Классические методы машинного обучения (базовый курс)

к.ф.-м.н. Михаил Игоревич Петровский
michael@cs.msu.ru

О лекторе

- доцент и ученый секретарь кафедры Интеллектуальных информационных технологий ВМК МГУ, автор и лектор курсов по машинному обучению, прикладной статистике и интеллектуальному анализу данных на ВМК МГУ
- руководитель магистерской программы «Интеллектуальный анализ больших данных» (кафедры Математической статистики и Интеллектуальных информационных технологий)
- более 25 лет опыта участия и руководства научно-исследовательскими и прикладными ИТ проектами в области анализа данных и машинного обучения
- Профессиональные сертификаты и награды:
  - Медаль РАН за лучшую работы в области информатики (по теме кандидатской диссертации)
  - Microsoft Certified Solution Developer (специализации desktop и distributed C++ разработка, SQL сервер)
  - PMI Certified Project Management Professional
  - SAS Certified Data Scientist, Big Data & Advanced Analytics Professional
- 2012-2022г. руководитель академической программы SAS в РФ/СНГ

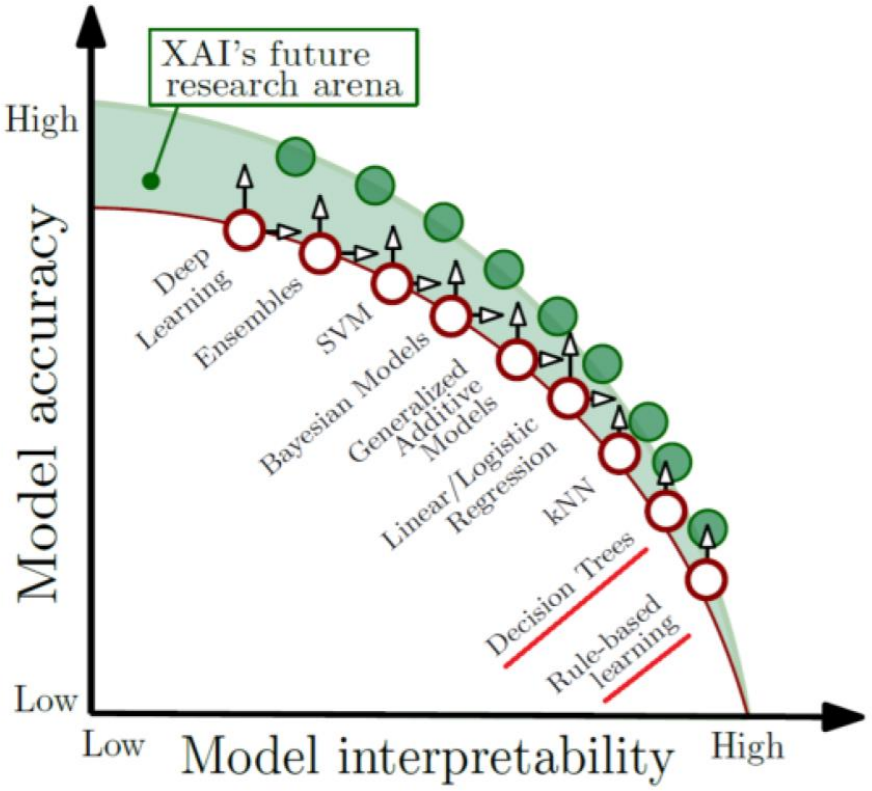
Аннотация курса

- Цели курса:
  - дать основные определения и терминологию (ИИ, ML, DS, Big data)
  - познакомить с базовыми методами построения обобщённых линейных моделей, деревьев решений и их ансамблей
  - выполнить практические задания из области анализа табличных данных в сфере клиентской аналитики (анализ рисков)
- Содержание курса:
  1. Введение
  2. Обобщенные линейные модели (???)
  3. Деревья решений
  4. Ансамбли
  5. Бустинг

Не надо быть «Карлсоном»!

Почему классические методы важны?

- Более «управляемые» и интерпретируемые модели
- Вычислительно менее требовательные
- Для многих задач не менее точные модели
- Но требуют более глубокого понимания математики



Введение

Немного истории и терминологии

Интуитивное определение ИИ

Искусственный интеллект – проблема определения термина

- **Нет общепризнанного** научного определения
- Сильный коммерческий **«хайп»**, смещающий акценты
- Часто термин ИИ **неправильно используется** в очень узком смысле, как машинное обучение, или даже нейросети, или даже глубокое обучение нейросетей
- Надо делать акцент на слово **«искусственный»**

Пример определения:

- «ИИ - междисциплинарная **область знаний**, занимающаяся исследованием и разработкой методов и артефактов (**устройств или программ**), которые способны **имитировать интеллектуальную** (разумную/рациональную) **деятельность** (мышление/принятие решение) **человека**»

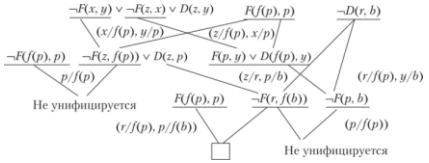
Почему «думать» и «делать» это разные области в ИИ?

«Думать» («мыслить») – оперировать знаниями

- Есть формальное **представление знаний** и интеллектуальная система, способная на их основе **генерировать** новые непротиворечивые знания или **проверять** утверждения, в том числе в условиях неопределенности
- Примеры задач ИИ из категории «думать»

  «рационально» – автоматическое доказательство теорем

  «как человек» – распознавание эмоций по фото или видео

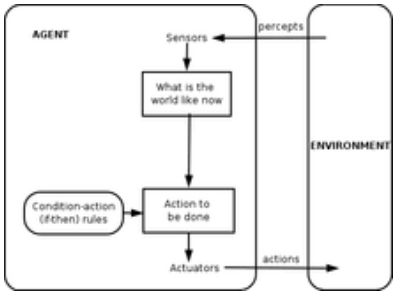


«Действовать» - взаимодействовать с окружающей средой (**интеллектуальный агент)**

- производит **действия**, получает отклик **среды**,
- самокорректируется (**учится**) с определенной **целью**
- Примеры задач ИИ из категории «действовать»

  «рационально» - беспилотный автомобиль

  «как человек» - чат-бот, голосовой помощник, игровой ИИ



Почему человек нерационален и плохо ли это?

Что значит «рационально»?

- Достижение заданной цели эффективным (а лучше оптимальным) непротиворечивым путем
- По сути – **задача оптимизации** (даже там, где это неочевидно, например, системы автоматических рассуждений не используют полный перебор вариантов)

Причины **нерациональности** человека:

- Недостаток информации
- Огромное пространство перебора при поиске решений (шахматы)
- Невозможность задать целевую функцию (помогает теория полезности)
- Биологические особенности работы мозга человека

Механизмы принятия решений человеком (все моделируются в ИИ):

- **Рефлексные** (не используют мозг, например, отдернуть обожженную руку)
- **Интуитивные/эмоциональные/спонтанные** (используют лимбическую систему, поощряются гормонально, приносят удовольствие) – «золотая жила» для ИИ (Эмоциональная экономика)
- **Рациональные** (работает неокортекс, ничего приятного, сильно устаешь, никто не любит думать)

Всегда ли **нерационально** значит **плохо? Нет!** (история про джинна, проблема «здравого смысла»)

Искусственный Интеллект

Общий ИИ (**AGI**)

- Философские и этические вопросы ИИ
- Футуристика
- Исследования принципов работы биологического интеллекта
- Вопросы создания универсального автономного интеллектуального агента («скайнеты» и прочие «матрицы»)

Большинство ученых считает, что в обозримом будущем в этой области **прогресс маловероятен:**

- нет работающих теорий, инструментов
- проблема «общечеловеческого бэкграунда» или «здравого смысла» - ограниченность знаний любой интеллектуальной системы
- **Но** есть надежда на **Big Data**!

ИИ в узком смысле (**ANI**)

Не интересуется общими вопросами, а изучает и развивает инструменты и приложения ИИ:

- Автоматические рассуждения
- **Машинное обучение (сейчас ключевой инструмент)**
- Поиск и оптимизация
- Человеко-машинное взаимодействие

«Дополненный» интеллект: **не AI, а IA** (Intelligence amplification) – не замена, а усиление

Бурное **развитие приложений** и алгоритмов из-за развития вычислительной техники

Но по сути застой в теории –последние фундаментальные результаты **20+ лет назад** (пожалуй кроме трансформеров)

Рождение ИИ и ранние успехи (1950е-1970е)

(1950) Краеугольная работа Тьюринга **«Computing Machinery and Intelligence»**:

- Тест Тьюринга, принципы машинного обучения, генетические и другие поисковые алгоритмы, обучение с подкреплением

(1956) **Дартмутский семинар** (2 месяца, 10 человек), итоги – «развод» с кибернетикой и теорией управления:

1. ИИ не математика, а информатика (без компьютера нельзя)
2. ИИ моделирует и изучает поведение и мышление человека (в том числе нерациональное)

Череда **успехов**:

- Изначальный список Тьюринга «машина никогда не сможет …» быстро сокращался
- Разработаны «универсальные» решатели (General Problem Solver, Prolog и др.)
- Разработан LISP, показал возможности символьного решения задач (в том числе математических)
- Усовершенствование методов обучения нейросетей (обратное распространение ошибки), персептрон Розенблата и теорема о его сходимости
- Прикладные успехи: экспертные системы в медицине, управлении и инженерии на основе сложных моделей представления знаний (типа фреймов), машинный перевод и распознавание образов

Зима ИИ (с 1960х до 80х)

«Зима ИИ» - сокращение финансирования и интереса общества, отток специалистов, коммерческий и научный провал многих проектов, оказалось, что многое **«без ИИ лучше и дешевле»** плюс **проблема здравого смысла** (common sense):

- Провал методов машинного перевода (с русского, кстати) и закрытие гос. финансирования, из-за проблемы **семантической неоднозначности**:

  «the spirit is willing, but the flesh is weak» $\xleftrightarrow[\text{на русский и обратно}]{}$ «the vodka is good, but the meat is rotten»

- **комбинаторный взрыв -** проблемы сложности вычислений в системах логического вывода и автоматических рассуждений (в принципе решит, но лет через 100)
- Провал идеи **«эволюции программ»** – самопрограммирующиеся программы по принципу генетических алгоритмов
- Принципиальные **ограничения персептронов** (например, задача XOR для однослойного), книга Минского и Пейперта с критикой ⇒ смерть Френка Розенблата
- **Крах** рынка **LISP машин** – оказались хороши в науке, плохи в бизнес-приложениях
- Провал идеи «**компьютера 5 поколения»** – «интеллектуального компьютера», например, на прологе
- **Неэффективность экспертных систем** на основе фреймворков и семантических сетей: сложно описывать, долго настраивать, низкая точность, противоречивость

Причины краха больших надежд

Основная причина – **изоляционизм** специалистов по ИИ от остальных компьютерных наук:

- Изначальная уверенность, что символьные вычисления, логические методы и формальные грамматики есть основа разумной деятельности и они решат все проблемы
- Оказалось, что «умение решать» математические задачи школьного уровня или проходить тест на IQ не делает умнее не только человека, но и компьютер
- Сложные модели представления знаний (фреймворки и семантические сети) не принесли существенной пользы в реальных задачах, оказалось, что настроить систему ИИ дороже, чем решить прикладную задачу традиционными методами

Стало понятно, что в будущем будут востребованы **гибридные интеллектуальные системы**:

- сочетающие в себе несколько методов ИИ или классические математические методы и ИИ, например машинное обучение + оптимальное управление, или LLM + информационный поиск = RAG (retrieval augmented generation)

Оттепель ИИ (90е)

Многие **классические методы** успешно пережили «зиму», например:

- Экспертные системы в медицине, логистике, проектировании и других областях
- Интеллектуальное планирование и распределение ресурсов в задачах управления
- Системы нечеткого вывода в задачах управления механизмами (автоматические коробки передач)
- Обучение с подкреплением для обнаружения и разрешения конфликтов в воздушном движении
- Нейросети в задачах распознавания визуальных и звуковых образов
- Системы на основе поиска в пространстве состояний в компьютерных играх
- Робототехника

Рывок в методах **машинного обучения** и интеллектуального анализа данных:

- В 80х заново «переизобрели» все, что было в нейросетях 50х, включая разные формы Back Propagation
- Архитектуры Deep Learning (CNN, RNN, AE, LSTM, …) и методы их обучения (да, да, им более 20 лет)
- Бустинг слабых моделей и другие ансамбли
- Метод опорных векторов – «убийца нейросетей», который так и не смог их убить
- Скрытые Марковские модели и обучаемые сети Байеса

Бум ИИ и связь с ML и Data Science

Относительный застой в теории - ничего принципиально нового уже больше 20 лет

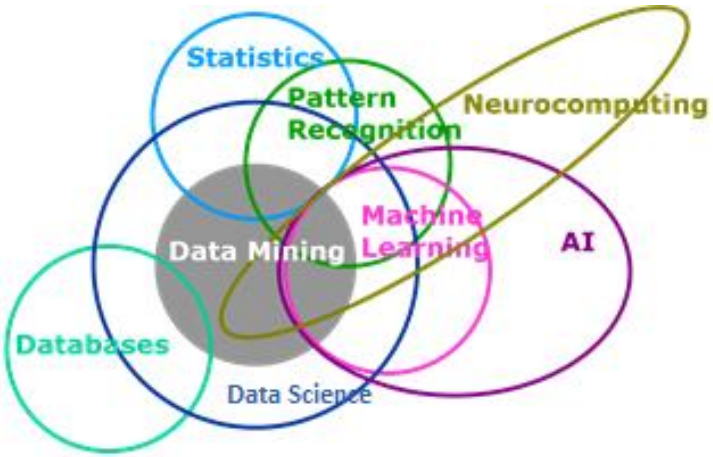
Пожалуй кроме **Трансформеров**

Прорыв в практике, почему? **Вычислительная техника** стала мощной и дешевой!

- Дешево накапливать и хранить большие объемы данных
- Можно просчитывать сложные модели за разумное время
- Математика подстраивается под вычислительную технику

В бизнес-сообществе часто термин ИИ используют (**неправильно!!!**) как синоним **Data Science** или **ML**

- **Машинное обучение** подраздел ИИ, изучающий методы построения алгоритмов, способных **обучаться на прецедентах** для решения задач: прогнозирования (классификации, ранжирования, регрессии), поиска скрытых структур в данных (ассоциаций, корреляций, кластеризации), обнаружения аномалий.
- **Data Science** (наука о данных) - раздел информатики, изучающий проблемы анализа, обработки (в том числе интеллектуальной) и представления данных в цифровой форме.
- Тесно связано с понятием **больших данных**.



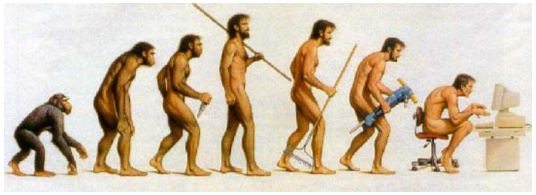
Большие данные

В научной среде термин используется с 1990х

(2008) «Как могут повлиять на будущее науки технологии, открывающие возможности работы с большими объёмами данных?», Клиффорд Линч (редактору журнала Nature)

(2011) «Big Data: The next frontier for innovation, competition and productivity», McKinsey Global Institute

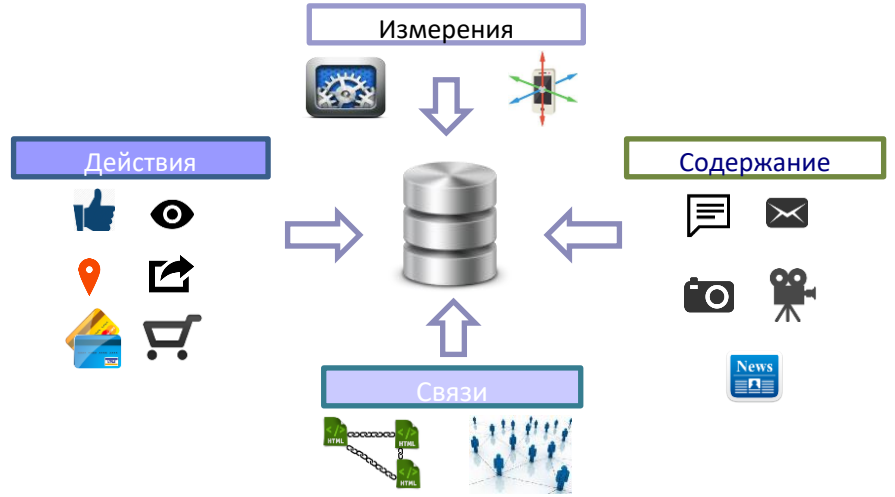
(2015) – термин Data Science



Начало цивилизации — 2003

5 экзабайт

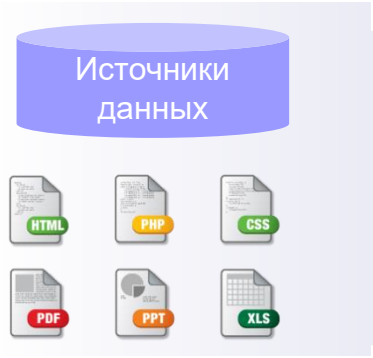
20+ экзабайт в сутки!



Volume (объем)	Velocity (скорость)	Variety (разнообразие)	Veracity (достоверность)
Большие объемы данных.	Быстрота поступления и обновления данных.	Разнообразные типы данных: текстовые, графы, медиа и другие	Низкое качество, недостоверность анализируемых данных.

Роль человека в аналитике больших данных

До эры больших данных:



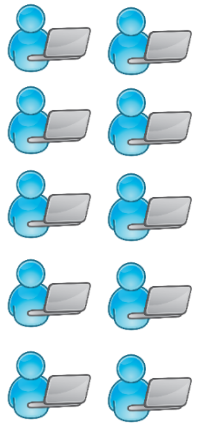
Корпоративное хранилище данных

Высококачественные надежные данные, последовательные, актуальные

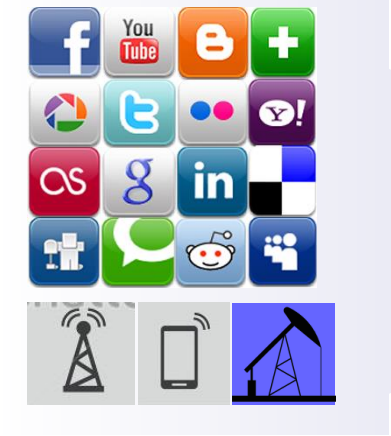
Витрина данных

Витрина аналитики

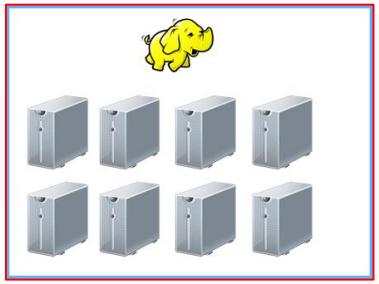
Традиционные аналитики



Сейчас:



Хранилище больших данных



Хранение “as is”

Data scientists:
Математики +
Программисты +
аналитики-прикладники

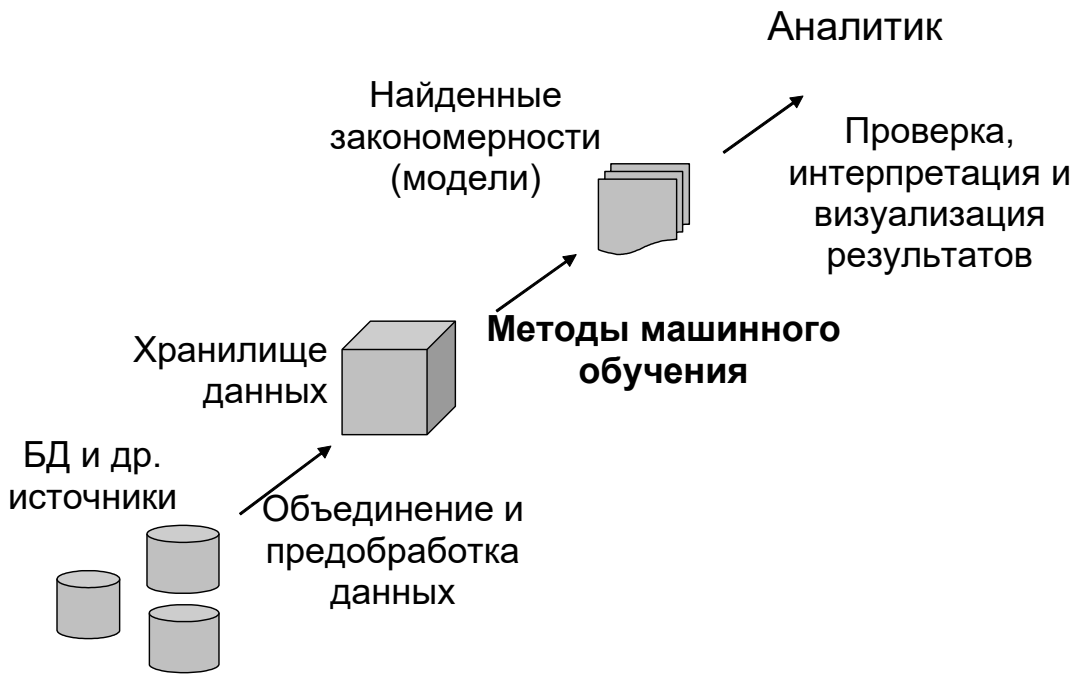
Успехи современного ИИ

Адаптируемый (с обучением) **ИИ + Большие данные + мощная вычислительная техника = заявка на AGI**

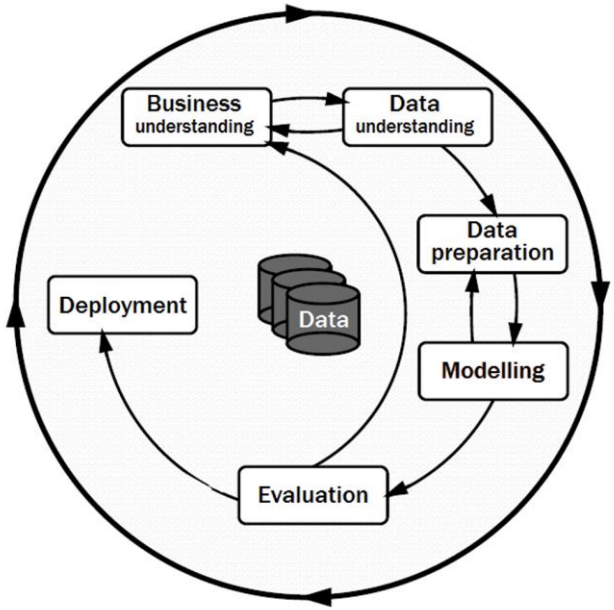
Еще 10 лет назад ученые были уверены, что все, что перечислено ниже, невозможно:

- Нейросети глубокого обучения распознают лица людей лучше чем сами люди
- Самообучающийся ИИ для игр (шахматы и го) обыгрывает любого человека, причем играет «по-человечески» (технически не всегда рационально), пример – претензии Каспарова к Deep Blue
- Большие языковые модели в задачах текстовой аналитики справляются лучше чем люди и практически не используют методы прикладной лингвистики
- Беспилотные автомобили на реальных дорогах

Интеллектуальный анализ данных



CRISP-DM: CRoss Industry Standard Process for Data Mining (1999)



Системы *интеллектуального анализа данных* – класс программных систем поддержки принятия решений, задачей которых является поиск *скрытых, ранее неизвестных, содержательных и потенциально полезных* закономерностей в *больших объемах разнородных, сложно структурированных* данных.

Han J., Kamber M. Data Mining: Concepts and Techniques // Morgan Kaufmann, 2000

Процесс интеллектуального анализа данных

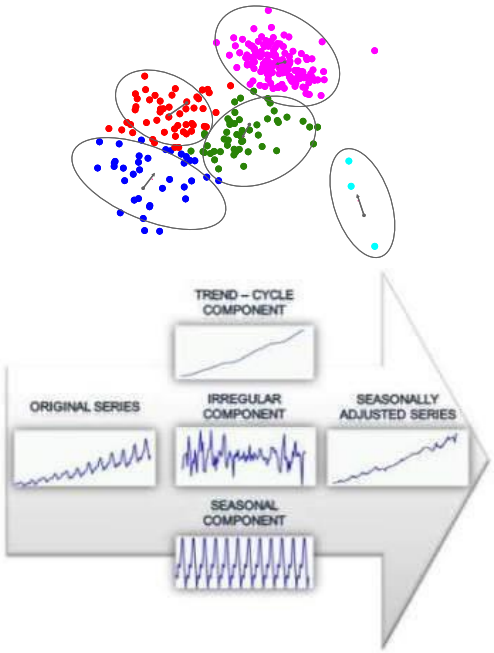
- Анализ предметной области:
  - выявление и формулировка необходимых априорных знаний о предметной области, целей анализа, задач приложения, сценариев использования
- Формирование и подготовка данных для анализа:
  - поиск (или выбор) «сырых» данных, возможно, реализация подсистемы сбора (консолидации)
  - предобработка данных (нормализация, дискретизация, обработка пропущенных значений, удаление артефактов, проверка консистентности)
  - уменьшение размерности, выбор значимых характеристик, расчет интегральных показателей и инвариантов
- Определение типа решаемой задачи анализа:
  - классификация, прогнозирование, кластеризация, поиск исключений, ассоциативный анализ и т.д.

Процесс интеллектуального анализа данных

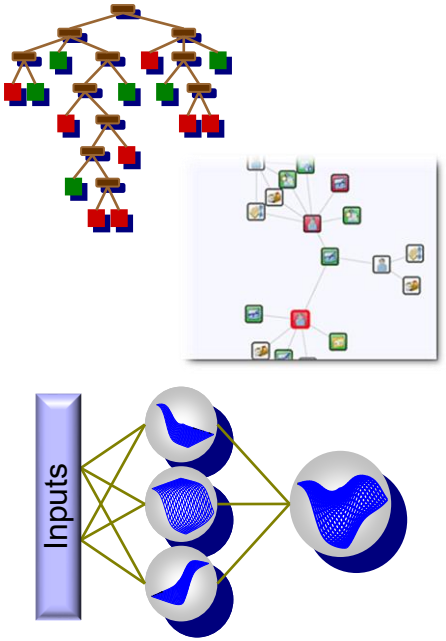
- Выбор (или разработка) алгоритма машинного обучения:
  - определение ограничений и требований к алгоритму по точности, размеру, интерпретируемости, скорости построения и применения получаемых моделей, по типу исходных данных
- Непосредственно построение моделей:
  - применение выбранного алгоритма анализа для поиска закономерностей выбранного типа и построение моделей
- Проверка моделей и представление результатов анализа:
  - визуализация, преобразование, удаление избыточности, оценка точности, достоверности моделей и т.д.
- Применение построенных моделей:
  - Descriptive data mining - информирование аналитика, «описательные» модели, основная цель – визуализация
  - Predictive data mining – прогнозирование неизвестных значений или характеристик в «новых» данных с помощью построенных моделей, основная цель – прогноз

Жизненный цикл аналитических моделей

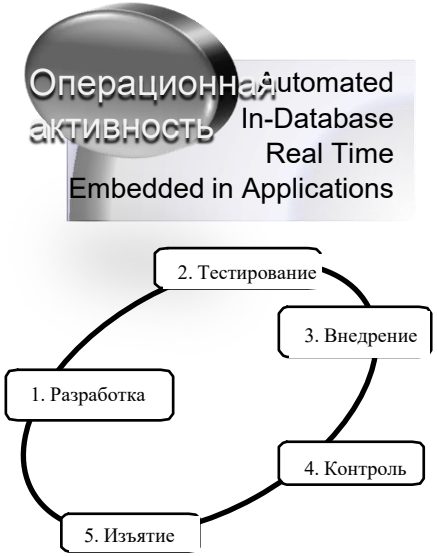
Выявление зависимостей



Построение моделей



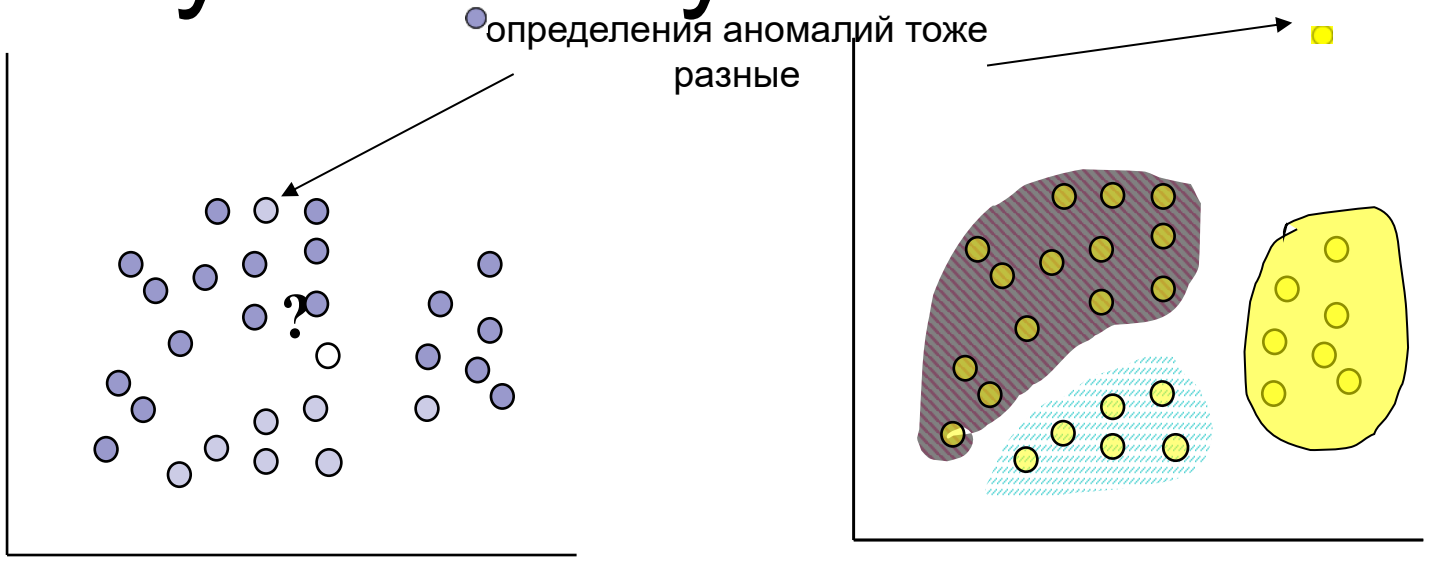
Внедрение моделей



Базовые задачи машинного обучения = типы выявляемых закономерностей

- Классификация («Обучение с учителем»)
  - Отнесение объектов к заранее определенным категориям
- Ранжирование («Обучение с учителем»)
  - Оценка степени соответствия объектов одной или более заранее определенным категориям
- Прогнозирование («Обучение с учителем»)
  - На основании известных значений атрибутов анализируемого объекта определяются значения неизвестных атрибутов
- Ассоциации («Обучение без учителя»)
  - Выявление зависимостей между атрибутами в виде правил или аналитических зависимостей, выявление скрытых свойств объектов
- Кластеризация («Обучение без учителя»)
  - Выделение компактных подгрупп «похожих» объектов
- Выявление исключений («Обучение с учителем и без»)
  - Поиск объектов, которые своими характеристиками значительно отличаются от остальных

Обучение с учителем и без



- «Размеченный» набор данных – выделен один или более признаков, которые могут быть неизвестны и которые нужно предсказывать, тогда задача обучения «с учителем», иначе «без учителя» («неразмеченный» набор данных):
  - «Выходные» признаки - нужно предсказывать (они же отклики, или «зависимые переменные», или …)
  - «Входные» признаки, которые считаются всегда известными (они же входы, или «независимые переменные», или регрессоры, …)

Обучение без учителя

- Иногда называют задачей «самоорганизации»
- Все признаки равнозначны, нет отклика, поэтому:
  - Обучение без учителя более «субъективное» (нет единых интуитивно понятных мер оценки качества типа «точности») и менее «автоматизируемо»
  - Сложнее подбирать метапараметры и сравнивать полученные модели
- Важность этого направления велико:
  - «Истинный data mining» - ищет неизвестные заранее зависимости без «подсказок» эксперта. Много важных прикладных задач по сегментации, выявление скрытых характеристик, зависимостей между атрибутами и т.д.
  - Для больших данных получить качественно размеченный набор тяжело или невозможно, часто возникают задачи *semisupervised learning*, когда размечена лишь часть набора
- Большинство задач сводится к поиску скрытых (латентных) признаков или скрытых структур (групп, отношений, зависимостей …) в данных:

$$z_j: D_1 \times \ \dots \times D_n \to D_{zj}, z_1 = F_1(f_1, \dots, f_n), \dots, z_k = F_k(f_1, \dots, f_n)$$

  - Основные случаи: z_j или отношения (например, кластеры, ассоциативные правила) или новые признаки (например, главные компоненты и степень принадлежности кластеру), иногда как в SOM (сетях Кохонена) и то, и другое сразу.

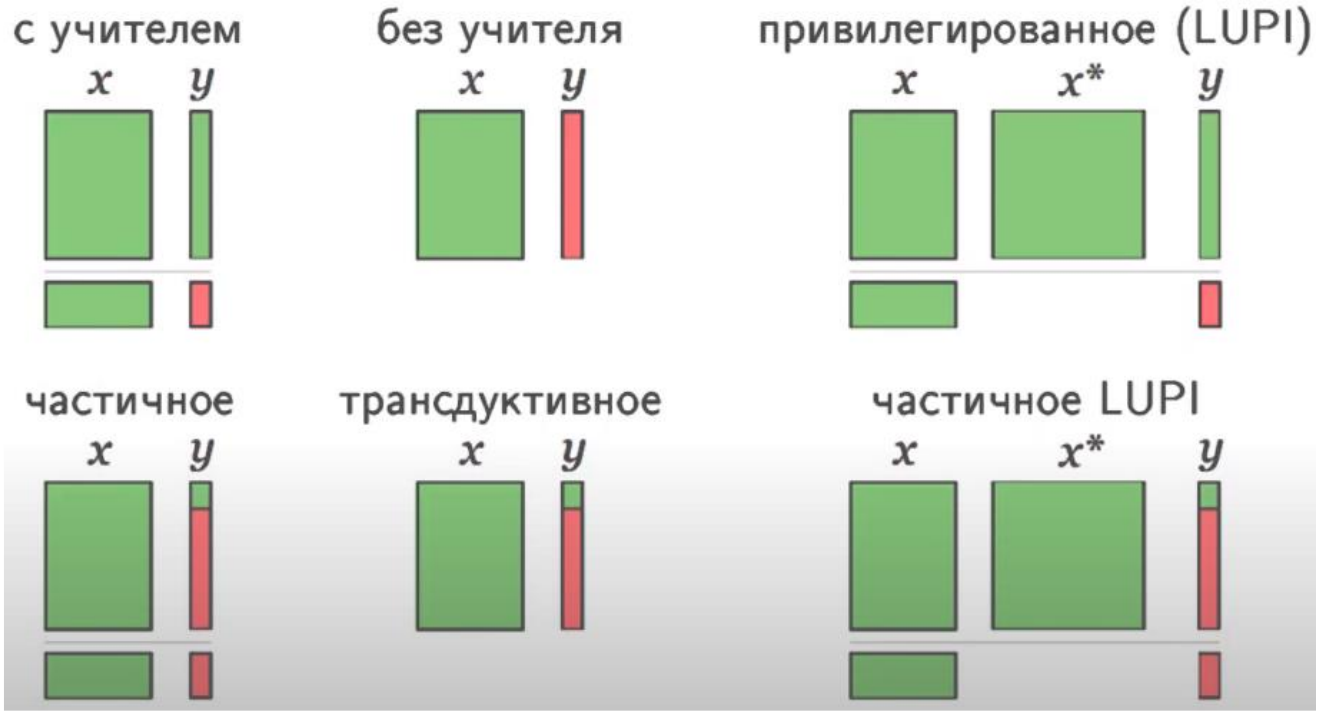
Обучение с учителем

- Дано: множество «размеченных» примеров :
  - обучающая выборка или тренировочный набор:
    $$Z = \{(x_i, y_i)\}_{i=1}^{l} \in X \times Y$$
  - $y_i \in Y$: известный «отклик» и его множество значений
  - Неизвестная зависимость, которую необходимо «восстановить» $\mathrm{y}: X \to Y$
- Постановка задачи:
  - Найти алгоритм (или гипотезу, или модель, или решающую функцию)
    $$a_Z: X \to Y$$
  - «качественно» приближающую неизвестную $\mathrm{y}: X \to Y$ на всем признаковом пространстве
- Два этапа:
  - Обучение (построение модели), метод обучения μ выбирает «лучший» алгоритм a (модель, гипотезу) среди заданного семейства A:
    $$a_Z = \mu(Z) \in A$$
  - Применение (скоринг модели), алгоритм a выдает (прогнозирует) значения отклика $\acute{y} = a_Z(\acute{x})$ для «новых» объектов (возможно для множества «новых» объектов) с неизвестным откликом

Типы задач обучения с учителем в зависимости от типа отклика

- Вообще тип задачи обучения с учителем определяются не только типом допустимых значений «отклика», но и **оценкой качества**, которая используется для выбора модели
- Типы задач:
  - Бинарная классификация $Y = \{0,1\}$ – решающая функция бинарная
  - Много-классовая классификация $Y = \{C_1, \dots, C_k\}$ – категориальный признак, значения взаимоисключающие, не задано отношение порядка, наблюдение не может принадлежать нескольким классам одновременно, решающая функция дискретная
  - Много-темная (multi-label) классификация $Y = \{0,1\}^k$ или Y – множество всех подмножеств $\{C_1, \dots, C_k\}$, решающая вектор функция выдает бинарный вектор релевантных тем
  - Регрессия $Y = \mathbb{R}$ или $Y = \mathbb{R}^k$, решающая функция – вещественная (вектор) функция
  - Порядковая регрессия $Y = \{1, \dots, K\}$ – дискретный признак, задано отношение порядка, решающая функция дискретная
  - Ранжирование $Y = \{C_1, \dots, C_k\}$ – категориальный признак, значения взаимоисключающие, решающая вектор функция выдает вектор степеней соответствия наблюдений каждому из классов

Типы задач обучения в зависимости от доступности разметки



- Трансдуктивное обучение – тестовая выборка известна заранее
- Привилегированное обучение – часть признаков известна только на этапе обучения

Основные типы исходных данных

- Транзакционные
  - Объекты анализа – «события» различной структуры с числовыми и категориальными атрибутами и с временной меткой
- Табличные
  - Объекты анализа представлены в виде реляционных таблиц, возможно взаимосвязанных (заданно ER-схемой), имеют разнотипные атрибуты
- Временные ряды и числовые данные большого объема
  - Обработка результатов наблюдений, научных экспериментов, характеристик технологических процессов
- Электронные тексты на естественном языке
  - анализ содержимого документов
- Графовые данные
  - Анализ взаимосвязей (SNA)
- Сложно структурированные данные
  - Мультимедия, геоданные, ДНК, программный код и многое другое

Исходные данные

- Объект анализа (или прецедент, или кейс, или наблюдение, …) x из некоторого возможно бесконечного множества объектов X задается набором признаков f_j (или атрибутов, или свойств, …)

$$f_j: X \to D_j,\ x \in X$$

- Домен (область определения, множество значений, тип) признака:
  - Категориальный: D_j как правило конечно, нет расстояния, не задан порядок
  - Ординальный (порядковый): как категориальные, но задан порядок в виде транзитивного, антисимметричного отношения, но нет расстояний

$$R_j: D_j \times D_j\ , \left(x_a R_j x_b\ \wedge\ x_b R_j x_c\right) \Rightarrow\ x_a R_j x_c, \left(x_a R_j x_b\ \wedge\ x_b R_j x_a\right) \Rightarrow\ \ x_a = x_b$$

  - Числовой – есть расстояние, $D_j = \mathbb{R}$
  - Бинарный $D_j = \{0,1\}$
- Вектор признаков, описывающий объект $(f_1(x), f_2(x), \ldots, f_n(x))$

Пространство признаков

- Набор данных - множестно $Z = \{x_1, \dots, x_l\}$, включает l объектов (наблюдений) из X и может быть представлено:
  - Матрицей признаков

$$\|F\|_{l \times p} = \begin{pmatrix} f_1(x_1) & f_2(x_1) & \dots & f_p(x_1) \\ & \dots & & \\ f_1(x_l) & f_2(x_l) & \dots & f_p(x_l) \end{pmatrix}$$

  - Симметричной матрицей сходства (или различия)

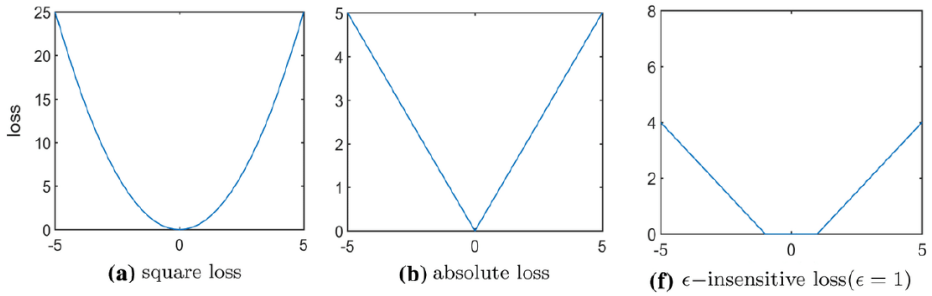
$$\|K\|_{l \times l} = \begin{pmatrix} d(x_1, x_1) & d(x_1, x_2) & \dots d(x_1, x_l) \\ & \dots & \\ d(x_l, x_1) & d(x_l, x_2) & \dots d(x_l, x_l) \end{pmatrix}$$

  где $d: X \times X \to \mathbb{R}$ некая симметричная, мера сходства или различия, не обязательно расстояние (правило треугольника может не выполняться)
- Вопрос построения признакового пространства – не простой, а иногда и критический, причины:
  - Мусор на вход – мусор на выход
  - Сложные структурированные объекты (например, графы, тексты и т.д.)
  - Противоречия, пропуски, ошибки, артефакты
  - Взаимозависимые и незначимые признаки

Функция потерь

- $L: Y \times Y \to \mathbb{R}^+$ характеризует отличие истинного отклика от спрогнозированного $L(y(x), a(x))$
- Примеры:
  - Классификация и регрессия:

$$L(y, y') = [y \neq y'],$$
$$L(y, y') = |y - y'|,$$
$$L(y, y') = (y - y')^2,$$
$$L(y, y') = [|y - y'| > \varepsilon]$$



  - Много-темная классификация:

$$L(y, y') = |y \nabla y'|, a \nabla b = (a \cup b) \backslash (a \cap b), a \subseteq Y, b \subseteq Y$$

  - Ранжирование:

$$L(y, y') = \frac{|\{(r, s): y_r \leq y_s \wedge y'_r \leq y'_s\}|}{|y||y'|}$$

Функционал качества и эмпирический риск

- Теоретический риск:

$$\mathrm{E}\big[L\big(y(x),a(x)\big)\big] = \int_{X\times Y} L\big(y(x),a(x)\big)\partial P(x,y) \to min$$

- Но $P(x,y)$ мы не знаем, а если бы знали, то и решать ничего не нужно, поэтому используем эмпирический риск:

$$Q(a,Z) = \frac{1}{l}\sum_{Z} L\big(y_i,a(x_i)\big) \to min$$

- Получаем, что метод обучения μ выбирает «лучший» алгоритм a_Z^* (модель, гипотезу) среди заданного семейства A на наборе данных Z как:

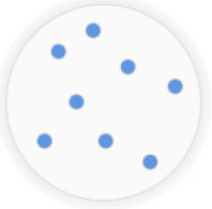
$$a_Z^* = \mu(Z) = \arg\underset{a\in A}{min}\, Q(a,Z)$$

- Можно ли использовать оценку качества на обучающей выборке как объективную? Ответ – НЕТ

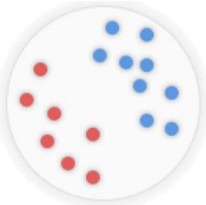
Гипотеза о компактности (непрерывности)

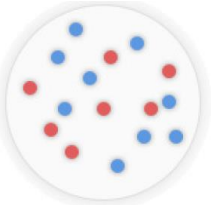
- Не формальная (эвристическая) гипотеза:
  - «Близкие» наблюдения в признаковом пространстве должны должны обладать похожими свойствами, для прогнозирования - иметь «похожие» отклики
- Непрерывность (для регрессии)

выполнена: 

не выполнена: 

- Компактность (для классификации)

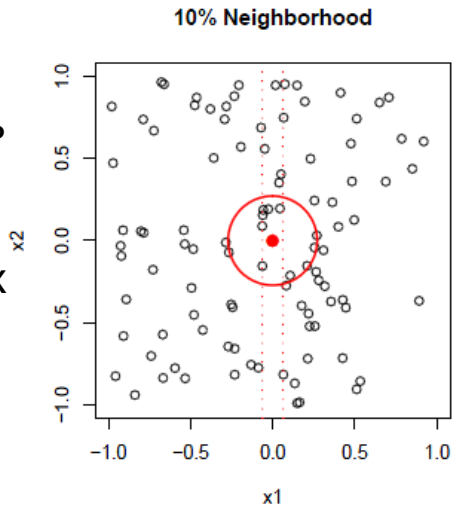
выполнена: 

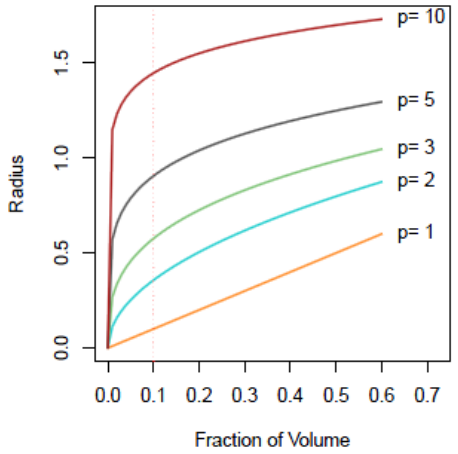
не выполнена: 

Проклятие размерности

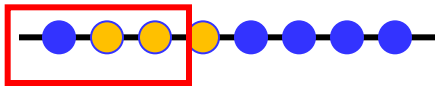
- Суть проблемы: экспоненциальный рост числа необходимых наблюдений при линейном росте размерности пространства
- Пример: ближайшие соседи как правило расположены далеко при больших размерностях пространства признаков.

Например, нам нужно получить значительную часть выборки, чтобы сгладить границу и снизить случайность в усредненном прогнозе, пусть 10%.
10% соседей для случая больших размерностей не может быть локализована, так что мы уже не можем оценить отклик на основе локального усреднения.

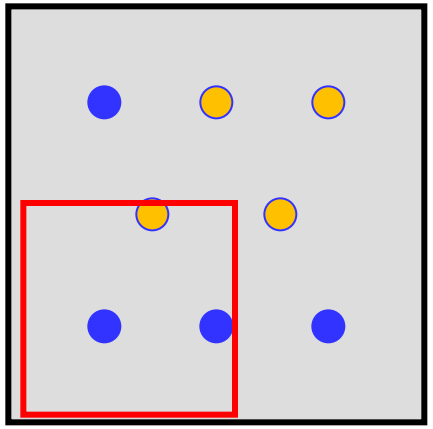




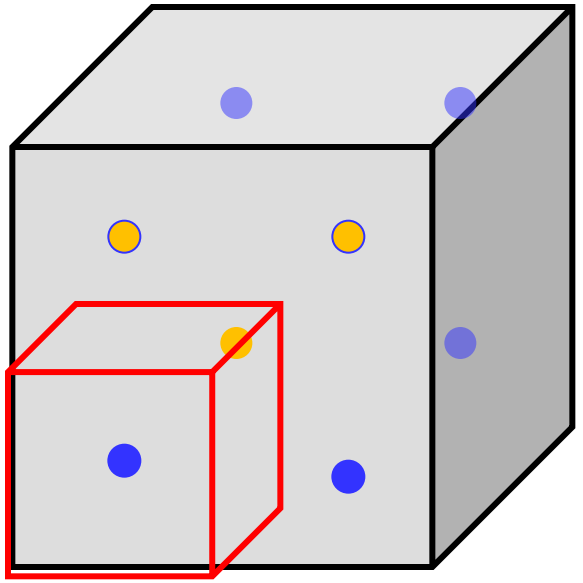
Модельный пример, демонстрирующий проклятие размерности



1–D



2–D



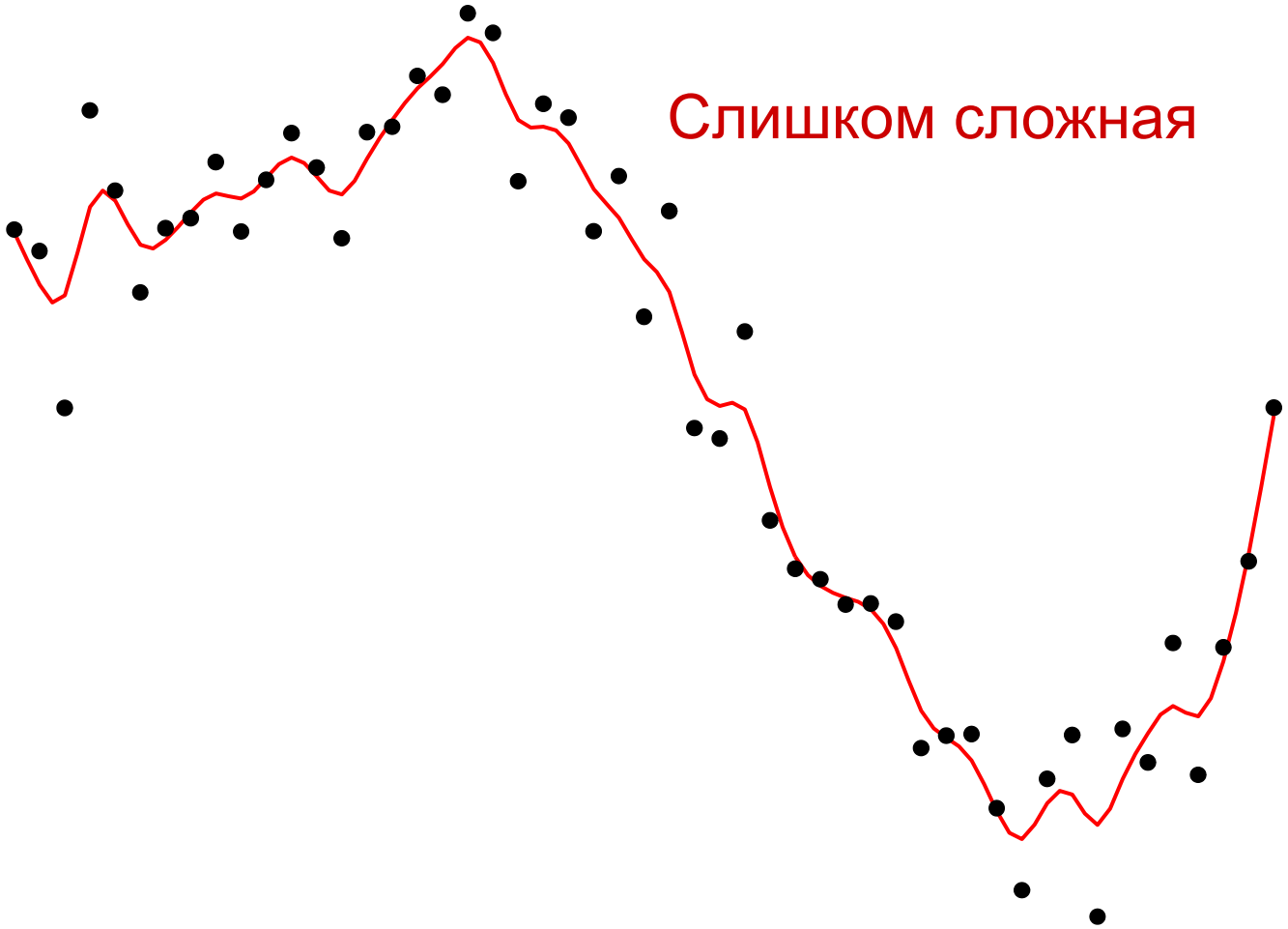
3–D

- $r=K/N$
- $E_p(r)=r^{1/p}$
- $E_{10}(0.01)=0.63$
- $E_{10}(0.1)=0.8$

Сложность модели

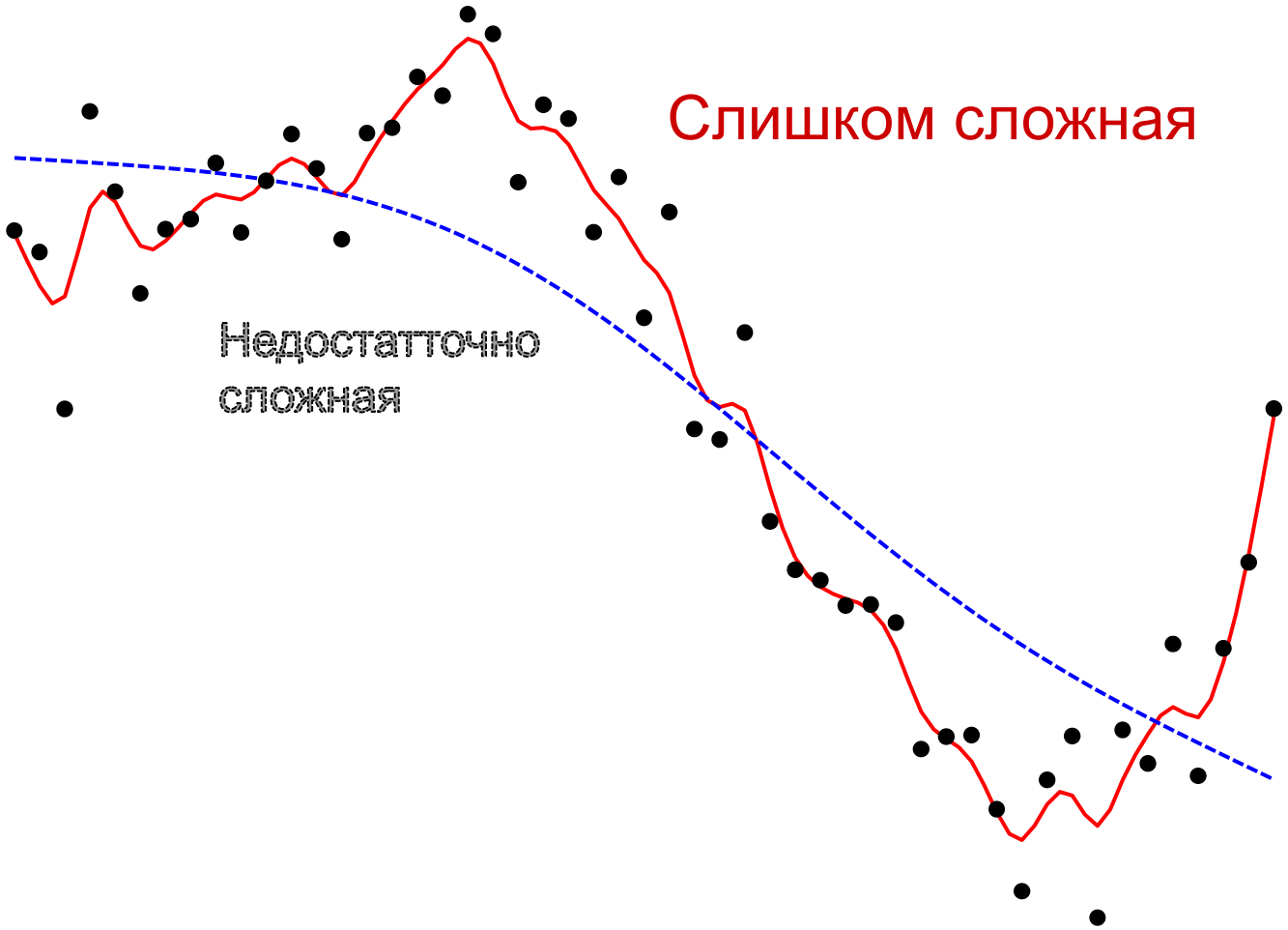
...

Сложность модели



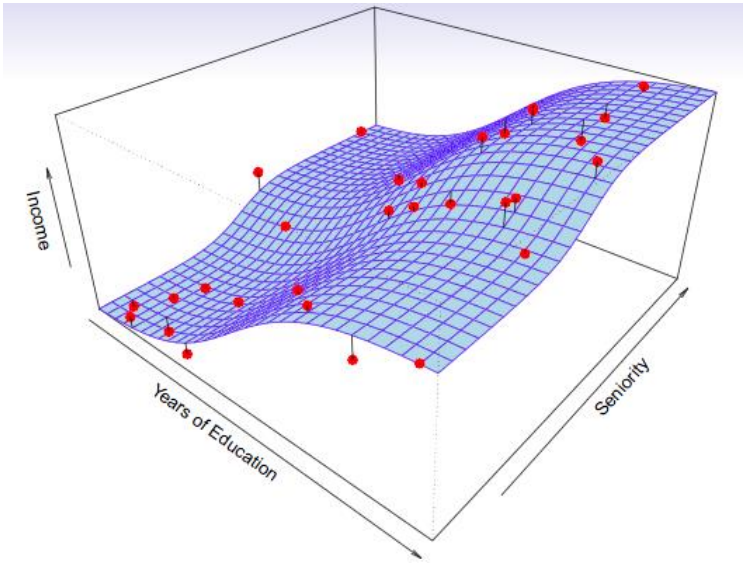
...

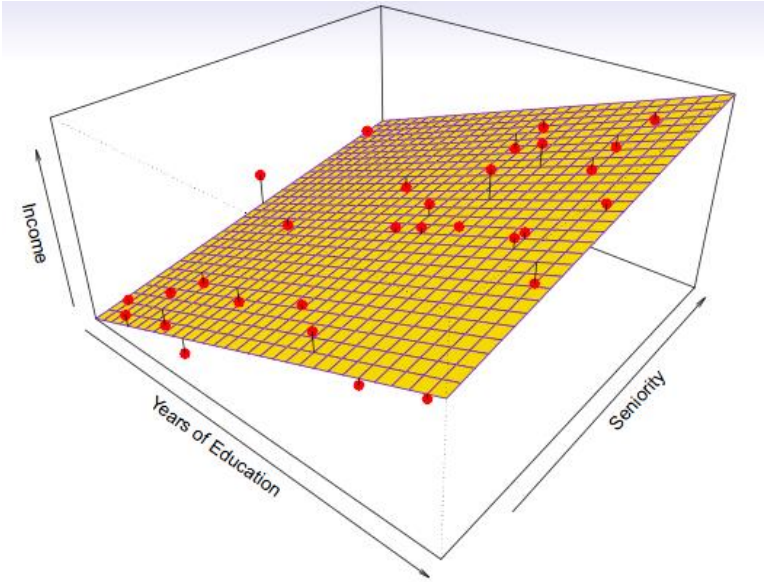
Сложность модели



...

Проблема недообучения и переобучения





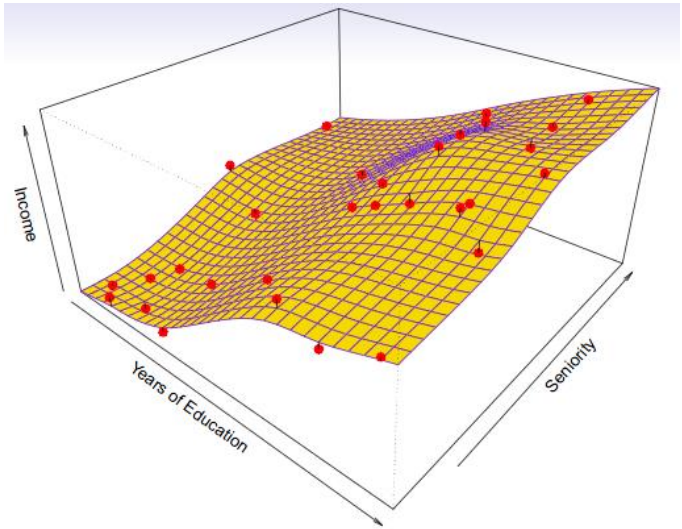
Модельный пример.

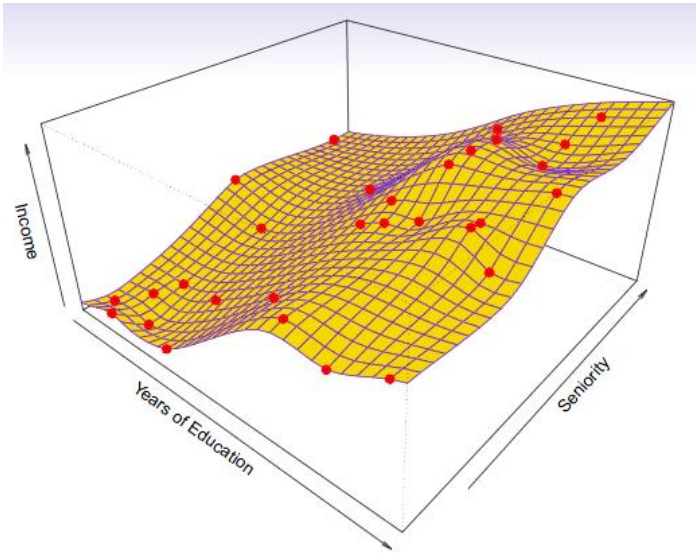
- Красные точки - наблюдения, синяя поверхность – истинная зависимость $\texttt{income} = f(\texttt{education}, \texttt{seniority}) + \epsilon$
- Желтая поверхность линейная модель

$$\hat{f}_L(\texttt{education}, \texttt{seniority}) = \hat{\beta}_0 + \hat{\beta}_1 \times \texttt{education} + \hat{\beta}_2 \times \texttt{seniority}$$

- Плохая точность приближения

Проблема недообучения и переобучения





Модельный пример.

- Более сложные модели (сплайны или полиномиальные регрессии или нейронные сети или еще что-то)
- Справа модель не допускает ошибок на обучающем наборе.
- Это хорошо? Нет!

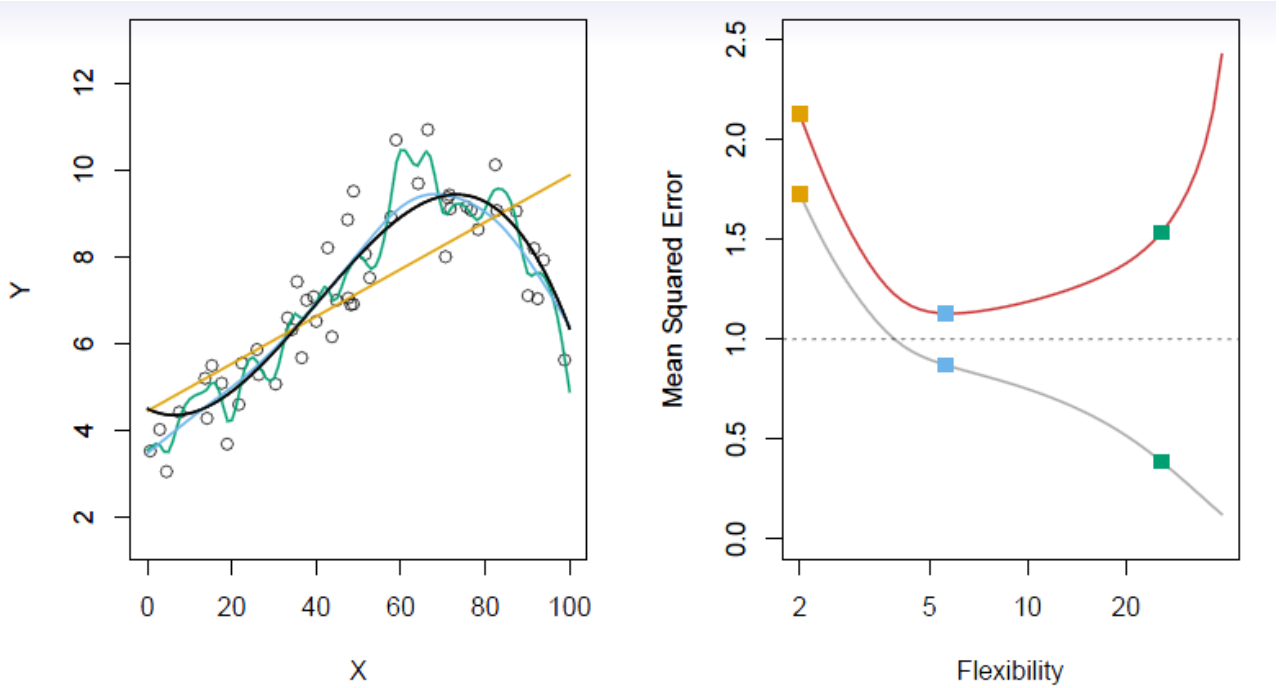
Недообучение vs переобучение

- Основная проблема машинного обучения!!!
  - Недообучение - низкое качество (большой эмпирический риск) на тренировочном наборе и на этапе скоринга
  - Переобучение - высокое качество (маленький эмпирический риск) на тренировочном наборе и плохое качество на этапе скоринга (большой эмпирический риск)
- Причины:
  - Сложность модели: например, для параметрических моделей много степеней свободы (параметров модели) или слишком сложное уравнение или большая норма вектора параметров
  - Плохое качество данных: шум и выбросы, малый объем или несбалансированность тренировочной выборки
  - Зависимости в пространстве признаков
- Обобщающая способность:
  - способность алгоритма «качественно» прогнозировать отклик для объектов одной природы (из одной генеральной совокупности), которых не было в тренировочном наборе
  - Как оценить?

Борьба с переобучением

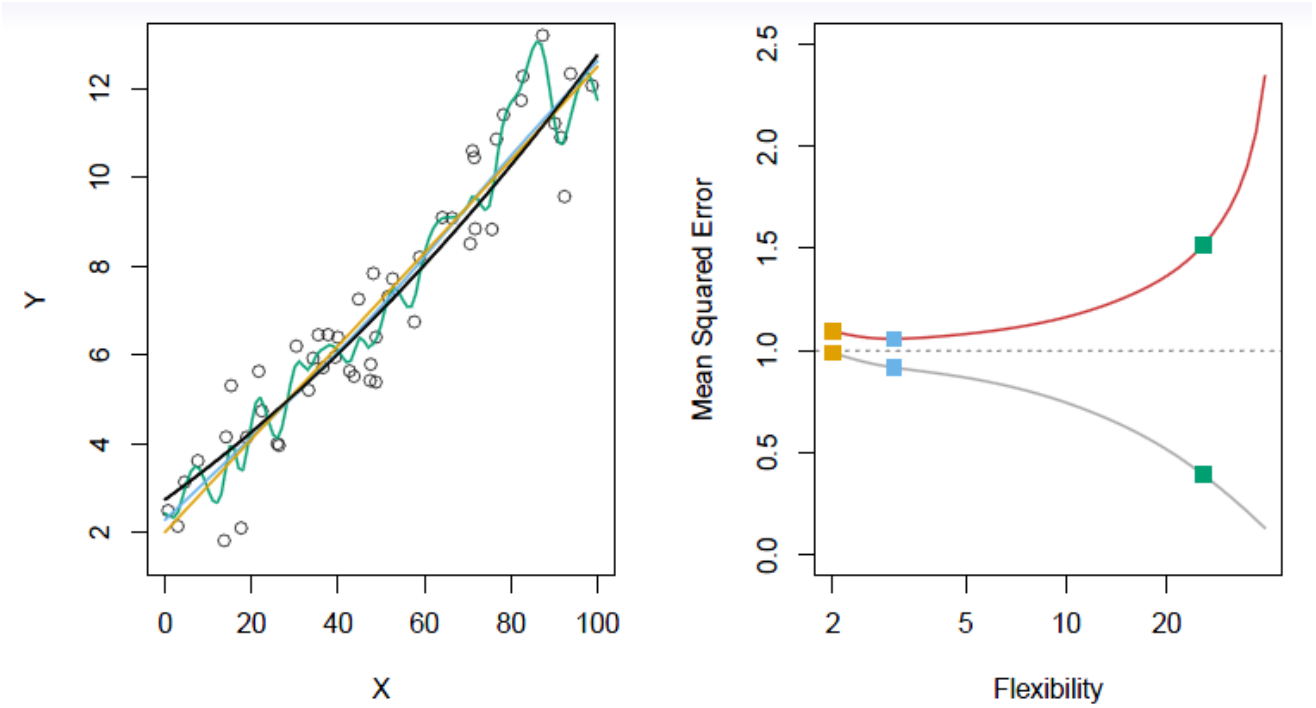
- Ограничить сложность модели (например, регуляризация)
- Преобразовать данные (удалять шум, уменьшать размерность и тд.)
- Использовать теоретические оценки обобщающей способности для некоторых методов обучения (обычно бесполезно, т.к. это оценки сверху)
- Эмпирически оценивать обобщающую способность с помощью тестовой выборки (или процедуры, имитируя проверку на тестовой выборке):
  - Строим модель на обучающем наборе данных и хотим, чтобы она была наилучшей.
  - Можем взять, например, квадратичную функцию потерь и оценить ее через среднеквадратичную ошибку MSE_{Tr}
  - Оценка может быть смещена в сторону более сложных моделей.
  - Поэтому мы вычисляем оценку MSE_{Te}, используя тестовый набор данных, который не участвовал в обучении модели

Оценка качества модели (сложная зависимость, много шума)



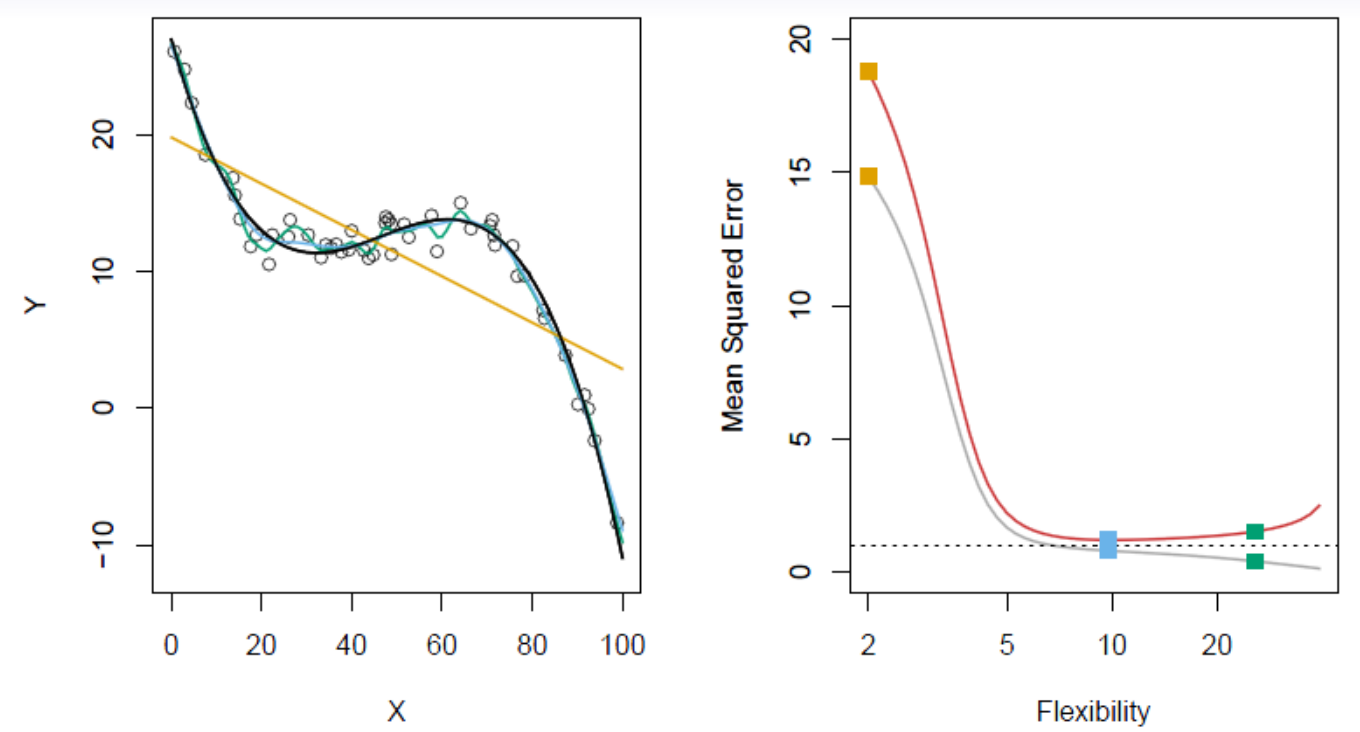
- Кривая, обозначенная черным цветом, - истинные значения.
- Красная кривая на правом рисунке – MSE_{Te} , серая кривая – MSE_{Tr}.
- Оранжевая, голубая и зеленая кривые соответствуют подгонке моделей различной сложности.
- Простые модели недообучены, сложные модели переобучены

Оценка качества модели (простая зависимость, много шума)



- Простые модели дают высокую обощающую способность
- Сложные модели переобучены

Оценка качества модели (сложная зависимость, мало шума)



- Простые модели недообучены
- Сложные обладают хорошей обобщающей способностью

MSE декомпозиция

$$MSE = E[(a(x) - y(x)]^2 = E[a(x)^2] + E[y(x)^2] - E[2a(x)y(x)] =$$
$$= Var(a(x)) + Var(y(x)) + (E[a(x)] - E[y(x)])^2$$

Дисперсия прогноза

Дисперсия шума

(не зависит от модели)

Квадрат смещения

Компромисс: Дисперсией vs Смещение!!!!

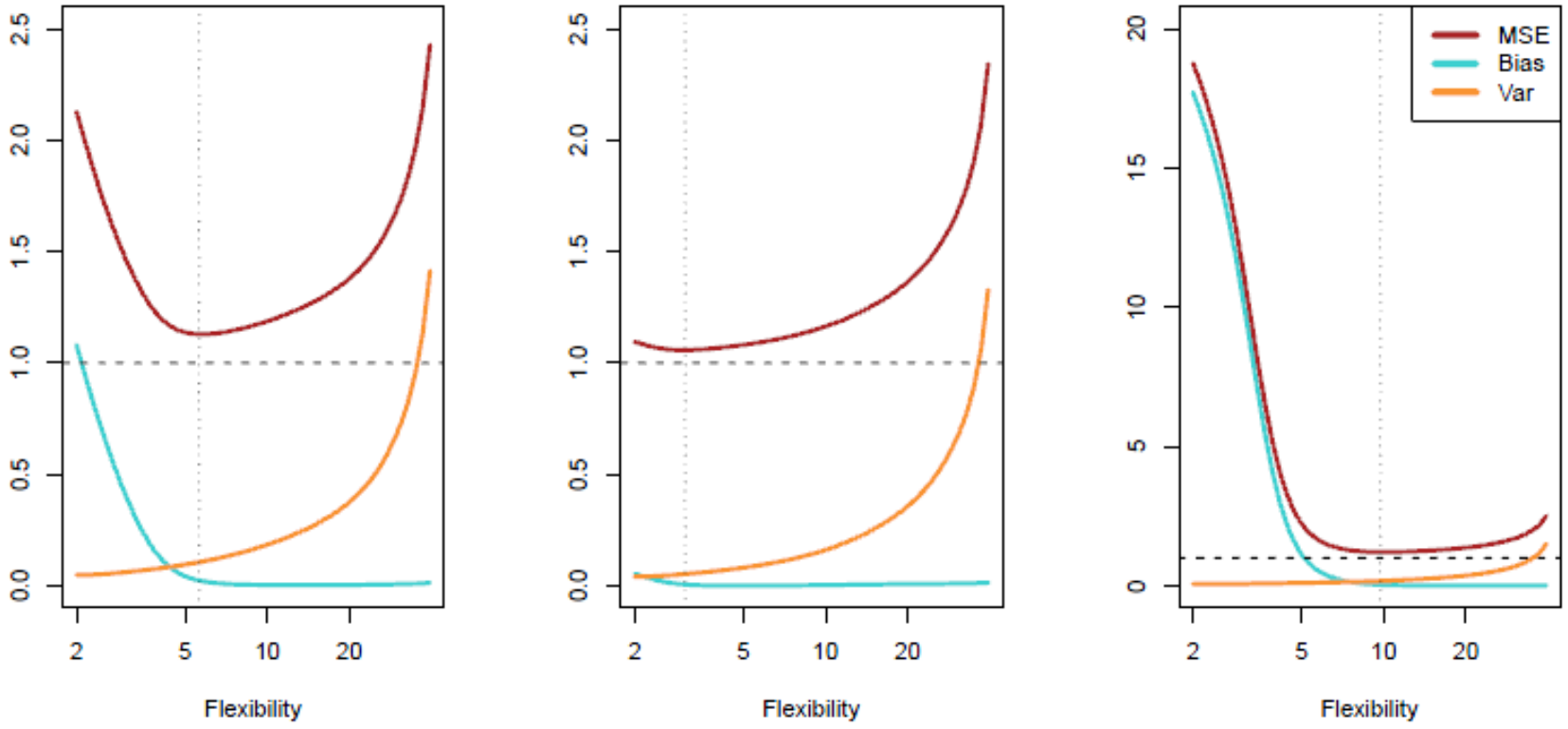
Сложнее модель => точнее приближение => меньше смещение +++

Сложнее модель => больше параметров => больше дисперсия ---

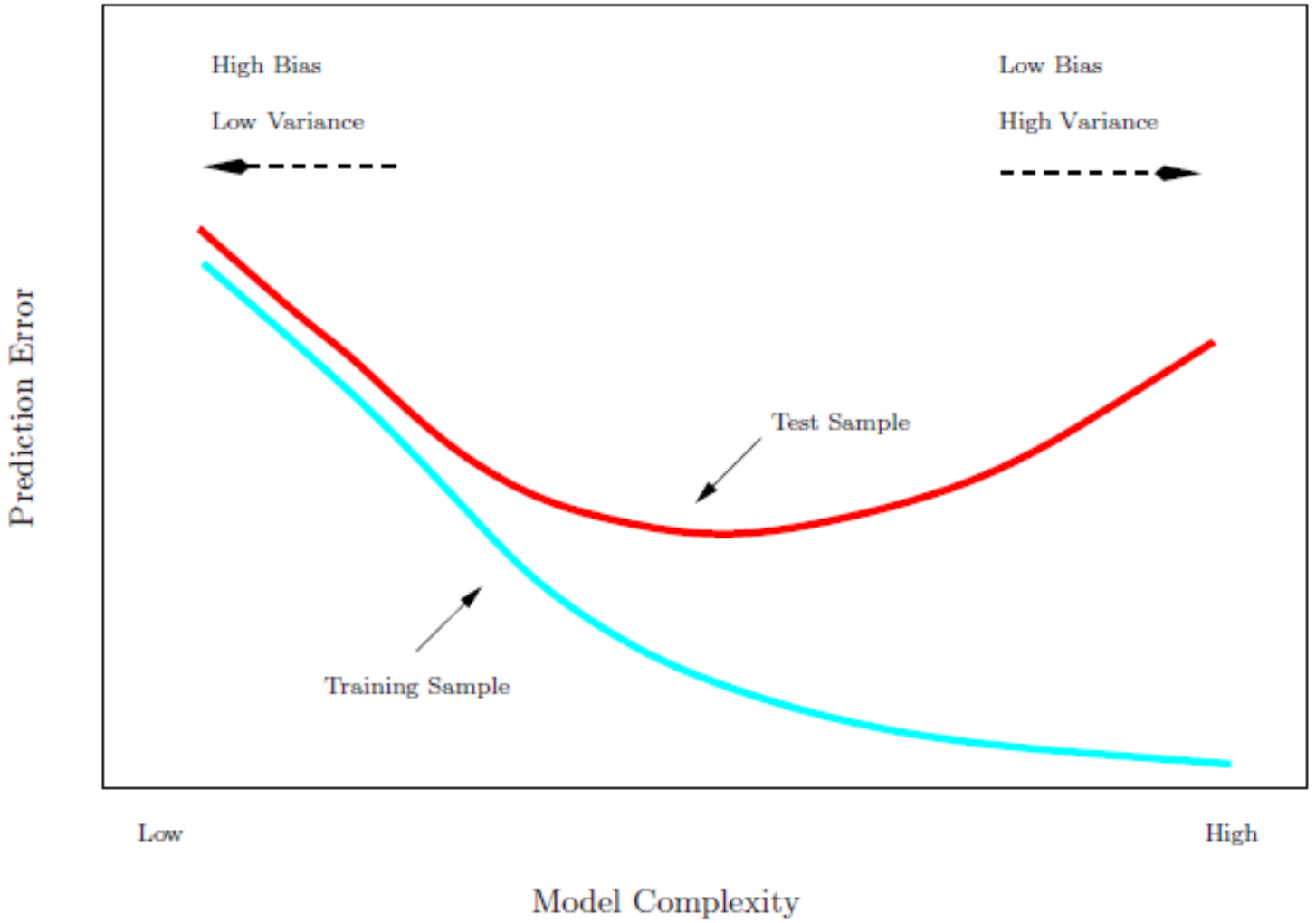
… и наоборот …

Поиск баланса между точностью и сложностью = поиск компромисса между смещением и дисперсией

Компромис отклонения и смещения для трех примеров



Качество на обучающем и тестовом наборе

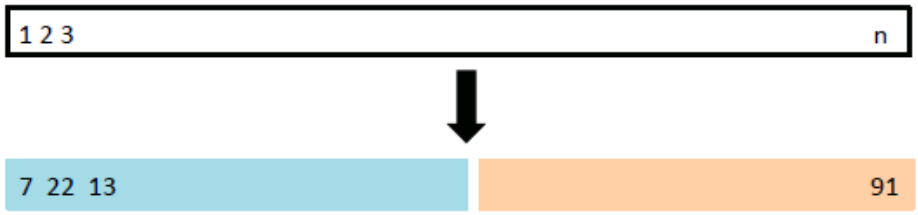


Валидация, кросс-валидация и бутстреппинг

- Эти методы позволяют:
  - оценить ошибки прогнозирования тестового набора
  - найти оценки параметров модели
  - выбрать лучшую модель
- Различия между *ошибкой тестирования* и *ошибкой обучения:*
  - Ошибка тестирования - это усредненная ошибка, которая возникает в результате применения модели машинного обучения для прогнозирования отклика на новом наблюдении, которое не было задействовано в процессе обучения.
  - Ошибка обучения вычисляется после применения алгоритма машинного обучения к наблюдениям, используемым в обучении.

Применение валидационного набора

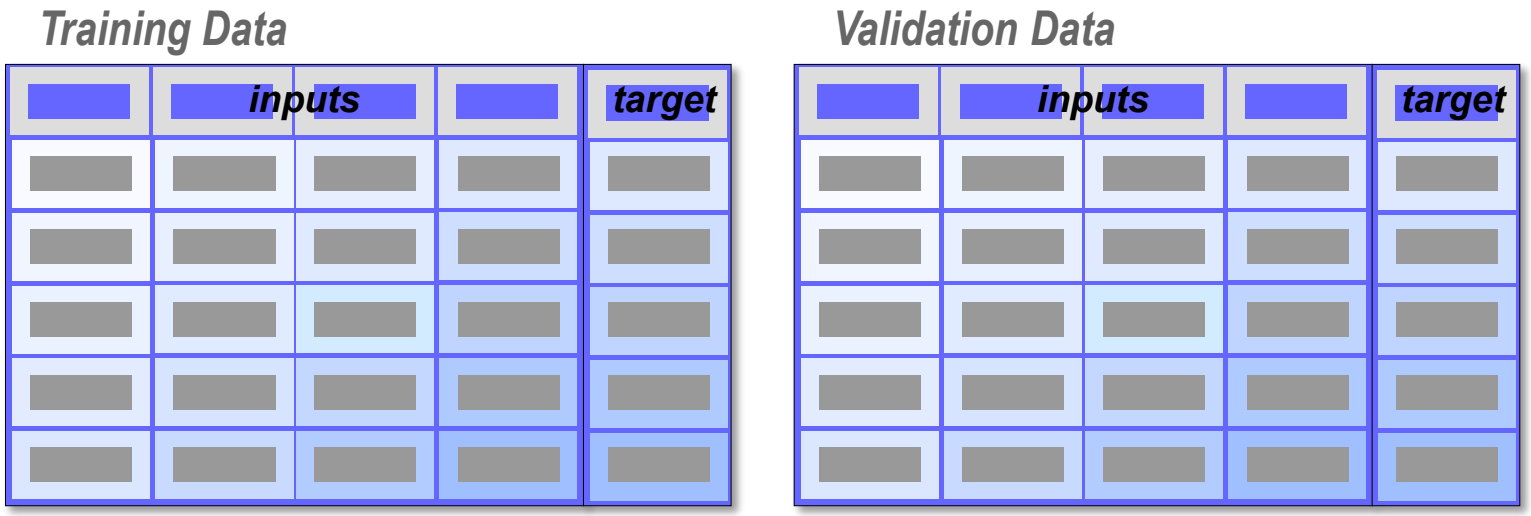
- Разделим случайным образом имеющийся набор образцов на две части*: обучающую и валидационную выборки*.



- Построим модель на обучающем наборе и используем ее для прогнозирования откликов наблюдений в валидационном наборе.
- Полученная ошибка на валидационном множестве дает оценку тестовой ошибки.

$$HO\ (\mu, Z, Z_{val}) = Q(\mu(Z \backslash Z_{val}), Z_{val})$$

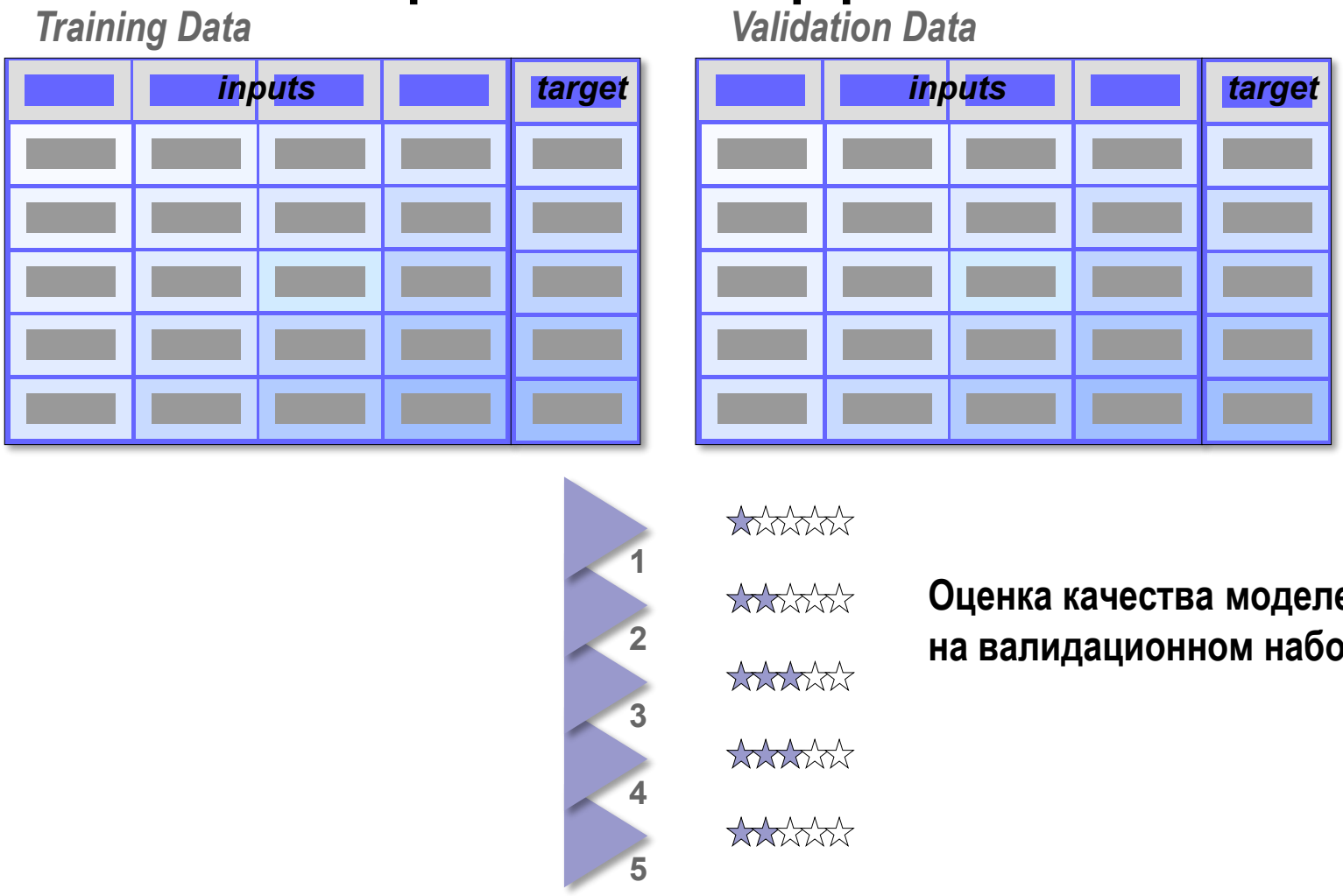
Использование валидационного набора данных



Основные методы генерации валидационного набора:

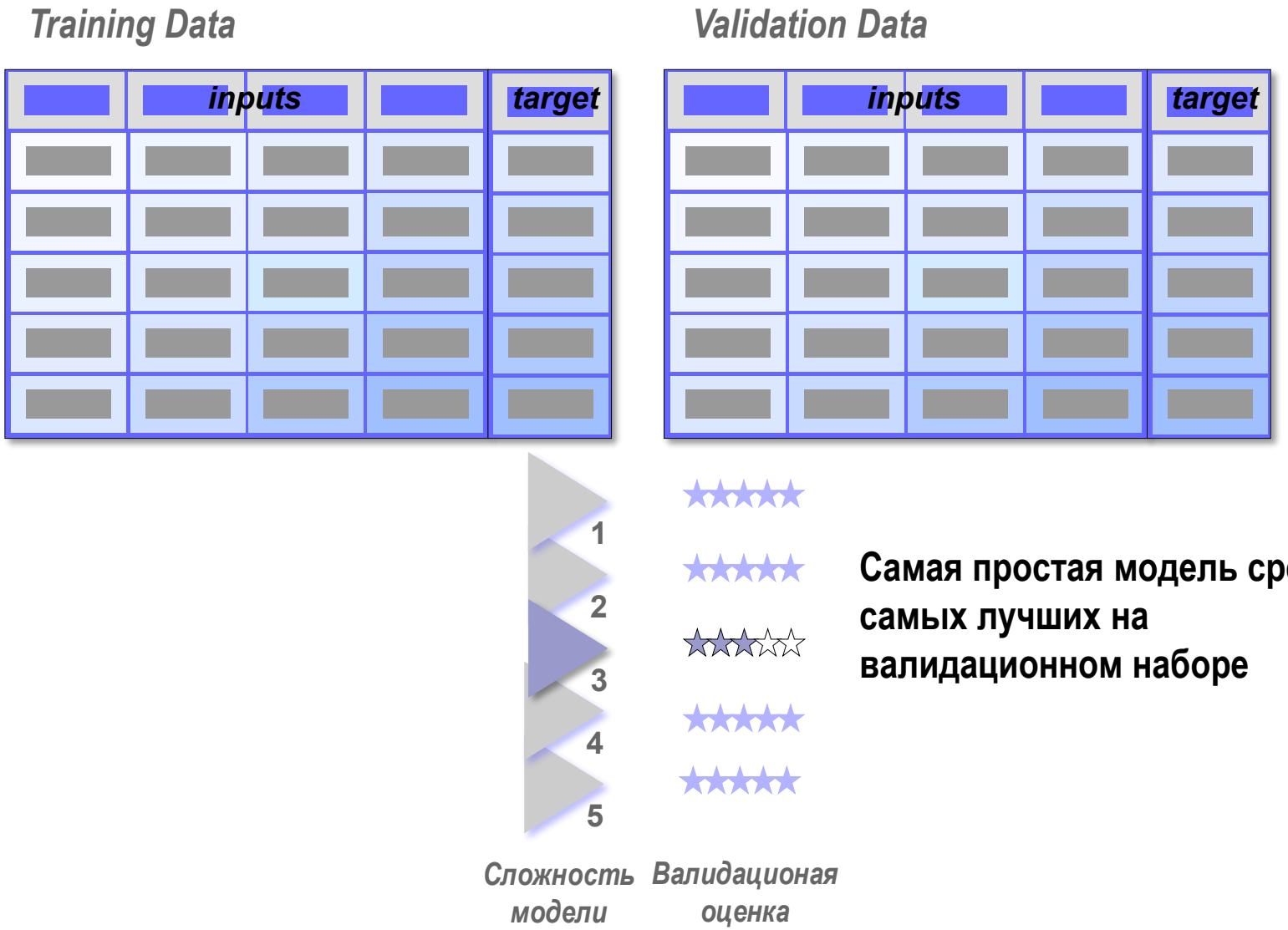
- Случайная выборка
- Стратифицированная выборка (сохраняем распределение выбранных переменых)
- Кластерная выборка (сохраняем пропорции кластеров)

Оценка моделей



Оценка качества моделей на валидационном наборе

Выбор модели



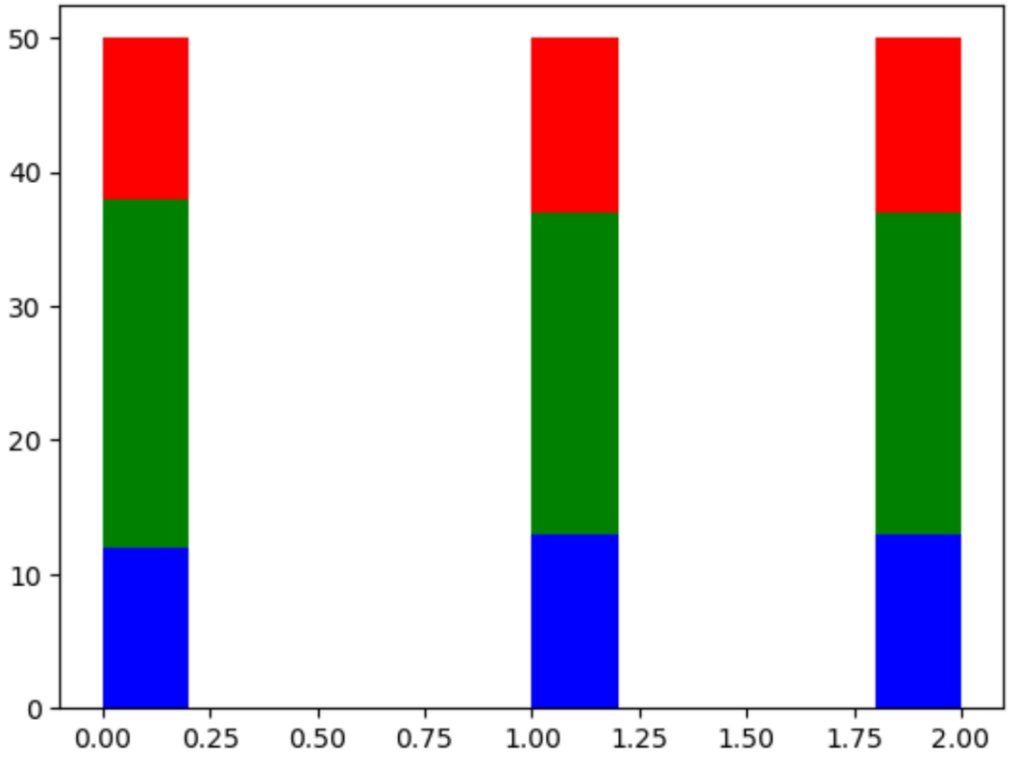
Самая простая модель среди самых лучших на валидационном наборе

Пример (Python)

```python
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris
```

```python
iris = load_iris()
X_train, X_test, y_train, y_test, = train_test_split(iris.data, iris.target, test_size=0.25,
    shuffle=True, # Random select
    stratify=iris.target, # Stratification
    random_state=123)
```

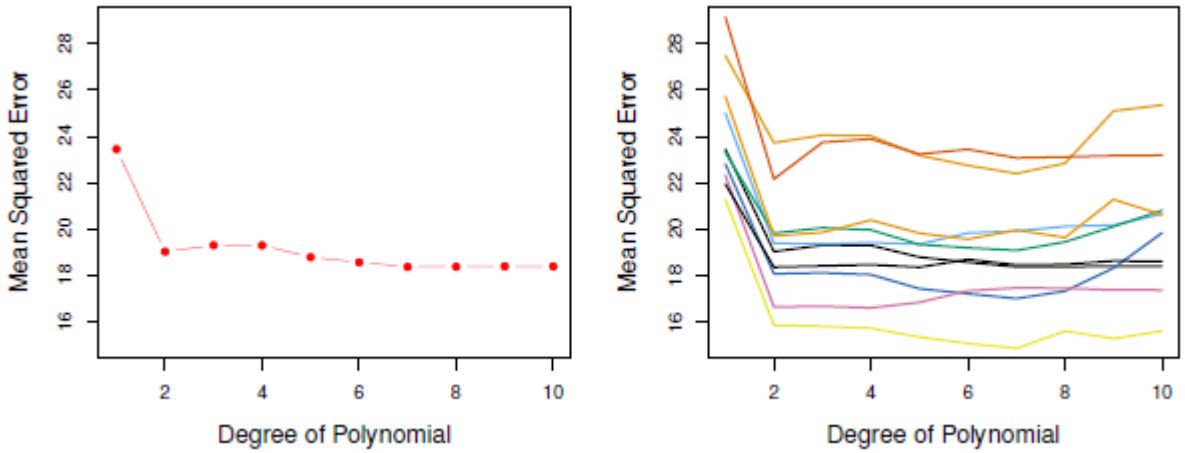
```python
plt.hist(iris.target, color="red")
plt.hist(y_train, color="green")
plt.hist(y_test, color="blue")
pass
```



Для кластерной выборки передаем в качестве stratify метки кластеров

Пример

- Хотим сравнить регрессионные модели с разными степенями полиномима
- Разделим случайным образом 392 наблюдения на две группы: обучающий набор, содержащий 196 объектов и валидационный набор, содержащий оставшиеся 196 объектов.



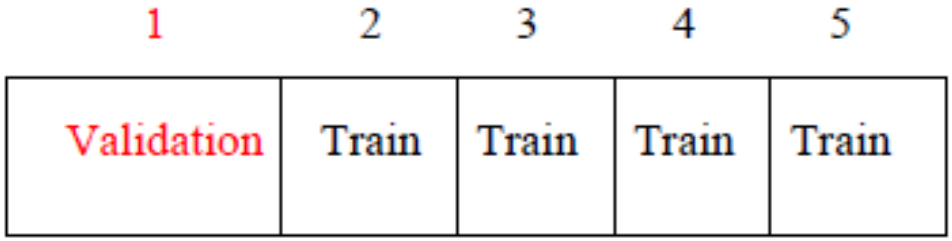
Слева показано одиночное разбиение, справа - множественное

Недостатки подхода применения валидационного набора

- Если плохое разбиение:
  - Валидационная оценка ошибки тестирования может сильно варьироваться в зависимости от того, какие именно наблюдения включены в обучающий наборе, а какие в валидационный.
- Не вся информация используется при обучении:
  - При валидационный подходе только подмножество наблюдений (те, которые включены в обучающий набора, а не в валидационный) используются для построения модели.
- Чрезмерный оптимизм:
  - Ошибка на валидационном наборе может иметь тенденцию *переоценивать* ошибку тестирования

Кросс-валидация

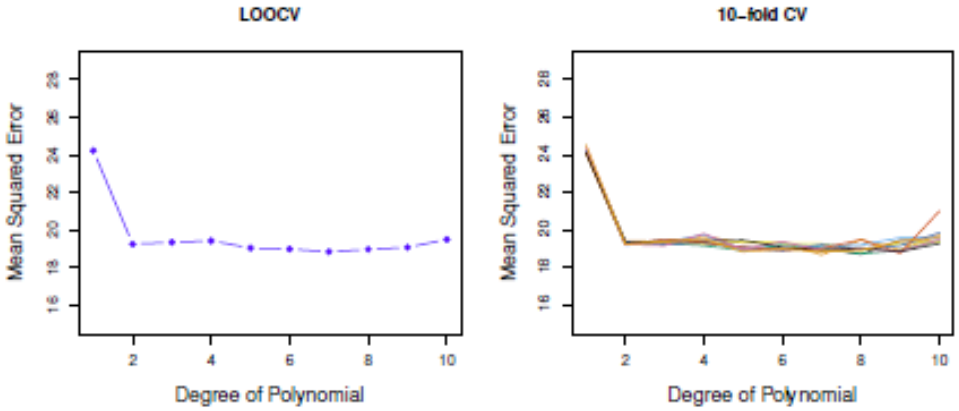
- *Широко используемый подход* для оценки ошибки тестирования.
- Оценки могут быть использованы для:
  - выбора оптимальной модели,
  - оценки тестовой ошибки результирующей выбранной модели.
- Идея - разделить данные на *K* частей равного размера. Мы удаляем часть *k,* строим модель на оставшихся частях, а затем получаем прогнозы для удаленной *k*-ой части.



- Это делается в свою очередь для каждой части *k = 1, 2, ..., K*, а затем результаты объединяются.

Кросс-валидация для оценки ошибки

- Обозначим *K* частей как $Z_1, \ldots, Z_K$, где Z_k - наблюдения в части k. l_k - наблюдений в части k, удобно брать $l_k = l/K$
- Вычислим: $CV_Z(\mu) = \sum_{k=1}^{K} \frac{l_k}{l} Q(\mu(Z \backslash Z_k), Z_k)$

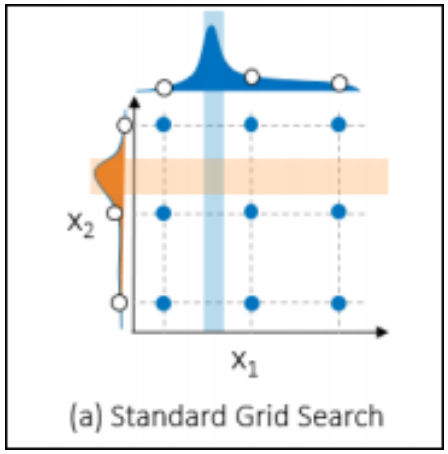


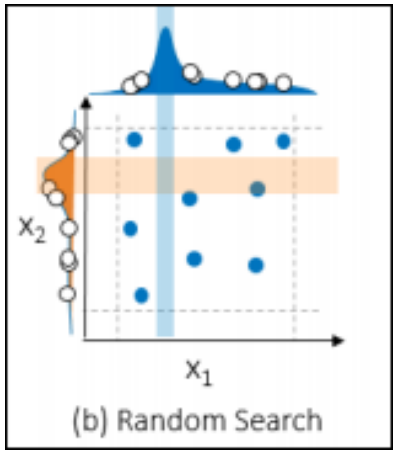
- При $K = l$ имеем l папок или кросс-валидацию с попеременным исключением одного наблюдения – скользящий контроль (*leave-one out cross-validation,* LOOCV).

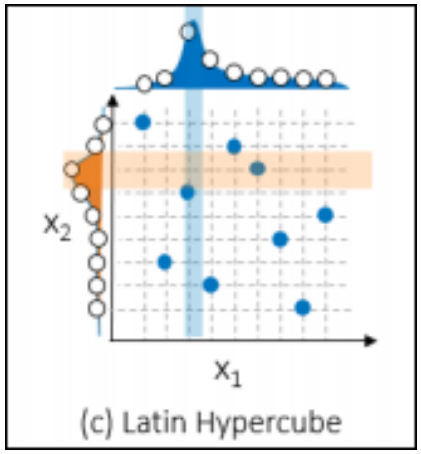
Кросс-валидация для оценки метапараметров (мета-обучение) и выбора модели

- Задают стратегию перебора вариантов метапараметров
- Запускают кросс-валидацию для разных значений метапараметров
- Рассчитывают кросс-валидационные ошибки для каждого варианта
- Выбирают лучшее значение метапараметра по кросс-валидационной ошибке
- Перестраивают модель на всей выборке с этим значением метапараметра

Кросс-валидация и валидация для выбора метапараметров (мета-обучение)







- AutoML:
  - Поиск по решетке
  - Случайный поиск
  - Латинский гиперкуб
  - Эволюционные и генетические алгоритмы поиска
  - «Мета» оптимизация
  - Байесовская оптимизация
- Оценка качества:
  - Не обязательно (и даже как правило) оценка качества для выбора модели совпадает с функцией потерь для обучения модели

Пример (Python)

```
from sklearn.utils import resample
```

```
X, y = fetch_california_housing(return_X_y=True, as_frame=True)
X
```

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-122.23
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-122.22
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.24
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	-122.25
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	-122.25
...	...	...	...	...	...	...	...	...
20635	1.5603	25.0	5.045455	1.133333	845.0	2.560606	39.48	-121.09
20636	2.5568	18.0	6.114035	1.315789	356.0	3.122807	39.49	-121.21
20637	1.7000	17.0	5.205543	1.120092	1007.0	2.325635	39.43	-121.22
20638	1.8672	18.0	5.329513	1.171920	741.0	2.123209	39.43	-121.32
20639	2.3886	16.0	5.254717	1.162264	1387.0	2.616981	39.37	-121.24

20640 rows × 8 columns

Пример – Grid Search (Python)

```
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.datasets import fetch_california_housing
```

```
X, y = fetch_california_housing(return_X_y=True)
```

```
N = 5000
X, y = X[:N], y[:N]
X.shape, y.shape
```

```
((5000, 8), (5000,))
```

```
scaler = StandardScaler()
VT = VarianceThreshold() # Preprocessing
KNN = KNeighborsRegressor() # Regressor
```

```
# Combined model - encapsulates all stages
model = Pipeline([("scaler", scaler), ("VT", VT), ("KNN", KNN)])
```

Пример – Grid Search (Python)

```
# Parameters to cycle through
# Pipeline parameters are passed as <STAGE>__<PARAMETER NAME>
parameters = {"KNN__n_neighbors": range(2, 20),
              "VT__threshold": [0, 1]}
```

```
# 5-fold cross-validation
GSCV = GridSearchCV(model, parameters, cv=5)
GSCV.fit(X, y)
pass
```

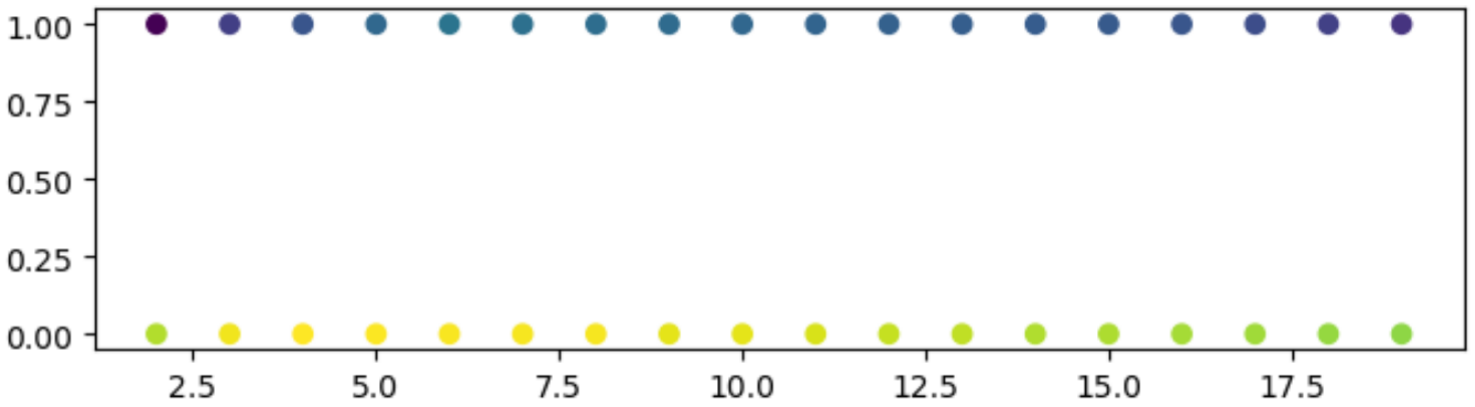
```
GSCV.best_params_
```

```
{'KNN__n_neighbors': 4, 'VT__threshold': 0}
```

```
pred = GSCV.predict(X) # GSCV is equal to the best estimator
```

Пример – Grid Search (Python)

```
plt.scatter(*pd.DataFrame(GSCV.cv_results_["params"]).T.values, c=GSCV.cv_results_["mean_test_score"])
plt.gcf().set_size_inches(8, 2)
```



Пример – Случайный поиск (Python)

```
from sklearn.model_selection import RandomizedSearchCV
from scipy.stats import randint, uniform
```

```
model = Pipeline([("scaler", scaler),
                  ("VT", VarianceThreshold()),
                  ("KNN", KNeighborsRegressor())])
```

```
distributions = {"KNN__n_neighbors": randint(2, 20),
                 "VT__threshold": uniform(0, 1)}
```

```
# 5-fold cross-validation
RSCV = RandomizedSearchCV(model, distributions, n_iter=20,
                          cv=5, random_state=123)
RSCV.fit(X, y)
pass
```

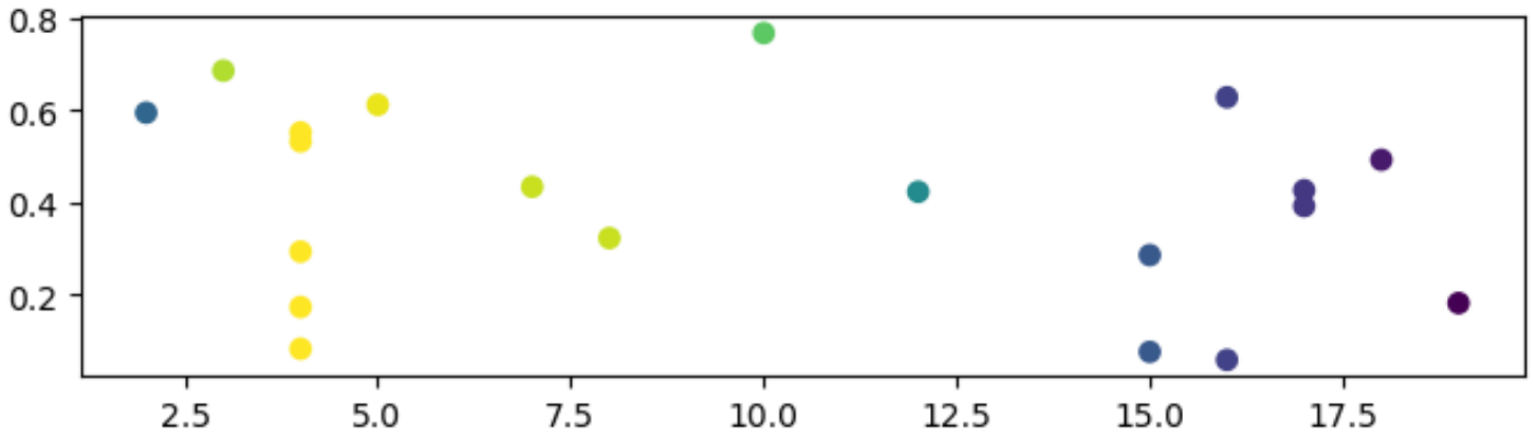
Пример – Случайный поиск (Python)

```
RSCV.best_params_
```

```
{'KNN__n_neighbors': 4, 'VT__threshold': 0.5513147690828912}
```

```
pred = RSCV.predict(X) # RSCV is equal to the best estimator
```

```
plt.scatter(*pd.DataFrame(RSCV.cv_results_["params"]).T.values, c=RSCV.cv_results_["mean_test_score"])
plt.gcf().set_size_inches(8, 2)
```



Пример – Отбор (Python)

```
from sklearn.experimental import enable_halving_search_cv # Required import
from sklearn.model_selection import HalvingGridSearchCV, HalvingRandomSearchCV
```

```
model = Pipeline([("scaler", scaler),
                  ("VT", VarianceThreshold()),
                  ("KNN", KNeighborsRegressor())])
```

```
distributions = {"KNN__n_neighbors": randint(2, 20),
                 "VT__threshold": uniform(0, 1)}
```

```
HRSV = HalvingRandomSearchCV(model, distributions, cv=5,
                             factor=2, # Candidate selection cut-off
                             # Resource increasing during selection:
                             resource="n_samples",
                             min_resources=100)
HRSV.fit(X, y)
pass
```

Пример – Отбор (Python)

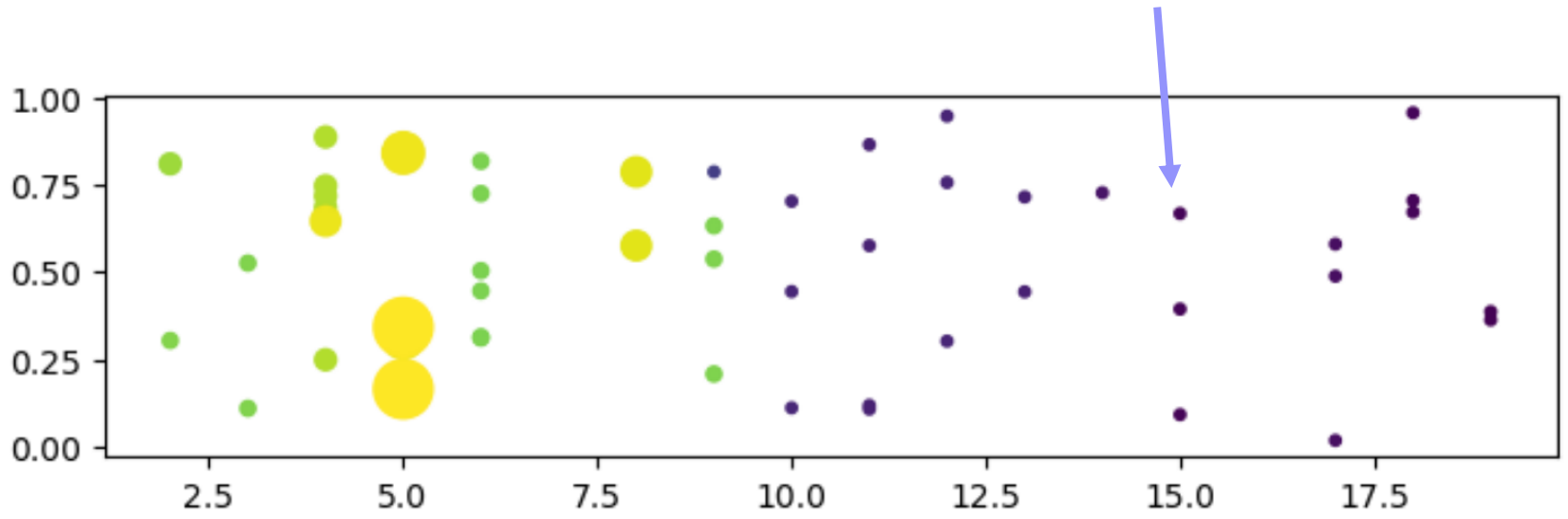
```
HRSV.best_params_
```

```
{'KNN__n_neighbors': 5, 'VT__threshold': 0.3417047325655804}
```

```
pred = HRSV.predict(X) # RSCV is equal to the best estimator
```

```
plt.scatter(*pd.DataFrame(HRSV.cv_results_["params"]).T.values,
            c=HRSV.cv_results_["mean_test_score"],
           s=HRSV.cv_results_["n_resources"]/10)
plt.gcf().set_size_inches(8, 2)
```

Размер отвечает за число семплов



Бутстрэпинг

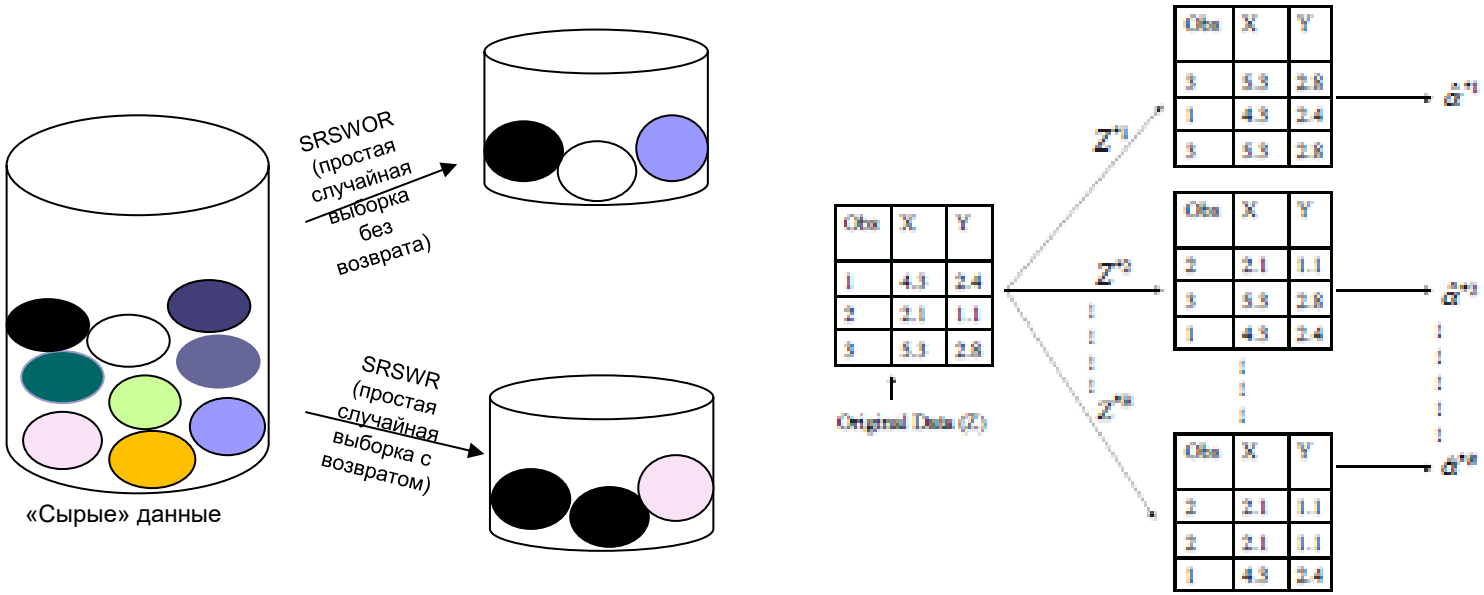
- *Бутстрэппинг* представляет собой мощный статистический инструмент, который может быть использован для количественной оценки неопределенности.
- Например, он может позволить произвести оценку стандартной ошибки коэффициента или доверительного интервала для этого коэффициента.
- Использование термина бутстреппинг происходит от фразы, чтобы *to pull oneself up by one's bootstraps*, - «пример» из книги «Удивительные приключения барона Мюнхгаузена»

Барон упал на дно глубокого озера. Когда казалось, что все было потеряно, он решил вытащить себя своими собственными силами.

Бутстрэппинг

- Подход бутстрэппинга позволяет имитировать процесс получения новых случайных наборов данных, так что мы можем оценить дисперсию нашей оценки, не создавая дополнительных образцов.
- Вместо того, чтобы постоянно получать независимые наборы данных, мы получаем различные наборы путем многократной выборки наблюдений из исходного набора *с замещением* (или *с возвращением*).
- Каждый из этих "наборов данных" создается путем выборки *с замещением* и имеет *такой же размер* как наш исходный набор данных. В результате некоторые наблюдения могут появляться более одного раза в наборе данных бутстреппинга, а некоторые нет вообще.

Демонстрационный пример



- Графическая иллюстрация бутсреппингового подхода на маленькой выборке
- Каждый бутсреппинговый набор данных содержит наблюдения, отобранные с заменой из исходного набора.
- Каждый такой набор данных начальной используется для получения оценки

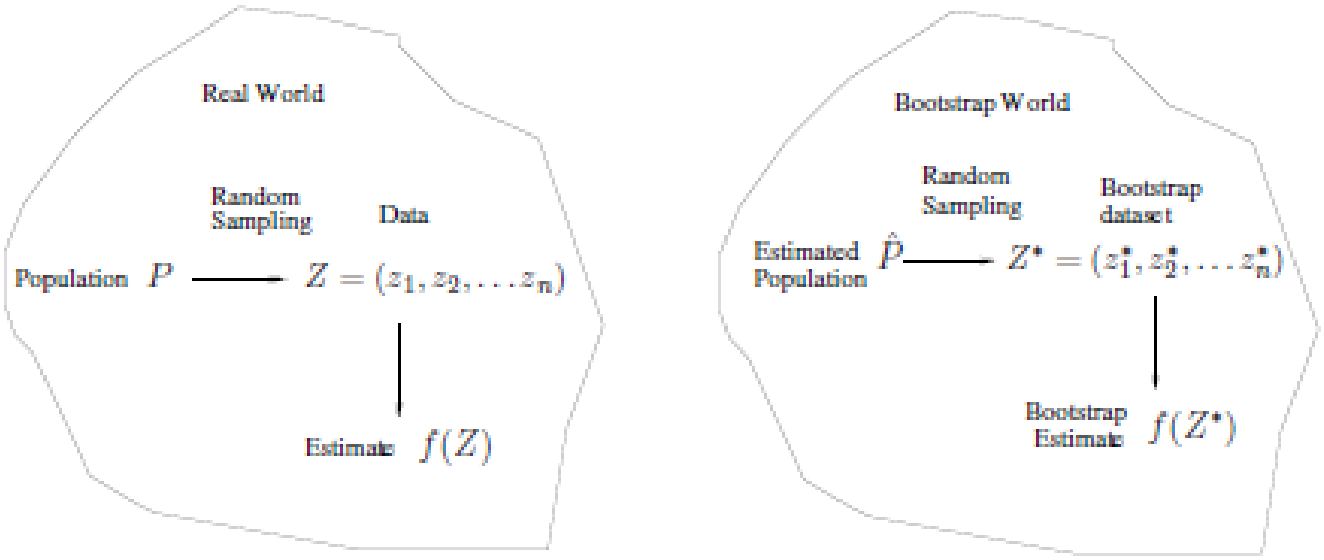
Бутстрэпинг

- Обозначая k-й набор данных бутстреппинга как Z^{*k}, мы используем его, чтобы выполнить новую оценку для a^{*k}
- Эта процедура повторяется *B* раз для некоторого большого значения *B* (например, 100 или 1000), чтобы получить *B* различных наборов данных бутстреппинга $Z^{*1}, Z^{*2}, \ldots, Z^{*B}$, и *B* соответствующих оценок $a^{*1}, a^{*2}, \ldots, a^{*B}$
- Оценим среднее и стандартную ошибку этих оценок бутстреппинга:

$$\hat{a} = \frac{1}{B}\sum_{r=1}^{B}(a^{*r}), SE_B(a) = \sqrt{\frac{1}{B-1}\sum_{r=1}^{B}(\hat{a} - a^{*r})^2}$$

- Они служат в качестве оценки, полученной на тестовом наборе данных.

Общая схема бутсрепинга



- В более сложных ситуациях, определение подходящего способа для получения выборок бутстрепинга может потребовать значительных усилий.
- Например, если данные представляют собой временные ряды, мы не можем просто выбирать наблюдения с замещением

Как бутстрепинг оценивает ошибку прогнозирования

- При кросс-валидации каждый из *K* блоков валидации отличается от других *K - 1*, используемых для обучения: *перекрытия нет*.
- Для оценки ошибки прогнозирования с помощью бутстреппинга мы могли бы использовать каждый набор данных бутстреппинга как валидационный набор для остальных (или наоборот), но:
  - каждая выборка бутстрепинга имеет значительное перекрытие с исходным набором (около двух третей).
  - это приведет бутстреппинг к существенному недооцениванию истинной ошибки прогнозирования
- Удаление перекрытия (*out of bag*) - можно частично решить эту проблему, используя для оценки только те наблюдения, которые не появились (случайно) в текущей выборке бутстреппинга $OOB(\mu) = \sum_{k=1}^{K} \frac{l_k}{l} Q(\mu(Z^{*k}), Z \backslash Z^{*k})$

Пример (Python)

```
ITER = 100
SAMPLES = 100
frame = []
for i in range(ITER):
    sample = resample(X, replace=True, n_samples=SAMPLES, stratify=None)
    stat = sample["HouseAge"].mean()
    frame.append(stat)
frame = np.array(frame).flatten()
frame = pd.Series(frame).sort_values()
```

Доверительные интервалы 90% для среднего возраста жилища:

```
frame.quantile(0.05), frame.quantile(0.95)
```

```
(26.248, 30.6515)
```

Бутстреп-регрессия (Python)

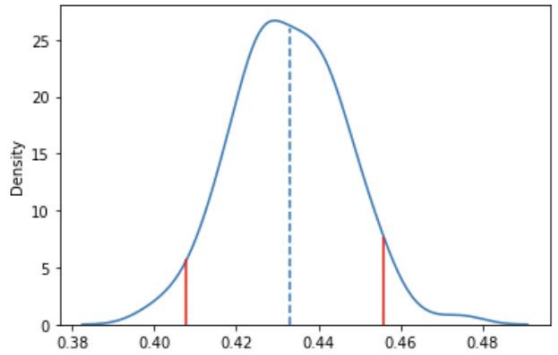
```
from sklearn.ensemble import BaggingRegressor
```

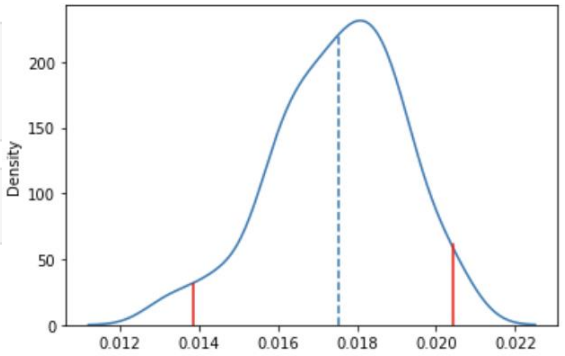
```
estimator = BaggingRegressor(LinearRegression(), n_estimators=100
                             bootstrap=True, max_samples=0.1,
                             random_state=42)
```

```
features = X[["MedInc", "HouseAge"]]
```

```
estimator.fit(features, y)
pass
```

```
coefs = np.array([x.coef_ for x in estimator.estimators_])
```






```
sns.kdeplot(data=coefs[:, 0])
plt.axvline(coefs[:, 0].mean(), 0, 0.93, ls="--")
plt.axvline(np.quantile(coefs[:, 0],0.025), 0, 0.20,c="red")
plt.axvline(np.quantile(coefs[:, 0],0.975), 0, 0.27,c="red")
```

```
sns.kdeplot(data=coefs[:, 1])
plt.axvline(coefs[:, 1].mean(), 0, 0.91, ls="--")
plt.axvline(np.quantile(coefs[:, 1],0.025), 0, 0.13,c="red")
plt.axvline(np.quantile(coefs[:, 1],0.975), 0, 0.25,c="red")
pass
```

Бутстреп-регрессия (Python)

```
pred = np.array([x.predict(features.values) for x in estimator.estimators_]).T
pred.shape
```

```
(20640, 100)
```

```
plt.plot(np.percentile(pred, q=5, axis=1)[:25], c="blue")
plt.plot(np.percentile(pred, q=95, axis=1)[:25], c="red")
plt.scatter(range(25), estimator.predict(features)[:25], c="green")
pass
```