

kNN, деревья решений и прочее

Маша Шеянова, masha.shejanova@gmail.com

Популярные алгоритмы классификации

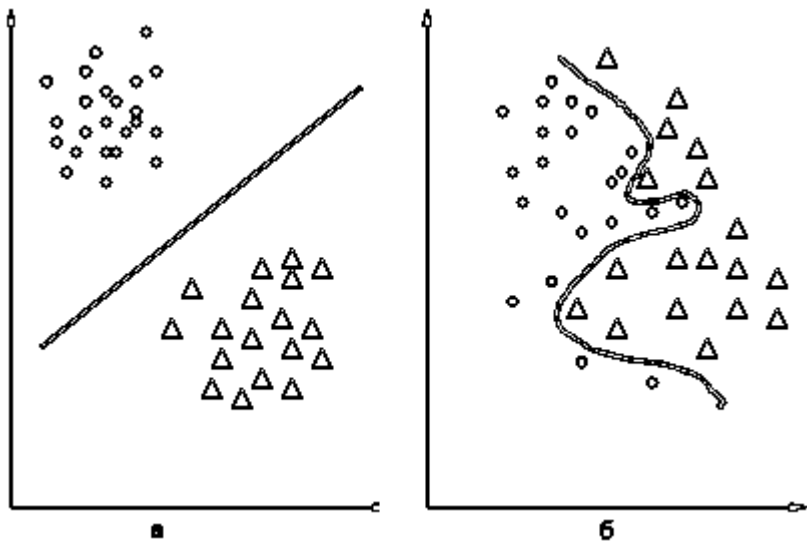
- логистическая регрессия
- **k ближайших соседей (kNN)**
- **деревья решений**
- наивный Байес
- метод опорных векторов (SVM)

На прошлой лекции мы обсуждали логистическую регрессию. Теперь разберёмся в kNN и деревьях решений.

Метрические классификаторы. kNN.

Гипотеза компактности

Это предположение о том, что схожие (близкие в пространстве признаков) объекты гораздо чаще лежат в одном классе, чем в разных.



Функция расстояния

Для того, чтобы сделать метрический классификатор, надо уметь считать расстояние между объектам; находить, какой ближе, а какой дальше.

Но как выглядят наши объекты? Это вектора! Каждая координата — значение того или иного признака.

А значит, в качестве функции расстояния можно воспользоваться:

- евклидовым расстоянием
- косинусным расстоянием

Функция расстояния

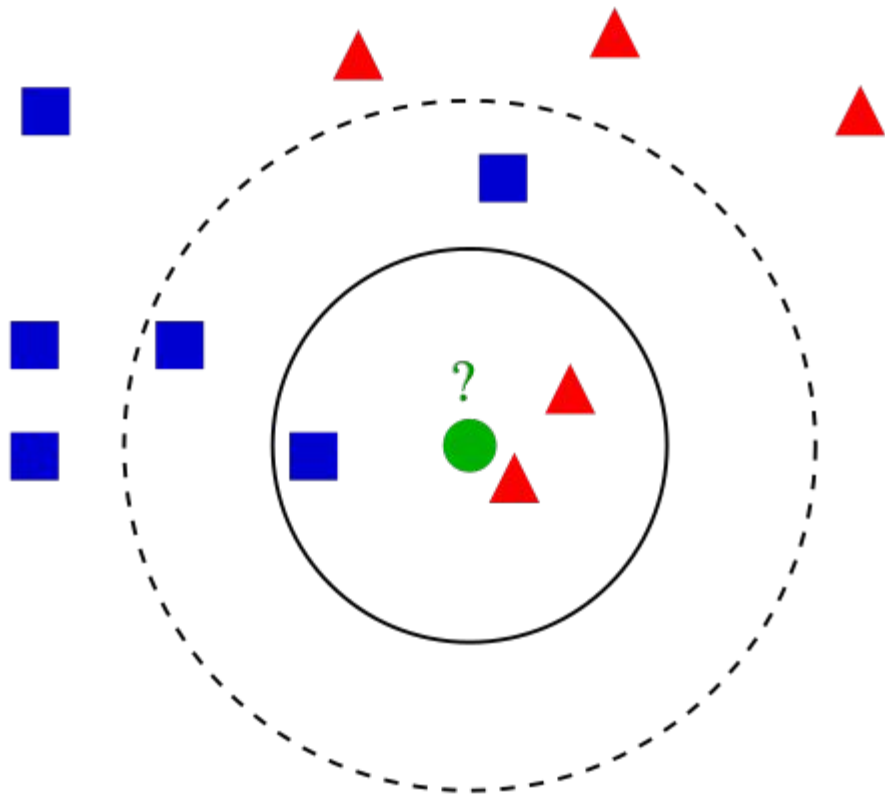
Евклидово расстояние, $[0; +\infty)$:

$$\begin{aligned} d(\mathbf{p}, \mathbf{q}) = d(\mathbf{q}, \mathbf{p}) &= \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2} \\ &= \sqrt{\sum_{i=1}^n (q_i - p_i)^2}. \end{aligned}$$

Косинусное расстояние — косинус углов между векторами $[0; 1]$:

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}},$$

к ближайших соседей (kNN)



Источник картинки.

К какому классу будет отнесён
кружок в центре при $k=3$?

При $k=5$?

kNN: преимущества и недостатки

Преимущества:

- соседи “голосуют” — не так страшны аутлаеры
- достаточно просто расстояний

Ещё особенности:

- параметр k можно настраивать для каждой задачи
- надо справляться с ситуациями когда среди k соседей одинаковое количество представителей разных классов

Главный недостаток: модель тяжёлая, потому что хранит все объекты обучающей выборки

kNN в sklearn

[sklearn.neighbors.KNeighborsClassifier](#)

Можно настраивать параметры:

- `n_neighbors` (по дефолту, 5)
- `weights` (по дефолту, “uniform” — ; ещё может быть “distance” и функция)
- `algorithm` — разные способы оптимизировать время поиска соседей

И ещё несколько других параметров.

Деревья решений

Идея

Каждый признак — критерий, чтобы выбрать, к какому классу относится объект. Мы можем построить **дерево**, где каждый **узел — разветвление по признаку**. Корень — самый значимый признак, дальше другие признаки.

Плюсы:

- очень интуитивно

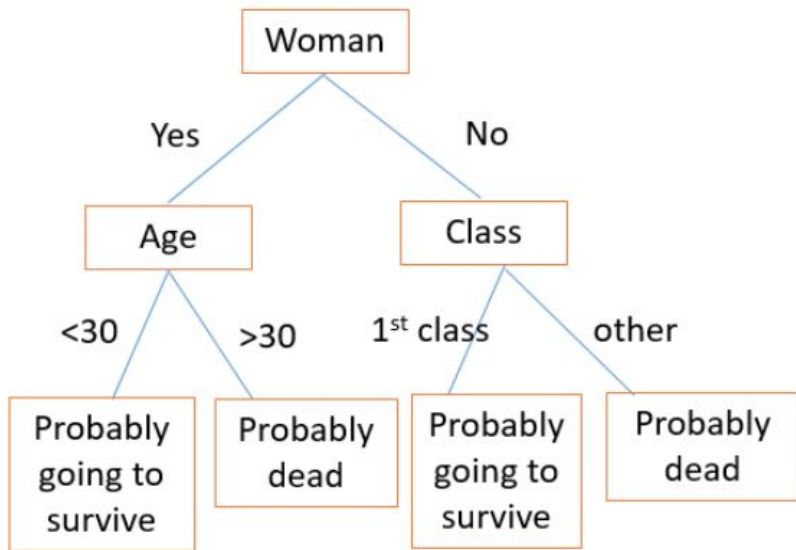
Минусы:

- склонны к переобучению

Как выглядит дерево решений

Источник картинки.

Дерево решений на примере датасета из титанка.



Как строятся деревья?

Сверху-вниз: сначала находим корень, потом в каждом из поддеревьев — новый корень, и так далее.

Как выбираем корень? Вводим “*impurity function*” — насколько плохо классифицирован датасет. *Impurity function* может быть разной, главное — чтобы она была большой, когда разбиение датасета на “кучки”

Когда каждое новое разделение классифицирует датасет чуть лучше.

Алгоритм

1. Вычислить impurity function для изначального датасета
2. Для каждого признака:
 - a. вычислить impurity function для каждого сплита
 - b. вычислить, насколько текущий атрибут лучше, чем было до него
3. Выбрать атрибут с лучшей разницей в impurity function
4. Повторять, пока мы не захотим остановиться

Энтропия


Один из алгоритмов использует энтропию. Как можно измерить насколько распределение "разнородное", или насколько "грязный" датасет?

Взять математическое ожидание количества бит, которое понадобится, чтобы закодировать один из исходов в **оптимальной** кодировке.

Это количество бит — **информация**
($-\log p$).

$$Entropy = - \sum p(X) \log p(X)$$

Мат. ожидание информации — **энтропия**.



here $p(x)$ is a fraction of
examples in a given class

Information gain

*“**Information gain (IG)** measures how much “information” a feature gives us about the class.”*

Сколько информации вносит родительский узел:

$$\text{Information gain} = \text{entropy (parent)} - [\text{weightes average}] * \text{entropy (children)}$$

Information gain

Разница между энтропией до и после разделения. Иными словами, насколько “чище”, “определённое” стали данные.

$$IG(A, S) = H(S) - \sum_{t \in T} p(t)H(t)$$

Where,

- $H(S)$ – Entropy of set S
- T – The subsets created from splitting set S by attribute A such that $S = \bigcup_{t \in T} t$
- $p(t)$ – The proportion of the number of elements in t to the number of elements in set S
- $H(t)$ – Entropy of subset t

Деревья решений в sklearn

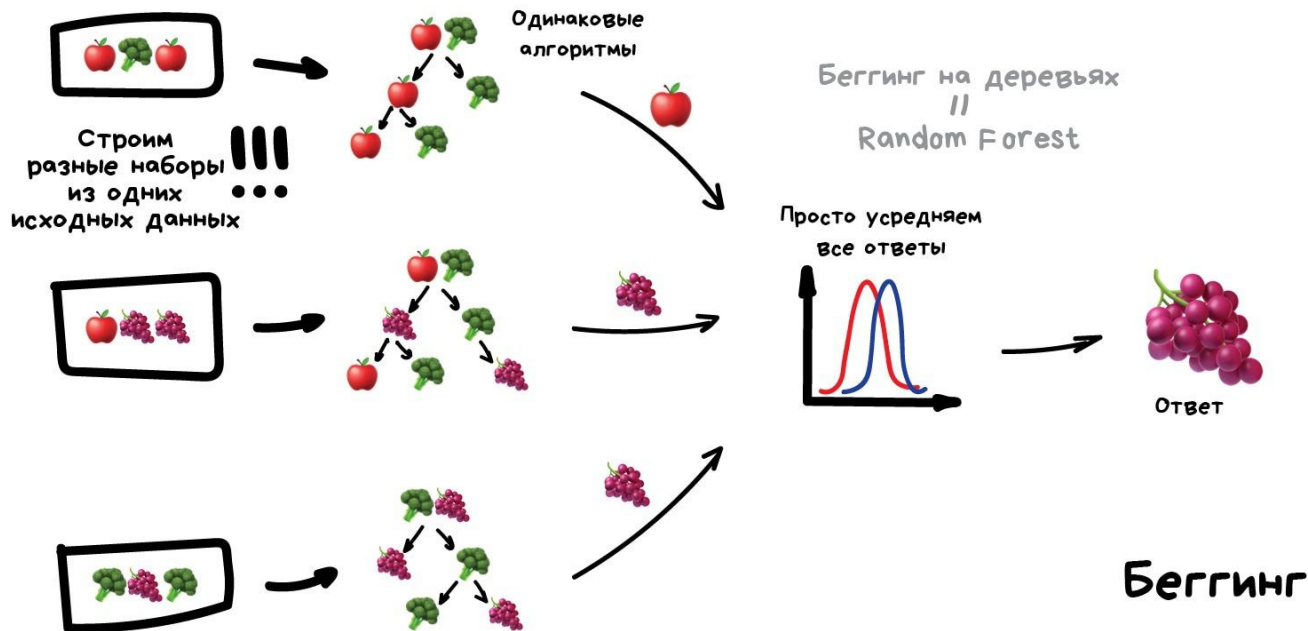
```
from sklearn.tree import DecisionTreeClassifier
```

Гиперпараметры:

- criterion — gini (ещё один способ разделить) или entropy
- min_samples_split — сколько должно быть точек данных, чтобы мы продолжили делить
- min_impurity_decrease
- max_depth — максимально возможная высота дерева
- max_leaf_nodes — максимально возможная “ширина” дерева

Случайный лес (random forest)

from sklearn.ensemble import **RandomForestClassifier**



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

Данные в случайных выборках могут повторяться.

О метриках качества

Accuracy

Метрика accuracy — самая простая оценка классификации: поделить все правильные ответы классификатора на количество всех ответов.

Достоинства: простота.

Недостатки: плохо работает, когда данные сильно перекошены.

(Например, если мы ищем у человека редкую болезнь, модель, которая про всех будет говорить “здоров”, будет права в 99% случаев)ю

Confusion matrix

А вот более подробная информация про то, какие ошибки делает модель.

TP: true positive (верно сказали да)

FP: false positive (сказали да, а надо было нет)

TN: true negative (верно сказали нет)

FN: false negative (сказали нет, а надо было да)


The diagram illustrates a confusion matrix for a classification model. It features a central 2x2 grid with 'Actual' on the horizontal axis and 'Predicted' on the vertical axis. The columns are labeled 'Yes' and 'No' under the 'Actual' header. The rows are labeled 'Yes' and 'No' under the 'Predicted' header. The cells contain the following values: TP=100, FP=10, FN=5, and TN=50. Marginal totals are shown: 105 for 'Actual Yes', 60 for 'Actual No', 110 for 'Predicted Yes', and 55 for 'Predicted No'. The total sample size is noted as 'Total n=165'.

		Actual		
		Yes	No	
Predicted	Yes	TP = 100	FP = 10	110
	No	FN = 5	TN = 50	55
		105	60	


Total n=165

Точность и полнота

How many selected items are relevant?

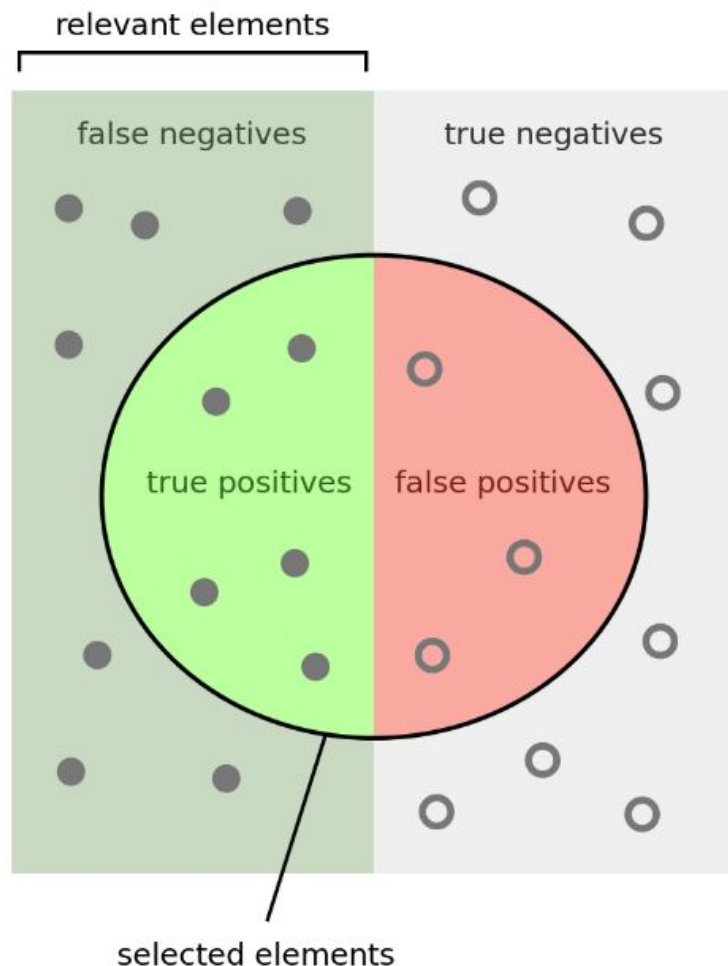
$$\text{Precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$


How many relevant items are selected?

$$\text{Recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$


Precision (точность): не отхватить лишнего

Recall (полнота): ничего не упустить



Точность и полнота

Когда важна точность (не отхватить случайно ничего лишнего):

- спам-фильтр (человек не хочет потерять важные письма)

Когда важна полнота (ничего не забыть):

- диагностика болезней
- поиск террористов
- ???

f1-мера

Но что, если важно и то, и то?

Можно было бы просто посчитать среднее между точностью и полнотой. Но тогда очень плохие результаты будут get away.

f1 — среднее гармоническое точности и полноты

$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$

Точность и полнота для нескольких классов

Но что если классов много — получается, точность, полнота и f1-мера не работают?

А вот и работают: надо для каждого класса считать, что положительный класс — это он сам, а отрицательный — все остальные.

Что такое baseline

Это тот результат, с которым вы сравниваете свой метод. Лучший результат, который можно было получить простым способом.

Например, в задаче бинарной классификации со сбалансированными классами можно “подбрасывать монетку”: accuracy будет ~ 0.5 .

Если вы изобрели новый метод для решения задачи, baseline — прошлый лучший метод. Если вы сравниваете, насколько важна лемматизация, baseline — результат без лемматизации.

Задача: придумайте простой baseline для определения оскорбительных твитов, для которого не нужно MO.

О предобработке текста

Что можно сделать с текстом?

Предобработка — в принципе, все изменения, которые вы делаете с данными до того, как извлечь из них признаки.

- почистить текст от мусора (например, от остатков markdown)
- убрать стоп-слова (*а, не, на, и, ...*), пунктуацию
- сделать умную токенизацию
- лемматизировать слова
- добавить информацию о частях речи
- добавить информацию о роли в предложении
- ...

Ресурсы

Почитать

- про деревья решений
- про энтропию и information gain для деревьев решений
- про kNN
- про precision, recall и f-меру