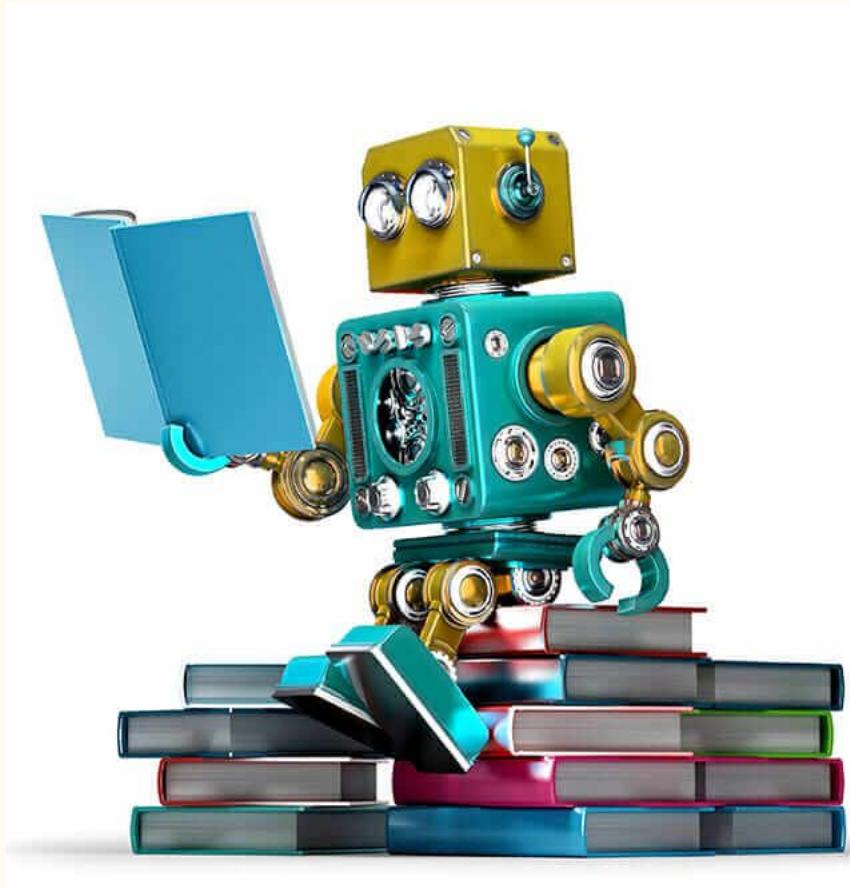


Lecture 2

Nadya Zueva
MIPT
Data Analysis dep., Yandex

План

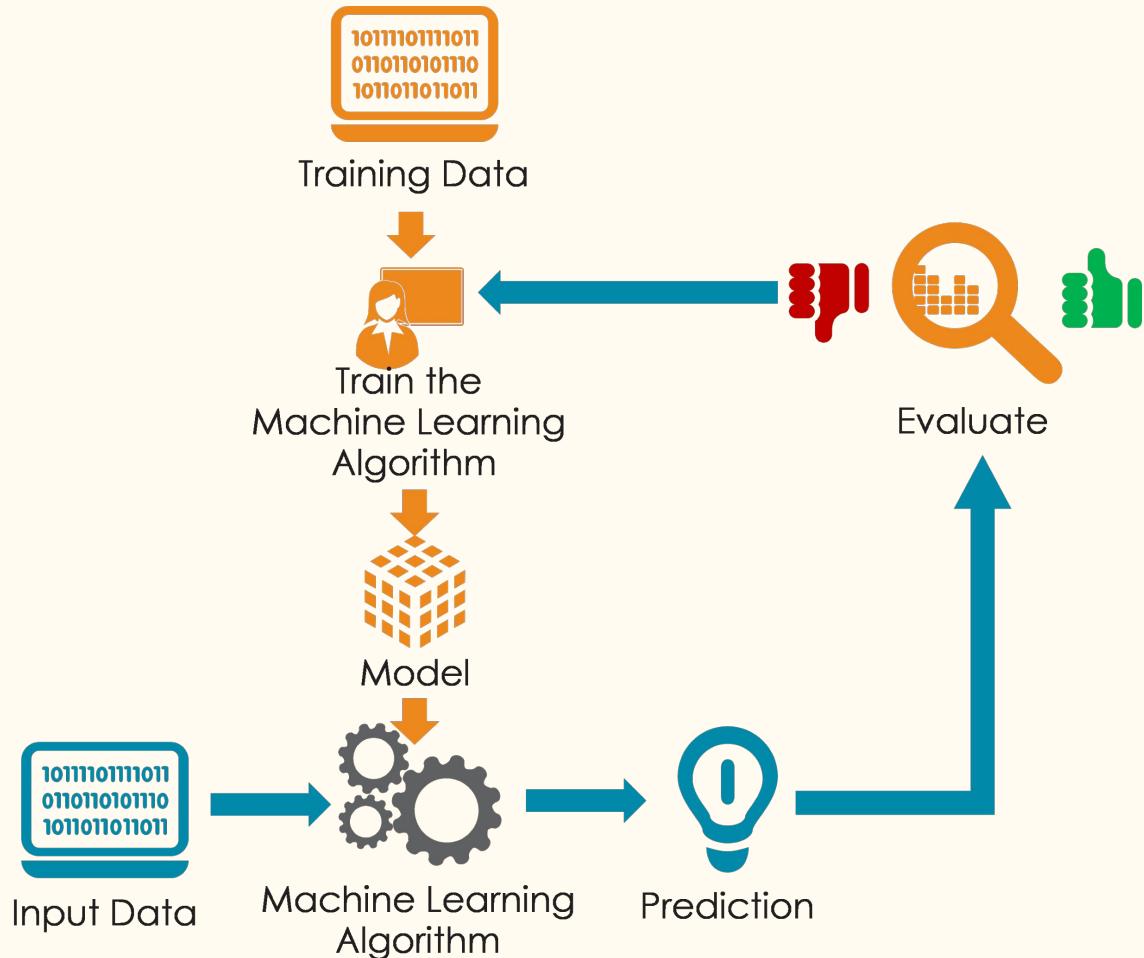
1. Основные понятия и обозначения
2. Зоопарк моделей
 - a. Обзор
 - b. Линейные модели
 - i. Линейная регрессия
 - ii. Логистическая регрессия
 - iii. SVM
 - c. Логические алгоритмы классификации
 - i. Закономерность и информативность
 - ii. Решающие деревья
3. Композиции алгоритмов
 - a. Градиентный бустинг (AnyBoost, XGBoost)
 - b. Простое голосование
 - c. Смеси алгоритмов
4. Scikit-Learn



ML Language

напоминание

100
010



X – множество **объектов**

Y – множество **допустимых ответов**

y^* – целевая функция, $y^*: X \rightarrow Y$, $y_i = y^*(x_i)$ известны только на **конечном** подмножестве объектов x_1, \dots, x_m из X

Пары (x_i, y_i) – прецеденты

Совокупность пар таких пар при i из $1, \dots, m$ – **обучающая выборка** (X_{train})

a – **решающая функция** (алгоритм), которая любому объекту из X ставит в соответствие допустимый ответ из Y и приближает целевую функцию y^*

X_{test} – **выборка прецедентов** для тестирования построенного алгоритма a

Для решения задачи обучения по прецедентам в первую очередь фиксируется восстанавливаемой зависимости.

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket
1	0	3	Braund, Mr. Owen Harris	male	22	1	0	A/5 21171
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Thayer)	female	38	1	0	PC 17599
3	1	3	Heikkinen, Miss. Laina	female	26	0	0	STON/O2. 310128
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35	1	0	113803
5	0	3	Allen, Mr. William Henry	male	35	0	0	373450
6	0	3	Moran, Mr. James	male		0	0	330877
7	0	1	McCarthy, Mr. Timothy J	male	54	0	0	17463
8	0	3	Palsson, Master. Gosta Leonard	male	2	3	1	349909

X

 y^*

features

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket
1	0	3	Braund, Mr. Owen Harris	male	22	1	0	A/5 21171
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Thayer)	female	38	1	0	PC 17599
3	1	3	Heikkinen, Miss. Laina	female	26	0	0	STON/O2. 310128
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35	1	0	113803
5	0	3	Allen, Mr. William Henry	male	35	0	0	373450
6	0	3	Moran, Mr. James	male		0	0	330877
7	0	1	McCarthy, Mr. Timothy J	male	54	0	0	17463
8	0	3	Palsson, Master. Gosta Leonard	male	2	3	1	349909

 $Y=\{0,1\}$

Признак (feature) f объекта x — это результат измерения некоторой характеристики объекта. Формально признаком называется отображение $f : X \rightarrow D_f$, где D_f — множество допустимых значений признака. В частности, любой алгоритм $a : X \rightarrow Y$ также можно рассматривать как признак

Пусть дан набор признаков $f_1(x), \dots, f_n(x)$.

Признаковое описание объекта x — вектор (одномерный массив) (f_1, \dots, f_n) . Совокупность признаковых описаний всех объектов выборки длины m , записанную в виде таблицы размера mp , называют матрицей объектов-признаков.

CATEGORY

RANGE

Excellent
(28% of people)

750 - 850

Good
(10% of people)

700 - 749

Fair
(16% of people)

650 - 699

Poor
(32% of people)

550 - 649

Very Poor
(14% of people)

350 - 549



1. Пол
2. Доход семьи
3. Опыт работы
4. Кредитная история
5. Возраст
6. Уровень образования
7. Профессия
8. Место проживания
9. Недвижимость в собственности

X

y*

features

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket
1	0	3	Braund, Mr. Owen Harris	male	22	1	0	A/5 21171
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Thayer)	female	38	1	0	PC 17599
3	1	3	Heikkinen, Miss. Laina	female	26	0	0	STON/O2. 310128
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35	1	0	113803
5	0	3	Allen, Mr. William Henry	male	35	0	0	373450
6	0	3	Moran, Mr. James	male		0	0	330877
7	0	1	McCarthy, Mr. Timothy J	male	54	0	0	17463
8	0	3	Palsson, Master. Gosta Leonard	male	2	3	1	349909

X_train=Object_ids+Features+y*

X

features

PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292
893	3	Wilkes, Mrs. James (Ellen Needs)	female	47	1	0	363272	7
894	2	Myles, Mr. Thomas Francis	male	62	0	0	240276	9.6875
895	3	Wirz, Mr. Albert	male	27	0	0	315154	8.6625
896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22	1	1	3101298	12.2875
897	3	Svensson, Mr. Johan Cervin	male	14	0	0	7538	9.225
898	3	Connolly, Miss. Kate	female	30	0	0	330972	7.6292
899	2	Caldwell, Mr. Albert Francis	male	26	1	1	248738	29
900	3	Abrahim, Mrs. Joseph (Sophie Halaut Easu)	female	18	0	0	2657	7.2292
901	3	Davies, Mr. John Samuel	male	21	2	0	A/4 48871	24.15

X_test=Object_ids+Features

По выборке X_{train} построить решающую функцию (*decisionfunction*)
 $a : X \rightarrow Y$, которая приближает целевую функцию y^* , причём не
только на объектах ***обучающей выборки, но и на всём***
множестве X .

Решающая функция a должна быть вычислимой.

Как строится функция а?

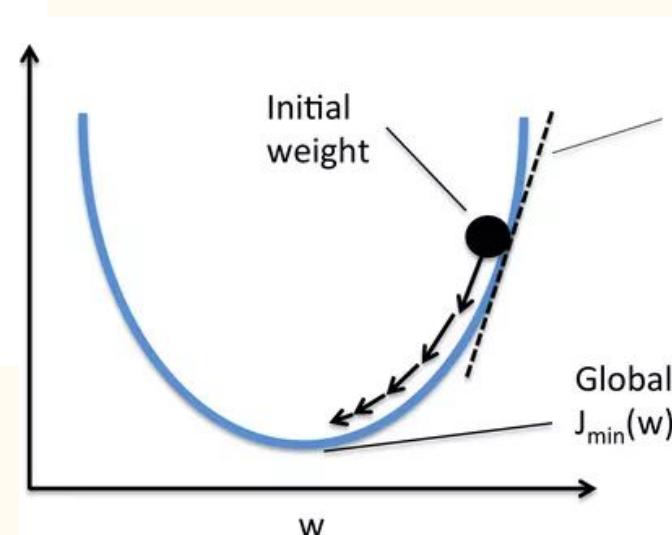
Обучающая выборка — выборка, по которой производится настройка (оптимизация параметров) модели зависимости.

Тестовая выборка — выборка, по которой оценивается качество построенной модели.

Функционал качества (обучение с учителем) — определяется как средняя ошибка ответов, выданных алгоритмом, по всем объектам выборки.

$$L(\hat{y}, y) = I(\hat{y} \neq y),$$

$$\text{logloss} = -\frac{1}{l} \cdot \sum_{i=1}^l (y_i \cdot \log(\hat{y}_i) + (1 - y_i) \cdot \log(1 - \hat{y}_i))$$



Зоопарк
моделей

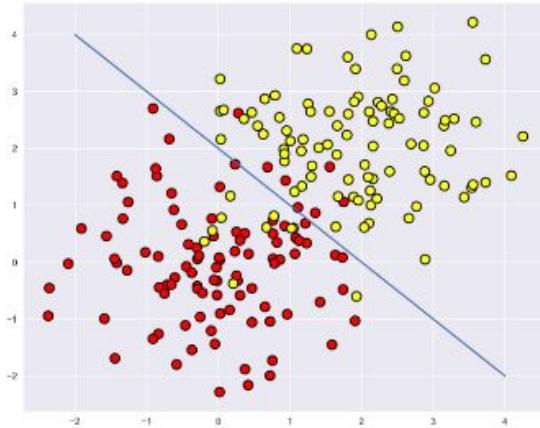
100
010

TODO

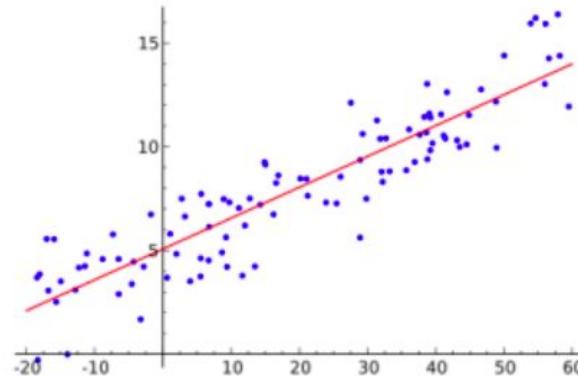
Параметрические непараметрические

Байесовские небайесовские

Регрессия VS Классификация



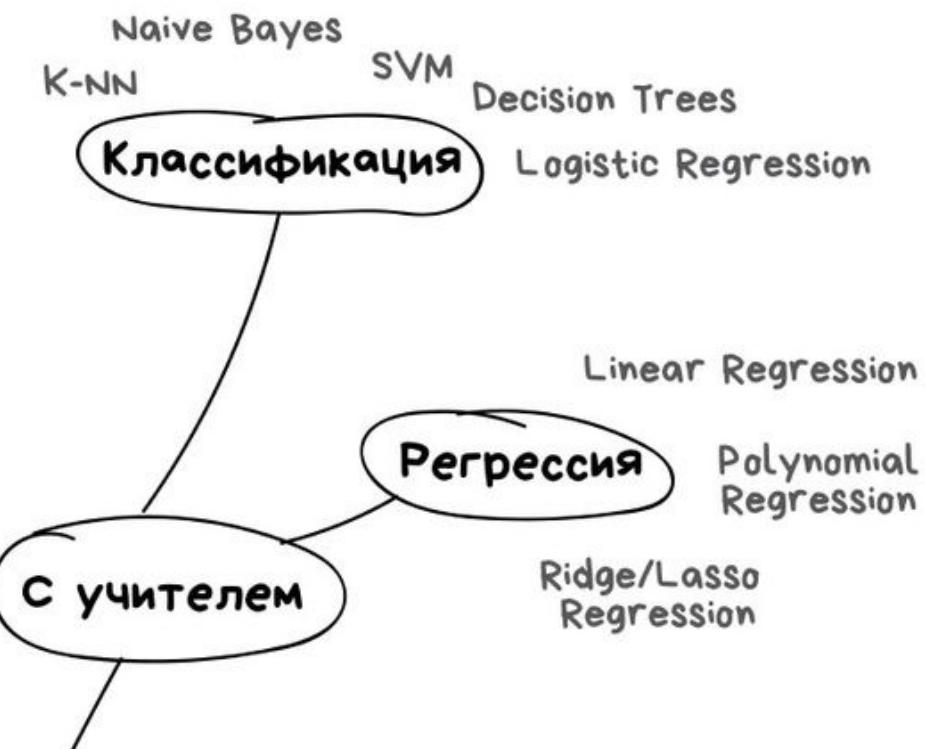
$$Y = \{1, \dots, N\}$$



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

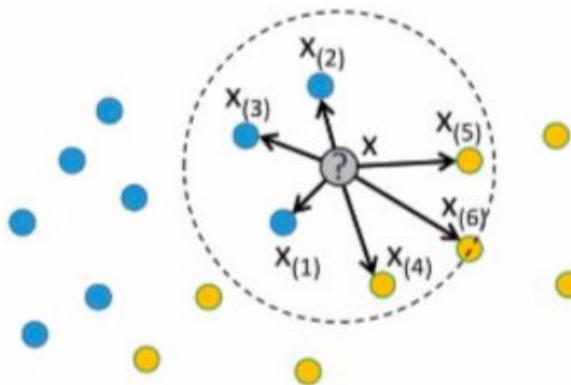






K-nearest neighbours

Пример классификации ($k = 6$):



Наивный байесовский классификатор

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

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

метка класса

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

объект для классификации

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

```
from __future__ import division
from collections import defaultdict
from math import log

def train(samples):
    classes, freq = defaultdict(lambda:0), defaultdict(lambda:0)
    for feats, label in samples:
        classes[label] += 1                      # count classes frequencies
        for feat in feats:
            freq[label, feat] += 1                # count features frequencies

    for label, feat in freq:                     # normalize features frequencies
        freq[label, feat] /= classes[label]
    for c in classes:                          # normalize classes frequencies
        classes[c] /= len(samples)

    return classes, freq                       # return P(C) and P(O|C)
```

подсчет
параметров

```
def classify(classifier, feats):
    classes, prob = classifier
    return min(classes.keys(),           # calculate argmin(-log(C|O))
               key = lambda cl: -log(classes[cl]) + \
                           sum(-log(prob.get((cl,feat), 10**(-7))) for feat in feats))
```

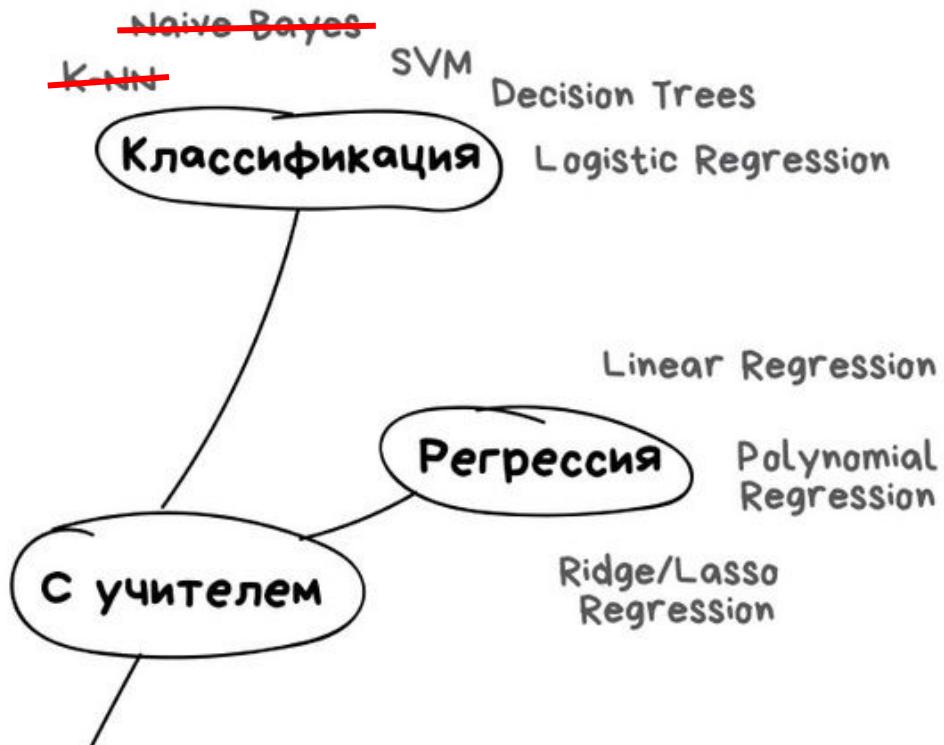
расчет
формулы

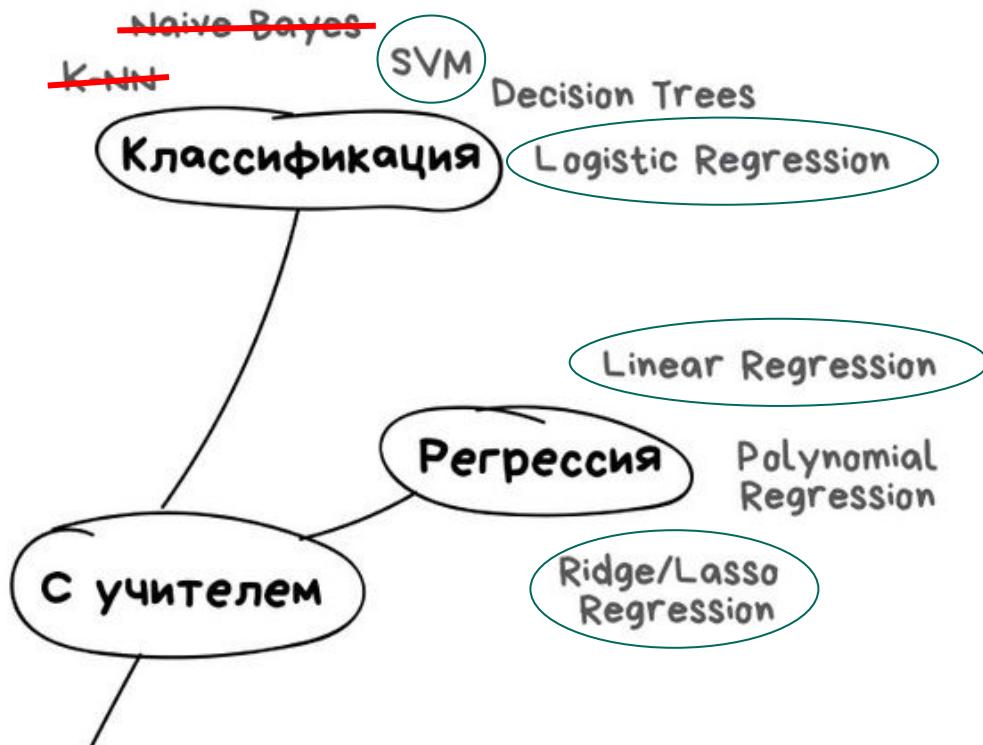
`GaussianNB` implements the Gaussian Naive Bayes algorithm for classification. The likelihood of the features is assumed to be Gaussian:

$$P(x_i | y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

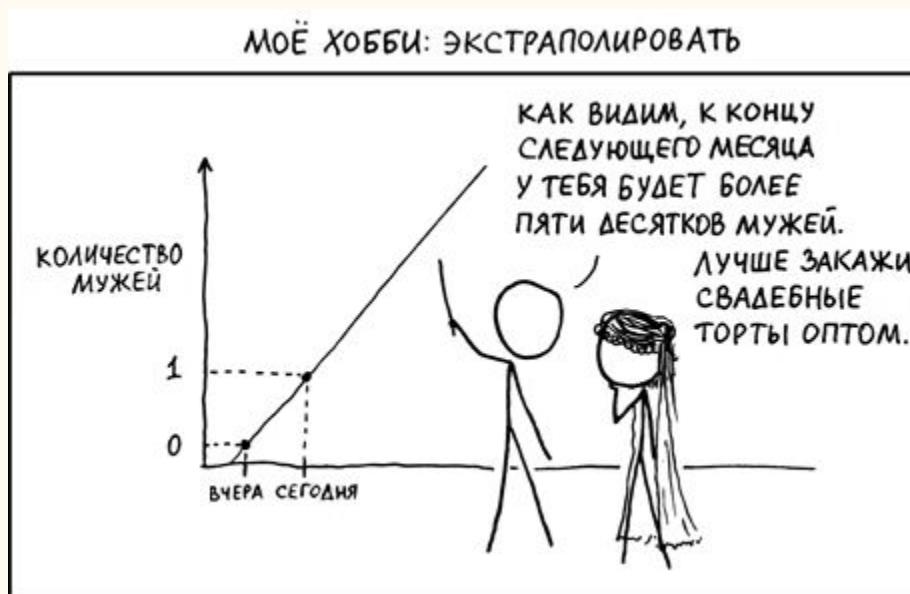
The parameters σ_y and μ_y are estimated using maximum likelihood.

```
>>> from sklearn import datasets
>>> iris = datasets.load_iris()
>>> from sklearn.naive_bayes import GaussianNB
>>> gnb = GaussianNB()
>>> y_pred = gnb.fit(iris.data, iris.target).predict(iris.data)
>>> print("Number of mislabeled points out of a total %d points : %d"
...      % (iris.data.shape[0],(iris.target != y_pred).sum()))
Number of mislabeled points out of a total 150 points : 6
```





Линейная регрессия

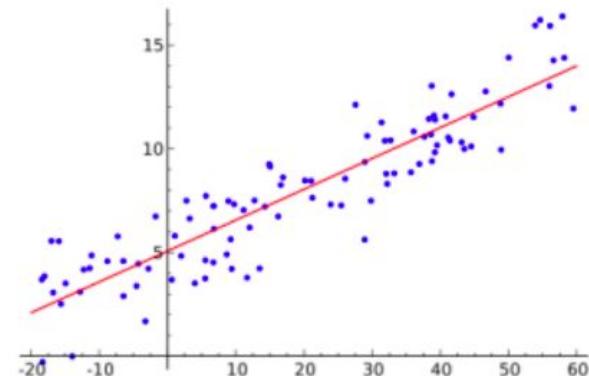


Линейная регрессия

$$\vec{y} = X\vec{w} + \epsilon,$$

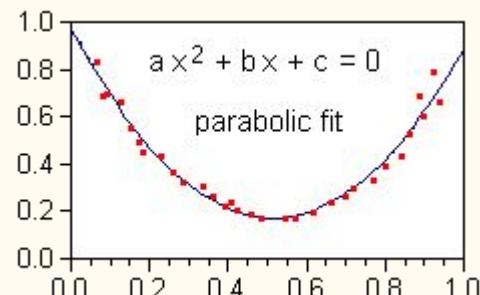
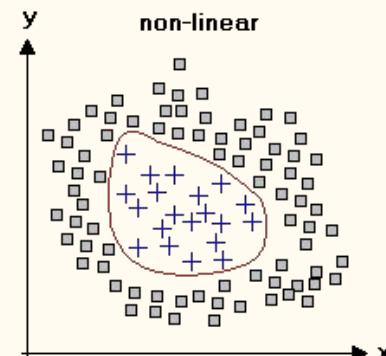
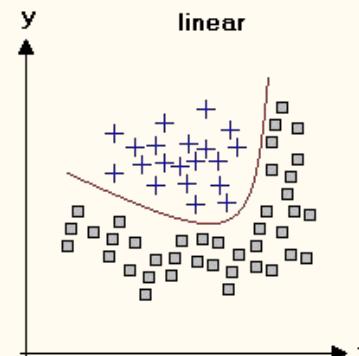
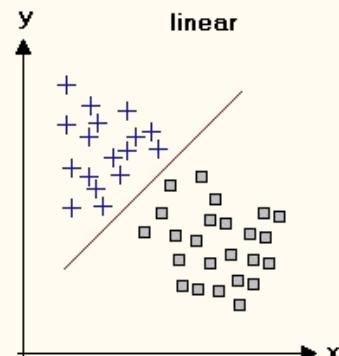
где

- $\vec{y} \in \mathbb{R}^n$ – объясняемая (или целевая) переменная;
- w – вектор параметров модели (в машинном обучении эти параметры часто называют весами);
- X – матрица наблюдений и признаков размерности n строк на $m + 1$ столбцов (включая фиктивную единичную колонку слева) с полным рангом по столбцам: $\text{rank}(X) = m + 1$;
- ϵ – случайная переменная, соответствующая случайной, непрогнозируемой ошибке модели.

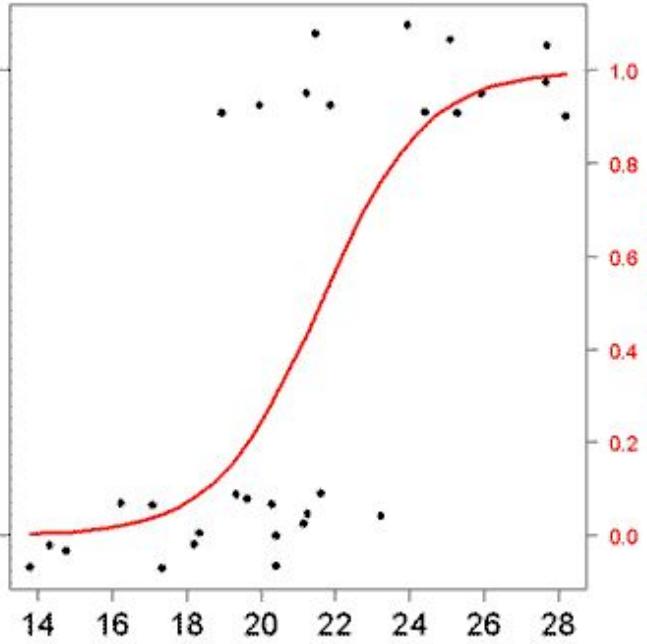


Линейная модель

$$y_i = \sum_{j=0}^m w_j X_{ij} + \epsilon_i$$



$$y_i = \pm \sum_{j=0}^m w_j X_{ij} + \epsilon_i$$



Возвращает вероятности
принадлежности к классам. При
этом каждое отдельное правило --
линейный классификатор

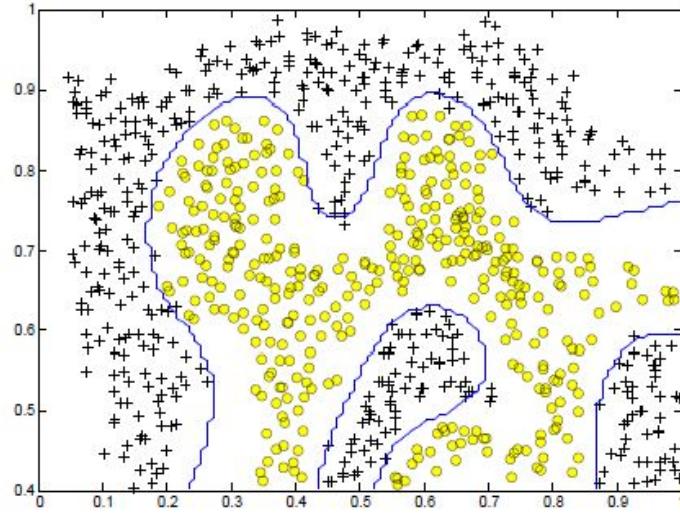


Figure 5: SVM (Gaussian Kernel) Decision Boundary (Example Dataset 2)

Штрафует за приближение к
границе

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

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

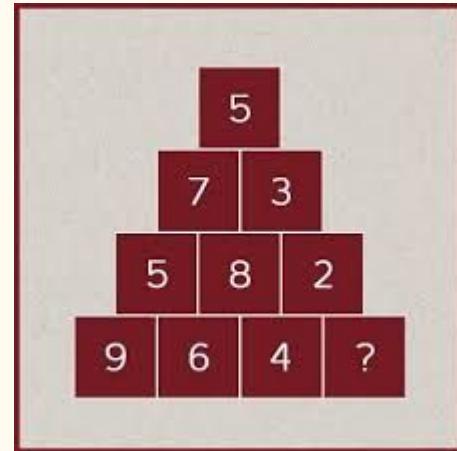
$$n_c(R) = \#\{x_i : R(x_i)=1 \text{ и } y_i \neq c\} \rightarrow \min;$$



1	2	3	1	
1	2		4	5

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.
2. Закономерность можно понимать как...
 - a. Высокоинформативный и интерпретируемый признак
 - b. Одноклассовый классификатор с отказами



Примеры закономерностей

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):

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

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· ℓ	IGain· ℓ	$\sqrt{p}-\sqrt{n}$
50	0	50	50	0.25	50	22.65	23.70	7.07
100	50	50	-150	0	1.96	2.33	1.98	2.93
50	9	41	5	0.16	5	7.87	7.94	4.07
5	0	5	5	0.03	5	2.04	3.04	2.24
100	0	100	100	0.5	100	52.18	53.32	10.0
140	20	120	40	0.5	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:

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

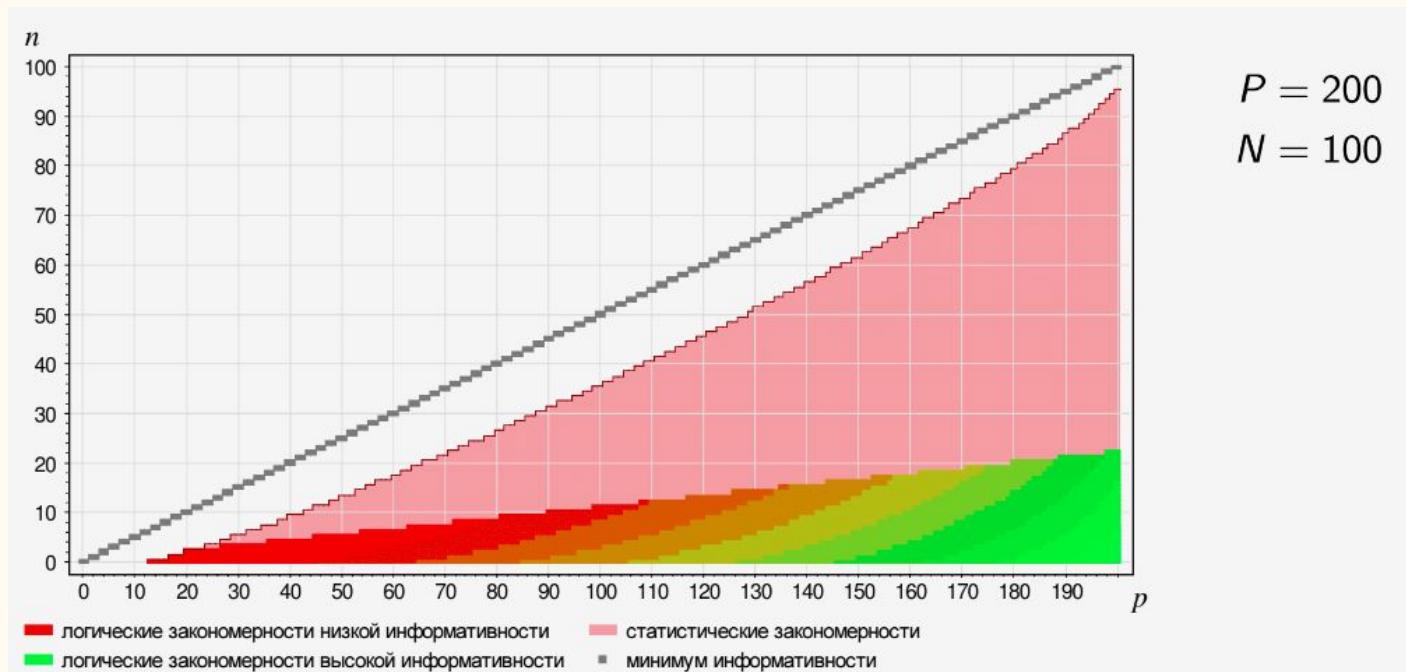
где $h(q) = -q \log_2 q - (1-q) \log_2(1-q)$

- критерий Джини (Gini impurity):

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

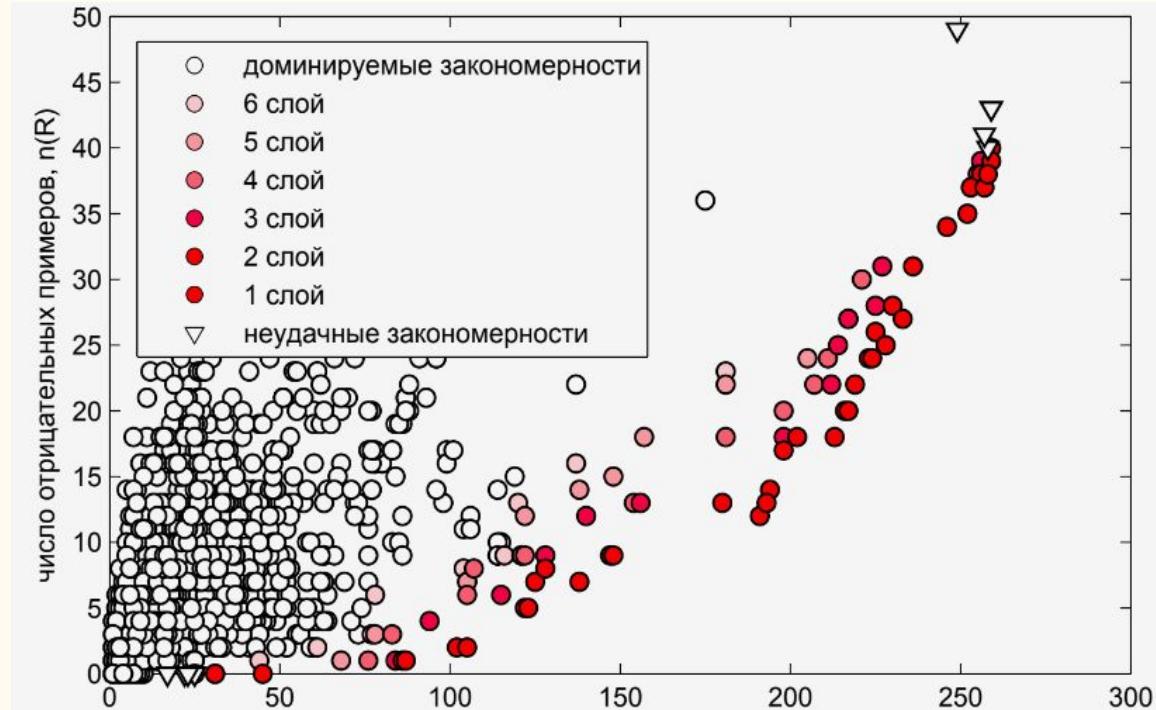
“загрязненность” распределения

Неслучайность = закономерность?



Парето-фронт

Множество
неулучшаемых
закономерностей

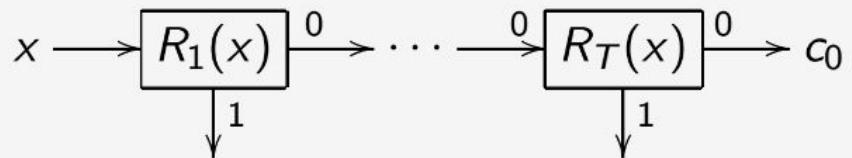


Композиции закономерностей

- Взвешенное (или простое) голосование

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

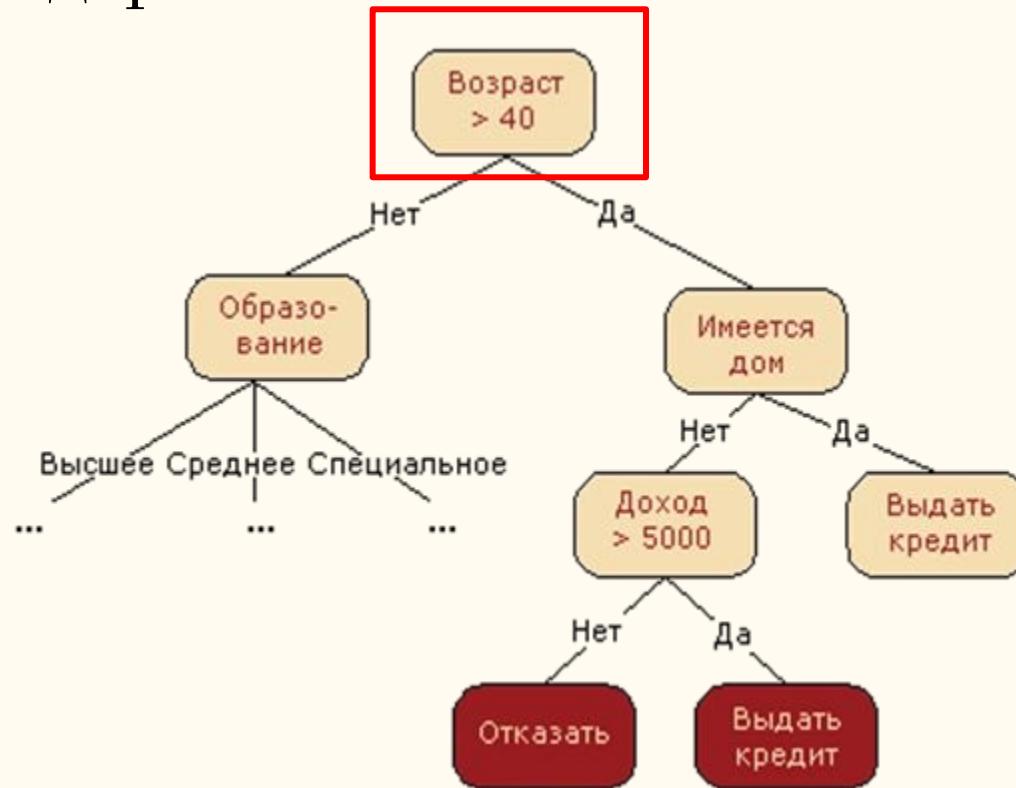
- Решающий список (по старшинству)



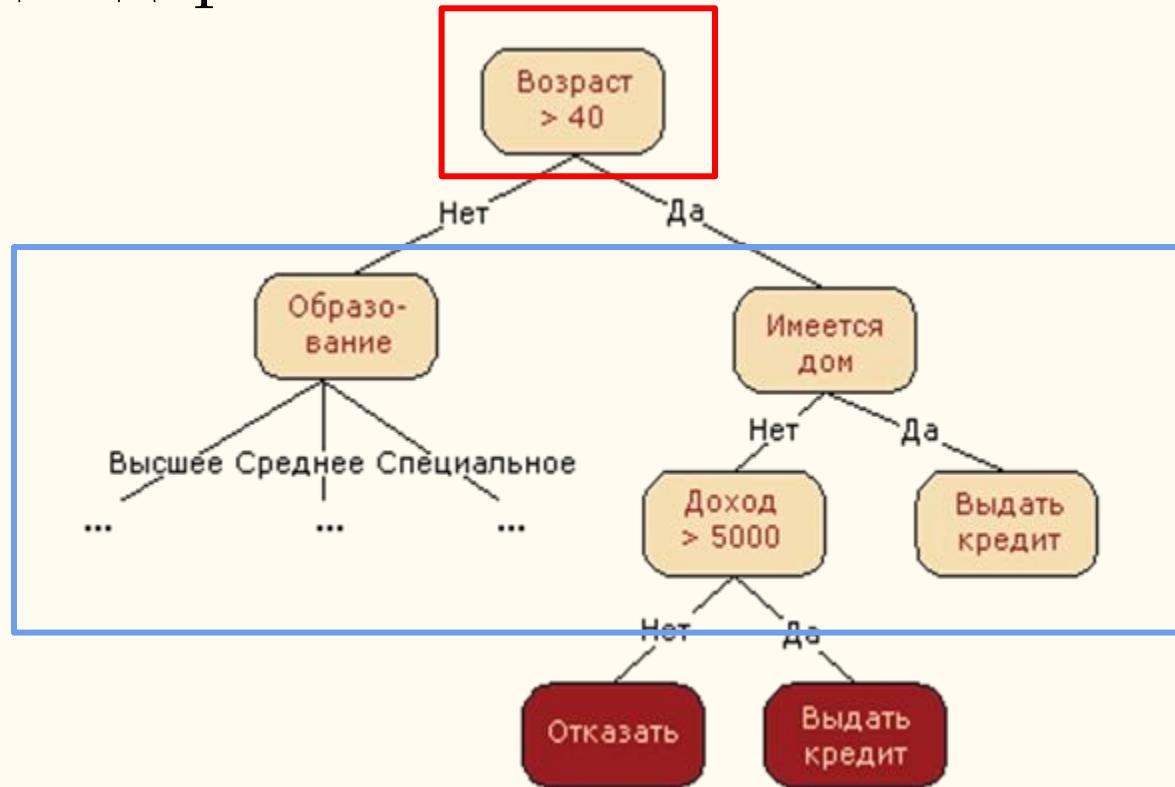
Решающее дерево



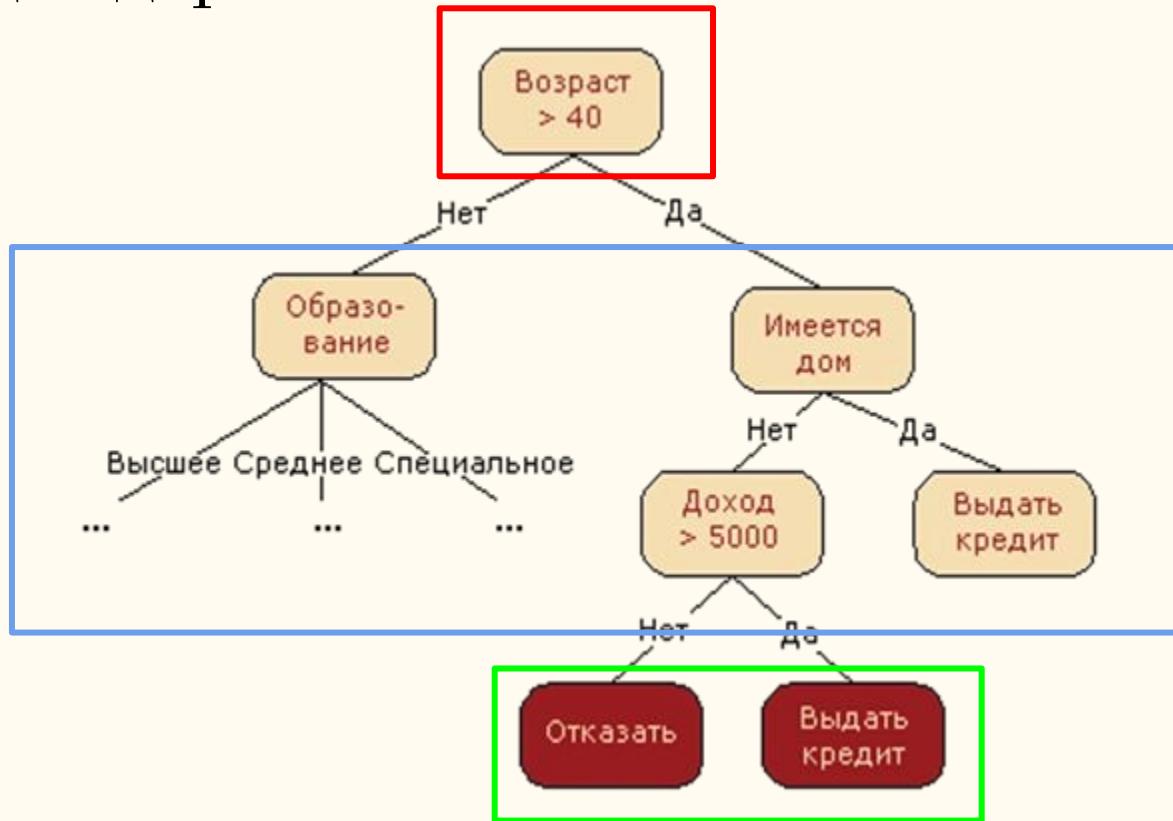
Решающее дерево



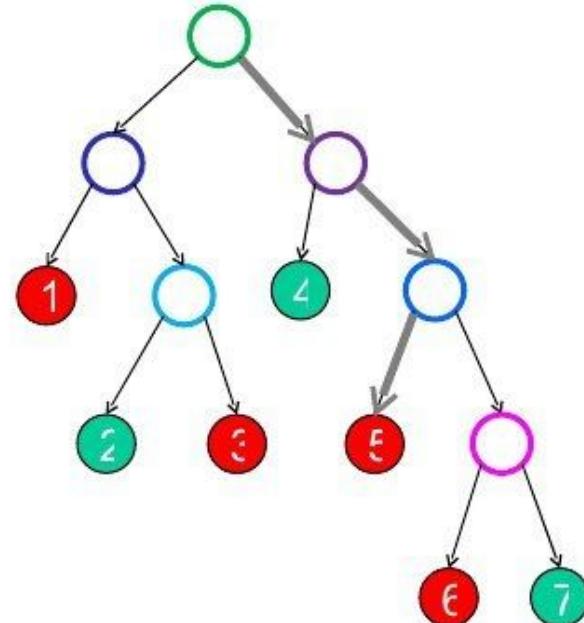
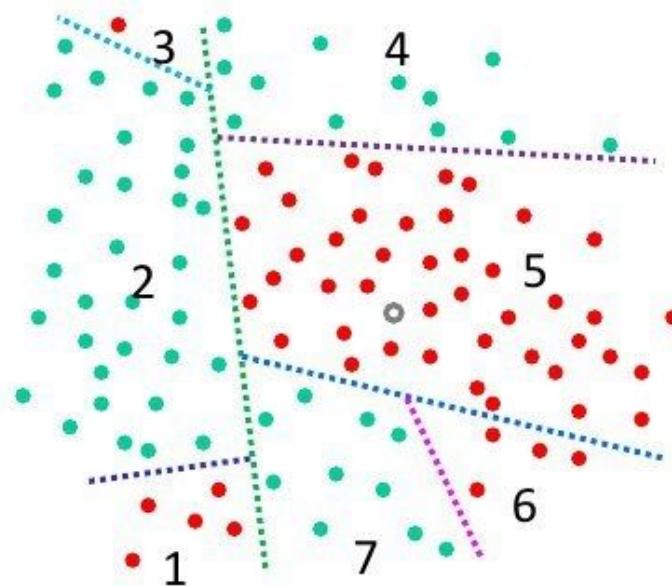
Решающее дерево



Решающее дерево

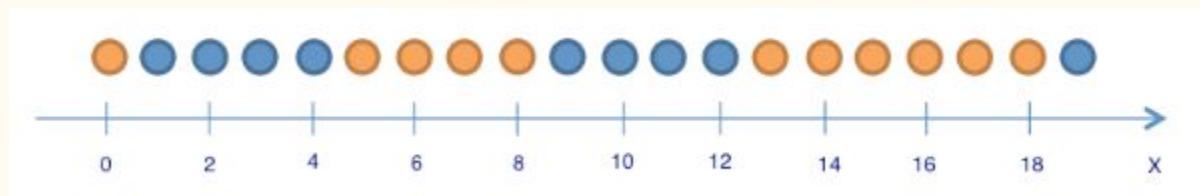


Решающее дерево



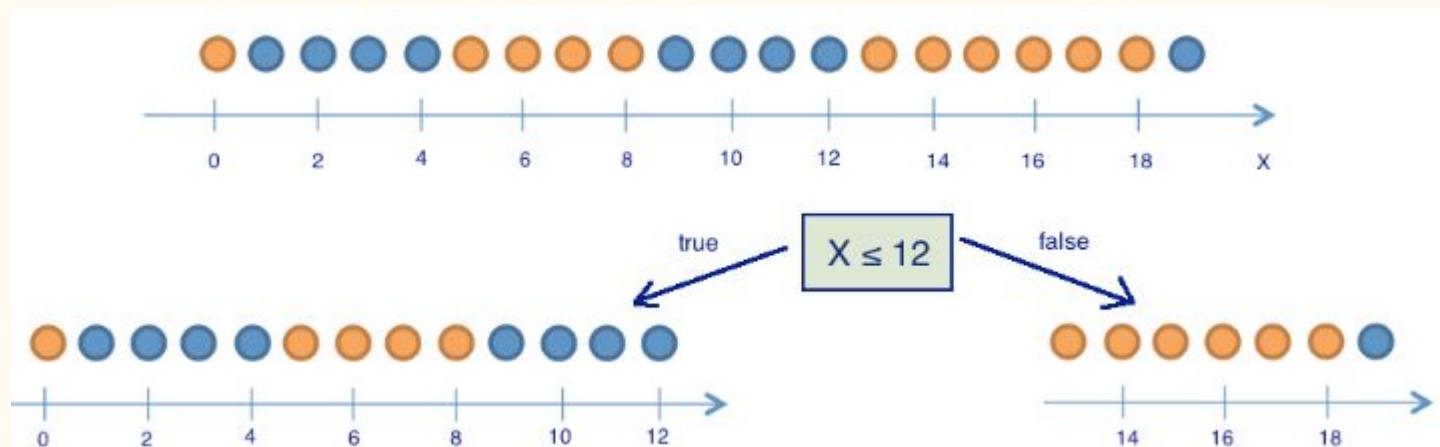
Обучение решающего дерева

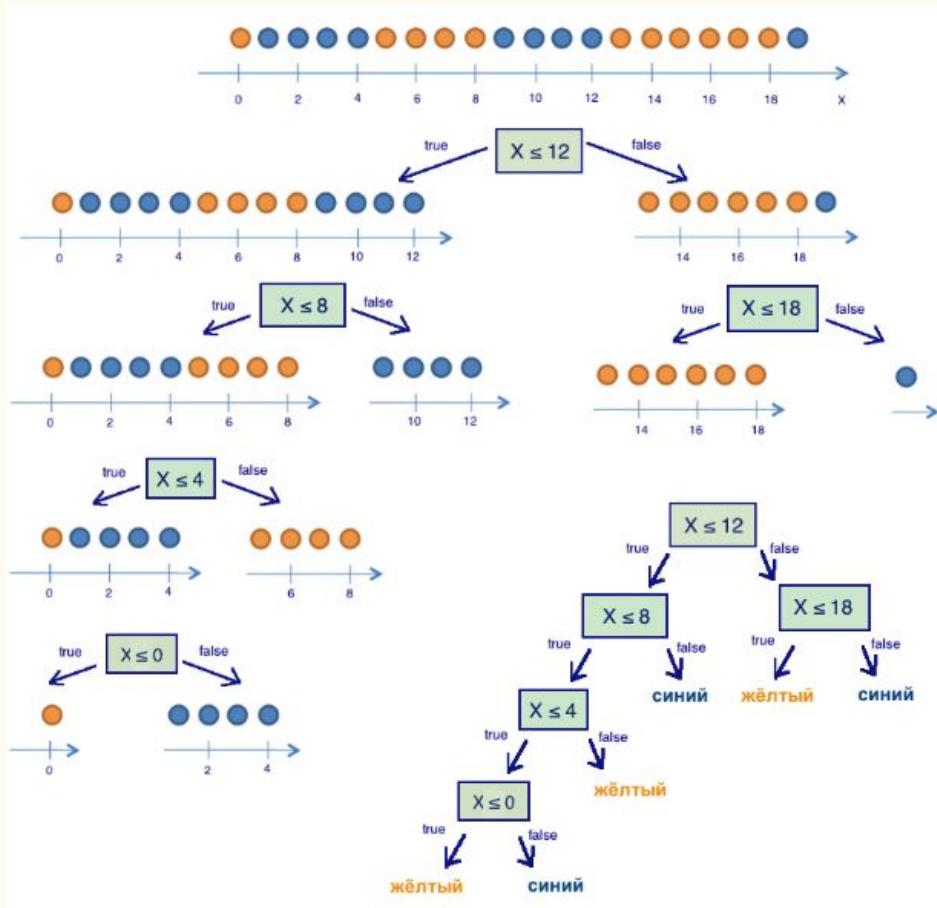
1. Стремимся к уменьшению энтропии (хаоса) в системе на каждом шаге



Обучение решающего дерева

- Стремимся к уменьшению энтропии (хаоса) в системе на каждом шаге



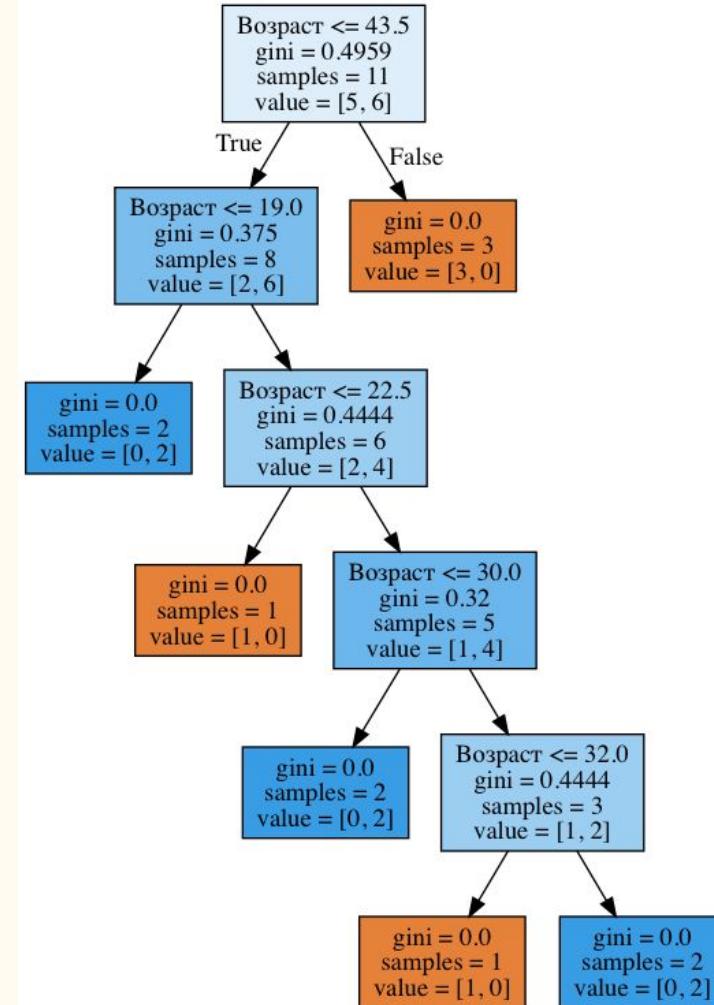


```

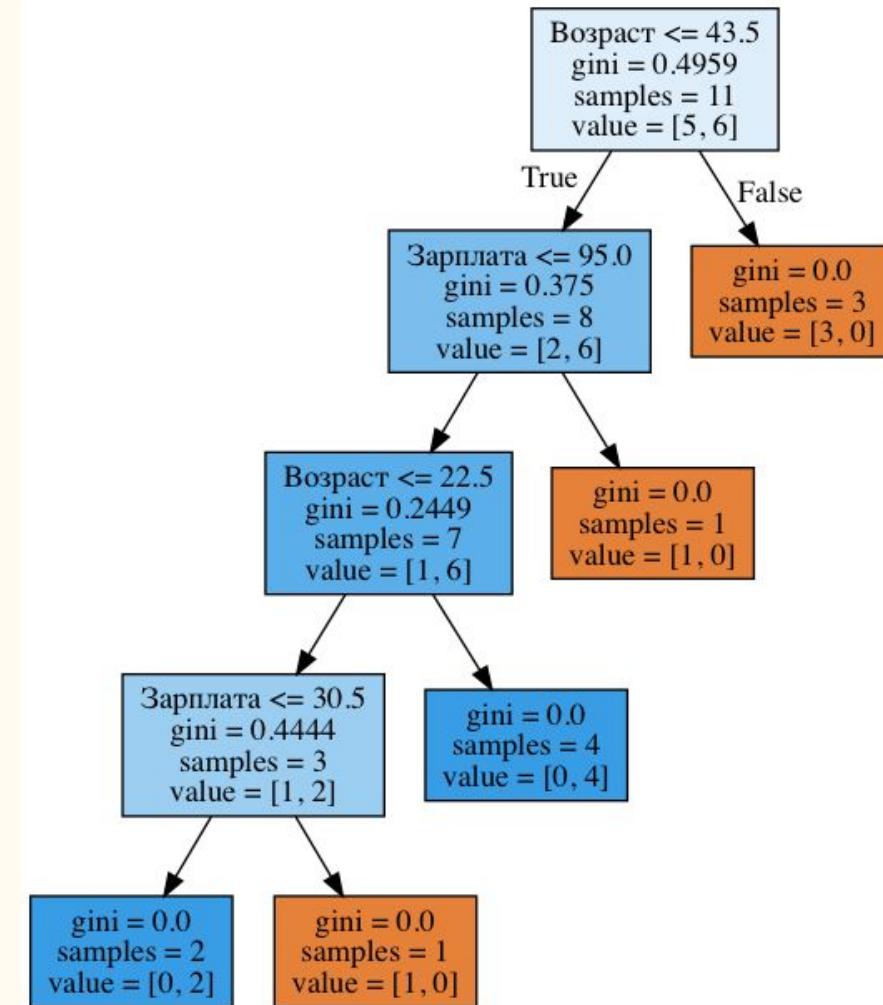
def build(L):
    create node t
    if the stopping criterion is True:
        assign a predictive model to t
    else:
        Find the best binary split L = L_left + L_right
        t.left = build(L_left)
        t.right = build(L_right)
    return t

```

	Возраст	Невозврат кредита
0	17	1
2	18	1
3	20	0
7	25	1
8	29	1
9	31	0
10	33	1
4	38	1
5	49	0
6	55	0
1	64	0



	Возраст	Зарплата	Невозврат кредита
0	17	25	1
1	64	80	0
2	18	22	1
3	20	36	0
4	38	37	1
5	49	59	0
6	55	74	0
7	25	70	1
8	29	33	1
9	31	102	0
10	33	88	1



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.

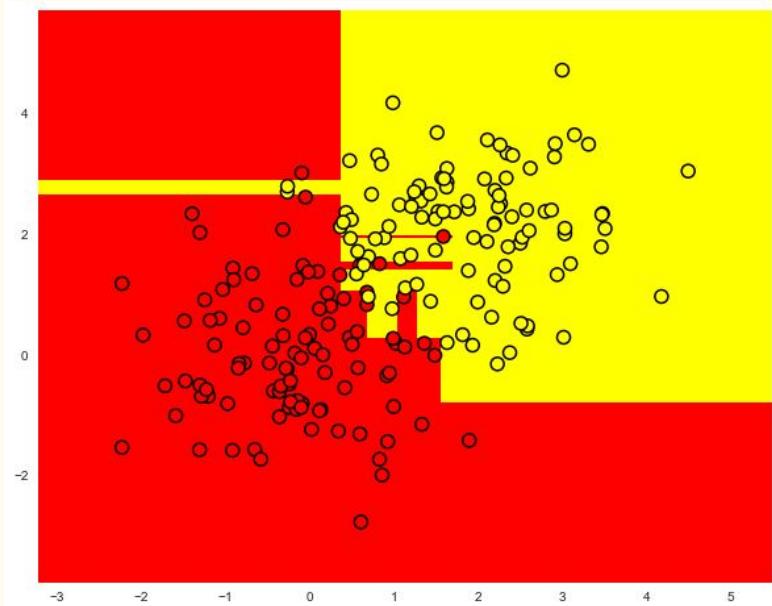
max_depth : int or None, optional (default=None)

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.

Проблема переобучения

Пути решения:

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



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

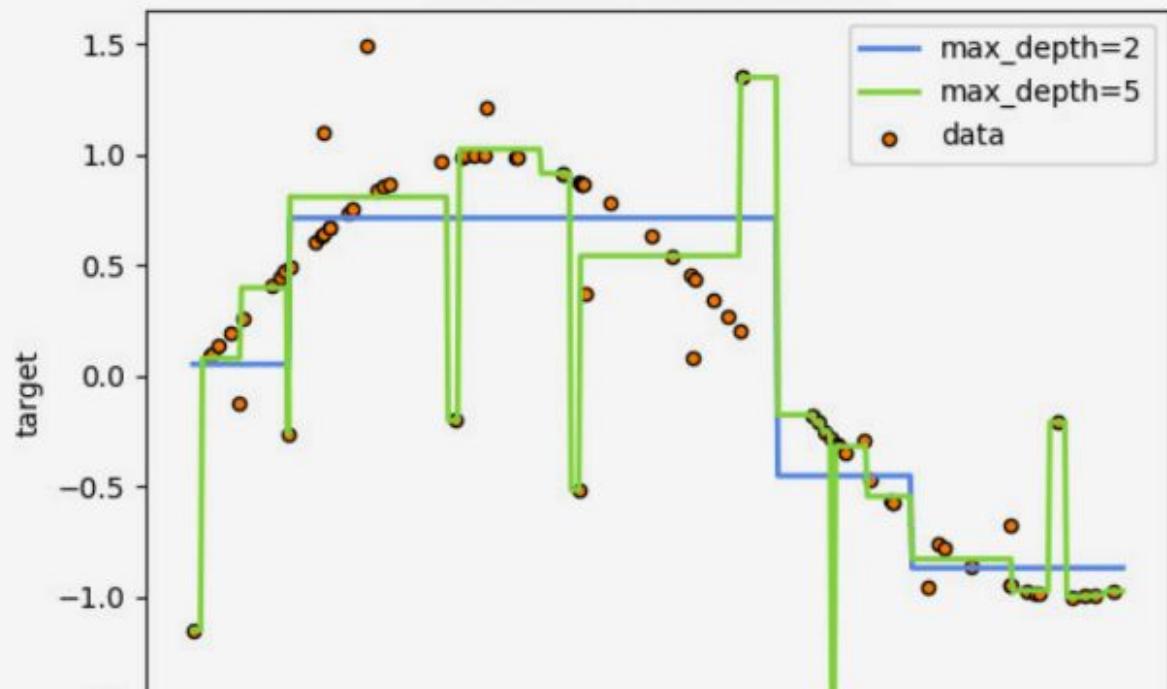
$$\Phi(U) = \min_{y \in Y} \frac{1}{|U|} \sum_{x_i \in U} (y - y_i)^2$$

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

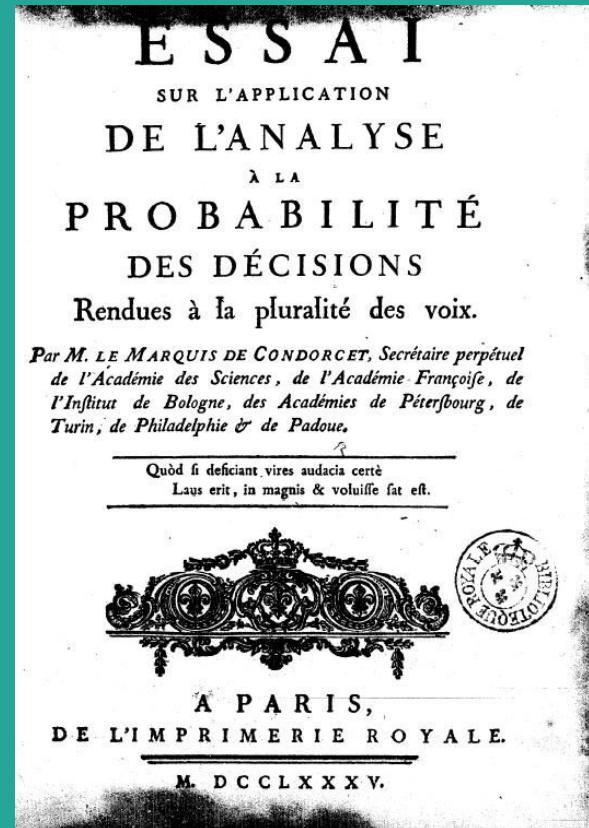


Композиции алгоритмов

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.



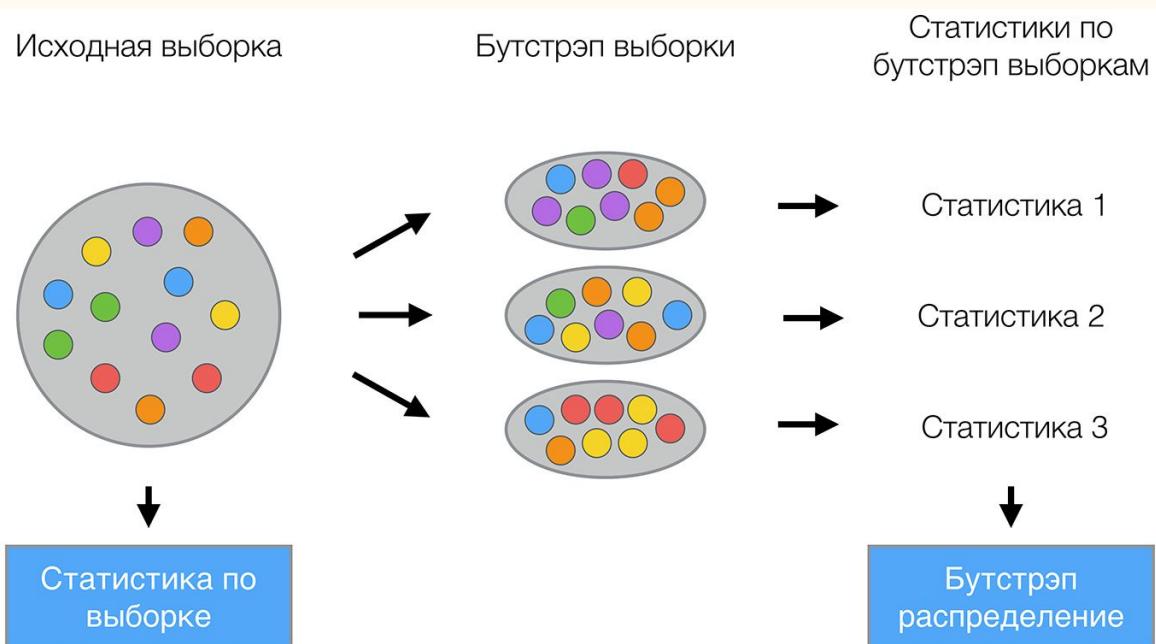
принцип Кондорсе, 1784

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



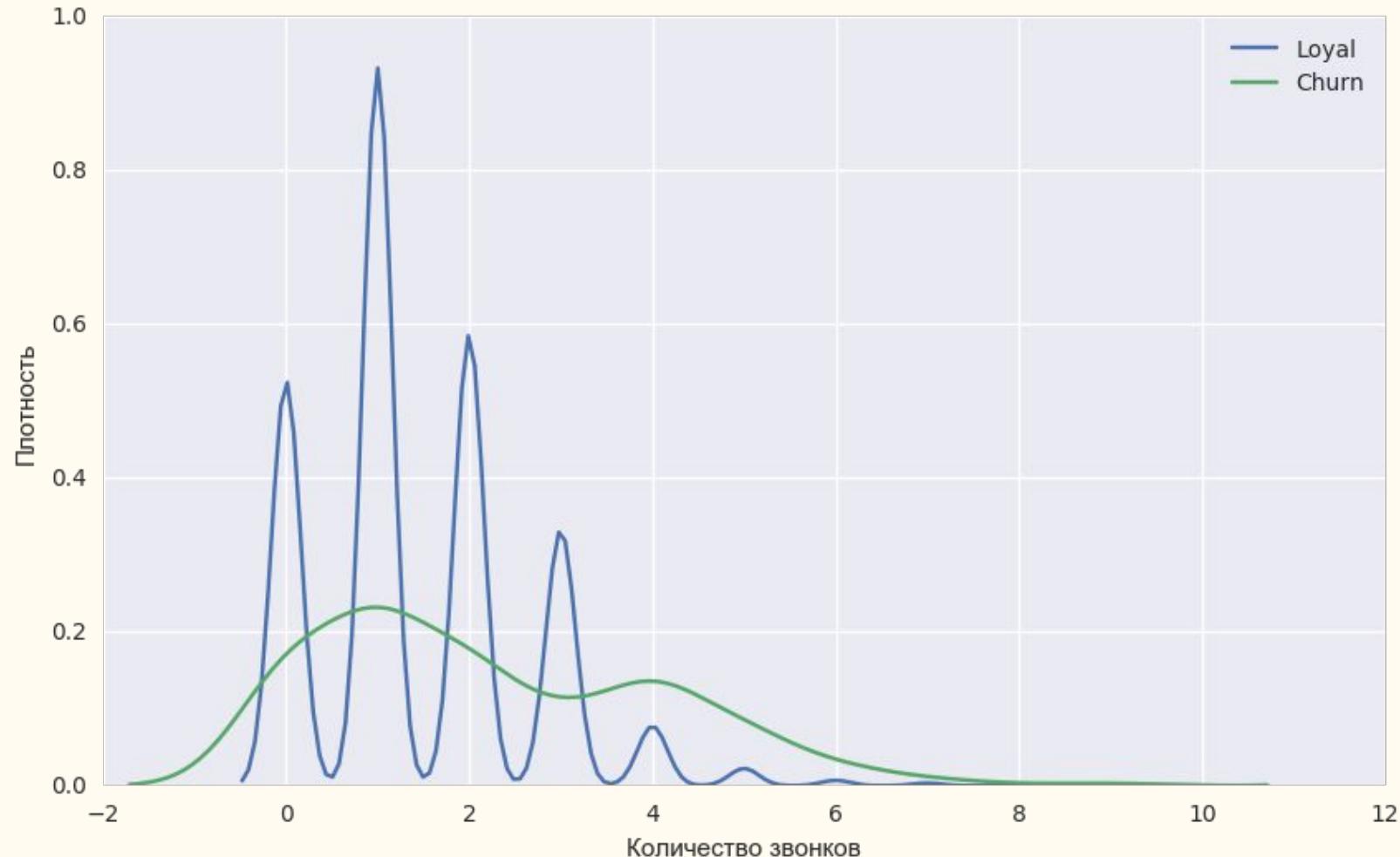
Бинарная классификация оттока клиентов

```
import pandas as pd
from matplotlib import pyplot as plt
plt.style.use('ggplot')
plt.rcParams['figure.figsize'] = 10, 6
import seaborn as sns
%matplotlib inline

telecom_data = pd.read_csv('data/telecom_churn.csv')

fig = sns.kdeplot(telecom_data[telecom_data['Churn'] == False]['Customer service calls'], label = 'Loyal')
fig = sns.kdeplot(telecom_data[telecom_data['Churn'] == True]['Customer service calls'], label = 'Churn')
fig.set(xlabel='Количество звонков', ylabel='Плотность')
plt.show()
```

Мало данных, одна из главных фич — количество звонков в сервисный центр



Оценим, сколько в среднем делает звонков каждая из групп.

Данных мало, поэтому искать среднее не совсем правильно, применим **bootstrap** и сделаем интервальную оценку среднего

```
import numpy as np
def get_bootstrap_samples(data, n_samples):
    # функция для генерации подвыборок с помощью бутстрэпа
    indices = np.random.randint(0, len(data), (n_samples, len(data)))
    samples = data[indices]
    return samples
def stat_intervals(stat, alpha):
    # функция для интервальной оценки
    boundaries = np.percentile(stat, [100 * alpha / 2., 100 * (1 - alpha / 2.)])
    return boundaries
```

Оценим, сколько в среднем делает звонков каждая из групп.

Данных мало, поэтому искать среднее не совсем правильно, применим **bootstrap** и сделаем интервальную оценку среднего

```
import numpy as np
def get_bootstrap_samples(data, n_samples):
    # функция для генерации подвыборок с помощью бутстрэпа
    indices = np.random.randint(0, len(data), (n_samples, len(data)))
    samples = data[indices]
    return samples
def stat_intervals(stat, alpha):
    # функция для интервальной оценки
    boundaries = np.percentile(stat, [100 * alpha / 2., 100 * (1 - alpha / 2.)])
    return boundaries
```

```
# сохранение в отдельные патру массивы данных по лояльным и уже бывшим клиентам
loyal_calls = telecom_data[telecom_data['Churn'] == False]['Customer service calls'].values
churn_calls = telecom_data[telecom_data['Churn'] == True]['Customer service calls'].values

# ставим seed для воспроизводимости результатов
np.random.seed(0)

# генерируем выборки с помощью бутстрэпа и сразу считаем по каждой из них среднее
loyal_mean_scores = [np.mean(sample)
                      for sample in get_bootstrap_samples(loyal_calls, 1000)]
churn_mean_scores = [np.mean(sample)
                      for sample in get_bootstrap_samples(churn_calls, 1000)]
```

```
# генерируем выборки с помощью бутстрэра и сразу считаем по каждой из них среднее
loyal_mean_scores = [np.mean(sample)
                      for sample in get_bootstrap_samples(loyal_calls, 1000)]
churn_mean_scores = [np.mean(sample)
                      for sample in get_bootstrap_samples(churn_calls, 1000)]

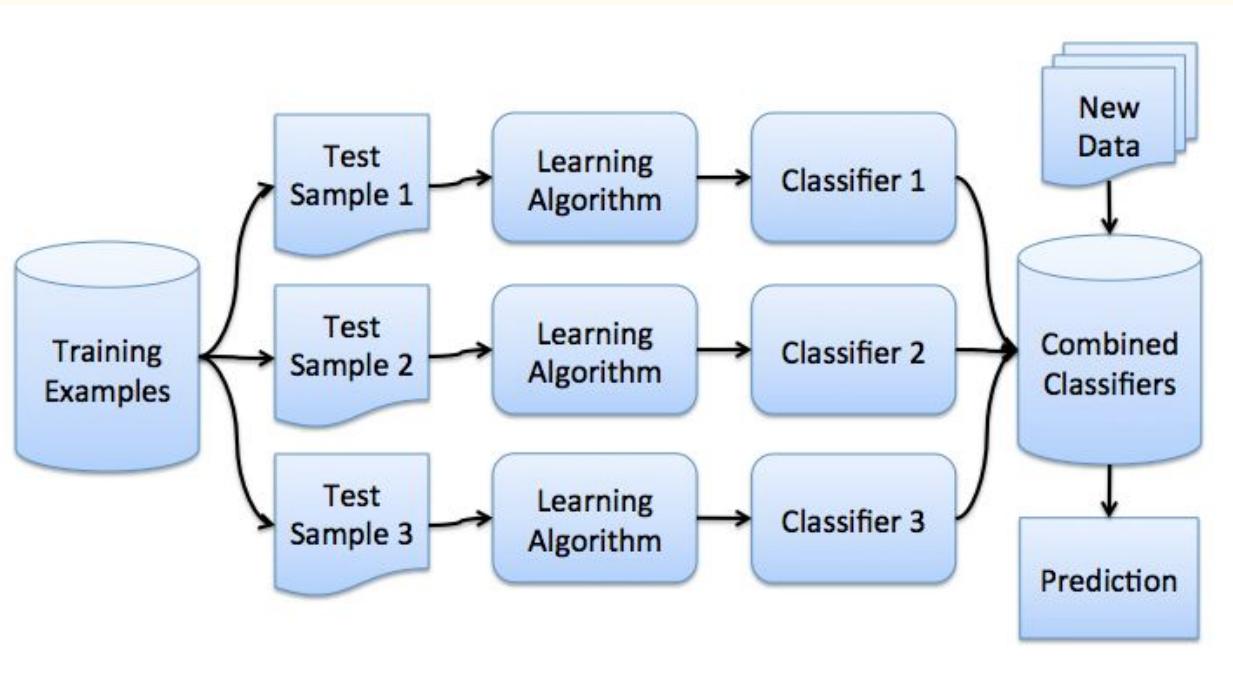
# выводим интервальную оценку среднего
print("Service calls from loyal: mean interval", stat_intervals(loyal_mean_scores,
0.05))
print("Service calls from churn: mean interval", stat_intervals(churn_mean_scores,
0.05))
```

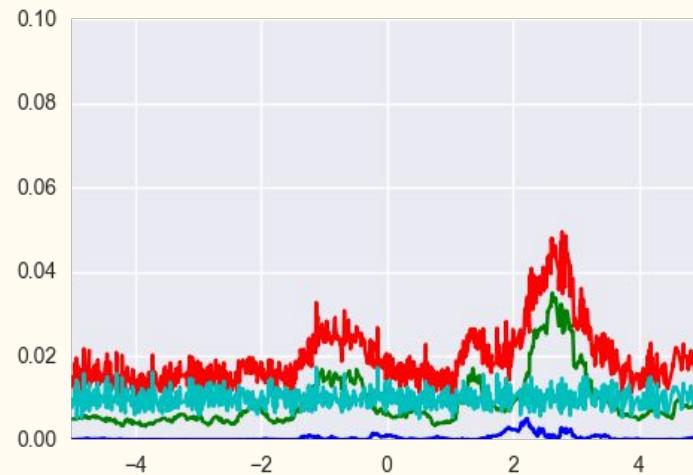
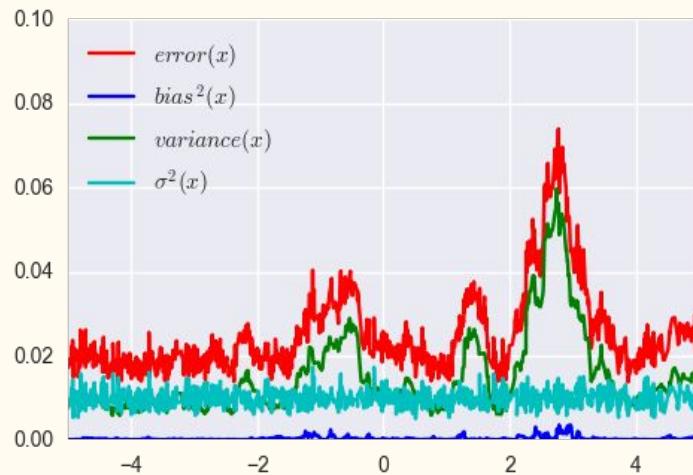
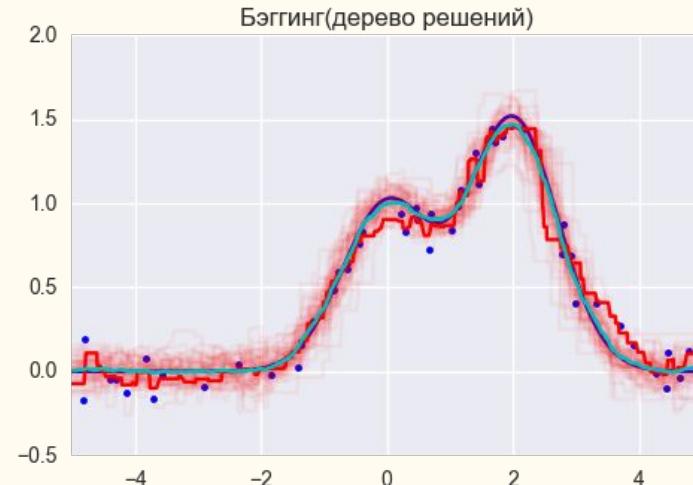
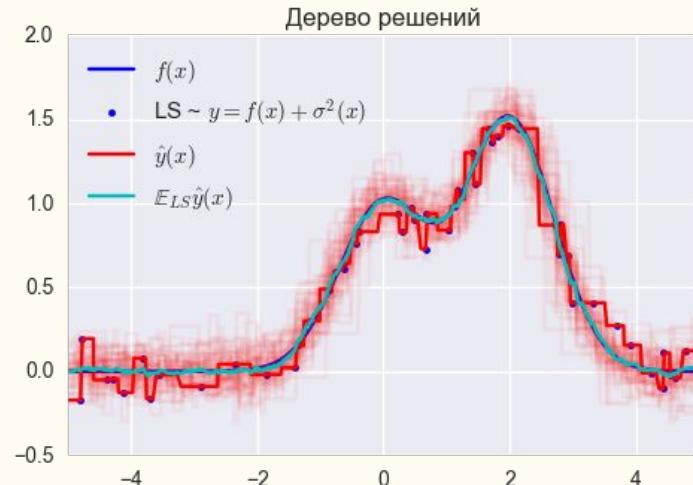
С 95% вероятностью среднее число звонков от лояльных клиентов будет лежать в промежутке между 1.40 и 1.50, в то время как наши бывшие клиенты звонили в среднем от 2.06 до 2.40 раз

Bagging

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:

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



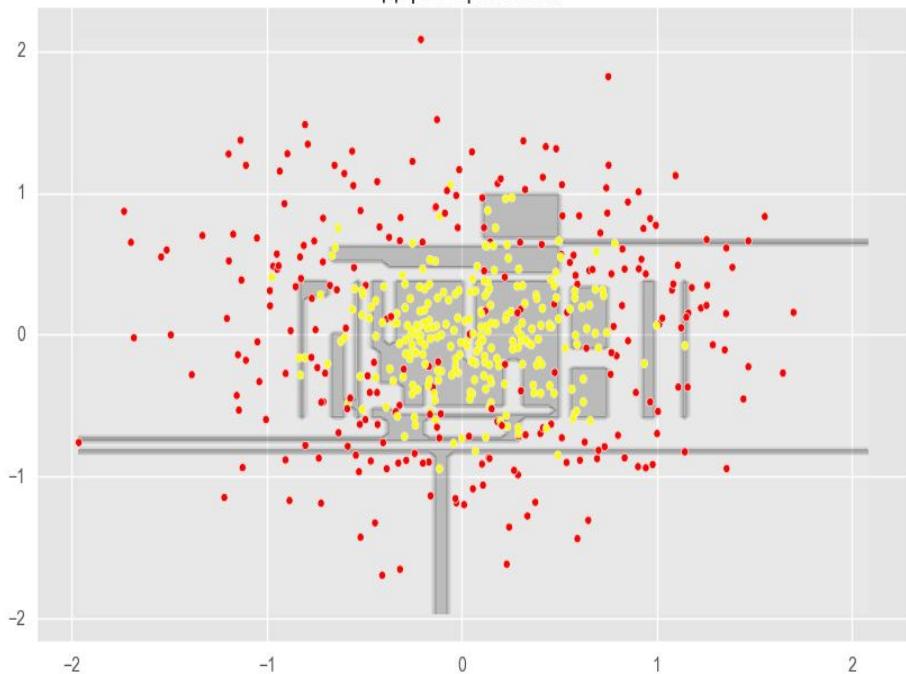


Random Forest

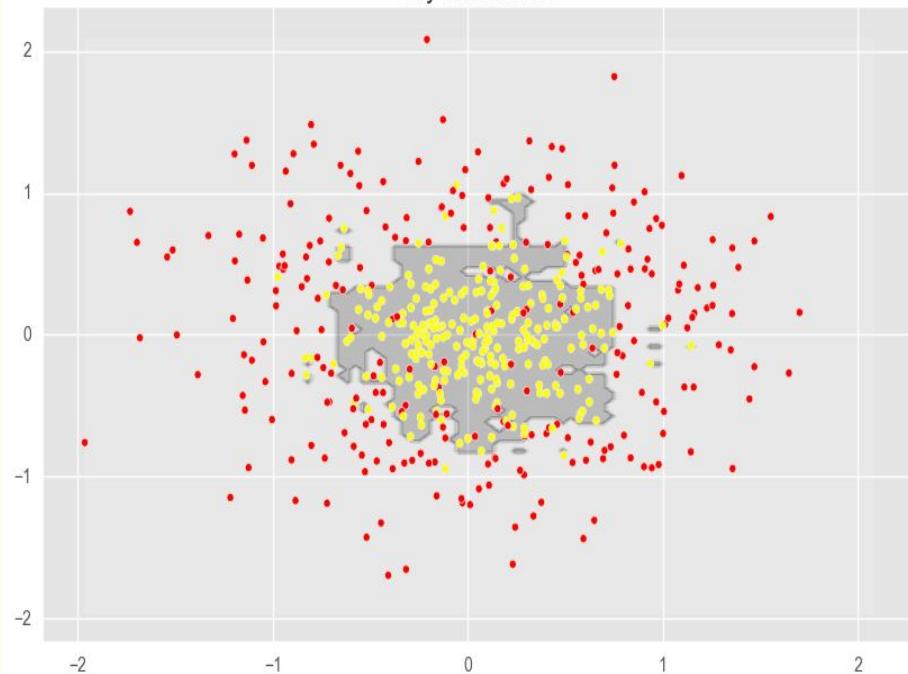
Случайный лес — это бэггинг над решающими деревьями, при обучении которых для каждого разбиения признаки выбираются из некоторого случайного подмножества признаков (bootstrap)



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/>

