



Лекция 06

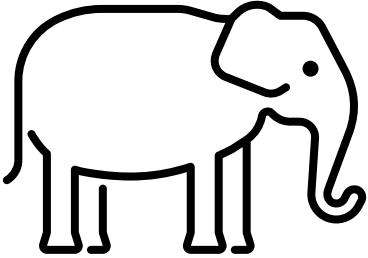
Деревья решений и ансамбли

- А. Деревья решений
- Б. Беггинг и Стекинг
- В. Бустинг

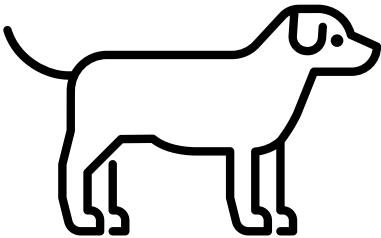


Деревья решений

Линейная регрессия



$\text{Вес} \cdot 0.73 + \text{Рост} \cdot 0.54 + \text{Хобот} \cdot 1.25 > 14.23$



$\text{Вес} \cdot 0.73 + \text{Рост} \cdot 0.54 + \text{Хобот} \cdot 1.25 < 14.23$

Человек

ЕСЛИ Вес > 3.75
И Рост > 1.29
И Хобот = есть
ТОГДА Слон

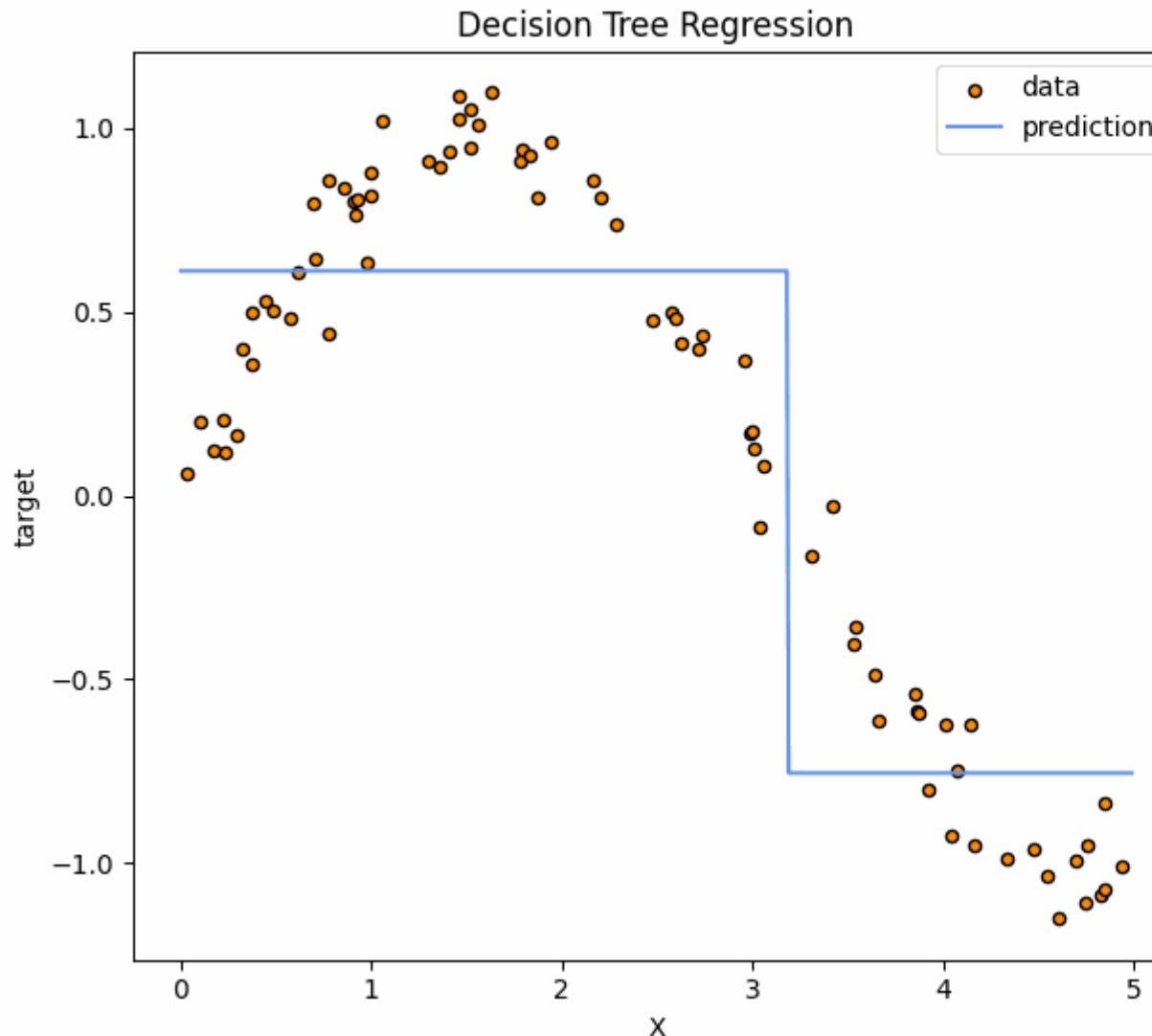
ЕСЛИ Вес < 3.75
И Рост < 1.29
И Хобот = нет
ТОГДА Собака



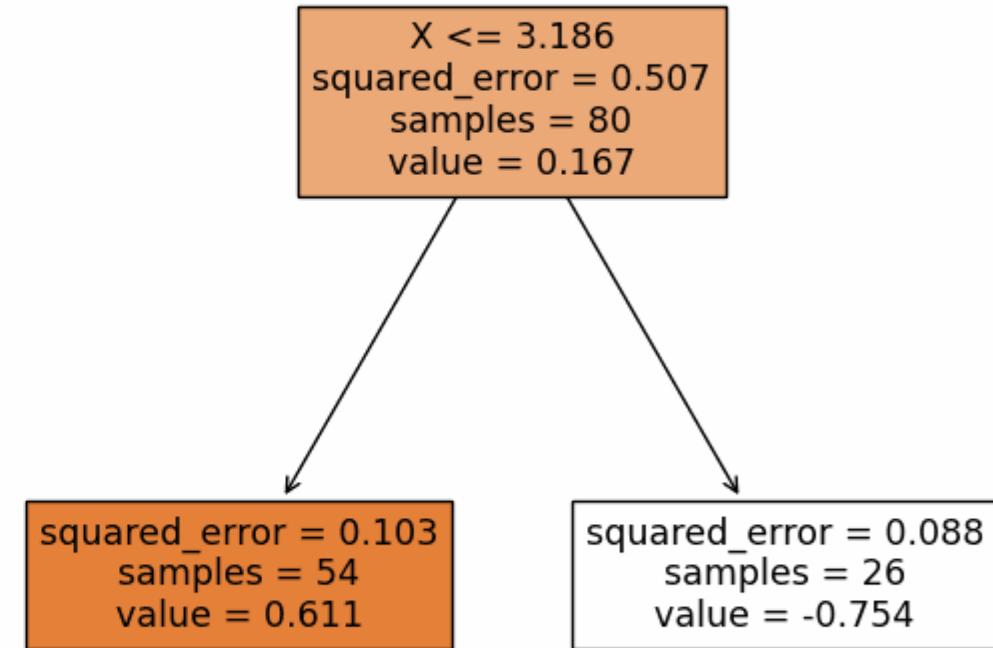
Линеаризация

3

Decision tree regressor with max depth=1



Decision Tree Structure

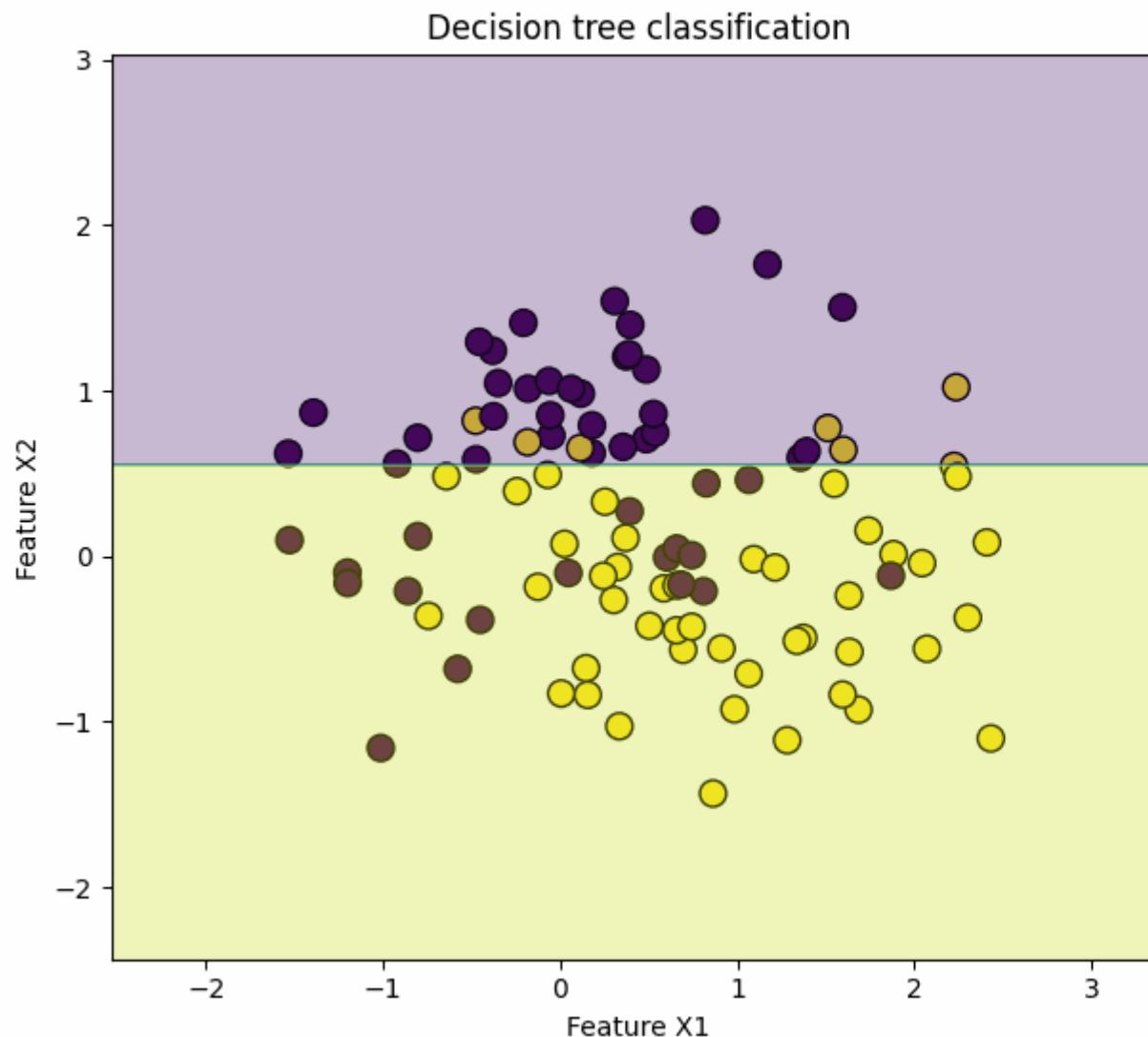




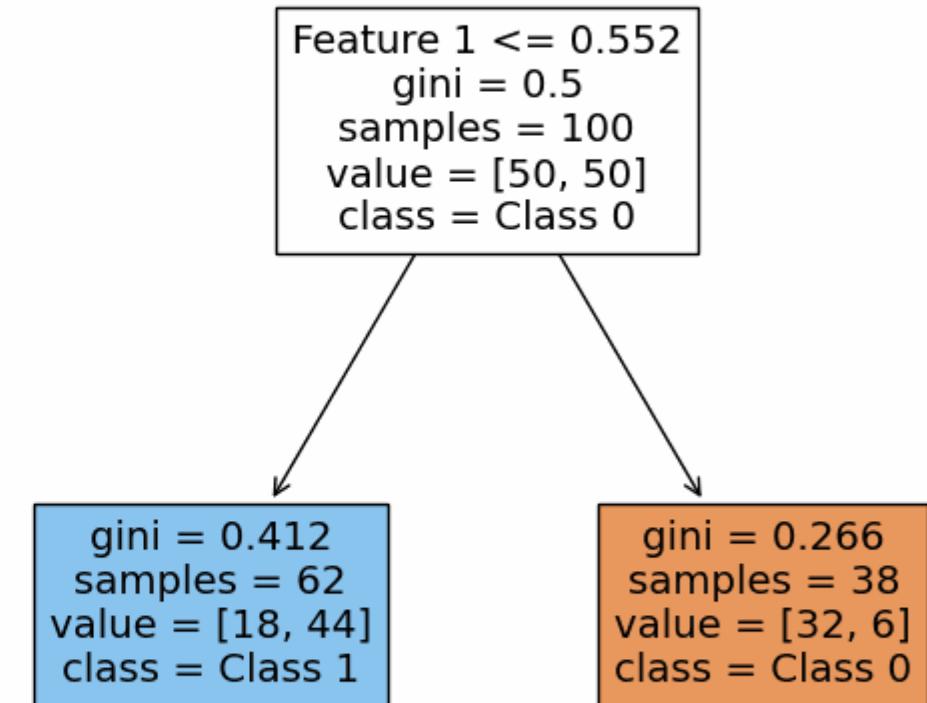
Линеаризация

4

Decision tree classifier for 2D feature with max_depth=1



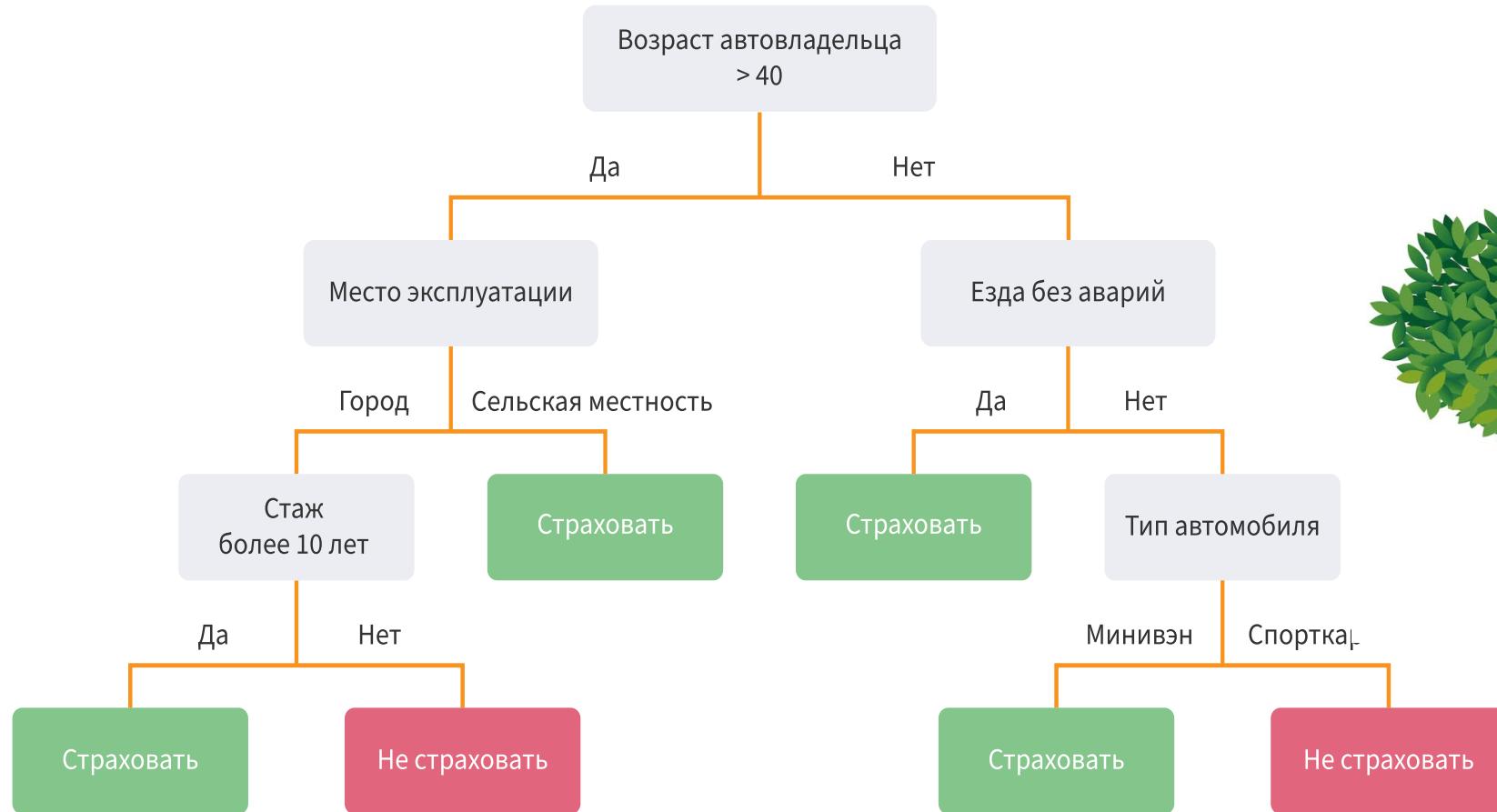
Decision Tree Structure





Деревья решений. Правила ЕСЛИ...ТОГДА

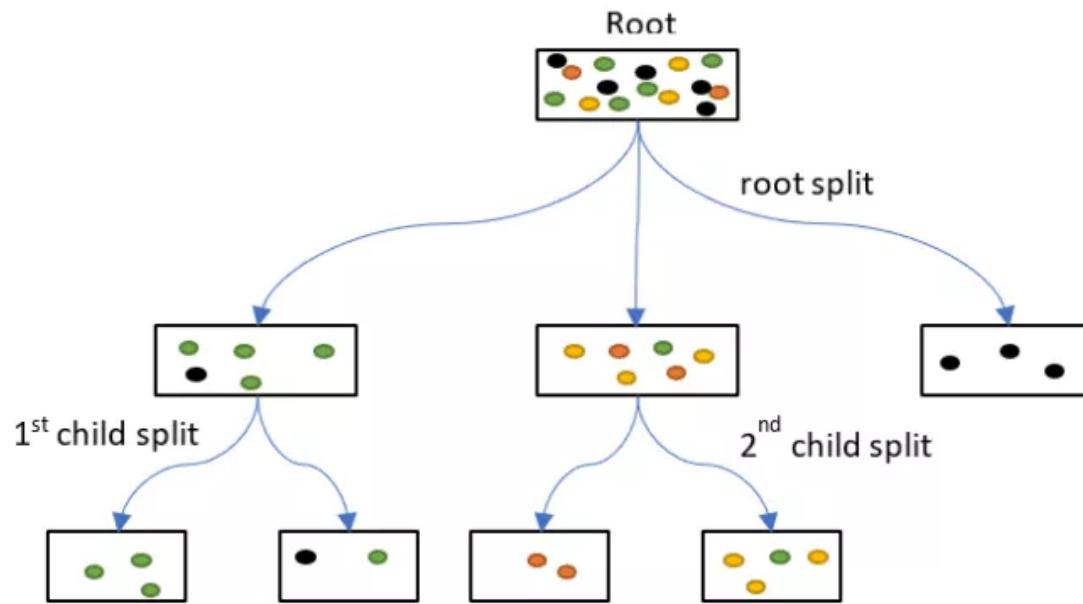
5



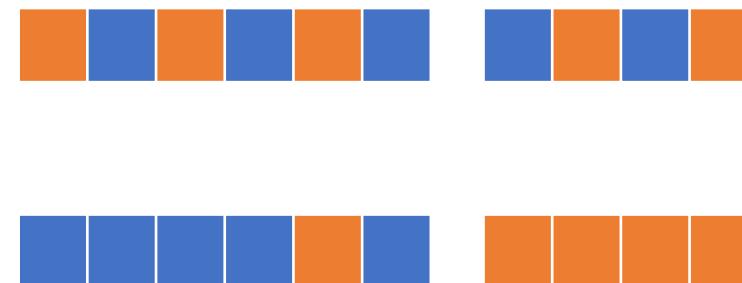


Деревья решений. Обучение CART

6



Признак $x_{ij} < \text{Порог } t_i$?
ДА / НЕТ



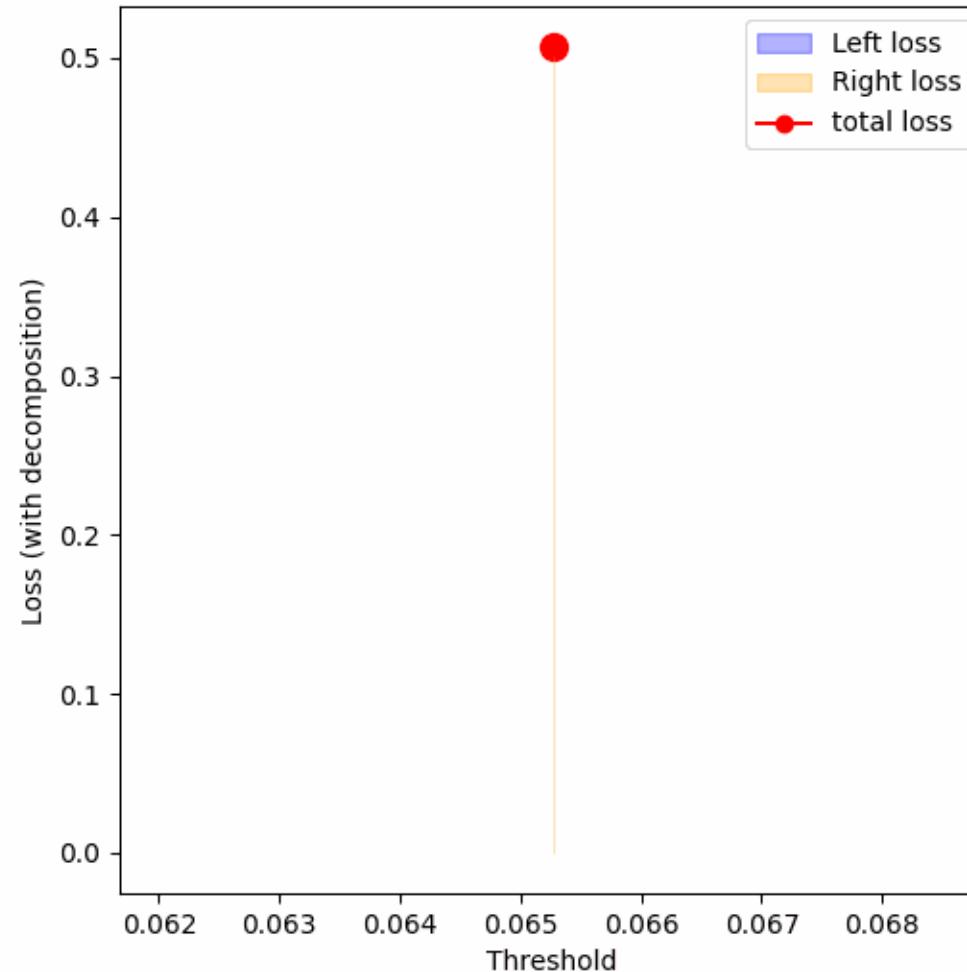
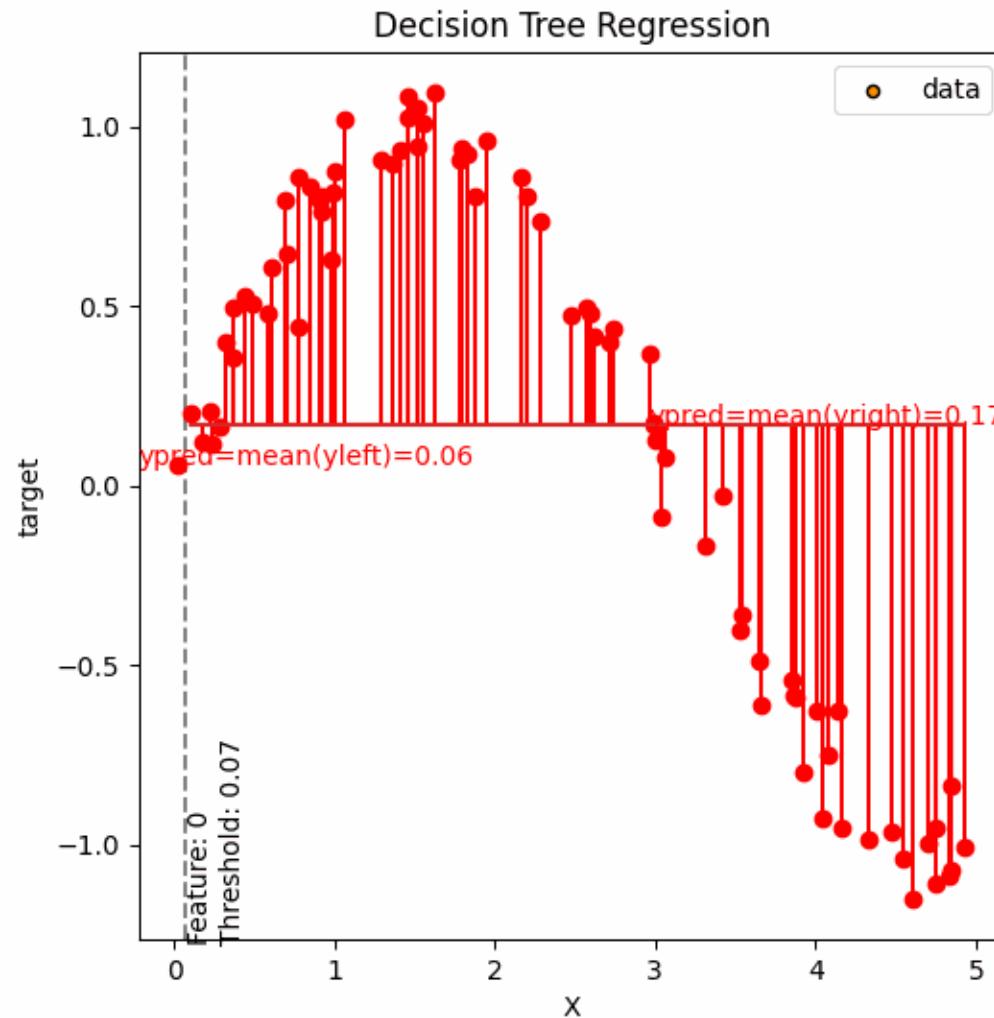
Неопределенность G

(число примеров в Ql)/(общее число примеров в Q)* H(Ql) +
(число примеров в Qr)/(общее число примеров в Q)* H(Qr)



Линеаризация

7





Деревья решений. Обучение CART

8

Неопределенность G

(число примеров в QI)/(общее число примеров в Q)* H(QI) +
(число примеров в Qr)/(общее число примеров в Q)* H(Qr)

КЛАССИФИКАЦИЯ

Неопределенность Джини

$I = 1$ если совпадает, 0 если не совпадает

$$p_k = \frac{1}{N_m} * \sum_{x_i \in Q_m} I(y_i = k),$$

$$H = \sum_k p_k * (1 - p_k)$$

Энтропия

$$H = - \sum_k p_k * \log(p_k)$$

Ошибка классификации

$$H = 1 - \max(p_k)$$

РЕГРЕССИЯ

$$y_{mean} = \frac{1}{N_m} \sum_{i \in N_m} y_i$$

$$H = \frac{1}{N_m} \sum_{i \in N_m} (y_i - y_{mean})^2$$



Деревья решений. Обучение CART

9





Деревья решений. Реализация sklearn

10

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, class_weight=None, ccp_alpha=0.0)
```

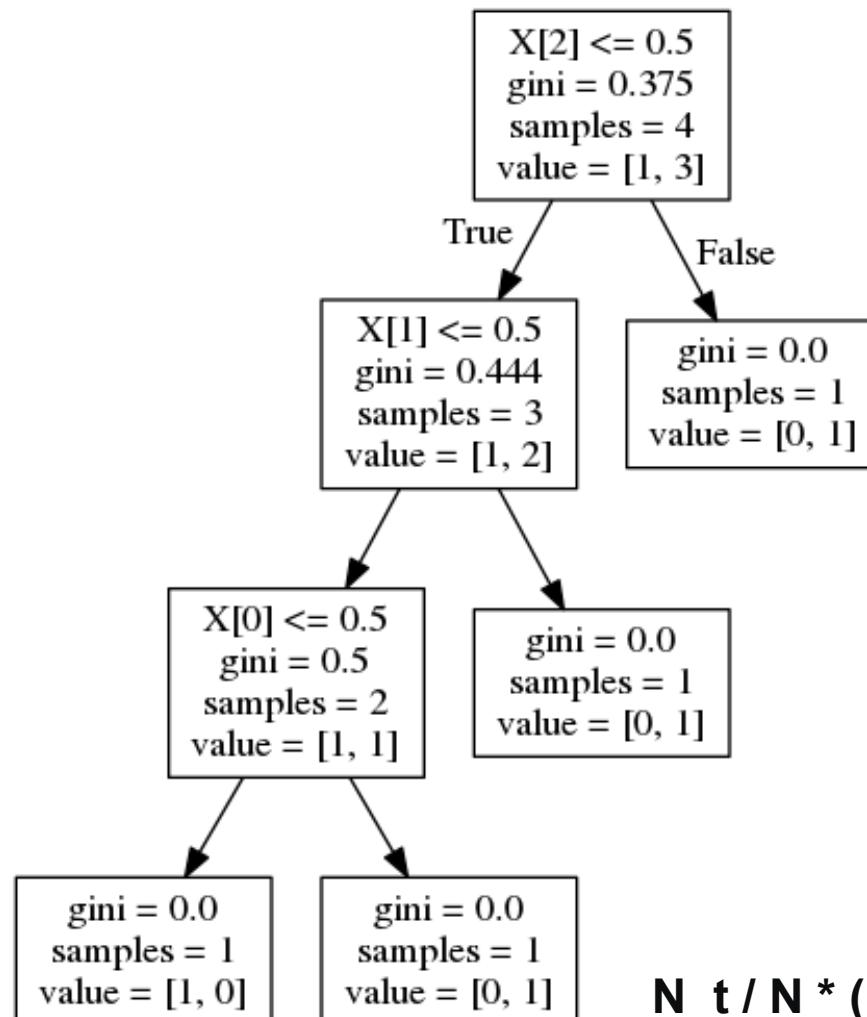
[source]

```
clf = tree.DecisionTreeClassifier() # создаем классификатор на основе дерева  
clf = clf.fit(X, y) # обучаем его, т.е. создаем само дерево  
tree.plot_tree(clf) # отображаем
```



Важность признаков

11



feature_importances

For $X[2]$:

$$\text{feature_importance} = (4 / 4) * (0.375 - (0.75 * 0.444)) = 0.042$$

For $X[1]$:

$$\text{feature_importance} = (3 / 4) * (0.444 - (2/3 * 0.5)) = 0.083$$

For $X[0]$:

$$\text{feature_importance} = (2 / 4) * (0.5) = 0.25$$

N – общее число примеров

N_t – примеров в текущем узле

N_{t_L} – примеров в левом потомке

N_{t_R} – примеров в правом потомке

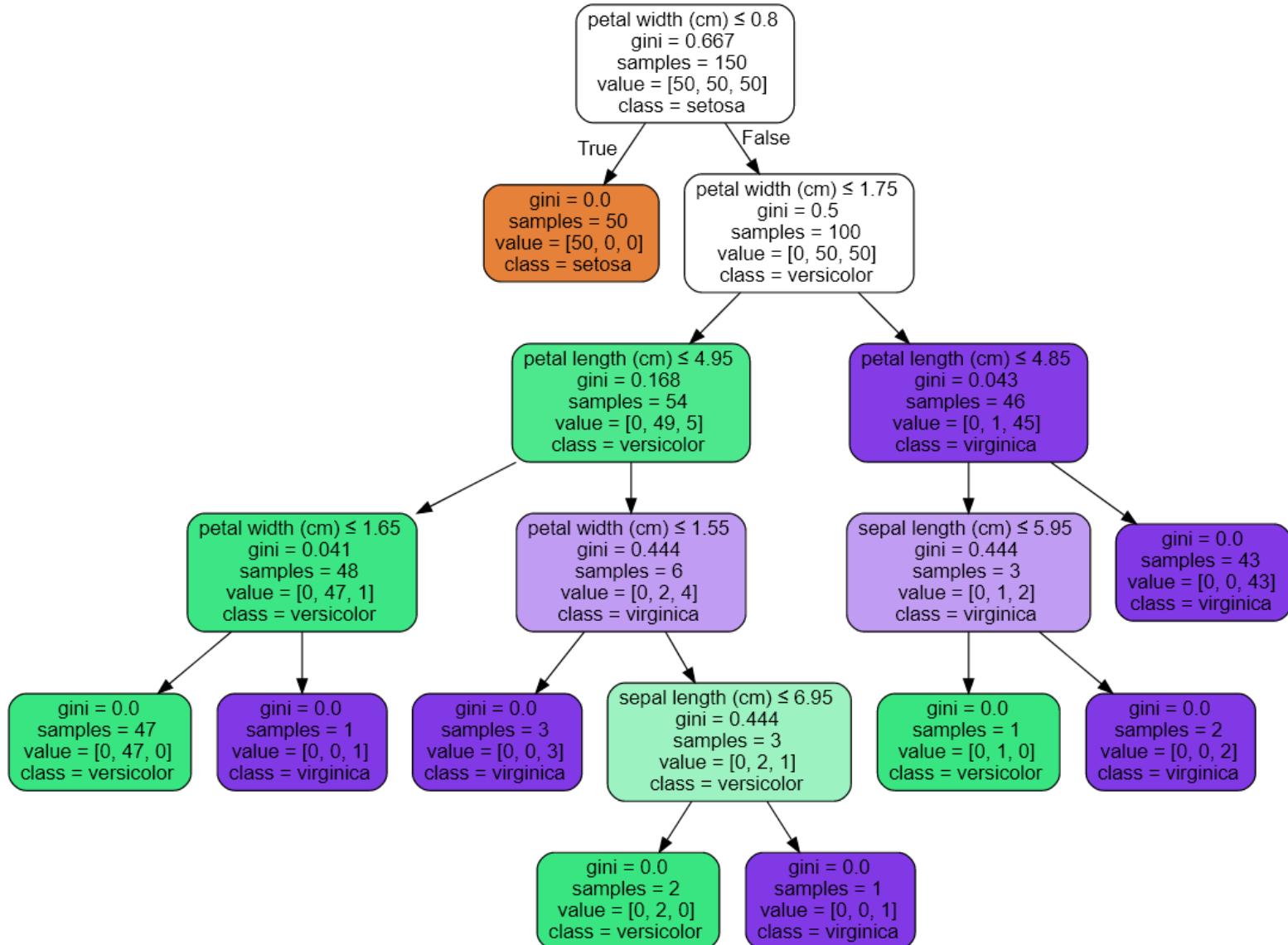
$$N_t / N * (\text{impurity} - N_{t_R} / N_t * \text{right_impurity} - N_{t_L} / N_t * \text{left_impurity})$$



Деревья решений. Реализация. Результаты

12

12

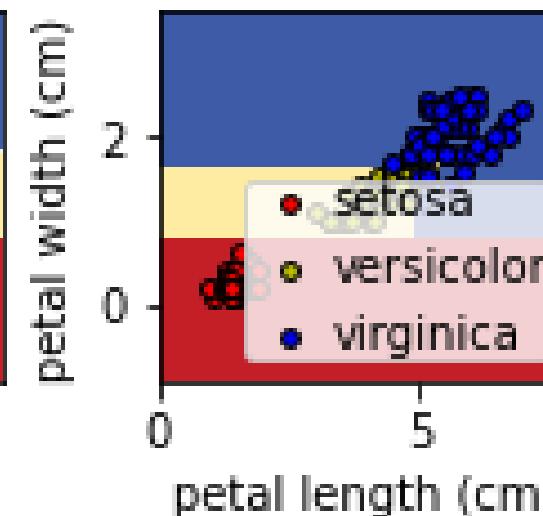
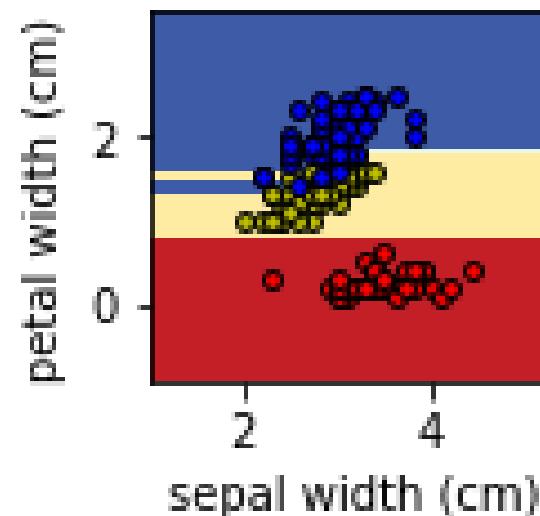
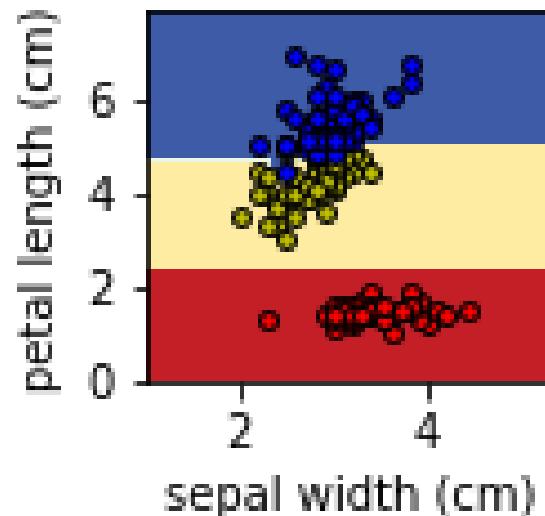
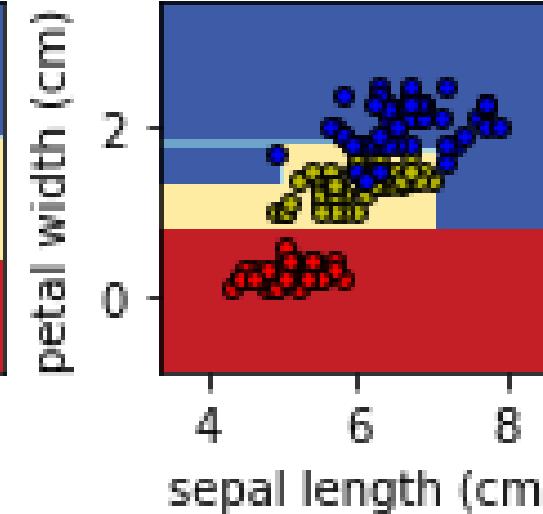
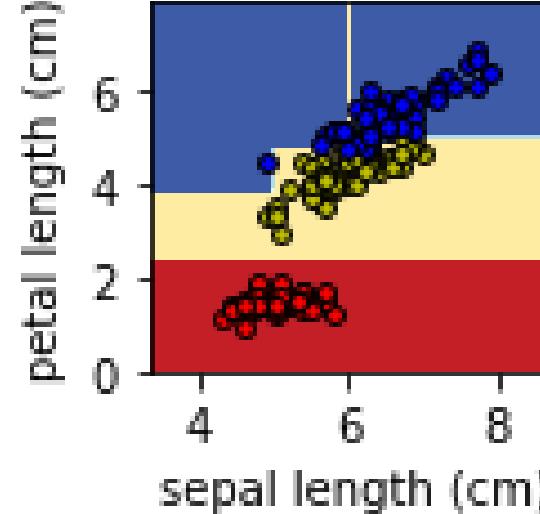
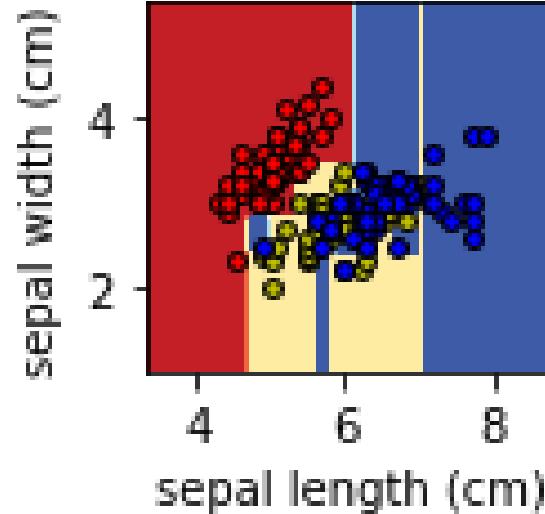




Деревья решений. Разделяющая поверхность

13

Decision surface of a decision tree using paired features





Деревья решений. Проблемы

14

ПРЕИМУЩЕСТВА:

- Интуитивно понятны и объяснимы
- Низкие требования к предобработке данных, нормализации
- Простая и понятная реализация

НЕДОСТАТКИ:

- Переобучение и чувствительность к данным
- Большое время обучения (перебора)

Совокупность деревьев – лес



Один в поле не воин

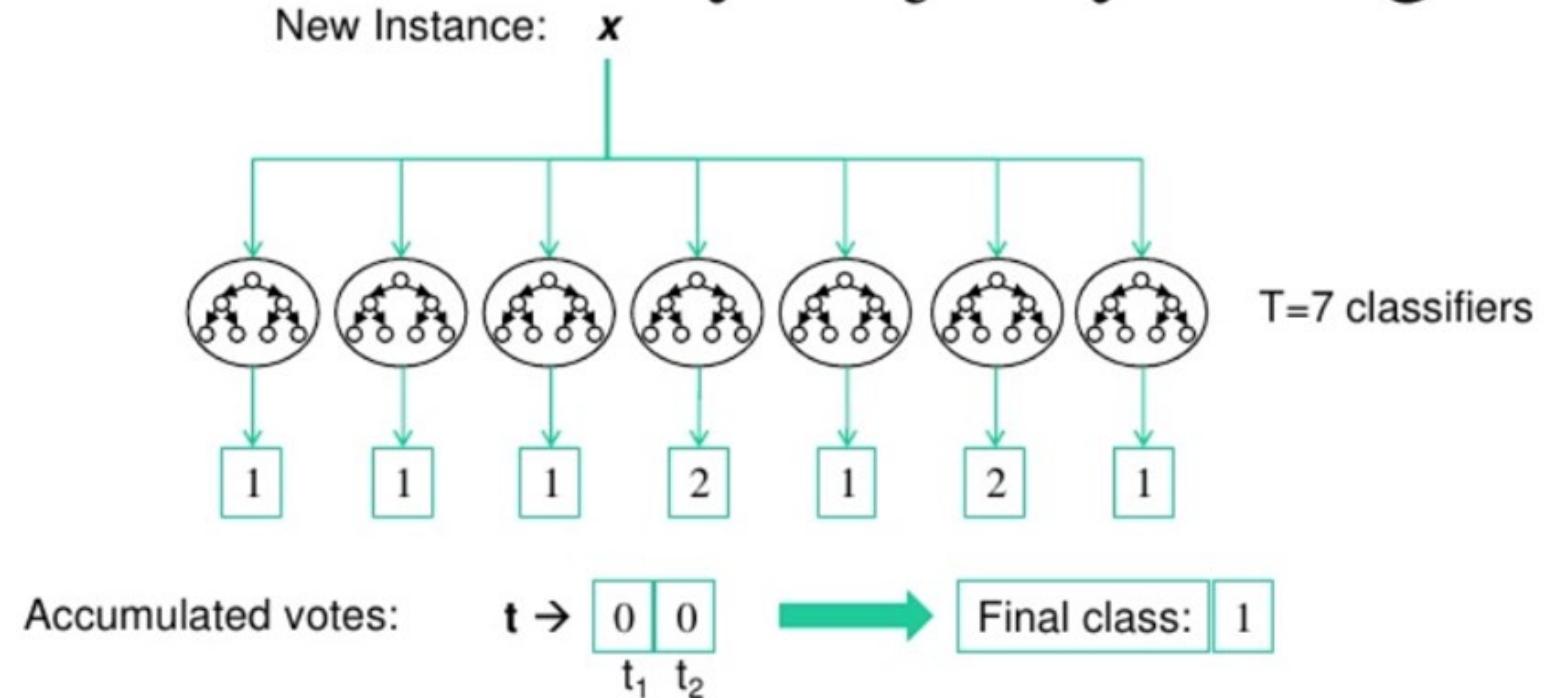


Требуется РАЗНООБРАЗИЕ!

Ансамбли моделей. Бэггинг Правило толпы

15

Classification by majority voting





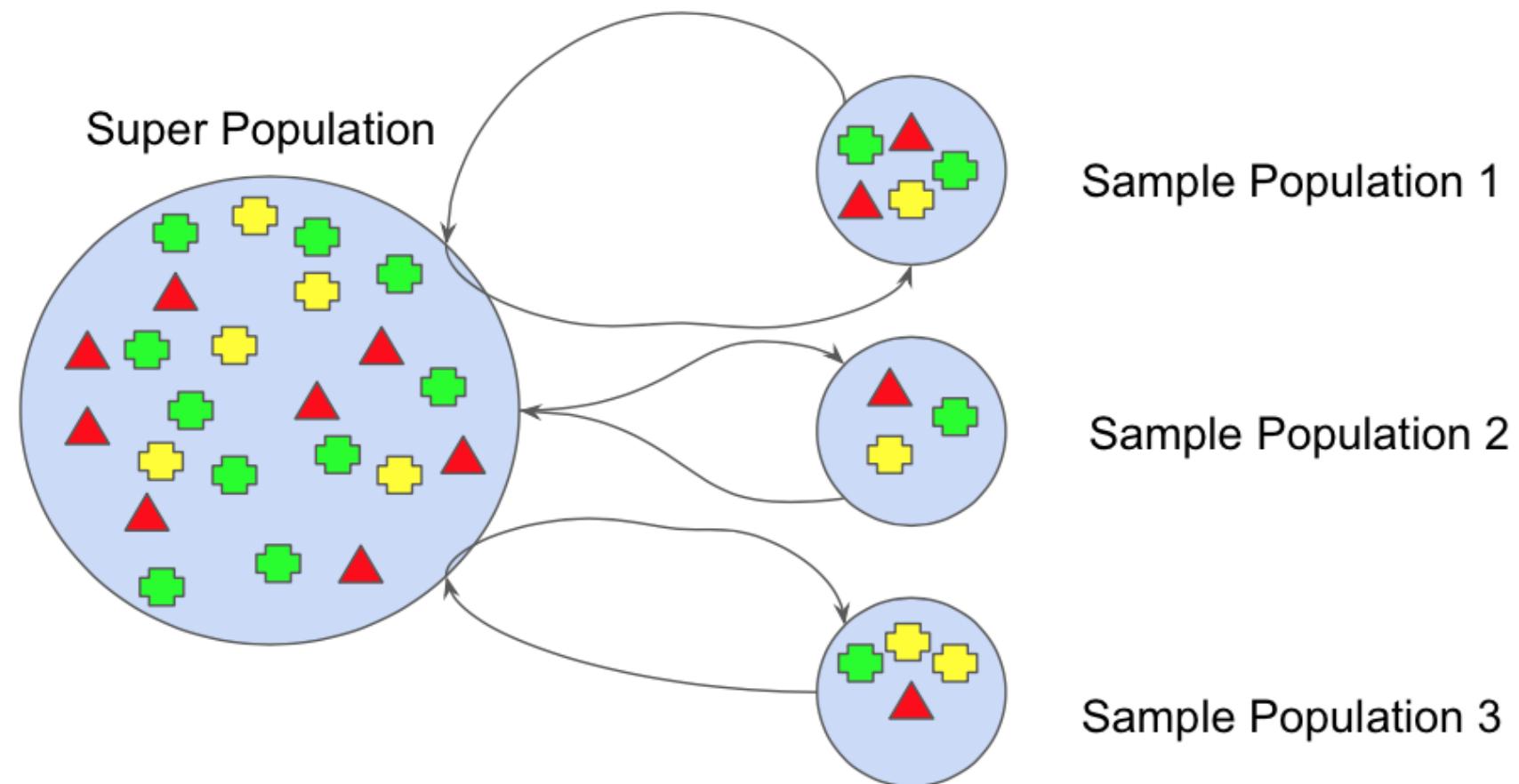
Ансамбли. Беггинг. Bootstrap

16

Красный цвет раздражает быков ? - Заблуждение



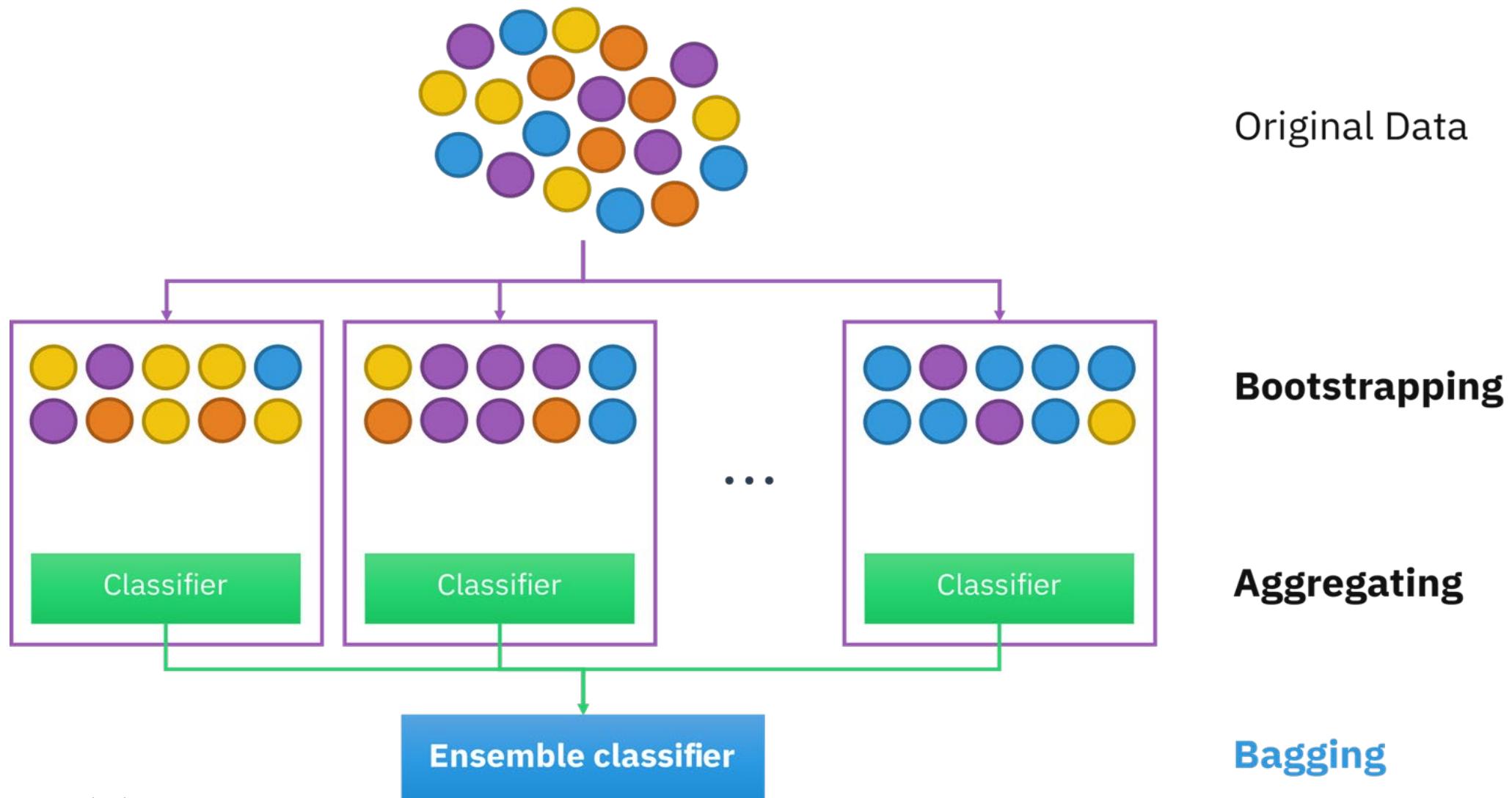
<https://delusionbook.ru/byk-i-krasnaya/>





Ансамбли. Бэггинг.

17





sklearn.ensemble.BaggingClassifier

```
class sklearn.ensemble.BaggingClassifier(base_estimator=None, n_estimators=10, *, max_samples=1.0,  
max_features=1.0, bootstrap=True, bootstrap_features=False, oob_score=False, warm_start=False, n_jobs=None,  
random_state=None, verbose=0)
```

[source]

- `base_estimator` - объект типа модели в ансамбле (все модели будут одного типа), по умолчанию - дерево.
- `n_estimators` - максимальное число моделей в ансамбле
- `max_features` - число или доля признаков в подвыборке моделей
- `max_samples` - число или доля примеров в подвыборке моделей (примеры выбираются случайно).
- `oob_score` - разрешить ли использование примеров не вошедших в подвыборку (`oob = out of bag`) для оценки ошибки модели, по умолчанию нет, `False`

- `base_estimator_` - тип моделей в ансамбле
- `estimators_` - список моделей в ансамбле
- `classes_` - метки классов
- `n_classes_` - число классов

```
from sklearn.ensemble import BaggingClassifier #  
from sklearn.tree import DecisionTreeClassifier # дерево для регрессии  
clf=BaggingClassifier(DecisionTreeClassifier(), n_estimators=10,max_samples=0.5)
```

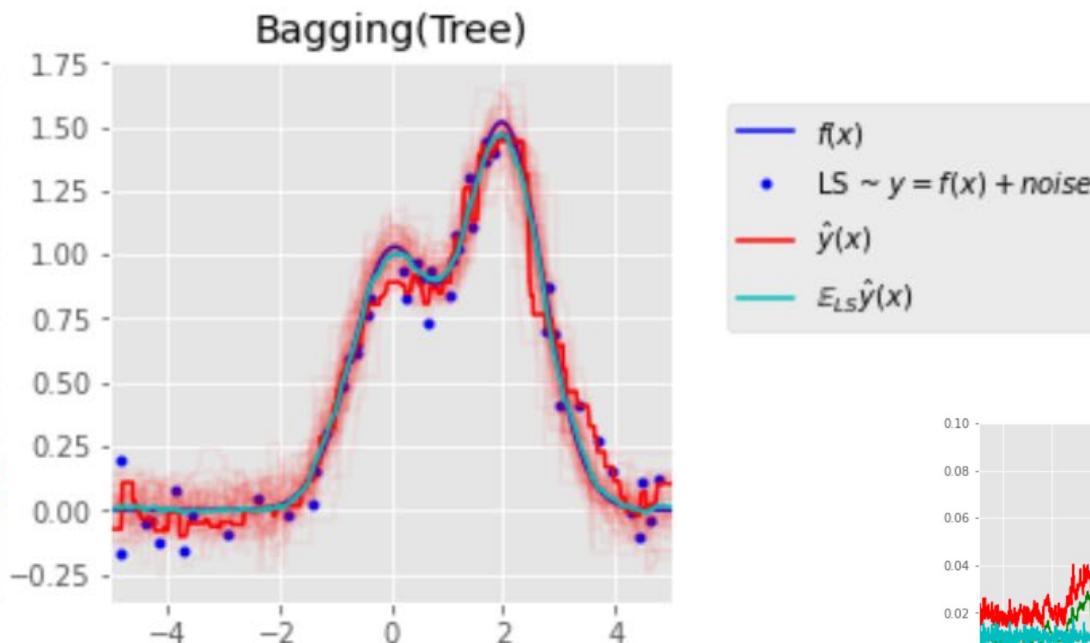
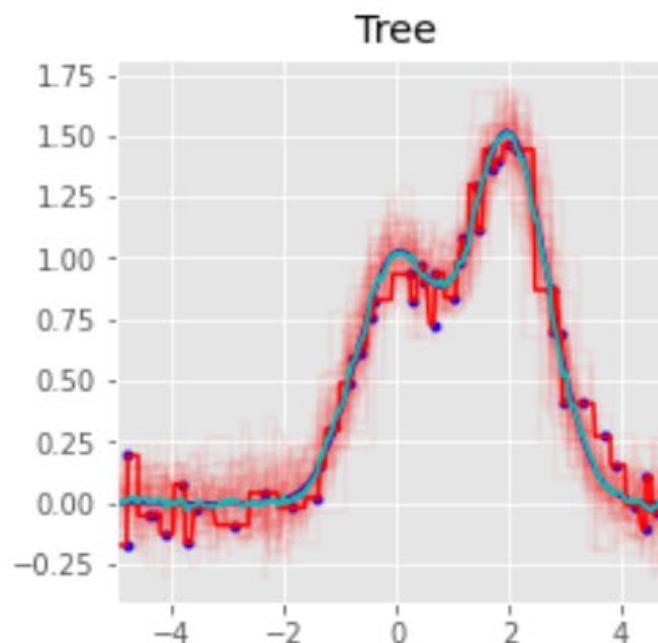


Ансамбли. Бэггинг. Ошибка

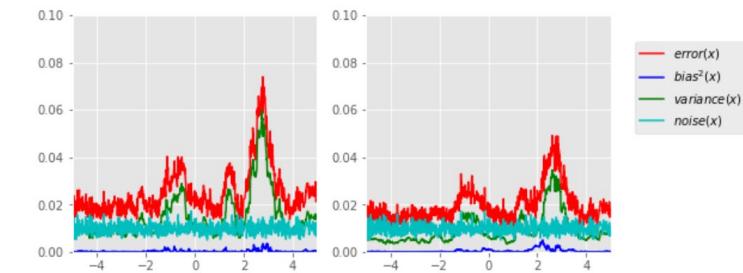
19

$$\text{ошибка} = \text{смещение}^2 + \text{дисперсия} + \text{шум}$$

| Tree : 0.0255 (error) = 0.0003 (bias²) + 0.0152 (var) + 0.0098 (noise)
Bagging(Tree): 0.0196 (error) = 0.0004 (bias²) + 0.0092 (var) + 0.0098 (noise)



— $f(x)$
• $LS \sim y = f(x) + noise$
— $\hat{y}(x)$
— $E_{LS}\hat{y}(x)$





Случайный лес. Модель

20

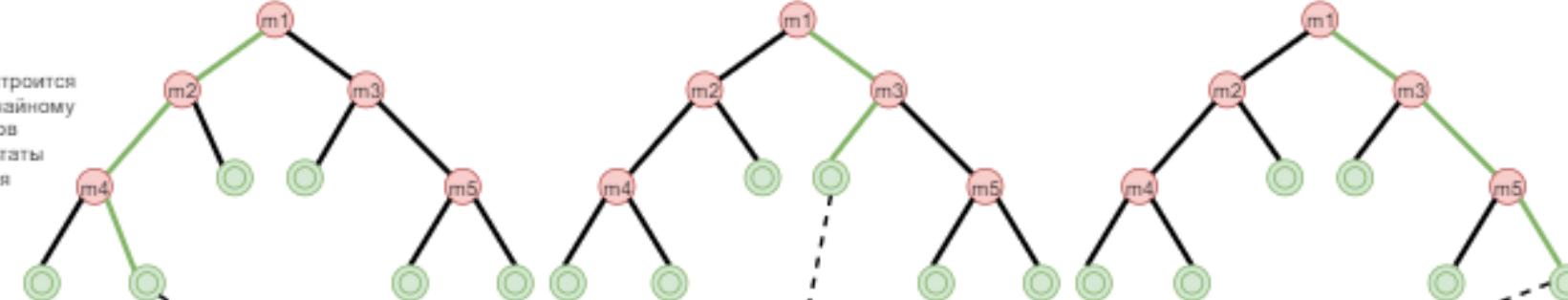
Bootstrap sampling

выбирается г (процент) примеров
(0.63 в классической реализации) в
п случайных подвыборок



Building the models

по каждой подвыборке строятся
дерево решений по случайному
набору m признаков
(ковариатов), результаты
попадают в листья



Bootstrap aggregating

собираются результаты со всех
построенных деревьев решений и
усредняются





Случайный лес. Реализация sklearn

21

sklearn.ensemble.RandomForestClassifier

```
class sklearn.ensemble.RandomForestClassifier(n_estimators=100, *, criterion='gini', max_depth=None,  
min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='sqrt', max_leaf_nodes=None,  
min_impurity_decrease=0.0, bootstrap=True, oob_score=False, n_jobs=None, random_state=None, verbose=0,  
warm_start=False, class_weight=None, ccp_alpha=0.0, max_samples=None)
```

[source]

- **n_estimators** - максимальное число деревьев в лесу
- **criterion** - критерий для создания узла: или критерий Джини **gini** (по умолчанию), или энтропия **entropy**.
- **max_depth** - максимальная глубина дерева
- **max_features** - максимальное число атрибутов, которые будут проверены при создании узла, по умолчанию это равно корню квадратному из числа всех атрибутов в данных.
- **max_samples** - максимальное число примеров используемых для одного дерева (примеры выбираются случайно).

- **estimators_** - список объектов деревьев (типа **DecisionTreeClassifier**) в этом лесу

- **classes_** - метки классов

- **n_classes_** - число классов

- **n_features_** - число атрибутов

- **n_outputs_** - число выходов

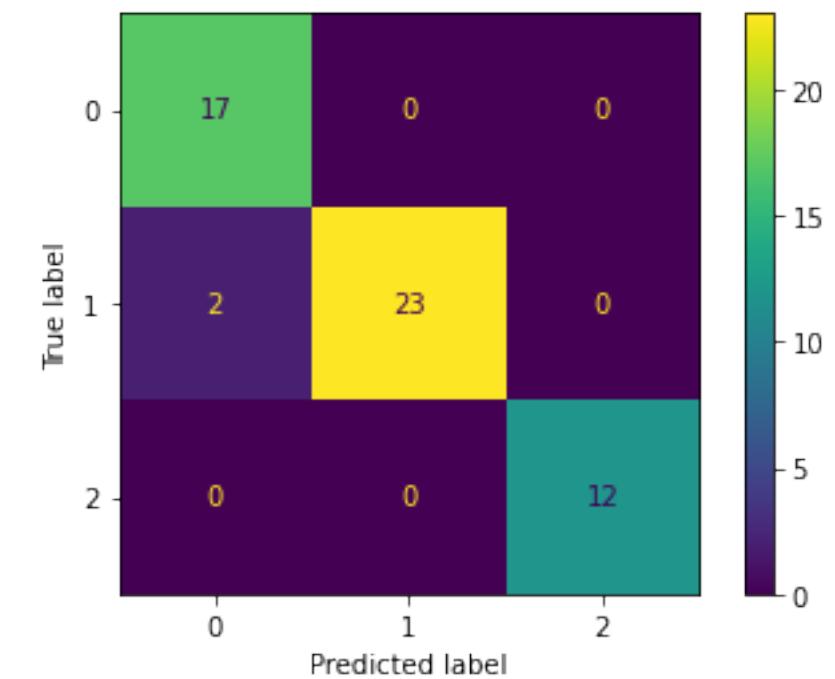
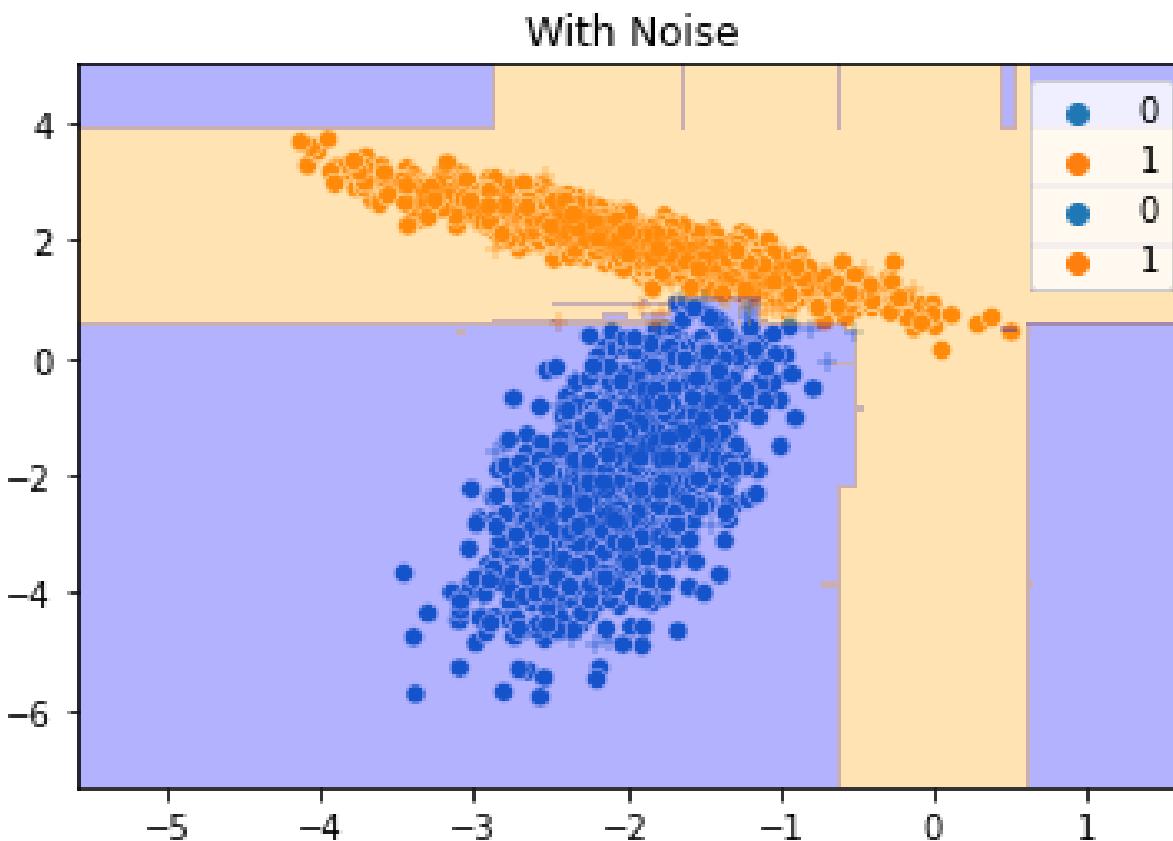
- **feature_importances_** - оценка важности атрибутов.

```
# Создаем классификатор на основе случайного леса. Изменяйте параметры  
clf = RandomForestClassifier(max_depth=5,# максимальная глубина дерева  
n_estimators=10,# число деревьев  
max_features=1)# максимальное число атрибутов, проверяемые при создании узла  
  
clf.fit(X_train, y_train) # обучаем  
  
y_pred = clf.predict(X_test) # проверяем на тестовых данных
```



Случайный лес. Результаты и разделяющая поверхность

22





Ансамбли моделей. Стекинг

23

Бэггинг.

Модели одного типа строятся и работают вместе, параллельно



Бустинг.

Модели одного типа строятся последовательно, компенсируя ошибки предыдущих



Стекинг.

Модели разного типа строятся и обучаются вместе, параллельно, решение принимает другая модель

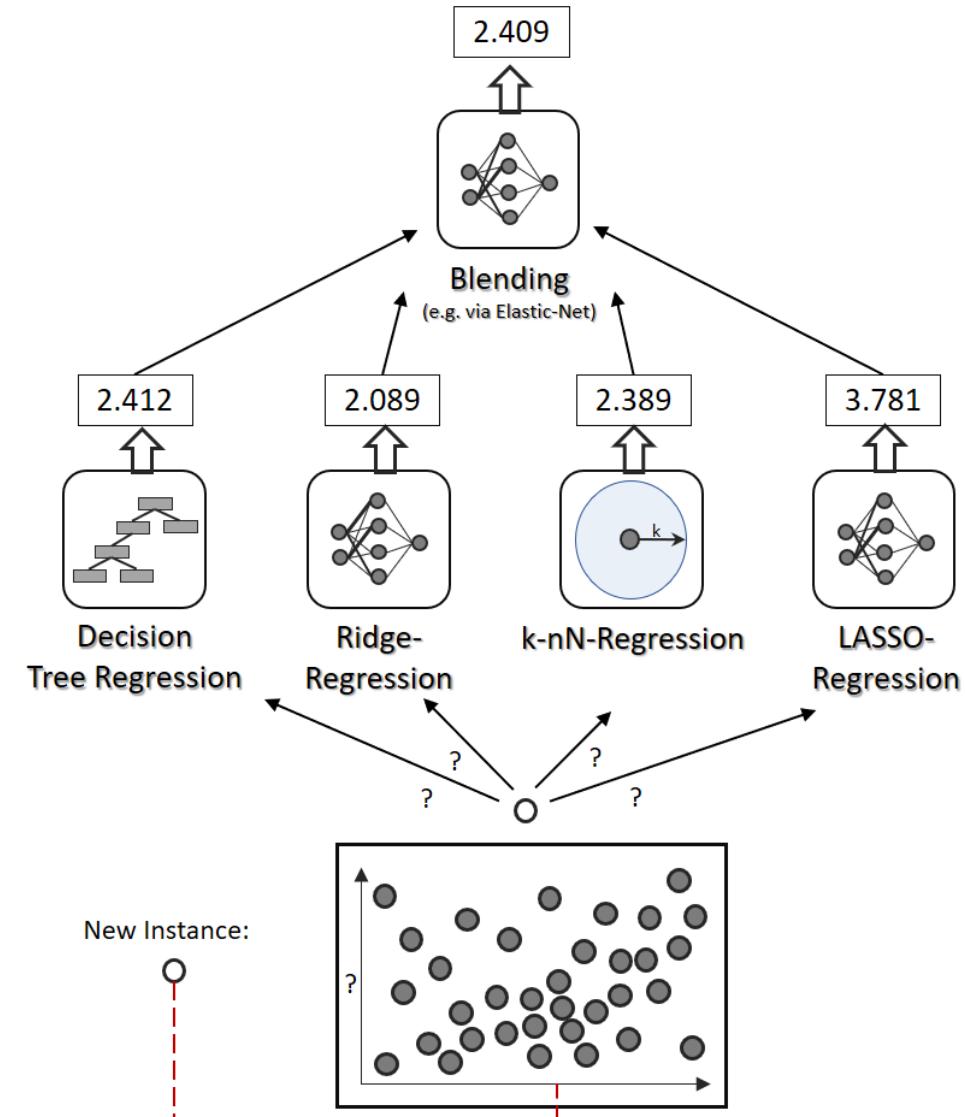




Ансамбли моделей. Стекинг

Stacking

24





sklearn.ensemble.StackingClassifier

```
class sklearn.ensemble.StackingClassifier(estimators, final_estimator=None, *,  
cv=None, stack_method='auto', n_jobs=None, passthrough=False,  
verbose=0)
```

[source]

estimators - кортеж из названий и объектов моделей ансамбля

final_estimator - модель для объединения результатов

cv - число разбиений для кроссвалидации (или объекты)

stack_method - по какому именно результату объединять модели: 'auto', 'predict_proba',

'decision_function', 'predict',

passthrough - использовать ли для обучения

финальной модели сами данные или только результаты моделей ансамбля

estimators_ - список обученных моделей в ансамбле

final_estimator_ - обученная финальная модель

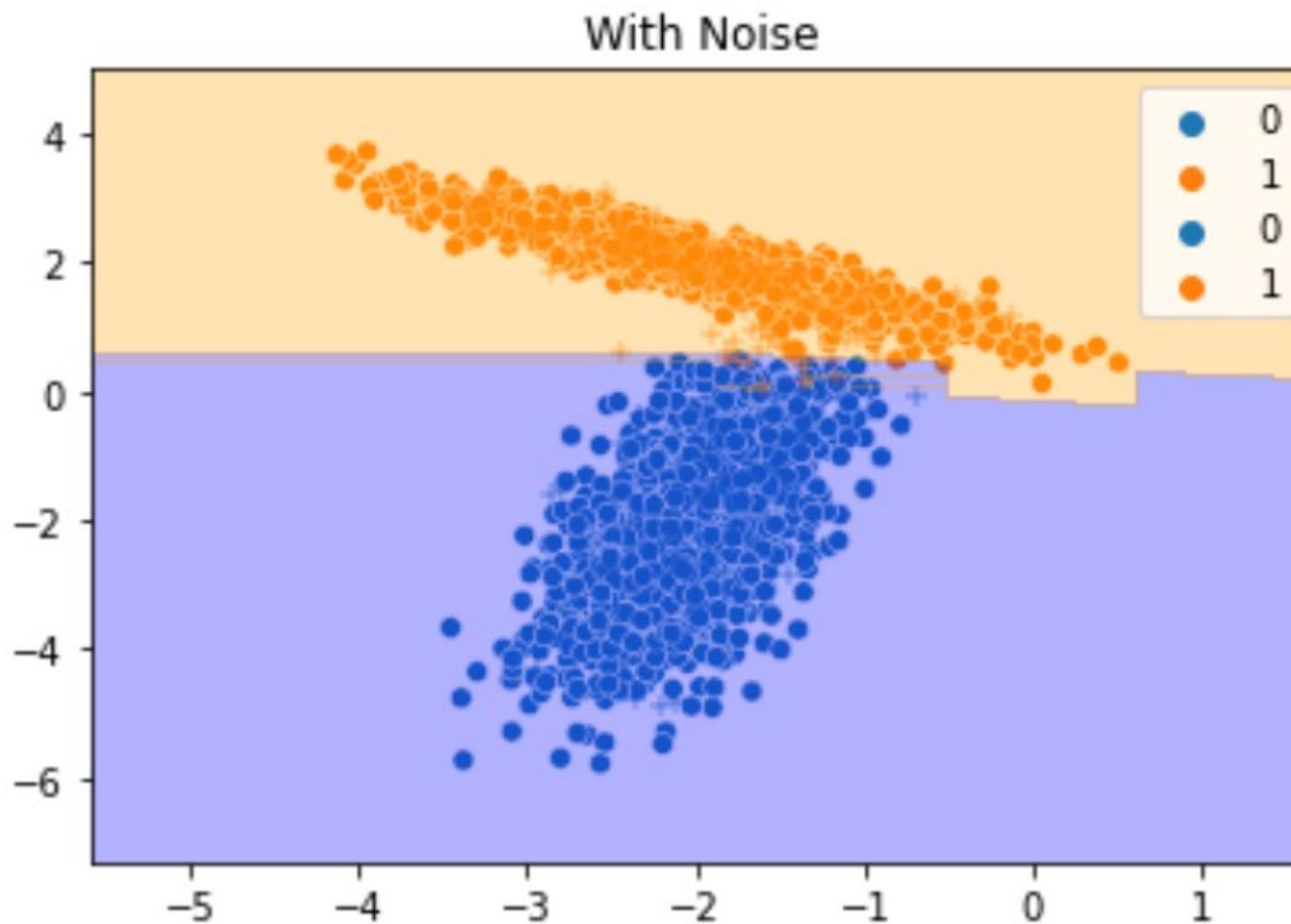
named_estimators_ - контейнер для доступа к параметрам моделей по их названию

classes_ - для классификатора метки классов



Ансамбли моделей. Стекинг

26





Ансамбли моделей. Бустинг

27

Бэггинг.

Модели строятся и работают
вместе, параллельно



Бустинг.

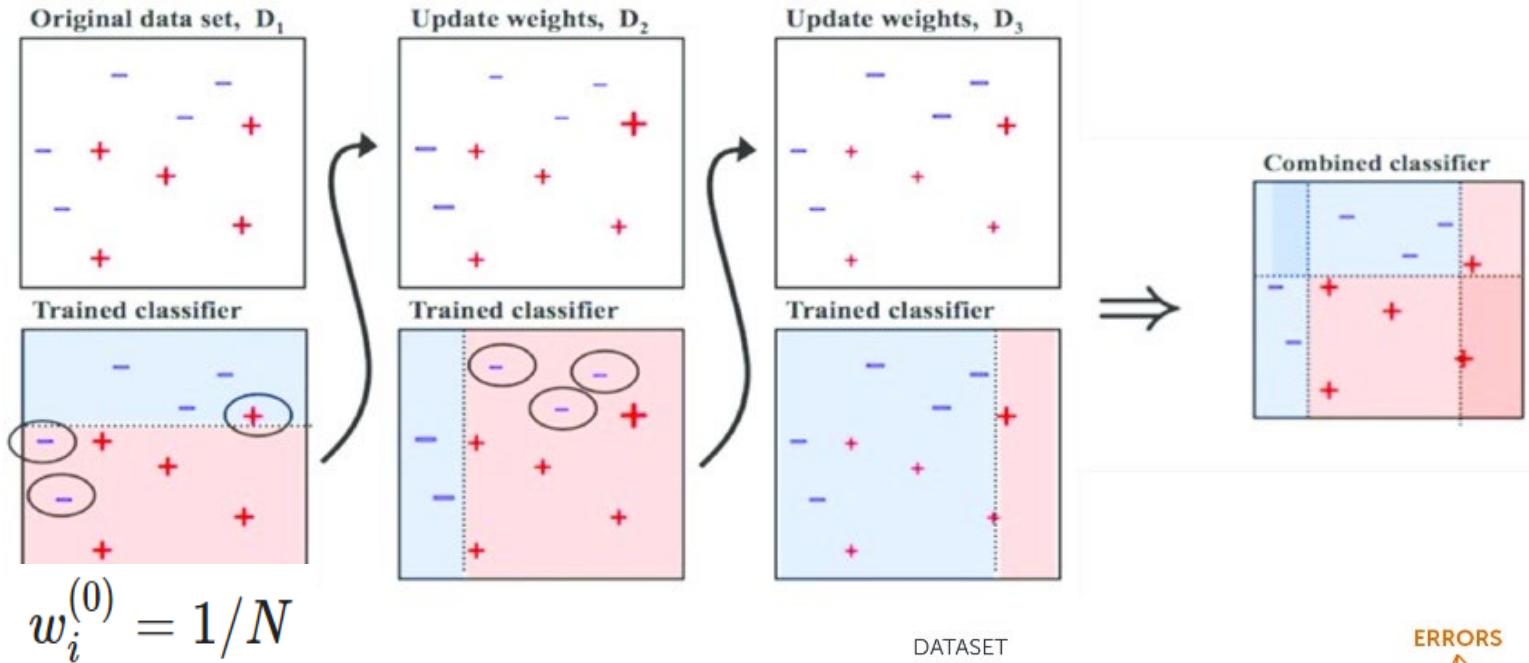
Модели строятся
последовательно,
компенсируя ошибки
предыдущих





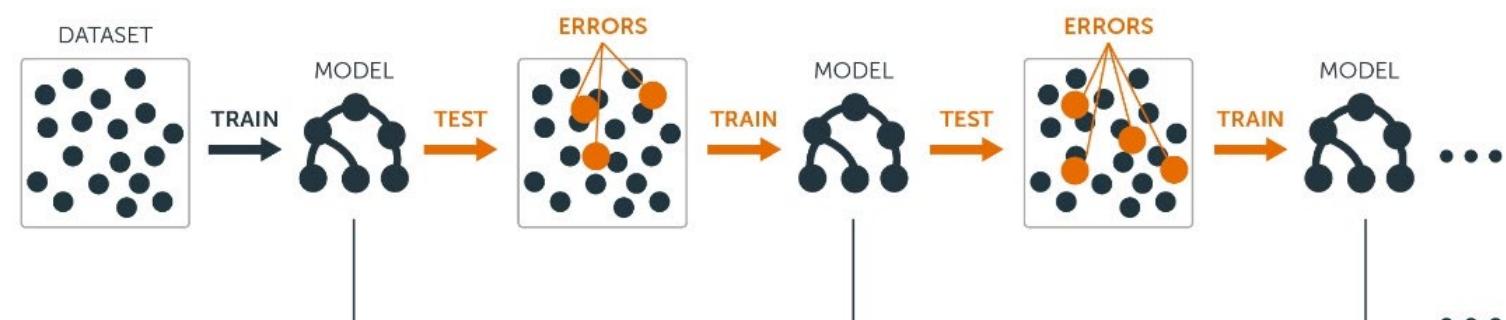
Ансамбли моделей. Бустинг. Adaboost

28



$$E = \sum_i w_i * e_i,$$

$$\sum_i w_i = 1, w_i \geq 0$$





Ансамбли моделей. Бустинг. AdaBoost

29

`sklearn.ensemble.AdaBoostClassifier`

```
class sklearn.ensemble.AdaBoostClassifier(base_estimator=None, *, n_estimators=50, learning_rate=1.0,  
algorithm='SAMME.R', random_state=None)
```

[\[source\]](#)

base_estimator — базовый алгоритм модели в ансамбле. По умолчанию используется дерево
n_estimators — максимальное количество моделей в ансамбле, после которого бустинг прекращается. Если ансамбль полностью обучится раньше, то моделей будет меньше.

learning_rate — ограничивает вклад каждой модели в изменение весов, по умолчанию равно 1. Снижение этого параметра будет означать, что весовые коэффициенты будут изменяться в меньшей степени, вынуждая модель дольше обучаться (но иногда повышается качество обучения).

base_estimator_ - тип моделей в ансамбле.

estimators_ - список обученных моделей ансамбля.

estimator_weights_ - вклад (вес) каждой модели в результат ансамбля

estimator_errors_ - ошибка каждой модели в ансамбле

feature_importances_ - важность признаков (если есть у базовой модели)

staged_decision_function(self, X) - для классификатора вычисляет уровни принадлежности к классу для каждой модели ансамбля по мере их создания, удобно для наблюдения за процессом обучения

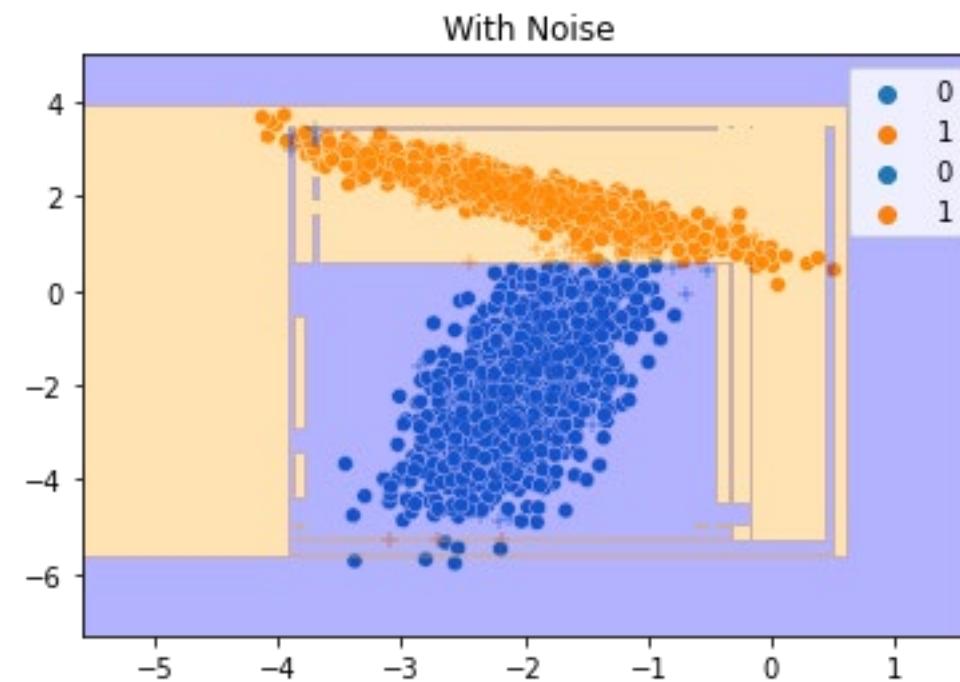
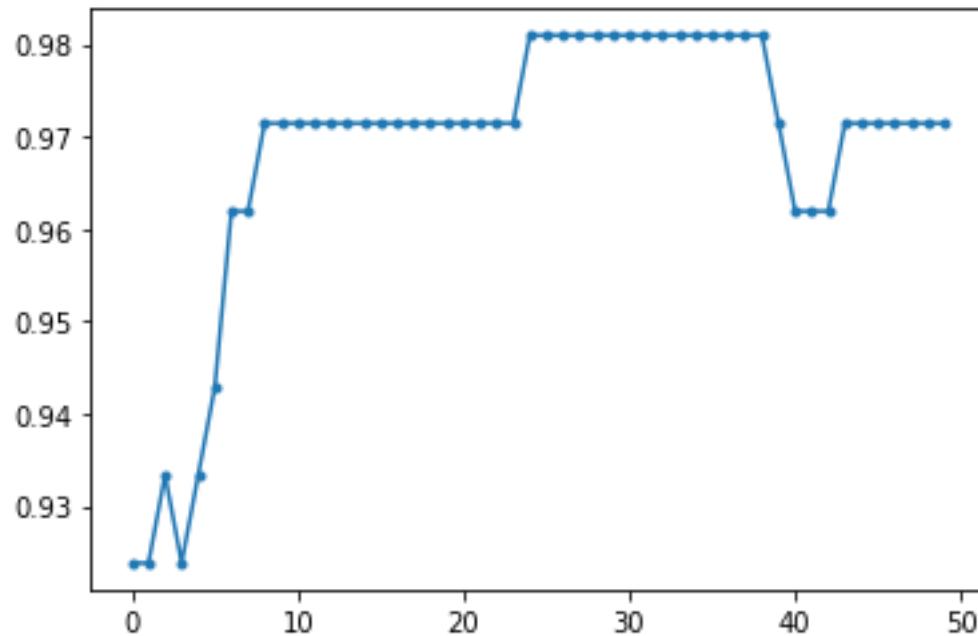
staged_predict(self, X) - вычисляет выходы для каждой модели ансамбля

staged_score(self, X, y[, sample_weight]) - вычисляет ошибки для каждой модели ансамбля.



Ансамбли моделей. Бустинг. Adaboost

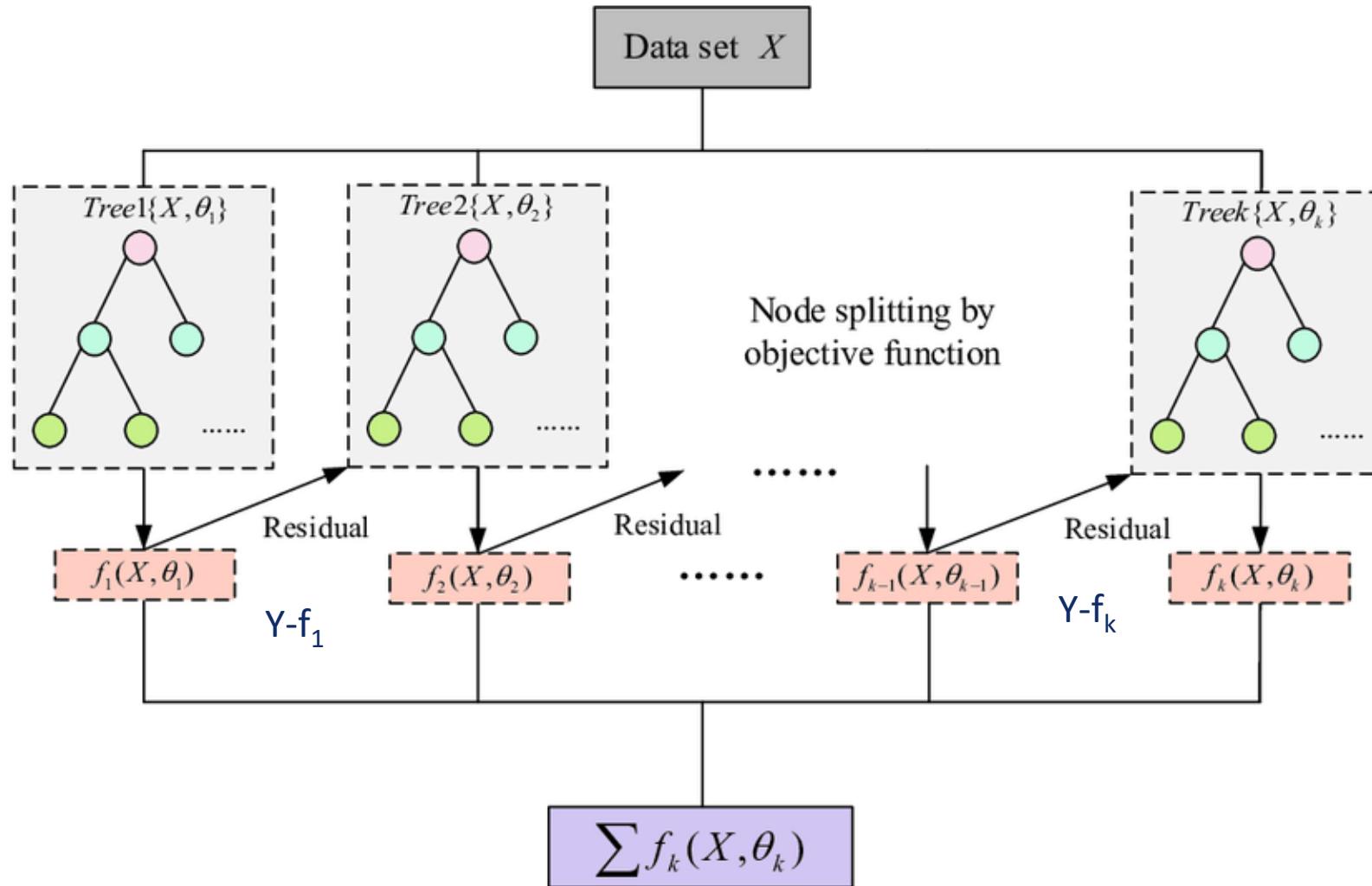
30





Ансамбли. Градиентный бустинг.

31





Ансамбли. Xgboost и Catboost

32

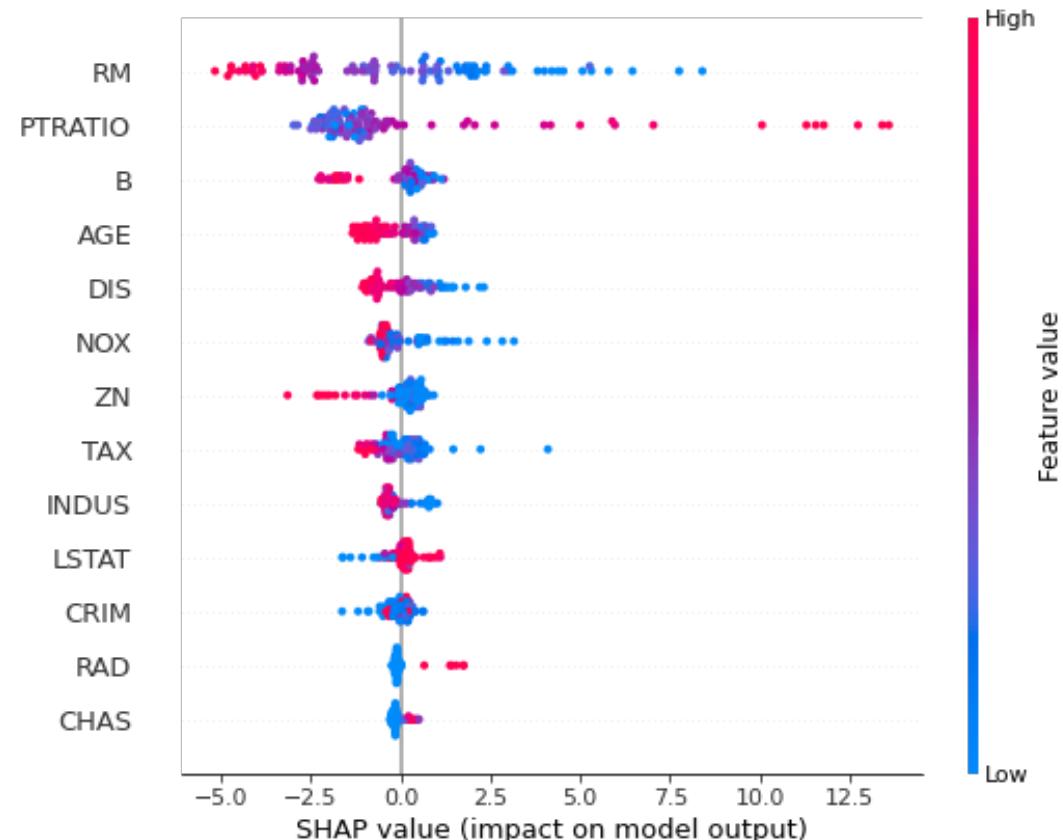
```
class xgboost.XGBClassifier(*, objective='binary:logistic', use_label_encoder=None, **kwargs) 🔗
```

https://xgboost.readthedocs.io/en/latest/python/python_api.html?highlight=xgbclassifier#xgboost.XGBClassifier

CatBoostClassifier

```
class CatBoostClassifier(iterations=None,  
                        learning_rate=None,  
                        depth=None,  
                        l2_leaf_reg=None,  
                        model_size_reg=None,  
                        rsm=None,  
                        loss_function=None,
```

https://catboost.ai/en/docs/concepts/python-reference_catboostclassifier





Группа по дисциплине:

<https://t.me/+8dShF1tFSDg0ZmJi>

