrics.roc_curve(y, scores, pos_label=2)
```

thresholds -- массив порогов для вычисления tpr и fpr

```
y_pred = [  
    [0.5, 0.4, 0.1],  
    [0.2, 0.3, 0.5],  
    [0.1, 0.1, 0.8],  
    [0.2, 0.4, 0.4],  
    [0.4, 0.2, 0.4],  
    [0.6, 0.1, 0.3],  
]
```



```
y_pred = [  
    0.1,  
    0.5,  
    0.8,  
    0.4,  
    0.4,  
    0.3  
]
```

threshold = 0.35  
A blue arrow pointing from the sorted y\_pred list to the binary classification result.

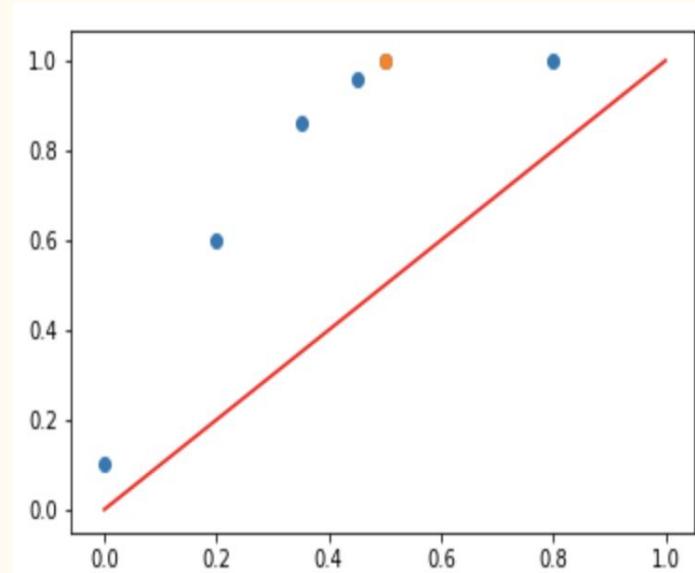
```
y_pred = [  
    0,  
    1,  
    1,  
    1,  
    1,  
    0  
]
```

# ROC-AUC

```
y_true = [    y_pred = [  
    0,  
    1,  
    1,  
    0,  
    0,  
    0]  
]  
]
```



$$\begin{aligned} \text{TPR} &= 1 \\ \text{FPR} &= 0.5 \end{aligned}$$



# ROC-AUC

```
>>> import numpy as np
>>> from sklearn import metrics
>>> y = np.array([1, 1, 2, 2])
>>> scores = np.array([0.1, 0.4, 0.35, 0.8])
>>> fpr, tpr, thresholds = metrics.roc_curve(y, scores, pos_label=2)
>>> fpr
array([0. , 0. , 0.5, 0.5, 1. ])
>>> tpr
array([0. , 0.5, 0.5, 1. , 1. ])
>>> thresholds
array([1.8 , 0.8 , 0.4 , 0.35, 0.1 ])
```

# ROC-AUC

Как подобрать thresholds?

```
y_pred = [    y_true = [  
    0.1,          0,  
    0.5,          1,  
    0.8,          1,  
    0.4,          0,  
    0.4,          0,  
    0.3           0  
]
```



```
y_pred = [    y_true = [  
    0.8,          1,  
    0.5,          1,  
    0.4,          0,  
    0.4,          0,  
    0.3,          0,  
    0.1           0  
]
```

Thresholds = y\_pred

# ROC-AUC

id	оценка	класс
1	0.5	0
2	0.1	0
3	0.2	0
4	0.6	1
5	0.2	1
6	0.3	1
7	0.0	0

Табл. 1

id	оценка	класс
4	0.6	1
1	0.5	0
6	0.3	1
3	0.2	0
5	0.2	1
2	0.1	0
7	0.0	0

Табл. 2

id	> 0.25	класс
4	1	1
1	1	0
6	1	1
3	0	0
5	0	1
2	0	0
7	0	0

Табл. 3

Теперь будем просматривать строки табл. 2 сверху вниз и прорисовывать на сетке линии, переходя их одного узла в другой. Стартуем из точки  $(0, 0)$ . Если значение метки класса в просматриваемой строке 1, то делаем шаг вверх; если 0, то делаем шаг вправо. Ясно, что в итоге мы попадём в точку  $(1, 1)$ , т.к. сделаем в сумме  $m$  шагов вверх и  $n$  шагов вправо.

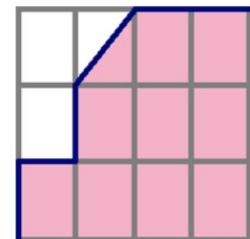
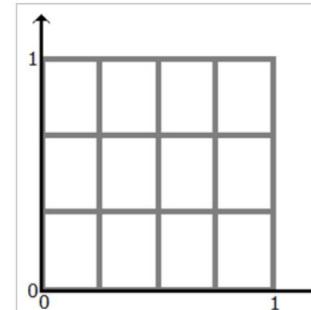
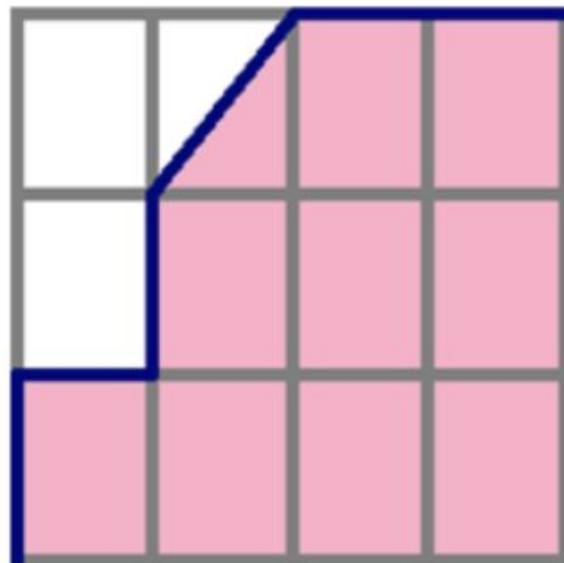


Рис.1. Построение ROC-кривой.

# ROC-AUC

threshold = 0.75



оценка

0  
0  
0  
0  
0  
0  
0

id	оценка	класс
4	0.6	1
1	0.5	0
6	0.3	1
3	0.2	0
5	0.2	1
2	0.1	0
7	0.0	0

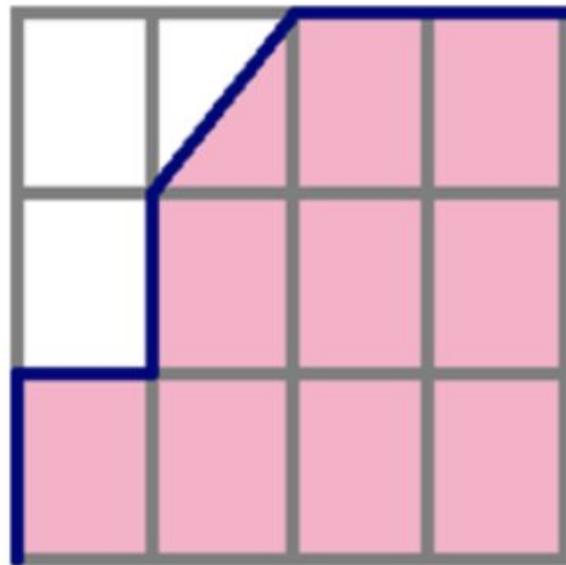
Табл. 2

id	> 0.25	класс
4	1	1
1	1	0
6	1	1
3	0	0
5	0	1
2	0	0
7	0	0

Табл. 3

# ROC-AUC

threshold = 0.6



оценка

1  
0  
0  
0  
0  
0  
0

id	оценка	класс
4	0.6	1
1	0.5	0
6	0.3	1
3	0.2	0
5	0.2	1
2	0.1	0
7	0.0	0

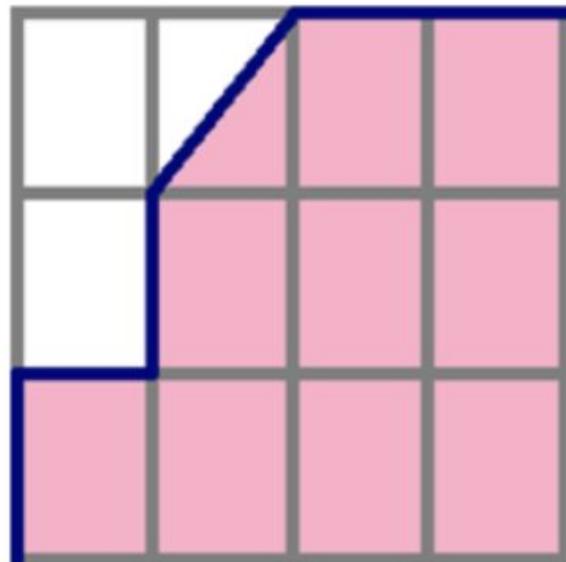
Табл. 2

id	> 0.25	класс
4	1	1
1	1	0
6	1	1
3	0	0
5	0	1
2	0	0
7	0	0

Табл. 3

# ROC-AUC

threshold = 0.6



оценка

1  
0  
0  
0  
0  
0  
0

id	оценка	класс
4	0.6	1
1	0.5	0
6	0.3	1
3	0.2	0
5	0.2	1
2	0.1	0
7	0.0	0

Табл. 2

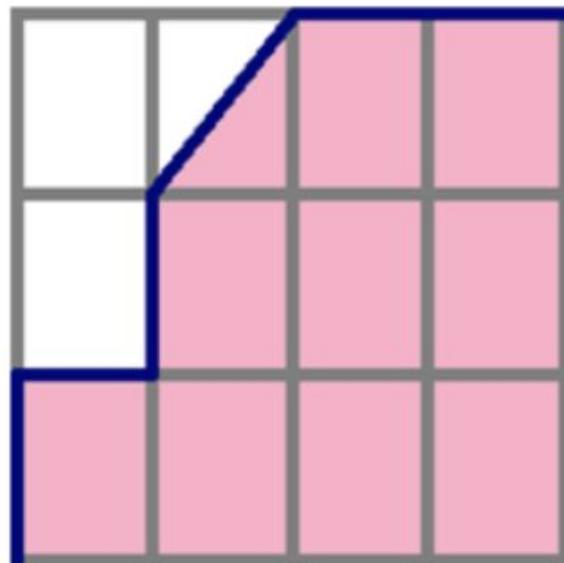
id	> 0.25	класс
4	1	1
1	1	0
6	1	1
3	0	0
5	0	1
2	0	0
7	0	0

Табл. 3

TPR вырос  
FPR не изменился

# ROC-AUC

threshold = 0.55



оценка

1  
0  
0  
0  
0  
0  
0

id	оценка	класс
4	0.6	1
1	0.5	0
6	0.3	1
3	0.2	0
5	0.2	1
2	0.1	0
7	0.0	0

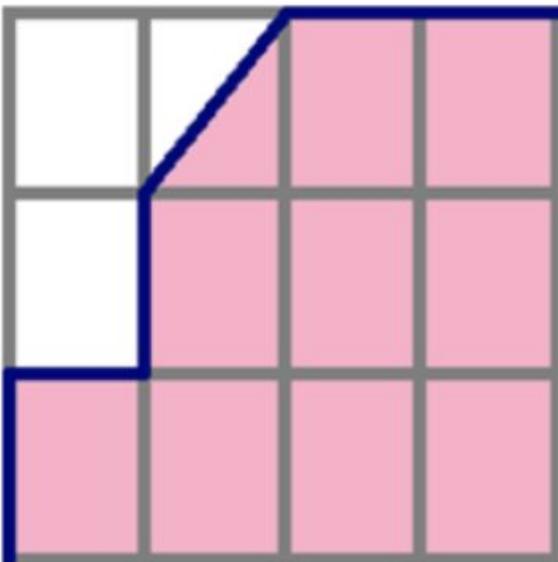
Табл. 2

id	> 0.25	класс
4	1	1
1	1	0
6	1	1
3	0	0
5	0	1
2	0	0
7	0	0

Табл. 3

# ROC-AUC

threshold = 0.6



оценка

1  
0  
0  
0  
0  
0  
0

id	оценка	класс
4	0.6	1
1	0.5	0
6	0.3	1
3	0.2	0
5	0.2	1
2	0.1	0
7	0.0	0

Табл. 2

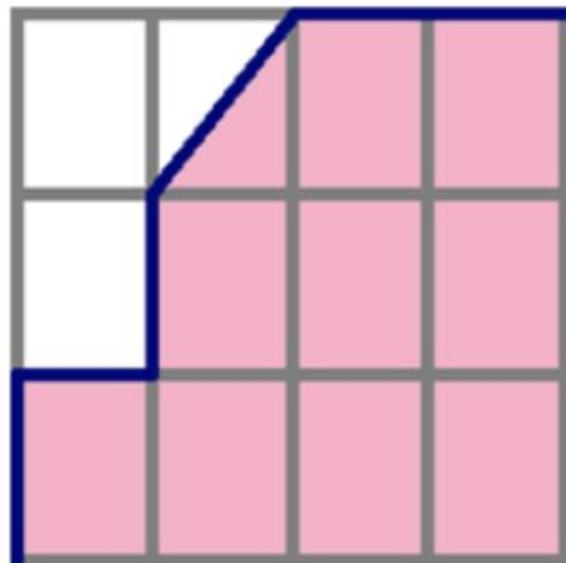
id	> 0.25	класс
4	1	1
1	1	0
6	1	1
3	0	0
5	0	1
2	0	0
7	0	0

Табл. 3

TPR не изменился  
FPR не изменился

# ROC-AUC

threshold = 0.5



оценка

1  
1  
0  
0  
0  
0  
0

id	оценка	класс
4	0.6	1
1	0.5	0
6	0.3	1
3	0.2	0
5	0.2	1
2	0.1	0
7	0.0	0

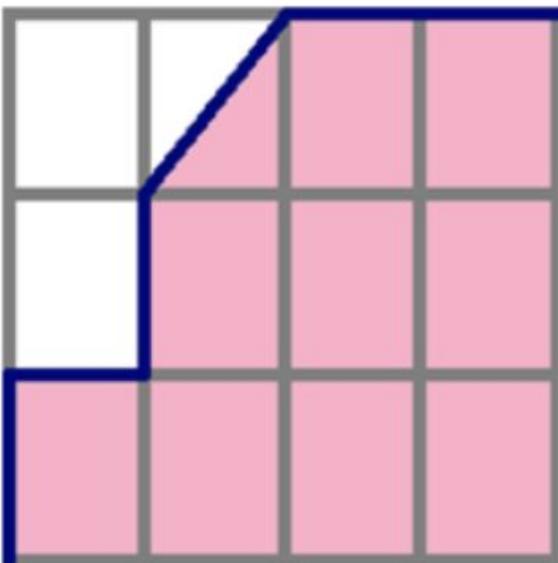
Табл. 2

id	> 0.25	класс
4	1	1
1	1	0
6	1	1
3	0	0
5	0	1
2	0	0
7	0	0

Табл. 3

# ROC-AUC

threshold = 0.5



оценка

1  
1  
0  
0  
0  
0  
0

id	оценка	класс
4	0.6	1
1	0.5	0
6	0.3	1
3	0.2	0
5	0.2	1
2	0.1	0
7	0.0	0

Табл. 2

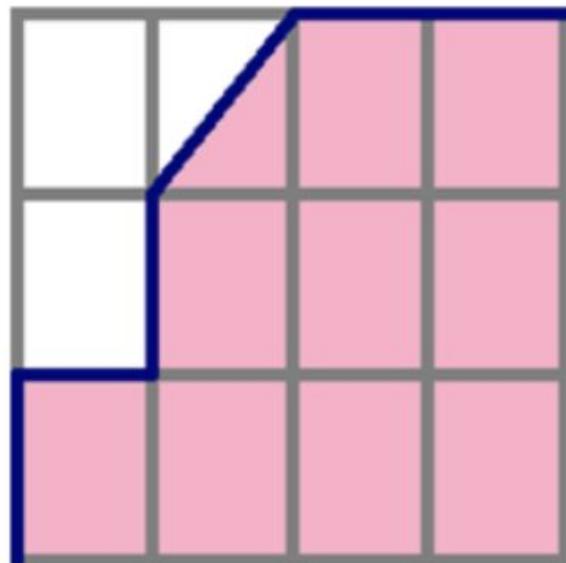
id	> 0.25	класс
4	1	1
1	1	0
6	1	1
3	0	0
5	0	1
2	0	0
7	0	0

Табл. 3

TPR не изменился  
FPR вырос

# ROC-AUC

threshold = 0.3



оценка

1  
1  
1  
0  
0  
0  
0

id	оценка	класс
4	0.6	1
1	0.5	0
6	0.3	1
3	0.2	0
5	0.2	1
2	0.1	0
7	0.0	0

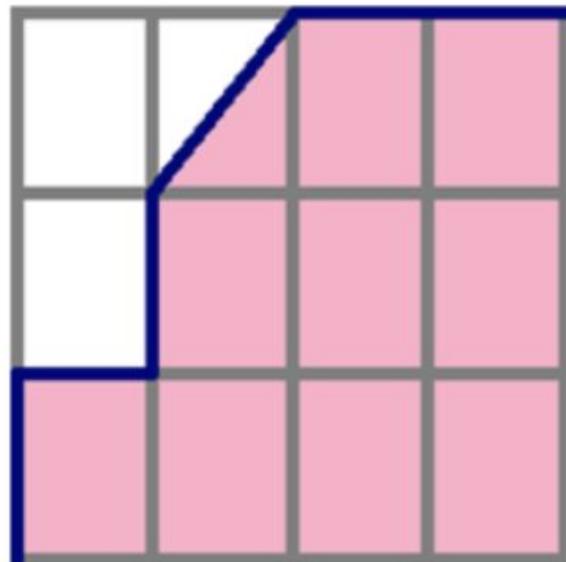
Табл. 2

id	> 0.25	класс
4	1	1
1	1	0
6	1	1
3	0	0
5	0	1
2	0	0
7	0	0

Табл. 3

# ROC-AUC

threshold = 0.3



оценка

1  
1  
1  
0  
0  
0  
0

id	оценка	класс
4	0.6	1
1	0.5	0
6	0.3	1
3	0.2	0
5	0.2	1
2	0.1	0
7	0.0	0

Табл. 2

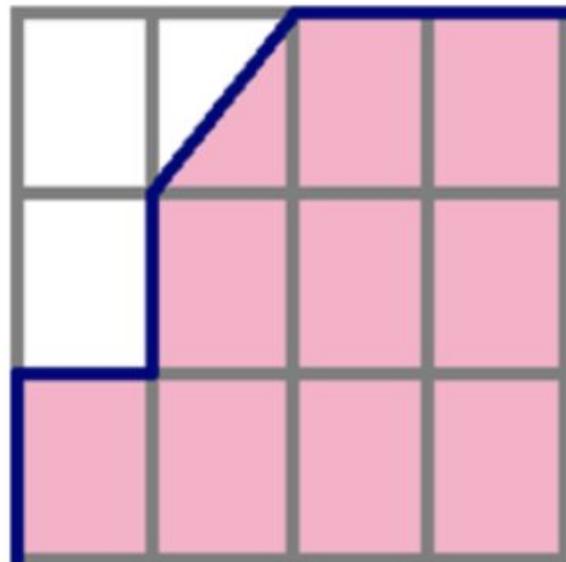
id	> 0.25	класс
4	1	1
1	1	0
6	1	1
3	0	0
5	0	1
2	0	0
7	0	0

Табл. 3

TPR вырос  
FPR не изменился

# ROC-AUC

threshold = 0.2



оценка

1  
1  
1  
1  
1  
0  
0

id	оценка	класс
4	0.6	1
1	0.5	0
6	0.3	1
3	0.2	0
5	0.2	1
2	0.1	0
7	0.0	0

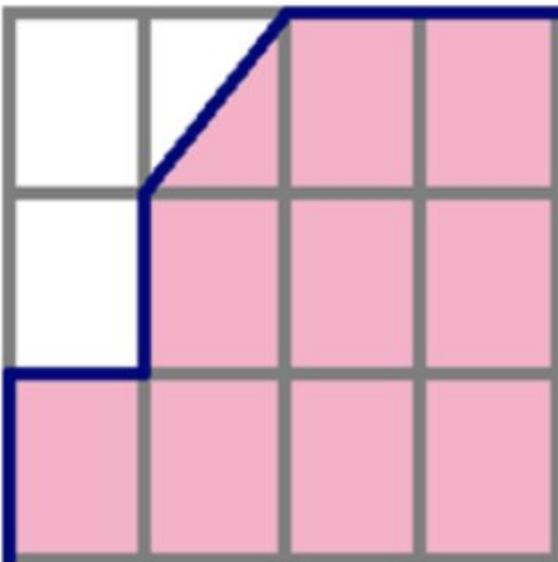
Табл. 2

id	> 0.25	класс
4	1	1
1	1	0
6	1	1
3	0	0
5	0	1
2	0	0
7	0	0

Табл. 3

# ROC-AUC

threshold = 0.2



оценка

1  
1  
1  
1  
1  
0  
0

id	оценка	класс
4	0.6	1
1	0.5	0
6	0.3	1
3	0.2	0
5	0.2	1
2	0.1	0
7	0.0	0

Табл. 2

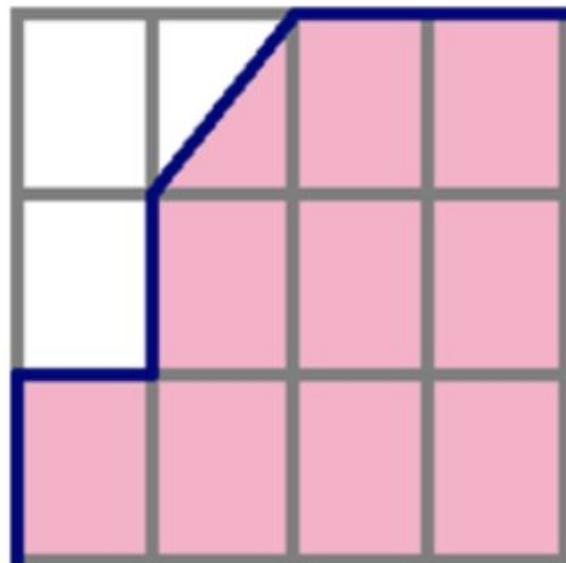
id	> 0.25	класс
4	1	1
1	1	0
6	1	1
3	0	0
5	0	1
2	0	0
7	0	0

Табл. 3

TPR вырос  
FPR вырос

# ROC-AUC

threshold = 0.1



оценка

1  
1  
1  
1  
1  
1  
0

id	оценка	класс
4	0.6	1
1	0.5	0
6	0.3	1
3	0.2	0
5	0.2	1
2	0.1	0
7	0.0	0

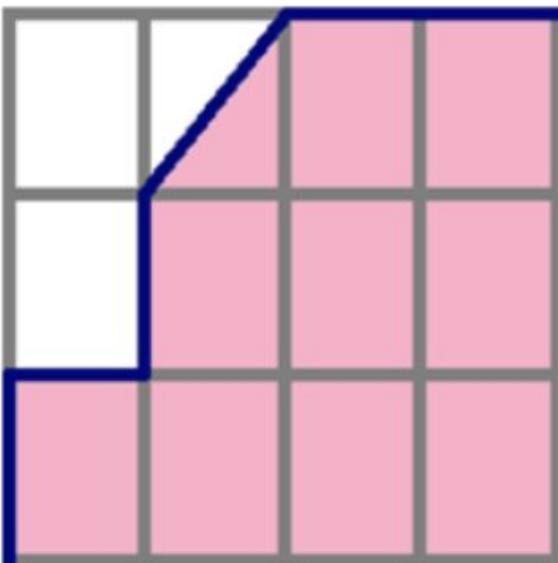
Табл. 2

id	> 0.25	класс
4	1	1
1	1	0
6	1	1
3	0	0
5	0	1
2	0	0
7	0	0

Табл. 3

# ROC-AUC

threshold = 0.1



оценка

1  
1  
1  
1  
1  
1  
0

id	оценка	класс
4	0.6	1
1	0.5	0
6	0.3	1
3	0.2	0
5	0.2	1
2	0.1	0
7	0.0	0

Табл. 2

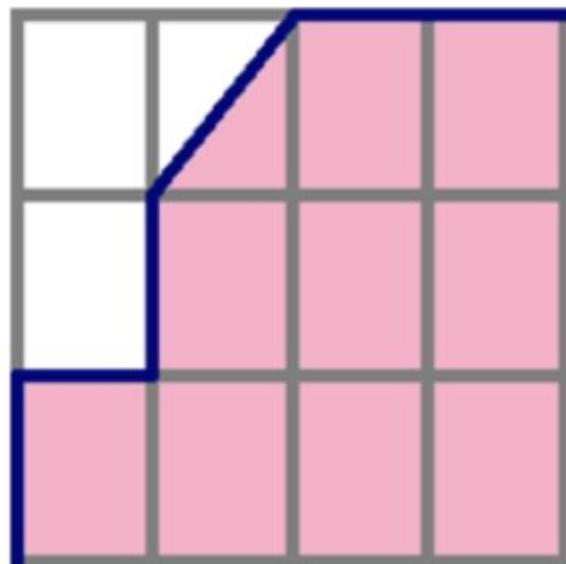
id	> 0.25	класс
4	1	1
1	1	0
6	1	1
3	0	0
5	0	1
2	0	0
7	0	0

Табл. 3

TPR не изменился  
FPR вырос

# ROC-AUC

threshold = 0.0



оценка

1  
1  
1  
1  
1  
1  
1

id	оценка	класс
4	0.6	1
1	0.5	0
6	0.3	1
3	0.2	0
5	0.2	1
2	0.1	0
7	0.0	0

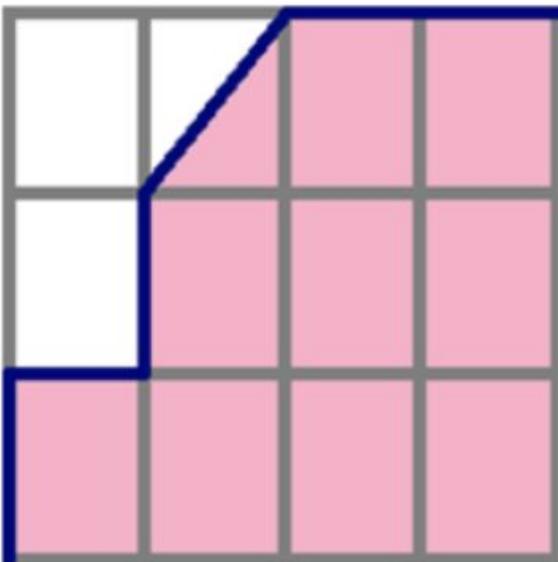
Табл. 2

id	> 0.25	класс
4	1	1
1	1	0
6	1	1
3	0	0
5	0	1
2	0	0
7	0	0

Табл. 3

# ROC-AUC

threshold = 0.0



оценка

1  
1  
1  
1  
1  
1  
1

id	оценка	класс
4	0.6	1
1	0.5	0
6	0.3	1
3	0.2	0
5	0.2	1
2	0.1	0
7	0.0	0

Табл. 2

id	> 0.25	класс
4	1	1
1	1	0
6	1	1
3	0	0
5	0	1
2	0	0
7	0	0

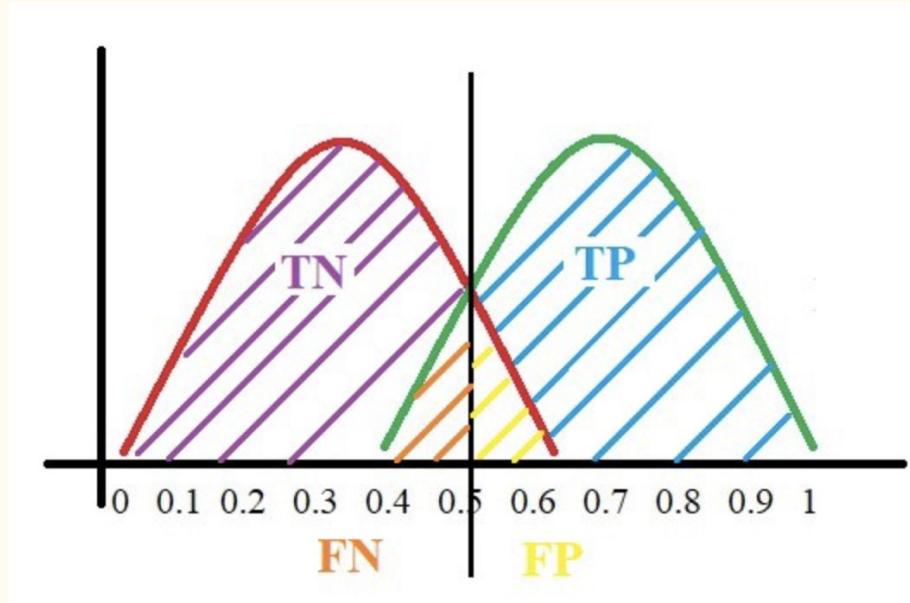
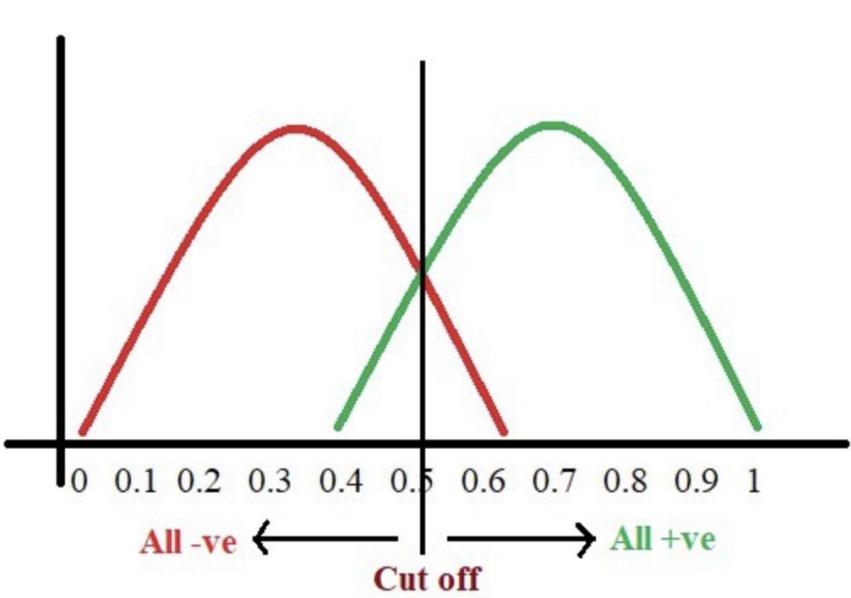
Табл. 3

TPR не изменился  
FPR вырос

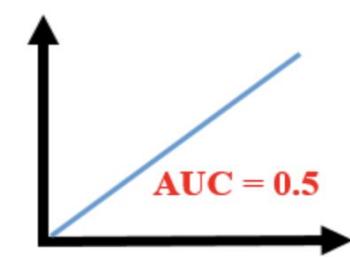
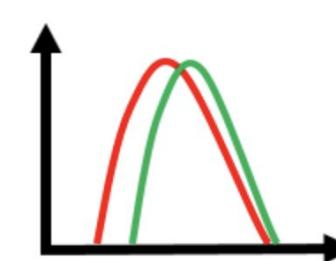
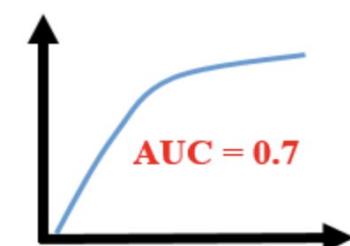
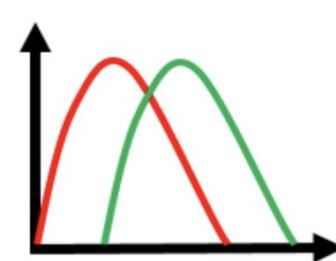
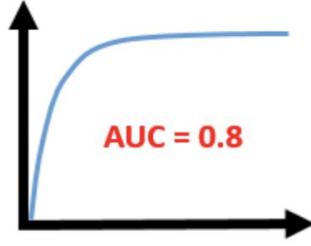
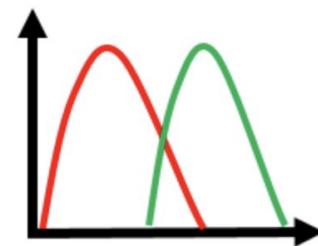
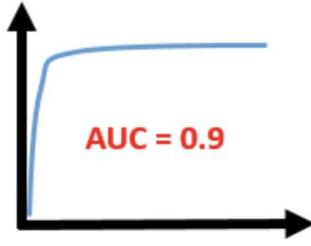
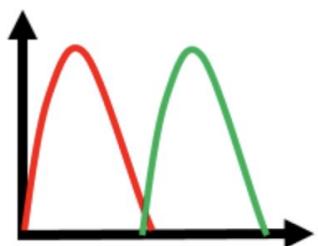
# ROC-AUC

Критерий AUC-ROC устойчив к несбалансированным классам и может быть интерпретирован как вероятность того, что случайно выбранный positive объект будет отранжирован классификатором выше (будет иметь более высокую вероятность быть positive), чем случайно выбранный negative объект.

# ROC-AUC



# ROC-AUC



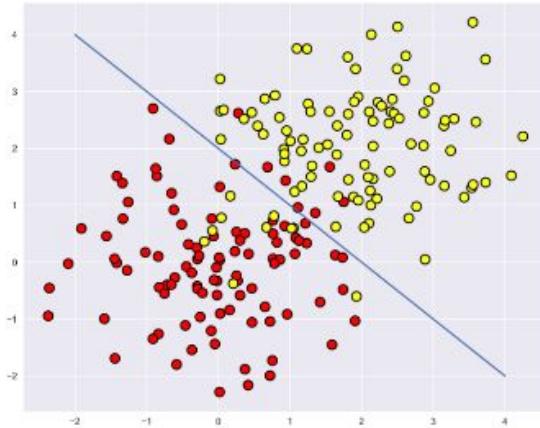
AUC ROC показывает, насколько хорошо вероятности положительного класса отделены от вероятностей отрицательного

# Naive Bayes

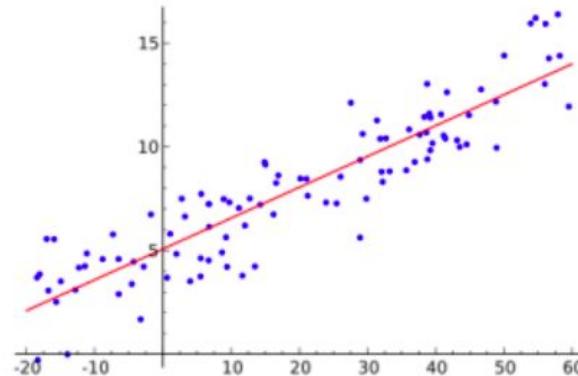
---

100  
010

# Регрессия VS Классификация

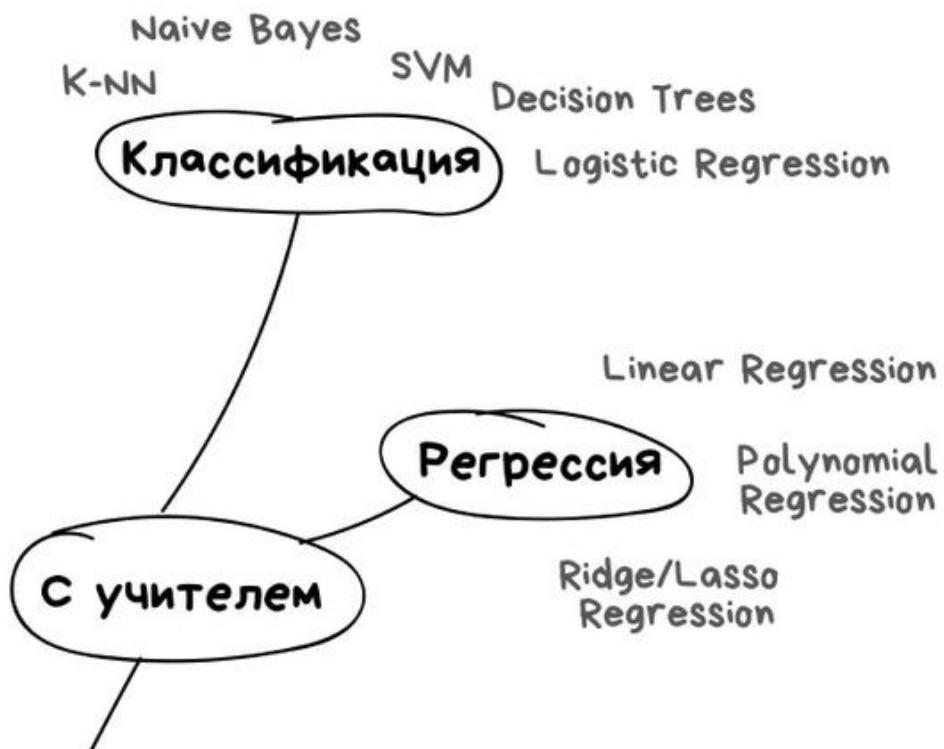


$$Y = \{1, \dots, N\}$$



$$Y \subseteq \mathbf{R}$$





# Наивный байесовский классификатор

- Spam detection
  - Сегментация новостных статей по их тематике;
  - Определение эмоционального окраса блока текста;
  - Программное обеспечение для распознавания лиц.

$$c = \arg \max_C P(C|O)$$

объект для классификации

$$P(A|B) = \frac{P(B|A) P(A)}{P(B)}$$

$$c = \arg \max_{c \in C} P(c|o_1 o_2 \dots o_n) = \arg \max_{c \in C} P(c) \prod P(o_i|c)$$

# Найвный Байес в SKLearn

**GaussianNB:** implements the Gaussian Naive Bayes algorithm for classification. The likelihood of the features is assumed to be Gaussian

**MultinomialNB:** implements the naive Bayes algorithm for multinomially distributed data

**ComplementNB:** implements the complement naive Bayes (CNB) algorithm. CNB is an adaptation of the standard multinomial naive Bayes (MNB) algorithm that is particularly suited for imbalanced data sets.

**BernoulliNB:** implements the naive Bayes training and classification algorithms for data that is distributed according to multivariate Bernoulli distributions

# Logical models

---

100  
010

# Логические методы классификации

Логическая закономерность – предикат  $R:X \rightarrow \{0,1\}$ :

1. Интерпретируем
  - a. записывается на естественном языке
  - b. зависит от небольшого числа параметров (1-7)
2. Информативен

$$p_c(R) = \#\{x_i : R(x_i)=1 \text{ и } y_i=c\} \rightarrow \max;$$

$$n_c(R) = \#\{x_i : R(x_i)=1 \text{ и } y_i \neq c\} \rightarrow \min;$$



# Логические методы классификации

Примеры информативности:

*Если возраст > 60 и пациент ранее перенёс инфаркт,  
то операцию не делать, риск отрицательного исхода 60%.*

*Если в анкете указан домашний телефон  
и зарплата > \$2000 и сумма кредита < \$5000  
то кредит можно выдать, риск дефолта 5%.*

# Закономерности: наборы правил

1. Пороговое условие (решающий пень, decision stump):

$$R(x) = [f_j(x) \leq a_j] \text{ или } [a_j \leq f_j(x) \leq b_j].$$

2. Конъюнкция пороговых условий:

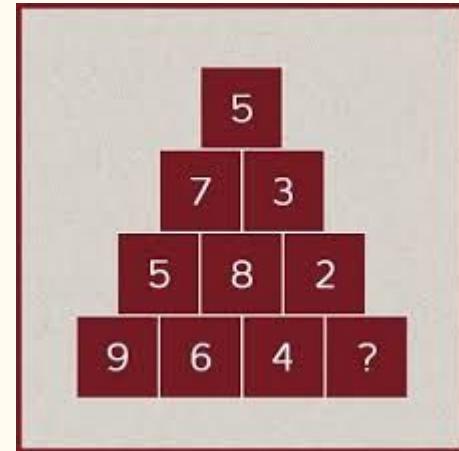
$$R(x) = \bigwedge_{j \in J} [a_j \leq f_j(x) \leq b_j].$$

3. Синдром — выполнение не менее  $d$  условий из  $|J|$ ,  
(при  $d = |J|$  это конъюнкция, при  $d = 1$  — дизъюнкция):

$$R(x) = \left[ \sum_{j \in J} [a_j \leq f_j(x) \leq b_j] \geq d \right],$$

# Обучение логических классификаторов

1. Шаги индукции правил (rule induction)
  - a. Выбор семейства правил для поиска закономерностей
  - b. Порождение правил (rule generation)
  - c. Отбор правил-закономерностей (rule selection)
  - d. Построение классификатора из правил как из признаков. Например, взвешенное голосование.



- ❶ Как изобретать признаки?
  - не наука, а искусство (озарения, мозговые штурмы,...)
- ❷ Какого вида закономерности нам нужны?
  - простые формулы от малого числа признаков
- ❸ Как определять информативность?
  - так, чтобы одновременно  $p_c \rightarrow \max$ ,  $n_c \rightarrow \min$
- ❹ Как строить отдельные закономерности?
  - методами отбора признаков
- ❺ Как объединять закономерности в алгоритм?
  - методами построения композиций классификаторов

*Закономерность* — это хорошо интерпретируемый одноклассовый классификатор с отказами.

# Оценка качества информативности

Проблема: надо сравнивать закономерности  $R$ .

Как свернуть два критерия в один критерий информативности?

$$\begin{cases} p(R) \rightarrow \max \\ n(R) \rightarrow \min \end{cases} \stackrel{?}{\Rightarrow} I(p, n) \rightarrow \max$$

Очевидные, но не всегда адекватные свёртки:

- $I(p, n) = \frac{p}{p + n} \rightarrow \max$  (precision);
- $I(p, n) = p - n \rightarrow \max$  (accuracy);

## Пример:

при  $P = 200$ ,  $N = 100$  и различных  $p$  и  $n$ .

Простые эвристики не всегда адекватны:

$p$	$n$	$p-n$	$p-5n$	$\frac{p}{P}-\frac{n}{N}$	$\frac{p}{n+1}$	IStat· $\ell$	IGain· $\ell$	$\sqrt{p}-\sqrt{n}$
50	0	<b>50</b>	50	0.25	50	22.65	23.70	7.07
100	50	<b>50</b>	-150	0	1.96	2.33	1.98	2.93
50	9	41	<b>5</b>	0.16	<b>5</b>	7.87	7.94	4.07
5	0	5	<b>5</b>	0.03	<b>5</b>	2.04	3.04	2.24
100	0	<b>100</b>	100	<b>0.5</b>	100	52.18	53.32	10.0
140	20	<b>120</b>	40	<b>0.5</b>	6.67	37.09	37.03	7.36

# Более адекватные свертки

сколько информации мы получим о разделении объектов на два класса, если узнаем  $\mathbf{R}$ ?

- энтропийный критерий прироста информации:

$$\text{IGain}(p, n) = h\left(\frac{P}{\ell}\right) - \frac{p+n}{\ell} h\left(\frac{p}{p+n}\right) - \frac{\ell-p-n}{\ell} h\left(\frac{P-p}{\ell-p-n}\right) \rightarrow \max,$$

где  $h(q) = -q \log_2 q - (1-q) \log_2(1-q)$

- критерий Джини (Gini impurity):

$$\text{IGini}(p, n) = \text{IGain}(p, n) \text{ при } h(q) = 4q(1-q)$$

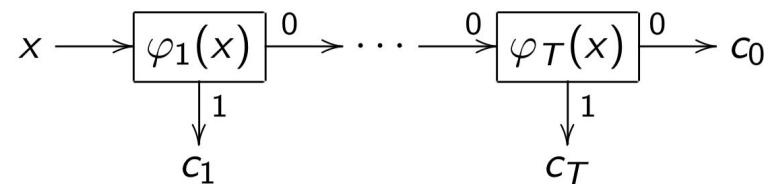
“загрязненность” распределения

# Композиции закономерностей

- Взвешенное (или простое) голосование

$$a(x) = \arg \max_{y \in Y} \sum_{t=1}^{T_y} w_{yt} R_{yt}(x)$$

- Решающий список (по старшинству)



# Резюме до этого момента

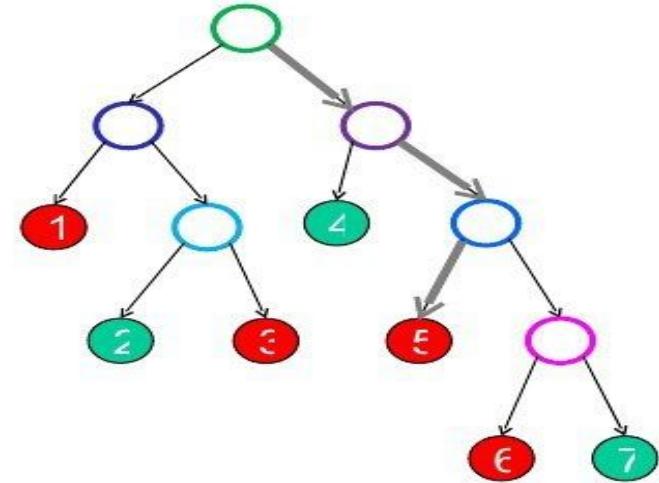
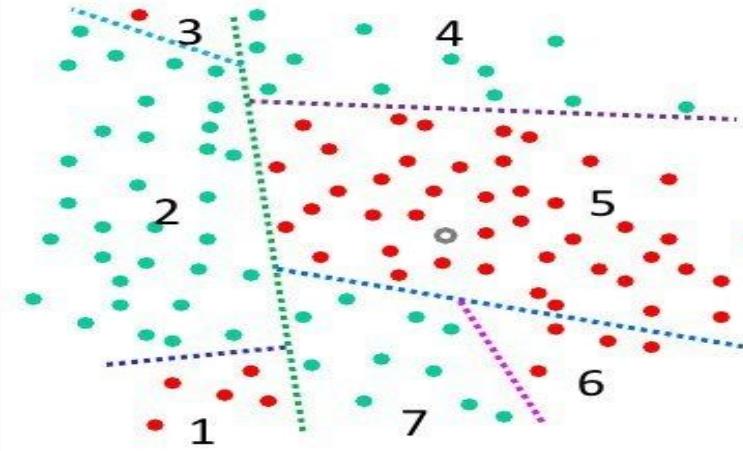
- Правило -- интерпретируемый предикат (функция)  $X: - \rightarrow \{0, 1\}$
- Закономерность -- информативное правило (набор правил)
- Критерий информативности существует много разных
- Как строить закономерность: отбирать признаки по их информативности
- Как строить композиции закономерностей:
  - голосование
  - решающий список
  - *решающее дерево*

# Решающее дерево



# Пример решающего дерева

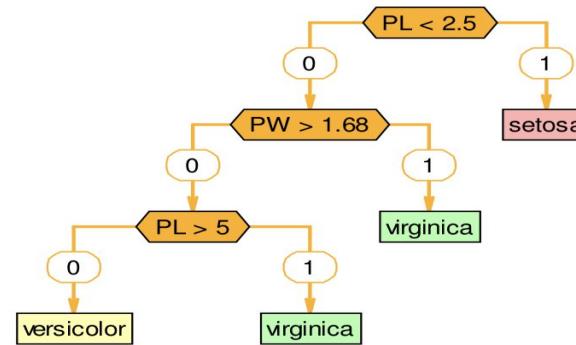
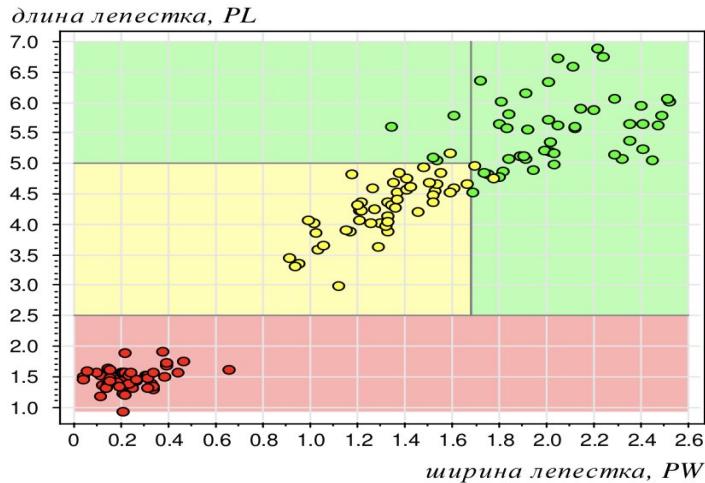
Решающее дерево



Slide by V.Lempitsky

# Пример решающего дерева

Задача Фишера о классификации цветков ириса на 3 класса, в выборке по 50 объектов каждого класса, 4 признака.



**На графике:** в осях двух самых информативных признаков (из 4) два класса разделились без ошибок, на третьем 3 ошибки.

# Построение решающего дерева

- 1: **ПРОЦЕДУРА** LearnID3 ( $U \subseteq X^\ell$ );
- 2: **если** все объекты из  $U$  лежат в одном классе  $c \in Y$  **то**
- 3:   **вернуть** новый лист  $v$ ,  $c_v := c$ ;
- 4: **найти** предикат с максимальной информативностью:  
 $\beta := \arg \max_{\beta \in \mathcal{B}} I(\beta, U)$ ;
- 5: **разбить** выборку на две части  $U = U_0 \cup U_1$  по предикату  $\beta$ :  
 $U_0 := \{x \in U : \beta(x) = 0\}$ ;  
 $U_1 := \{x \in U : \beta(x) = 1\}$ ;
- 6: **если**  $U_0 = \emptyset$  **или**  $U_1 = \emptyset$  **то**
- 7:   **вернуть** новый лист  $v$ ,  $c_v := \text{Мажоритарный класс}(U)$ ;
- 8: **создать** новую внутреннюю вершину  $v$ :  $\beta_v := \beta$ ;  
построить левое поддерево:  $L_v := \text{LearnID3}(U_0)$ ;  
построить правое поддерево:  $R_v := \text{LearnID3}(U_1)$ ;
- 9: **вернуть**  $v$ ;

# Решающие деревья: + и -

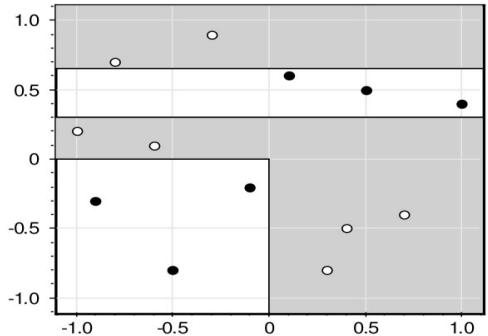
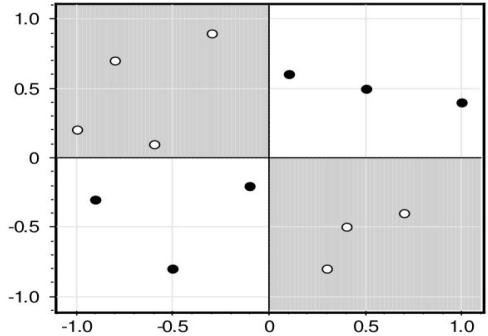
+

- Интерпретируемость, простота
- Гибкость: можно менять множества закономерностей
- Допустимы данные с пропусками
- Сложность построения линейна по длины выборки

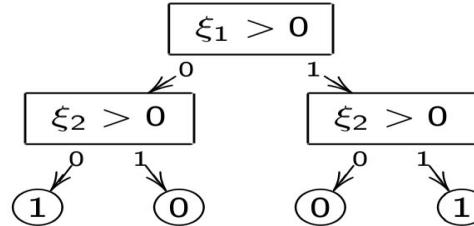
-

- Жадный алгоритм выбора правила для вершины пере усложняет дерево
- Чем дальше вершина от корня, тем слабее статистическая надежность решающего правила
- Высокая чувствительность к шуму, составу выборки

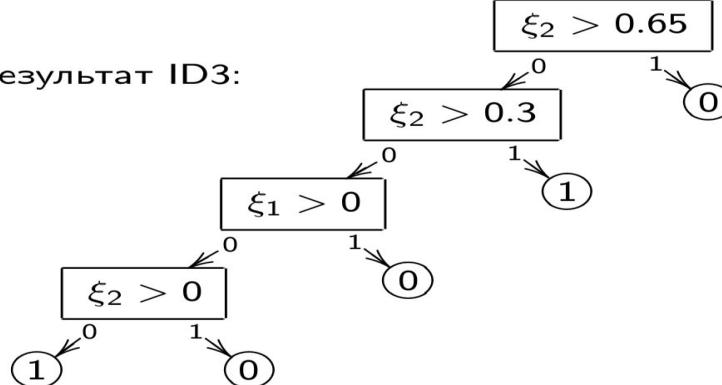
# Проблема переобучения



Оптимальное дерево для задачи XOR:



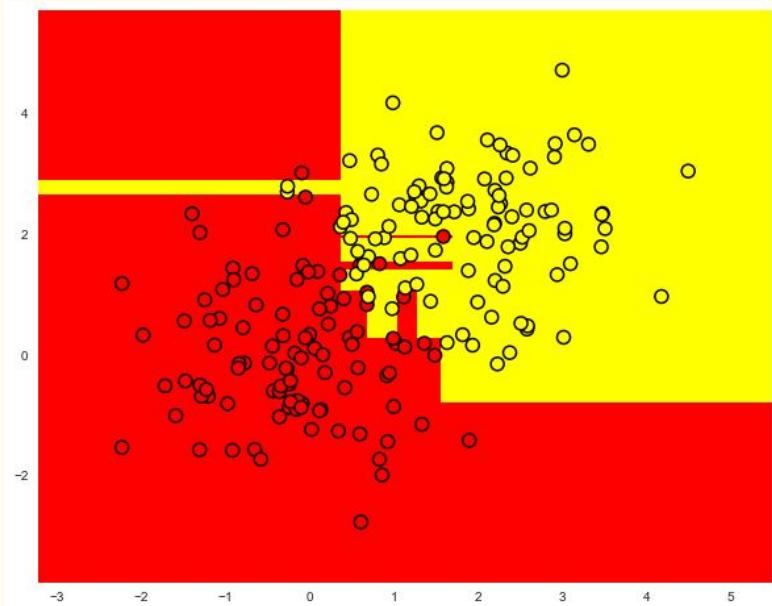
Результат ID3:



# Проблема переобучения

Пути решения:

1. Ограничение глубины дерева
2. Ограничение минимального числа объектов в листе
3. Стрижка дерева (pruning)



## sklearn.tree.DecisionTreeClassifier

```
class 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_decrease=0.0, min_impurity_split=None, class_weight=None, presort=False)
```

[source]

A decision tree classifier.

Read more in the [User Guide](#).

**Parameters:** **criterion** : string, optional (default="gini")

The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "entropy" for the information gain.

**splitter** : string, optional (default="best")

The strategy used to choose the split at each node. Supported strategies are "best" to choose the best split and "random" to choose the best random split.

**max\_depth** : int or None, optional (default=None)

The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min\_samples\_split samples.

Обобщение на случай регрессии:  $Y = \mathbb{R}$ ,  $y_v \in \mathbb{R}$ .

$U$  — множество объектов  $x_i$ , дошедших до вершины  $v$

Мера неопределённости — среднеквадратичная ошибка

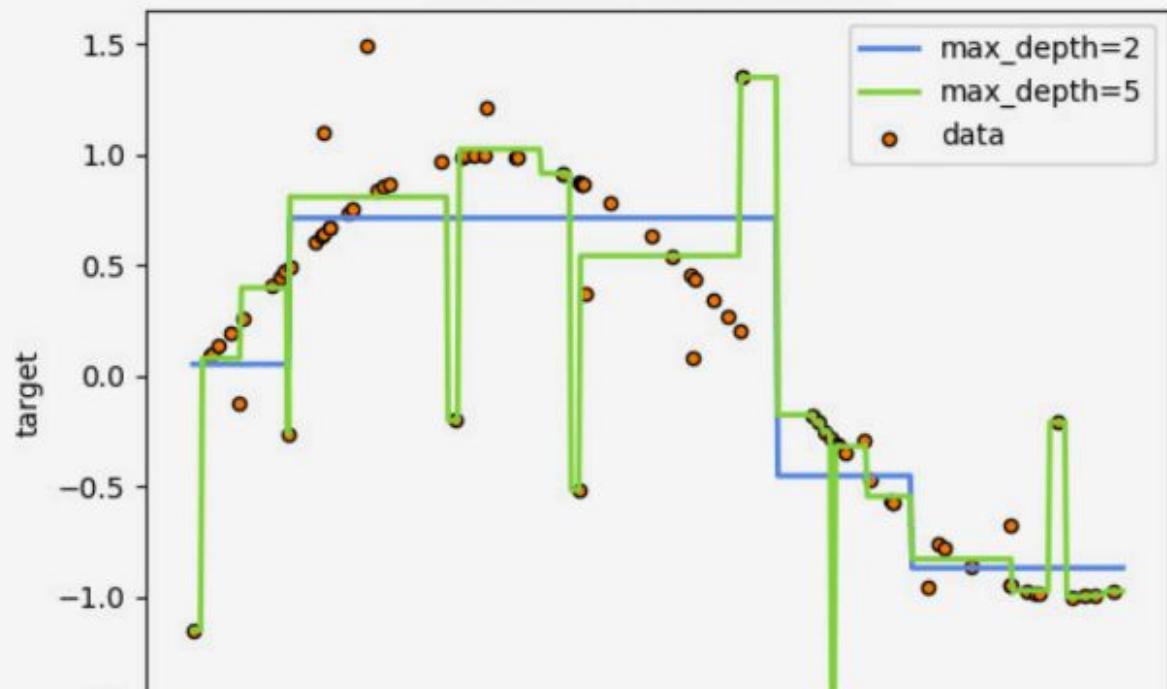
$$\Phi(U) = \min_{y \in Y} \frac{1}{|U|} \sum_{x_i \in U} (y - y_i)^2$$

Значение  $y_v$  в терминальной вершине  $v$  — МНК-решение:

$$y_v = \frac{1}{|U|} \sum_{x_i \in U} y_i$$

Дерево регрессии  $a(x)$  — это кусочно-постоянная функция.

## Decision Tree Regression

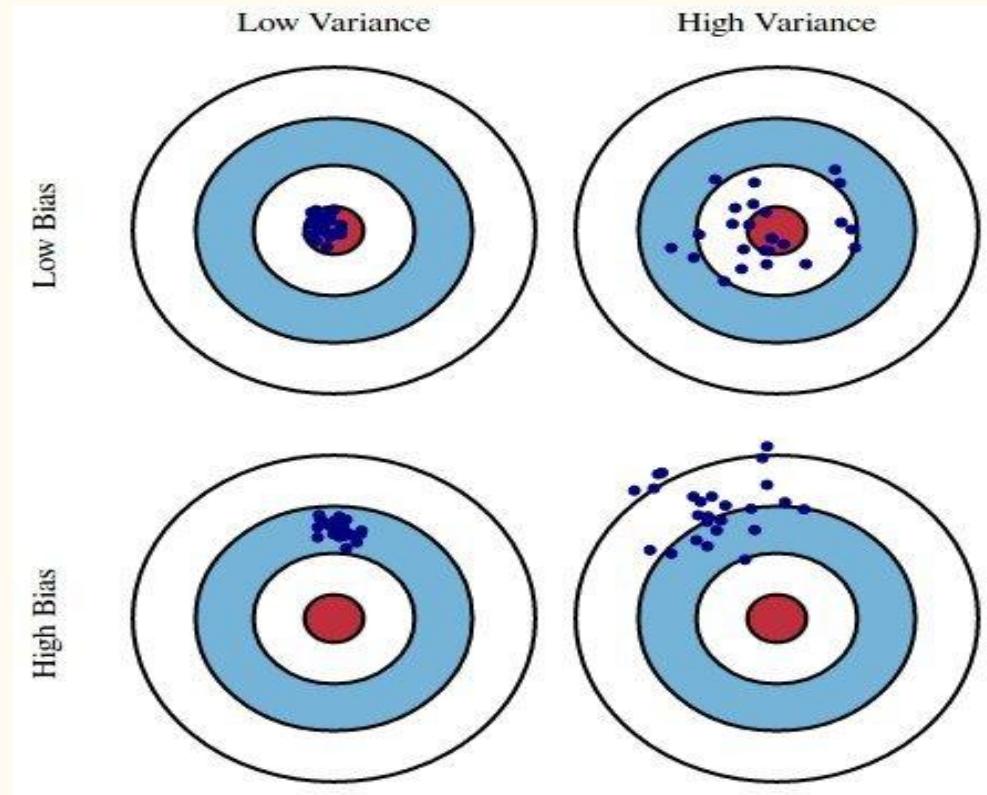


# Bias-Variance tradeoff

---

100  
010

# Bias and variance



# Разложение ошибки алгоритма

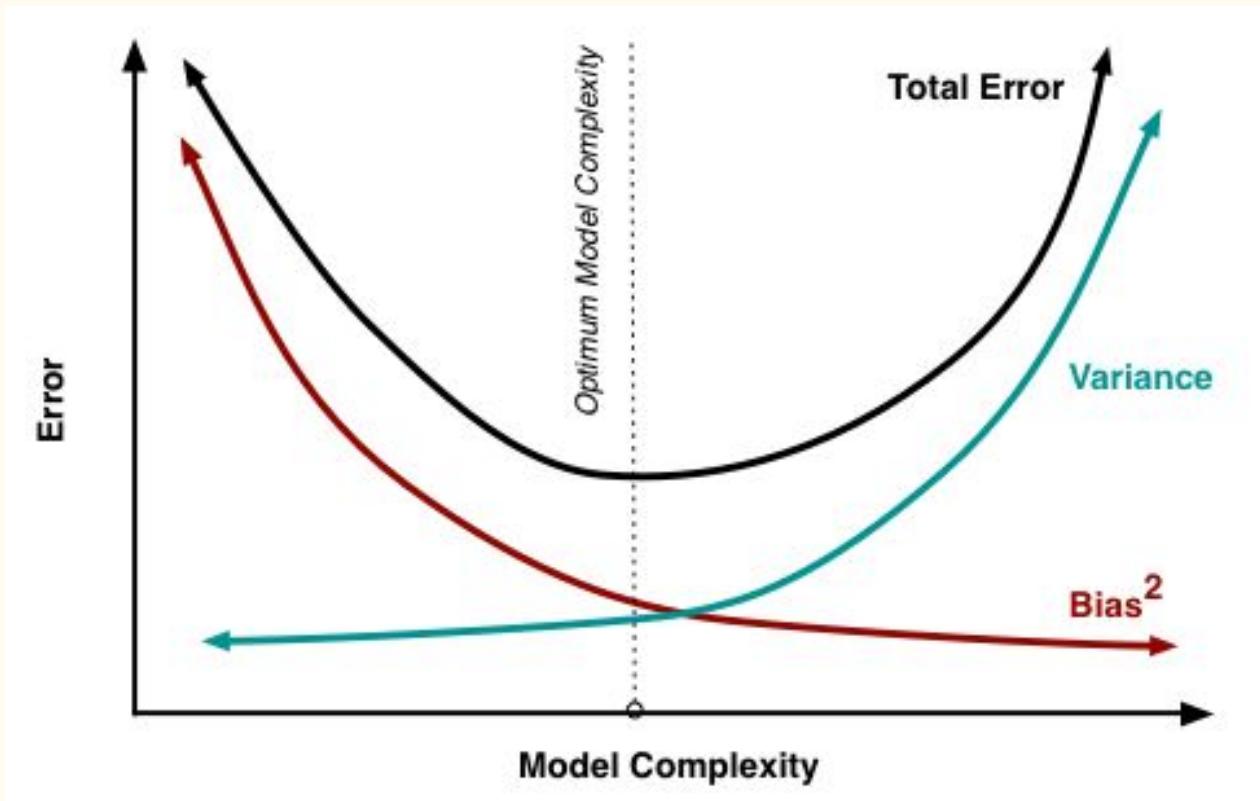
$$Err(x) = E \left[ (Y - \hat{f}(x))^2 \right]$$

The Err(x) can be further decomposed as

$$Err(x) = \left( E[\hat{f}(x)] - f(x) \right)^2 + E \left[ \left( \hat{f}(x) - E[\hat{f}(x)] \right)^2 \right] + \sigma_e^2$$

$$Err(x) = \text{Bias}^2 + \text{Variance} + \text{Irreducible Error}$$

# Типичная картина обучения



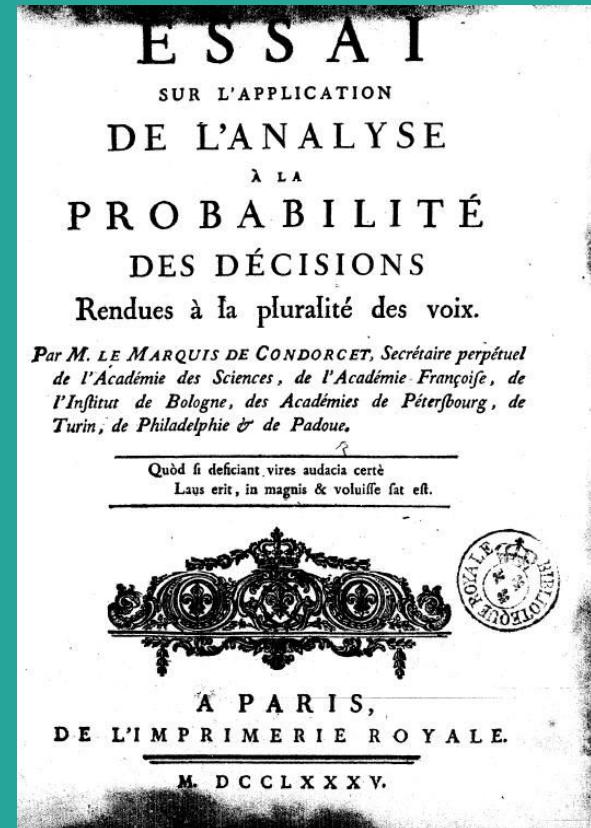
# Compositions

---

100  
010

Если каждый член жюри имеет независимое мнение, и если вероятность правильного решения члена жюри больше  $0.5$ , то тогда вероятность правильного решения присяжных в целом возрастает с увеличением количества членов жюри и стремится к единице.

Если же вероятность быть правым у каждого из членов жюри меньше  $0.5$ , то вероятность принятия правильного решения присяжными в целом монотонно уменьшается и стремится к нулю с увеличением количества присяжных.



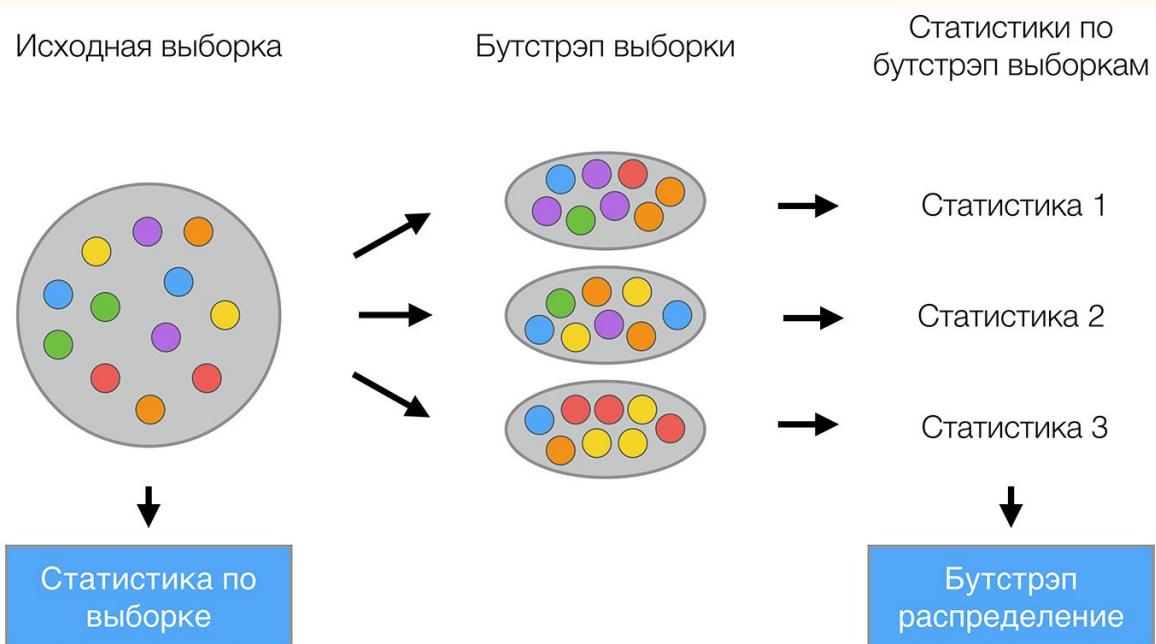
принцип Кондорсе, 1784

Собралось около 800 человек, которые попытались угадать вес быка на ярмарке. Бык весил 1198 фунтов. Ни один крестьянин не угадал точный вес быка, но если посчитать среднее от их предсказаний, то получим 1197 фунтов.



Гальтон, 1906 год

# Bootstrap

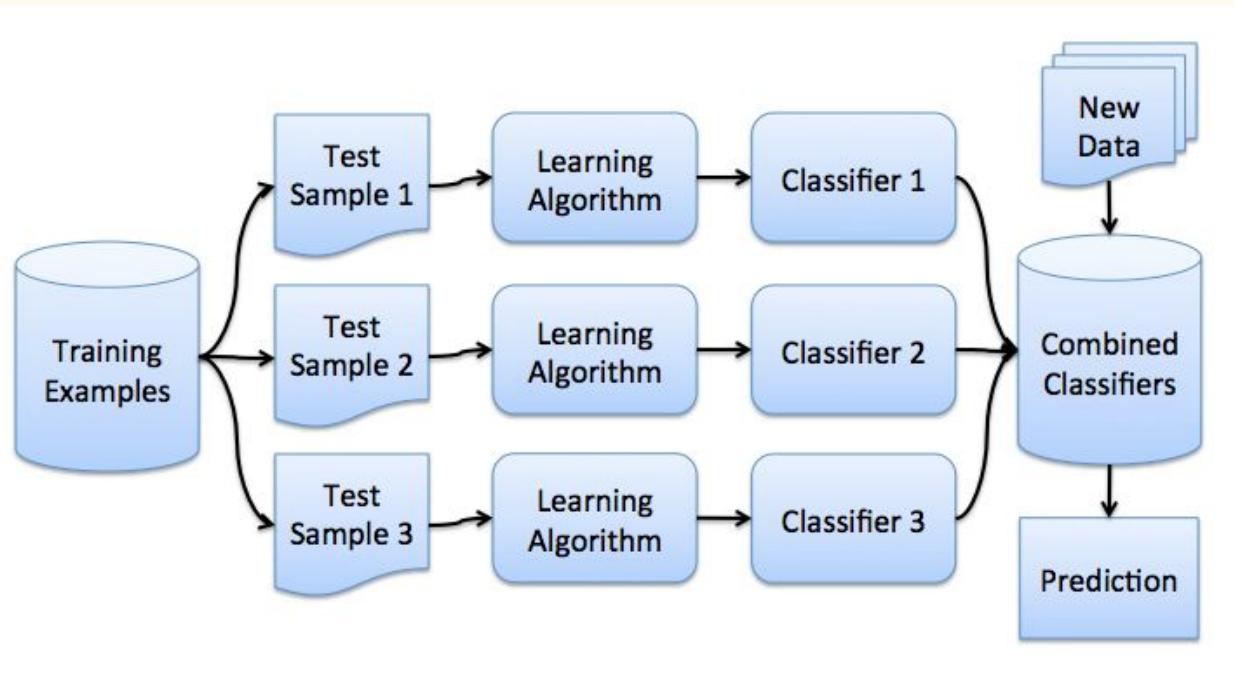


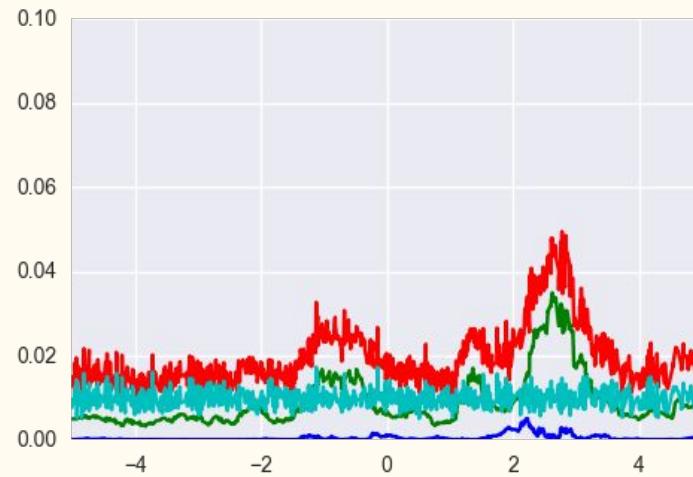
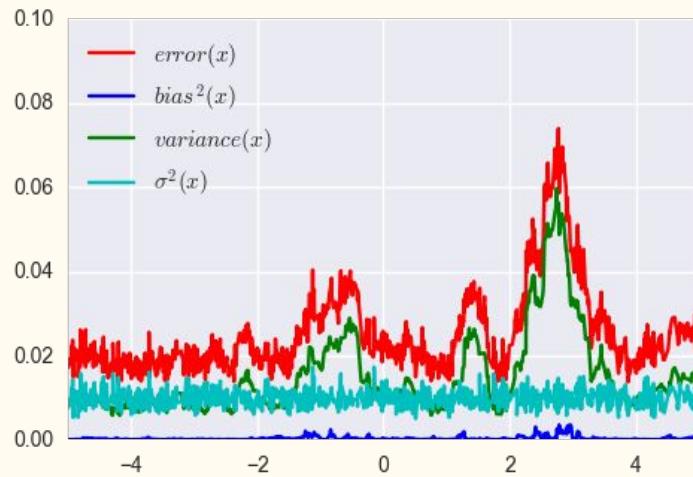
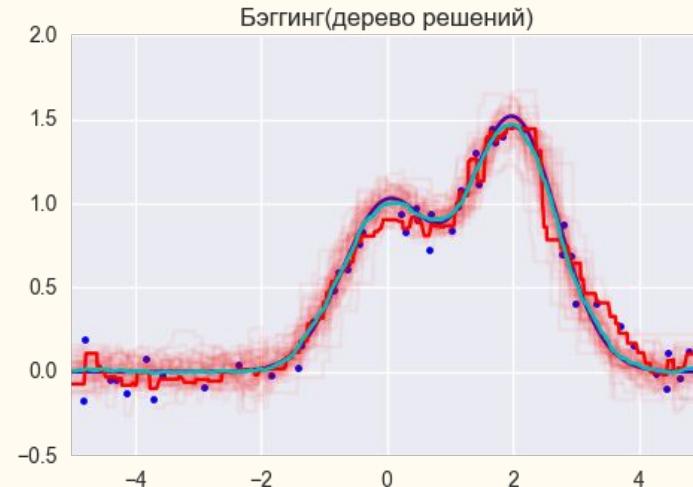
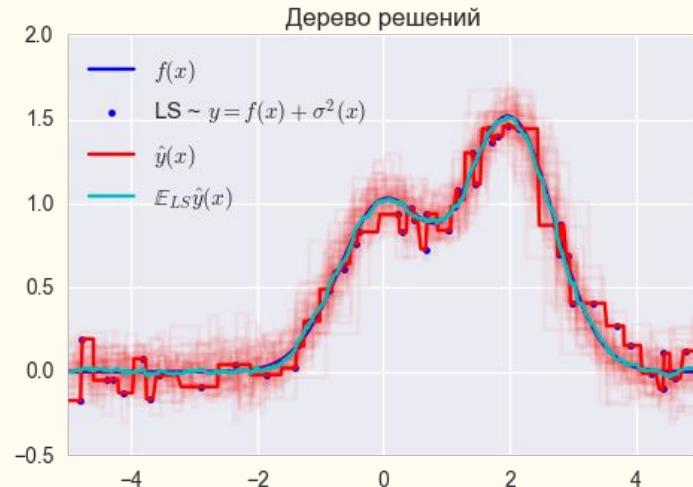
# Bagging (bootstrap aggregating)

Дана train sample X. С помощью **bootstrap** сгенерируем из неё M выборок. Теперь на каждой выборке обучим свой классификатор.

Итоговый классификатор будет усреднять ответы всех этих алгоритмов:

$$a(x) = \frac{1}{M} \sum_{i=1}^M a_i(x)$$



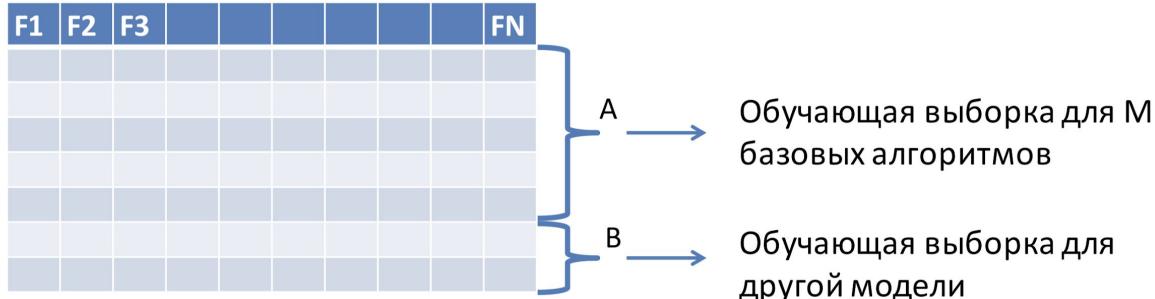


## Stacking

## Обучающая выборка:

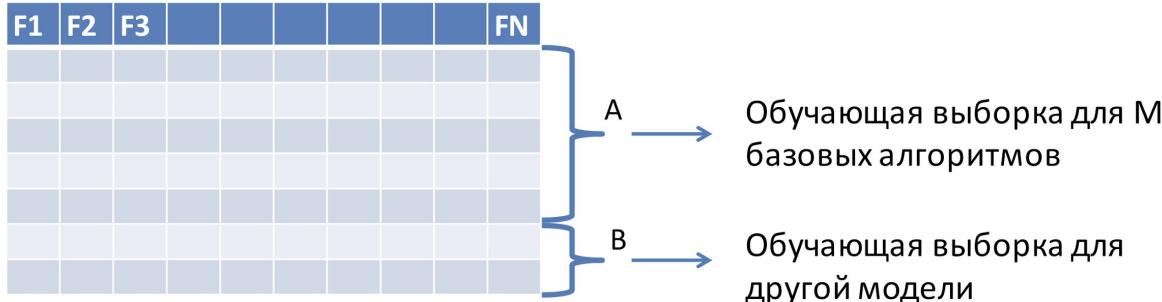
# Stacking

Обучающая выборка:



# Stacking

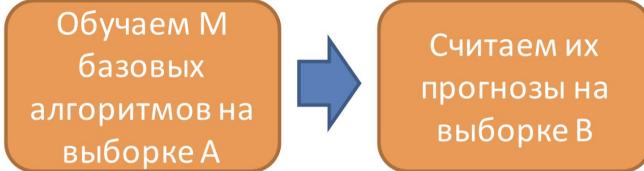
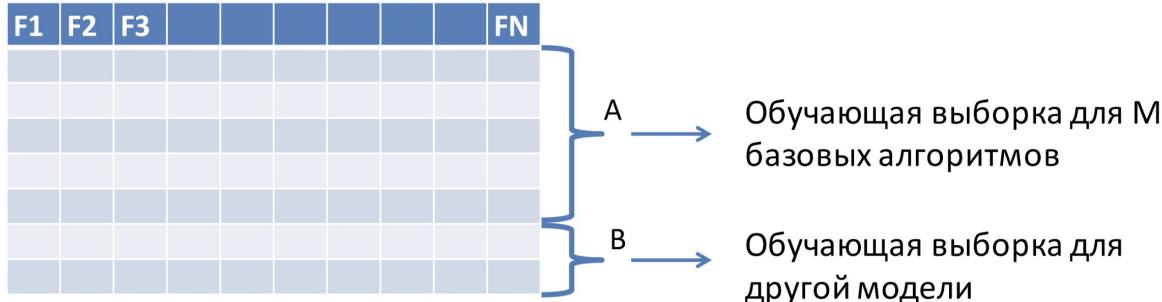
Обучающая выборка:



Обучаем M  
базовых  
алгоритмов на  
выборке A

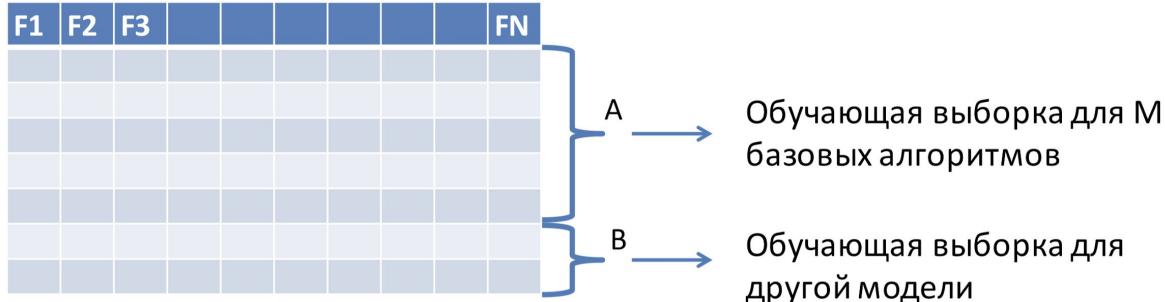
# Stacking

Обучающая выборка:



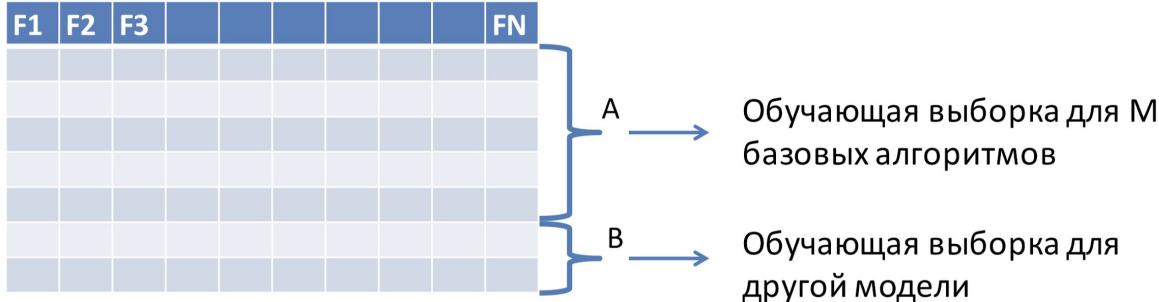
# Stacking

Обучающая выборка:



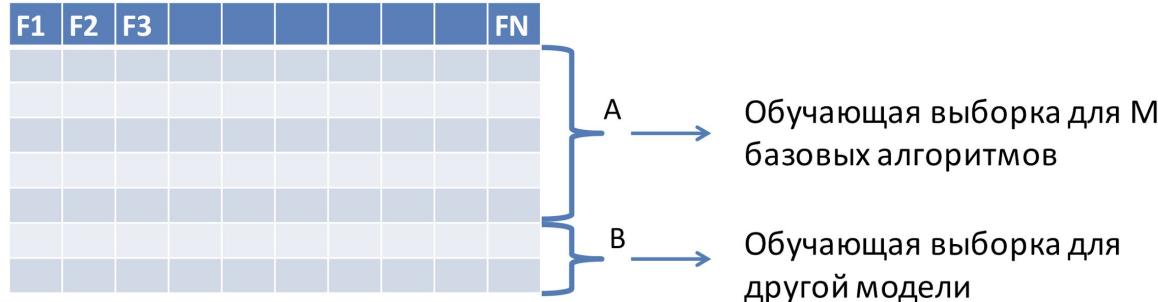
# Stacking

Обучающая выборка:



# Stacking

Обучающая выборка:



# Blending

Смесь нескольких сильных классификаторов:

$$a(x) = \sum_{t=1}^T \alpha_t b_t(x)$$

+ веса неотрицательны и дают в сумме единицу

# Blending

Смесь нескольких сильных классификаторов:

$$a(x) = \sum_{t=1}^T \alpha_t b_t(x)$$

+ веса неотрицательны и дают в сумме единицу

Преимущества и недостатки:

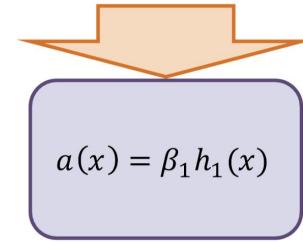
- Очень прост идеально, хорошо работает, логичен
- Иногда надо перебирать веса или использовать дискретную оптимизацию
- Не всегда композиция в виде взвешенной суммы – то, что надо. Иногда нужна более сложная композиция

# Boosting

Бустинг – жадное построение  
взвешенной суммы базовых  
алгоритмов  $h_k(x)$

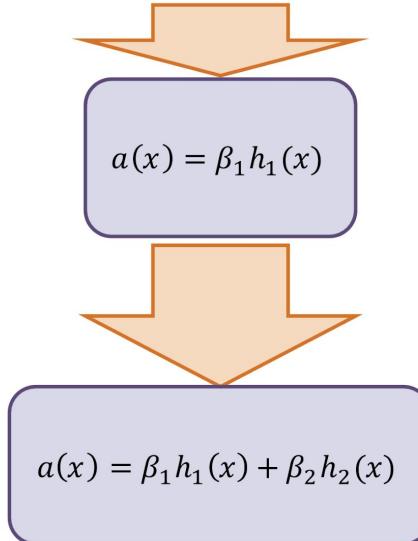
# Boosting

Бустинг – жадное построение  
взвешенной суммы базовых  
алгоритмов  $h_k(x)$



# Boosting

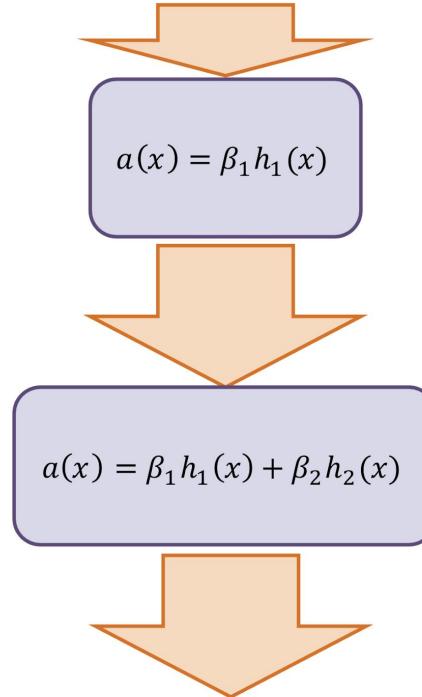
Бустинг – жадное построение  
взвешенной суммы базовых  
алгоритмов  $h_k(x)$



# Boosting

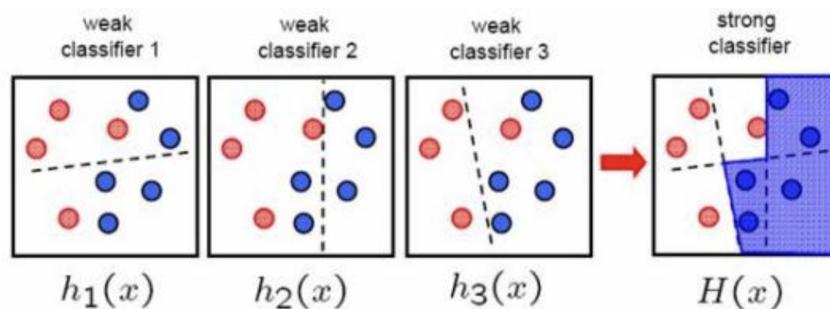
Бустинг – жадное построение  
взвешенной суммы базовых  
алгоритмов  $h_k(x)$

$$a(x) = \sum_{t=1}^T \beta_t h_t(x)$$



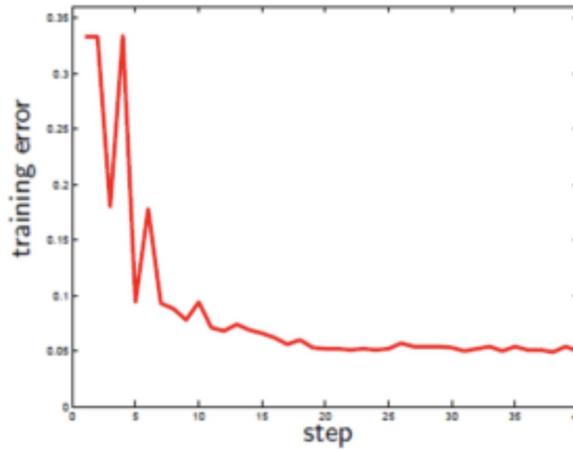
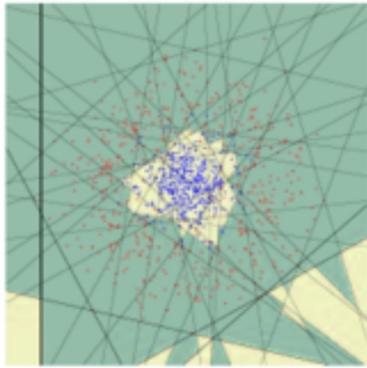
# «Слабые» алгоритмы

$h_k(x)$  – как правило, решающие деревья небольшой глубины или линейные модели



# Пример: бустинг над линейными классификаторами

$t = 40$

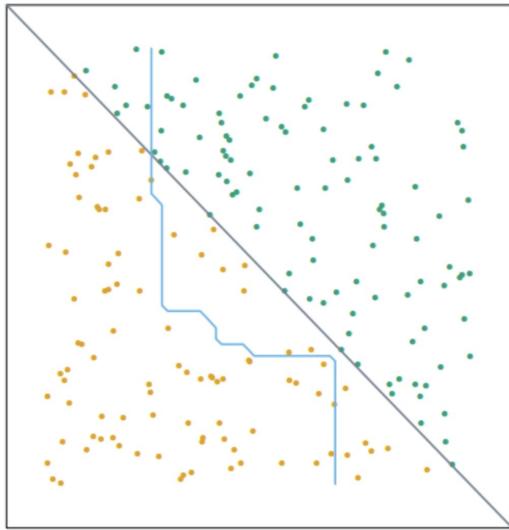


# Алгоритмы бустинга

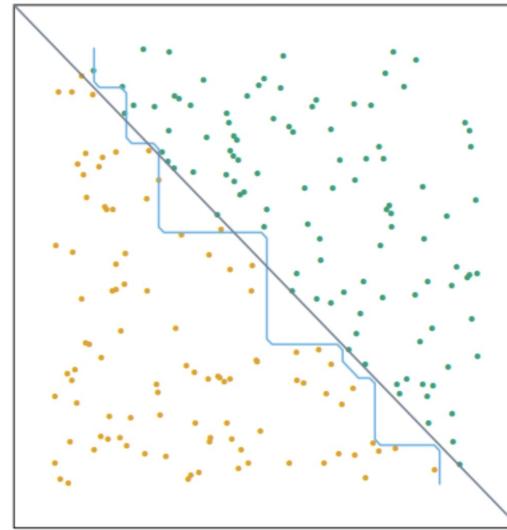
- Основные алгоритмы:
  - Градиентный бустинг
  - Адаптивный бустинг (AdaBoost)
- Вариации AdaBoost:
  - AnyBoost (произвольная функция потерь)
  - BrownBoost
  - GentleBoost
  - LogitBoost
  - ....

# Бэггинг и бустинг

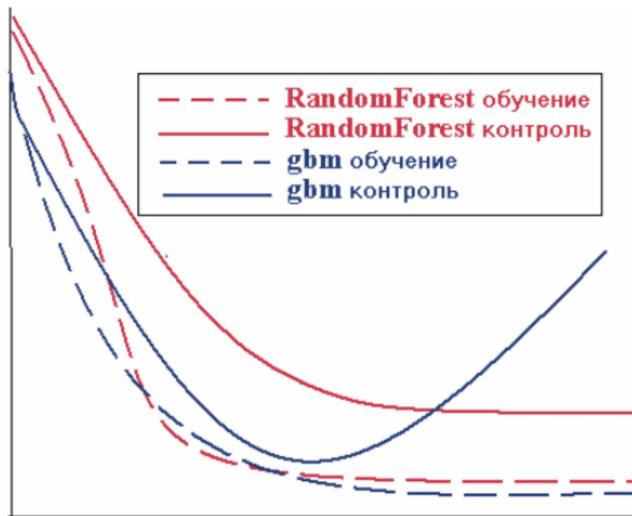
Bagged Decision Rule



Boosted Decision Rule



# Бэггинг и бустинг: переобучение



# Преимущества и недостатки бустинга

- Позволяет очень точно приблизить восстанавливаемую функцию или разделяющую поверхность классов
- Плохо интерпретируем
- Композиции могут содержать десятки тысяч базовых моделей и долго обучаться
- Переобучение на выбросах при избыточном количестве классификаторов

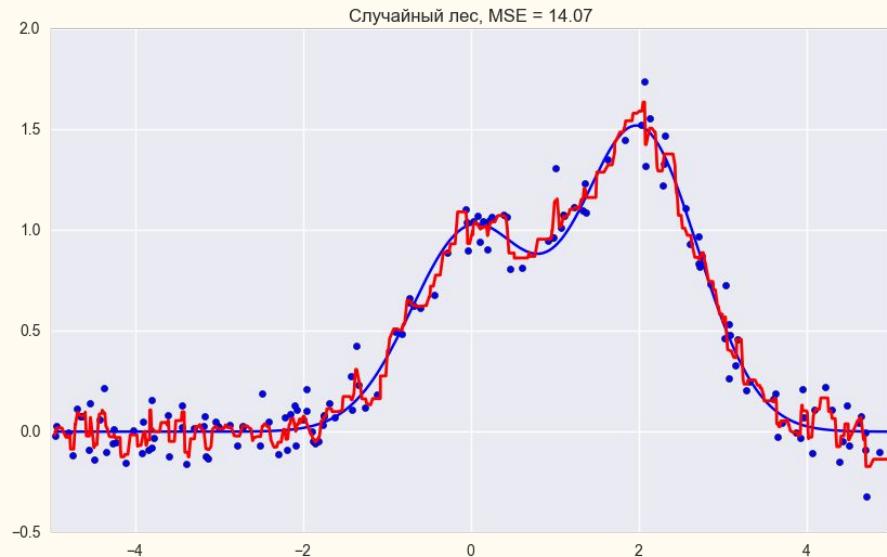
# Compositions with Decision Trees

---

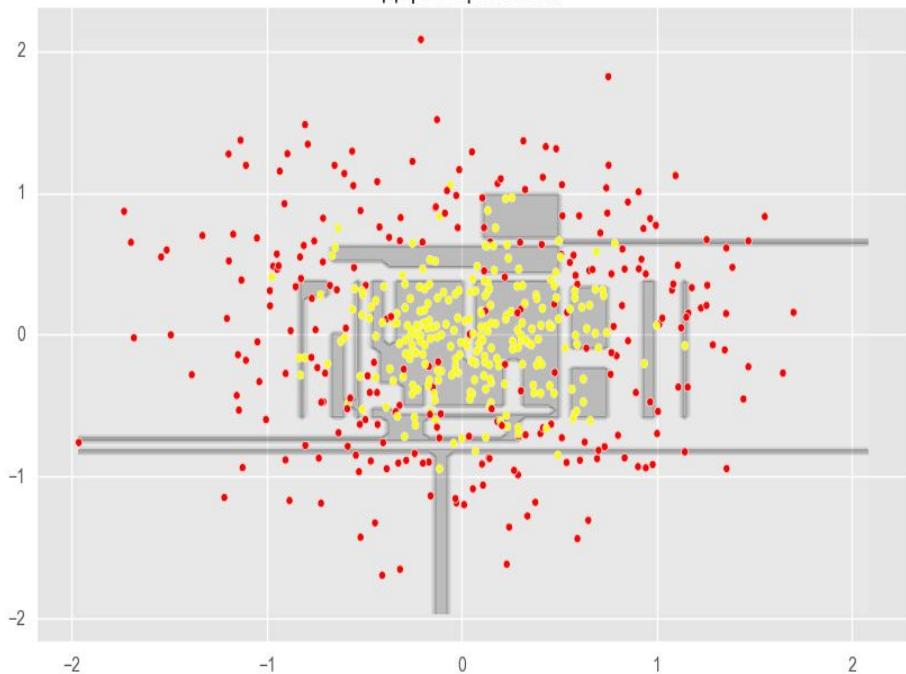
100  
010

# Random Forest

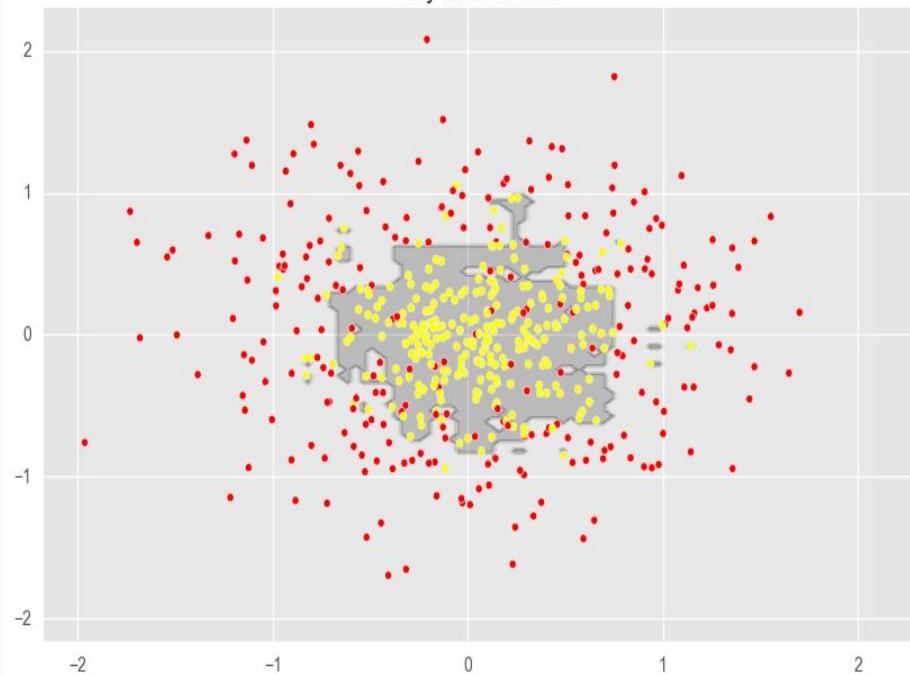
Случайный лес — это бэггинг над решающими деревьями, при обучении которых для каждого разбиения признаки выбираются из некоторого случайного подмножества признаков



Дерево решений



Случайный лес



### 3.2.4.3.1. `sklearn.ensemble.RandomForestClassifier`

```
class sklearn.ensemble. RandomForestClassifier(n_estimators=10, criterion='gini', max_depth=None,
min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto',
max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, bootstrap=True, oob_score=False,
n_jobs=1, random_state=None, verbose=0, warm_start=False, class_weight=None)           [source]
```

A random forest classifier.

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is always the same as the original input sample size but the samples are drawn with replacement if `bootstrap=True` (default).

Read more in the [User Guide](#).

**Parameters:** `n_estimators` : integer, optional (default=10)

The number of trees in the forest.

`criterion` : string, optional (default="gini")

The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "entropy" for the information gain. Note: this parameter is tree-specific.

`max_features` : int, float, string or None, optional (default="auto")

The number of features to consider when looking for the best split:

- If int, then consider `max_features` features at each split.
- If float, then `max_features` is a percentage and `int(max_features * n_features)` features are considered at each split.
- If "auto", then `max_features=sqrt(n_features)`.
- If "sqrt", then `max_features=sqrt(n_features)` (same as "auto").

## 3.2.4.3.2. `sklearn.ensemble.RandomForestRegressor`

```
class sklearn.ensemble. RandomForestRegressor (n_estimators=10, criterion='mse', max_depth=None,  
min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto',  
max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, bootstrap=True, oob_score=False,  
n_jobs=1, random_state=None, verbose=0, warm_start=False) [source]
```

A random forest regressor.

A random forest is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is always the same as the original input sample size but the samples are drawn with replacement if `bootstrap=True` (default).

Read more in the [User Guide](#).

**Parameters:** `n_estimators` : integer, optional (default=10)

The number of trees in the forest.

`criterion` : string, optional (default="mse")

The function to measure the quality of a split. Supported criteria are "mse" for the mean squared error, which is equal to variance reduction as feature selection criterion, and "mae" for the mean absolute error.

*New in version 0.18:* Mean Absolute Error (MAE) criterion.

`max_features` : int, float, string or None, optional (default="auto")

# XGBoost

XGBoost — библиотека градиентного бустинга на деревьях решений с открытым исходным кодом.



# LightGBM

LightGBM — библиотека градиентного бустинга на деревьях решений с открытым исходным кодом.



# CatBoost

CatBoost — библиотека градиентного бустинга на деревьях решений с открытым исходным кодом.



<https://tech.yandex.ru/catboost/>