Системы искусственного интеллекта

Лекция 8

Случайный лес

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

Ансамбли

Комитеты (голосование) / усреднение	в том числе, различные усреднения, с предварительной деформацией, калибровкой, бэгинг (bagging) + обобщения (RF)
Перекодировки ответа	кодирование целевого вектора, ECOC (error-correcting output coding)
Стекинг (stacking)	построение метапризнаков — ответов алгоритмов на объектах выборки, обучение на них мета-алгоритма
Бустинг (boosting)	построение суммы алгоритмов: каждое следующее слагаемое строится с учётом ошибок предыдущих
«Ручные методы»	эвристические способы комбинирования ответов базовых алгоритмов
Однородные ансамбли	рекурсия в формуле мета-алгоритм(базовые) + общая схема оптимизации (пример: нейросети)

Ручные методы ансамблирования

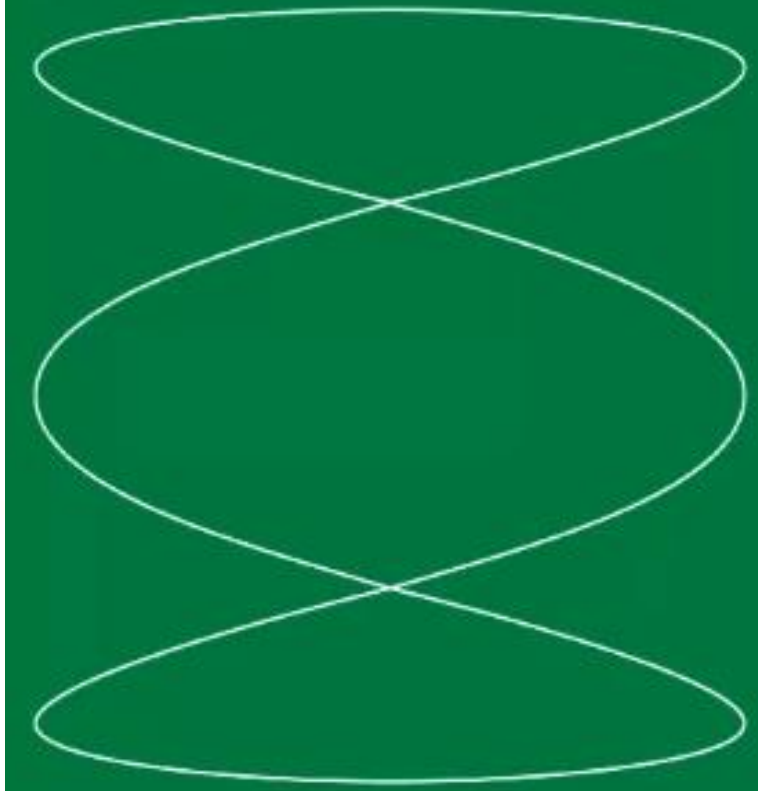
Метод Ефимова

$$f(a_1, a_2)$$

	$a_1 \le 0.1$	$0.1 < a_1 < 0.9$	$a_1 \ge 0.9$
$a_2 \le 0.1$	$\min(a_1, a_2)$	$\min(a_1, a_2)$	$0.55a_1 + 0.45a_2$
$0.1 < a_2 < 0.9$	$0.1a_1 + 0.9a_2$	$\text{mean}(a_1, a_2)$	$0.9a_1 + 0.1a_2$
$a_2 \ge 0.9$	$0.75a_1 + 0.25a_2$	$\max(a_1, a_2)$	$\max(a_1, a_2)$

Итог

Ключевые идеи ассамблирования



01

Объединение ответов разных алгоритмов

усреднение / голосование / стекинг ...

02

Повышения разнообразия / независимости базовых алгоритмов

«варьирование» признаков, объектов, моделей, в модели и т.п.

Использование подвыборок / весов

03

Ансамблирование: параллельное и последовательное

Parallel ensembles –

все алгоритмы строятся независимо

Идея: усреднить (high complexity, low bias) – модели, для снижения variance

Sequential ensembles –

алгоритмы строятся последовательно

Случайный лес (Random Forest)

Специальный метод ансамблирования

Лео Брейман,
1928 – 2005

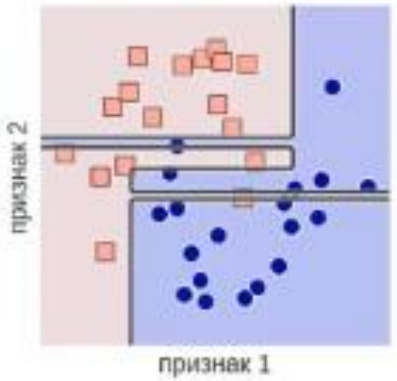
Случайный лес

= бэггинг + специальное построение деревьев
(подмножество признаков при расщеплении)

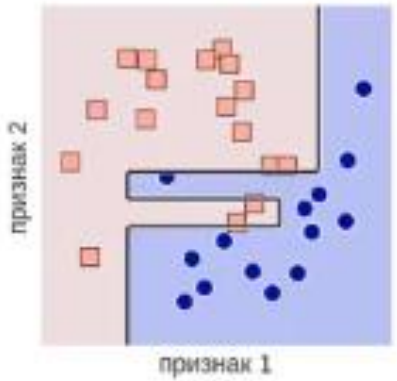
Качество одного дерева очень низкое,
но у ансамбля – высокое!

Случайный лес (Random Forest)

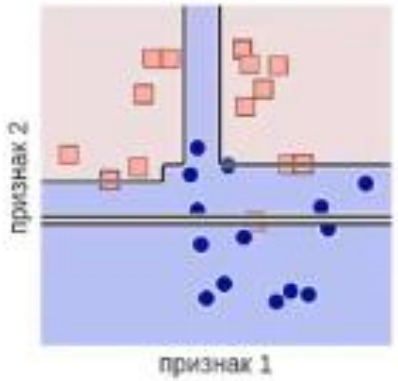
Дерево
№1



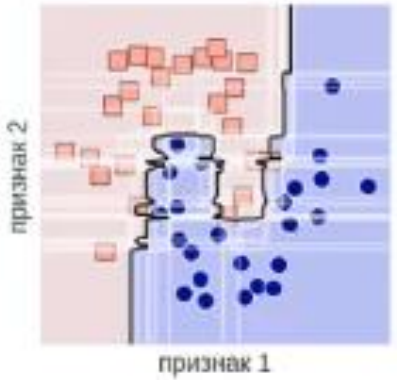
Дерево
№2



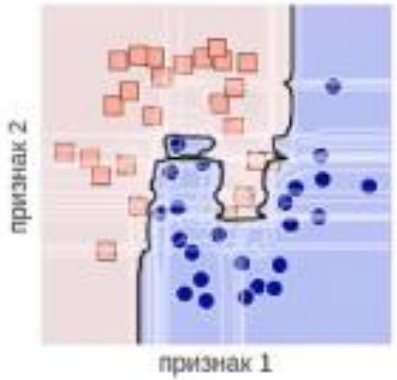
Дерево
№3



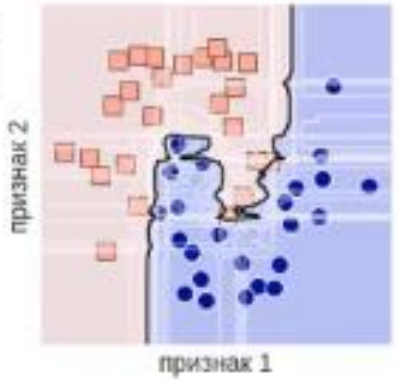
RF, число
деревьев
10



RF, число
деревьев
100



RF, число
деревьев
1000



Построение случайного леса

01 **Выбирается подвыборка**
`max_samples` (м.б. с повторением) – **на ней строится дерево**

Чаще всего используется bootstrap

02 **Строим дерево**

2.1. Для построения каждого расщепления просматриваем `max_features` случайных признаков

2.2. Как правило, дерево строится до исчерпания выборки (без прунинга)

Ответ леса в задачах классификации:

По большинству,
вероятность = процент деревьев (R)

Сравниваем вероятность с порогом /
по максимальной вероятности, вероятность =
среднее арифметическое вероятностей в листьях деревьев ансамбля (sklearn)

Ответ леса в задачах регрессии:

Среднее арифметическое

Автоматически при построении RF получаем

Энтропийный → $H(R) = -\sum_j p_j \log_2 p_j$

Джини → $H(R) = \sum_j p_j(1-p_j) = 1 - \sum_j p_j^2$

$$\frac{|R|}{m}\left(H(R) - \frac{|R_{\text{left}}|}{|R|}H(R_{\text{left}}) - \frac{|R_{\text{right}}|}{|R|}H(R_{\text{right}})\right)$$

Мера неоднородности (impurity)

- **Рейтинг признаков –** `.feature_importances_`

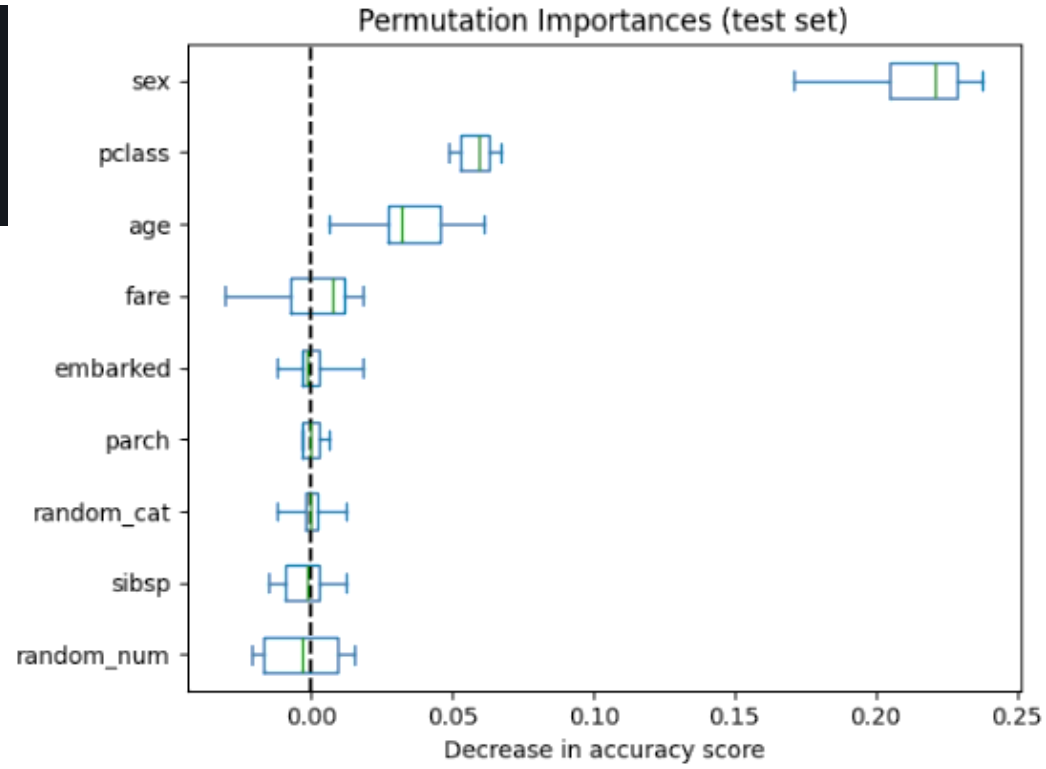
  Отдельная тема

- **Уверенность в ответе – дисперсия ответов деревьев**

Permutation Importance

Height at age 20 (cm)	Height at age 10 (cm)
182	155
175	147
...	...
156	142
153	130

На рисунке ниже показана важность признаков перестановки `RandomForestClassifier`, обученной на расширенной версии набора данных Titanic, который содержит признаки `random_cat` и `random_num`, т.е. категориальный и числовой признак, который никак не коррелирует с целевой переменной:



Подходит для «черного ящика»

Реализация случайного леса

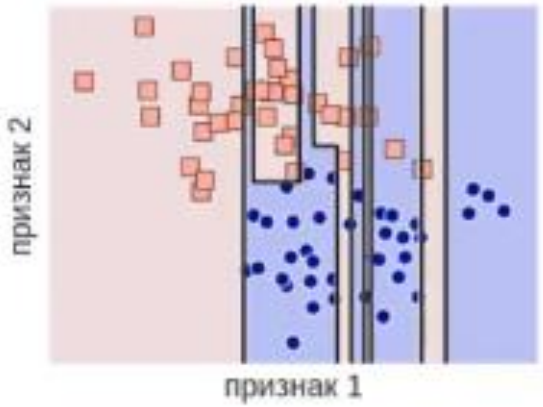
```
from sklearn ensemble import RandomForestClassifier
clf = RandomForestClassifier(n_estimators=100, # число деревьев
                             criterion='gini', # базис
                             max_depth=None, # глуб
                             min_samples_split=2, # базис
                             min_samples_leaf=1, # базис
                             min_weight_fraction_leaf=0.0, # вспом
                             max_features='sqrt', # базис
                             max_leaf_nodes=None, # базис
                             min_impurity_decrease=0.0, # базис
                             bootstrap=True, # метод на бутстрапе
                             oob_score=False, # ООН-оценка качества
                             n_jobs=None,
                             random_state=None,
                             verbose=0
                             warm_start=False, # дополнить ли существующий лес
                             class_weight=None, # вспом
                             ccp_alpha=0.0, # базис
                             max_samples=None) # размер выборки (при bootstrap=True)

clf.fit(X, y)
```

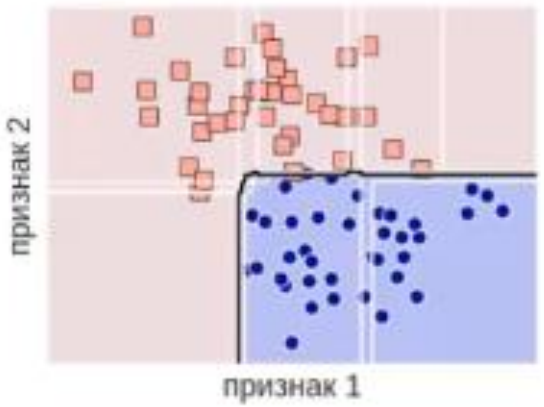
«Случайный лес»

```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(n_estimators=1)
rf.fit(X_train, y_train)
```

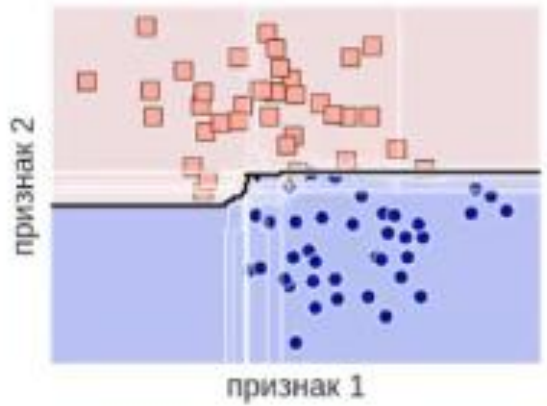
n_estimators=1



n_estimators=5

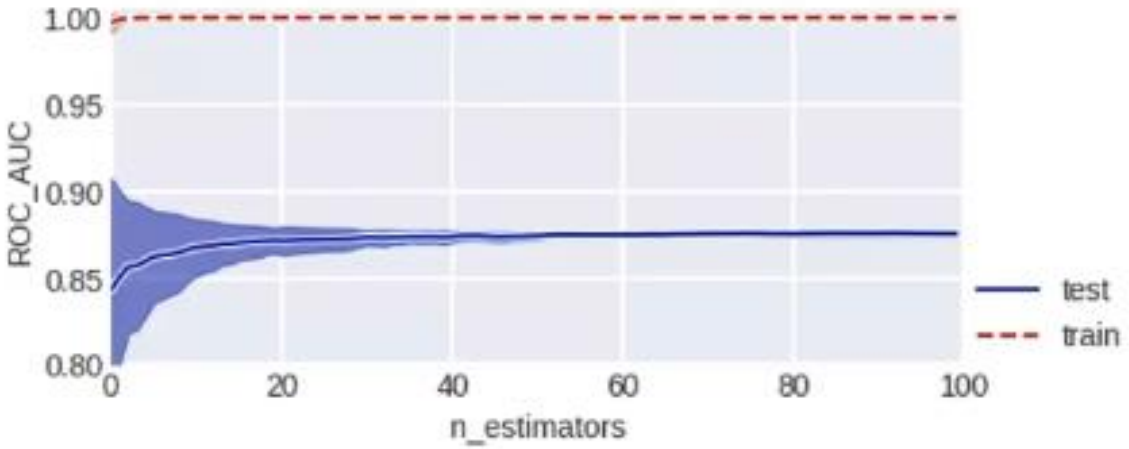


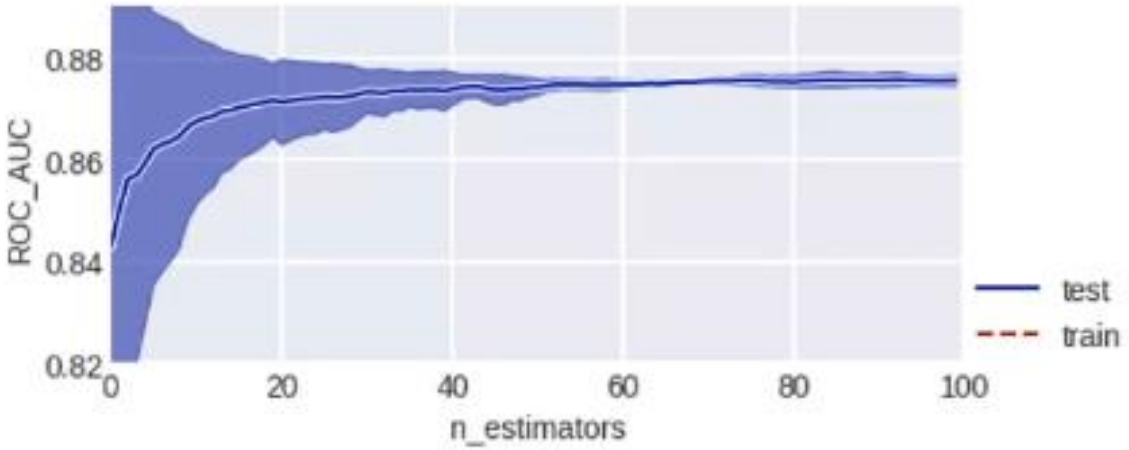
n_estimators=100



Настройка гиперпараметров

`n_estimators`





«Чем больше, тем лучше»

Настройка гиперпараметров

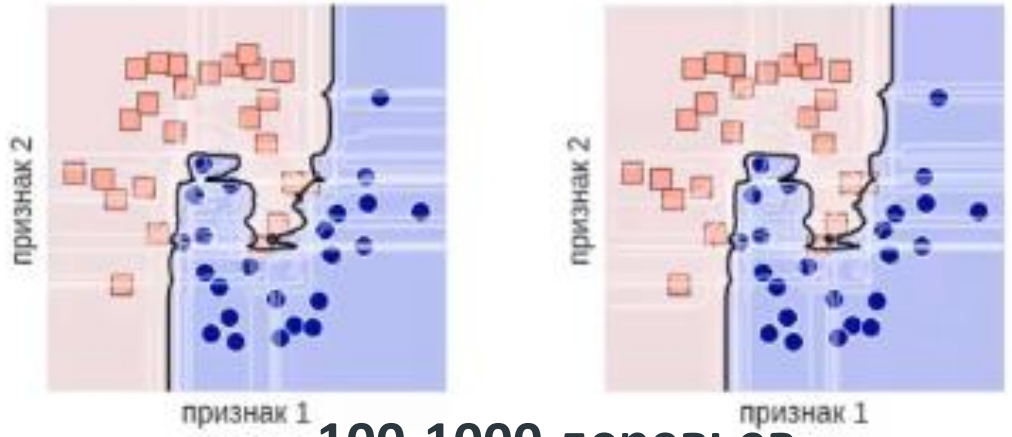
`criterion`

Энтропийный → $H(R) = -\sum_j p_j \log_2 p_j$

Джини → $H(R) = \sum_j p_j (1 - p_j) = 1 - \sum_j p_j^2$

$$\frac{|R|}{m}\left(H(R) - \frac{|R_{\text{left}}|}{|R|}H(R_{\text{left}}) - \frac{|R_{\text{right}}|}{|R|}H(R_{\text{right}})\right)$$

Различные критерии расщепления



`criterion='gini'` `criterion='entropy'`

100-1000 деревьев

В модельных задачах «на глаз» разницы не видно

В авторском коде был реализован Джини...

Настройка гиперпараметров

`max_features`

Число признаков при расщеплении – самый серьёзный гиперпараметр

Настраивается в первую очередь при достаточном числе деревьев

Зависимость унимодальная

По умолчанию часто:

$\sqrt{n}$ – классификация

$n/3$ – регрессия (в sklearn – n)

- Зависит от числа шумовых признаков
- Надо перенастраивать при добавлении новых признаков

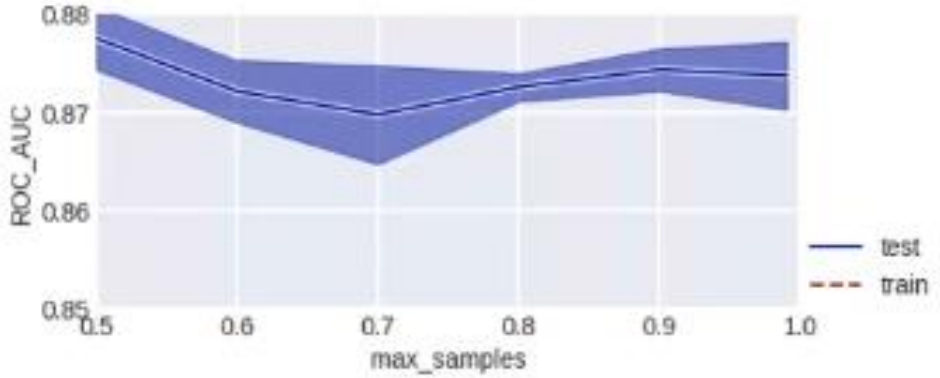
Чем больше – тем однотипнее деревья.

Чем больше – тем медленнее настройка!

Kaggle: часто суммируют алгоритмы с разными `max_features`

Настройка гиперпараметров

`max_samples`



В sklearn при `bootstrap=True` можно регулировать долю выборки

Часто «чем больше, тем лучше»

Чем больше – тем однотипнее деревья / дольше построение деревьев

- Можно настраивать гиперпараметр не в первую очередь
- Определитесь с типом выбора с возвратом / без возврата

Настройка гиперпараметров – «сложность»

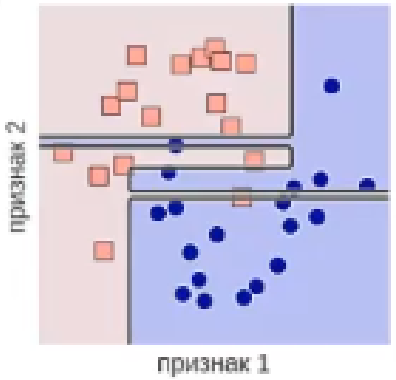
Гиперпараметры сложности:

- Число объектов в листе
- Число объектов для расщепления
- Максимальная глубина дерева

Оптимальные значения, как правило:

- Несколько объектов в листе
- Бесконечная глубина (как «в классике»)

«Good results are often achieved when setting `max_depth=None` in combination with `min_samples_split=1`»



↓

От гиперпараметров существенно зависит скорость построения леса

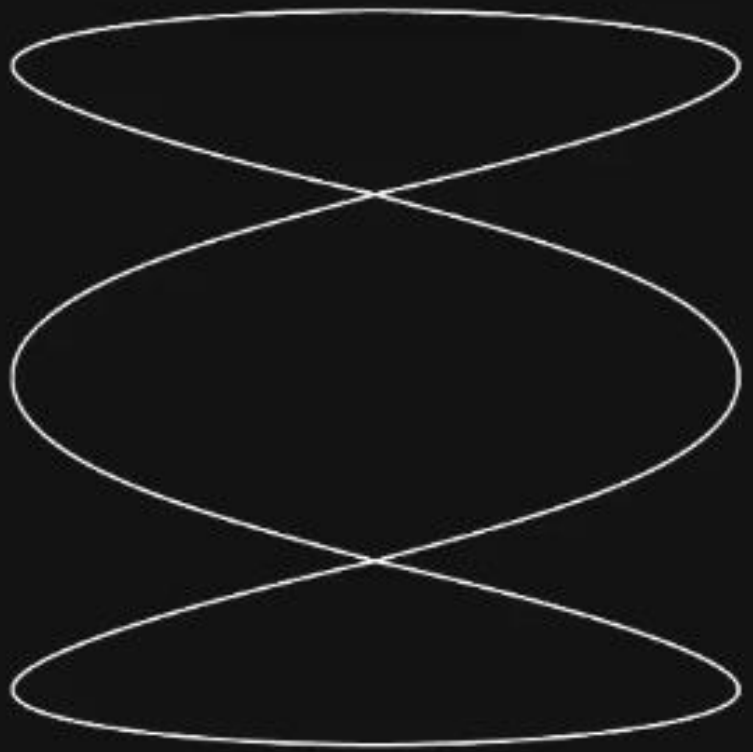
↓

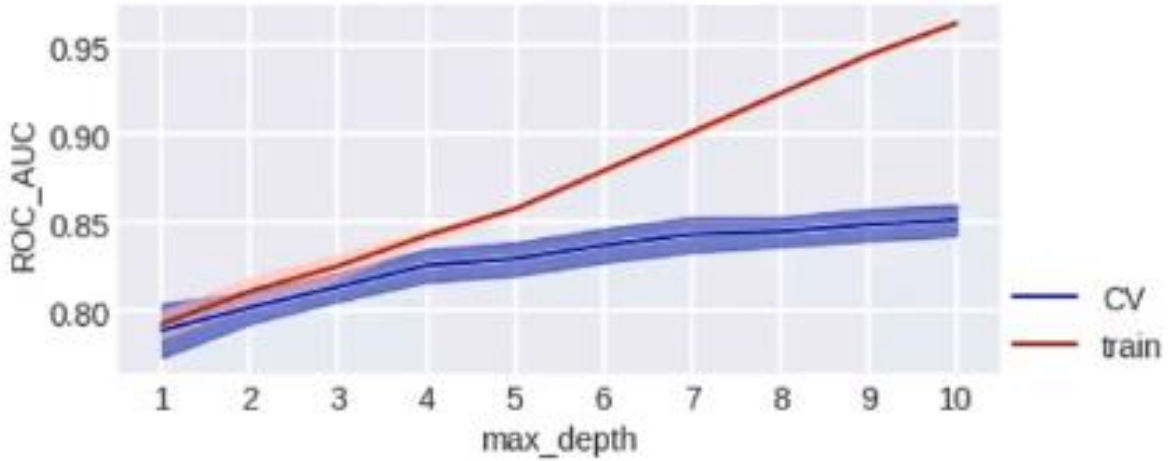
Настраиваются не в первую очередь

`min_samples_leaf` (1 – классификация, 5 – регрессия)

Настройка гиперпараметров

`max_depth`





Редко используются неглубокие деревья:

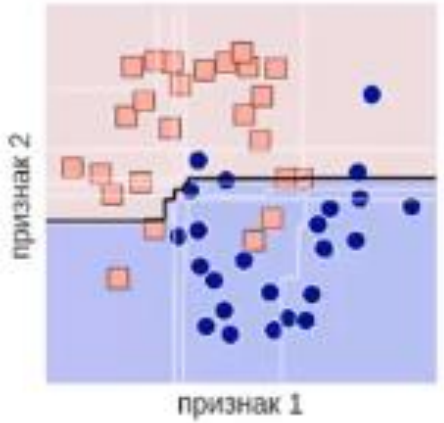
- В задачах с выбросами
- Когда много объектов
  (деревья большие и долго строятся)
- При этом настройка некоторых других гиперпараметров не имеет смысла

Каких?

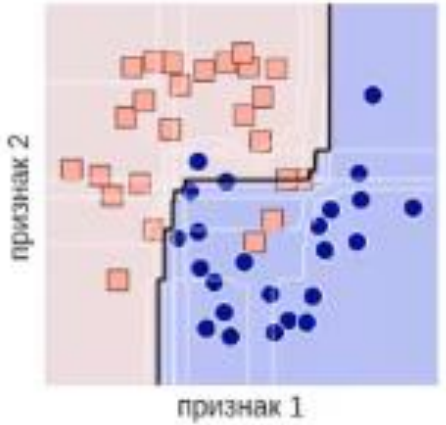
Настройка гиперпараметров

max_depth

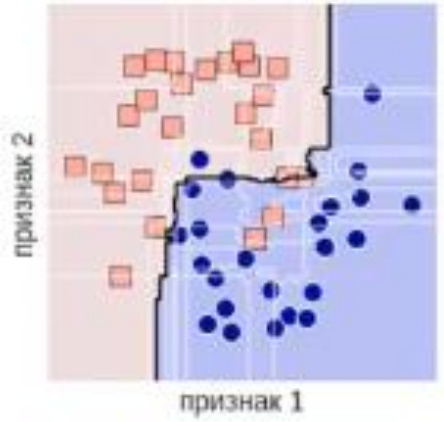
max_depth=1



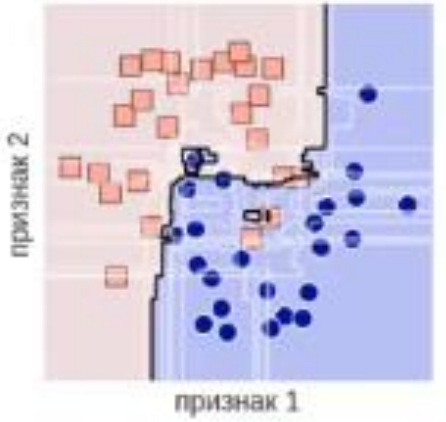
max_depth=2



max_depth=3



max_depth=4



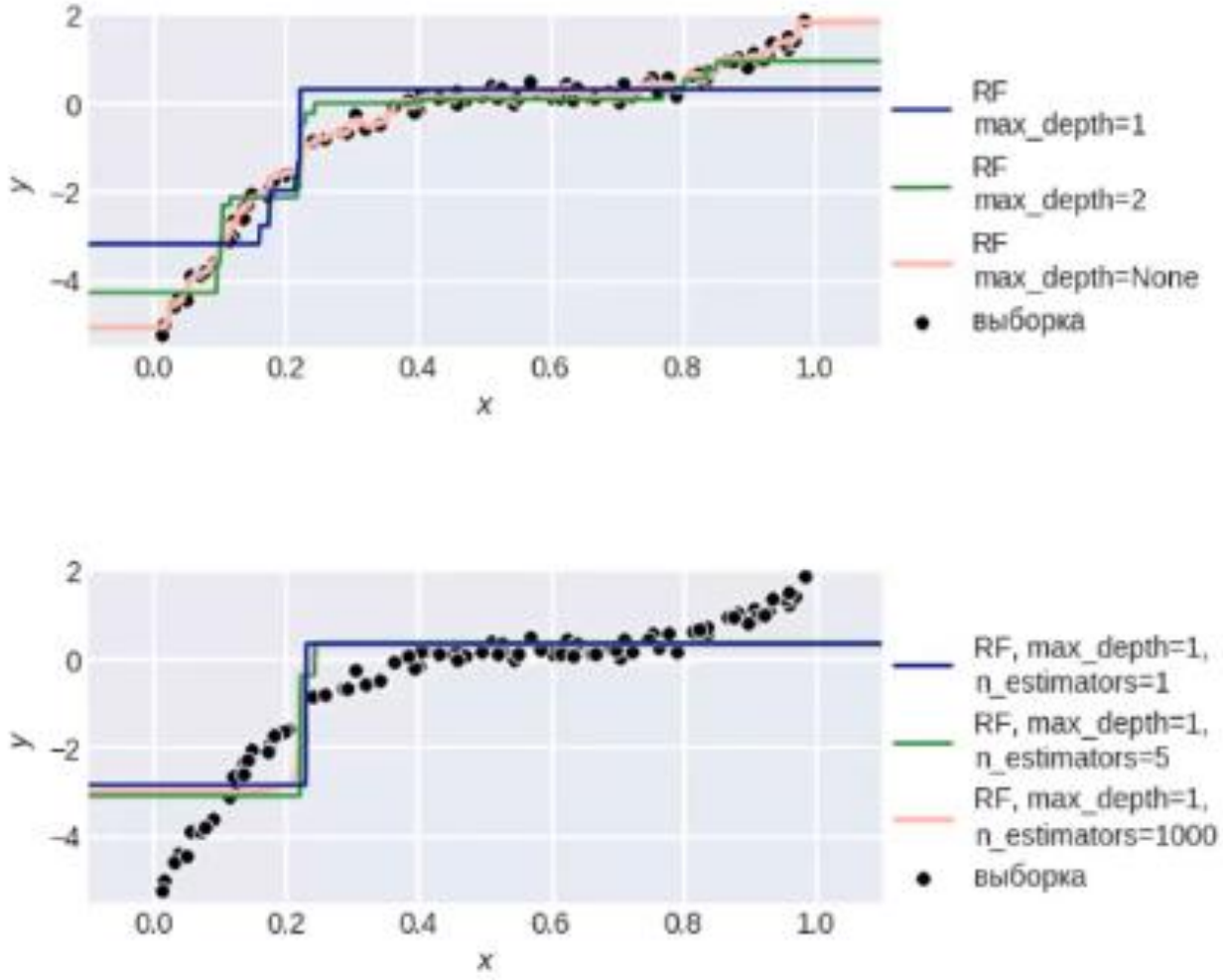
Extreme Random Trees (ExtraTrees)

- Нет бутстрепа (используем всю выборку)
- Генерируем несколько пар (признак, порог) и выбираем оптимальную для разбиения пару
- Также есть гиперпараметр **max_features** «число признаков для просмотра»

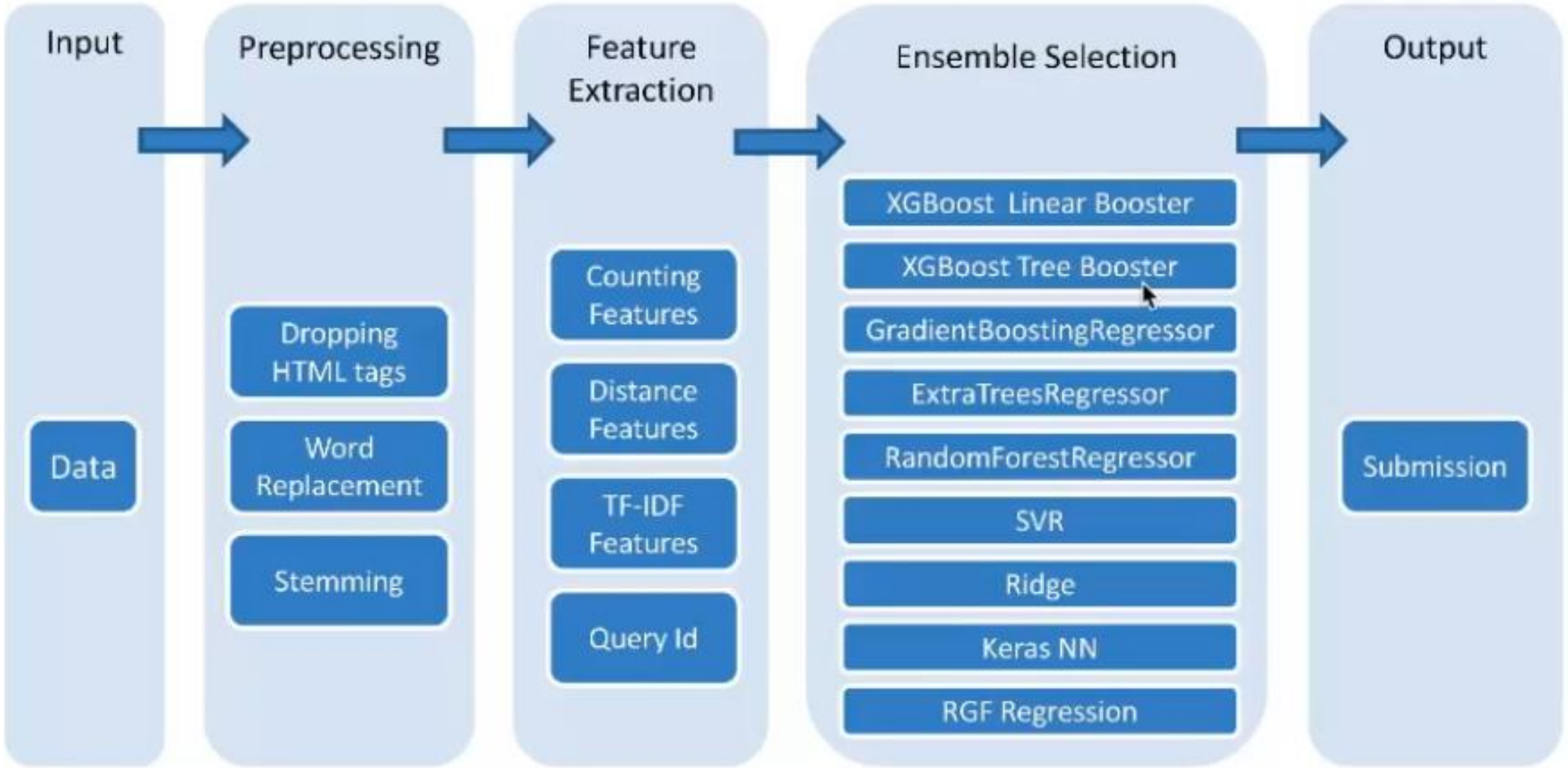
- ET обучается быстрее RF
- ET чуть хуже RF, когда много шумных признаков

```
from sklearn.ensemble import ExtraTreesClassifier
clf = ExtraTreesClassifier(n_estimators=10, max_depth=None,
                           min_samples_split=2, random_state=0)
```

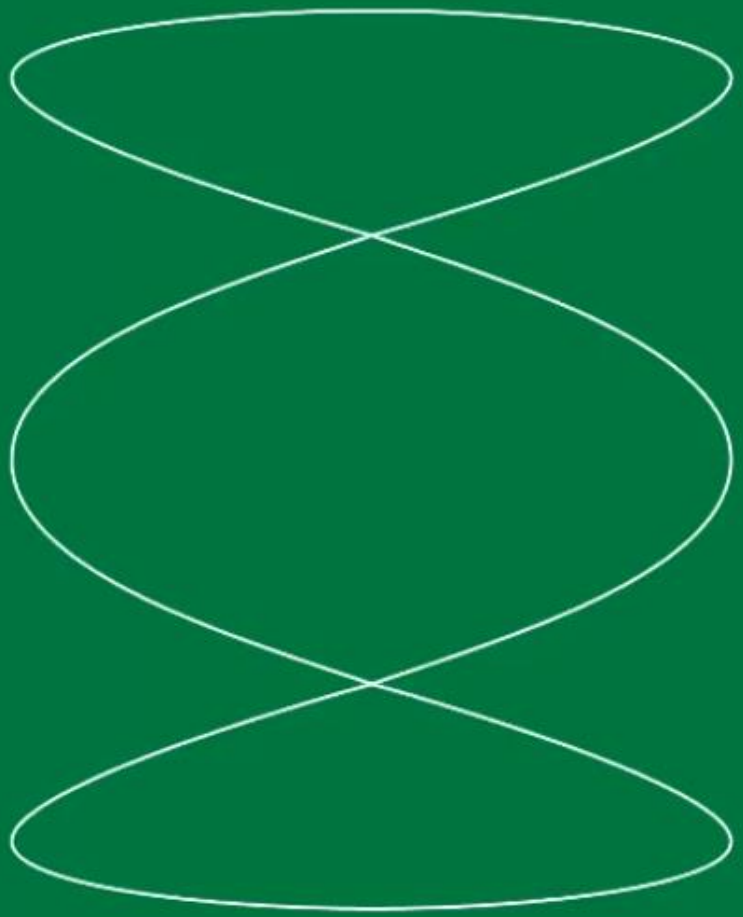
Когда плохи методы, основанные на деревьях...



Ансамбль над ансамблями



Итог



- **Число признаков (max_features)** – самый важный гиперпараметр (унимодальность)
- **Число базовых алгоритмов (n_estimators)** – чем больше, тем лучше
- **Глубина (max_depth)** – скорее всего, максимальная
- **Гиперпараметры сложности (min_samples_leaf, min_samples_split)** – чуть подкорректировать
- **Подвыборка (samsize)** – чуть подкорректировать (часто и выбора нет)
- **Леса наследуют главный недостаток деревьев – плохи для экстраполяции**

Спасибо за внимание!

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