Искусственный Интеллект

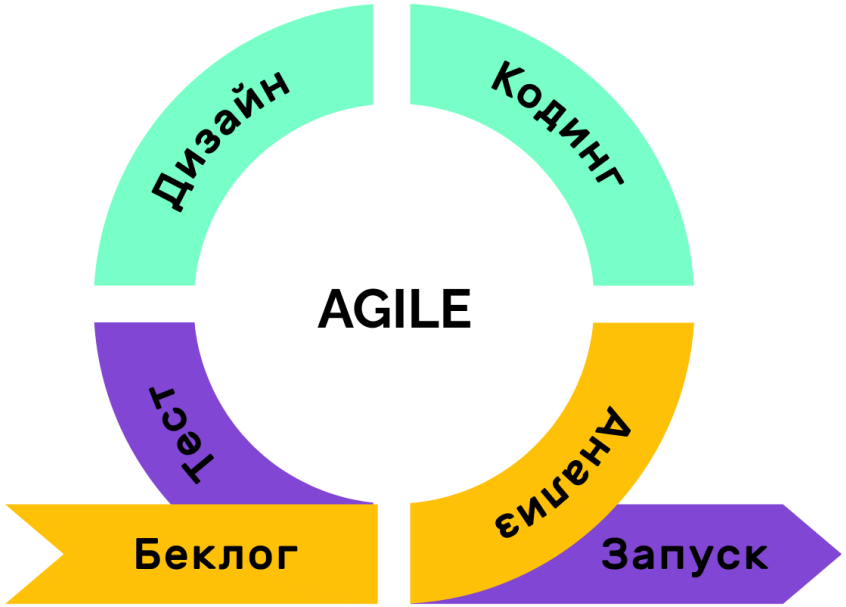
Лекция 2: Цикл работы с моделью ИИ.
Линейная регрессия. Функция ошибки. Градиентный спуск.

Мартынюк Полина Антоновна

telegram: @PAMartynyuk
email: pa-martynyuk@yandex.ru

Особенности работы с проектами, включающими модели машинного обучения

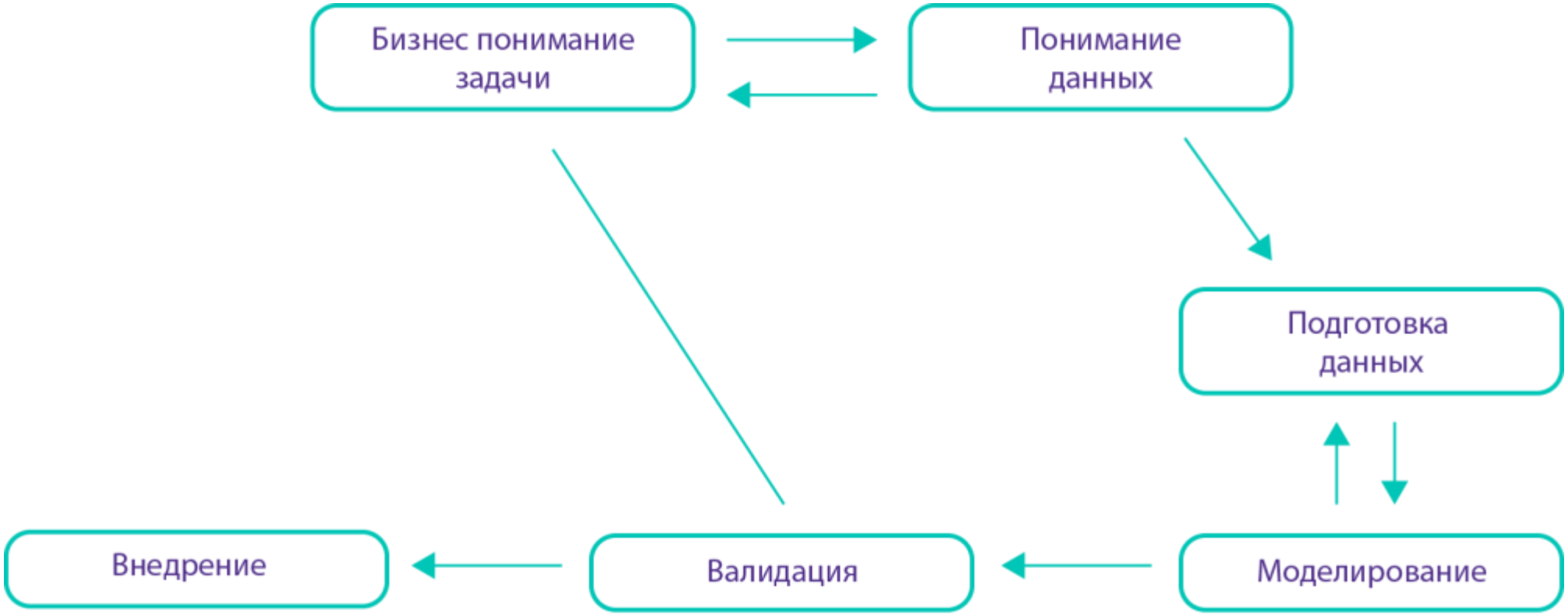
В разработке ПО существуют стандарты по управлению проектами



Эти стандарты плохо подходят для проектов по анализу данных, поэтому в анализе данных появились свои.

Стандарт CRISP-DM

- **CRISP-DM** — Cross-Industry Standard Process for Data Mining (Межотраслевой стандартный процесс интеллектуального анализа данных).
- Стандарт был создан в 1999 г., 42% компаний все еще используют именно его



Стандарт CRISP-DM

1. Бизнес-понимание задачи

На этом этапе мы **формулируем задачу** и делаем **план проекта**.

Проект стартует с того, что у бизнеса есть какая-то **проблема**, и он хочет решить ее при помощи анализа данных.

Например, банк хочет выдавать кредиты более надежным заемщикам. Но **бизнес не умеет формулировать задачу с точки зрения машинного обучения**.

Главная задача на этапе бизнес-понимания задачи — перевести проблему с языка бизнеса на язык машинного обучения.

Стандарт CRISP-DM

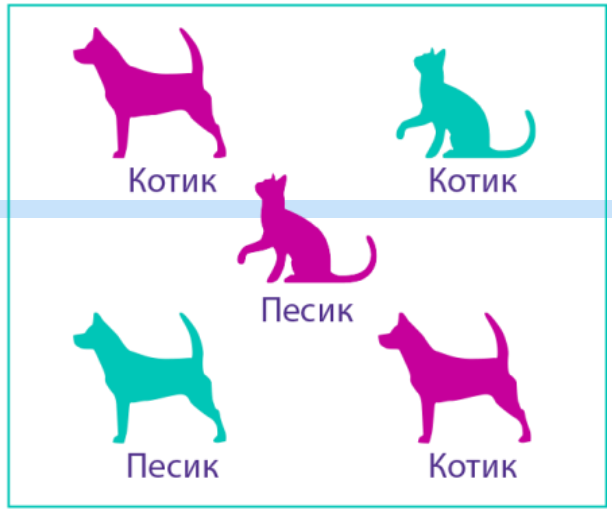
1. Бизнес-понимание задачи

Также нам надо выбрать **метрику**, по которой мы поймем, решили мы проблему бизнеса или нет.

Бизнес использует свои метрики: объем продаж, прибыль, средняя доходность с заемщика и т. д.

В команде аналитиков данных используются **метрики машинного обучения**, например точность.

Надо понять, какую бизнес-метрику необходимо улучшить, подобрать максимально похожую метрику машинного обучения и по ней потом оценивать модель.



Верное предсказание

Котик

Ошибочное предсказание

Котик

Точность модели — доля правильных ответов

$$\text{Точность} = \frac{\text{2 верно предсказанных животных}}{\text{5 животных в выборке}}$$

Точность = 0,4

Стандарт CRISP-DM

2. Понимание данных

На этом этапе аналитик данных определяет, **какие данные нужны для решения задачи**. Половина успеха проектов по машинному обучению — качественные данные и правильная разметка.

На этом этапе мы решаем:

- Какие данные нужны
- Какие данные у нас есть внутри компании
- Какие данные придется закупать во внешних источниках
- Нуждаются ли данные в разметке

Процесс поиска данных внутри компании и получения доступа к ним иногда может занимать 2 месяца и более (проблемы наличия данных, персональные данные, безопасность и т.д.).

Стандарт CRISP-DM

3. Подготовка данных

На этом этапе решается целый комплекс задач. Мы должны:

- понять, какие типы данных у нас есть

Данные могут быть представлены в виде таблицы, необработанных текстов, изображений или графов. С каждым из этих типов данных хорошо работают разные модели машинного обучения.

- проверить качество данных

Данные могут содержать ошибки, пропуски, аномальные значения. Необходимо хорошо изучить данные и решить, как мы будем чистить и трансформировать данные.

- почистить данные

Тут мы заполняем пропуски, исправляем ошибки, где это возможно, удаляем аномальные объекты.

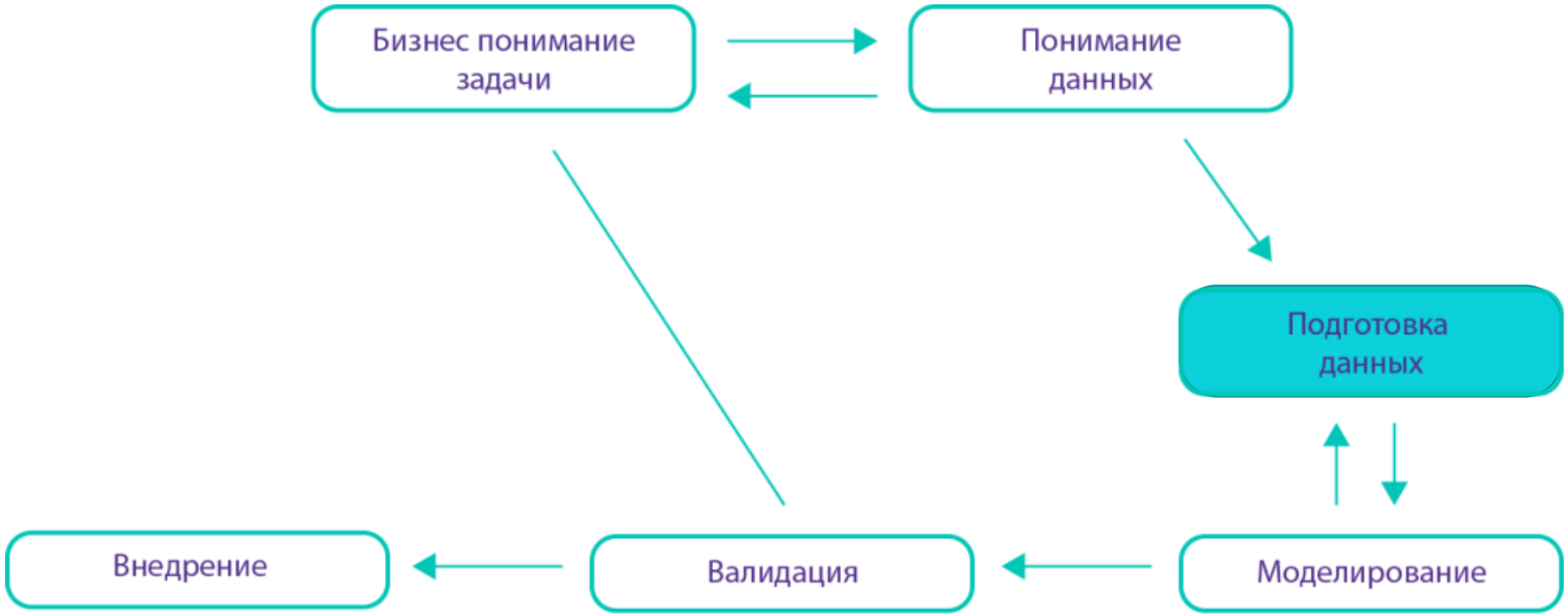
- преобразовать данные

Не все модели умеют работать с любыми типами признаков, например с категориальными переменными. Нужно преобразовать такие переменные в формат, понятный модели.

- отобрать релевантные признаки

Не всегда больше признаков — лучше. Иногда бывает, что в выборке более 3 тысяч признаков, но не все из них полезны. Нам надо оставить только самые актуальные. Обычно оставляют 20-30 самых подходящих. Однако этот этап нужен не всегда. Например, глубокие нейронные сети могут сами извлекать признаки из неструктурированных данных: текстов, картинок, транзакций.

Стандарт CRISP-DM



> Сбор данных, очистка и подготовка данных занимают больше 50 % времени аналитика данных, в то время как моделирование может занимать только 20 %.

Стандарт CRISP-DM

4. Моделирование

На этом этапе нам надо:

- **выбрать** модель машинного обучения
- **обучить** ее
- **проверить качество** модели по выбранным метрикам
- **подобрать параметры** модели.

Обычно тестируют сначала самую простую модель, ее называют **бейзлайн** (baseline). Потом проверяют более сложные модели, которые могут дать более высокое качество.

Некоторые модели требуют иного набора данных. Поэтому часто необходимо повторить фазу подготовки данных.

Этап моделирования заканчивается тем, что аналитики данных сравнивают несколько моделей, выбирают лучшую и отдают на проверку бизнесу.

Стандарт CRISP-DM

5. Валидация модели

На этом этапе бизнес решает, подходит ли ему построенная модель.

Существует несколько **причин не принять модель**:

- модель не решает проблему, **плохо работает на бизнес-метрике**, хотя на метрике машинного обучения работала хорошо
- модель **дорого или сложно поддерживать**
- модель **плохо работает на некоторой группе объектов**

Основная цель этапа — проверить, существуют ли какие-то другие **нюансы**, связанные с моделью, **которые мы не рассмотрели ранее**. Если все хорошо, бизнес принимает решение о внедрении модели.

Если модель отклонена, мы возвращаемся на более ранние этапы. При этом то, на какой этап мы возвращаемся, зависит от того, чем именно заказчик недоволен.

Стандарт CRISP-DM

6. Внедрение

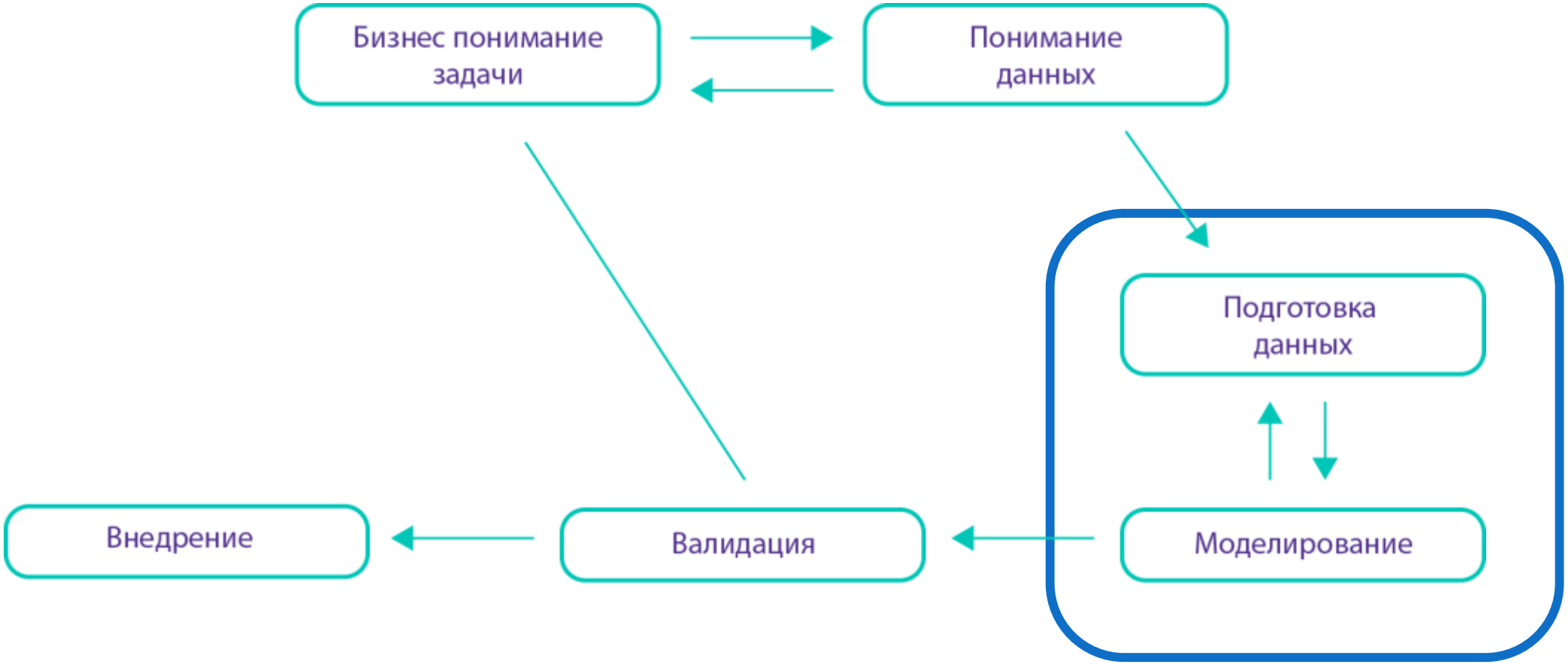
В самом конце нужно внедрить модель и поддерживать ее.

Внедрением модели обычно занимается бизнес, однако часто приходится помогать бизнесу на этапе внедрения.

Также модель со временем может **устаревать**, так как данные меняются. Периодически необходимо **переобучать модели на новых данных**.

Иногда цель проекта — это не построить модель, а улучшить понимание данных для принятия более эффективных решений. Тогда на этапе внедрения полученные знания структурируются и представляются в виде отчета, презентации и т. д.

Стандарт CRISP-DM



Работа с данными и моделью

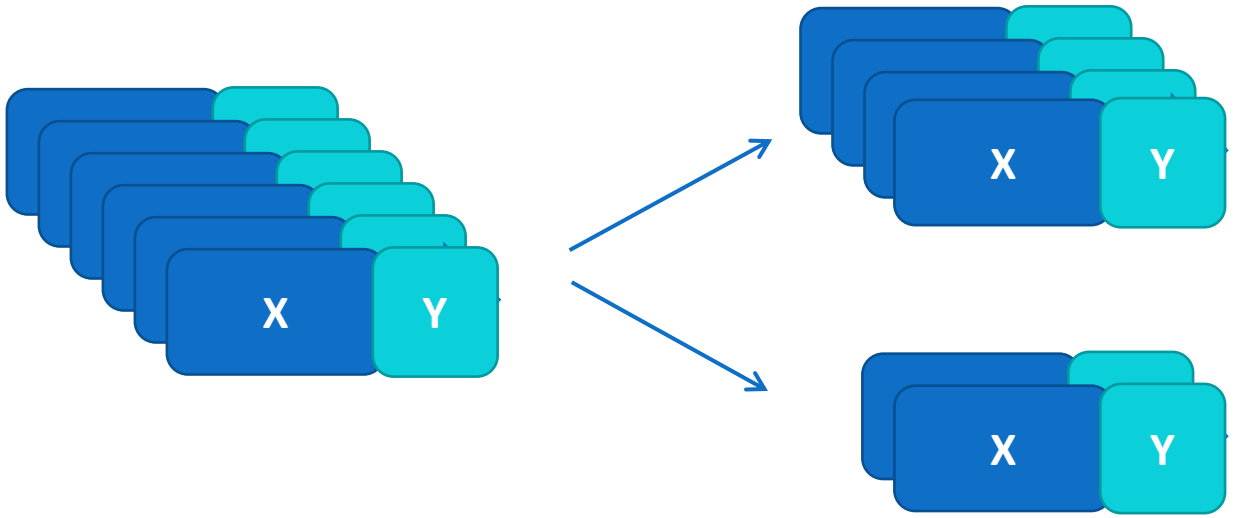
- Что мы имеем?

$$x \xrightarrow[y \approx f(x)]{f(x)} y$$

Площадь квартиры (X1, feature)	Количество комнат (X2, feature)	Цена квартиры (Y, label, target)
73	2	5.5
150	4	15.8
34	1	4.2
101	3	10.9

Работа с данными и моделью

- Необходимо добиться того, чтобы модель, обученная на тех данных, которые у нас есть, была способна корректно обрабатывать аналогичные данные, которые она “не видела” при обучении.
- Решение – выделить часть данных, которые не будут использоваться при обучении, но будут – при оценке модели.



Тренировочные данные *train*

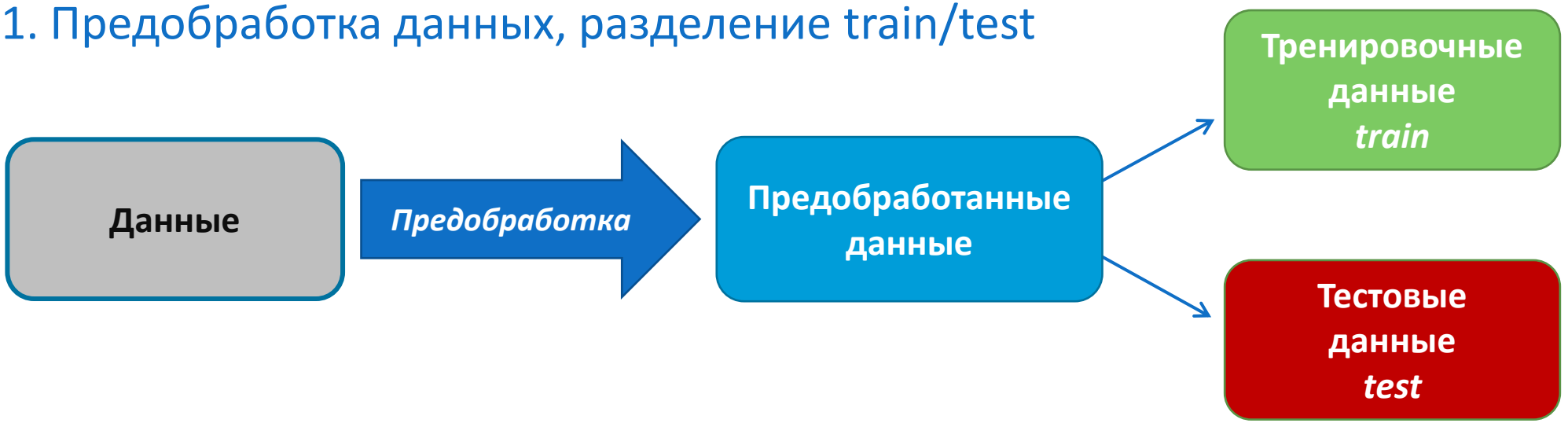
80% : 20%

Тестовые данные *test*

70% : 30%

Работа с данными и моделью

1. Предобработка данных, разделение train/test



2. Обучение модели

Работа с данными и моделью

3. Оценка модели

Работа с данными и моделью

Условные обозначения

Формат данных

Площадь квартиры (X1, feature)	Количество комнат (X2, feature)	Цена квартиры (Y, label, target)
73	2	5.5
150	4	15.8
34	1	4.2
101	3	10.9

Экземпляр размеченных данных

. Наблюдение (значения признаков)

. Метка/целевое значение

Количество экземпляров наблюдений, *n*

Площадь квартиры (X1, feature)	Количество комнат (X2, feature)	Цена квартиры (Y, label, target)
73	2	5.5
150	4	15.8
34	1	4.2
101	3	10.9

Количество параметров/фич, *p*

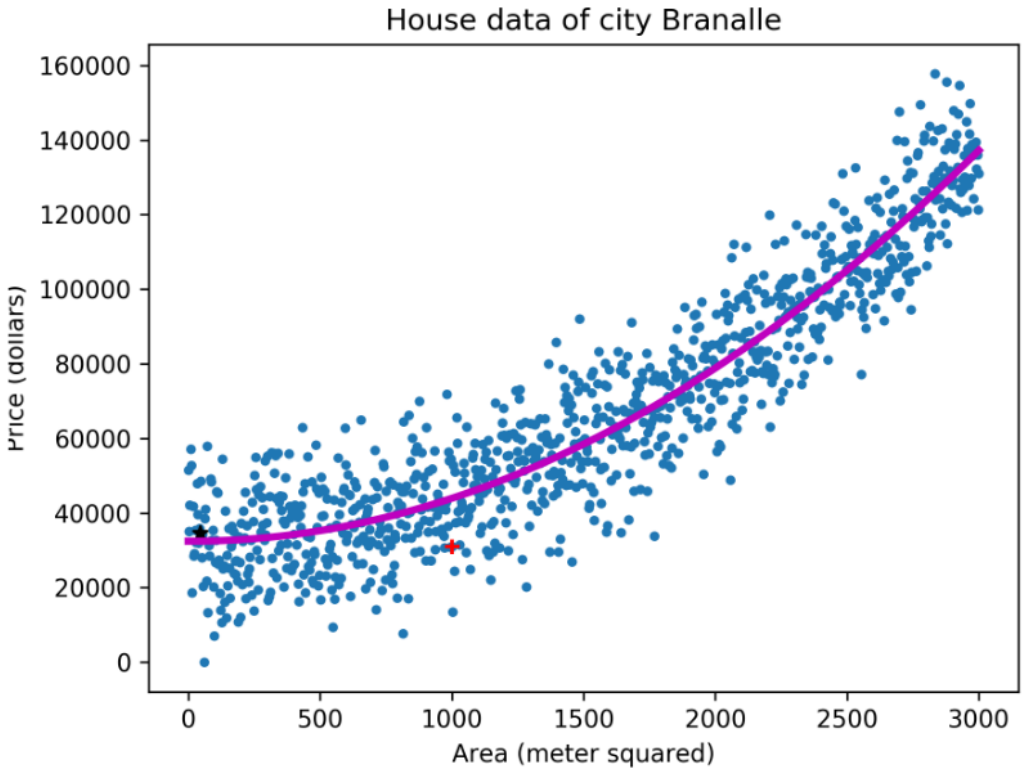
Регрессия

Регрессия – предсказание числового значения по входным данным.

Целевое значение Y представляет собой числовое значение, которое модель пытается предсказать.

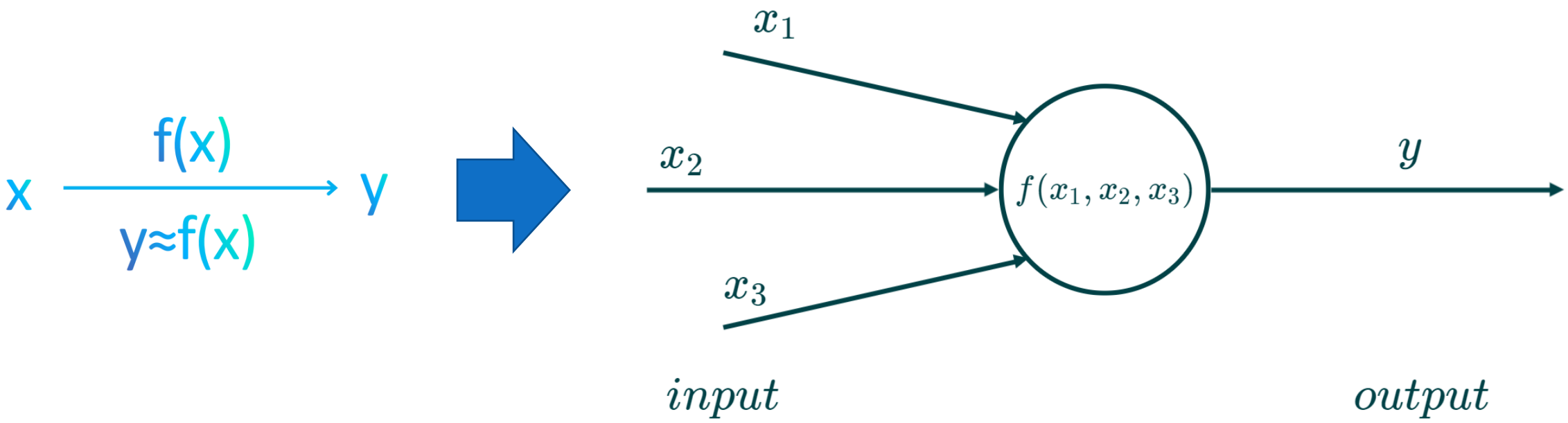
Например, при прогнозировании цены дома целевое значение может быть числом, представляющим стоимость.

Площадь квартиры (X1, feature)	Количество комнат (X2, feature)	Цена квартиры (Y, label, target)
73	2	5.5
150	4	15.8
34	1	4.2
101	3	10.9



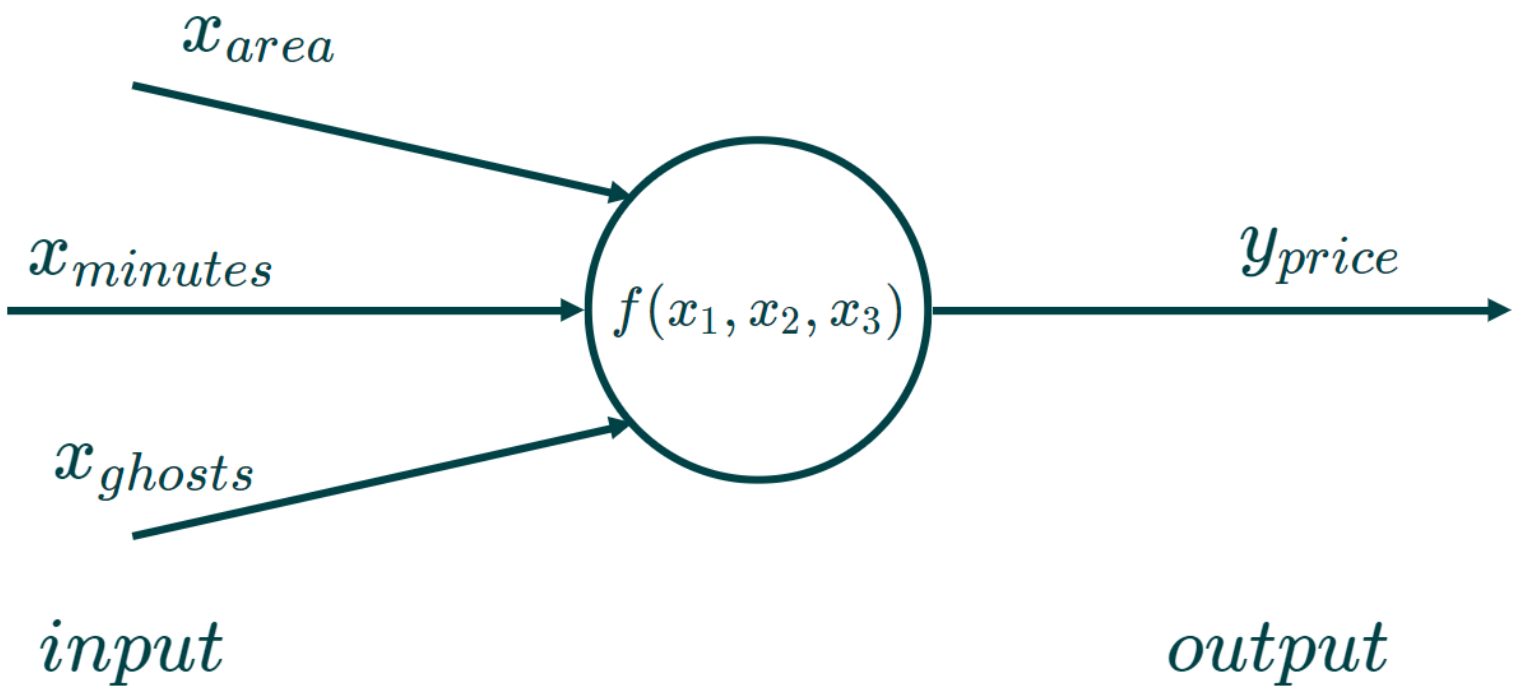
Регрессия

- Заданы три признака: x_1, x_2, x_3
- По ним требуется прогнозировать целевое значение y



Регрессия

- Пусть задача – предсказание цены на квартиру
- Признаки – площадь, время пути до метро и число призраков



- Как смоделировать функцию $f(x_1, x_2, x_3)$?

Регрессия

- Интуитивный подход – всё сложить!

$$x_1 + x_2 + x_3 = y$$

- Проблема?

Степень значимости признаков – разная!

- Решение – веса для признаков!

$$w_1 * x_1 + w_2 * x_2 + w_3 * x_3 = y$$

Регрессия

- Каждому признаку – свой вес:

$$w_{area} * x_{area} + w_{minutes} * x_{minutes} + w_{ghosts} * x_{ghosts} = y$$

- Чем важнее признак. тем больше его вес:

$$\uparrow w_{area} * x_{area} + w_{minutes} * x_{minutes} + w_{ghosts} * x_{ghosts} = y \uparrow$$

- Если признак негативно влияет на результат, его вес отрицателен:

$$w_{area} * x_{area} + \downarrow w_{minutes} * x_{minutes} + w_{ghosts} * x_{ghosts} = y \downarrow$$

- Если признак почти не влияет на результат, его вес близок к нулю:

$$w_{area} * x_{area} + w_{minutes} * x_{minutes} + w_{ghosts} * x_{ghosts} = y$$

Регрессия

- Что не учтено?

$$w_1 * x_1 + w_2 * x_2 + w_3 * x_3 = y$$

- Если все **признаки** равны **нулю**, **целевое значение** тоже **приравнивается к нулю**, что верно далеко не всегда с точки зрения задачи
- Решение – добавить дополнительный параметр смещения:

$$w_1 * x_1 + w_2 * x_2 + w_3 * x_3 + b = y$$

$$b - bias$$

Регрессия

- Варианты записи смещения:

1) $w_1 * x_1 + w_2 * x_2 + w_3 * x_3 + b = y$

$b - bias$

2) $w_0 * x_0 + w_1 * x_1 + w_2 * x_2 + w_3 * x_3 = y$

$w_0 - bias,\ x_0 = 1$

Регрессия

- Функция модели в общем виде:

1) Смещение – *b*

$$y = \sum_{i=1}^{p}(x_i w_i) + b$$

2) Смещение – w_0

$$y = \sum_{i=0}^{p}(x_i w_i)$$

$$x_0 = 1$$

Регрессия

Формальная постановка задачи

- Задана выборка значений признаков:

$$X_n : \{x_1, x_2, \ldots, x_n \mid x_i \in R^p\}$$

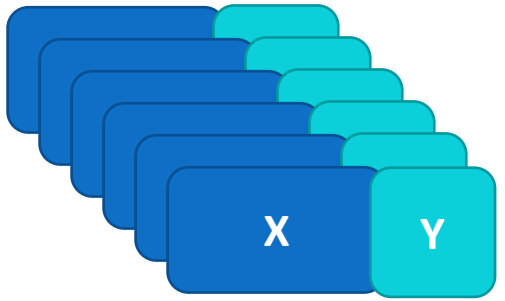
Здесь n - количество элементов в выборке входных данных, p - размерность признакового пространства.

- Задана выборка соответствующих значений целевой переменной:

$$Y_n : \{y_1, y_2, \ldots, y_n \mid y_i \in R\}$$

- Получаем множество исходных данных:

$$D : \{(x, y)_i\}, \; i = 1 \ldots n$$



Регрессия

Формальная постановка задачи

- Задано параметрическое семейство функций *f(w,x)* зависящее от параметров *W* и от входных признаков *X*:

$$f(w,x) = x_0 w_0 + x_1 w_1 + \ldots + x_p w_p$$

- Нужно построить модель, предсказывающую по x_i значение $\hat{y}_i$, наиболее близкое к истинному значению y_i :

$$\hat{y}_i = f(w, x_i)$$

$$|\hat{y}_i - y_i| \to 0$$

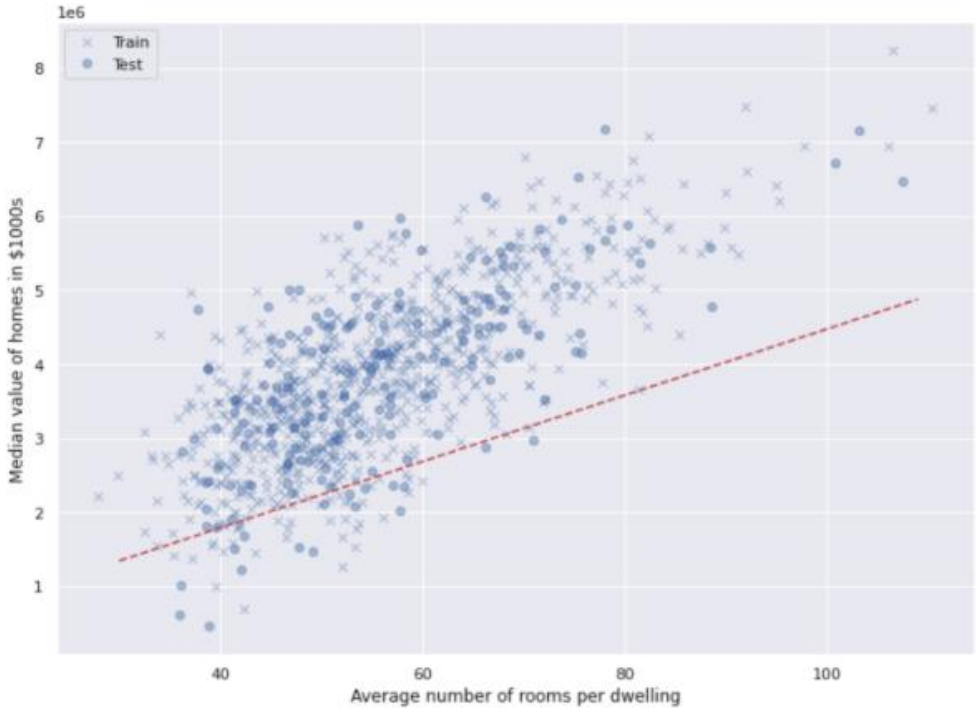
- **Цель обучения – найти такие веса модели $\hat{W}$, при которых разница между истинными и предсказанными значениями минимальна**

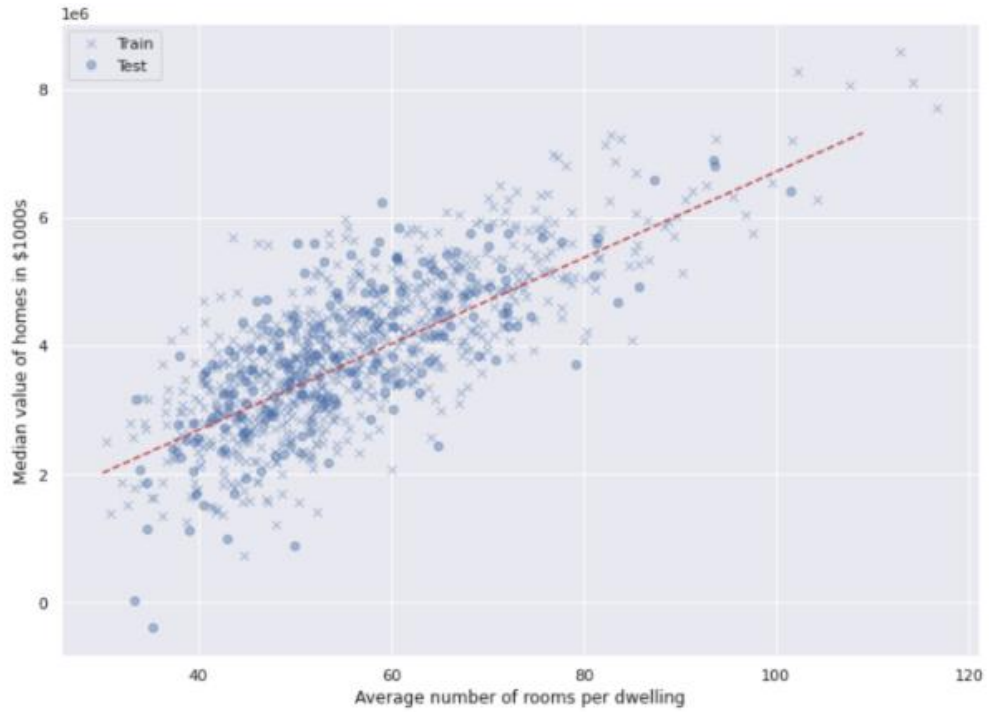
Регрессия: обучение модели

- Возьмем простейший случай с единственным признаком:

$$\hat{y} = wx + b$$

- Определим случайно две модели с разными значениями весов

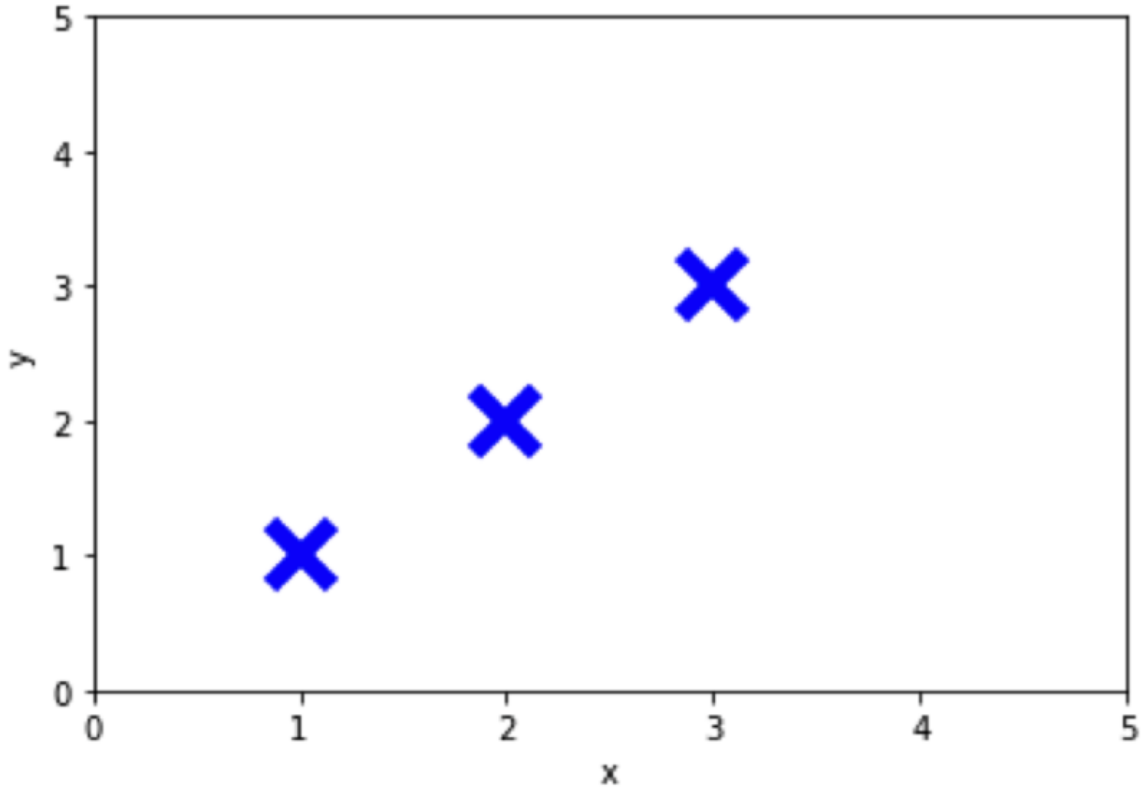




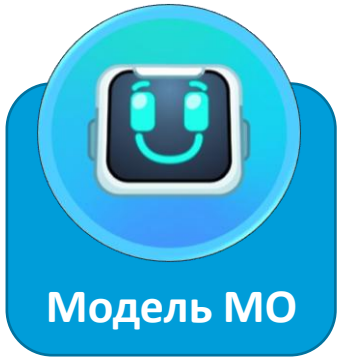
- Как понять, какая модель лучше?

Регрессия: обучение модели

- Имеем: **модель** (линейная регрессия для единственного признака *x*) и **датасет**



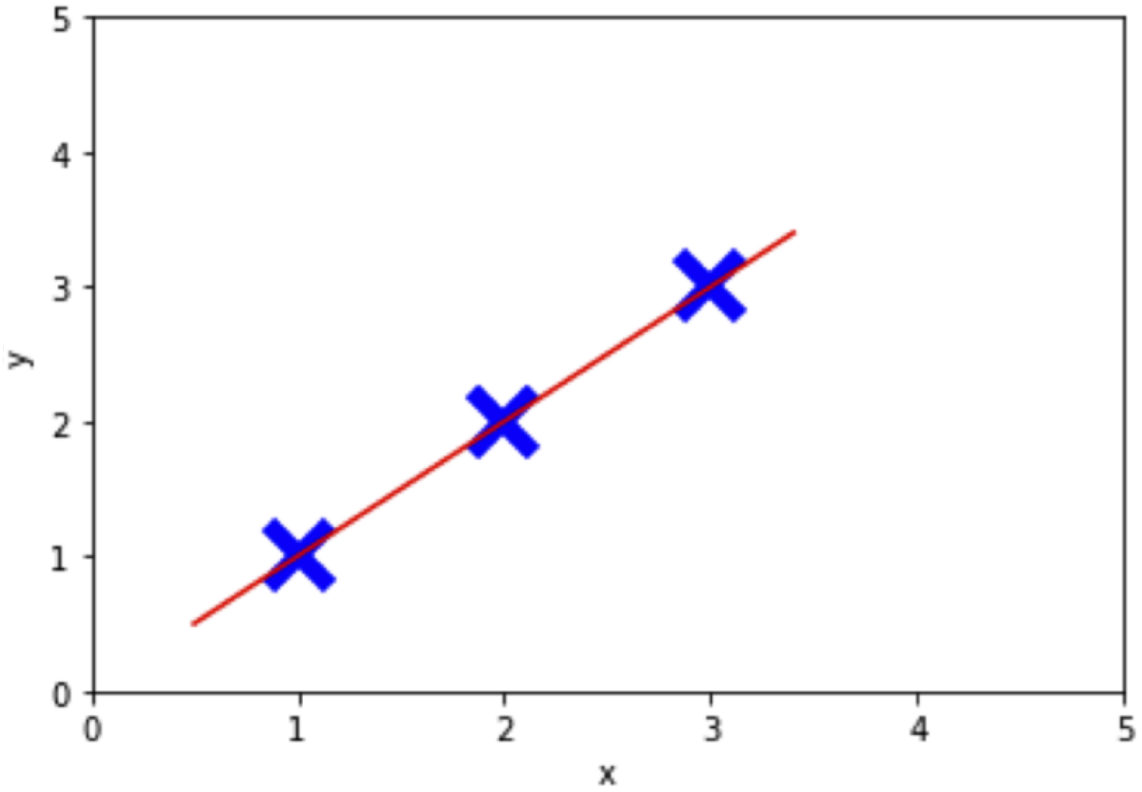
$$\hat{y} = wx + b$$



X	Y
1	1
2	2
3	3

Регрессия: обучение модели

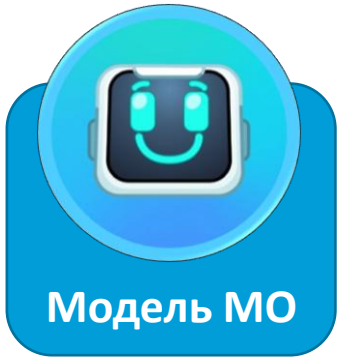
- Аналитически мы понимаем, что **идеальная модель** имеет параметры: w = 1, b = 0 (предсказанные и истинные значения совпали)



$$\hat{y} = wx + b$$

$$w = 1$$

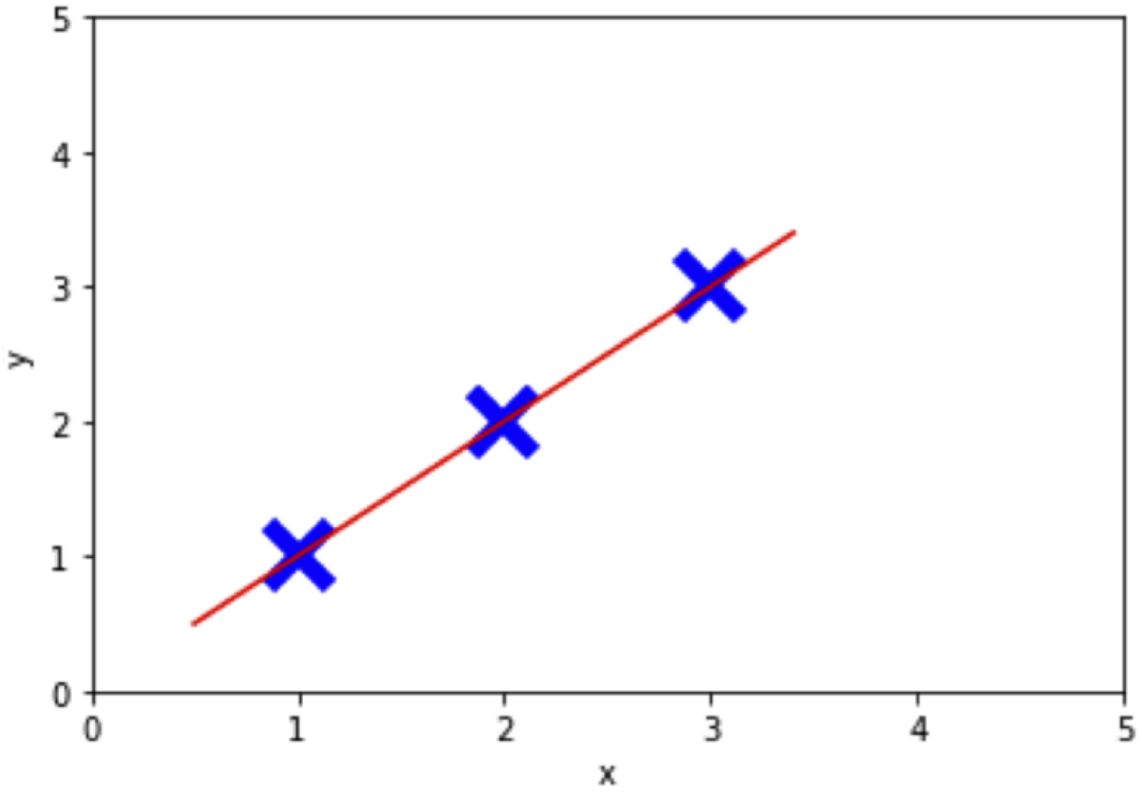
$$b = 0$$



X	Y	Y_pred
1	1	1
2	2	2
3	3	3

Регрессия: обучение модели

- Чтобы выразить **степень различия между истинными и предсказанными значениями математически**, введем функцию *L:*

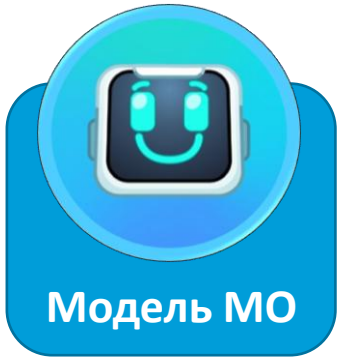


$$\hat{y} = wx + b$$

$$w = 1$$

$$b = 0$$

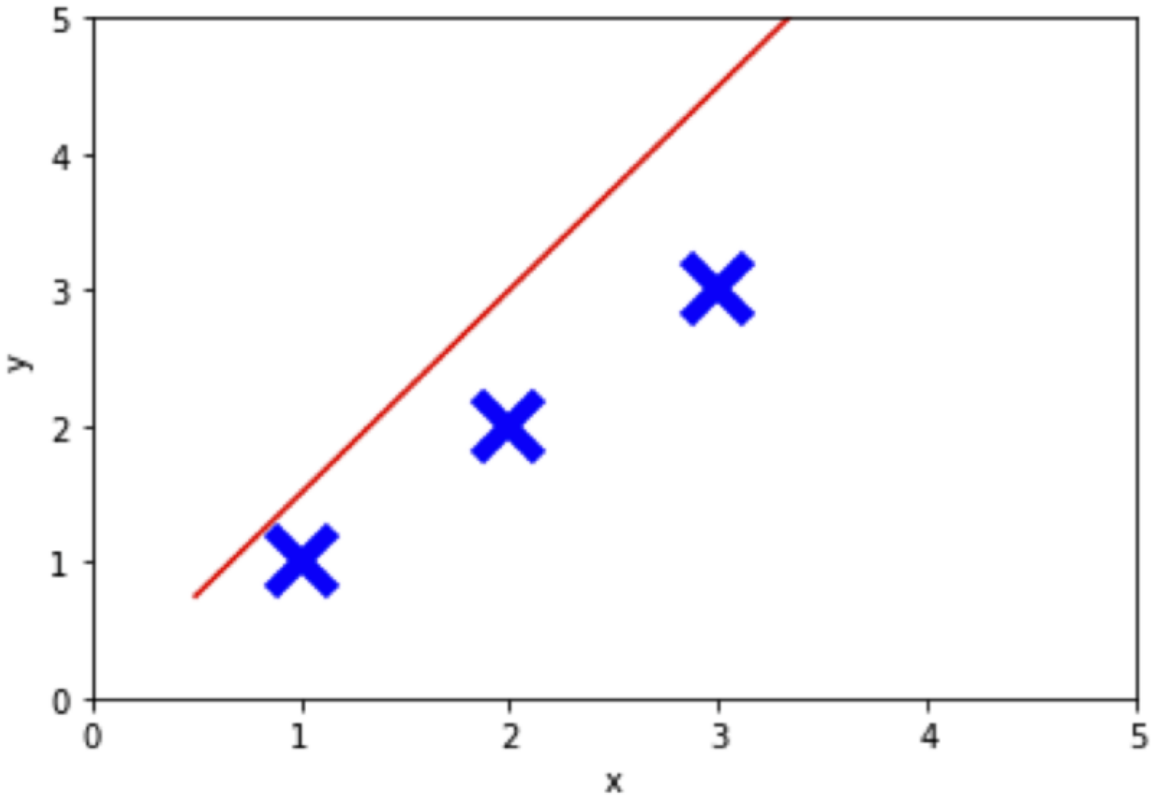
$$L(y, \hat{y}) = 0$$



X	Y	Y_pred
1	1	1
2	2	2
3	3	3

Регрессия: обучение модели

- Возьмем другую модель: w = 1.5, b = 0
- Как для этой модели оценить значение функции *L*?

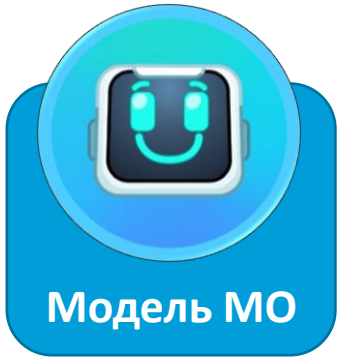


$$\hat{y} = wx + b$$

$$w = 1.5$$

$$b = 0$$

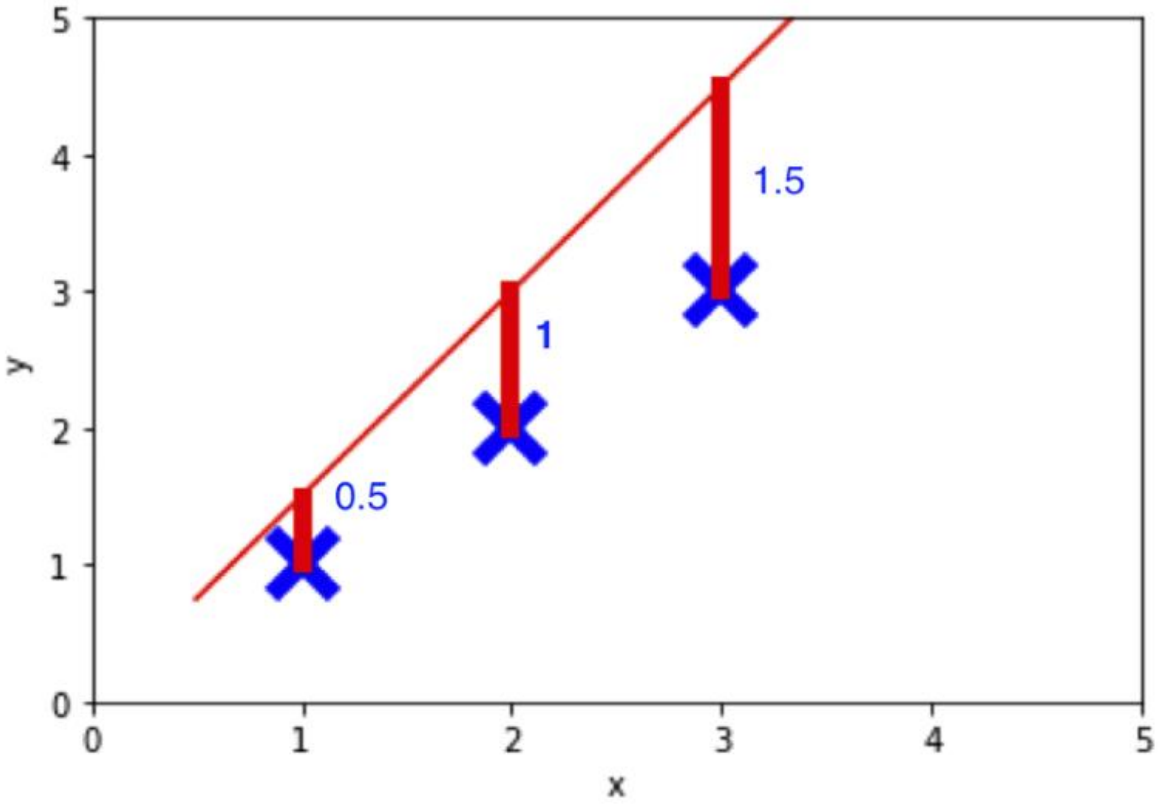
$$L(y, \hat{y}) = \ ?$$

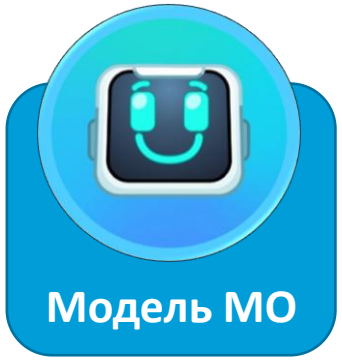


X	Y	Y_pred
1	1	1.5
2	2	3
3	3	4.5

Регрессия: обучение модели

- Самый простой вариант – вычислить **разницу между истинным и предсказанным значениями**
- Вычислили разницу для всех точек. Что дальше?





$$\hat{y} = wx + b$$

$$w = 1.5$$

$$b = 0$$

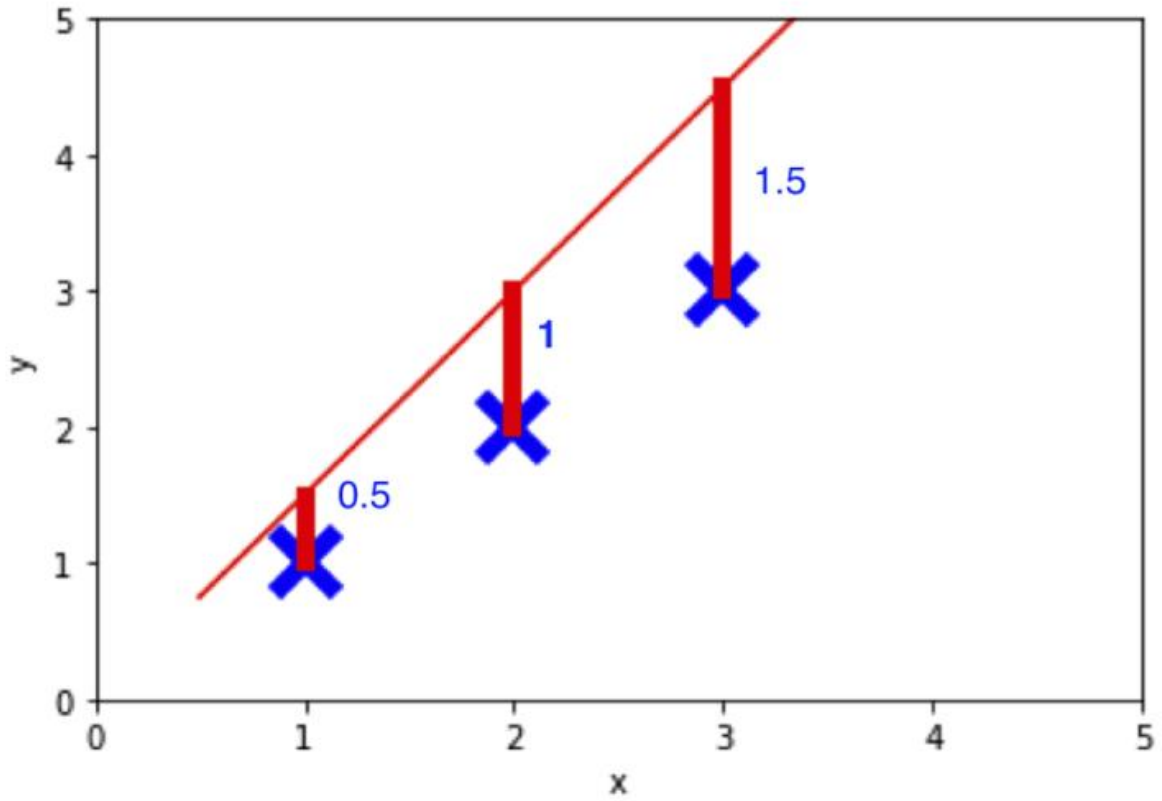
$$L(y, \hat{y}) = \hat{y} - y$$

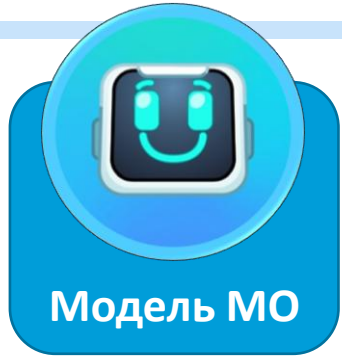
X	Y	Y_pred	ΔY
1	1	1.5	0.5
2	2	3	1
3	3	4.5	1.5

Регрессия: обучение модели

- Суммируем все вычисленные значения:

  0.5 + 1 + 1.5 = 3





$$\hat{y} = wx + b$$

$$w = 1.5$$

$$b = 0$$

$$L(y, \hat{y}) = \sum \hat{y} - y$$

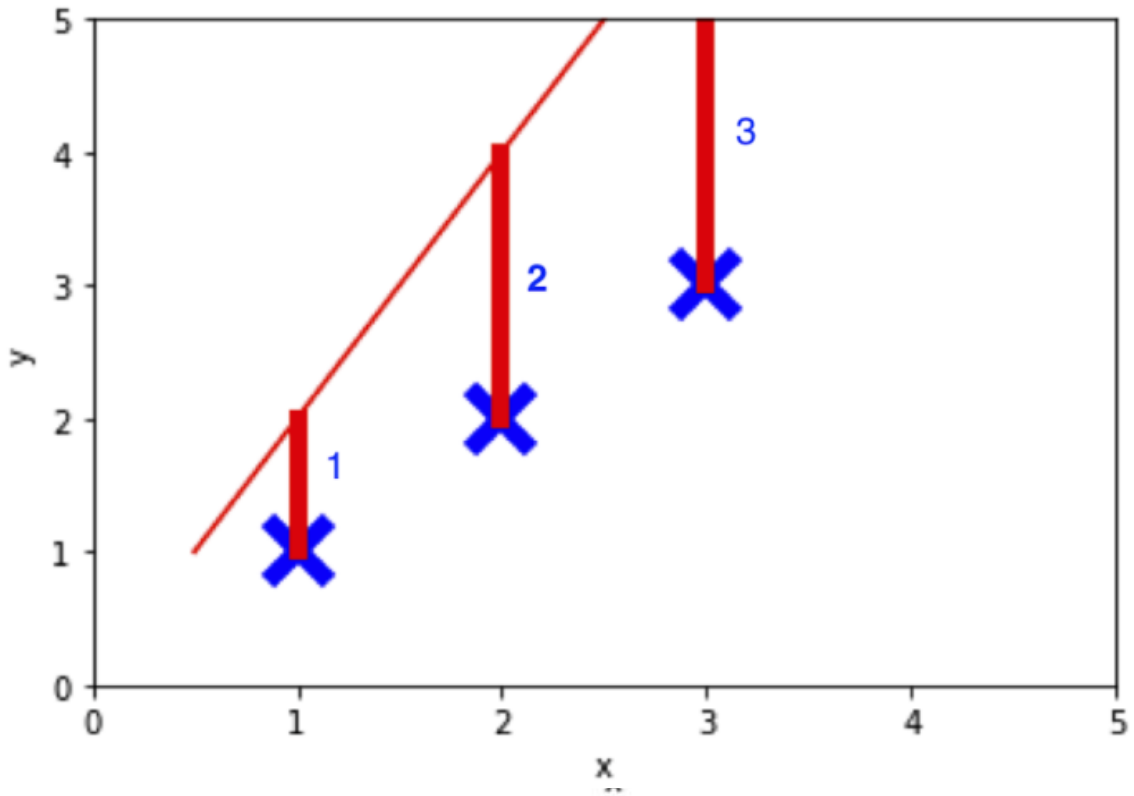
$$L(y, \hat{y}) = 3$$

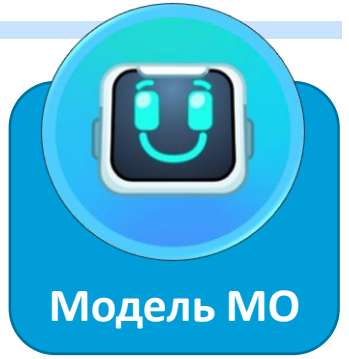
X	Y	Y_pred	ΔY
1	1	1.5	0.5
2	2	3	1
3	3	4.5	1.5

Регрессия: обучение модели

- Возьмем другую модель: w = 2, b = 0

  1 + 2 + 3 = 6





$$\hat{y} = wx + b$$

$$w = 2$$

$$b = 0$$

$$L(y, \hat{y}) = \sum \hat{y} - y$$

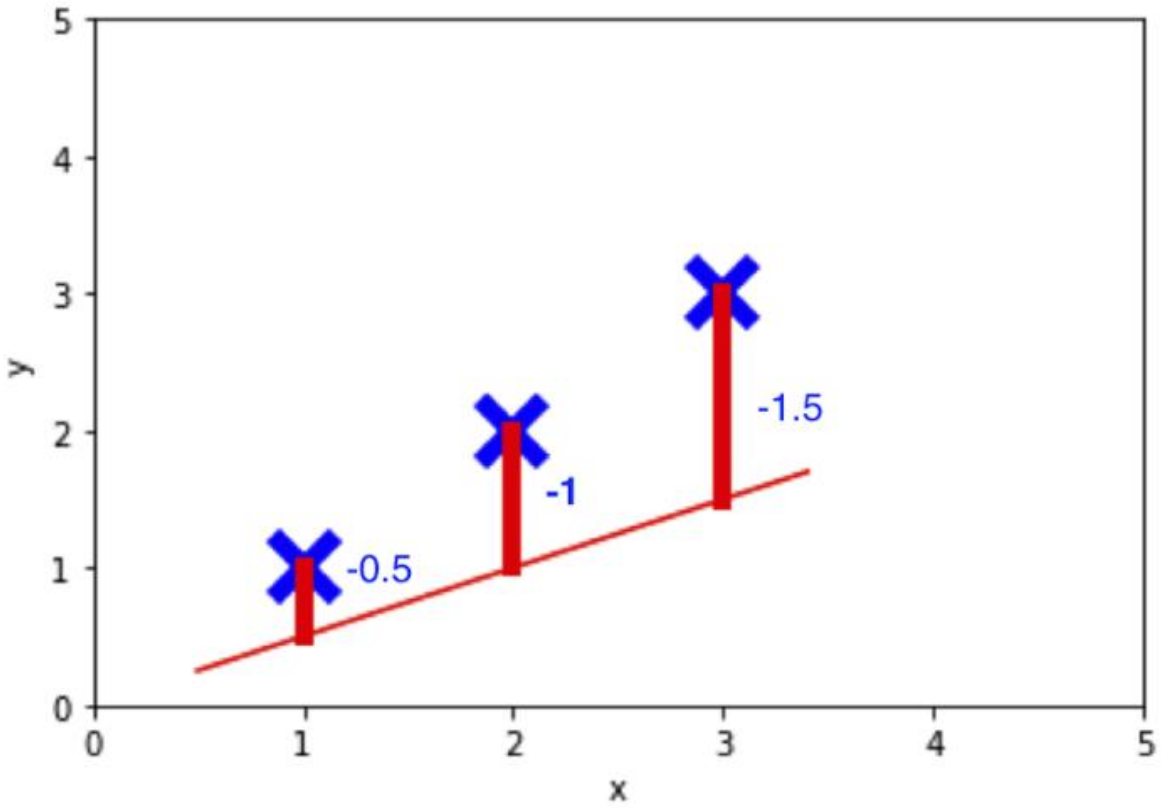
$$L(y, \hat{y}) = 6$$

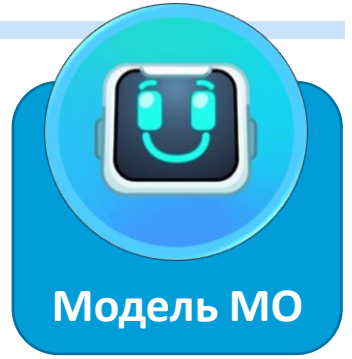
X	Y	Y_pred	ΔY
1	1	2	1
2	2	4	2
3	3	6	3

Регрессия: обучение модели

- Возьмем еще одну модель: w = 0.5, b = 0:

  (-0.5) + (-1) + (-1.5) = -3





$$\hat{y} = wx + b$$

$$w = 0.5$$

$$b = 0$$

$$L(y, \hat{y}) = \sum \hat{y} - y$$

$$L(y, \hat{y}) = -3$$

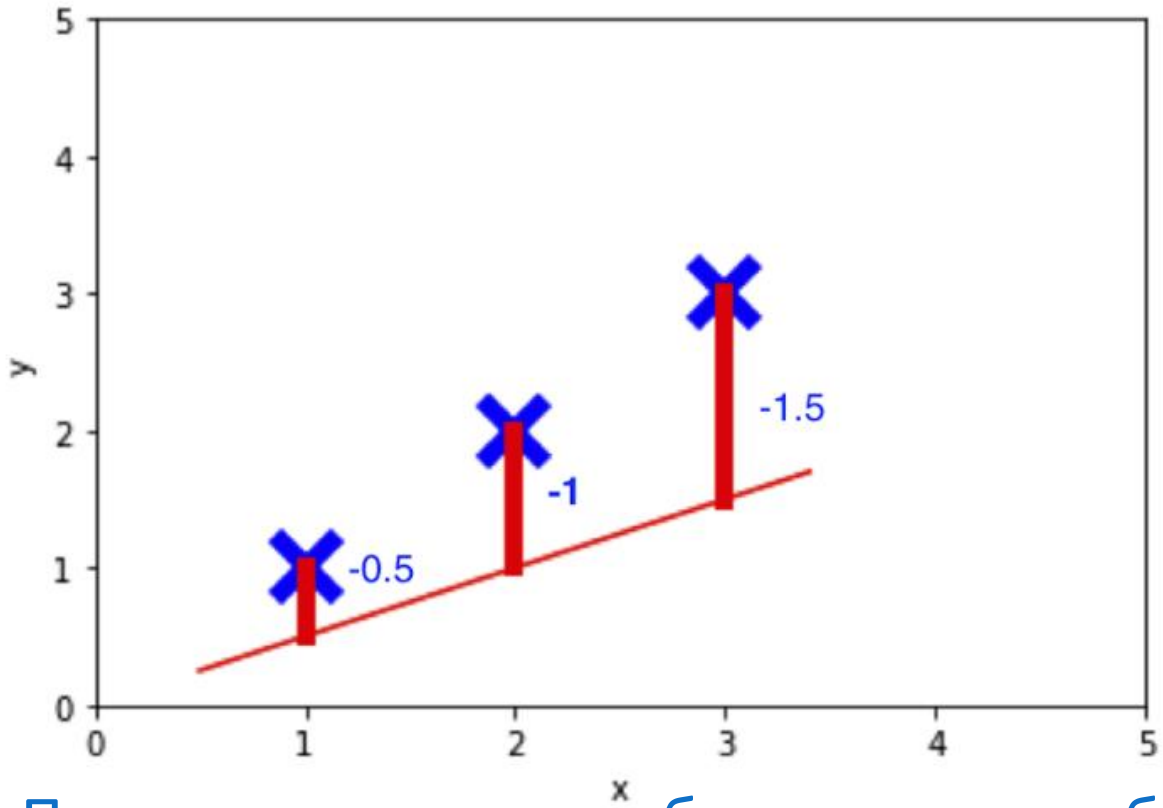
Нелогично

X	Y	Y_pred	ΔY
1	1	0.5	-0.5
2	2	1	-1
3	3	1.5	-1.5

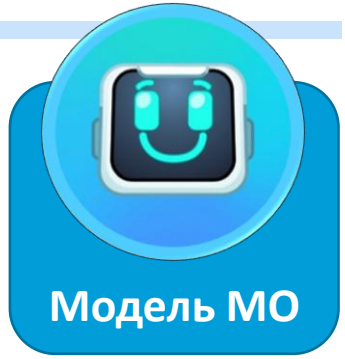
Регрессия: обучение модели

- Будем вычислять не просто разницу, а абсолютное значение разницы:

  |-0.5| + |-1| + |-1.5| = 3



- Получаем **сумму** ошибок, а хотелось бы получать **среднюю ошибку**



$$\hat{y} = wx + b$$

$$w = 0.5$$

$$b = 0$$

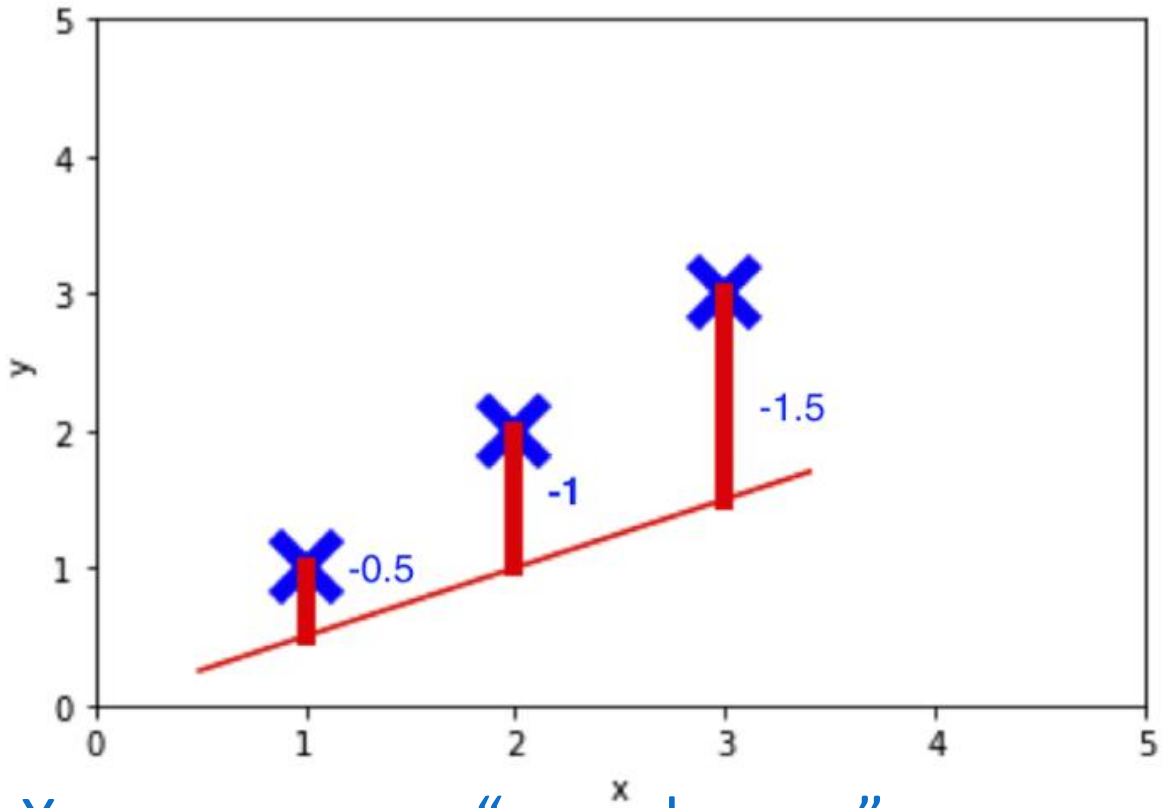
$$L(\hat{y}, y) = \sum |\hat{y} - y|$$

$$L(\hat{y}, y) = 3$$

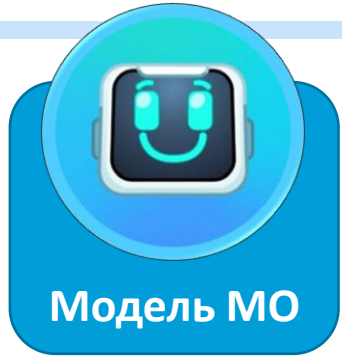
X	Y	Y_pred	ΔY
1	1	0.5	-0.5
2	2	1	-1
3	3	1.5	-1.5

Регрессия: обучение модели

- Для получения усредненной ошибки разделим на число наблюдений:
  (|-0.5| + |-1| + |-1.5|)/3 = 1



- Хотим сильнее “штрафовать” за большие ошибки



$$\hat{y} = wx + b$$

$$w = 0.5$$

$$b = 0$$

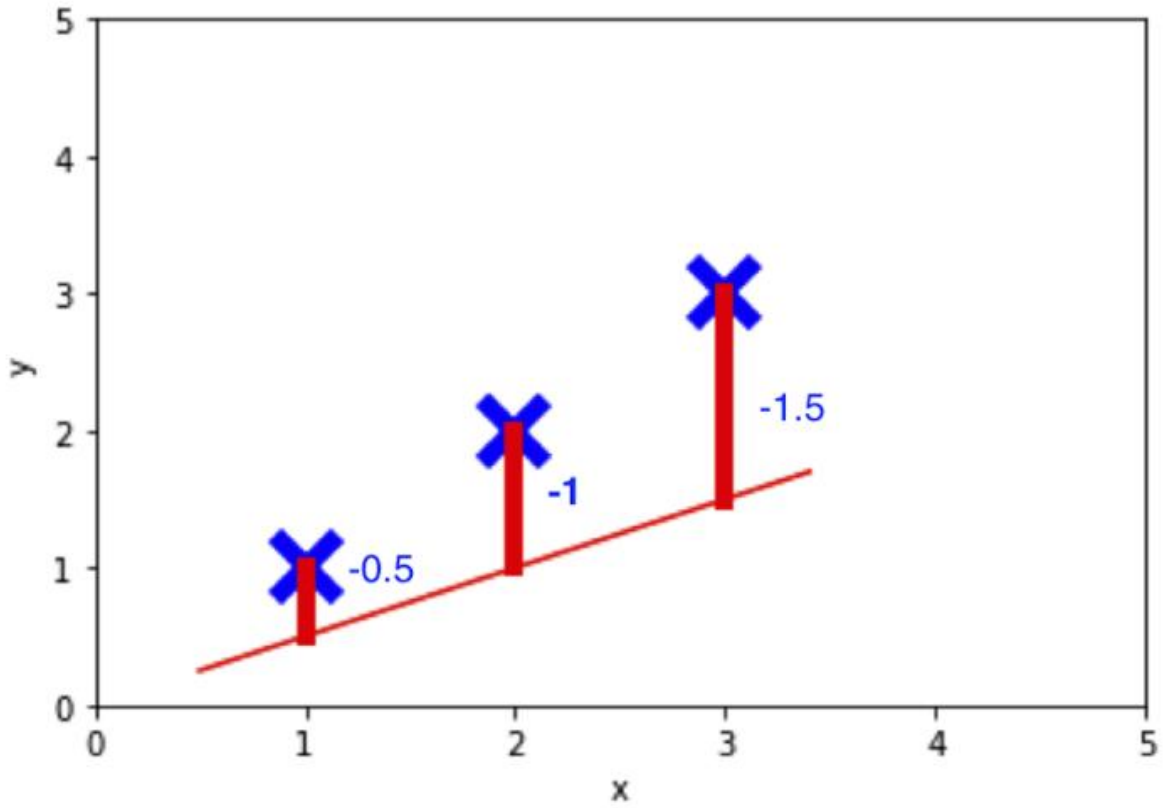
$$L(\hat{y}, y) = \frac{1}{n} \sum |\hat{y} - y|$$

$$L(\hat{y}, y) = 1$$

X	Y	Y_pred	ΔY
1	1	0.5	-0.5
2	2	1	-1
3	3	1.5	-1.5

Регрессия: обучение модели

- Будем вычислять квадрат разницы:
  ((-0.5)^2 + (-1)^2 + (-1.5)^2)/3 = 1.17



$$\hat{y} = wx + b$$

$$w = 0.5$$

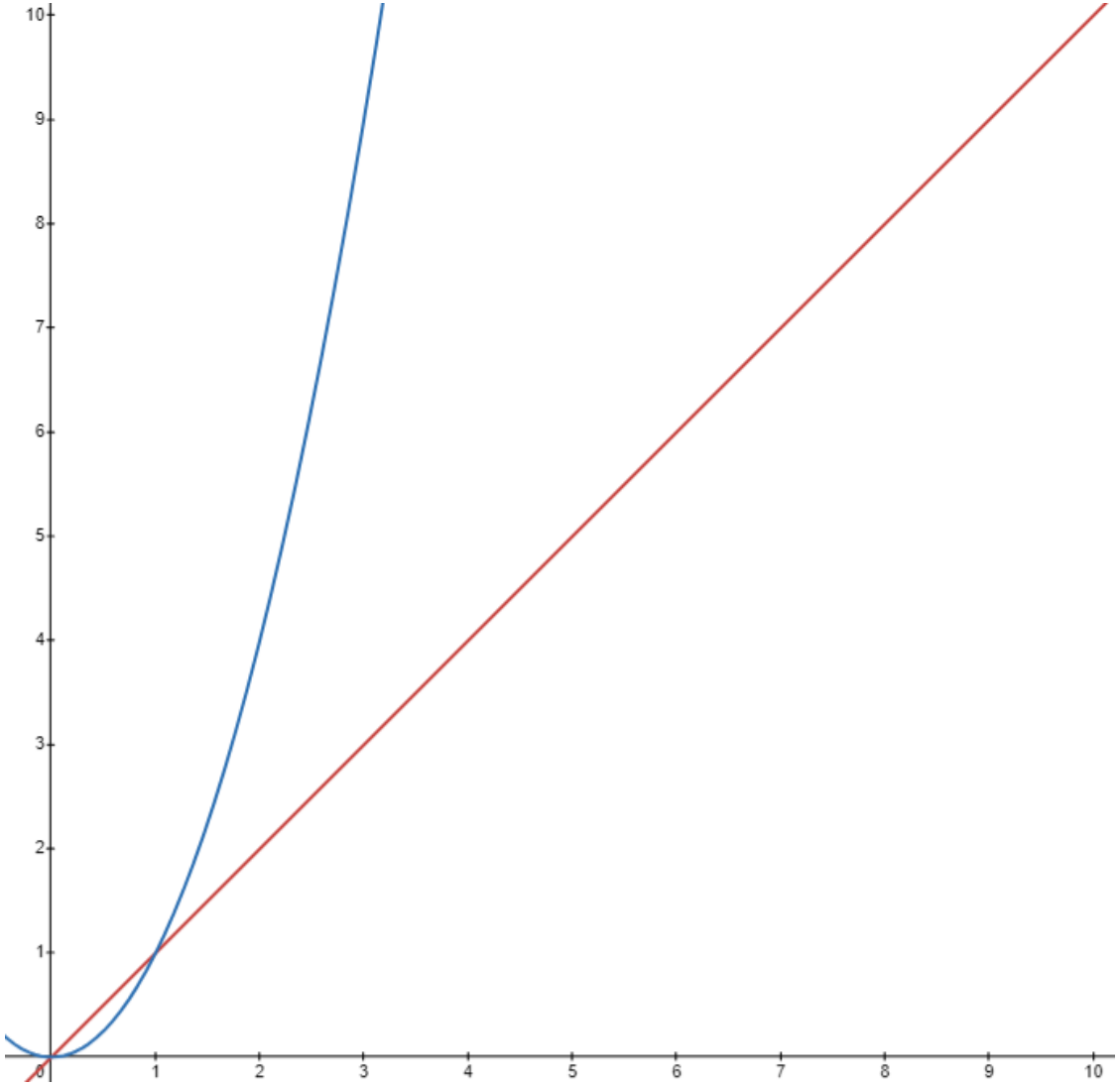
$$b = 0$$

$$L(\hat{y}, y) = \frac{1}{n} \sum (\hat{y} - y)^2$$

$$L(\hat{y}, y) = 1.17$$

X	Y	Y_pred	(ΔY)^2
1	1	0.5	0.25
2	2	1	1
3	3	1.5	2.25

Регрессия: обучение модели



- Оба варианта применимы, однако зачастую используют именно квадрат, поскольку он сильнее “наказывает” за рост ошибки

$$L(\hat{y}, y) = \frac{1}{n} \sum (\hat{y} - y)^2$$

$$L(\hat{y}, y) = \frac{1}{n} \sum |\hat{y} - y|$$

Регрессия: обучение модели

$$MSE = \frac{1}{n}\sum(\hat{y} - y)^2$$

- Mean Squared Error, MSE
- Среднеквадратичная ошибка
- **MSE** усиливает влияние крупных ошибок за счёт возведения отклонений в квадрат. Это значит, что большие ошибки (выбросы) влияют на метрику больше, чем маленькие.

$$MAE = \frac{1}{n}\sum|\hat{y} - y|$$

- Mean Absolute Error, MAE
- Средняя абсолютная ошибка
- **MAE** не чувствительна к выбросам, в отличие от метрик, основанных на квадратичных отклонениях, поскольку она работает с абсолютными значениями ошибок.

n - количество наблюдений (или точек данных) в наборе данных.

y_i - фактическое значение для i-го наблюдения.

$\hat{y}_i$ - предсказанное значение для i-го наблюдения.

Регрессия: обучение модели

- Функция L (Loss) – **Функция потерь**

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

$$\hat{y} = \sum_{i=1}^{p}(x_i \hat{w}_i) + \hat{b}$$

$$L(w_1, \ldots, w_p, b) = \frac{1}{n}\sum_{i=1}^{n}(y_i - (w_1 x_1 + \ldots + w_p x_p + b))^2$$

- **Задача обучения – найти такие параметры модели (W, b), при которых значение функции потерь будет минимальным**

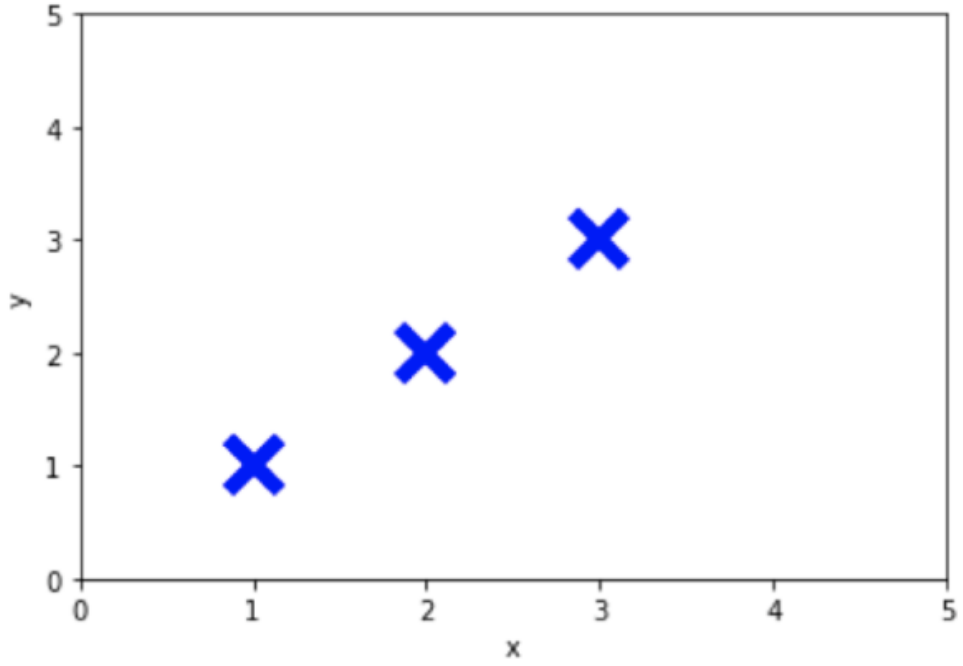
$$W, b = argmin(L(w_1, \ldots, w_p, b))$$

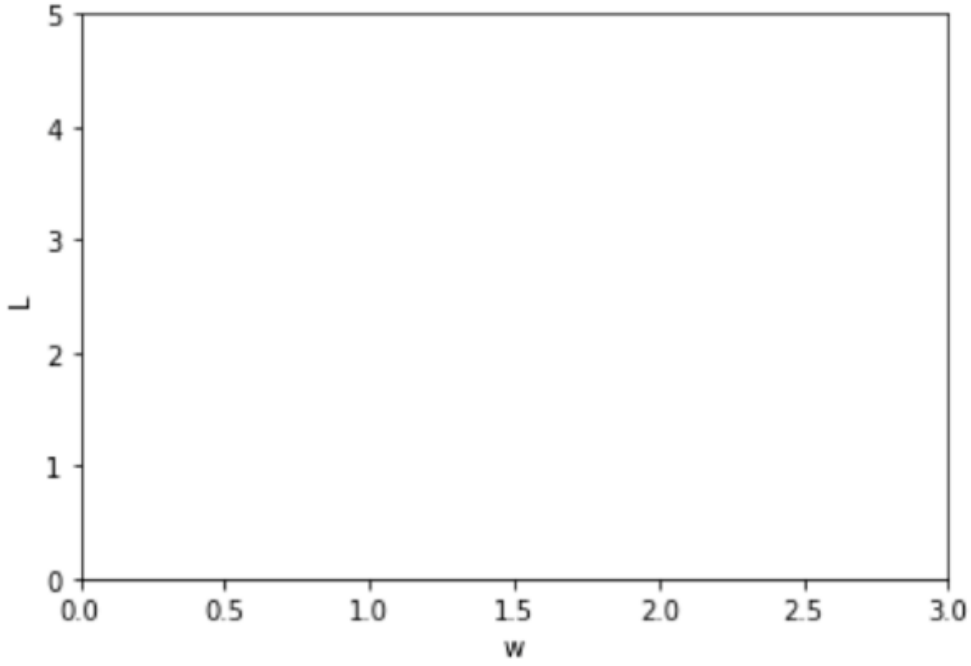
Регрессия: обучение модели

- Визуализируем функцию потерь

$$\hat{y} = wx + b$$

$$L(\hat{y}, y) = \frac{1}{n} \sum (\hat{y} - y)^2$$





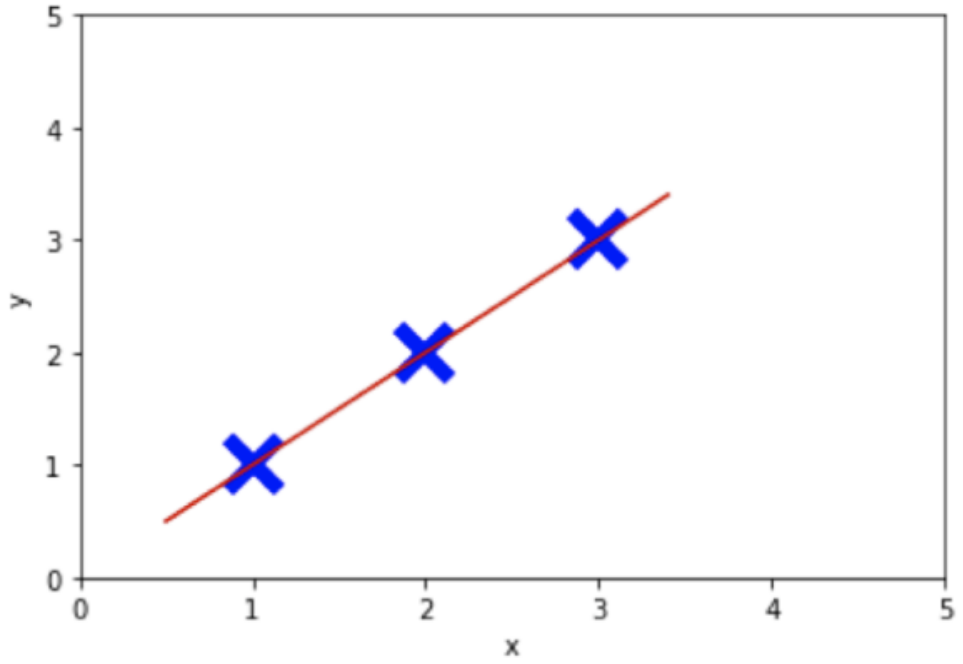
Регрессия: обучение модели

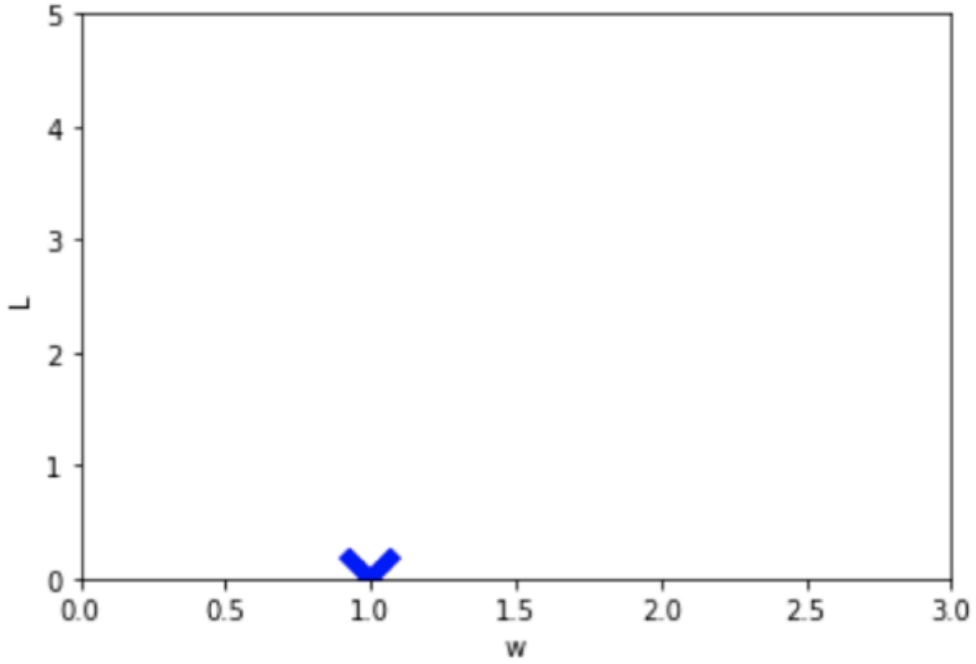
- Визуализируем функцию потерь

$$L(\hat{y}, y) = \frac{1}{n} \sum (\hat{y} - y)^2$$

$$\hat{y} = wx + b$$

$$w = 1$$





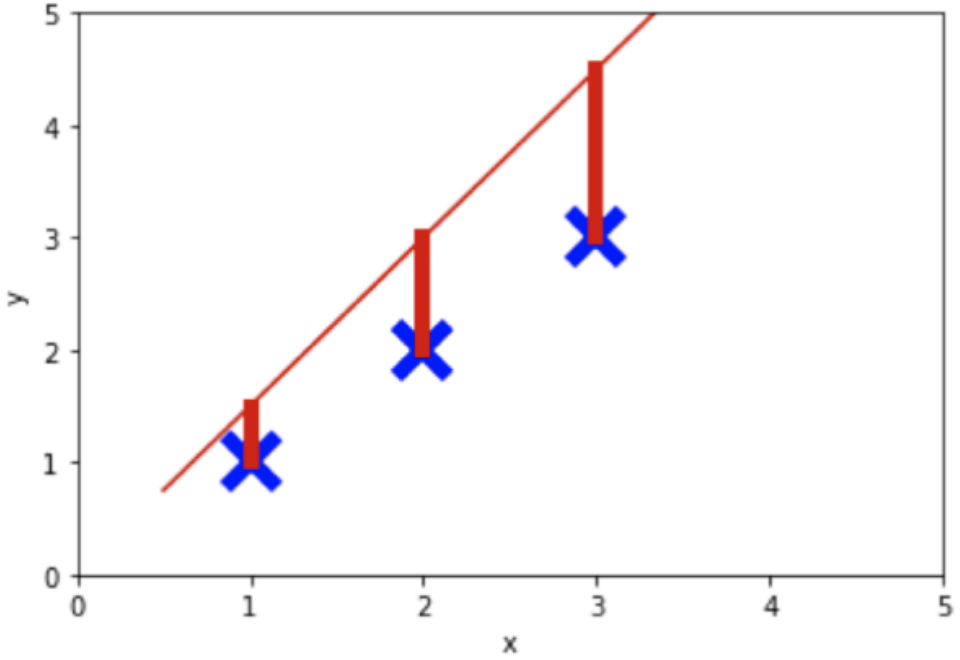
Регрессия: обучение модели

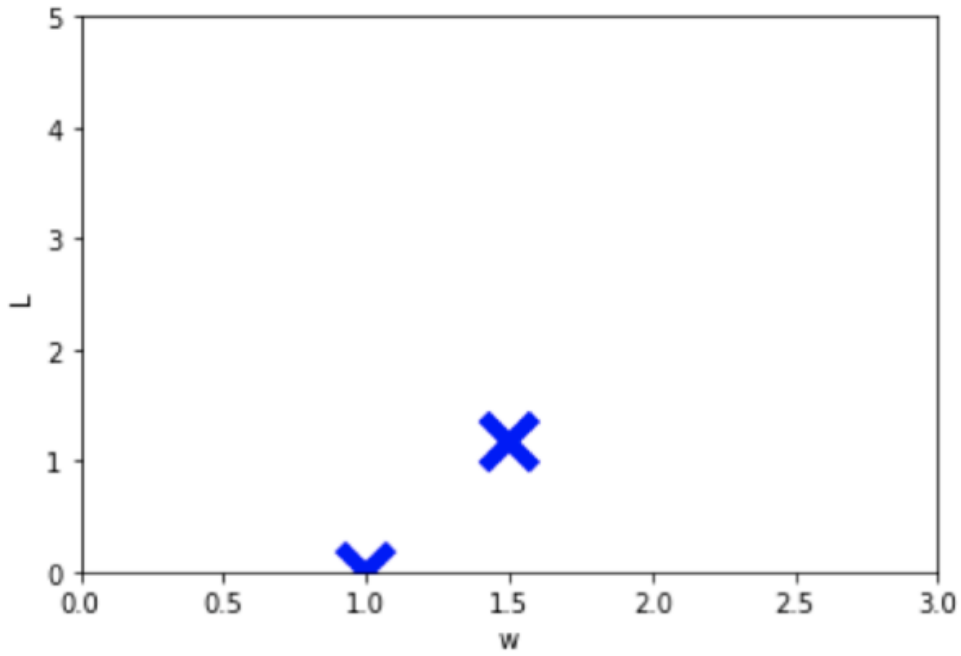
- Визуализируем функцию потерь

$$\hat{y} = wx + b$$

$$L(\hat{y}, y) = \frac{1}{n} \sum (\hat{y} - y)^2$$

$$w = 1.5$$





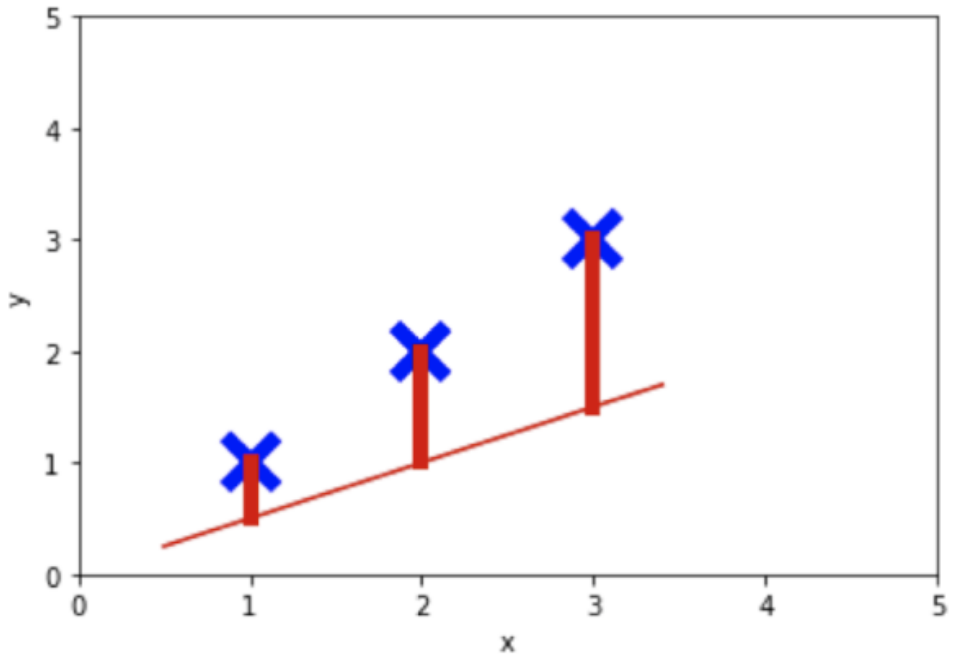
Регрессия: обучение модели

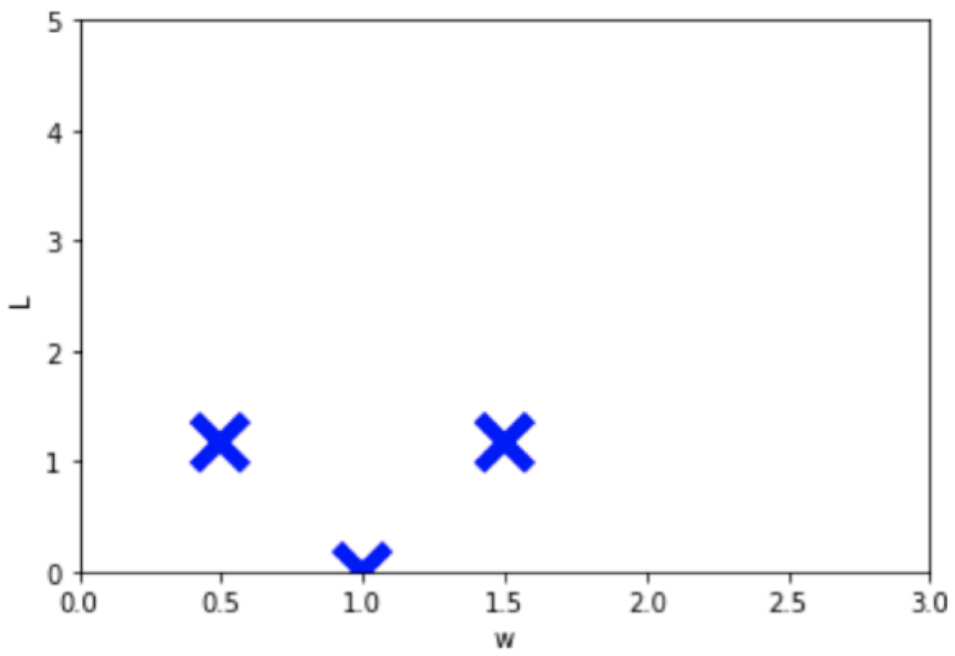
- Визуализируем функцию потерь

$$L(\hat{y}, y) = \frac{1}{n} \sum (\hat{y} - y)^2$$

$$\hat{y} = wx + b$$

$$w = 0.5$$





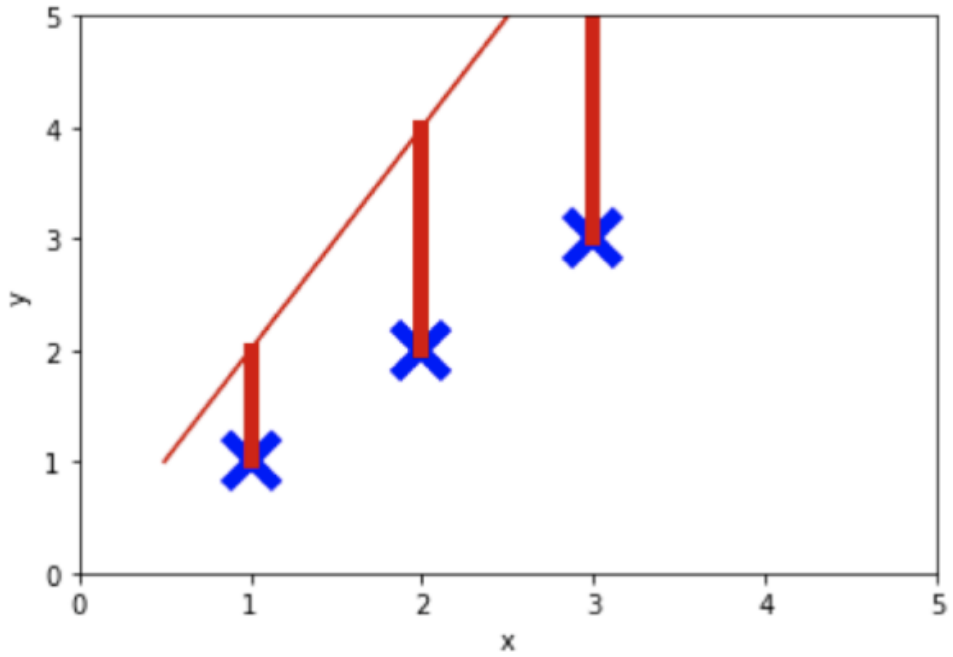
Регрессия: обучение модели

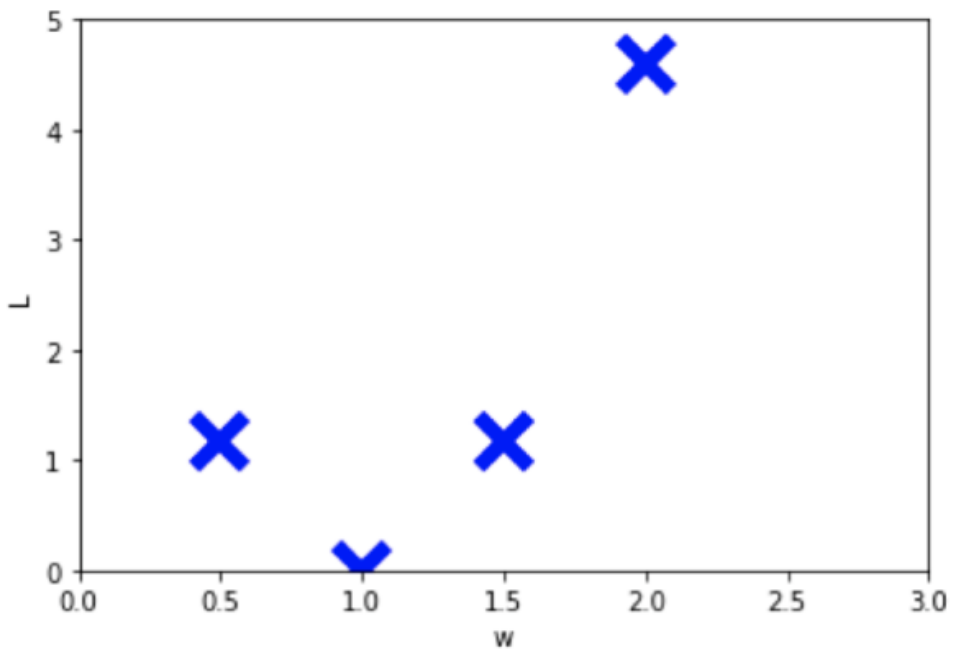
- Визуализируем функцию потерь

$$L(\hat{y}, y) = \frac{1}{n} \sum (\hat{y} - y)^2$$

$$\hat{y} = wx + b$$

$$w = 2$$





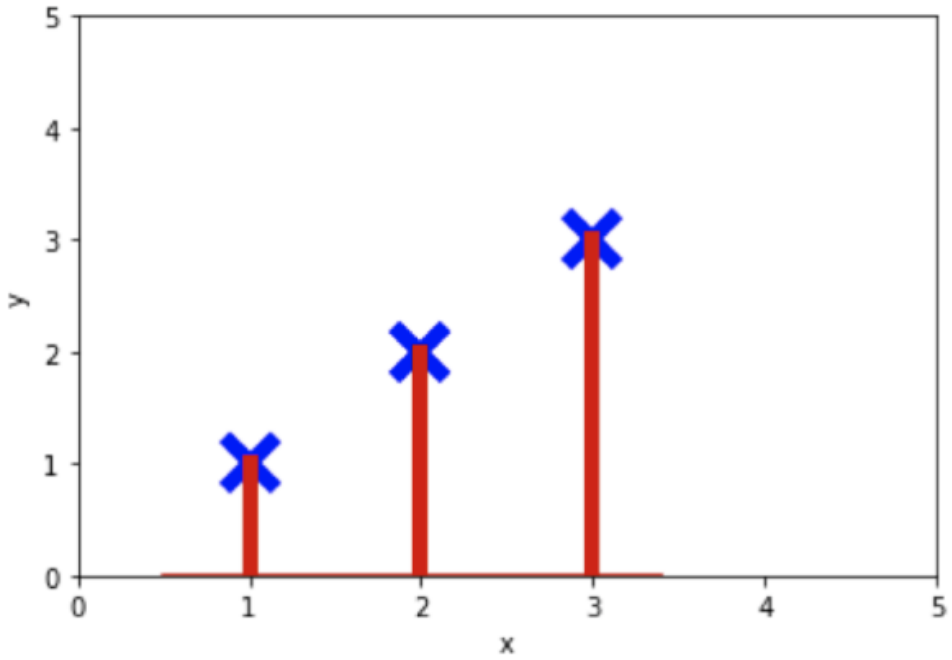
Регрессия: обучение модели

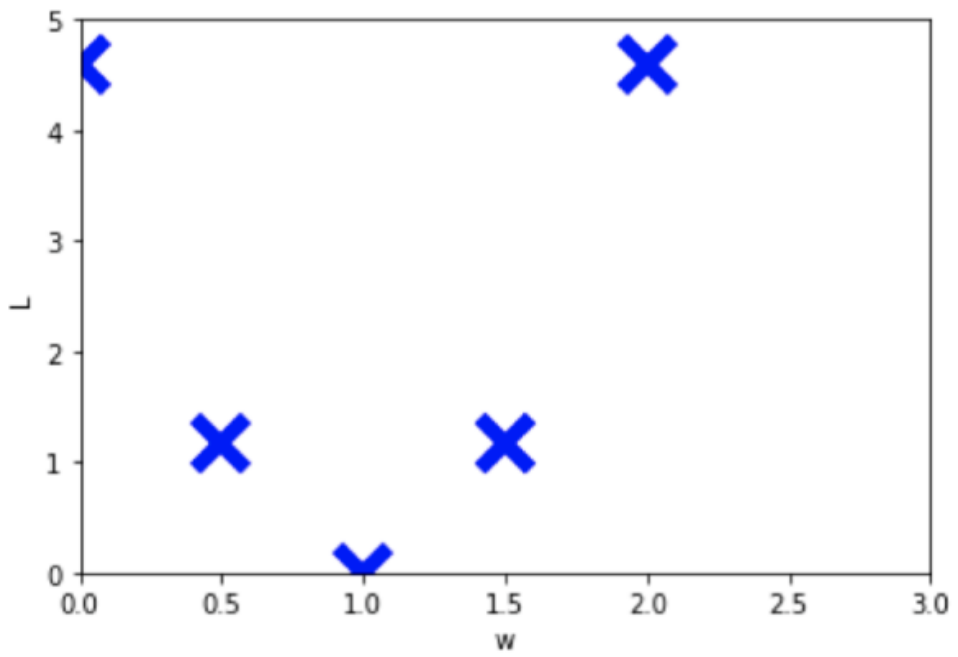
- Визуализируем функцию потерь

$$L(\hat{y}, y) = \frac{1}{n} \sum (\hat{y} - y)^2$$

$$\hat{y} = wx + b$$

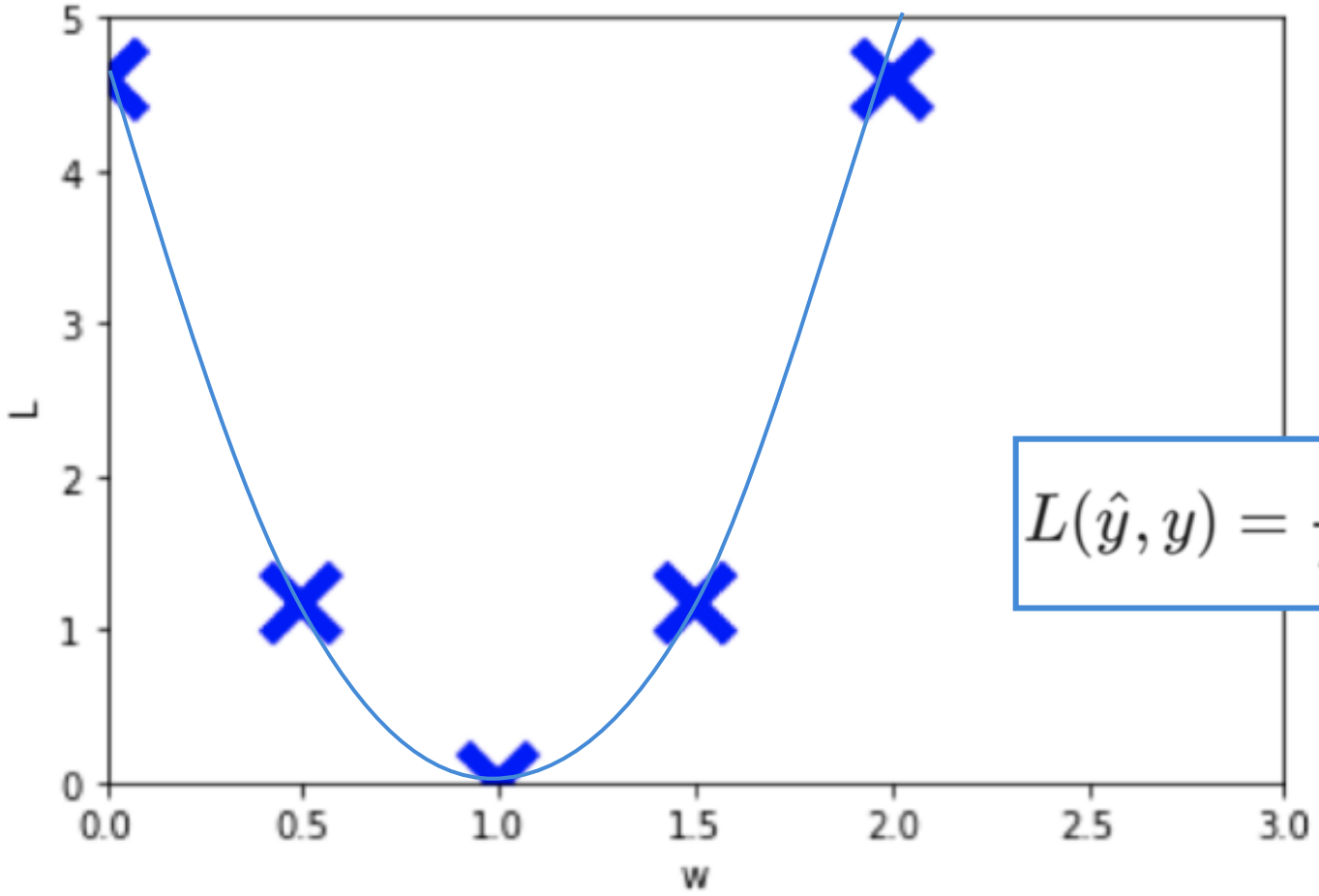
$$w = 0$$





Регрессия: обучение модели

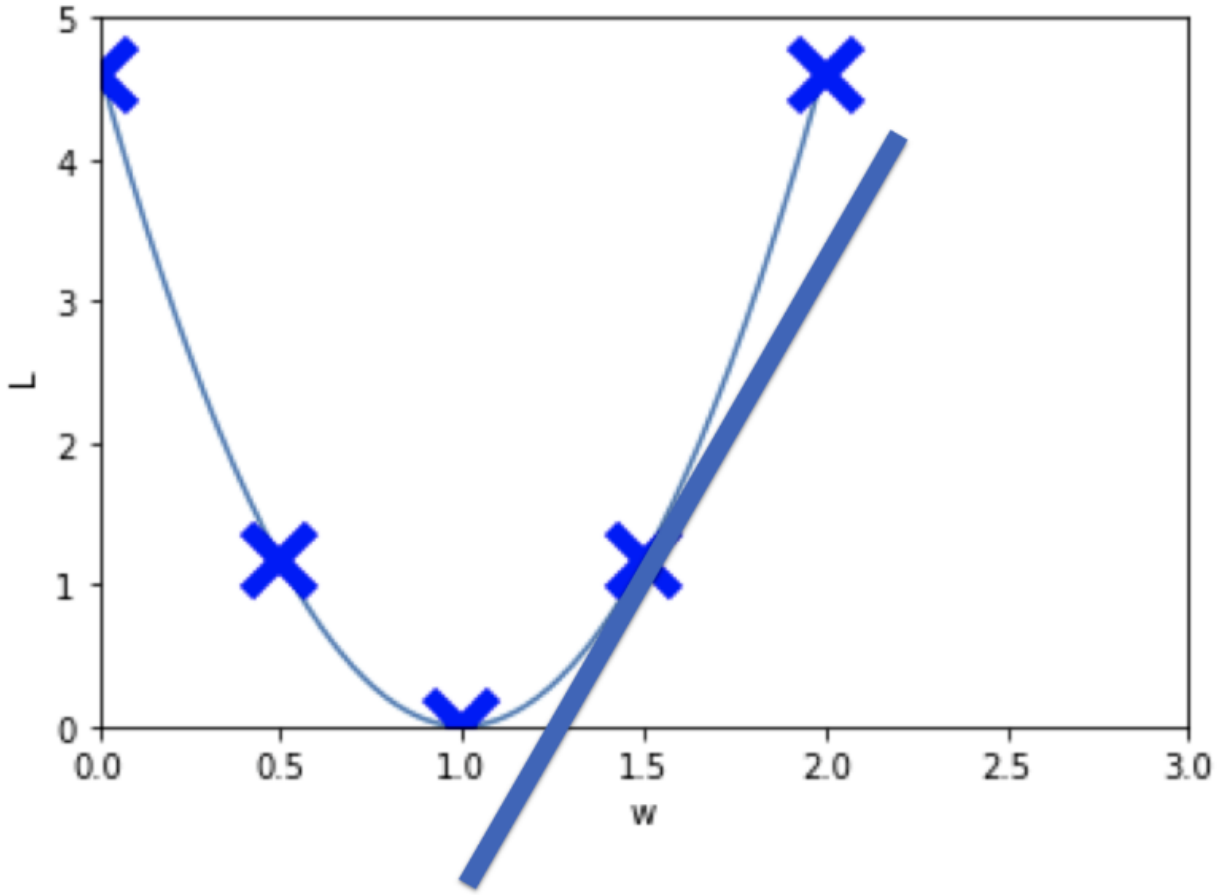
- Что получилось и почему?



$$L(\hat{y}, y) = \frac{1}{n} \sum (\hat{y} - y)^2$$

Регрессия: обучение модели

- Надо минимизировать функцию
- Производная указывает направление роста функции



Если

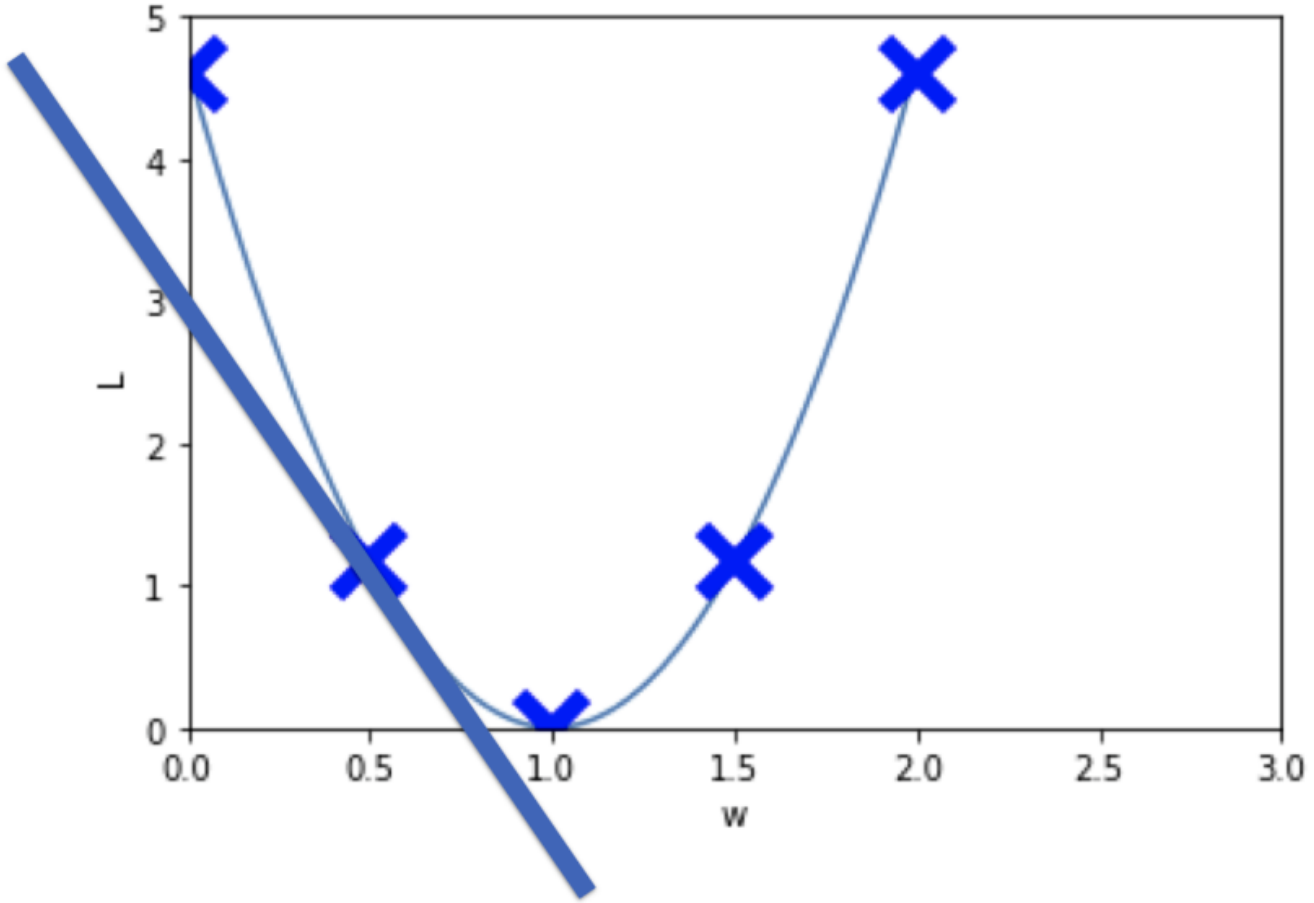
$$f(x) = x^2$$

то

$$f'(x) = 2x$$

Регрессия: обучение модели

- Надо минимизировать функцию
- Производная указывает направление роста функции



Если

$$f(x) = x^2$$

то

$$f'(x) = 2x$$

Регрессия: обучение модели

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

$$L(w_1, \ldots, w_p) = \frac{1}{n}\sum_{i=1}^{n}(y_i - (x_1 w_1 + \ldots + x_p w_p + b))^2$$

$$\frac{\partial L}{\partial w_1} = \frac{2}{n}\sum_{i=1}^{n}(x_1 w_1 + \ldots + x_p w_p + b - y_i)x_1 = \frac{2}{n}\sum_{i=1}^{n}(wx + b - y_i)x_1$$

$$\ldots$$

$$\frac{\partial L}{\partial w_p} = \frac{2}{n}\sum_{i=1}^{n}(x_1 w_1 + \ldots + x_p w_p + b - y_i)x_p = \frac{2}{n}\sum_{i=1}^{n}(wx + b - y_i)x_p$$

$$\frac{\partial L}{\partial b} = \frac{2}{n}\sum_{i=1}^{n}(x_1 w_1 + \ldots + x_p w_p + b - y_i) = \frac{2}{n}\sum_{i=1}^{n}(wx + b - y_i)$$

Регрессия: обучение модели

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

$$L(w_1, \ldots, w_p) = \frac{1}{n}\sum_{i=1}^{n}(y_i - (x_1 w_1 + \ldots + x_p w_p + b))^2$$

$$\frac{\partial L}{\partial w_1} = \frac{2}{n}\sum_{i=1}^{n}(x_1 w_1 + \ldots + x_p w_p + b - y_i)x_1 = \frac{2}{n}\sum_{i=1}^{n}(wx + b - y_i)x_1$$

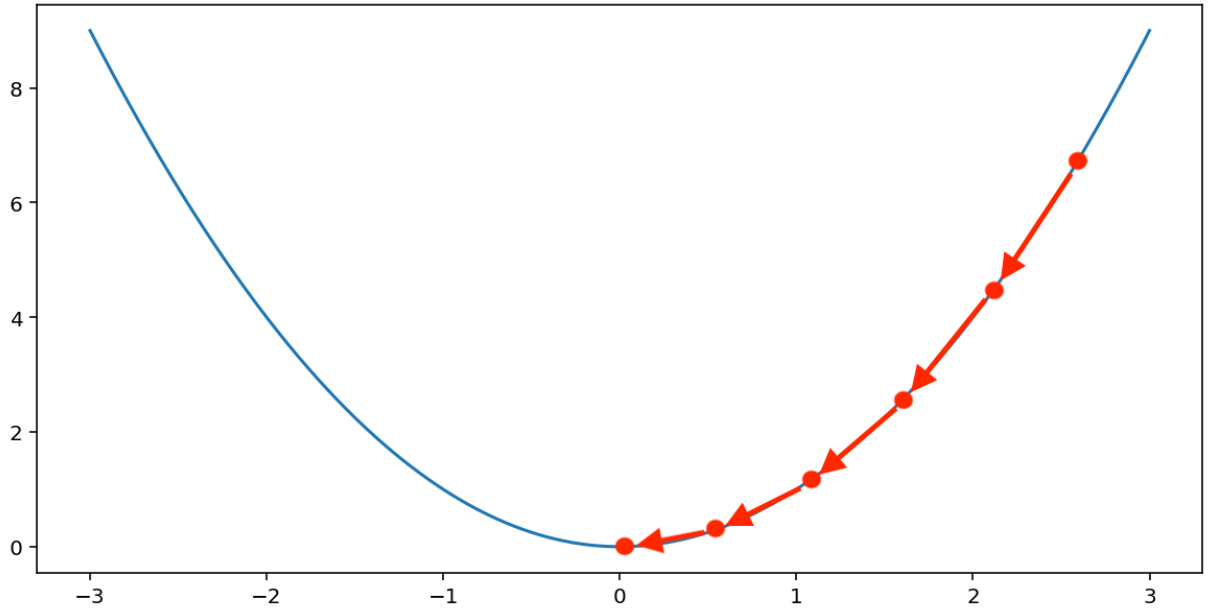
$$\ldots$$

$$\frac{\partial L}{\partial w_p} = \frac{2}{n}\sum_{i=1}^{n}(x_1 w_1 + \ldots + x_p w_p + b - y_i)x_p = \frac{2}{n}\sum_{i=1}^{n}(wx + b - y_i)x_p$$

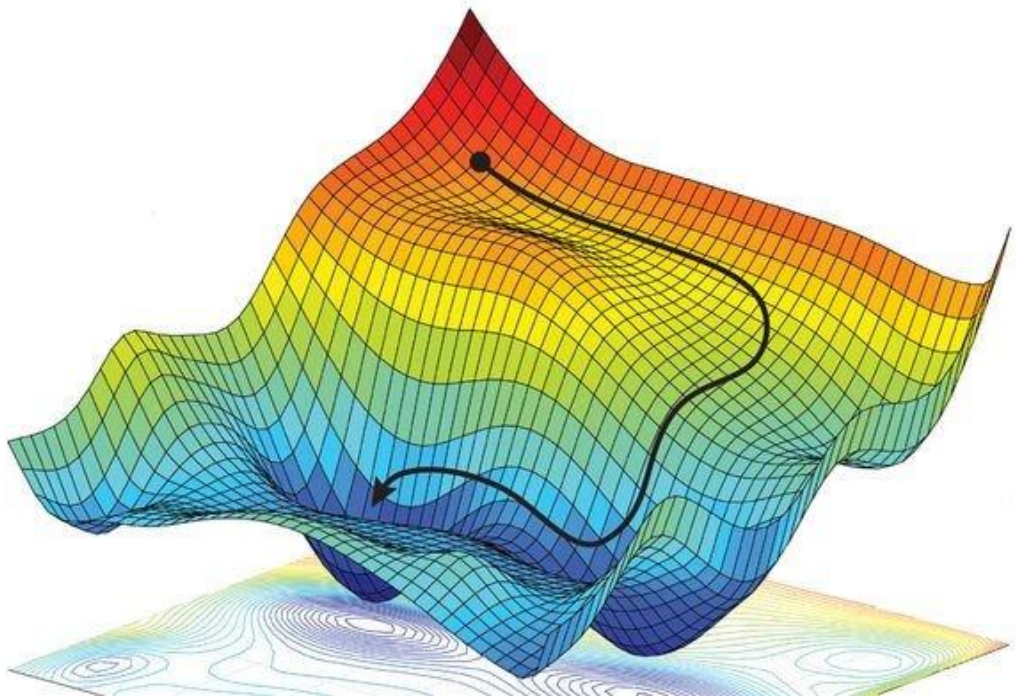
$$\frac{\partial L}{\partial b} = \frac{2}{n}\sum_{i=1}^{n}(x_1 w_1 + \ldots + x_p w_p + b - y_i) = \frac{2}{n}\sum_{i=1}^{n}(wx + b - y_i)$$

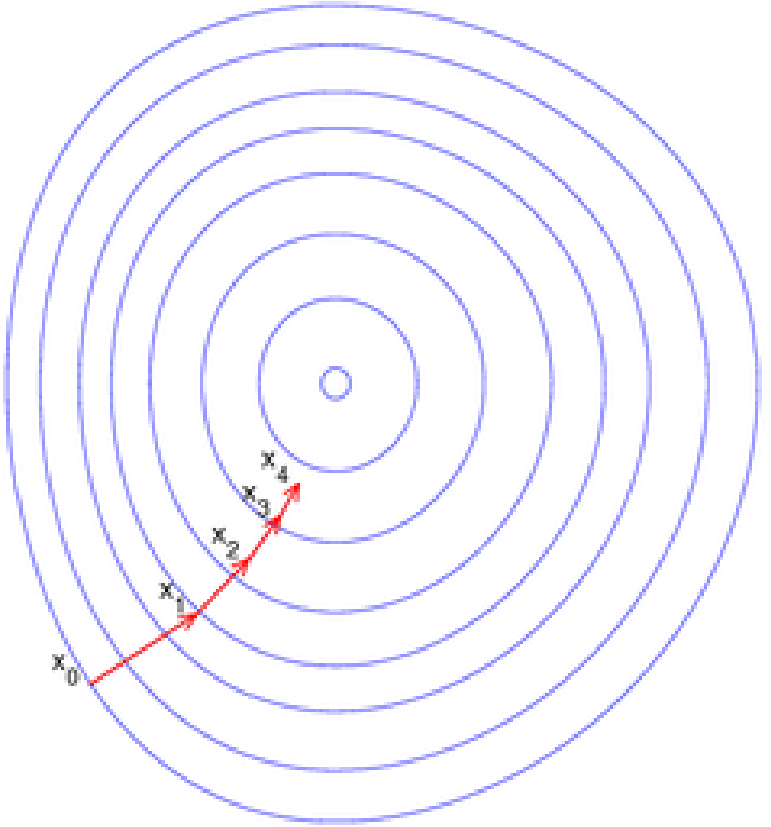
Регрессия: градиентный спуск

- **Градиентный спуск** - это оптимизационный алгоритм, который используется для минимизации функций, в частности, для настройки параметров модели в машинном обучении.
- Главная идея заключается в поиске **минимума функции**, двигаясь по направлению, **противоположному градиенту** (первой производной) функции.
- Это означает, что алгоритм "спускается" по функции, пока не достигнет локального минимума.

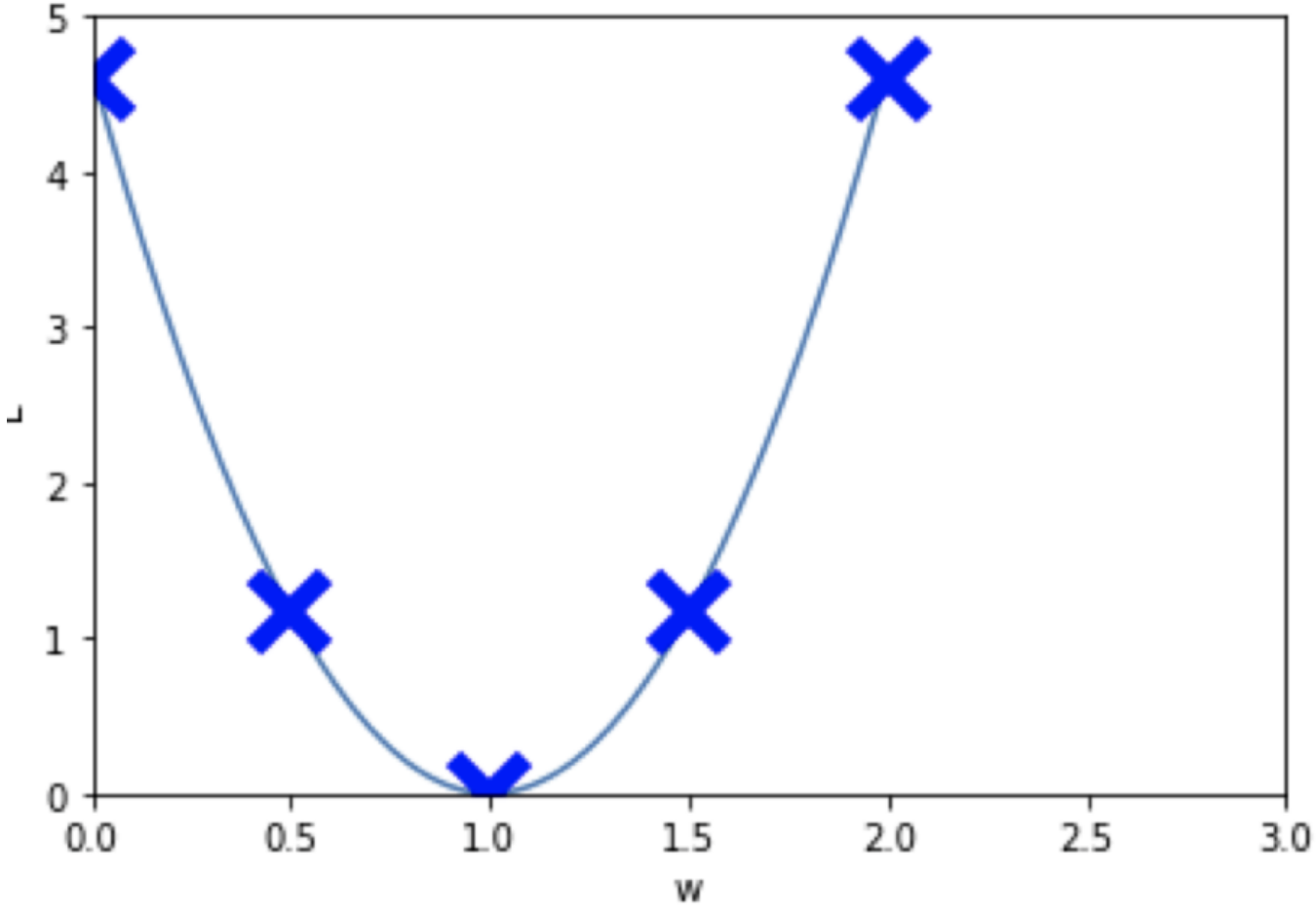


Регрессия: градиентный спуск





Регрессия: градиентный спуск



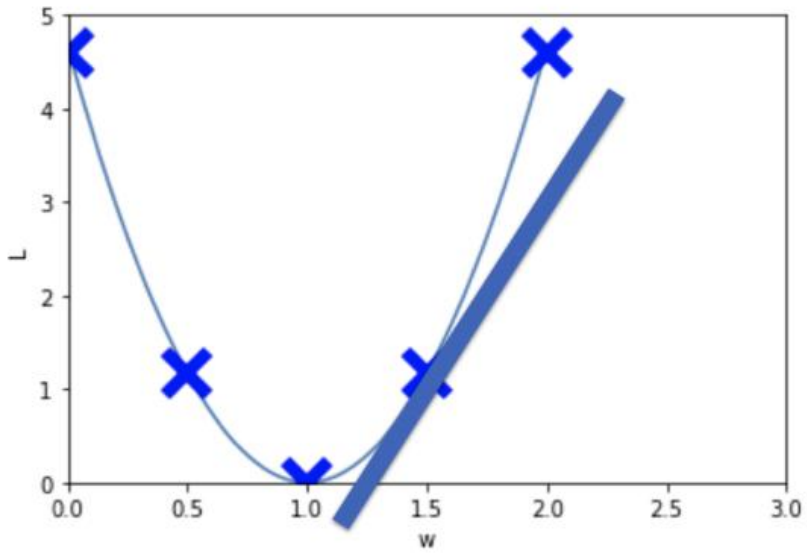
$$w_1 := w_1 - \frac{\partial L(w_1)}{\partial w_1}$$

$$\dots$$

$$w_p := w_p - \frac{\partial L(w_p)}{\partial w_p}$$

$$b := b - \frac{\partial L(b)}{\partial b}$$

Регрессия: градиентный спуск

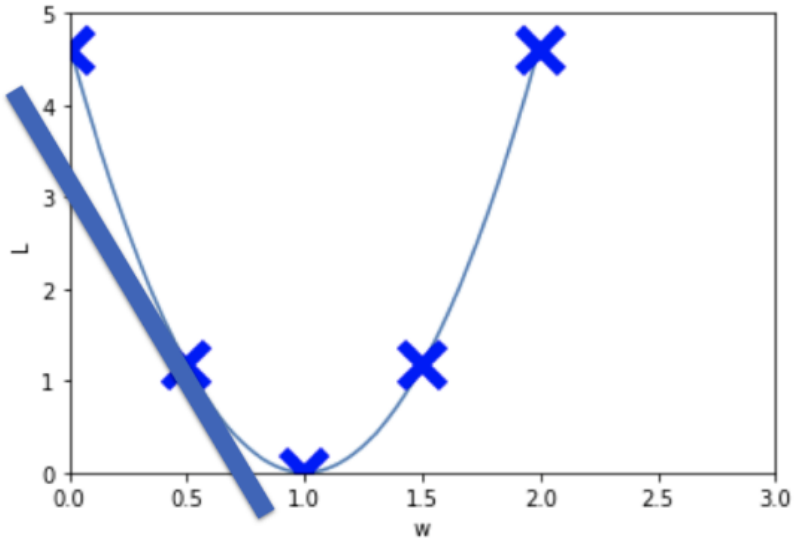


$$X = [1, 2, 3] \quad Y = [1, 2, 3] \quad w = 1.5 \quad b = 0$$

1. $$f(x) = \frac{1}{n} \sum (wx + b - y)^2$$
2. $$\frac{\partial f(x)}{\partial w} = \frac{2}{n} \sum (wx + b - y)x$$
3. $$\frac{\partial f(x)}{\partial w} = \frac{2}{3} \sum (1.5x - y)x$$

$$\frac{\partial f(x)}{\partial w} = \frac{2}{3}((1.5 \times 1 - 1) \times 1 + (1.5 \times 2 - 2) \times 2 + (1.5 \times 3 - 3) \times 3) = 4.6667$$

Регрессия: градиентный спуск



$$X = [1, 2, 3] \quad Y = [1, 2, 3] \quad w = 0.5 \quad b = 0$$

1. $$f(x) = \frac{1}{n} \sum (wx + b - y)^2$$
2. $$\frac{\partial f(x)}{\partial w} = \frac{2}{n} \sum (wx + b - y)x$$
3. $$\frac{\partial f(x)}{\partial w} = \frac{2}{3} \sum (1.5x - y)x$$

$$\frac{\partial f(x)}{\partial w} = \frac{2}{3}((0.5 \times 1 - 1) \times 1 + (0.5 \times 2 - 2) \times 2 + (0.5 \times 3 - 3) \times 3) = -4.6667$$

Регрессия: градиентный спуск

Градиент функции потерь

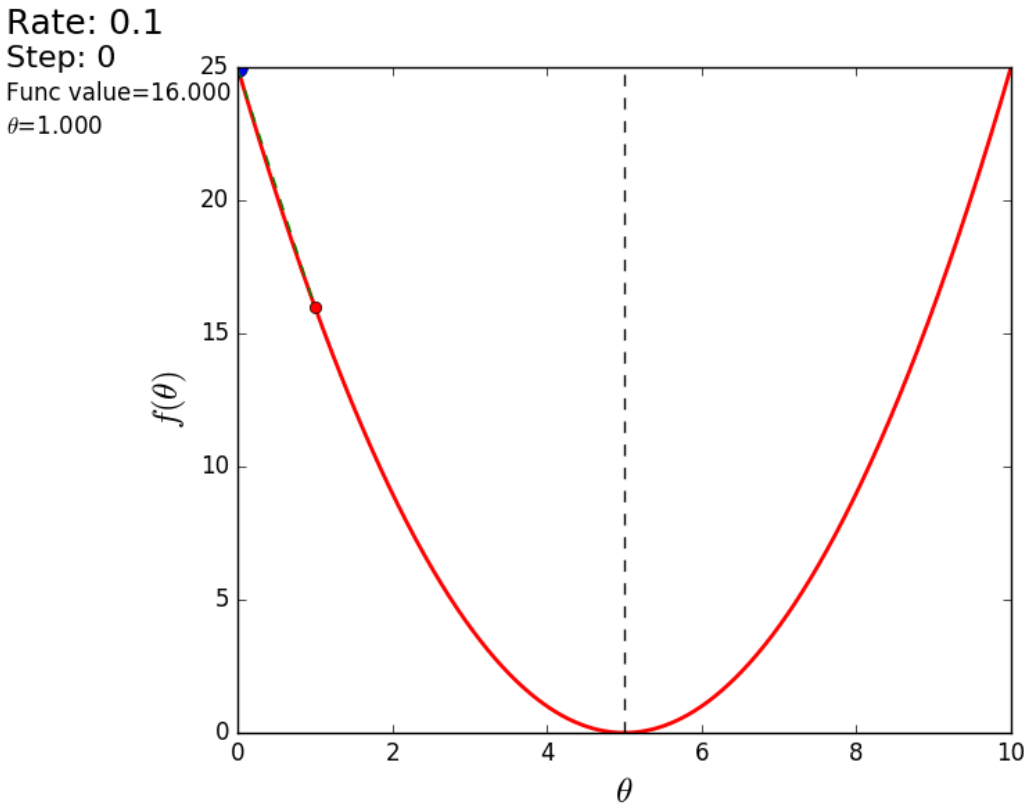
$$\nabla L(w) = \left(\frac{\partial L}{\partial w_1}, \frac{\partial L}{\partial w_2}, \ldots, \frac{\partial L}{\partial w_n} \right)$$

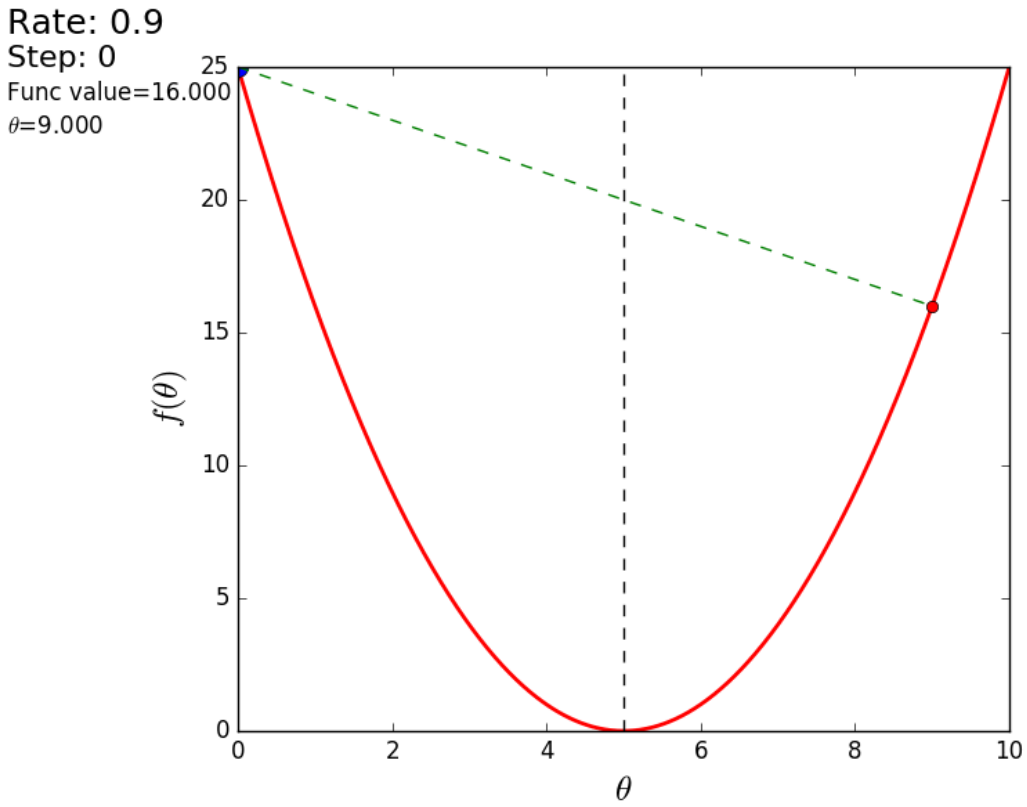
Обновление весов на каждом шаге

$$w = w - \alpha \cdot \nabla L(w)$$

***α* - скорость обучения (learning rate)**

Регрессия: градиентный спуск





Регрессия: градиентный спуск

- Алгоритм градиентного спуска

$$w_1, \ldots, w_p := 0$$
$$b := 0$$
$$for\ i\ in\ range(n\_iter):$$
$$w_1 := w_1 - \alpha \frac{\partial L}{\partial w_1}$$
$$\ldots$$
$$w_p := w_p - \alpha \frac{\partial L}{\partial w_p}$$
$$b := b - \alpha \frac{\partial L}{\partial b}$$

Проход по всем экземплярам данных в обучающей выборке - ***эпоха***

Гиперпараметры

n_iter - число эпох

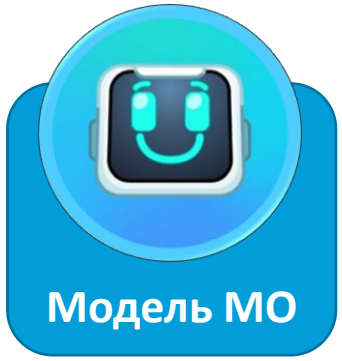
α - скорость обучения

Регрессия: градиентный спуск

- **Гиперпараметры** - это параметры, которые **не настраиваются автоматически** алгоритмом машинного обучения в процессе обучения модели, а **задаются вручную** до начала обучения.
- Они представляют собой конфигурационные параметры модели, которые влияют на ее обучение и производительность, но не определяются непосредственно из данных.

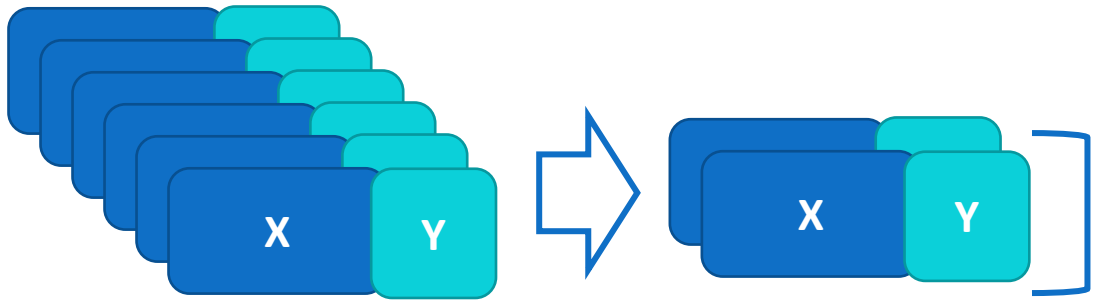
n_iter - число эпох

α - скорость обучения



Батчи

- Проблема – на всём датасете считать градиент **очень долго**!
- Решение – использовать не весь, а **часть**
- В контексте машинного обучения, "**батч**" (batch) - это небольшой поднабор данных, который используется для одной итерации (одного шага) обучения модели.
- Батчи используются в алгоритмах градиентного спуска
- Способ, при котором шаг градиентного спуска делается не один раз в эпоху, а несколько раз.



Обучающий датасет

Батч

Батчи

Размер батча

- Размер батча определяет, сколько обучающих примеров будет использоваться на каждой итерации обучения. Например, в задачах глубокого обучения типичные размеры батчей могут составлять 32, 64, 128 и так далее. Выбор размера батча может влиять на скорость обучения, использование памяти и стабильность сходимости модели.

Преимущества батчей:

- Ускорение обучения: Использование батчей позволяет распараллеливать вычисления, что может значительно ускорить обучение на многопроцессорных системах и графических процессорах.
- Управление памятью: Многие модели и наборы данных могут быть слишком большими для хранения в оперативной памяти целиком, поэтому батчи позволяют обрабатывать данные по частям.

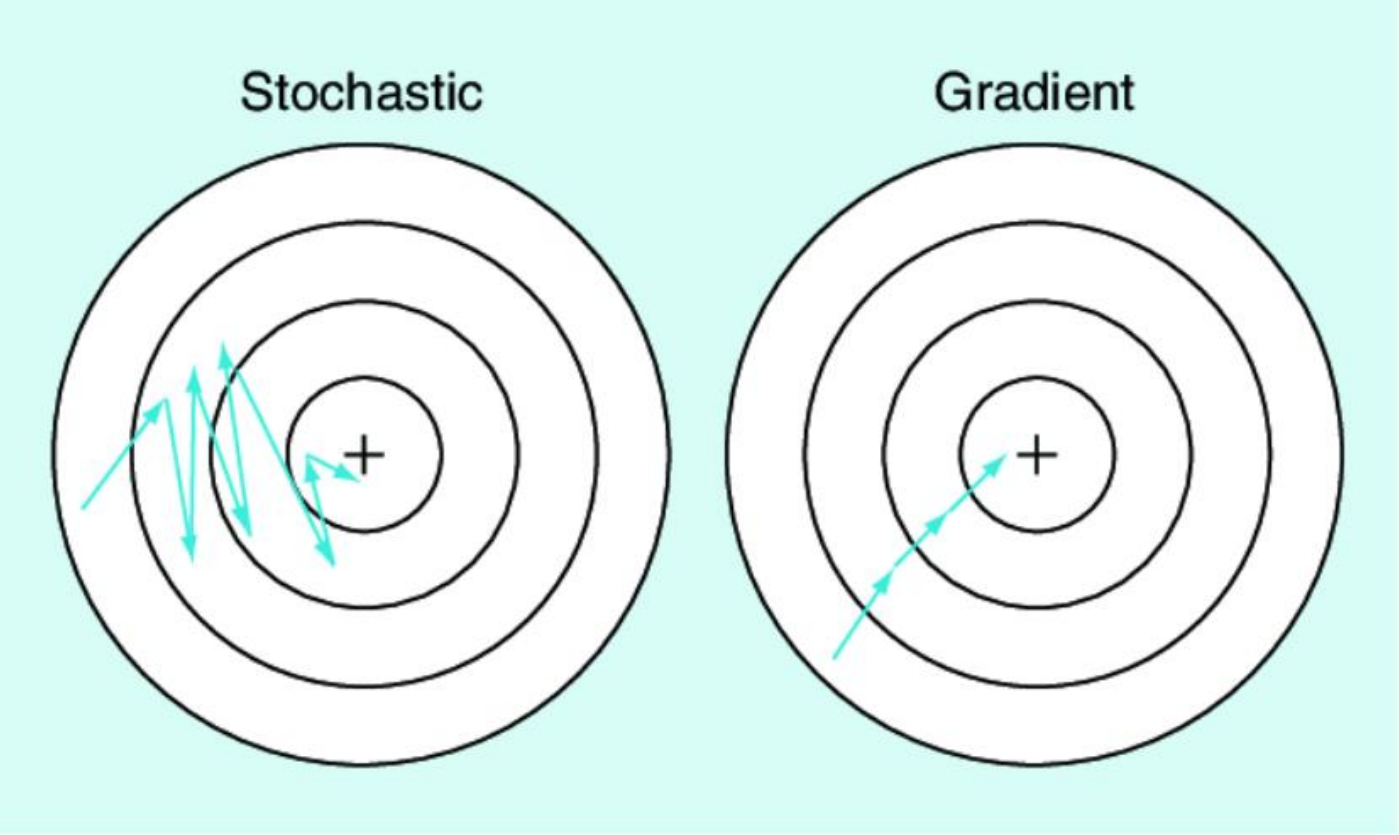
Батчи

- **Типы батчей:**

  - **Полный батч (Batch GD):** Все доступные данные используются на каждой итерации. Это обеспечивает стабильную и точную сходимость, но может быть вычислительно затратным для больших наборов данных.
  - **Стохастический батч (SGD):** На каждой итерации случайно выбирается один обучающий пример. Это ускоряет сходимость, но может привести к шумным обновлениям параметров.
  - **Мини-батч (Mini-batch GD):** Используется случайное подмножество данных заданного размера. Это компромисс между Batch GD и SGD и широко используется в глубоком обучении.

Стохастический градиентный спуск

- Способ, при котором шаг градиентного спуска делается не один раз в эпоху, а на каждый экземпляр данных



Стохастический градиентный спуск

- **Обработка данных:**
  - **Обычный градиентный спуск (Batch GD):** В этом методе на каждой итерации используется вся тренировочная выборка данных для вычисления градиента функции потерь. Это означает, что все обучающие примеры участвуют в одной итерации.
  - **Стохастический градиентный спуск (SGD):** В этом методе на каждой итерации случайно выбирается один обучающий пример (или небольшой мини-пакет) для вычисления градиента. Это означает, что данные обрабатываются по одному или нескольким примерам за итерацию.
- **Скорость сходимости:**
  - **Обычный градиентный спуск (Batch GD):** Обычно сходится более стабильно и предсказуемо, так как каждая итерация использует полный набор данных. Однако, в зависимости от размера выборки, он может быть более медленным и требовательным к памяти, особенно для больших наборов данных.
  - **Стохастический градиентный спуск (SGD):** Сходится быстрее на начальных этапах обучения, так как обновления параметров происходят более часто. Однако, из-за случайного выбора обучающих примеров, SGD может иметь большие колебания в процессе обучения и требует тщательной настройки скорости обучения.
- **Шум и вариативность:**
  - **Обычный градиентный спуск (Batch GD):** Использование всей тренировочной выборки сглаживает градиент и уменьшает шум, что делает его менее подверженным вариативности в обновлениях параметров.
  - **Стохастический градиентный спуск (SGD):** Из-за случайного выбора примеров SGD может создавать более шумные обновления параметров, что может помочь выходить из локальных минимумов, но также может привести к более непостоянной сходимости.

Спасибо за внимание!