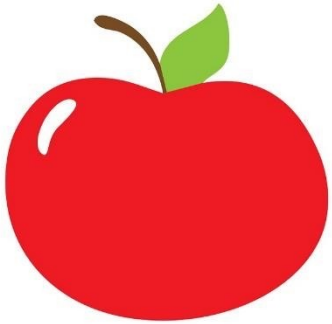
ПРОСТЫЕ НЕЛИНЕЙНЫЕ АЛГОРИТМЫ

НАИВНЫЙ БАЙЕСОВСКИЙ КЛАССИФИКАТОР

НАИВНЫЙ БАЙЕСОВСКИЙ КЛАССИФИКАТОР

Наивный байесовский классификатор – это алгоритм классификации, основанный на теореме Байеса с допущением о независимости признаков.



Пример: фрукт может считаться яблоком,

если:

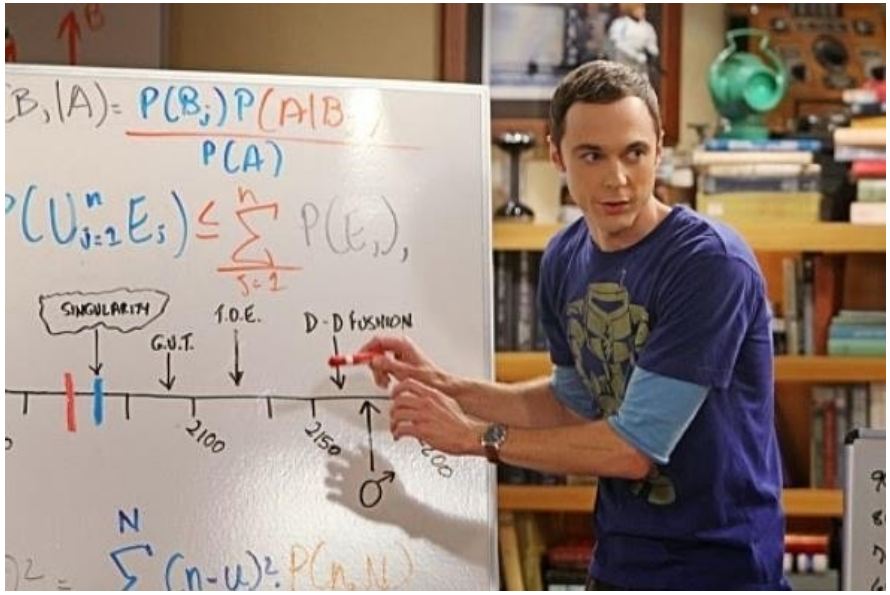
1) он красный
2) круглый
3) его диаметр составляет порядка 8 см

Предполагаем, что признаки вносят независимый вклад в вероятность того, что фрукт является яблоком.

ТЕОРЕМА БАЙЕСА

Теорема Байеса:

$$P(c|x) = \frac{P(x|c) \cdot P(c)}{P(x)}$$



- $P(c|x)$ - вероятность того, что объект со значением признака x принадлежит классу c.
- $P(c)$ – априорная вероятность класса c.
- $P(x|c)$ - вероятность того, что значение признака равно x при условии, что объект принадлежит классу c.
- $P(x)$ – априорная вероятность значения признака x.

ПРИМЕР РАБОТЫ БАЙЕСОВСКОГО АЛГОРИТМА

Пример: на основе данных о погодных условиях необходимо определить, состоится ли матч.

Weather	Play
Sunny	No
Overcast	Yes
Rainy	Yes
Sunny	Yes
Sunny	Yes
Overcast	Yes
Rainy	No
Rainy	No
Sunny	Yes
Rainy	Yes
Sunny	No
Overcast	Yes
Overcast	Yes
Rainy	No

- Преобразуем набор данных в следующую таблицу:

Weather	No	Yes
Overcast	0	4
Rainy	3	2
Sunny	2	3
Grand Total	5	9

ПРИМЕР РАБОТЫ БАЙЕСОВСКОГО АЛГОРИТМА

Решим задачу с помощью теоремы Байеса:

$$P(Yes|Sunny) = P(Sunny|Yes) \cdot P(Yes)/P(Sunny)$$

Таблица частот				
Weather	**No**	**Yes**		
Overcast	0	4	**=4/14**	**0.29**
Rainy	3	2	**=5/14**	**0.36**
Sunny	2	3	**=5/14**	**0.36**
Grand Total	5	9		
	=5/14	**=9/14**		
	0.36	**0.64**		

- $P(Sunny|Yes) = \frac{3}{9}, P(Sunny) = \frac{5}{14}, P(Yes) = \frac{9}{14}.$
- $P(Yes|Sunny) = \frac{3}{9} \cdot \frac{9}{14} : \frac{5}{14} = \frac{3}{5} = 0{,}6 \Rightarrow 60\%.$

В СЛУЧАЕ НЕСКОЛЬКИХ ПРИЗНАКОВ

Пусть $x_1, \dots, x_n$ - признаки объекта, y - целевая переменная.

Тогда теорема Байеса записывается в виде

$$P(y|x_1, \dots, x_n) = \frac{P(x_1|y)P(x_2|y) \dots P(x_n|y)P(y)}{P(x_1)P(x_2) \dots P(x_n)}$$

Почитать статью про Байесовский классификатор

GAUSSIAN NAÏVE BAYES

Пусть $x_1, \dots, x_n$ - **числовые признаки объекта, распределенные по нормальному закону**.

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y}} \exp(-\frac{\left(x_i - \mu_y\right)^2}{2\sigma_y^2})$$

MULTINOMIAL NAÏVE BAYES

Пусть $x_1, \dots, x_n$ - **числовые признаки объекта, имеющие дискретное распределение (например, количества или частоты)**

$$P(x_i|y) = \frac{N_{yi} + \alpha}{N_y + \alpha n},$$

где

- N_{yi} - число строк в таблице с категорией i и значением таргета y
- N_y - число строк в таблице со значением таргета y
- α – параметр сглаживания (чтобы предотвратить нулевые вероятности)
- n – число признаков

БАЙЕСОВСКИЙ АЛГОРИТМ ДЛЯ КЛАССИФИКАЦИИ

Плюсы и минусы:

+ классификация быстрая и простая

+ в случае, если выполняется предположение о независимости, классификатор показывает очень высокое качество

- если в тестовых данных присутствует категория, не встречавшаяся в данных для обучения, модель присвоит ей нулевую вероятность

НАИВНЫЙ БАЙЕС + KDE

Можно улучшить качество работы наивного байесовского алгоритма, используя Kernel Density Estimator для оценки плотности распределения.

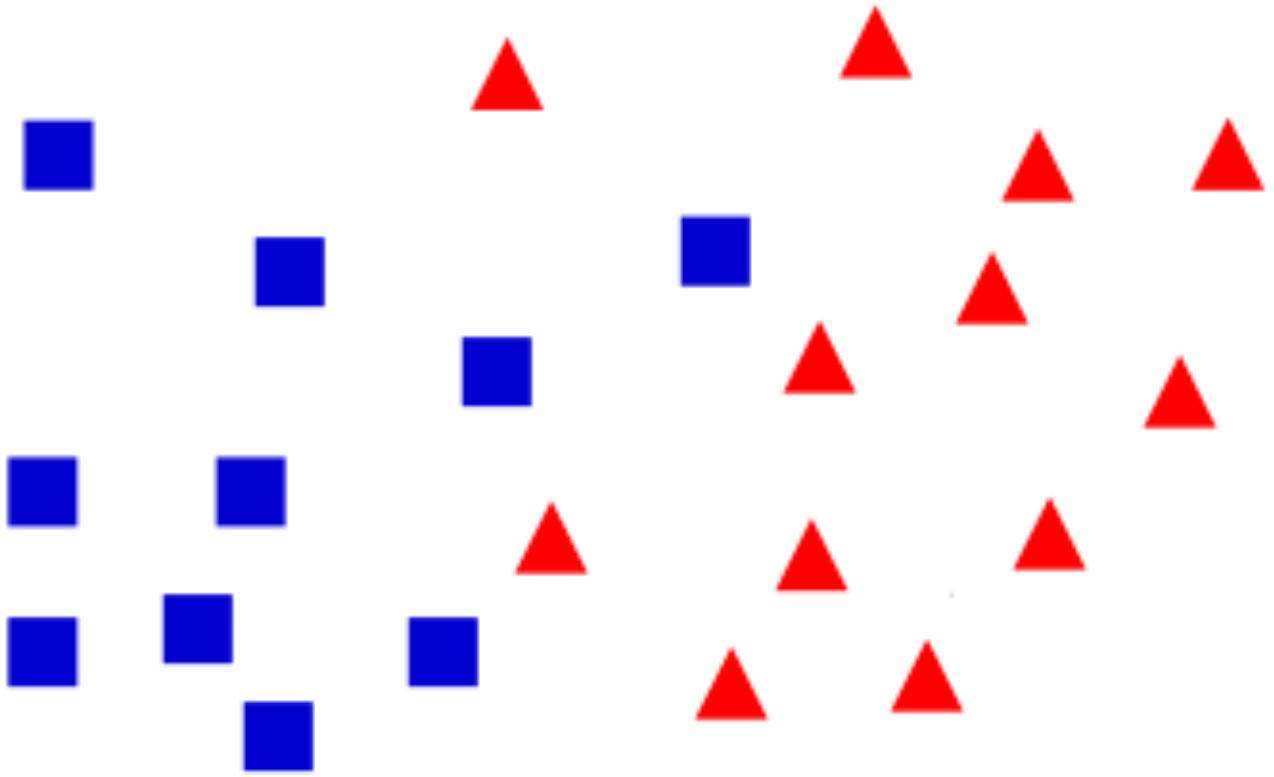
Материал выходит за рамки нашего курса, но с реализацией можно ознакомиться тут:

https://github.com/sampath9dasari/NaiveBayesClassifier-KDE/tree/master

МЕТОД БЛИЖАЙШИХ СОСЕДЕЙ

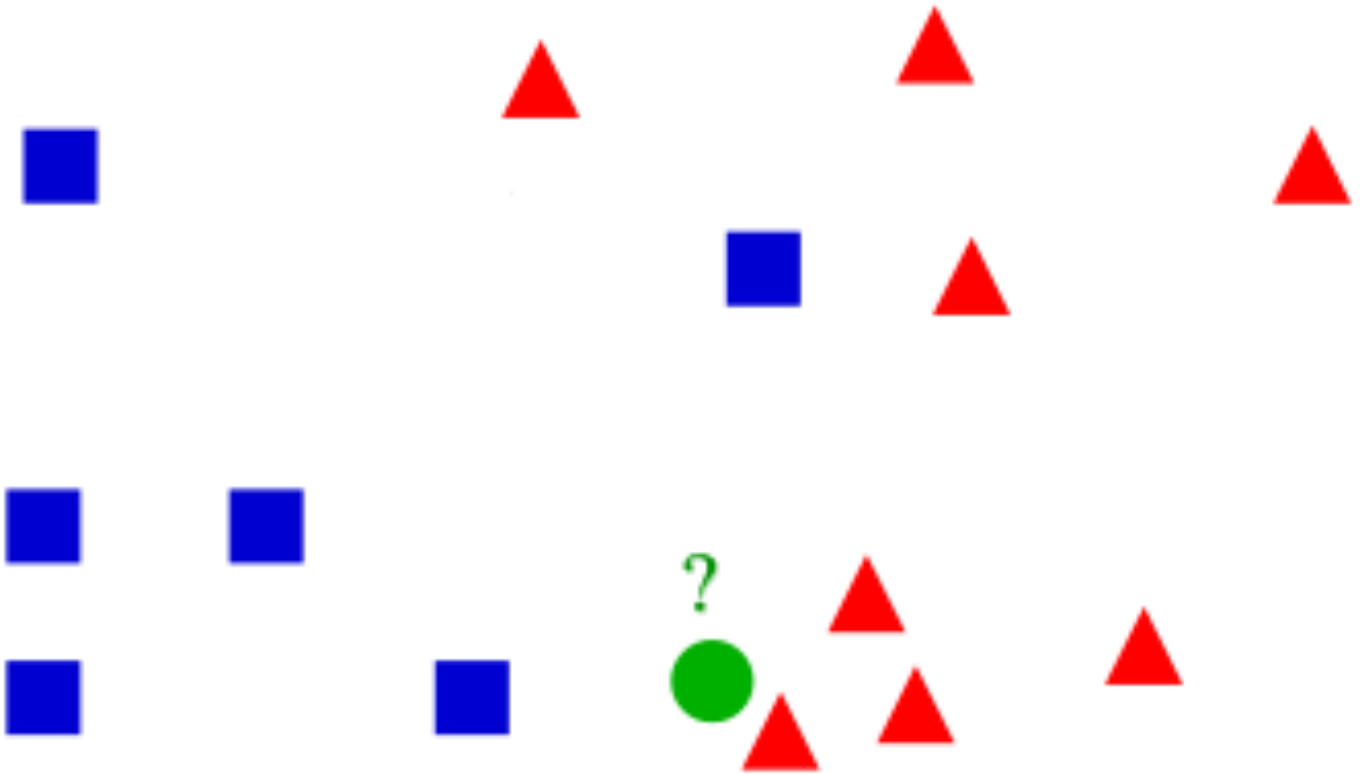
МЕТОД БЛИЖАЙШИХ СОСЕДЕЙ

Идея: схожие объекты находятся близко друг к другу в пространстве признаков.



МЕТОД БЛИЖАЙШИХ СОСЕДЕЙ

Как классифицировать новый объект?



МЕТОД БЛИЖАЙШИХ СОСЕДЕЙ

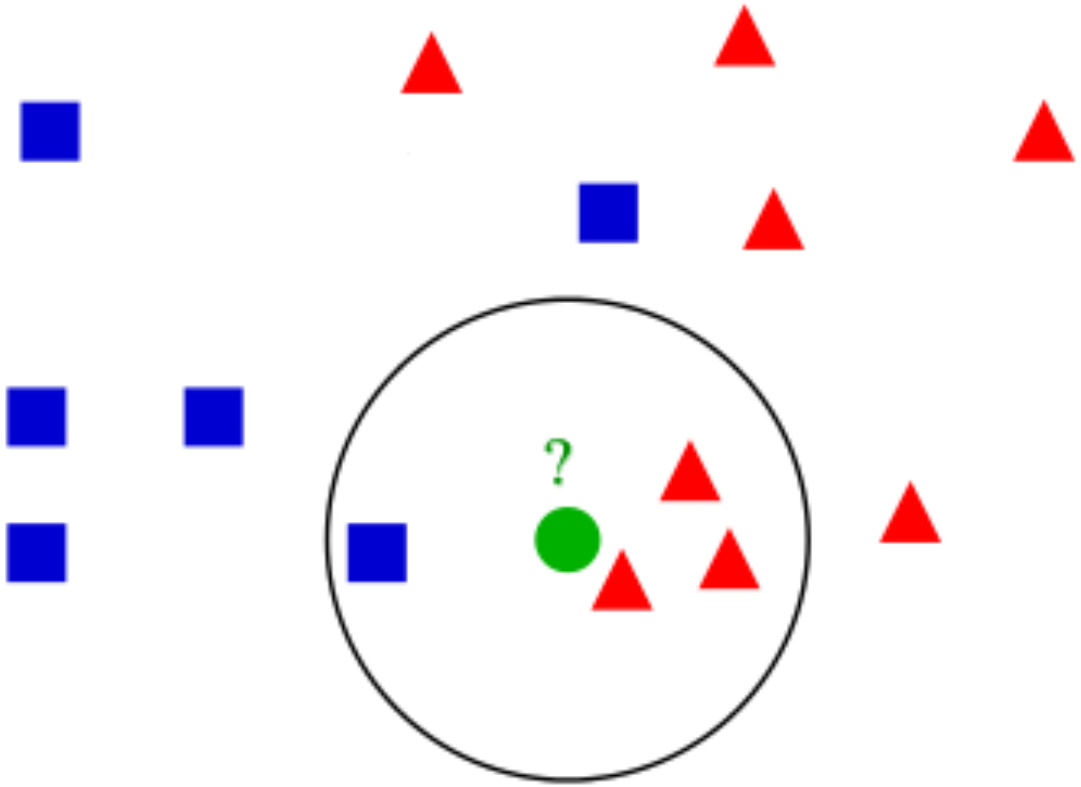
Чтобы классифицировать новый объект, нужно:

- Вычислить расстояние до каждого из объектов обучающей выборки.
- Выбрать k объектов обучающей выборки, расстояние до которых минимально.
- Класс классифицируемого объекта — это класс, наиболее часто встречающийся среди k ближайших соседей.

МЕТОД БЛИЖАЙШИХ СОСЕДЕЙ

Число ближайших соседей k – гиперпараметр метода.

Например, для $k = 4$ получим:



То есть объект будет отнесён к классу *треугольников*.

ФОРМАЛИЗАЦИЯ МЕТОДА

Пусть k - количество соседей. Для каждого объекта u возьмём k ближайших к нему объектов из тренировочной выборки:

$$x_{(1;u)}, x_{(2;u)}, \dots, x_{(k;u)}.$$

Тогда класс объекта u определяется следующим образом:

$$a(u) = argmax_{y \in Y} \sum_{i=1}^{k} I[y_i = y]$$

ФОРМАЛИЗАЦИЯ МЕТОДА

Пусть k - количество соседей. Для каждого объекта u возьмём k ближайших к нему объектов из тренировочной выборки:

$$x_{(1;u)}, x_{(2;u)}, \dots, x_{(k;u)}.$$

Тогда класс объекта u определяется следующим образом:

$$a(u) = argmax_{y \in Y} \sum_{i=1}^{k} I[y_i = y]$$

Ближайшие объекты - это объекты, расстояние от которых до данного объекта наименьшее по некоторой метрике ρ.

ФОРМАЛИЗАЦИЯ МЕТОДА

Пусть k - количество соседей. Для каждого объекта u возьмём k ближайших к нему объектов из тренировочной выборки:

$$x_{(1;u)}, x_{(2;u)}, \dots, x_{(k;u)}.$$

Тогда класс объекта u определяется следующим образом:

$$a(u) = argmax_{y \in Y} \sum_{i=1}^{k} I[y_i = y]$$

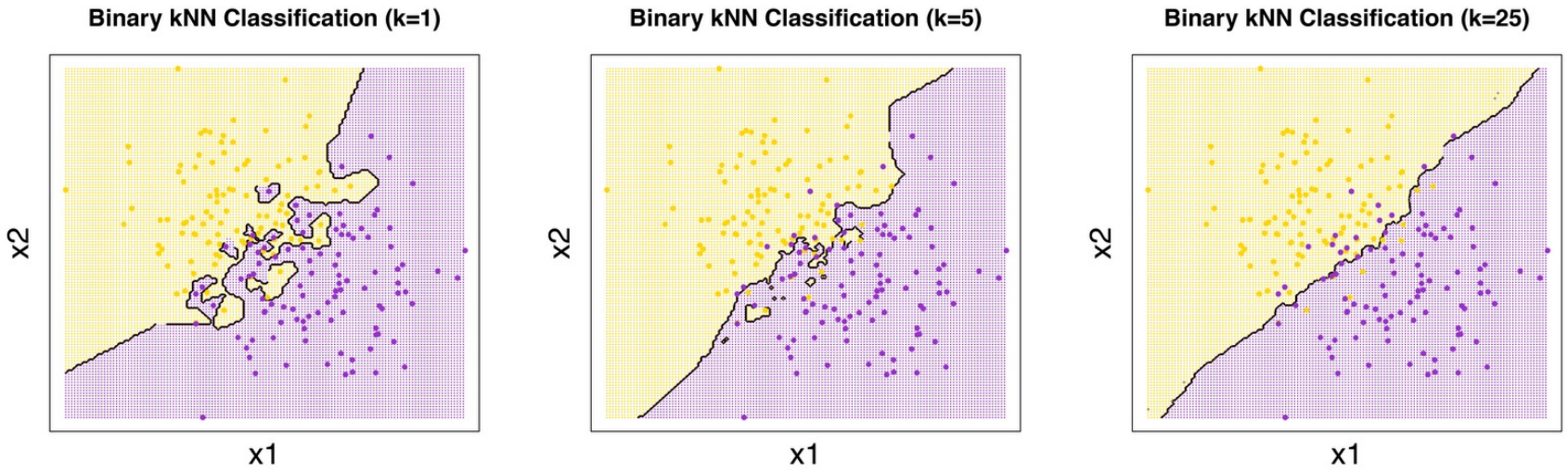
Ближайшие объекты - это объекты, расстояние от которых до данного объекта наименьшее по некоторой метрике ρ.

- В качестве метрики ρ как правило используют евклидово расстояние, но можно использовать и другие метрики.
- Перед использованием метода необходимо масштабировать данные, иначе признаки с большими числовыми значениями будут доминировать при вычислении расстояний.

ВЛИЯНИЕ ГИПЕРПАРАМЕТРА k

Алгоритм определения классов очень простой. На самом деле ***у метода нет фазы обучения***, потому что нет параметров, которые подбираются в процессе обучения по выборке. В этом смысле метод очень простой.

Несмотря на свою простоту, метод легко переобучается. Например, если взять число соседей очень маленьким (*k*=2 или *k*=3), метод будет делать предсказания только по двум или трем самым близким точкам, и поэтому сильно подгонится под данные.

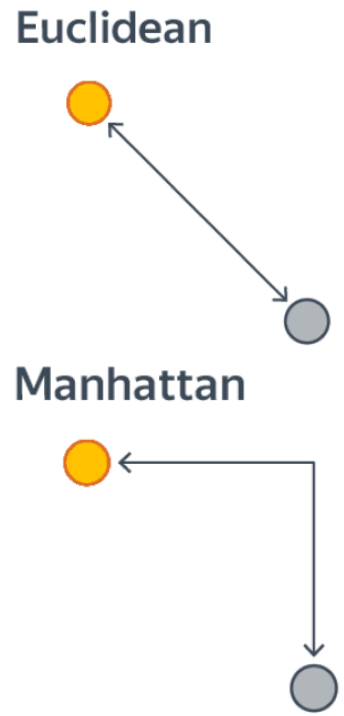


ВЫБОР МЕТРИКИ

В методе ближайших соседей мы вычисляем расстояния между объектами. Способов вычислить расстояние очень много, каждый из них задается своей формулой (метрикой). Каждая метрика больше подходит для своего типа задач.

Евклидова метрика - классический способ измерить расстояние между объектами. Пусть объекты *a* и *b* имеют координаты $a = (x_1, y_1), b = (x_2, y_2)$. Тогда евклидово расстояние между ними

$$\rho(a, b) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$



Манхеттенское расстояние - другой способ измерить расстояние между объектами:

$$\rho(a, b) = |x_1 - x_2| + |y_1 - y_2|$$

МАСШТАБИРОВАНИЕ ДАННЫХ

При использовании KNN в ситуации, когда объекты описываются вектором из числовых признаков **необходимо масштабировать данные.**

Приведем пример

Пусть мы ищем ближайшие объекты к объекту a=(40,1,100000),
где
40 - возраст человека,
1 - пол (1 - мужчина, 0 - женщина),
100000 - зарплата.

Какой объект будет ближайшим к объекту a по евклидовой метрике?
b=(80,0,90000)
c=(38,0,50000)
d=(25,1,1000000)

ОБОБЩЕНИЯ KNN

У классического KNN есть один большой недостаток - он никак не учитывает расстояния до ближайших объектов. В то же время понятно, что объекты, находящиеся ближе к объекту запроса, должны вносить больший вклад в ответ. Также учёт расстояний будет очень важен, когда мы будем применять KNN для решения задач регрессии.

- можно упорядочить соседей по увеличению расстояния от объекта запроса u и давать им вес, обратно пропорциональный номеру соседа: $w_k = \frac{1}{k}$, то есть

$$a(u) = argmax_{y \in Y} \sum_{i=1}^{k} \frac{I[y_i = y]}{k}$$

ОБОБЩЕНИЯ KNN

У классического KNN есть один большой недостаток - он никак не учитывает расстояния до ближайших объектов. В то же время понятно, что объекты, находящиеся ближе к объекту запроса, должны вносить больший вклад в ответ. Также учёт расстояний будет очень важен, когда мы будем применять KNN для решения задач регрессии.

- способ учесть расстояния называется *методом Парзеновского окна:*

$$a(u) = argmax_{y \in Y} \sum_{i=1}^{k} K\left(\frac{\rho(u, x_i)}{h}\right) I[y_i = y],$$

где u - вектор признаков запроса, x_i - вектор признаков *i*-го ближайшего соседа, h - ширина окна.

ОБОБЩЕНИЯ KNN

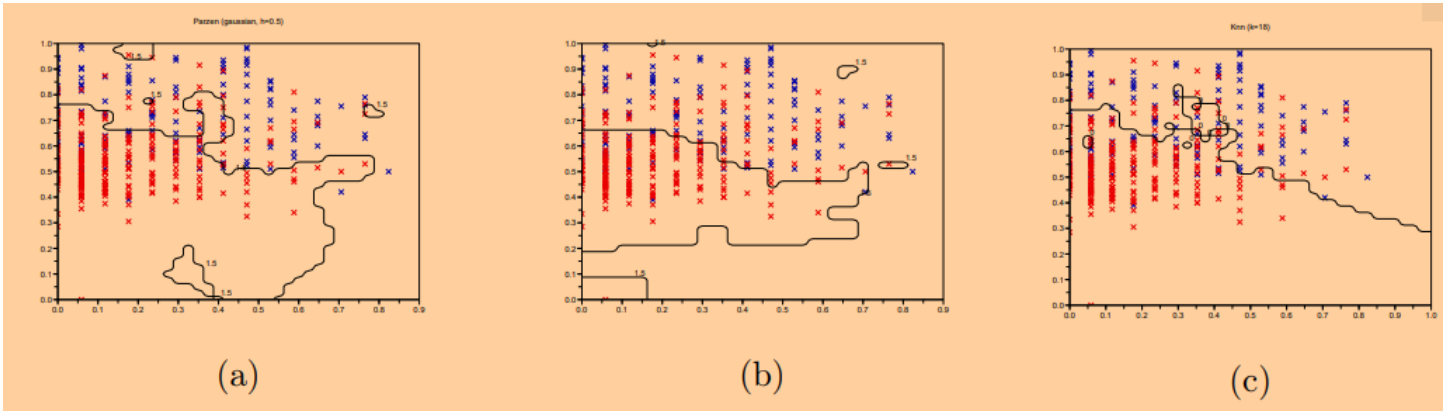
- способ учесть расстояния называется *методом Парзеновского окна:*

$$a(u) = argmax_{y \in Y} \sum_{i=1}^{k} K\left(\frac{\rho(u, x_i)}{h}\right) I[y_i = y],$$

где u - вектор признаков запроса, x_i - вектор признаков *i*-го ближайшего соседа, h - ширина окна.

Функция $K(x)$ называется ядром. Существует множество различных ядер, например:

- $K(x) = \frac{1}{2} I[|x| \leq 1]$ (прямоугольное ядро)
- $K(x) = (1 - |x|) \cdot I[|x| \leq 1]$ (треугольное ядро)
- $K(x) = \frac{1}{\sqrt{2\pi}} e^{-2x^2}$ (гауссовское ядро)



KNN В ЗАДАЧЕ РЕГРЕССИИ

С помощью KNN можно решать не только задачи классификации, но и задачи регрессии. Существует как простой, так и более сложные подходы - все они полностью аналогичны подходам в задачах классификации.

- Простой способ:

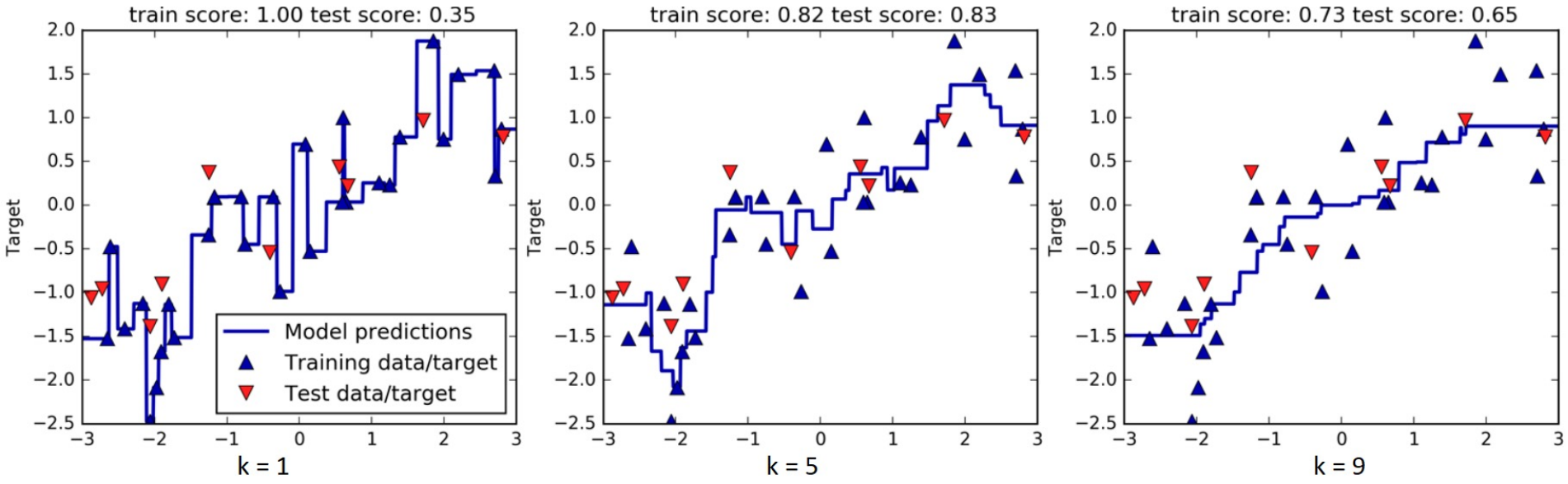
$$a(q) = \frac{1}{k}\sum_{i=1}^{k} y_k$$

- Взвешенный вариант (формула Надарая-Ватсона):

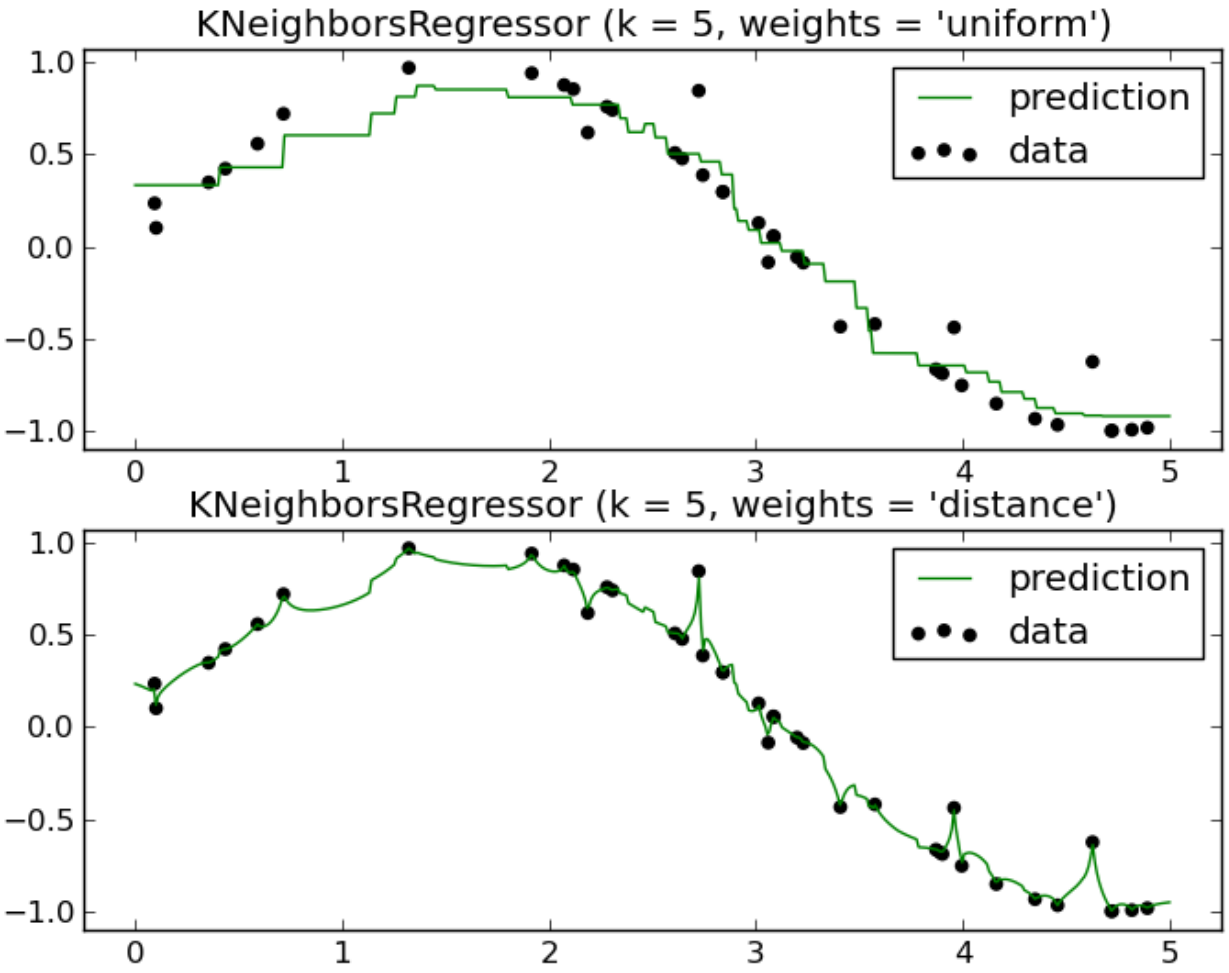
$$a(q) = \frac{\sum_{i=1}^{k} K\left(\frac{\rho(q, x_i)}{h}\right) y_i}{\sum_{i=1}^{k} K(\frac{\rho(q, x_i)}{h})}$$

На результат предсказания, как и в задаче классификации, влияет и число соседей, и выбор формулы для предсказания.

ВЛИЯНИЕ ЧИСЛА СОСЕДЕЙ



ВЛИЯНИЕ СПОСОБА УЧЕТА РАССТОЯНИЙ



ПЛЮСЫ И МИНУСЫ KNN

Преимущества KNN:

- Простой алгоритм (для объяснения и для интерпретации)
- Метод не делает никаких предположений о данных (об их линейной разделимости, о распределении данных)
- В некоторых задачах достаточно хорошо работает
- Применяется и для классификации, и для регрессии

Недостатки KNN:

- Требует больших ресурсов по памяти, так как хранит всю выборку
- Требует больших ресурсов по времени, так как вычисляет расстояния до всех объектов выборки
- Чувствителен к масштабу данных
- Зависит от выбранной метрики, которая в свою очередь должна отражать реальное сходство объектов. Найти такую метрику не всегда просто или даже возможно