

Метрики качества. Предобработка данных.

Татьяна Гайнцева

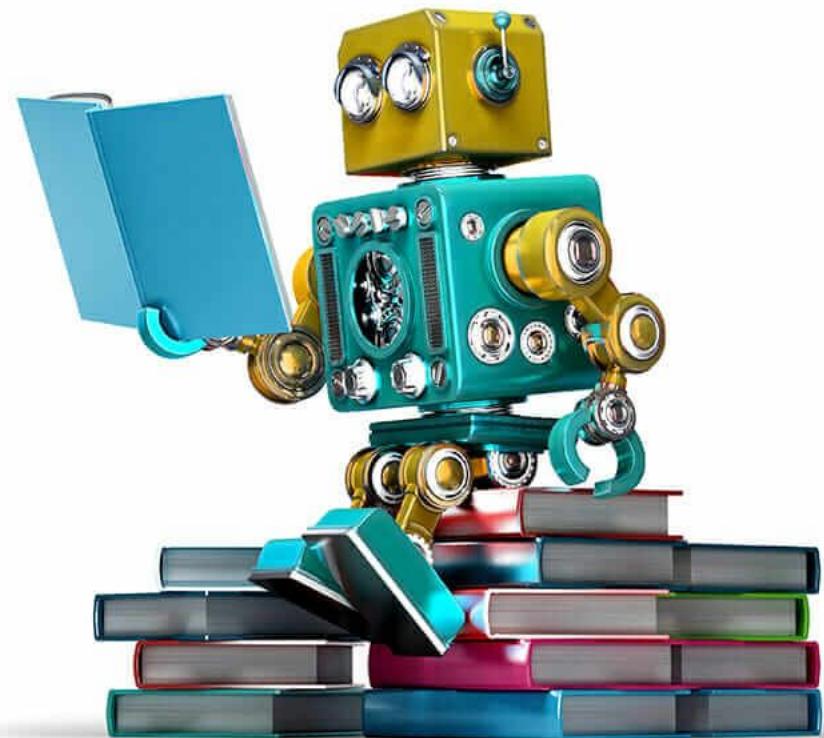
МФТИ, ШАД,

DLSchool

@atmyre

План

1. Оценки качества классификации
2. Критерии обобщающей способности
3. Методы отбора признаков
4. Методы работы с разреженными признаками
5. тестирование модели



Метрики качества классификации

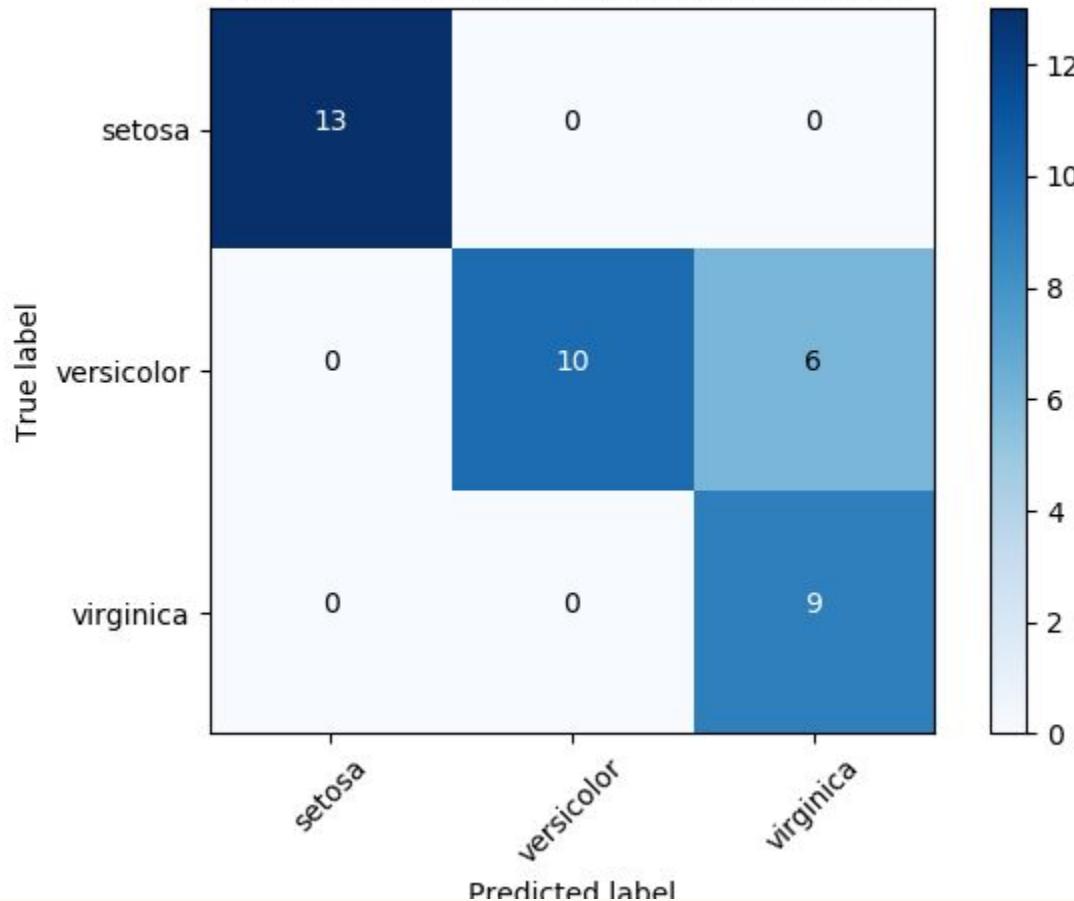


Confusion matrix

		Actual Value (as confirmed by experiment)	
		positives	negatives
Predicted Value (predicted by the test)	positives	TP True Positive	FP False Positive
	negatives	FN False Negative	TN True Negative

`sklearn.metrics.confusion_matrix`

Confusion matrix, without normalization



Accuracy

Интуитивная, понятная и почти неиспользуемая, так как абсолютно бесполезна для работы с несбалансированными классами

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

`sklearn.metrics.accuracy_score`

Пример: spam prediction

Допустим, мы хотим оценить работу спам-фильтра почты. У нас есть 100 не-спам писем, 90 из которых наш классификатор определил верно (True Negative = 90, False Positive = 10), и 10 спам-писем, 5 из которых классификатор также определил верно (True Positive = 5, False Negative = 5).

Тогда accuracy:

$$accuracy = \frac{5 + 90}{5 + 90 + 10 + 5} = 86,4$$

Однако если мы просто будем предсказывать все письма как не-спам, то получим более высокую accuracy:

$$accuracy = \frac{0 + 100}{0 + 100 + 0 + 10} = 90,9$$

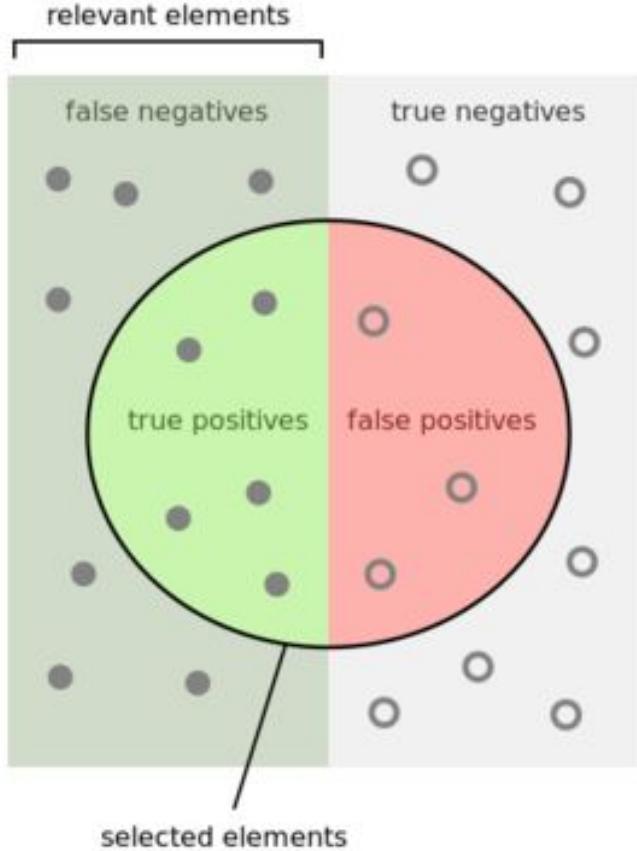
Precision, Recall, F-мера

Для оценки качества работы алгоритма на каждом из классов по отдельности введем метрики precision (точность) и recall (полнота).

$$precision = \frac{TP}{TP + FP}$$

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

Precision можно интерпретировать как долю объектов, названных классификатором положительными и при этом действительно являющимися положительными, а recall показывает, какую долю объектов положительного класса из всех объектов положительного класса нашел алгоритм.



How many selected items are relevant?

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

How many relevant items are selected?

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

`sklearn.metrics.precision_score`

`sklearn.metrics.recall_score`

F-мера

Существует несколько различных способов объединить precision и recall в агрегированный критерий качества. F-мера (в общем случае F_β) — среднее гармоническое precision и recall :

$$F_\beta = (1 + \beta^2) \cdot \frac{precision \cdot recall}{(\beta^2 \cdot precision) + recall}$$

при $\beta = 0$ получаем точность

при $\beta = 1$ — непараметрическую F-меру, (сбалансированная)

при $\beta = \infty$ — полноту.

`sklearn.metrics.f1_score`

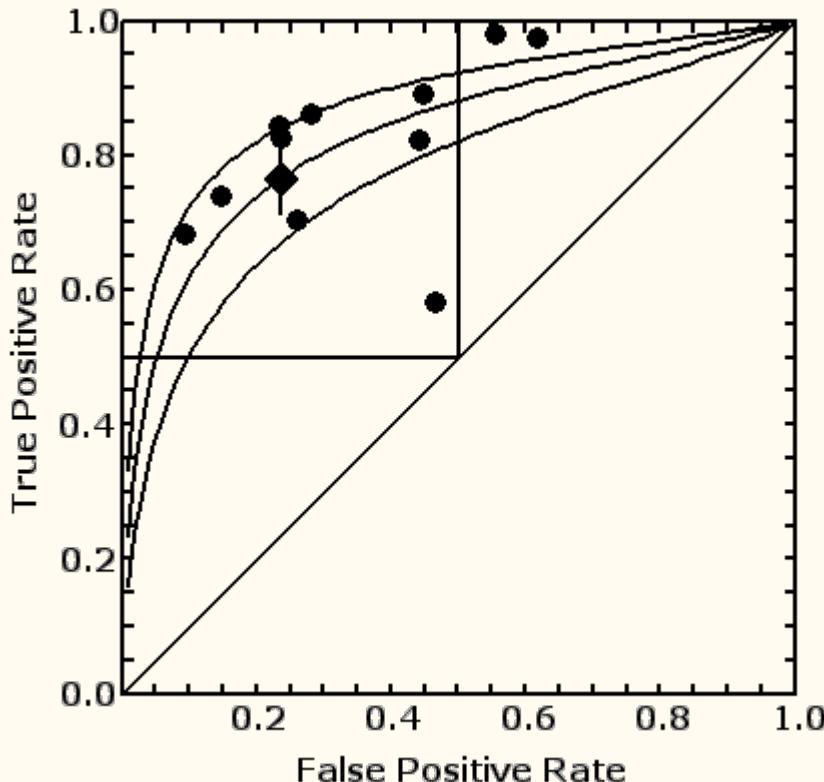
ROC-AUC

При конвертации вещественного ответа алгоритма в бинарную метку, мы должны выбрать какой-либо порог, при котором 0 становится 1. Естественным и близким кажется порог, равный 0.5, но он не всегда оказывается оптимальным, например, при отсутствии баланса классов.

Одним из способов оценить модель в целом, не привязываясь к конкретному порогу, является AUC-ROC — площадь (Area Under Curve) под кривой ошибок (Receiver Operating Characteristic curve).

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

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



ROC-AUC

```
from sklearn import metrics
```

```
fpr, tpr, thresholds = metrics.roc_curve(y, scores, pos_label=2)
```

```
y_true = [0, 2, 2, 1, 1, 0]
```



```
y_true = [0, 1, 1, 0, 0, 0]
```

```
y_pred = [  
    [0.5, 0.4, 0.1],  
    [0.2, 0.3, 0.5],  
    [0.1, 0.1, 0.8],  
    [0.2, 0.4, 0.4],  
    [0.4, 0.2, 0.4],  
    [0.6, 0.1, 0.3],  
]
```



```
y_pred = [  
    0.1,  
    0.5,  
    0.8,  
    0.4,  
    0.4,  
    0.3  
]
```

ROC-AUC

```
from sklearn import metrics
```

```
fpr, tpr, thresholds = metrics.roc_curve(y, scores, pos_label=2)
```

thresholds -- массив порогов для вычисления tpr и fpr

```
y_pred = [  
    [0.5, 0.4, 0.1],  
    [0.2, 0.3, 0.5],  
    [0.1, 0.1, 0.8],  
    [0.2, 0.4, 0.4],  
    [0.4, 0.2, 0.4],  
    [0.6, 0.1, 0.3],  
]
```



```
y_pred = [  
    0.1,  
    0.5,  
    0.8,  
    0.4,  
    0.4,  
    0.3  
]
```

threshold = 0.35
A blue arrow pointing from the sorted y_pred list to the binary classification result.

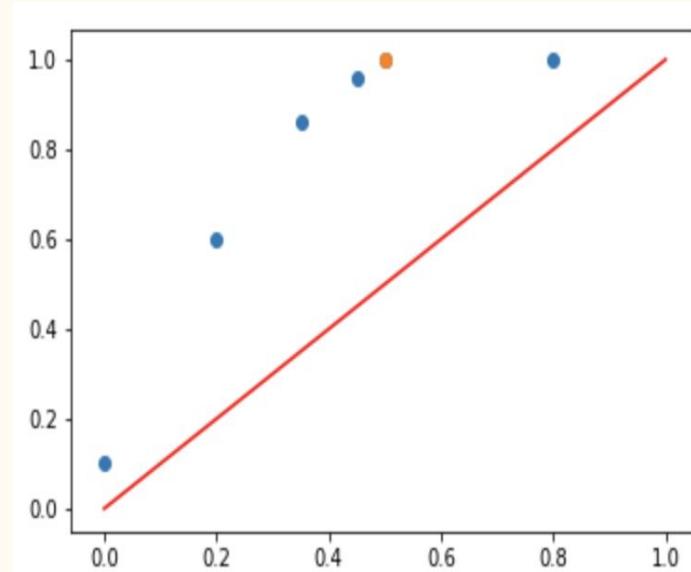
```
y_pred = [  
    0,  
    1,  
    1,  
    1,  
    1,  
    0  
]
```

ROC-AUC

```
y_true = [    y_pred = [  
    0,  
    1,  
    1,  
    0,  
    0,  
    0]  
]
```



$$\begin{aligned} \text{TPR} &= 1 \\ \text{FPR} &= 0.5 \end{aligned}$$



ROC-AUC

```
>>> import numpy as np
>>> from sklearn import metrics
>>> y = np.array([1, 1, 2, 2])
>>> scores = np.array([0.1, 0.4, 0.35, 0.8])
>>> fpr, tpr, thresholds = metrics.roc_curve(y, scores, pos_label=2)
>>> fpr
array([0. , 0. , 0.5, 0.5, 1. ])
>>> tpr
array([0. , 0.5, 0.5, 1. , 1. ])
>>> thresholds
array([1.8 , 0.8 , 0.4 , 0.35, 0.1 ])
```

ROC-AUC

Как подобрать thresholds?

```
y_pred = [    y_true = [  
    0.1,          0,  
    0.5,          1,  
    0.8,          1,  
    0.4,          0,  
    0.4,          0,  
    0.3           0  
]
```



```
y_pred = [    y_true = [  
    0.8,          1,  
    0.5,          1,  
    0.4,          0,  
    0.4,          0,  
    0.3,          0,  
    0.1           0  
]
```

Thresholds = y_pred

ROC-AUC

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

Теперь будем просматривать строки табл. 2 сверху вниз и прорисовывать на сетке линии, переходя их одного узла в другой. Стартуем из точки $(0, 0)$. Если значение метки класса в просматриваемой строке 1, то делаем шаг вверх; если 0, то делаем шаг вправо. Ясно, что в итоге мы попадём в точку $(1, 1)$, т.к. сделаем в сумме m шагов вверх и n шагов вправо.

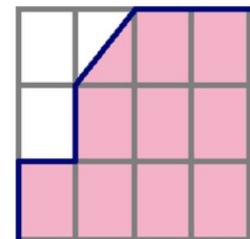
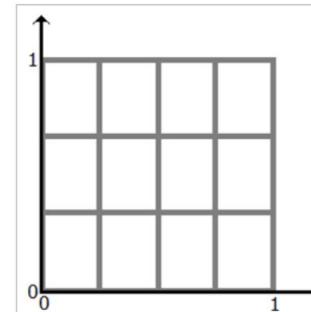
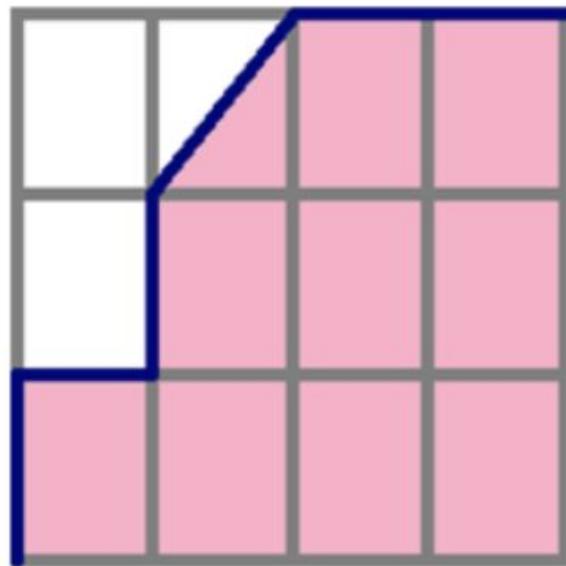


Рис.1. Построение ROC-кривой.

ROC-AUC

threshold = 0.75



оценка

0
0
0
0
0
0
0

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

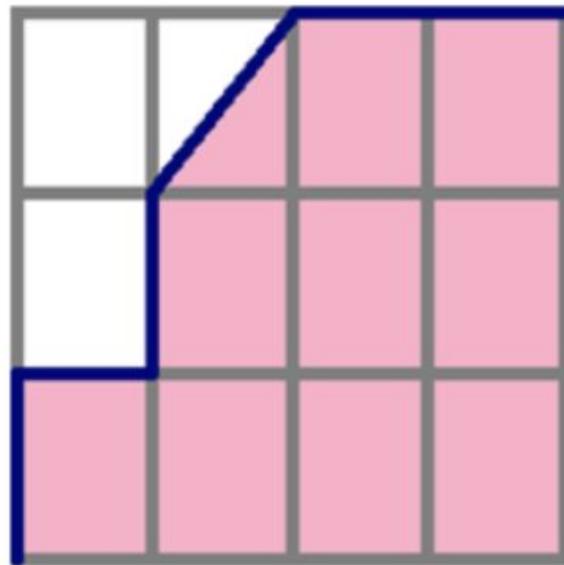
Табл. 2

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

ROC-AUC

threshold = 0.6



оценка

1
0
0
0
0
0
0

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

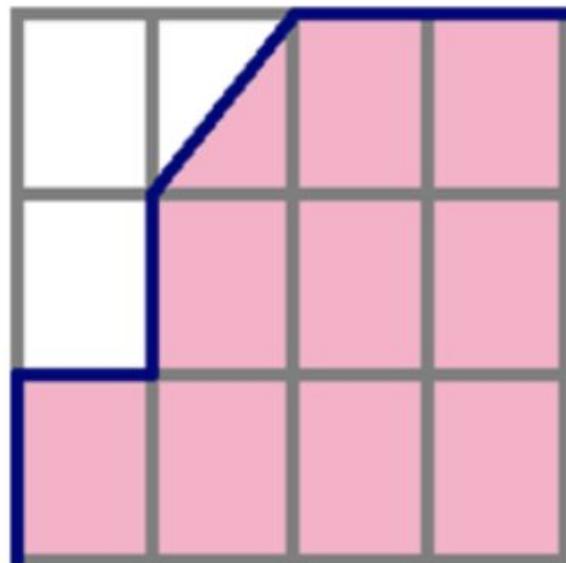
Табл. 2

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

ROC-AUC

threshold = 0.6



оценка

1
0
0
0
0
0
0

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

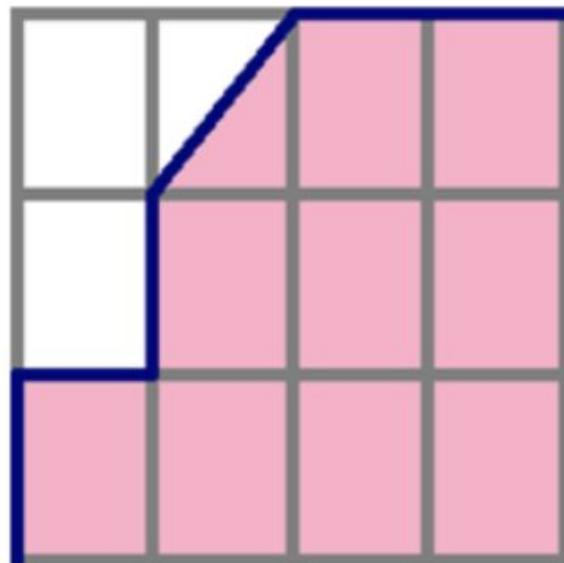
id	> 0.25	класс
4	1	1
1	1	0
6	1	1
3	0	0
5	0	1
2	0	0
7	0	0

Табл. 3

TPR вырос
FPR не изменился

ROC-AUC

threshold = 0.55



оценка

1
0
0
0
0
0
0

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

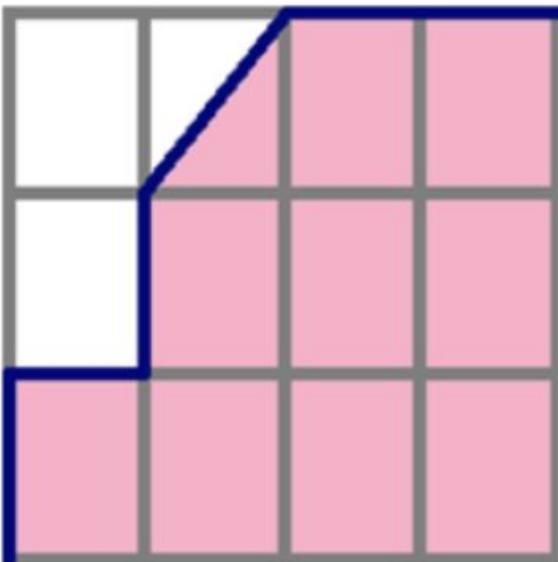
Табл. 2

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

ROC-AUC

threshold = 0.6



оценка

1
0
0
0
0
0
0

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

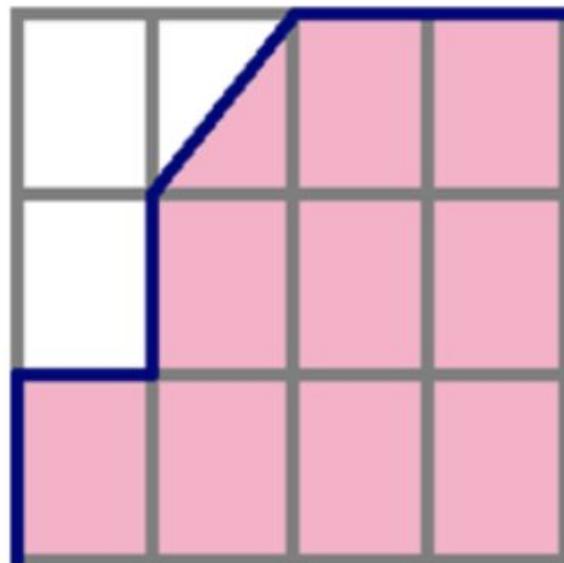
id	> 0.25	класс
4	1	1
1	1	0
6	1	1
3	0	0
5	0	1
2	0	0
7	0	0

Табл. 3

TPR не изменился
FPR не изменился

ROC-AUC

threshold = 0.5



оценка

1
1
0
0
0
0
0

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

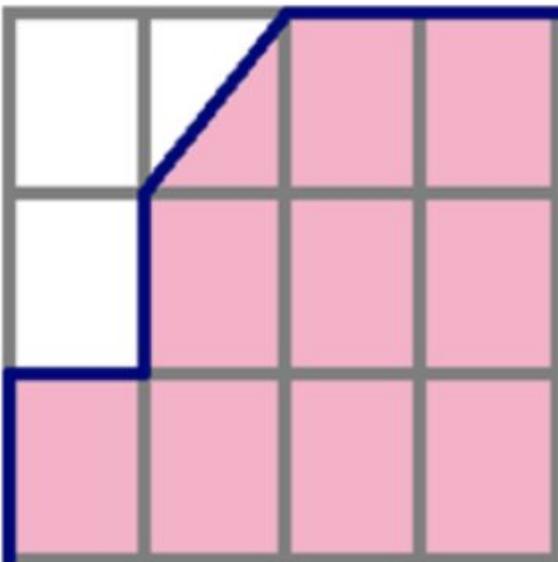
Табл. 2

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

ROC-AUC

threshold = 0.5



оценка

1
1
0
0
0
0
0

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

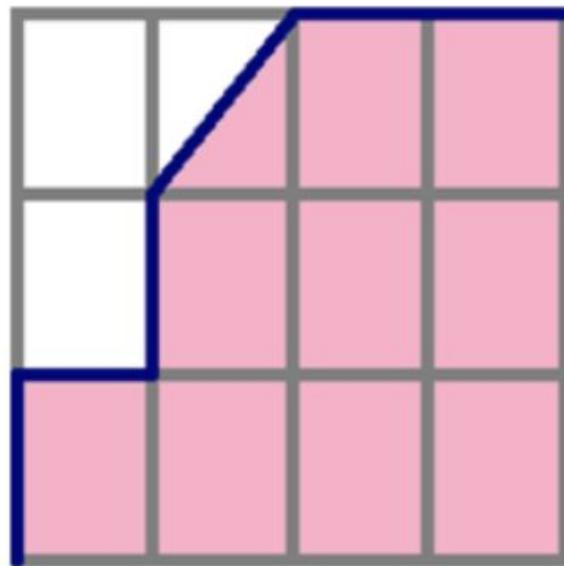
id	> 0.25	класс
4	1	1
1	1	0
6	1	1
3	0	0
5	0	1
2	0	0
7	0	0

Табл. 3

TPR не изменился
FPR вырос

ROC-AUC

threshold = 0.3



оценка

1
1
1
0
0
0
0

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

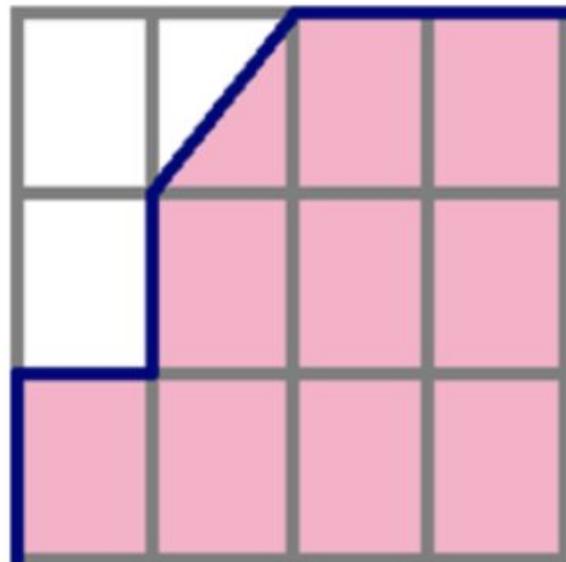
Табл. 2

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

ROC-AUC

threshold = 0.3



оценка

1
1
1
0
0
0
0

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

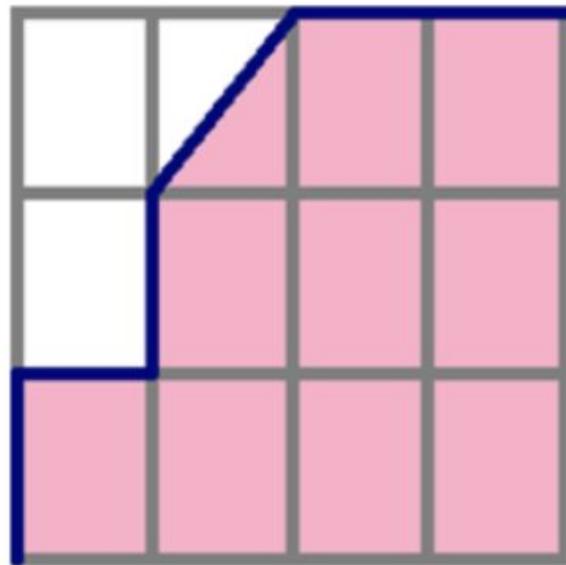
id	> 0.25	класс
4	1	1
1	1	0
6	1	1
3	0	0
5	0	1
2	0	0
7	0	0

Табл. 3

TPR вырос
FPR не изменился

ROC-AUC

threshold = 0.2



оценка

1
1
1
1
1
0
0

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

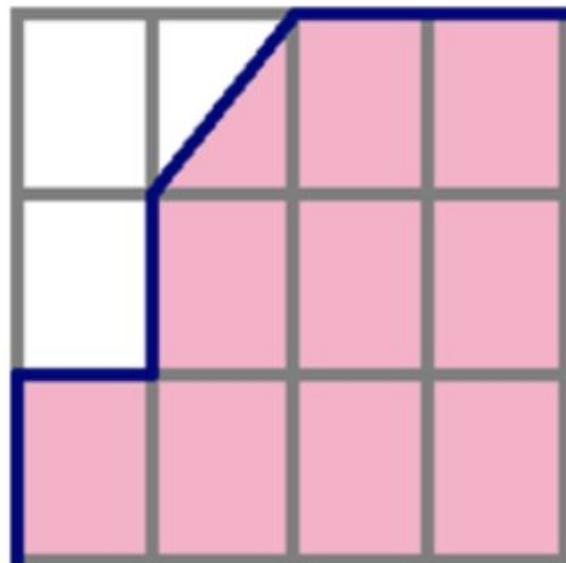
Табл. 2

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

ROC-AUC

threshold = 0.2



оценка

1
1
1
1
1
0
0

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

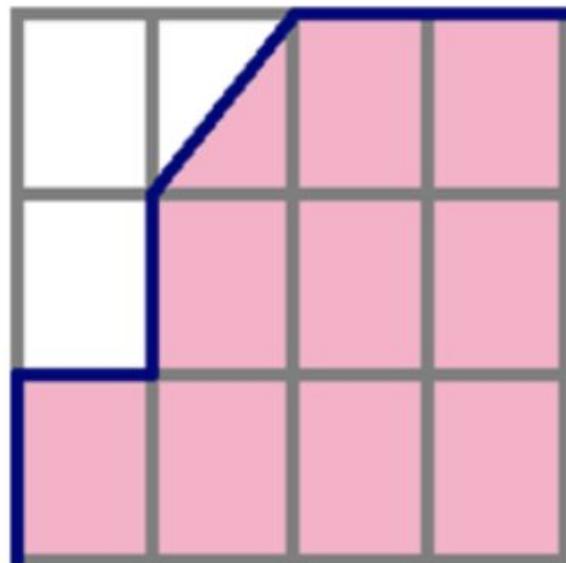
id	> 0.25	класс
4	1	1
1	1	0
6	1	1
3	0	0
5	0	1
2	0	0
7	0	0

Табл. 3

TPR вырос
FPR вырос

ROC-AUC

threshold = 0.1



оценка

1
1
1
1
1
1
0

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

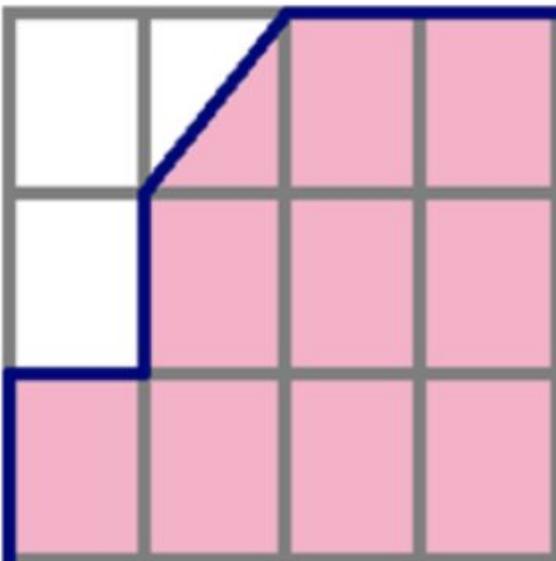
Табл. 2

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

ROC-AUC

threshold = 0.1



оценка

1
1
1
1
1
1
0

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

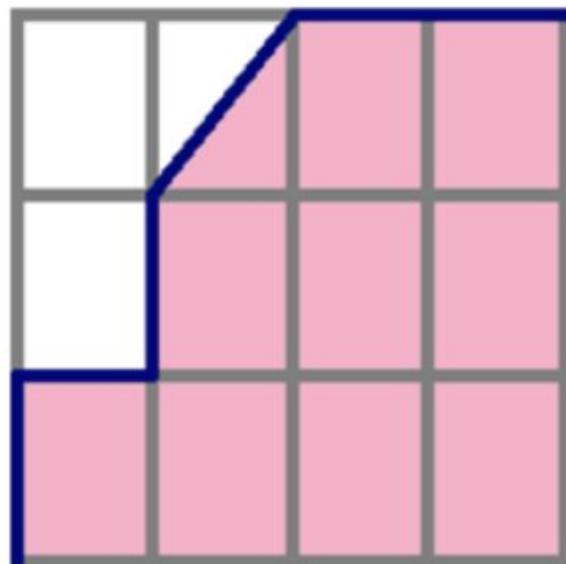
id	> 0.25	класс
4	1	1
1	1	0
6	1	1
3	0	0
5	0	1
2	0	0
7	0	0

Табл. 3

TPR не изменился
FPR вырос

ROC-AUC

threshold = 0.0



оценка

1
1
1
1
1
1
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

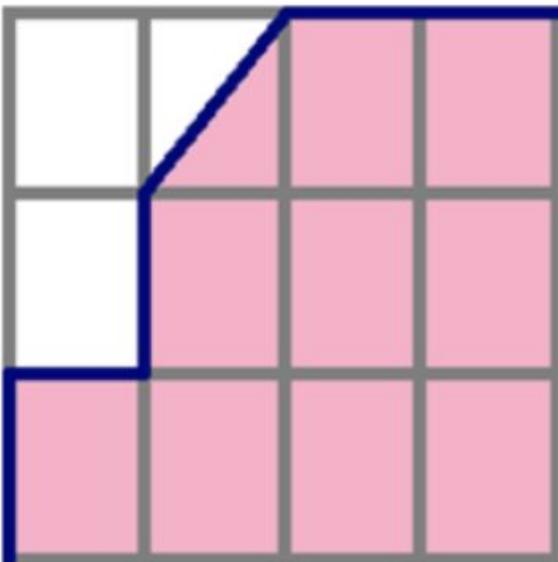
Табл. 2

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

ROC-AUC

threshold = 0.0



оценка

1
1
1
1
1
1
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

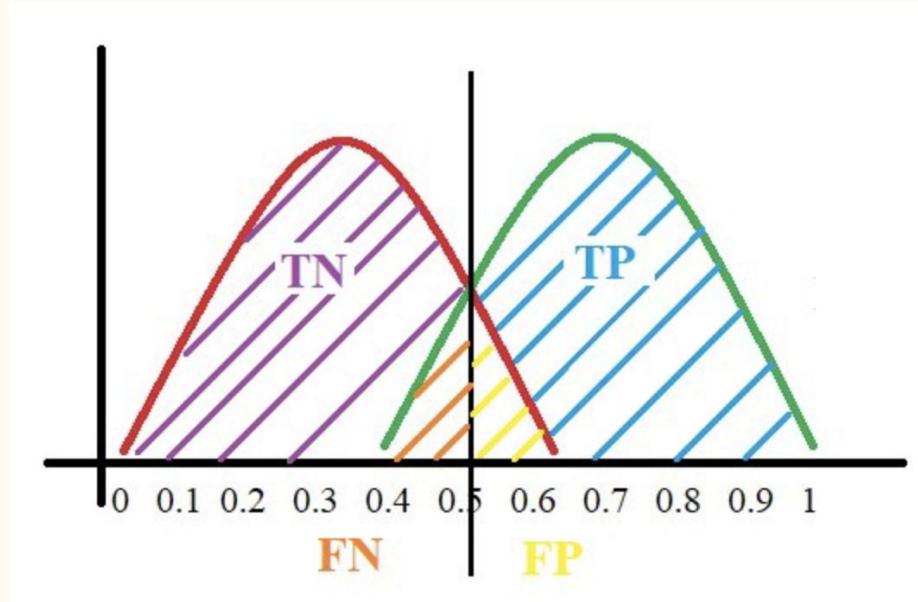
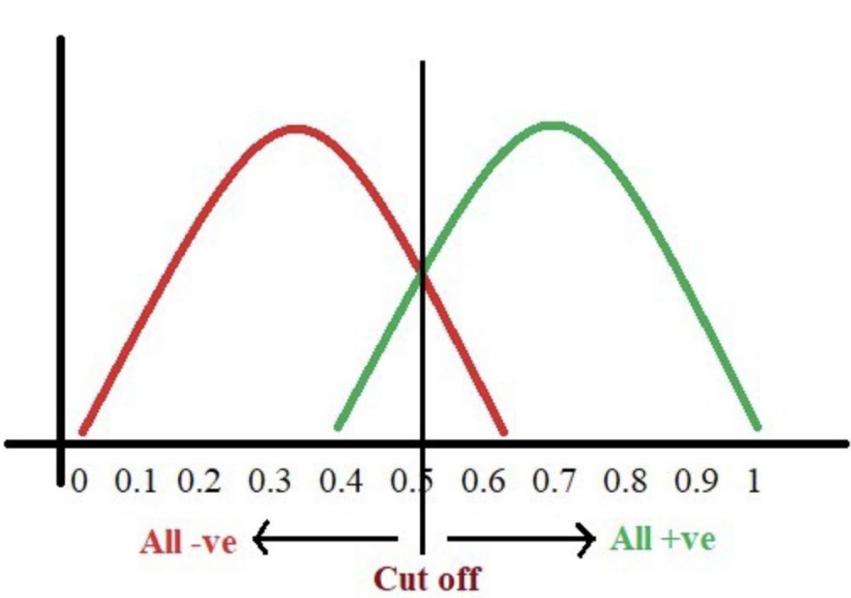
Табл. 3

TPR не изменился
FPR вырос

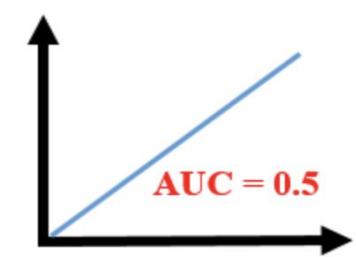
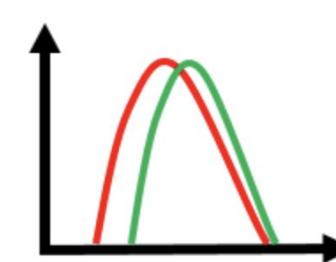
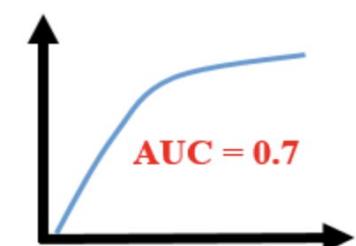
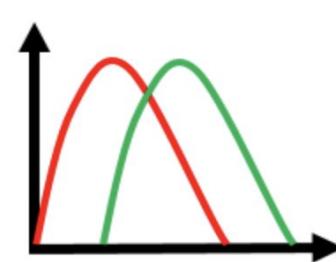
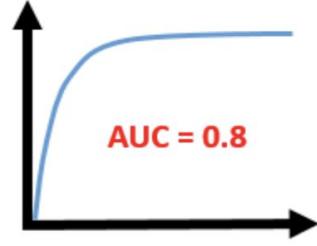
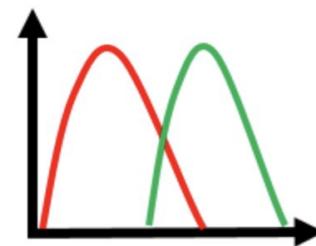
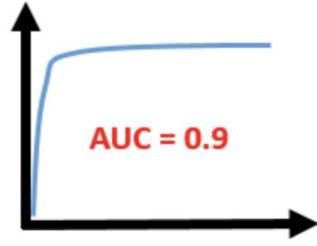
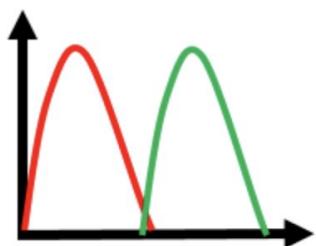
ROC-AUC

Критерий AUC-ROC устойчив к несбалансированным классам и может быть интерпретирован как вероятность того, что случайно выбранный positive объект будет отранжирован классификатором выше (будет иметь более высокую вероятность быть positive), чем случайно выбранный negative объект.

ROC-AUC



ROC-AUC



AUC ROC показывает, насколько хорошо вероятности положительного класса отделены от вероятностей отрицательного

ROC-AUC

Задача: в базе данных Google 1М документов. Нужно вернуть релевантные документы к запросу.

- **Алгоритм 1** возвращает 100 документов, 90 из которых релевантны. Таким образом,

$$TPR = \frac{TP}{TP + FN} = \frac{90}{90 + 10} = 0.9$$

$$FPR = \frac{FP}{FP + TN} = \frac{10}{10 + 999890} = 0.00001$$

ROC-AUC

Задача: в базе данных Google 1М документов. Нужно вернуть релевантные документы к запросу.

- **Алгоритм 2** возвращает 2000 документов, 90 из которых релевантны. Таким образом,

$$TPR = \frac{TP}{TP + FN} = \frac{90}{90 + 10} = 0.9$$

$$FPR = \frac{FP}{FP + TN} = \frac{1910}{1910 + 997990} = 0.00191$$

Разница в False Positive Rate между этими двумя алгоритмами *крайне* мала — всего **0.0019**.

Это является следствием того, что AUC-ROC измеряет долю False Positive относительно True Negative и в задачах, где нам не так важен больший класс, может давать не совсем адекватную картину при сравнении алгоритмов.

Задача: в базе данных Google 1M документов. Нужно вернуть релевантные документы к запросу.

- Алгоритм 1

$$precision = \frac{TP}{TP + FP} = 90/(90 + 10) = 0.9$$

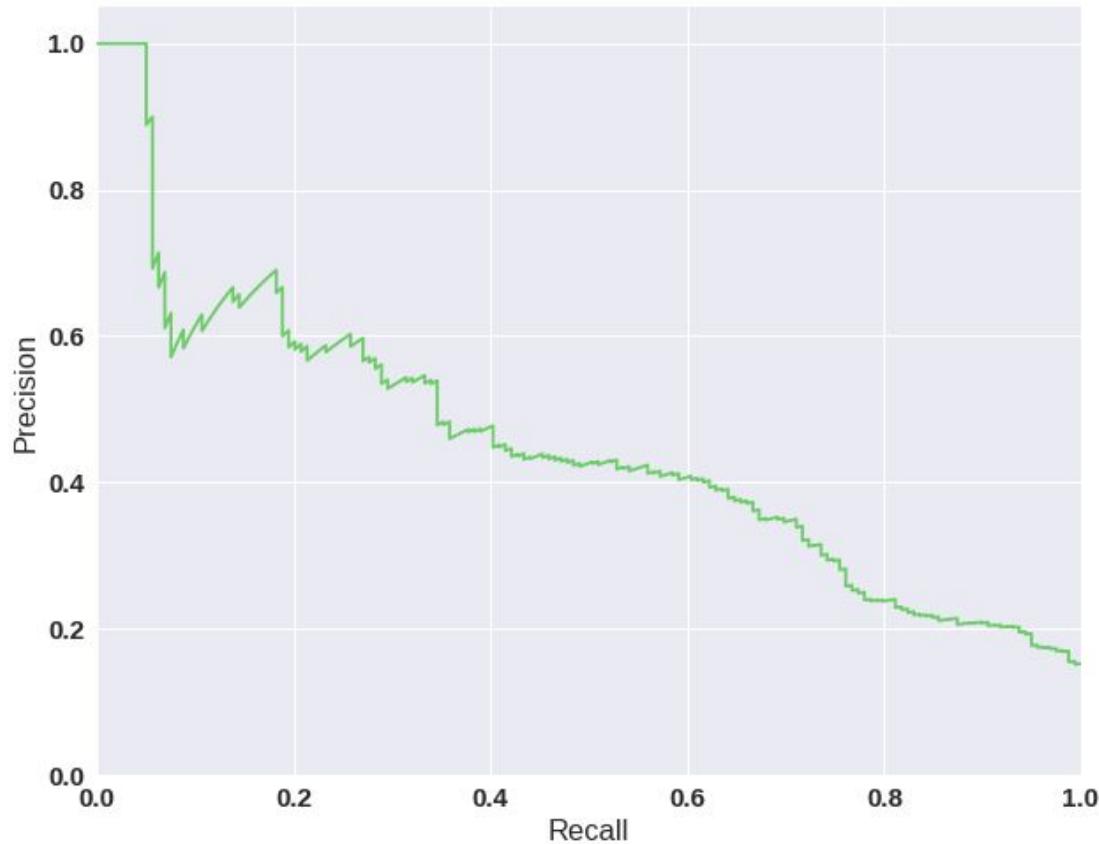
$$recall = \frac{TP}{TP + FN} = 90/(90 + 10) = 0.9$$

- Алгоритм 2

$$precision = \frac{TP}{TP + FP} = \frac{90}{90 + 1910} = 0.045$$

$$recall = \frac{TP}{TP + FN} = \frac{90}{90 + 10} = 0.9$$

Precision-Recall curve



ROC vs PR

Curve	x-axis		y-axis	
	Concept	Calculation	Concept	Calculation
Precision-recall	Recall	$TP / (TP + FN)$	Precision	$TP / (TP + FP)$
ROC	1-specificity	$FP / (FP + TN)$	Sensitivity	$TP / (TP + FN)$

True Negative не используется для построения PR-curve

LogLoss

Рассмотрим задачу **бинарной** классификации

Посмотрим на выборку со следующей точки зрения: каждый элемент обучающей выборки имеет ответ 1 с вероятностью p и ответ 0 с вероятностью $1-p$

Для каждого объекта вероятность p своя!

Задача: максимизировать правдоподобие выборки:

$a_i = a(x_i | w)$ – ответ алгоритма, зависящего от параметров w , на i -м объекте

$$p(y | X, w) = \prod_i p(y_i | x_i, w) = \prod_i a_i^{y_i} (1 - a_i)^{1-y_i} \rightarrow \max$$

$$\sum_i (-y_i \log a_i - (1 - y_i) \log(1 - a_i)) \rightarrow \min$$

LogLoss

$$-\begin{cases} \log a_i, & y_i = 1, \\ \log(1 - a_i), & y_i = 0. \end{cases}$$

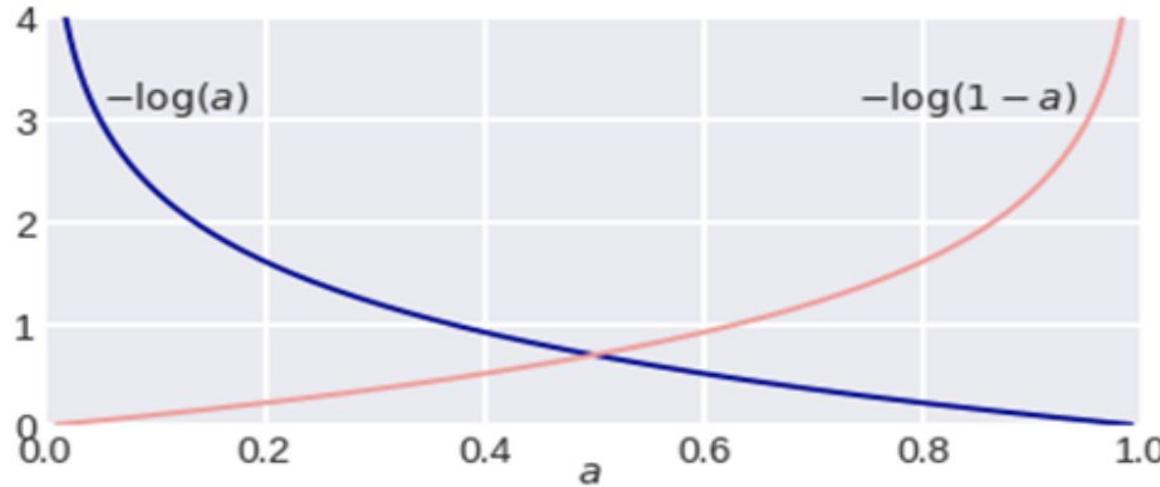


Рис. 1. logloss-ошибка на одном объекте.

LogLoss

$a_i = a(x_i | w)$ – ответ алгоритма, зависящего от параметров w , на i -м объекте

$$p(y | X, w) = \prod_i p(y_i | x_i, w) = \prod_i a_i^{y_i} (1 - a_i)^{1-y_i} \rightarrow \max$$

$$\sum_i (-y_i \log a_i - (1 - y_i) \log(1 - a_i)) \rightarrow \min$$

матожидание нашей ошибки

Оптимальный a -- такой,
который для каждого объекта
выдает вероятность
принадлежности объекта к
классу 1

$$-p \log(a_i) - (1 - p) \log(1 - a_i)$$

$$\frac{p}{a_i} - \frac{1-p}{1-a_i} = 0$$

$$a_i = p$$

LogLoss

Из этого следует, что нам нужно стремить к минимуму такую функцию потерь:

$$\text{LogLoss} = -p \log(p) - (1-p) \log(1-p).$$

Cross-Entropy

Для многоклассовой классификации LogLoss превращается в
Cross-Entropy

Выводы

- В случае многоклассовой классификации нужно внимательно следить за метриками каждого из классов и следовать логике решения задачи, а не оптимизации метрики
- В случае неравных классов нужно подбирать баланс классов для обучения и метрику, которая будет корректно отражать качество классификации
- Выбор метрики нужно делать с фокусом на предметную область, предварительно обрабатывая данные и, возможно, сегментируя (как в случае с делением на богатых и бедных клиентов)

Несбалансированность выборки



Как решать?

- Over-sampling
- Under-sampling
- Ensembles of sampling
- Custom sampling
- Special metrics

The screenshot shows a blue header bar with the text "imbalanced-learn" and "stable". Below it is a search bar labeled "Search docs". The main content area has a dark grey sidebar on the left containing "GETTING STARTED", "Install and contribution", and "DOCUMENTATION". A "User Guide" section is expanded, showing a list of 9 items: 1. Introduction, 2. Over-sampling, 3. Under-sampling, 4. Combination of over- and under-sampling, 5. Ensemble of samplers, 6. Miscellaneous samplers, 7. Metrics, 8. Dataset loading utilities, and 9. Utilities for Developers.

Docs » User guide: contents

User Guide

- 1. Introduction
 - 1.1. API's of imbalanced-learn samplers
 - 1.2. Problem statement regarding imbalanced data sets
- 2. Over-sampling
 - 2.1. A practical guide
 - 2.1.1. Naive random over-sampling
 - 2.1.2. From random over-sampling to SMOTE and ADASYN
 - 2.1.3. Ill-posed examples
 - 2.1.4. SMOTE variants
 - 2.2. Mathematical formulation
 - 2.2.1. Sample generation
 - 2.2.2. Multi-class management
- 3. Under-sampling
 - 3.1. Prototype generation
 - 3.2. Prototype selection
 - 3.2.1. Controlled under-sampling techniques

<https://imbalanced-learn.readthedocs.io/en/stable/miscellaneous.html>

Preprocessing



Standardization

Mean removal and variance scaling

```
>>> from sklearn import preprocessing  
>>> import numpy as np  
>>> X_train = np.array([[ 1., -1.,  2.],  
...                      [ 2.,  0.,  0.],  
...                      [ 0.,  1., -1.]])  
>>> X_scaled = preprocessing.scale(X_train)  
  
>>> X_scaled  
array([[ 0. ..., -1.22...,  1.33...],  
       [ 1.22...,  0. ..., -0.26...],  
       [-1.22...,  1.22..., -1.06...]])
```

```
>>> X_scaled.mean(axis=0)  
array([0., 0., 0.])  
  
>>> X_scaled.std(axis=0)  
array([1., 1., 1.])
```

- RBF Kernel for SVM
- L1, L2 regularizations for LM

Scaling features to a range

```
>>> X_train = np.array([[ 1., -1.,  2.],
...                      [ 2.,  0.,  0.],
...                      [ 0.,  1., -1.]])
...
>>> min_max_scaler = preprocessing.MinMaxScaler()
>>> X_train_minmax = min_max_scaler.fit_transform(X_train)
>>> X_train_minmax
array([[0.5        , 0.         , 1.         ],
       [1.         , 0.5        , 0.33333333],
       [0.         , 1.         , 0.         ]])
```

- robustness to very small standard deviations of features
- preserving zero entries in sparse data

Mapping to a uniform distribution

```
>>> from sklearn.datasets import load_iris
>>> from sklearn.model_selection import train_test_split
>>> iris = load_iris()
>>> X, y = iris.data, iris.target
>>> X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)
>>> quantile_transformer = preprocessing.QuantileTransformer(random_state=0)
>>> X_train_trans = quantile_transformer.fit_transform(X_train)
>>> X_test_trans = quantile_transformer.transform(X_test)
>>> np.percentile(X_train[:, 0], [0, 25, 50, 75, 100])
array([ 4.3,  5.1,  5.8,  6.5,  7.9])
```

- it smooths out unusual distributions and is less influenced by outliers than scaling methods.
- It does, however, distort correlations and distances within and across features

Normalization

```
>>> X = [[ 1., -1.,  2.],
...         [ 2.,  0.,  0.],
...         [ 0.,  1., -1.]]
>>> X_normalized = preprocessing.normalize(X, norm='l2')

>>> X_normalized
array([[ 0.40..., -0.40...,  0.81...],
       [ 1.    ...,  0.    ...,  0.    ...],
       [ 0.    ...,  0.70..., -0.70...]])
```

- metric algorithms
- quadratic kernels

Encoding categorical features

```
>>> enc = preprocessing.OrdinalEncoder()
>>> X = [['male', 'from US', 'uses Safari'], ['female', 'from Europe', 'uses Firefox']]
>>> enc.fit(X)
OrdinalEncoder(categories='auto', dtype=<... 'numpy.float64'>)
>>> enc.transform([['female', 'from US', 'uses Safari']])
array([[0., 1., 1.]])
```

Binarization

```
>>> X = [[ 1., -1.,  2.],
...        [ 2.,  0.,  0.],
...        [ 0.,  1., -1.]]
>>> binarizer = preprocessing.Binarizer().fit(X) # fit does nothing
>>> binarizer
Binarizer(copy=True, threshold=0.0)
>>> binarizer.transform(X)
array([[1., 0., 1.],
       [1., 0., 0.],
       [0., 1., 0.]])
```

Discretization

```
>>> X = np.array([[ -3.,  5., 15 ],
...                 [  0.,  6., 14 ],
...                 [  6.,  3., 11 ]])
>>> est = preprocessing.KBinsDiscretizer(n_bins=[3, 2, 2], encode='ordinal').fit(X)
```

```
>>> est.transform(X)
array([[ 0.,  1.,  1.],
       [ 1.,  1.,  1.],
       [ 2.,  0.,  0.]])
```

discretizers features into k equal width bins

Generating polynomial features

```
>>> import numpy as np
>>> from sklearn.preprocessing import PolynomialFeatures
>>> X = np.arange(6).reshape(3, 2)
>>> X
array([[0, 1],
       [2, 3],
       [4, 5]])
>>> poly = PolynomialFeatures(2)
>>> poly.fit_transform(X)
array([[ 1.,  0.,  1.,  0.,  0.,  1.],
       [ 1.,  2.,  3.,  4.,  6.,  9.],
       [ 1.,  4.,  5., 16., 20., 25.]])
```

- Generating new features
- Train polynomial algorithms

Выбор метода обучения



Дано: X — пространство объектов; Y — множество ответов;
 $X^\ell = (x_i, y_i)_{i=1}^\ell$ — обучающая выборка, $y_i = y^*(x_i)$;
 $A_t = \{a: X \rightarrow Y\}$ — модели алгоритмов, $t \in T$;
 $\mu_t: (X \times Y)^\ell \rightarrow A_t$ — методы обучения, $t \in T$.

Найти: метод μ_t с наилучшей обобщающей способностью.

Частные случаи:

- выбор лучшей модели A_t (model selection);
- выбор метода обучения μ_t для заданной модели A (в частности, оптимизация гиперпараметров);
- отбор признаков (features selection):
 $F = \{f_j: X \rightarrow D_j: j = 1, \dots, n\}$ — множество признаков;
метод обучения μ_J использует только признаки $J \subseteq F$.

Как оценить качество обучения?

$\mathcal{L}(a, x)$ — функция потерь алгоритма a на объекте x ;

$Q(a, X^\ell) = \frac{1}{\ell} \sum_{i=1}^{\ell} \mathcal{L}(a, x_i)$ — функционал качества a на X^ℓ .

Внутренний критерий оценивает качество на обучении X^ℓ :

$$Q_\mu(X^\ell) = Q(\mu(X^\ell), X^\ell).$$

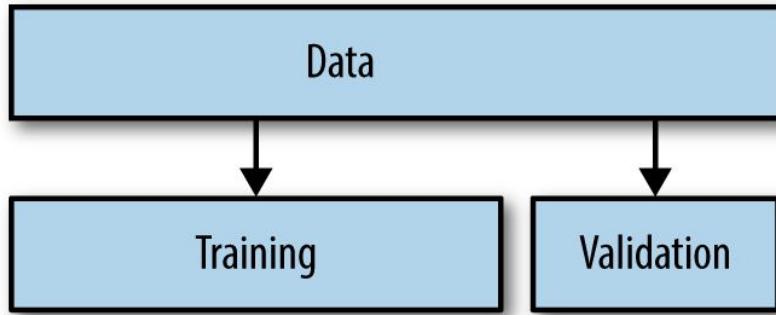
Недостаток: эта оценка смещена, т.к. μ минимизирует её же.

Внешний критерий оценивает качество «вне обучения»,
например, по отложенной (hold-out) контрольной выборке X^k :

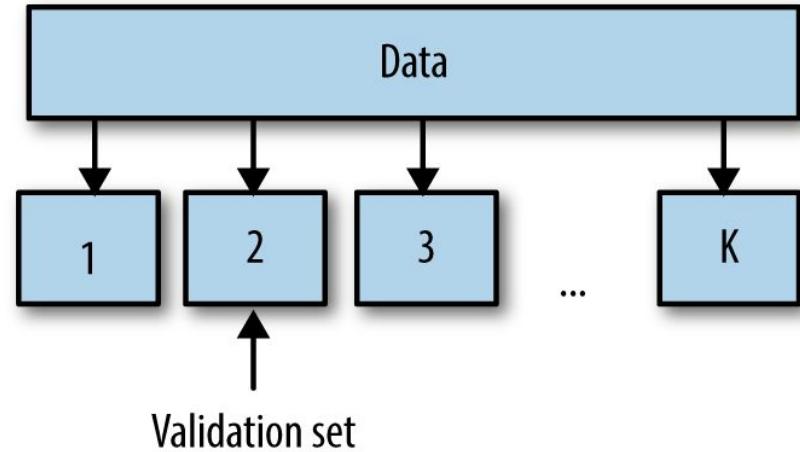
$$Q_\mu(X^\ell, X^k) = Q(\mu(X^\ell), X^k).$$

Недостаток: эта оценка зависит от разбиения $X^L = X^\ell \sqcup X^k$.

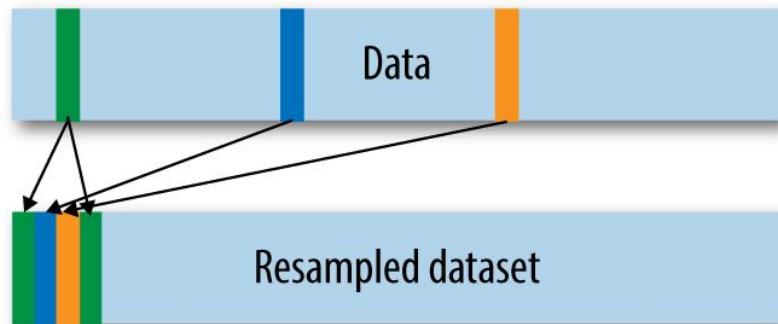
Hold-out validation



K-fold cross validation

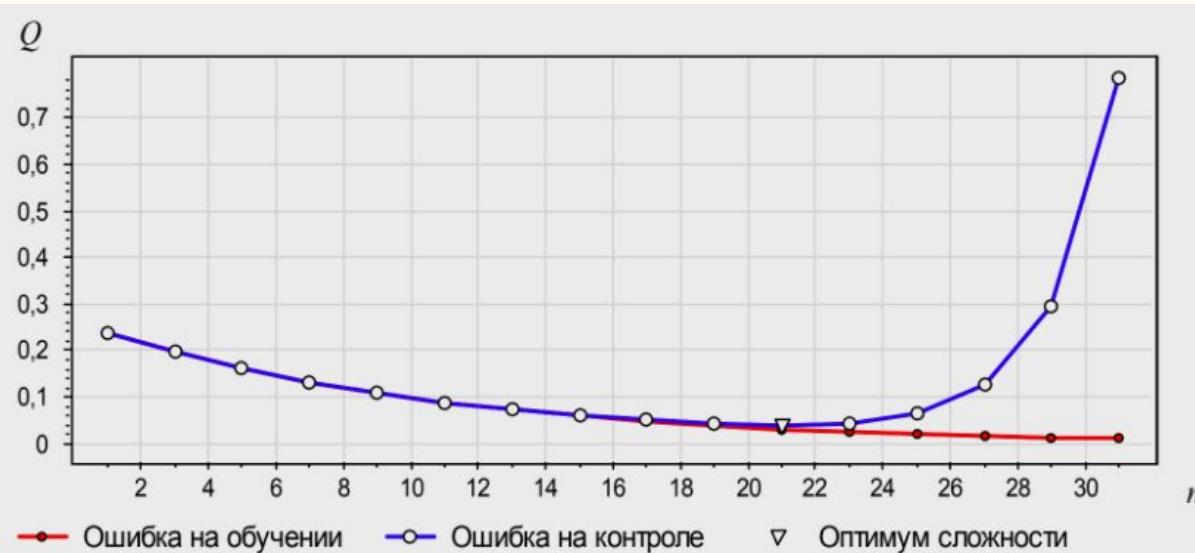


Bootstrap resampling



Внутренний VS Внешний критерий

Ошибка на внутреннем монотонно убывает, в то время как на внешнем имеется характерный минимум



Cross-Validation

Усреднение оценок hold-out по заданному N — множеству разбиений $X^L = X_n^\ell \sqcup X_n^k$, $n = 1, \dots, N$:

$$CV(\mu, X^L) = \frac{1}{|N|} \sum_{n \in N} Q_\mu(X_n^\ell, X_n^k).$$

Способы задания N :

1. Случайное множество разбиений
2. Полная проверка (слишком сложно, используются комбинаторные оценки)

- 3.

$$LOO(\mu, X^L) = \frac{1}{L} \sum_{i=1}^L Q_\mu(X^L \setminus \{x_i\}, \{x_i\}).$$

leave-one-out

- 4.

$$CV_q(\mu, X^L) = \frac{1}{q} \sum_{n=1}^q Q_\mu(X^L \setminus X_n^{\ell_n}, X_n^{\ell_n}).$$

Q-fold CV

q -fold t -times

$$CV_{t \times q}(\mu, X^L) = \frac{1}{t} \sum_{s=1}^t \frac{1}{q} \sum_{n=1}^q Q_\mu(X^L \setminus X_{sn}^{\ell_n}, X_{sn}^{\ell_n}).$$

Преимущества $t \times q$ -fold CV:

- увеличением t можно улучшать точность оценки (компромисс между точностью и временем вычислений);
- каждый объект участвует в контроле ровно t раз;
- оценивание доверительных интервалов (95% при $t = 40$).

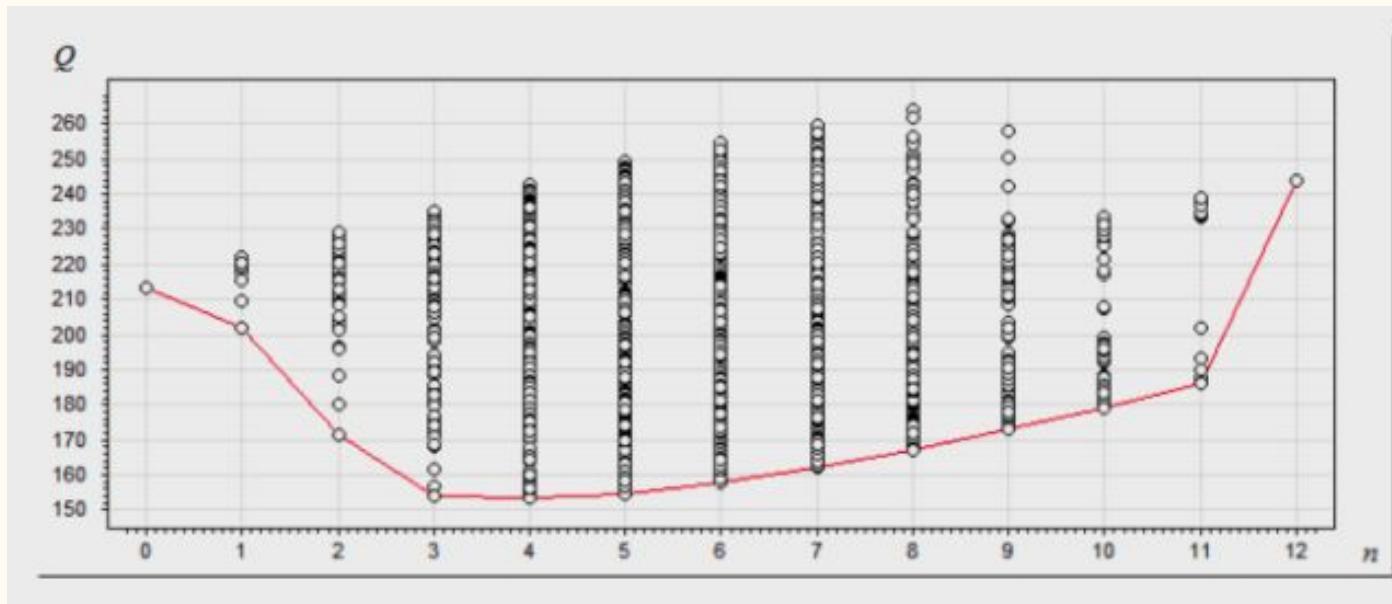
Задача отбора признаков

F -- набор признаков, M_j -- метод обучения, использующий только j выделенных признаков из F . X -- обучающая выборка.

$$Q(j) = Q(M_j, X) \rightarrow \min$$



Полный перебор



Эвристики

1. Жадный алгоритм отбора признаков
2. Чередование добавлений и удалений признаков
3. BFS
4. Эволюционный алгоритм поиска
 - a. Увеличивать вероятность перехода от более успешного родителя
 - b. Применение совокупности критериев
 - c. При стагнации -- мутации
 - d. Параллельно выращивать несколько изолированных популяций

