Naive Bayes. SVM. Регуляризация. Ансамбли

Маша Шеянова, masha.shejanova@gmail.co

Ещё несколько классификаторов

Наивный Байесовский классификатор

Города и сёла

Возьмём 1000 случайных человек и спросим их, где они живут: в городе или в селе.

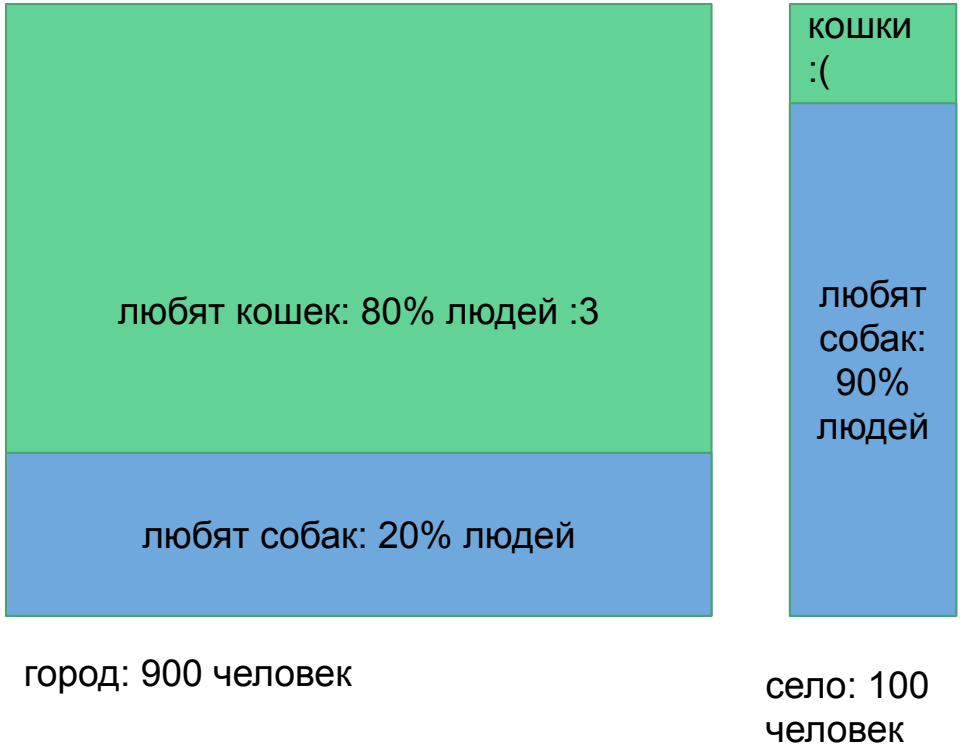
В городе живёт больше людей, поэтому в нашей выборке таких оказалось 900, а из села — всего 100.

Кошки и собаки

А теперь расспросим их, кого они бы предпочли в качестве домашнего питомца: кошек или собак.

Оказалось, что в сёлах собак любят гораздо больше, а в городе предпочитают кошек.

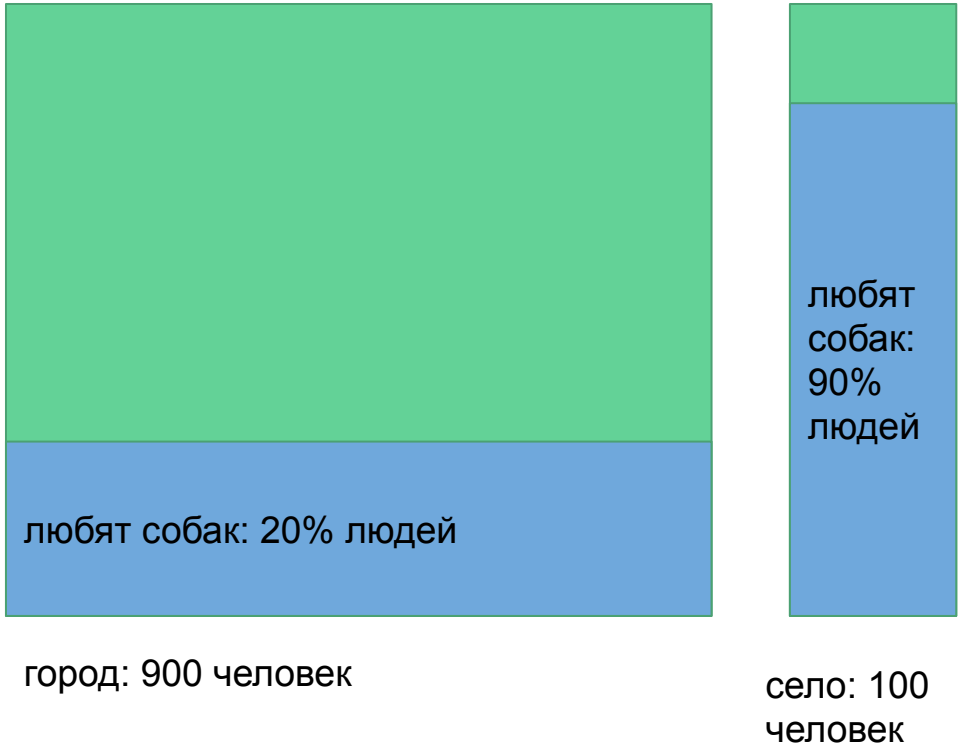
Пусть событие A = человек из села. Событие B = человек любит собак.



Вероятность события B

Мы встретили случайного человека из нашей тысячи.

С какой вероятностью он любит собак?



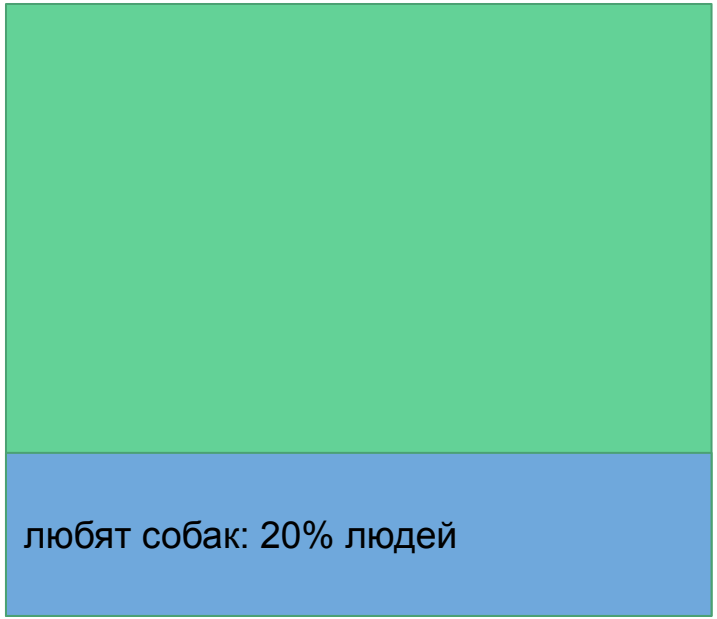
Вероятность события B

Мы встретили случайного человека из нашей тысячи. С какой вероятностью он любит собак?

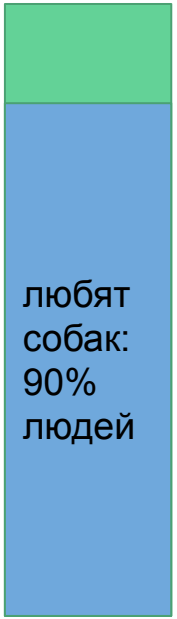
P(городской и собачник) = 900/1000 * 0.1 = 0.9 * 0.2 = 0.18

P(сельский и собачник) = 100/1000 * 0.1 = 0.1 * 0.9 = 0.09

P(собачник) = 0.18 + 0.09 = 0.27



город: 900 человек



село: 100 человек

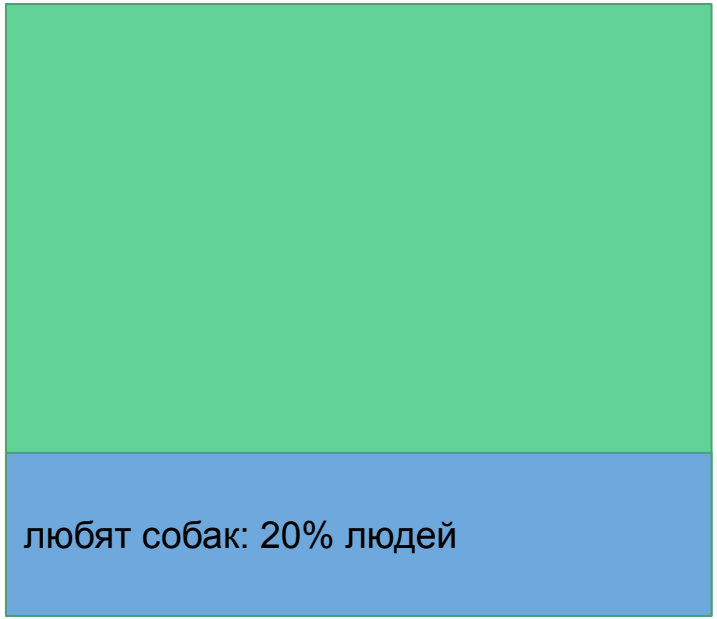
Вероятность события B

Мы встретили случайного человека из нашей тысячи. С какой вероятностью он любит собак?

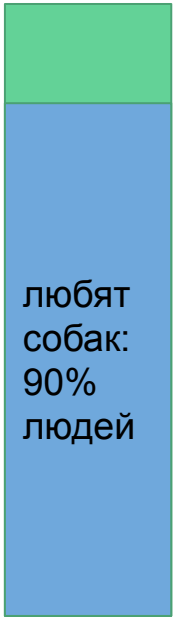
P(городской и собачник) = 900/1000 * 0.1 = 0.9 * 0.2 = 0.18

P(сельский и собачник) = 100/1000 * 0.1 = 0.1 * 0.9 = 0.09

P(собачник) = 0.18 + 0.09 = 0.27



город: 900 человек

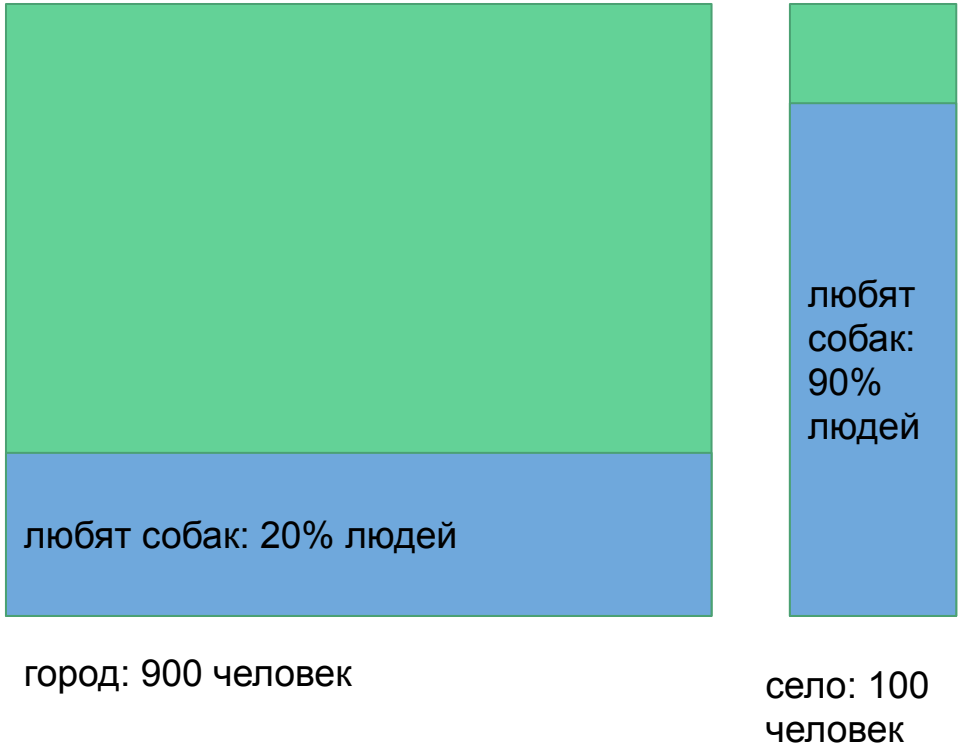


село: 100 человек

Вероятность события A

Мы встретили случайного человека из нашей тысячи, и оказалось, что он любит собак.

С какой вероятностью он живёт в селе?



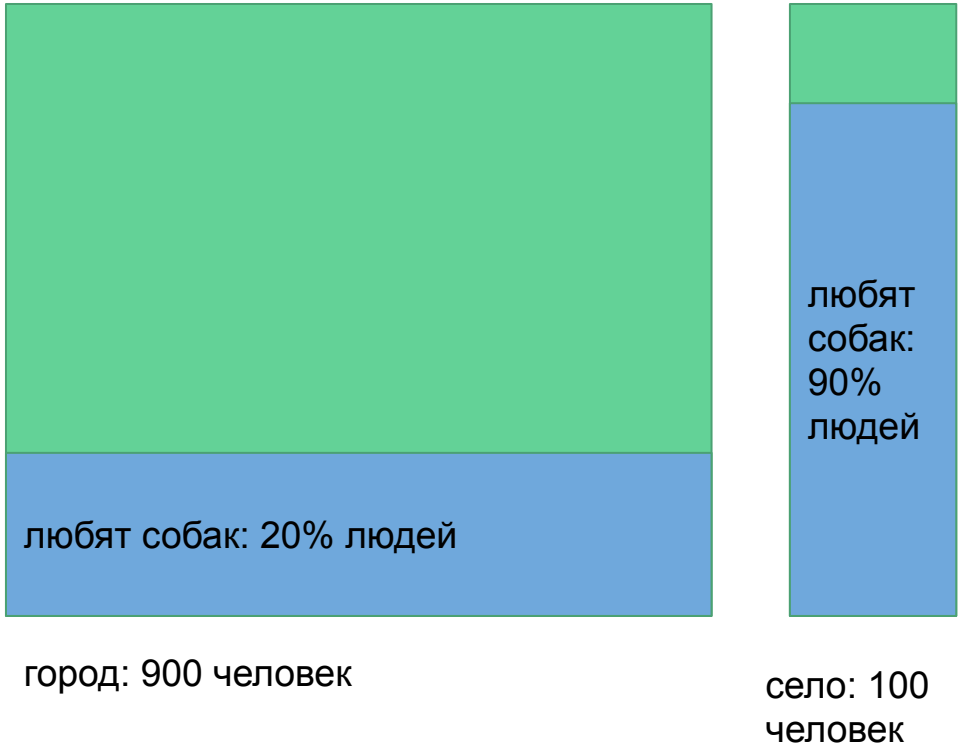
Вероятность события A

Случайный человек из нашей тысячи любит собак. С какой вероятностью он из села?

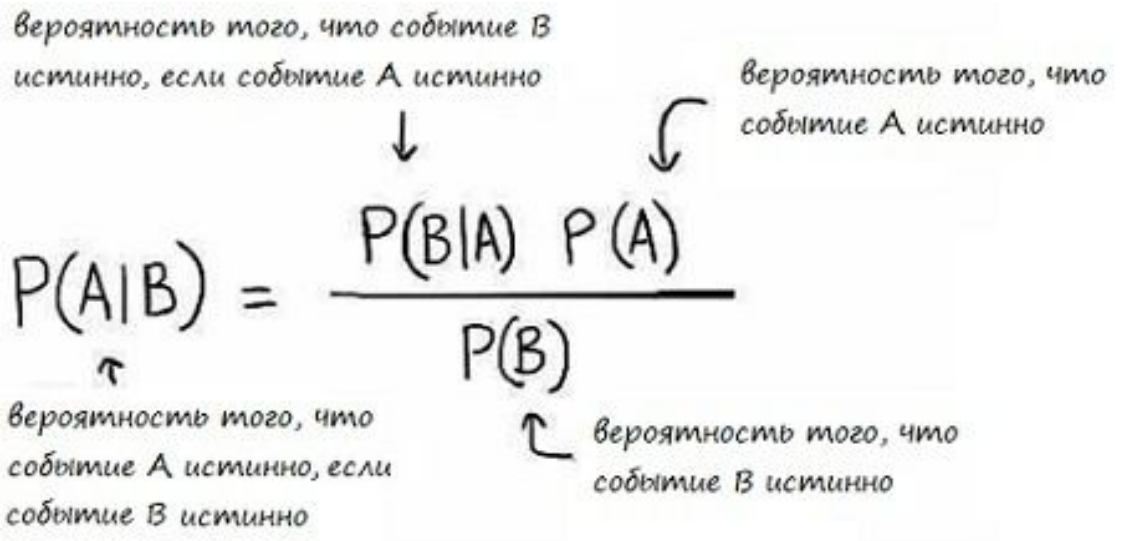
P(сельский и собачник) = 100/1000 * 0.1 = 0.1 * 0.9 = 0.09

P(собачник) = 0.18 + 0.09 = 0.27

P(сельский | собачник) = P(сельский и собачник) / P(собачник) = 0.09/0.27 = 1/3



Формула Байеса



(Источник картинки)

если P(B) не дано изначально, его можно расписать как

P(B|A)*P(A) + P(B|не A)*P(не A)

Определение спама

На почту студента Вани в 7 % случаев приходит спам. В обычных входящих письмах для Вани слово *вклады* встречается в 5% писем. В спаме, который приходит к Ване, это слово встречается в 60% случаев.

К Ване пришло письмо, в котором есть слово *вклады*. С какой вероятностью это нормальное письмо (не спам)?

Определение спама

На почту студента Вани в 7 % случаев приходит спам. В обычных входящих письмах для Вани слово *вклады* встречается в 5% писем. В спаме, который приходит к Ване, это слово встречается в 60% случаев.

К Ване пришло письмо, в котором есть слово *вклады*. С какой вероятностью это нормальное письмо (не спам)?

P(норм | *вклады*) = P(*вклады* | норм) * P(норм) / P(*вклады*)

Определение спама

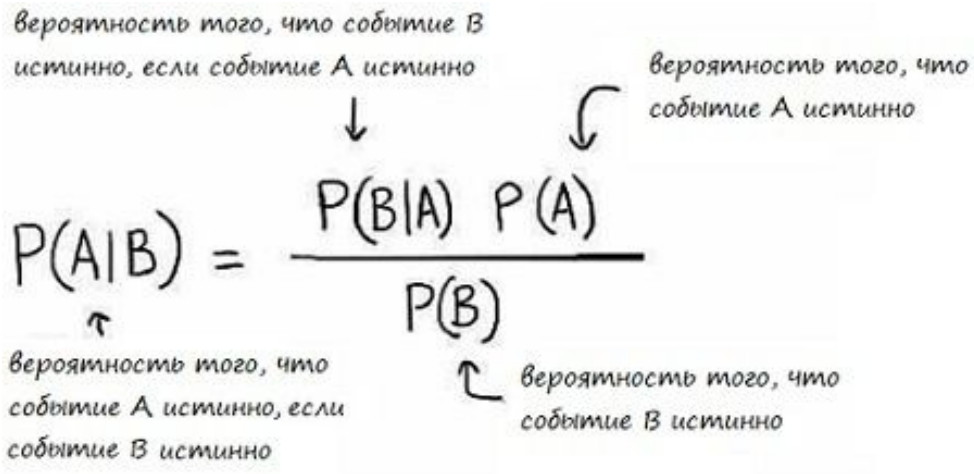
На почту студента Вани в 7 % случаев приходит спам. В обычных входящих письмах для Вани слово *вклады* встречается в 5% писем. В спаме, который приходит к Ване, это слово встречается в 60% случаев.

К Ване пришло письмо, в котором есть слово *вклады*. С какой вероятностью это нормальное письмо (не спам)?

P(*вклады*) = P(*вклады* | норм) * P(норм) + P(*вклады* | спам) * P(спам) = **0.05 * 0.93** + **0.6 * 0.07** = **0.0465** + **0.042** = **0.0885**

P(норм письмо| *вклады*) = P(*вклады* | норм письмо) * P(норм) / P(*вклады*) = **0.0465** / **0.0885 = 0.525423729**

Наивный Байесовский классификатор



Итак, формула Байеса — это про то, **насколько вероятно наше предположение о мире при условии того, что мы видим**.

На этой идее и держится наивный Байесовский классификатор: насколько вероятно, что событие A (класс) произошло при условии события B (признака).

Реализации в питоне

- Gaussian Naive Bayes — для непрерывных признаков (float)
- Multinomial Naive Bayes — для дискретных признаков (int)
- Bernoulli Naive Bayes — для бинарных признаков (0 или 1)

Наивный Байес ≠ байесовские методы!

Наивный Байес — самый простой классификатор в машинном обучении.

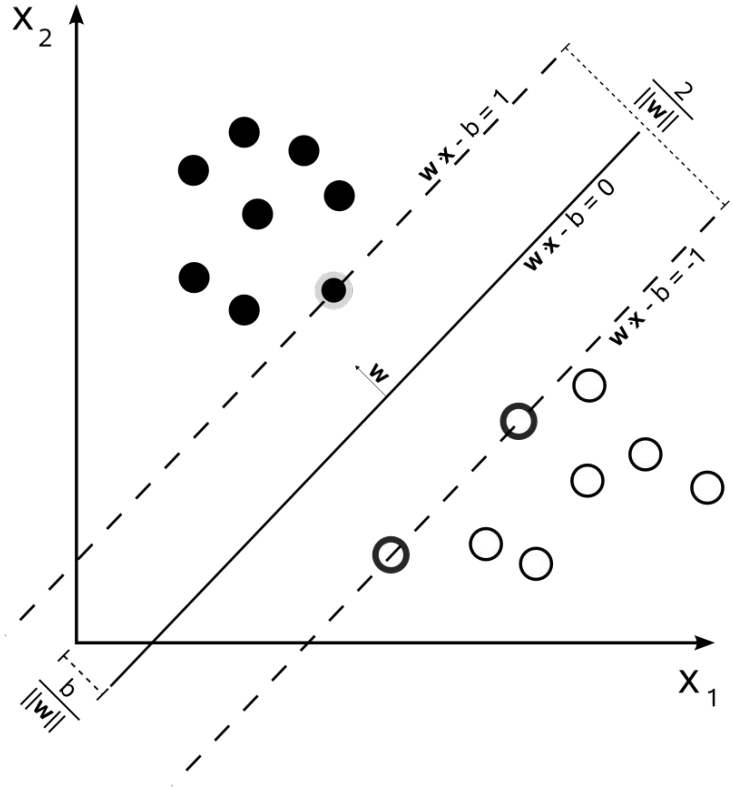
Байесовские методы — нестандартный подход к нейросетям, по которому защищают диссертации и ведут целые курсы!

SVM: метод опорный векторов

Идея

Есть данные, относящиеся к двум классам. Мы хотим построить разделяющую плоскость с наибольшим зазором между двумя классами.

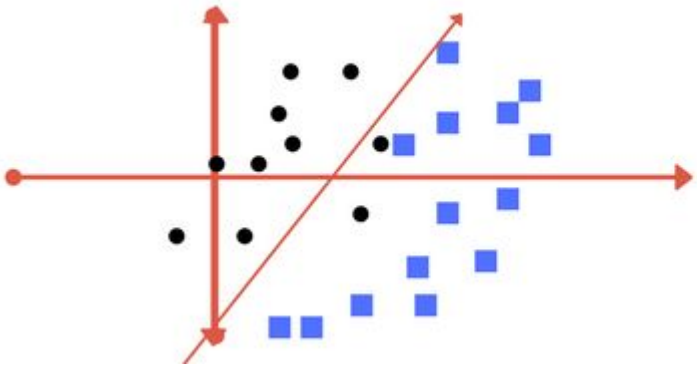
Подробнее SVM описан здесь.



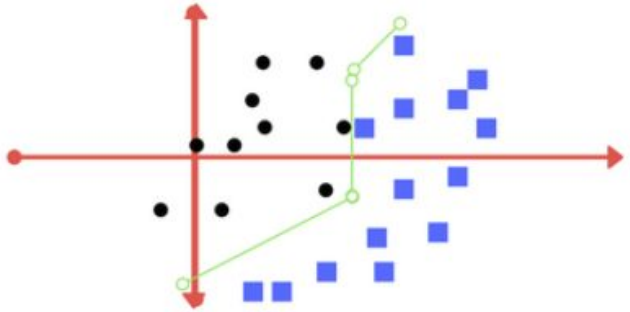
C parameter

For **large** values of C, the optimization will choose a **smaller-margin hyperplane**. Conversely, a very **small** value of C will cause the optimizer to look for a **larger-margin separating hyperplane**, even if that hyperplane misclassifies more points.

Маленький C:



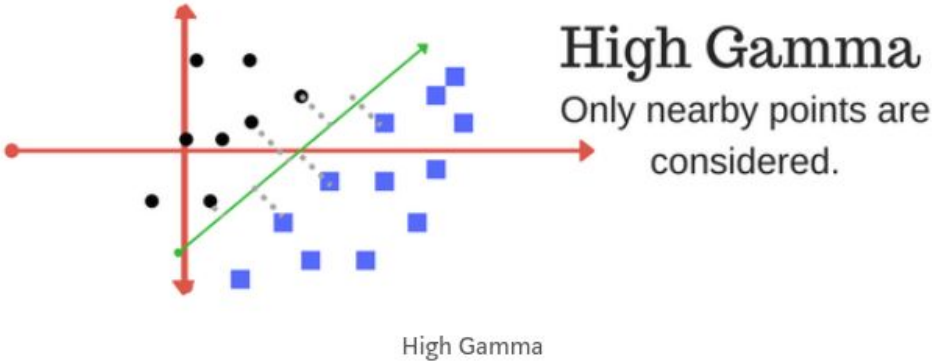
Большой C:



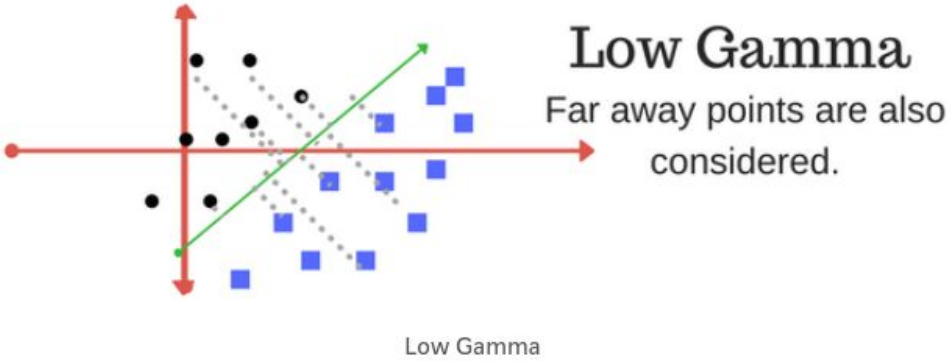
Gamma

With **low** gamma, points **far away** from plausible separation line **are considered** in calculation for the separation line.

High gamma means the points **close** to plausible line are considered in calculation.



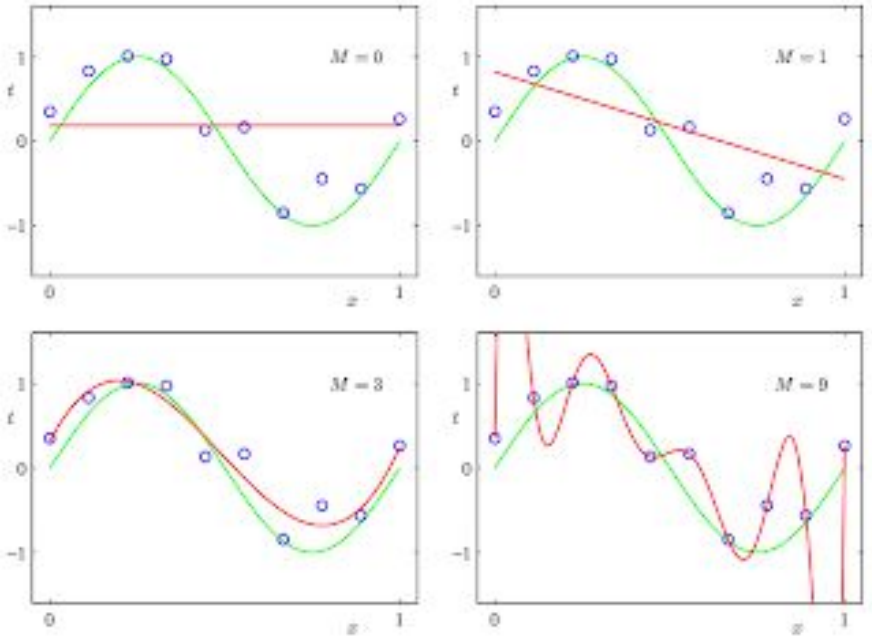
High Gamma



Low Gamma

Регуляризация

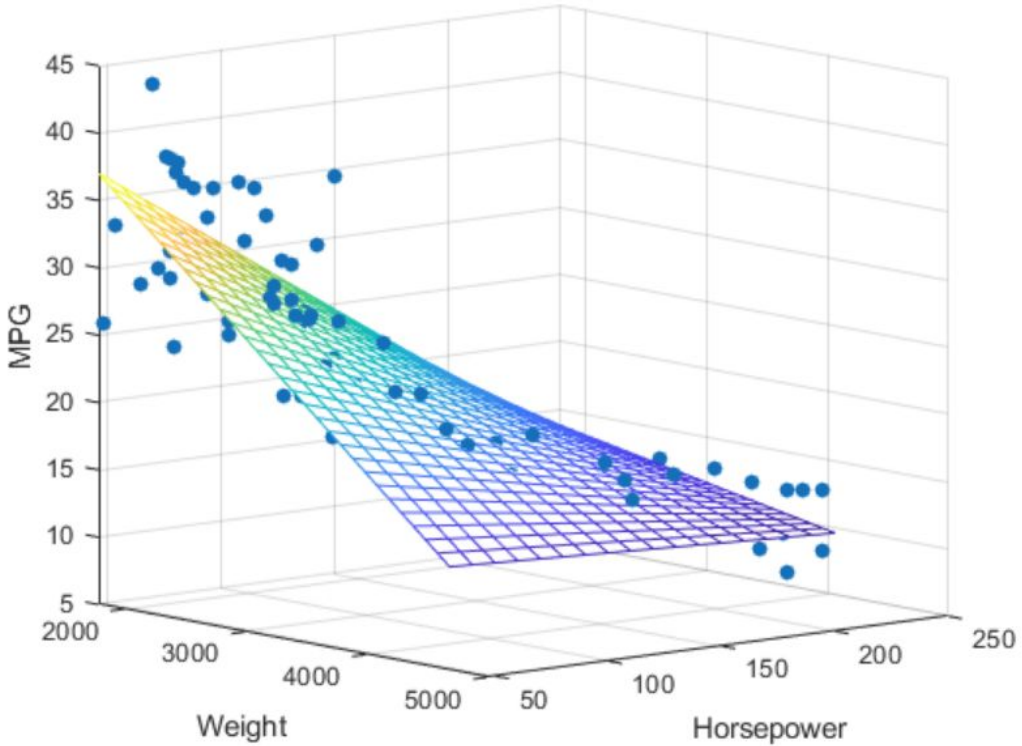
Недообучение и переобучение



Две верхние картинки — недообучение, нижняя слева — оптимальная аппроксимация, нижняя справа — переобучение.

(Источник: «Pattern Recognition and Machine Learning» (Figure 1.4) («Распознавание образов и машинное обучение» (рис. 1.4)).)

Неустранимая ошибка



Расход топлива зависит от двух признаков: вес машины, лошадиные силы.

Почему точки не лежат на одной прямой?

- неучтённые признаки
- шум в данных

Шум и неучтённые признаки — причина **неустранимой ошибки**.

Проблема баланса сложности модели

- Переусложнённые модели (что-то вроде $a * x^{39} + b * x^{35} + c * x^{31} + ...$) имеют высокую дисперсию (их сильно кидает по точкам обучающих данных), и они будут склонны к переобучению.
- Простые модели (вроде $a * x + b$) имеют большой риск недообучения

Наша цель — выбрать оптимальный баланс в сложности модели.

Регуляризация

В алгоритмах с градиентным спуском, вместо того, чтобы минимизировать просто функцию потерь J(X), будем минимизировать J(X) + λ * <какой-то из способов сложить коэффициенты>.

Тем самым, мы добиваемся того, что если вес какого-то коэффициента оказался очень большим, это было **действительно** нужно.

Например, изначальная J(X) была разницей квадратов между предсказанием модели $h_\theta(x^i)$ и истинным ответом y^i.

Тогда, с регуляризацией получится:

$$\min_\theta J(\theta) \qquad J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^{n}\theta_j^2\right]$$

L1 (Lasso)

L1 регуляризация (lasso regression) добавляет штраф, равный абсолютному значению суммы коэффициентов.

$$\sum_{i=1}^{n}(Y_i - \sum_{j=1}^{p} X_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

Cost function

Просто складываем значения коэффициентов и умножаем на lambda.

L2 (Ridge Regression)

L2 regularization (Ridge regression) adds an L2 penalty equal to the square of the magnitude of coefficients. L2 will not yield sparse models and all coefficients are shrunk by the same factor (none are eliminated).

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

Cost function

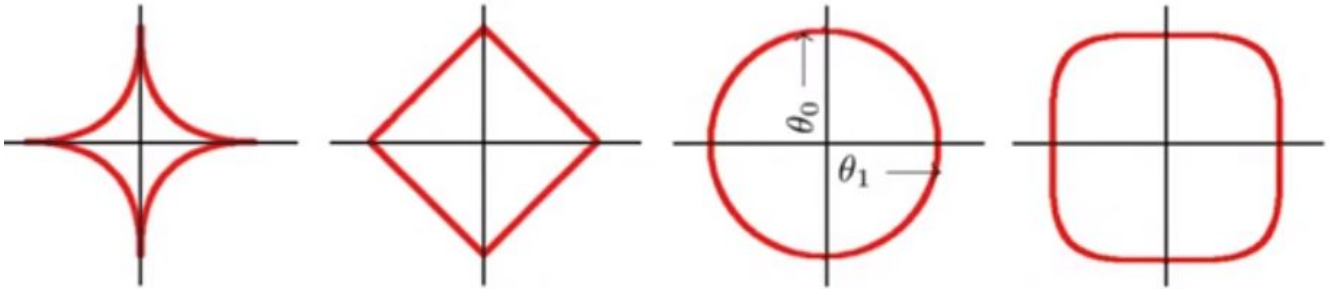
Возводим каждый коэффициент в квадрат и умножаем на lambda.

Разная регуляризация

Different regularization functions

- More generally, for the L_p regularizer: $\left(\sum_i |\theta_i|^p\right)^{\frac{1}{p}}$

Isosurfaces: $\|\theta\|_p = \text{constant}$



p=0.5

p=1

Lasso

p=2

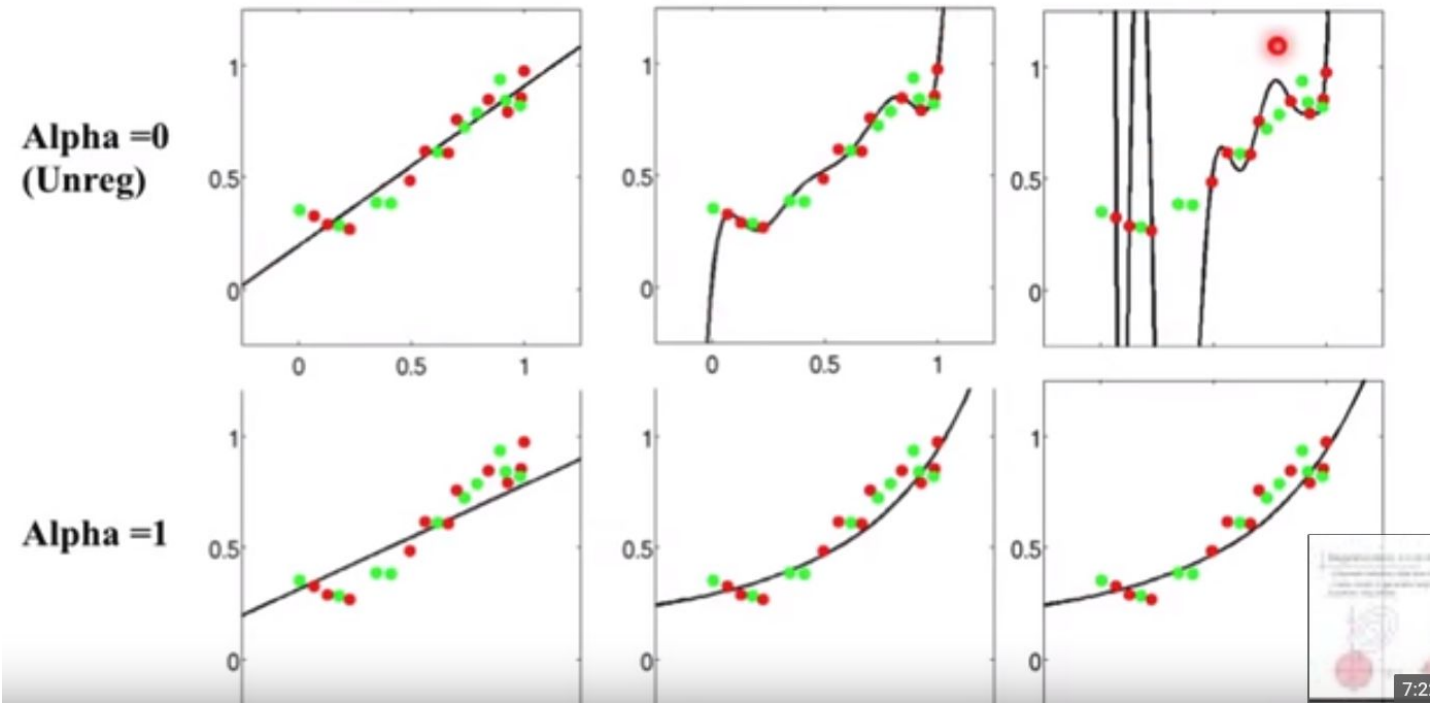
Quadratic

p=4

альфа / лямбда

Чем больше эта величина, тем больше значения мы придаем штрафу за величину коэффициентов.

Источник.



Как подбирать гиперпараметры

Параметры vs гиперпараметры

Параметр — это внутренняя характеристика модели, значение которой может быть выведенно из данных. Это, например, коэффициент при признаке “слово *котик*”.

Гиперпараметр — это характеристика, “внешняя” по отношении к модели. Например, k в kNN. Его нельзя вывести из данных при обучении, и надо подбирать как-то отдельно.

(Определения из статьи).

Подбор гиперпараметров

Как подобрать гиперпараметры?

- можно пробовать менять разные варианты руками
- можно перебирать их в цикле

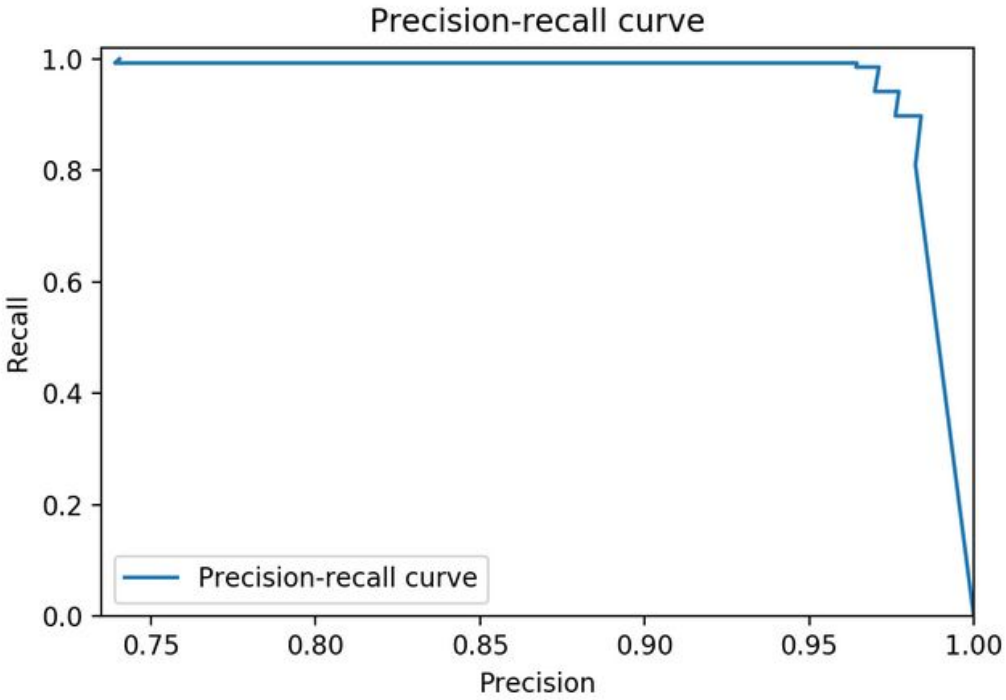
```
for df in range(0, 20):
    vec = TfidfVectorizer(min_df=df)
```

- а можно использовать Grid Search

```
from sklearn.model_selection import GridSearchCV
```

Ещё метрики и их визуализации

Precision-recall curve



With the precision-recall curve, the closer it is to the top-right corner, the better the algorithm.

And hence a larger **area under the curve** (AUC) indicates that the algorithm has higher recall and higher precision.

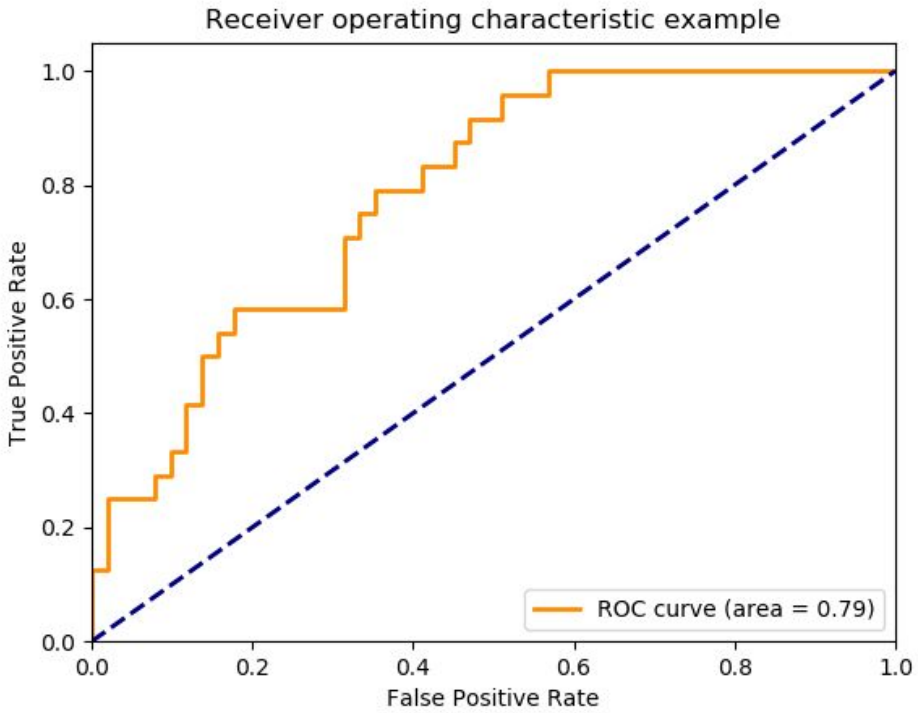
Receiver Operating Characteristic (ROC)

TPR — то же самое, что recall — TP / (TP + TN)

FPR — доля FP / (FP + TN)

Receiver Operating Characteristic — нахождение баланса между TPR и TFR при разных порогах принятия решения.

Receiver Operating Characteristic



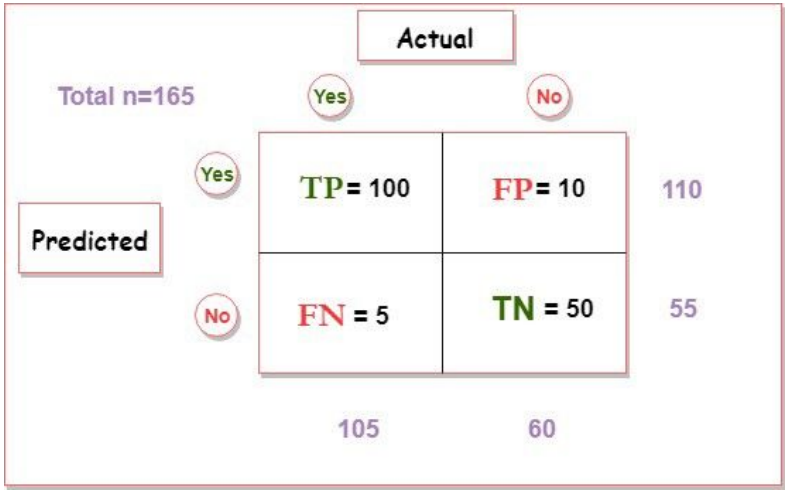
Источник каринки. Там же показано, как построить её с помощью matplotlib.

Area under the ROC curve (AUC) is a good measure of the performance of the classification algorithm. If it is near 0.5, the classifier is not much better than random guessing, whereas **it gets better as the area gets close to 1**.

Confusion matrix

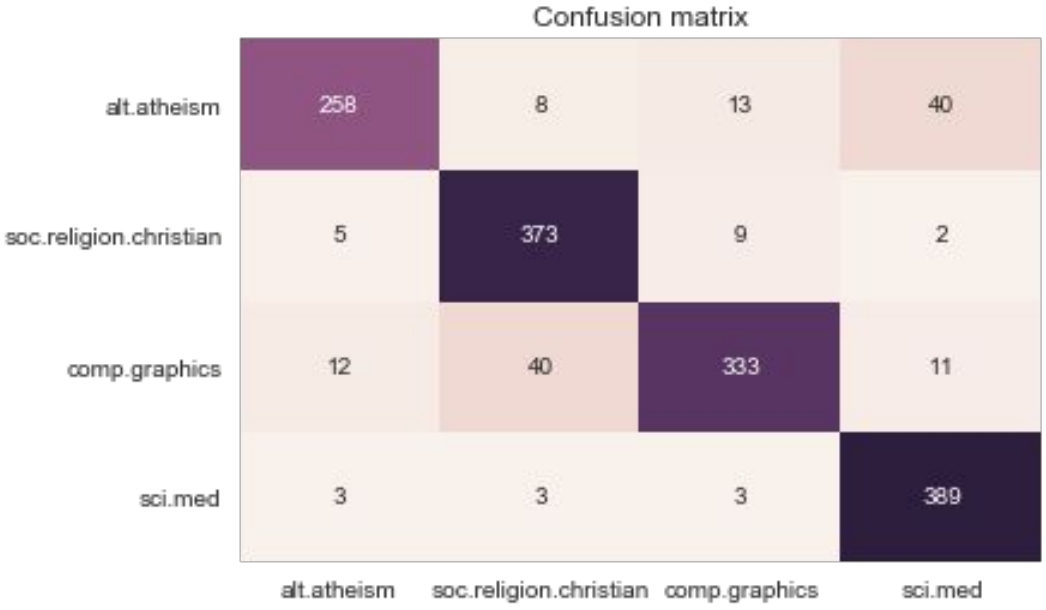
Когда мы говорили о точности и полноте, мы имели дело с confusion matrix:

- TP — правильно отнесено к классу
- TN — правильно не отнесено к классу
- FP — неправильно отнесено к классу
- FN — неправильно не отнесено к классу



pr = TP / (TP + **FP**) rec = TP / (TP + **FN**)

Confusion matrix



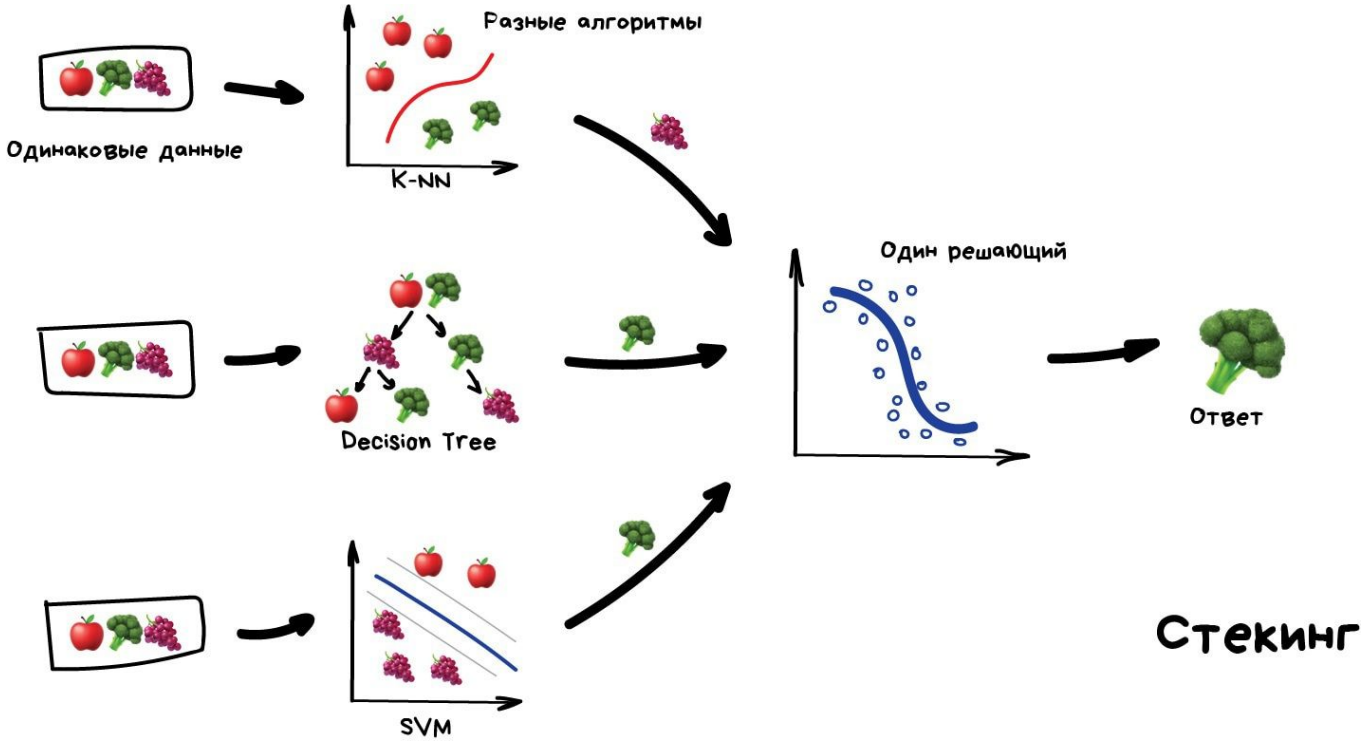
Раньше мы рассматривали confusion matrix 2x2.

Но в задаче мультиклассовой классификации они могут быть и больше, и бывают очень полезны.

Источник картинки, данные про классификацию новостей.

Ансамбли
(Все картинки отсюда)

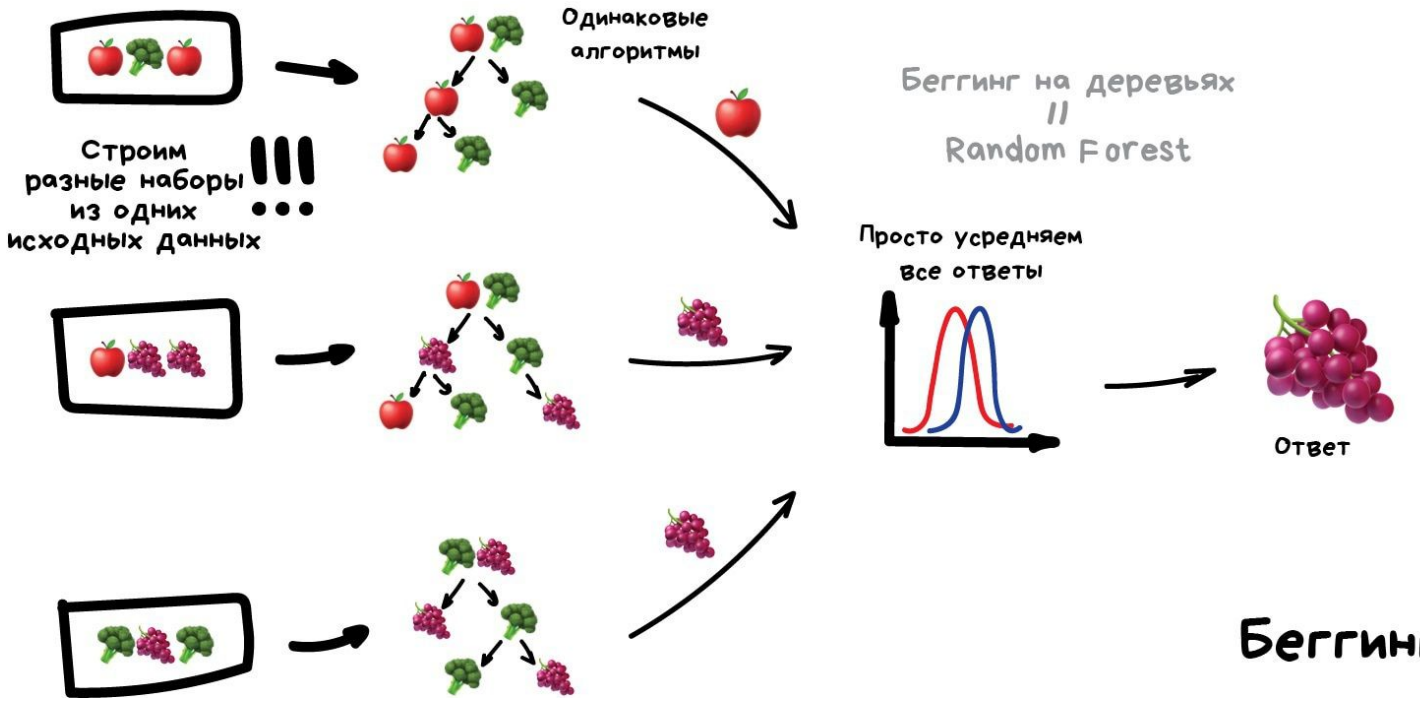
Stacking



Обучаем n разных алгоритмов и передаём их результаты на вход последнему. Ключевое слово — разных.

На практике применяется редко.

Bagging



Обучаем один алгоритм много раз на случайных выборках из данных. Потом усредняем ответы.

Данные в случайных выборках могут повторяться.

Random Forest (случайный лес)

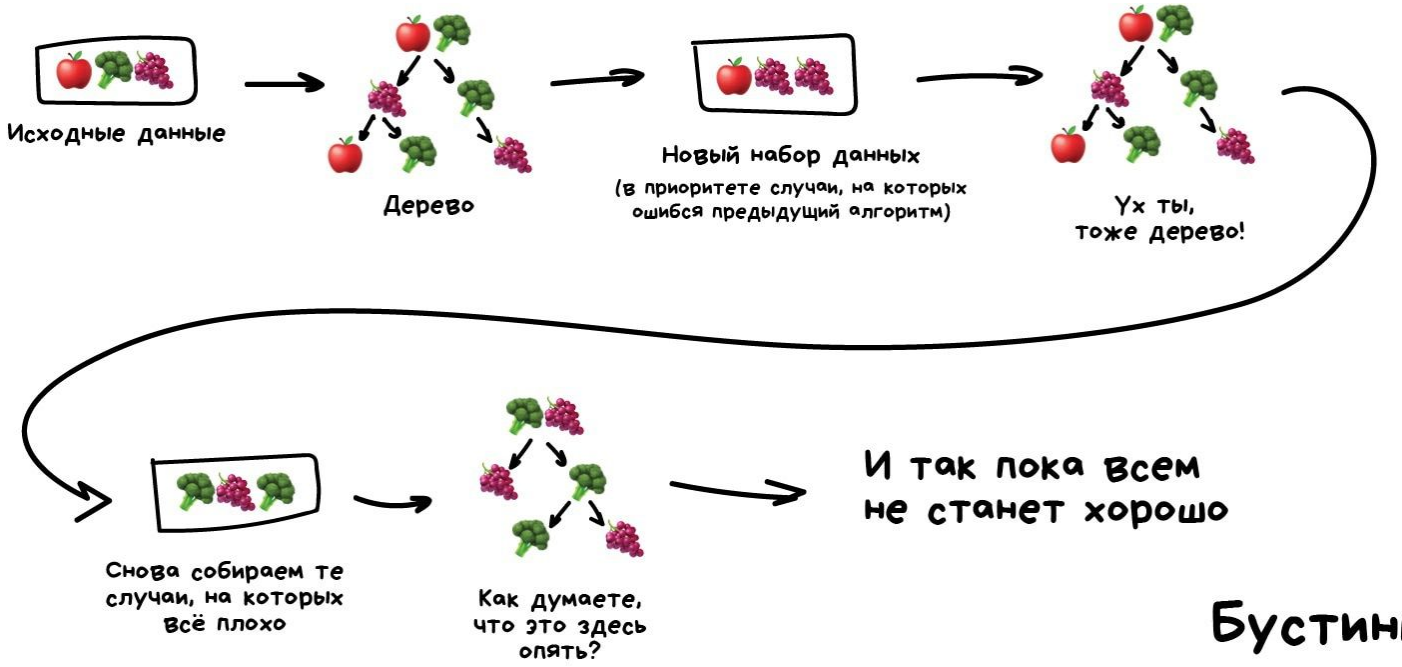
from sklearn.ensemble import RandomForestClassifier

Самый популярный вид бэггинга (потому что деревья склонны переобучаться).

Некоторые гиперпараметры:

- n_estimators — сколько деревьев
- параметры, используемые для деревьев (критерий, глубина дерева etc)

Boosting



Обучаем алгоритмы последовательно, каждый следующий уделяет особое внимание тем случаям, на которых ошибся предыдущий.

Бустинг

Boosting sklearn

Самые популярные алгоритмы для бустинга:

- XGBoost
- Light GBM
- CatBoost

Здесь разбираются их различия.

Ресурсы

Почитать / посмотреть

- SVM с картинками, SVM с кодом
- Про Байесовский наивный классификатор
- Regularization in Machine Learning
- Видео про регуляризацию
- introduction to data visualization in python
- Машинное обучение для людей, ансамбли (рус)
- объяснение про ROC curve
- про Grid Search

Домашнее задание: классификация отзывов

Данные: отзывы на фильмы.

Что сделать:

- 3 балла — считать датасет, обучить на нём любой из описанных сегодня классификаторов, измерить качество
- 3 балла — перебрать как минимум 3 классификатора, найти лучший
- 2 балла (*) — попробовать разные гиперпараметры, найти лучший
- бонусный 1 балл за понятный и чистый код :)