# Регрессия

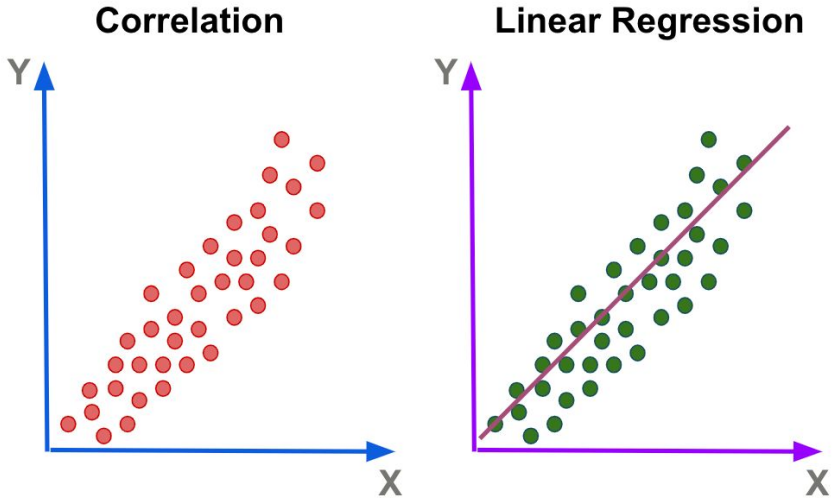
В основе - презентация Анны Дмитриевой

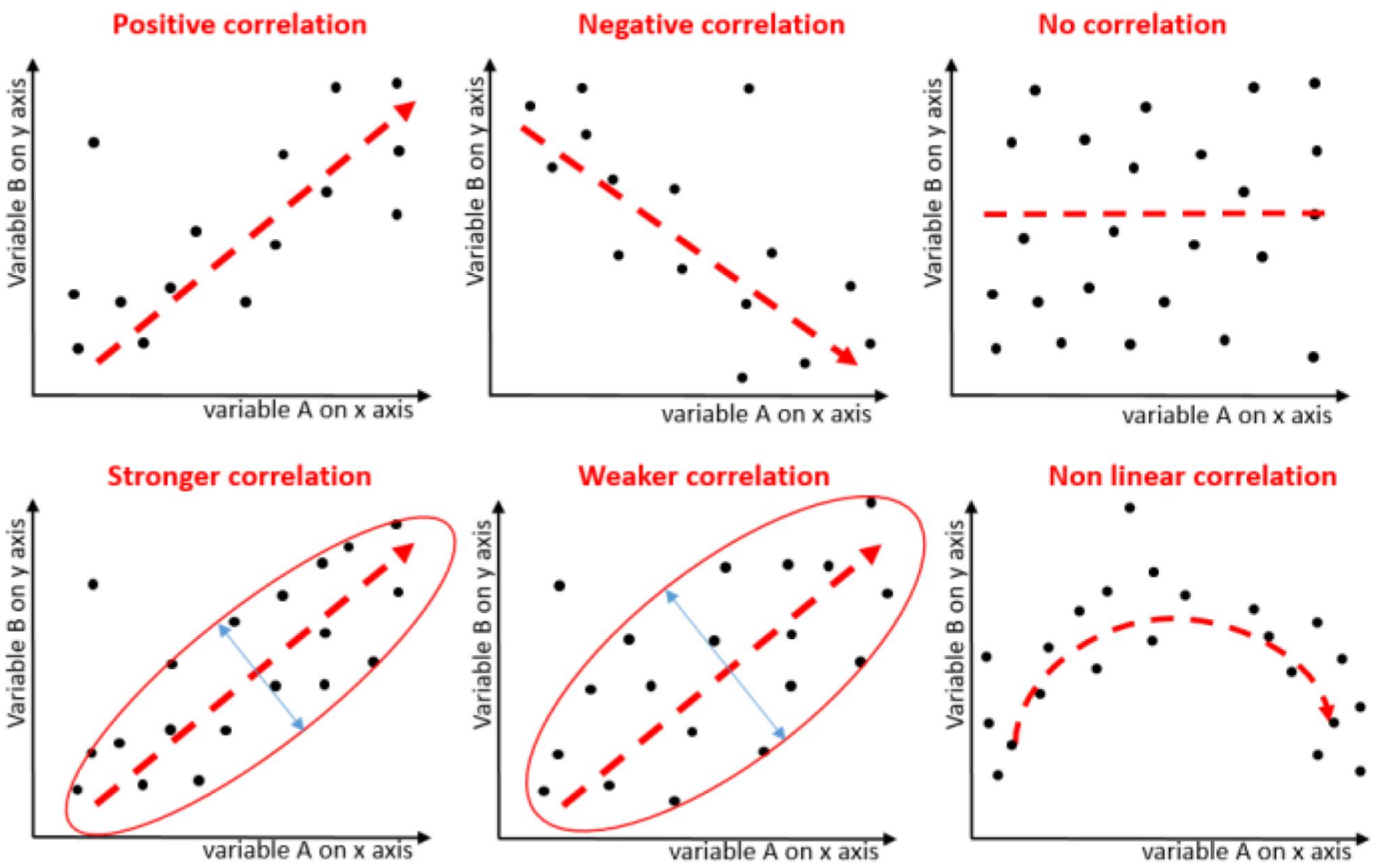
ДПО Компьютерная лингвистика

# Регрессия

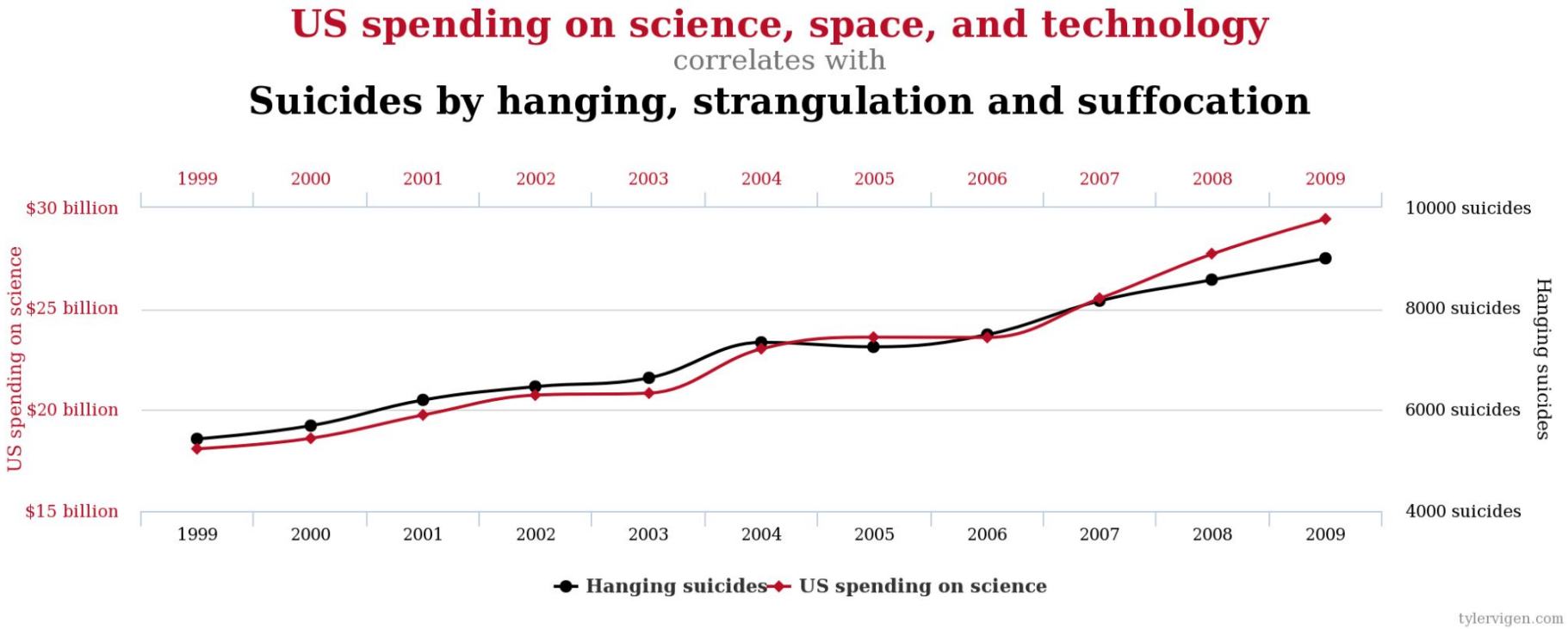
Задача: исследовать влияние независимых переменных на зависимую.

Регрессия отличается от корреляции прежде всего тем, что задача, которую мы решаем, содержит прогноз

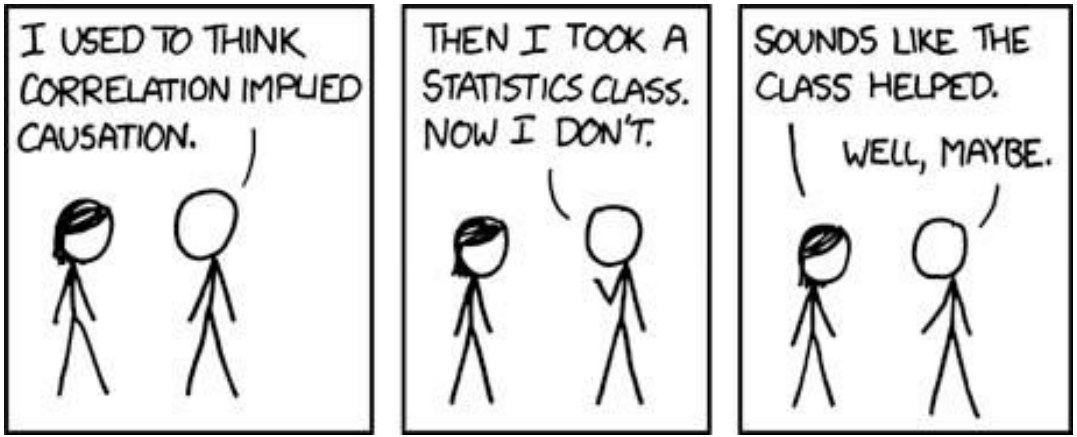



Positive correlation
Variable B on y axis
variable A on x axis
Negative correlation
Variable B on y axis
variable A on x axis
No correlation
Variable B on y axis
variable A on x axis
Stronger correlation
Variable B on y axis
variable A on x axis
Weaker correlation
Variable B on y axis
variable A on x axis
Non linear correlation
Variable B on y axis
variable A on x axis

# Важно! Корреляция ≠ каузация



# Важно! Корреляция ≠ каузация




# Еще одно отличие от корреляции:

Целевая (зависимая) переменная y

По ней мы делаем прогноз

Одна или несколько независимых переменных (x), которые влияют на y

Другие названия: признак, предиктор, регрессор, фича (в ML)

Важно! Эти признаки не должны коррелировать друг с другом (больше чем на 0,7) - иначе возникнет мультиколлинеарность

Мультиколлинеарность приводит к увеличению статистической неопределенности — дисперсии оценок. Это означает, что конкретные результаты оценки могут сильно различаться для разных выборок несмотря на то, что выборки однородны

# Линейная регрессия

Задача: предсказать значение зависимой переменной.

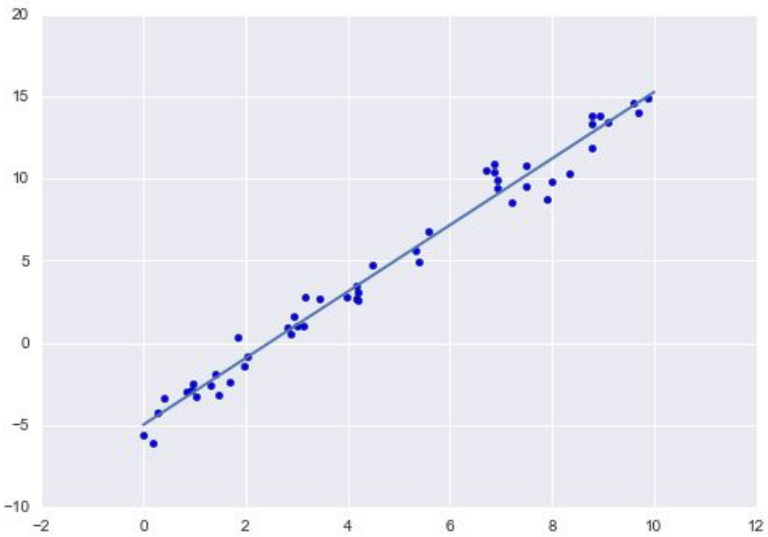
Функция в простейшей форме имеет вид:

$$y = wx + w_0$$

Где w - коэффициент, $w_0$ - intercept.

При увеличении количества признаков:

$$y = w_0 + w_1 x_1 + w_2 x_2 + [...]$$

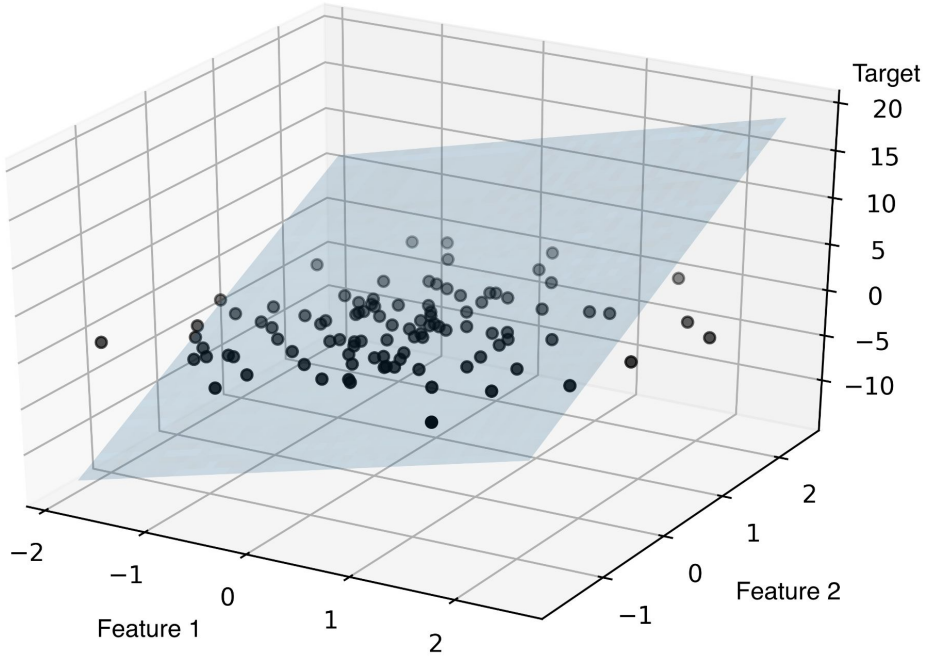



Картинка:
https://jakevdp.github.io/PythonDataScienceHandbook/05.06-linear-regression.html

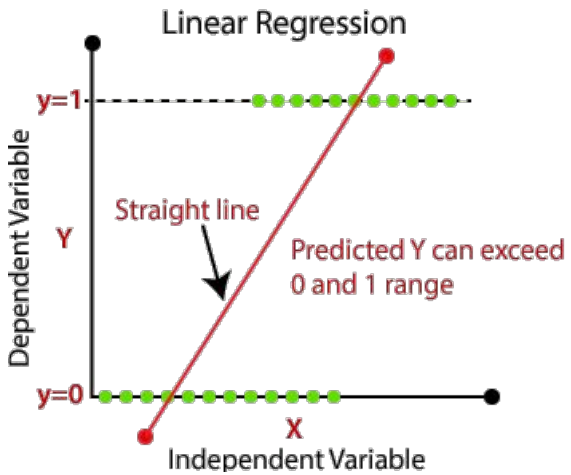
# Многомерная линейная регрессия

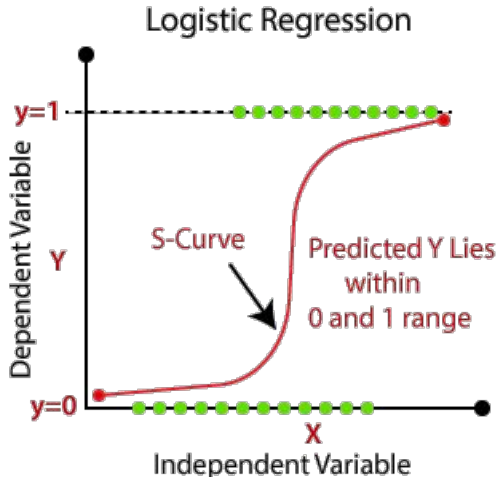
Вместо линии будем искать гиперплоскость

$$y = w_0 + w_1 x_1 + w_2 x_2 + [...]$$



# Разница между линейной регрессией и логистической




y = -1, 0, 0.1, 0.12… 1, 2…

y = 0 | y = 1

Мы еще вернемся к этому в конце

Картинка: https://www.javatpoint.com/linear-regression-vs-logistic-regression-in-machine-learning

# Оптимизация

# Как подобрать значения параметров?

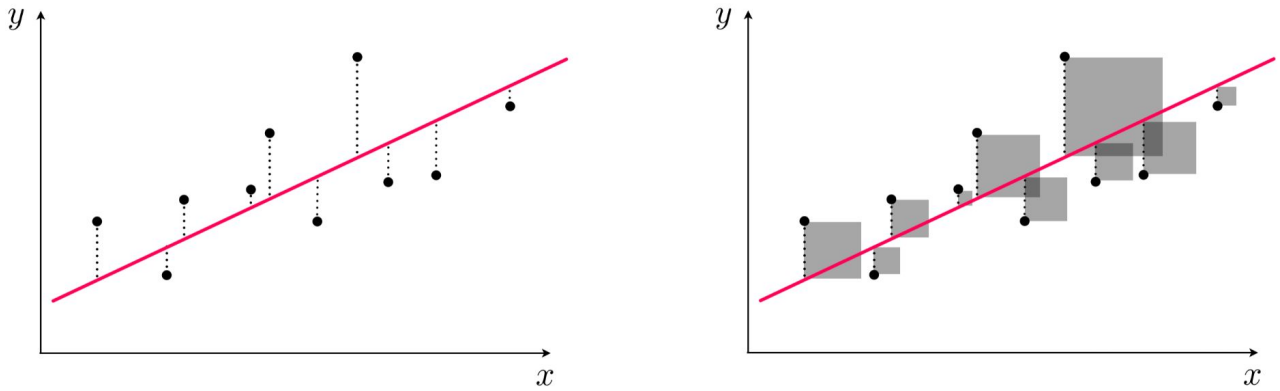
Подбор точных значений параметров - очень дорогая операция в многомерном пространстве. Поэтому мы будем итеративно приближать значения параметров к идеальным.

Мы начинаем со случайными параметрами, а потом постепенно исправляем их с помощью функции потерь (loss). Существует много видов функций потерь. Значения лосса большие, если модель работает плохо, и маленькие, если хорошо.

Для регрессий в качестве функции потерь чаще всего используется метод наименьших квадратов.

Важно! Мы не делаем это сами, алгоритм найдет идеальные параметры сам

# Метод наименьших квадратов




Задача - минимизировать длину линий ошибок, возведенных в квадрат.

Формула: $$Loss(w, x, y) = \frac{1}{N}\sum_{n=1}^{N}(wx_n + w_0 - y_n)^2$$

Картинка: https://jermwatt.github.io/machine_learning_refined/notes/5_Linear_regression/5_2_Least.html

# Другие виды регрессий

# Непараметрическая регрессия

Есть виды регрессии, в которых предиктор не принимает заранее определенную форму (например, прямой), а собирается на основе информации, выделенной из данных.

Примеры:

- kNN для регрессий;
- Деревья принятия решений;
- …

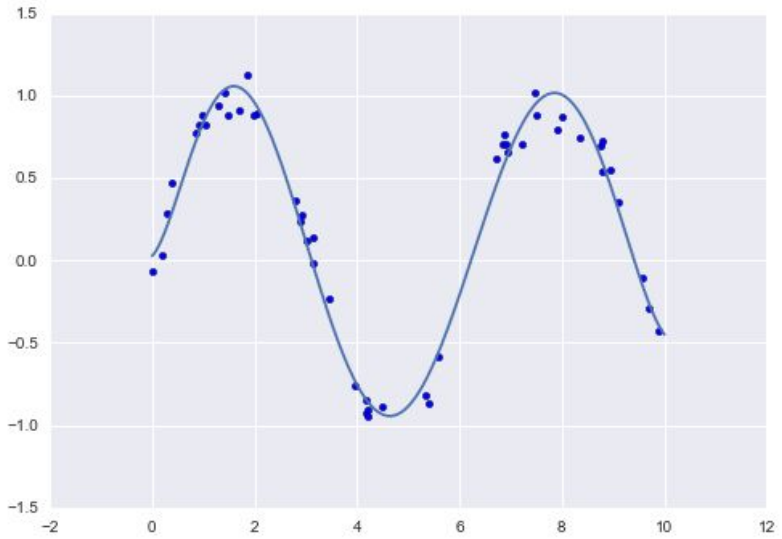
# Полиномиальная регрессия

Это регрессия вида:

$$y = w_0 + w_1 x_1 + w_2 x_2^2 + [...] + w_i x_i^n$$

Степенью этого уравнения будет максимальная степень n.

Полиномиальные регрессии помогают, когда зависимость между переменными нельзя выразить прямой линией.




Картинка:
https://jakevdp.github.io/PythonDataScienceHandbook/05.06-linear-regression.html

# Регуляризация

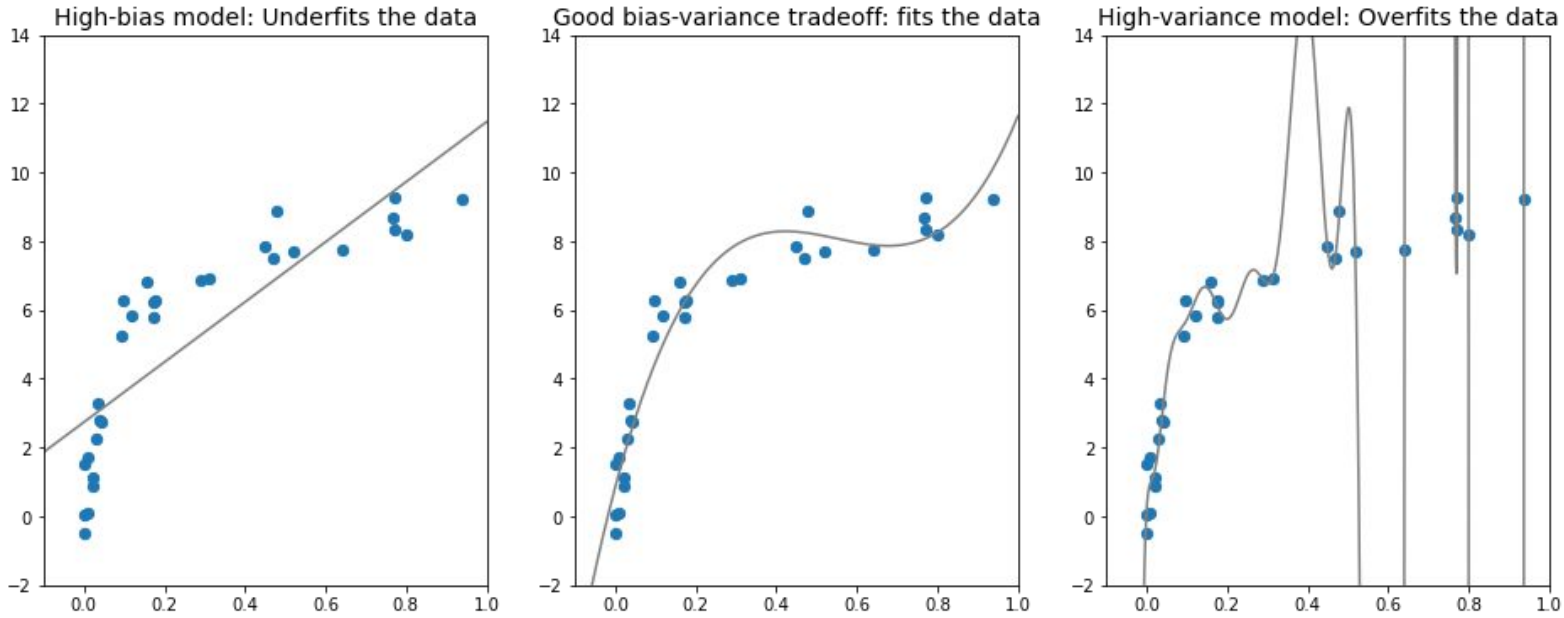
# Регуляризация регрессий

Регуляризация — метод добавления некоторых дополнительных ограничений к условию с целью решить некорректно поставленную задачу или предотвратить переобучение. (Википедия)

NB! В широком смысле,

- **Регуляризация** - предотвращение переобучения,
- **Оптимизация** - предотвращение недообучения (улучшение поиска минимума функции ошибки).

# Bias-variance tradeoff




Картинка нарисована по коду отсюда: https://jakevdp.github.io/PythonDataScienceHandbook/06.00-figure-code.html#Bias-Variance-Tradeoff. Степени полиномов: 1, 3, 20.

# Регуляризация регрессий

Мы должны добавить к функции потерь параметр **регуляризации**, который будет штрафовать модель за величину коэффициентов.

Чем больше параметр регуляризации, тем больше модель штрафуется за величину коэффициентов и их количество.

В хорошей модели у релевантных признаков, хорошо объясняющих зависимую переменную, должны быть коэффициенты ($\omega$) больше, чем у незначимых признаков.

# Ridge & LASSO

Это виды регрессии, которые по-разному регуляризуют функции потерь:

$$Loss_{Ridge}(w, x, y) = \frac{1}{N}\sum_{n=1}^{N}(wx_n + w_0 - y_n)^2 + \lambda w^2$$

$$Loss_{Lasso}(w, x, y) = \frac{1}{N}\sum_{n=1}^{N}(wx_n + w_0 - y_n)^2 + \lambda |w|$$

Loss - функция потерь, w - веса, λ - настраиваемый параметр/коэффициент регуляризации, т.е. число, которое мы можем выбрать.

# Разница между Ridge и Lasso

- Lasso имеет более выраженную тенденцию к занулению коэффициентов (=избавлению от признаков). Она может быть полезна, если вы:
  - Заведомо знаете, что не все признаки будут вам полезны;
  - Имеете ограничения по скорости построения предсказаний, и вам выгодно избавляться от “лишних” признаков;
  - Имеете выборку, где объектов меньше, чем признаков.
- Гребневая регрессия не зануляет коэффициенты, а скорее старается уменьшить слишком большие. Этот метод подходит, если вы уверены, что все ваши независимые переменные будут иметь эффект на зависимую, пусть небольшой.

# Оценка качества

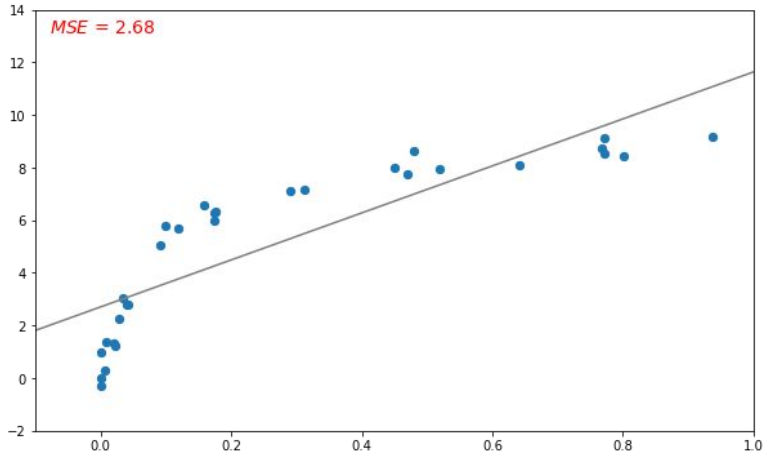
# Оценка качества: ошибки модели

Residuals = y_true - y_pred

Среднеквадратичная ошибка:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(true_i - pred_i)^2$$

$$ResidualSquaredError = \sqrt{MSE}$$

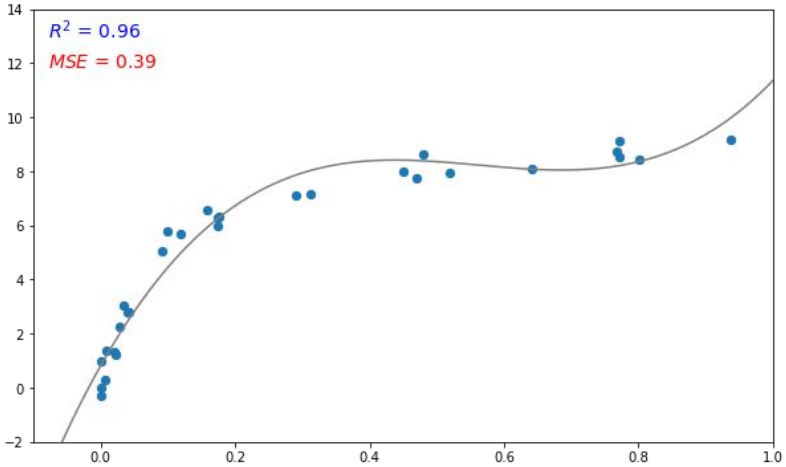


# Оценка качества

$R^2$, или коэффициент детерминации - насколько условная дисперсия модели отличается от дисперсии реальных значений.

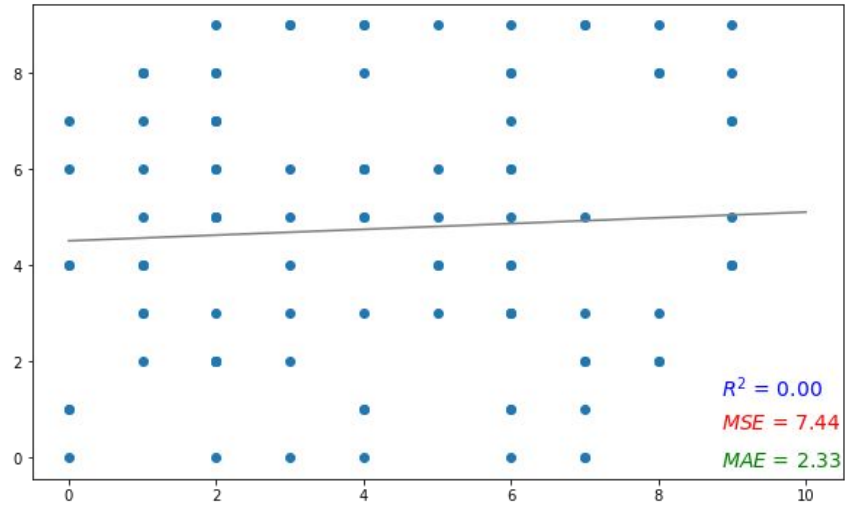
$$R^2 = 1 - \frac{\sum(true_i - pred_i)^2}{\sum(true_i - avg(true))^2}$$

pred = y_pred, true = y_true, avg - среднее (average).



# $R^2$

| | |
|---|---|
| $R^2 <= 0$ | Модель предсказывает значения так же или хуже, чем прямая линия |
| $R^2 > 0$ | Модель имеет какую-то предсказательную способность |
| $R^2 = 1$ | Модель идеально предсказывает всю выборку |

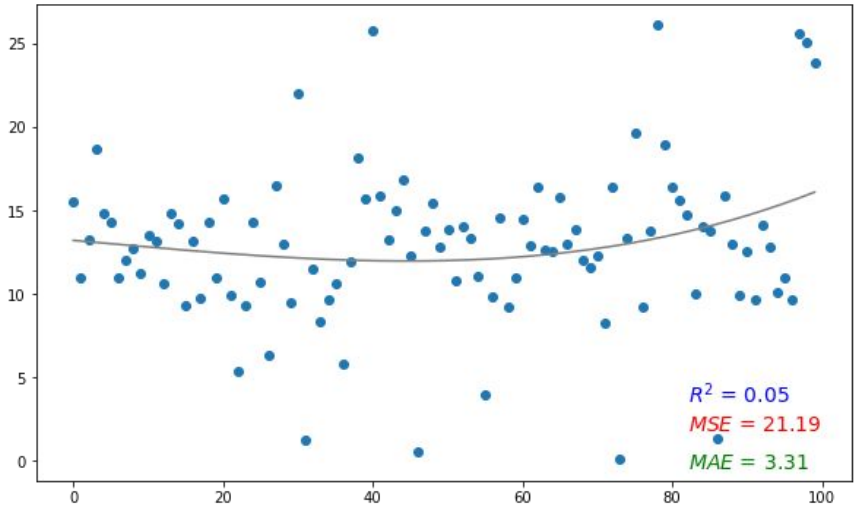


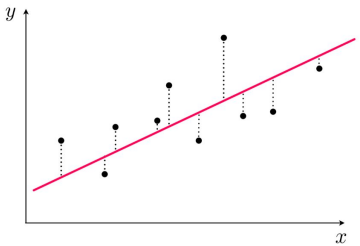
# Оценка качества: ошибки модели

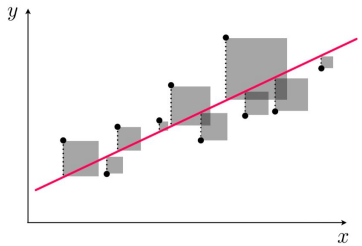
Есть и другие способы оценивать ошибки, например, средняя абсолютная ошибка:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|true_i - pred_i|$$

MAE более устойчива к выбросам.



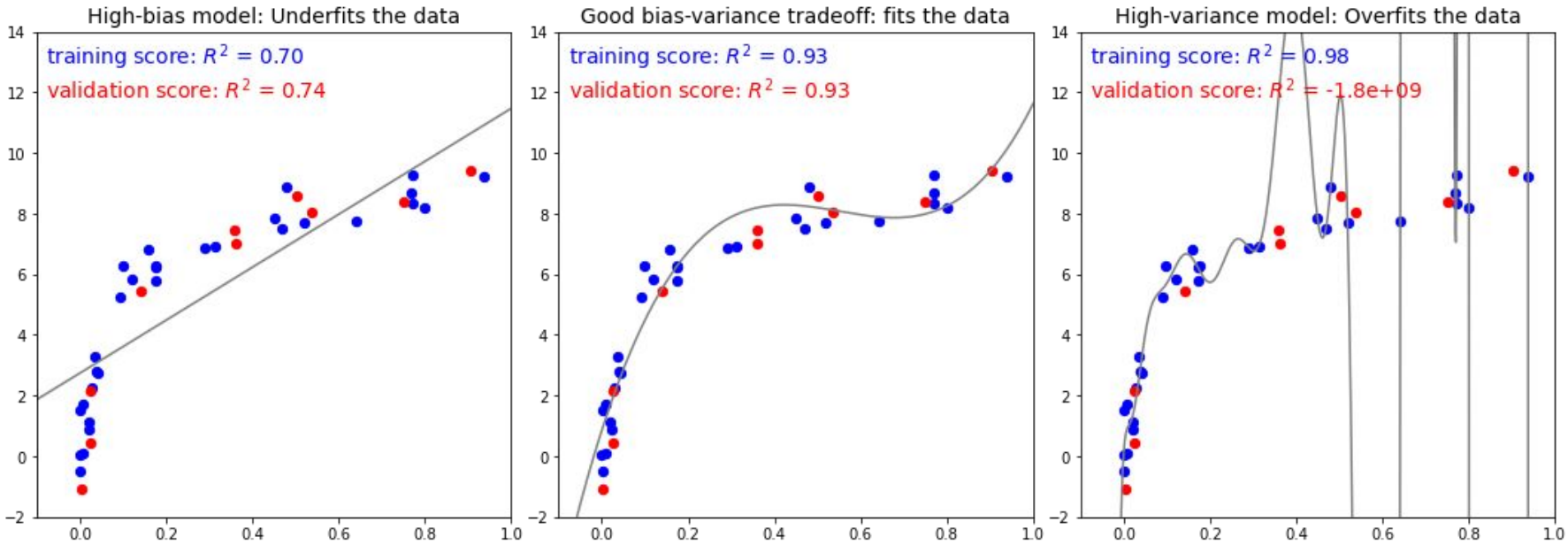



# Обучение и переобучение: терминология

- Underfitting: недообучение, недостаточная обобщающая способность модели;
- Overfitting: переобучение. Модель слишком хорошо выучивает тренировочные данные и теряет предсказательную способность на тестовых. В широком смысле переобучением называют любой случай, при котором качество предсказаний модели искусственно завышается;
- Bias: смещение. Показывает, насколько сильно средний ответ алгоритма отличается от «идеального предсказания»;
- Variance: дисперсия/разброс. Показывает чувствительность модели к изменениям в обучающих данных.

# Bias-variance tradeoff




Картинка нарисована по коду отсюда: https://jakevdp.github.io/PythonDataScienceHandbook/06.00-figure-code.html#Bias-Variance-Tradeoff-Metrics.
Степени полиномов: 1, 3, 20.

# Как предотвратить переобучение и недообучение

Переобучение:

- Регуляризовать модель;
- Добавить данных;
- Не позволять знаниям из тестового множества проникать в тренировочное. **Простейшее решение** - иметь три выборки: тренировочную, валидационную и тестовую, и никогда не менять модель, исходя только из результатов на тесте.

Недообучение:

- Дать модели больше времени;
- Увеличить набор параметров;
- Выбрать другой метод оптимизации или другую модель.