



Линейная регрессия

обзор раздела



Линейная регрессия

- Первый наш алгоритм машинного обучения – линейная регрессия – является одним из старейших алгоритмов!
- Давайте быстро посмотрим, что нас ждёт в этом разделе курса



Линейная регрессия

- Линейная регрессия
 - Теория линейной регрессии
 - Простая реализация в Python
 - Обзор Scikit-learn
 - Линейная регрессия в Scikit-learn
 - Полиномиальная регрессия
 - Регуляризация
 - Обзор набора данных для упражнений



Линейная регрессия

- В отличие от разделов курса для других алгоритмов, упражнения по линейной регрессии распределены по нескольким разделам. Сначала нам нужно обсудить построение признаков (feature engineering) и кросс-валидацию, и только после них выполнять полноценный проверочный проект по линейной регрессии



Давайте начнём!



Линейная регрессия

Теория алгоритма - часть 1

История и мотивация



Линейная регрессия

- Прежде чем написать какой-то код, давайте сначала получим интуитивное понимание теории линейной регрессии



Линейная регрессия

- Мы обсудим следующие темы:
 - Краткая история
 - Линейные связи между переменными
 - Наименьшие квадраты
 - Функции стоимости
 - Градиентный спуск
 - Векторизация



Линейная регрессия

- Ранние формы алгоритма линейной регрессии возникли ещё в 1700 годах. Термин регрессия стали применять намного позднее.
- Изначально алгоритм применяли в астрономии, биологии, для улучшения методов морской навигации. Затем в экономике и других сферах.



Линейная регрессия

- 1722 – Роджер Коутс обнаруживает, что объединение нескольких точек наблюдения может улучшить оценки истинного значения
- 1750 – Тобиаш Майер изучает усреднение различных результатов в похожих условиях, наблюдая за поведением луны



Линейная регрессия

- 1757 – Роджер Джозеф Боскович продолжает изучать объединение нескольких измерений, наблюдая за Землёй
- 1788 – Пьер-Симон Лаплас разрабатывает похожие теории усреднения значений, объясняя разницу в движении между Юпитером и Сатурном



Линейная регрессия

- 1805 – Первая публикация по линейной регрессии с методом наименьших квадратов, Адриен Мари Лежандр





Линейная регрессия

- 2005 – было обнаружено, что портрет Адриена Мари Лежандра в профиль принадлежит не ему, а Луису Лежандру!
- Единственный достоверный портрет учёного – следующая карикатура...





Линейная регрессия

- Единственный достоверный портрет Адриена Мари Лежандра – следующая карикатура





Линейная регрессия

- 1809 – Карл Фридрих Гаусс публикует методы вычисления орбит небесных тел
- Он утверждает, что изобрёл метод наименьших квадратов ещё в 1795 году!





Линейная регрессия

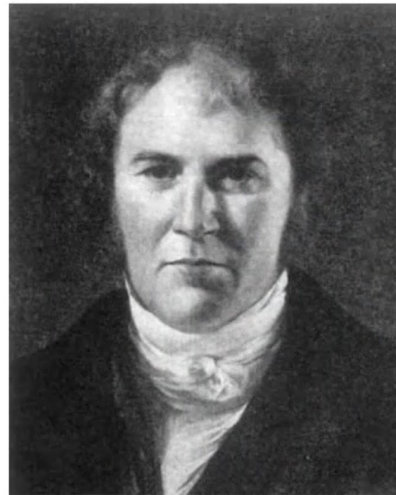
- Карл Фридрих Гаусс родился в 1777 году, так что на момент заявленного изобретения ему было всего 18 лет!





Линейная регрессия

- 1808 – Роберт Адрайн опубликовал свою формулировку наименьших квадратов (за год до публикации Гаусса)





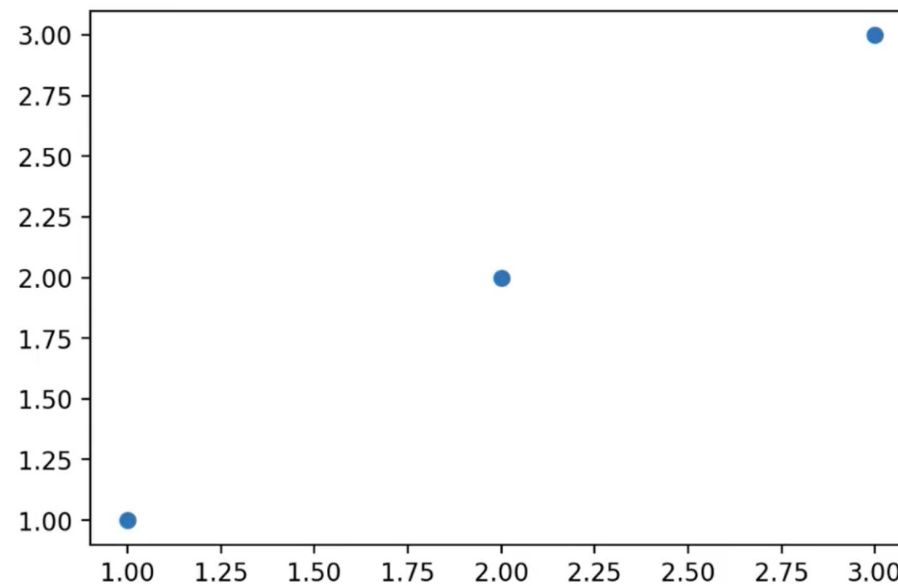
Линейная регрессия

- Итак, вне зависимости от истинного автора, давайте получим интуитивное понимание теории линейной регрессии!



Линейная регрессия

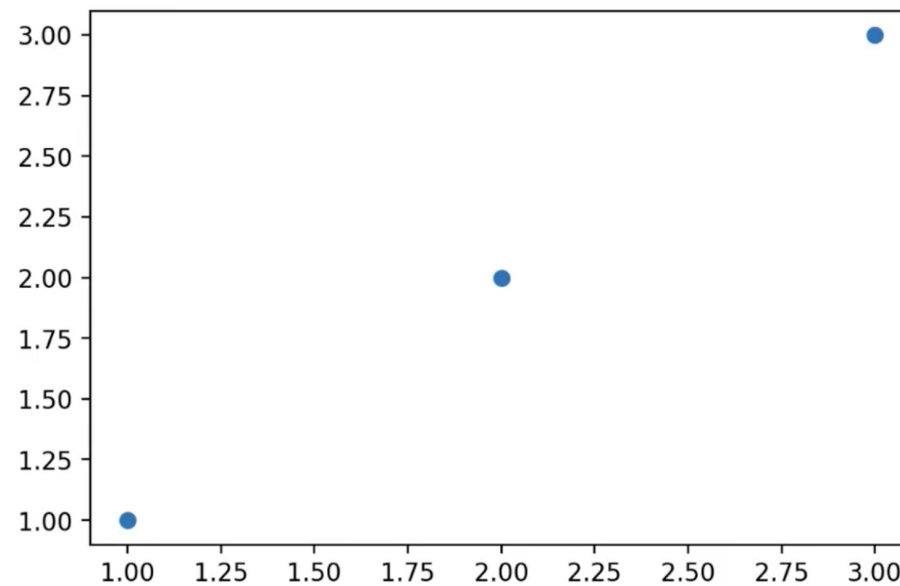
- Простыми словами: линейная связь между двумя переменными – это связь в виде прямой линии
- Простейший пример: $y=x$





Линейная регрессия

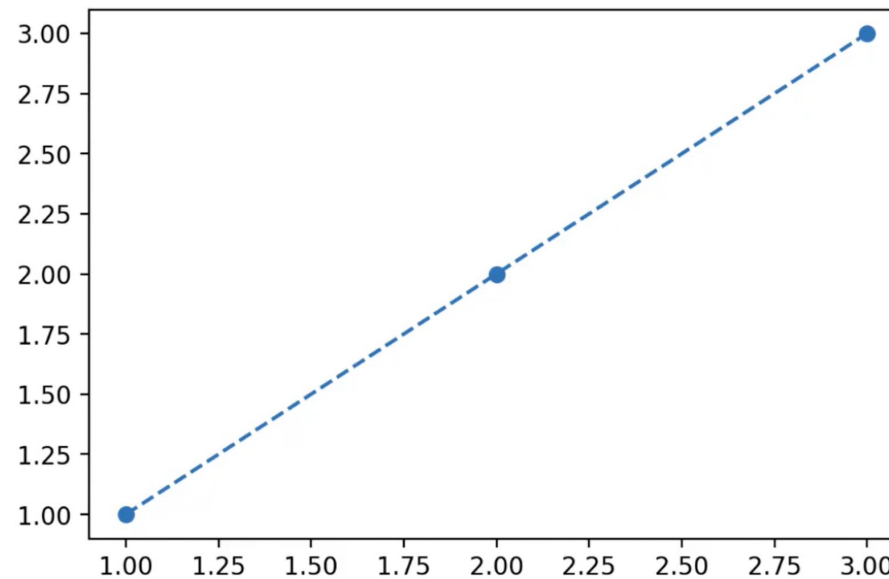
- Здесь мы видим точки $x=[1,2,3]$ и $y=[1,2,3]$





Линейная регрессия

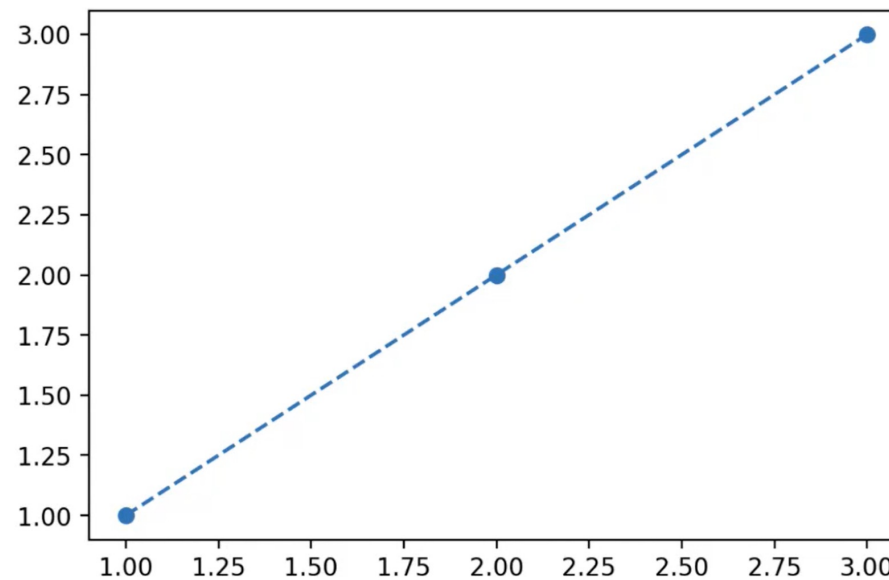
- Здесь мы видим точки $x=[1,2,3]$ и $y=[1,2,3]$
- По этим трём точкам мы можем провести линию





Линейная регрессия

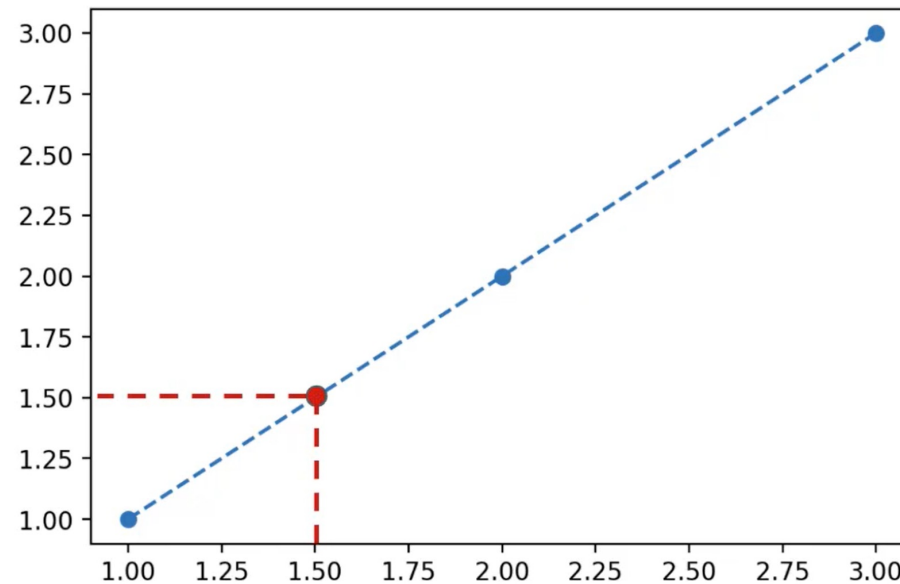
- Здесь мы видим точки $x=[1,2,3]$ и $y=[1,2,3]$
- По этим трём точкам мы можем провести линию
- Эта линия – связь $y=x$ между переменными x и y





Линейная регрессия

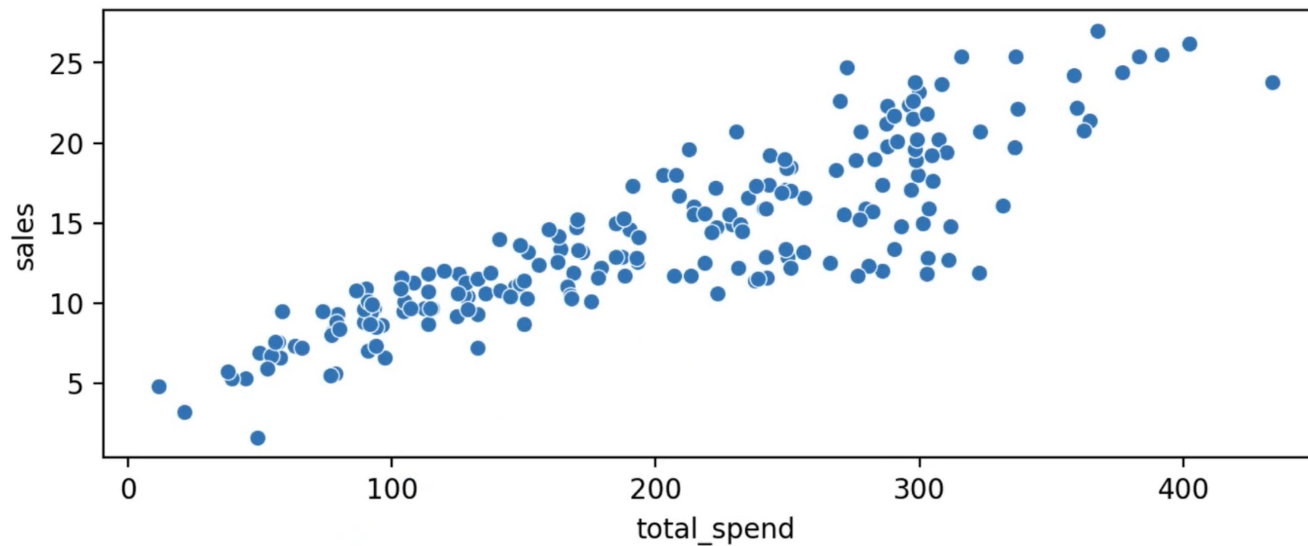
- Это значит, что для какого-то нового значения x мы можем предсказать значение y





Линейная регрессия

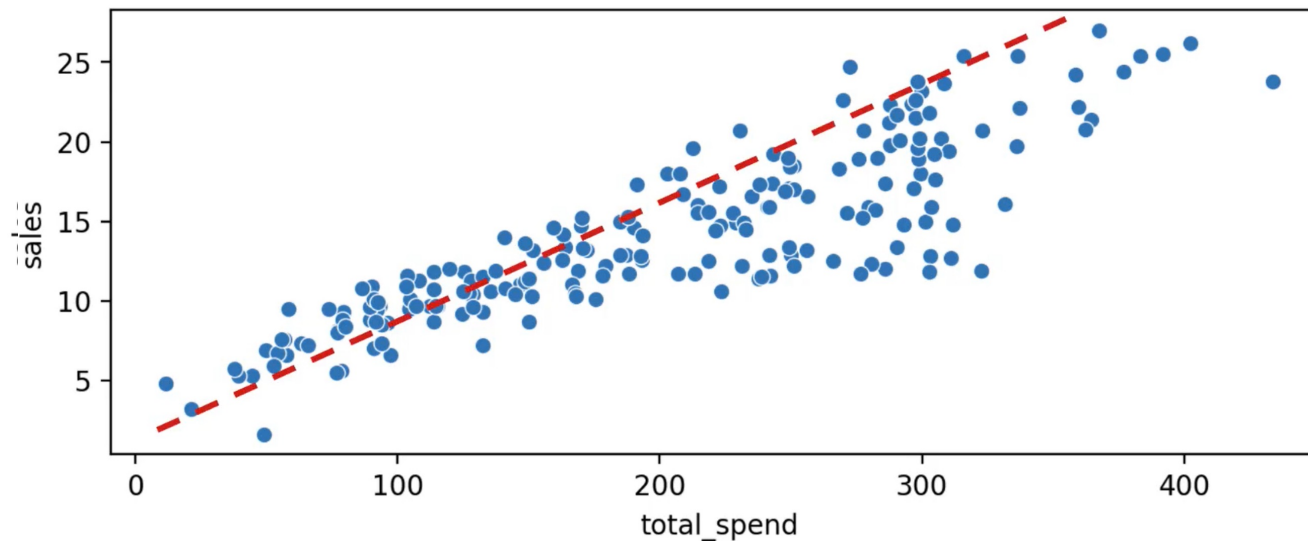
- Но что делать с реальными данными? Как нам провести здесь линию?





Линейная регрессия

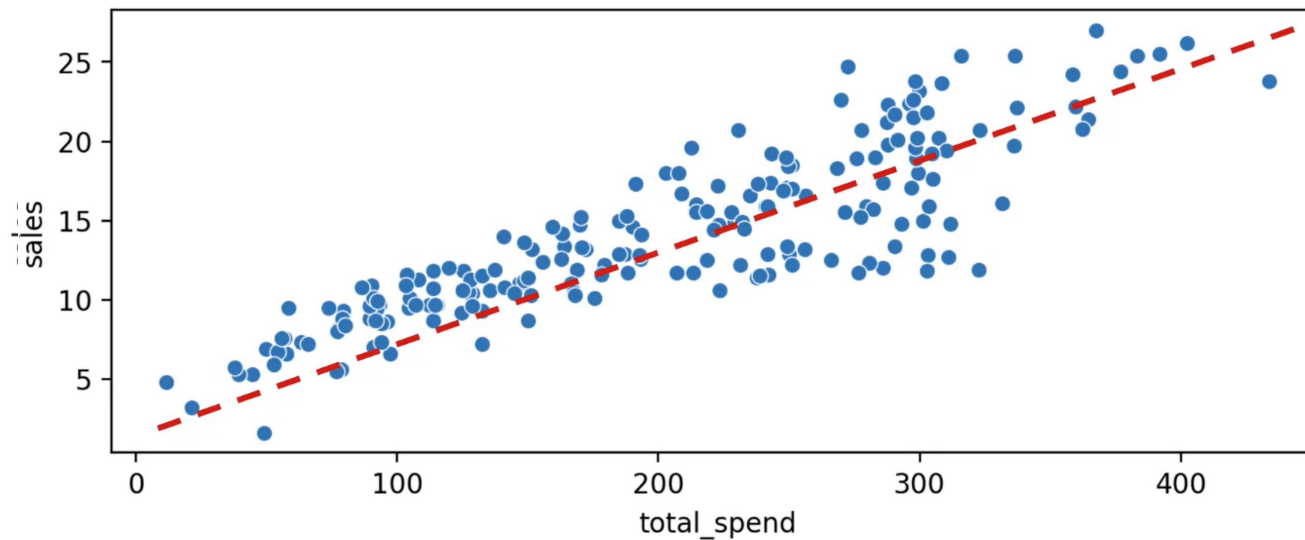
- Но что делать с реальными данными? Как нам провести здесь линию?





Линейная регрессия

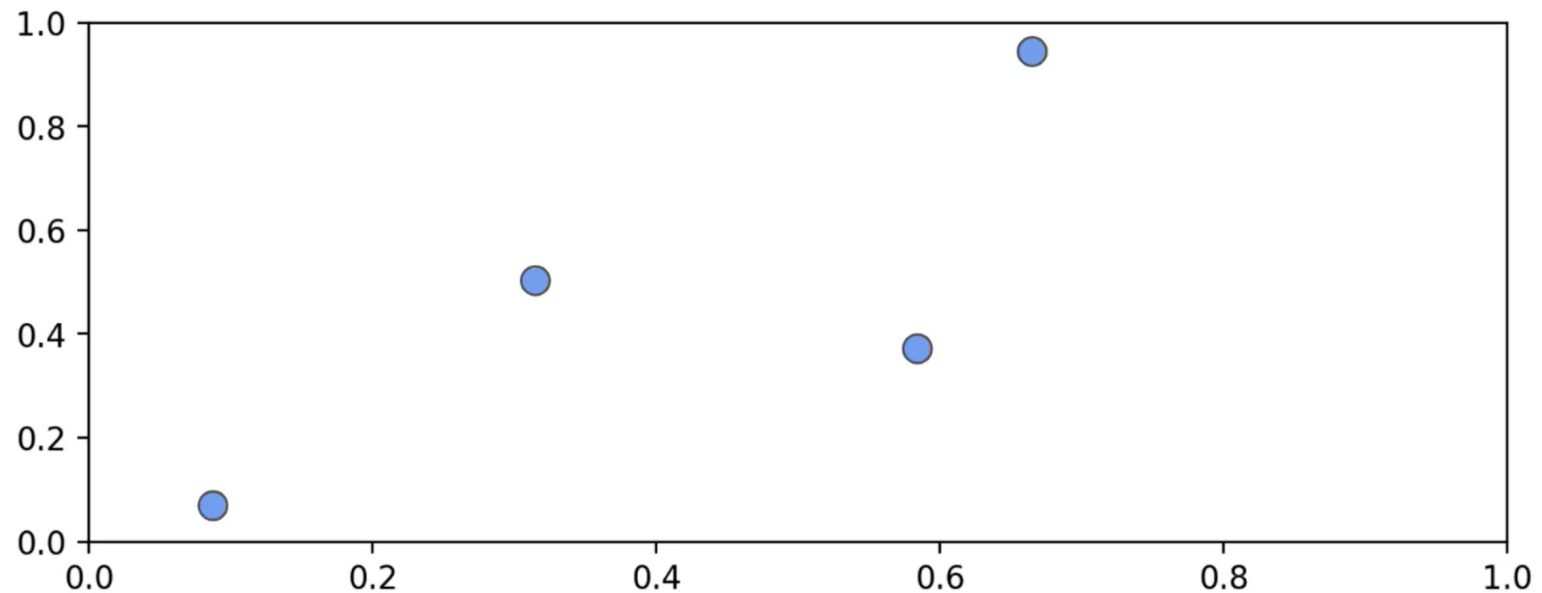
- Но что делать с реальными данными? Как нам провести здесь линию?





Линейная регрессия

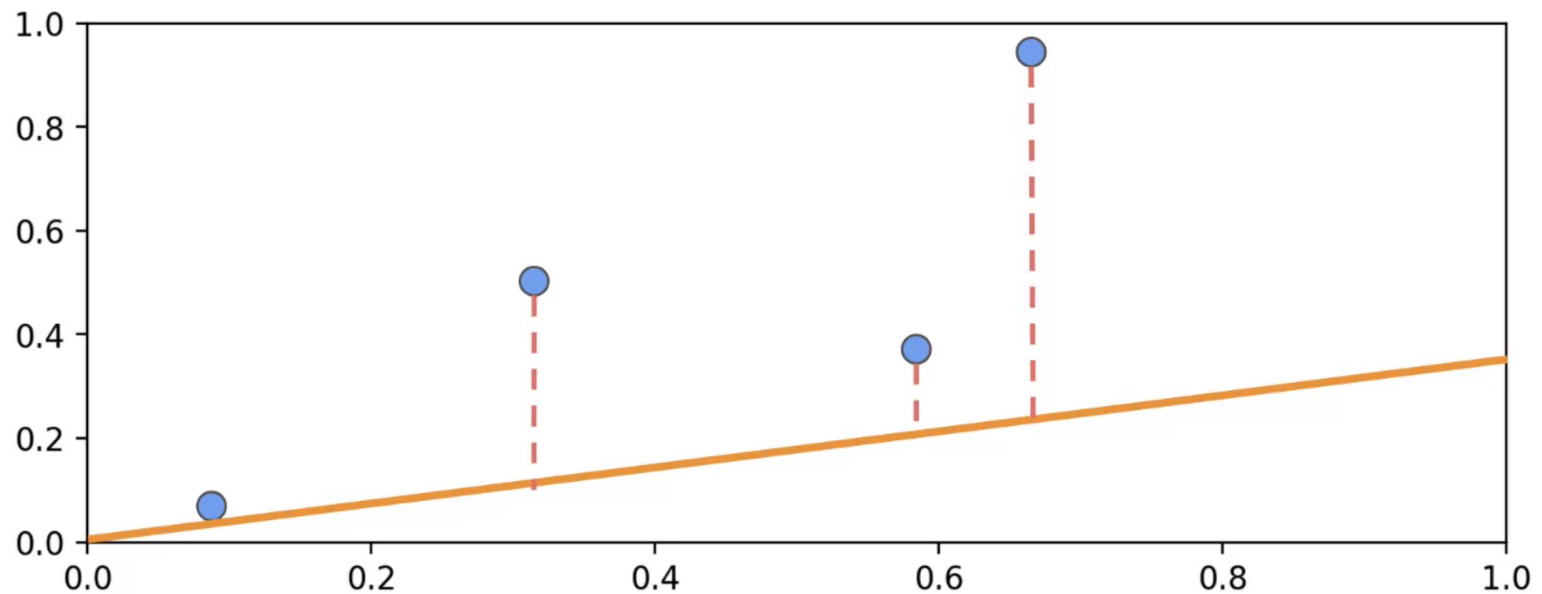
- По сути, нам хотелось бы минимизировать общее расстояние между точками и линией





Линейная регрессия

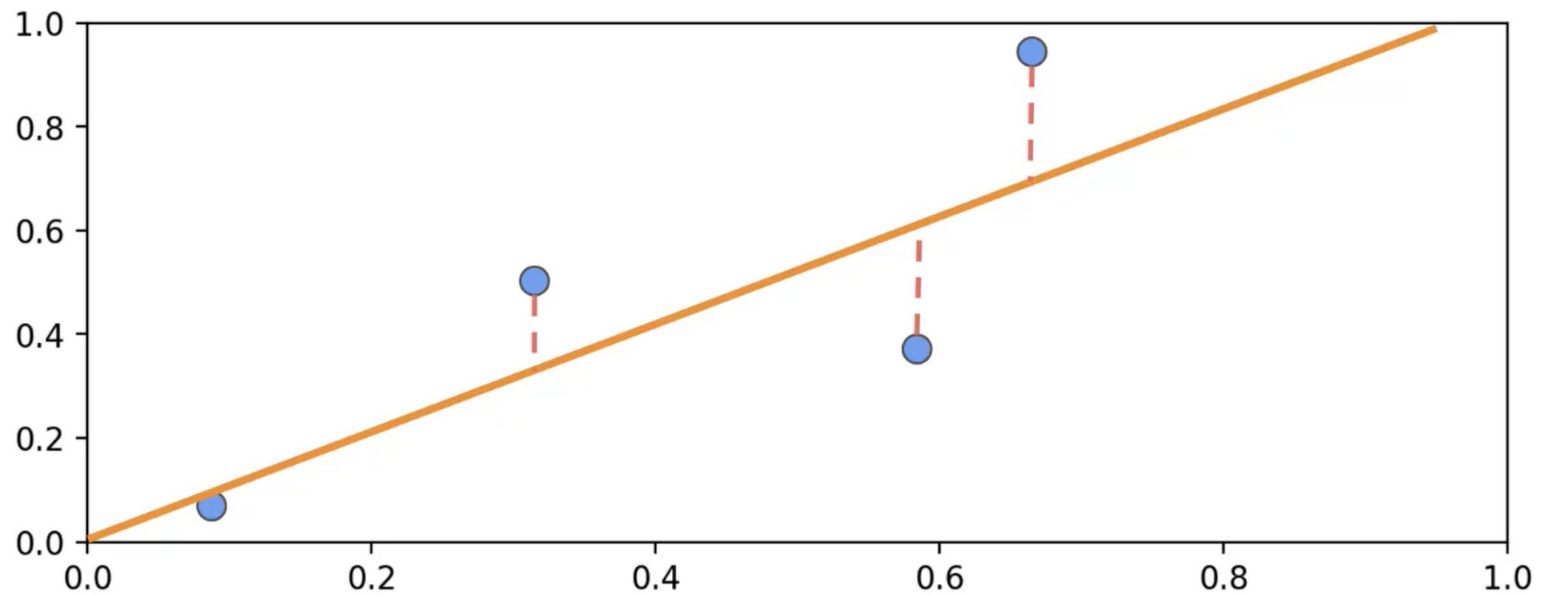
- Проведя линию, мы можем измерить расстояния от точек до линии – это будут ошибки (остатки).





Линейная регрессия

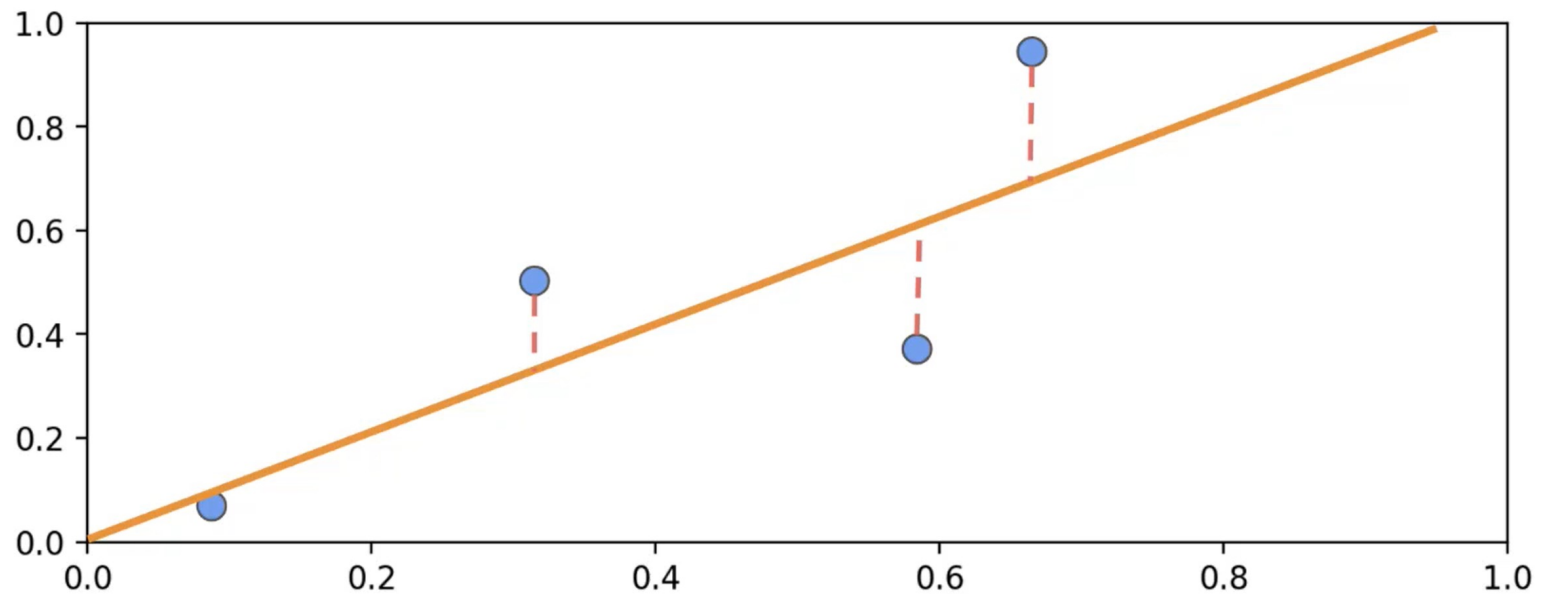
- Некоторые линии явно лучше других





Линейная регрессия

- Некоторые линии явно лучше других
- Ошибки бывают положительными и отрицательными





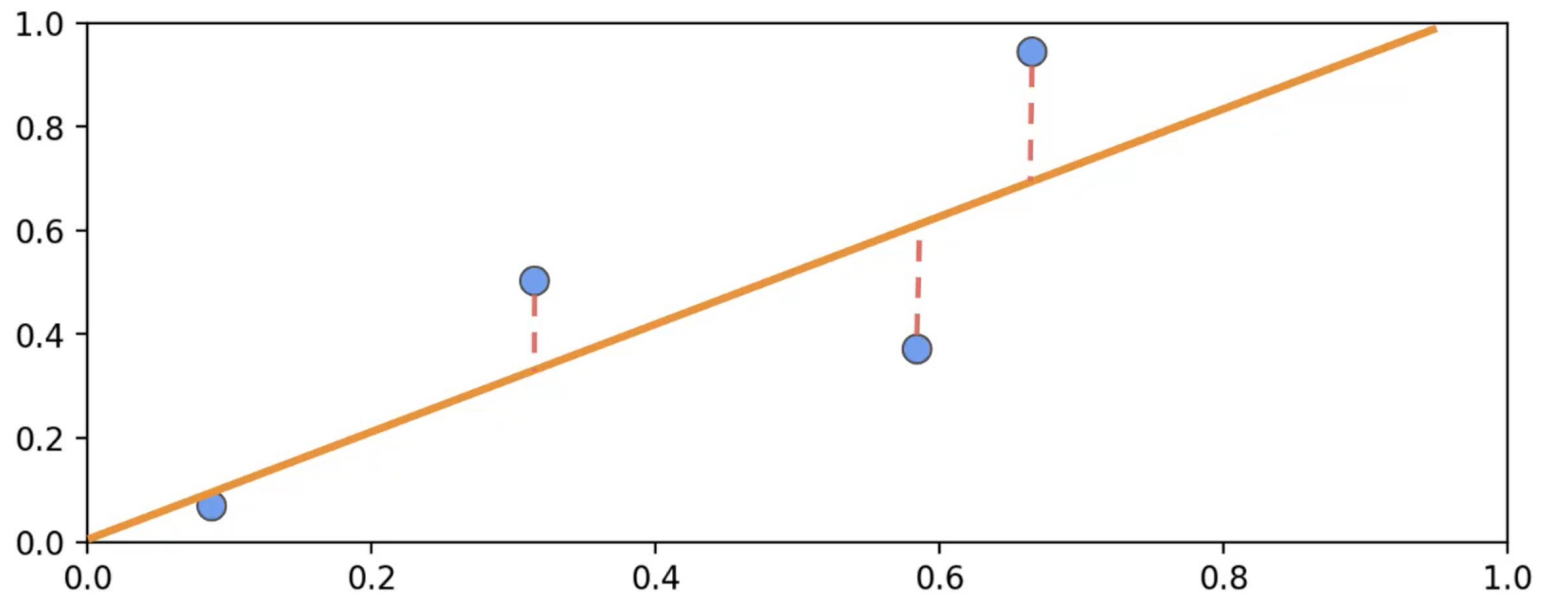
Линейная регрессия

- Метод наименьших квадратов стремится минимизировать сумму квадратов разницы между наблюдаемым значением в наборе данных и значением, предсказанным линейной регрессией



Линейная регрессия

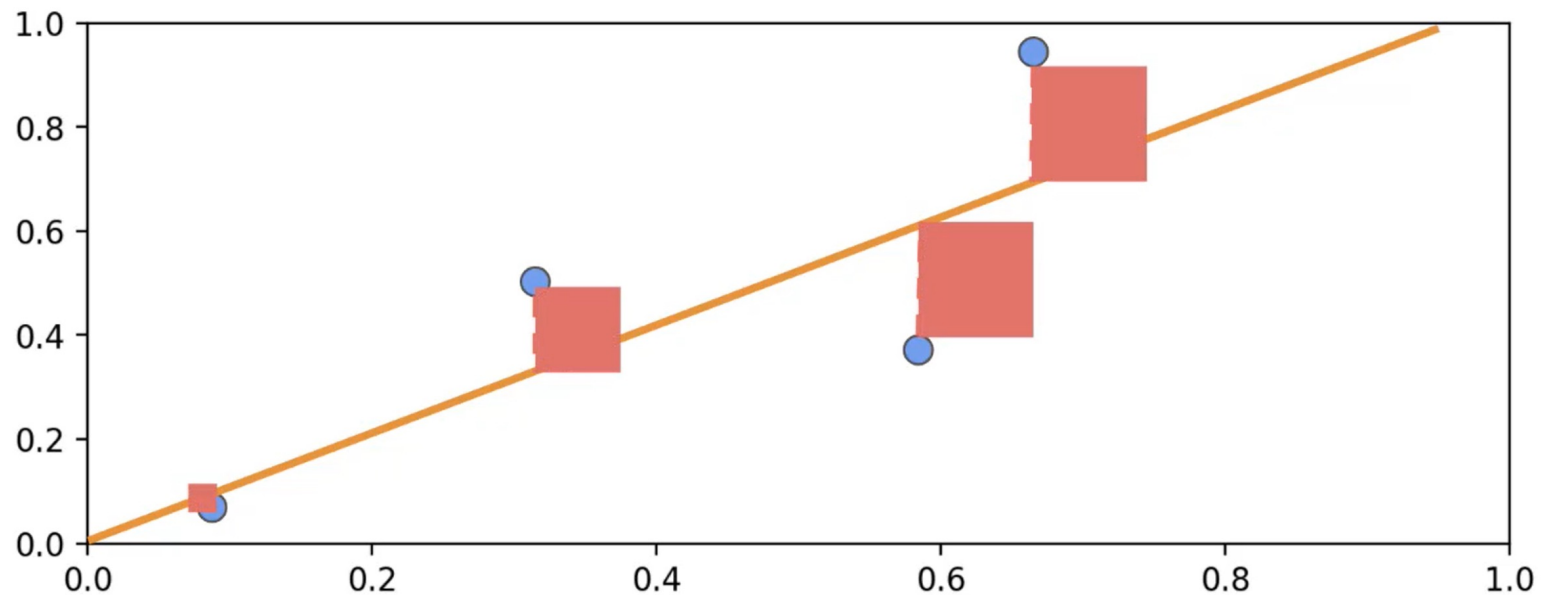
- Мы можем визуализировать квадраты ошибок, которые нужно минимизировать:





Линейная регрессия

- Мы можем визуализировать квадраты ошибок, которые нужно минимизировать:





Линейная регрессия

- Применение квадрата ошибок вместо самих ошибок поможет упростить вычисления, когда мы будем смотреть на производные
- Давайте продолжим изучение метода наименьших квадратов, приведя набор данных к математической нотации, и затем решив линейное уравнение между признаками и целевой переменной.



Линейная регрессия

Теория алгоритма - часть 2

наименьшие квадраты



Линейная регрессия

- Метод наименьших квадратов
 - Мы знаем уравнение для прямой линии:
 - $y = mx + b$
 - m это наклон линии
 - b это пересечение с осью y



Линейная регрессия

- Метод наименьших квадратов
 - В уравнении $y=mx+b$ есть место только для одной переменной x
 - Метод наименьших квадратов поможет нам решить это уравнение – найти коэффициенты m и b
 - Позже мы узнаем, как распространить этот метод на несколько переменных, применяя метод градиентного спуска



Линейная регрессия

- Давайте переведем набор с данными в математическую нотацию для линейной регрессии
- Затем решим задачу для одной переменной, чтобы увидеть работу метода наименьших квадратов в действии
- Далее перейдем к градиентному спуску и более реалистичным сценариям нескольких переменных



Линейная регрессия

- Можно выразить это математическими терминами

X

y

Area m ²	Bedrooms	Bathrooms	Price
200	3	2	\$500,000
190	2	1	\$450,000
230	3	3	\$650,000
180	1	1	\$400,000
210	2	2	\$550,000



Линейная регрессия

- Можно выразить это математическими терминами

X

y

x_1	x_2	x_3	y
200	3	2	\$500,000
190	2	1	\$450,000
230	3	3	\$650,000
180	1	1	\$400,000
210	2	2	\$550,000



Линейная регрессия

- Можно выразить это математическими терминами

X

y

x_1	x_2	x_3	y
x_1^1	3	2	\$500,000
x_1^2	2	1	\$450,000
x_1^3	3	3	\$650,000
x_1^4	1	1	\$400,000
x_1^5	2	2	\$550,000



Линейная регрессия

- Можно выразить это математическими терминами

X

y

x_1	x_2	x_3	y
x_1^1	x_2^1	x_3^1	y_1
x_1^2	x_2^2	x_3^2	y_2
x_1^3	x_2^3	x_3^3	y_3
x_1^4	x_2^4	x_3^4	y_4
x_1^5	x_2^5	x_3^5	y_5



Линейная регрессия

- Далее построим линейную связь между X и y

X

y

x_1	x_2	x_3	y
-------	-------	-------	-----



Линейная регрессия

- Поменяем переменные местами





Линейная регрессия

- У каждой переменной X должен быть коэффициент

y		X		
y	x_1	x_2	x_3	

$$\hat{y} = \beta_0 x_0 + \dots + \beta_n x_n$$



Линейная регрессия

- У каждой переменной X должен быть коэффициент
- Это очень похоже на уравнение $y=mx+b$

y		X		
y	x_1	x_2	x_3	

$$\hat{y} = \beta_0 x_0 + \dots + \beta_n x_n$$



Линейная регрессия

- Это же уравнение можно переписать как сумму

y		x		
y	x_1	x_2	x_3	

$$\hat{y} = \beta_0 x_0 + \dots + \beta_n x_n$$

$$\hat{y} = \sum_{i=0}^n \beta_i x_i$$



Линейная регрессия

- Обратите внимание, что здесь написано $\hat{y}$ с крышкой – оценка значения. Потому что обычно нет таких коэффициентов бетта, чтобы точно воспроизвести y .

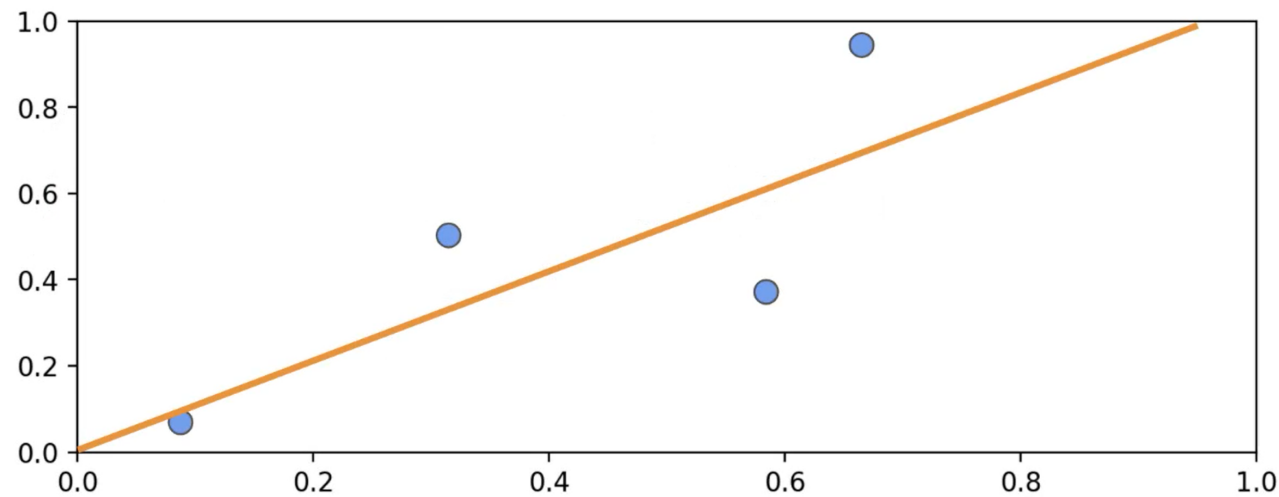
$$\hat{y} = \sum_{i=0}^n \beta_i x_i$$



Линейная регрессия

- Линейное уравнение:

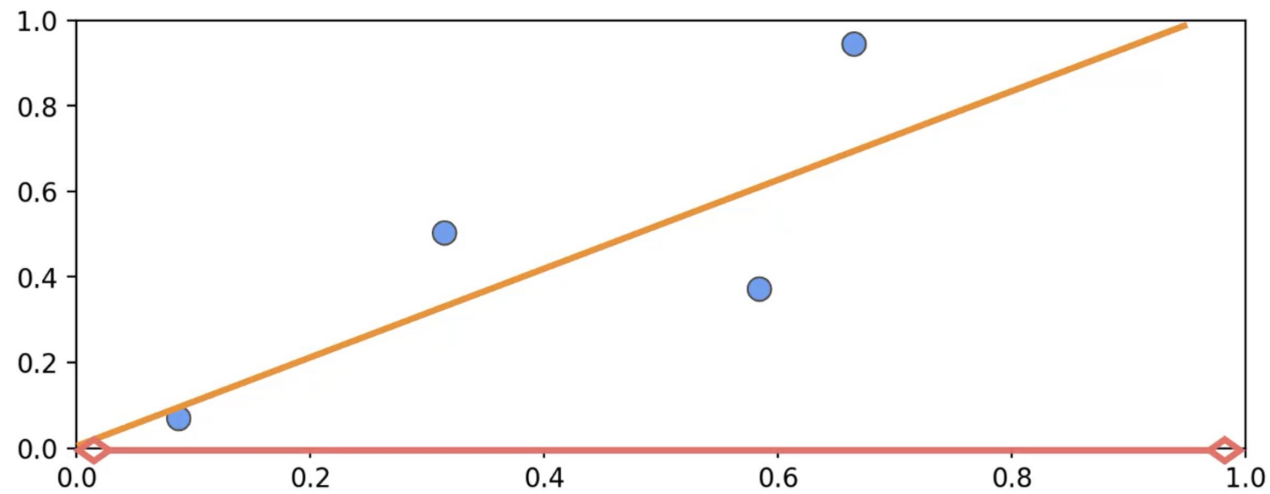
$$\hat{y} = \sum_{i=0}^n \beta_i x_i$$





Линейная регрессия

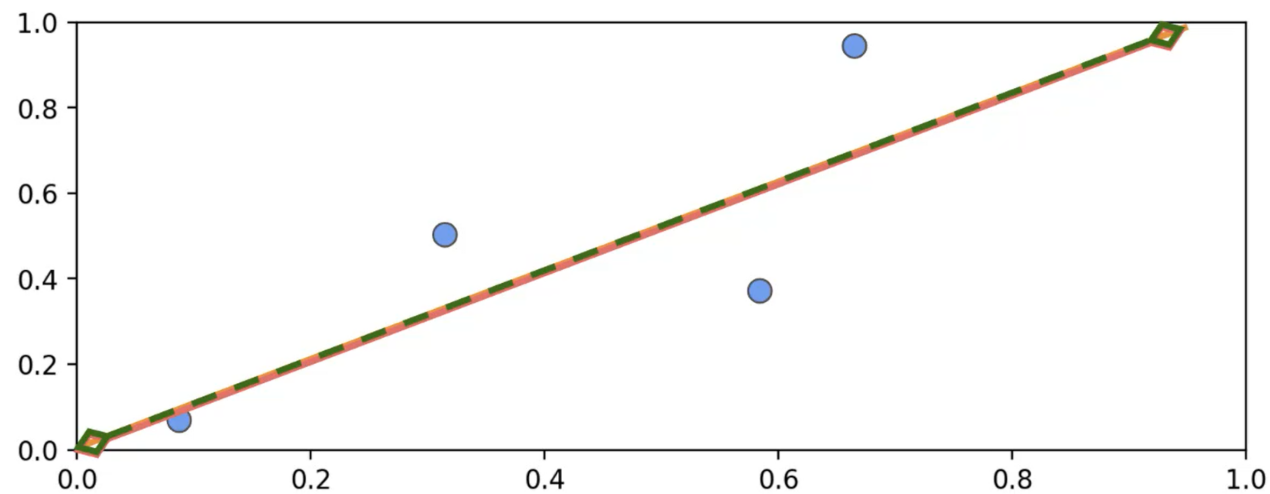
$$\hat{y} = \sum_{i=0}^n \beta_i x_i$$





Линейная регрессия

$$\hat{y} = \sum_{i=0}^n \beta_i x_i$$





Линейная регрессия

- Для простых задач с одной переменной можно вывести решение с помощью уравнений
- Давайте решим простую задачу для примера, а затем перейдём к нескольким переменным и градиентному спуску



Линейная регрессия

- Как мы обсуждали, уравнение прямой линии имеет вид

$$y = mx + b$$

m это наклон линии

b это пересечение с осью y при $x=0$



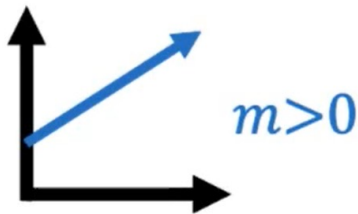
Линейная регрессия

- Как мы обсуждали, уравнение прямой линии имеет вид

$$y = mx + b$$

m это наклон линии

b это пересечение с осью y при $x=0$



Положительный наклон



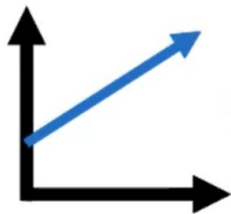
Линейная регрессия

- Как мы обсуждали, уравнение прямой линии имеет вид

$$y = mx + b$$

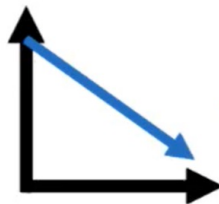
m это наклон линии

b это пересечение с осью y при $x=0$



$m > 0$

Положительный наклон



$m < 0$

Отрицательный наклон



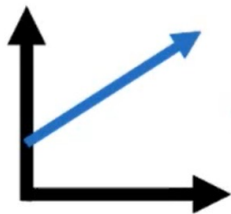
Линейная регрессия

- Как мы обсуждали, уравнение прямой линии имеет вид

$$y = mx + b$$

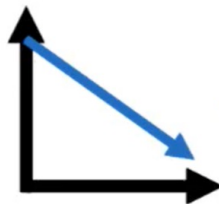
m это наклон линии

b это пересечение с осью y при $x=0$



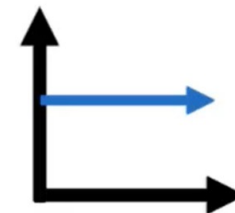
$m > 0$

Положительный наклон



$m < 0$

Отрицательный наклон



$m = 0$

Нулевой наклон



Линейная регрессия

- В линейной регрессии, где мы пытаемся сформулировать связь между переменными, уравнение $y=mx+b$ превращается в уравнение

$$\hat{y} = b_0 + b_1x$$



Линейная регрессия

- В линейной регрессии, где мы пытаемся сформулировать связь между переменными, уравнение $y=mx+b$ превращается в уравнение

$$\hat{y} = b_0 + b_1x$$

- Цель – предсказать значение **зависимой переменной** (y) на основе **независимой переменной** (x)



Линейная регрессия

$$\hat{y} = b_0 + b_1x$$

- Для коэффициентов b_1 и b_0 есть формулы:

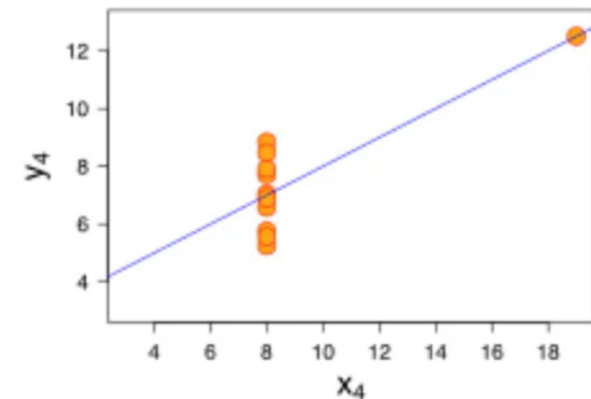
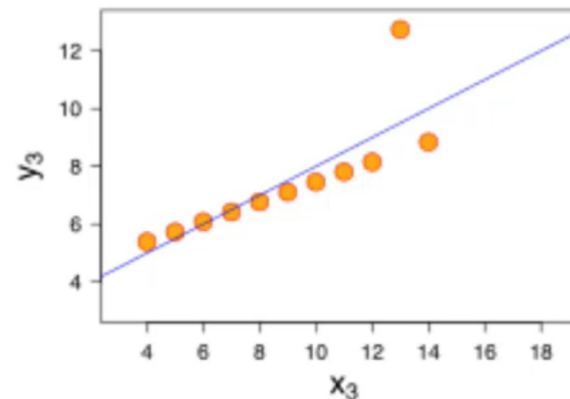
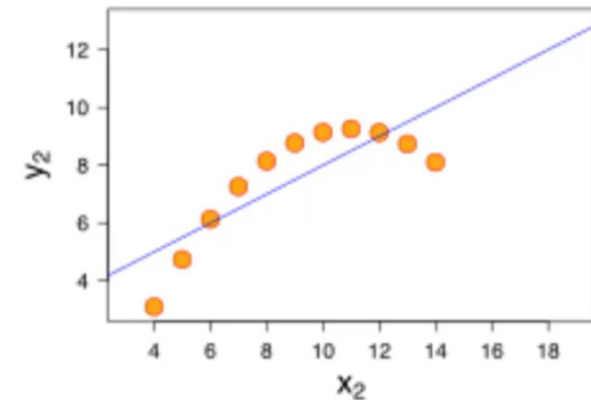
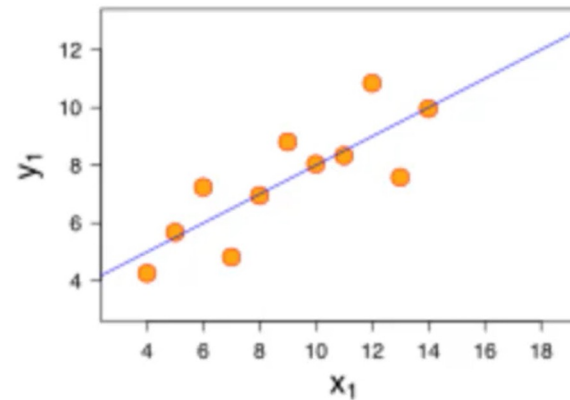
$$b_1 = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sum (x_i - \bar{x})^2}$$

$$b_0 = \bar{y} - b_1\bar{x}$$



Ограничения линейной регрессии

- Квартет Энскомба показывает возможные недостатки регрессии
- Каждый из 4х графиков имеет одну и ту же линию регрессии





Линейная регрессия

- Пример задачи. Мы хотим найти зависимость между количеством часов, которые завод работает в неделю, и объёмом выпускаемой продукции.





Линейная регрессия

- Пример задачи. Мы хотим найти зависимость между количеством часов, которые завод работает в неделю, и объёмом выпускаемой продукции.
- Здесь независимая переменная x — количество часов
- Зависимая переменная y — объём продукции





Линейная регрессия

- Пусть у нас есть следующие данные:

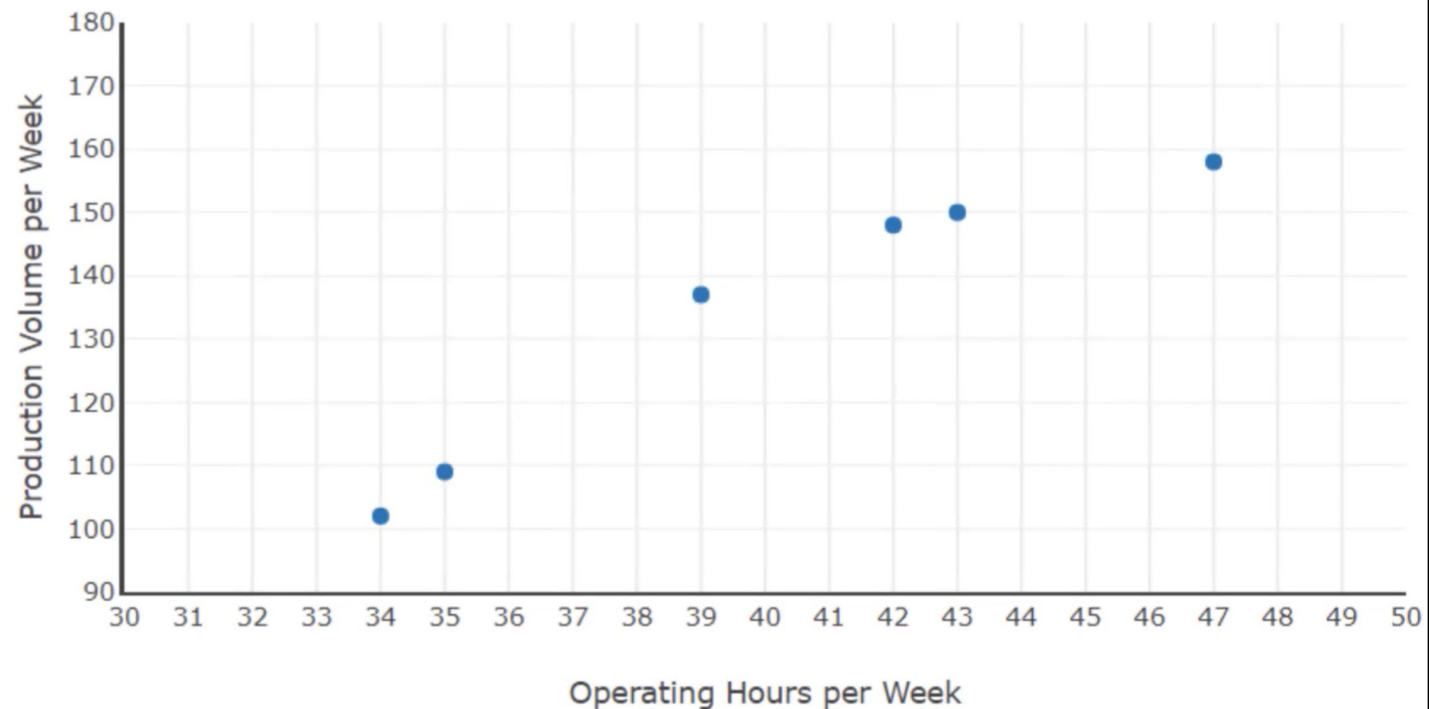
Production Hours (x)	Production Volume (y)
34	102
35	109
39	137
42	148
43	150
47	158



Линейная регрессия

- Прежде всего нанесём данные на график:

Production Hours (x)	Production Volume (y)
34	102
35	109
39	137
42	148
43	150
47	158

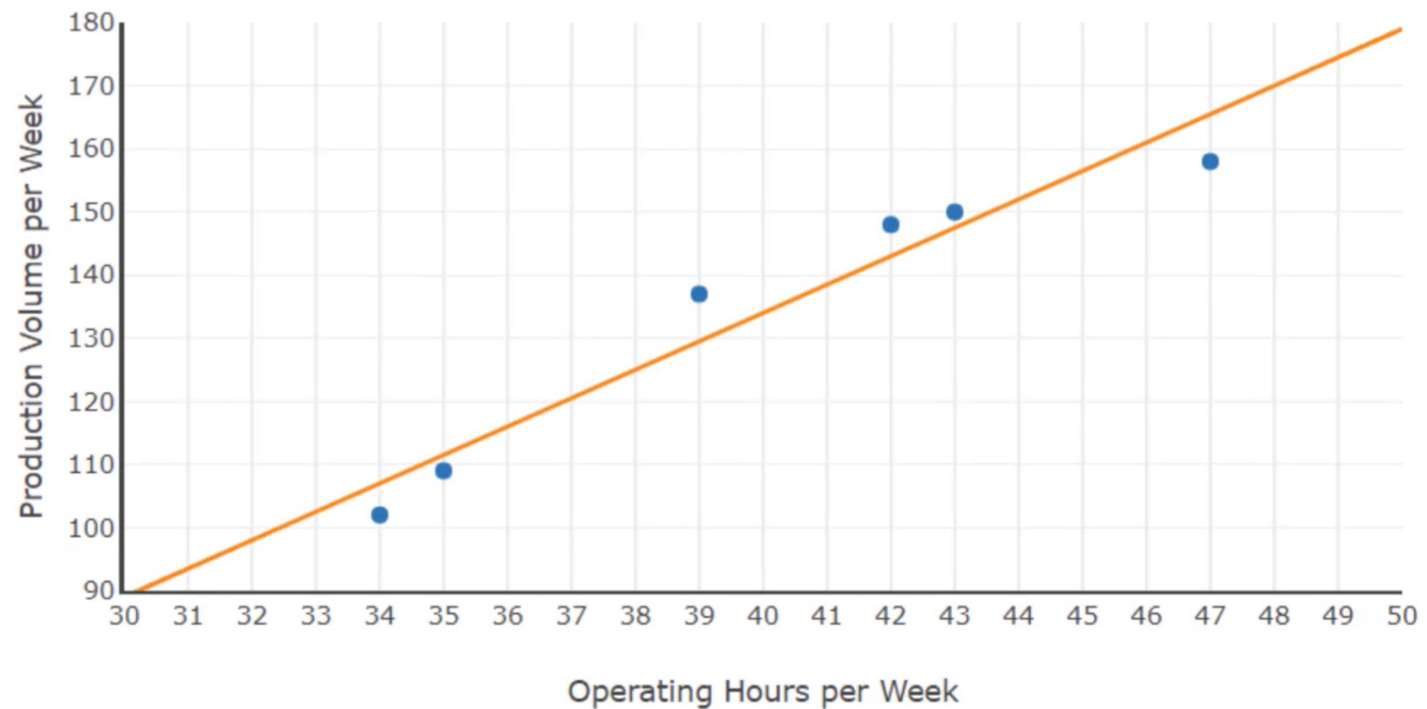




Линейная регрессия

- Прежде всего нанесём данные на график:

Production Hours (x)	Production Volume (y)
34	102
35	109
39	137
42	148
43	150
47	158





Линейная регрессия

- Выполним вычисления:

Production Hours (x)	Production Volume (y)
34	102
35	109
39	137
42	148
43	150
47	158

$$\hat{y} = b_0 + b_1x$$
$$b_1 = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})^2}$$
$$b_0 = \bar{y} - b_1\bar{x}$$



Линейная регрессия

- Выполним вычисления:

	Production Hours (x)	Production Volume (y)
	34	102
	35	109
	39	137
	42	148
	43	150
	47	158
$\bar{x}, \bar{y}$	40	134

$$\hat{y} = b_0 + b_1x$$
$$b_1 = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})^2}$$
$$b_0 = \bar{y} - b_1\bar{x}$$



Линейная регрессия

- Выполним вычисления:

Production Hours (x)	Production Volume (y)	$(x - \bar{x})$	$(y - \bar{y})$
34	102	-6	-32
35	109	-5	-25
39	137	-1	3
42	148	2	14
43	150	3	16
47	158	7	24
$\bar{x}, \bar{y}$	40	134	

$$\hat{y} = b_0 + b_1x$$
$$b_1 = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})^2}$$
$$b_0 = \bar{y} - b_1\bar{x}$$



Линейная регрессия

- Выполним вычисления:

$$\hat{y} = b_0 + b_1x$$
$$b_1 = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})^2}$$
$$b_0 = \bar{y} - b_1\bar{x}$$

Production Hours (x)	Production Volume (y)	$(x - \bar{x})$	$(y - \bar{y})$	$(x - \bar{x})(y - \bar{y})$
34	102	-6	-32	192
35	109	-5	-25	125
39	137	-1	3	-3
42	148	2	14	28
43	150	3	16	48
47	158	7	24	168
$\bar{x}, \bar{y}$	40	134		



Линейная регрессия

- Выполним вычисления:

$$\hat{y} = b_0 + b_1x$$
$$b_1 = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})^2}$$
$$b_0 = \bar{y} - b_1\bar{x}$$

Production Hours (x)	Production Volume (y)	$(x - \bar{x})$	$(y - \bar{y})$	$(x - \bar{x})(y - \bar{y})$	$(x - \bar{x})^2$
34	102	-6	-32	192	36
35	109	-5	-25	125	25
39	137	-1	3	-3	1
42	148	2	14	28	4
43	150	3	16	48	9
47	158	7	24	168	49
$\bar{x}, \bar{y}$	40	134			



Линейная регрессия

- Выполним вычисления:

$$\hat{y} = b_0 + b_1x$$
$$b_1 = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})^2}$$
$$b_0 = \bar{y} - b_1\bar{x}$$

Production Hours (x)	Production Volume (y)	$(x - \bar{x})$	$(y - \bar{y})$	$(x - \bar{x})(y - \bar{y})$	$(x - \bar{x})^2$
34	102	-6	-32	192	36
35	109	-5	-25	125	25
39	137	-1	3	-3	1
42	148	2	14	28	4
43	150	3	16	48	9
47	158	7	24	168	49
$\bar{x}, \bar{y}$	40 134		Sum:	558	124



Линейная регрессия

- Выполним вычисления:

$$\hat{y} = b_0 + b_1x$$
$$b_1 = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})^2}$$
$$b_0 = \bar{y} - b_1\bar{x}$$

Production Hours (x)	Production Volume (y)	$(x - \bar{x})$	$(y - \bar{y})$	$(x - \bar{x})(y - \bar{y})$	$(x - \bar{x})^2$
34	102	-6	-32	192	36
35	109	-5	-25	125	25
39	137	-1	3	-3	1
42	148	2	14	28	4
43	150	3	16	48	9
47	158	7	24	168	49
$\bar{x}, \bar{y}$	40	134	Sum:	558	124
				$\Sigma(x - \bar{x})(y - \bar{y})$	$\Sigma(x - \bar{x})^2$



Линейная регрессия

- Выполним вычисления:

Production Hours (x)	Production Volume (y)
34	102
35	109
39	137
42	148
43	150
47	158
$\bar{x}, \bar{y}$	40 134

Sum:	558	124
	$\Sigma(x - \bar{x})(y - \bar{y})$	$\Sigma(x - \bar{x})^2$

$$\hat{y} = b_0 + b_1x$$
$$b_1 = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})^2}$$
$$b_0 = \bar{y} - b_1\bar{x}$$



Линейная регрессия

- Выполним вычисления:

$$\hat{y} = b_0 + b_1x$$
$$b_1 = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})^2}$$
$$b_0 = \bar{y} - b_1\bar{x}$$

Production Hours (x)	Production Volume (y)
34	102
35	109
39	137
42	148
43	150
47	158
$\bar{x}, \bar{y}$	40 134

$$b_1 = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})^2}$$

Sum:	558	124
	$\sum(x - \bar{x})(y - \bar{y})$	$\sum(x - \bar{x})^2$



Линейная регрессия

- Выполним вычисления:

$$\hat{y} = b_0 + b_1x$$
$$b_1 = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})^2}$$
$$b_0 = \bar{y} - b_1\bar{x}$$

Production Hours (x)	Production Volume (y)
34	102
35	109
39	137
42	148
43	150
47	158
$\bar{x}, \bar{y}$	40 134

$$b_1 = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})^2} = \frac{558}{124} = 4.5$$

Sum:	558	124
	$\sum(x - \bar{x})(y - \bar{y})$	$\sum(x - \bar{x})^2$



Линейная регрессия

- Выполним вычисления:

Production Hours (x)	Production Volume (y)
34	102
35	109
39	137
42	148
43	150
47	158
$\bar{x}, \bar{y}$	40 134

$$b_1 = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})^2} = \frac{558}{124} = 4.5$$

$$b_0 = \bar{y} - b_1\bar{x}$$

$$\begin{aligned}\hat{y} &= b_0 + b_1x \\ b_1 &= \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})^2} \\ b_0 &= \bar{y} - b_1\bar{x}\end{aligned}$$

Sum:	558	124
	$\sum(x - \bar{x})(y - \bar{y})$	$\sum(x - \bar{x})^2$



Линейная регрессия

- Выполним вычисления:

	Production Hours (x)	Production Volume (y)
	34	102
	35	109
	39	137
	42	148
	43	150
	47	158
$\bar{x}, \bar{y}$	40	134

$$b_1 = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})^2} = \frac{558}{124} = 4.5$$

$$b_0 = \bar{y} - b_1\bar{x} = 134 - (4.5 \times 40) = -46$$

$$\begin{aligned}\hat{y} &= b_0 + b_1x \\ b_1 &= \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})^2} \\ b_0 &= \bar{y} - b_1\bar{x}\end{aligned}$$

Sum:	558	124
	$\sum(x - \bar{x})(y - \bar{y})$	$\sum(x - \bar{x})^2$



Линейная регрессия

- Выполним вычисления:

Production Hours (x)	Production Volume (y)
34	102
35	109
39	137
42	148
43	150
47	158
$\bar{x}, \bar{y}$	40 134

$$b_1 = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})^2} = \frac{558}{124} = 4.5$$

$$b_0 = \bar{y} - b_1\bar{x} = 134 - (4.5 \times 40) = -46$$

$$\hat{y} = -46 + 4.5x$$

$$\begin{aligned}\hat{y} &= b_0 + b_1x \\ b_1 &= \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})^2} \\ b_0 &= \bar{y} - b_1\bar{x}\end{aligned}$$

Sum:	558	124
	$\sum(x - \bar{x})(y - \bar{y})$	$\sum(x - \bar{x})^2$



Линейная регрессия

- По формуле, если мы хотим выпустить 125 единиц продукции в неделю, то завод должен работать столько часов:

	Production Hours (x)	Production Volume (y)
	34	102
	35	109
	39	137
	42	148
	43	150
	47	158
$\bar{x}, \bar{y}$	40	134



Линейная регрессия

- По формуле, если мы хотим выпустить 125 единиц продукции в неделю, то завод должен работать столько часов:

Production Hours (x)	Production Volume (y)
34	102
35	109
39	137
42	148
43	150
47	158
$\bar{x}, \bar{y}$	40 134

$$\hat{y} = b_0 + b_1x$$

$$125 = -46 + 4.5x$$



Линейная регрессия

- По формуле, если мы хотим выпустить 125 единиц продукции в неделю, то завод должен работать столько часов:

Production Hours (x)	Production Volume (y)
34	102
35	109
39	137
42	148
43	150
47	158
$\bar{x}, \bar{y}$	40 134

$$\hat{y} = b_0 + b_1x$$

$$125 = -46 + 4.5x$$

$$x = \frac{171}{4.5} = \mathbf{38} \text{ часов в неделю}$$



Линейная регрессия

- Однако в случае нескольких переменных решение с помощью формул быстро становится немасштабируемым
- Вместо этого применяется минимизация стоимостной функции с помощью метода градиентного спуска



Линейная регрессия

- С помощью градиентного спуска мы найдём коэффициенты бетта в уравнении ниже.

$$\hat{y} = \sum_{i=0}^n \beta_i x_i$$



Линейная регрессия

Теория алгоритма - часть 3

функции стоимости



Линейная регрессия

- Что мы уже знаем:
 - Линейное уравнение: $y=mx+b$
 - Метод наименьших квадратов – есть формулы решения для одной переменной
 - Для нескольких переменных не масштабируется
 - Мы перевели табличные данные в матричную нотацию
 - Есть общая формула для бетта-коэффициентов



Линейная регрессия

- Мы ищем наилучшие бетта-коэффициенты в уравнении ниже

$$\hat{y} = \sum_{i=0}^n \beta_i x_i$$



Линейная регрессия

- Мы ищем наилучшие бетта-коэффициенты в уравнении ниже
- “Наилучшие” коэффициенты – те, которые минимизируют сумму квадратов ошибок

$$\hat{y} = \sum_{i=0}^n \beta_i x_i$$



Линейная регрессия

- Остаток / ошибка (residual error) для строки j вычисляется так:

$$y^j - \hat{y}^j$$



Линейная регрессия

- Квадрат ошибок для строки j :

$$\left(y^j - \hat{y}^j\right)^2$$



Линейная регрессия

- Сумма квадратов ошибок для m строк:

$$\sum_{j=1}^m \left(y^j - \hat{y}^j \right)^2$$



Линейная регрессия

- Средняя сумма квадратов ошибок для m строк:

$$\frac{1}{m} \sum_{j=1}^m \left(y^j - \hat{y}^j \right)^2$$



Линейная регрессия

- Средняя сумма квадратов ошибок для m строк
- Это как раз то, что нужно для функции стоимости!

$$\frac{1}{m} \sum_{j=1}^m \left(y^j - \hat{y}^j \right)^2$$



Линейная регрессия

- Функция стоимости измеряет степень ошибки



Линейная регрессия

- Функция стоимости измеряет степень (размер) ошибки
- Мы хотим минимизировать эту функцию



Линейная регрессия

- Функция стоимости измеряет степень (размер) ошибки
- Мы хотим минимизировать эту функцию
- Давайте определим функцию стоимости вот так:

$$J(\beta) = \frac{1}{2m} \sum_{j=1}^m \left(y^j - \hat{y}^j \right)^2$$



Линейная регрессия

- Функция стоимости измеряет степень (размер) ошибки
- Мы хотим минимизировать эту функцию
- Давайте определим функцию стоимости вот так:

$$J(\beta) = \frac{1}{2m} \sum_{j=1}^m (y^j - \hat{y}^j)^2$$

Функция от коэффициентов бетта.
Хотим минимизировать функцию.



Линейная регрессия

- Функция стоимости измеряет степень (размер) ошибки
- Мы хотим минимизировать эту функцию
- Давайте определим функцию стоимости вот так:

$$J(\beta) = \frac{1}{2m} \sum_{j=1}^m \left(y^j - \hat{y}^j \right)^2$$

Разница между настоящим значением y
и прогнозируемым значением $\hat{y}$ с крышкой



Линейная регрессия

- Функция стоимости измеряет степень (размер) ошибки
- Мы хотим минимизировать эту функцию
- Давайте определим функцию стоимости вот так:

$$J(\beta) = \frac{1}{2m} \sum_{j=1}^m \left(y^j - \hat{y}^j \right)^2$$

**Квадрат ошибки всегда положителен -
и для положительных, и для отрицательных ошибок**



Линейная регрессия

- Функция стоимости измеряет степень (размер) ошибки
- Мы хотим минимизировать эту функцию
- Давайте определим функцию стоимости вот так:

$$J(\beta) = \frac{1}{2m} \sum_{j=1}^m (y^j - \hat{y}^j)^2$$

Сумма для m строк



Линейная регрессия

- Функция стоимости измеряет степень (размер) ошибки
- Мы хотим минимизировать эту функцию
- Давайте определим функцию стоимости вот так:

$$J(\beta) = \frac{1}{2m} \sum_{j=1}^m \left(y^j - \hat{y}^j \right)^2$$

↑
Делим на m - получаем среднее значение



Линейная регрессия

- Функция стоимости измеряет степень (размер) ошибки
- Мы хотим минимизировать эту функцию
- Давайте определим функцию стоимости вот так:

$$J(\beta) = \frac{1}{2m} \sum_{j=1}^m \left(y^j - \hat{y}^j \right)^2$$

Делим ещё на 2 для удобства вычисления производных



Линейная регрессия

- Итого мы получили функцию от коэффициентов бетта и признаков X:

$$\begin{aligned} J(\beta) &= \frac{1}{2m} \sum_{j=1}^m \left(y^j - \hat{y}^j \right)^2 \\ &= \frac{1}{2m} \sum_{j=1}^m \left(y^j - \sum_{i=0}^n \beta_i x_i^j \right)^2 \end{aligned}$$



Линейная регрессия

- Чтобы минимизировать функцию, нужно взять её производную и приравнять к нулю:

$$\begin{aligned}\frac{\partial J}{\partial \beta_k}(\boldsymbol{\beta}) &= \frac{\partial}{\partial \beta_k} \left(\frac{1}{2m} \sum_{j=1}^m \left(y^j - \sum_{i=0}^n \beta_i x_i^j \right)^2 \right) \\ &= \frac{1}{m} \sum_{j=1}^m \left(y^j - \sum_{i=0}^n \beta_i x_i^j \right) (-x_k^j)\end{aligned}$$



Линейная регрессия

- Чтобы минимизировать функцию, нужно взять её производную и приравнять к нулю:

$$\frac{\partial J}{\partial \beta_k}(\beta) = \frac{\partial}{\partial \beta_k} \left(\frac{1}{2m} \sum_{j=1}^m \left(y^j - \sum_{i=0}^n \beta_i x_i^j \right)^2 \right)$$
$$= \frac{1}{m} \sum_{j=1}^m \left(y^j - \sum_{i=0}^n \beta_i x_i^j \right) (-x_k^j)$$



Линейная регрессия

- К сожалению, попытки решить это уравнение с помощью формул не приведут к масштабируемому решению
- В следующей лекции мы посмотрим, как решить эту задачу с помощью метода градиентного спуска



Линейная регрессия

Теория алгоритма - часть 4

градиентный спуск



Линейная регрессия

- Итак, у нас есть функция стоимости, которую нужно минимизировать
- Если взять производную этой функции и приравнять нулю, чтобы найти коэффициенты бетта, то аналитическое решение в виде формул будет слишком сложным



Линейная регрессия

- Вместо этого мы можем описать функцию стоимости в матричном виде, и применить метод градиентного спуска, чтобы компьютер сам нашёл коэффициенты бетта, минимизирующие функцию



Линейная регрессия

- Ранее мы нашли производную функции стоимости

$$\frac{\partial J}{\partial \beta_k}(\beta) = \frac{1}{m} \sum_{j=1}^m \left(y^j - \sum_{i=0}^n \beta_i x_i^j \right) (-x_k^j)$$



Линейная регрессия

- Ранее мы нашли производную функции стоимости
- Наши данные – матрица признаков X и вектор целевой переменной y

$$\frac{\partial J}{\partial \beta_k}(\beta) = \frac{1}{m} \sum_{j=1}^m \left(y^j - \sum_{i=0}^n \beta_i x_i^j \right) (-x_k^j)$$



Линейная регрессия

- Все производные по отдельным бетта- можно собрать в единый вектор производных

$$\nabla_{\beta} J = \begin{bmatrix} \frac{\partial J}{\partial \beta_0} \\ \vdots \\ \frac{\partial J}{\partial \beta_n} \end{bmatrix}$$



Линейная регрессия

- Все производные по отдельным бетта- можно собрать в единый вектор производных

$$\nabla_{\beta} J = \begin{bmatrix} \frac{\partial J}{\partial \beta_0} \\ \vdots \\ \frac{\partial J}{\partial \beta_n} \end{bmatrix} = \begin{bmatrix} -\frac{1}{m} \sum_{j=1}^m \left(y^j - \sum_{i=0}^n \beta_i x_i^j \right) x_0^j \\ \vdots \\ -\frac{1}{m} \sum_{j=1}^m \left(y^j - \sum_{i=0}^n \beta_i x_i^j \right) x_n^j \end{bmatrix}$$



Линейная регрессия

- Также мы знаем, как представить наши данные в векторном виде

$$\mathbf{X} = \begin{bmatrix} 1 & x_1^1 & x_2^1 & \dots & x_n^1 \\ 1 & x_1^2 & x_2^2 & \dots & x_n^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^m & x_2^m & \dots & x_n^m \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} \quad \boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_n \end{bmatrix}$$



Линейная регрессия

- Разобъём производную на две составляющие

$$\nabla_{\beta} J = \begin{bmatrix} -\frac{1}{m} \sum_{j=1}^m \left(y^j - \sum_{i=0}^n \beta_i x_i^j \right) x_0^j \\ \vdots \\ -\frac{1}{m} \sum_{j=1}^m \left(y^j - \sum_{i=0}^n \beta_i x_i^j \right) x_n^j \end{bmatrix}$$



Линейная регрессия

- Получаем следующее:

$$\nabla_{\beta} J = -\frac{1}{m} \begin{bmatrix} \sum_{j=1}^m y^j x_0^j \\ \vdots \\ \sum_{j=1}^m y^j x_n^j \end{bmatrix} + \frac{1}{m} \begin{bmatrix} \sum_{j=1}^m \sum_{i=0}^n \beta_i x_i^j x_0^j \\ \vdots \\ \sum_{j=1}^m \sum_{i=0}^n \beta_i x_i^j x_n^j \end{bmatrix}$$



Линейная регрессия

- Теперь мы можем вычислить производную для любого набора коэффициентов бетта

$$\nabla_{\beta} J = -\frac{1}{m} \begin{bmatrix} \sum_{j=1}^m y^j x_0^j \\ \vdots \\ \sum_{j=1}^m y^j x_n^j \end{bmatrix} + \frac{1}{m} \begin{bmatrix} \sum_{j=1}^m \sum_{i=0}^n \beta_i x_i^j x_0^j \\ \vdots \\ \sum_{j=1}^m \sum_{i=0}^n \beta_i x_i^j x_n^j \end{bmatrix}$$



Линейная регрессия

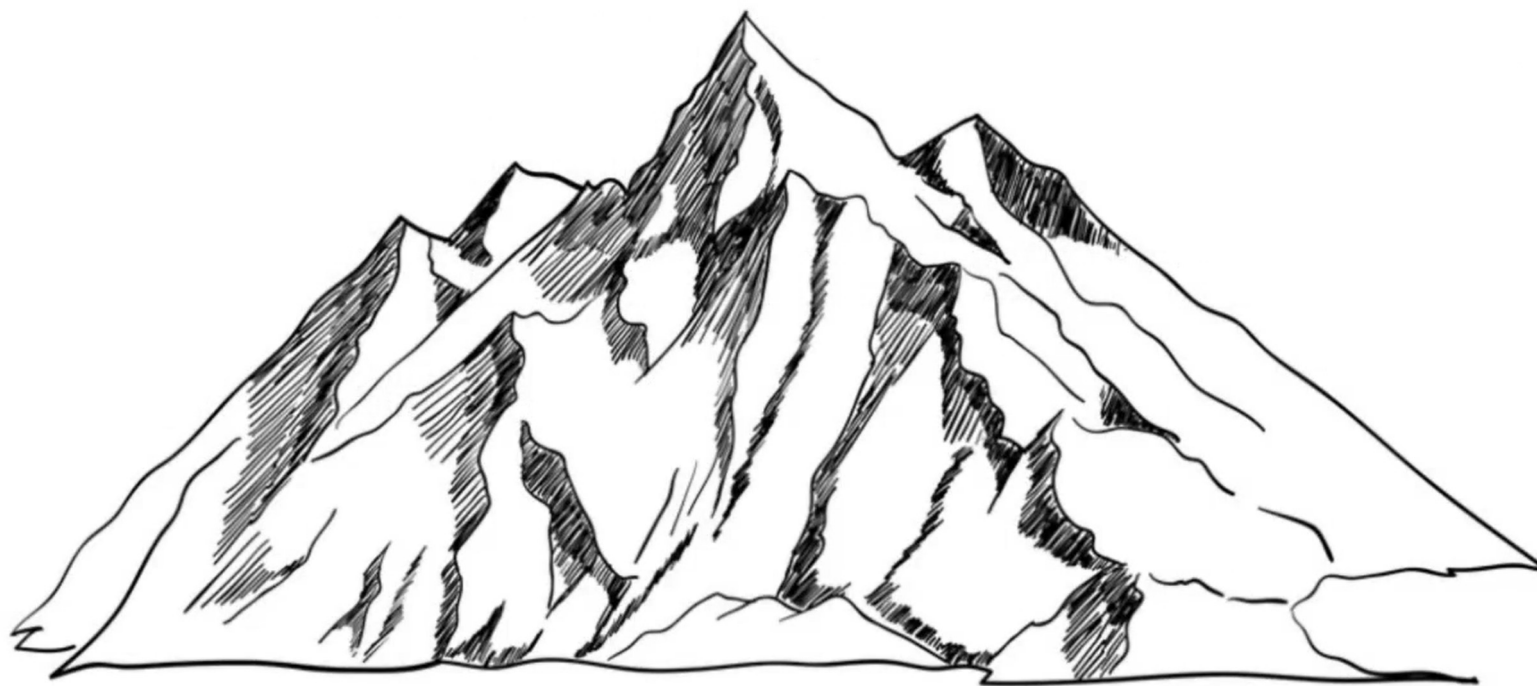
- Теперь мы можем вычислить производную для любого набора коэффициентов бетта

$$\nabla_{\beta} J = -\frac{1}{m} \begin{bmatrix} \sum_{j=1}^m y^j x_0^j \\ \vdots \\ \sum_{j=1}^m y^j x_n^j \end{bmatrix} + \frac{1}{m} \begin{bmatrix} \sum_{j=1}^m \sum_{i=0}^n \beta_i x_i^j x_0^j \\ \vdots \\ \sum_{j=1}^m \sum_{i=0}^n \beta_i x_i^j x_n^j \end{bmatrix}$$



Линейная регрессия

- Аналогия с вершиной горы





Линейная регрессия

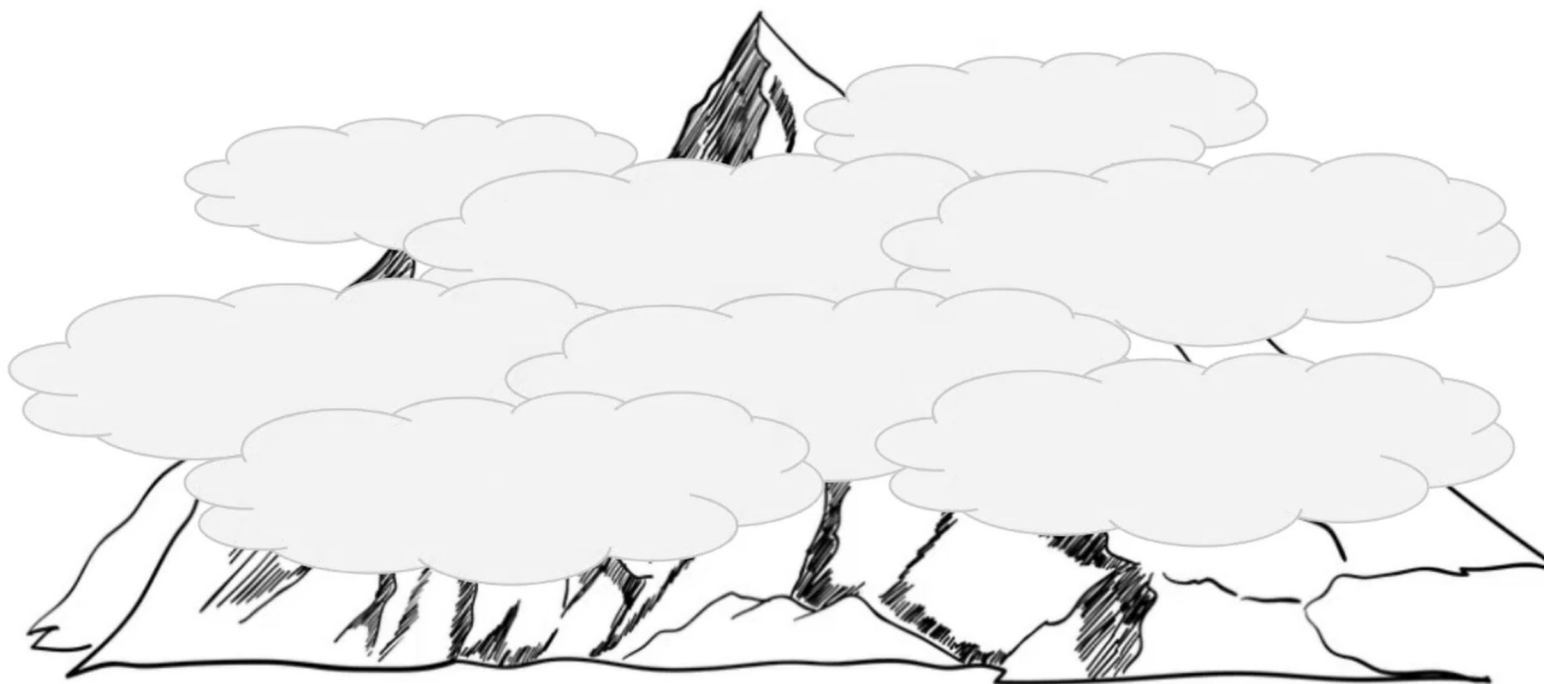
- Аналогия с вершиной горы





Линейная регрессия

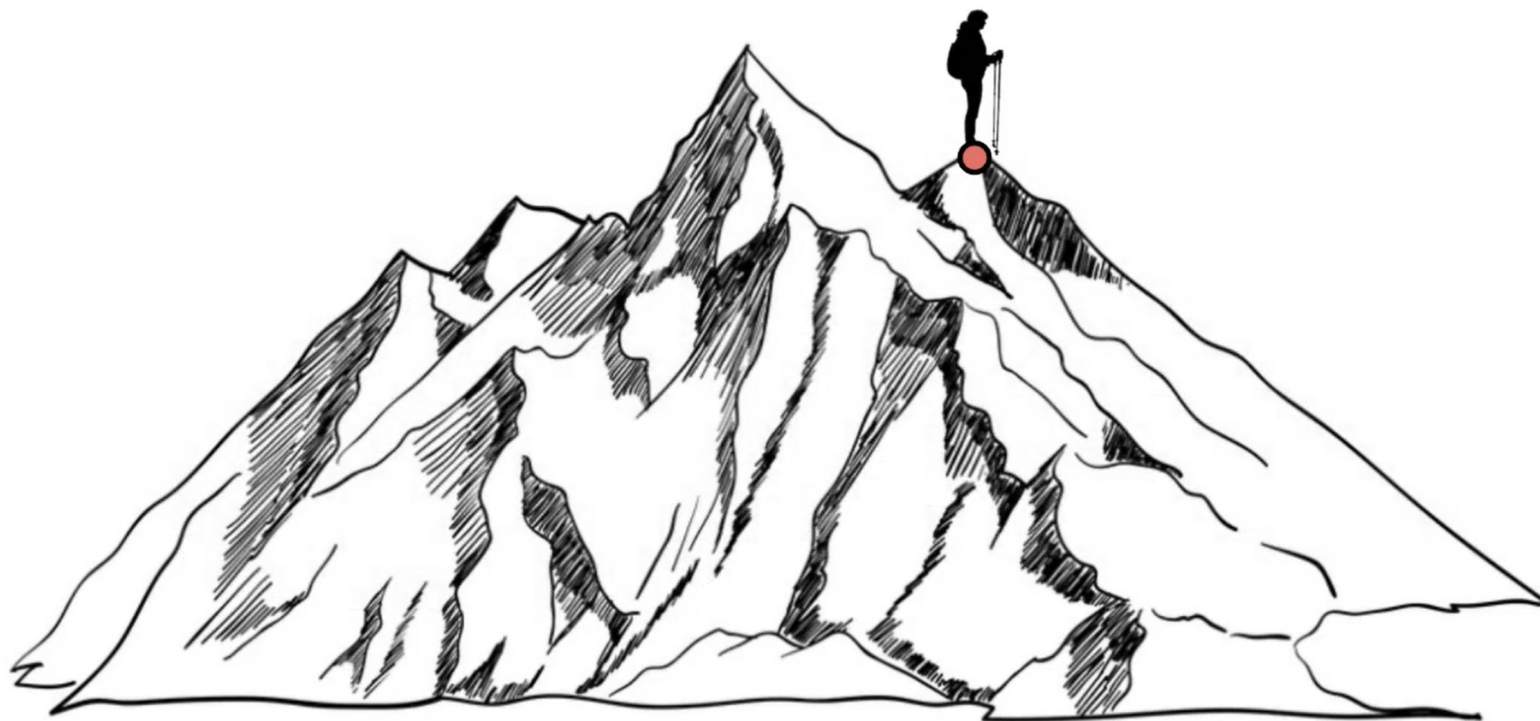
- Аналогия с вершиной горы





Линейная регрессия

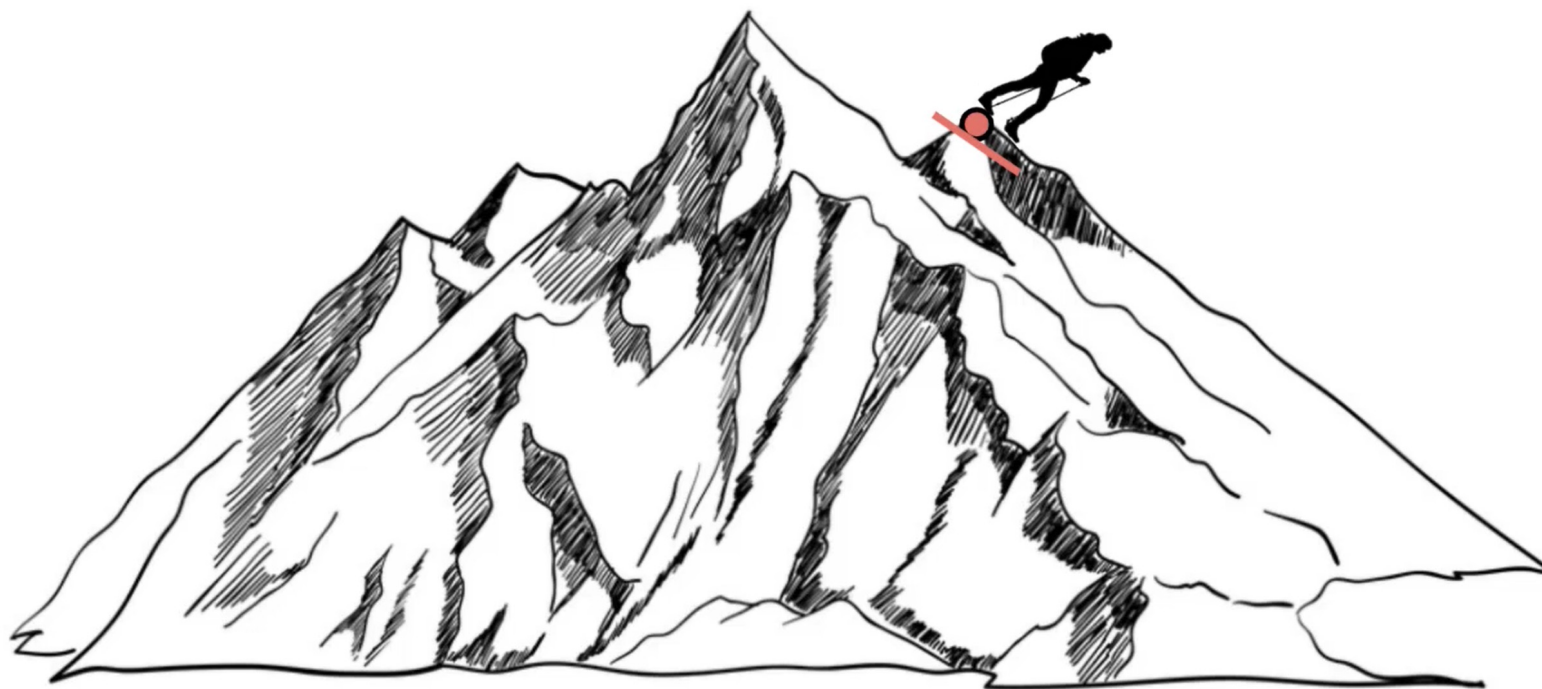
- Аналогия с вершиной горы





Линейная регрессия

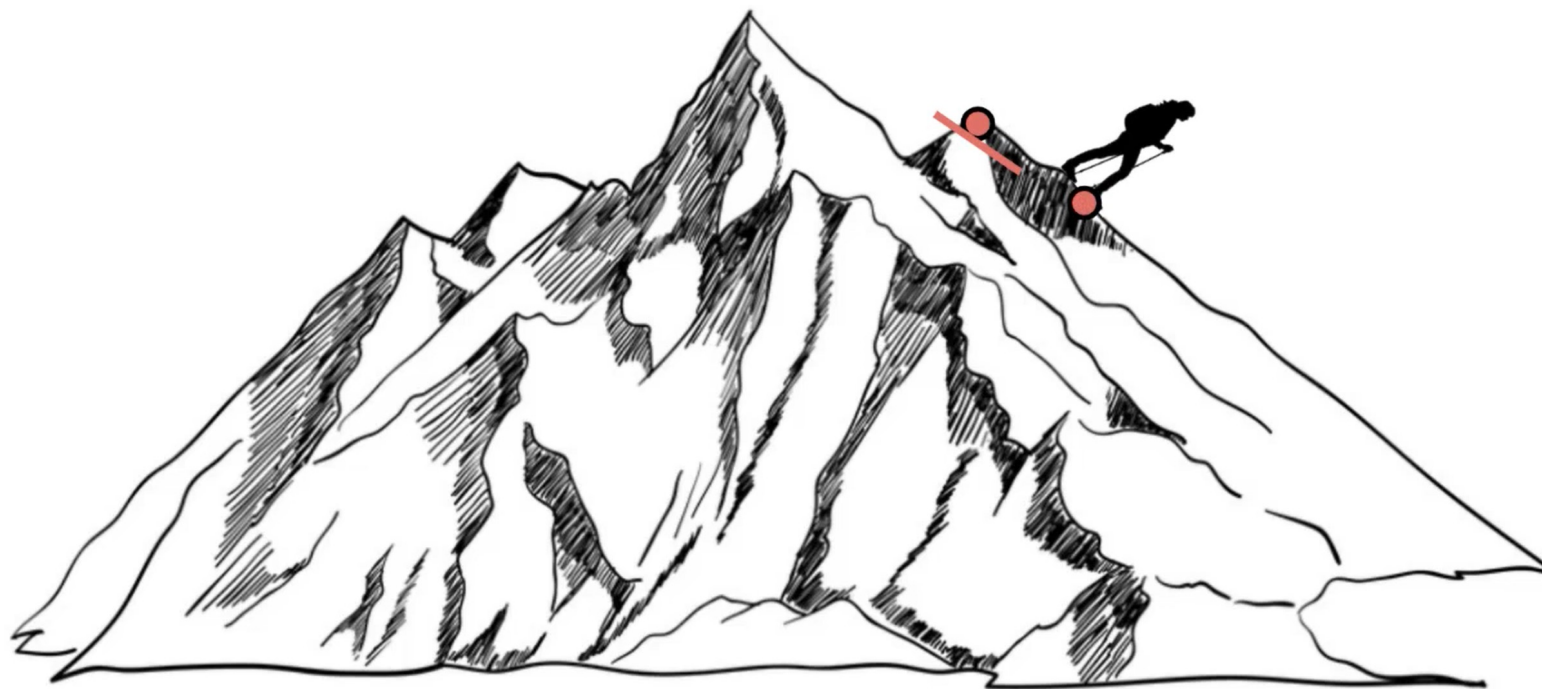
- Аналогия с вершиной горы





Линейная регрессия

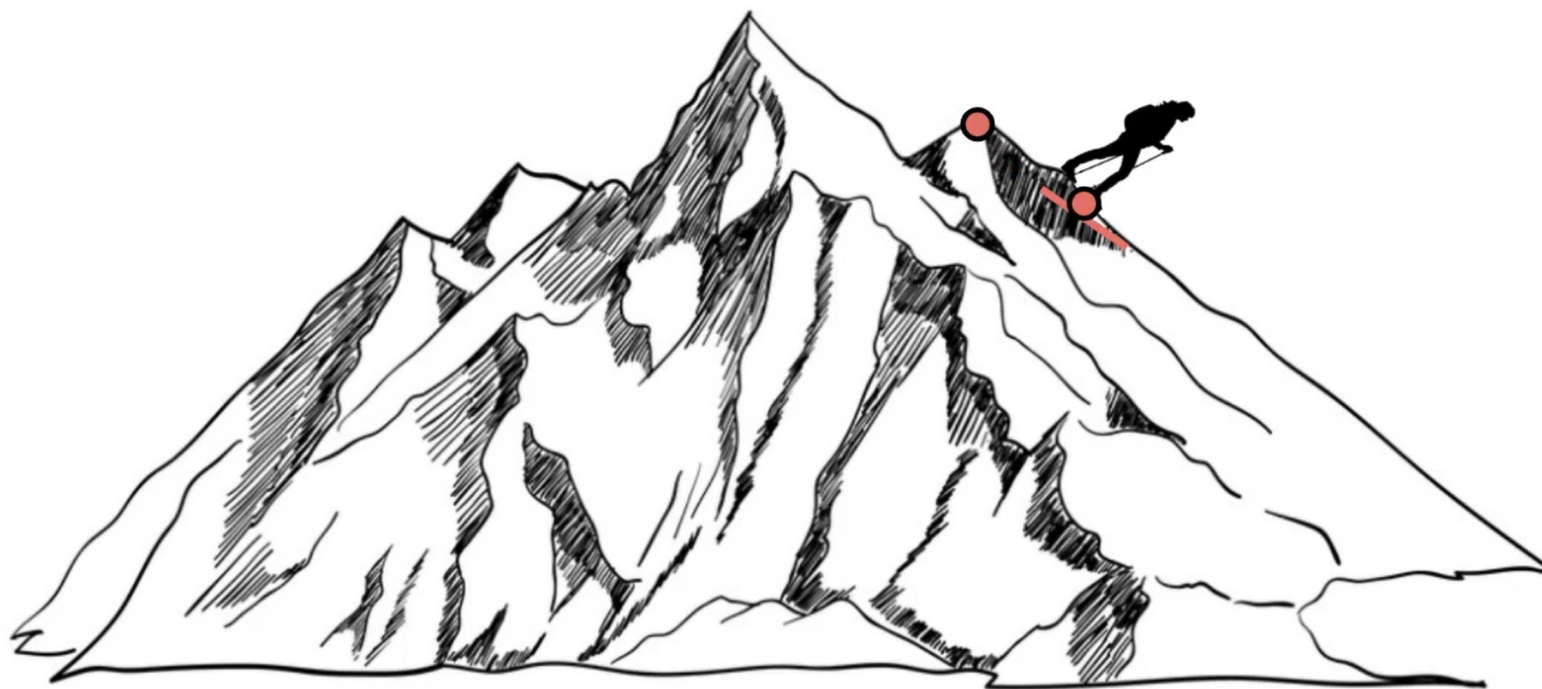
- Аналогия с вершиной горы





Линейная регрессия

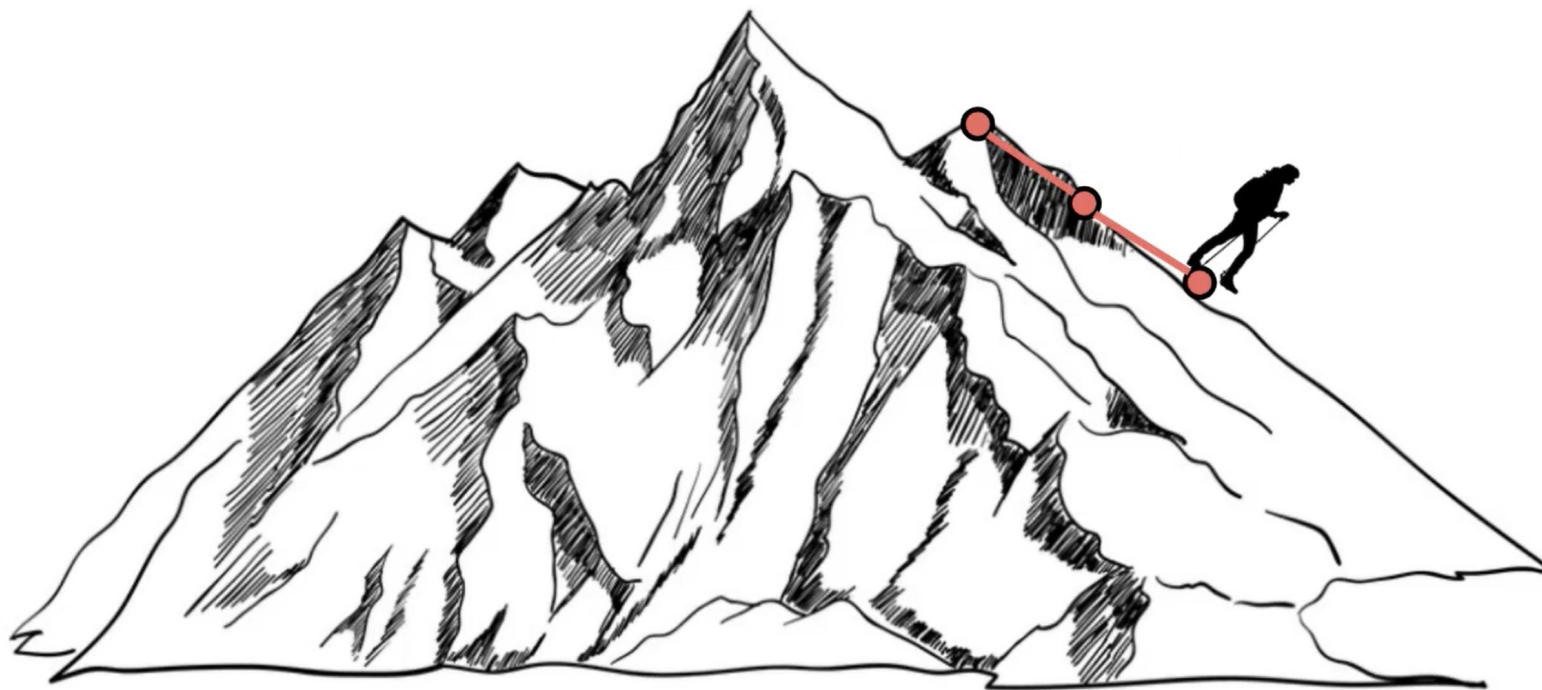
- Аналогия с вершиной горы





Линейная регрессия

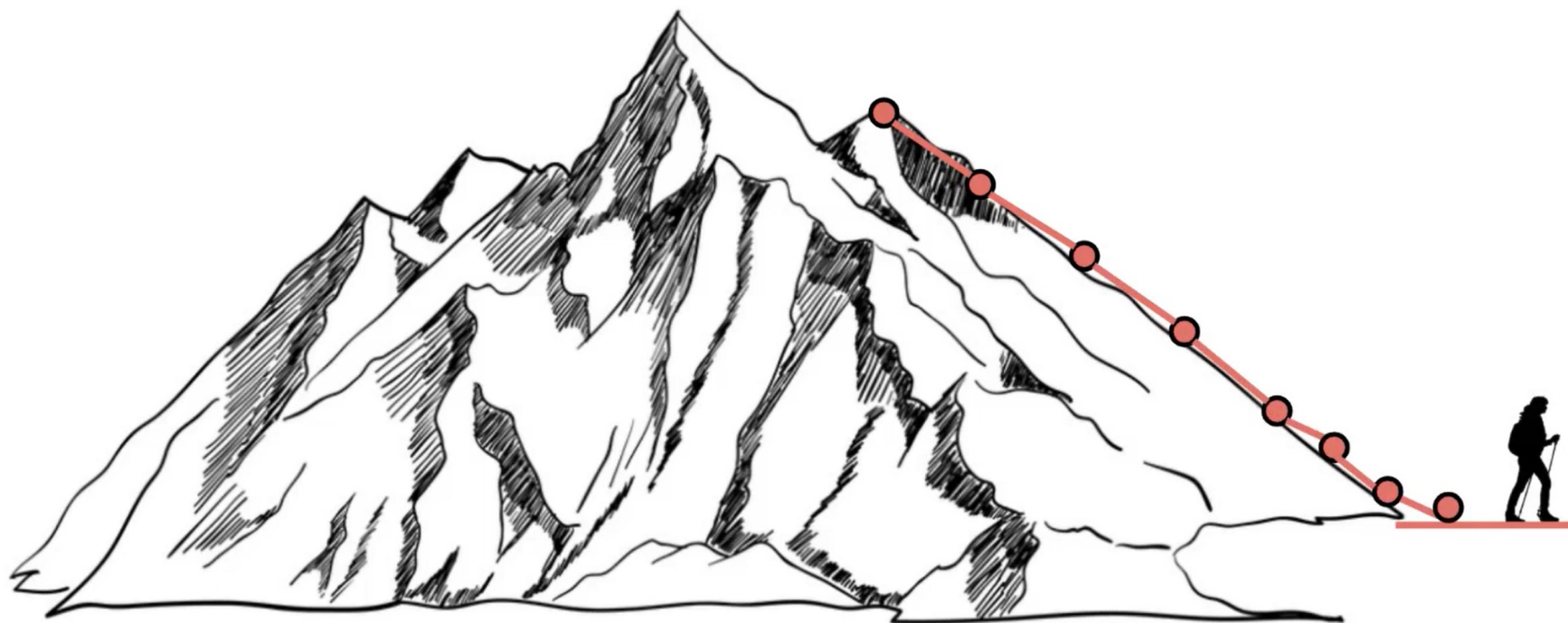
- Аналогия с вершиной горы





Линейная регрессия

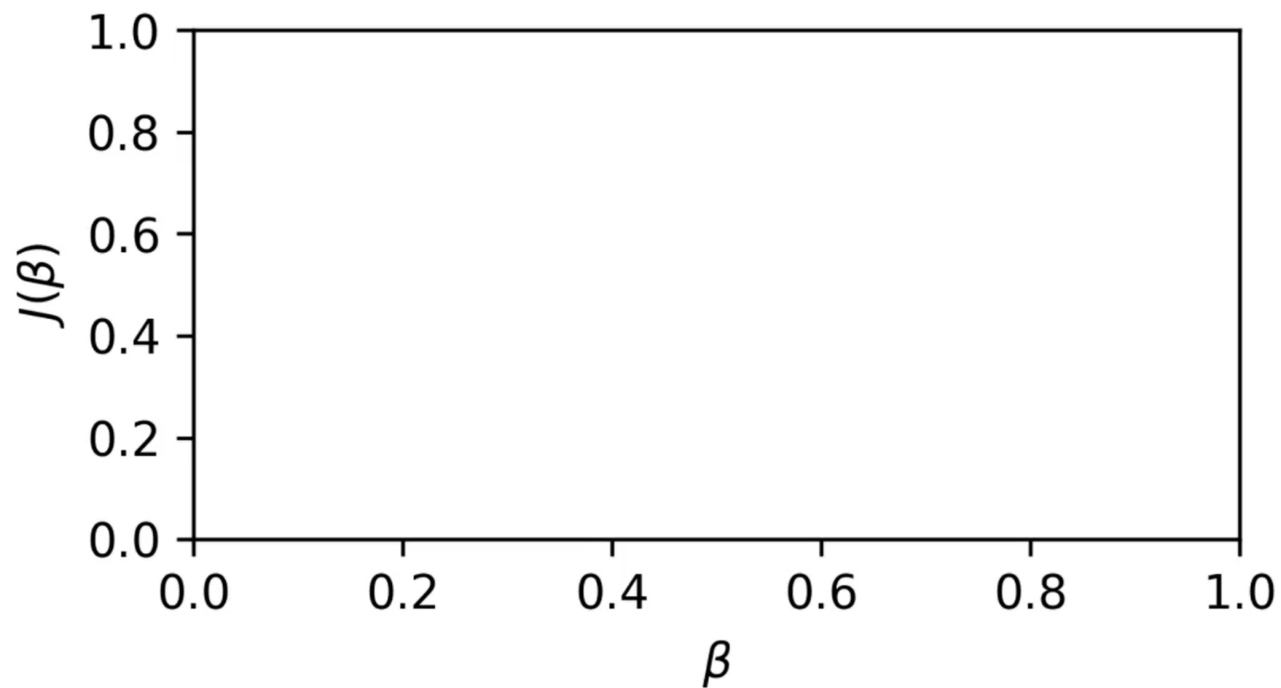
- Аналогия с вершиной горы





Линейная регрессия

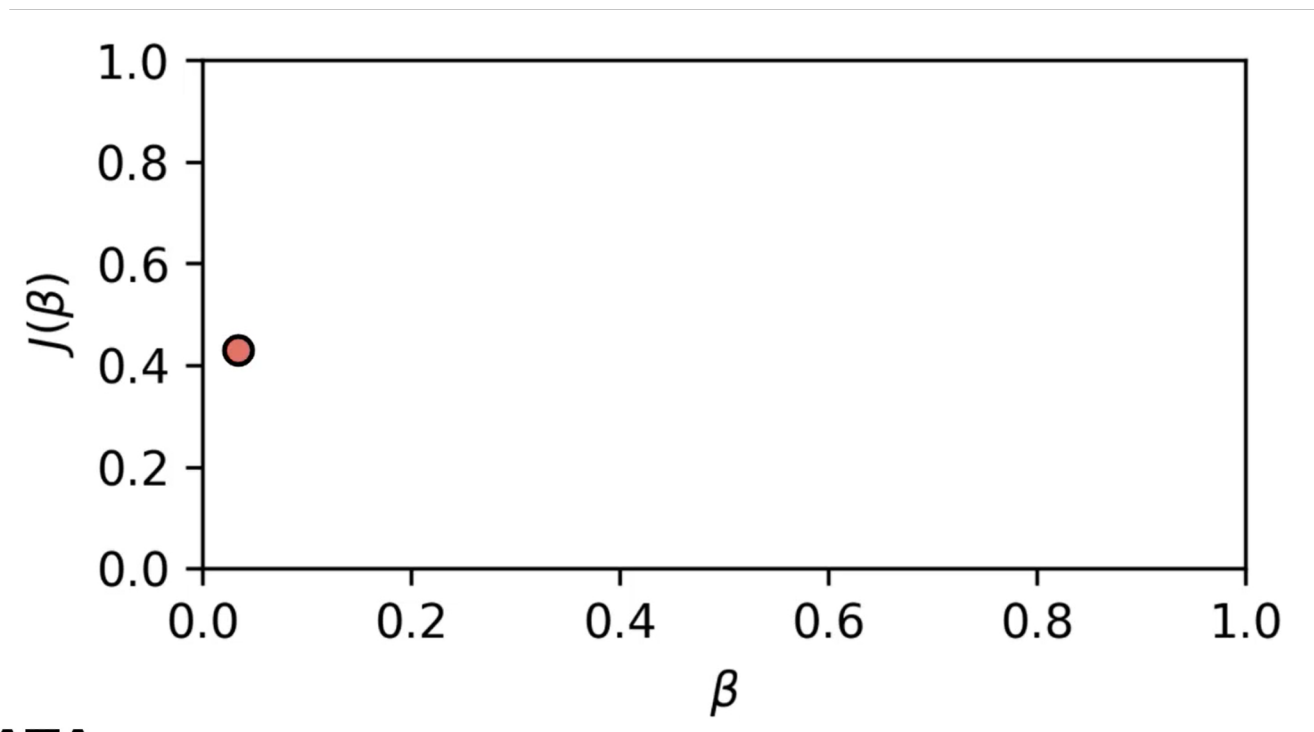
- Одномерная функция стоимости – одно бетта





Линейная регрессия

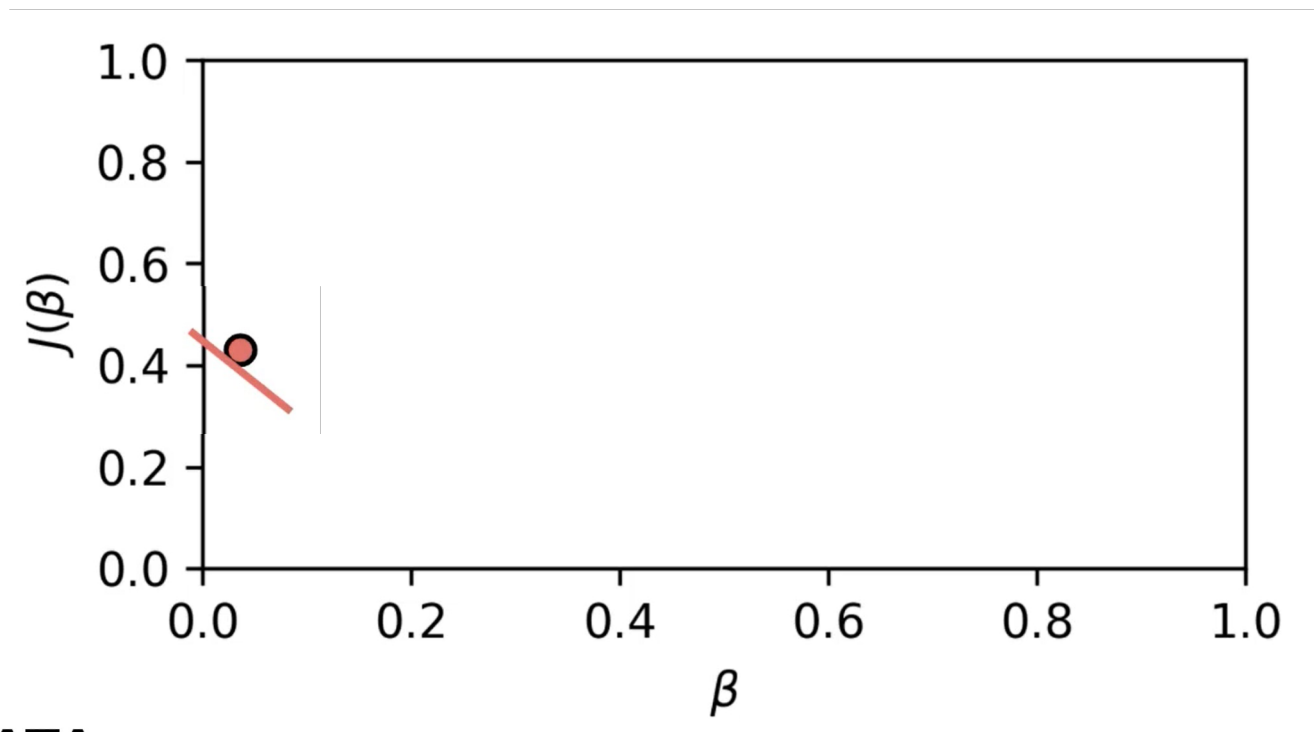
- Выбираем начальную точку





Линейная регрессия

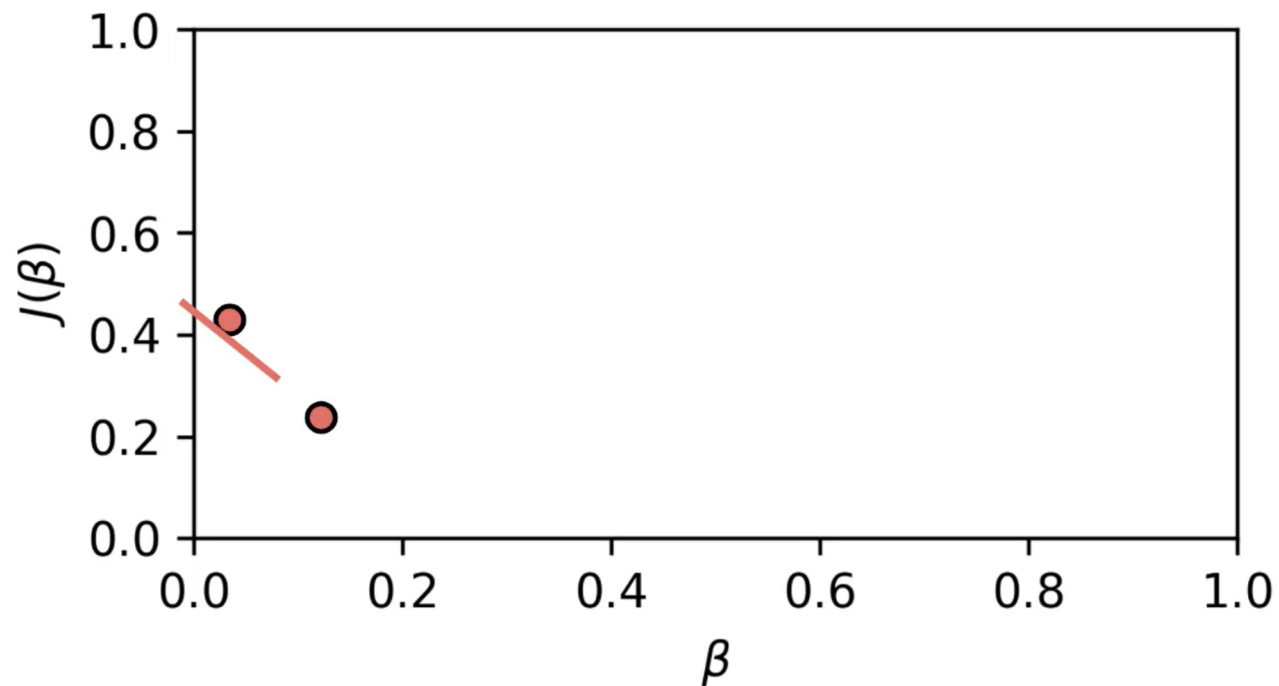
- Делаем шаг пропорционально производной * -1





Линейная регрессия

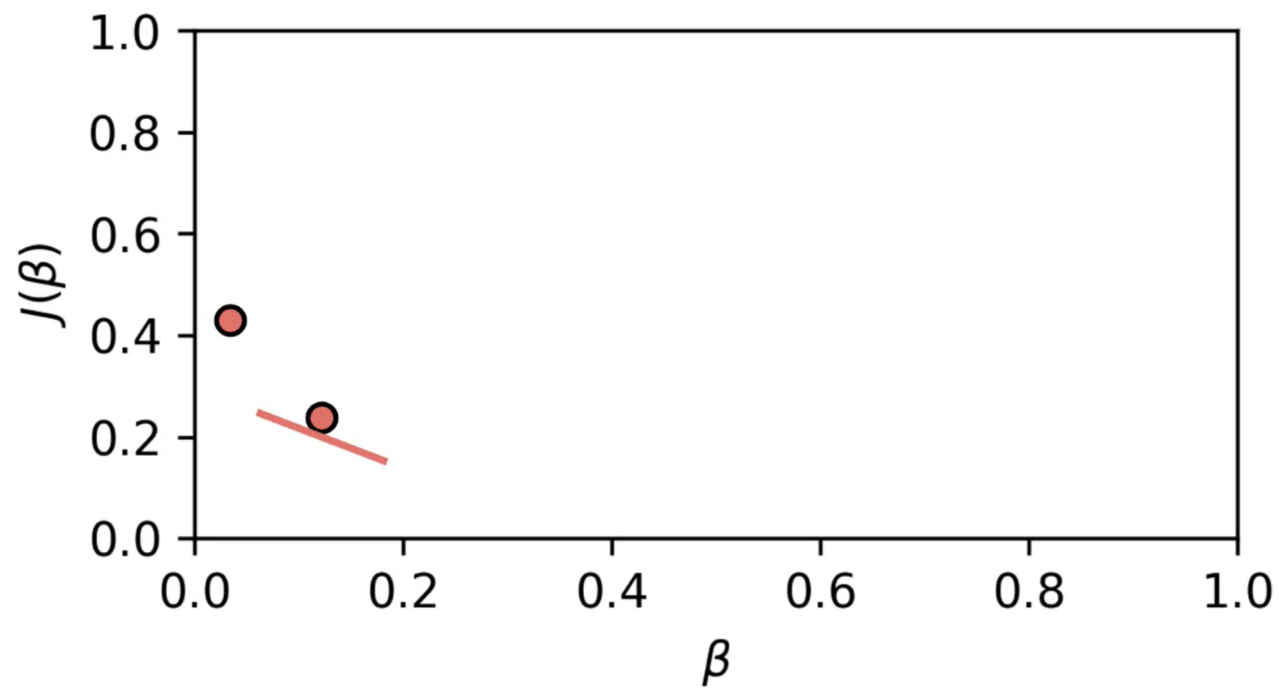
- Делаем шаг пропорционально производной * -1





Линейная регрессия

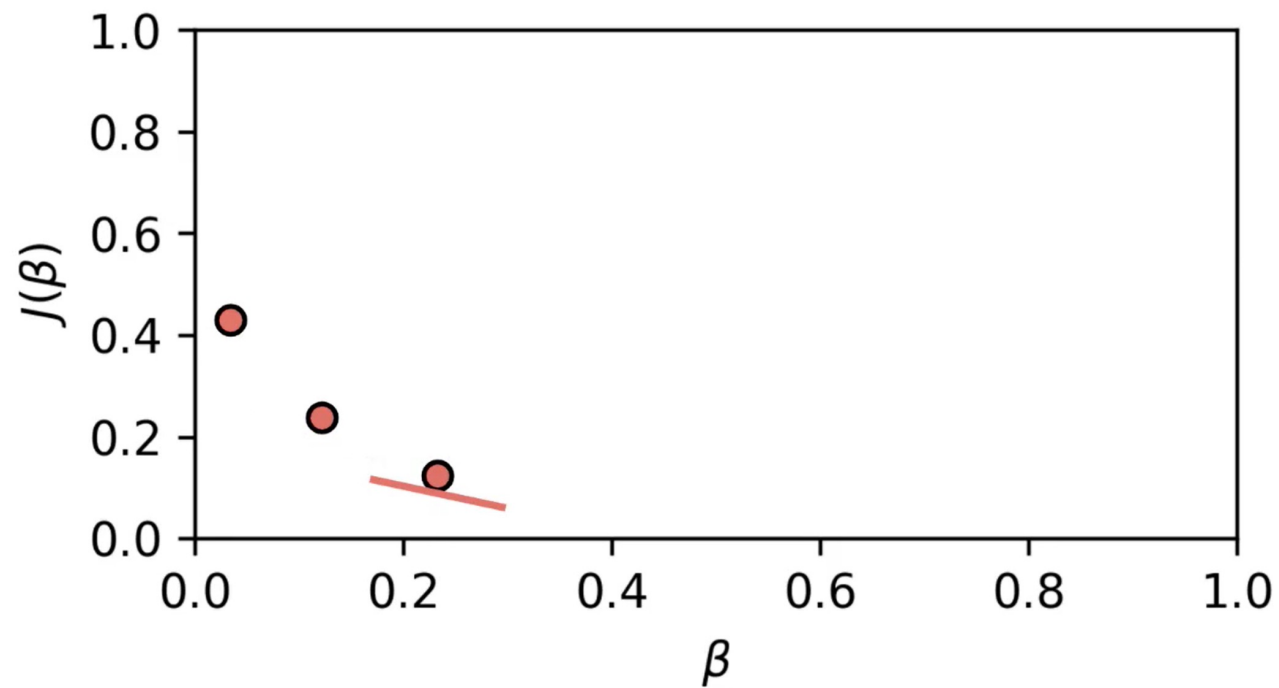
- Повторяем шаги





Линейная регрессия

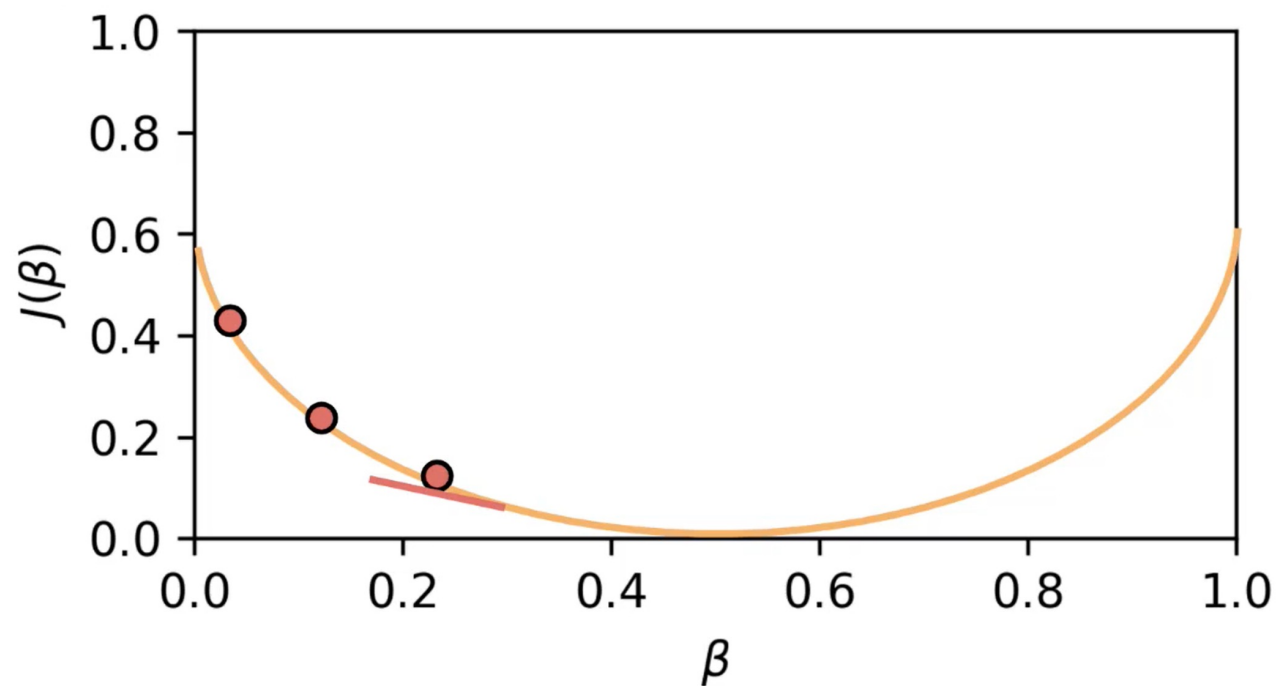
- Повторяем шаги





Линейная регрессия

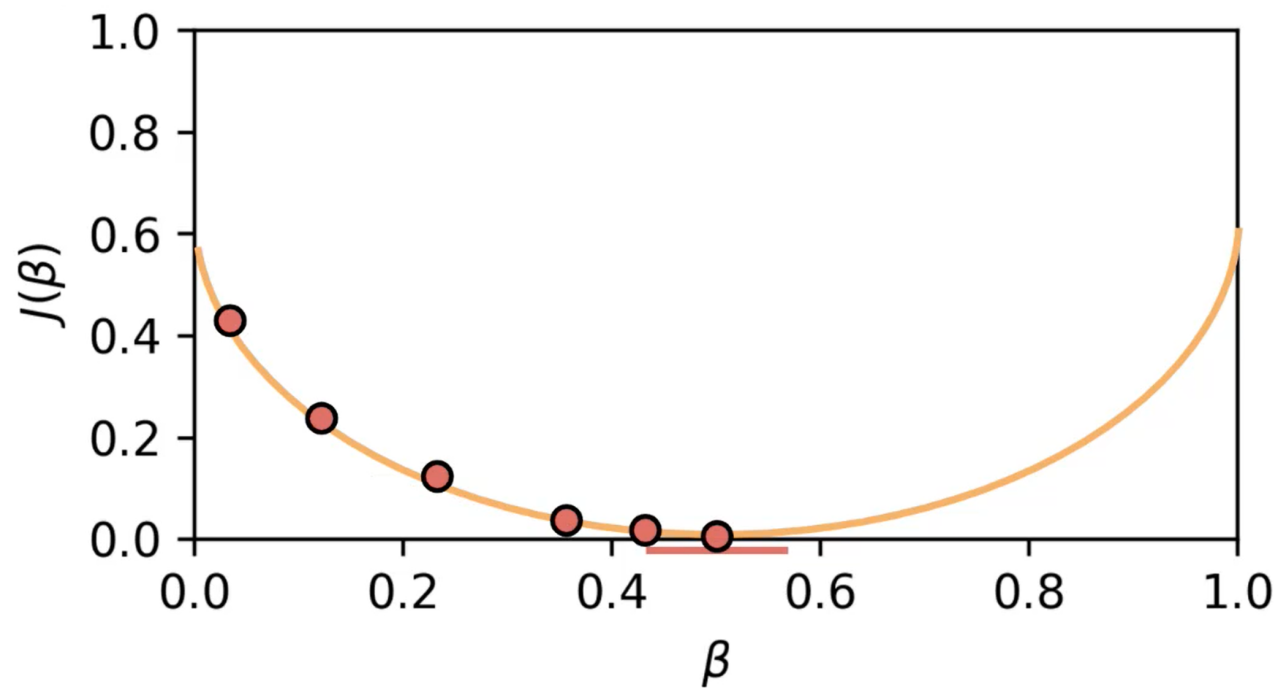
- По сути мы рисуем производную (градиент)!





Линейная регрессия

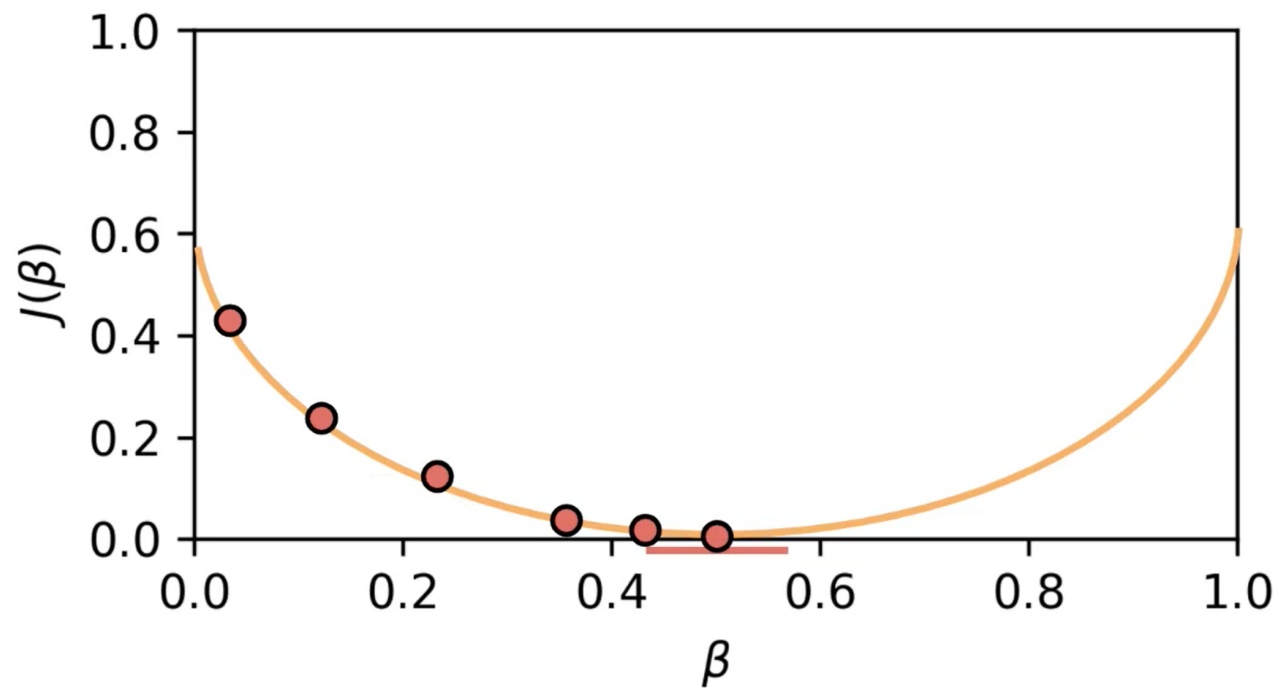
- В конце концов мы найдём минимум!





Линейная регрессия

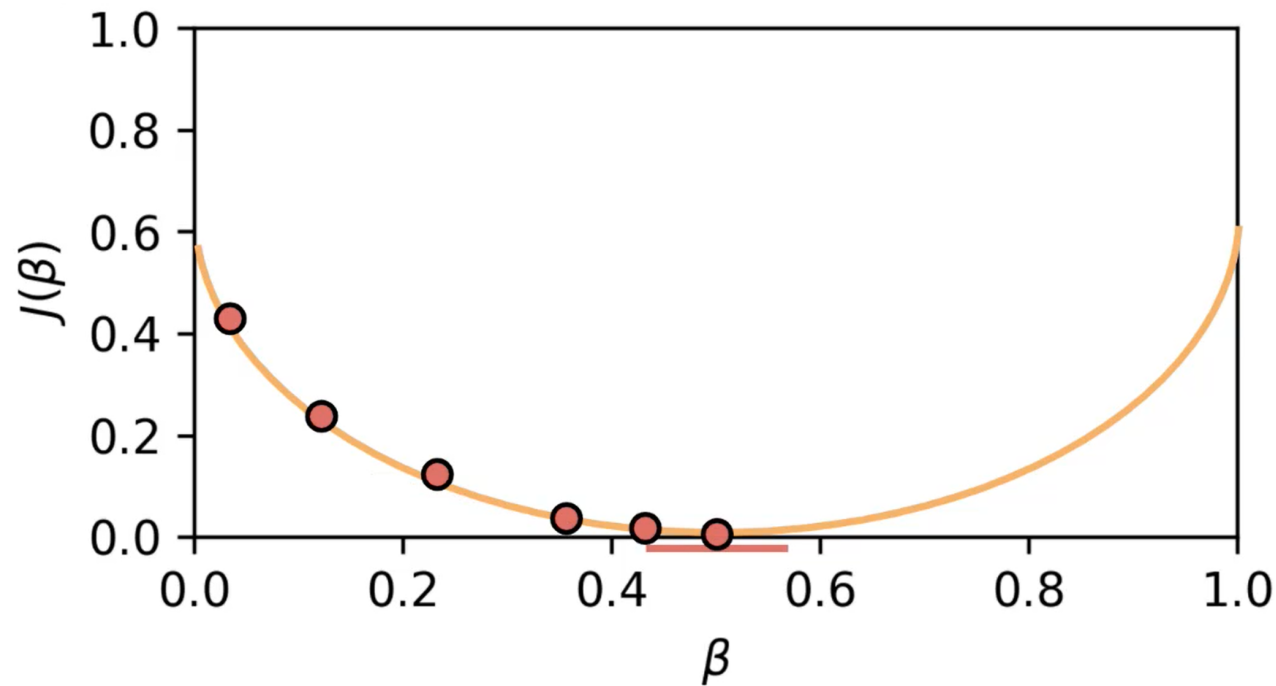
- Шаги пропорциональны отрицательному градиенту





Линейная регрессия

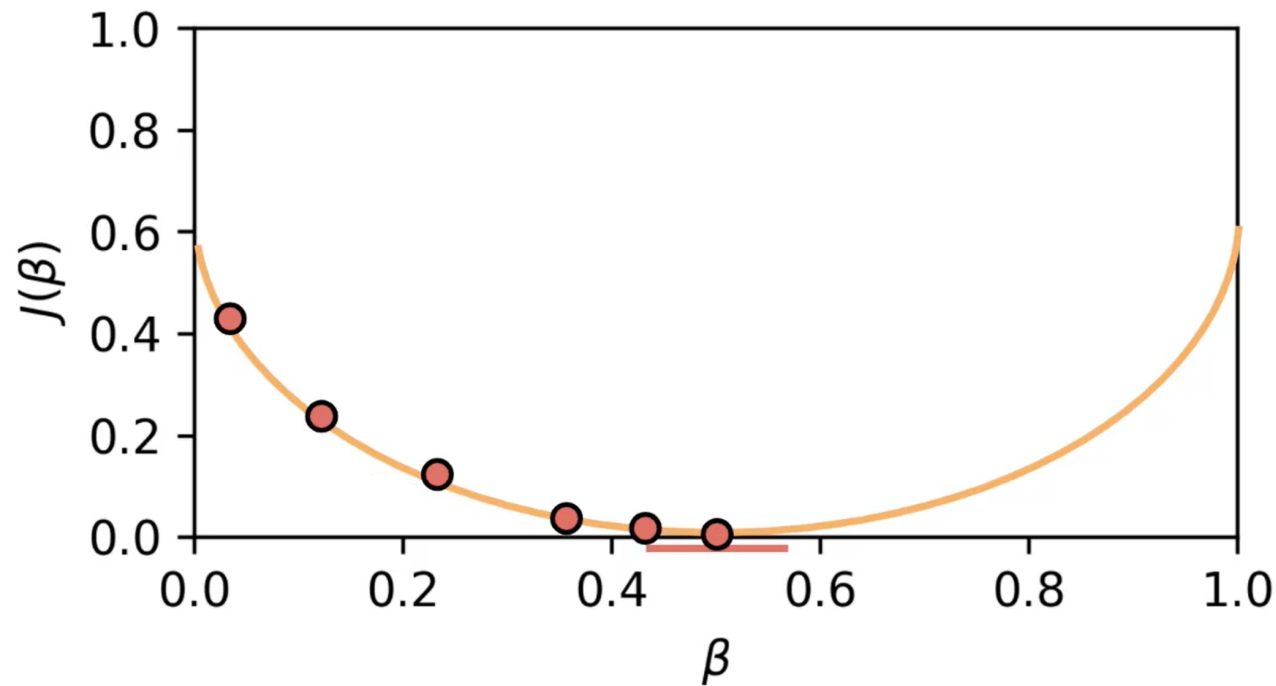
- Шаги пропорциональны отрицательному градиенту
- Сначала шаги большие, затем они меньше





Линейная регрессия

- Шаги пропорциональны отрицательному градиенту
- Сначала шаги большие, затем они меньше



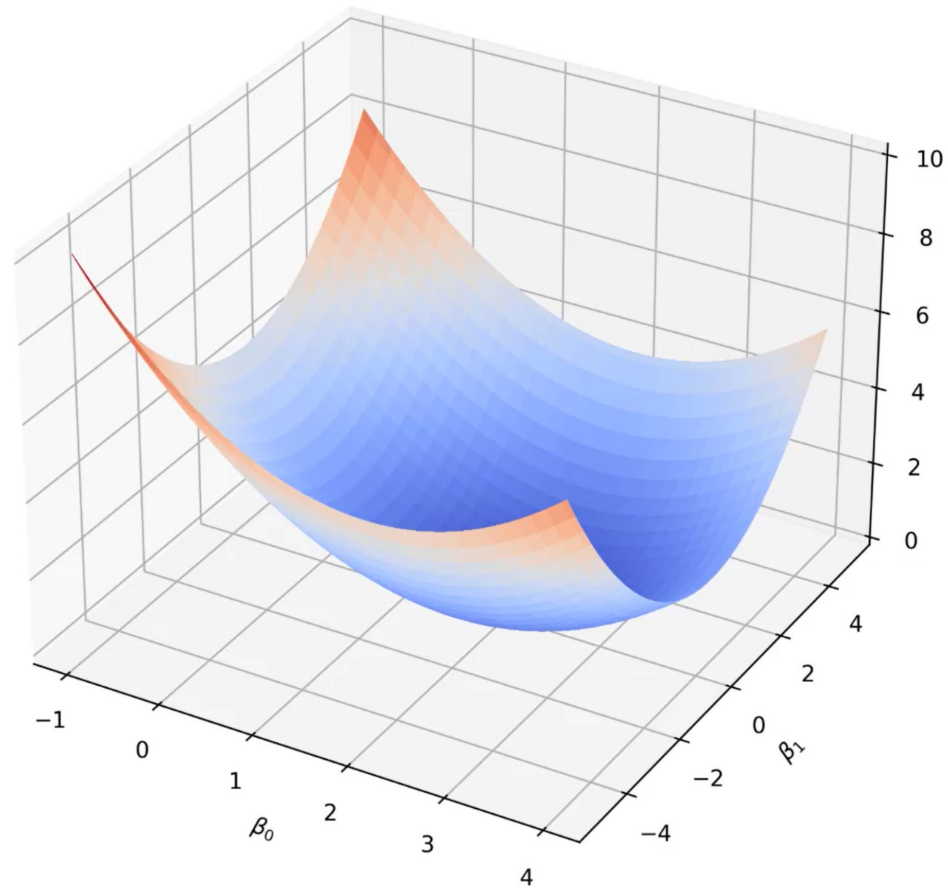


Линейная регрессия

- Рассмотрим случай двух переменных
- Шаги те же самые:
 - Вычисляем в точке производную (градиент)
 - Двигаемся в направлении, обратном градиенту
 - Величина шага пропорциональна градиенту
 - Повторяем шаги, пока не найдём минимум функции

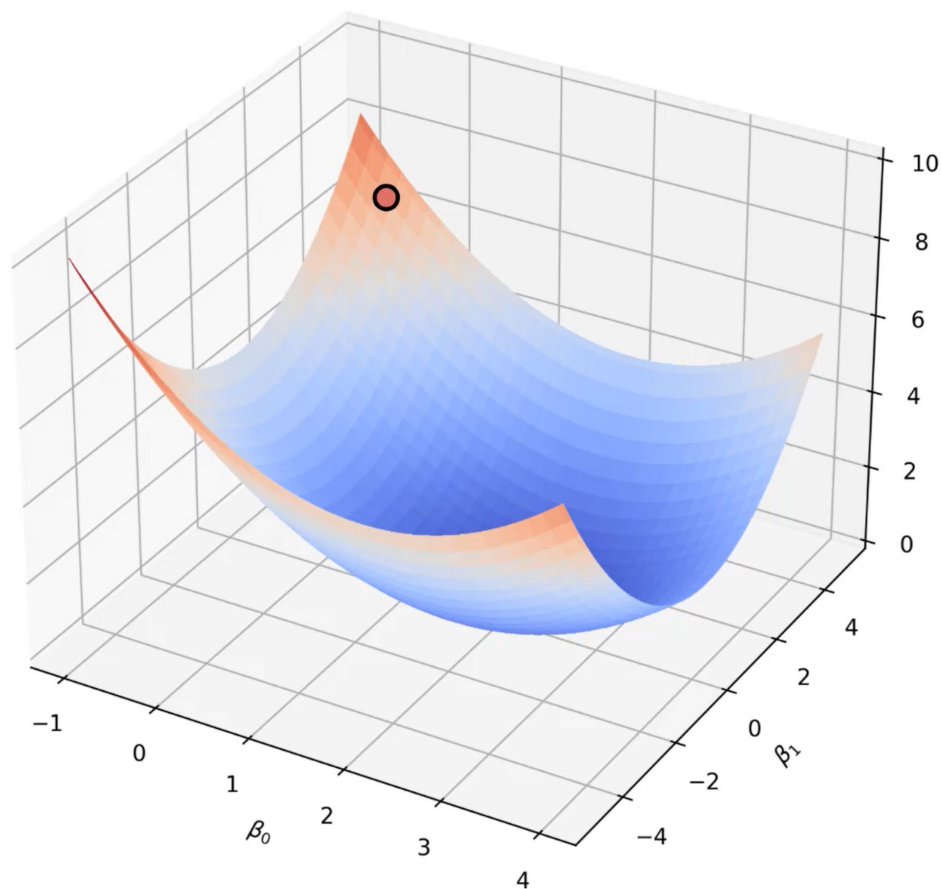


Линейная регрессия



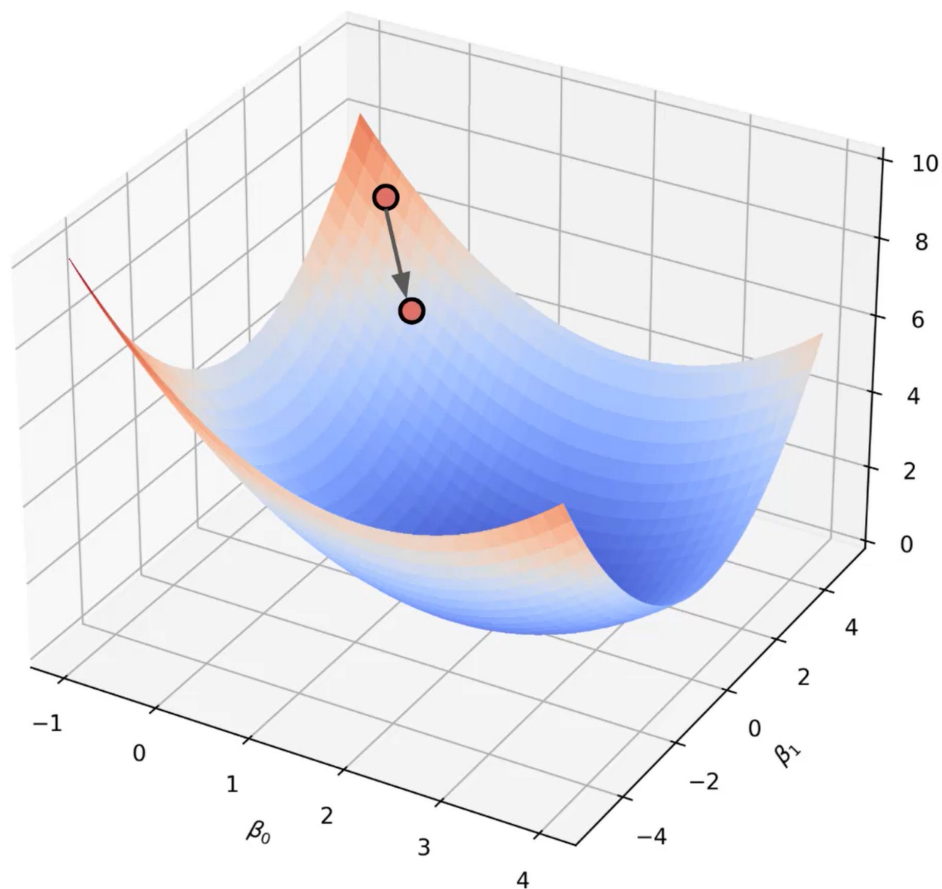


Линейная регрессия



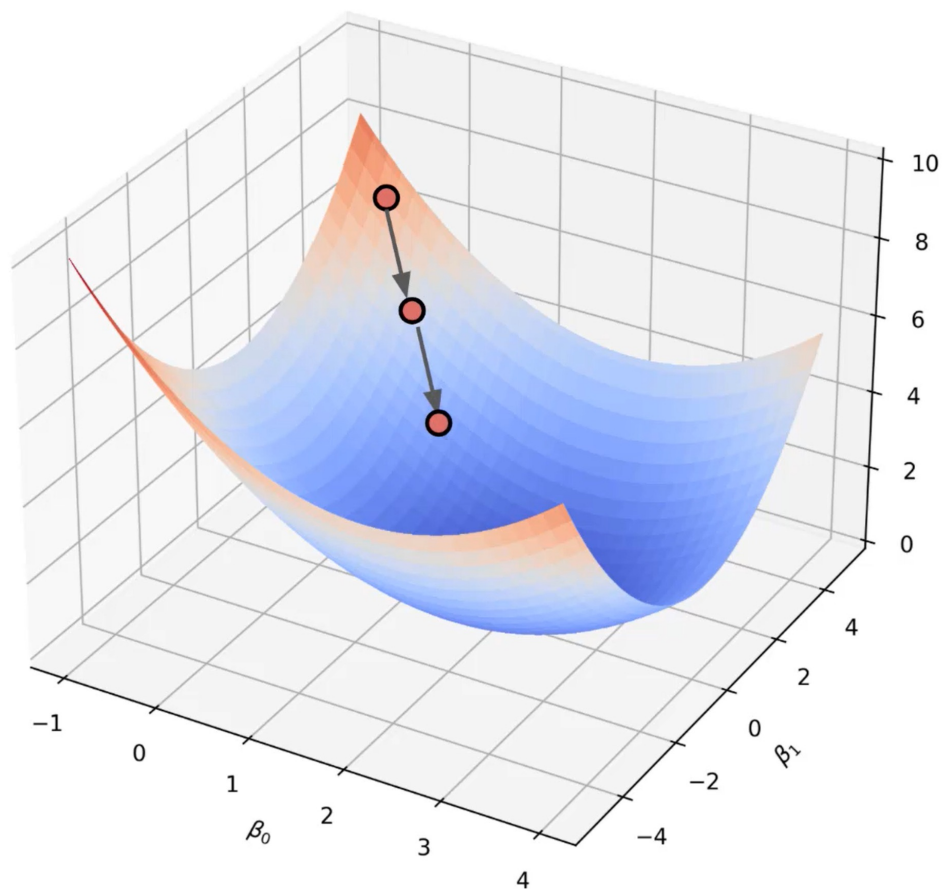


Линейная регрессия



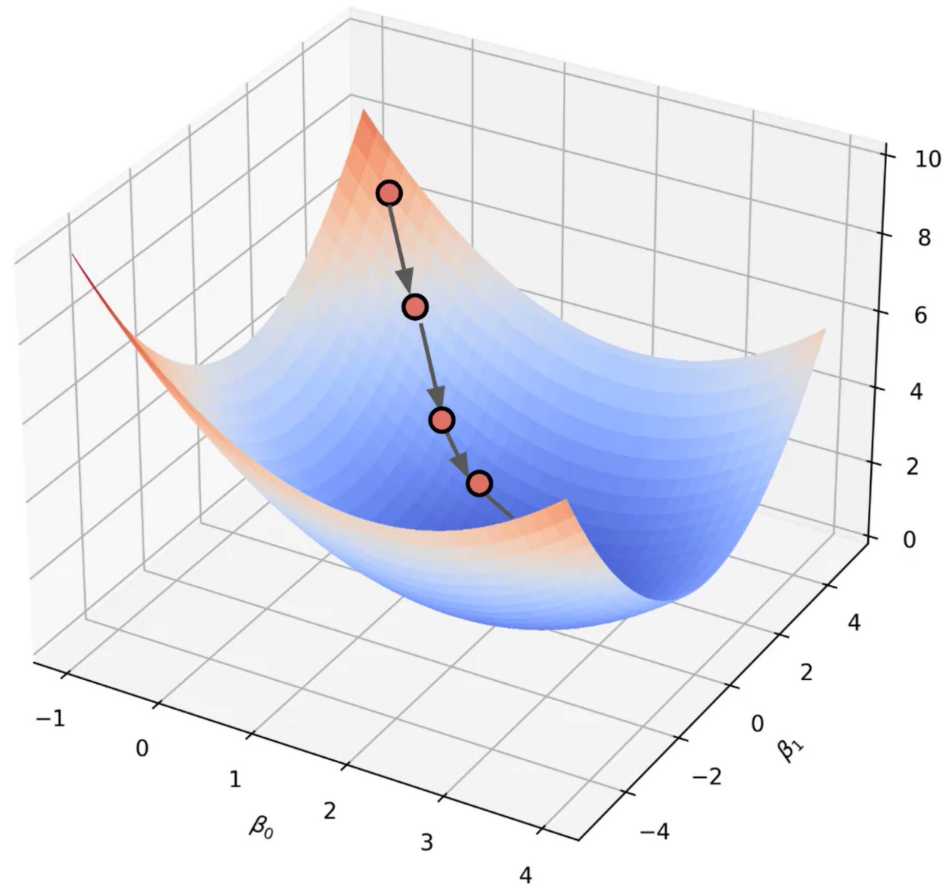


Линейная регрессия



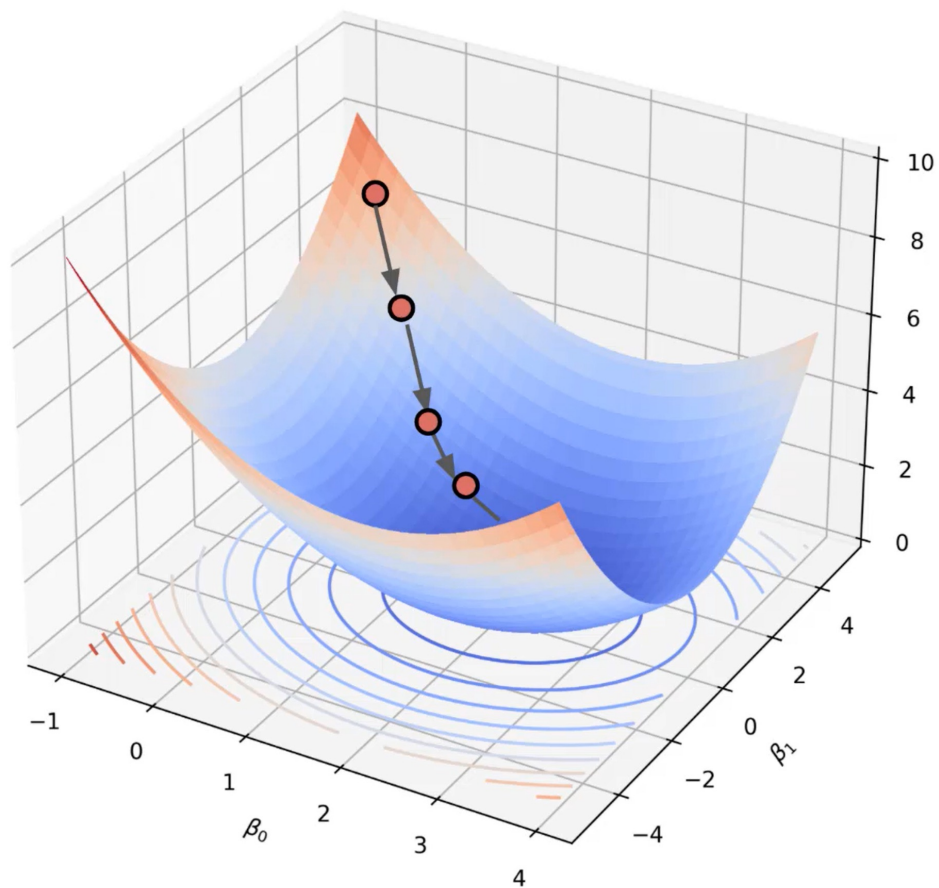


Линейная регрессия



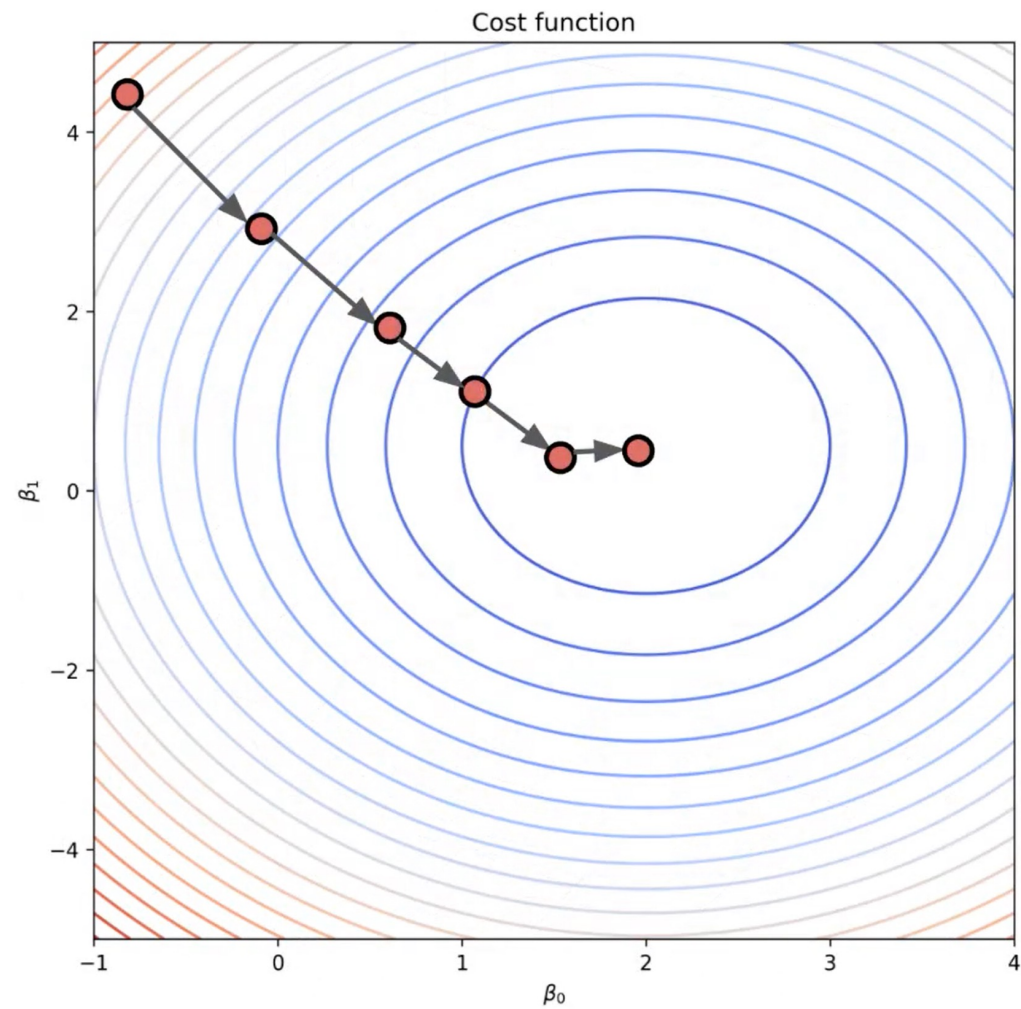


Линейная регрессия





Линейная регрессия





Линейная регрессия

- Итак, мы наконец-то знаем, как можно найти оптимальные коэффициенты бетта, минимизирующие функцию стоимости – чтобы найти оптимальную линию регрессии!
- Мы готовы к тому, чтобы написать код для линейной регрессии



Простая линейная регрессия



Простая линейная регрессия

- Теперь мы знаем, как линейная регрессия устроена внутри. Давайте сначала напишем код для простой линейной регрессии.
- В этой лекции мы также начнём думать об оценке эффективности работы модели, а также о регрессии с несколькими переменными.



Простая линейная регрессия

- Простая линейная регрессия использует только одну переменную X : $y=mx+b$
- Мы рассмотрим пример с данными о рекламе, где изучим линейную связь между общим бюджетом на рекламу и продажами товаров
- Давайте перейдём в блокнот!



Обзор Scikit-Learn



Обзор Scikit-Learn

- Как мы видели, в NumPy есть встроенные методы для простой линейной регрессии. Однако для более сложных моделей нам определённо понадобится Scikit-Learn!
- Прежде чем переходить к машинному обучению в Scikit-Learn и Python, давайте поймём общую философию sklearn.



Обзор Scikit-Learn

- Scikit-Learn – это библиотека со многими алгоритмами машинного обучения.
- Для работы с моделями применяется общий фреймворк "Estimator API".
- Это значит, что методы импорта, обучения и использования алгоритмов выполняются одинаково для разных алгоритмов.



Обзор Scikit-Learn

- Такой подход позволяет быстро заменять один алгоритм на другой, проверяя разные подходы.
- Важное замечание:
 - Как следствие, пользователи могут применять практически любой алгоритм, даже не до конца понимая, что делает тот или иной алгоритм!



Обзор Scikit-Learn

- Scikit-Learn содержит много полезных методов, включая функции разбиения данных на обучающий и тестовый наборы, функции кросс-валидации, а также метрики оценки модели
- В итоге эта библиотека содержит всё необходимое для многих наших задач по машинному обучению



Обзор Scikit-Learn

- Философия Scikit-Learn:
 - При построении моделей особое внимание уделяется применению моделей и оценке их эффективности
 - Это более практичный подход по сравнению с академическим подходом, где описываются модели и её параметры



Обзор Scikit-Learn

- Философия Scikit-Learn:
 - Академические пользователи, привыкшие к отчётам в стиле языка R, при работе с Python могут обратить внимание на библиотеку statsmodels, в которой можно найти более детальное статистическое описание моделей



Процесс машинного обучения с учителем (supervised learning)

- Вспомним разбиение данных на обучающий и тестовый наборы данных



TRAIN

TEST

Area m ²	Bedrooms	Bathrooms	Price
200	3	2	\$500,000
190	2	1	\$450,000
230	3	3	\$650,000
180	1	1	\$400,000
210	2	2	\$550,000



Процесс машинного обучения с учителем (supervised learning)

- Вспомним также, что данные у нас разбиваются на 4 компонента:

	Area m ²	Bedrooms	Bathrooms	Price	
X TRAIN	200	3	2	\$500,000	Y TRAIN
	190	2	1	\$450,000	
	230	3	3	\$650,000	
X TEST	180	1	1	\$400,000	Y TEST
	210	2	2	\$550,000	



Процесс машинного обучения с учителем (supervised learning)

- Scikit-Learn может легко выполнить такое разбиение данных (а также более продвинутый вариант – кросс-валидацию)



TRAIN

TEST

Area m ²	Bedrooms	Bathrooms	Price
200	3	2	\$500,000
190	2	1	\$450,000
230	3	3	\$650,000
180	1	1	\$400,000
210	2	2	\$550,000



Обзор Scikit-Learn

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y)
```



Процесс машинного обучения с учителем (supervised learning)

- Вспомним также, что для оценки модели мы хотим сравнить результаты модели с тестовыми значениями



Predictions	Area m ²	Bedrooms	Bathrooms	Price
\$410,000	180	1	1	\$400,000
\$540,000	210	2	2	\$550,000



Обзор Scikit-Learn

```
from sklearn.model_family import ModelAlgo
```



Обзор Scikit-Learn

```
from sklearn.model_family import ModelAlgo  
mymodel = ModelAlgo(param1,param2)
```



Обзор Scikit-Learn

```
from sklearn.model_family import ModelAlgo  
mymodel = ModelAlgo(param1,param2)  
mymodel.fit(X_train,y_train)
```




Обзор Scikit-Learn

```
from sklearn.model_family import ModelAlgo  
mymodel = ModelAlgo(param1,param2)  
mymodel.fit(X_train,y_train)  
predictions = mymodel.predict(X_test)
```



Обзор Scikit-Learn

```
from sklearn.model_family import ModelAlgo  
mymodel = ModelAlgo(param1,param2)  
mymodel.fit(X_train,y_train)  
predictions = mymodel.predict(X_test)  
  
from sklearn.metrics import error_metric
```



Обзор Scikit-Learn

```
from sklearn.model_family import ModelAlgo  
mymodel = ModelAlgo(param1,param2)  
mymodel.fit(X_train,y_train)  
predictions = mymodel.predict(X_test)
```

```
from sklearn.metrics import error_metric  
performance = error_metric(y_test,predictions)
```



Обзор Scikit-Learn

- Похожий подход будет применяться для всех алгоритмов машинного обучения с учителем (supervised machine learning).
- Давайте посмотрим на примере линейной регрессии!



Линейная регрессия в Scikit-Learn

Часть 1: настройка данных и
обучение модели



Линейная регрессия в Scikit-Learn

- Ранее мы исследовали вопрос “есть ли связь между общим бюджетом на рекламу и продажами?”
- Теперь мы перейдём к вопросу “какова связь между каждым каналом рекламы (ТВ, радио, газеты) и продажами?”



Оценка работы модели

Метрики регрессии



Оценка работы модели (performance evaluation)

- Теперь у нас есть обученная модель, которая может делать предсказания на основе признаков. Но как мы решим, хороши ли эти предсказания?
- К счастью, у нас есть тестовые данные, с которыми можно сравнить предсказания модели.



Оценка работы модели (performance evaluation)

- Давайте уделим время на то, чтобы обсудить оценку работы регрессионной модели.
- Регрессия – это когда модель предсказывает значения непрерывной переменной (в отличие от дискретной переменной, такая задача называется классификацией).



Оценка работы модели (performance evaluation)

- Например, задача предсказать цену дома на основе его характеристик – это задача регрессии.
- А вот задача предсказать страну, в которой находится дом, на основе его характеристик – это задача классификации.



Оценка работы модели (performance evaluation)

- Возможно Вы слышали про такие метрики оценки, как точность (accuracy) и полнота (recall).
- Но эти метрики не подходят для регрессии – здесь нужны метрики для непрерывных переменных!



Оценка работы модели (performance evaluation)

- Обсудим наиболее частые метрики для регрессии:
 - Mean Absolute Error – средняя абсолютная ошибка
 - Mean Squared Error – среднеквадратическая ошибка
 - Root Mean Square Error – среднеквадратическое отклонение
- Эти метрики подойдут для любой регрессии, не только линейной регрессии



Оценка работы модели (performance evaluation)

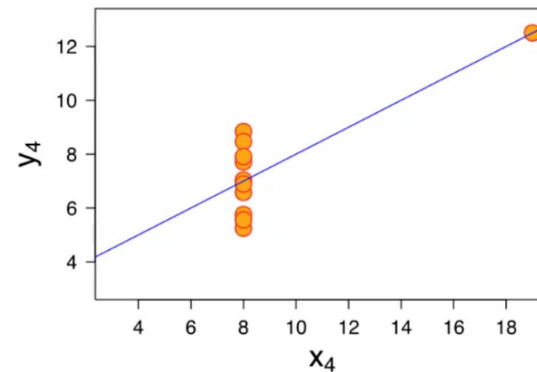
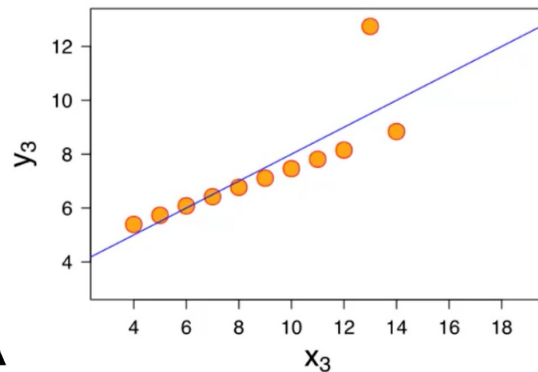
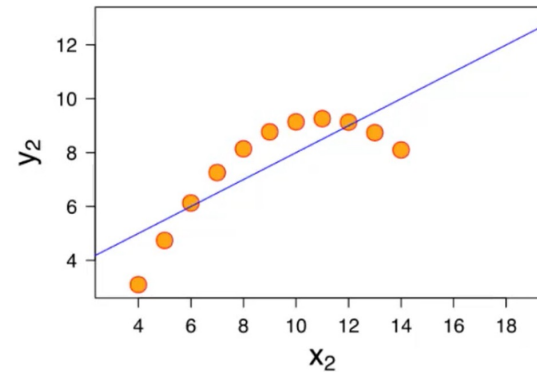
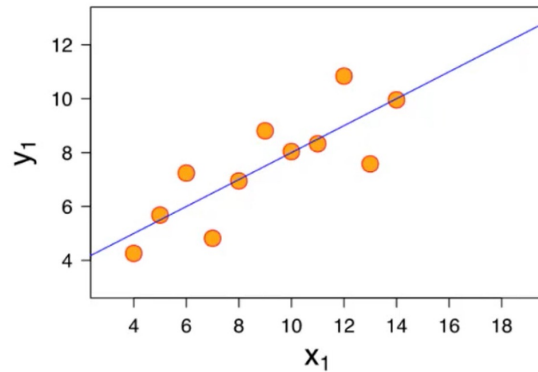
- Mean Absolute Error (MAE) – средняя абсолютная ошибка
 - Берём ошибки, убираем минусы, и усредняем
 - Понятная метрика

$$\frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|$$



Оценка работы модели (performance evaluation)

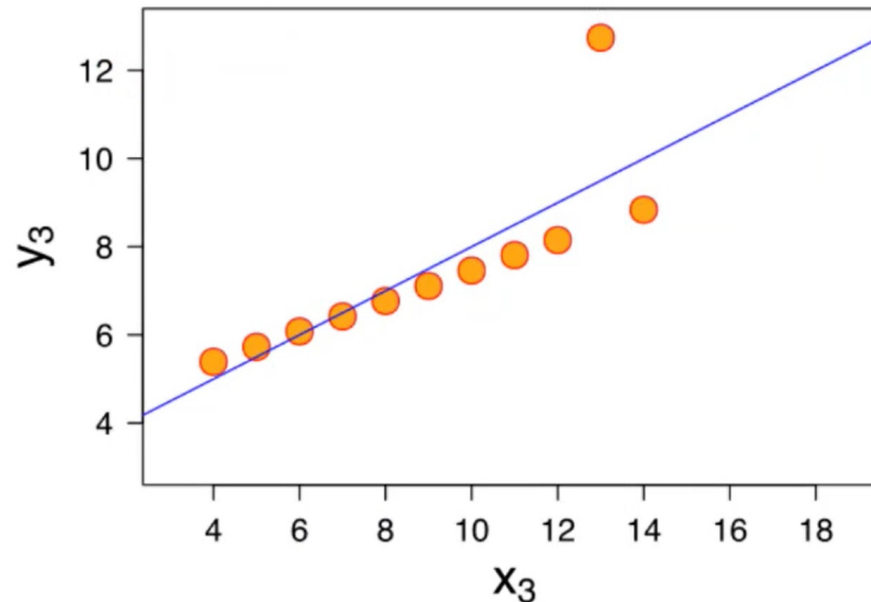
- Однако метрика MAE не “наказывает” большие ошибки





Оценка работы модели (performance evaluation)

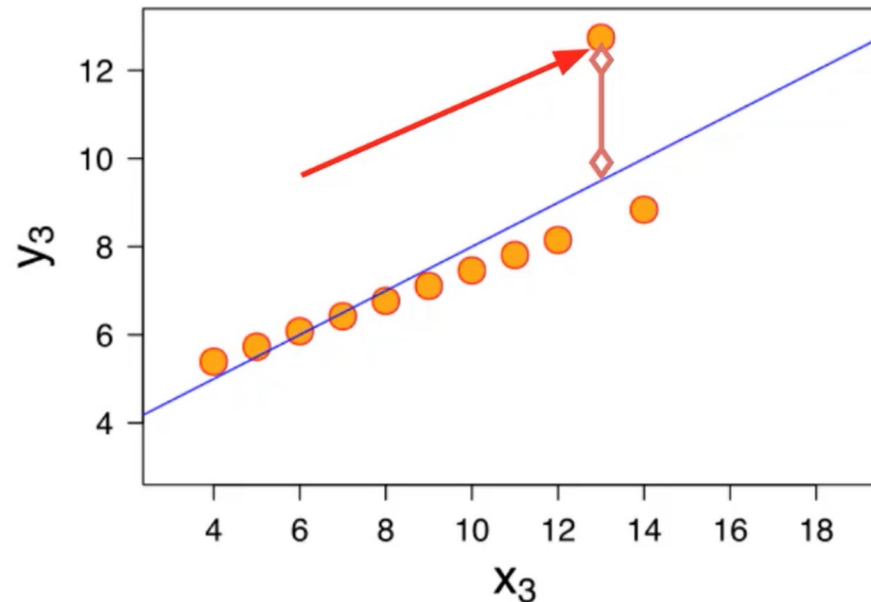
- Однако метрика MAE не “наказывает” большие ошибки





Оценка работы модели (performance evaluation)

- Мы хотим учитывать вот такие ошибки:





Оценка работы модели (performance evaluation)

- Mean Squared Error (MSE) – среднеквадратическая ошибка
 - Большие ошибки “наказываются” сильнее, поэтому метрика MSE популярнее, чем MAE

$$\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$



Оценка работы модели (performance evaluation)

- Mean Squared Error (MSE) – среднеквадратическая ошибка
 - Неудобство – единицы измерения в квадрате

$$\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$



Оценка работы модели (performance evaluation)

- Root Mean Squared Error (RMSE) –
среднеквадратическое отклонение
 - Единицы измерения те же, что и для y

$$\sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2}$$



Оценка работы модели (performance evaluation)

- Частый вопрос – какое значение RMSE считать хорошим?
- Всё зависит от ситуации!
- Отклонение в 10\$ это прекрасный результат для оценки цены загородного дома, но для оценки цены шоколадки это слишком много



Оценка работы модели (performance evaluation)

- Другой пример – если модель будет предсказывать, какую дозу лекарства дать пациенту, то в такой задаче метрика RMSE очень важна для избежания передозировки
- Если модель создаётся для улучшения текущего решения, то в нём уже может быть базовый RMSE, с которым можно сравниваться



Оценка работы модели (performance evaluation)

- Давайте вычислим эти метрики в блокноте с помощью Scikit-Learn!



Анализ остатков (residuals)



Анализ остатков

- Для линейной регрессии имеет смысл смотреть не только на метрику RMSE, но и на остатки (residuals):

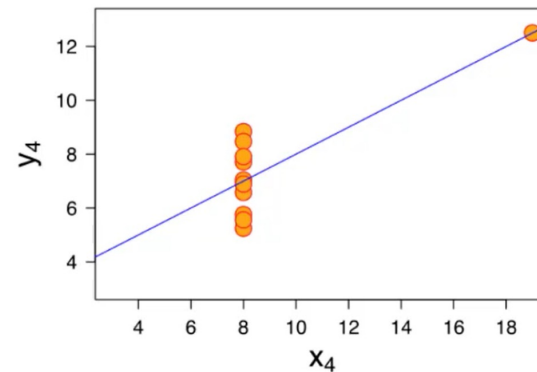
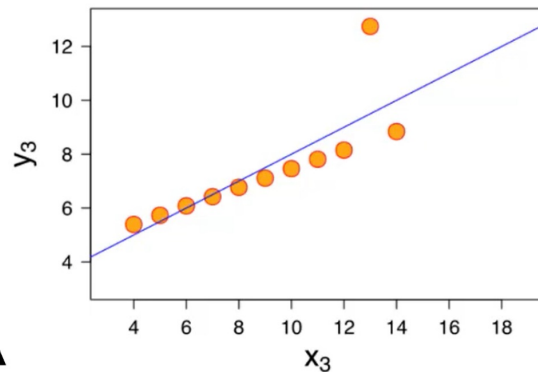
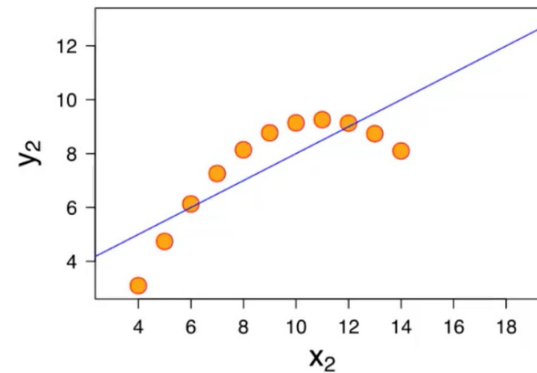
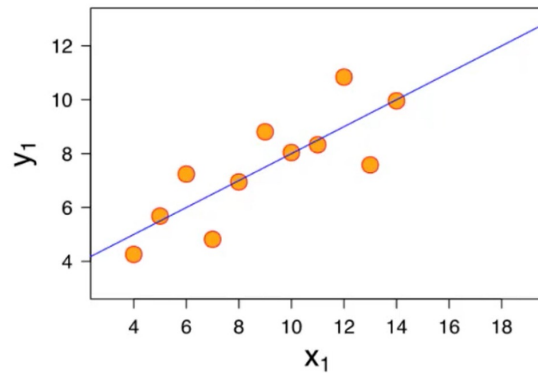
$$(y - \hat{y})$$

- Давайте обсудим, почему это важно...



Анализ остатков

- Вспомните квартет Энскомба:

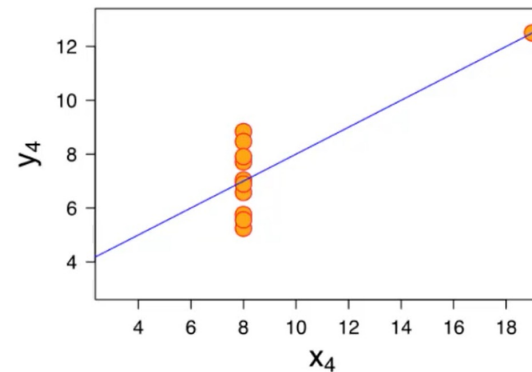
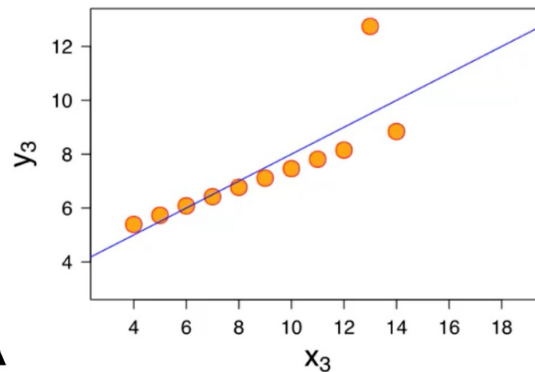
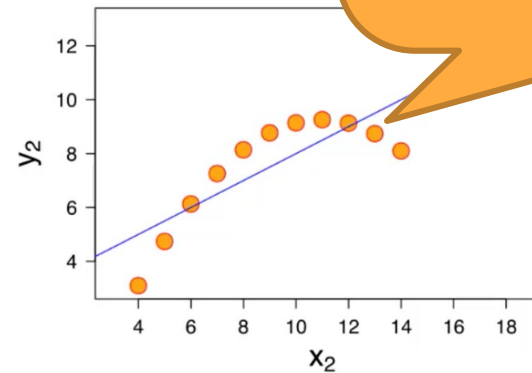
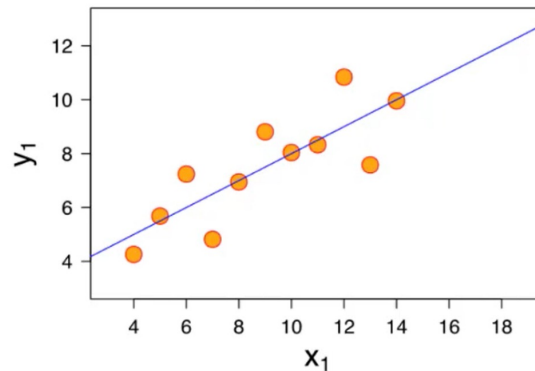




Анализ остатков

- Вспомните квартет Энскомба:

Линейная
регрессия
не работает





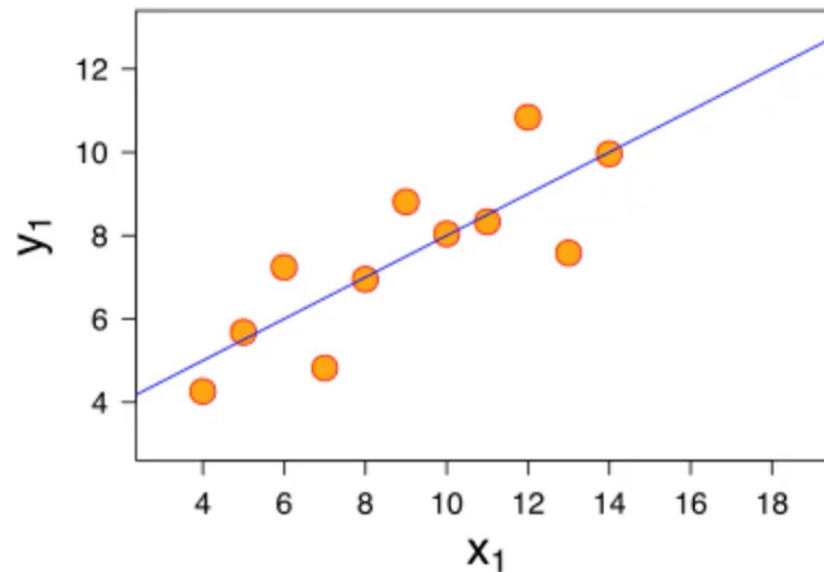
Анализ остатков

- Но как мы можем увидеть это в случае нескольких переменных X ?
- Ведь мы уже не сможем нарисовать наглядные визуальные графики для нескольких переменных одновременно.



Анализ остатков

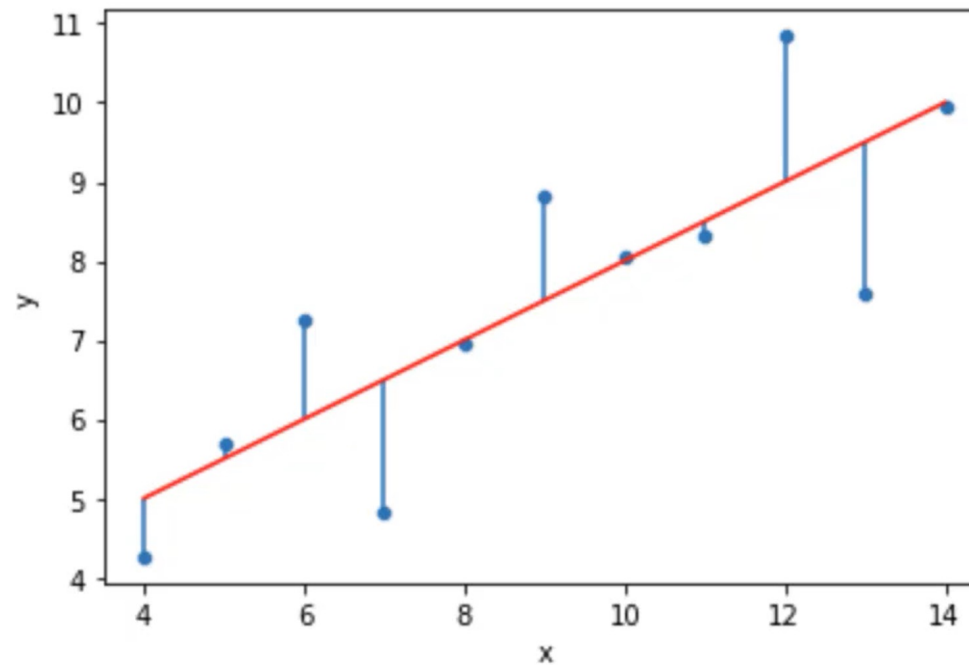
- Что мы можем сделать – это нарисовать график остатков относительно истинных значений y
- Рассмотрим следующий график:





Анализ остатков

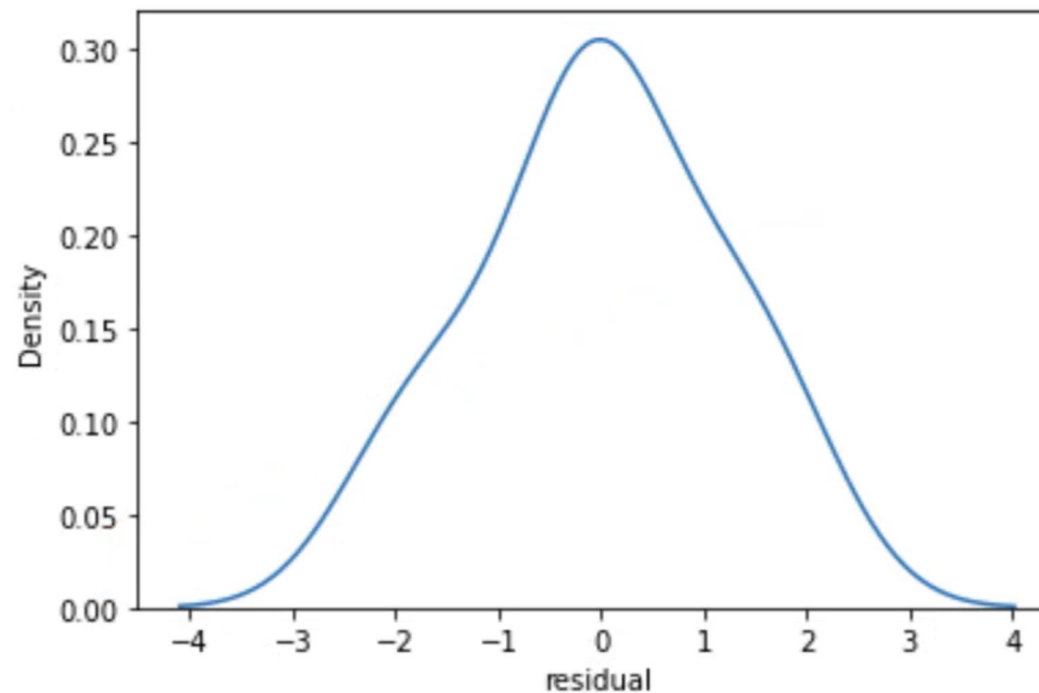
- Остатки должны быть случайны, и их распределение близко к нормальному распределению:





Анализ остатков

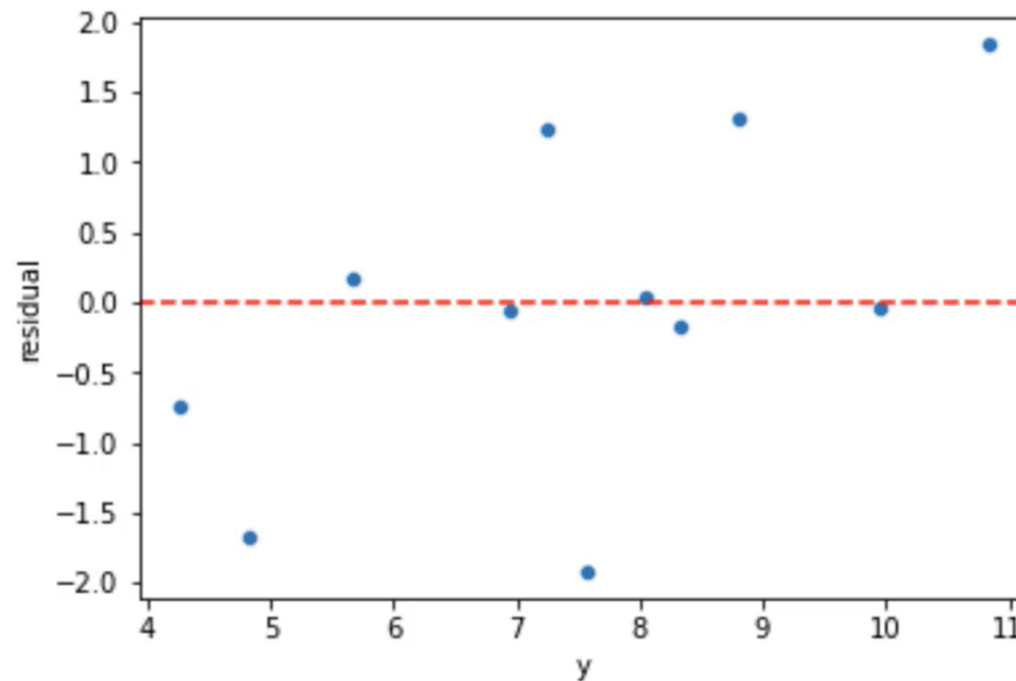
- Остатки должны быть случайны, и их распределение близко к нормальному распределению:





Анализ остатков

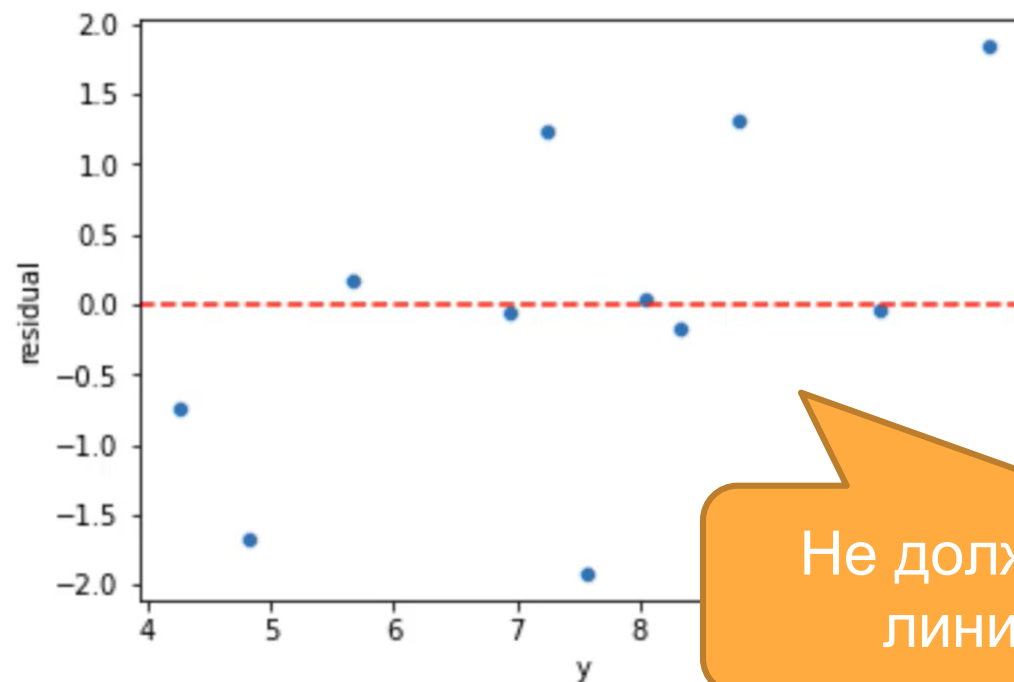
- График остатков (residual plot) показывает остатки и истинные значения y :





Анализ остатков

- График остатков (residual plot) показывает остатки и истинные значения y :

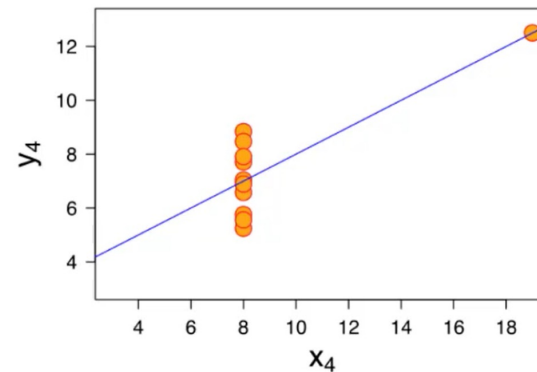
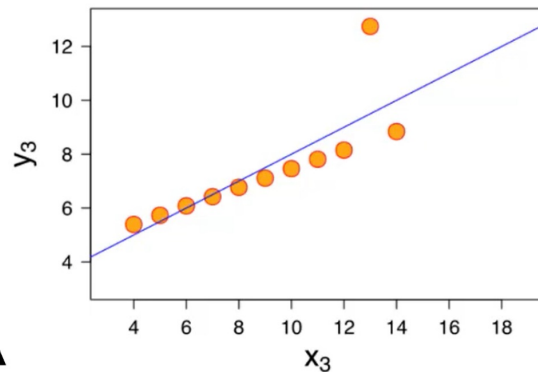
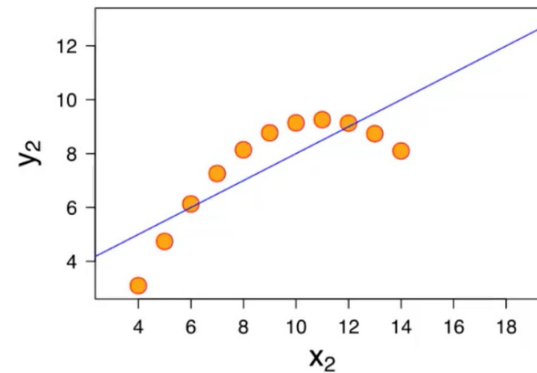
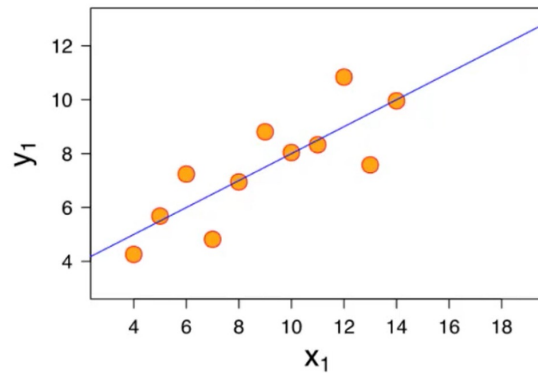


Не должно быть чёткой
линии или кривой



Анализ остатков

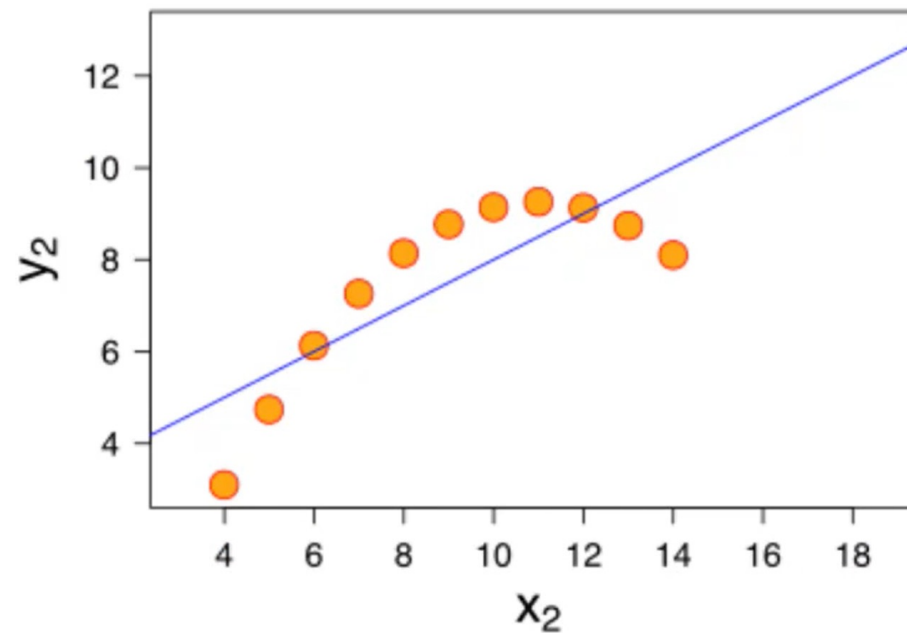
- Как насчёт данных, где линию не провести?





Анализ остатков

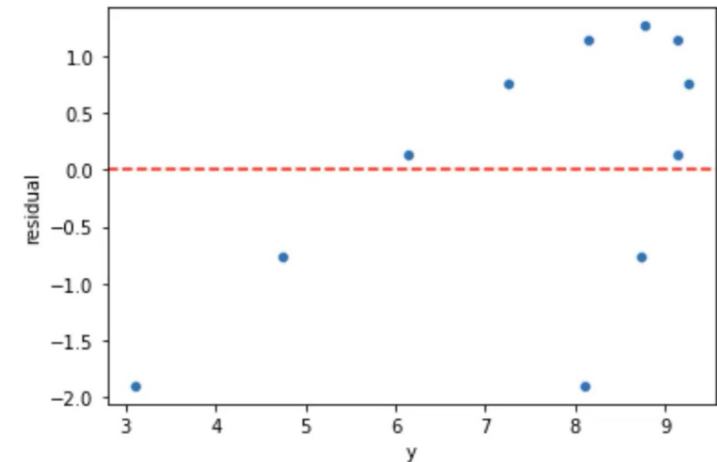
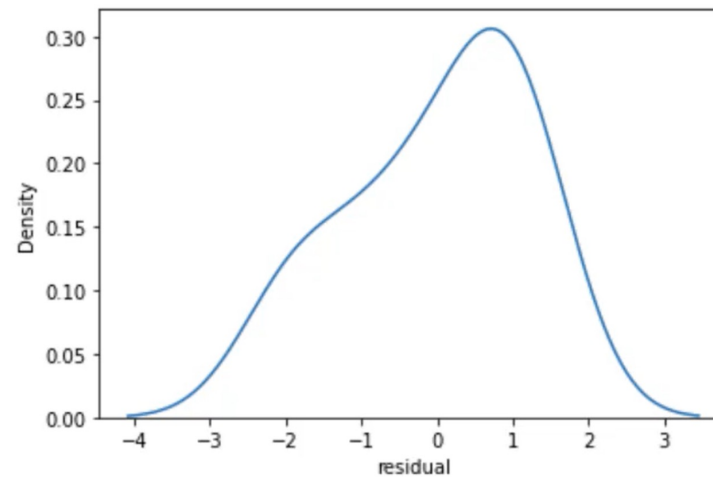
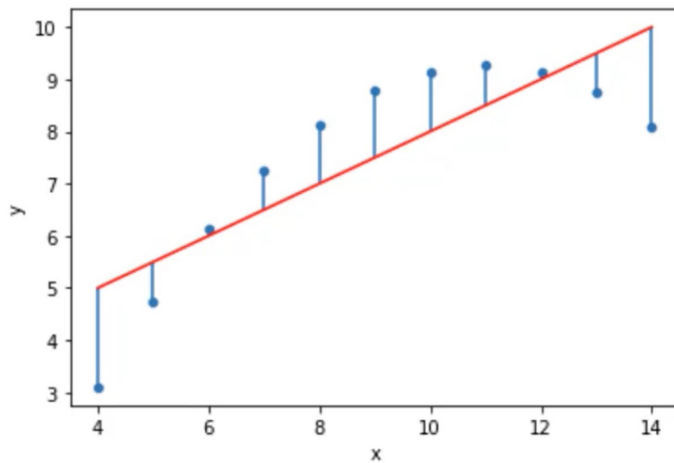
- Как насчёт данных, где линию не провести?





Анализ остатков

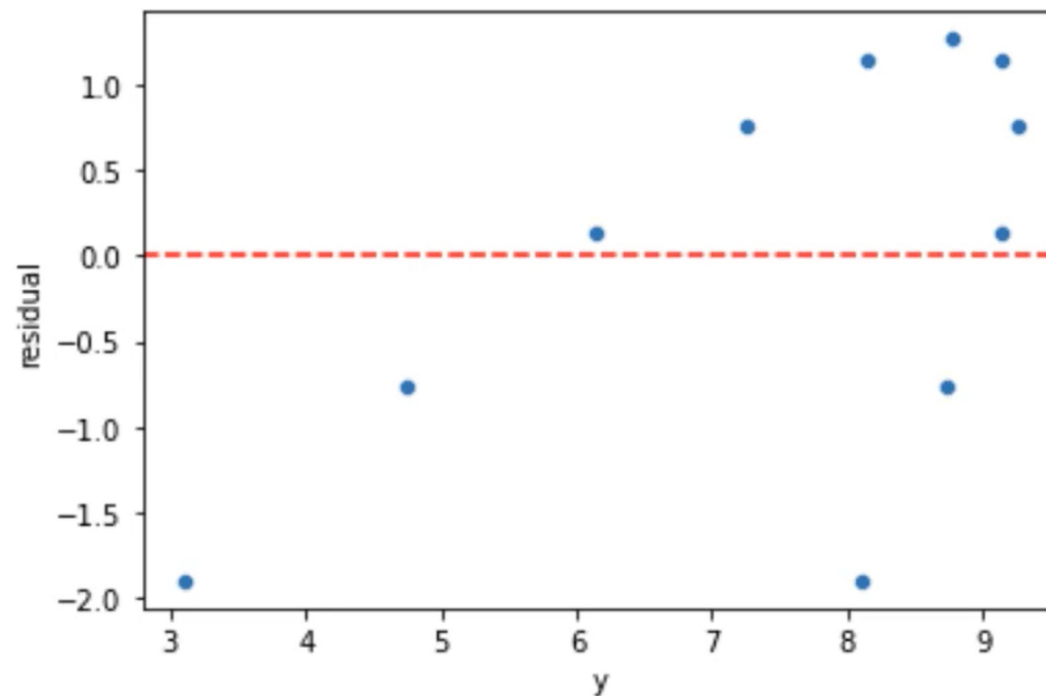
- Здесь мы видим некоторую неслучайную картину – это значит, что линейная регрессия неприменима!





Анализ остатков

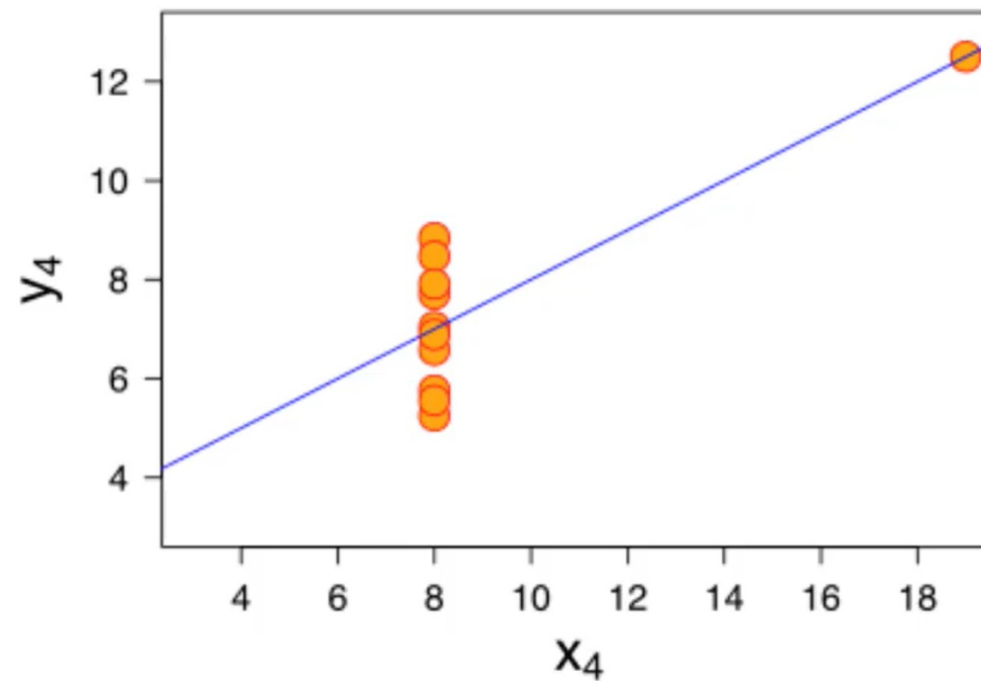
- Здесь мы видим некоторую неслучайную картину – это значит, что линейная регрессия неприменима!





Анализ остатков

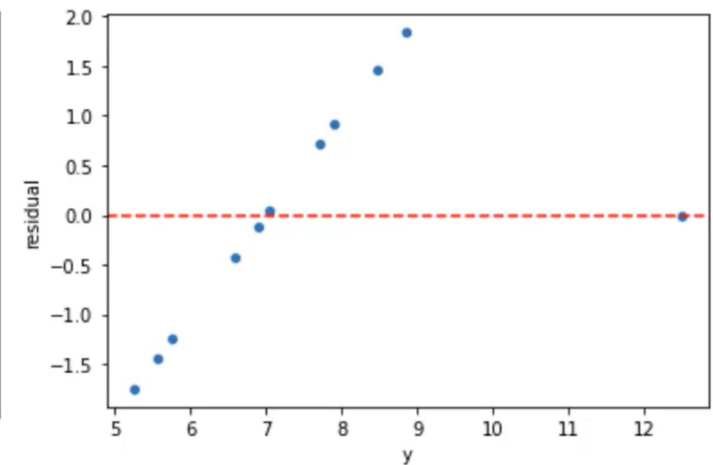
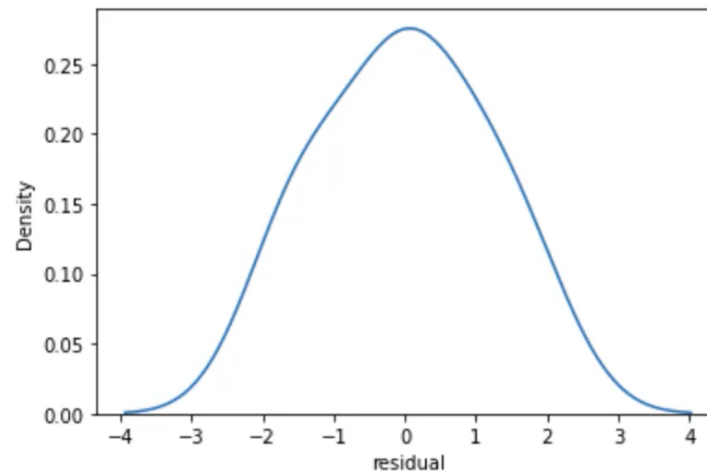
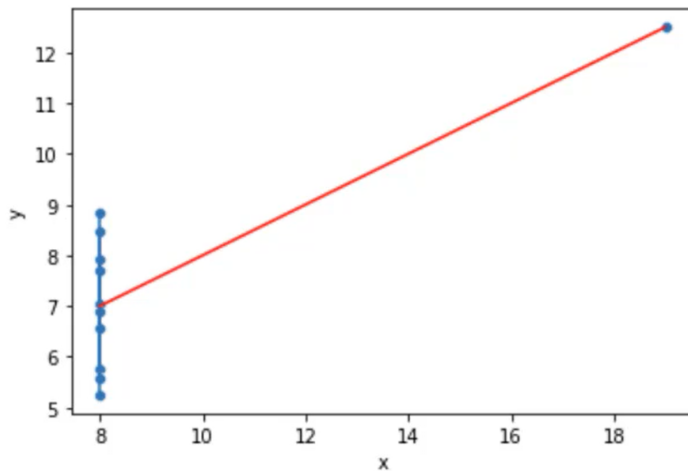
- Как насчёт данных, где линию не провести?





Анализ остатков

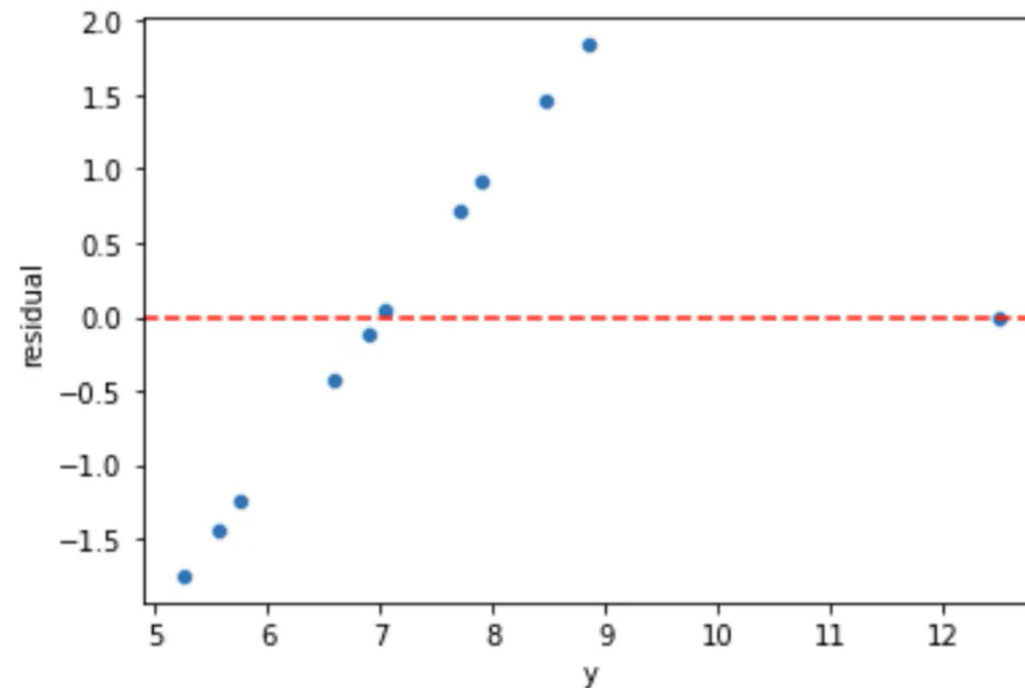
- Здесь мы видим некоторую неслучайную картину – это значит, что линейная регрессия неприменима!





Анализ остатков

- Здесь мы видим некоторую неслучайную картину – это значит, что линейная регрессия неприменима!





Анализ остатков

- Давайте посмотрим, как нарисовать такие графики в Python для наших моделей!



Внедрение модели

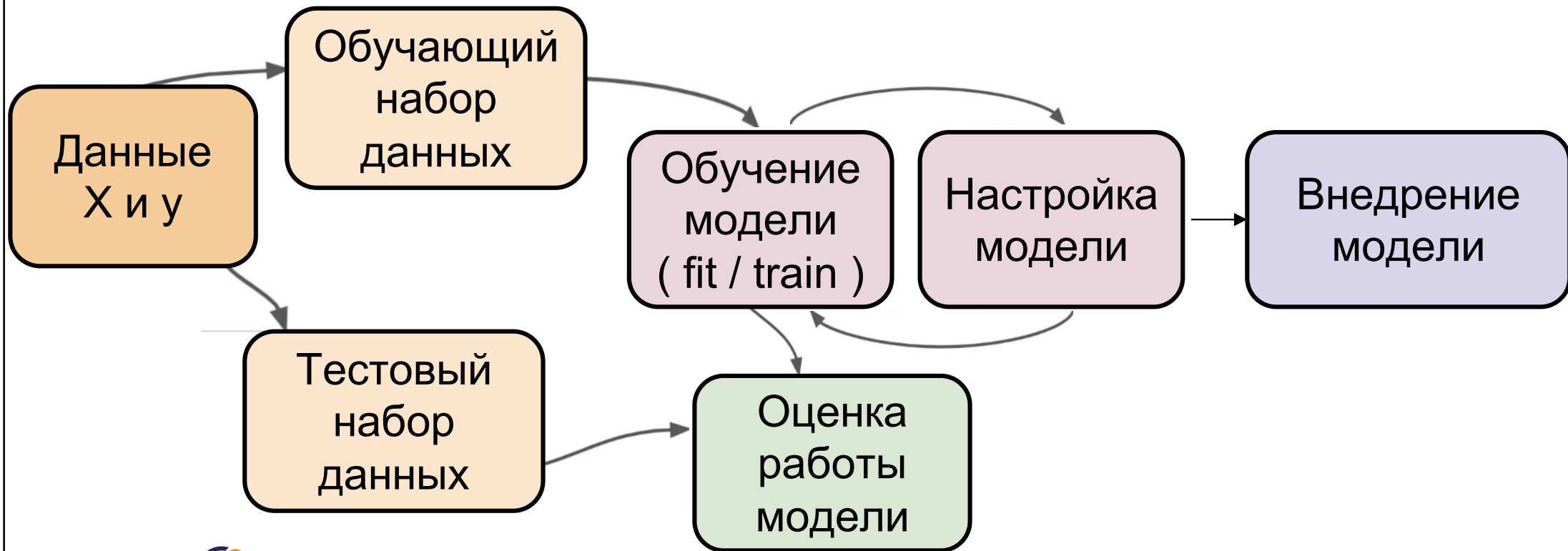


Внедрение / развёртывание модели (deployment)

- Мы почти завершили наш первый проход по шагам процесса машинного обучения!
- Давайте кратко вспомним, какие это шаги, и что мы уже сделали.

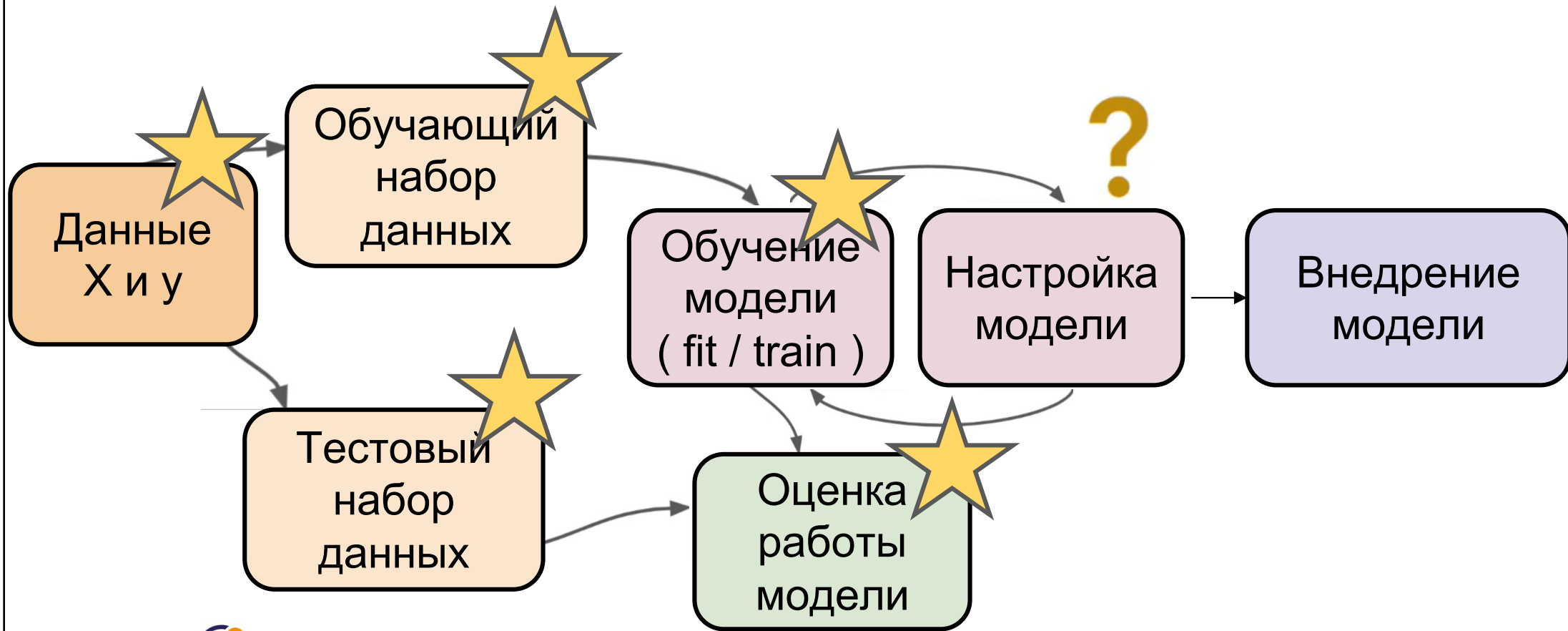


Процесс машинного обучения с учителем (supervised learning)





Процесс машинного обучения с учителем (supervised learning)





Внедрение / развёртывание модели (deployment)

- Позже для настройки модели мы будем применять полиномиальную регрессию и регуляризацию.
- Сейчас давайте пока рассмотрим простое “внедрение” модели – как мы можем сохранить и затем загрузить модель, и применить её к новым данным.



Полиномиальная регрессия

Теория и мотивация



Полиномиальная регрессия

- Мы завершили изучение линейной регрессии - для задачи предсказания целевой переменной на основе набора признаков (features).
- Можем ли мы как-то улучшить линейную регрессию?
- Один из вариантов – применить для признаков полиномы более высокой степени.



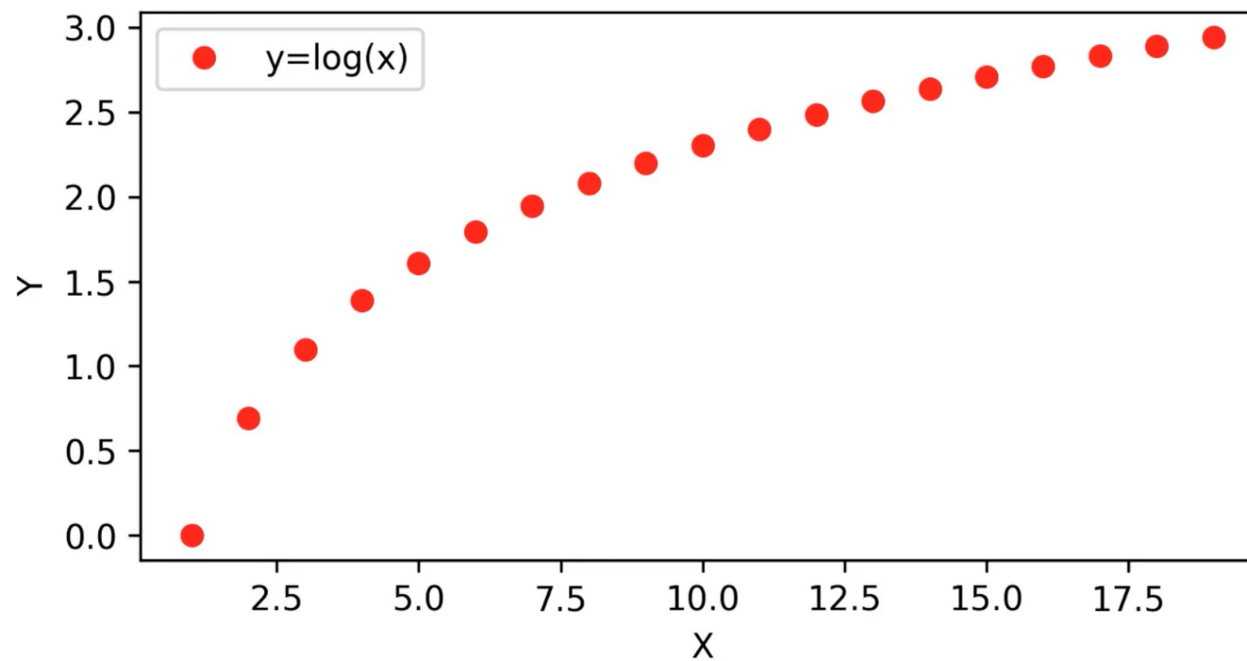
Полиномиальная регрессия

- Полиномы помогут решить две задачи:
 - Нелинейные связи между отдельным признаком и целевой переменной
 - Взаимодействие (зависимости) между признаками
- Начнём с первой задачи – когда связь не является линейной.



Полиномиальная регрессия

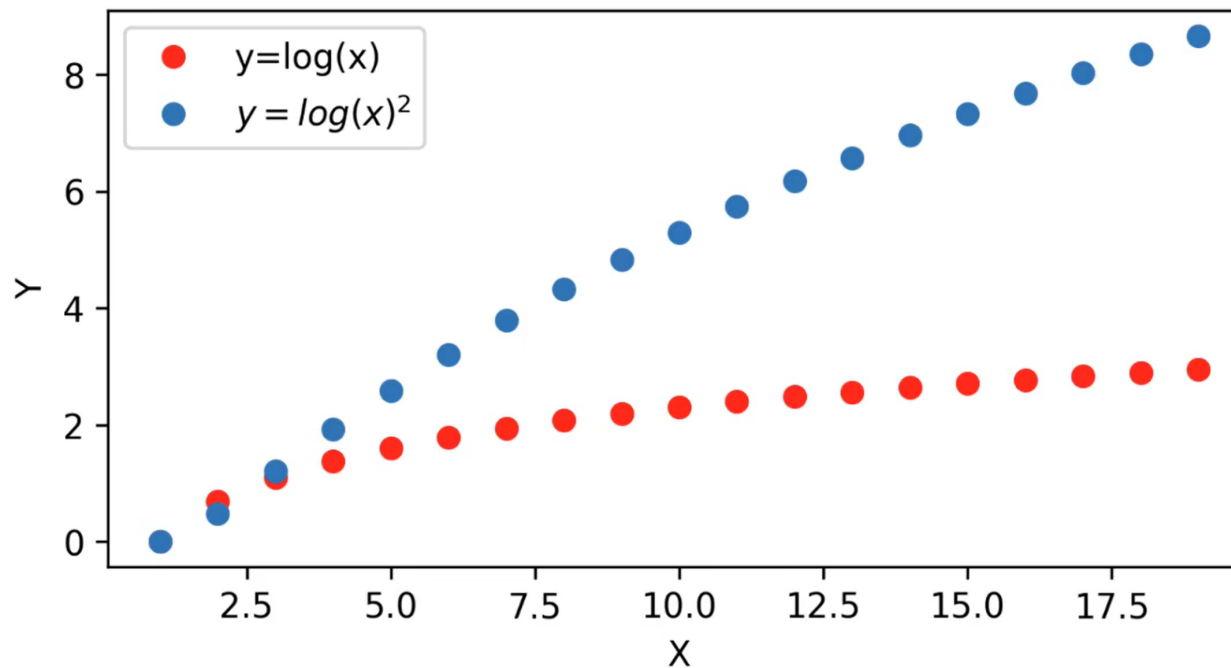
- Пример признака, который не является линейным:





Полиномиальная регрессия

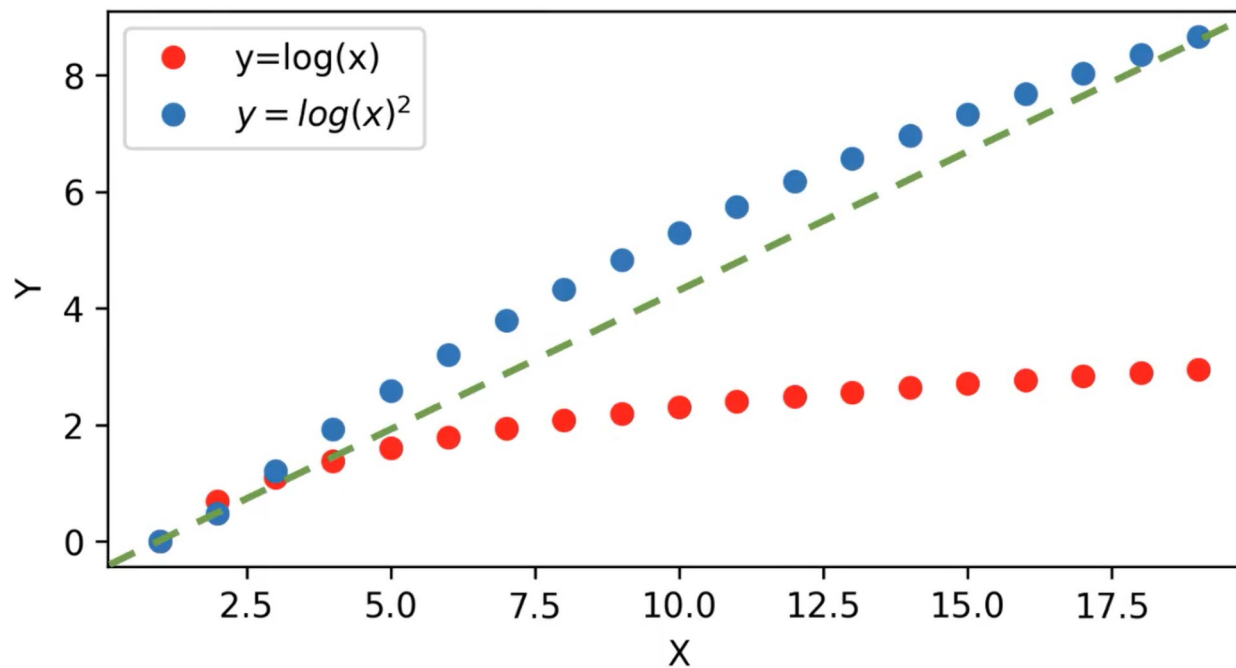
- Другой пример – квадрат логарифма:





Полиномиальная регрессия

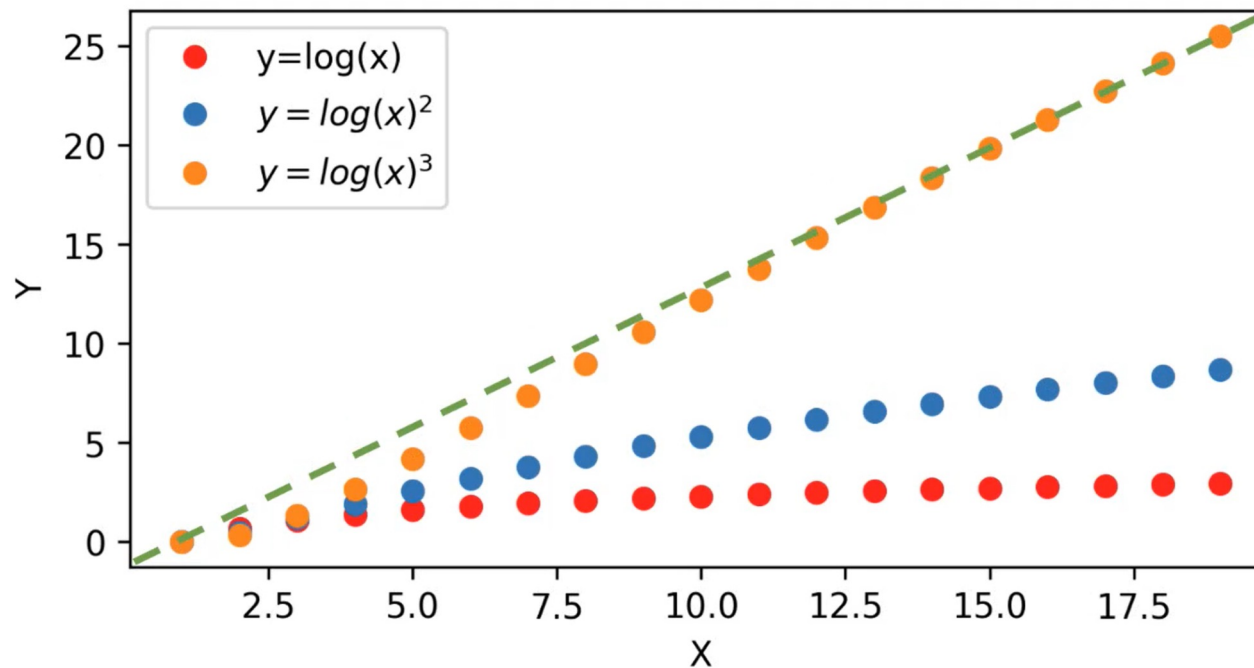
- Другой пример – квадрат логарифма:





Полиномиальная регрессия

- Более высокие степени ближе в прямой линии:





Полиномиальная регрессия

- Конечно это несколько надуманный пример, и далеко не каждый признак будет похож на полином более высокого порядка.
- Основная мысль здесь в том, что при поиске коэффициентов бетта в линейном уравнении можно брать не только сами признаки, но и степени этих признаков



Полиномиальная регрессия

- Также между признаками возможны зависимости. Что если один признак важен, только когда важен и другой?
- Например, возможно реклама в газете сама по себе мало эффективна, но если добавить рекламу на ТВ, то эффективность газетной рекламы заметно возрастёт?



Полиномиальная регрессия

- Кто посмотрел только ТВ – возможно купят товар. Но будут и те, кто сначала увидел рекламу по ТВ, а потом реклама в газете “напомнила” ему/ей о товаре. В итоге результат может быть выше, чем отдельно ТВ и отдельно газеты.
- Можем ли мы как-то это проверить?



Полиномиальная регрессия

- Простейший способ – добавить произведение двух признаков (interaction term), в дополнение к каждому из признаков в отдельности
- Это можно сделать в Scikit-Learn с помощью методов preprocessing.



Полиномиальная регрессия

- В библиотеке Scikit-Learn preprocessing есть много полезных методов для обработки данных перед обучением модели
- Один из них – PolynomialFeatures – автоматически создаёт полиномы более высоких порядков, а также произведения комбинаций признаков.



Полиномиальная регрессия

- Примеры слагаемых, которые создаются:
 - Постоянное число (значение 1.0)
 - Возведение в степени (x^1 , x^2 , x^3 и т.д.)
 - Произведения пар признаков ($x_1 * x_2$, $x_1 * x_3$ и т.д.)



Полиномиальная регрессия

- В случае двух признаков A и B:

$1, A, B, A^2, AB, B^2$



Полиномиальная регрессия

- В случае двух признаков A и B:

$$1, A, B, A^2, AB, B^2$$

- Общая форма для двух признаков X_1 и X_2 :

$$1, X_1, X_2, X_1^2, X_1X_2, X_2^2$$



Полиномиальная регрессия

- В случае двух признаков A и B:

$$1, A, B, A^2, AB, B^2$$

- Общая форма для двух признаков X_1 и X_2 :

$$1, X_1, X_2, X_1^2, X_1X_2, X_2^2$$

- Пример – если $X_1=2$ и $X_2=3$:

$$1, 2, 3, 4, 6, 9$$



Полиномиальная регрессия

Создание признаков



Полиномиальная регрессия

Обучение и оценка модели



Дилемма смещения-дисперсии (Bias-Variance Trade-off)

недообучение v.s. переобучение



Недообучение и переобучение

- Мы видели, что полиномы более высоких порядков показывают результаты лучше, чем обычная линейная регрессия
- Но как выбрать оптимальную степень полинома?
- Какие плюсы и минусы нужно учитывать, повышая сложность модели?



Недообучение и переобучение

- В общем случае усложнение модели приводит к дилемме смещения-дисперсии (bias-variance trade off).
- Мы хотим получить модель, которая хорошо работает на новых неизвестных данных, а также которая хорошо объясняет известные данные.



Недообучение и переобучение

- Большое смещение (bias) – недообучение (underfitting)
- Большая дисперсия (variance) – переобучение (overfitting), здесь “пере-” означает “слишком”.
- Оба случая – плохая модель
- Рассмотрим сначала случай переобученной модели...



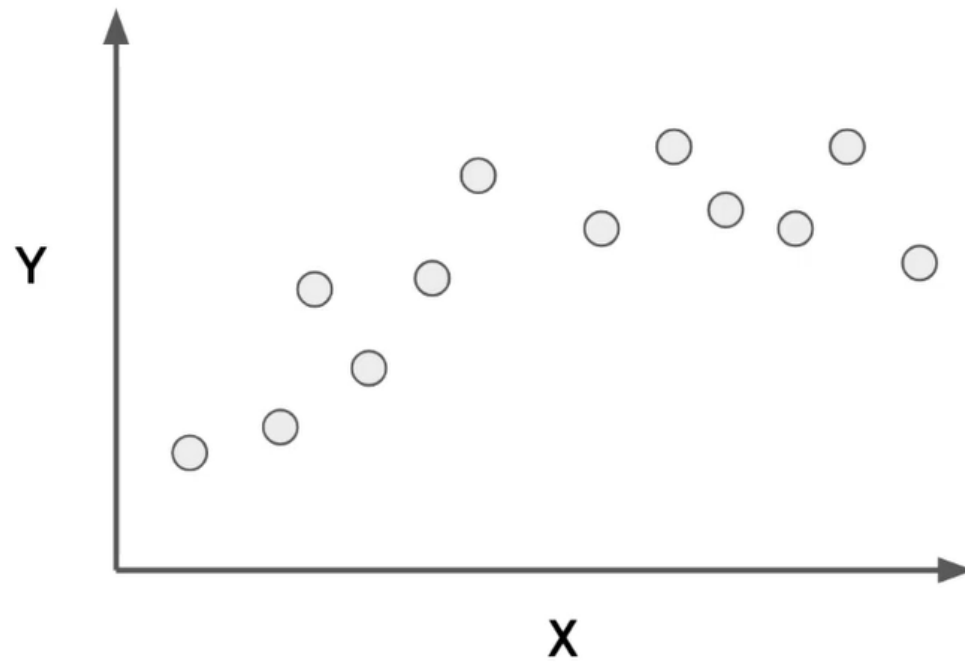
Недообучение и переобучение

- Переобучение (overfitting):
 - Модель слишком точно повторяет шумы и неточности в данных
 - Это часто приводит к малым ошибкам на обучающем наборе данных, но к большим ошибкам на тестовых / проверочных данных



Недообучение и переобучение

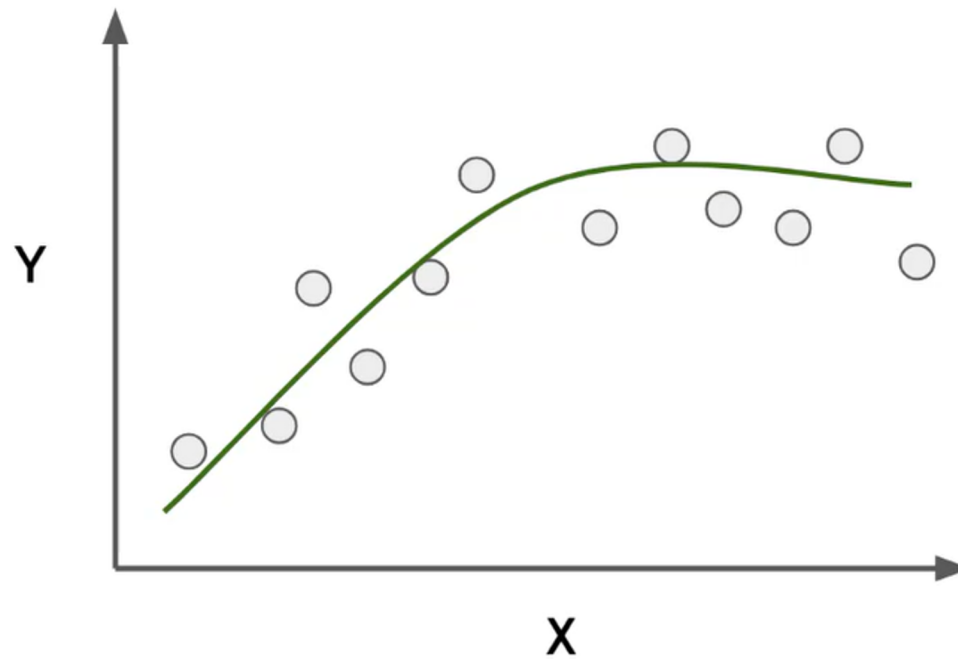
- Данные





Недообучение и переобучение

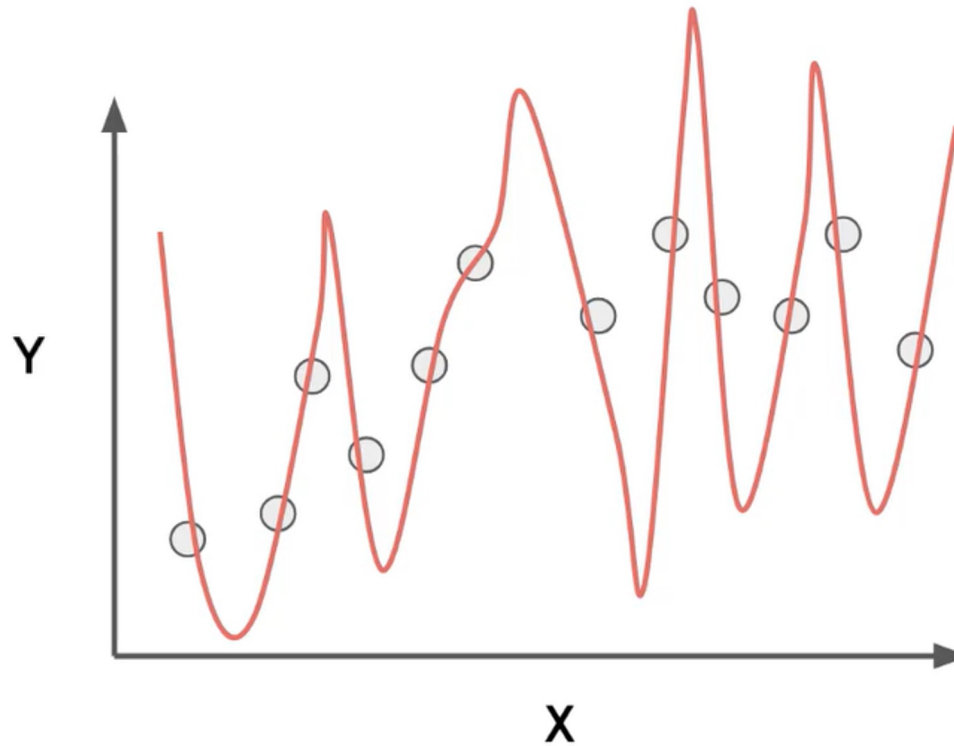
- Хорошая модель





Недообучение и переобучение

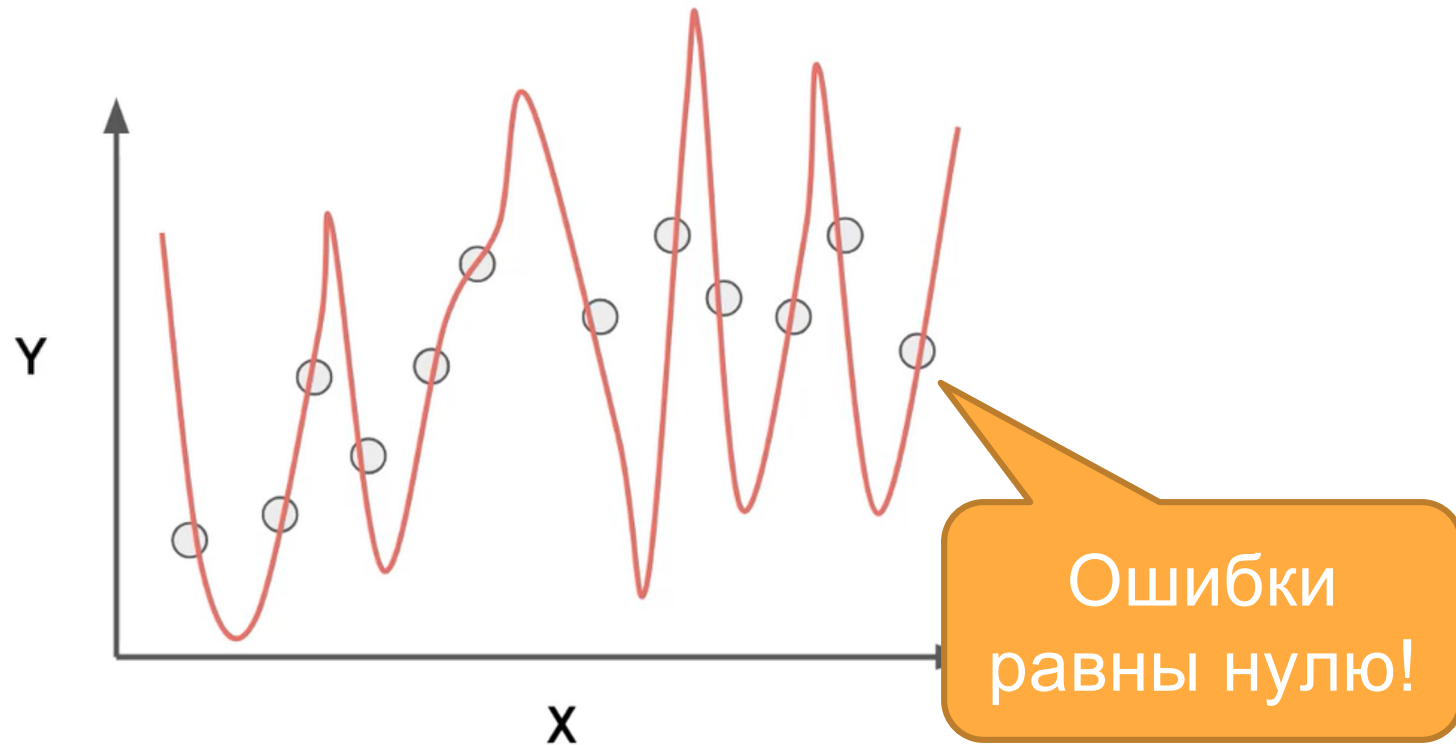
- Переобученная модель (overfitting)





Недообучение и переобучение

- Переобученная модель (overfitting)





Недообучение и переобучение

- Переобучение (overfitting):
 - Модель слишком точно повторяет шумы и неточности в данных
 - Это часто приводит к малым ошибкам на обучающем наборе данных, но к большим ошибкам на тестовых / проверочных данных



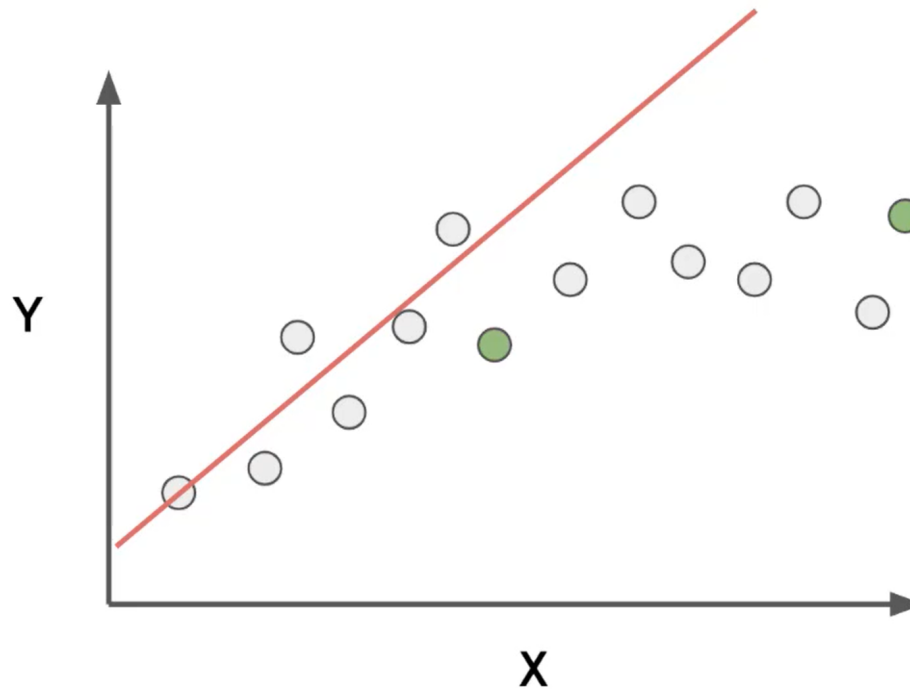
Недообучение и переобучение

- Недобучение (underfitting):
 - Модель не описывает тренды в данных и недостаточно хорошо работает на обучающем наборе
 - Малая дисперсия (variance) и большое смещение (bias)
 - Как правило это говорит о том, что модель слишком простая



Недообучение и переобучение

- Недообученная модель (underfitting)





Недообучение и переобучение

- Недобучение (underfitting):
 - Модель имеет большое смещение (bias) и слишком сильно обобщает данные
 - Обычно показывает слабые результаты и на обучающем, и на тестовом наборах данных



Недообучение и переобучение

- Недобучение v.s. переобучение
 - Обычно переобучение сложнее заметить, поскольку модель показывает хорошие результаты на обучающем наборе данных и выглядит всё так, что она работает хорошо



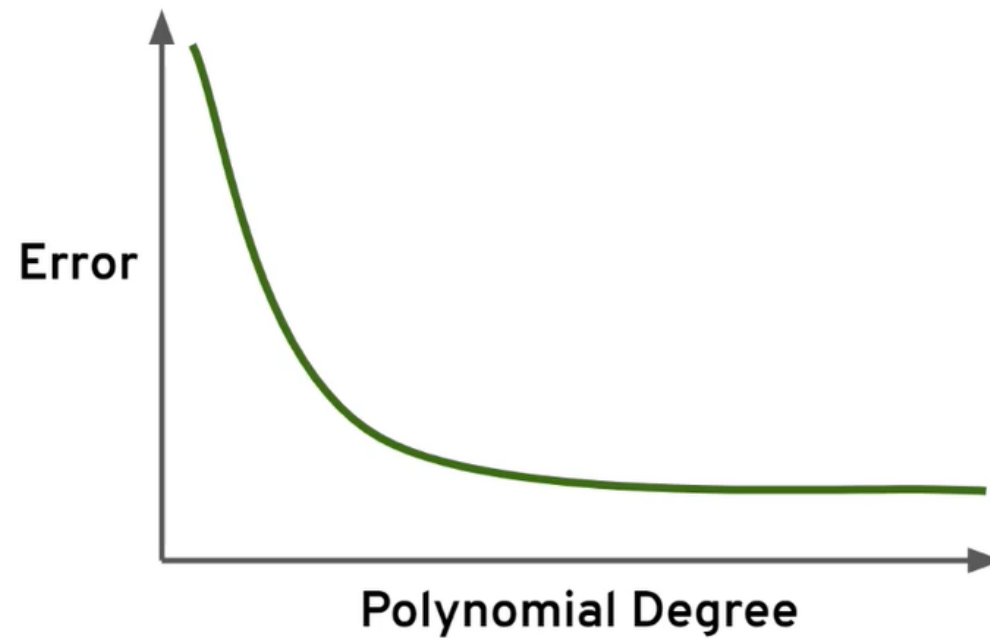
Недообучение и переобучение

- В наших примерах для одной переменной можно рисовать визуальные графики. Но как мы увидим недообучение и переобучение в случае многих переменных?
- Для начала представим, что мы обучили модель и затем измерили ошибки для разной сложности модели (степеней полинома).



Недообучение и переобучение

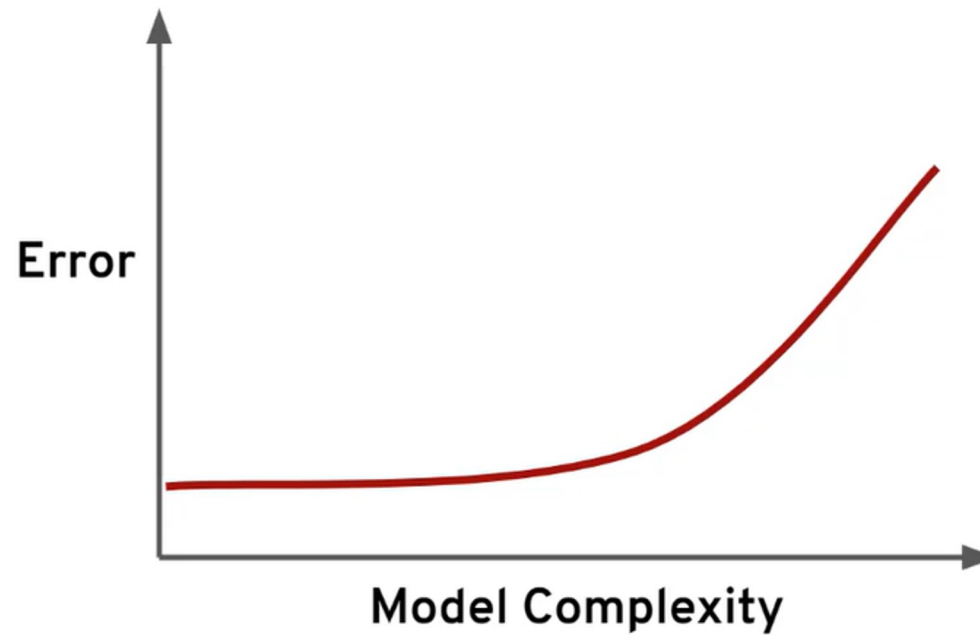
- Хорошая модель





Недообучение и переобучение

- Плохая модель





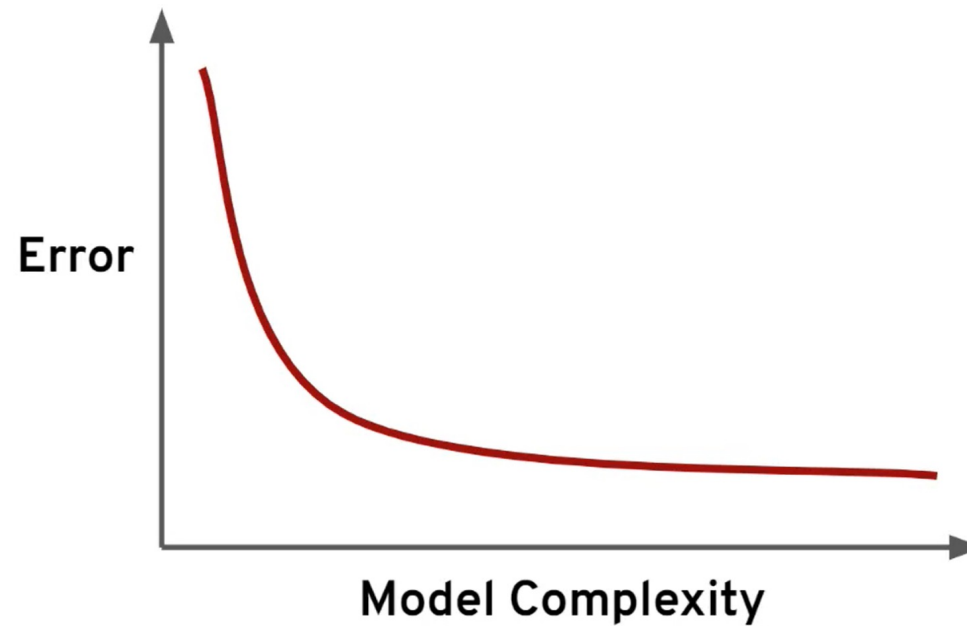
Недообучение и переобучение

- При изучении недообученности и переобученности модели нужно смотреть результаты модели на **обучающем наборе данных (training set)**, сравнивая их с результатами на **тестовом наборе данных (test set)**.



Недообучение и переобучение

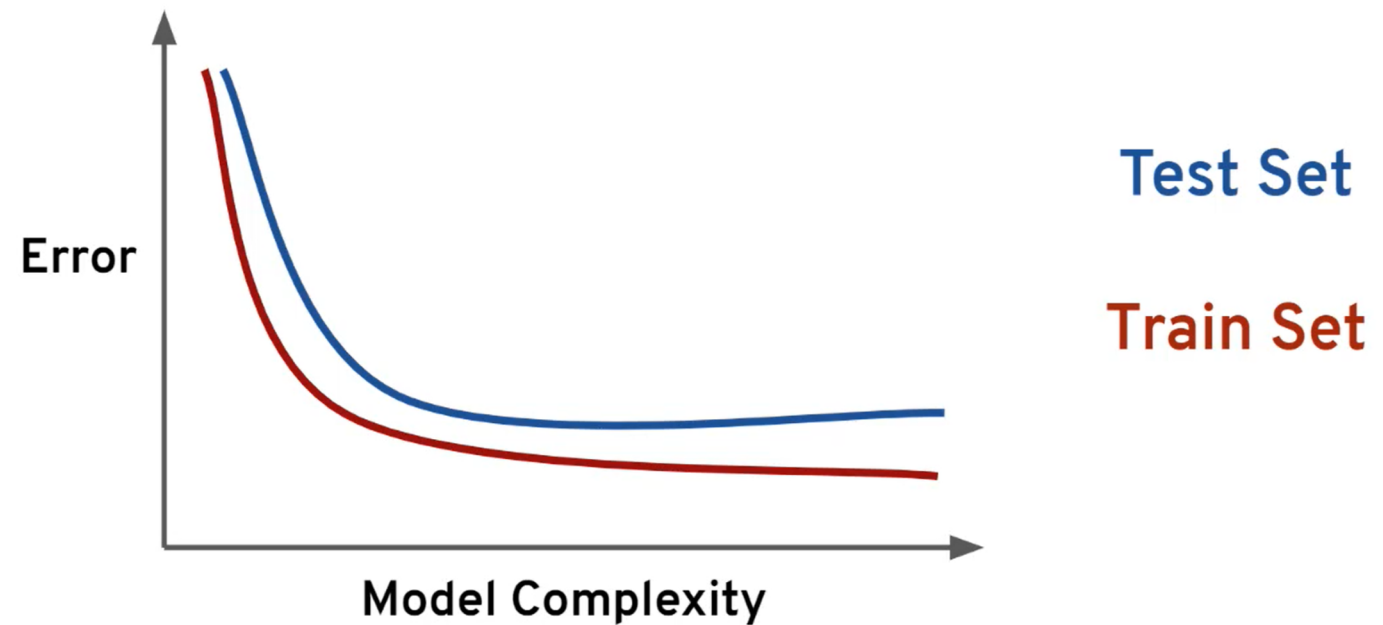
- Сначала на **обучающем наборе данных (training set)**





Недообучение и переобучение

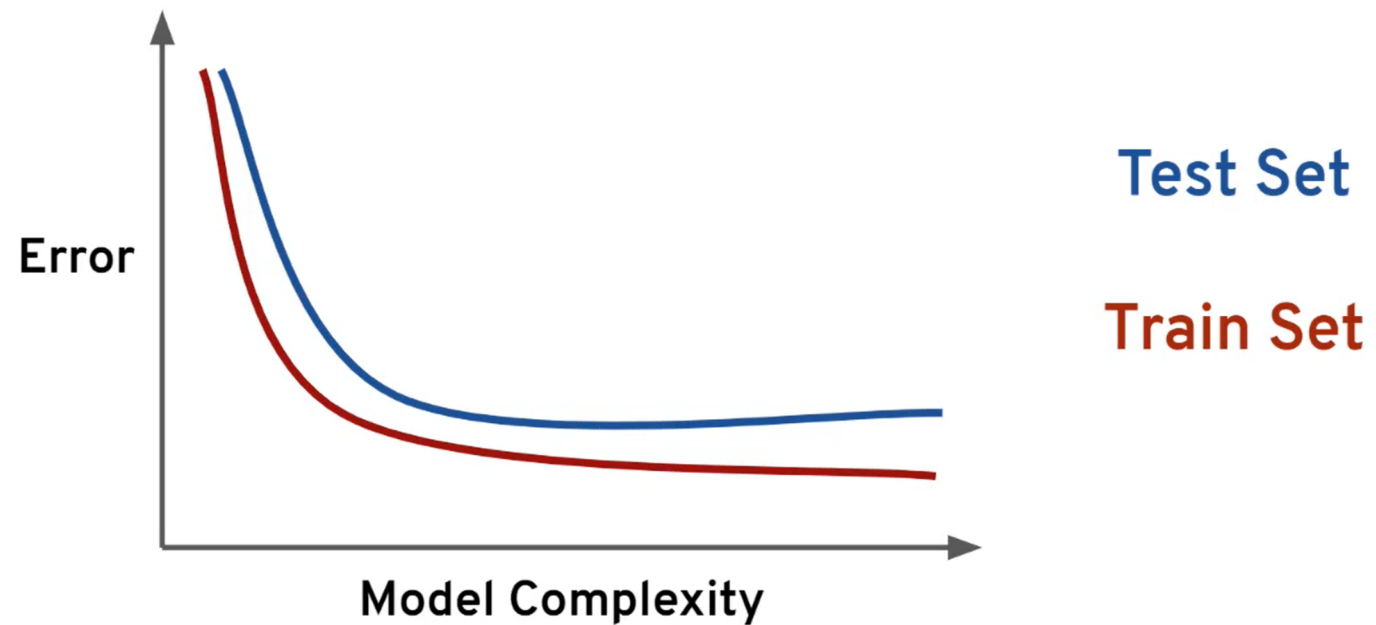
- Затем на **тестовом наборе данных (test set)**





Недообучение и переобучение

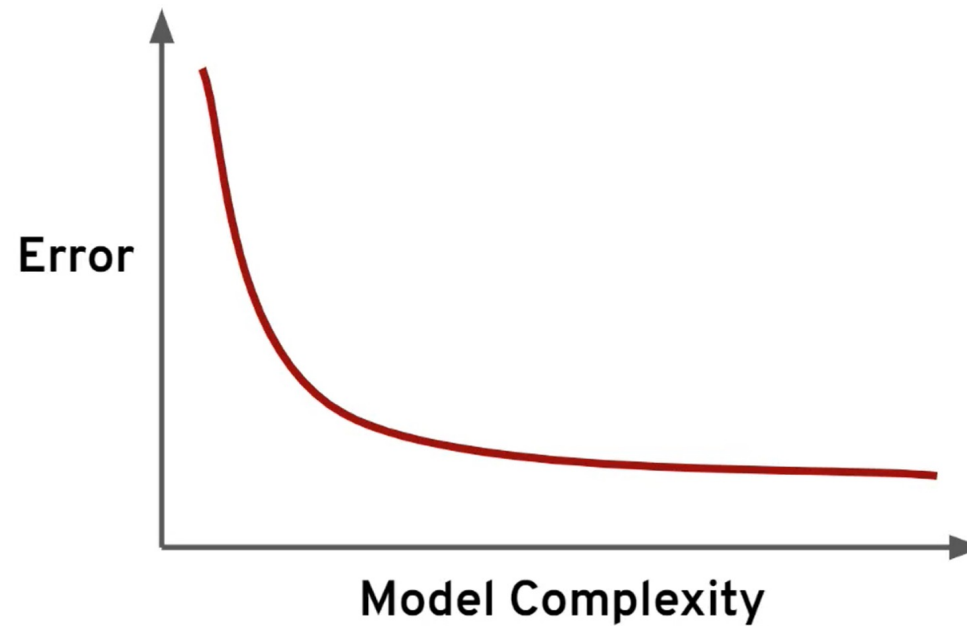
- В идеале модель должна хорошо работать в обоих случаях, показывая похожее поведение





Недообучение и переобучение

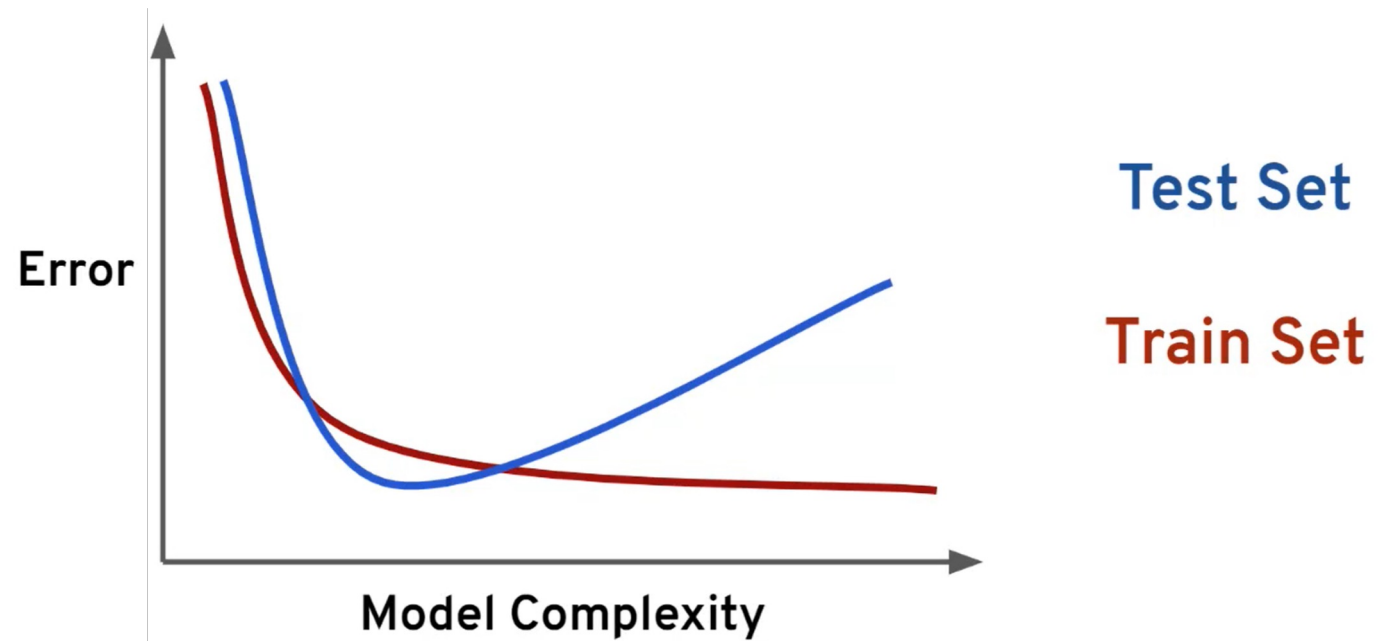
- Но что если мы переобучим модель (overfitting)?
Модель будет плохо работать на тестовых данных!





Недообучение и переобучение

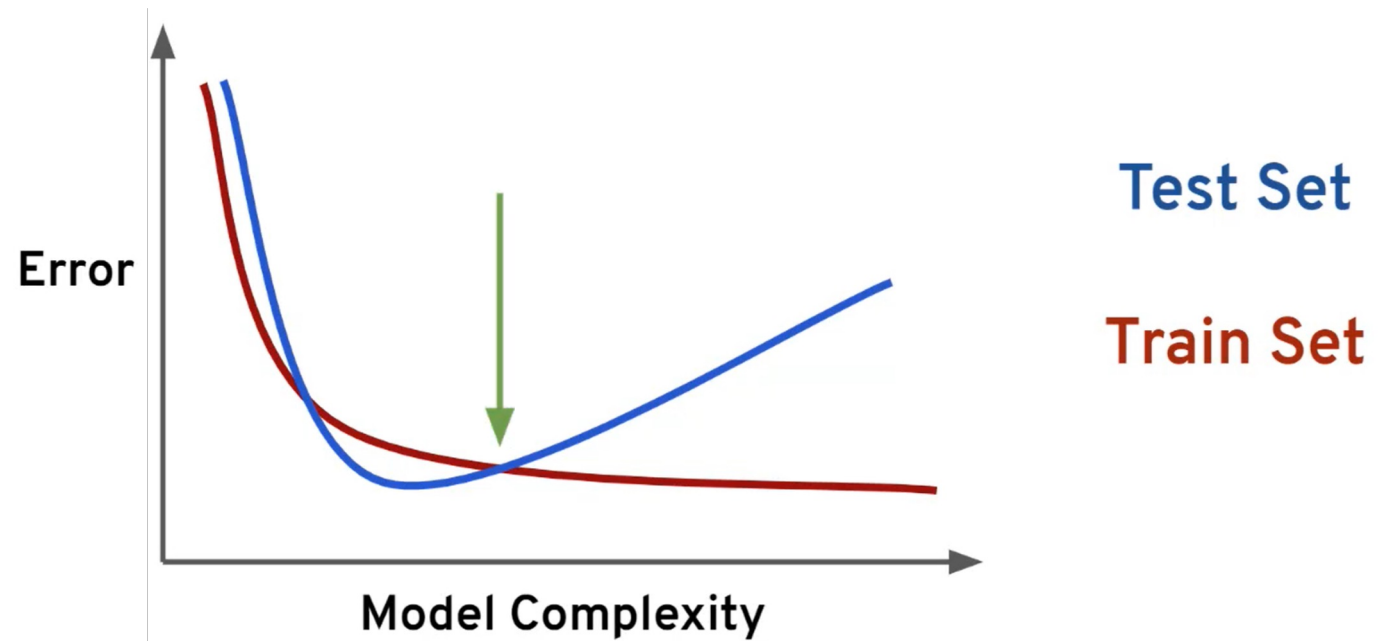
- Но что если мы переобучим модель (overfitting)?
Модель будет плохо работать на тестовых данных!





Недообучение и переобучение

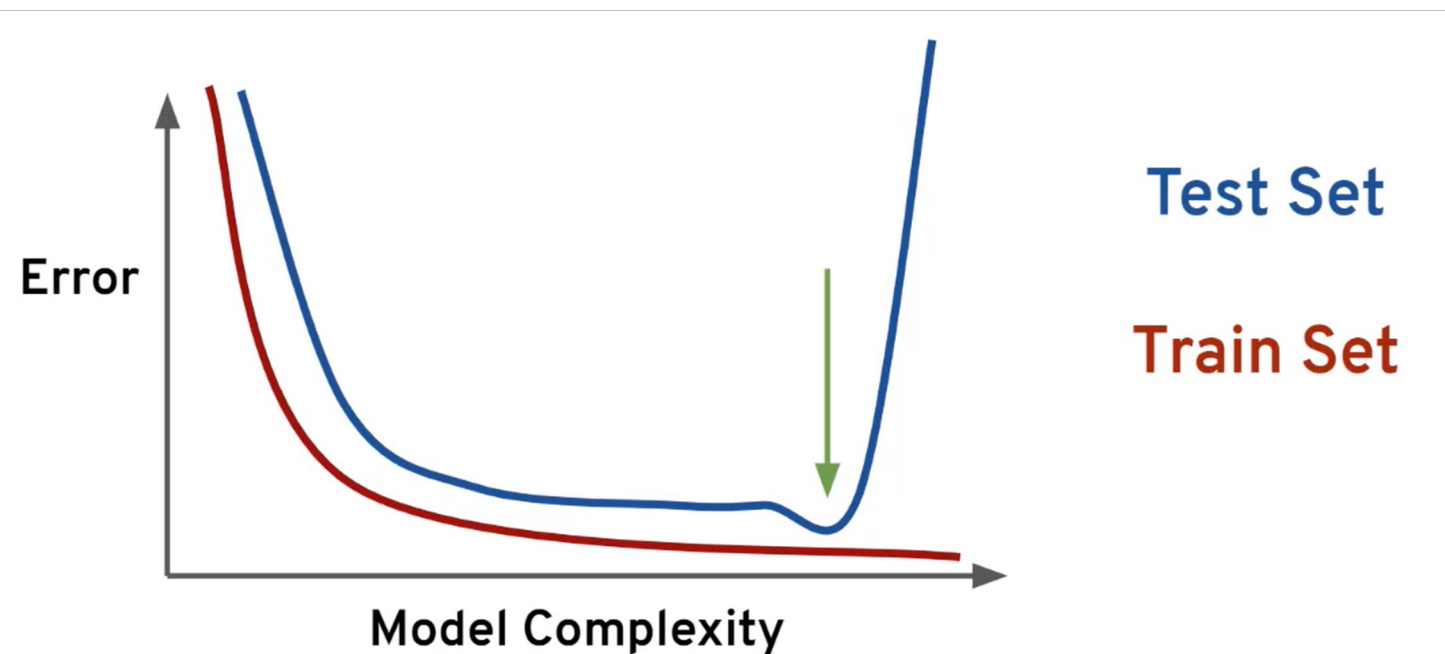
- Это индикатор того, что модель слишком сложная. Нужно искать точку, где остановиться.





Недообучение и переобучение

- В некоторых алгоритмах ошибка может начать расти очень резко.





Недообучение и переобучение

- Это значит, что когда мы хотим выбрать сложность модели, а также хотим оценить работу модели, то нам нужно изучать ошибки модели как на обучающем, так и на тестовом наборах данных.



Недообучение и переобучение

- Это значит, что когда мы хотим выбрать сложность модели, а также хотим оценить работу модели, то нам нужно изучать ошибки модели как на обучающем, так и на тестовом наборах данных.
- Для полиномиальной регрессии это степень полинома, но другие алгоритмы могут иметь свои гиперпараметры, определяющие сложность модели.



Недообучение и переобучение

- Это значит, что когда мы хотим выбрать сложность модели, а также хотим оценить работу модели, то нам нужно изучать ошибки модели как на обучающем, так и на тестовом наборах данных.
- Для полиномиальной регрессии это степень полинома, но другие алгоритмы могут иметь свои гиперпараметры, определяющие сложность модели.

Рассмотрим подробнее в следующей лекции



Полиномиальная регрессия

Настройка параметров модели



Полиномиальная регрессия

- Давайте выберем оптимальную сложность модели (степень полинома).
- Как мы обсуждали раньше, нам потребуется обращать внимание на ошибки как на обучающем, так и на тестовом наборах данных — чтобы избежать переобучения (overfitting).



Полиномиальная регрессия

Внедрение модели



Регуляризация



Регуляризация

- Регуляризация помогает решить несколько общих проблем моделей, применяя:
 - Минимизацию сложности модели
 - Штрафы на функцию потерь
 - Уменьшая переобученность модели (добавляя больше смещения, чтобы уменьшить дисперсию)



Регуляризация

- В двух словах регуляризация – это уменьшение переобученности модели (overfitting):
 - Требуется добавления смещения (bias)
 - Требуется поиска оптимального значения гиперпараметра для штрафа



Регуляризация

- Три основных типа регуляризации:
 - L1-регуляризация – регрессия LASSO
 - L2-регуляризация – регрессия RIDGE
 - Комбинация L1 и L2 – Elastic Net



Регуляризация

- L1-регуляризация добавляет штраф, равный абсолютному значению величин коэффициентов
 - Ограничивает размер коэффициентов
 - Может создавать разреженные модели, в которых некоторые коэффициенты становятся нулями



Регуляризация

- L1-регуляризация добавляет штраф, равный абсолютному значению величин коэффициентов

$$\sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^p |\beta_j| = \text{RSS} + \lambda \sum_{j=1}^p |\beta_j|$$



Регуляризация

- L2-регуляризация добавляет штраф, равный квадрату величин коэффициентов
 - Все коэффициенты уменьшаются на один порядок
 - Вовсе не обязательно обнуляет коэффициенты



Регуляризация

- L2-регуляризация добавляет штраф, равный квадрату величин коэффициентов

$$\sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^p \beta_j^2 = \text{RSS} + \lambda \sum_{j=1}^p \beta_j^2$$



Регуляризация

- Elastic Net комбинирует L1 и L2, определяя соотношение между ними с помощью параметра alpha:

$$\sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 + \lambda \left(\frac{1-\alpha}{2} \sum_{j=1}^p \beta_j^2 + \alpha \sum_{j=1}^p |\beta_j| \right)$$



Регуляризация

- Elastic Net комбинирует L1 и L2, определяя соотношение между ними с помощью параметра alpha:

$$\alpha = 0$$

$$\sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 + \lambda \left(\frac{1-\alpha}{2} \sum_{j=1}^p \beta_j^2 + \alpha \sum_{j=1}^p |\beta_j| \right)$$



Регуляризация

- Elastic Net комбинирует L1 и L2, определяя соотношение между ними с помощью параметра alpha:

$$\alpha = 0$$

$$\sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 + \lambda \left(\frac{1-\alpha}{2} \sum_{j=1}^p \beta_j^2 + \alpha \sum_{j=1}^p |\beta_j| \right)$$



Регуляризация

- Elastic Net комбинирует L1 и L2, определяя соотношение между ними с помощью параметра alpha:

$$\alpha = 1$$

$$\sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 + \lambda \left(\frac{1-\alpha}{2} \sum_{j=1}^p \beta_j^2 + \alpha \sum_{j=1}^p |\beta_j| \right)$$



Регуляризация

- Elastic Net комбинирует L1 и L2, определяя соотношение между ними с помощью параметра alpha:

$$\alpha = 1$$

$$\sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 + \lambda \left(\frac{1-\alpha}{2} \sum_{j=1}^p \beta_j^2 + \alpha \sum_{j=1}^p |\beta_j| \right)$$



Регуляризация

- Elastic Net комбинирует L1 и L2, определяя соотношение между ними с помощью параметра alpha:

$$\sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 + \lambda \left(\frac{1-\alpha}{2} \sum_{j=1}^p \beta_j^2 + \alpha \sum_{j=1}^p |\beta_j| \right)$$



Регуляризация

- Эти методы регуляризации имеют накладные расходы:
 - Добавляется ещё один гиперпараметр, который необходимо оптимизировать
 - Мультипликатор для штрафа определяет “силу” этого штрафа



Регуляризация

- И начнём мы изучение с L2-регуляризации (RIDGE-регрессия), поскольку после возведения в квадрат это слагаемое становится более интуитивно понятным
- Но перед этим нам нужно предварительно изучить:
 - Масштабирование признаков (feature scaling)
 - Кросс-валидацию



Масштабирование признаков



Масштабирование признаков (feature scaling)

- Масштабирование признаков приносит много пользы процессам машинного обучения!
- Для некоторых алгоритмов, измеряющих расстояние между точками (например, KNN) – масштабирование обязательно для получения хороших результатов



Масштабирование признаков (feature scaling)

- Масштабирование признаков улучшает сходимость итераций для алгоритмов, которые зависят от масштаба признаков
- Если признаки имеют разный масштаб, то одни веса (weights) могут обновляться быстрее других, поскольку сами значения признаков участвуют в обновлении весов



Масштабирование признаков (feature scaling)

- Некоторые алгоритмы не зависят от масштаба признаков - например, алгоритмы на основе деревьев классификации и регрессии



Масштабирование признаков (feature scaling)

- Сведение разных признаков к единой шкале позволяет сравнивать признаки, имеющие разные единицы измерения
- Это позволяет напрямую сравнивать разные коэффициенты модели между собой.



Масштабирование признаков (feature scaling)

- Возможные подводные камни:
 - Новые данные должны быть смасштабированы перед тем, как добавлять их в модель
 - Влияет на интерпретацию коэффициентов – легче сравнивать коэффициенты между собой, но сложнее масштабироваться обратно на исходные шкалы.



Масштабирование признаков (feature scaling)

- Преимущества:
 - Может заметно повысить производительность
 - Для некоторых моделей это просто необходимо
 - Серьёзных минусов практически нет



Масштабирование признаков (feature scaling)

- Два основных способа масштабирования:
 - Стандартизация
 - Данные должны получить среднее значение $\mu=0$ и среднеквадратическое отклонение $\sigma=1$
 - Нормализация
 - Данные должны оказаться в диапазоне от 0 до 1



Масштабирование признаков (feature scaling)

- Стандартизация:
 - Данные должны получить среднее значение $\mu=0$ и среднеквадратическое отклонение $\sigma=1$

$$X_{changed} = \frac{X - \mu}{\sigma}$$



Масштабирование признаков (feature scaling)

Другое название: Z-score normalization

- Стандартизация:
 - Данные должны получить среднее значение $\mu=0$ и среднеквадратическое отклонение $\sigma=1$

$$X_{changed} = \frac{X - \mu}{\sigma}$$



Масштабирование признаков (feature scaling)

- Нормализация
 - Данные должны оказаться в диапазоне от 0 до 1

$$X_{changed} = \frac{X - X_{min}}{X_{max} - X_{min}}$$



Масштабирование признаков (feature scaling)

- Есть много других методов для масштабирования признаков. Scikit-learn предоставляет классы для обучения (“fit”) и трансформации данных (“transform”).
- Давайте подробнее обсудим вызовы “fit” и “transform” с точки зрения масштабирования данных.



Масштабирование признаков (feature scaling)

- Метод `.fit()` вычисляет нужные метрики – $X_{\min}$, $X_{\max}$, среднее значение, дисперсия и т.д.
- Метод `.transform()` масштабирует данные и возвращает новую версию данных.
- Мы видели похожий процесс ранее, для полиномиальной регрессии.



Масштабирование признаков (feature scaling)

- Важные моменты:
 - Мы вызываем `.fit()` только для обучающих данных
 - Вычисление статистической информации должно выполняться только на обучающих данных
 - Не нужно добавлять в модель какие-то знания, которые содержатся в тестовых данных



Масштабирование признаков (feature scaling)

- Если использовать все данные, то произойдёт “утечка данных”:
 - Вычисление статистик на полном наборе данных приведёт к тому, что некоторые крупницы информации из тестовых данных перетекут в обучающие данные во время трансформации `.transform()`.



Масштабирование признаков (feature scaling)

- Процесс масштабирования признаков:
 - Разбиваем данные на обучающий и тестовый наборы
 - Вызываем `.fit()` только для обучающих данных
 - Вызываем `.transform()` для обучающих данных
 - Вызываем `.transform()` для тестовых данных



Масштабирование признаков (feature scaling)

- Нужно ли масштабировать целевую переменную?
 - В общем случае это не требуется
 - Нормализуя распределение на выходе, мы меняем определение целевой переменной – это будет отличаться от того, что мы пытаемся предсказывать.



Масштабирование признаков (feature scaling)

- Нужно ли масштабировать целевую переменную?
 - В общем случае это не требуется
 - Нормализуя распределение на выходе, мы меняем определение целевой переменной – это будет отличаться от того, что мы пытаемся предсказывать.
 - Возможны даже негативные эффекты – см. статью ниже:

stats.stackexchange.com/questions/111467



Масштабирование признаков (feature scaling)

- Теперь, когда мы знаем полезные свойства масштабирования признаков, давайте перейдём к изучению полезных свойств кросс-валидации!



Кросс-валидация



Кросс-валидация (перекрёстная проверка)

- Кросс-валидация – это набор более продвинутых методов для разбиения данных на обучающий и тестовый наборы
- Когда мы делаем простое разбиение, то теряем возможность отлаживать гиперпараметры модели на всём объёме данных.



Кросс-валидация (перекрёстная проверка)

- Но можно ли как-то сделать следующее?
 - Обучать модель на всех данных
 - Проверять модель на всех данных
- Это можно сделать с помощью кросс-валидации!
- Давайте сделаем обзор, что это такое...



Кросс-валидация (перекрёстная проверка)

- Для начала преобразуем данные ниже в разноцветные блоки

x

y

Area m ²	Bedrooms	Bathrooms	Price
200	3	2	\$500,000
190	2	1	\$450,000
230	3	3	\$650,000
180	1	1	\$400,000
210	2	2	\$550,000



Кросс-валидация (перекрёстная проверка)

- Преобразуем в общую форму

X			y
x_1	x_2	x_3	y
x_1^1	x_1^1	x_1^1	y_1
x_1^2	x_1^2	x_1^2	y_2
x_1^3	x_1^3	x_1^3	y_3
x_1^4	x_1^4	x_1^4	y_4
x_1^5	x_1^5	x_1^5	y_5



Кросс-валидация (перекрёстная проверка)

- Разбиваем на обучающий и тестовый наборы

	X			y
	x_1	x_2	x_3	y
TRAIN	x_1^1	x_1^1	x_1^1	y_1
	x_1^2	x_1^2	x_1^2	y_2
	x_1^3	x_1^3	x_1^3	y_3
TEST	x_1^4	x_1^4	x_1^4	y_4
	x_1^5	x_1^5	x_1^5	y_5



Кросс-валидация (перекрёстная проверка)

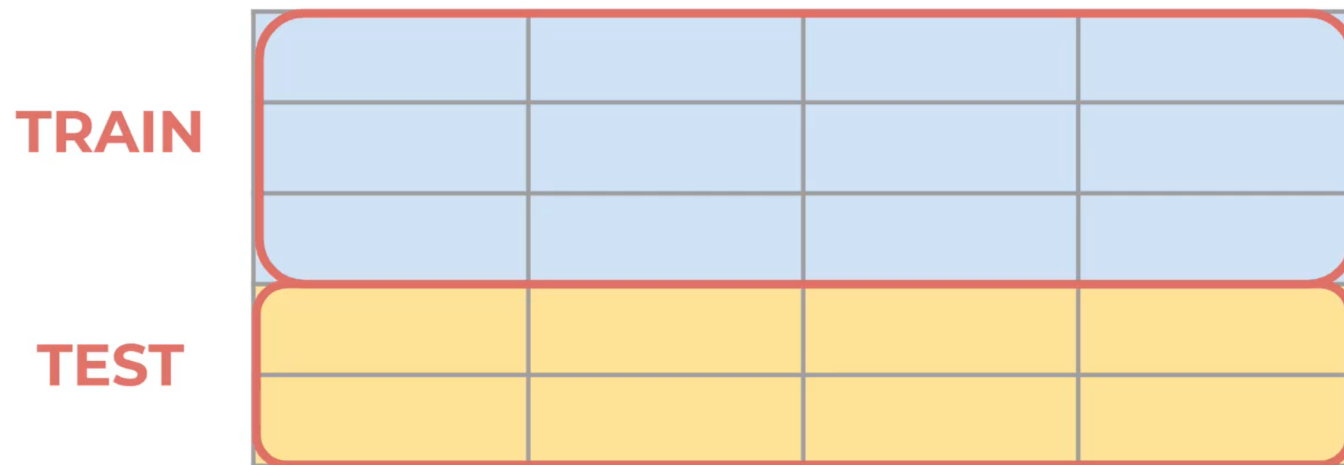
- Разбиваем на обучающий и тестовый наборы

TRAIN	x_1^1	x_1^1	x_1^1	y_1
	x_1^2	x_1^2	x_1^2	y_2
	x_1^3	x_1^3	x_1^3	y_3
TEST	x_1^4	x_1^4	x_1^4	y_4
	x_1^5	x_1^5	x_1^5	y_5



Кросс-валидация (перекрёстная проверка)

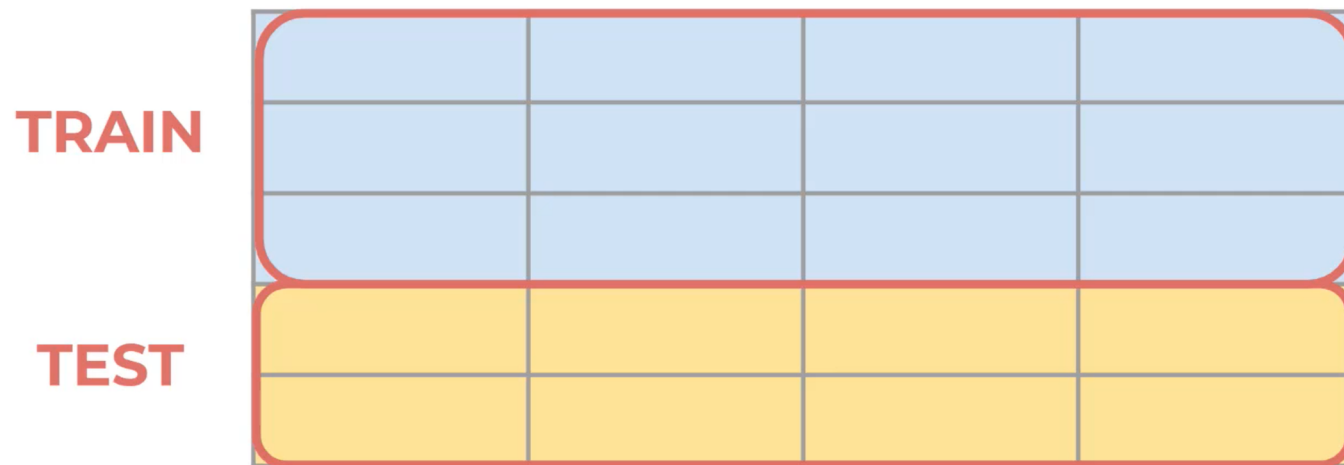
- Разбиваем на обучающий и тестовый наборы





Кросс-валидация (перекрёстная проверка)

- Далее повернём и поменяем размер:





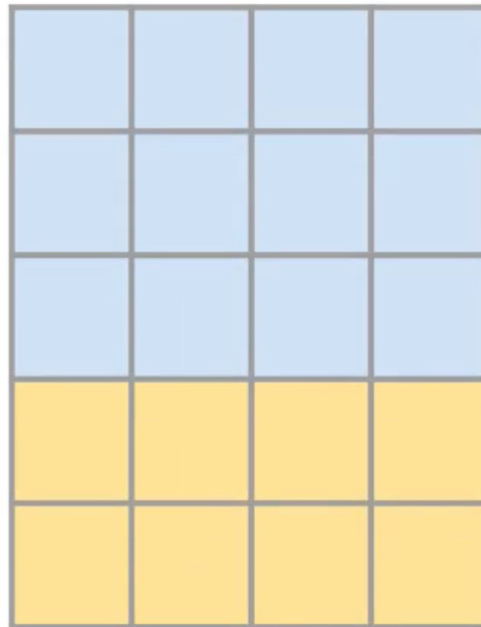
Кросс-валидация (перекрёстная проверка)

- Далее повернём и поменяем размер:



Кросс-валидация (перекрёстная проверка)

- Далее повернём и поменяем размер:





Кросс-валидация (перекрёстная проверка)

- Далее повернём и поменяем размер:





Кросс-валидация (перекрёстная проверка)

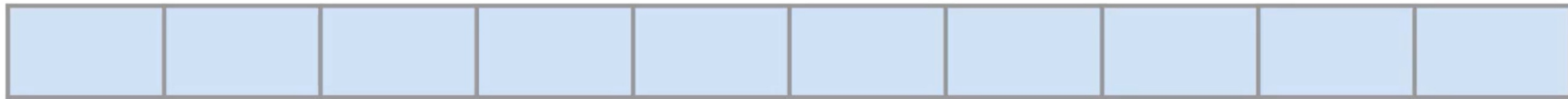
- В итоге получаем следующее представление:





Кросс-валидация (перекрёстная проверка)

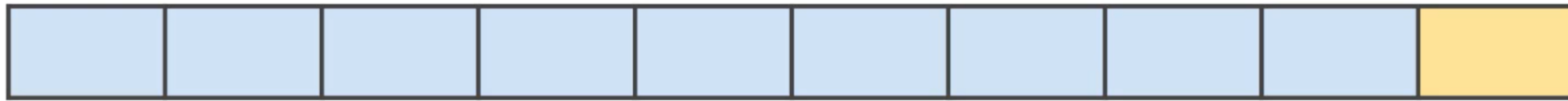
- Начнём с полного набора всех данных:





Кросс-валидация (перекрёстная проверка)

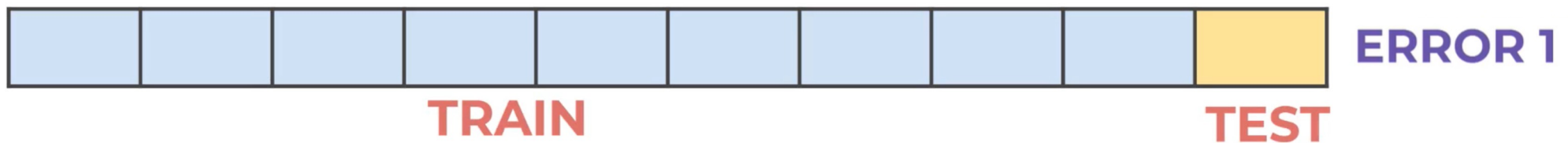
- $1/K$ часть всех данных отводим на тестовый набор:





Кросс-валидация (перекрёстная проверка)

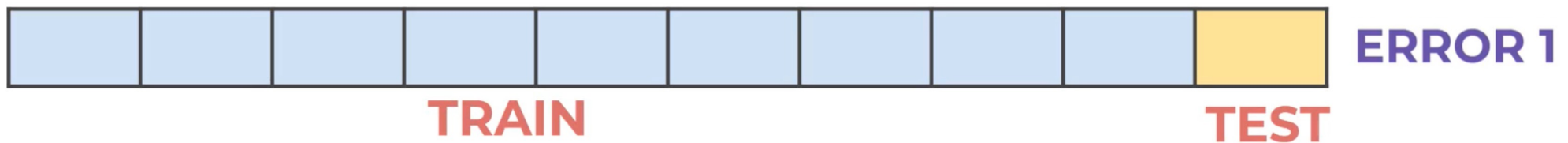
- $1/K$ часть всех данных отводим на тестовый набор:





Кросс-валидация (перекрёстная проверка)

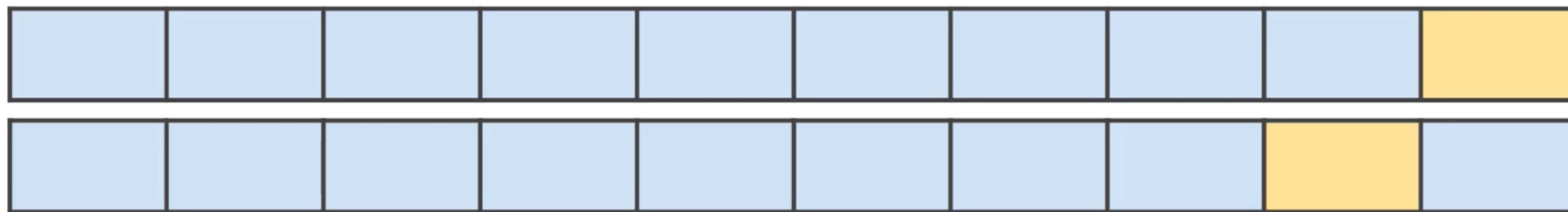
- Обучаем модель и вычисляем метрики:





Кросс-валидация (перекрёстная проверка)

- Делаем то же самое для другого разбиения:



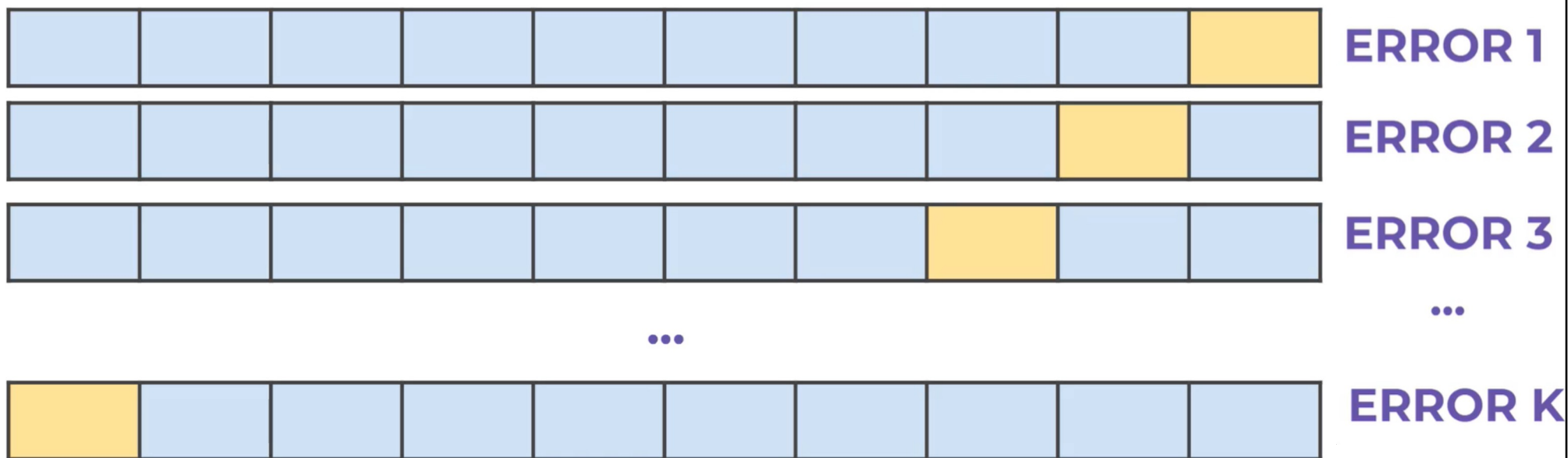
ERROR 1

ERROR 2



Кросс-валидация (перекрёстная проверка)

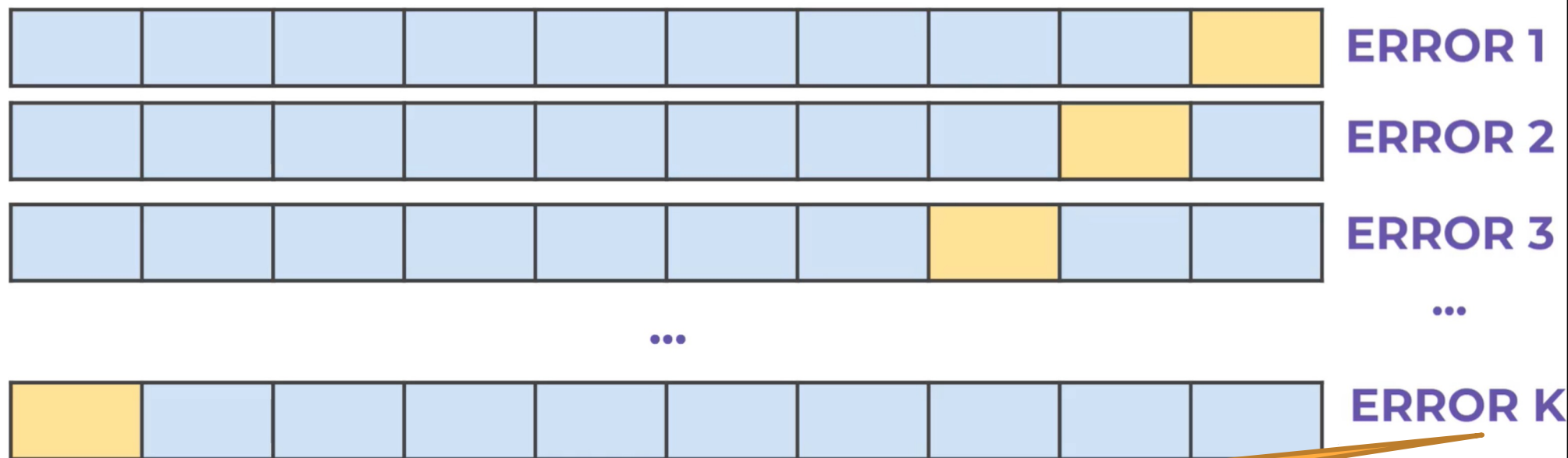
- Повторяем для всех возможных разбиений:





Кросс-валидация (перекрёстная проверка)

- Повторяем для всех возможных разбиений:



Усредняем ошибки



Кросс-валидация (перекрёстная проверка)

- Мы смогли обучать модель на всех данных и тестировать на всех данных!
- Мы получили лучшее понимание того, как ведёт себя модель на разных разбиениях



Кросс-валидация (перекрёстная проверка)

- Мы смогли обучать модель на всех данных и тестировать на всех данных!
- Мы получили лучшее понимание того, как ведёт себя модель на разных разбиениях
- Чего это нам стоило?
 - Мы повторяли вычисления K раз!
- Обычно выбирают $K=10$, т.е. тестовый набор 10%



Кросс-валидация (перекрёстная проверка)

- Один вопрос, который возникает при таком подходе – это объективная настройка гиперпараметров
- Когда мы используем все данные для подбора гиперпараметров, корректно ли мы оцениваем метрики работы модели?



Кросс-валидация (перекрёстная проверка)

- Как мы можем понять работу модели на данных, которые модели не видела при обучении и при подборе гиперпараметров?
- Для этого можно отложить в сторону часть данных – “hold-out” test set.



Кросс-валидация (перекрёстная проверка)

- В самом начале отделяем часть данных





Кросс-валидация (перекрёстная проверка)

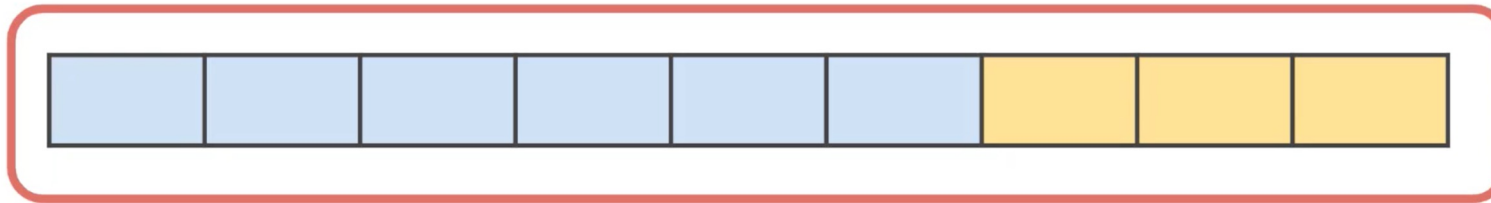
- Выполняем разбиение, как мы делали раньше





Кросс-валидация (перекрёстная проверка)

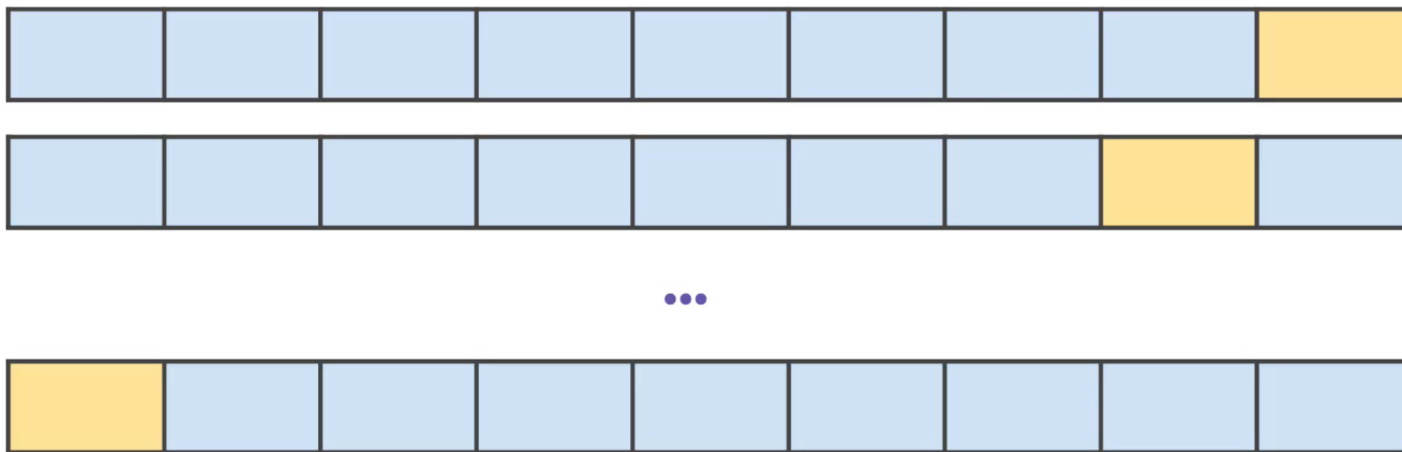
- Обучение/тюнинг только на этих данных





Кросс-валидация (перекрёстная проверка)

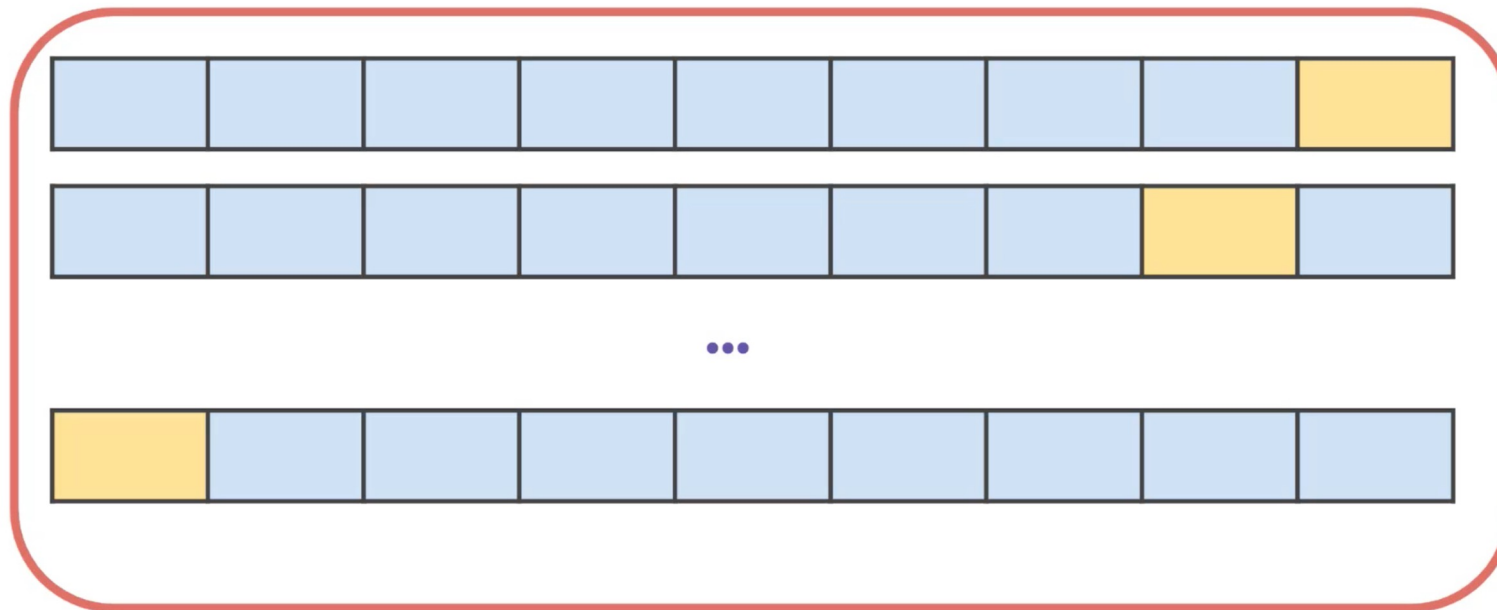
- Либо то же самое для кросс-валидации





Кросс-валидация (перекрёстная проверка)

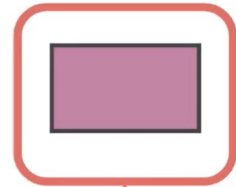
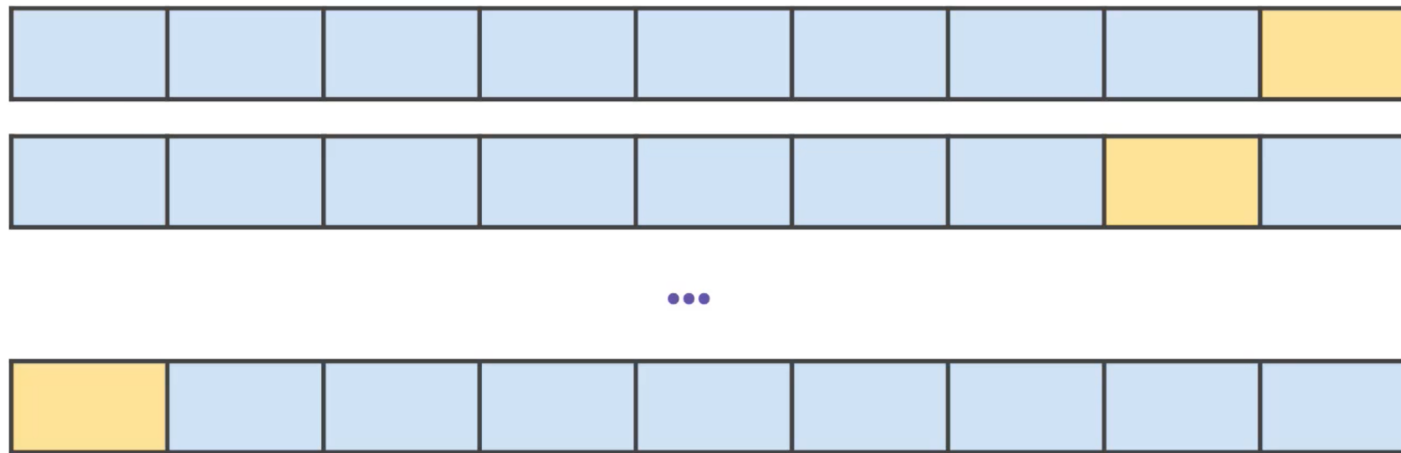
- Обучение/тюнинг только на этих данных





Кросс-валидация (перекрёстная проверка)

- После обучения/тюнинга выполняем финальную проверку на данных “hold-out”



После этого уже нельзя
продолжать тюнинг
параметров!



Кросс-валидация (перекрёстная проверка)

- Разбиение “Train – Validation – Test”



TRAIN

VALIDATION

TEST

- Позволяет получить корректные финальные оценки производительности модели
- После этого больше не редактируем модель!



Кросс-валидация (перекрёстная проверка)

- Все эти подходы корректны, каждая ситуация уникальна!
- Учитывайте:
 - Ранее выполненные работы
 - Требования к отчётности по модели
 - Корректность оценки модели
 - Контекст применения данных и модели



Кросс-валидация (перекрёстная проверка)

- Во многих методах регуляризации есть параметры для настройки с помощью методов кросс-валидации
- Во время курса, для простоты изложения, мы часто будем использовать простое разбиение на обучающий и тестовый наборы данных.



Регуляризация для линейной регрессии

Настройка данных



Ridge-регрессия

Теория



Ridge-регрессия

- Ridge-регрессия – это метод регуляризации, позволяющий снизить вероятность переобученности модели (overfitting) на обучающем наборе данных
- Это достигается добавлением штрафующего слагаемого к ошибке, с квадратом значений коэффициентов



Ridge-регрессия

- Вспомним общую формулу линейной регрессии:

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \dots + \hat{\beta}_p x_p$$



Ridge-регрессия

- Находим коэффициенты бетта, минимизируя сумму квадратов остатков (residual sum of squares)

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \dots + \hat{\beta}_p x_p$$



Ridge-регрессия

- Находим коэффициенты бетта, минимизируя сумму квадратов остатков (residual sum of squares)

$$\text{RSS} = \sum_{i=1}^n (y_i - \hat{y}_i)^2$$



Ridge-регрессия

- Мы можем подставить формулу для $\hat{\mathbf{y}}$:

$$\begin{aligned}\text{RSS} &= \sum_{i=1}^n (y_i - \hat{y}_i)^2 \\ &= \sum_{i=1}^n (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_{i1} - \hat{\beta}_2 x_{i2} - \dots - \hat{\beta}_p x_{ip})^2\end{aligned}$$



Ridge-регрессия

- Получается следующее:

$$\text{RSS} = \sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2$$



Ridge-регрессия

- Ridge-регрессия добавляет такой штраф (penalty):

$$\text{Error} = \sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^p \beta_j^2$$



Ridge-регрессия

- Ridge-регрессия добавляет такой штраф (penalty)
- В итоге мы минимизируем RSS+штраф

$$\text{Error} = \sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^p \beta_j^2$$



Ridge-регрессия

- Ridge-регрессия добавляет такой штраф (penalty)
- В итоге мы минимизируем RSS+штраф

$$\text{Error} = \sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^p \beta_j^2$$

Квадраты



Ridge-регрессия

- Ridge-регрессия добавляет такой штраф (penalty)
- В итоге мы минимизируем RSS+штраф

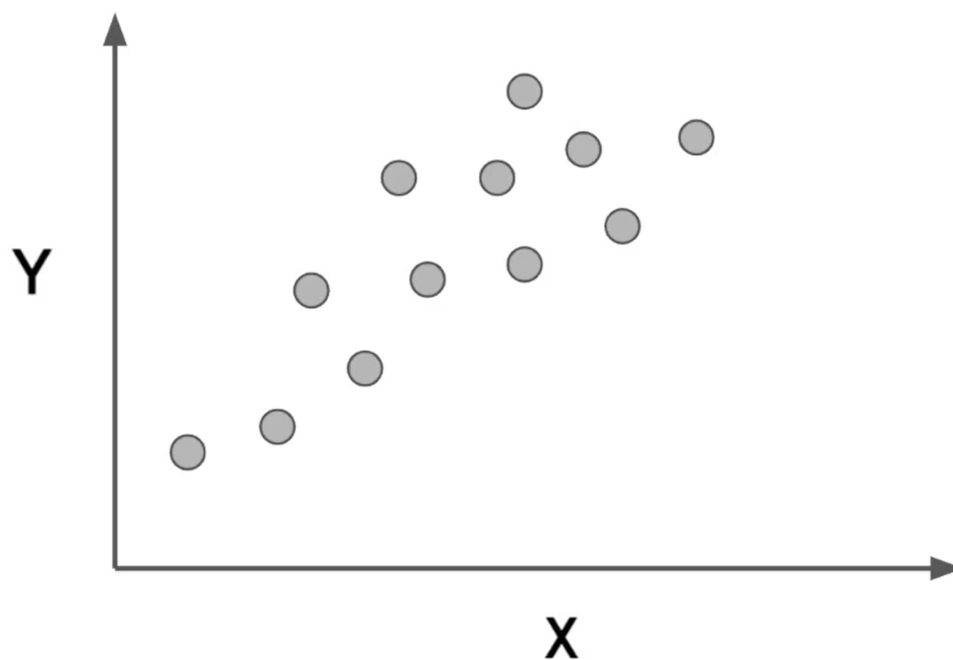
$$\text{Error} = \sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^p \beta_j^2$$

Параметр
лямбда



Ridge-регрессия

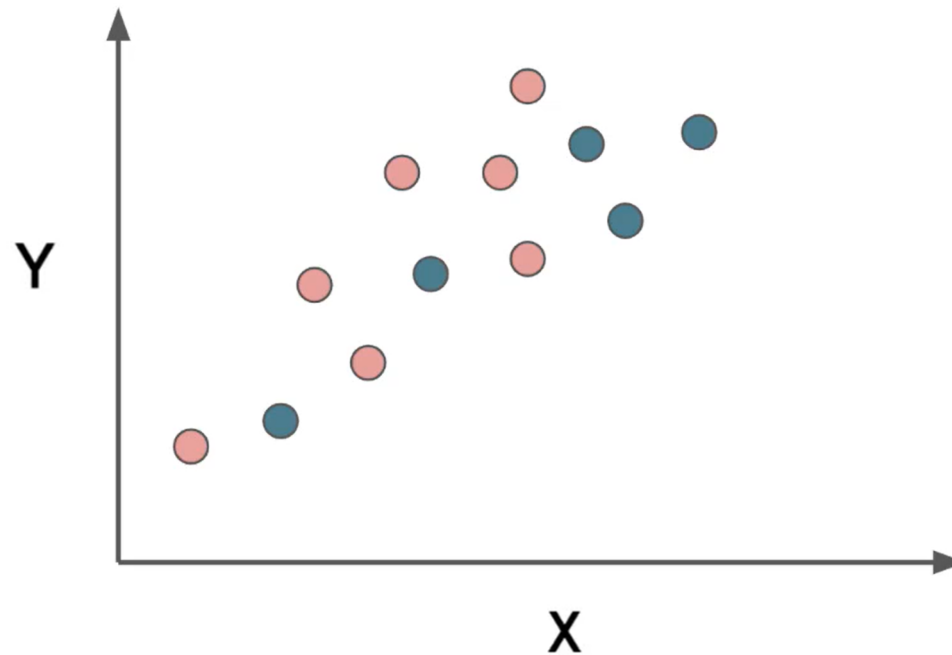
- Представим себе следующие данные





Ridge-регрессия

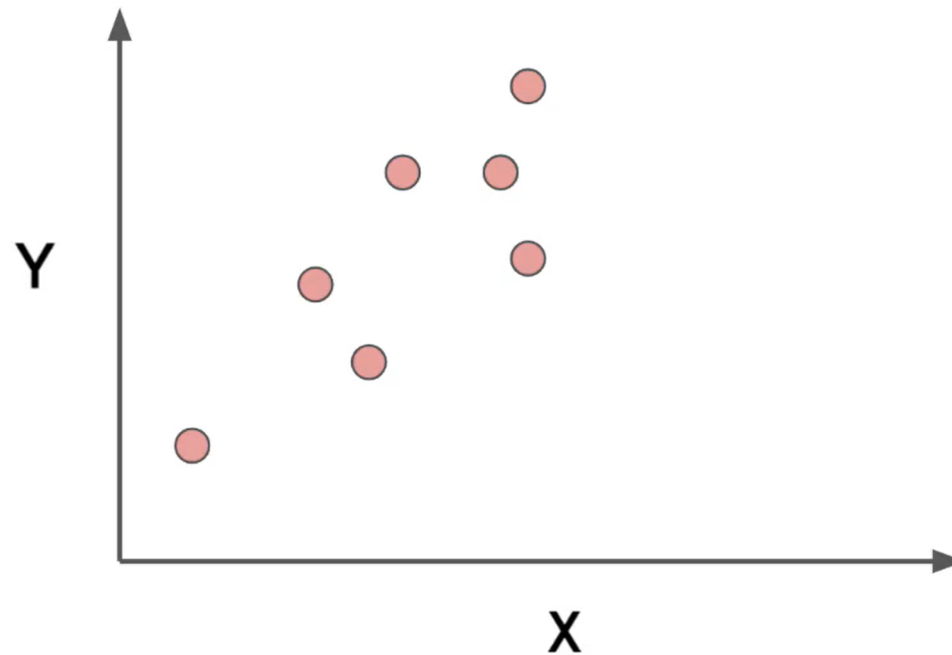
- Разбиваем на обучающий и тестовый наборы





Ridge-регрессия

- На обучающем наборе ищем линию $\hat{y} = \beta_1 x + \beta_0$

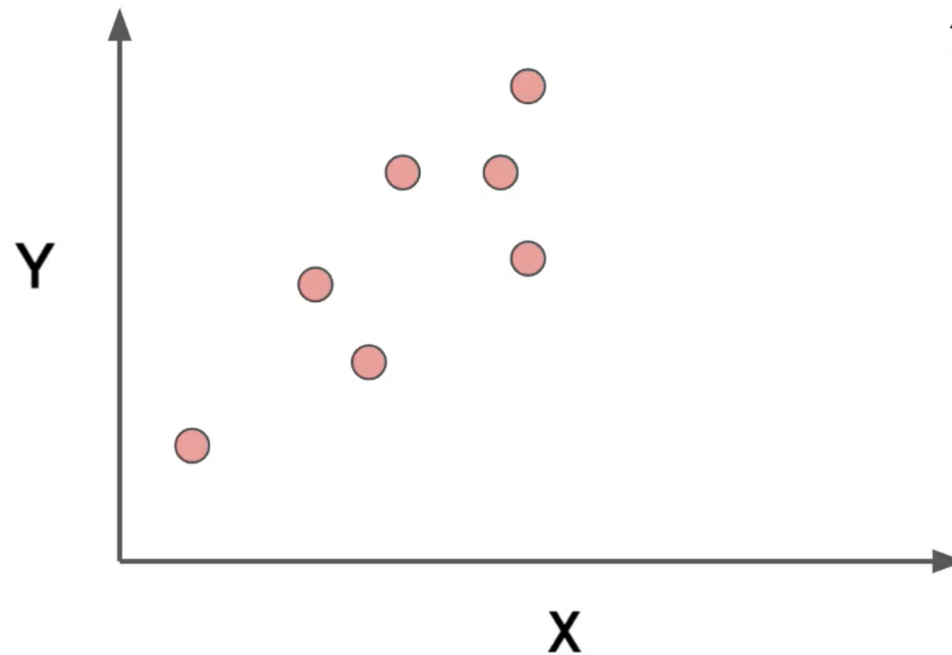




Ridge-регрессия

- Сначала применим только формулу RSS

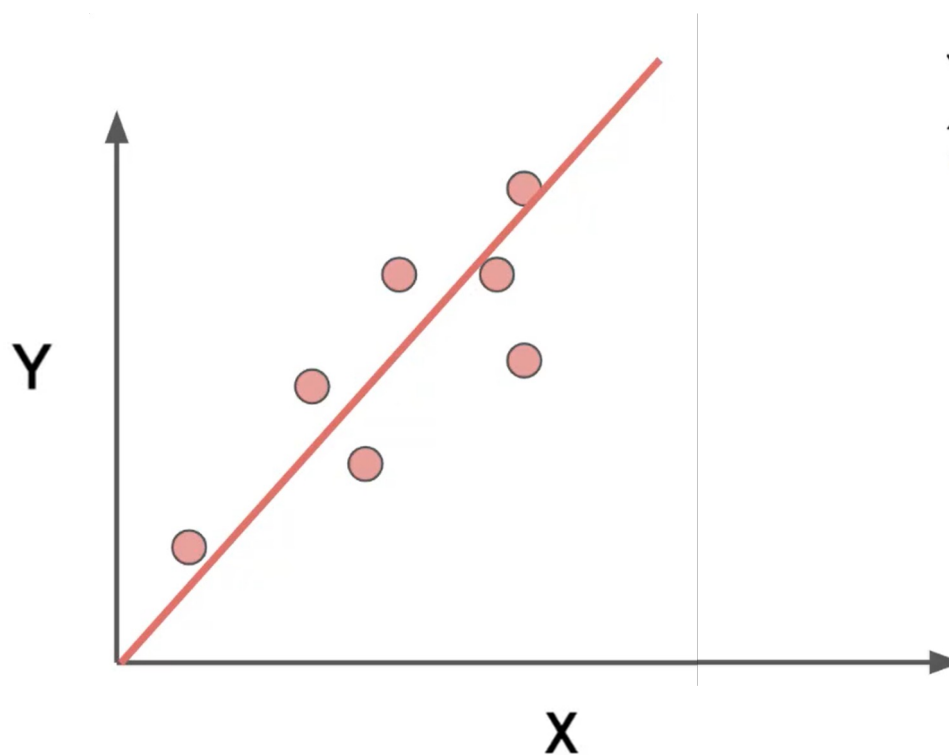
$$\sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2$$





Ridge-регрессия

- Сначала применим только формулу RSS

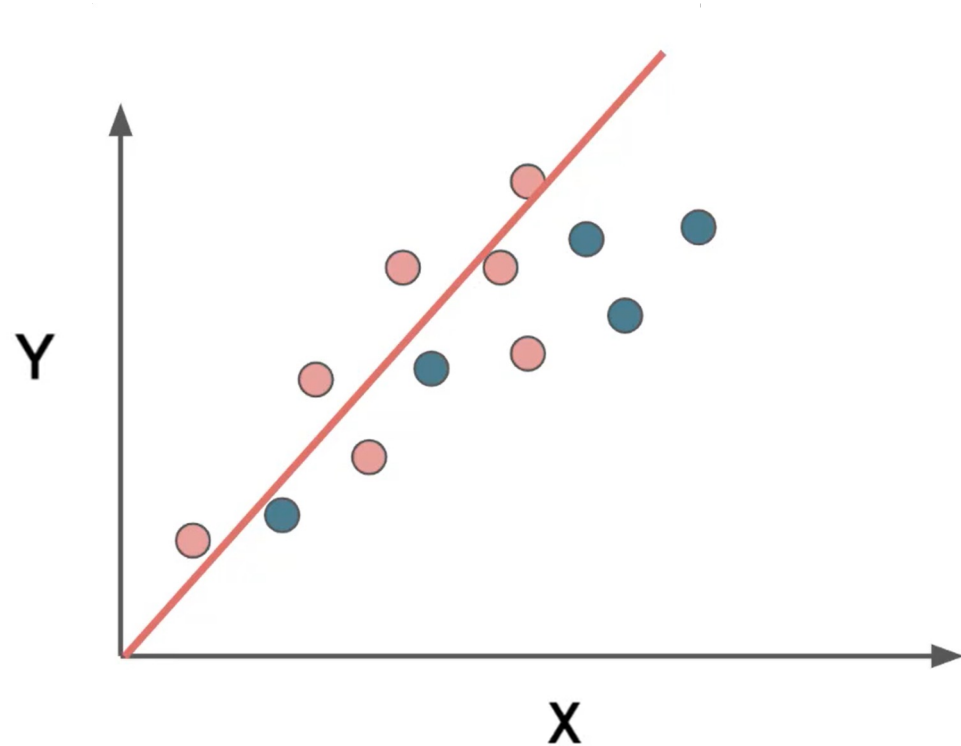


$$\sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2$$



Ridge-регрессия

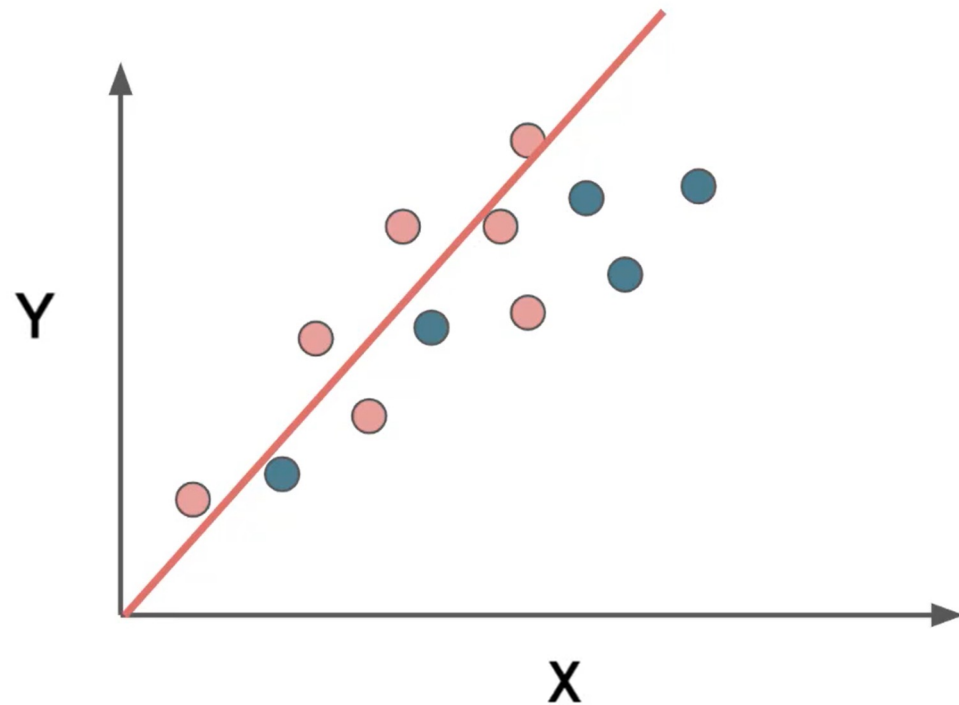
- Кажется, мы немного переобучили модель (overfit)





Ridge-регрессия

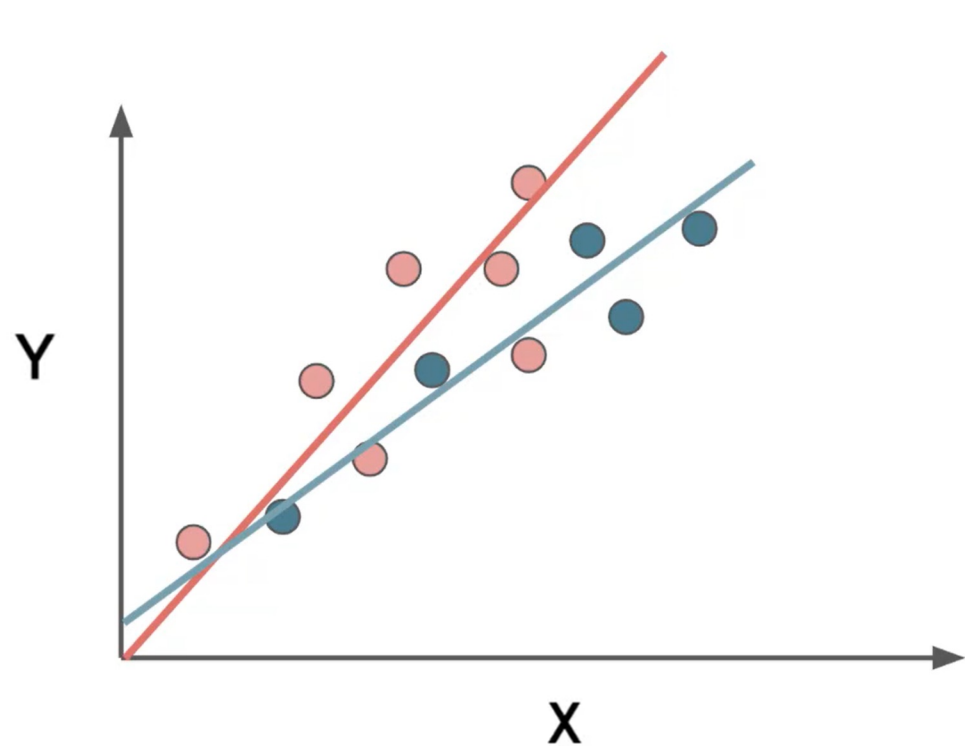
- Можем ли мы добавить немного смещения (bias), чтобы заметно уменьшить дисперсию (variance)?





Ridge-регрессия

- Тогда мы получим лучший вариант линии $\hat{y} = \beta_1 x + \beta_0$

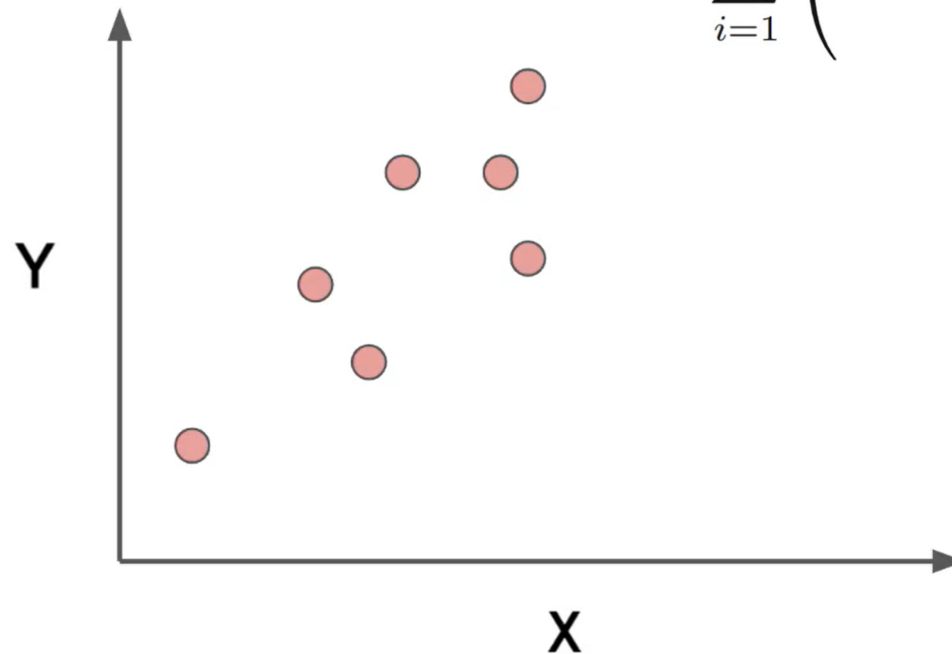




Ridge-регрессия

- Представим, что мы минимизируем RSS+Penalty:

$$\sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^p \beta_j^2$$

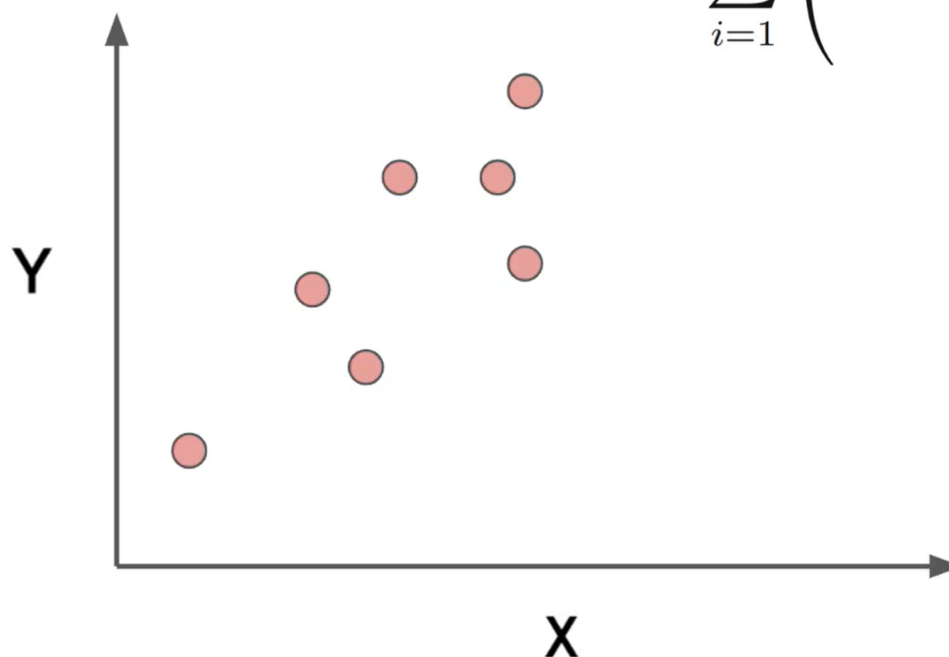




Ridge-регрессия

- Представим, что мы минимизируем RSS+Penalty:

$$\sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^p \beta_j^2$$

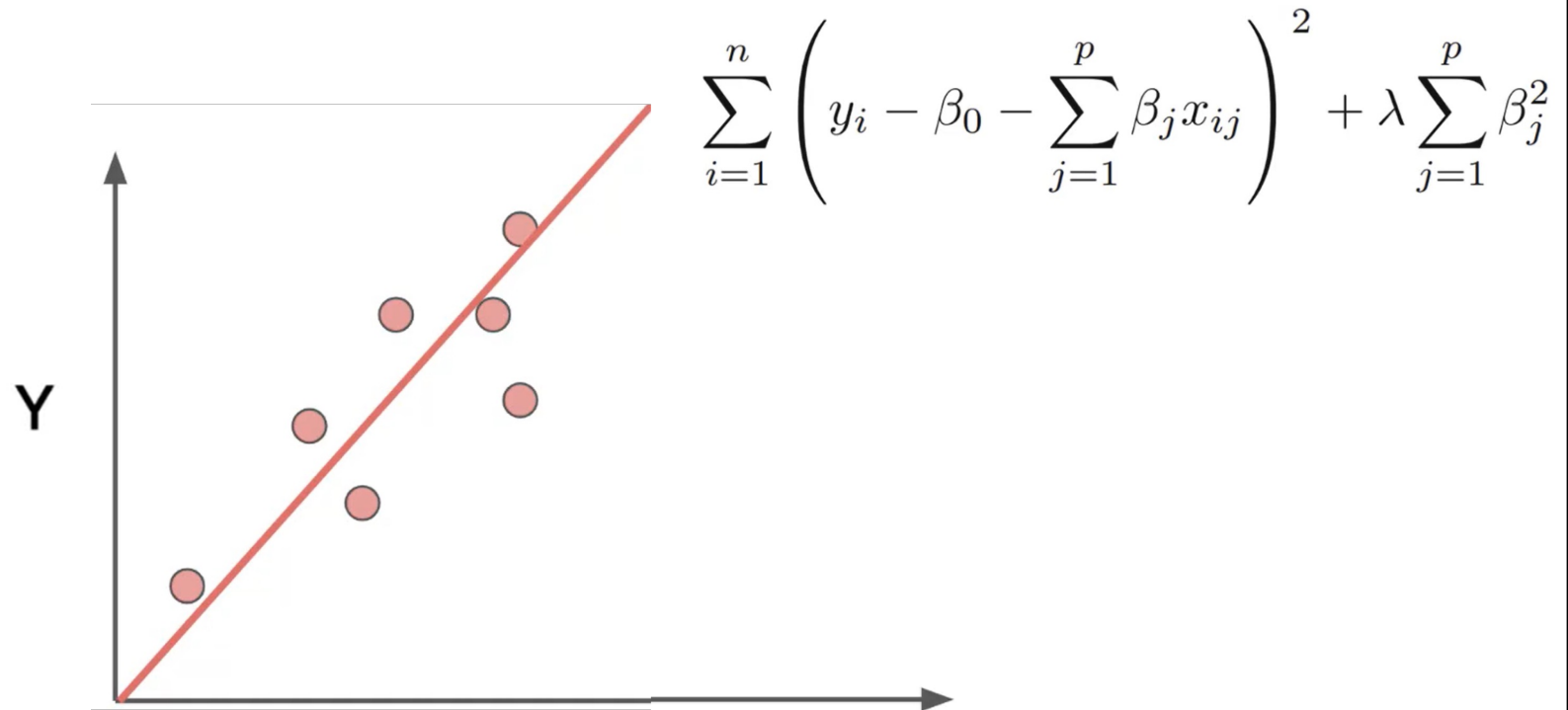


Альфа и
квадраты бетта



Ridge-регрессия

- Мы штрафуем большой наклон линии $\hat{y} = \beta_1 x + \beta_0$

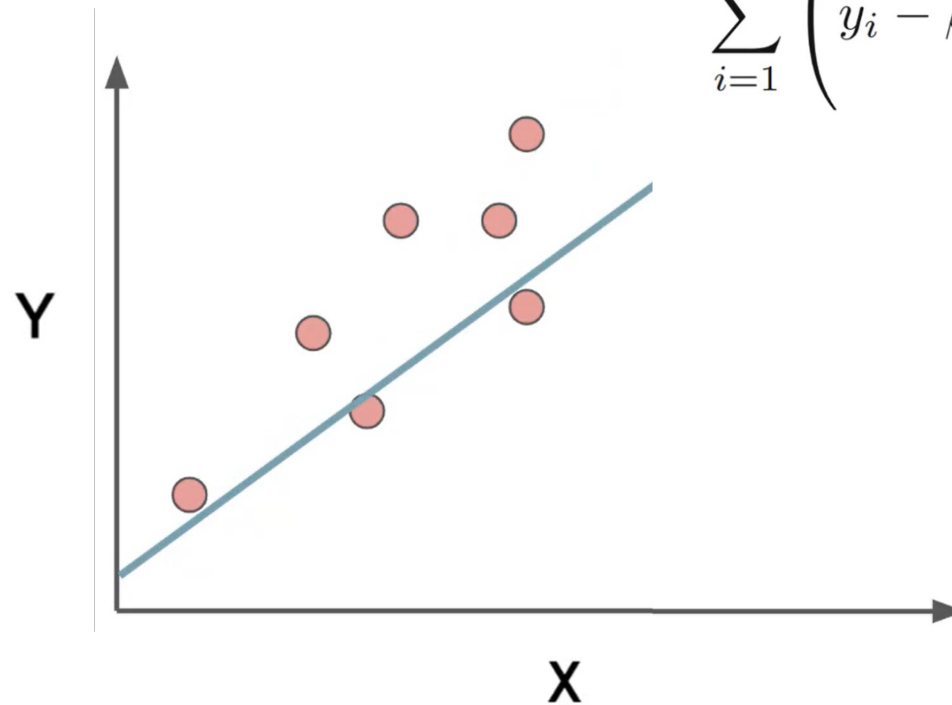




Ridge-регрессия

- Для одного признака уменьшается наклон линии:

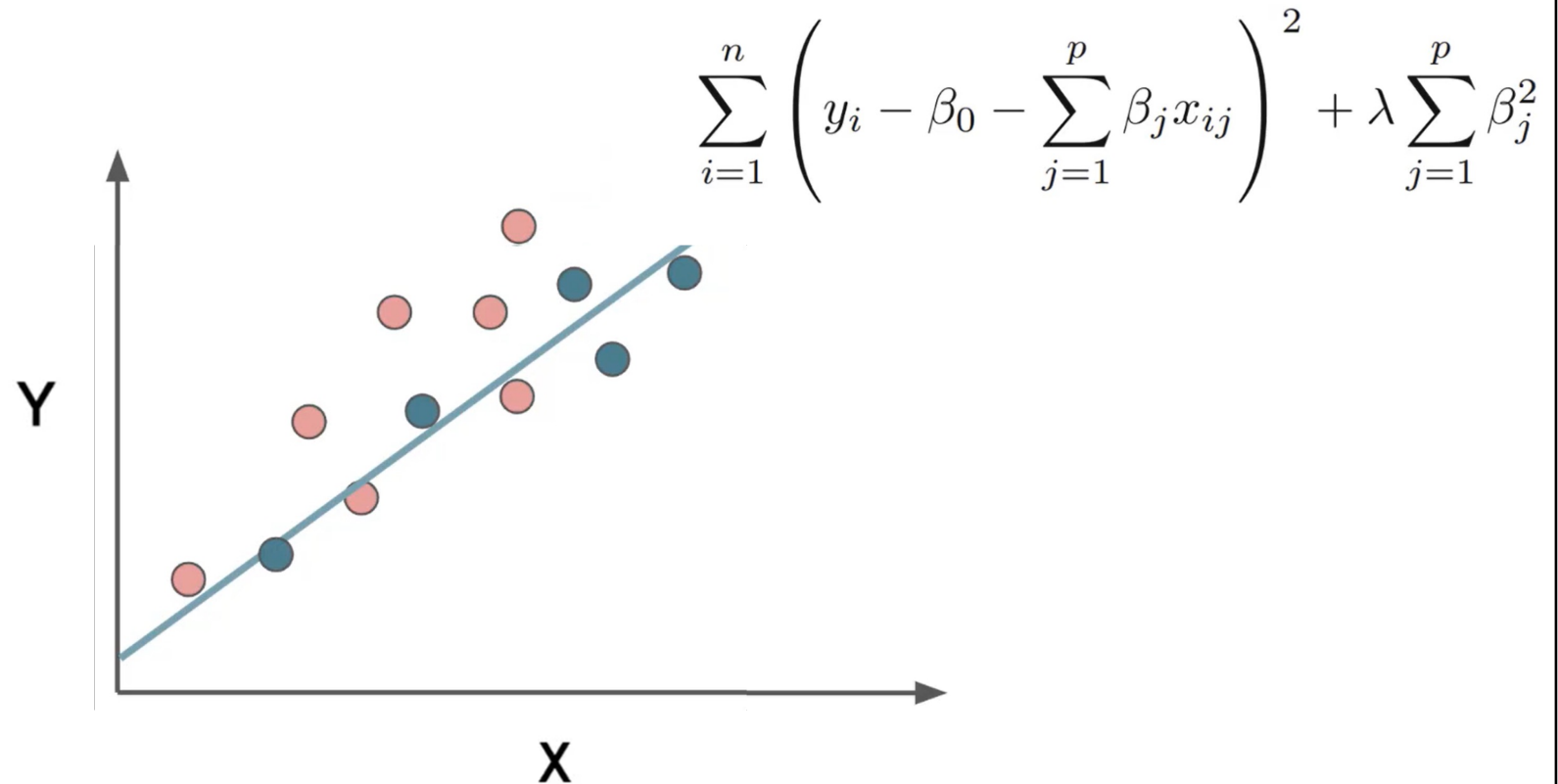
$$\sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^p \beta_j^2$$





Ridge-регрессия

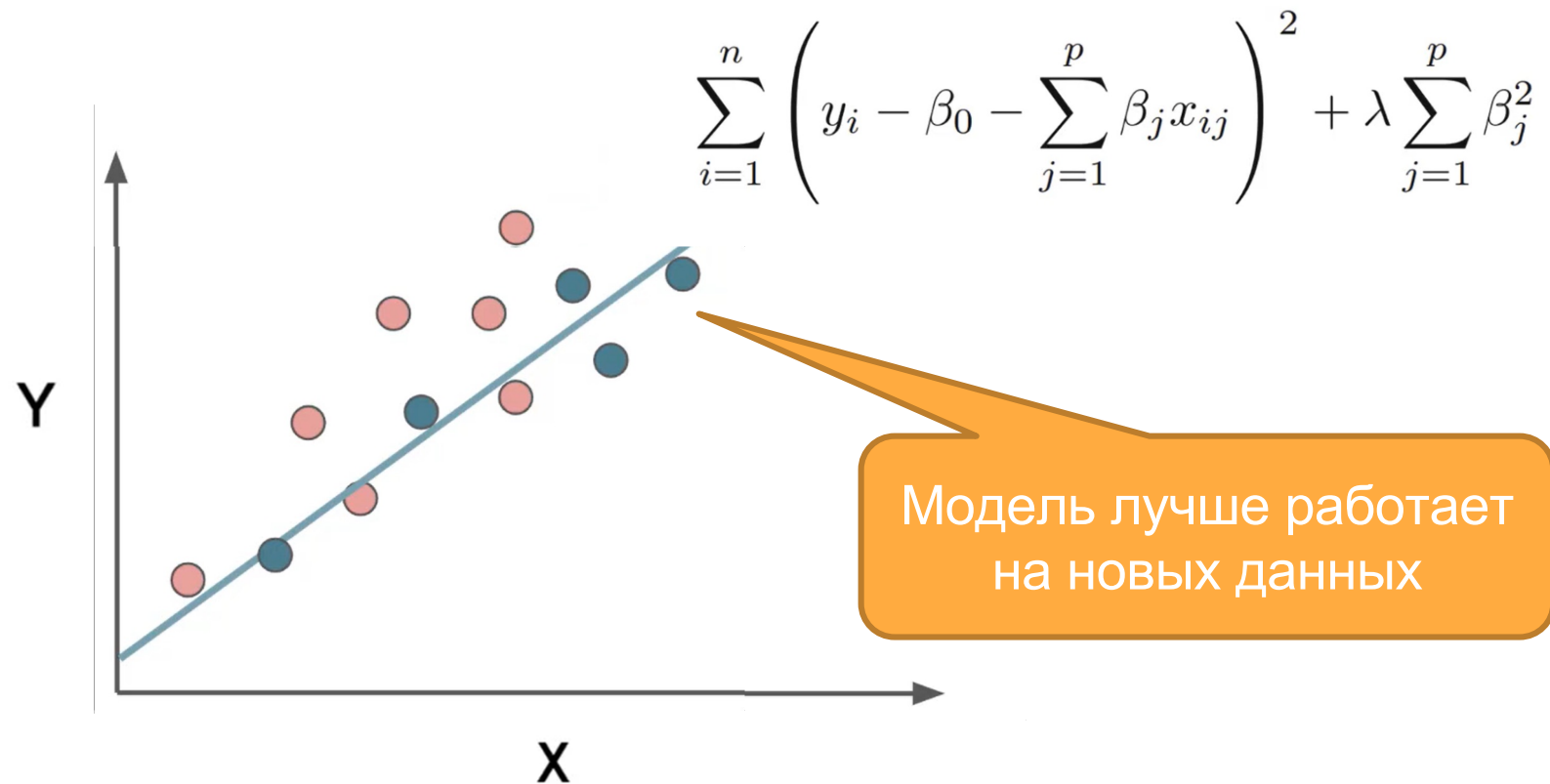
- Для одного признака уменьшается наклон линии:





Ridge-регрессия

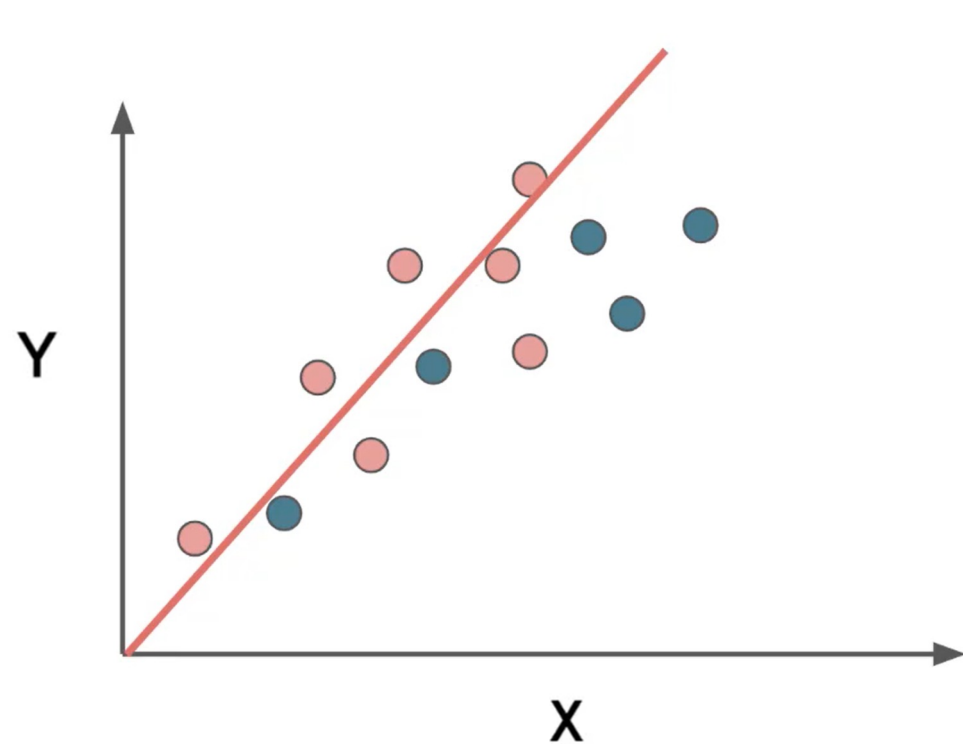
- Для одного признака уменьшается наклон линии:





Ridge-регрессия

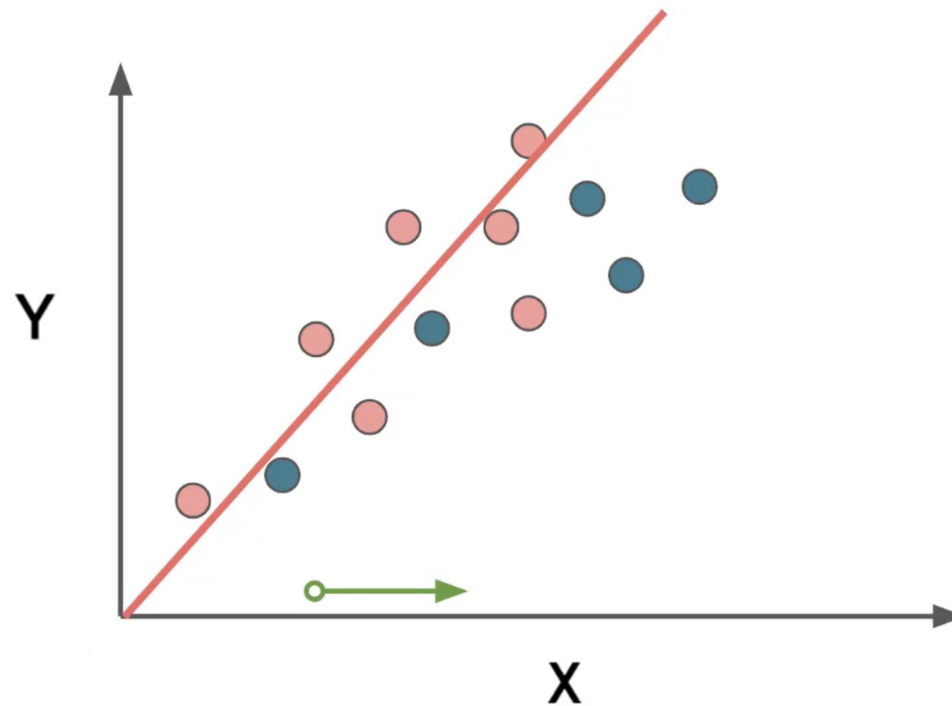
- Сравним это со случаем переобученности модели





Ridge-регрессия

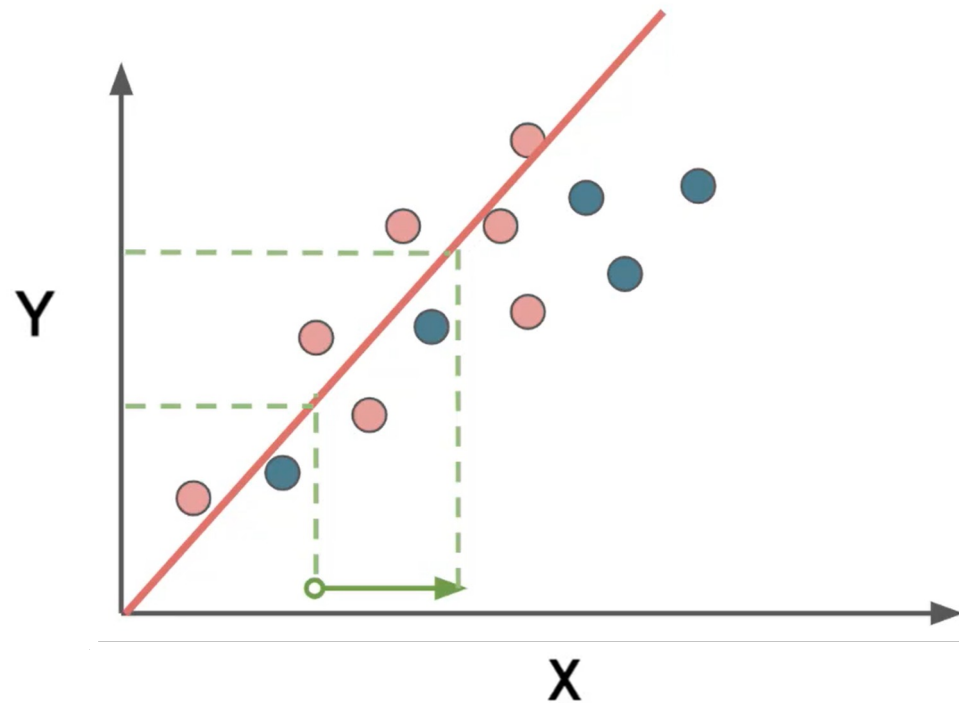
- Увеличение X приводит к бОльшему увеличению y





Ridge-регрессия

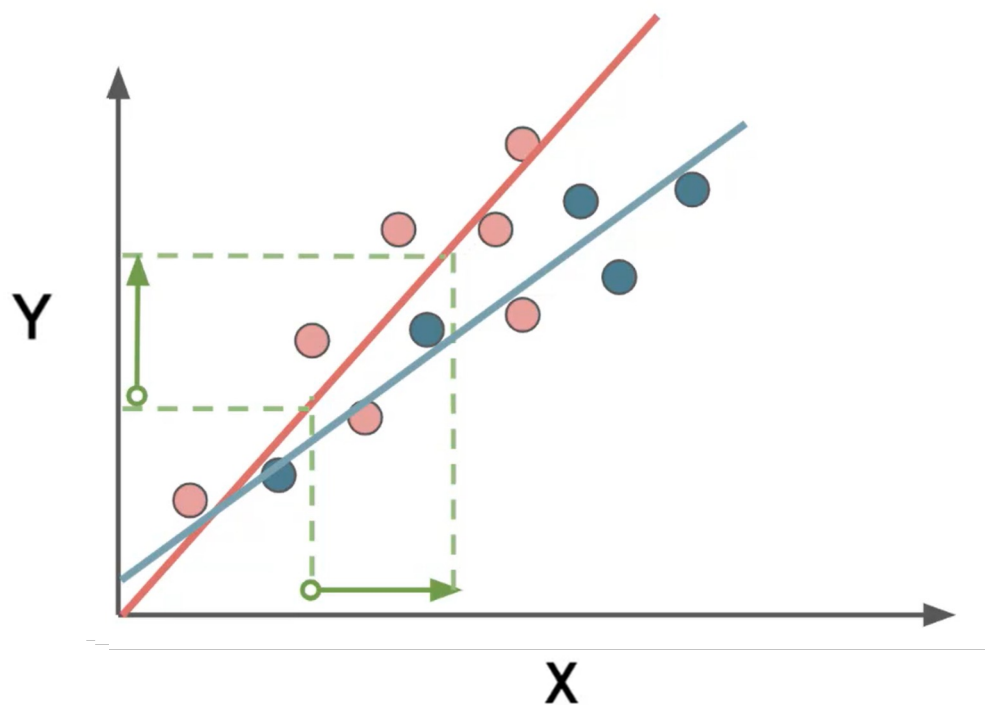
- Увеличение X приводит к бОльшему увеличению y





Ridge-регрессия

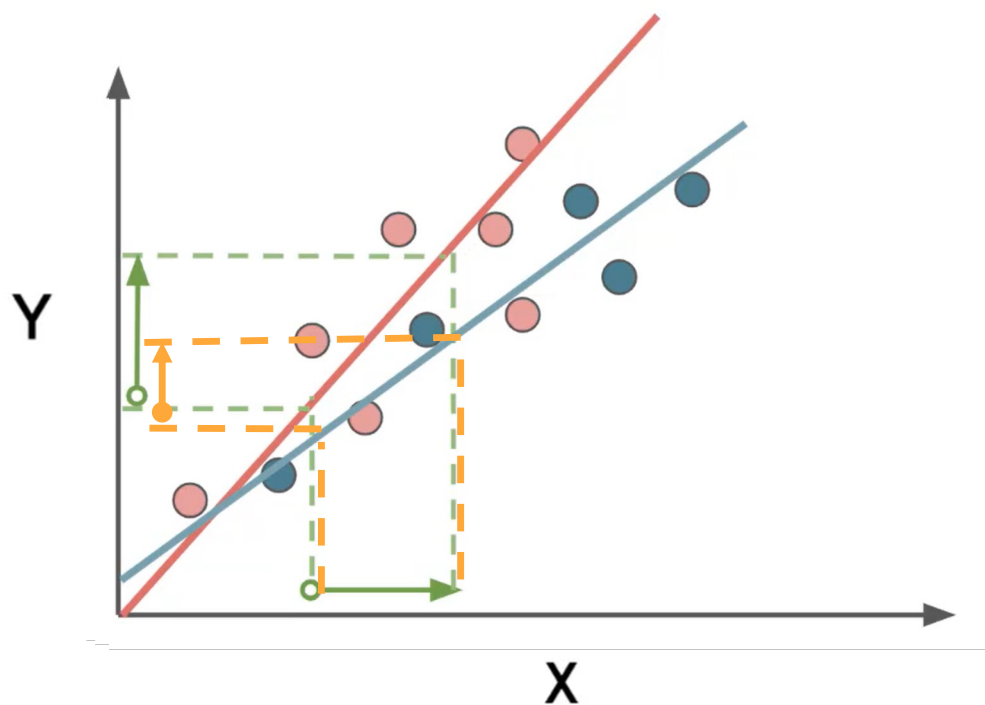
- Сравним это с результатом ridge-регрессии





Ridge-регрессия

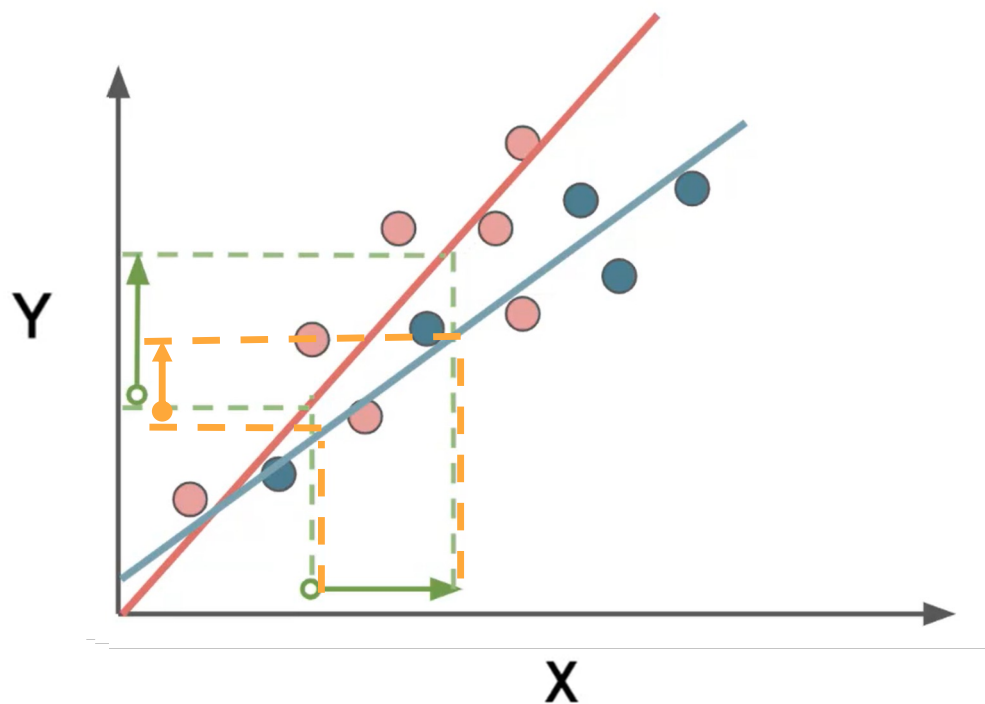
- Сравним это с результатом ridge-регрессии





Ridge-регрессия

- Сравним это с результатом ridge-регрессии
- Штраф квадратов бетта штрафует большие значения



$$\lambda \sum_{j=1}^p \beta_j^2$$



Ridge-регрессия

- А параметр альфа? Как сильно должны мы наказывать большие значения бета?

$$\lambda \sum_{j=1}^p \beta_j^2$$



Ridge-регрессия

- А параметр альфа? Как сильно должны мы наказывать большие значения бета?
- Мы можем с помощью кросс-валидации попробовать разные значения альфа, и выбрать оптимальный вариант!

$$\lambda \sum_{j=1}^p \beta_j^2$$



Ridge-регрессия

L2-регуляризация



Ridge-регрессия

- Важное замечание:
 - в Sklearn параметр лямбда называется alpha:

$$\text{Error} = \sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 + \boxed{\lambda} \sum_{j=1}^p \beta_j^2$$



Ridge-регрессия

- Важное замечание:
 - Для метрик кросс-валидации Sklearn использует объект “scorer”
 - Все объекты “scorer” следуют такому принципу, что бОльшие результирующие значения лучше, чем меньшие



Ridge-регрессия

- Важное замечание:
 - Например, более высокая точность лучше
 - Но вот для метрики RMSE это хуже!
 - Поэтому для RMSE в Sklearn в объекте “scorer” используется отрицательное значение метрики



Ridge-регрессия

- Важное замечание:
 - Например, более высокая точность лучше
 - Но вот для метрики RMSE это хуже!
 - Поэтому для RMSE в Sklearn в объекте “scorer” используется отрицательное значение метрики
 - Это позволяет получить универсальный подход для разных метрик. Это же относится и к параметру alpha.



Lasso-регрессия

L1-регуляризация



Lasso-регрессия

- L1-регуляризация добавляет штраф пропорционально абсолютному значению коэффициентов

$$\sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^p |\beta_j| = \text{RSS} + \lambda \sum_{j=1}^p |\beta_j|$$



Lasso-регрессия

- L1-регуляризация добавляет штраф пропорционально абсолютному значению коэффициентов
 - Ограничивает размер коэффициентов
 - Может приводить к разреженным моделям, в которых некоторые коэффициенты становятся нулевыми



Lasso-регрессия

- LASSO может приравнивать некоторые коэффициенты в точности нулю, когда параметр альфа достаточно большой.
- По сути, LASSO выбирает некоторые переменные
- Модели, которые получаются в результате работы LASSO, обычно легче интерпретировать.



Elastic Net

L1 и L2 регуляризация



Elastic Net

- Мы видели работу Ridge и Lasso регрессии.
- Ранее обсуждали, что Lasso может обнулять значения, но мы не изучили как следует, почему и как это происходит
- Это станет понятнее, когда мы рассмотрим Elastic Net, который объединяет в себе Ridge и Lasso!



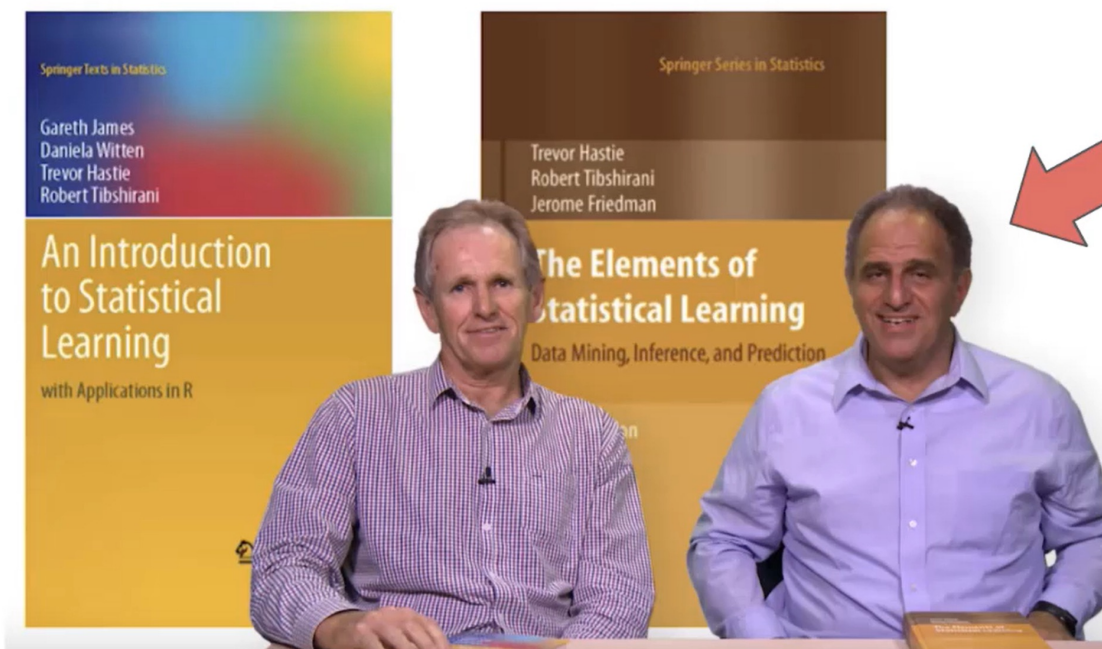
Elastic Net

- Рассмотрим Lasso подробнее.
- Lasso изначально появился в геофизике в 1986 году
- Позднее этот метод был снова открыт и популяризирован в 1996 году Робертом Тибширани, который предложил термин Lasso.



Elastic Net

- Роберт Тибширани является одним из авторов книги ISLR – Introduction to Statistical Learning





Elastic Net

- Перепишем условие задачи, указав сумму s :

$$\text{минимизируем по } \beta \left\{ \sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 \right\}$$

и

$$\text{минимизируем по } \beta \left\{ \sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 \right\}$$

$$\text{при условии } \sum_{j=1}^p |\beta_j| \leq s$$

$$\text{при условии } \sum_{j=1}^p \beta_j^2 \leq s$$



Elastic Net

- Начнём с такого эксперимента:
 - Рассмотрим простое уравнение $\hat{y} = \beta_1 x_1 + \beta_2 x_2$
 - Мы знаем, что регуляризацию можно рассматривать как некоторое условие для RSS



Elastic Net

- Начнём с такого эксперимента:
 - Рассмотрим простое уравнение $\hat{y} = \beta_1 x_1 + \beta_2 x_2$
 - L1 ограничивает сумму абсолютных значений $\sum |\beta|$
 - L2 ограничивает сумму квадратов значений $\sum \beta^2$



Elastic Net

- Начнём с такого эксперимента:
 - Рассмотрим простое уравнение $\hat{y} = \beta_1 x_1 + \beta_2 x_2$
 - L1 ограничивает сумму абсолютных значений $\sum |\beta|$
 - L2 ограничивает сумму квадратов значений $\sum \beta^2$
 - Ограничиваем штрафы некоторой суммой s



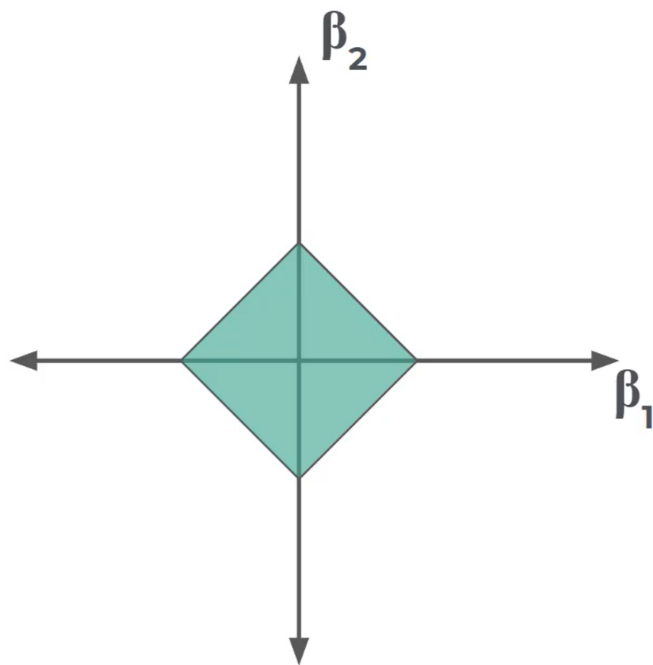
Elastic Net

- В случае только двух признаков: $\hat{y} = \beta_1 x_1 + \beta_2 x_2$
- Штраф Lasso-регрессии: $|\beta_1| + |\beta_2| \leq s$
- Штраф Ridge-регрессии: $\beta_1^2 + \beta_2^2 \leq s$



Elastic Net

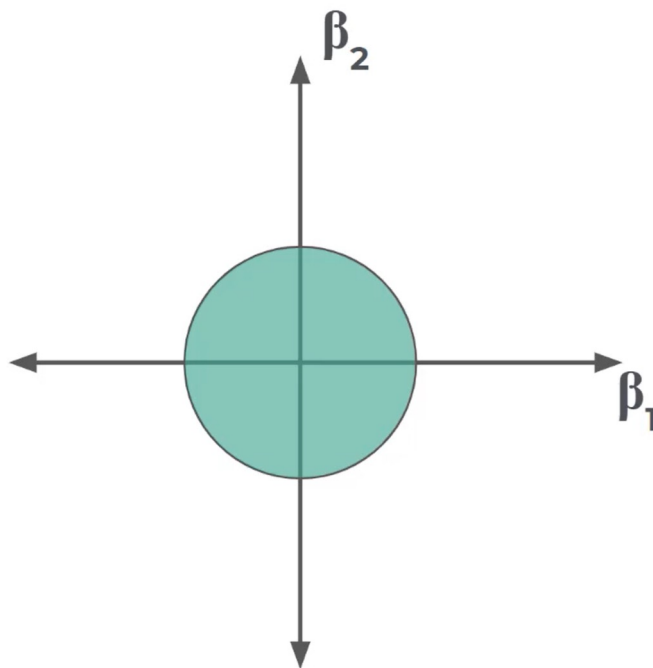
- Нарисуем график Lasso-регрессии: $|\beta_1| + |\beta_2| \leq s$





Elastic Net

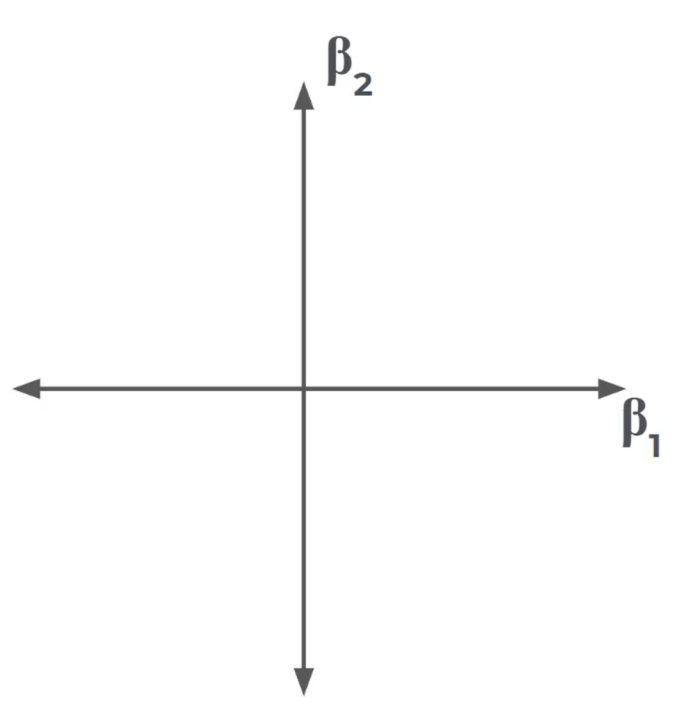
- Нарисуем график Ridge-регрессии: $\beta_1^2 + \beta_2^2 \leq s$





Elastic Net

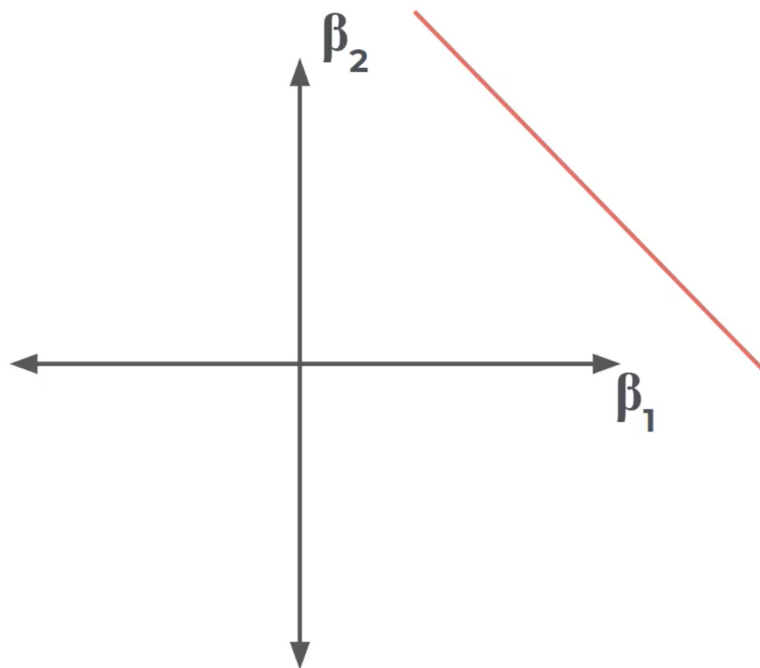
- Как будет выглядеть сумма остатков RSS?





Elastic Net

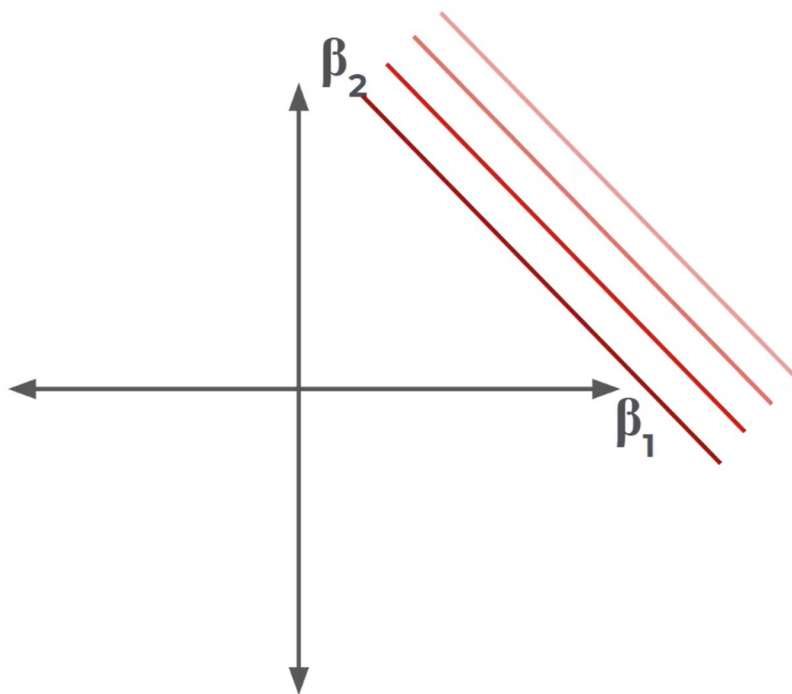
- Как будет выглядеть сумма остатков RSS?





Elastic Net

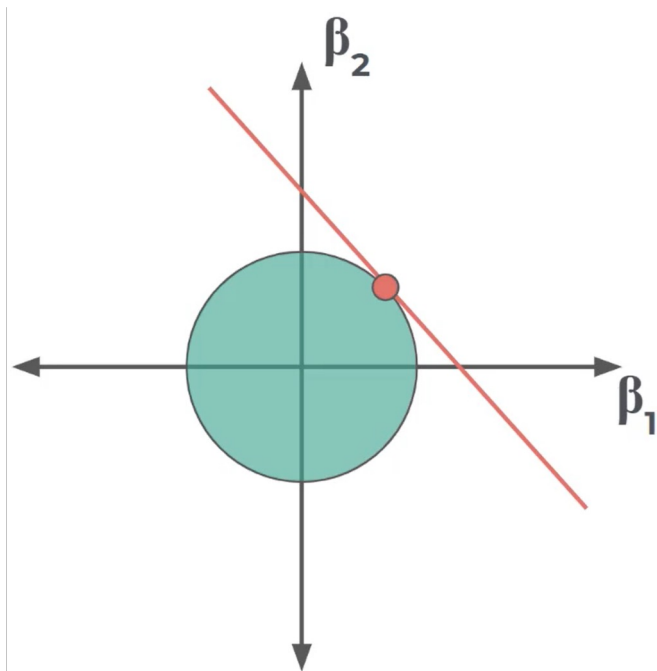
- Как будет выглядеть сумма остатков RSS?





Elastic Net

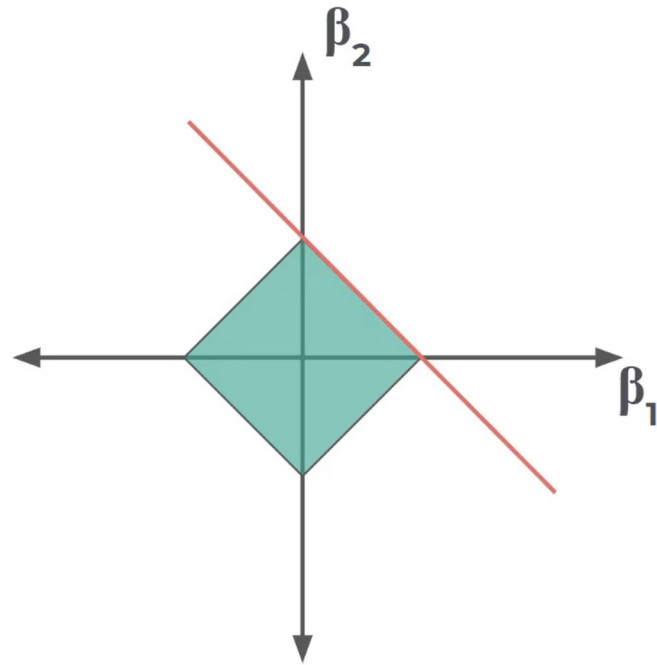
- Штраф для Ridge: $\beta_1^2 + \beta_2^2 \leq s$





Elastic Net

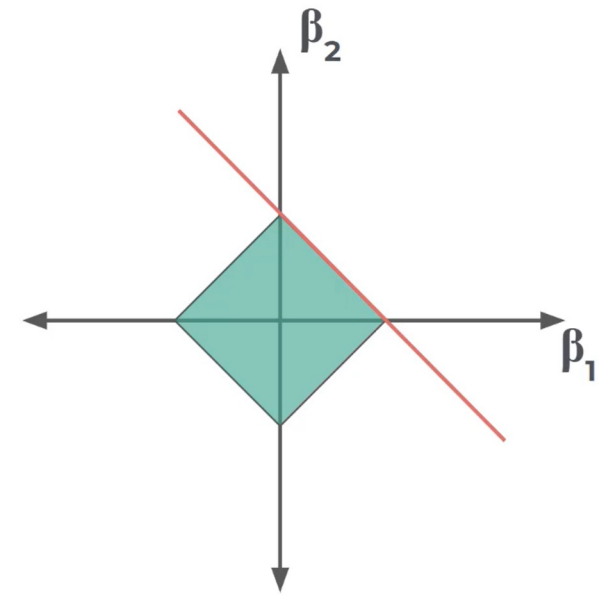
- Штраф для Lasso: $|\beta_1| + |\beta_2| \leq s$





Elastic Net

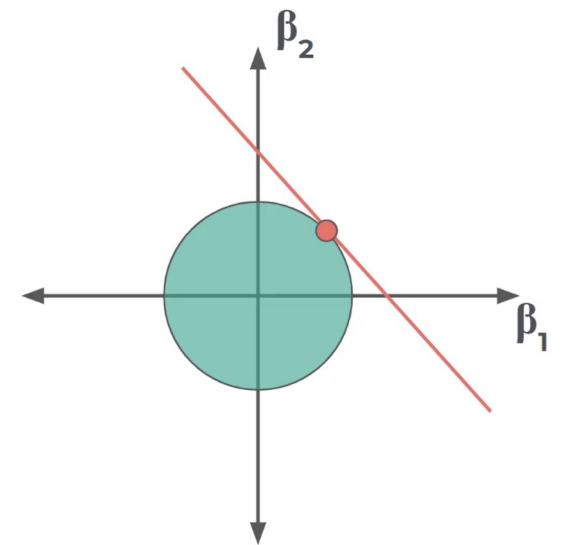
- Lasso: выпуклый объект, располагающийся по касательной относительно границы, скорее всего пройдёт через угол гиперкуба, в котором некоторые компоненты бетта в точности равны нулю.





Elastic Net

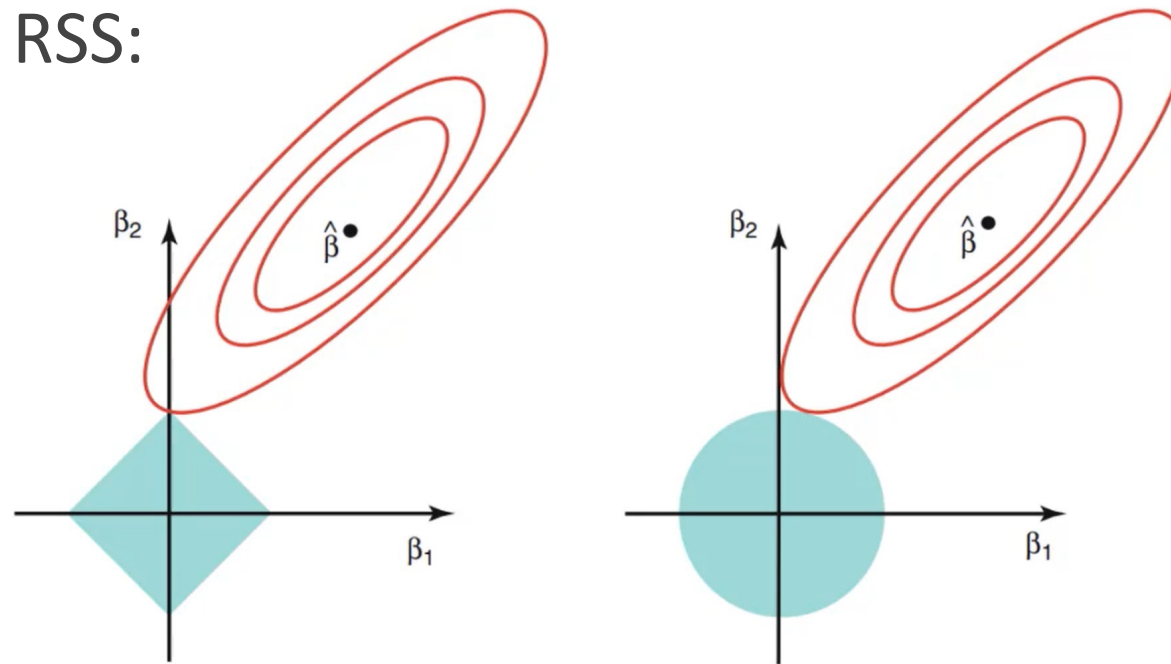
- Ridge: в случае n -сферы, точки границы, где некоторые компоненты бетта равны нулю, не отличаются от других точек, которых может коснуться выпуклый объект. Вероятность коснуться таких точек не выше, чем для любых других точек, где все компоненты бетта не равны нулю.





Elastic Net

- Поэтому в Lasso более вероятны нули, чем в Ridge.
- Это можно удобно изобразить с помощью контурных графиков для RSS:





Elastic Net

- Elastic Net старается улучшить и L1, и L2 регуляризацию, объединяя их вместе:

$$\text{Error} = \sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 + \lambda_1 \sum_{j=1}^p \beta_j^2 + \lambda_2 \sum_{j=1}^p |\beta_j|$$



Elastic Net

- Elastic Net старается улучшить и L1, и L2 регуляризацию, объединяя их вместе:

$$\text{Error} = \sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 + \lambda_1 \sum_{j=1}^p \beta_j^2 + \lambda_2 \sum_{j=1}^p |\beta_j|$$

Обратите внимание на два
разных параметра лямбда



Elastic Net

- Альтернативный вариант – рассматривать параметр “соотношения” между L1 и L2:

$$\sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 + \lambda \left(\frac{1-\alpha}{2} \sum_{j=1}^p \beta_j^2 + \alpha \sum_{j=1}^p |\beta_j| \right)$$



Elastic Net

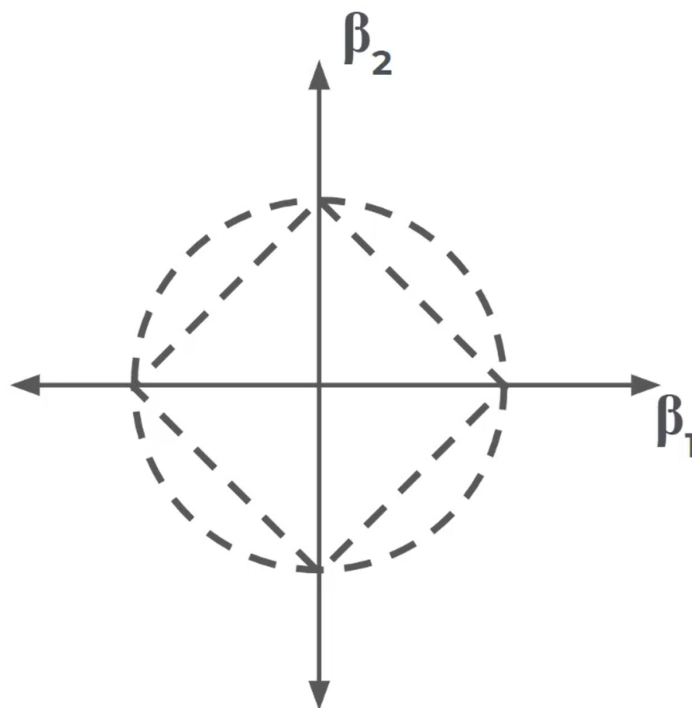
- Ещё можно записать в упрощённой нотации:

$$\hat{\beta} \equiv \underset{\beta}{\operatorname{argmin}}(\|y - X\beta\|^2 + \lambda_2 \|\beta\|^2 + \lambda_1 \|\beta\|_1)$$



Elastic Net

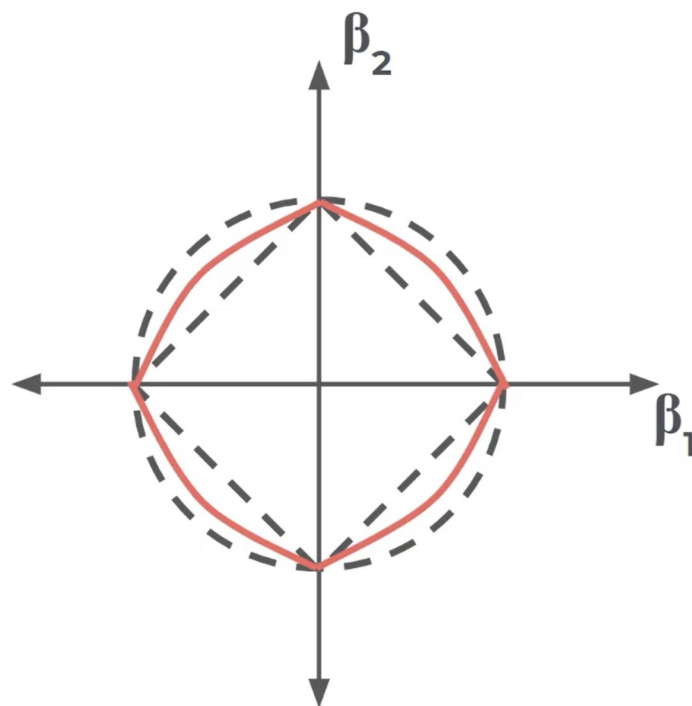
- Область штрафов для Elastic Net:





Elastic Net

- Область штрафов для Elastic Net:





Elastic Net

- Давайте посмотрим на Elastic Net в Python и Scikit-learn!

$$\sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 + \lambda \left(\frac{1-\alpha}{2} \sum_{j=1}^p \beta_j^2 + \alpha \sum_{j=1}^p |\beta_j| \right)$$



Проект по линейной регрессии

Данные



Проект - данные

- Большинство наборов данных требуют очистки, анализ и построение признаков перед их применением в машинном обучении
- Давайте посмотрим на данные для проекта по линейной регрессии
- В следующих разделах мы рассмотрим подготовку данных к машинному обучению!