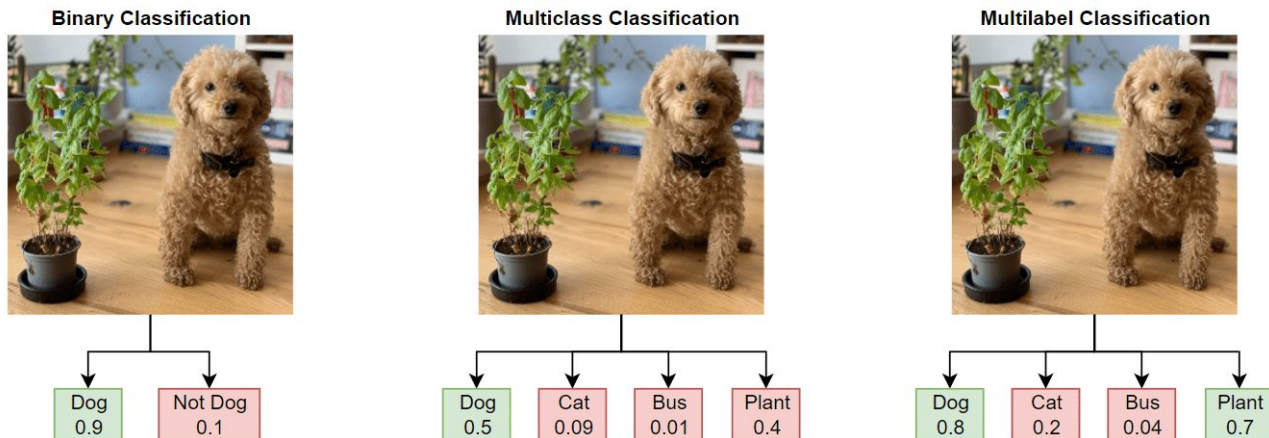


# Классификация

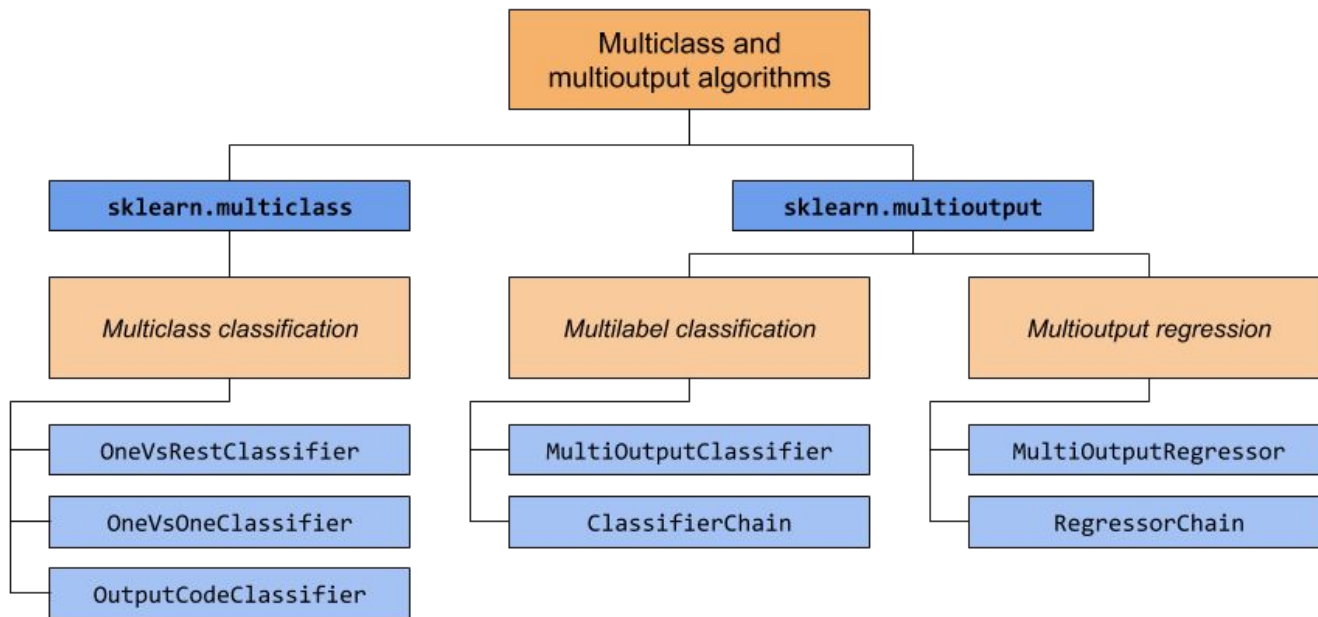
# Виды классификации

# Мультикласс, мультилейбл, мультиаутпут

Напоминание из прошлой лекции:



# Мультикласс, мультилейбл, мультиаутпут





# Мультикласс, мультилейбл, мультиаутпут

- Multiclass: выберите один из  $n > 2$  лейблов классов. Например: определите, какое животное из трех возможных изображено на картинке, исходя из того, что на каждой картинке изображено только одно животное.
- Multilabel: присвойте каждому вхождению  $m$  лейблов из  $n$  возможных классов, где  $m \leq n$  (предскажите названия всех животных, которые есть на картинке);
- Multiclass-multioutput\*: присвойте каждому вхождению  $n$  лейблов, каждый из которых может разделяться на  $m$  возможных классов (предскажите породу и цвет кошки).

Перед началом работы

# Подготовка: стратификация

- 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`).

# Подготовка: баланс классов

- Если классы не сбалансированы, модель может склоняться к превалирующему классу при предсказании. Этого можно избежать, либо выкинув лишние примеры из большего класса, либо присвоив классам веса:

```
lr_binary = LogisticRegression(class_weight={0:0.72, 1:0.28})
```

На картинке класс “1” составляет большую часть данных, поэтому ему присвоен меньший вес.

- В sklearn веса можно вычислить, например, с помощью функции `compute_class_weight`, а потом передать в параметр `class_weight` классификатора; либо использовать параметр `class_weight`, встроенный во многие модели.



# Примеры классификаторов

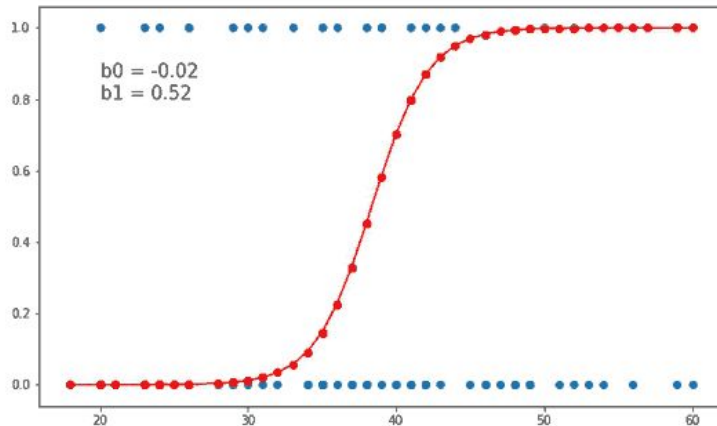
# Логистическая регрессия

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,  $x_1 \dots x_n$  - наши признаки,  $e$  - экспонента.

Почему экспонента? Потому что экспонента - это основание натурального логарифма.



Картинка:

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

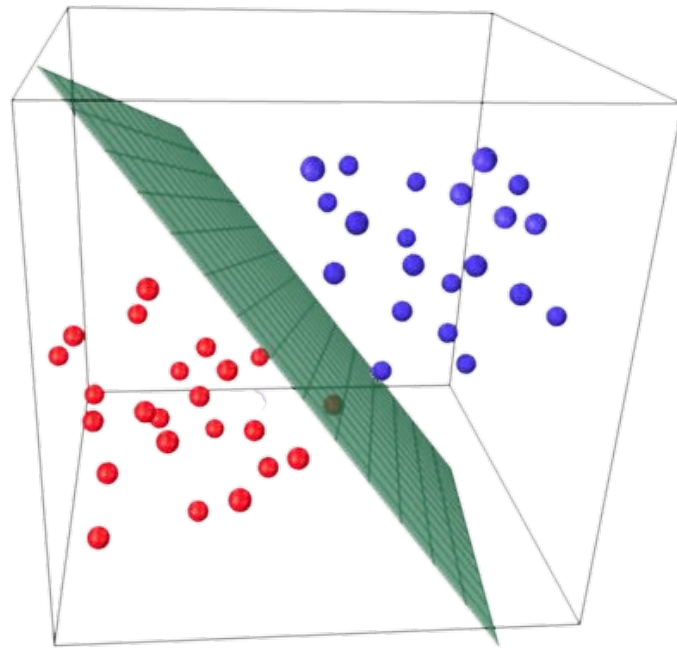
# Немного о том, как это работает

Нам необходимо разделить точки в пространстве гиперплоскостью.

Уравнение плоскости будет иметь вид:


$$w_0 + w_1x_1 + w_2x_2 + [\dots] + w_nx_n = 0$$

Здесь  $w$  - параметры (веса), а  $x$  - наши признаки. По значению весов (коэффициентов) можно понять важность того или иного признака.



## Немного о том, как это работает

Нам нужно найти вероятность положительного класса, т.е. число от 0 до 1. Но наше выражение с предыдущего слайда имеет пределы  $[-\infty; \infty]$ . Нам нужно, чтобы обе части выражения имели одинаковые пределы. Для этого нам понадобится формула отношения шансов и натуральный логарифм.

$$\ln\left(\frac{p}{1-p}\right) = w_0 + w_1x_1 + w_2x_2 + [\dots] + w_nx_n$$


Это формула отношения шансов. Сама по себе она имеет пределы от 0 до бесконечности  $[0; \infty]$ . Мы применили логарифм к отношению шансов, и теперь оно имеет пределы  $[-\infty; \infty]$ .

# Немного о том, как это работает

Теперь перепишем это выражение и найдем вероятность  $p$ :

$$\ln\left(\frac{p}{1-p}\right) = w_0 + w_1x_1 + w_2x_2 + [\dots] + w_nx_n$$

$$\frac{p}{p-1} = e^{w_0+w_1x_1+w_2x_2+[\dots]+w_nx_n}$$

Зависимость вероятности от отношения шансов OR рассчитывается по следующей формуле:

$$P = \frac{OR}{1 + OR}$$

# Немного о том, как это работает

Таким образом:

$$p = \frac{e^{(w_0 + w_1 x_1 + w_2 x_2 + [\dots] + w_n x_n)}}{1 + e^{(w_0 + w_1 x_1 + w_2 x_2 + [\dots] + w_n x_n)}} \longrightarrow$$

Похоже на эту формулу с  
первого слайда?

$$P(Y = 1 | x_1, [\dots]) = \frac{e^{(w_0 + w_1 x_1 + [\dots])}}{1 + e^{(w_0 + w_1 x_1 + [\dots])}}$$

$$p = \frac{1}{1 + e^{-(w_0 + w_1 x_1 + w_2 x_2 + [\dots] + w_n x_n)}} \longrightarrow$$

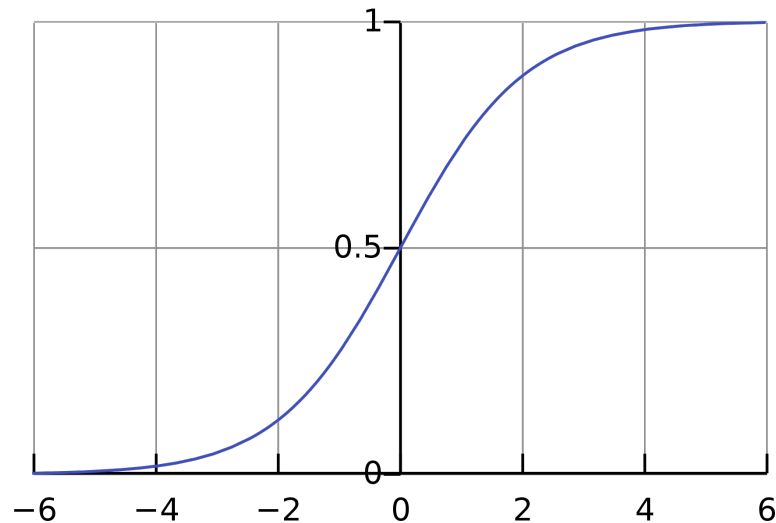
А это похоже на сигмоидную  
функцию

# Немного о том, как это работает

В итоге мы получили формулу, очень похожую на сигмоидную функцию:

$$F(x) = \frac{1}{1 + e^{-x}}$$

Сигмоида принимает значения от 0 до 1 и часто применяется в алгоритмах классификации, в том числе в нейронных сетях.



Картинка:

<https://commons.wikimedia.org/wiki/File:Logistic-curve.svg#/media/File:Logistic-curve.svg>

# 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.

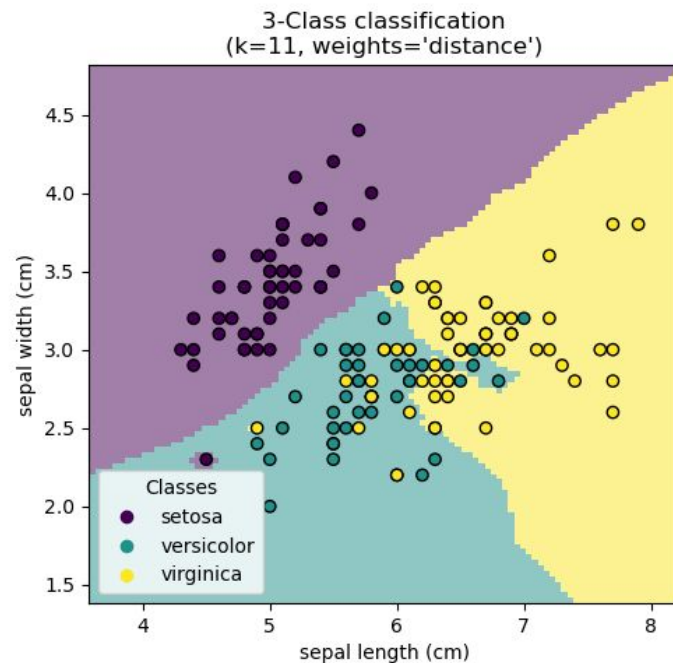
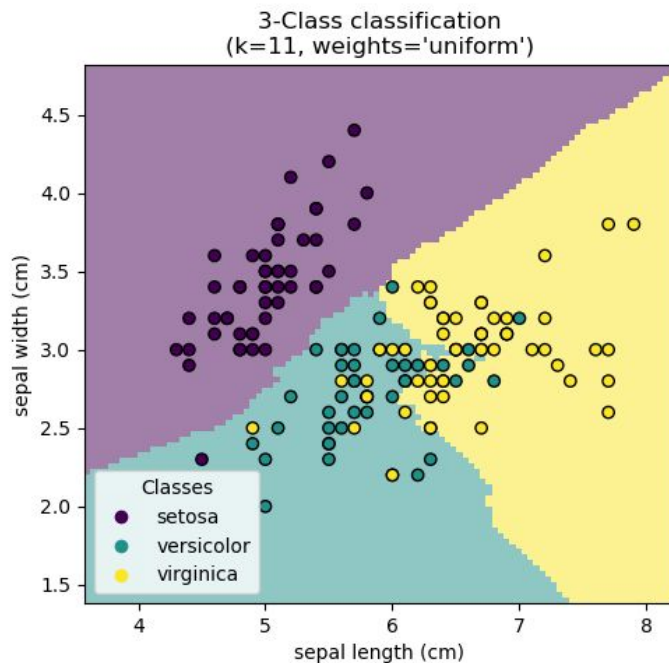


## К ближайших соседей

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](https://scikit-learn.org/stable/auto_examples/neighbors/plot_classification.html)

# Внимание: неоднозначная терминология

- LDA != LDA
  - Linear Discriminant Analysis - supervised алгоритм, который можно использовать для снижения размерности и классификации. Latent Dirichlet Allocation - unsupervised алгоритм, может использоваться для выделения тем в неаннотированных данных.
- kNN != k-means
  - В отличие от К ближайших соседей, К средних - unsupervised алгоритм, часто используемый для кластеризации.
- Decision trees != Random Forest
  - Деревья принятия решений - это семейство алгоритмов классификации и регрессии. Рандомный лес тоже основан на древесном алгоритме, но используется для ансамблевого обучения.

# Методы оценки качества

# Оценка качества классификации

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

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

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

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

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.

# Почему F-мера, а не accuracy?

Accuracy не очень хорошо работает для несбалансированных классов. Например, имеется 100 тестовых образцов, 90 из которых принадлежат классу 1, а 10 - классу 0. Модель предсказывает, что все 100 образцов принадлежат классу 1. Таким образом, accuracy =  $90/100 = 0.9$ , что на самом деле неприемлемо.

F-мера, как гармоническое среднее, будет близка к нулю, когда хотя бы один из аргументов (точность или полнота) близок к нулю. Рассмотрим, например, случай, когда в  $y\_true$  50 нулей и 50 единиц, а в  $y\_pred$  - 20 единиц и 80 нулей, но все 20 единиц предсказаны верно. Это даст нам precision = 1, но recall всего лишь 0.4. F-мера при этом будет равна 0.57, а accuracy = 0.7.

Ноутбук про значения accuracy и f1-score на несбалансированных данных:  
[https://colab.research.google.com/drive/1uVG12mOAylkalQK9QV2bC-tRN3lmylfn?  
usp=sharing](https://colab.research.google.com/drive/1uVG12mOAylkalQK9QV2bC-tRN3lmylfn?usp=sharing)

# Микро-, макро-, взвешенное усреднение

|    | precision    | recall | f1-score | support |    |
|----|--------------|--------|----------|---------|----|
| 0  | 0.67         | 0.67   | 0.67     | 6       |    |
| 1  | 1.00         | 0.33   | 0.50     | 9       |    |
| 2  | 0.45         | 1.00   | 0.62     | 5       |    |
| ?? | accuracy     |        | 0.60     | 20      |    |
|    | macro avg    | 0.71   | 0.67     | 0.60    | 20 |
|    | weighted avg | 0.76   | 0.60     | 0.58    | 20 |



# Микро-, макро-, взвешенное усреднение

На примере точности:

$\text{Precision} = \text{true\_positive} / (\text{true\_positive} + \text{false\_positive})$

- Микроусреднение: считаем среднее количество TP и FP по всем классам. Эти числа подставляем в формулу precision;
- Макроусреднение: считаем точность по всем классам, усредняем;
- Взвешенное усреднение: макроусреднение, но каждое значение точности домножается на вес класса (отношение числа примеров для этого класса к числу всех примеров в выборке).

Тетрадка с примером:

<https://colab.research.google.com/drive/1QNKgcg4wSxgq8vmZ5cMxA3mWJiQIdDS4?usp=sharing>

# ROC curve

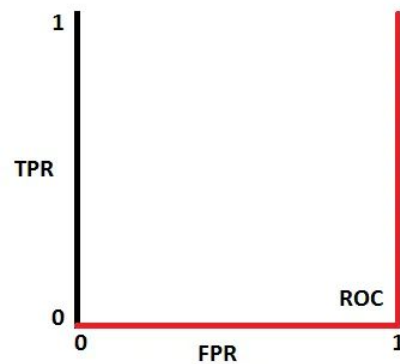
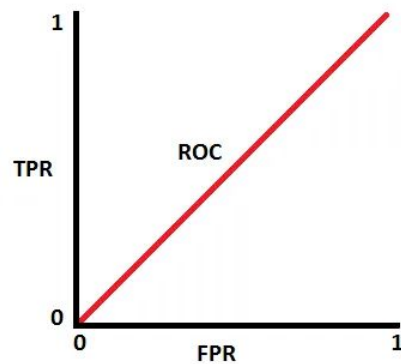
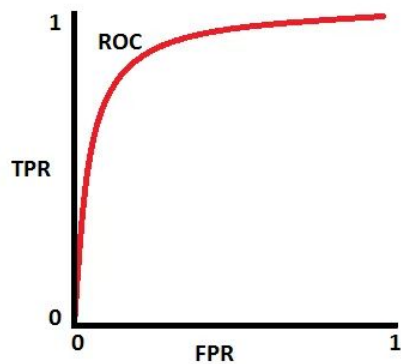
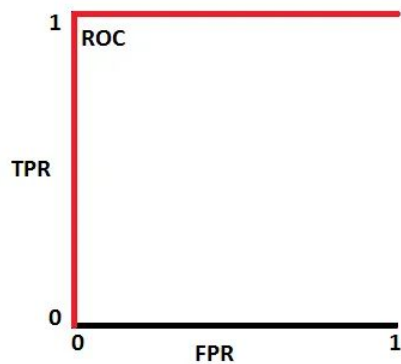
- ROC = receiver operating curve или кривая ошибок. Отражает отношение между чувствительностью алгоритма классификации (true positive rate) и его специфичностью (false positive rate).

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

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

- Классификаторы выдают вероятности принадлежности элемента к какому-либо классу. Чувствительность и специфичность будут отличаться в зависимости от того, какой порог вероятности мы возьмем. Обычно берется порог  $\geq 0.5$ . Идеальный классификатор, который всегда прав, при любом пороге будет выдавать  $TPR = 1$ . Кривая ошибок отражает зависимость TPR и FPR от выбранного порога вероятности.

# ROC curve



# ROC curve - вычисление

В таблице 1 представлены оценки вероятности, выданные классификатором (“оценка”), и истинные значения классов (“класс”). В таблице 2 все столбцы упорядочены по убыванию оценок. В таблице 3 представлены округленные значения вероятностей с порогом  $> 0.25$ .

| id | оценка | класс |
|----|--------|-------|
| 1  | 0.5    | 0     |
| 2  | 0.1    | 0     |
| 3  | 0.2    | 0     |
| 4  | 0.6    | 1     |
| 5  | 0.2    | 1     |
| 6  | 0.3    | 1     |
| 7  | 0.0    | 0     |

Табл. 1

| id | оценка | класс |
|----|--------|-------|
| 4  | 0.6    | 1     |
| 1  | 0.5    | 0     |
| 6  | 0.3    | 1     |
| 3  | 0.2    | 0     |
| 5  | 0.2    | 1     |
| 2  | 0.1    | 0     |
| 7  | 0.0    | 0     |

Табл. 2

| id | $> 0.25$ | класс |
|----|----------|-------|
| 4  | 1        | 1     |
| 1  | 1        | 0     |
| 6  | 1        | 1     |
| 3  | 0        | 0     |
| 5  | 0        | 1     |
| 2  | 0        | 0     |
| 7  | 0        | 0     |

Табл. 3

Источник в блоге Александра Дьяконова:

<https://alexanderdyakonov.wordpress.com/2017/07/28/auc-roc-%D0%BF%D0%BB%D0%BE%D1%89%D0%B0%D0%B4%D1%8C-%D0%BF%D0%BE%D0%B4-%D0%BA%D1%80%D0%B8%D0%B2%D0%BE%D0%B9-%D0%BE%D1%88%D0%B8%D0%B1%D0%BE%D0%BA/>

# ROC curve - вычисление

Для построения кривой ошибок необходимо посчитать FPR и TPR для таблицы 3, а также для еще нескольких подобных таблиц для разных пороговых значений вероятности. В идеале, каждое уникальное значение оценки может быть порогом.

| id | оценка | класс |
|----|--------|-------|
| 1  | 0.5    | 0     |
| 2  | 0.1    | 0     |
| 3  | 0.2    | 0     |
| 4  | 0.6    | 1     |
| 5  | 0.2    | 1     |
| 6  | 0.3    | 1     |
| 7  | 0.0    | 0     |

Табл. 1

| id | оценка | класс |
|----|--------|-------|
| 4  | 0.6    | 1     |
| 1  | 0.5    | 0     |
| 6  | 0.3    | 1     |
| 3  | 0.2    | 0     |
| 5  | 0.2    | 1     |
| 2  | 0.1    | 0     |
| 7  | 0.0    | 0     |

Табл. 2

| id | > 0.25 | класс |
|----|--------|-------|
| 4  | 1      | 1     |
| 1  | 1      | 0     |
| 6  | 1      | 1     |
| 3  | 0      | 0     |
| 5  | 0      | 1     |
| 2  | 0      | 0     |
| 7  | 0      | 0     |

Табл. 3

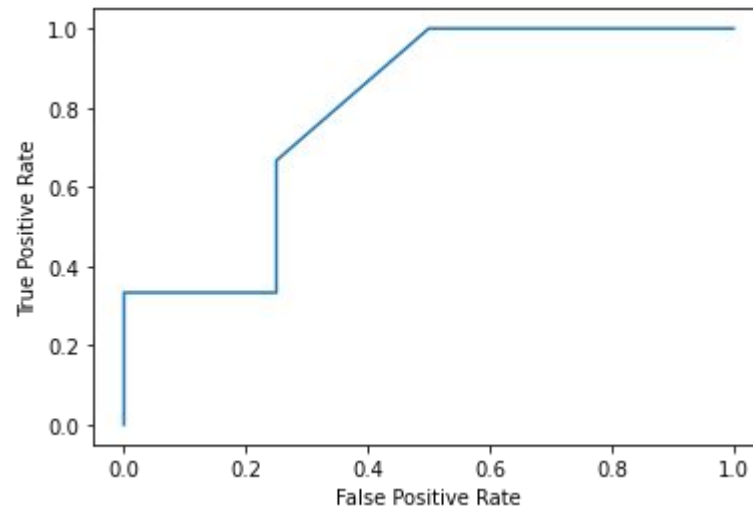
$$\begin{aligned} \text{TPR} &= 2/3 \\ \text{FPR} &= 1/4 \end{aligned}$$

Источник в блоге Александра Дьяконова:

<https://alexanderdyakonov.wordpress.com/2017/07/28/auc-roc-%D0%BF%D0%BB%D0%BE%D1%89%D0%B0%D0%B4%D1%8C-%D0%BF%D0%BE%D0%B4-%D0%BA%D1%80%D0%B8%D0%B2%D0%BE%D0%B9-%D0%BE%D1%88%D0%B8%D0%B1%D0%BE%D0%BA/>

# ROC curve

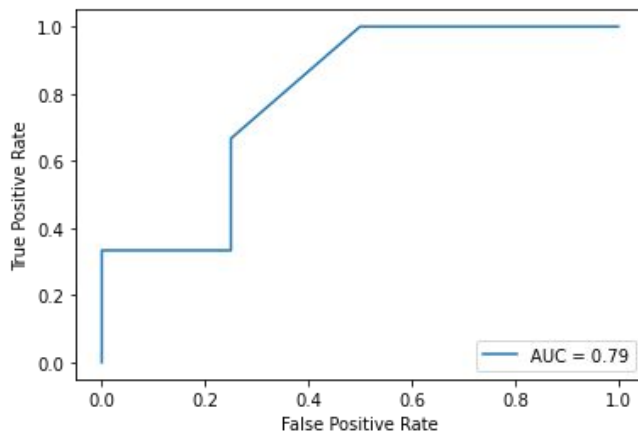
| TPR   | FPR  | Threshold |
|-------|------|-----------|
| 1     | 1    | 0         |
| 1     | 0.75 | 0.1       |
| 1     | 0.5  | 0.2       |
| 1     | 0.5  | 0.3       |
| 0.(6) | 0.25 | 0.4       |
| 0.(6) | 0.25 | 0.5       |
| 0.(3) | 0    | 0.6       |
| 0     | 0    | 1.1       |



Добавим придуманное большое значение, чтобы кривая дошла до нуля

# 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. (вопрос: а что значит  $AUC < 0.5$ ?)

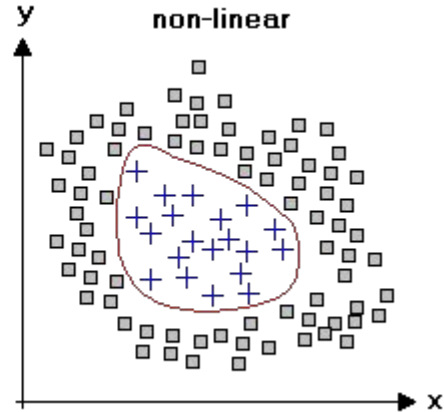
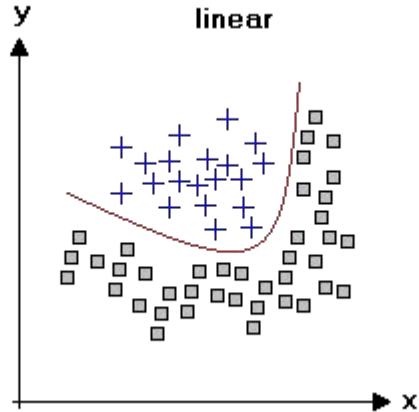
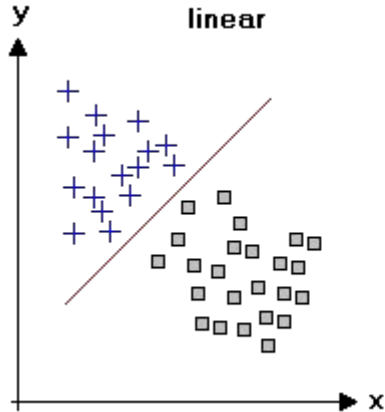




Дополнительно

# Линейность и нелинейность\*

Нелинейность модели  $\neq$  нелинейность функции



# Линейность и нелинейность\*

В моделях линейность определяется относительно параметров, а не предиктора:

$E[Y] = \beta_0 + \beta_1 + \beta_2 \log(X)$  - линейна относительно  $\beta$ ;

$E[Y] = \beta_0 + \beta_1 + \log(\beta_2 X)$  - НЕлинейна относительно  $\beta$ .

# Практика

<https://colab.research.google.com/drive/1aF7fXCwE8AzWEnUcrMTcmIREAf8itsHc?usp=sharing>