Лекция. Модели машинного обучения

Типы машинного обучения.

Выделяется четыре параметра разбиения методов машинного обучения на типы:

1. С учителем и без учителя. Обучению с учителем соответствует случай, когда «опыт» – обучающий пример – содержит существенную информацию, отсутствующую в еще не виденных «тестовых примерах», к которым будут применены обретенные в ходе обучения знания. При таком подходе цель приобретения знаний – предсказать отсутствующую в тестовых данных информацию. В случае обучения без учителя между обучающими и тестовыми данными нет никакого различия. Обучаемый обрабатывает входные данные, имея целью составить на их основе некоторый дайджест, или сжатое представление. Типичный пример такой задачи – кластеризация набора данных, т. е. разбиение его на подмножества похожих объектов.

Существует также промежуточный тип обучения, когда обучающие примеры содержат больше информации, чем тестовые, а обучаемый должен предсказать для тестовых примеров еще больше информации. Такие подходы обычно изучаются в дисциплине, называемой обучение с подкреплением.

2. Активный и пассивный обучаемый. Парадигмы обучения могут классифицироваться по роли обучаемого. Различаются «активный» и «пассивный» обучаемые. Активный обучаемый взаимодействует с учителем на этапе обучения, например, задавая вопросы или ставя эксперименты, тогда как пассивный только наблюдает за информацией, поставляемой учителем, не оказывая на нее никакого влияния и не направляя процесс обучения.

3. Полезность учителя. Модель «статистического обучения» предполагает, что обучающие данные (опыт обучаемого) порождаются некоторым случайным процессом, за которым мы наблюдаем. Модель «противодействующего учителя» – так называемое состязательное обучение – применяется в условиях, когда учитель активно противодействует обучению.

4. Онлайновое и пакетное обучение. Этот тип описывает ситуации, в которых обучаемый должен отвечать в режиме онлайн на протяжении всего процесса обучения, и когда разрешается применить обретенные знания уже после обработки значительного объема данных.

Модели и схема обучения

Модели – центральная концепция машинного обучения, поскольку это именно то, что порождается в результате обучения на данных с целью решения поставленной задачи. Разнообразие моделей велико. Можно выделить систематичность в моделях машинного обучения. Существуют следующие группы моделей:

Процесс машинного обучения: Классы моделей

Классы моделей в ML

- **Обучение с учителем** (supervised learning)
  - Классификация
  - Регрессия
  - Ранжирование
- **Обучение без учителя** (unsupervised learning)
  - Кластеризация
  - Уменьшение размерности
- **Обучение с частичным привлечением учителя** (semi-supervised learning)
- **Обучение с подкреплением** (reinforcement learning)

Классификация: Важные моменты

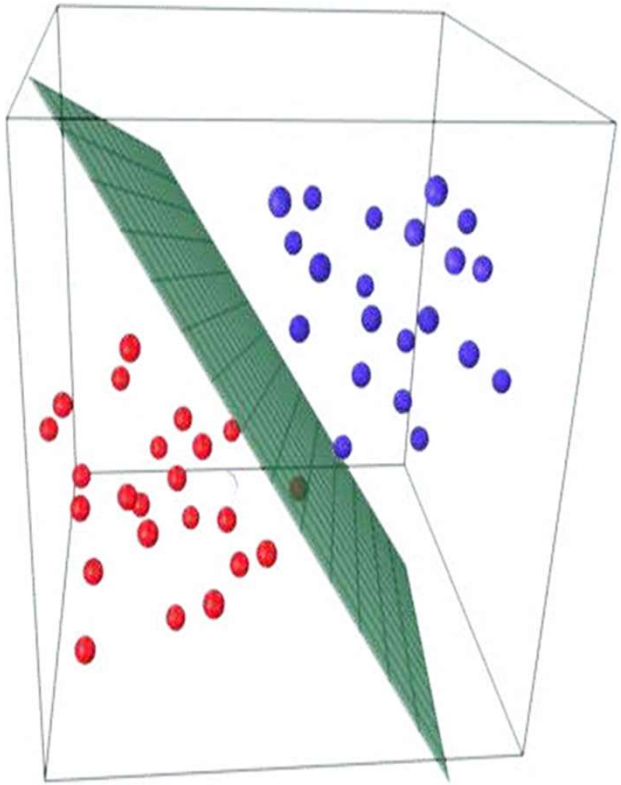
- Классификация – это метод обучения с учителем.
  Требуется размеченная выборка
- Метка принимает набор дискретных значений
  (эквивалентно номерам классов + словарю)
- Вид признакового пространства и расположение классов внутри него определяют какой классификатор справится с задачей (но визуализировать это практически невозможно для большого числа измерений)
- Частный популярный класс классификаторов – линейные классификаторы с разделяющей гиперплоскостью в качестве границы.

Классификация: Популярные методы

- Логистическая регрессия
- Машина опорных векторов (Support vector machine, SVM)
- Наивный Байес
- Деревья решений
- Нейросети
- Ансамбли (Random Forests, Gradient Boosted Trees)
- k-NN
- ...

Классификация: Логистическая регрессия

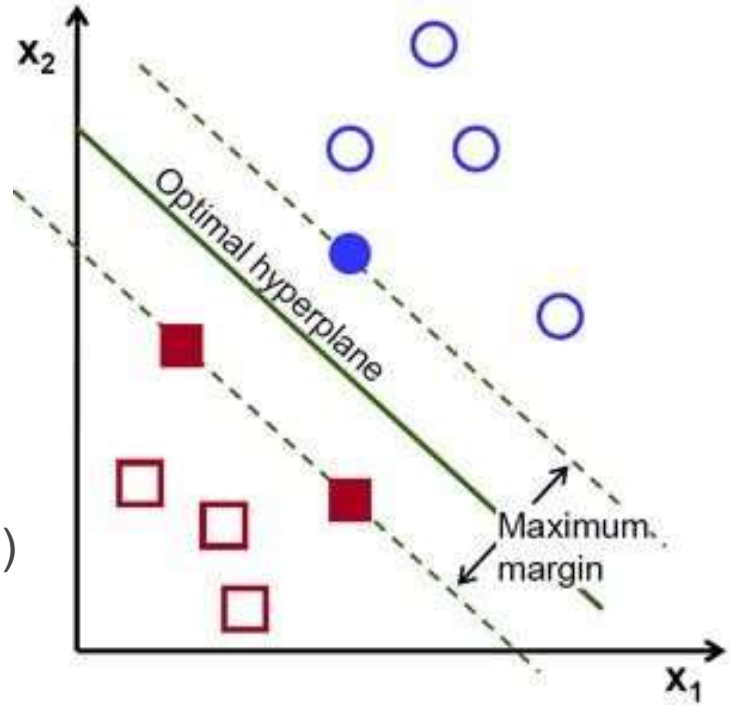
- Один из самых простых классификаторов
- Довольно эффективен на малом объёме данных и может превзойти более сложные методы (деревья решений, deep learning), но проиграет на большом объёме.
- Линейный классификатор (но можно сделать нелинейным с помощью генерации новых признаков, например, квадратичных)
- Вид модели: линейное правило + нелинейная функция после него (для приведения к диапазону от 0 до 1)



$$z = \theta^T x = \theta_1 x_1 + \ldots + \theta_n x_n \qquad f(z) = \frac{1}{1 + e^{-z}}.$$

Классификация: SVM

- SVM = Support Vector Machine = Машина опорных векторов
- Простой вариант линейный, но часто используются более сложные варианты с использованием kernel trick, который по сути производит перевод в новое пространство
- Идея: различных разделяющих плоскостей может быть много, надо выбрать “лучшую”.
- Алгоритм использует предположение, что чем больше расстояние между разделяющей гиперплоскостью и ближними к ней объектами классов, тем меньше будет средняя ошибка классификатора в последующей работе.
  Те самые ближайшие к границе (разделяющей гиперпроскости) вектора и называются опорными.

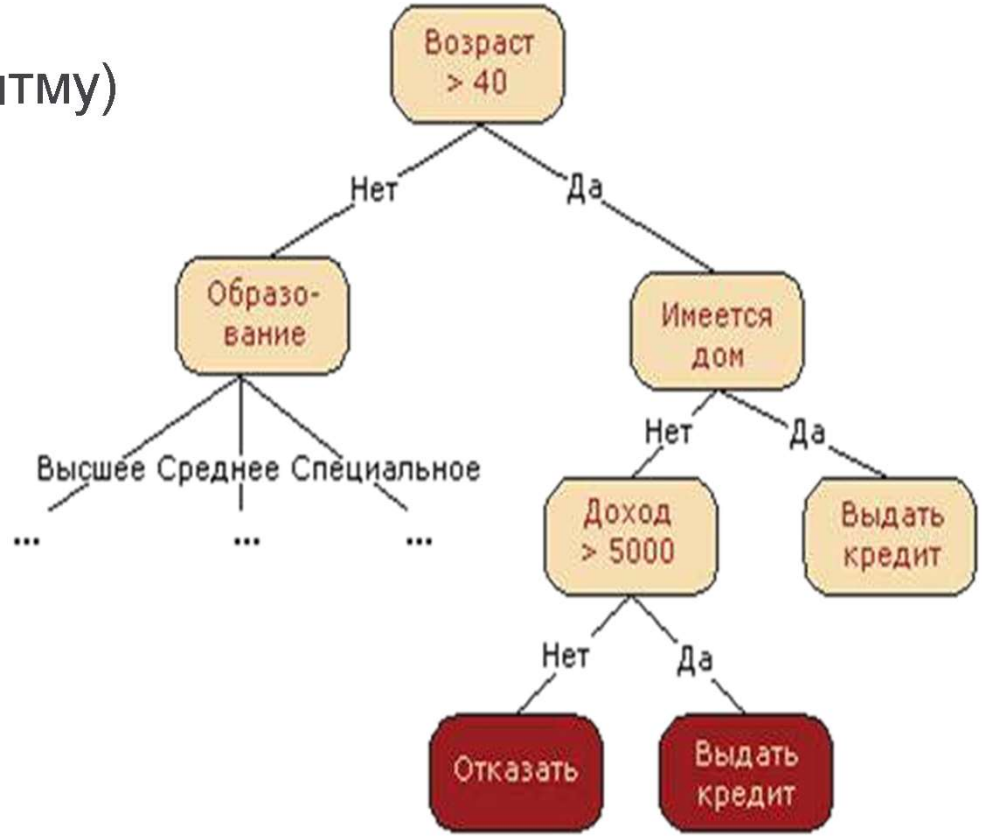


Классификация: Деревья решений

Набор узлов, в каждом из которых принимается решение по одной из переменных, куда двигаться дальше.

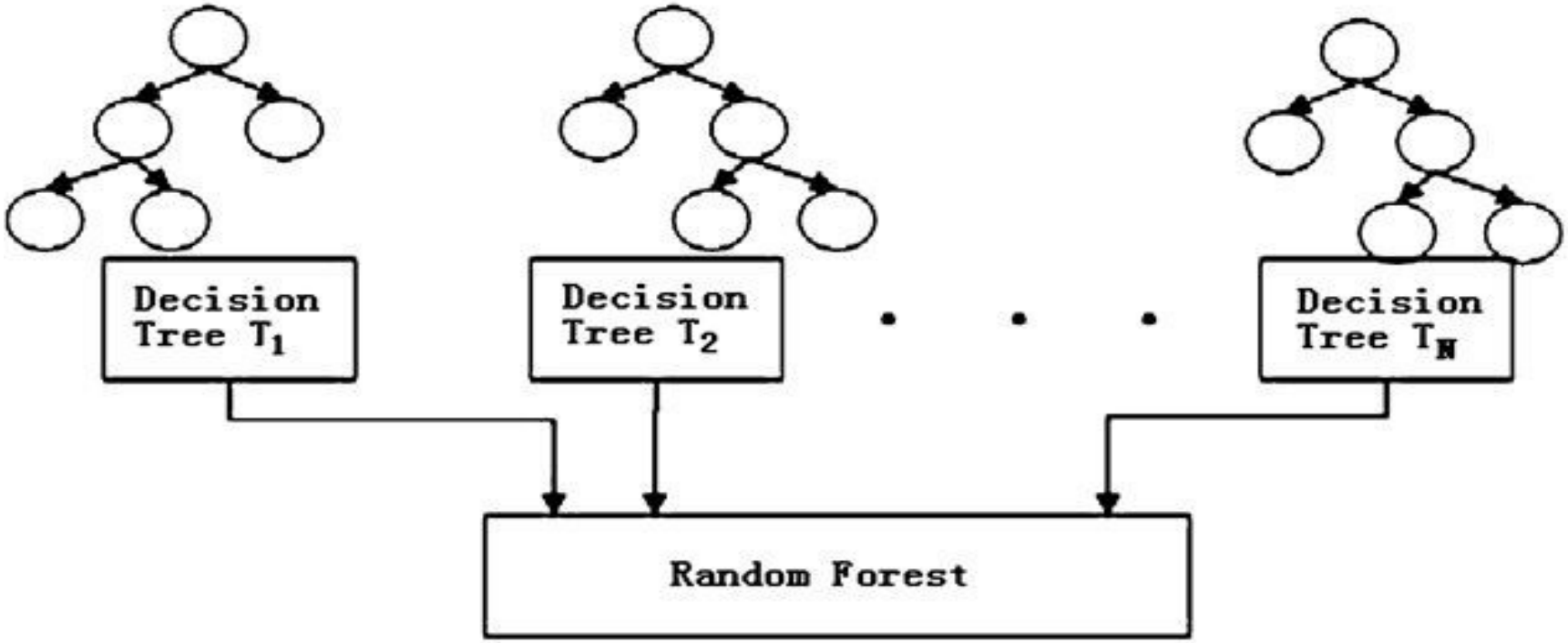
Удобны в интерпретации

(это может быть требованием к алгоритму)



Классификация: Ансамбли

Идея: Использовать много простых классификаторов (например, 50 простых деревьев) и агрегировать результат. Пример: Random Forest.



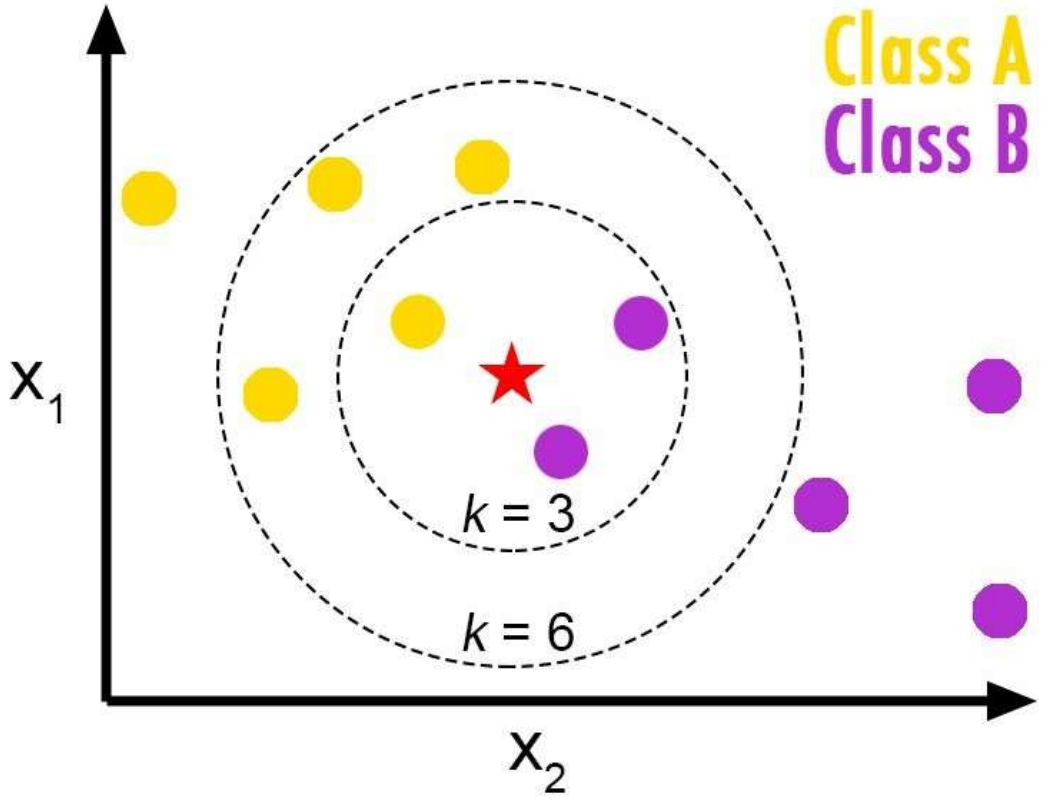
Классификация: kNN

K ближайших соседей (k nearest neighbors).

Не путать с NN (нейросети).

Простой классификатор с сильно нелинейной границей.

Требует запоминания обучающих примеров, а сама классификация происходит по “голосованию” среди K соседей нового объекта ещё не известного класса.



Регрессия

Есть обучающая выборка, в которой представлены объекты в виде их признакового описания (вектор признаков) и значения целевой переменной (непрерывной в отличие от классификации).

Надо найти алгоритм, который для каждого нового объекта (его признакового описания) спрогнозирует значение целевой переменной.

Регрессия: Важные моменты

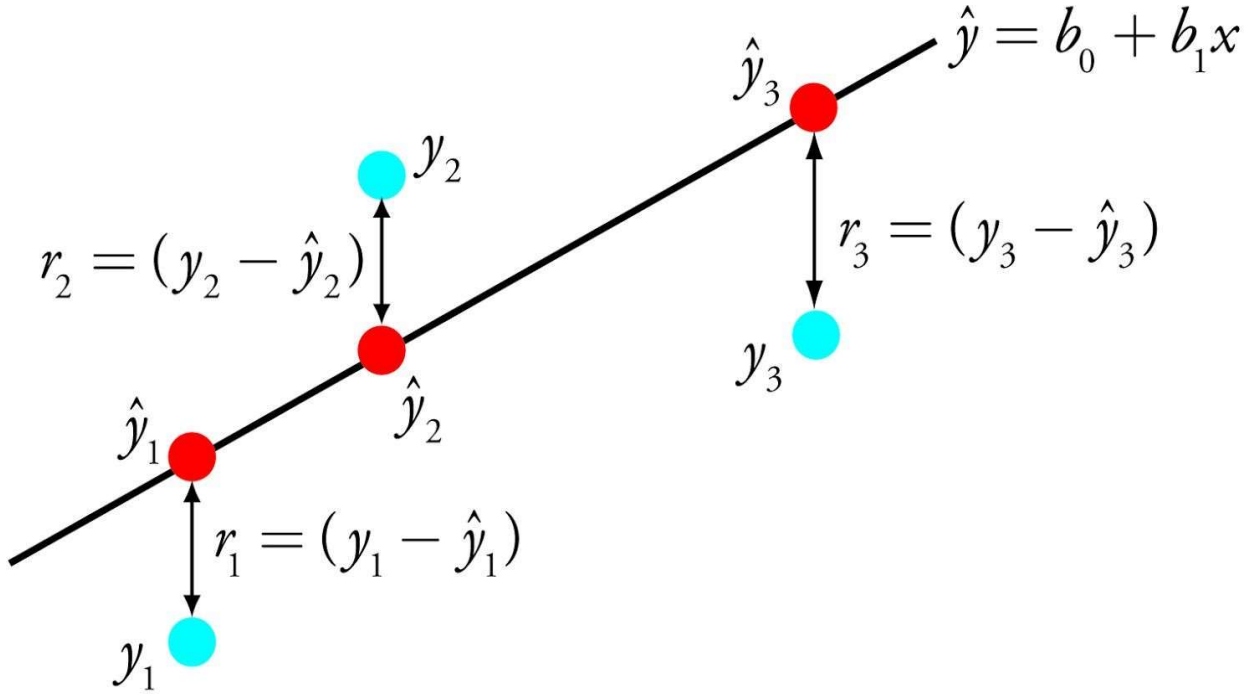
- Регрессия – это метод обучения с учителем. Требуется размеченная выборка
- Целевое значение может быть любым действительным числом
- Аналогично классификации есть линейные и нелинейные модели. Нелинейную регрессию часто можно получить с помощью генерации новых признаков из старых.
- Может быть чувствительна к выбросам (но есть методы борьбы)

Регрессия: Методы

- Линейная регрессия (в том числе множественная)
- Регрессионные деревья
- Регрессия через SVM
- Нейросети
- ...

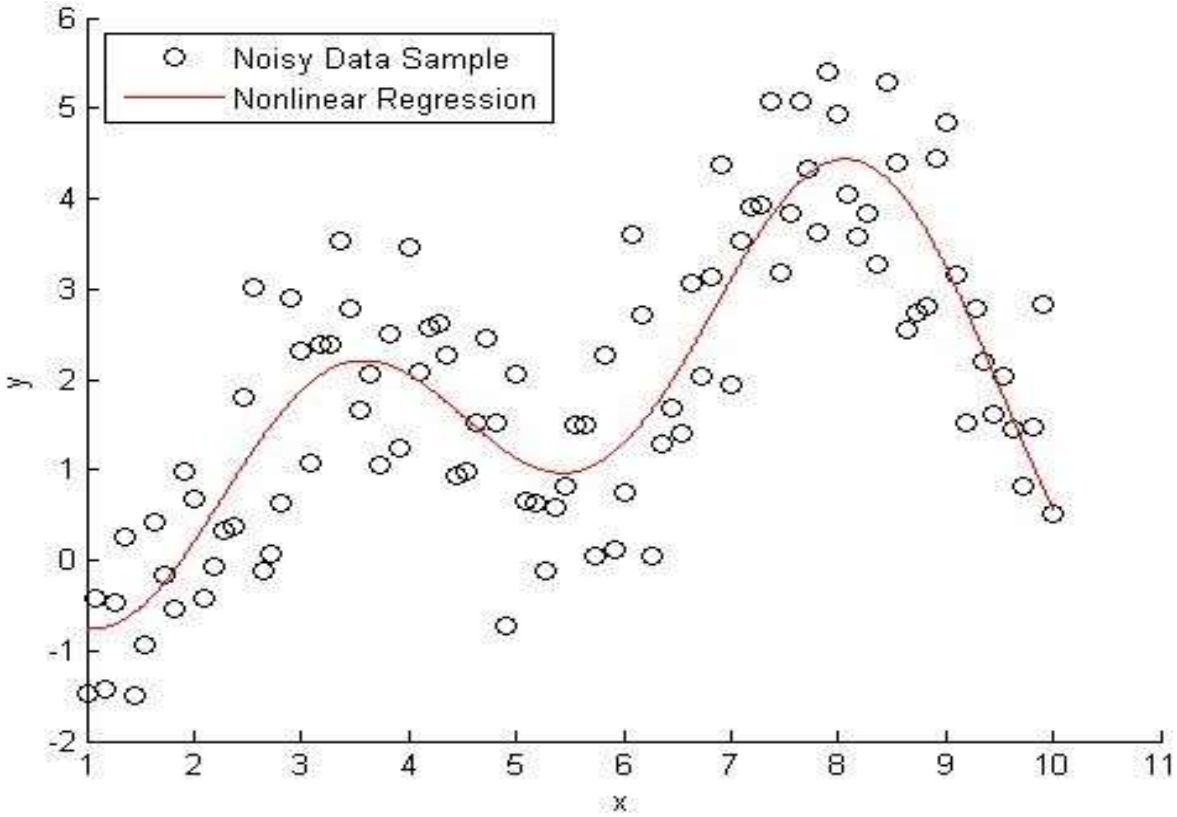
Регрессия: Линейная регрессия

Простой алгоритм. Вписать линию, которая “лучшим” образом проходит через облако точек (наших примеров). Решение методом наименьших квадратов.



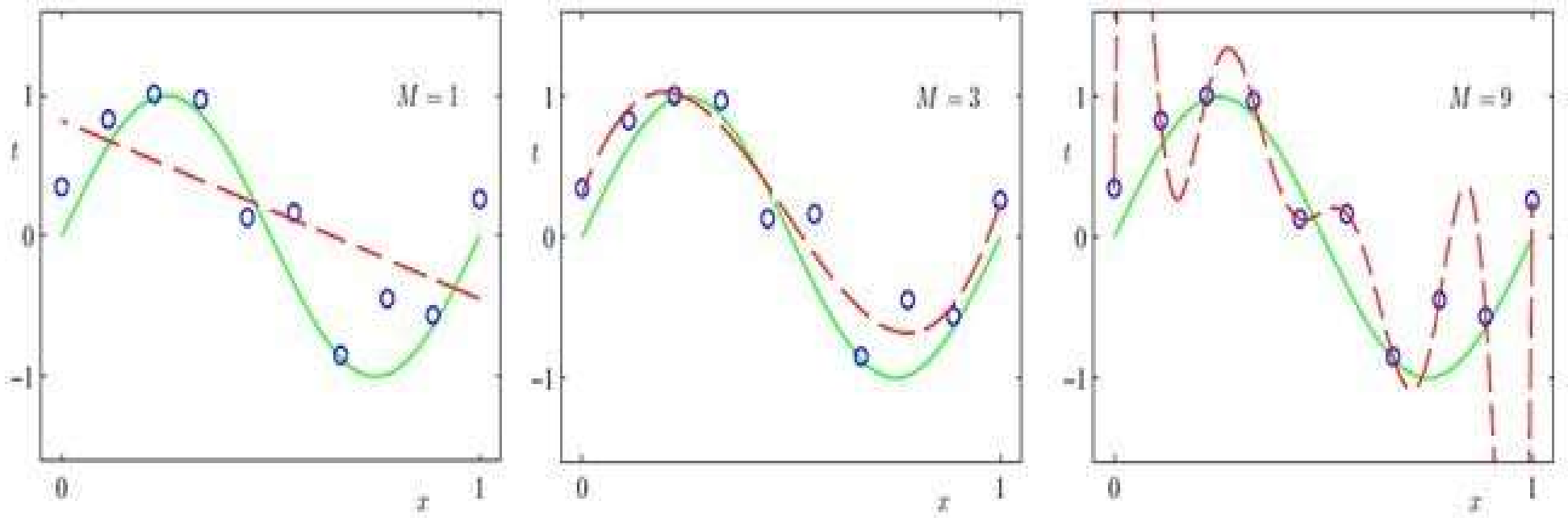
Регрессия: Нелинейная регрессия

Можно получить не меняя математический аппарат, добавив сгенерированные признаки (например, полиномиальные).

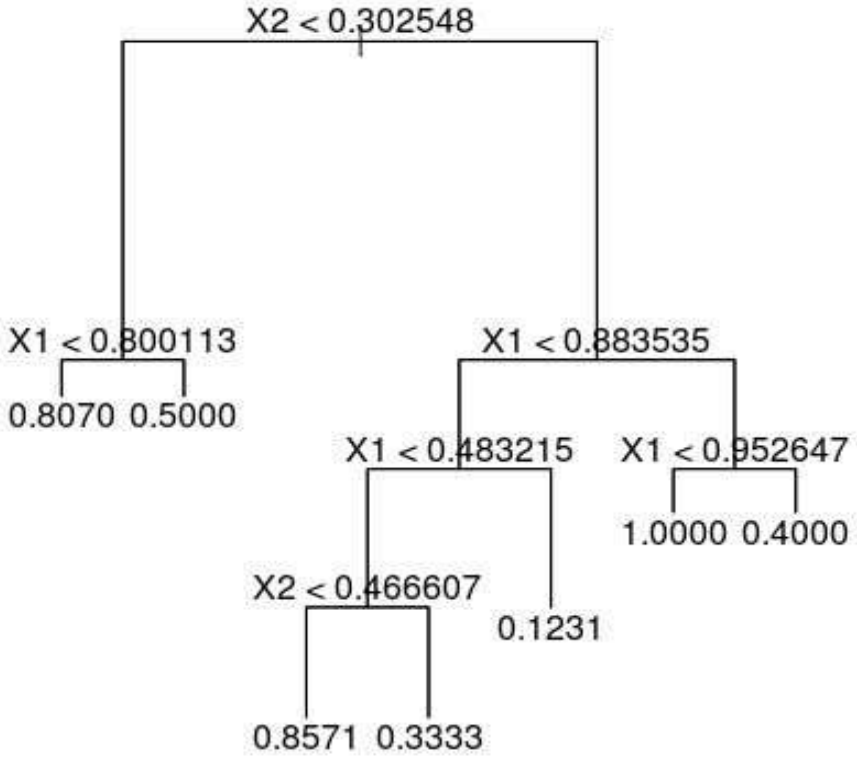


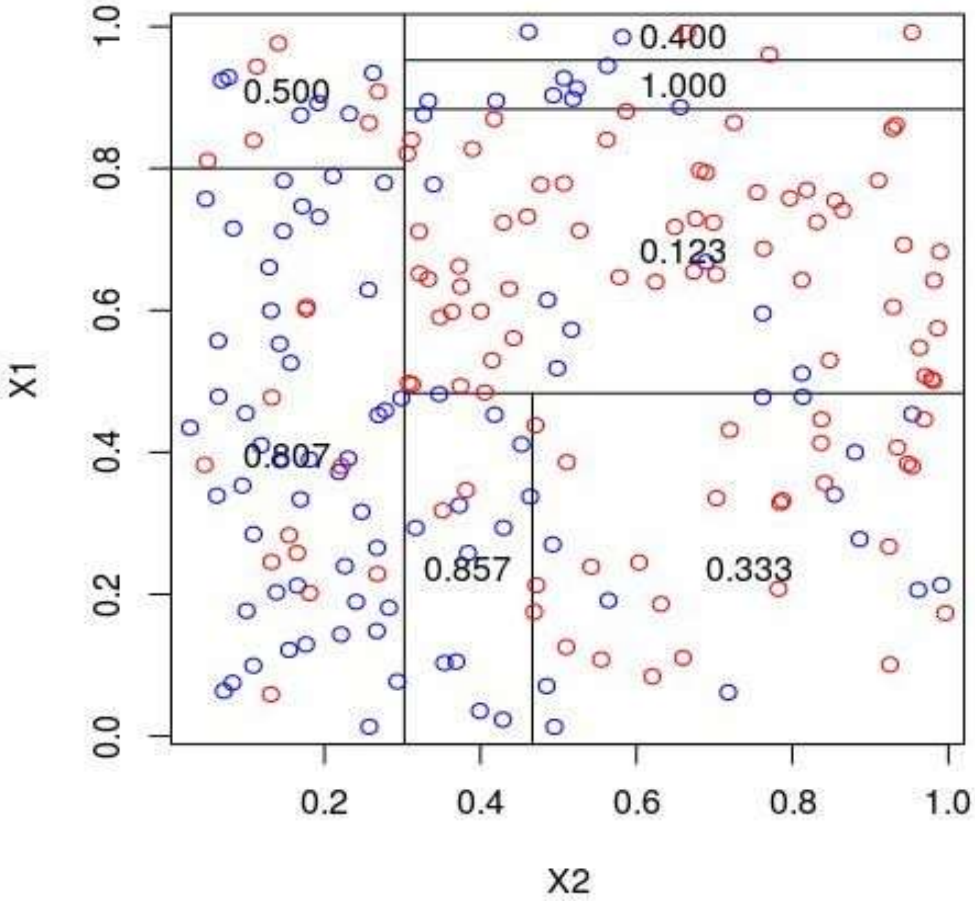
Регрессия: Нелинейная регрессия

Важно не переусердствовать при добавлении нелинейных признаков



Регрессия: Деревья и ансамбли





Ранжирование

- Даны списки объектов и частичные порядки на этих списках (например, известно, какой элемент идёт за каким).

- Задача: построить ранжирующую модель, обобщающую способ ранжирования на новые данные

Классы моделей в ML

- **Обучение с учителем** (supervised learning)
  - Классификация
  - Регрессия
  - Ранжирование
- **Обучение без учителя** (unsupervised learning)
  - Кластеризация
  - Уменьшение размерности
- **Обучение с частичным привлечением учителя**
  - (semi-supervised learning)
- **Обучение с подкреплением** (reinforcement learning)

Обучение без учителя

Типичные задачи:

- Кластеризация
  - Объединить элементы в группы по их похожести
  - Пример: группировка клеток по профилю экспрессии; домохозяйств по паттерну потребления электроэнергии; пользователей сервиса по поведению; кластеризация клиентов на группы.
- Уменьшение размерности
  - Перевести данные в пространство меньшей размерности, с сохранением отношений между элементами

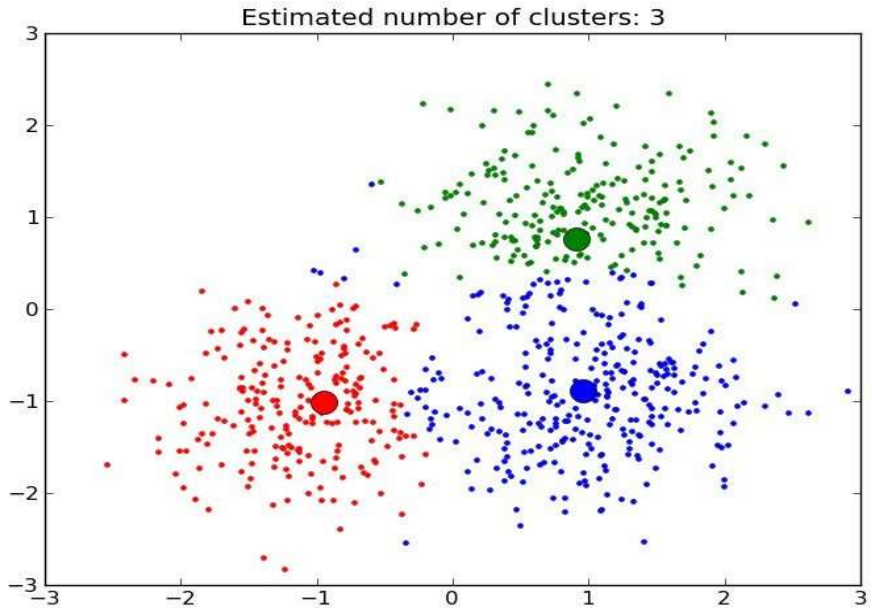
Пример: визуализация многомерных данных

Кластеризация

• Каждый объект представлен в виде многомерного вектора $<x_1, x_2, \ldots, x_n>$

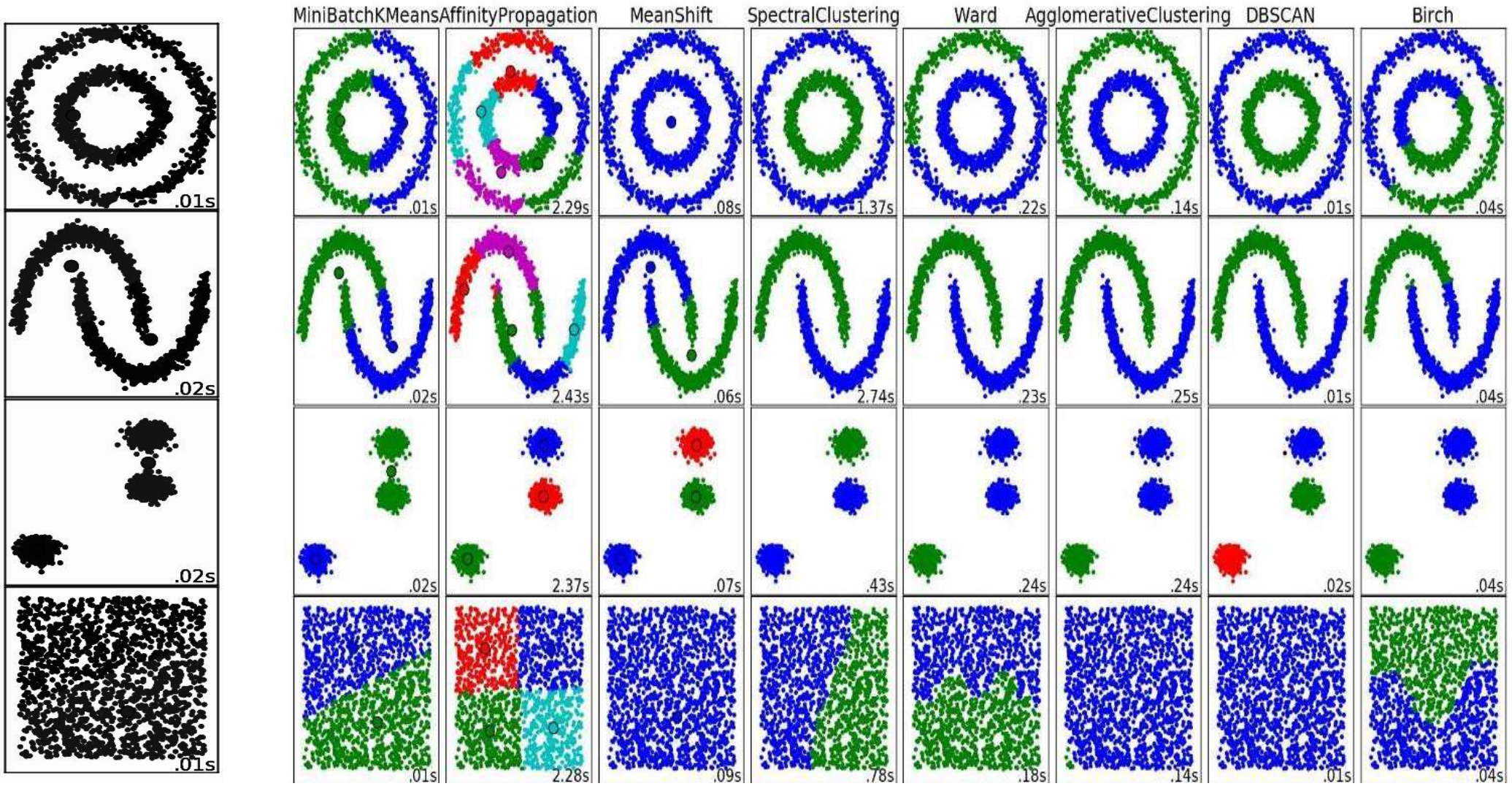
Задача: сгруппировать объекты по “похожести” так, чтобы:

- Объекты внутри одной группы были более похожи друг на друга, чем на объекты из любой другой группы
- Количество групп может как задаваться изначально, так и определяться по ходу дела
- Под группировкой имеется в виду приписать каждому объекту номер кластера



Кластеризация: Важные моменты

- Обучение без учителя (нет размеченной выборки)
- Похожесть или близость объектов обычно определяется через расстояние в многомерном пространстве признаков
- Надо определить само пространство (что учитываем) и метрику близости (как именно считаем близость)
- Может быть полезно для исследования датасета и получения интуиции о его структуре
- Результат трудно визуализировать, если число измерений больше четырёх
- Результат можно использовать для других методов (например, классификации и регрессии) как новый признак


MiniBatchKMeans
AffinityPropagation
MeanShift
SpectralClustering
Ward
AgglomerativeClustering
DBSCAN
Birch
.01s
2.29s
.08s
1.37s
.22s
.14s
.01s
.04s
.02s
2.43s
.06s
2.74s
.23s
.25s
.01s
.04s
.02s
2.37s
.07s
.43s
.24s
.24s
.02s
.04s
.01s
2.28s
.09s
.78s
.18s
.14s
.01s
.04s

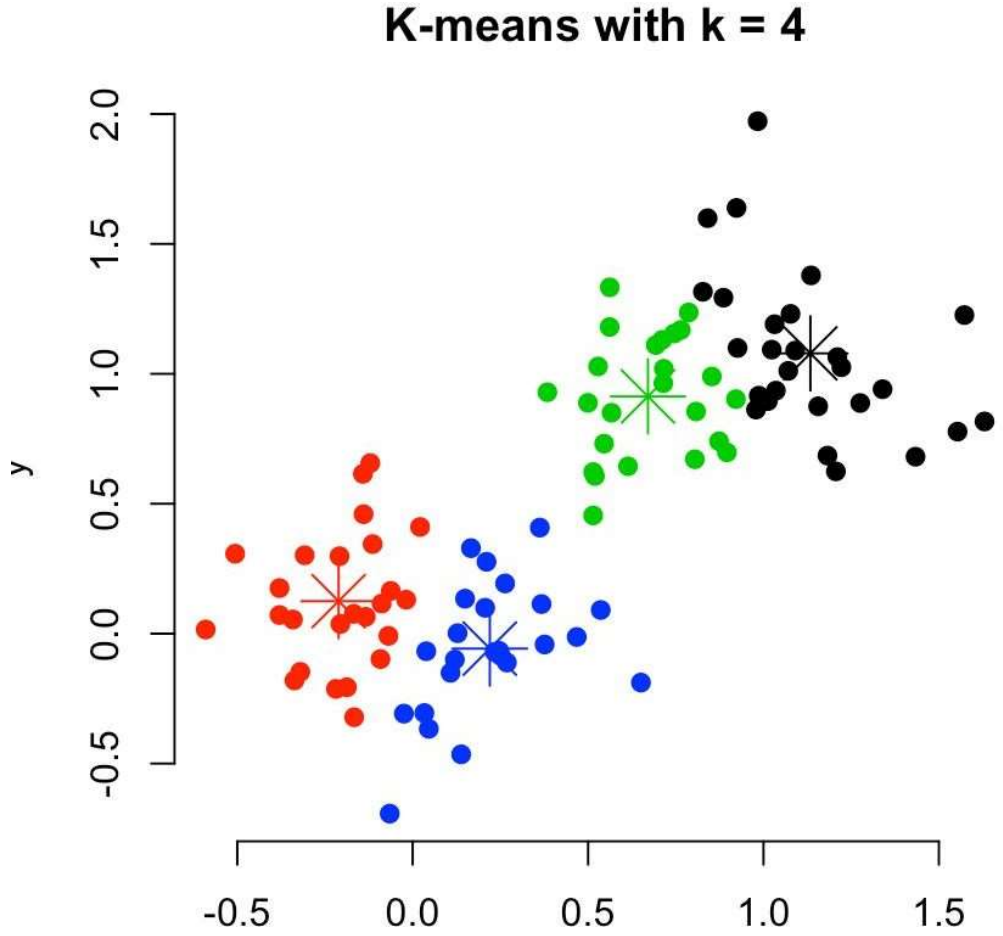
Кластеризация: Методы

- K-means (K-means++, K-means||)
- Иерархическая кластеризация
- Кластеризация на основе распределений (EM-алгоритм)
- Кластеризация на основе плотности (DBSCAN)
- Графовая кластеризация (определение клик)
- ...

Кластеризация: K-means

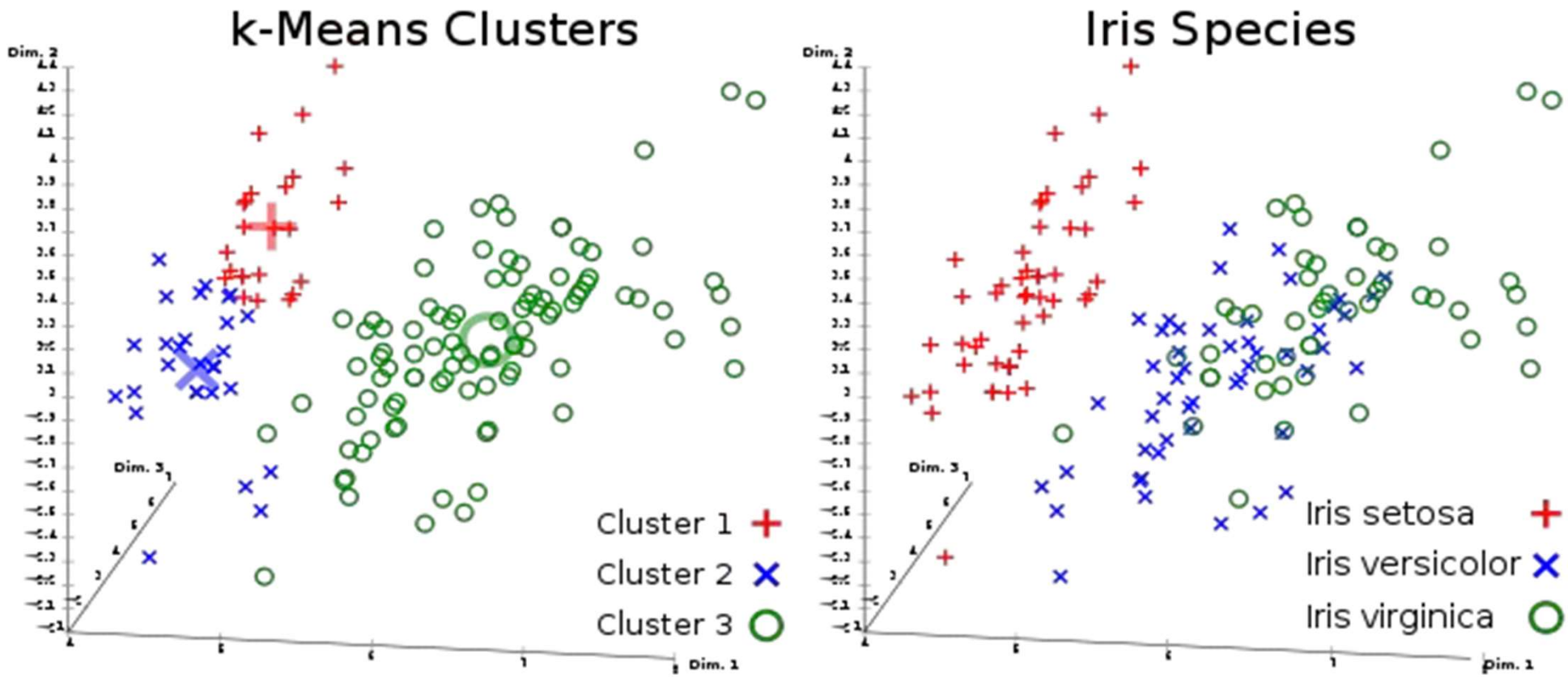
Простой и лёгкий для вычисления алгоритм. Количество кластеров (k) задаётся изначально.

Мы можем не знать заранее, сколько кластеров есть в данных, но можно Перебрать несколько вариантов и выбрать лучший (по какому-либо критерию, в том числе и автоматически)



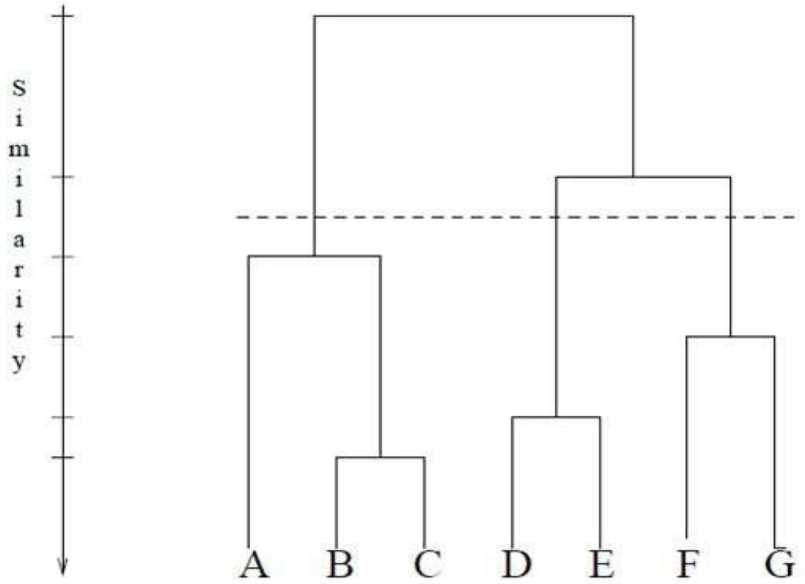
Кластеризация: K-means

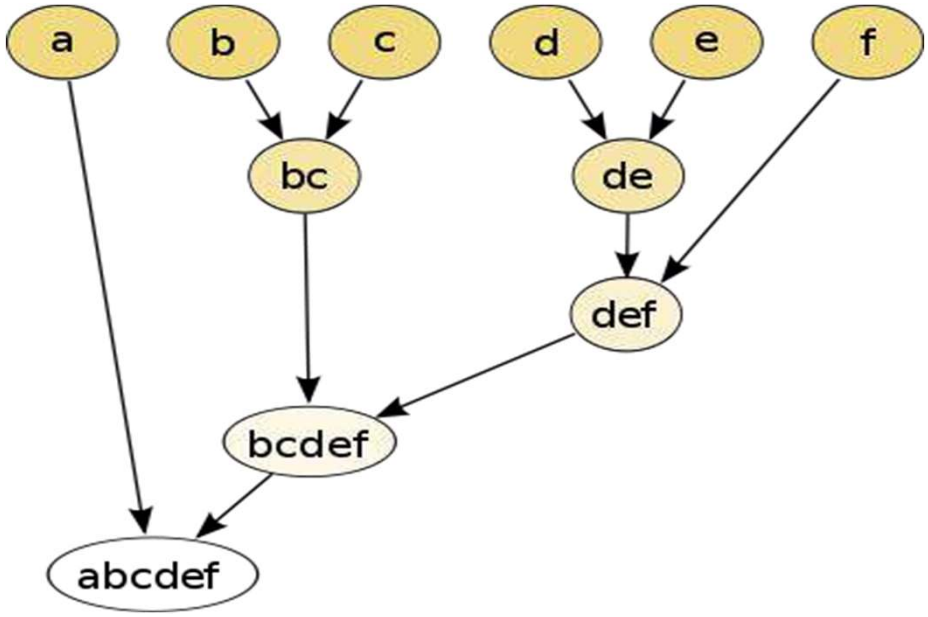
Для кластеров сложной формы может давать плохой результат



Иерархическая кластеризация

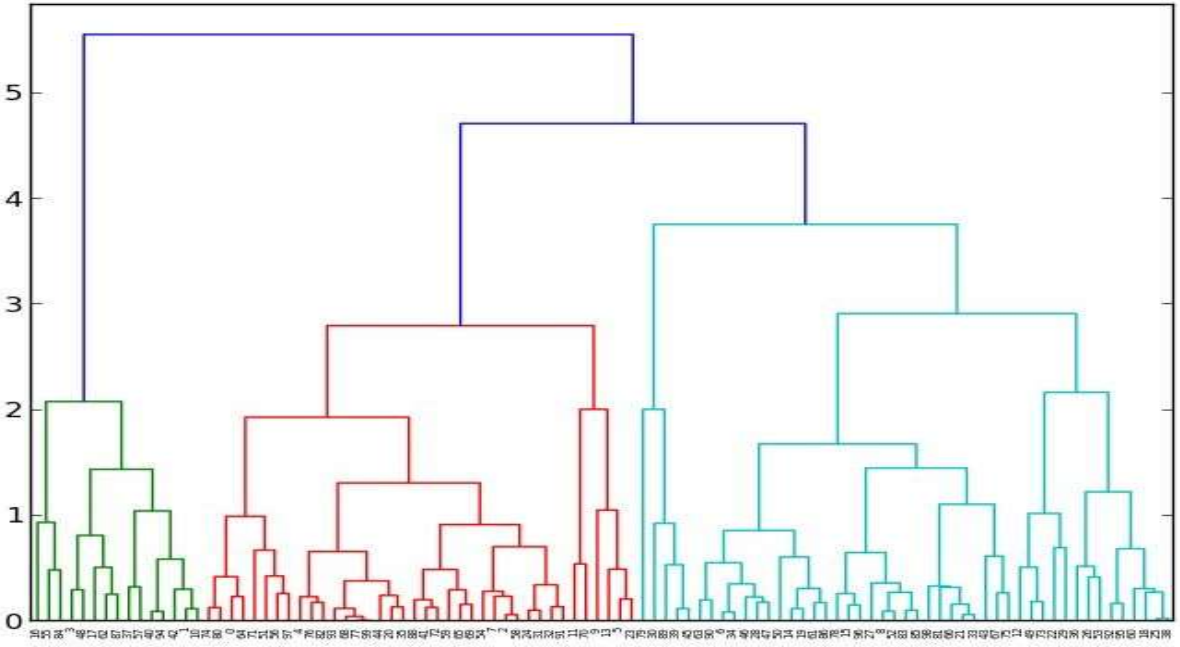
- Алгоритм простой, но вычислительно сложный (не используется с Big Data)
- Использует идею последовательного объединения самых похожих объектов

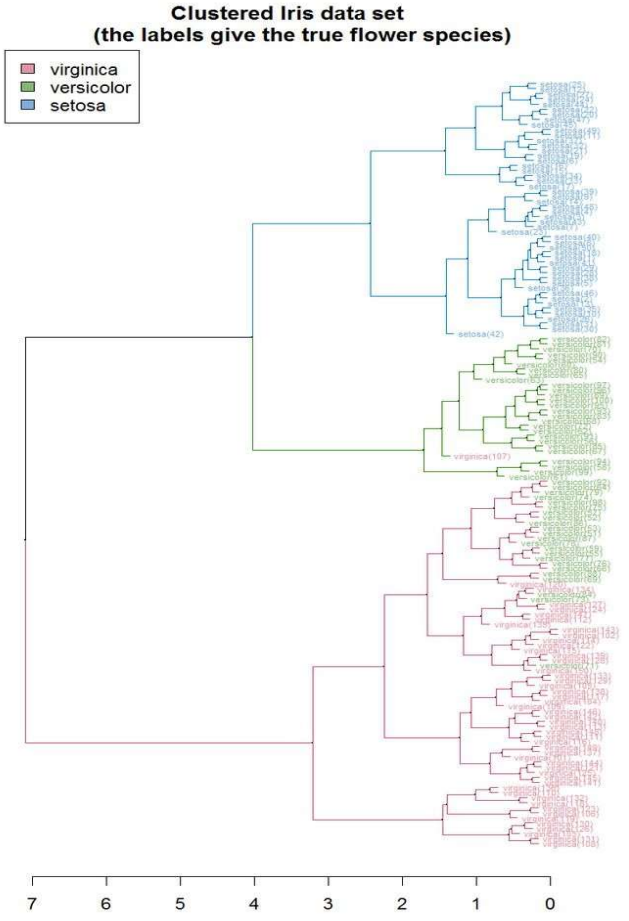




Иерархическая кластеризация

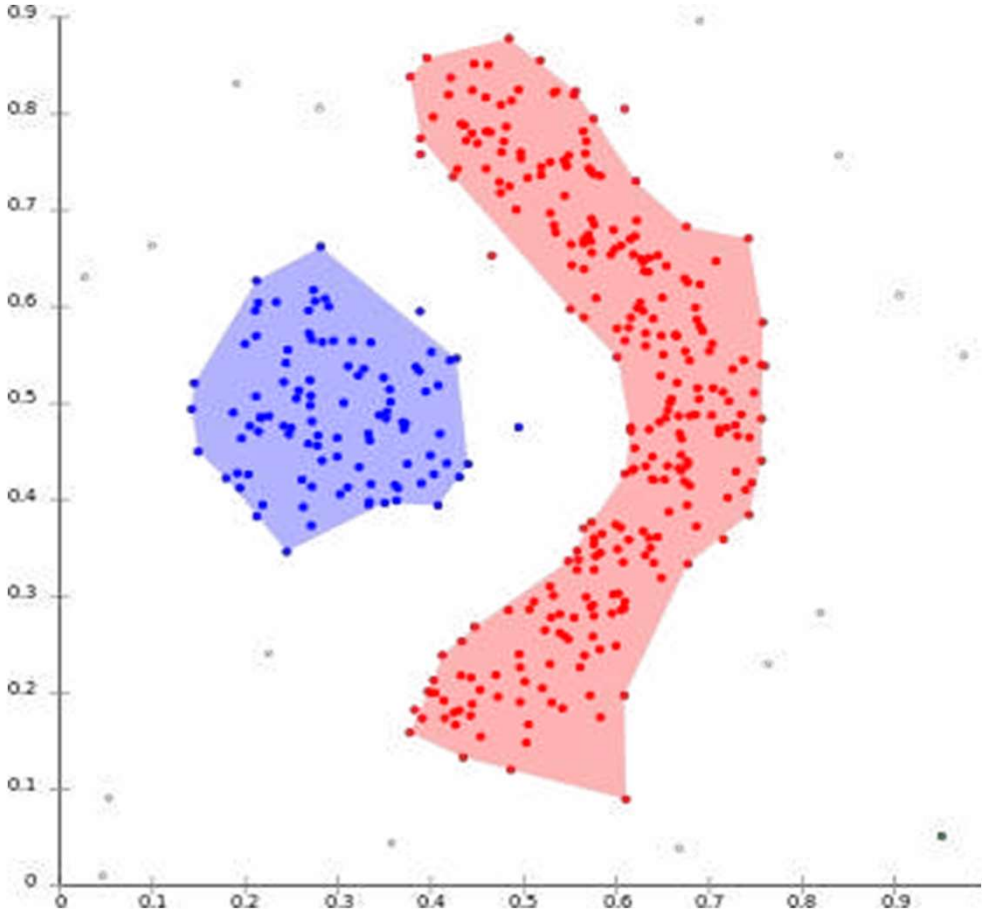
- Результатом является дендрограмма
- Проблемы с отображением при большом числе объектов





Кластеризация: DBSCAN

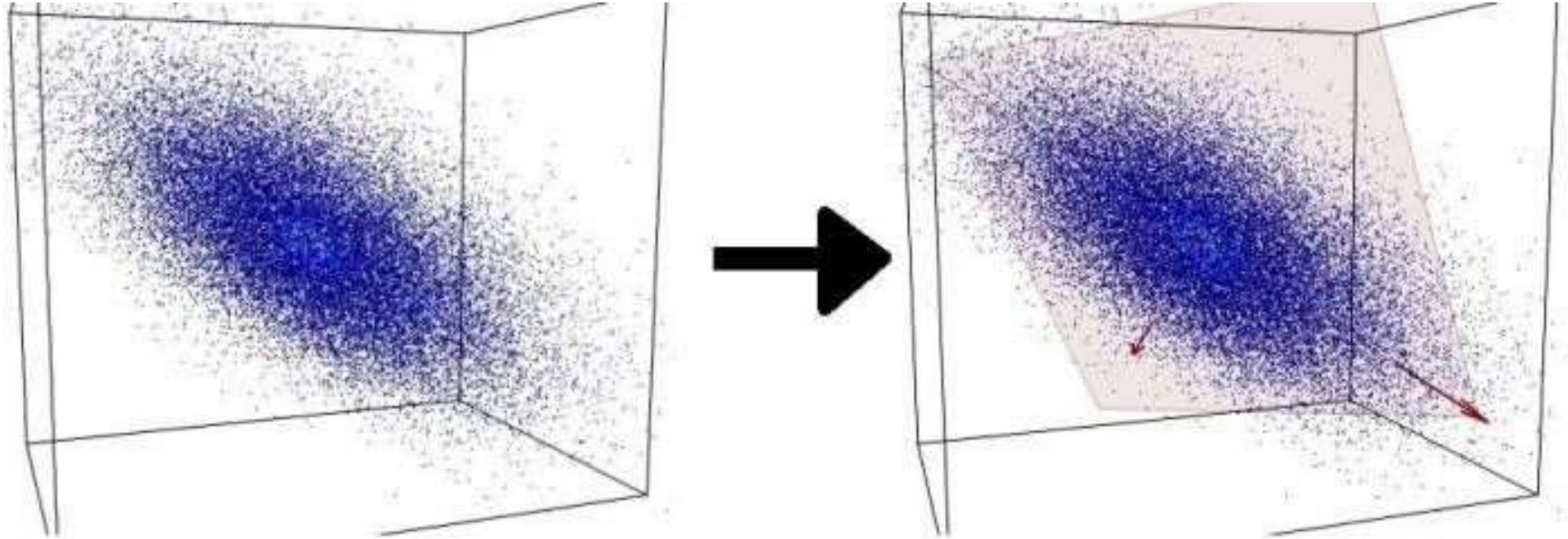
- Плотностный алгоритм: группирует вместе “плотные” участки
- Может находить кластеры очень сложной формы
- Часть точек не относятся никуда и считаются шумом
- Проблемы при кластеризации данных с сильно различающейся плотностью.
- Может быть трудно выбрать параметры для новых неизученных датасетов (а уже изученные кластеризовать неинтересно)



Уменьшение размерности

Каждый объект представлен в виде многомерного вектора <x1,x2,...,xn>

Задача: получить более компактное признаковое описание объекта <x1,x2,...,xk> где k<n



Уменьшение размерности: Важные моменты

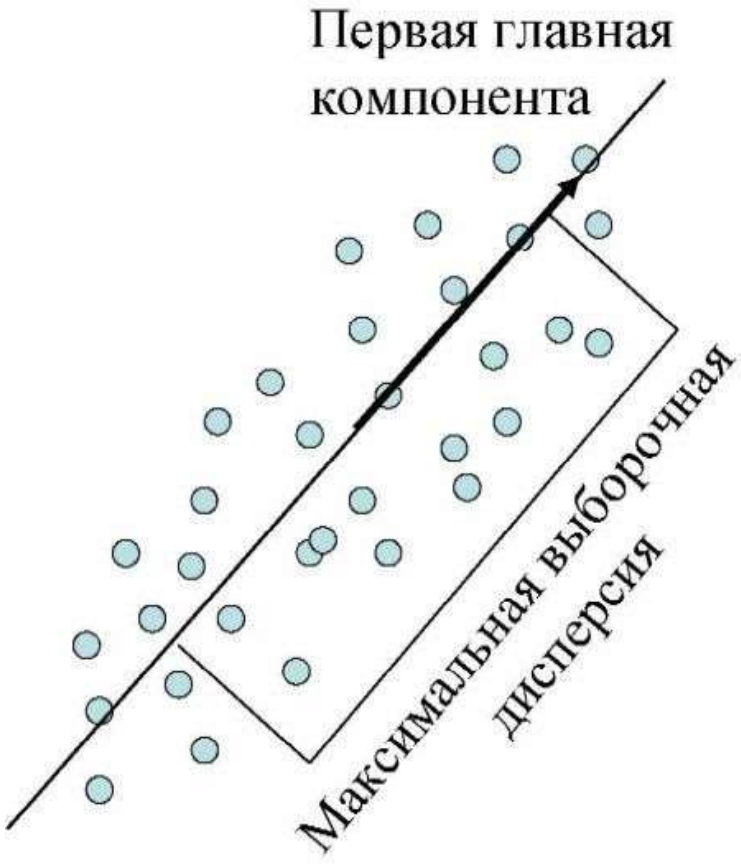
- Обучение без учителя (нет размеченной выборки)
- Помогает устранять “проклятие размерности”: данные быстро становятся разреженными при увеличении размерности
- Удобно использовать для визуализации, когда многомерный набор данных сводится к 2-3 измерениям, на которых могут быть видны закономерности
- Удобно использовать для разведочного анализа
- Уменьшает размер датасета, требуемое для хранения место и время для обработки
- Может помочь другим методам машинного обучения за счёт устранения избыточных данных (например, сильно скореллированных)

Уменьшение размерности: методы

- Метод главных компонент (principal component analysis, PCA)
- Метод независимых компонент (ICA)
- Многомерное шкалирование (Multidimensional scaling, MDS)
- Автоэнкодеры
- t-distributed stochastic neighbor embedding (t-SNE)
- ...

Уменьшение размерности: PCA

- Преобразует данные (возможно, коррелированные) в набор линейно независимых компонент (главных компонент).
- Первая компонента имеет максимальную дисперсию, вторая следующую по величине и т.д.
- Фактически находим базис в пространстве меньшей размерности.
- Очень широко применяемый метод.



Обучение с частичным привлечением учителя

Вариант обучения с учителем (supervised learning), в котором также используются неразмеченные данные.

Например,

- Есть небольшой размеченный датасет (большой датасет д**о**рог или его не всегда можно получить).
- Есть большой неразмеченный датасет (неразмеченных данных много).
- По неразмеченным данным модель “учит” структуру и закономерности.
- По размеченным данным на уже выделенной структуре учим модель с использованием разметки.

Обучение с подкреплением

Постановка задачи:

- Есть среда, в которой действует агент. У среды есть состояние.
- Агент может совершать действия.
- Действия приносят некий результат (reward) (не обязательно мгновенно)
- Надо научиться такой модели поведения, которая максимизирует результат

Типичные задачи:

- Управление роботом
- Оптимизация в играх

Метрики качества

- Метрики качества регрессии

Mean squared error	$\mathrm{MSE} = \frac{1}{n}\sum_{t=1}^{n} e_t^2$
Root mean squared error	$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{t=1}^{n} e_t^2}$
Mean absolute error	$\mathrm{MAE} = \frac{1}{n}\sum_{t=1}^{n} \lvert e_t \rvert$
Mean absolute percentage error	$\mathrm{MAPE} = \frac{100\%}{n}\sum_{t=1}^{n} \left\lvert \frac{e_t}{y_t} \right\rvert$

Метрики качества классификации

Идея: насколько хорошо мы угадываем классы. Важно, что цена ошибки может быть разной для разных случаев.

При классификации на два класса (по сути да/нет) у нас есть 4

различных исхода:

- True Positive (TP) -- истинное значение было “да” и мы предсказали “да”
- True Negative (TN) -- истинное значение было “нет” и мы предсказали “нет”
- False Positive (FP) -- истинное значение было “нет”, а мы предсказали “да”. Ложное срабатывание, ошибка I рода.

• False Negatie (FN) -- истинное значение было “да”, а мы предсказали “нет”.

Пропуск цели, ошибка II рода.

		Предсказанный	
		true	**false**
Действительный	**true**	TP	FN
	false	FP	TN

Метрики качества классификации

- **Accuracy** (аккуратность) – процент верных предсказаний
- **Precision** (точность) – сколько верных среди предсказанных как “Класс 1/да”
- **Recall** (полнота) – сколько из настоящих “Класс 1/да” мы определили верно (то же самое, что True Positive Rate, Sensitivity)
- **F-мера** (гармоническое среднее P и R)
- **Log-loss** (учитывает значения вероятности)

...

		Предсказанный	
		true	**false**
Действительный	**true**	TP	FN
	false	FP	TN

$$precision(p.) = \frac{TP}{TP + FP}$$

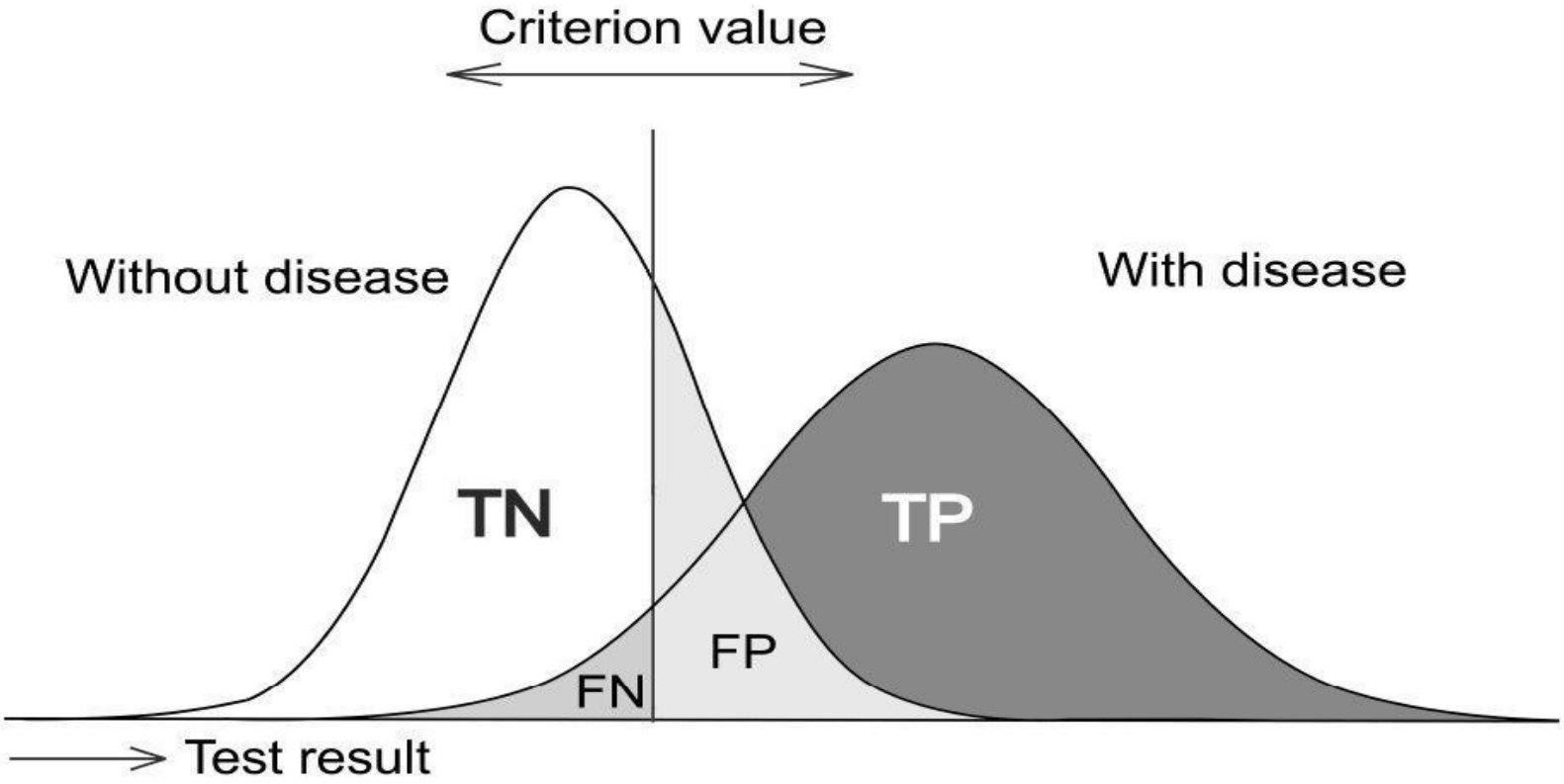
$$recall(r.) = \frac{TP}{TP + FN}$$

$$accuracy(acc.) = \frac{TP + TN}{TP + TN + FP + FN}$$

$$F1 = \frac{2 \times precision \times recall}{precision + recall}$$

Метрики качества классификации

В реальности у классификатора часто есть порог, который можно регулировать (тем самым играя на разнице в цене ошибок I и II рода)



ROC-кривая

- Поскольку у классификатора есть порог, мы можем получить общую картину для всех значений порога.
- Зависимость количества **верно классифицированных положительных примеров** (чувствительность, sensitivity, true positive rate, TPR, hit rate, recall) от количества **неверно классифицированных отрицательных примеров** (1-специфичность, fall-out, false positive rate, FPR).

$$TPR = TP/P = TP/(TP + FN)$$

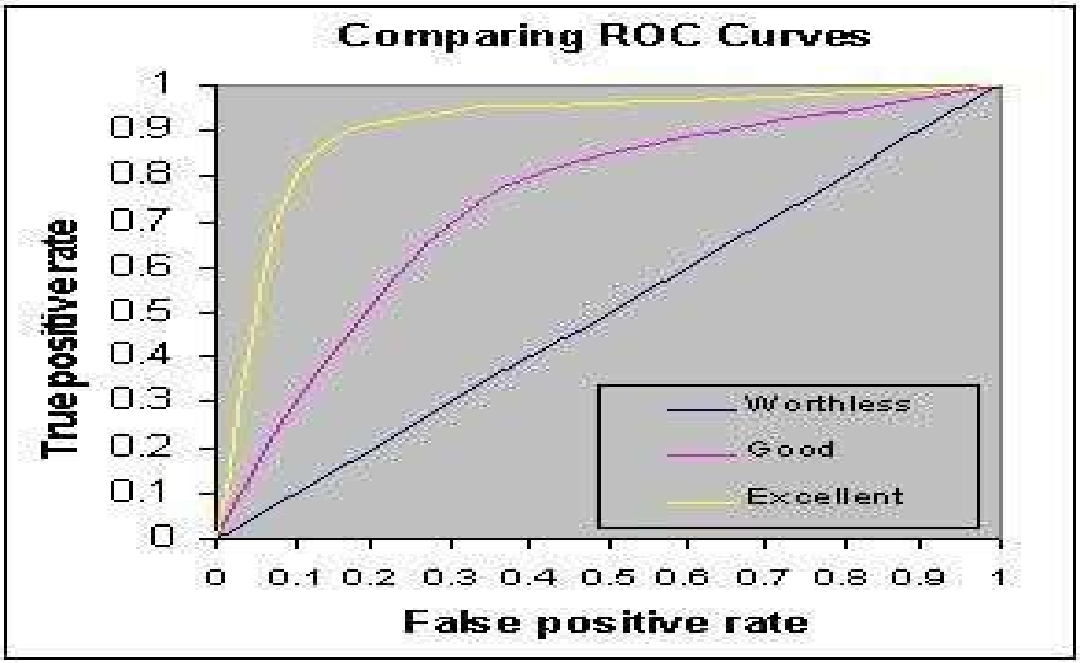
$$FPR = FP/N = FP/(FP + TN) = 1 - SPC$$

		Предсказанный	
		true	**false**
Действительный	**true**	TP	FN
	false	FP	TN

ROC-кривая

- Зависимость количества **верно классифицированных положительных примеров** от количества **неверно классифицированных отрицательных примеров**.
- Разные кривые можно сравнивать по площади, которую они ограничивают (ROC AUC).
- 0.5 -- случайный классификатор

  1.0 -- идеальный классификатор



Метрики качества кластеризации

- Внутренние метрики:
  - средняя похожесть объектов внутри кластера: должна быть высокой)
  - среднее расстояние между кластерами: должно быть выше между кластерами, чем внутри одного кластера (расстояние обычно обратная мера к похожести)

Метрики качества кластеризации

- Со внутренними метриками проблема, так как их часто невозможно перевести в бизнес-метрики, поэтому используют внешние метрики, например, оценку по размеченной части датасета:
  - Можно считать метрики аналогичные метрикам для классификации, считая, что целевое значение -- это номер кластера
  - Можно придумать свою метрику на размеченном датасете
  - Это требует времени (ручной труд на разметку), но даёт гораздо более практический и понятный результат

Практические соображения по метрикам

1. Если возможно, старайтесь использовать одну и ту же метрику для обучения и оценки качества.Но это не всегда возможно, так как по некоторым метрикам нельзя или трудно напрямую оптимизироваться (AUC, NDCG)

2. Внимательно и осознанно выбирайте метрику под ваш датасет.
   - В случае сильно несбалансированных классов можно получить бесполезную метрику
   - Пример: Поиск мошеннических транзакций
     - Датасет: 1 млн примеров, из них мошеннических транзакций 1%
     - Метрика: Аккуратность (Accuracy)
     - Классификатор: всех относим к классу “Честные”

Получаем качество 99%

3. Учитываете, что может быть сильно разная цена ошибки.
   - Возможно, надо скорректировать метрику либо выбирать метрику исходя из этого знания.

4. Выбросы могут сильно исказить метрику (регрессия)

   - Используйте устойчивую метрику (например, медиана устойчивее арифметического среднего) или фильтруйте выбросы.

Обучение модели

Разбиение датасета для обучения и оценки.

- Проблема: если оценивать качество на той же выборке, на которой алгоритм обучался, оценка будет слишком оптимистичная и не соответствовать реальному качеству алгоритма.

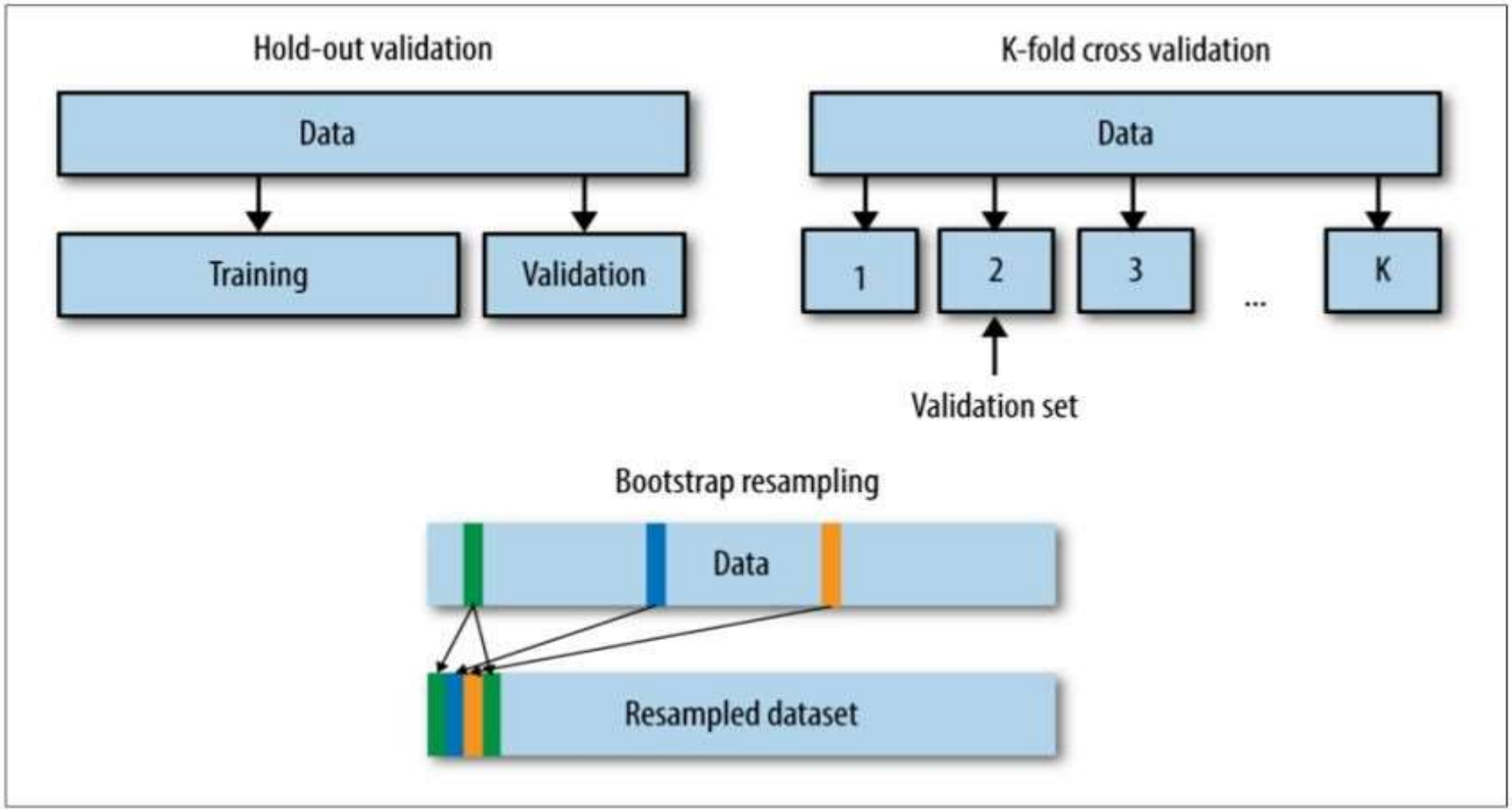
- В предельном случае алгоритм “переобучается” и начинает отлично предсказывать примеры из своей выборки. Он их может “выучить” или запомнить (тогда новые примеры он может совсем не угадать), либо же находит закономерности в шуме, который всегда есть в данных (а шум на другой выборке примет иные значения, так что предсказание алгоритма будет неверным).

- Метод борьбы с этой проблемой: использовать

отдельные наборы данных для обучения и оценки.

- Скрытые данные (Hold-out validation):
  - train/validation/test датасеты
  - Обучаем модель на train
  - Выбираем модель по validation
  - Финальная проверка на test (используется только один раз для получения неоптимистичной оценки качества)
- Кросс-валидация (k-fold cross-validation)
  - Разбиваем данные на k кусочков
  - Обучаемся на (k-1) кусочке, проверяемся на 1
  - Результат усредняется
- Bootstrap resampling
  - Генерируем новый датасет, делая выборку из оригинального

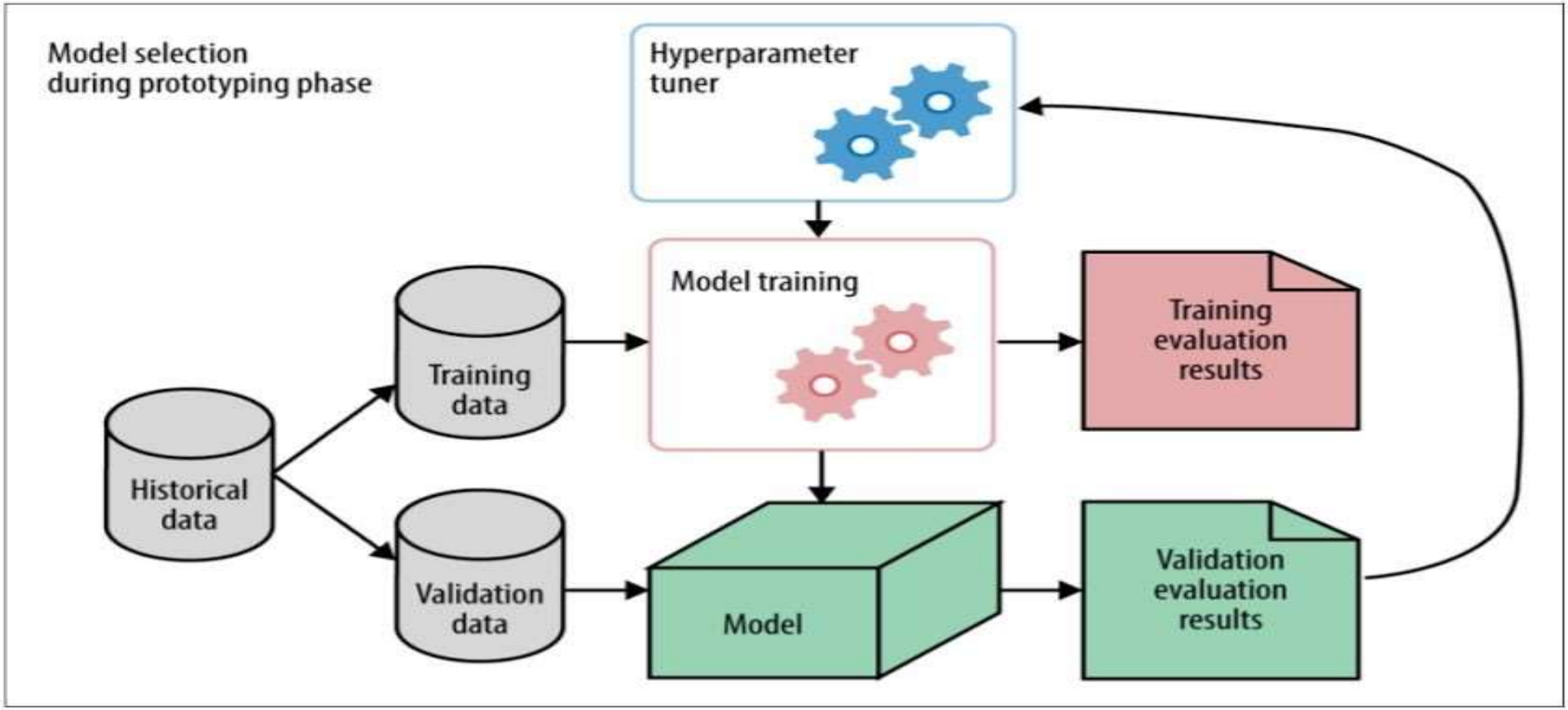
Механизмы валидации

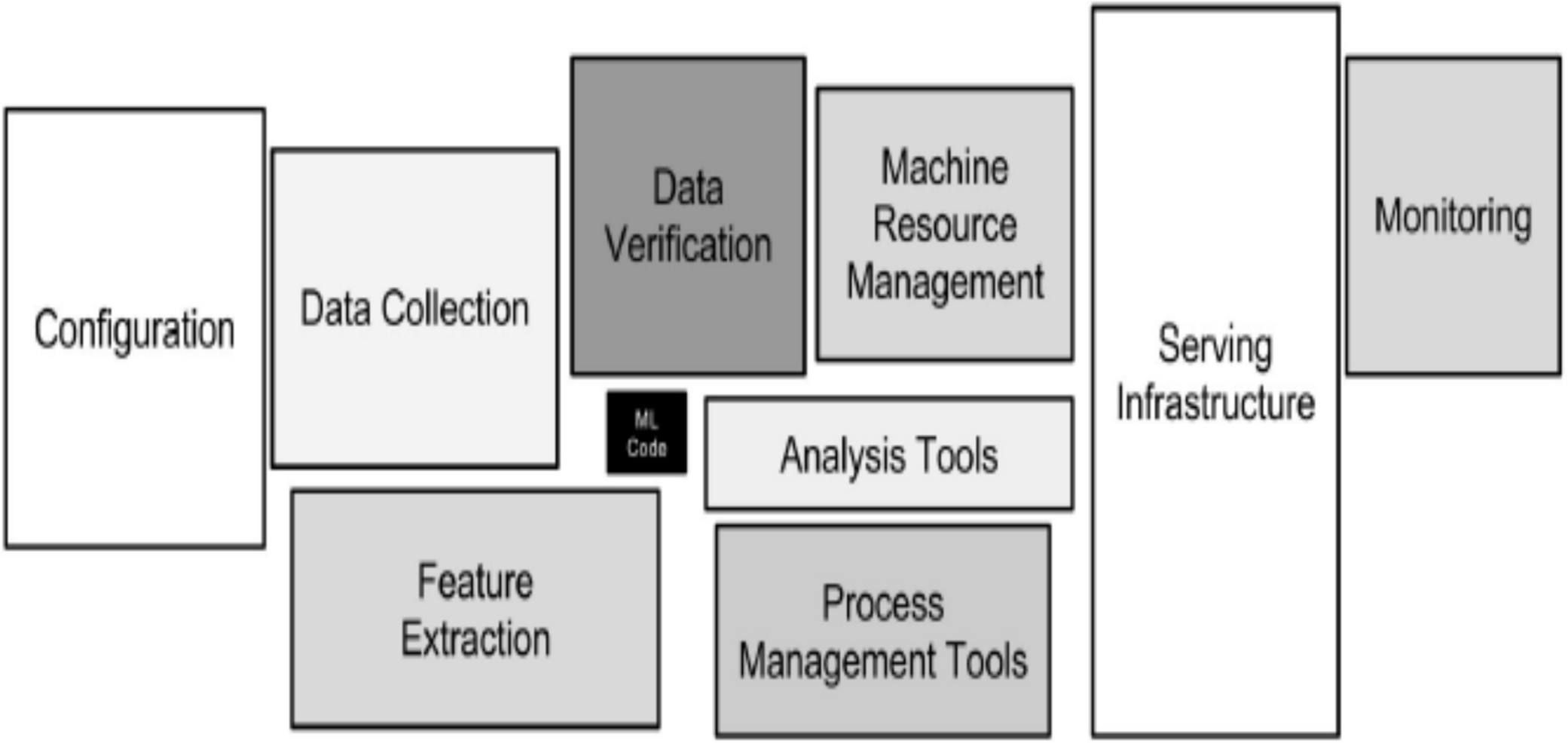


Обучение модели

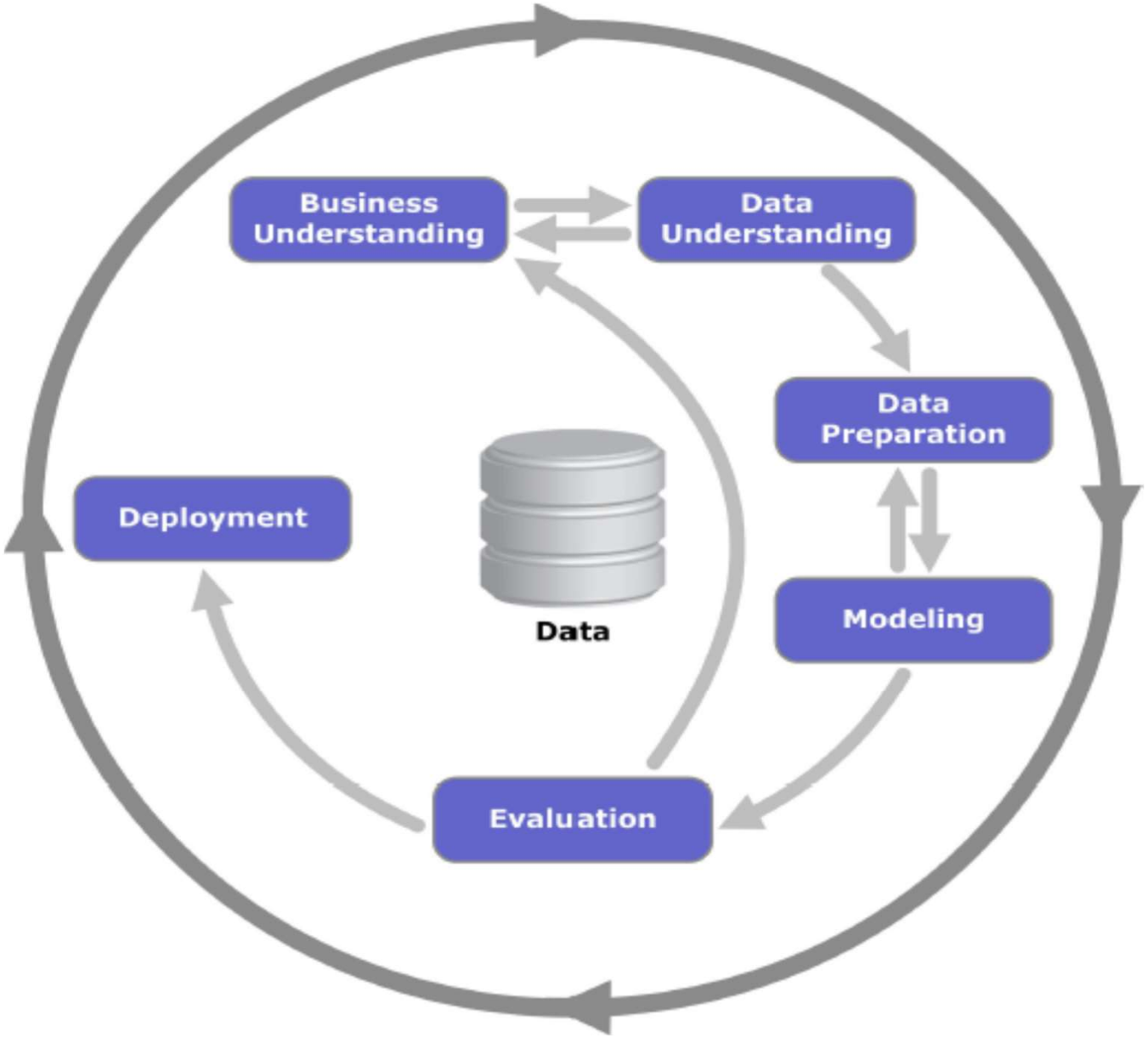
- Модель обучается на тренировочных данных (train set)
  - Обучение может означать подстройку каких-то численных параметров модели или (реже) изменение структуры модели.
  - В результате мы получаем параметры модели, которые позволяют ей (мы надеемся) эффективно выполнять свою функцию.
- Редко бывает только одна модель, обычно требуется пробовать разные варианты одной модели или модели различных классов
  - Для оценки качества и выбора лучшей модели мы используем валидационную выборку (validation set), на которой модель не обучалась
- В конце работы мы оцениваем модель на тестовой выборке
  - (test set)
  - Это нужно для получения не слишком оптимистичной оценки качества модели
- В идеале тестовую выборку нужно использовать только один раз!

Процесс создания модели ML




Configuration
Data Collection
Data Verification
Machine Resource Management
Monitoring
Serving Infrastructure
ML Code
Analysis Tools
Feature Extraction
Process Management Tools

CRISP-DM


Business Understanding
Data Understanding
Data Preparation
Modeling
Evaluation
Deployment
Data

Data Science Lifecycle