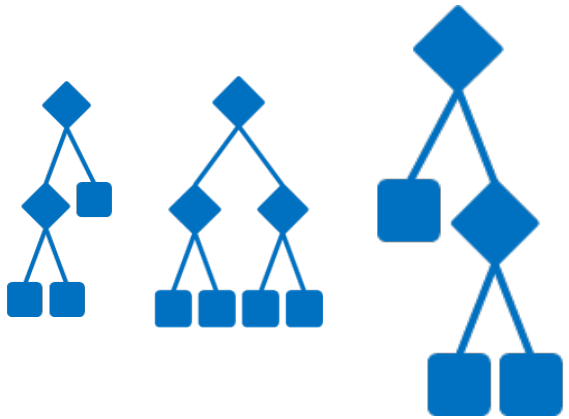
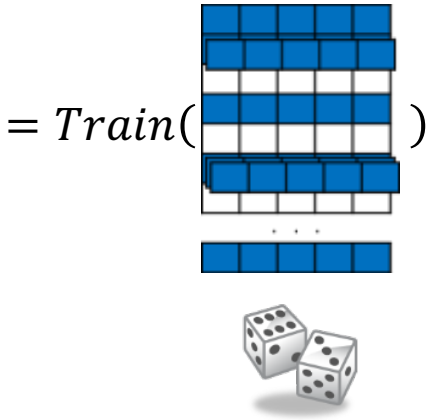
LSML #4

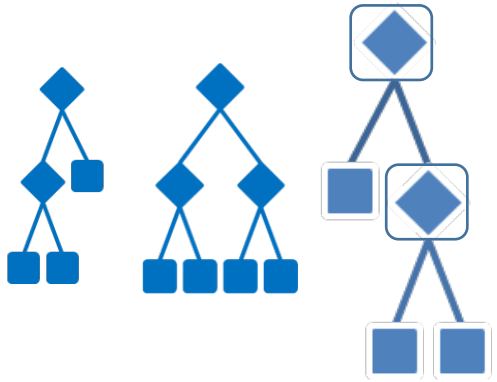
Градиентный бустинг

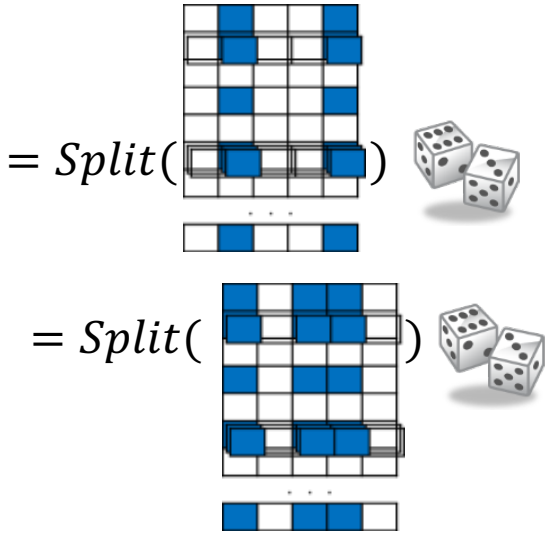
А как же Bagging и Random Forest?

- **Bagging** – учим каждое дерево на bootstrap выборке
- **RF** – bootstrap + семплирование признаков при каждом разбиении
- Легко параллелятся по деревьям
- Локальные для машины данные могут заменять bootstrap





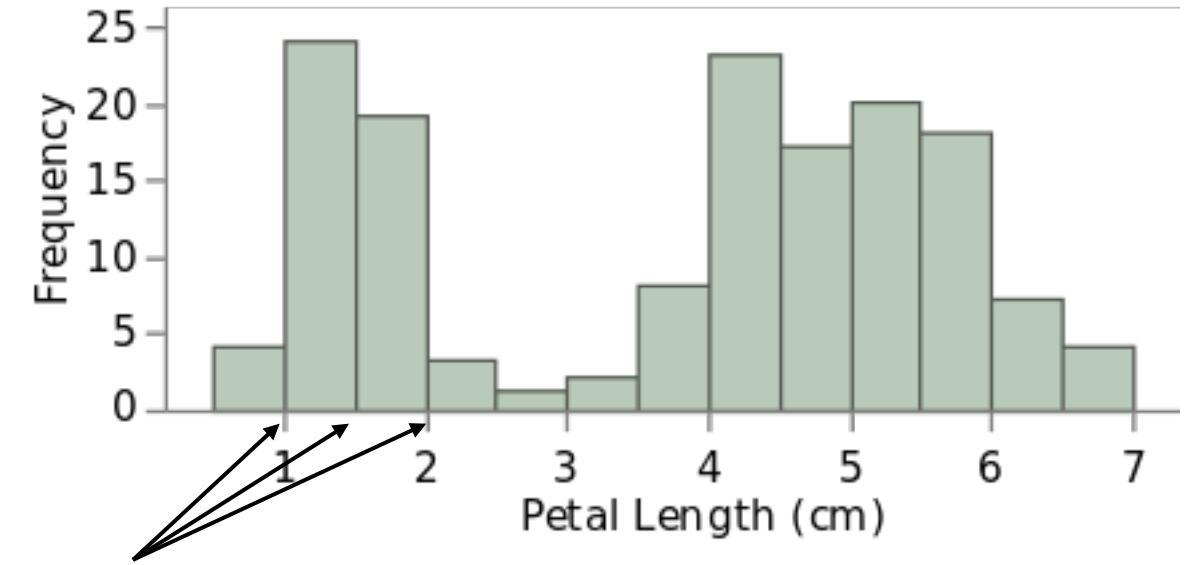




http://hunch.net/~large_scale_survey/

Gradient Boosting

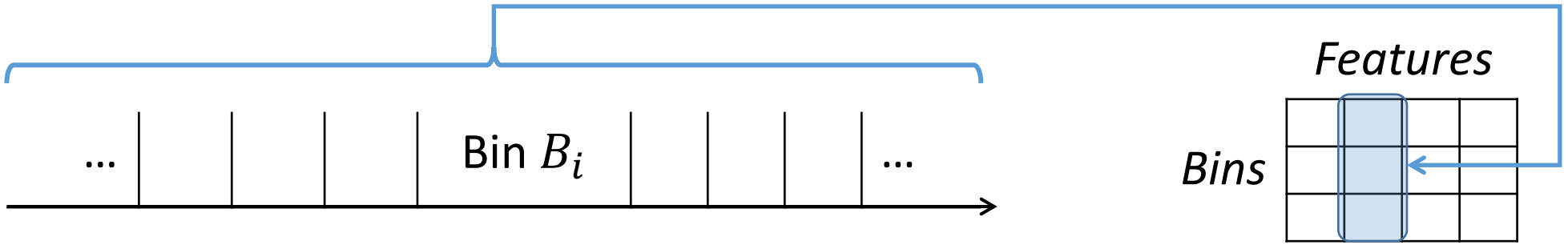
- Бустинг итерационный – надо параллелить создание дерева
- Для каждой вершины перебираются все признаки и пороги
  - Вещественные признаки дискретизируем по корзинкам (binning)
  - Порогами будут являться границы корзинок



Пороги

Feature Binning

- На примере задачи **регрессии**, разбили признак на k корзинок



- **Первый шаг**: собираем статистики

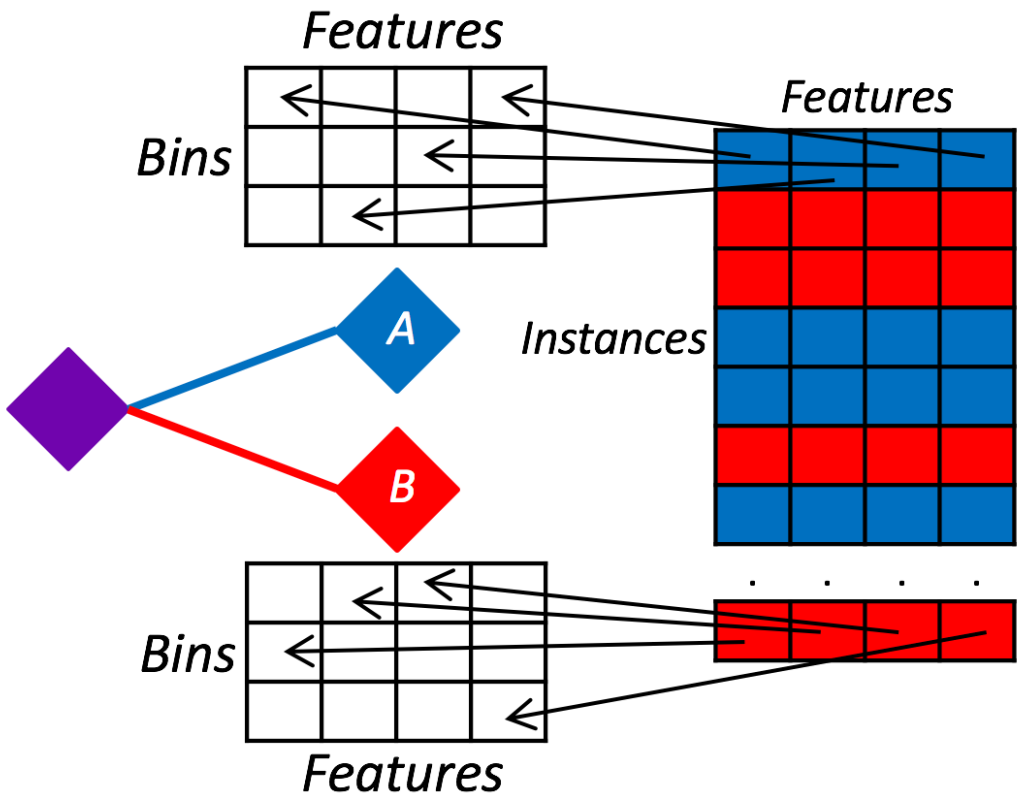
$$N_i = \sum 1 \qquad Y_i = \sum y \qquad N = \sum N_i \qquad Y = \sum Y_i$$

- **Второй шаг**: считаем пользу сплита по правой границе корзинки B_i

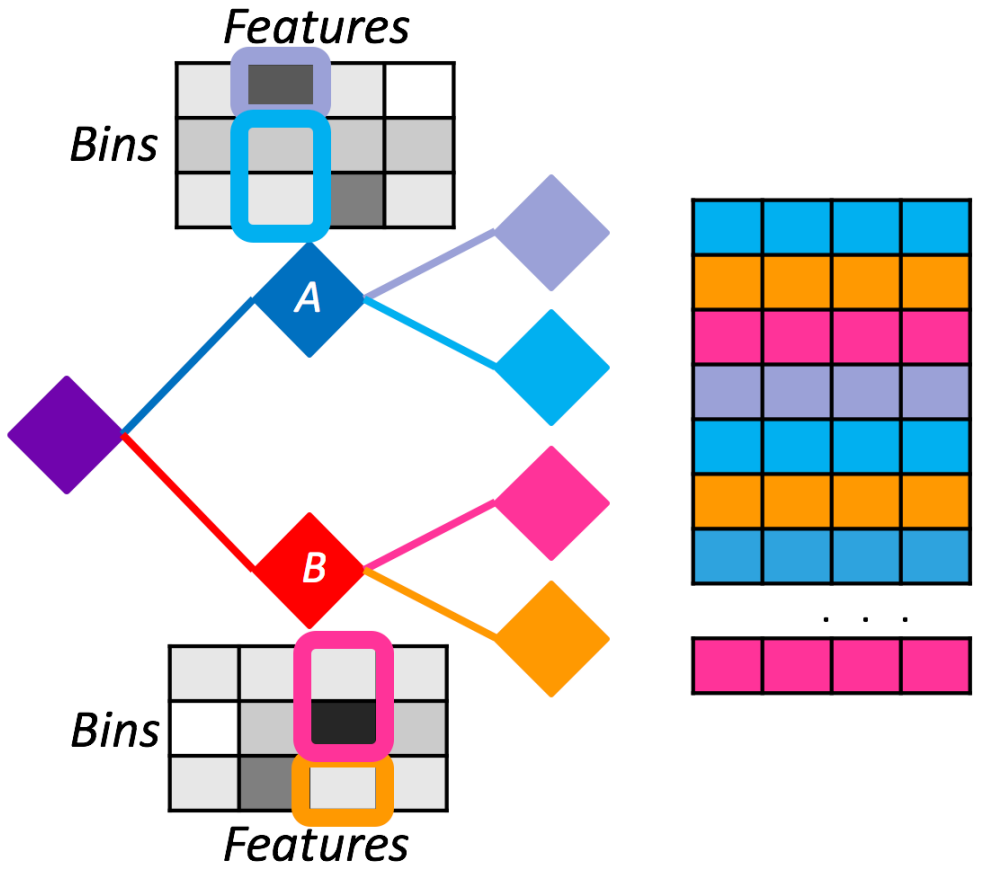
$$-\frac{Y_{0:i}^2}{N_{0:i}} - \frac{(Y - Y_{0:i})^2}{N - N_{0:i}}$$

$0{:}\,i$ – нотация суммирования

Feature Binning



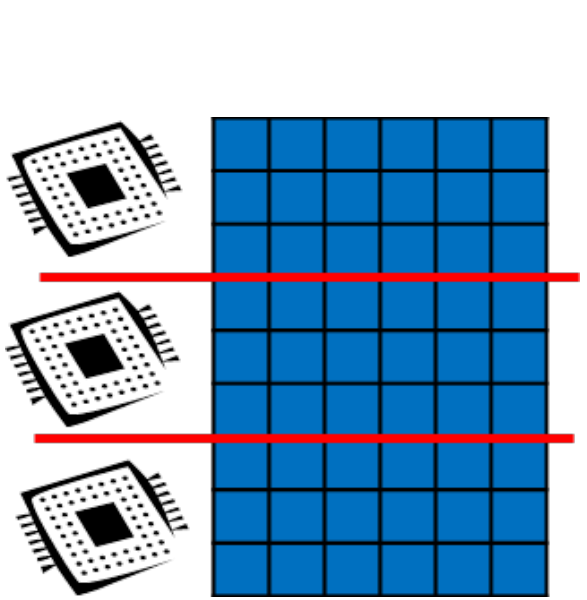
Первый шаг: считаем статистики

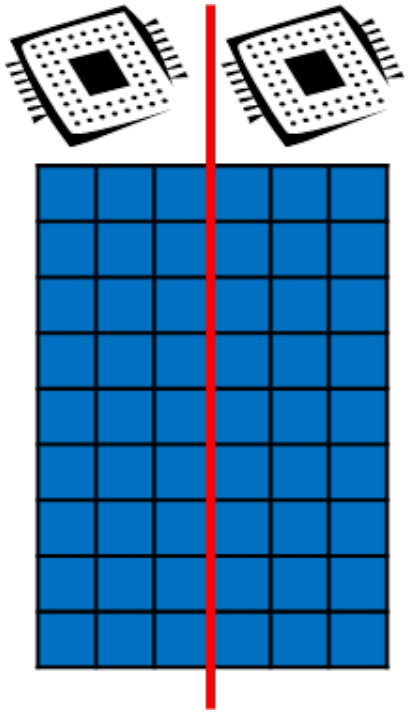


Второй шаг: выбираем сплит

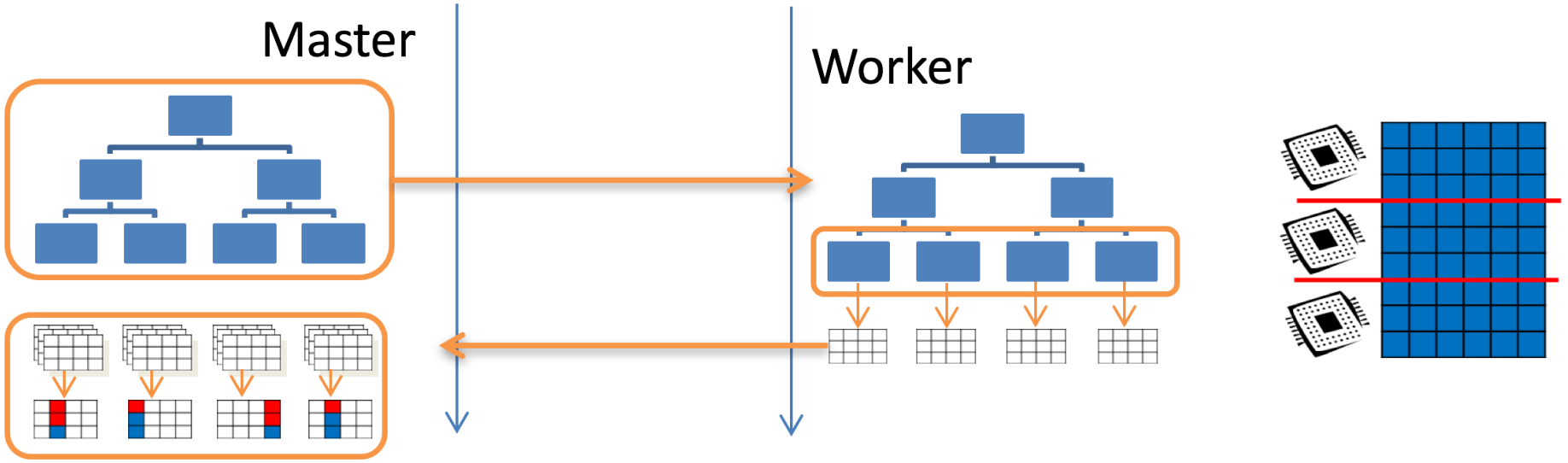
Feature Binning

- Один проход по данным для каждого уровня дерева
- Проход по данным можно распределять по объектам или признакам



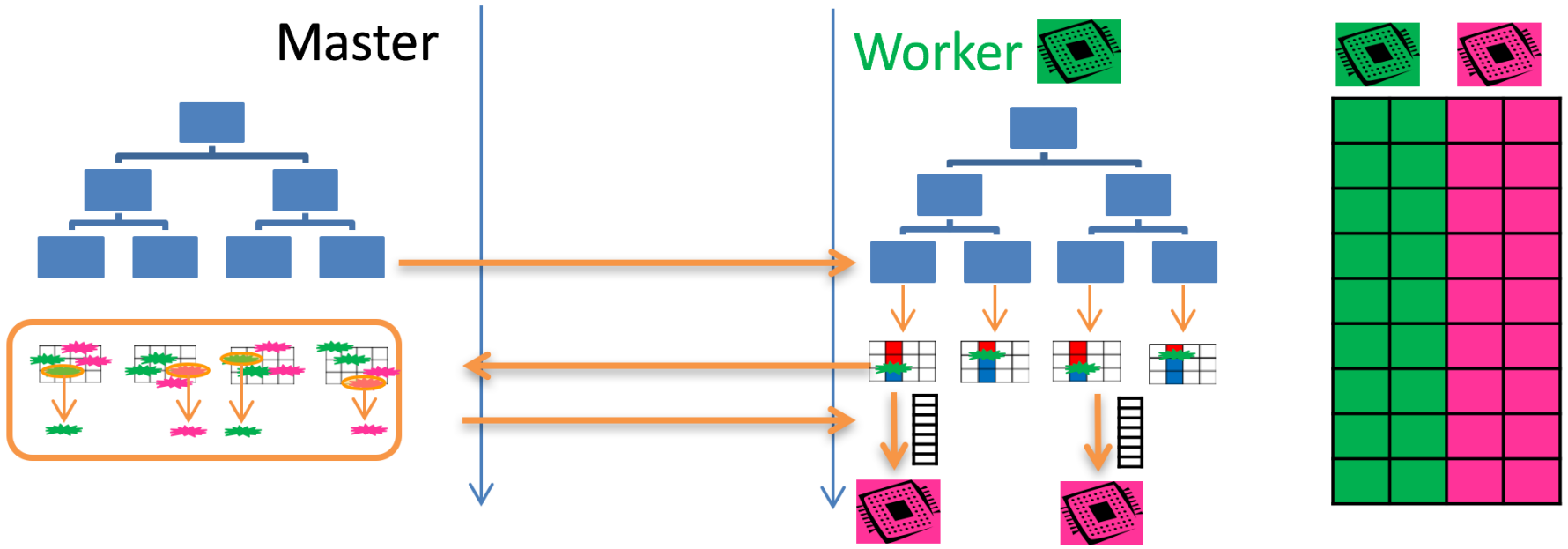


Распределяем объекты



- Мастер
  - Посылает воркерам текущую модель
  - Агрегирует локальные гистограммы (Features-Bins) воркеров и выбирает лучший сплит
- Воркеры
  - Делают проход по своим данным и заполняют локальные гистограммы

Распределяем признаки



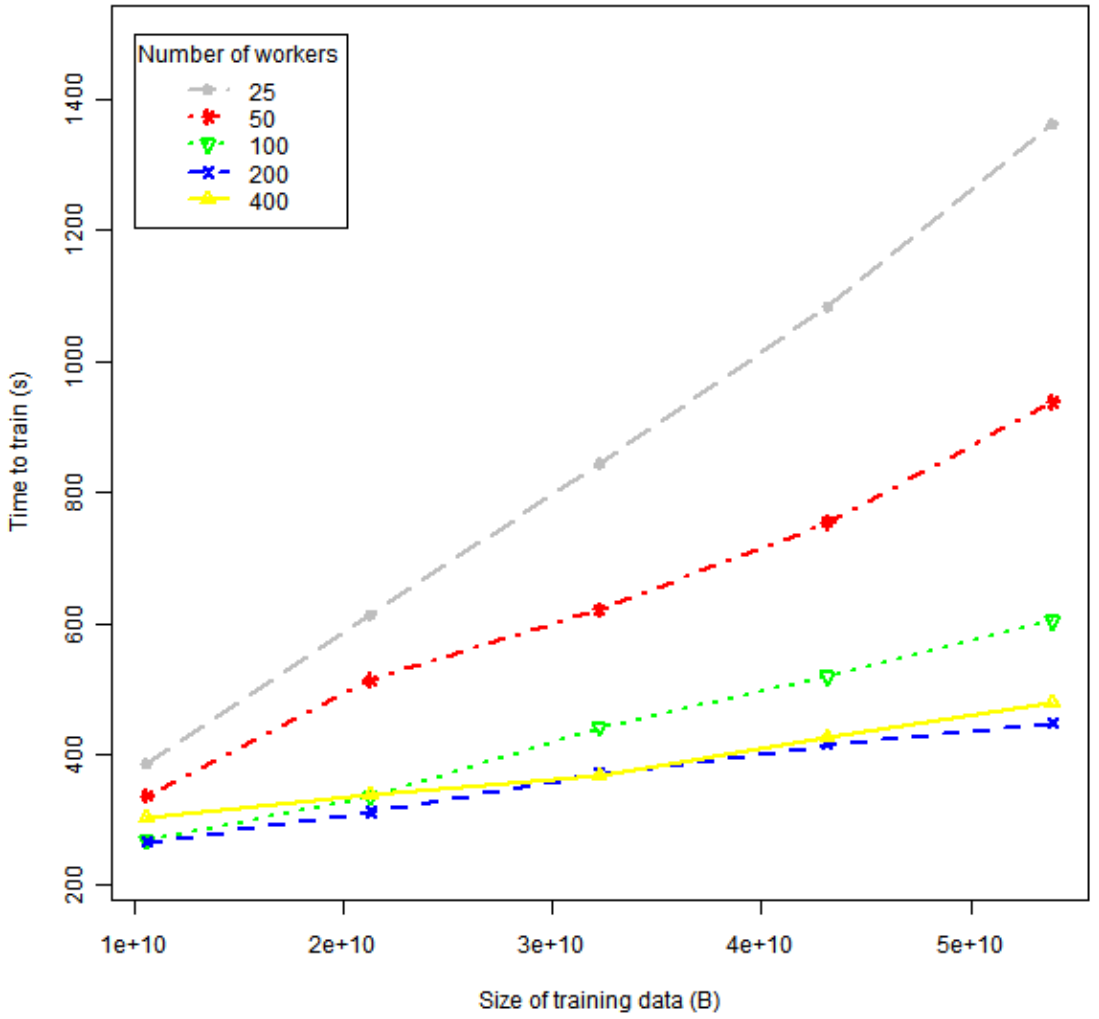
- Мастер
  - Получает лучшие сплиты по признакам от воркеров и выбирает лучший среди них
  - **Просит лучших разослать всем остальным информацию про новый выбранный сплит**
- Воркеры
  - **Всех признаков нет**, помнят в какой лист попадает каждый объект и какие там предсказания
  - Делают проход по своим признакам и выбирают лучший сплит

Пример реализации: PLANET (2011)

- MapReduce, RPC
- Разбиение по объектам
- Binning во время инициализации
  - За один проход оценивают квантили

Пример реализации: PLANET (2011)

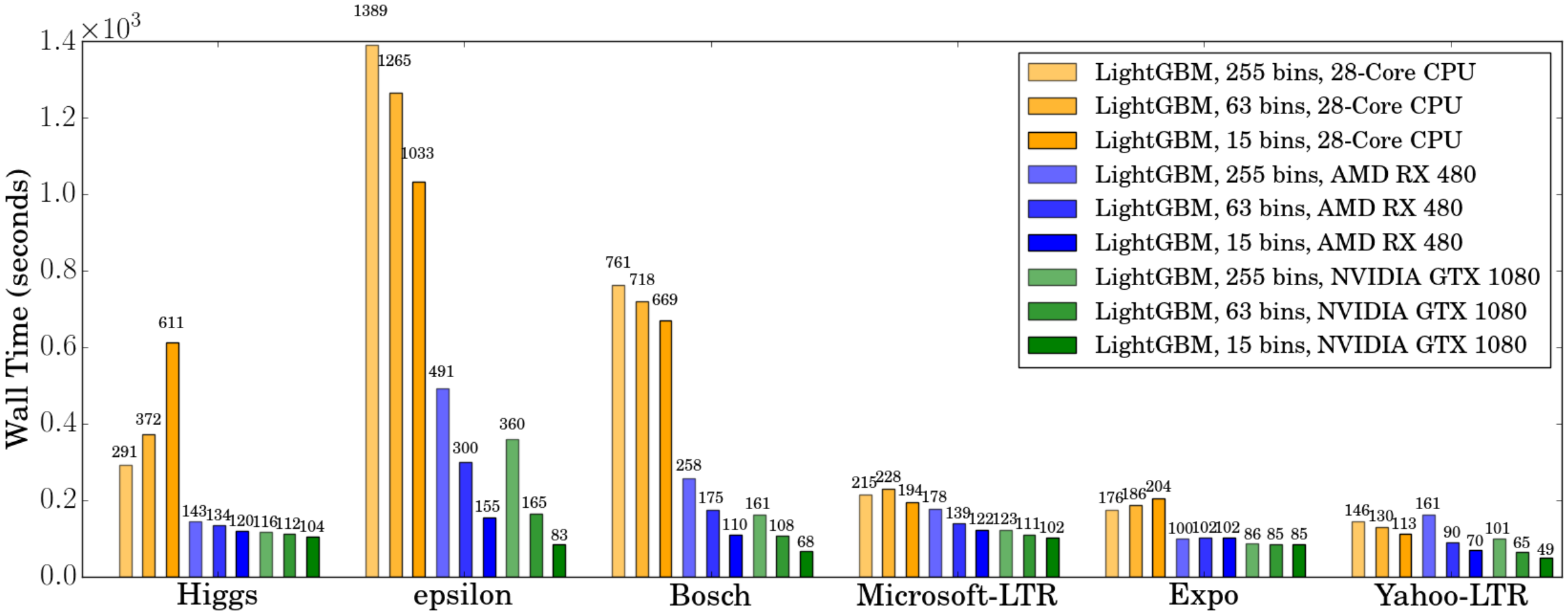
- Бинарная классификация
- Признаки:
  - 4 вещественных
  - 6 категориальных, $|C| \in$[2-500]
- 314 млн объектов
- Деревья глубины 3



Пример реализации: Yahoo! GBDT (2009)

- Hadoop, MPI
- Разбиение по объектам или признакам
- Результаты:
  - MapReduce Horizontal: 211 minutes x 2500 trees = 366 days (100 machines)
  - MapReduce Vertical: 28 seconds x 2500 trees = 19.4 hours (20 machines)
  - MPI: 5 seconds x 2500 trees = 3.4 hours (10 machines)

Пример реализации: LightGBM



https://github.com/Microsoft/LightGBM/blob/master/docs/GPU-Performance.md

Пример реализации: Матрикснет

- Разбиение по документам
- Ускорение обучения (CPU, GPU)
- Новые регуляризации и функции ошибок
- Хорошие модели без подбора параметров

Пример реализации: Матрикснет

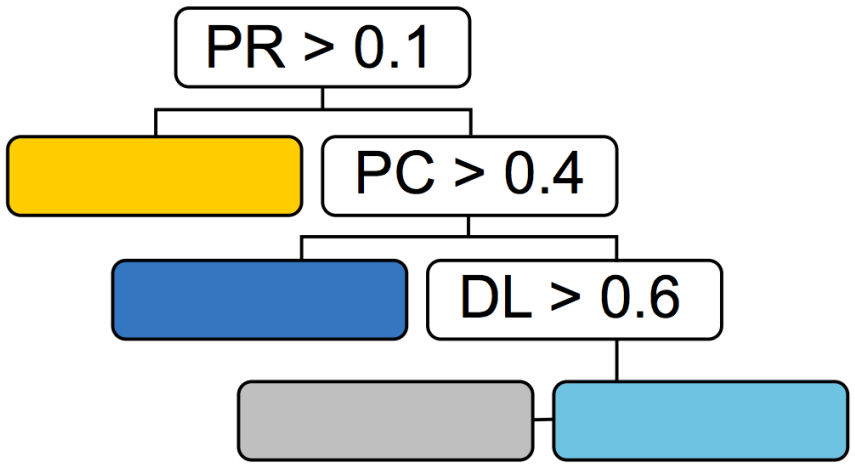
	Matrixnet	Azure Boosted DT	XGBoost	LightGBM
Pol	0,994	0,922 ↓0,14%	0,991 ↓0,23%	0,991 ↓0,23%
2dplanes	0,9476	0,9474 ↓0,02%	0,9474 ↓0,02%	0,9474 ↓0,01%
Elevator	0,915	0,909 ↓0,67%	0,9 ↓1,54%	0,908 ↓0,74%
Ailerons	0,86	0,856 ↓0,45%	0,837 ↓2,67%	0,856 ↓0,55%
Fried	0,957	0,955 ↓0,22%	0,954 ↓0,32%	0,955 ↓0,17%
House	0,677	0,68 ↑0,51%	0,658 ↓2,72%	0,661 ↓2,23%

Сравнение на открытых наборах данных
с сайта www.openml.org
Метрика – R2-коэффициент детерминации

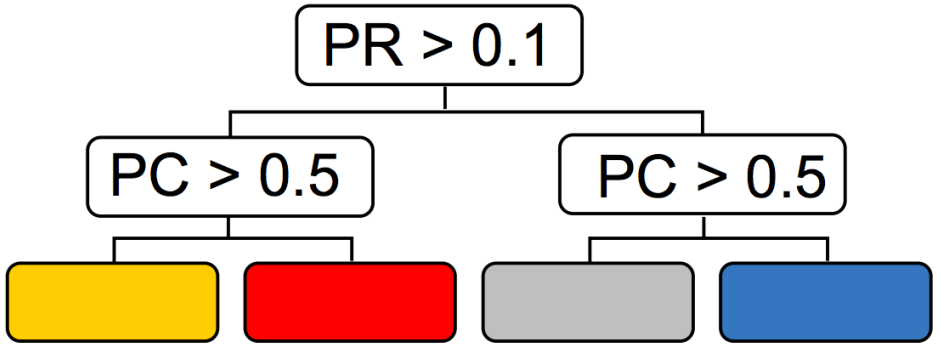
https://yadi.sk/i/3nzyqx063EAYmi

Oblivious trees в Матрикснет

Дерево решений



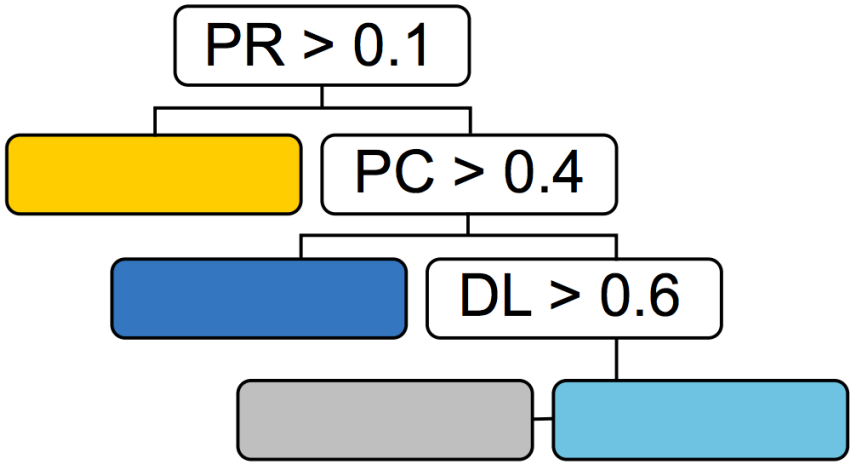
Oblivious дерево



https://yadi.sk/i/3nzyqx063EAYmi

Oblivious trees в Матрикснет

Дерево решений



Oblivious дерево

	PR ≤ 0.1	PR > 0.1
PC > 0.5		
PC ≤ 0.5		

Ссылки

- Как устроен xgboost http://www.kdd.org/kdd2016/papers/files/rfp0697-chenAemb.pdf
- xgboost: https://homes.cs.washington.edu/~tqchen/pdf/BoostedTree.pdf
- LightGBM и обучение на GPU https://github.com/Microsoft/LightGBM/blob/master/docs/GPU-Performance.md