

# **Системы искусственного интеллекта**

## **Лекция 7 Ансамбли**

Запорожцев Иван Федорович  
zaporozhtsev.if.work@gmail.com

# Ансамбль алгоритмов

Ensemble /  
Multiple Classifier System



– алгоритм, который состоит из нескольких алгоритмов машинного обучения  
(**базовых алгоритмов** – base learners)

Простой ансамбль в регрессии:

$$a(x) = \frac{1}{n}(b_1(x) + \dots + b_n(x))$$

комитет большинства

Простой ансамбль в классификации:

$$a(x) = \text{mode}(b_1(x), \dots, b_n(x))$$

В чём может быть усложнение?

# Ансамбль алгоритмов

$$a(x) = b(b_1(x), \dots, b_n(x))$$



- $b$   $\longrightarrow$  мета-алгоритм (meta-estimator),
- $b_i$   $\longrightarrow$  базовые алгоритмы (base learners)  
в бустинге – слабые (weak)

# Реализация в scikit-learn

```
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier, VotingClassifier

clf1 = LogisticRegression(multi_class='multinomial', random_state=1)
clf2 = RandomForestClassifier(n_estimators=50, random_state=1)
clf3 = GaussianNB()

ecf1 = VotingClassifier(estimators=[
    ('lr', clf1), ('rf', clf2), ('gnb', clf3)],
    voting='hard', # по большинству или soft - сумма вероятностей
    weights=None, # веса алгоритмов
    flatten_transform=True) # для явного голосования - форма ответа

ecf1 = ecf1.fit(X, y)
print(ecf1.predict(X))
```

# Ошибка суммы регрессоров: теоретическое обоснование

---

Если ответы регрессоров на объекте –  
независимые случайные величины  
с одинаковым матожиданием и дисперсией

---

$$\xi = \frac{1}{n}(\xi_1 + \dots + \xi_n)$$

$$E\xi = \frac{1}{n}(E\xi_1 + \dots + E\xi_n) = E\xi_i$$

$$D\xi = \frac{1}{n^2}(D\xi_1 + \dots + D\xi_n) = \frac{D\xi_i}{n}$$

---

А если есть корреляция  
между базовыми алгоритмами?

# Ошибка комитета большинства: теоретическое обоснование

Пусть три (независимых) бинарных классификатора с вероятностью ошибки  $p$

Пусть верный ответ – 0

$$\left. \begin{array}{l} (0,0,0) \\ (1,0,0) \\ (0,1,0) \\ (0,0,1) \end{array} \right\} \begin{array}{l} (1-p)(1-p)(1-p) \\ p(1-p)(1-p) \\ (1-p)p(1-p) \\ (1-p)(1-p)p \end{array}$$



Верный ответ

$$\left. \begin{array}{l} (1,1,1) \\ (1,1,0) \\ (0,1,1) \\ (1,0,1) \end{array} \right\} \begin{array}{l} ppp \\ pp(1-p) \\ (1-p)pp \\ p(1-p)p \end{array}$$



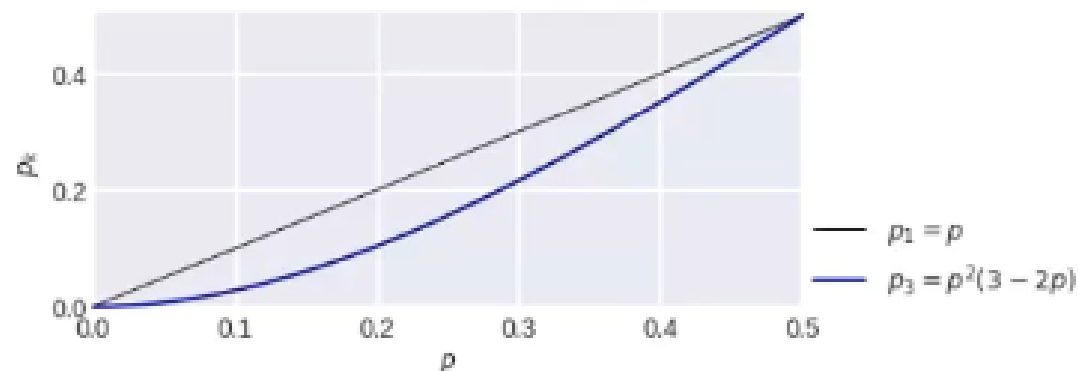
Ошибка

$$p^3 + 3(1-p)p^2 = p^2(3-2p)$$



Вероятность ошибки

# Ошибка комитета большинства



- При малых  $p$  ошибка комитета очень мала!
- При  $p = 0.2$  — почти в два раза меньше

# Ошибка комитета большинства

Общий случай



$$\sum_{t=0}^{\lfloor n/2 \rfloor} C_n^t (1-p)^t p^{n-t} \leq e^{-\frac{1}{2}n(2p-1)^2}$$

Неравенство Хёфдинга (Hoeffding)

Ошибка экспоненциально  
снижается с увеличением  
числа базовых алгоритмов...

Но это в теории

# Выход

Пытаться делать алгоритмы разнообразными

Пусть алгоритмы ошибаются, но по-разному:  
ошибки одних компенсируются правильными ответами других

## Одинаковые

1111000000  $q=0.6$   
1111000000  $q=0.6$   
1111000000  $q=0.6$

$q_{ens} = 0.6$

## Похожие

1111000000  $q=0.6$   
1110100000  $q=0.6$   
1101100000  $q=0.6$

$q_{ens} = 0.5$

## Разные

0100101100  $q=0.6$   
1001010100  $q=0.6$   
0010100011  $q=0.6$   
**0000100100**  
 $q_{ens} = 0.8$

Нет ли здесь обмана?

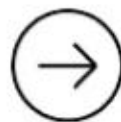
# Повышения разнообразия – что «варьируют»



Обучающую выборку  
(бэггинг)



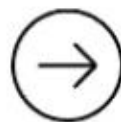
Признаки  
(Random Subspaces)



Целевой вектор  
(ECOC,  $f(y)$ )



Модели  
(стекинг)

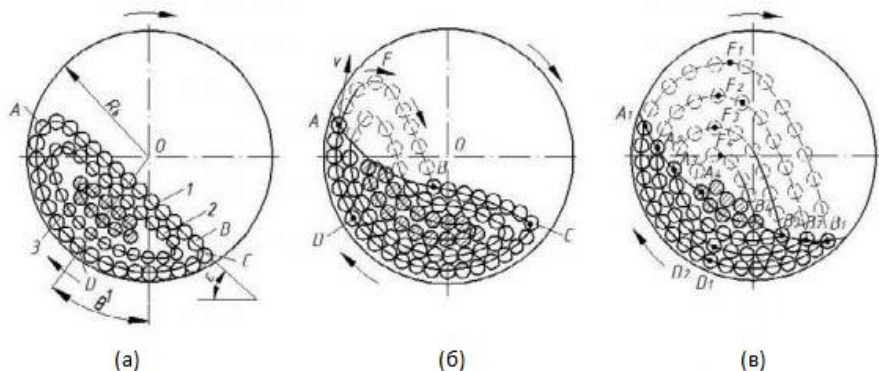


Алгоритмы в модели  
(разные гиперпараметры, инициализации,  
snapshot, разные random seed в RF, ...)

И результаты «усредняют» —  
тоже есть разные способы

# Способы усреднения

## Комитеты (голосование, Voting Ensembles)



Контур и схема движения мелющих тел (шаров) при каскадном (а), смешанном (б) и водопадном (в) режимах работы мельницы

Голосование по большинству (Majority vote)

$$a(x) = \text{mode}(b_1(x), \dots, b_n(x))$$

Комитеты единогласия  
в бинарной задаче классификации

$$a(x) = \min(b_1(x), \dots, b_n(x))$$

Комитеты единогласия  
в бинарной задаче классификации

«тревога при малейшем подозрении»

$$a(x) = \max(b_1(x), \dots, b_n(x))$$

# Способы усреднения

Обобщения среднего  
арифметического

---

«Среднее арифметическое»

$$a(x) = \frac{1}{n} (b_1(x) + \dots + b_n(x))$$

+ любые другие средние (ех: по Колмогорову)

$$a(x) = \frac{1}{n} f^{-1} (f(b_1(x)) + \dots + f(b_n(x)))$$

---

Ранговое усреднение (Rank Averaging)

$$a(x) = \frac{1}{n} (\text{rank}(b_1(x)) + \dots + \text{rank}(b_n(x)))$$

ориентировано на конкретный AUC ROC

## О пользе рангов! Выбираем лучший вариант Feature Subspace для нескольких тестов (0 – общий, 1-4 – частные)

22.12 - 01.01 ( 978): MAE = 0.4869, R2 = 0.6082

01.01 - 12.01 ( 948): MAE = 0.4343, R2 = -0.12

12.01 - 23.01 (1011): MAE = 0.5454, R2 = 0.4687

23.01 - 31.01 ( 755): MAE = 0.5427, R2 = 0.5159

	extra	r2_0	r2_1	r2_2	r2_3	r2_4	mae_0	mae_1	mae_2	mae_3	mae_4	r2_r	mae_r
(Air 5, Tok 4, Tok 5, Tok 6)		0.149593	0.075103	0.214961	0.092916	0.224181	0.698724	0.736177	0.694671	0.741633	0.667615	1	1
(Air 5, Air 6, Tok 4, Tok 5, Tok 6)		0.150085	0.076349	0.215344	0.091124	0.224627	0.698559	0.735596	0.694573	0.742509	0.667301	1	1
(Air 5, Tok 4, Tok 5, Tok 6, in_FE)		0.150506	0.074022	0.213734	0.095591	0.226256	0.698824	0.738008	0.695285	0.740758	0.666819	1	1
(Air 4, Air 5, Tok 4, Tok 5, Tok 6, in_FE)		0.150169	0.072479	0.213778	0.095761	0.226110	0.699000	0.738650	0.695353	0.740677	0.666951	1	1
(Air 5, Air 6, Tok 4, Tok 5, Tok 6, in_FE)		0.150974	0.075230	0.214112	0.093874	0.226663	0.698653	0.737457	0.695184	0.741554	0.666537	1	1

```
df['r2_r'] = df.groupby(exp_meta_part)['r2_1', 'r2_2', 'r2_3', 'r2_4']\
              .transform(lambda x: x.rank(pct=True)).sum(axis=1)
inds = df.sort_values(exp_meta_part + ['r2_r'], ascending=False).groupby(exp_meta_part)\
        .head(10).index
df.r2_r = 0
df.loc[inds, 'r2_r'] = 1
```

# Способы усреднения

Усреднение с весами  
(weighted averaging)

---

Усреднение (регрессия)

$$a(x) = \frac{1}{w_1 + \dots + w_n} (w_1 \cdot b_1(x) + \dots + w_n \cdot b_n(x))$$

---

Голосование (классификация)

$$a(x) = \arg \max_j \left[ \sum_{t: b_t(x)=j} w_t \right]$$

---

Feature-Weighted Linear Stacking

$$\begin{aligned} a(x) &= w_1(x) \cdot b_1(x) + \dots + w_n(x) \cdot b_n(x) = \\ &= \sum_t \left( \sum_i w_{ti} x_{[i]} \right) b_t(x) = \sum_{t,i} w_{ti} x_{[i]} b_t(x) \end{aligned}$$

# Обоснования применения ансамблей



## Статистическое (Statistical)

– ошибка может быть меньше



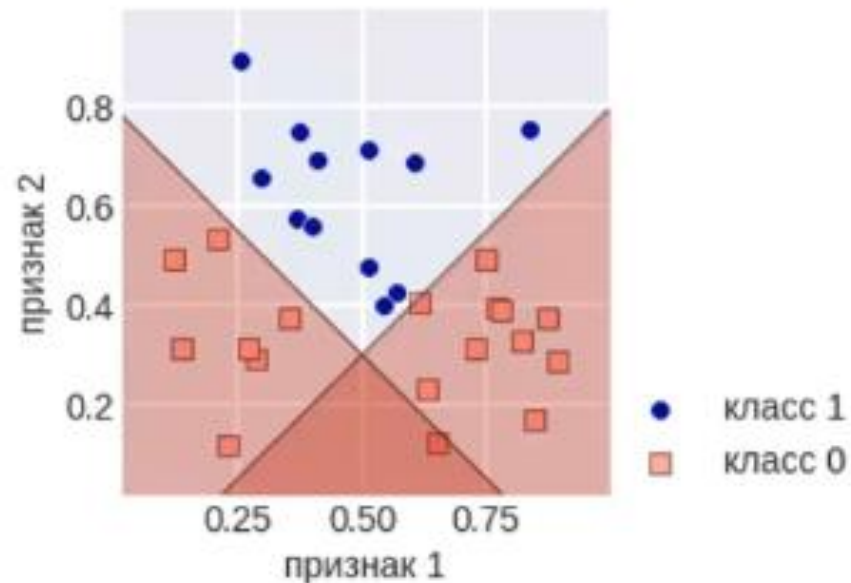
## Вычислительное (Computational)

– обучение = оптимизация функции,  
а ансамбль «распараллеливает» процесс



## Функциональное (Representational)

– можно представить функции, которые нельзя  
было с помощью базовых алгоритмов



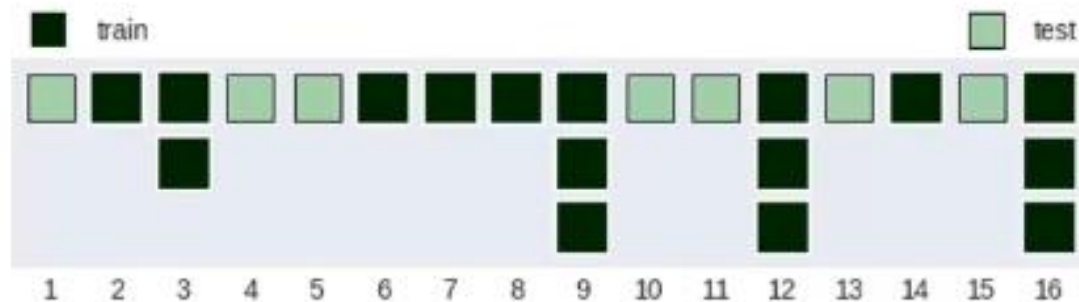
# Варьирование выборки: Бэгинг (Bagging)

## bootstrap aggregating

Каждый базовый алгоритм  
настраивается на случайной  
подвыборке обучения

С помощью выбора с возвращением формируется  
подвыборка полного объёма  $m$ , на которой  
производится обучение модели

На остальных объектах (которые не попали  
в обучение) – контроль



Какова вероятность, что объект не попадёт в трэйн:

$$\left(1 - \frac{1}{m}\right)^m \approx e^{-1} \approx 0.37 = 37\% \text{ выборки}$$

$$\lim_{m \rightarrow \infty} \left(1 + \frac{1}{m}\right)^m = e$$

$$\lim_{m \rightarrow \infty} \left(1 - \frac{1}{m}\right)^m = \left\{ \begin{array}{l} \text{замена} \\ t := -m \end{array} \right\} = \lim_{t \rightarrow \infty} \left(1 + \frac{1}{t}\right)^{-t} =$$

$$= \lim_{t \rightarrow \infty} \left( \underbrace{\left(1 + \frac{1}{t}\right)^t}_{\rightarrow e \text{ при } t \rightarrow \infty} \right)^{-1} = e^{-1}$$



модель учится на выборке  
того же объёма, что и итоговая  
(которую мы обучим по всей выборке)

- использует не все данные
- есть дубликаты



с точки зрения распределения  
бутстреп-выборка похожа на исходную

# Бэгинг (Bagging)

1

Цикл по  $t$  (номер базового алгоритма)

1.1 Взять подвыборку  $[X', y']$   
обучающей выборки  $[X, y]$

1.2 Обучить  $t$ -й базовый алгоритм  
на этой подвыборке:

$$b_t = \text{fit}(X', y')$$

2

Ансамбль

$$a(x) = \frac{1}{n}(b_1(x) + \dots + b_n(x))$$

(для задач регрессии).

Каждый базовый алгоритм обучается  
~ на 63% данных, остальные называются –  
out-of-bag-наблюдениями (OOB)

$$1 - \frac{1}{e} \approx 0.632$$

- процедура снижения variance  
в статистическом обучении

# OOB- prediction



На OOB-части выборки  
можно получить ответы  
алгоритма

Пусть на  $i$ -й итерации это часть:  $\text{OOB}_i$   
и мы построили алгоритм  $b_i$



OOB-ответы бэгинга (OOB-prediction)

$$a_{\text{OOB}}(x_j) = \frac{1}{|\{i : x_j \in \text{OOB}_i\}|} \sum_{i: x_j \in \text{OOB}_i} b_i(x_j)$$



Можно вычислить OOB-ошибку бэгинга

Хорошая оценка ошибки  
на тесте похожа на CV-ошибку...

# Реализация в scikit-learn

```
from sklearn.svm import SVC
from sklearn.ensemble import BaggingClassifier

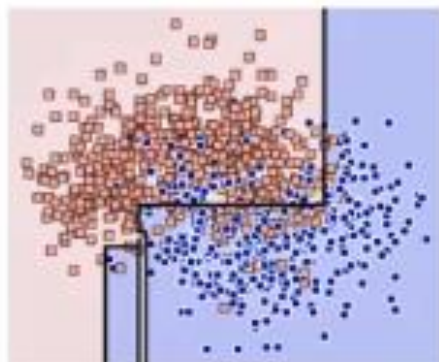
clf = BaggingClassifier(estimator=SVC(), # Базовый модель (принимает base_estimator)
                        n_estimators=10, # Число ансамблевых моделей
                        max_samples=1.0, # Размер подвыборки (доля или число)
                        max_features=1.0, # Число / доля признаков для подвыборки
                        bootstrap=True, # Выбирать ли подвыборки с повторениями
                        bootstrap_features=False, # Аналогичная опция для признаков
                        oob_score=False, # Включить ли OOB-оценку
                        warm_start=False) # Продолжить ли с ранее обученными
                                     # приближенными старыми моделями

clf.fit(X, y)
```

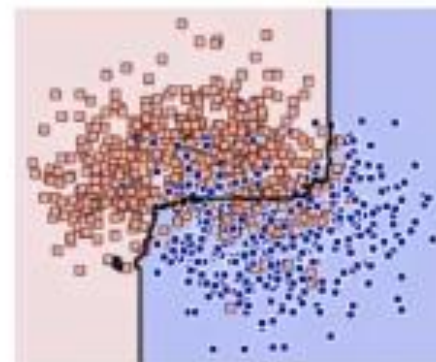
есть ещё  
`ensemble.BaggingRegressor`

# Примеры бэгинга

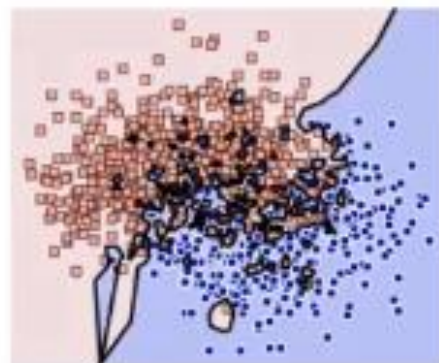
Одно  
дерево



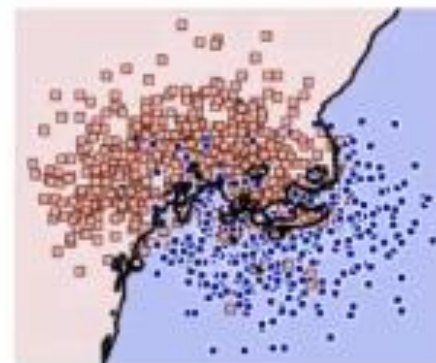
Бэгинг  
100 деревьев



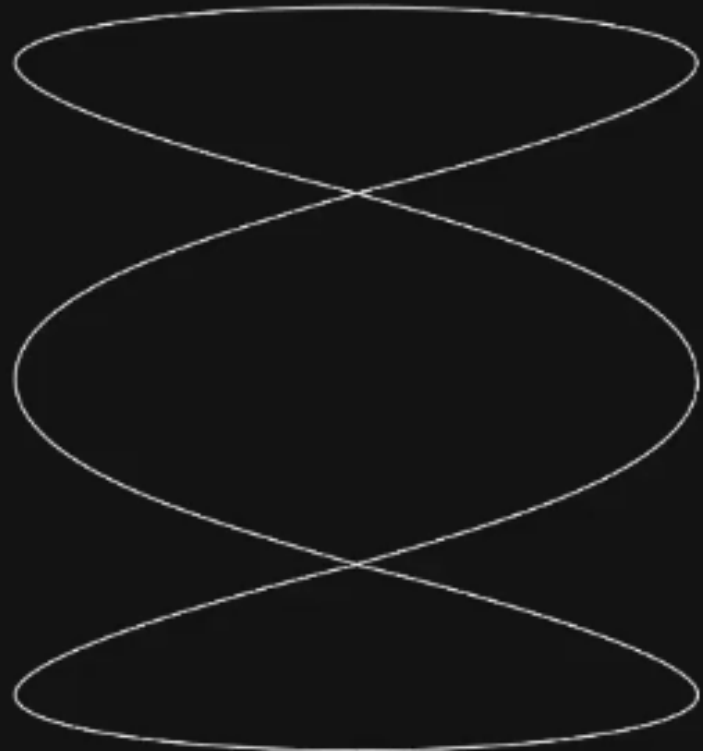
Ближайший  
сосед



Бэгинг 100  
ближайших соседей



# Варьирование признаков



## Случайные подпространства (Random Subspaces)

случайное подмножество признаков



## Бэгинг (Bagging)

подвыборка обучающей выборки  
берётся с помощью бутстрепа



## Пэстинг (Pasting)

случайная обучающая подвыборка



## Cross-Validated Committees

$k$  обучений на  $(k-1)$ -м фолде



## Случайные патчи (Random Patches)

одновременно берём случайное  
подмножество объектов и признаков

# Случайный лес (Random Forest)

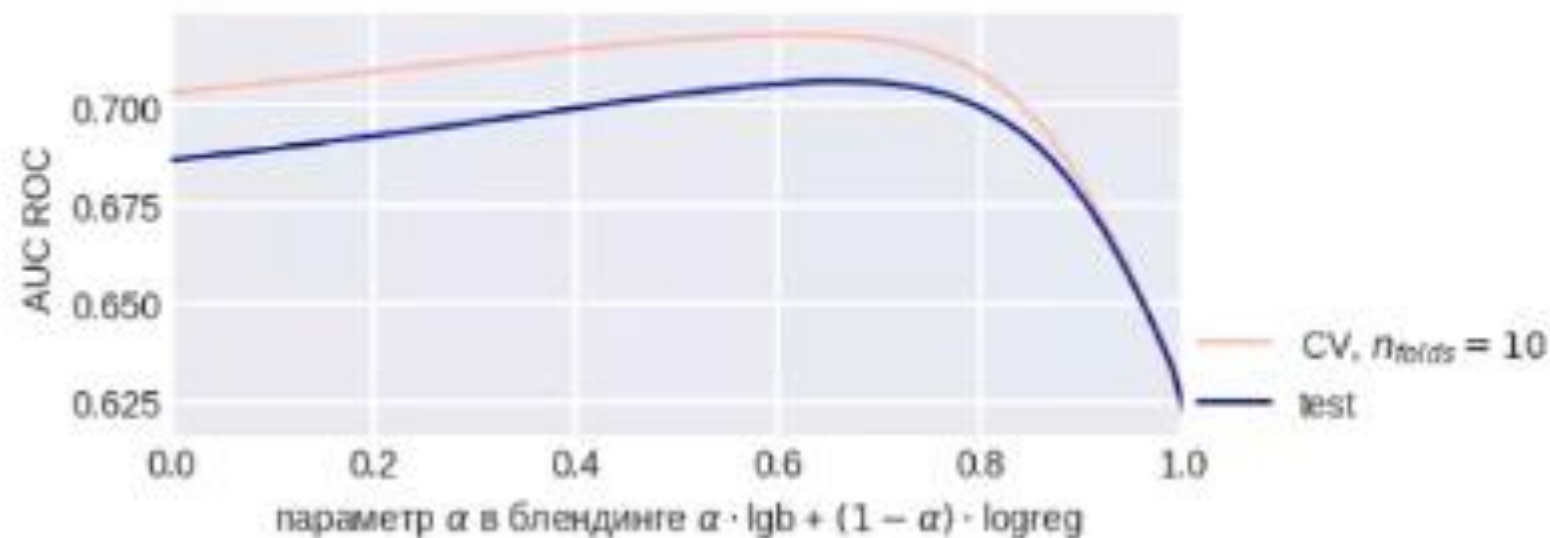
Дальнейшие улучшения  
независимости базовых  
классификаторов



бэггинг + случайности при построении деревьев

Отдельная лекция

# Варьирование моделей: Блендинг (Blending)



Варьирование моделей +  
обобщение усреднения

## Стекинг (stacking)

Идея:

---

Хорошо усреднять алгоритмы,  
но почему именно усреднять?

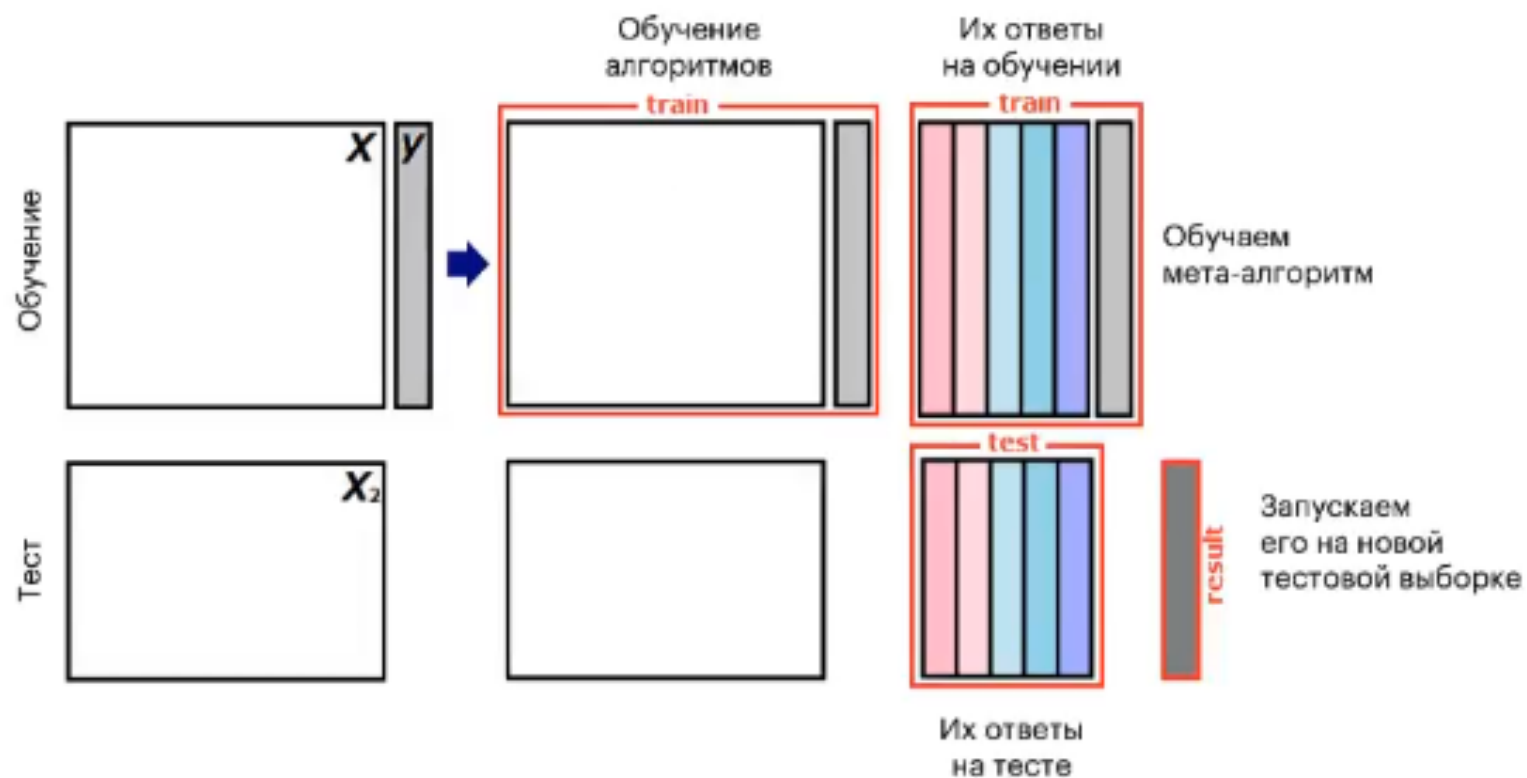
приходит в голову всем...



$$a(x) = b(b_1(x), \dots, b_n(x))$$

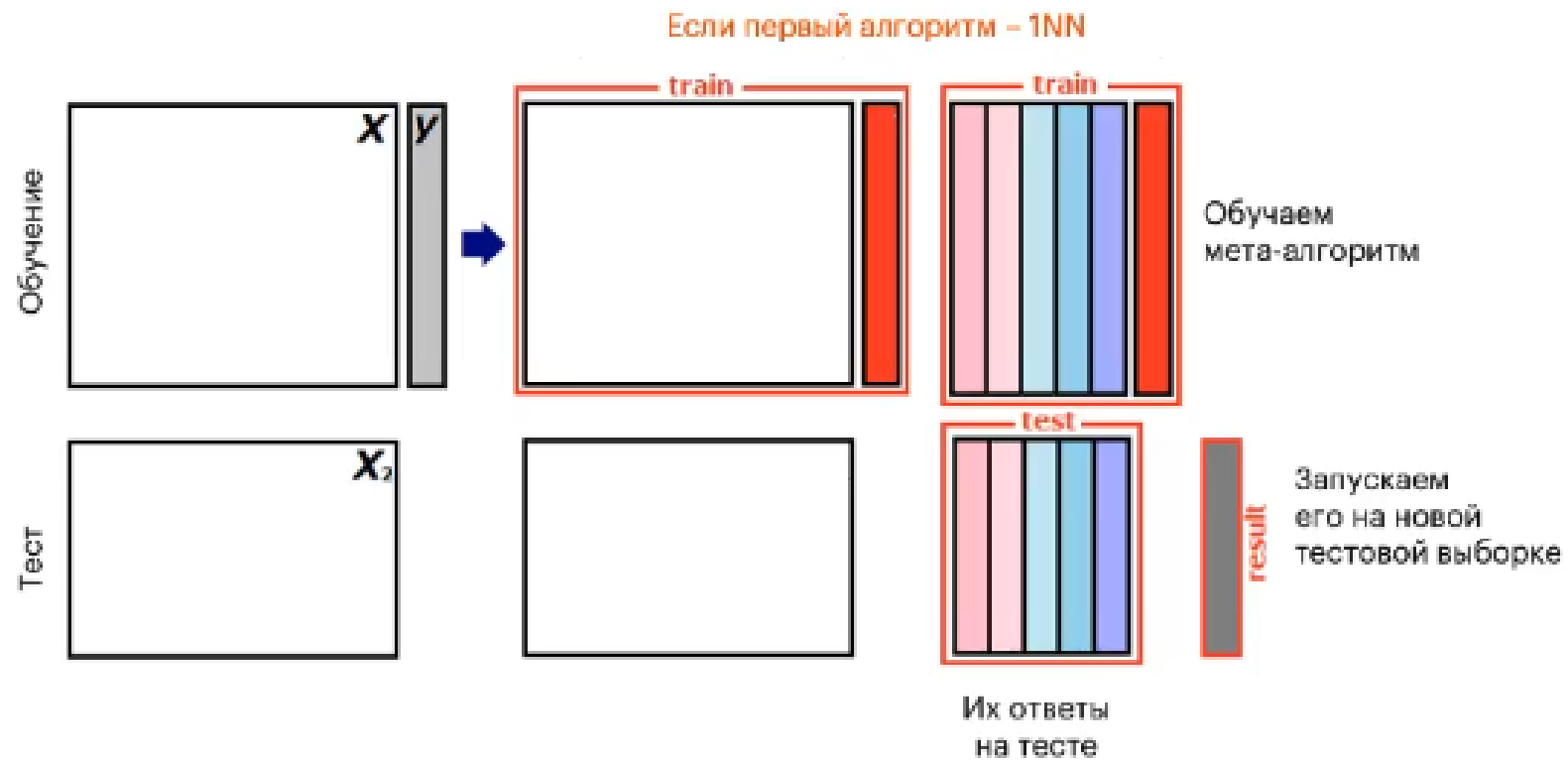
$b \longrightarrow$  мета-алгоритм, который нужно  
отдельно настроить!

# Наивная форма стекинга



Что здесь  
неправильно?

# Наивная форма стекинга



Происходит  
переобучение



базовый алгоритм  
на обучении воспроизводит  
истинные метки,  
метаалгоритм ему  
доверяет... но на тесте  
он уже не знает  
правильных меток

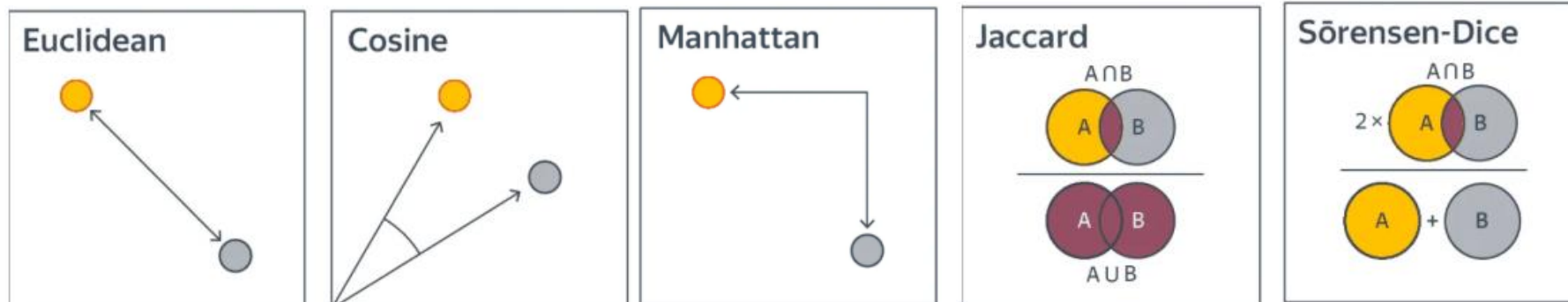
# Метод k-ближайших соседей (KNN)

k-nearest neighbors

Хотим предсказать класс

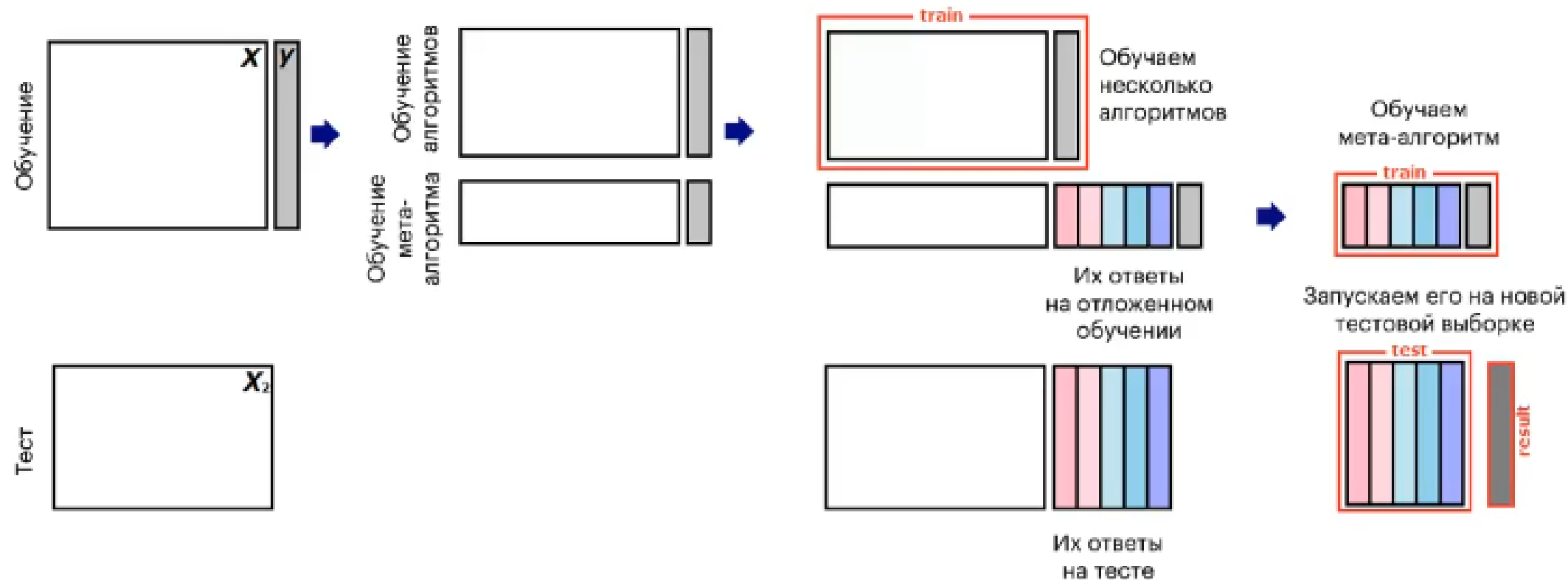


Относится к классу метрических методов. А какая метрика?



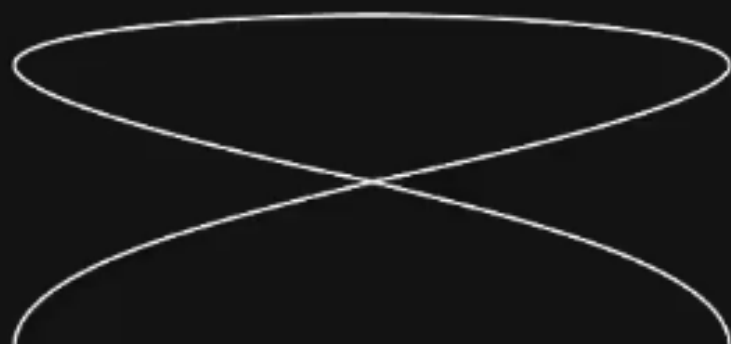
# Стекинг (stacking)

## Простейшая форма стекинга



# Блендинг

Термин введён  
победителями  
конкурса Netflix



Сейчас блендингом называются  
простейшие формы стекинга, например,  
выпуклую комбинацию алгоритмов

$$\alpha_i \geq 0 \text{ и } \alpha_1 + \alpha_2 + \dots + \alpha_n = 1$$

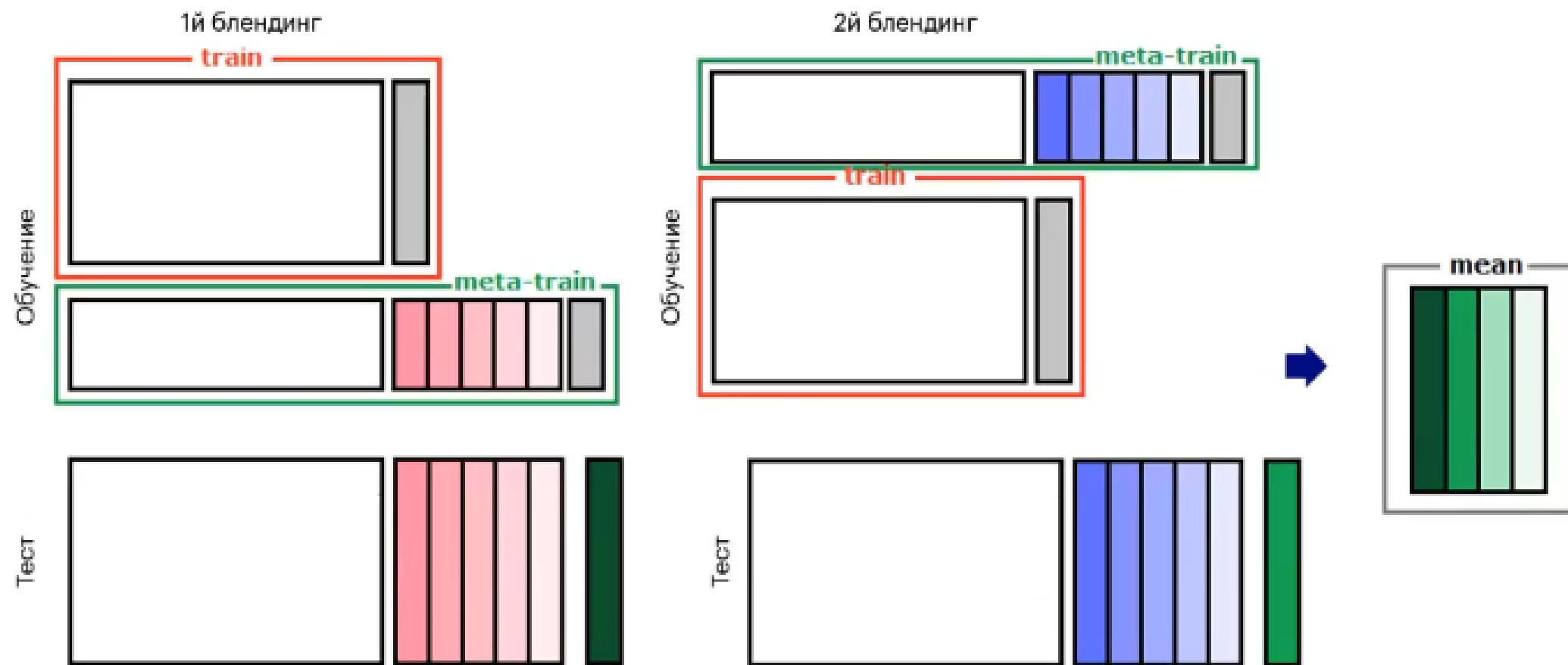


Недостатки:

Используется не вся обучающая выборка

- можно усреднить несколько блендингов

# Блендинг: усреднение ответов



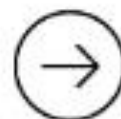
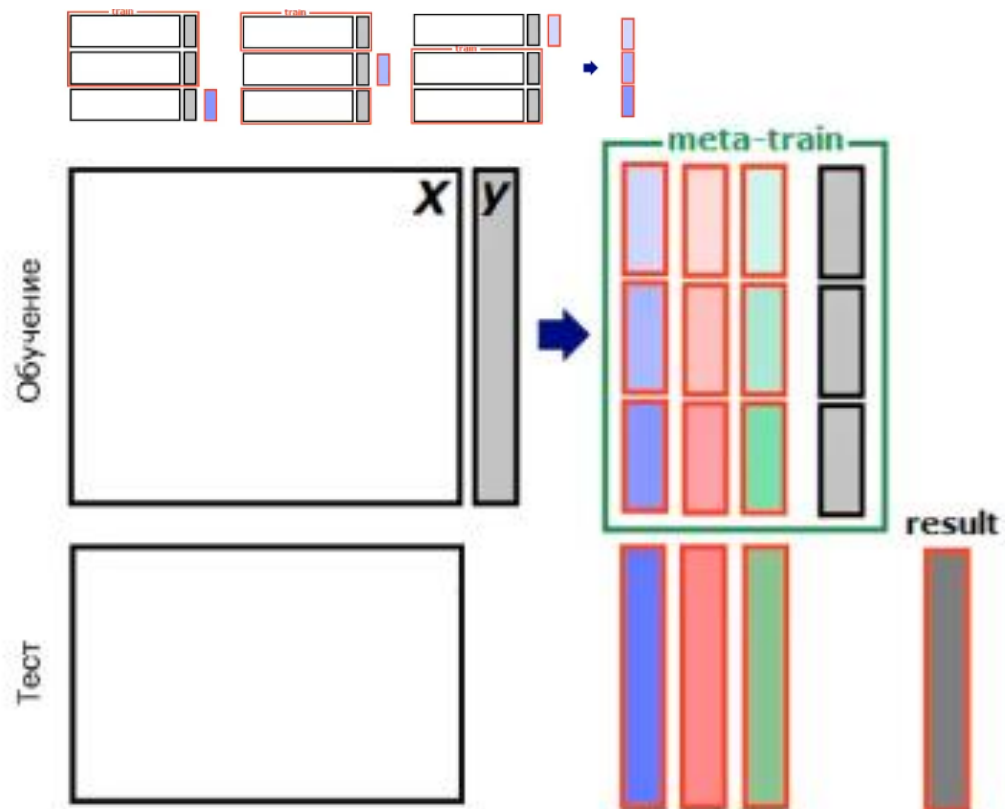
# Стекинг

Хотим использовать всю обучающую выборку



# Стекинг

Хотим использовать всю обучающую выборку



Получаем k-Fold-методом  
все метапризнаки



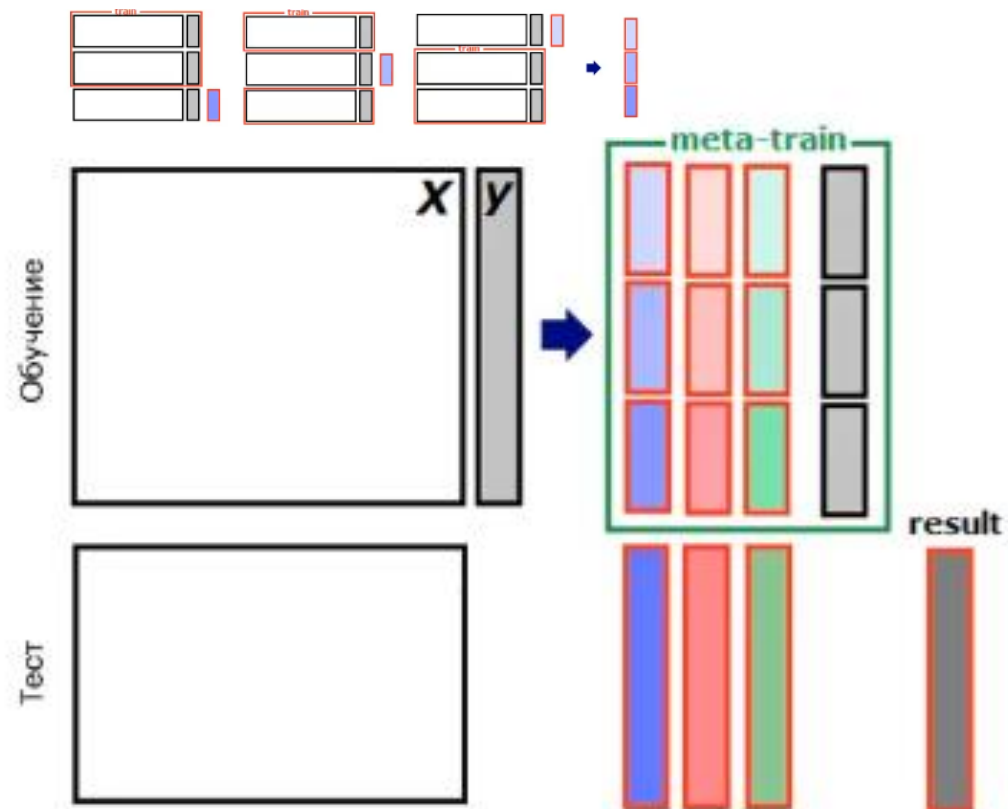
Используем все  
базовые алгоритмы



Обучаем мета-алгоритм

# Стекинг

Хотим использовать всю обучающую выборку



Результат агрегированной модели или  
усреднение ответов нескольких моделей



Получаем k-Fold-методом  
все метапризнаки

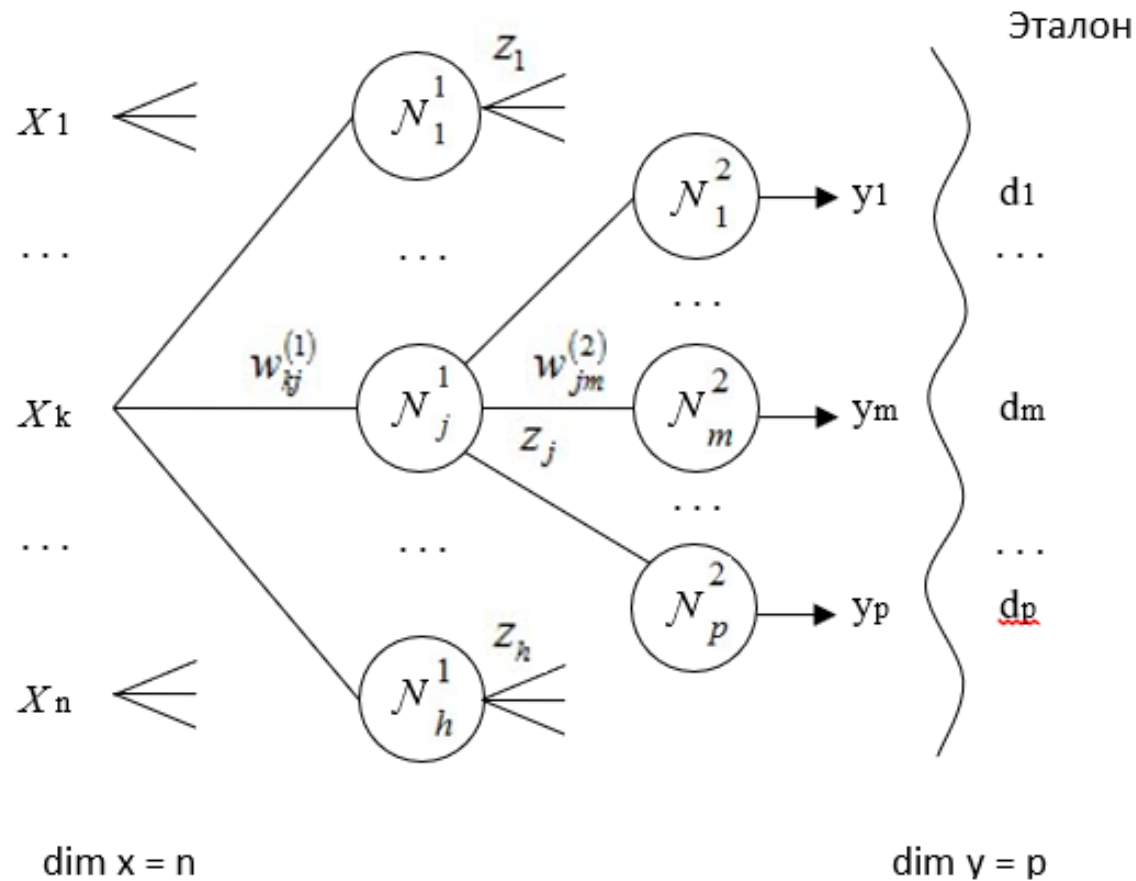


Используем все  
базовые алгоритмы



Обучаем мета-алгоритм

# Стекинг в нейронных сетях



Базовые алгоритмы = нейроны одного слоя  
 Мета-алгоритм = нейрон следующего слоя  
 по отношению к текущему



**Спасибо за внимание!**



Запорожцев Иван Федорович  
zaporozhtsev.if.work@gmail.com