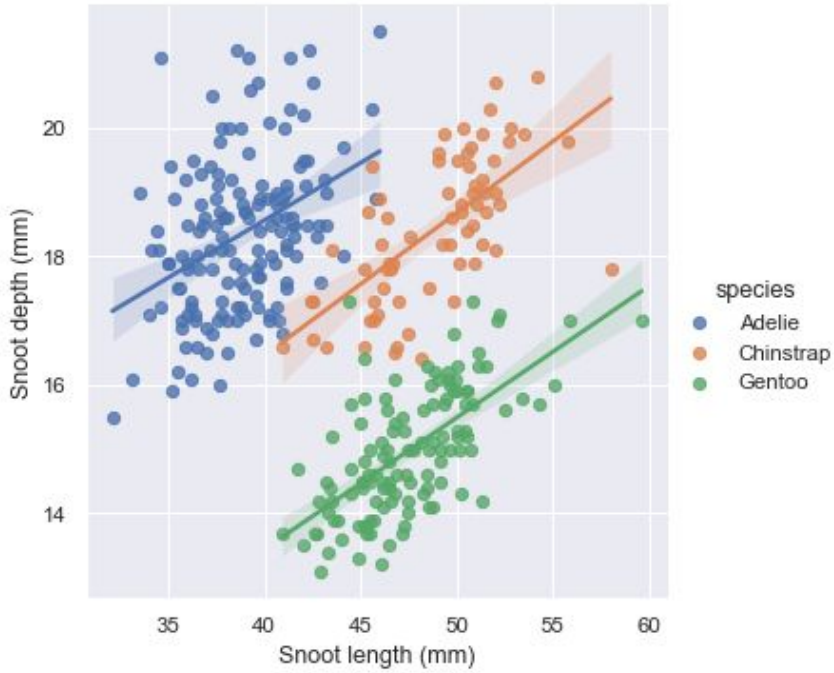
Регрессия

Регрессия

Задача: исследовать влияние независимых переменных на зависимую.

Примеры:

- Анализ данных айтрекинга (влияние различных свойств слова на длину фиксации);
- Влияние различных свойств текста на его читабельность;
- …



Картинка: Multiple linear regression in Seaborn, https://seaborn.pydata.org/examples/multiple_regression.html

Линейная регрессия

Задача: предсказать значение зависимой переменной.

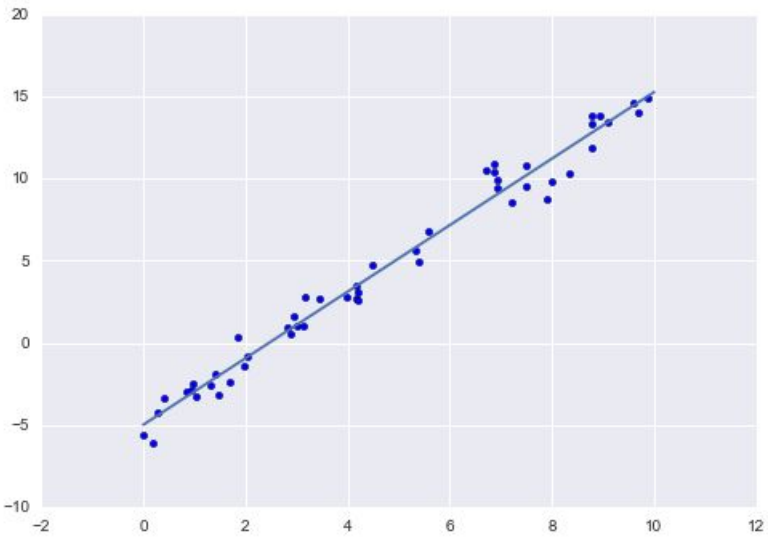
Функция в простейшей форме имеет вид:

$$y = wx + w_0$$

Где w - коэффициент, w_0 - intercept.

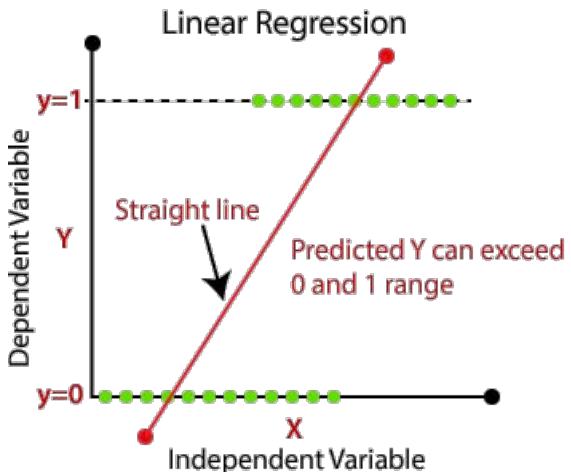
При увеличении количества признаков:

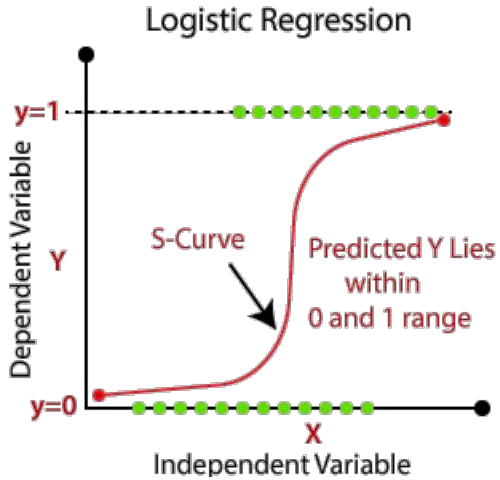
$$y = w_0 + w_1 x_1 + w_2 x_2 + [...]$$



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

Разница между линейной регрессией и логистической





y = -1, 0, 0.1, 0.12… 1, 2…

y = 0 | y = 1

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

Оптимизация

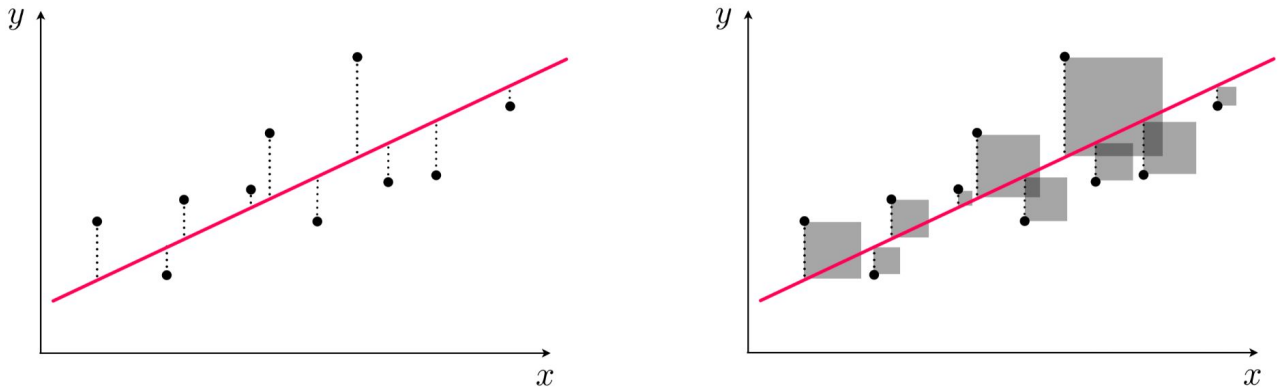
Как подобрать значения параметров?

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.

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

Как подобрать значения параметров?

Итак, мы посчитали значения нашей функции со случайными параметрами, получили значение ошибки. Но как его минимизировать? Куда двигаться?

Нужно найти точку, в которой лосс минимален. Это и будет оптимальным значением параметров.

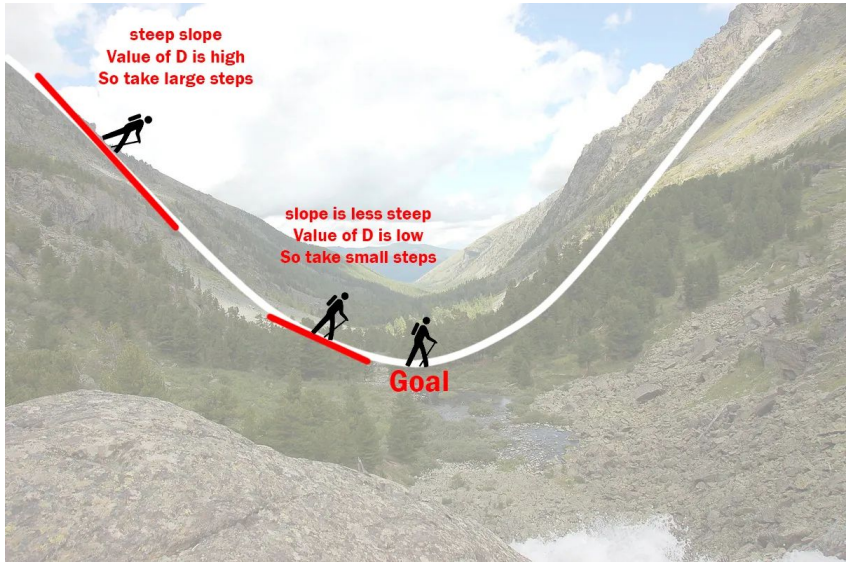
Таким образом, происходит **оптимизация.**

Одним из самых популярных алгоритмов оптимизации, т.е. поиска оптимальных значений параметров, является градиентный спуск.

Оптимизация: градиентный спуск

Идея: мы берем частную производную функции потерь по каждому из весов. Получаем градиент - вектор из частных производных. Потом к каждому весу прибавляем значение этой производной, но не всё полностью, а умноженное на некое небольшое число (learning rate), чтобы не проскочить минимум. Потом вычисляем новый лосс, пока не дойдем* до 0.

* стоит сказать, что на самом деле количество подобных проходов, или эпох, ограничено.



D - это значение частной производной, или крутизна склона, learning rate - скорость движения, D*lr - длина шага.

Картинка: https://towardsdatascience.com/linear-regression-using-gradient-descent-97a6c8700931

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;
- …

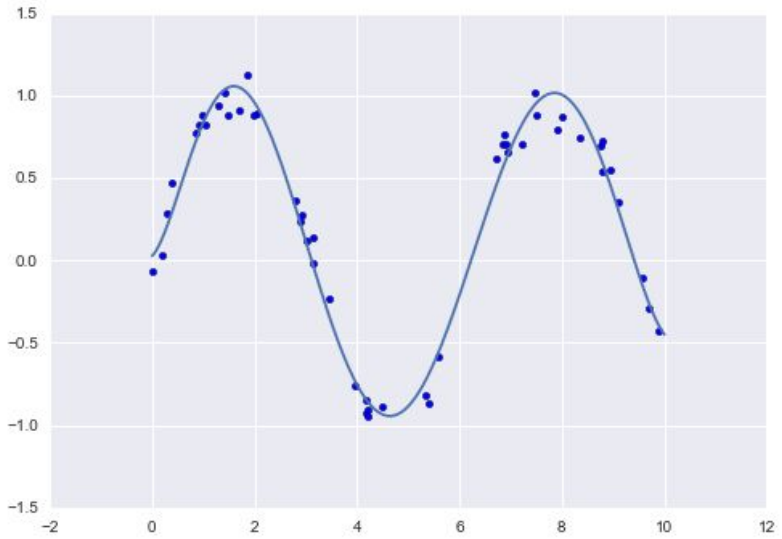
Полиномиальная регрессия

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 линией.



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

Полиномиальная регрессия

Как решается такое выражение?

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

Ответ: признаки преображаются в новые признаки:

$$z_n = [x_1, x_2^2, [...], x_n^n]$$

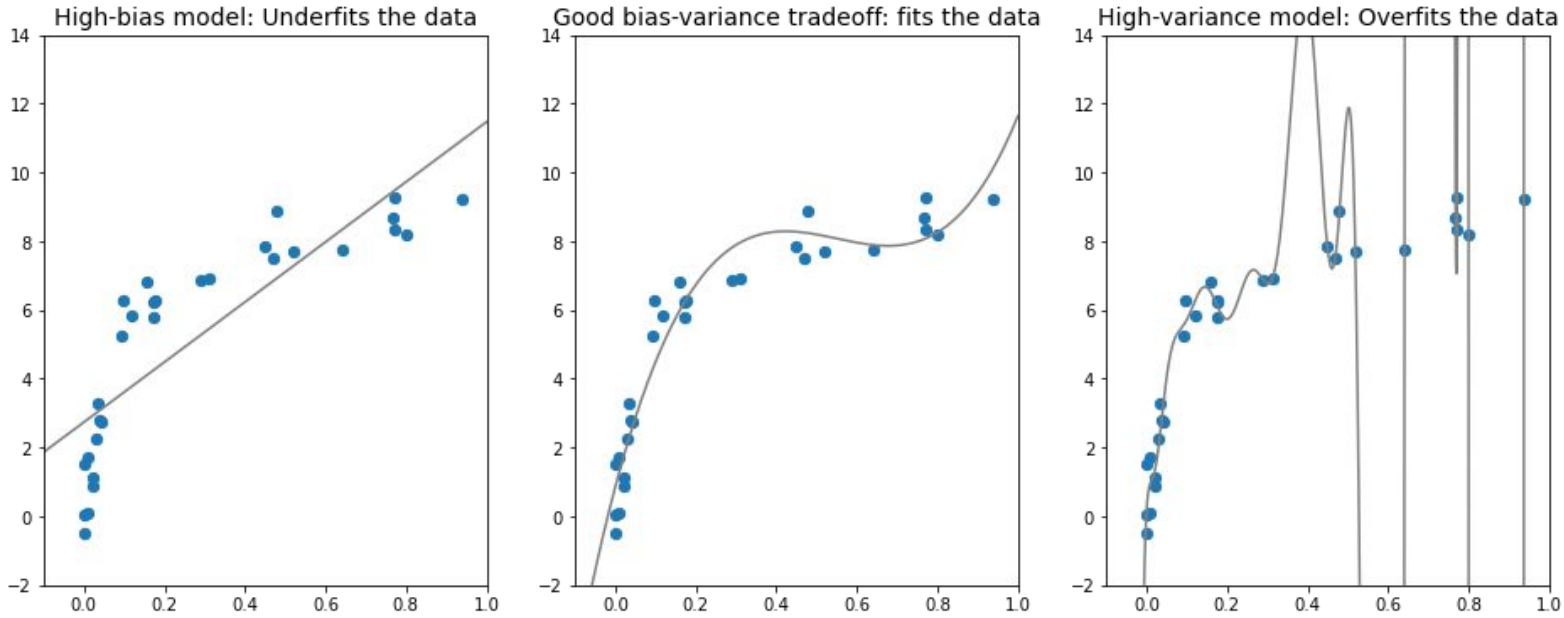
$$y = w_0 + w_1 z_1 + w_2 z_2 + [...] + w_i z_n$$

Валидация - общая терминология

Обучение и переобучение

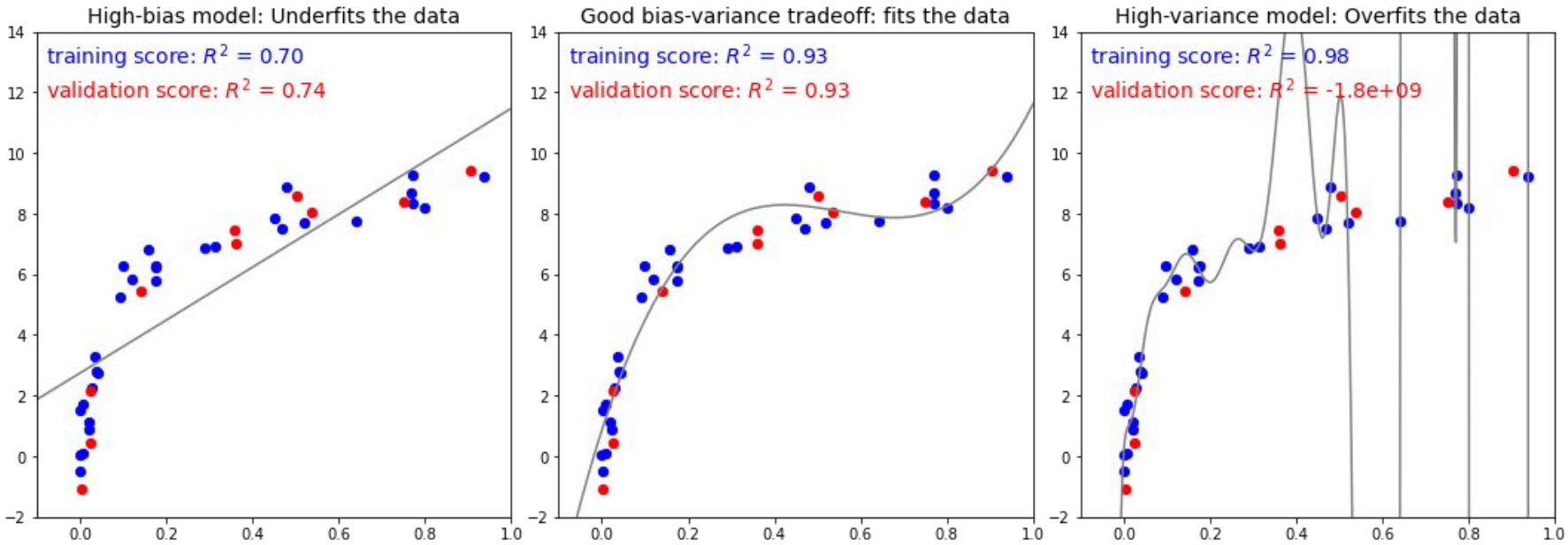
- Underfitting: недообучение, недостаточная обобщающая способность модели;
- Overfitting: переобучение. Модель слишком хорошо выучивает тренировочные данные и теряет предсказательную способность на тестовых. В широком смысле переобучением называют любой случай, при котором качество предсказаний модели 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. Степени полиномов: 1, 3, 20.

Bias-variance tradeoff



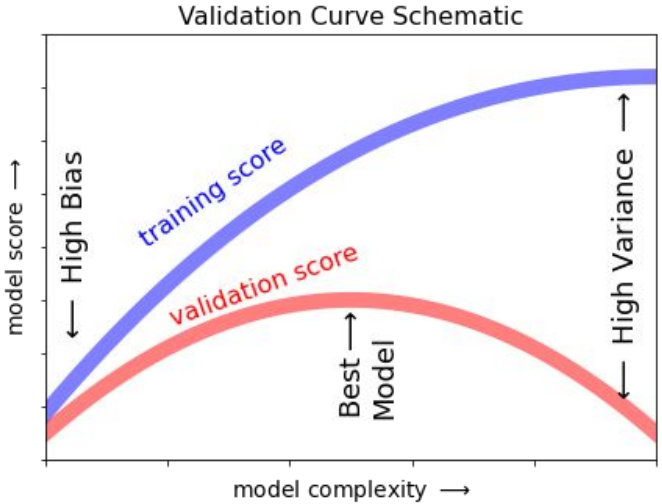
Картинка нарисована по коду отсюда: https://jakevdp.github.io/PythonDataScienceHandbook/06.00-figure-code.html#Bias-Variance-Tradeoff-Metrics.
Степени полиномов: 1, 3, 20.

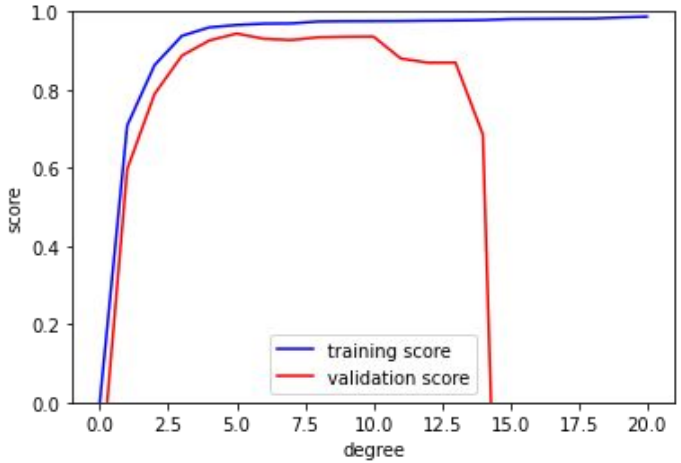
Зачем это знать?

Чтобы найти лучшую модель.

- При большом смещении качество во время тренировки близко к качеству на валидации, но, как правило, достаточно мало́;
- При большой дисперсии качество на тренировочной выборке растет, а на валидационной со временем уменьшается.

=> Нужно выбирать что-то среднее.





Код картинок:
https://jakevdp.github.io/PythonDataScienceHandbook/06.00-figure-code.html#Validation-Curve;
https://jakevdp.github.io/PythonDataScienceHandbook/05.03-hyperparameters-and-model-validation.html#Learning-curves-in-Scikit-Learn

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.

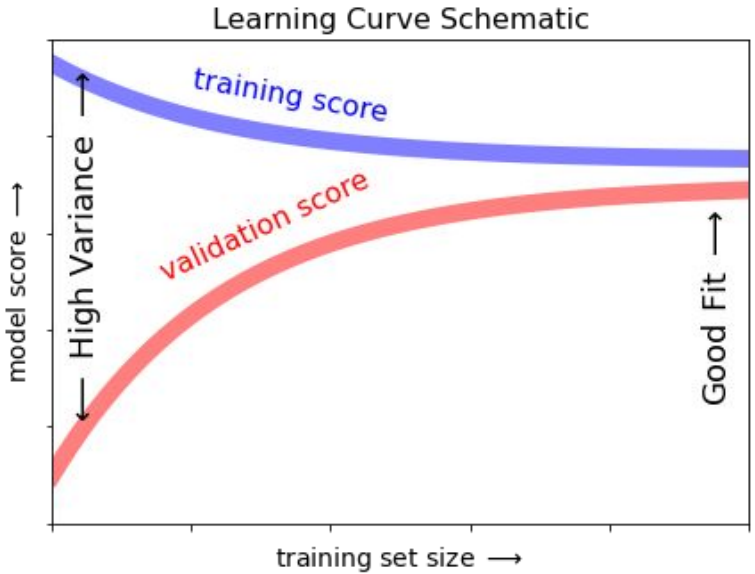
Недообучение:

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

Как понять, что данных достаточно?*

По learning curve, тренировочной кривой.

После прохождения точки, в которой валидационная и тренировочная кривая сближаются, добавление новых данных больше не будет улучшать модель.



Код картинки:
https://jakevdp.github.io/PythonDataScienceHandbook/06.00-figure-code.html#Learning-Curve

Регуляризация - для регрессий

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).

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 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.

y - average cat jump length

x1 - age, x2 - color, x3 - kind, … x20 - favorite food

$y = w_1x_1 + w_2x_2 + \ldots + w_{20}x_{20}$

Оценка качества

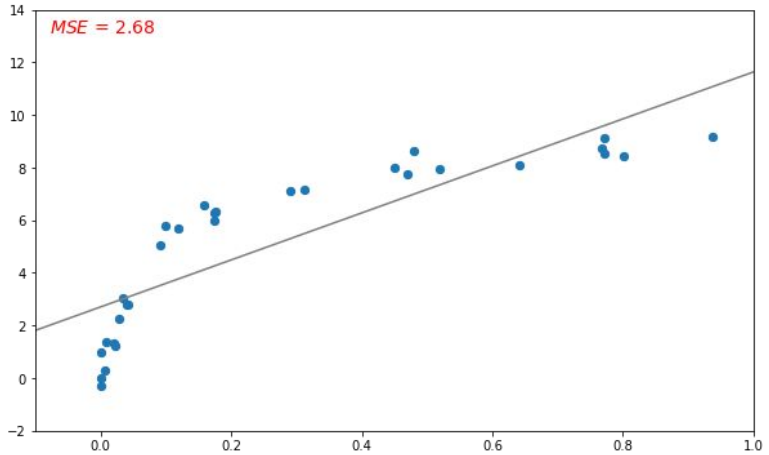
Оценка качества: ошибки модели

Residuals = y_true - y_pred

Среднеквадратичная ошибка:

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

$$ResidualSquaredError = \sqrt{MSE}$$

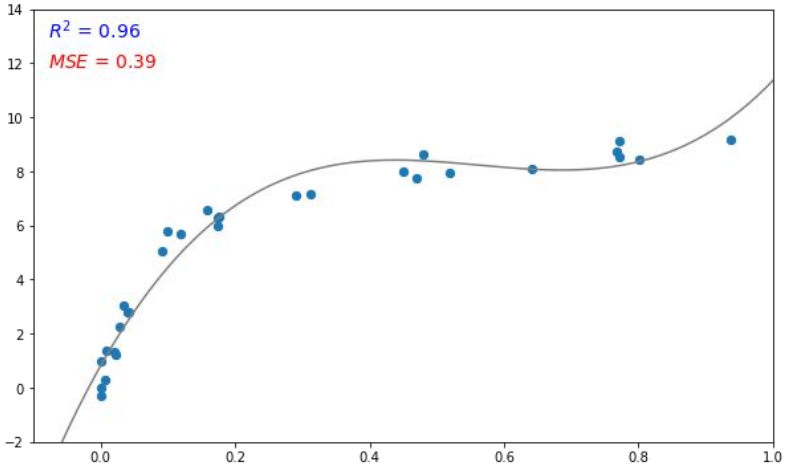


Оценка качества

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

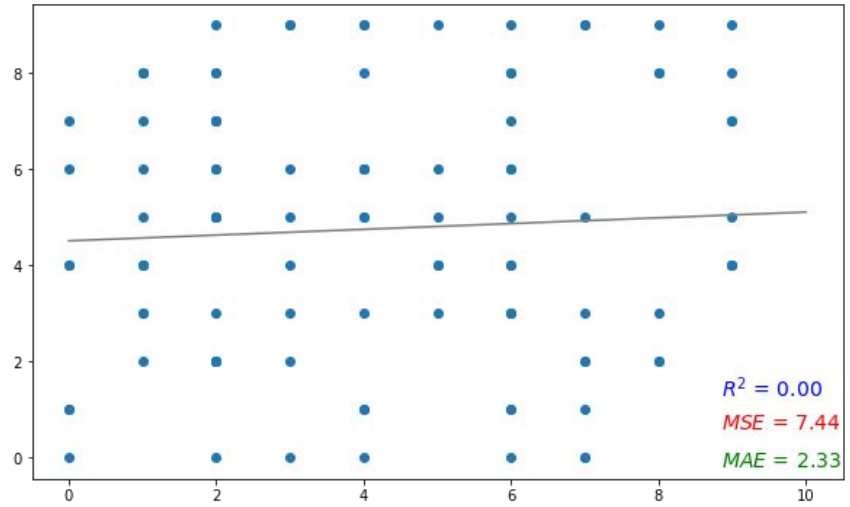
$$R^2 = 1 - \frac{\sum(true_i - pred_i)^2}{\sum(true_i - avg(true))^2}$$

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

Недостаток R^2: возрастает с увеличением числа предикторов (признаков).

Решение: Adjusted R^2:

$$Adj\,R^2 = 1 - (1 - R^2)\frac{k-1}{k-n-1}$$

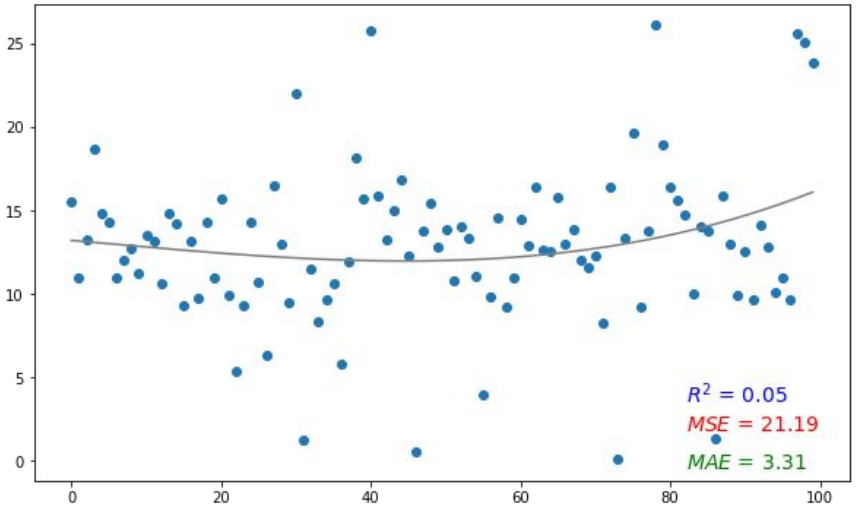
n - количество предикторов, k - количество наблюдений.

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.



F-статистика и p-value*

Напоминание:

- F-статистика позволяет понять, является ли группа переменных статистически значимой, измеряя их совместный эффект на зависимую переменную;
- p-value позволяет понять, можно ли отвергнуть нулевую гипотезу.

В sklearn нет простого способа посчитать эти статистики для уже готовой регрессии, но есть класс, который позволяет посчитать их отдельно: https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.f_regression.html

Практика:
https://colab.research.google.com/drive/12td3H9NcCExNvImbGGtgaMEIpLtHQ-9c?usp=sharing

Дополнительно

- https://pozdniakov.github.io/tidy_stats/340-lm.html
- https://colab.research.google.com/github/fbeilstein/machine_learning/blob/master/lecture_09_linear_regression.ipynb#scrollTo=Mz3BqxXrcnlk