# MLLib - машинное обучение в Spark

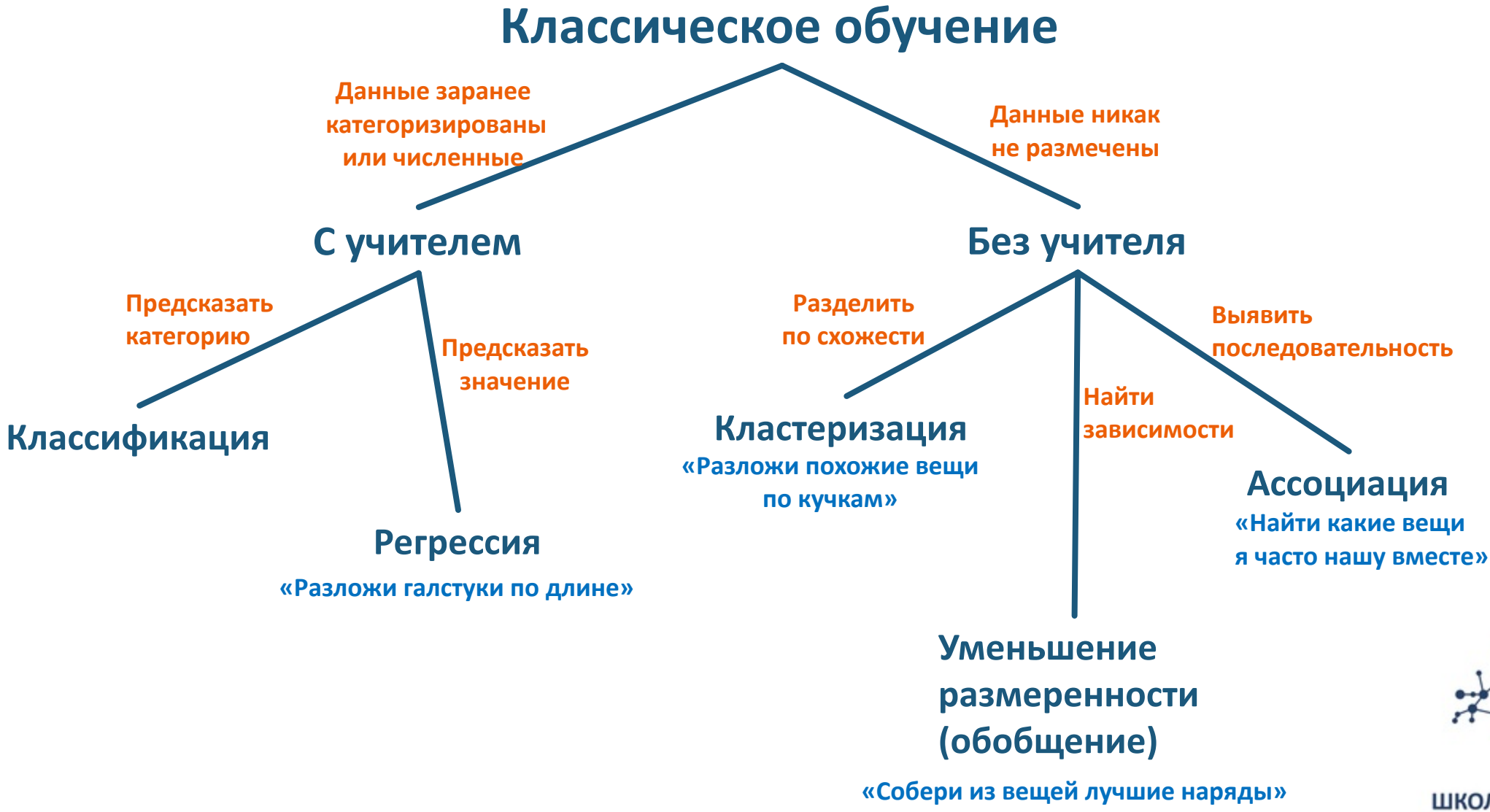
МОДУЛЬ 11


ШКОЛА БОЛЬШИХ ДАННЫХ

# Что такое ML

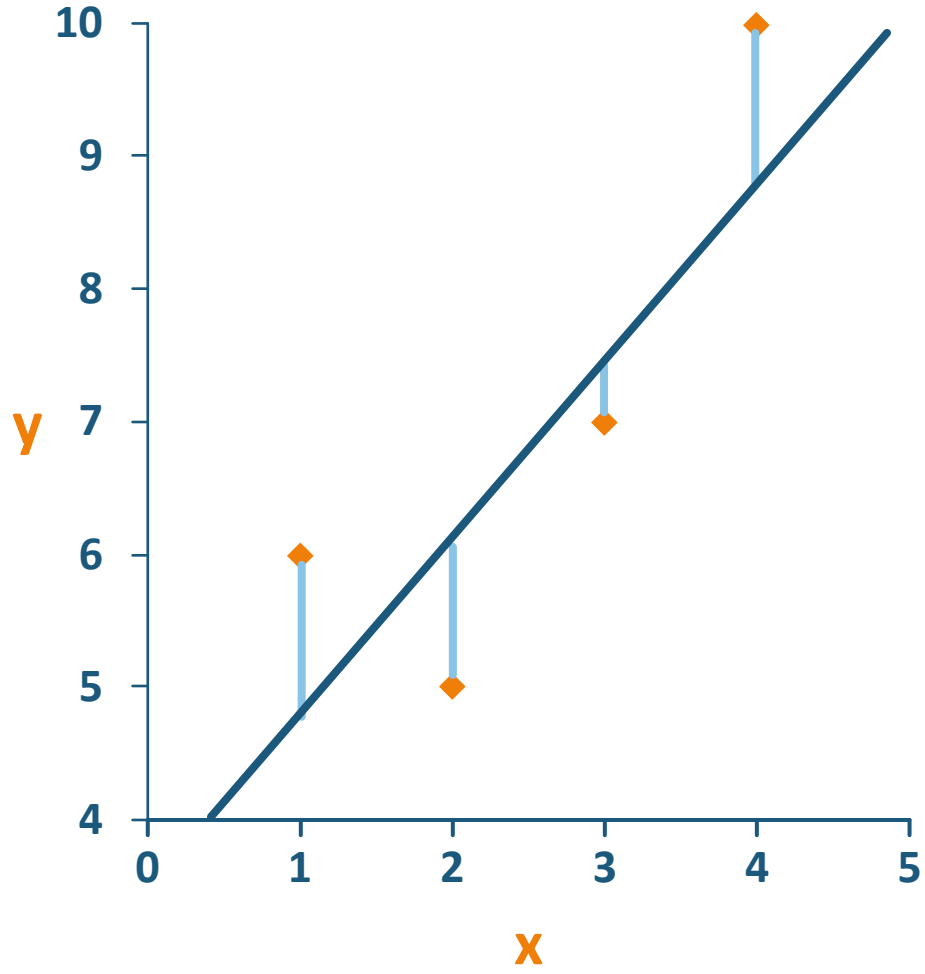
- формулируется вопрос
- собираются и подготавливаются необходимые данные
- специальным образом настраиваются параметры алгоритма
  - ✓ используются данные
  - ✓ алгоритм называют “модель” (“**model**”)
  - ✓ процесс настройки - `обучение` (“**train**” или “**fit**”)
- **модель** - это функция от данных, возвращающая один результат (**число**)

# Классификация алгоритмов




ШКОЛА БОЛЬШИХ ДАННЫХ

# Пример алгоритма: линейная регрессия




ШКОЛА БОЛЬШИХ ДАННЫХ

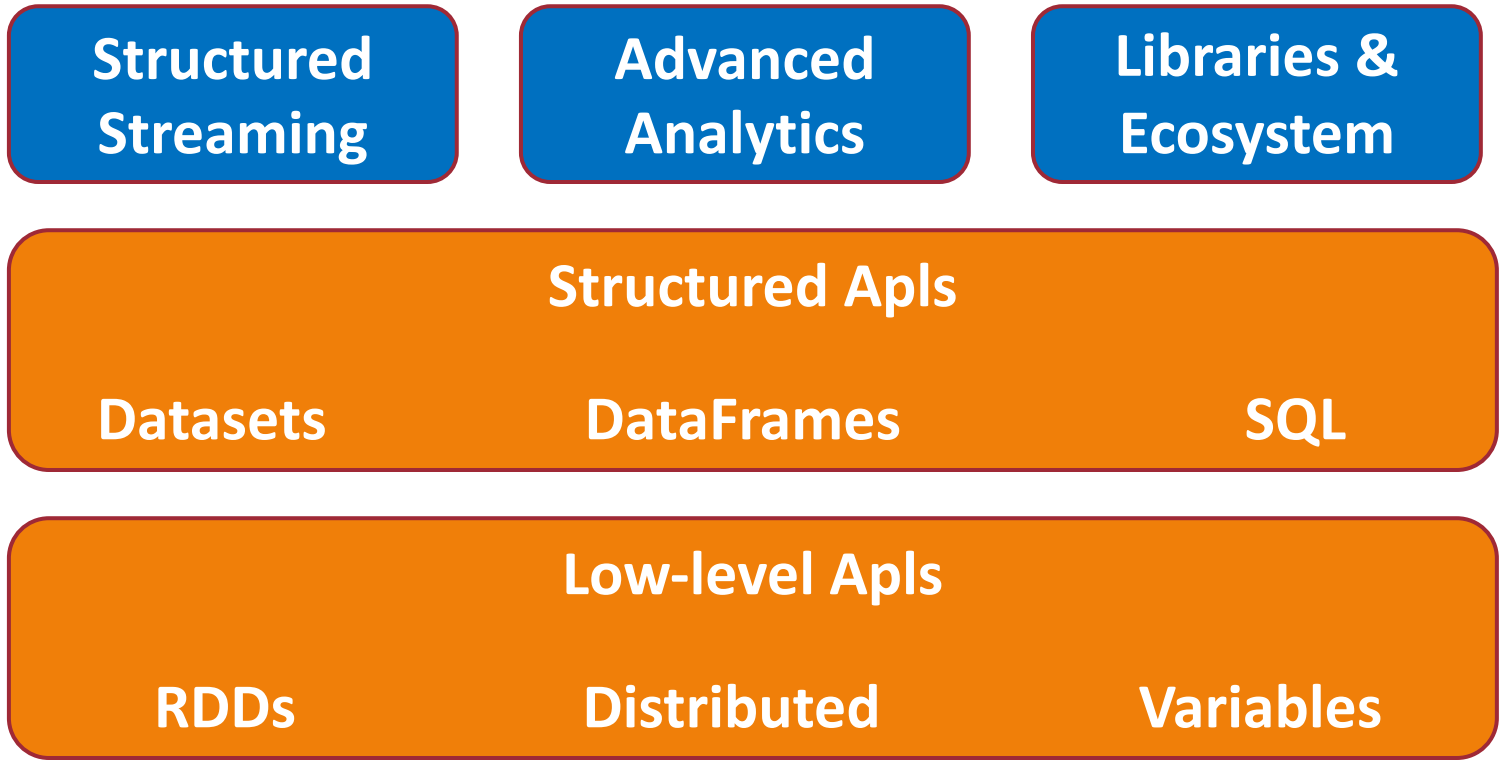
# Процесс обучения

- делим данные на **“train”**/ **“test”**
- обучаем на **“train”**
- проверяем на **“test”**

**Качество полученной модели**

- **метрики**: позволяют получить количественную оценку качества
  - ✓ пример: средняя ошибка (**MAE**)
- **проблемы**
  - ✓ низкое качество (**“underfit”**)
  - ✓ **“overfit”**
- **важно**: метрики есть (но не всегда)

# Состав Spark



# Почему MLlib

- есть множество **ML библиотек**
  - ✓ **single machine**
- Spark для подготовки данных
  - ✓ возможно без **MLlib**
- MLlib для обучения на **больших наборах данных**
- **единая технологическая платформа**

**НО**

- ✓ нет out-of-the-box способов **развертывания** полученных моделей

# Дополнительные абстракции

- **“Vector”**: данные для моделей
- **“Estimator”**: класс ML алгоритмов
- **“Transformer”**: модель, построенная с помощью **“estimator.fit()”**
- **“Evaluator”**: оценка качества модели

`Technically, an Estimator produces a Model (i.e. a Transformer) for a given DataFrame and parameters (as ParamMap). It fits a model to the input DataFrame and ParamMap to produce a Transformer (a Model) that can calculate predictions for any DataFrame-based input datasets.`

`A evaluator is a transformation that maps a DataFrame into a metric indicating how good a model is.`

# Базовый workflow

- готовим данные (**“dataframe”**)
- преобразуем в датафрейм с двумя колонками
  - ✓ **“label”**: целевая переменная
  - ✓ **“features”**: вектор фич
- разбиваем на **тренировочные** и **тестовые** данные
- **обучаем** модель на тренировочных данных
- **проверяем** модель на тестовых данных
- **используем** модель для предсказания на новых данных

# Использование pipelines

- **“pipeline”**: конвейер трансформаций
- **“stage”**: стадия трансформации
- **“pipeline.fit()”**: обучение конвейера
- **“pipeline.transform()”**: обработка данных

# Feature engineering

- два этапа
  - ✓ подготовка датасета (**“dataframe”**)
  - ✓ обработка датасета (**“dataframe с vector”**)
- **“Estimator”** и **“Transformer”**
- использование в **“pipeline”**
- примеры преобразований
  - ✓ непрерывные данные
  - ✓ категориальные данные
  - ✓ преобразование текста
- ориентация на **Spark MLLib** (дальнейшее использование)

# Обзор возможностей MLlib

- классификация
  - ✓ логистическая регрессия
  - ✓ деревья решений
  - ✓ случайный лес и градиентный бустинг
  - ✓ наивный байес
- регрессия
  - ✓ линейная регрессия
  - ✓ обобщенная линейная регрессия
  - ✓ деревья решений
  - ✓ случайный лес и градиентный бустинг
- рекомендации
  - ✓ ALS (Alternating Least Squares)
- обучение без учителя
  - ✓ k-means
  - ✓ гауссово моделирование смеси (GMM)
  - ✓ латентное размещение Дирихле (LDA)
- глубокое обучение
  - ✓ использование возможностей кластера и предобученных моделей

# Вопросы?