

Летняя школа по анализу данных

Задача классификации

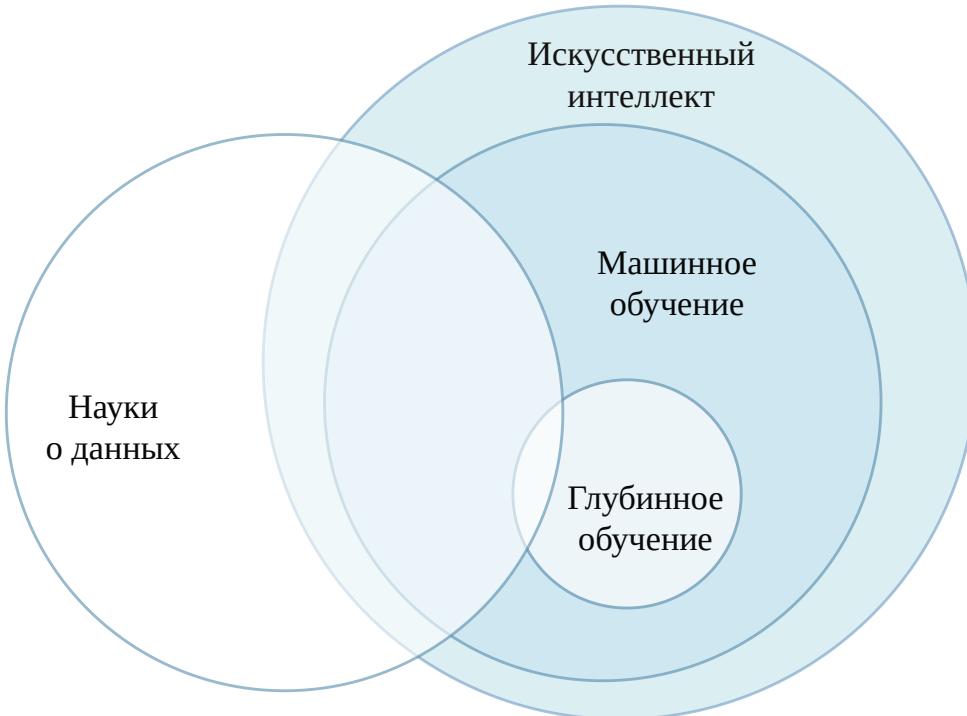
Алексей Болдырев

25/06/2024

Примеры классификации

1. В отделение неотложной помощи поступает пациент с набором симптомов, которые могут быть отнесены к одному из трех медицинских заболеваний. Какое из этих заболеваний в действительности у человека?
2. Вам приходит сообщение в ТГ. Бот-классификатор, определяет, нужно ли оно вам, или это спам.
3. Рекомендательная система предлагает вам прочитать учебник по машинному обучению в зависимости от списка учебников, которые вы уже прочитали.

Что такое машинное обучение?



Машинное обучение
находится на перекрестке:

- Математической статистики
- Теории вероятностей
- Математического анализа
- Линейной алгебры
- ...
- **Предметной области**

Машинное обучение находится на перекрестке

- **Математической статистики и теории вероятности**
 - Оценка неопределенности, представление зашумленных данных
- **Линейной алгебры**
 - Компактное представление линейных преобразований данных
 - Методы уменьшения размерности
- **Других отраслей математики**
 - Теория оптимизации, гарантии сходимости и аппроксимации
- Кроме того требуется **опыт работы в конкретной области**
 - Физика, лингвистика, генетика, компьютерное зрение, FinTech, ...

Зачем в машинном обучении нужна математика?

Предположим, ваша модель показывает **81%** точности

И что теперь?

- Это хорошо/плохо, полезно/бесполезно?
- Можно ли это улучшить?
- Если да, то до какого предела улучшать?

Статистическая теория дает ответы на эти вопросы

Зачем в машинном обучении нужна математика?

Статистическая теория предоставляет:

- Выбрать:
 - **правильный алгоритм** для решения задачи
 - **лучшие гиперпараметры**
 - **стратегии валидации** (проверки) модели
- Распознавать и устранять **недообучение и переобучение**
- **Устранять проблемы** с плохими и неоднозначными результатами
- Оценивать неопределенность прогноза
- **Оптимизировать алгоритмы** и строить эффективный программный код
 - Модели должны быть (экономически) жизнеспособными

Что такое машинное обучение?

Это фреймворк для
(статистического) нахождения
неизвестных функций из наблюдений.

Что такое машинное обучение? Пример

Цель - увеличить продажи некоторого продукта в 200 случаях

- У нас есть бюджет на рекламу в телевидении, радио и газетах (признаки)
- Необходимо определить взаимосвязь с бюджетом на рекламу и продажами
- Какой признак сильнее всего влияет на целевую переменную?
- Каким образом распределить бюджет, чтобы повысить продажи?

Нужно найти функцию, лучше всего связывающую y с X



переменные

признаки

цель

TV radio newspaper sales

	переменные	признаки	цель	
	TV	radio	newspaper	sales
0	230.1	37.8	69.2	22.1
1	44.5	39.3	45.1	10.4
2	17.2	45.9	69.3	9.3
...
197	177.0	9.3	6.4	12.8
198	283.6	42.0	66.2	25.5
199	232.1	8.6	8.7	13.4

Наблюдения

200 rows × 4 columns

Что такое машинное обучение?

В общем виде:

- Пусть X будет **матрицей данных** с N наблюдениями (точками, случаями) и p признаками (предикторами, фичами):

$$X = [X_1, X_2, \dots, X_p] \in \mathbb{R}^{N \times p}$$

- Пусть Y будет вектором **ответов** с N соответствующими метками: $Y \in \mathbb{R}^N$
- Мы хотим найти **фиксированную**, но **неизвестную** функцию f :

$$Y = f(X) + \varepsilon$$

- Где ε это **независимая** случайная величина со следующими свойствами:

$$\varepsilon \perp X_i, \quad \mathbb{E}\varepsilon = 0, \quad \mathbb{V}\varepsilon < \infty$$

Зачем оценивать f ?

- Предсказание
 - \hat{f} это неинтерпретируемый **черный ящик**
 - Мы ищем наилучшие **предсказания** для имеющихся X
 - Цель - уменьшить **устранимую** ошибку
- Вывод (inference)
 - \hat{f} это интерпретируемый **белый ящик**
 - Мы хотим:
 - понять, насколько **чувствителен** Y к изменениям X
 - найти **важные** признаки
 - определить, является ли f **линейной** или более сложной функцией

Как нам оценить f ?

- Большинство методов машинного обучения оценивают \hat{f} из **обучающих наблюдений** $(x'_i, y'_i) \in \mathbb{R}^{p+1}$
 - x'_i - i -й вектор значений признаков

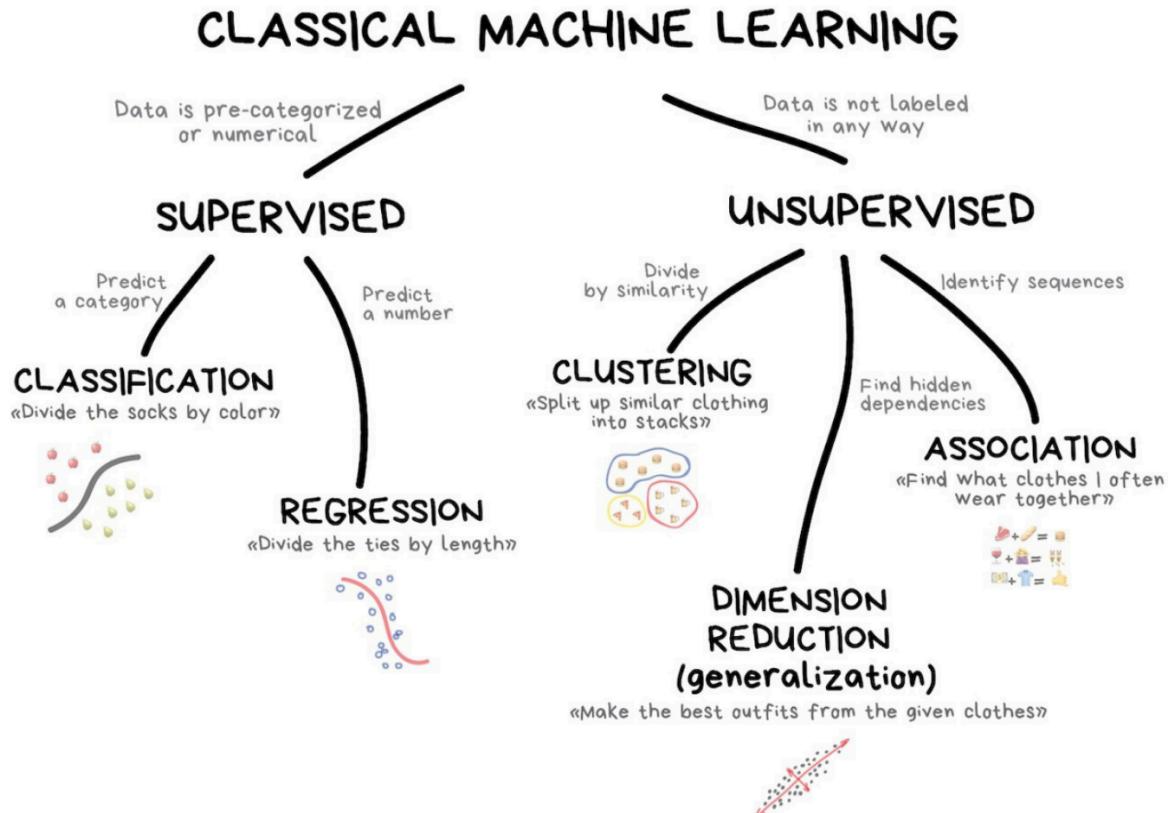
Вектор признаков	X_1 TV	X_2 radio	X_3 newspaper	y sales
230.1	37.8	69.2	$\} x'_1$	22.1
44.5	39.3	45.1	$x_{1:}$	10.4
17.2	45.9	69.3	$x_{1:}$	9.3
...	\dots	...
177.0	9.3	6.4	$x_{2:}$	12.8
283.6	42.0	66.2	$x'_{2:}$	25.5
232.1	8.6	8.7	$\} x'_{2:}$	13.4

Транспонированный вектор значений признаков

(транспонированный) вектор наблюдений

200 rows × 4 columns

Ландшафт проблем машинного обучения

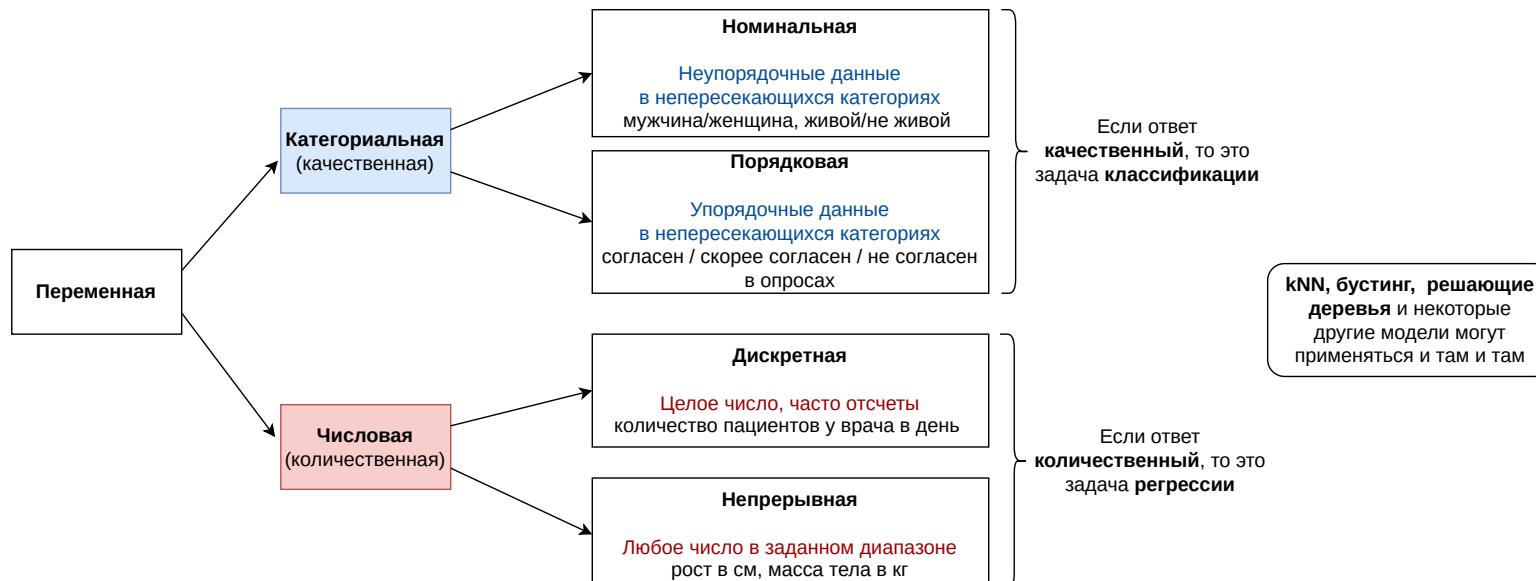


Обучение с учителем или обучение без учителя

- **Обучение с учителем:** каждое наблюдение сопровождается ответами (метками), y
 - Мы хотим соотнести:
 - строки наблюдений, x_i , с их предсказаниями, y_i
или
 - столбцы признаков, X_i , со столбцом ответов
- **Обучение без учителя:** без ответов (меток)
 - Мы ищем соотношения между
 - столбцами признаков, X_i
 - Группируем похожие признаки вместе (PCA, NMF, Gaussian Mixture)
 - строками наблюдений, x_i с x_j
 - Группируем похожие наблюдения (kMeans, DBScan, HAC)

Задачи регрессии и классификации

- Категориальные переменные принимают значения в k классах (категориях)
 - Если такие переменные порядковые, то можно это использовать
 - Часто мы можем перейти от одной задачи к другой



Основные принципы выбора модели

Итак, существует множество моделей и множество типов проблем, к тому же бесплатного завтрака не бывает.

Так как же выбрать лучшую для нашей конкретной задачи?

Есть несколько ключевых выборов / компромиссов, которые необходимо сделать:

- Параметрическая модель или непараметрическая
- Настраиваемая или интерпретируемая модель
 - решение задач предсказания или вывода, соответственно
- Модель с малым смещением или с малой дисперсией?

Параметрические модели

Используют функции с **конечным** числом параметров

- Шаг 1: **Зафиксируем семейство** параметрических моделей \mathcal{F}
 - Также мы определяем размерность пространства параметров
- Шаг 2: **Зададим модель** на тренировочных наблюдениях
 - Т.е. оценить параметры модели (с помощью м.н.к., градиентного спуска, ...)

Пример:

- Возьмем **логистическую регрессию** с $p + 1$ параметрами

$$\log \left(\frac{p(X)}{1 - p(X)} \right) = \beta_0 + \beta_1 X$$

- Оценим коэффициенты регрессии β с помощью

метода максимального правдоподобия: $\ell(\beta_0, \beta_1) = \prod_{i:y_i=1} p(x_i) \prod_{j:y_j=1} (1 - p(x_j))$

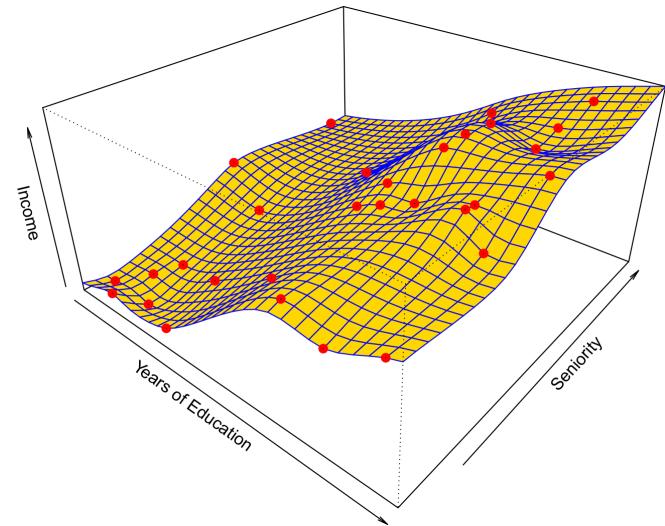
Параметрические модели

Свойства параметрических моделей:

- **Переобученная** модель запоминает шум в обучающих наблюдениях и не может хорошо обобщить новые наблюдения
 - Хорошо работает на обучающих данных, но плохо - на новых (тестовых)
- **Недообученная** модель не может отразить основные взаимосвязи в данных
 - Плохо работает на обучающих и новых данных
- Сложные модели (например, деревья решений или нейронные сети) склонны переобучаться, если их не контролировать и не использовать регуляризацию

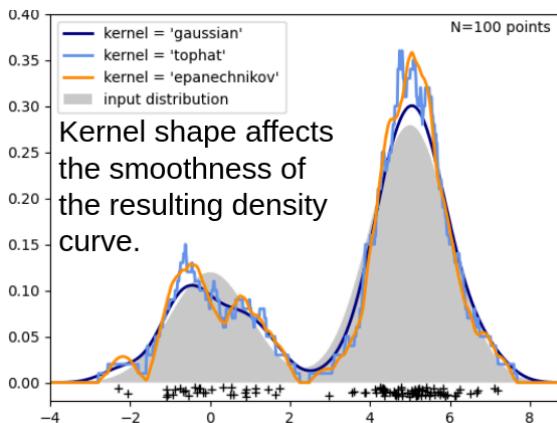
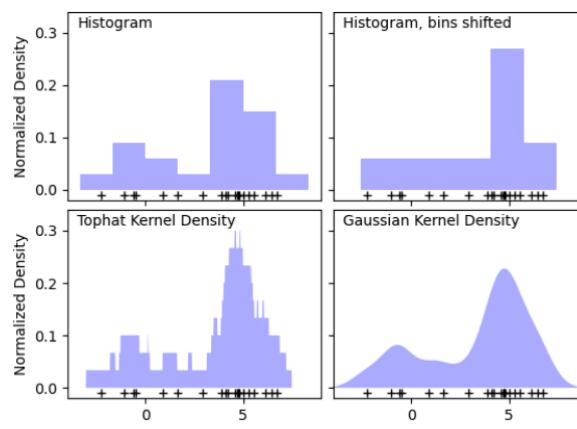
Непараметрические модели

- Мы не выбираем семейство функций
- У таких моделей **нет** параметров обучения или их количество **переменное**
 - Исследователю по-прежнему доступны **гиперпараметры** (не обучаются)
- Могут производить гладкие функции
 - Можно контролировать их гладкость



Непараметрические модели: примеры

- Классификатор k-ближайших соседей (**kNN**)
- Гистограмма, определяющая функцию плотности распределения
- Определение плотности ядра (**KDE**)
- Метод опорных векторов (**SVM**)
- Непараметрическая логистическая регрессия



Непараметрические модели и индуктивная предвзятость

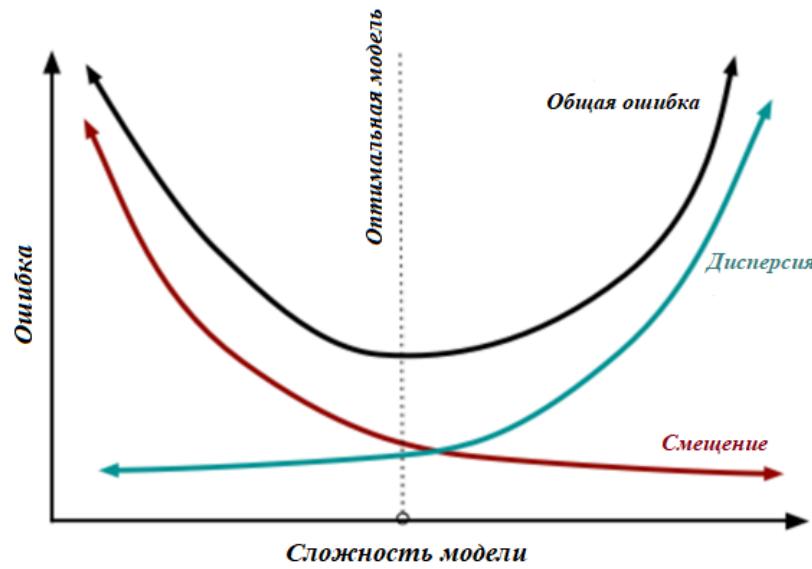
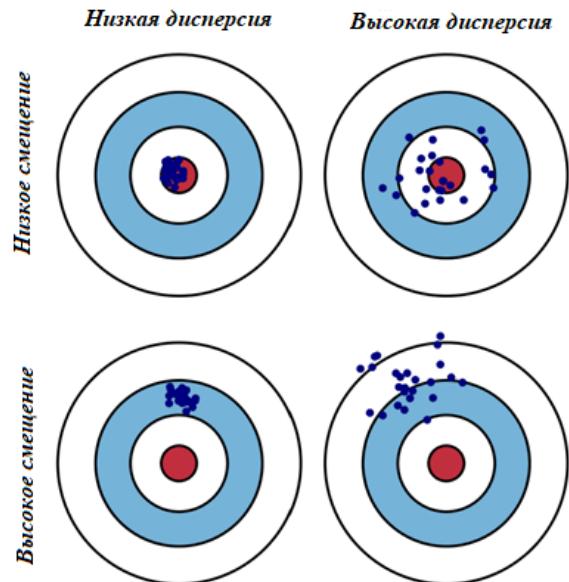
Непараметрические модели кажутся лучшей идеей - почему мы не можем использовать их повсеместно?

- Это потому, что у них все еще есть некоторые предположения относительно данных, или **индуктивная предвзятость**
- В **параметрических** моделях мы делаем **явные** предположения об истинном значении f
- В **непараметрических** моделях мы делаем **неявные** предположения

Настраиваемость или интерпретируемость модели



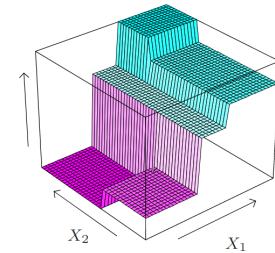
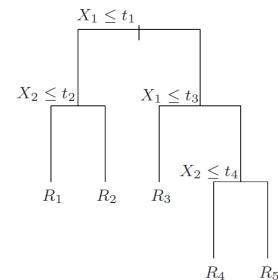
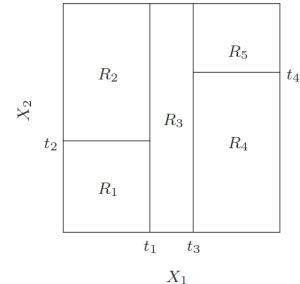
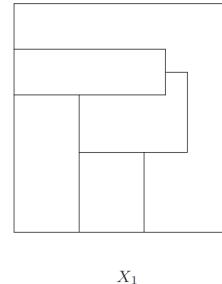
Смещение или дисперсия?



Деревья решений

Деревья решений

- Мы делим пространство признаков на **прямоугольные области** $\{R_m\}_{1:M}$ и присваиваем c_m каждой из них
- Используем **бинарное рекурсивное разделение**
- **Разделение** делается исходя из “лучшего” различия получившегося
 - “лучшее” означает:
 - В регрессии:
 - Наименьшая общая ошибка в каждой ветви
 - В классификации:
 - **Самые чистые классы в каждой ветви**



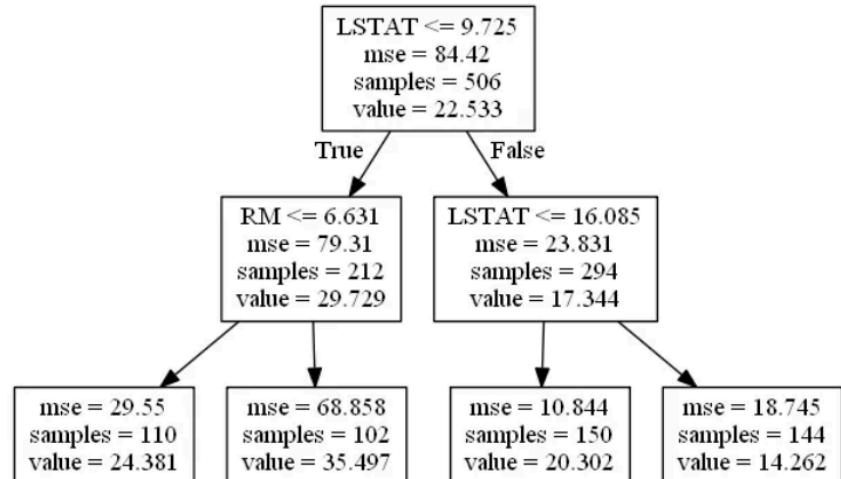
Источник: [ESL Fig. 9.2](#)

Построение деревьев регрессии

- Выбираем $\{R_m\}$ для получения минимального RSS в каждой области R_m
- Минимизация $\hat{c}_m := \bar{y}|_{R_m} = \frac{1}{N_m} \sum_{x_i \in R_m} y_i$
 - усредненный ответ на R_m
 - $N_m := \#\{x_i \in R_m\}$
 - Поиск всевозможных комбинаций $\{R_m\}$ вычислительно труден
- "Жадный" способ:
 - В каждом узле мы ищем разделяющую переменную j и точку s для получения минимального RSS: $\min_{j,s} [\min_{c_1} \text{RSS}_{\text{Left}}(j, s) + \min_{c_2} \text{RSS}_{\text{Right}}(j, s)] = \min_{j,s} [\min_{c_1} \sum_{x_i \in R_1(j,s)} (y_i - c_1)^2 + \min_{c_2} \sum_{x_i \in R_2(j,s)} (y_i - c_2)^2]$
 - Повторяем разделение на каждой получившейся области
 - Без дополнительных ограничений мы получаем переобучение

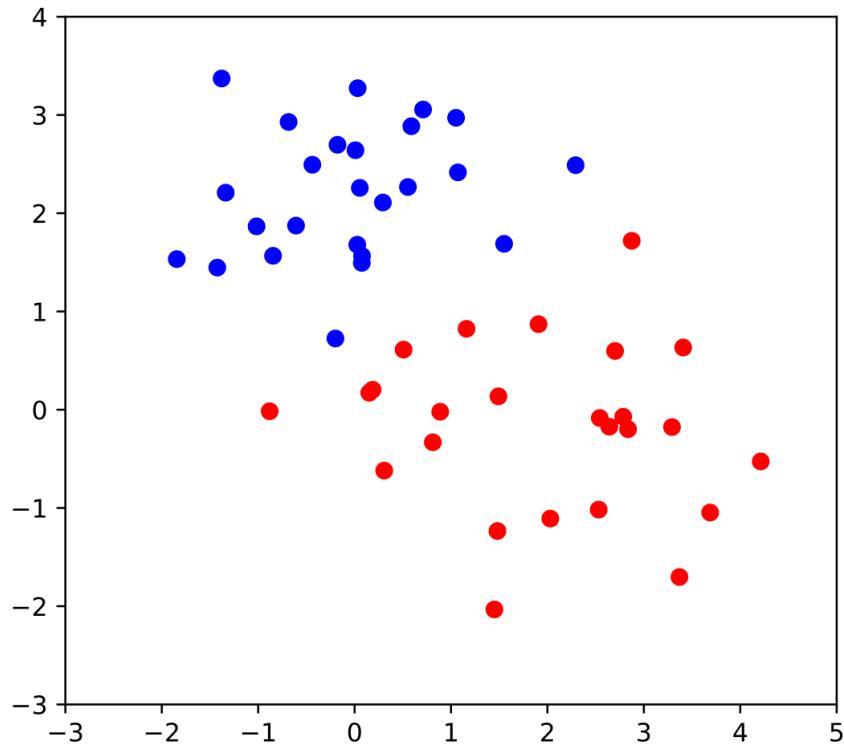
Контроль переобучения

- Один из подходов:
 - Рассмотрим порог τ
 - Если $\Delta RSS > \tau$, продолжаем разделение
- ΔRSS (или ΔMSE) могут возрасти или уменьшиться с ростом дерева
 - Мы должны предусмотреть остановку алгоритма
- Пример:
 - Если $\tau_{MSE} = 6$, тогда левая ветвь не построится
 - Но MSE уменьшится на 30.5

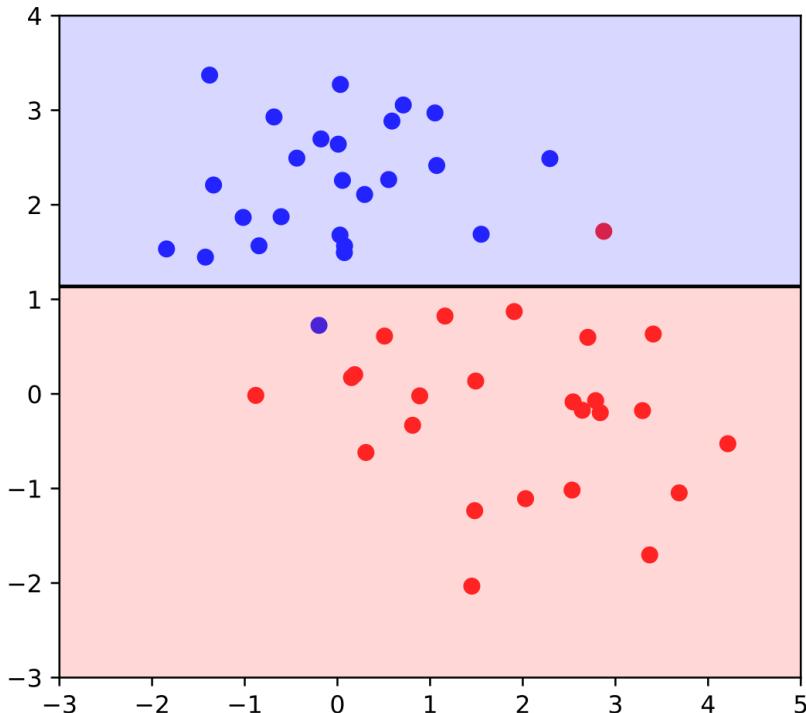


Источник: <https://towardsdatascience.com/decision-trees-explained-3ec41632ceb6>

Построение деревьев решений



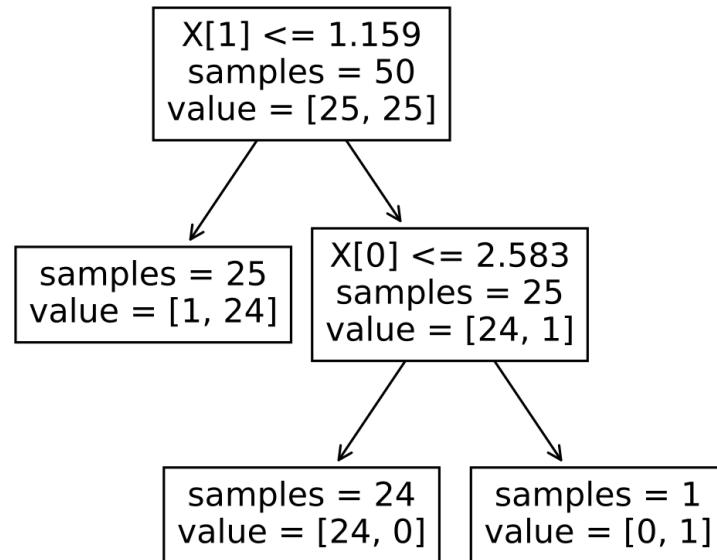
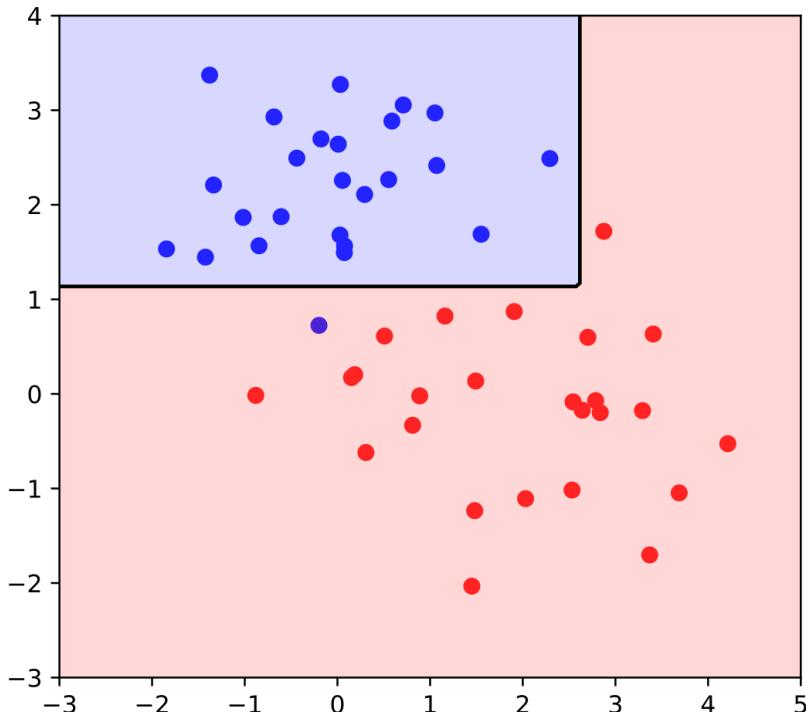
Построение деревьев решений



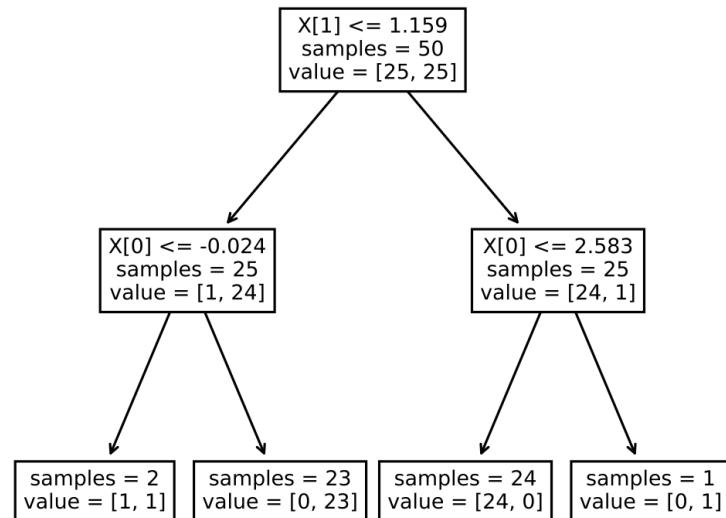
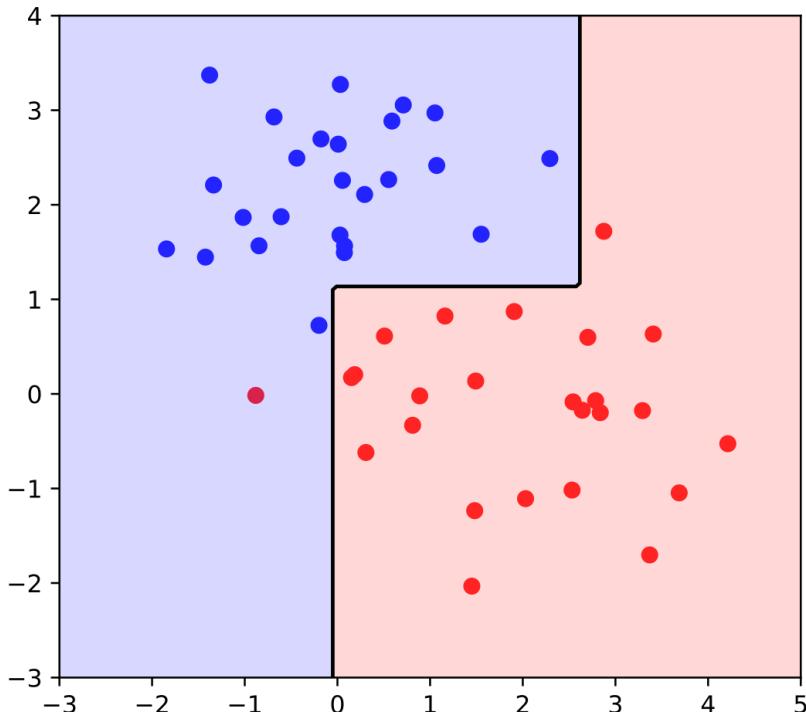
$X[1] \leq 1.159$
samples = 50
value = [25, 25]

samples = 25
value = [1, 24]
samples = 25
value = [24, 1]

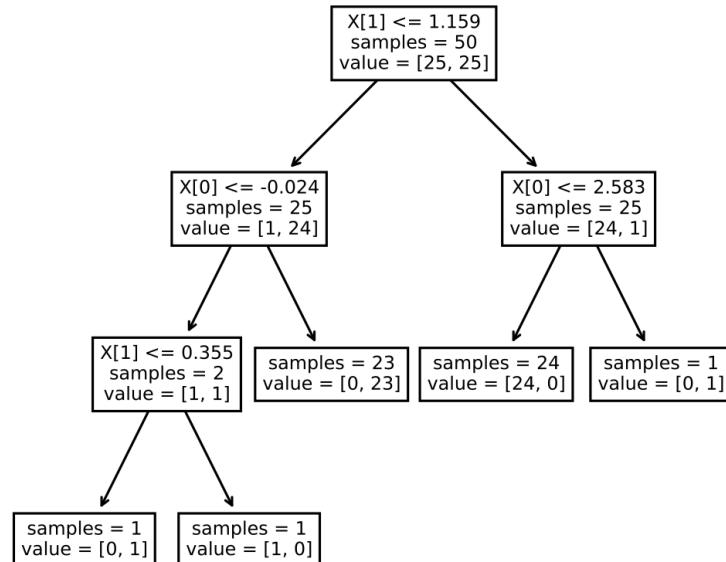
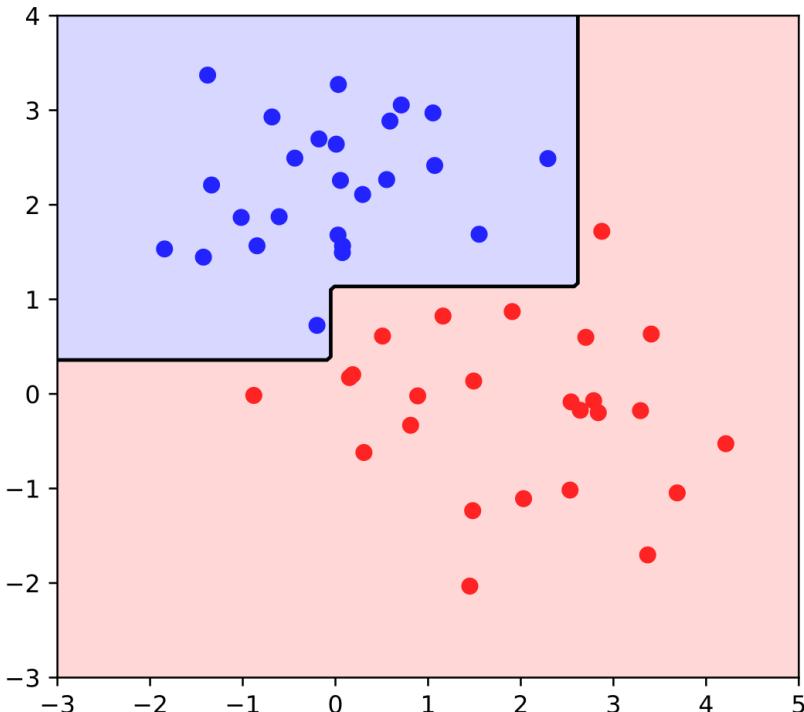
Построение деревьев решений



Построение деревьев решений



Построение деревьев решений



Демо различных методов классификации

Демо различных методов классификации

Метрики качества классификации

Матрица ошибок

Рассмотрим задачу классификации с 4 классами

		Ожидаемый класс			
		1	2	3	4
Предсказанный класс	1	52	3	7	2
	2	2	28	2	0
	3	5	2	25	12
	4	1	1	9	40

В матрице:

- Значение - число объектов j -го класса, которые были предсказаны как i -й класс
- Диагональные элементы - **верные** классификации
- Внедиагональные элементы - **неверные** классификации

Бинарная классификация

Рассмотрим задачу бинарной классификации с классами + и -

		Ожидаемый класс	
		+	-
Предсказанный класс	+	TP (True Positive)	FP (False Positive)
	-	FN (False Negative)	TN (True Negative)

Бинарная классификация

Рассмотрим задачу бинарной классификации с классами + и -

		Ожидаемый класс		Метрики качества:	
		+	-	Accuracy = $\frac{TP + TN}{TP + TN + FP + FN}$	
Предсказанный класс	+	TP (True Positive)	FP (False Positive)		
	-	FN (False Negative)	TN (True Negative)		

Бинарная классификация

Рассмотрим задачу бинарной классификации с классами + и -

Ожидаемый класс	
Предсказанный класс	+
	-
+	TP (True Positive)
-	FP (False Positive) FN (False Negative)
-	TN (True Negative)

Метрики качества:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

True positive rate, TPR, Recall =

$$\frac{\text{TP}}{\text{TP} + \text{FN}}$$

Бинарная классификация

Рассмотрим задачу бинарной классификации с классами + и -

		Ожидаемый класс	
		+	-
Предсказанный класс	+	TP (True Positive)	FP (False Positive)
	-	FN (False Negative)	TN (True Negative)

Метрики качества:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

$$\text{True positive rate, TPR, Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{False positive rate, FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}$$

Бинарная классификация

Рассмотрим задачу бинарной классификации с классами + и -

		Ожидаемый класс	
		+	-
Предсказанный класс	+	TP (True Positive)	FP (False Positive)
	-	FN (False Negative)	TN (True Negative)

Метрики качества:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

$$\text{True positive rate, TPR, Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{False positive rate, FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Бинарная классификация

Рассмотрим задачу бинарной классификации с классами + и -

		Ожидаемый класс	
		+	-
Предсказанный класс	+	TP (True Positive)	FP (False Positive)
	-	FN (False Negative)	TN (True Negative)

Метрики качества:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

$$\text{True positive rate, TPR, Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

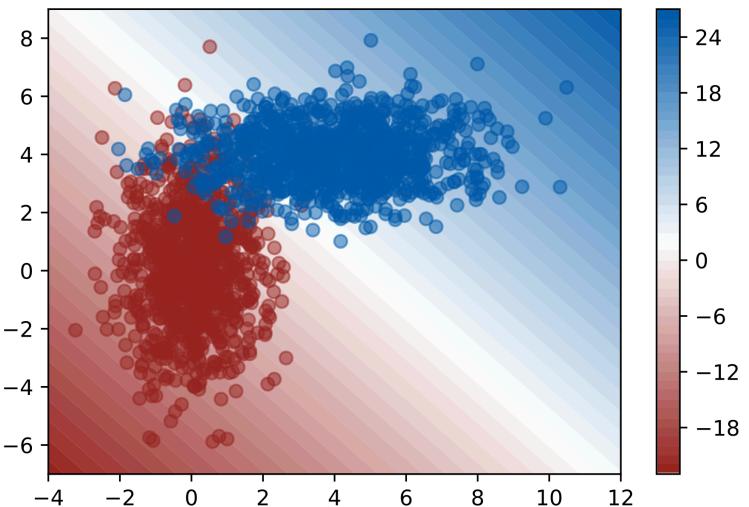
$$\text{False positive rate, FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

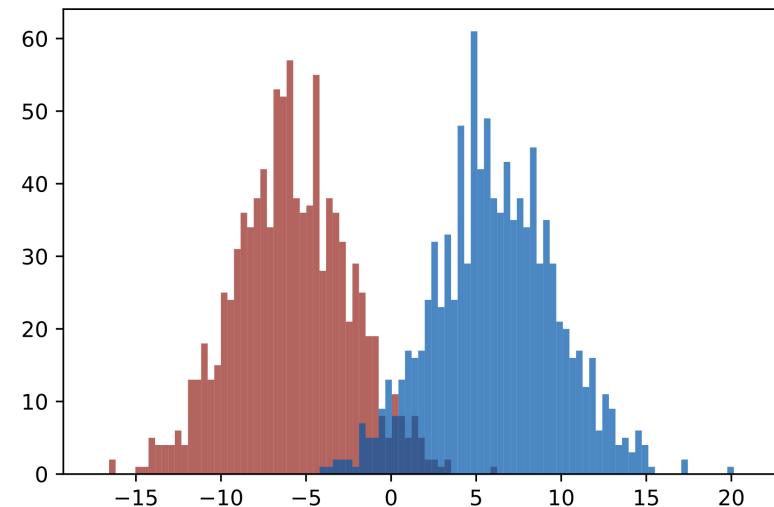
$$\text{F}_1\text{-score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Предсказание непрерывных значений

Многие алгоритмы классификации работают с непрерывными функциями оценки



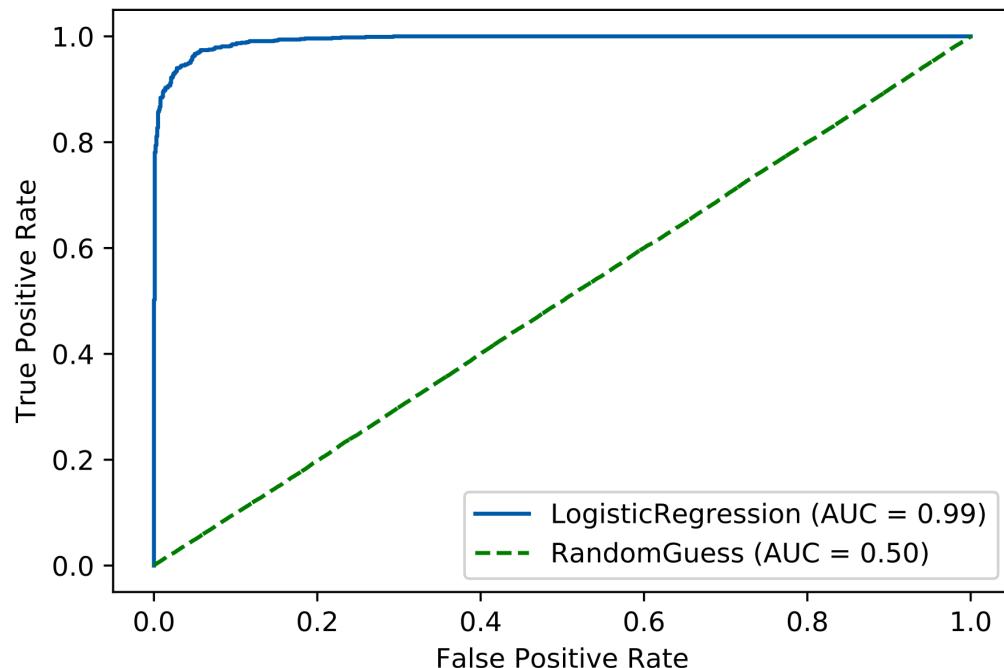
Предсказание модели классификации



Значение функции оценки в данных

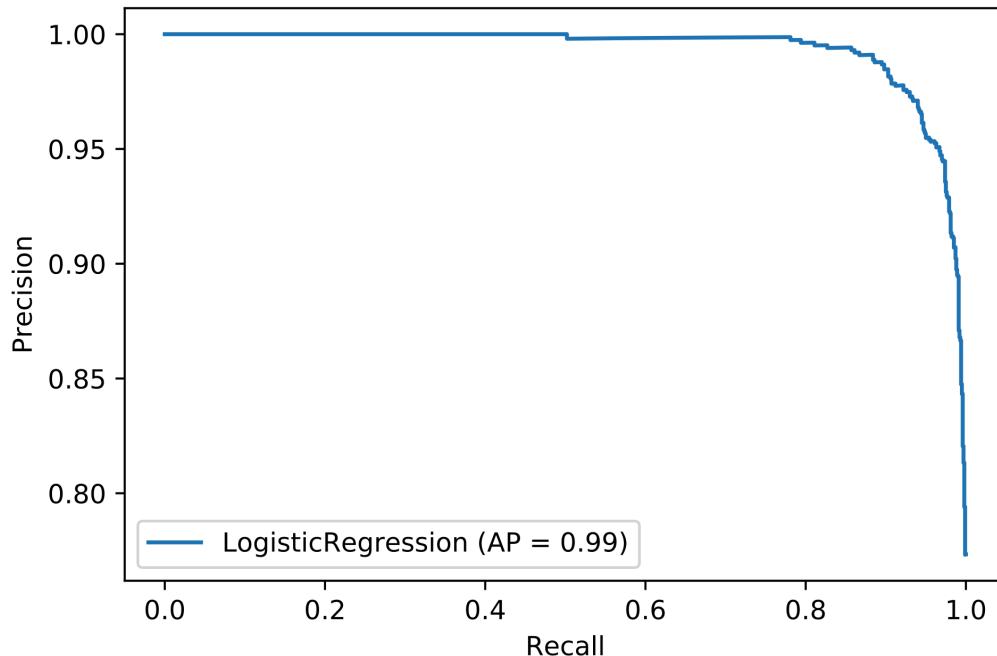
ROC-кривая

Receiver Operating Characteristic = TPR как функция FPR



Интерактивный пример

Кривая Precision-Recall



Кривая Precision-Recall показывает компромисс между Precision и Recall для различных пороговых значений. Большая площадь под кривой означает как высокое значение Precision (низкий уровень ложноположительных результатов), так и высокое значение Recall (низкий уровень ложноотрицательных результатов)

Пересемплирование

Что такое пересемплирование?

Это отбор образцов из выборки (как правило, многократный, с заменой)

Виды:

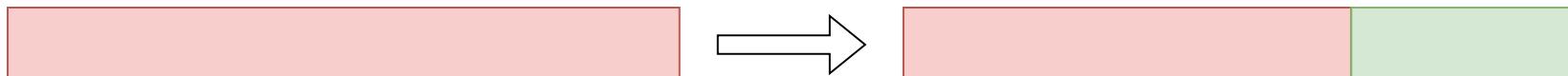
- **Валидация и Валидация+Тест**
- **Кросс-валидация (CV)** используется для
 - **выбора** лучшей модели или поиска ее **гиперпараметров**, а затем
 - для **оценки ошибки обобщения (тестовой ошибки)**
 - или оценки некоторой меры качества модели (R^2 , ...)
- **Bootstrap** используется для
 - анализа **распределения** некоторой *выборочной статистики*
 - Среднее значение, медиана, процентили, дисперсия, p -значение, доверительный интервал (CI), параметры модели и т.д.

Что такое пересемплирование?

- Каждое наблюдение участвует в обучении/тестировании модели много раз
 - В CV наблюдения используются $k = 5$ или 10 раз и до n раз
 - Мы повторно подгоняем новую модель или новый набор гиперпараметров k раз
- В бутстрэпе наблюдения могут быть использованы $B \sim$ тысячи раз
 - Мы повторно вычисляем рассматриваемую статистику B раз на каждой подвыборке
- Повторная выборка требует больших вычислительных затрат, но может быть распараллелена

Валидационный подход

- Мы **разбиваем** исходную выборку размером n на 2 подвыборки:
 - **Обучающая**, размером n_T , используемое для обучения модели
 - На которой модель находит истинные связи между входами и выходами (I/O)
 - **Валидационная** - используется для оценки работы модели на тестовых данных
- При этом "**тестовая ошибка**" - это теоретическая концепция, которая известна только в том случае, если:
 - мы знаем **полную совокупность** наблюдений
 - мы **симулировали** наблюдения и знаем базовое распределение
- Разделение на обучающую/тестовую выборки может быть **80/20, 70/30, 60/40, 50/50** (в % от исходной выборки)



Перемешивание и предположение об i.i.d.

- Если наблюдения действительно независимы, мы можем разделить выборку как угодно
 - Однако большинство реальных наблюдений имеют некоторую зависимость.
 - Например, собранные за определенное время, случайно упорядоченные по какому-то столбцу, сгруппированные по категориям и т.д.
- Чтобы модель не подстраивалась под локальные зависимости, **перемешивайте наблюдения перед разбиением!**
 - Это помогает **разрушить зависимость между соседними наблюдениями**, которые могут быть связаны между собой
- Некоторые модели хорошо подходят для использования этих локальных эффектов.
 - Например, модели временных рядов могут использовать автокорреляции и **сезонные эффекты**

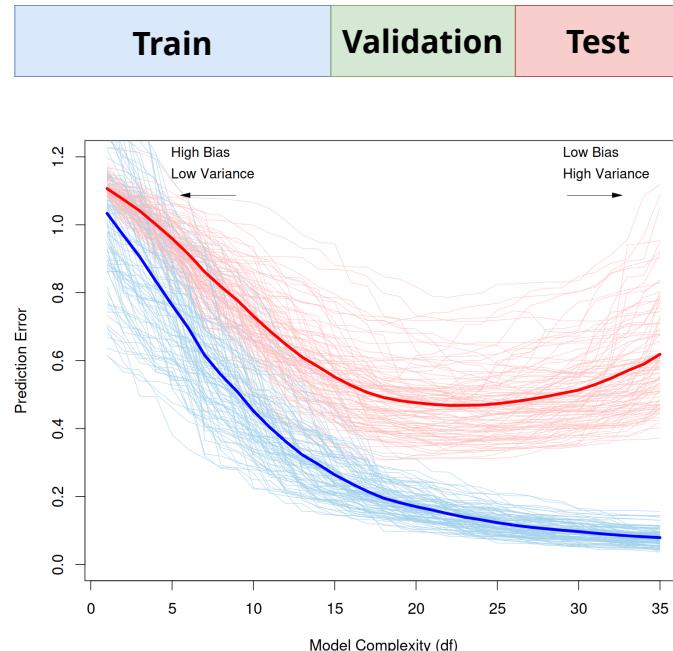
Валидационный+Тестовый подход

- **Большая** выборка может быть разделена на следующие подвыборки:
 - **Обучающую**: для подгонки наших моделей или одной модели с разными гиперпараметрами
 - **Валидационную**: для выбора лучшей модели
 - Мы оцениваем семейство моделей (регрессионные, kNN, ...), гиперпараметры, признаки, преобразования, ...
 - **Тестовую**: для оценки ошибки обобщения или итоговой ошибки
 - Использование только валидационного набора необъективно, так как мы уже знаем, что лучшая модель показала хорошие результаты
- Разделение зависит от конкретного случая, но может составлять **50/25/25**

Смещение, дисперсия и сложность модели

Цели:

1. **выбрать** модель, которая хорошо обобщает
 - т.е. дает малую ошибку на новых наблюдениях
 - Выполняется на **валидационной** выборке
 2. **измерить** ее качество (в виде ошибки или точности)
 - Выполняется на **тестовой** выборке
- Более сложные модели приводят к переобучению
(высокое смещение и низкая дисперсия)
- Сложность модели коррелирует с количеством степеней свободы и количеством оцениваемых параметров

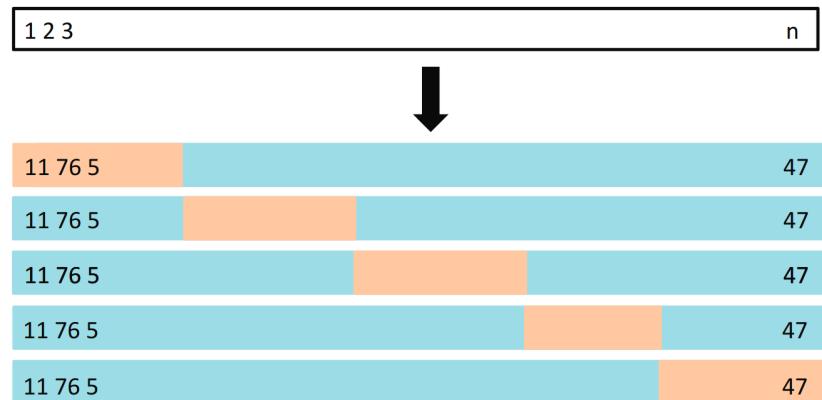


Источник изображения: ESL Fig. 7.1

Кросс-валидация (CV)

- В k CV мы разбиваем набор наблюдений на k разбиений
 - Каждое разбиение используется как валидационное ($v\text{MSE}_{d,i}$),
остальные $k - 1$ разбиений используются в обучении
- Каждый обучающий набор содержит $n_k \approx (k - 1)/k$ наблюдений
- $k = 5$ и до 10 - разумный баланс между 1 CV и n CV
- Оценка качества CV с k разбиениями:

$$\text{CV}_{d,k} := \frac{1}{k} \sum_{i=1}^k v\text{MSE}_{d,i}$$



(Не)правильный способ использования CV

■ Неправильный:

- Допустим, у нас есть $X_{N \times 1M}$ и мы делаем следующее:
 1. Находим наиболее объясняющие признаки $\tilde{X}_{N \times 100}$
 2. Применяем классификатор, используя \tilde{X}
 3. Используем CV для поиска гиперпараметров и оценки ошибки тестирования
- Проблема в том, что для выбора признаков мы использовали **ВСЕ** наблюдения (обучающие и тестовые)

■ Правильный:

- Выберем k разбиений и сделаем CV:
 1. Найдем наиболее объясняющие признаки $\tilde{X}_{N \times 100}$
 2. Построим классификатор, используя \tilde{X}
 3. Оценим ошибку тестовой выборки

Bootstrap

Используется для оценки неопределенности или распределения параметров модели

Минимальный пример: рассмотрим выборку $x = 1, 2, 3, 4$

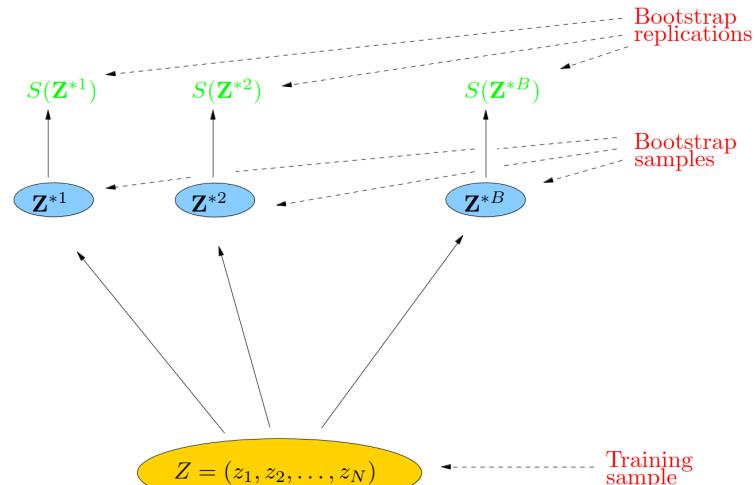
- Мы можем вычислить:
 - $\bar{x} = 2.5$ - среднее значение
 - $s_x \approx 1.29$ - дисперсия
 - $s_{\bar{x}} = \frac{s_x}{\sqrt{n}} \approx 0.65$ - стандартная ошибка среднего значения
- Каково распределение \bar{x} ?
 - Мы не можем оценить распределение, потому что у нас есть только одно наблюдение

Bootstrap

- Каково распределение \bar{x} ?
 - Статистическая теория утверждает, что \bar{x} приближается к гауссовскому со средним \bar{x} и средним отклонением $s_{\bar{x}}$
 - Во многих сложных ситуациях теория не может оценить распределение параметров и оценок
- Мы можем использовать **повторную выборку / пересемплирование**: сгенерировав B бутстррап-выборок z^{*1}, \dots, z^{*B}
 - Вычислим среднее, $\hat{\mu}^{*i}$, для каждого z^{*i}
 - Изучим распределение этих бутстррап-средних

Bootstrap

- Бутстрэпинг - это повторная выборка с заменой
 - Она помогает оценить распределение оценок, которые в противном случае наблюдались бы только один раз
- Мы берем B бутстрэп-выборок, $\{\mathbf{Z}^{*b}\}_{1:B}$ из $\mathbf{Z} = \{z_i\}_{1:N}, z_i := (x_i, y_i)$
- Пусть $S(\mathbf{Z})$ - любая интересующая нас статистика ($\mathbb{E}z, \mathbb{V}z, \hat{y}, \beta, \dots$)
- С помощью бутстрэп-выборок мы можем оценить распределение $S(\mathbf{Z})$
$$\hat{\mathbb{V}}S(\mathbf{Z}) := \frac{1}{B-1} \sum_{b=1}^B [S(\mathbf{Z}^{*b}) - \bar{S}]^2$$



Пример Bootstrap

```
import numpy as np
np.random.seed(1)
x = [1,2,3,4] # исходная выборка

B, n = 10, len(x) # число бутстррап-образцов и их размер
bootstrap = lambda x, n=len(x): np.random.choice(x, n)
z = np.array(bootstrap(x, n*B)).reshape((B,n))
# z = np.array([bootstrap(x) for i in range(B)]) # медленее
z
>> array([[2, 4, 1, 1], [4, 2, 4, 2], [4, 1, 1, 2], [1, 4, 2, 1], [3, 2, 3, 1], \
[3, 2, 3, 1], [4, 1, 3, 1], [2, 3, 3, 1], [4, 4, 2, 2], [4, 3, 1, 3]])

mu = np.mean(z, axis=1) # вычисление среднего значения для каждого образца
mu
>> array([2. , 3. , 2. , 2. , 2.25, 2.25, 2.25, 2.25, 3. , 2.75])

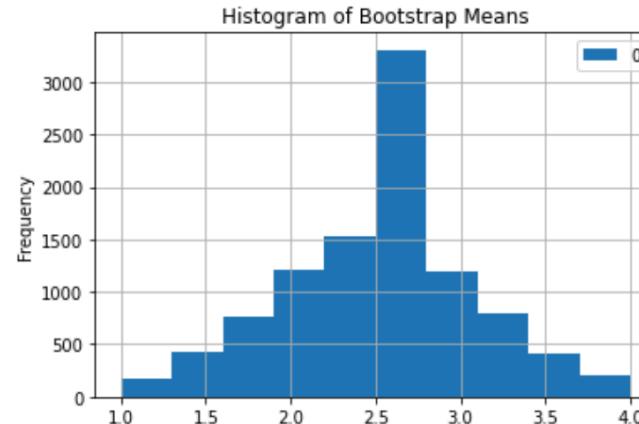
# Вычислим стандартную ошибку бутстрата,
# т.е. оценим стандартную ошибку распределения средних
np.std(mu, ddof=1) # число степеней свободы (ddof) = 1 для стандартного отклонения образца
>> 0.395
```

Пример Bootstrap. Увеличение числа образцов

- $\text{Std.Dev.}(Z^*|B = 1000) \approx 0.5532$
- $\text{Std.Dev.}(Z^*|B = 10000) \approx 0.5608$
- $\text{Std.Dev.}(Z^*|B = 100000) \approx 0.5585$

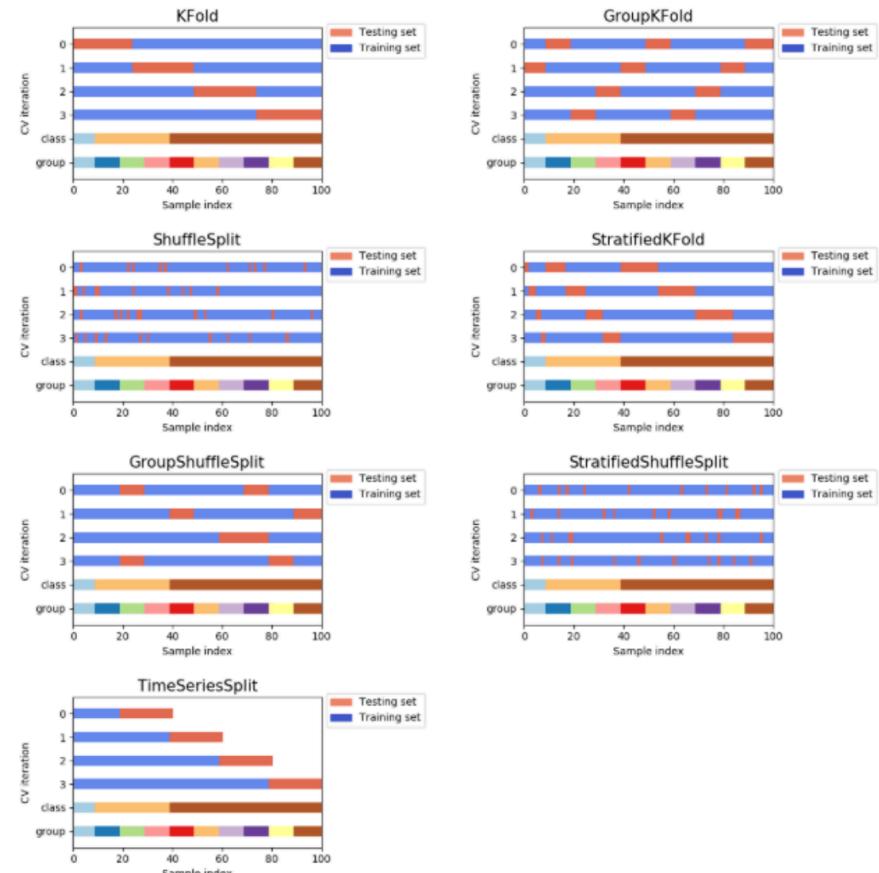
```
import pandas as pd
df = pd.DataFrame(mu)
df.plot(kind='hist', grid=True, title='Histogram of Bootstrap Means');
df.describe().round(2).T
```

	count	mean	std	min	25%	50%	75%	max
0	10000.0	2.5	0.56	1.0	2.0	2.5	3.0	4.0



Использование SKLearn для пересемплирования

- Scikit-Learn имеет удобный набор инструментов для разделения выборки
 - Некоторые из них позволяют также перемешивать события



Обучение без учителя

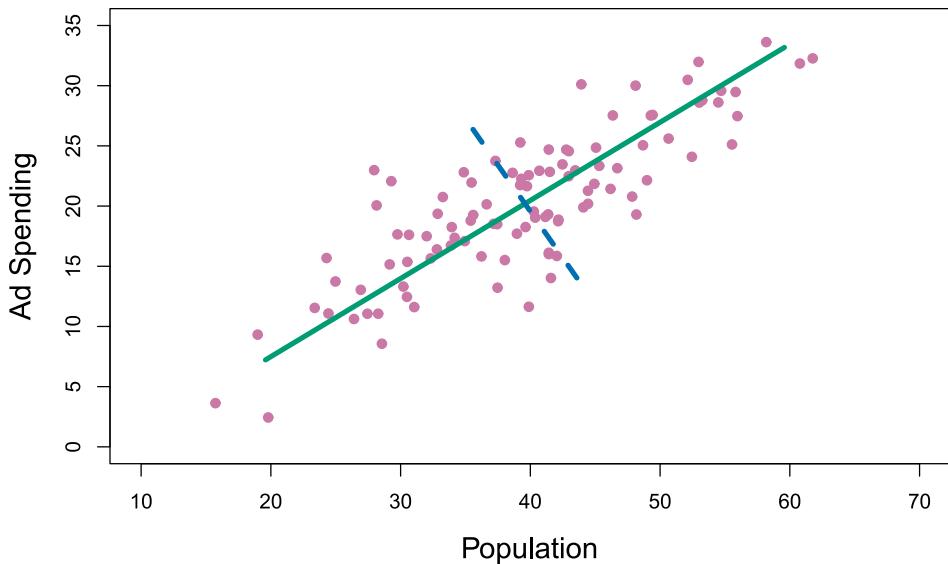
Снижение размерности

Снижение размерности: метод главных компонент

Метод главных компонент (PCA) - линейный метод снижения размерности, который преобразует исходные признаки в новый набор признаков (главных компонентов), сортируя их по величине дисперсии.

- Преимущества:
 - Уменьшение мультиколлинеарности
 - Преобразует коррелированные признаки в набор независимых главных компонентов
 - Сжатие данных
 - Часто позволяет сохранить большую часть информации в меньшем числе признаков

Метод главных компонент



Источник изображения: ISLP Fig. 6.14

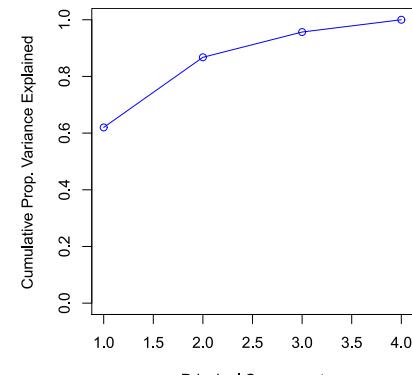
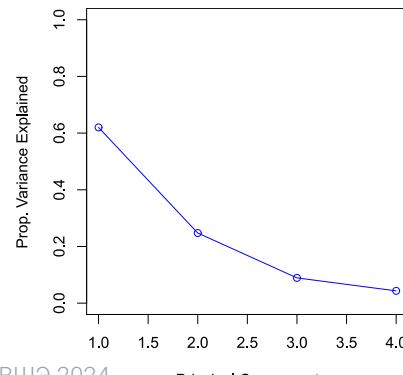
PCA в sklearn

```
from sklearn import datasets, decomposition
pca = decomposition.PCA(n_components=3)
pca.fit(X)
X = pca.transform(X)
```

Не забудьте провести стандартизацию признаков перед использованием PCA!

Выбор числа главных компонент

- Мы предпочитаем число главных компонент, которые объясняют **большую** изменчивость всех признаков X_i
 - Предполагается, что необъясненная изменчивость является результатом шума
- Для оценки часто используются графики типа Scree или Elbow
 - Ищите наибольший спад в доле объясненной дисперсии
 - Или для наибольшего роста накопленной доли объясненной дисперсии
- Это сжатие лучше всего работает для линейно связанных предикторов
- Вы можете "сжать" тысячи признаков до десятка с незначительной "потерей информации"



Линейный Дискриминантный Анализ

Линейный Дискриминантный Анализ (LDA) - линейный метод, который стремится найти линейные комбинации признаков, максимизирующие различие между классами.

- Преимущества:
 - Улучшение различимости классов
 - Хорошо работает для задач классификации
 - Уменьшение размерности
 - снижает размерность до $K - 1$, где K – количество классов

t-распределенное стохастическое вложение соседей

t-распределенное стохастическое вложение соседей (t-SNE) - нелинейный метод, направленный на визуализацию высокоразмерных данных путем моделирования вероятностей соседства точек.

- Преимущества:
 - Визуализация
 - Хорош для визуализации высокоразмерных данных в двумерном или трёхмерном пространстве
 - Учет сложных структур
 - Способен сохранять сложные нелинейные структуры данных

Демо

Самоорганизующиеся карты

Самоорганизующиеся карты (SOM) - нейронная сеть, обучающаяся отображать высокоразмерные данные в двумерное пространство, сохраняя топологические свойства данных.

- Преимущества:
- Топологическое упорядочение
 - служит для кластеризации и визуализации данных
- Интерпретируемость
 - Карта легко интерпретируется, поскольку сохраняет топологию исходных данных

Автокодировщики

Автокодировщики (автоэнкодеры) - нейронные сети, обучающиеся кодировать входные данные в сжатое представление и затем восстанавливать исходные данные из этого представления.

- Преимущества:
 - Обучение существенных признаков
 - способны выявлять скрытые, важные представления во входных данных
 - Гибкость
 - применимы к сложным, нелинейным структурам данных

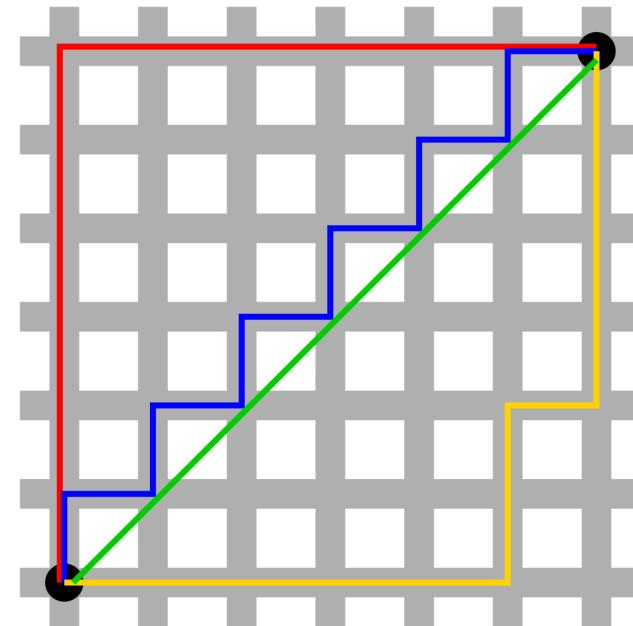
Дополнительные материалы

Условные обозначения

- $a \in A$ - принадлежность множеству: a - элемент множества A
- $|B|$ - мощность множества: количество элементов в множестве B
- $\|\mathbf{v}\|$ - норма: "длина" вектора \mathbf{v}

Расстояния:

- манхэттенское (L_1)
- евклидово (L_2)
- \sum - сумма
- \int - интеграл
- \mathbb{R} - пространство вещественных чисел
- \mathbb{R}^n - то же, но размерности n



Источник:

https://ru.wikipedia.org/wiki/Расстояние_городских_кварталов

Условные обозначения

- $\frac{dy}{dx}$ - производная функции y по переменной x
- $f : A \rightarrow B$ - функция (отображение) области определений A в область значений B
 - A и B могут быть подмножествами многомерного пространства \mathbb{R}^n , $n > 1$
 - Например: $f(x), f(\mathbf{x}), \mathbf{f}(x), \mathbf{f}(\mathbf{x})$
- $\frac{\partial y}{\partial x_i}$ - частная производная функции y по i -ой компоненте вектора \mathbf{x}
- $X_{n \times p}$ - матрица данных размера $n \times p$
 - столбцы X_i признаков
(фичей, переменных, предикторов)
 - строки x_j наблюдений
- $\mathbf{x}, \mathbf{y}, \mathbf{z}$ - векторы; $\mathbf{A}, \mathbf{B}, \mathbf{X}$ - матрицы

вектор

Refund	Marital Status	Taxable Income	Cheat
Yes	Single	125K	No
No	Married	100K	No
No	Single	70K	No
Yes	Married	120K	No
No	Divorced	95K	Yes
No	Married	60K	No
Yes	Divorced	220K	No
No	Single	85K	Yes
No	Married	75K	No
No	Single	90K	Yes

матрица

Бесплатного завтрака не бывает (Теорема NFL)

- Зачем нам множество функций, моделей, подходов?
- Почему просто не использовать нейронные сети (или градиентный бустинг) **всюду**?
- Пусть A - любой алгоритм обучения для задачи бинарной классификации с потерями $0 - 1$ в области X , и $N \leq 0.5|X|$. Тогда существует распределение (набор данных) D при следующих условиях:
 1. Существует функция f с $\mathcal{L}(D, f) = 0$
 2. С вероятностью по крайней мере $1/7$ при выборе $S \sim D^N$, мы получим $\mathcal{L}(D, A(S)) \geq 1/8$ (см. [understanding-machine-learning/](#))
- Даже если мы найдем очень хороший алгоритм A , который хорошо работает на одних данных, он все равно может потерпеть неудачу на других данных
- **Это справедливо только при отсутствии знаний о данных!**

END