# Лекция 4

Деревья: от пня до градиентного бустинга

Аминов Тимур. email: aminov.tv@phystech.edu , telegram: @Sifonsoul

# План

1. Trees
2. Bias-variance tradeoff
3. Ensembling
    a. Forest
    b. AdaBoost
    c. Gradient Boosting
4. Stacking

# Полезные ссылки

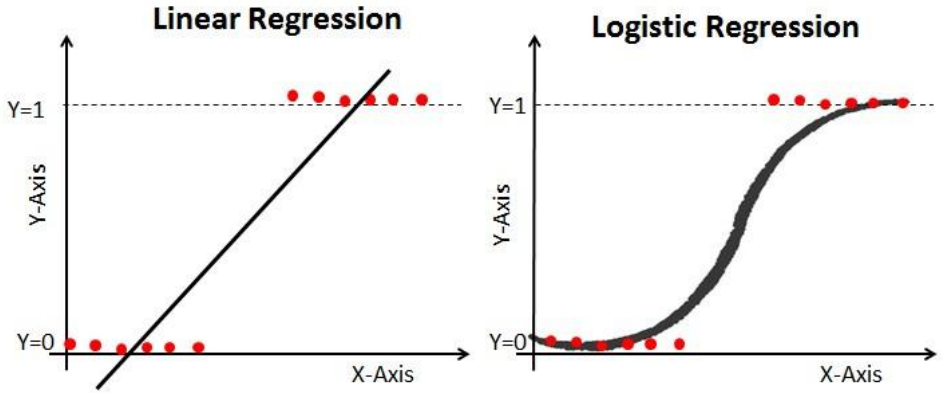
- Про деревья https://towardsdatascience.com/decision-trees-d07e0f420175
- Обзор по алгоритмам бустинга
  https://medium.com/diogo-menezes-borges/boosting-with-adaboost-and-gradient-boosting-9cbab2a1af81
- Подробный разбор параметров LGBM и сравнение с XGBoost
  https://www.analyticsvidhya.com/blog/2017/06/which-algorithm-takes-the-crown-light-gbm-vs-xgboost/

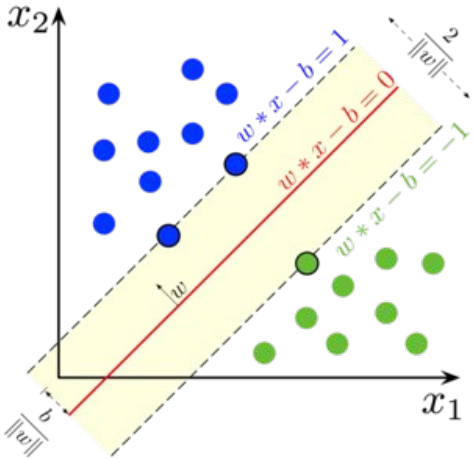
Ссылки внизу слайдов тоже полезные, но более углубленные по теме слайда.

# Повторение

# Линейные модели

- Линейная регрессия
- Логистическая регрессия
- SVM



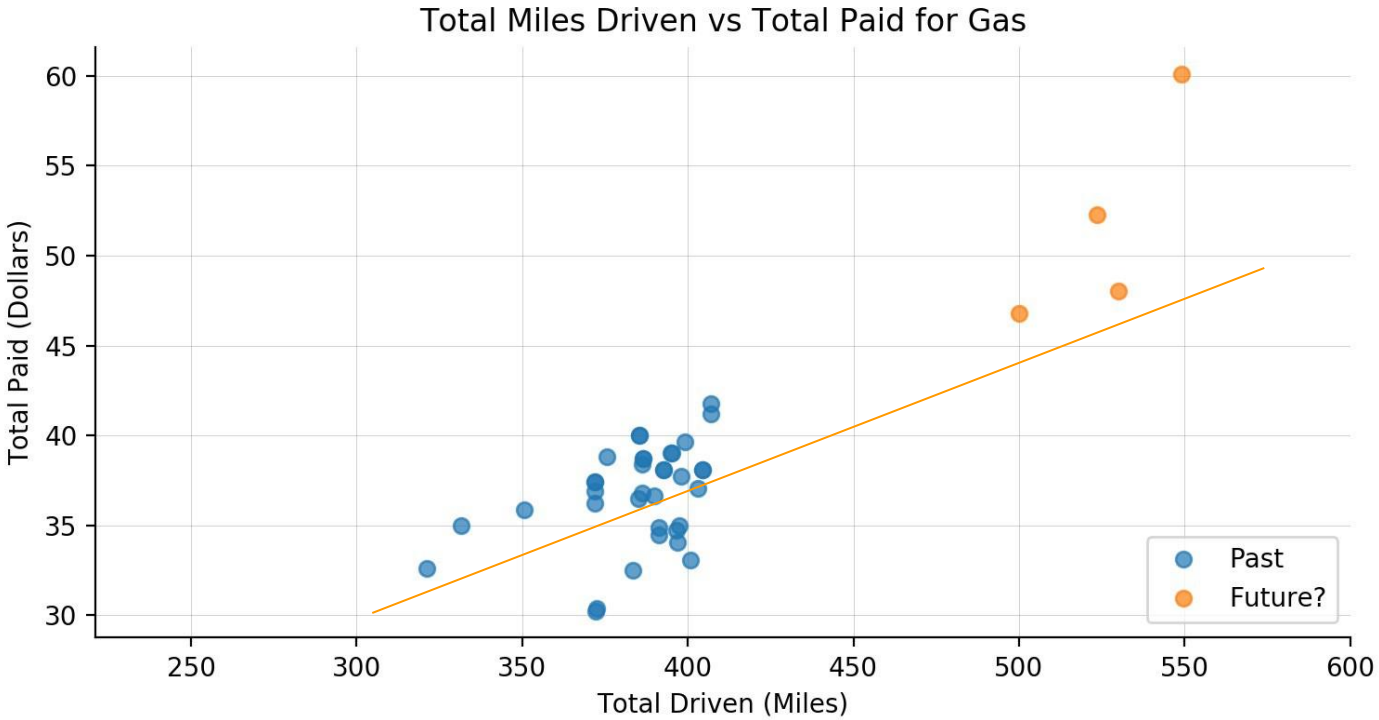



$$\hat{y}_i = wx_i + b$$

$$\mathcal{L} = \sum_i (wx_i + b - y_i)^2$$

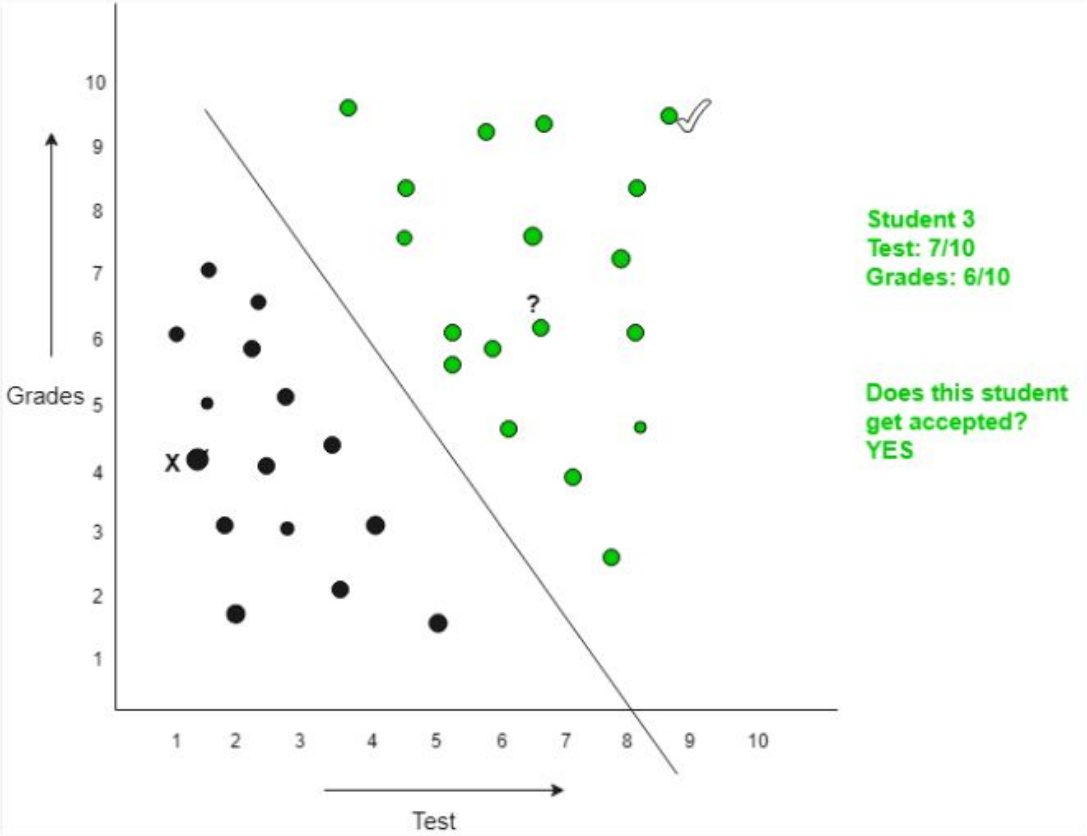
# Пример линейной регрессии

| | A | B |
|---|---|---|
| 1 | Total Payed | Total Miles |
| 2 | 36.66 | 390 |
| 3 | 37.05 | 403 |
| 4 | 34.71 | 396.5 |
| 5 | 32.5 | 383.5 |
| 6 | 32.63 | 321.1 |
| 7 | 34.45 | 391.3 |
| 8 | 36.79 | 386.1 |
| 9 | 37.44 | 371.8 |
| 10 | 38.09 | 404.3 |
| 11 | 38.09 | 392.6 |
| 12 | 38.74 | 386.49 |
| 13 | 39 | 395.2 |
| 14 | 40 | 385.50 |
| 15 | 36.21 | 372 |
| 16 | 34.05 | 397 |
| 17 | 41.79 | 407 |
| 18 | 30.25 | 372.33 |
| 19 | 38.83 | 375.6 |
| 20 | 39.66 | 399 |



# Пример SVM

```
Student 1 : Test Score: 9/10, Grades: 8/10  Result: Accepted
Student 2 : Test Score: 3/10, Grades: 4/10, Result: Rejected
Student 3 : Test Score: 7/10, Grades: 6/10, Result: to be tested
```

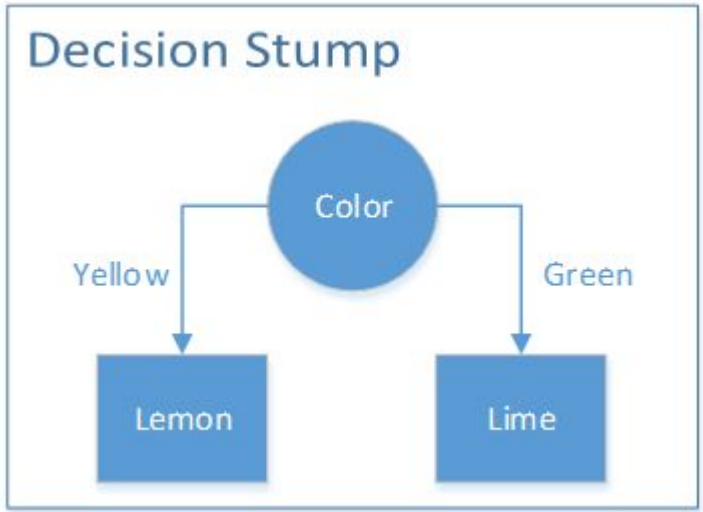


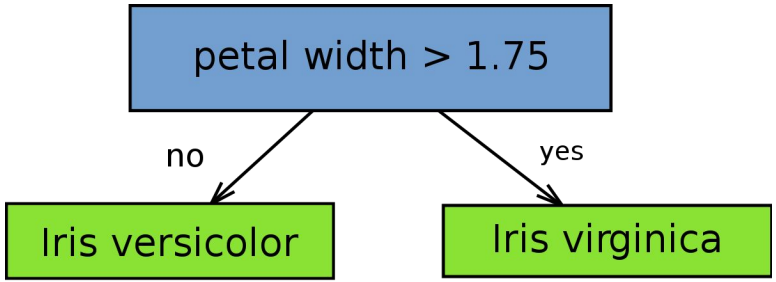
# Деревья

# Логические условия

Логические операторы делят датасет на несколько частей

- Пороговое условие
  - применяется к числовым признакам
  - делит датасет на две части
- Деление по категориям
  - делит датасет используя один категориальный признак



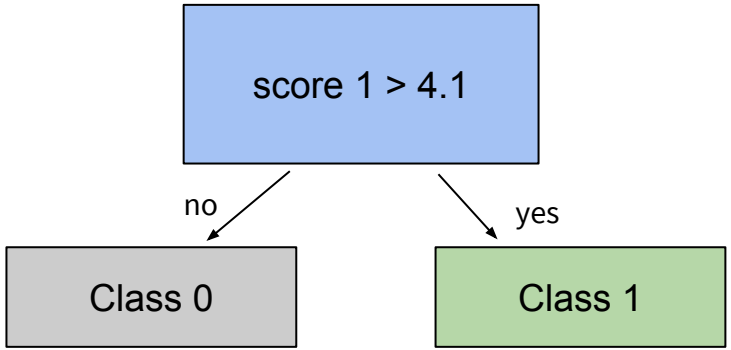



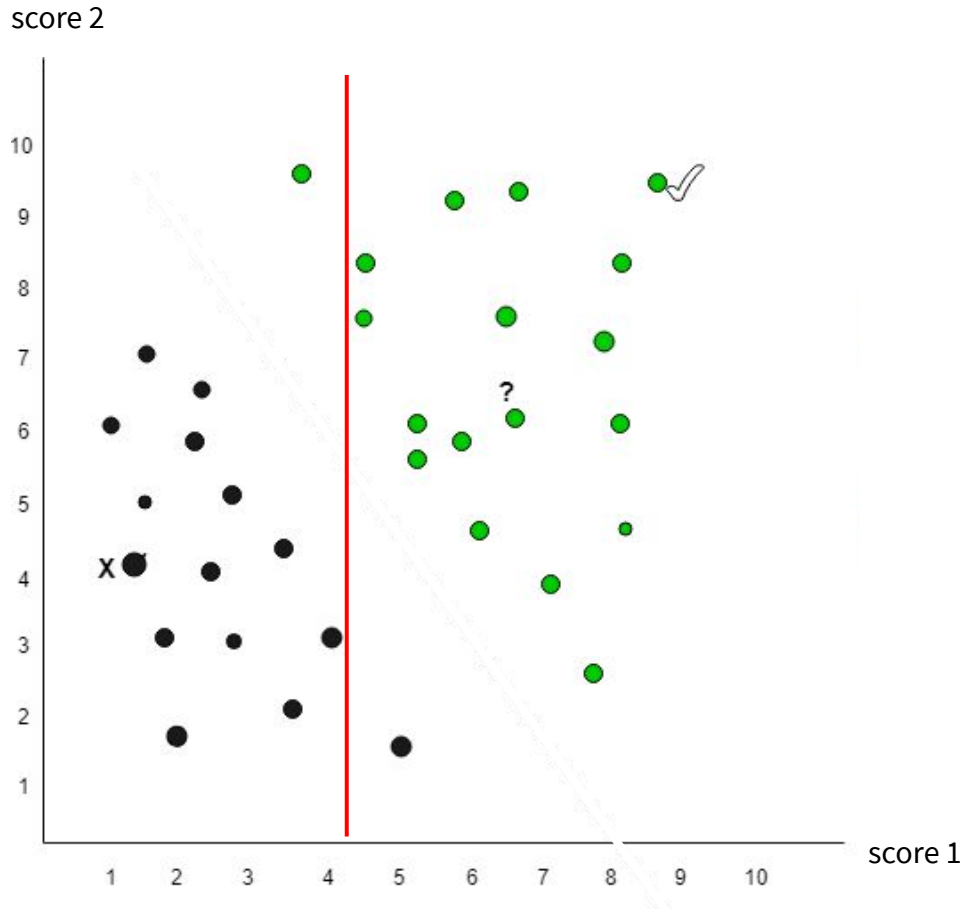
# Decision stump

Модель, которая использует одно правило называется решающий пень

Пороговое правило не может разделить датасет по наклонной линии

Пень - это дерево с *глубиной* 1



# Критерии качества

Чтобы выбрать наилучшее логическое условие для разделения тренировочной на несколько подвыборок, используют следующие критерии:

- Для классификации:
  - Энтропия
  - Критерий Джини
- Для регрессии:
  - Стандартное отклонение

$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

$$G.I\ (Y) = 1 - \sum_{i=1}^{k} [p(Y)]^2$$

Все эти критерии говорят о том, насколько **уменьшилась неопределенность** при разделении тренировочного набора на подвыборки с помощью данного правила

$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

# Decision tree

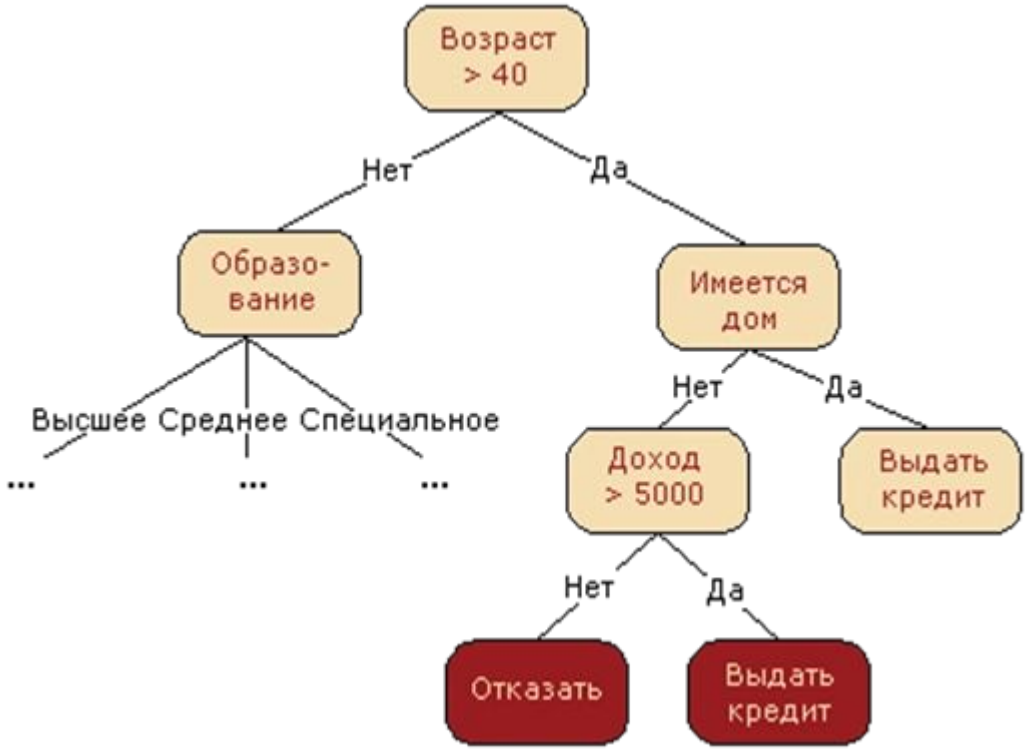
В каждом узле выбрано из всех возможных правил то, которое делит датасет с максимальным уменьшением

a) энтропии
b) критерия Джини
c) дисперсии

Каждая подвыборка снова делится.

Деление заканчивается когда:

a) В подвыборке остался один класс
b) Достигнута максимальная глубина решающего дерева



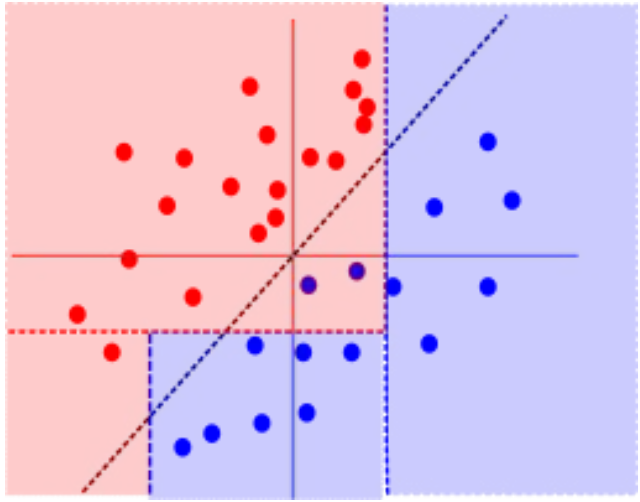
# Решающее дерево

Минусы:

- Не всегда находит лучшее разбиение
- Легко переобучается

Плюсы:

- Интерпретируемо
- Может подстроиться под задачу любой сложности
- Быстрое обучение



# Bias&variance tradeoff

# Bias&variance tradeoff

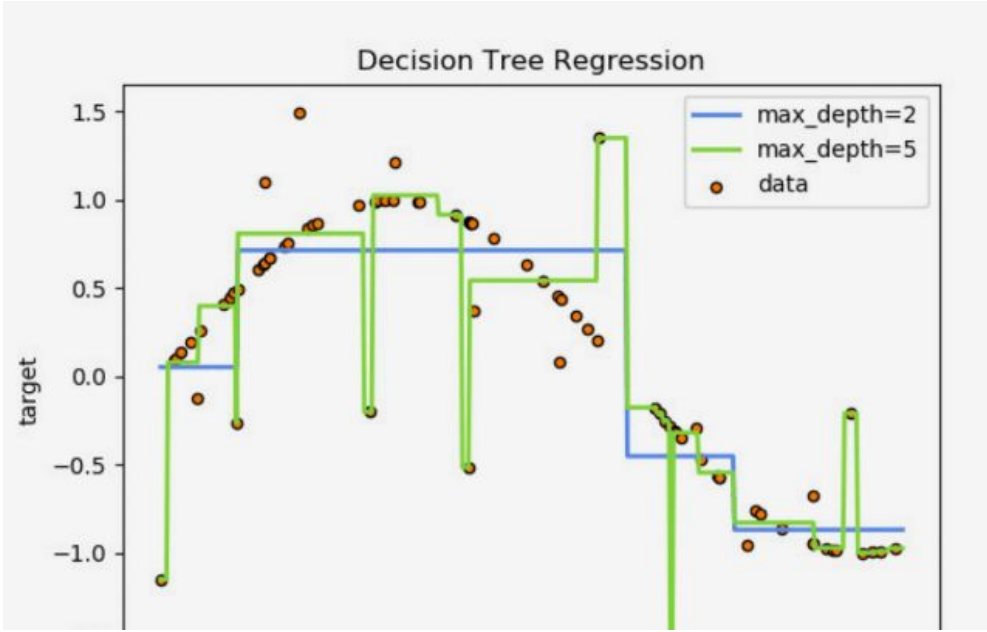
Существует идея раскладывать ошибку в теории на смещение - bias, и дисперсию - variance.

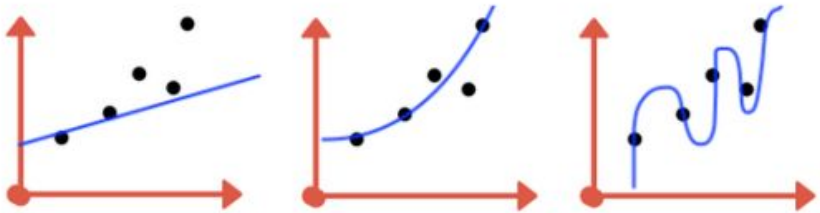
Смещение - среднее смещение предсказаний от реальной закономерности

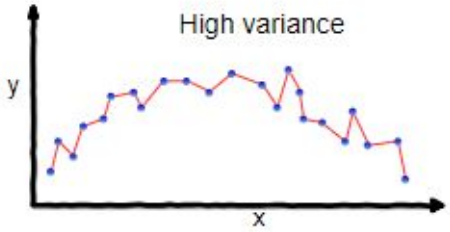
Дисперсия - средняя дисперсия предсказания в каждой точке при обучении на случайной тренировочной выборке

**error = bias^2 + variance + noise**

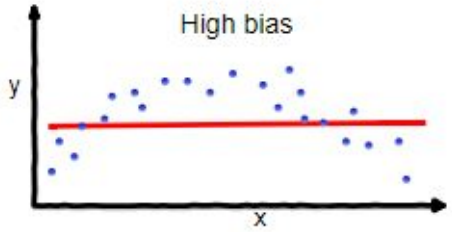
- У слишком простой модели высокий bias - **недообучение**
- У слишком сложной модели высокое variance - **переобучение**



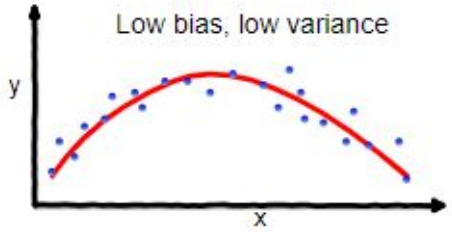






**overfitting**




**underfitting**




**Good balance**

**error = bias^2 + variance + noise**

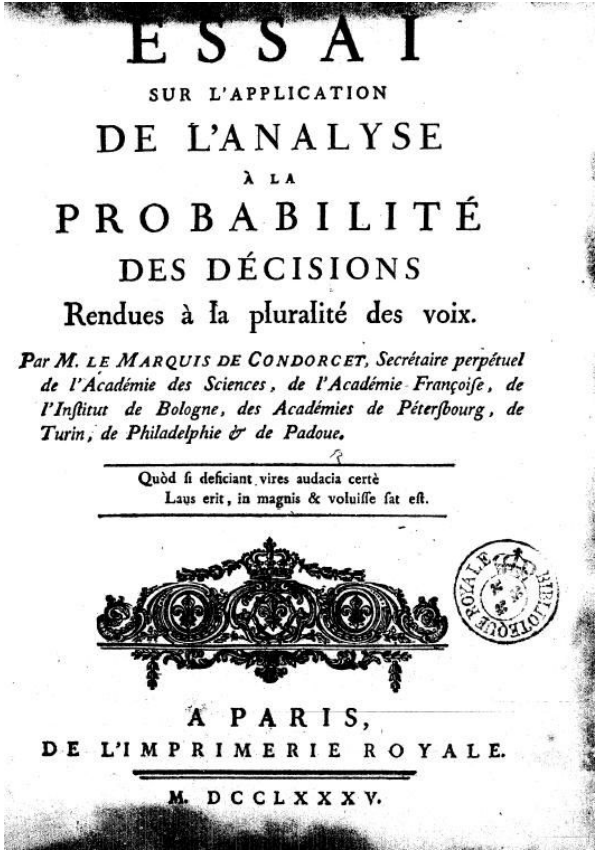
# Композиции алгоритмов

# Bagging

# Голосование

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.



ESSAI

SUR L'APPLICATION

DE L'ANALYSE

À LA

PROBABILITÉ

DES DÉCISIONS

Rendues à la pluralité des voix.

*Par M. LE MARQUIS DE CONDORCET, Secrétaire perpétuel de l'Académie des Sciences, de l'Académie Françoise, de l'Inſtitut de Bologne, des Académies de Péterſbourg, de Turin, de Philadelphie & de Padoue.*

Quòd ſi deficiant vires audacia certè
Laus erit, in magnis & voluiſſe ſat eſt.



A PARIS,

DE L'IMPRIMERIE ROYALE.

M. DCCLXXXV.

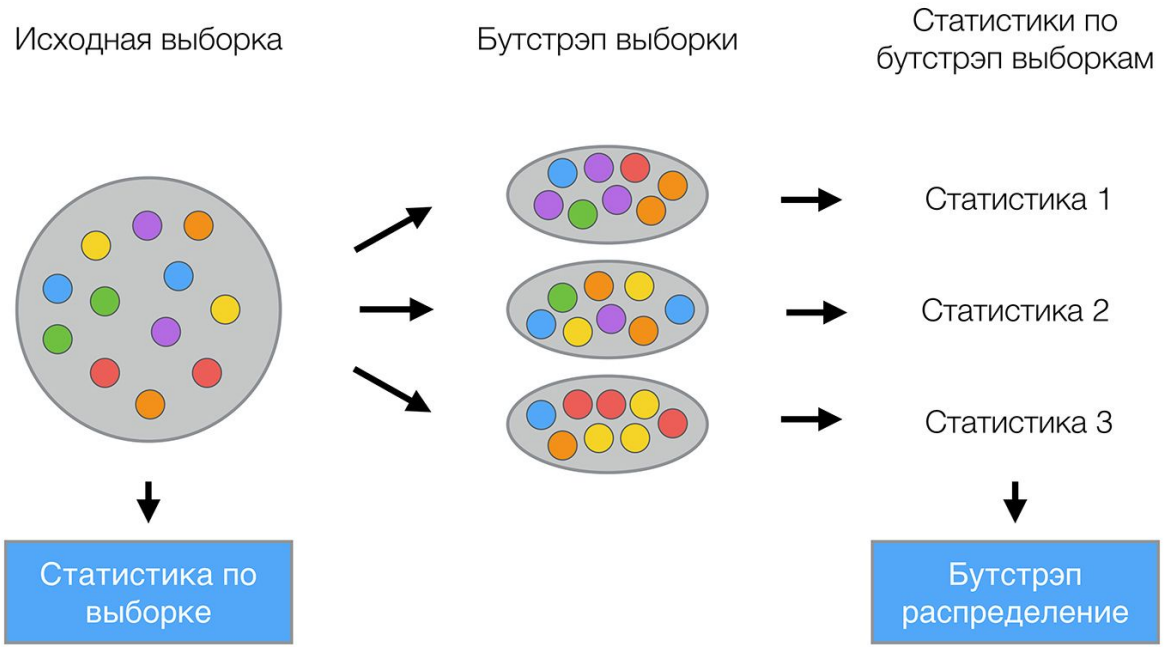
принцип Кондорсе, 1784

# Усреднение

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




https://habr.com/ru/company/ods/blog/324402/

1 2 3 — $p$ — вероятность не верной классификации

$y^n$ — голосование — 1, 2, 3 — независимы

$A$

$P\{\text{ошибки коллеги}\} = P\{\text{все ошибись}\} + P\{\text{ошибось двое}\} =$

$= p^3 + (1-p)\,p^2 \cdot C_3^2 = C_3^{3-2} = C_3^1 = 3$

$= p^3 + 3p^2 - 3p^3 = 3p^2 - 2p^3$

$P\{A\} < p$

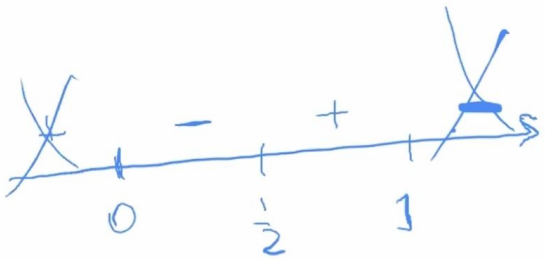
$p^2 < p$

$3p^2 - 2p^3 < p$

$p(3p - 2p^2 - 1) < 0$

$p(p-1)(-2p+1) < 0$

| | -2 | 3 | -1 |
|---|---|---|---|
| 1 | -2 | 1 | |

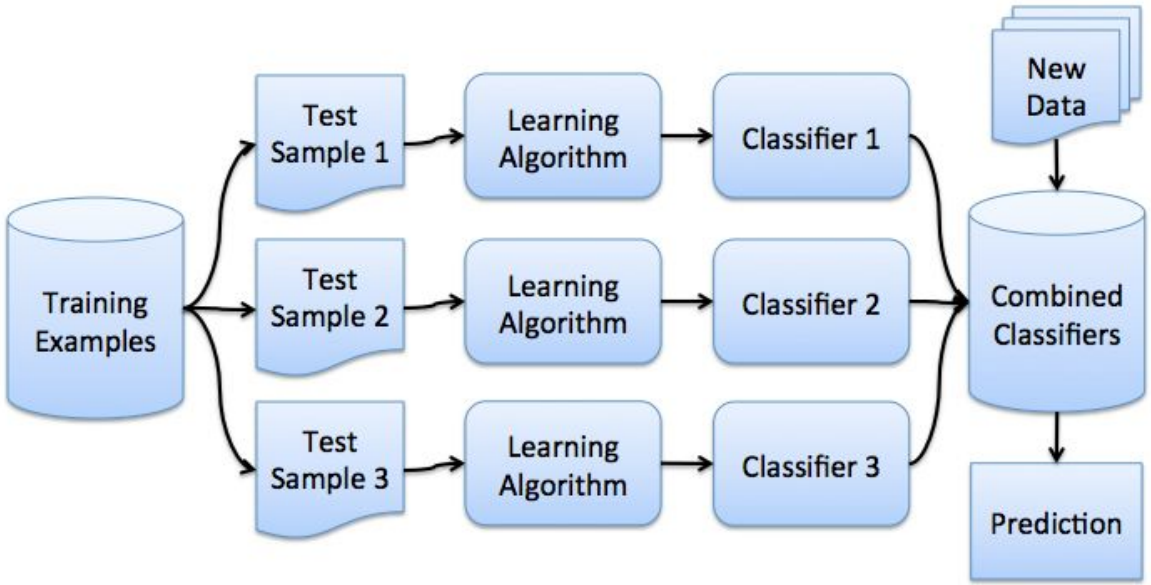


# Bagging

С помощью bootstrap сгенерируем из тренировочной выборки M выборок.

На каждой выборке обучим свой классификатор.

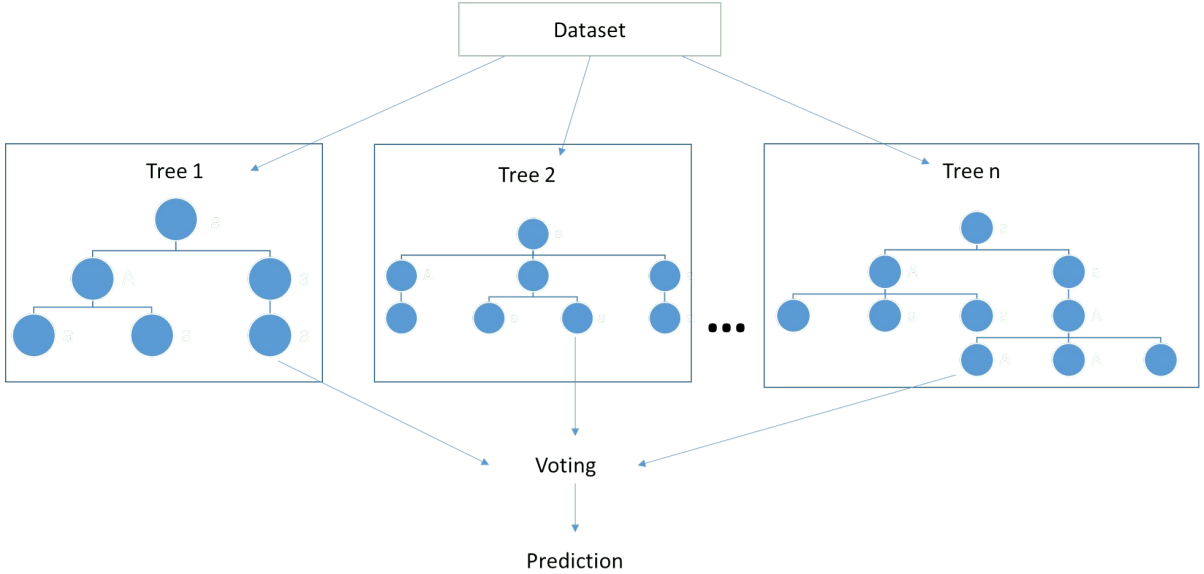
Итоговый классификатор будет усреднять ответы всех этих алгоритмов.



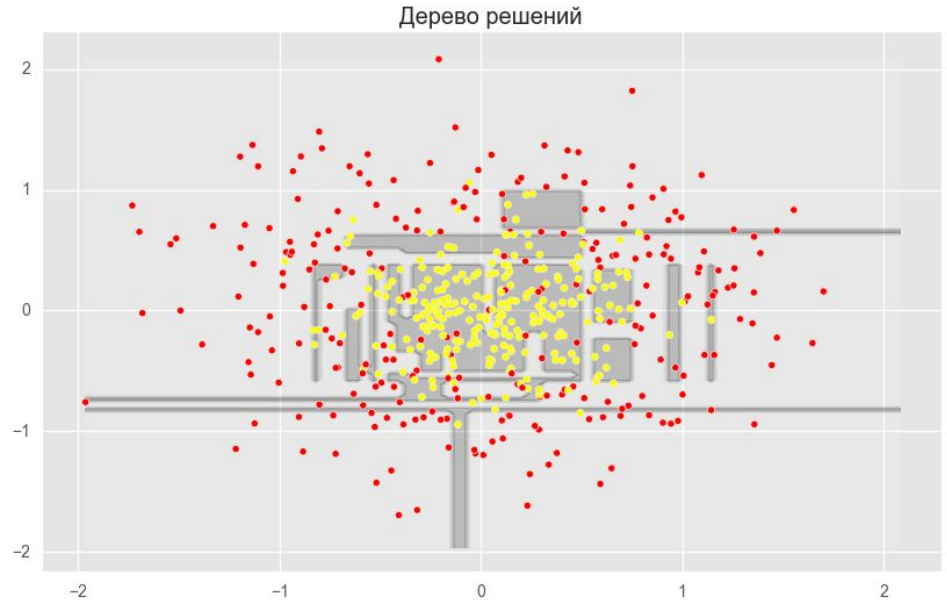

$$a(x) = \frac{1}{M} \sum_{i=1}^{M} a_i(x)$$

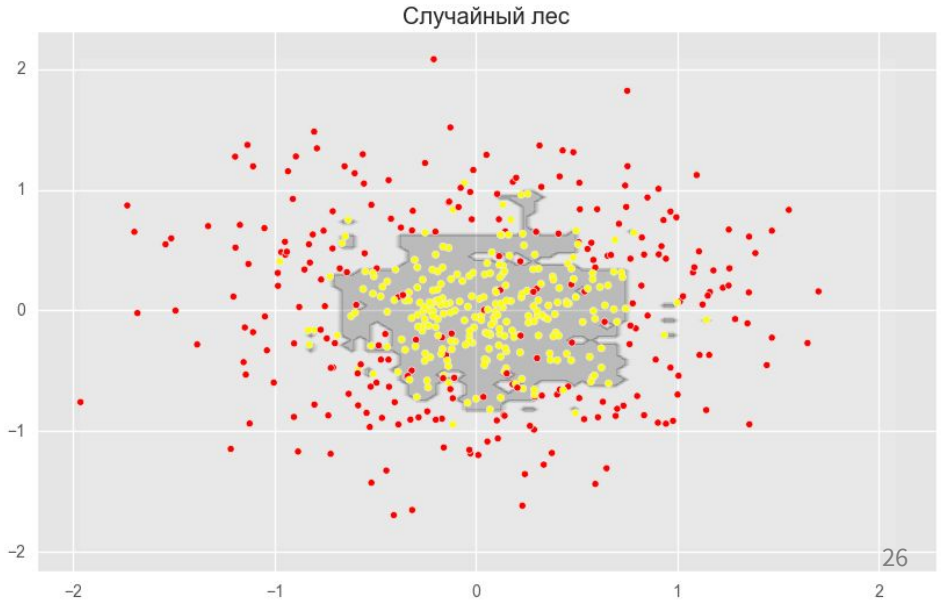
# Случайный лес

- Лес - это ансамбль деревьев, образованный с помощью bootstrap выборок.
- Также это называется bagging над решающими деревьями
- Уменьшает variance, не меняя bias

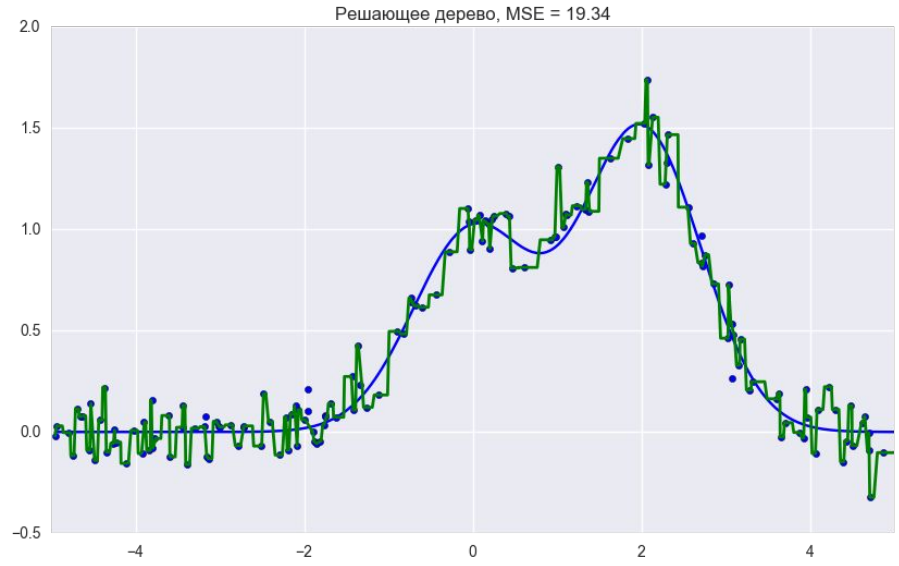


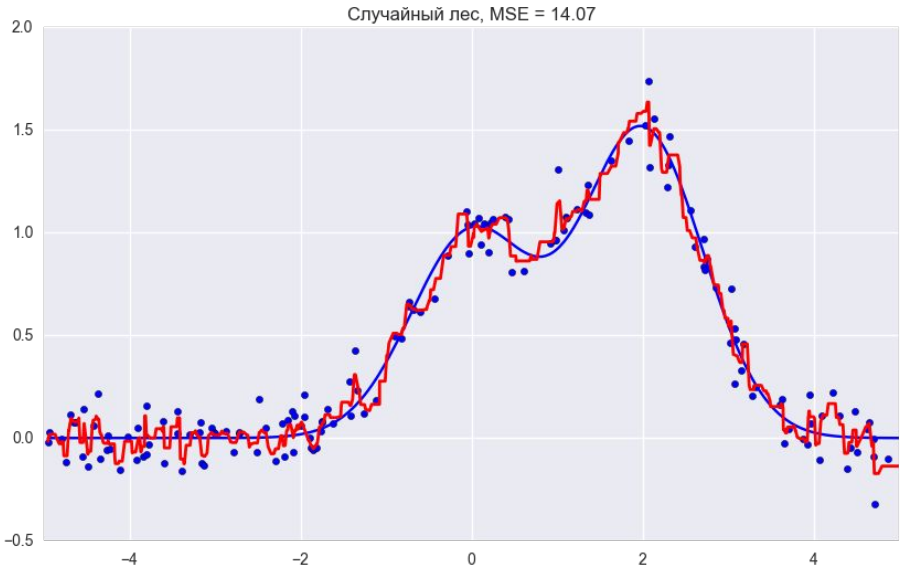
# Случайный лес



# Случайный лес



# Boosting

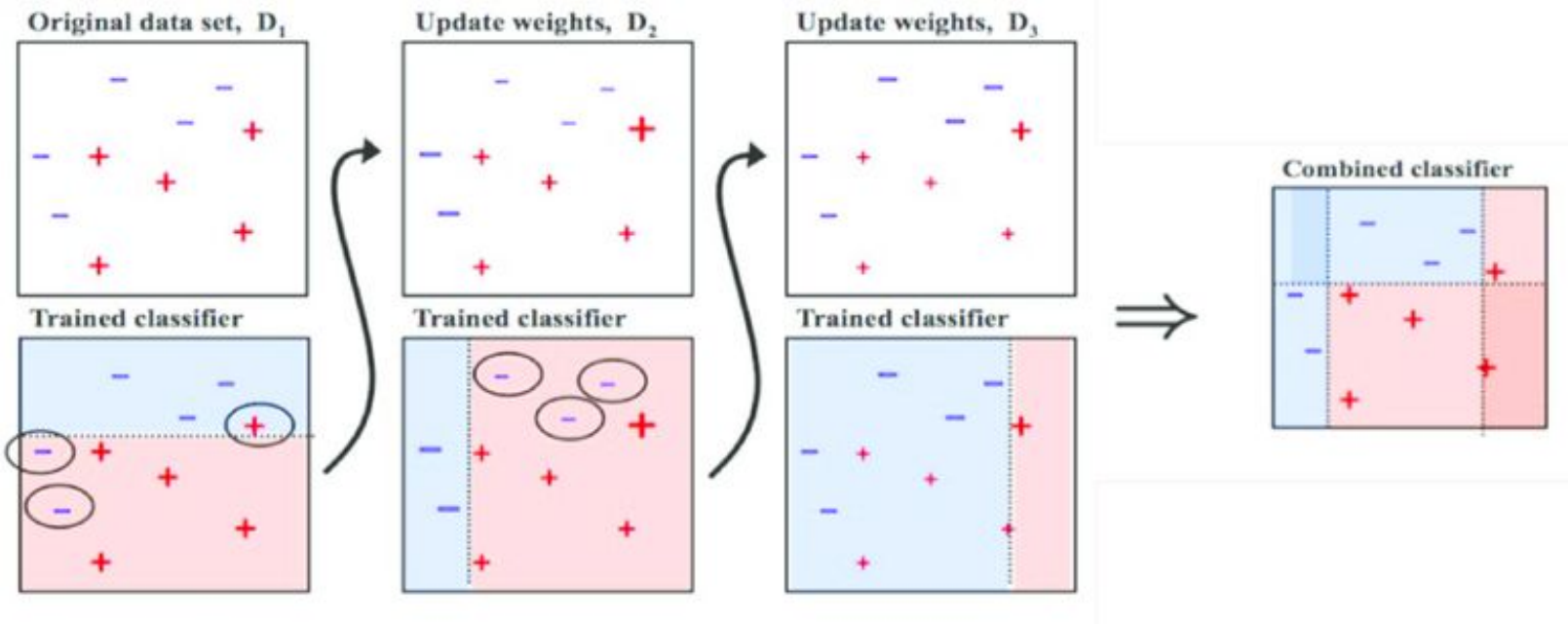
# Adaptive Boosting

1. Всем объектам задается одинаковый вес
2. На подвыборке обучается базовая модель (по-умолчанию - решающий пень)
3. Делаются предсказания и измеряется качество на всей выборке
4. Веса изменяются в соответствие с ошибкой: чем больше ошибка - тем больше вес данного примера
5. Следующая модель обучается на взвешенном датасете: больше внимания тем примерам, у которых больше вес
6. Процесс повторяется до тех пор, пока ошибка не перестанет уменьшаться или пока не будет достигнуто нужное кол-во итераций
7. Предсказания на тестовой выборке делаются по принципу взвешенного голосования

# Adaptive Boosting

“В конце обучения каждой модели мы увеличиваем (boost) веса тех примеров, которые были неверно классифицированы”

# Adaptive Boosting



# Adaptive Boosting

Плюсы:

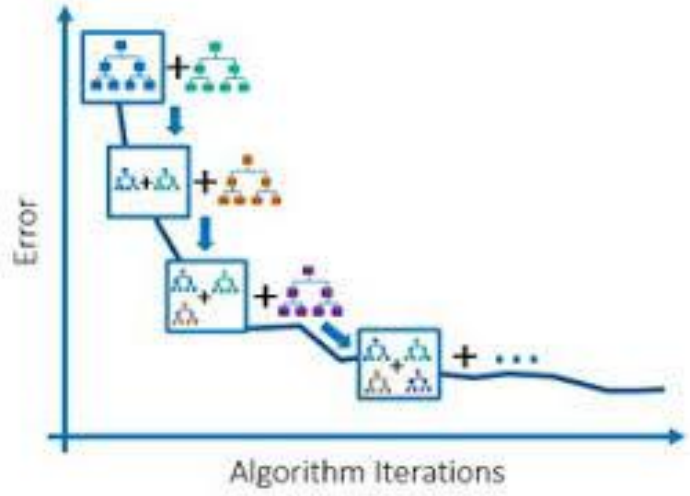
- Уменьшаются как bias, так и variance
- Простой базовый классификатор

Минусы:

- Обучение не распараллеливается

# Gradient Boosting

1. Базовая модель (по-умолчанию - дерево глубиной 3) обучается на подвыборке тренировочного набора
2. Делаются предсказания на всей обучающей выборке
3. Относительная ошибка этих предсказаний становится целевой переменной для следующей модели
4. Предсказания новой модели складываются с предсказаниями предыдущей
5. Вычисляется новая относительная ошибка и делается целевой переменной для следующей модели
6. Процесс повторяется до тех пор, пока ошибка не перестанет уменьшаться или пока не будет достигнуто нужное кол-во итераций



# Сравнение

AdaBoost:

- Абсолютное значение ошибки
- Используется как вес
- Функция loss такая же, как у базовой модели

AdaBoost является частным случаем GB

Gradient Boosting:

- Относительное смещение
- Используется как целевая переменная
- Функция loss может быть произвольной

GB появился позже, как теоретическое обобщение AdaBoost

https://web.stanford.edu/~hastie/ElemStatLearn/printings/ESLII_print12.pdf - стр 347, 350

# Loss для регрессии

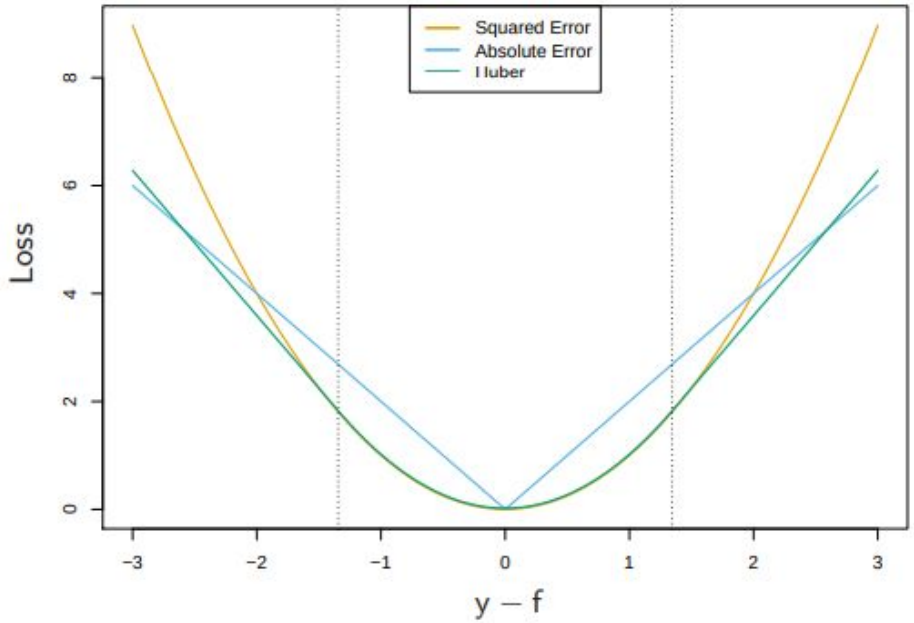



**FIGURE 10.5.** *A comparison of three loss functions for regression, plotted as a function of the margin $y-f$. The Huber loss function combines the good properties of squared-error loss near zero and absolute error loss when $|y - f|$ is large.*

# Loss для классификации

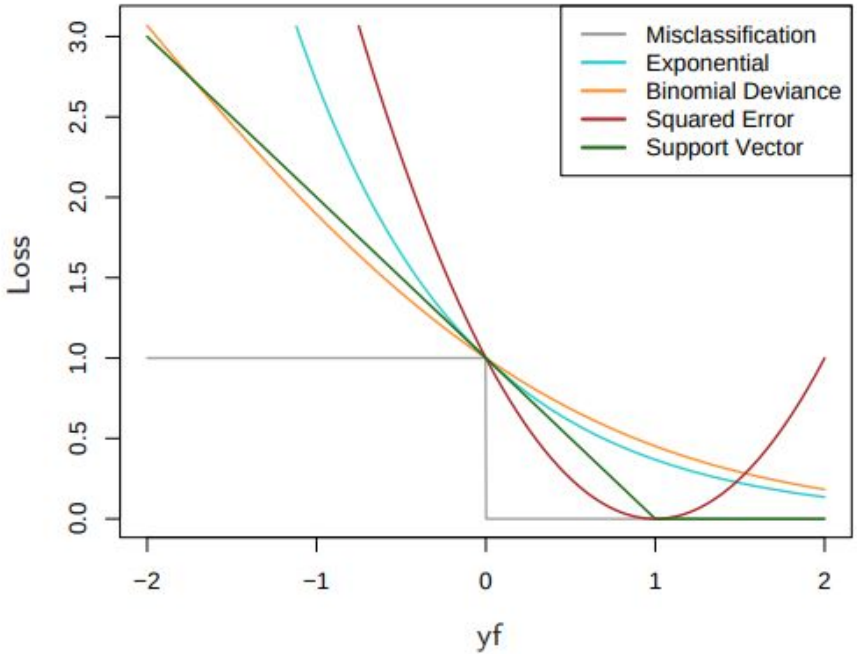



**FIGURE 10.4.** *Loss functions for two-class classification. The response is* $y = \pm 1$*; the prediction is* $f$*, with class prediction* $\text{sign}(f)$*. The losses are misclassification:* $I(\text{sign}(f) \neq y)$*; exponential:* $\exp(-yf)$*; binomial deviance:* $\log(1 + \exp(-2yf))$*; squared error:* $(y - f)^2$*; and support vector:* $(1 - yf)_+$ *(see Section 12.3). Each function has been scaled so that it passes through the point* $(0, 1)$*.*

# Градиентный бустинг

Метод градиентного бустинга считается **самым сильным** классическим алгоритмом обучения с учителем как для регрессии, так и для классификации

4 самых известных реализации:

- sklearn
- XGBoost
- LightGBM
- CatBoost

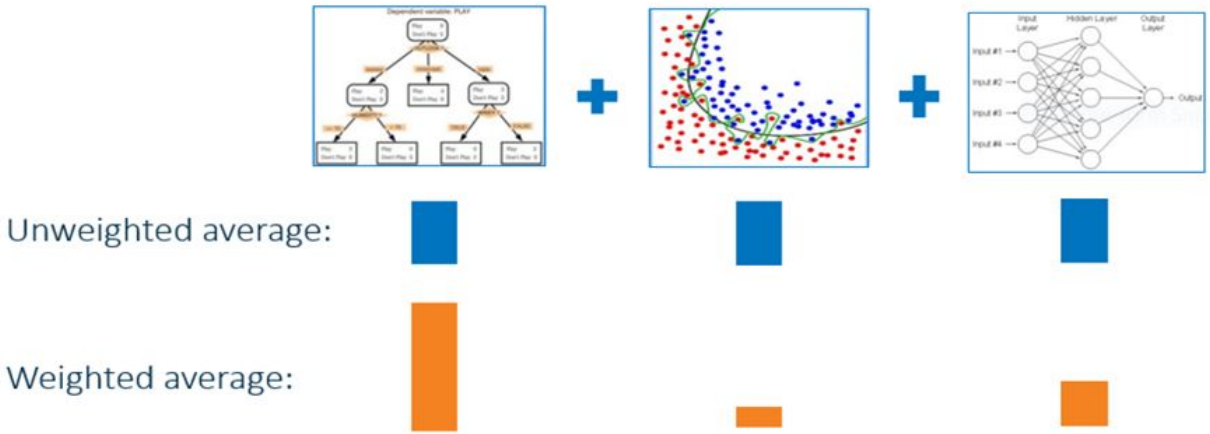
Эти библиотеки широко применяются в соревнованиях на kaggle и в промышленности

# Stacking

# Stacking

Другой способ объединить предсказания нескольких моделей.

В отличие от bagging, проводится взвешенное голосование, где веса разных моделей (base learners) подбираются новой моделью (meta learner).

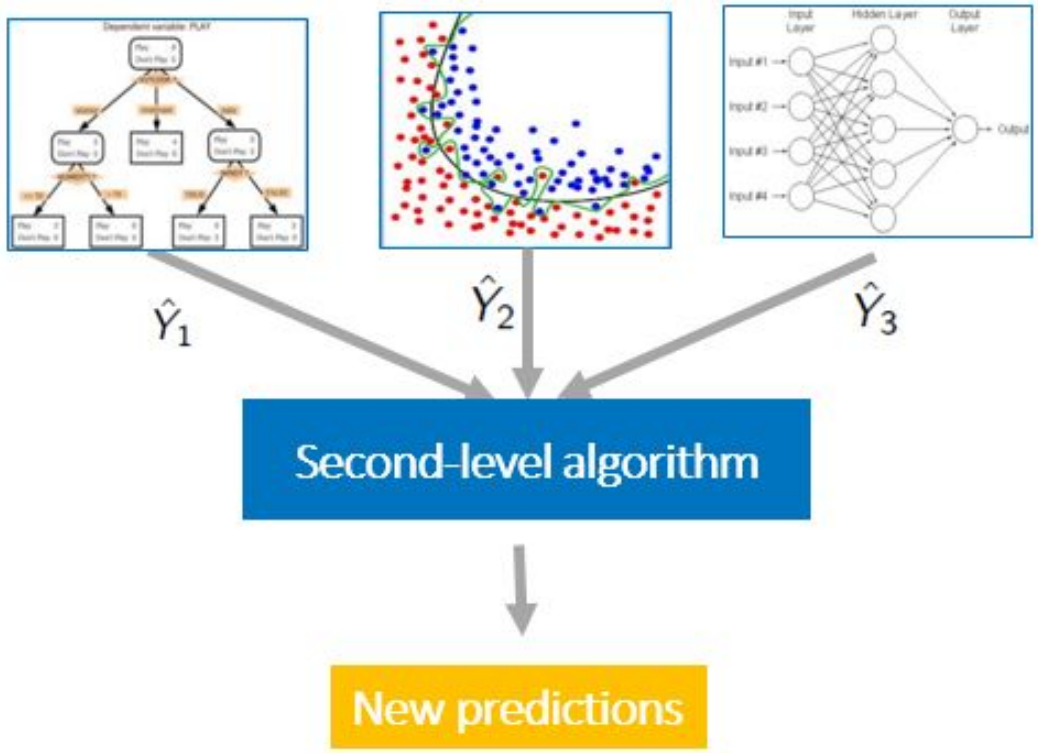


# Stacking

Входные данные для meta learner’a - это предсказания базовых моделей.

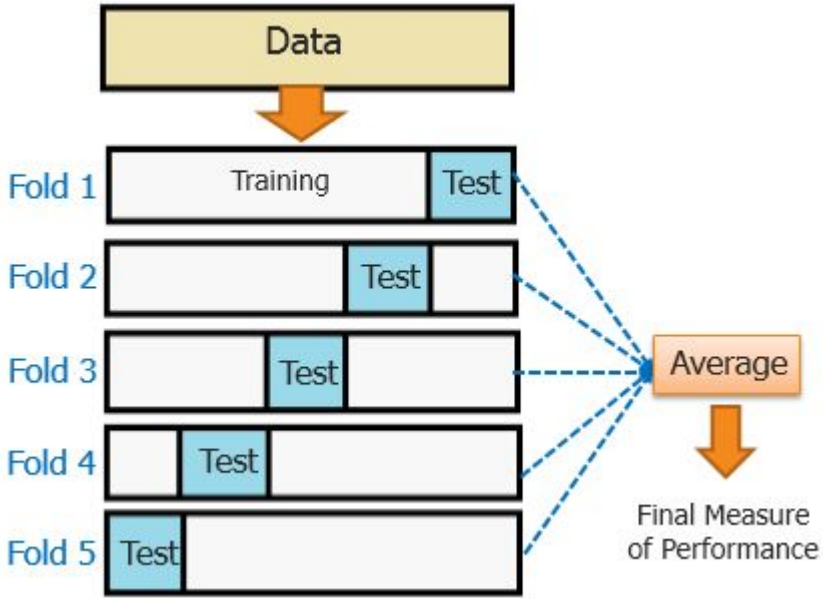
Целевая переменная остается та же.

Чтобы мета-модель правильно обучалась, нужны несмещенные предсказания.

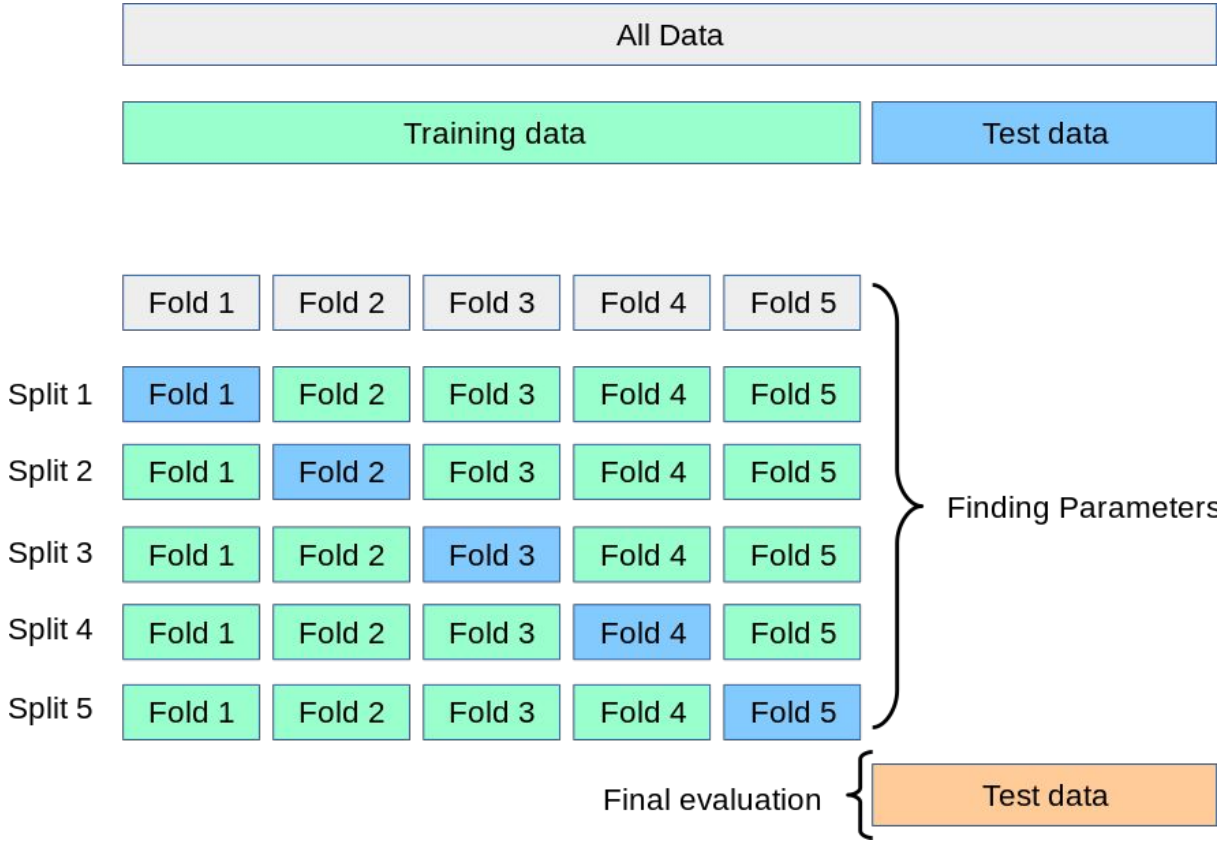


# Напоминание: K-fold cross-validation

- Получаем несмещенную оценку качества модели
- Используем весь тренировочный набор



# Out-of-fold prediction



# Out-of-fold prediction

Предсказания на тренировочной выборке делаются несколькими разными моделями (то есть, одна модель, но обученная на разных данных).

Предсказания на тестовой выборке делаются на каждом fold’е, а затем берется среднее.

# Стандартный модуль

```
from mlxtend.classifier import StackingCVClassifier
```

```
# Initializing models
clf1 = xgb_grid.best_estimator_
clf2 = gbc_grid.best_estimator_
clf3 = rf_grid.best_estimator_
clf4 = svc_grid.best_estimator_

lr = LogisticRegression()
st_clf = StackingCVClassifier(classifiers=[clf1, clf2, clf3, clf4], meta_classifier=lr)
```

https://www.kaggle.com/dejavu23/titanic-survival-seaborn-and-ensembles

# Много слоев


•Jeong-Yoon Lee, Winning Data Science Competitions

https://blogs.sas.com/content/subconsciousmusings/2017/05/18/stacked-ensemble-models-win-data-science-competitions/

# Обучение с учителем - summary

Модели:

- K Nearest Neighbors
- Naive Bayes Classifier
- Linear and logistic regression
- Support vector machines
- Trees
- Forests
- AdaBoost, Gradient Boost
- XGBoost, LightGBM, CatBoost

Все модели (кроме Naive Bayes) применяются как для классификации, так и для регрессии

Ансамбли:

- Bagging, boosting
- Stacking

# Краткая инструкция

Какую же модель все таки применить?

1) Попробовать линейную модель (linear regression или logistic regression)
2) Измерить качество. Выяснить: результаты предсказаний полезны хоть сколько-то вообще, или нет?
3) Попробовать SVM и случайный лес. Посмотреть, насколько улучшится качество. Если модель работает адекватно, то в 90% случаев **можно так и оставить**
4) Если хочется улучшить качество на пару процентов, попробуйте xgboost, lgbm, catboost. У какого получится выше точность, тот и использовать.
5)

Если каждые полпроцента на счету, то можно экспериментировать с ансамблями:

- Попробовать объединить все свои модели с помощью голосования
- Попробовать объединить все свои модели с помощью еще одной линейной модели

Сложные модели легко запрограммировать, но они требуют большого кол-ва экспериментов