Обратная связь

Метрики

Матрица ошибок

		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

Точность и полнота

- Точность и полнота напрямую зависят от порога, по которому мы переводим предсказанные моделью вероятности в классы.

Пример:
- если взять высокий порог, например, 0.8 или 0.9, то мы будем называть +1 только объекты, в которых классификатор очень сильно уверен. Тогда мы будем максимизировать точность
- если же сдвигать порог ближе к нулю, то наоборот, точность будет падать, зато полнота подрастет.

Точность и полнота

- Точность и полнота напрямую зависят от порога, по которому мы переводим предсказанные моделью вероятности в классы.

Пример:

- если взять высокий порог, например, 0.8 или 0.9, то мы будем называть +1 только объекты, в которых классификатор очень сильно уверен. Тогда мы будем максимизировать точность
- если же сдвигать порог ближе к нулю, то наоборот, точность будет падать, зато полнота подрастет.

Точность и полнота

- Точность и полнота напрямую зависят от порога, по которому мы переводим предсказанные моделью вероятности в классы.
- Эти метрики зависят от порога
- Хочется, чтобы была метрика, учитывающая всевозможные пороги

ROC-AUC: интуиция

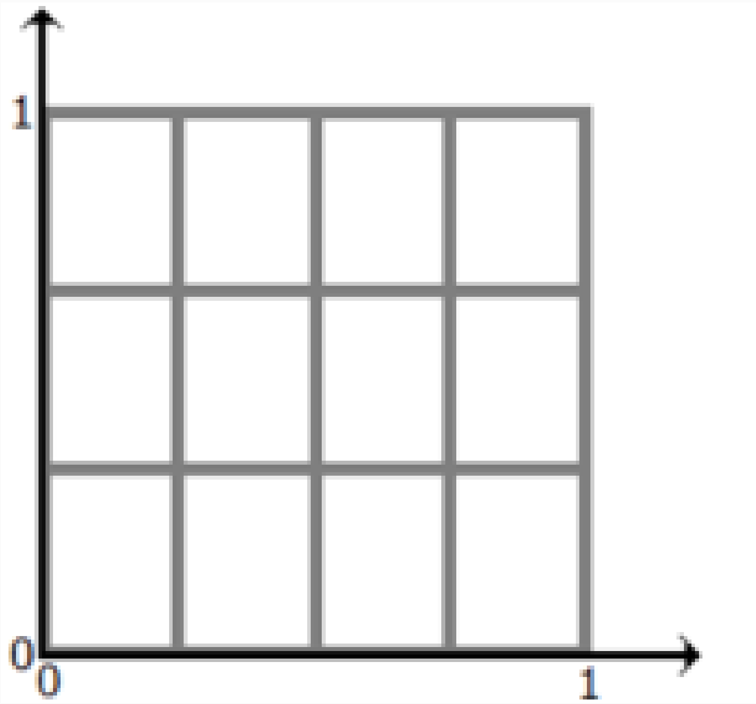
- Пример:

p	класс
0.5	0
0.1	0
0.25	0
0.6	1
0.2	1
0.3	1
0.0	0

p	класс
0.6	1
0.5	0
0.3	1
0.25	0
0.2	1
0.1	0
0.0	0

ROC-AUC: алгоритм

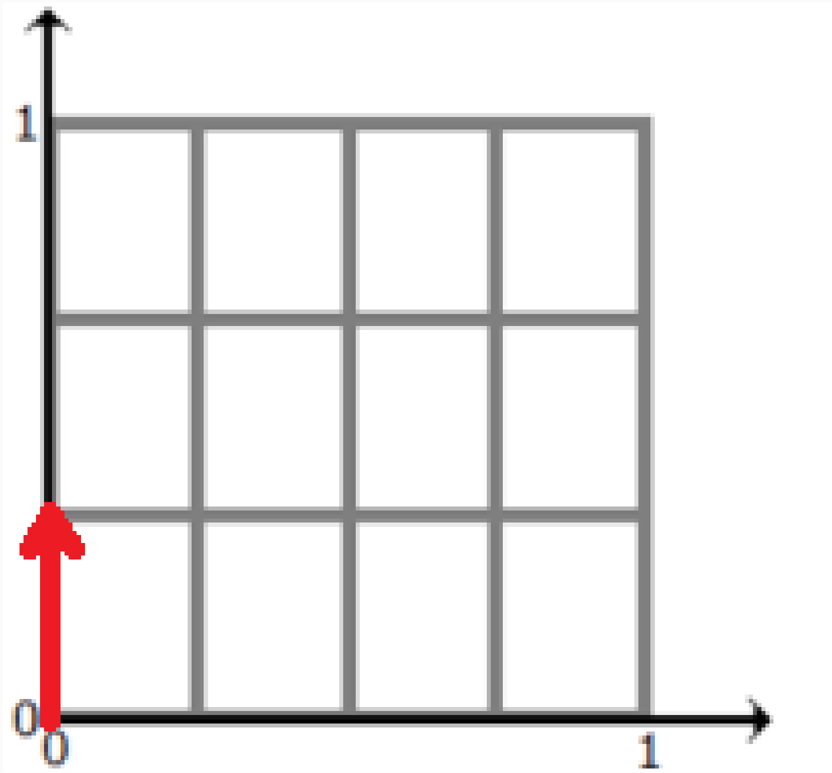
- Нарисуем квадрат 1 на 1.
- Горизонтальную сторону квадрата разобъем на равные отрезки, число которых равно числу 0 в данных
- Вертикальную сторону разобъем на равные отрезки, число которых равно числу 1



ROC-AUC: алгоритм

- Нарисуем квадрат 1 на 1.
- Горизонтальную сторону квадрата разобъем на равные отрезки, число которых равно числу 0 в данных
- Вертикальную сторону разобъем на равные отрезки, число которых равно числу 1

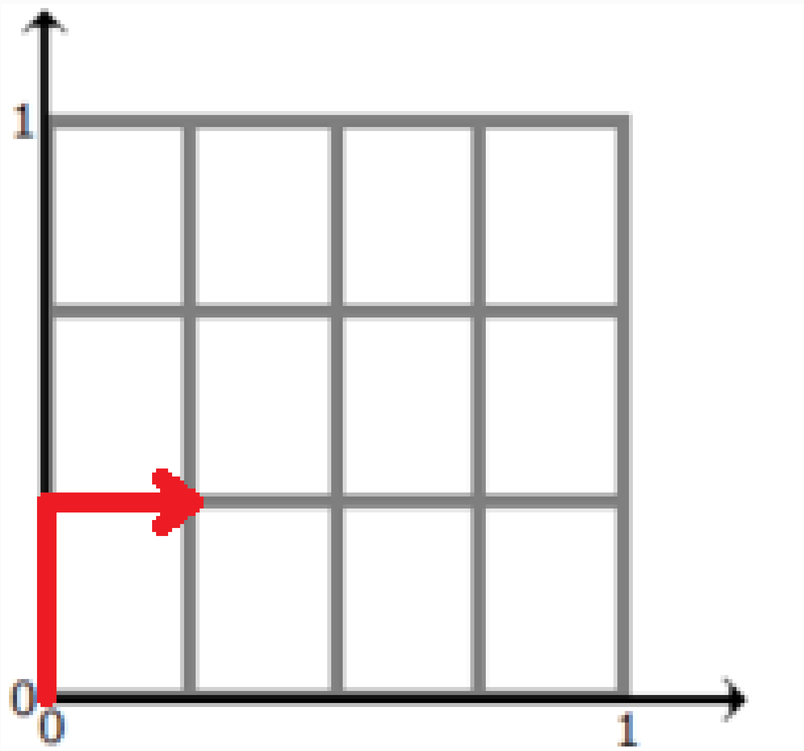
p	класс
0.6	1
0.5	0
0.3	1
0.25	0
0.2	1
0.1	0
0.0	0



ROC-AUC: алгоритм

- Нарисуем квадрат 1 на 1.
- Горизонтальную сторону квадрата разобъем на равные отрезки, число которых равно числу 0 в данных
- Вертикальную сторону разобъем на равные отрезки, число которых равно числу 1

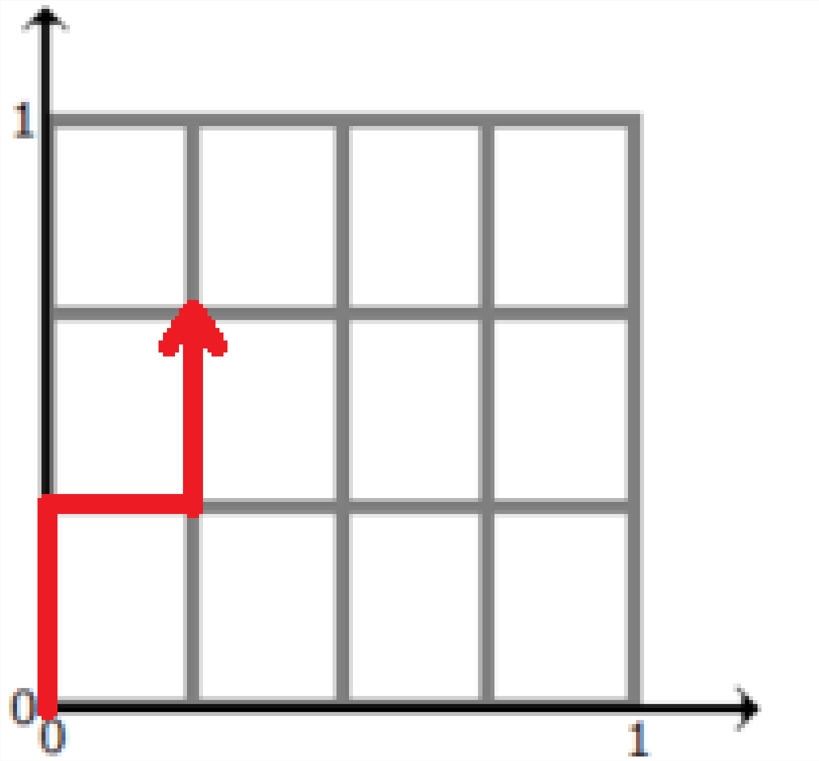
p	класс
0.6	1
0.5	0
0.3	1
0.25	0
0.2	1
0.1	0
0.0	0



ROC-AUC: алгоритм

- Нарисуем квадрат 1 на 1.
- Горизонтальную сторону квадрата разобъем на равные отрезки, число которых равно числу 0 в данных
- Вертикальную сторону разобъем на равные отрезки, число которых равно числу 1

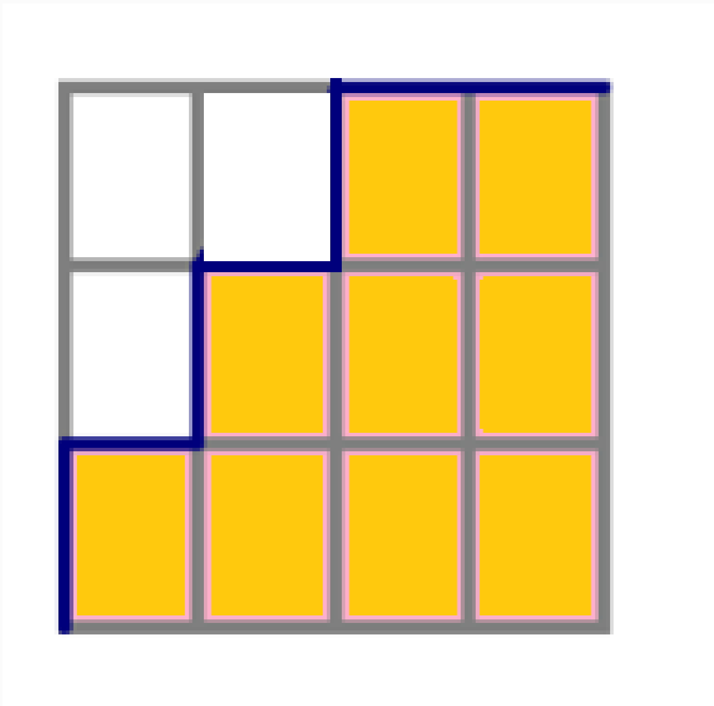
р	класс
0.6	1
0.5	0
0.3	1
0.25	0
0.2	1
0.1	0
0.0	0



ROC-AUC: алгоритм

- Пойдем по отсортированной таблице по столбцу класс сверху вниз
- Будем стартовать из точки (0,0) на квадрате. И если мы встречаем 1, сдвигаемся на одну клеточку вверх, а если 0 - то вправо
- В итоге мы придём в точку (1,1).

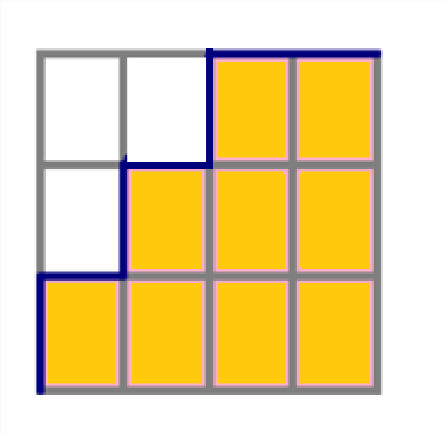
p	класс
0.6	1
0.5	0
0.3	1
0.25	0
0.2	1
0.1	0
0.0	0



Полученная кривая называется ROC-кривой, а метрика, равная площади под ней - AUC-ROC.

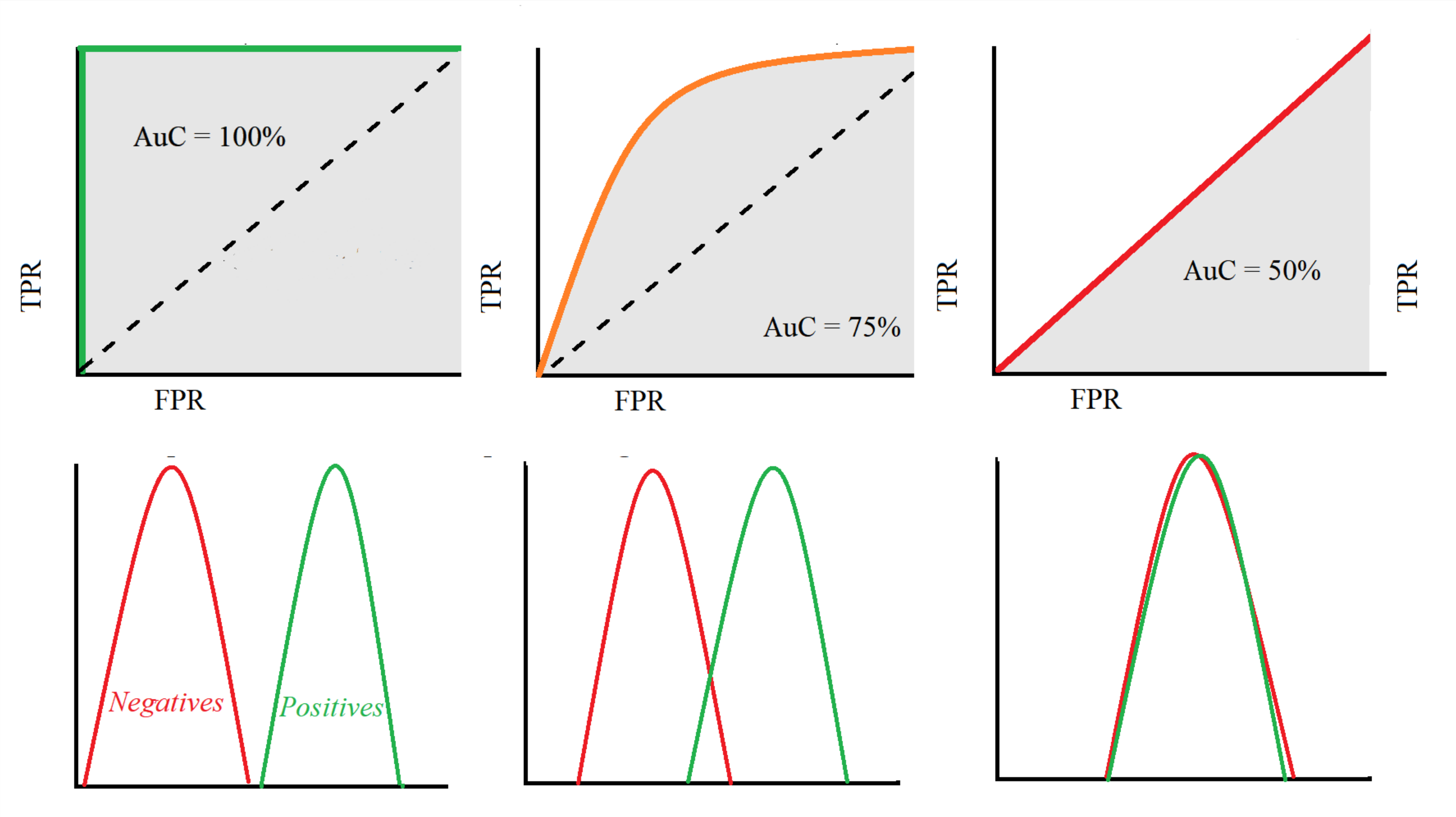
ROC-AUC: пояснение

- Если взять идеальную модель, то все 1 будут идти друг за другом в отсортированной таблице, то есть сначала мы из (0,0) будем идти всё время вверх, и только затем в таблице пойдут нули, и мы будем идти по кривой всё время вправо. То есть для идеальной модели ROC-кривая - это квадрат 1 на 1, и AUC-ROC = 1.
- Если же модель просто произвольно угадывает ответ, то мы кривая будет похожа на диагональ квадрата, и площадь под ней будет 0.5.



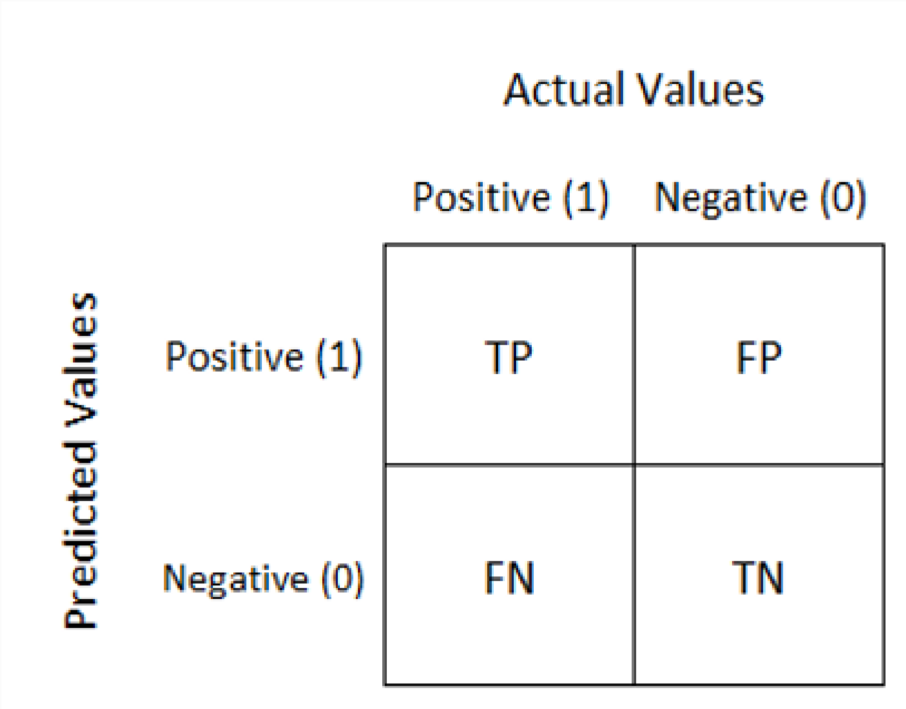
Чем лучше модель (чем разумнее она предсказывает вероятности и тем самым чем лучше упорядочивает объекты классов), тем больше у неё AUC-ROC.

ROC-AUC: примеры



TPR и FPR

- Переведем вероятности в классы по некоторому порогу и построим матрицу ошибок:



TPR и FPR

		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

По матрице ошибок можно посчитать

- False Positive Rate:

$$FPR = \frac{FP}{FP + TN}$$

- это доля неверно принятых объектов отрицательного класса.

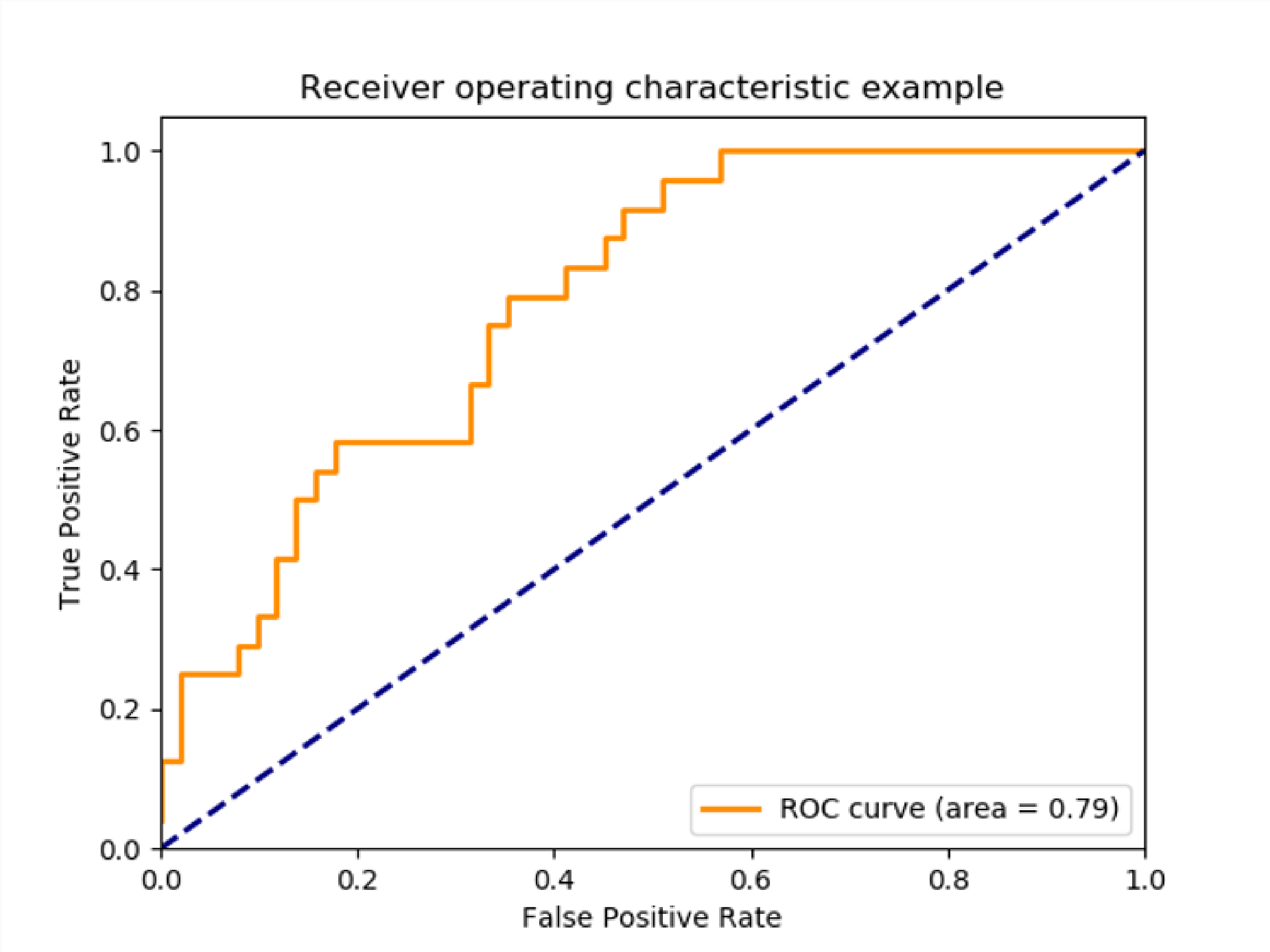
- True Positive Rate:

$$TPR = \frac{TP}{TP + FN}$$

- это доля верно принятых объектов положительного класса.

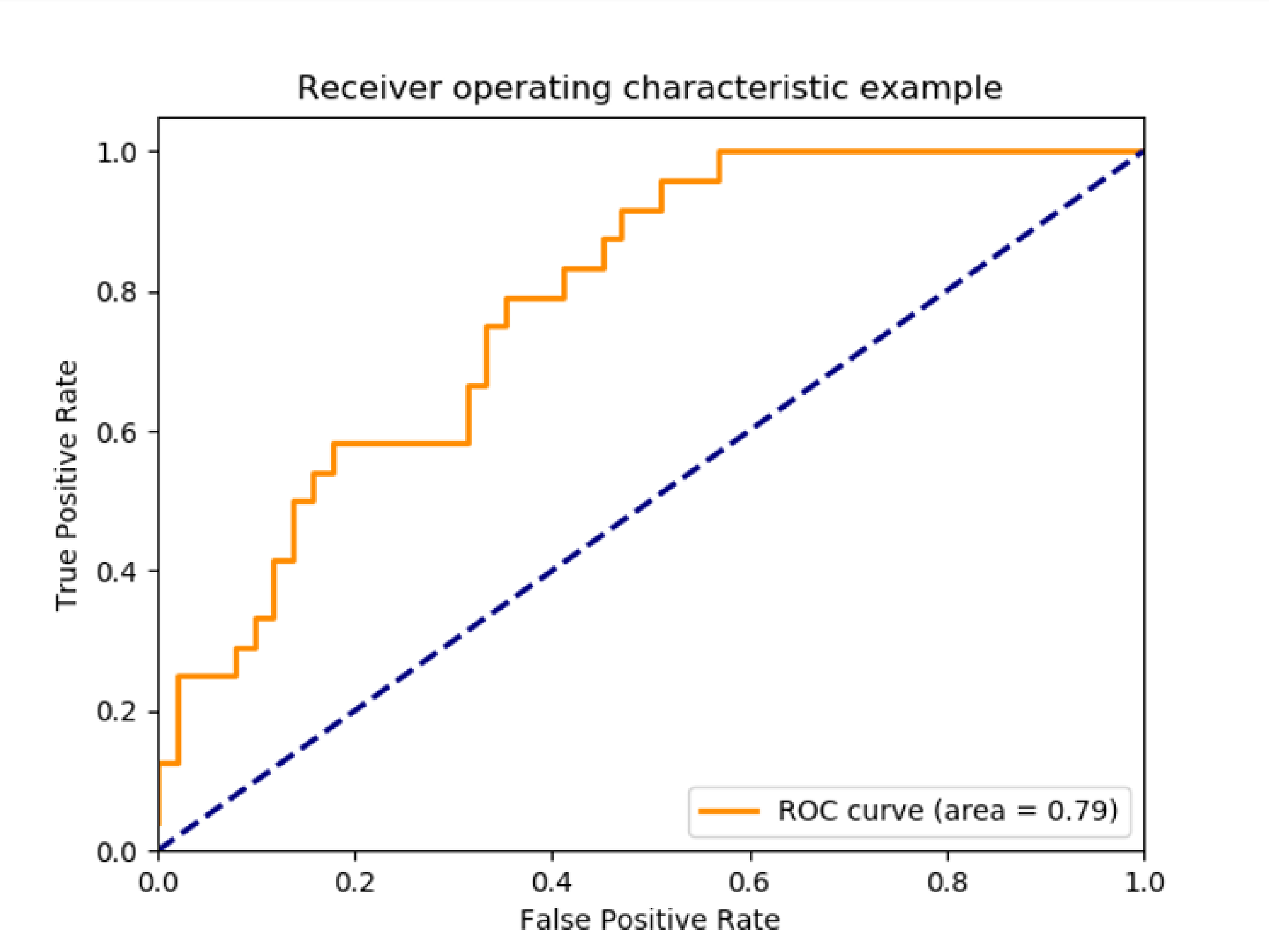
ROC-AUC

- Для каждого возможного порога переведем вероятности, предсказанные моделью, в классы
- Затем посчитаем пару значений (FPR, TPR) и отметим точку с этими координатами на плоскости
- Затем соединим полученные точки кривой - эта кривая и называется **ROC-кривой**.



ROC-AUC

- Так как объектов в выборке конечное число, то в реальности не нужно рассматривать всевозможные числа на отрезке [0; 1] в качестве порогов.
- Можно взять только пороги, совпадающие с предсказанными моделью вероятностями (число порогов равное числу различных предсказанных вероятностей)



ROC-AUC: формальный алгоритм

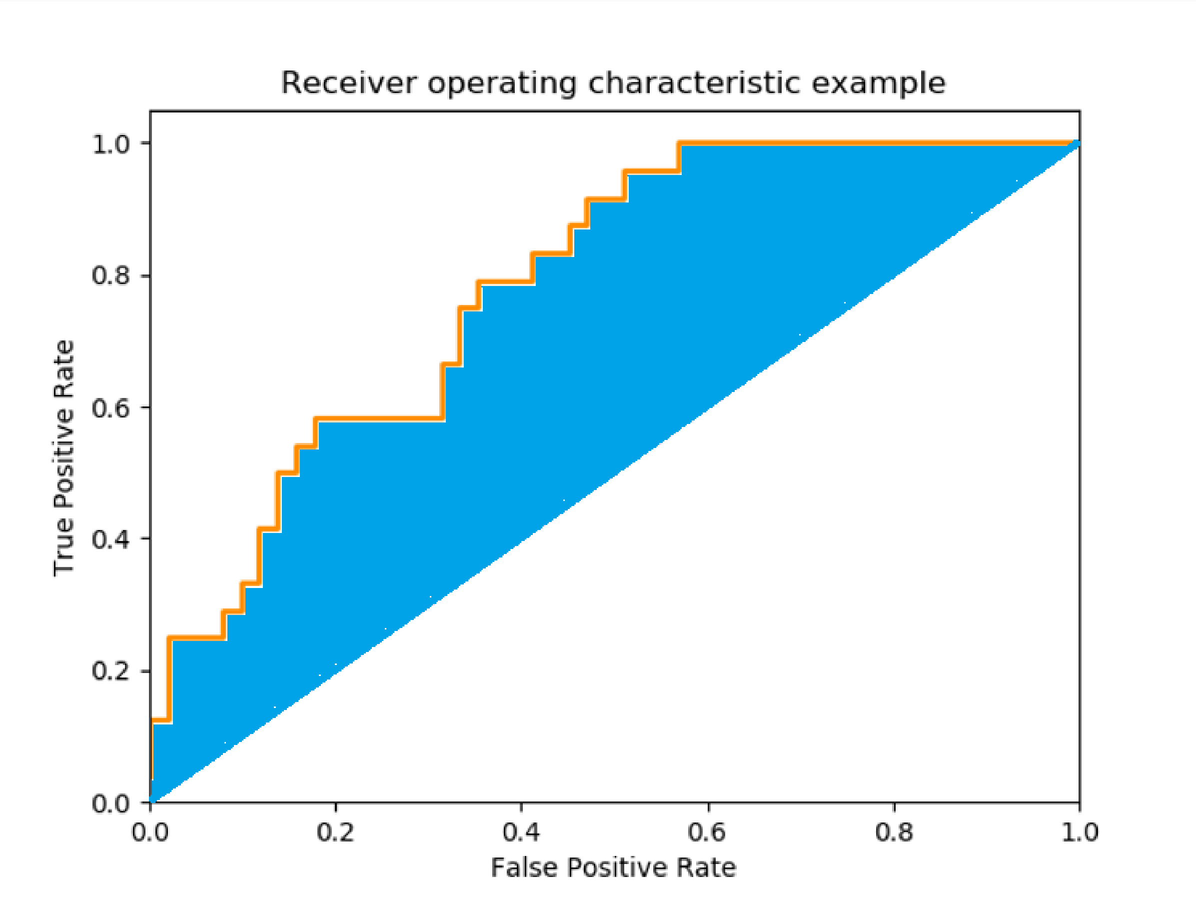
- Для каждого из порогов, заданных предсказанными моделью вероятностями, необходимо вычислить значения (FPR,TPR) и поставить точку с этими координатами на плоскости.
- Затем получим ROC-кривую и посчитаем площадь под ней.

Коэффициент Gini

- Коэффициент Gini (Gini Impurity) – это удвоенная площадь под ROC-кривой и над главной диагональю квадрата:

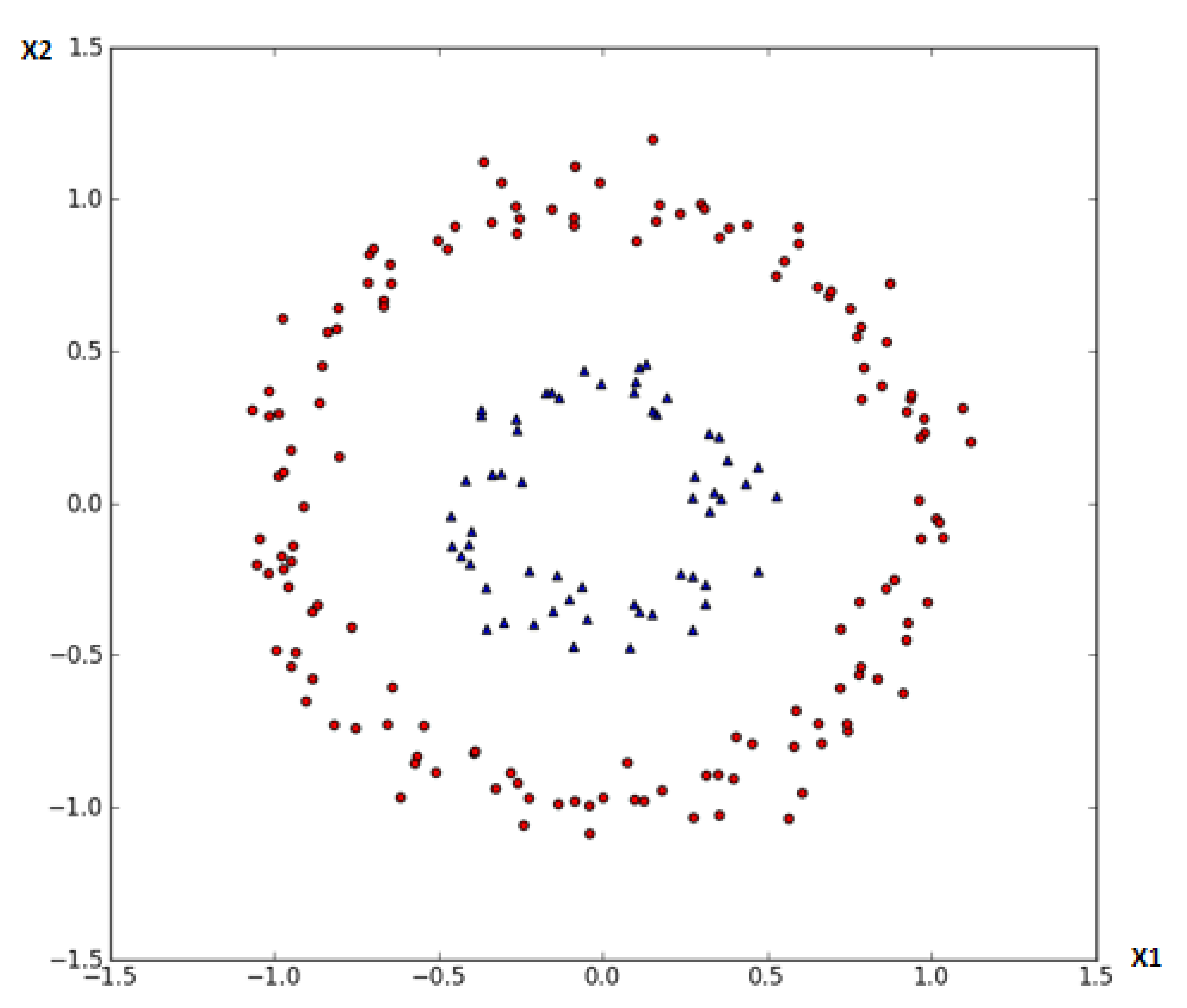
$$Gini = 2 \cdot AUC - 1$$

- $Gini \in [0; 1]$ и ведёт себя похожим образом на ROC-AUC.



Ядровой SVM

Пример



Пример

Добавим признак

$$z = \sqrt{x_1^2 + x_2^2}$$

Тогда в новом пространстве признаков (x_1, x_2, z) точки становятся линейно-разделимы!

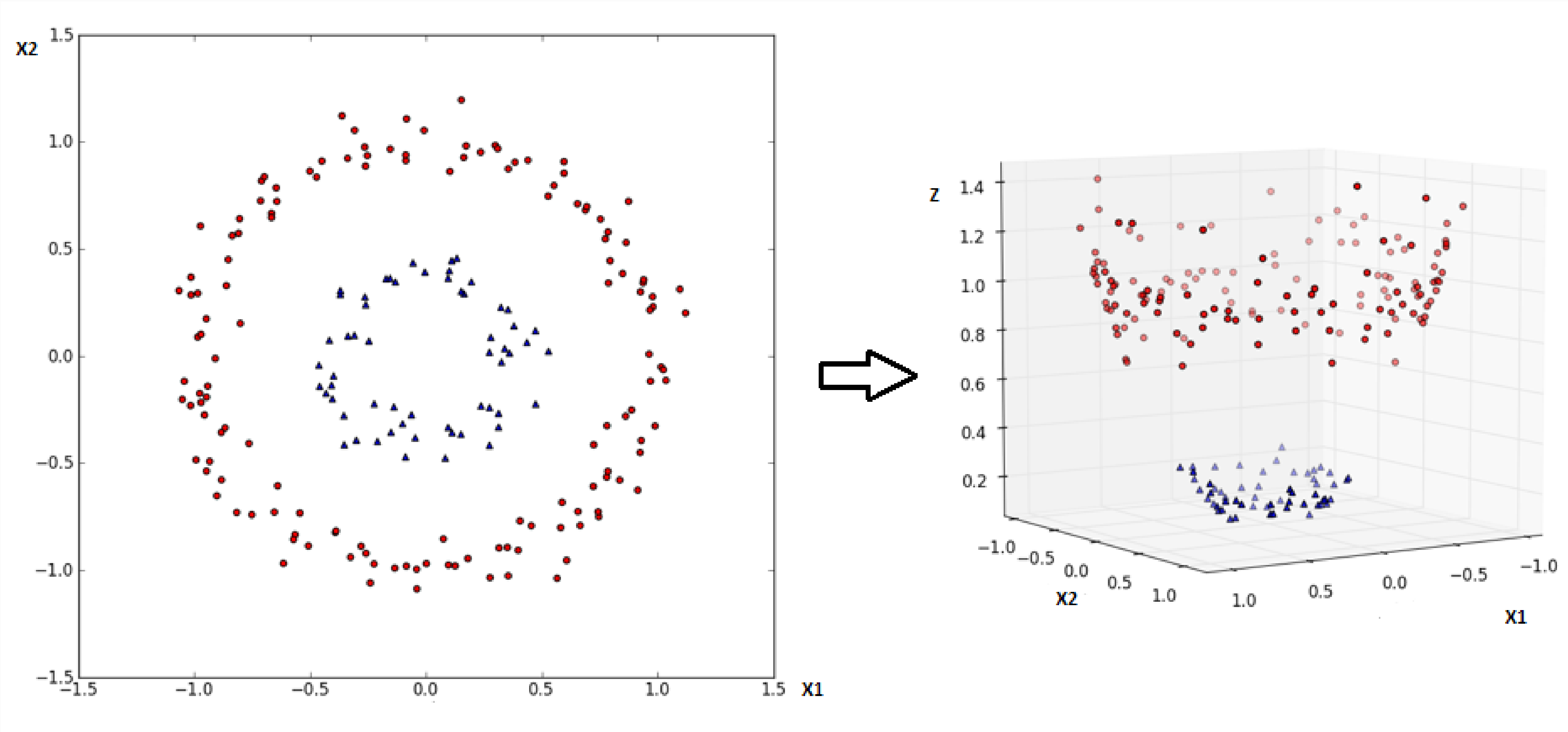


Схема решения

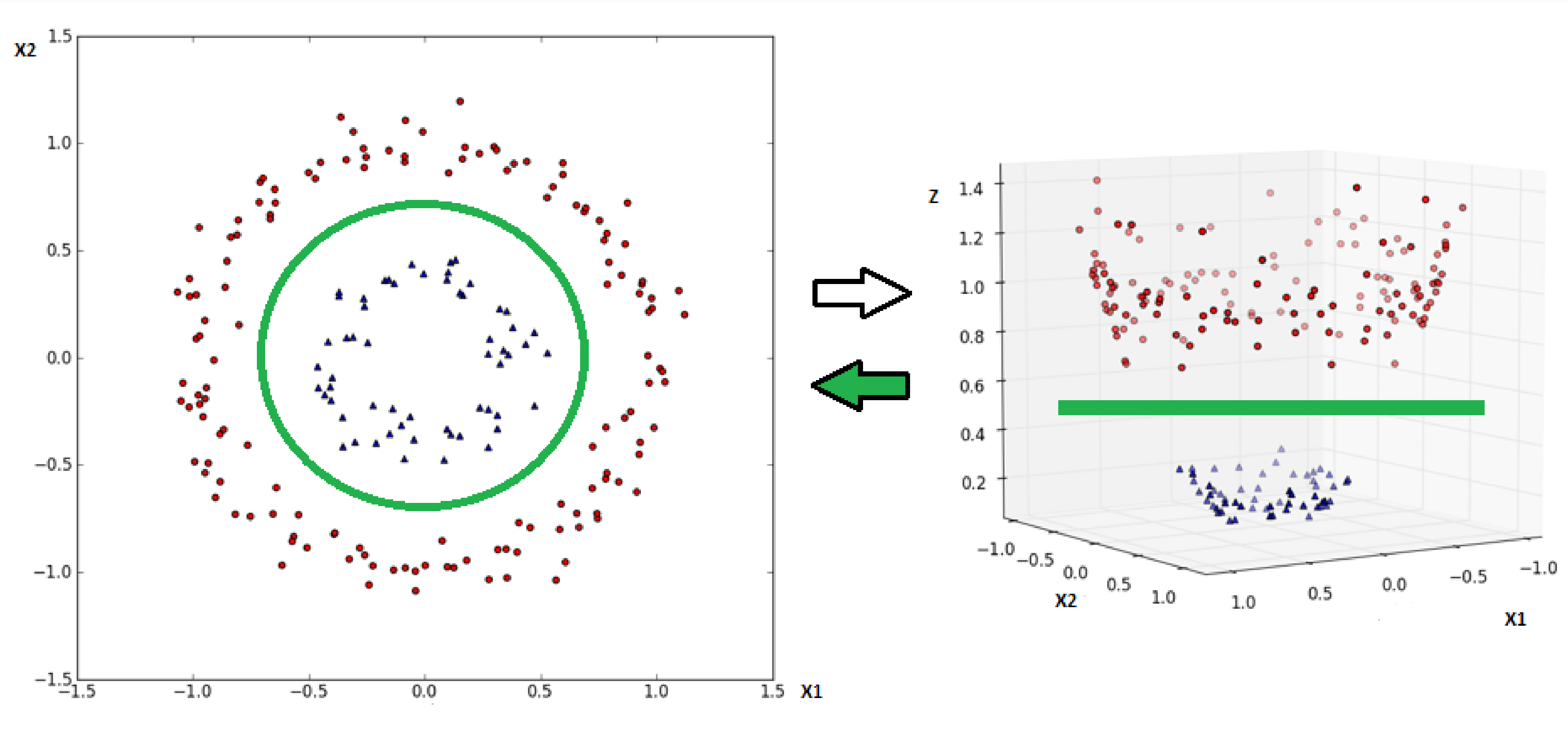
В задачах, где целевая переменная имеет более сложную зависимость от признаков, чем линейная, можно поступить так:

- Подбираем нелинейное преобразование признаков
$$x \to \varphi(x),$$
Чтобы в новом признаковом пространстве классы стали линейно разделимы
- Обучаем линейный классификатор на новых признаках $\varphi(x)$

Тем самым, с помощью преобразования признаков можно решать нелинейные задачи линейными классификаторами!

Схема решения

- Если нарисовать в исходном признаковом пространстве разделяющую поверхность обученного в новых признаках линейного классификатора, то получим нелинейную разделяющую поверхность!



Ядро

Пусть мы применили некоторое преобразование φ к исходным признакам x и получили новые признаки объекта $\varphi(x)$.

- Тогда ядро

$$K(a, b) = \left(\varphi(a), \varphi(b)\right)$$

- это скалярное произведение объектов a и b в новом признаковом пространстве.

Ядро

Пусть мы применили некоторое преобразование φ к исходным признакам x и получили новые признаки объекта $\varphi(x)$.

- Тогда ядро

$$K(a, b) = \big(\varphi(a), \varphi(b)\big)$$

- это скалярное произведение объектов a и b в новом признаковом пространстве.

- Ядро задает правила, по которым вычисляются расстояния и углы между объектами. Это необходимая информация для обучения модели.

Ядро

- Можно задавать преобразование φ и по нему считать функцию ядра K

$$K(a, b) = \big(\varphi(a), \varphi(b)\big)$$

- А можно сразу задавать K, не задавая φ.

- Ядро задает правила, по которым вычисляются расстояния и углы между объектами, поэтому знать только функцию K зачастую достаточно для обучения модели.

Ядра: примеры

- Функция является ядром, если она симметрична и неотрицательно определена (то есть ведёт себя как скалярное произведение)

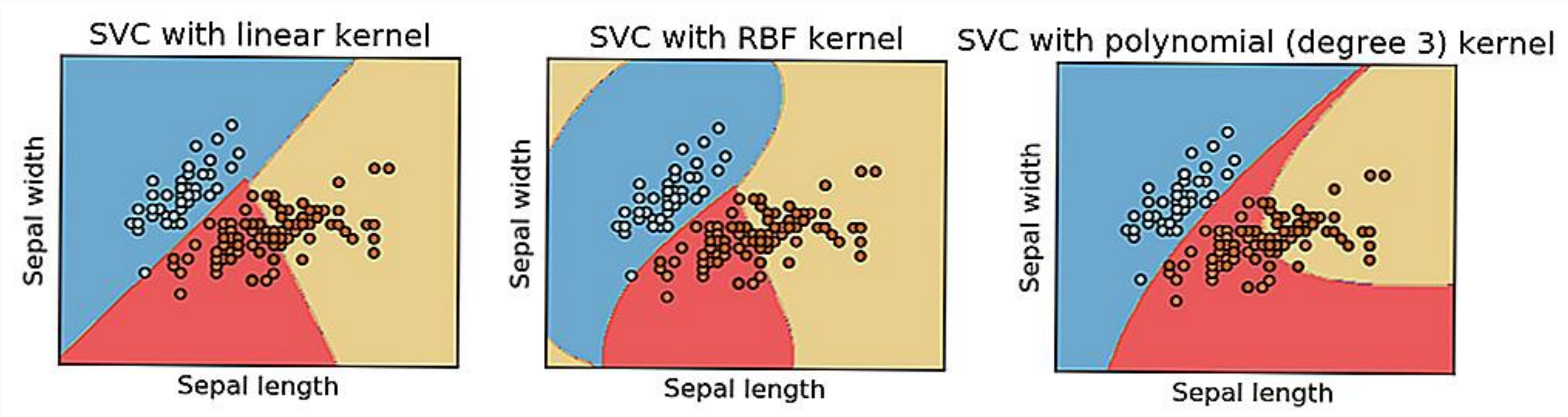
Примеры:

- $K(a, b) = (a, b)^2$ - ядро, так как эта функция симметрична и неотрицательно определена
- $K(a, b) = a - b$ – не ядро, так как функция не симметрична

Популярные ядра

Популярные ядра:

- Полиномиальное: $K(a,b) = (\gamma \cdot (a,b) + r)^d$
- Радиальное: $K(a,b) = \exp(-\gamma \cdot ||a-b||^2)$
- Сигмоидальное: $K(a,b) = \tanh(\gamma \cdot (a,b) + r)$



Ядровой SVM в деталях

https://github.com/esokolov/ml-course-hse/blob/master/2016-spring/lecture-notes/lecture16-kernels.pdf