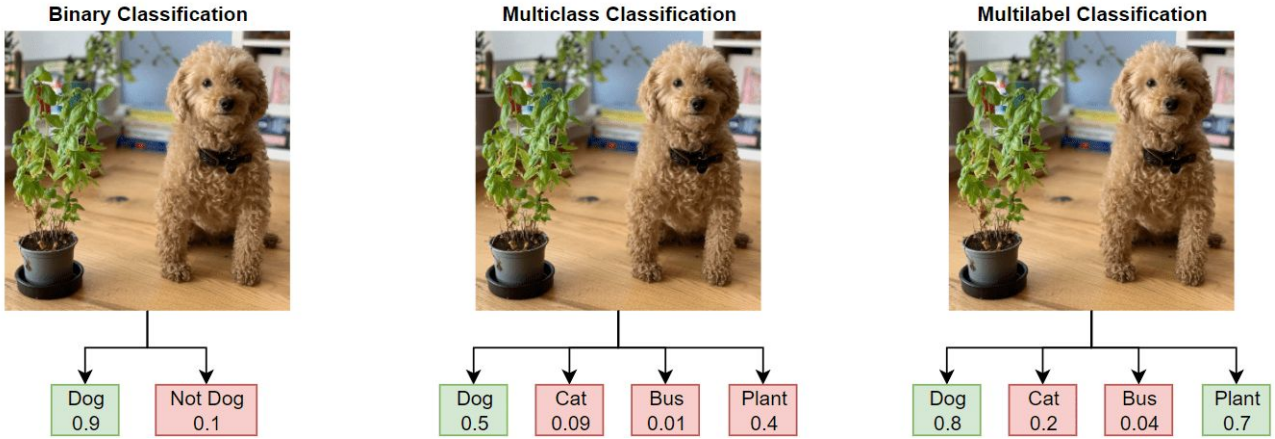
Классификация

В основе - презентация Анны Дмитриевой

Виды классификации

Мультикласс, мультилейбл, мультиаутпут



Картинка: www.mathworks.com

Стратификация и вес классов

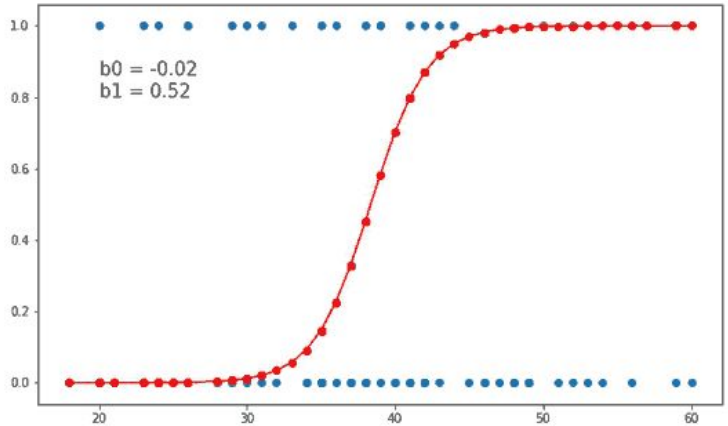
- Стратификация в машинном обучении - это разделение набора данных на выборки (тренировочную, *валидационную (!)*, тестовую) таким образом, что во всех выборках **соотношение** классов остается одинаковым
- В sklearn производится с помощью встроенных классов и параметров (например, параметр **stratify** функции **train_test_split**).
- Можно настроить и нужный вам вес классов или задать автоматический баланс классов (напр., в логистической регрессии): **class_weight="balanced"**

Примеры классификаторов

Логистическая регрессия

В sklearn это линейная модель для классификации, которая возвращает вероятности каждого класса.

$$P(Y = 1|x_1, x_2, [...]x_n) = \frac{e^{(w_0+w_1x_1+w_2x_2+[...]+w_nx_n)}}{1 + e^{(w_0+w_1x_1+w_2x_2+[...]+w_nx_n)}}$$

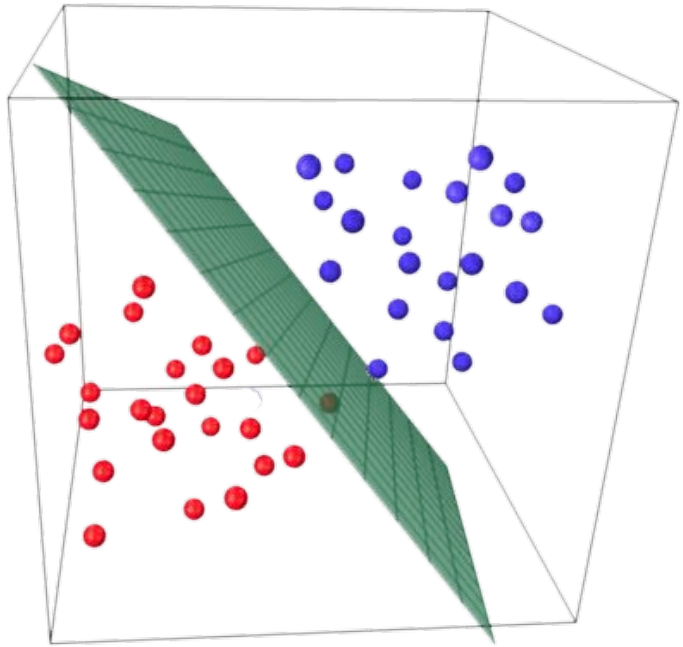


Здесь w - коэффициенты/веса (внимание: на картинке они обозначены буквой b), w_0 - константа, $x_1 \ldots x_n$ - наши признаки, e - экспонента

Картинка: https://towardsdatascience.com/logistic-regression-explained-and-implemented-in-python-880955306060

Немного о том, как это работает

Нам необходимо разделить точки в пространстве гиперплоскостью.



Картинка: https://habr.com/ru/companies/io/articles/265007/

К ближайших соседей

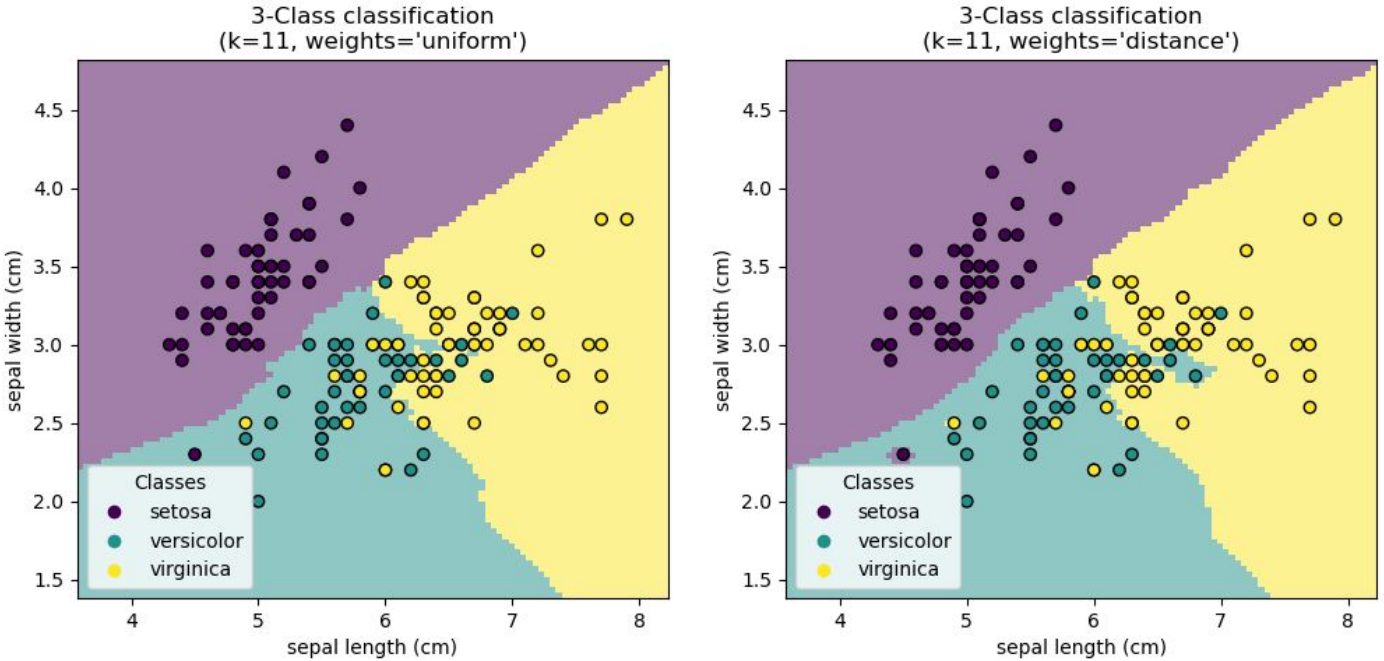
- В многомерном пространстве существует множество векторов признаков $X_{1...i}$. Каждый из них относится к одному из N классов. Предполагается, что вектора одного класса будут располагаться рядом. Когда нам нужно определить, к какому классу относится каждый новый вектор, мы проецируем его в то же пространство и смотрим, кто его соседи.
- k nearest neighbors (kNN) может предсказывать как непрерывные, так и категориальные переменные, т.е. может использоваться как для регрессии, так и для классификации.

К ближайших соседей

Простое решение: каждый новый вектор сравниваете со всем набором тренировочных векторов, каждый раз вычисляя расстояние между векторами. Затем сортируем вектора по расстоянию от нового и берем k ближайших.

- При регрессии: значением функции будет среднее от ближайших векторов;
- При классификации: модель выдаст тот класс, который преобладает среди k ближайших векторов.

К ближайших соседей

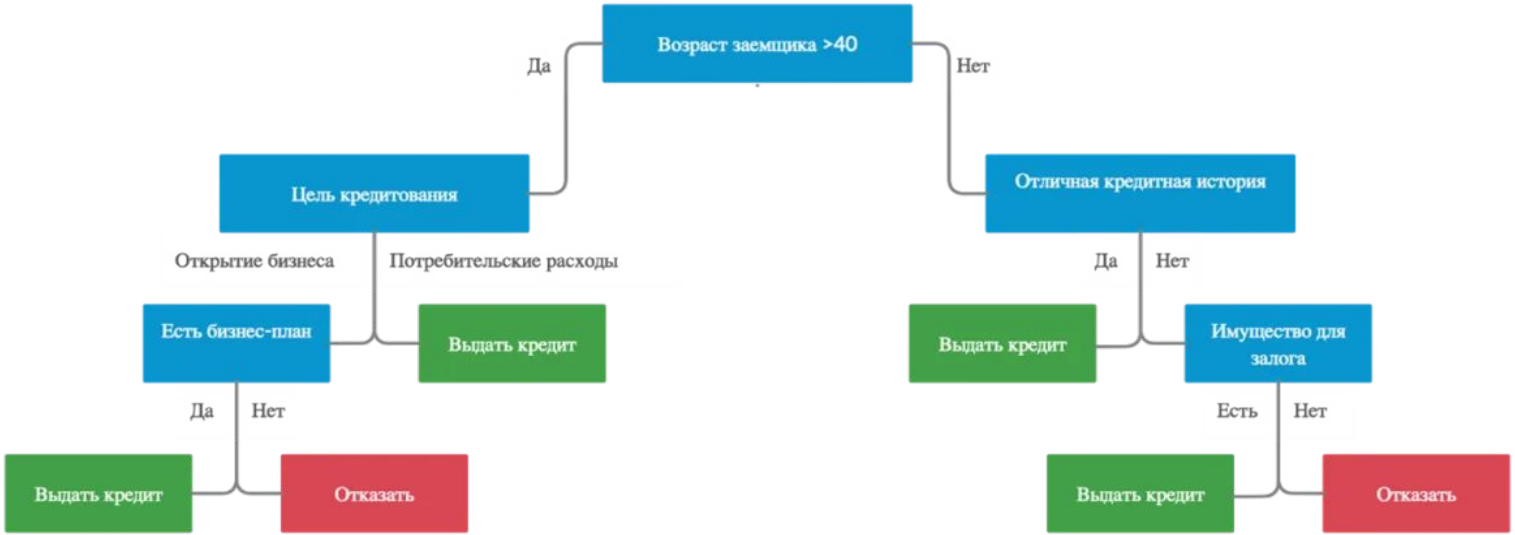


Картинка: kNN-классификация на датасете с ирисами с разными весами. https://scikit-learn.org/stable/auto_examples/neighbors/plot_classification.html

Деревья решений

Подробнее рекомендую здесь: https://mipt-stats.gitlab.io/courses/ad_fivt/trees.html

Дерево решений представляет собой иерархическую древовидную структуру, состоящую из правила вида «Если …, то ...». Правила генерируются автоматически в процессе обучения.



Суть алгоритма

Процесс разбиения узлов продолжают до того, пока все узлы в конце ветвей не станут листами.

Узел становится листом в двух случаях:

- естественным образом — когда он содержит единственный объект или объект только одного класса;
- после достижения заданного условия остановки алгоритм — например, минимально допустимое число примеров в узле или максимальная глубина дерева.

Суть алгоритма

В основе построения лежат «жадные» алгоритмы, допускающие локально-оптимальные решения на каждом шаге (разбиения в узлах), которые приводят к оптимальному итоговому решению. То есть при выборе одного атрибута и произведении разбиения по нему на подмножества, алгоритм не может вернуться назад и выбрать другой атрибут, даже если это даст лучшее итоговое разбиение.

Популярные алгоритмы, используемых для обучения деревьев решений, строятся на базе принципа «разделяй и властвуй».

Методы оценки качества

Оценка качества классификации

Confusion matrix		True labels	
		Positive	Negative
Predicted labels	Positive	True positive	False positive
	Negative	False negative	True negative

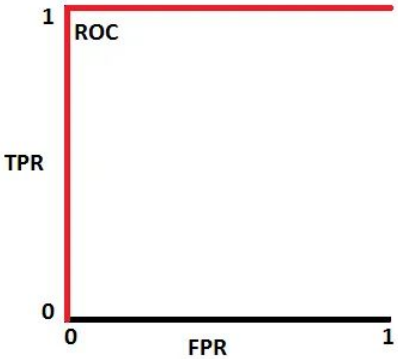
$$Precision = \frac{TP}{TP + FP} \qquad Recall = \frac{TP}{TP + FN}$$

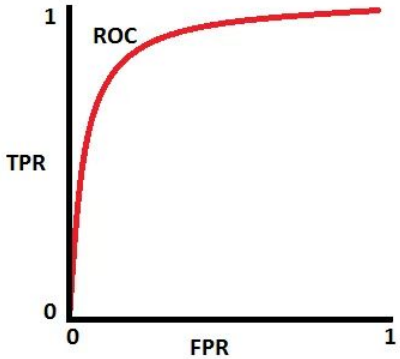
$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

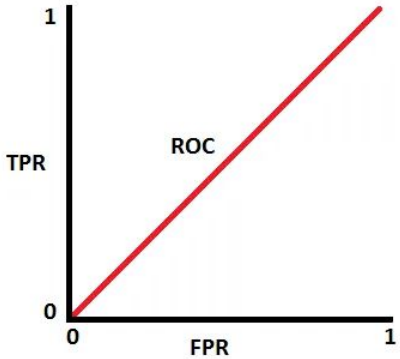
$$F1\ Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

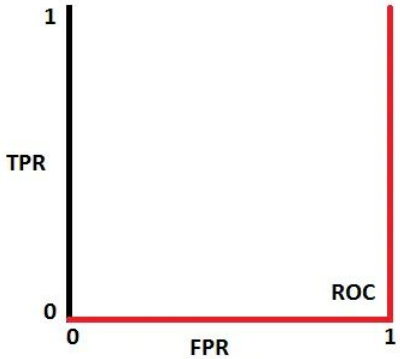
Картинка: Seol, Da & Choi, Jeong & Kim, Chan & Hong, Sang. (2023). Alleviating Class-Imbalance Data of Semiconductor Equipment Anomaly Detection Study. Electronics. 12. 585. 10.3390/electronics12030585.

ROC curve







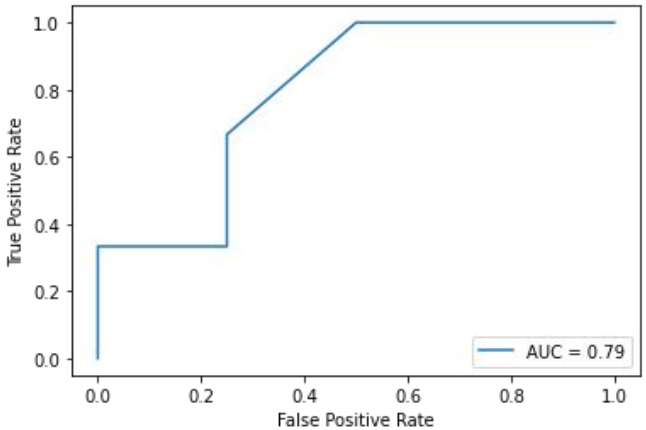


обычно принимают за пороговое значение

Картинки: https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5

AUC

- AUC = area under the curve, площадь пространства между кривой и осью false positive rate. AUC идеальной модели должна приближаться к 1. AUC=0.5 означает, что модель не умеет разделять классы

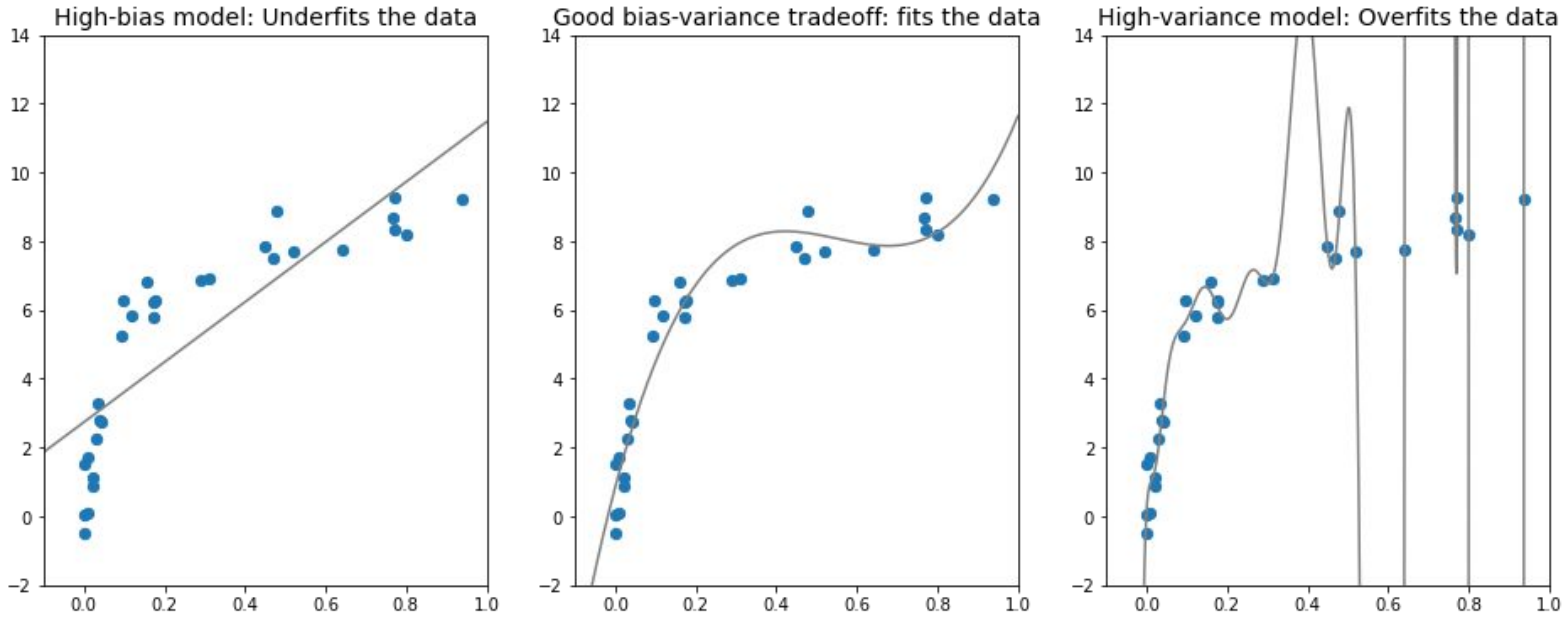


Валидация и подбор гиперпараметров

Обучение и переобучение: терминология

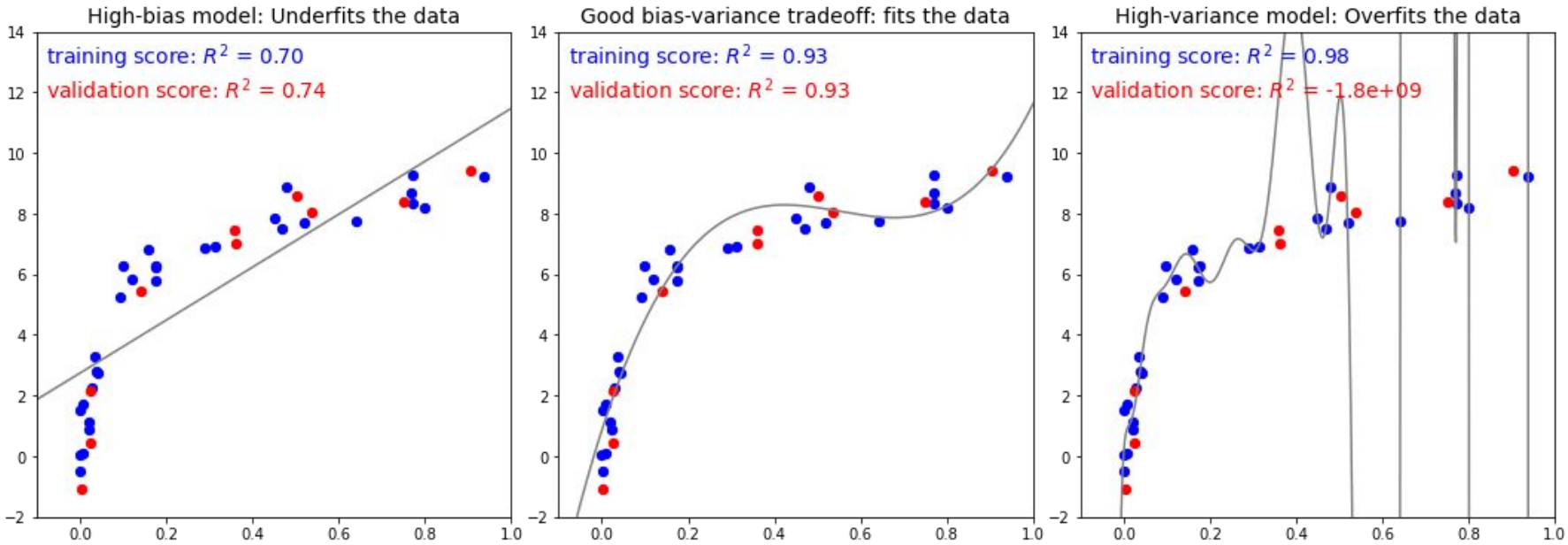
- Underfitting: недообучение, недостаточная обобщающая способность модели;
- Overfitting: переобучение. Модель слишком хорошо выучивает тренировочные данные и теряет предсказательную способность на тестовых. В широком смысле переобучением называют любой случай, при котором качество предсказаний модели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

Bias-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.

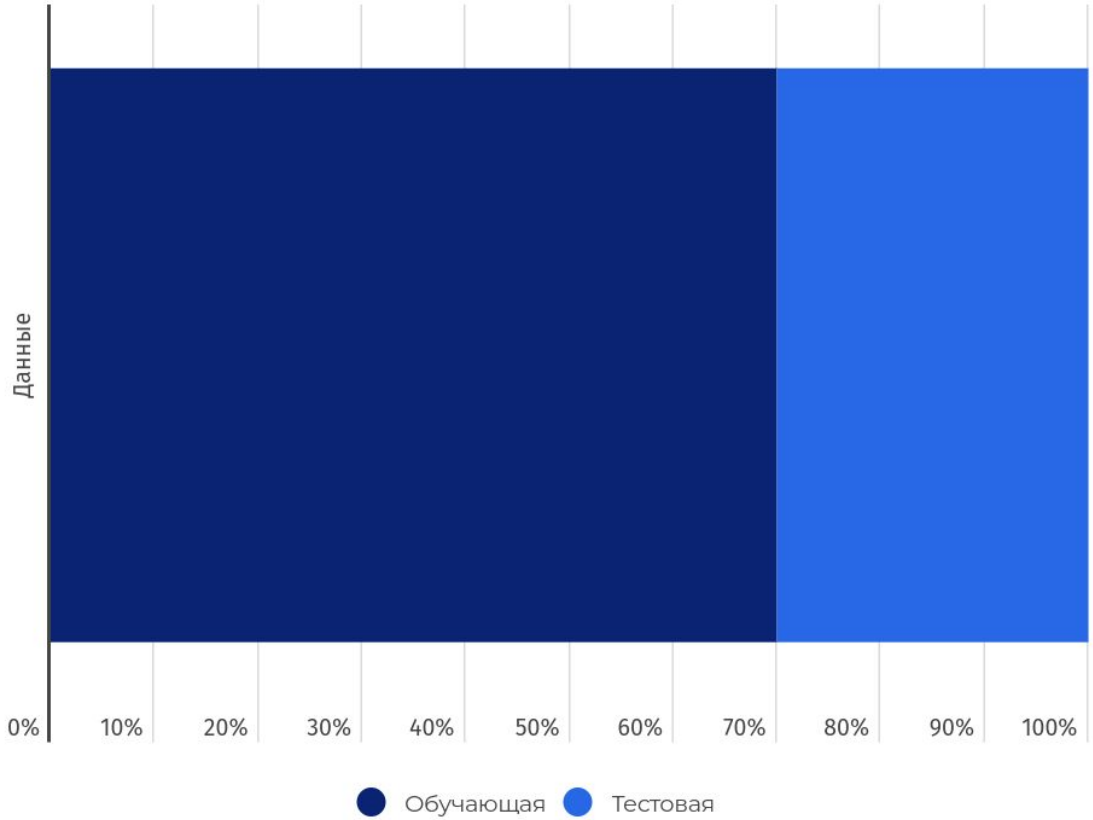
Bias-variance tradeoff



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

Кросс-валидация

предсказываем



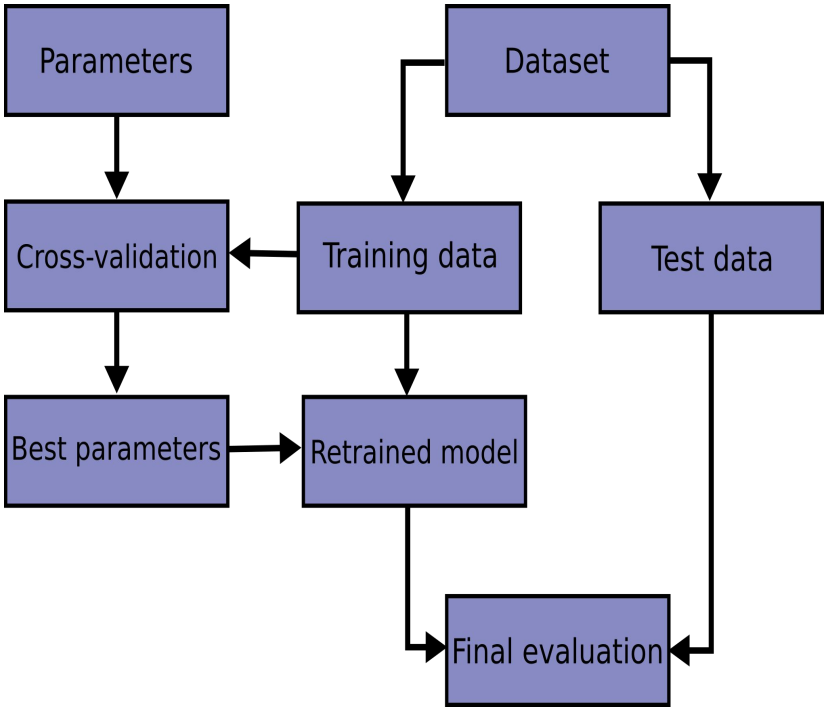
Обычно мы делим наши данные на обучающую и тестовую выборку примерно так

Кросс-валидация

Идея: мы извлекаем максимум из наших данных, итеративно улучшая параметры модели, и при этом избегаем переобучения.

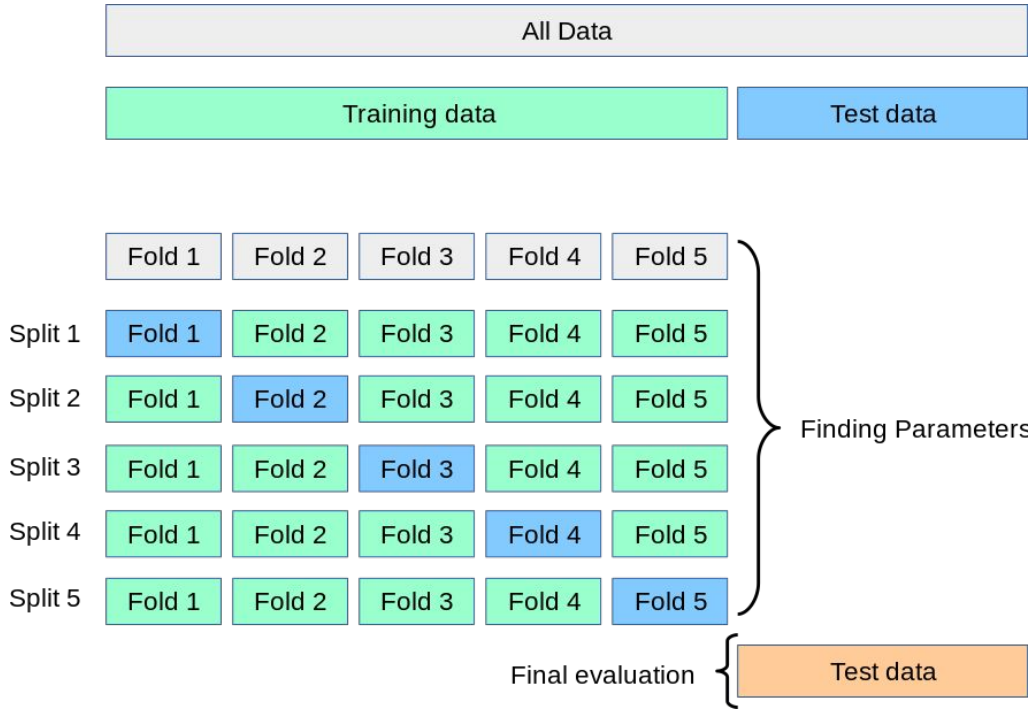
Как мы это делаем:

- Делим тренировочную выборку на К кусочков;
- Используем К-1 кусочков для тренировки, 1 для валидации;
- Повторяем К раз.



Картинка: https://scikit-learn.org/stable/modules/cross_validation.html

Кросс-валидация



Картинка: https://scikit-learn.org/stable/modules/cross_validation.html

Подбор гиперпараметров

- Параметры настраиваются в процессе обучения модели на данных. Например, веса в линейной регрессии, нейросетях, структура решающего дерева;
- Гиперпараметры — это характеристики модели, которые фиксируются до начала обучения: глубина решающего дерева, значение силы регуляризации в линейной модели (очень грубо: это те дополнительные атрибуты модели, которые мы можем указать в скобках при ее загрузке)

Как найти оптимальные значения гиперпараметров?

Самый простой способ - перебрать все возможные комбинации.

Grid Search:

- Для каждого гиперпараметра фиксируется несколько значений;
- Перебираются все комбинации значений различных гиперпараметров, на каждой из этих комбинаций модель обучается и тестируется;
- Выбирается комбинация, на которой модель показывает лучшее качество.