

# Метрики и валидация

# План

- Мотивация
- Метрики
  - Классификация
    - Confusion Matrix, Accuracy
    - Метрики из Confusion Matrix
    - Roc Auc
    - Log Loss
  - Регрессия
    - MSE, RMSE, R-squared
    - MAE
    - (R)MSPE, MAPE
    - (R)MSLE
  - Оптимизация метрик
- Валидация
  - Стратегии валидации
  - Выбор числа и способа разбиений
  - Анализ причин способов избежать валидационных проблем

# Мотивация

# Метрики

- Почему их столько?
- Зачем нам о них задумываться?

# Метрики

<https://www.kaggle.com/c/rsna-pneumonia-detection-challenge#evaluation>

- This competition is evaluated on the mean average precision at different intersection over union (IoU) thresholds. The IoU of a set of predicted bounding boxes and ground truth bounding boxes is calculated as:
- The metric sweeps over a range of IoU thresholds, at each point calculating an average precision value. The threshold values range from 0.4 to 0.75 with a step size of 0.05: (0.4, 0.45, 0.5, 0.55, 0.6, 0.65, 0.7, 0.75). In other words, at a threshold of 0.5, a predicted object is considered a "hit" if its intersection over union with a ground truth object is greater than 0.5. At each threshold value  $t$ , a precision value is calculated based on the number of true positives (TP), false negatives (FN), and false positives (FP) resulting from comparing the predicted object to all ground truth objects
- A true positive is counted when a single predicted object matches a ground truth object with an IoU above the threshold. A false positive indicates a predicted object had no associated ground truth object. A false negative indicates a ground truth object had no associated predicted object. Important note: if there are no ground truth objects at all for a given image, ANY number of predictions (false positives) will result in the image receiving a score of zero, and being included in the mean average precision. The average precision of a single image is calculated as the mean of the above precision values at each IoU threshold:

$$IoU(A, B) = \frac{A \cap B}{A \cup B}.$$

$$\frac{TP(t)}{TP(t) + FP(t) + FN(t)}.$$

$$\frac{1}{|thresholds|} \sum_t \frac{TP(t)}{TP(t) + FP(t) + FN(t)}.$$

# Мотивация

- Accuracy 0.99 – это хорошо?

# Мотивация

- Accuracy 0.99 – это хорошо?
- Нет, если наши данные распределены 990 – 10 и мы предсказали константный класс

# Константные предсказания

Зачем?

- Ваша модель предсказывает почти 100%?
- Нет возможности использовать cross-validation для проверки
- Дисбаланс, плохо подобраны параметры, неправильная модель

# Константные предсказания

```
class sklearn.dummy. DummyClassifier (strategy='stratified', random_state=None, constant=None)
```

[source]

DummyClassifier is a classifier that makes predictions using simple rules.

This classifier is useful as a simple baseline to compare with other (real) classifiers. Do not use it for real problems.

Read more in the [User Guide](#).

**Parameters:** `strategy : str, default="stratified"`

Strategy to use to generate predictions.

- “stratified”: generates predictions by respecting the training set’s class distribution.
- “most\_frequent”: always predicts the most frequent label in the training set.
- “prior”: always predicts the class that maximizes the class prior (like “most\_frequent”) and `predict_proba` returns the class prior.
- “uniform”: generates predictions uniformly at random.
- “constant”: always predicts a constant label that is provided by the user. This is useful for metrics that evaluate a non-majority class

*New in version 0.17:* Dummy Classifier now supports prior fitting strategy using parameter `prior`.

# Классификация

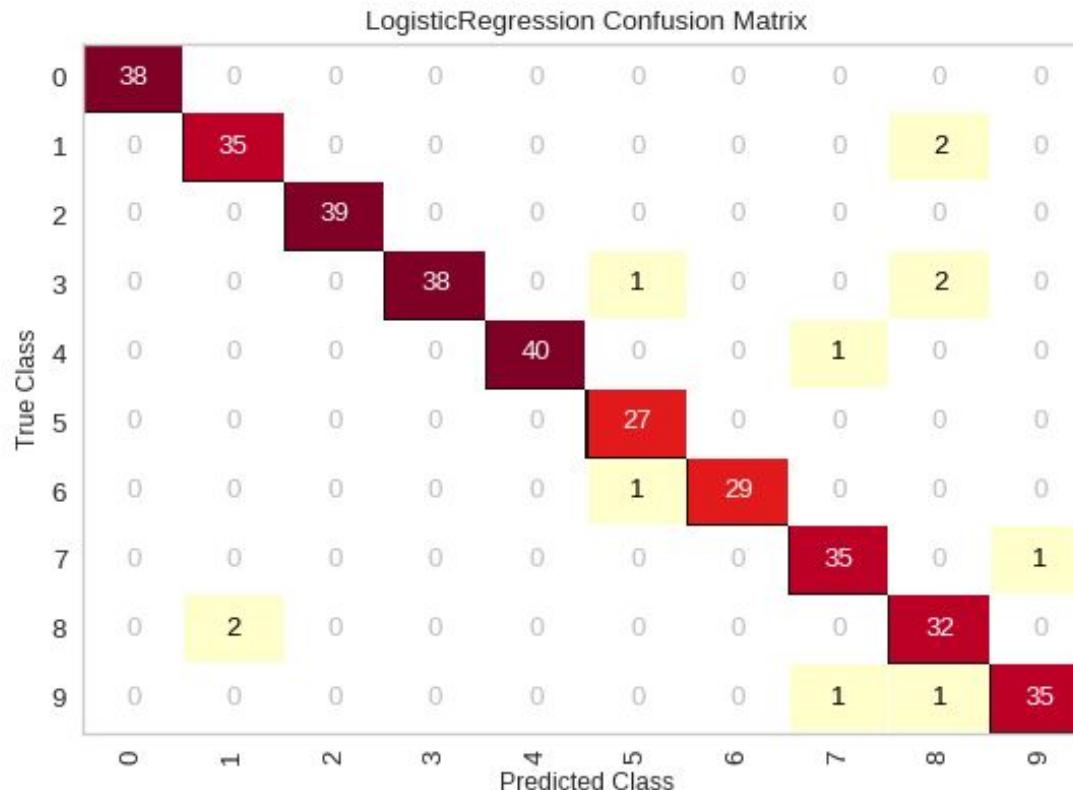
# План

- Обозначения
- Accuracy
- Confusion matrix (TP, FP, TN, FN)
- Метрики из Confusion matrix:
  - Sensitivity, Specificity, Prevalence, Detection Rate
  - Balanced Accuracy, Precision, F1-score
- ROC AUC
- Log-loss (multiclass)

# Обозначения

- $N$  - количество объектов
- $L$  - количество классов
- $y$  - исходные значения
- $\hat{y}$  - предсказания
- $[a = b]$  - indicator function

# Confusion matrix



# Confusion matrix - 2 класса

Пример:

```
In [14]: # Важно: первый аргумент - true values, второй - predicted values  
# получаем матрицу 2x2  
print(metrics.confusion_matrix(y_test, y_pred_class))
```

```
[[118  12]  
 [ 47  15]]
```

N = 192	Предсказан: 0	Предсказана: 1
Истинный: 0	118	12
Истинный: 1	47	15

# Accuracy score

$$\text{Accuracy} = \frac{1}{N} \sum_{i=1}^N [\hat{y}_i = y_i]$$

Доля правильных  
предсказаний

# Accuracy score

$$\text{Accuracy} = \frac{1}{N} \sum_{i=1}^N [\alpha = y_i]$$

Лучшая константа –  
предсказывать  
самый частый класс

Пример:

```
In [8]: # Изучаем распределение классов тестового сета (с помощью Pandas Series)
y_test.value_counts()
```

```
Out[8]: 0    130
1     62
Name: label, dtype: int64
```

```
In [9]: # Считаем процент единиц
# y_test содержит только нули и единицы, мы можем просто посчитать mean = процент единиц
y_test.mean()
```

```
Out[9]: 0.3229166666666667
```

# Accuracy score

Пример:

```
In [10]: # Находим процент нулей  
1 - y_test.mean()
```

```
Out[10]: 0.6770833333333333
```

```
In [11]: # Находим константную точность в одну строчку кода  
# Только для бинарных признаков (1/0)  
max(y_test.mean(), 1 - y_test.mean())
```

```
Out[11]: 0.6770833333333333
```

- Это значит, что данная 'глупая' модель, которая всегда предсказывает 0, будет права в ~68% случаев
- Это хороший способ узнать, какого минимума мы должны достичь с другими моделями

# TP, FP, TN, FN

N = 192	Предсказан: 0	Предсказана: 1
Истинный: 0	TN = 118	FP = 12
Истинный: 1	FN = 47	TP = 15

- True Positives (TP): мы правильно предсказали 1
  - 15
- True Negatives (TN): мы правильно предсказали 0
  - 118
- False Positives (FP): мы неправильно предсказали 1
  - 12
  - Ошибка 1 типа
- False Negatives (FN): мы неправильно предсказали 0
  - 47
  - Ошибка 2 типа

# Метрики из Confusion matrix

- Accuracy:  $\frac{(TP+TN)}{(TP+TN+FP+FN)} = 0.693$
- Sensitivity:  $\frac{TP}{(TP+FN)} = 0.242$ 
  - Когда исходное значение позитивны(1), как часто предсказания верны?
  - Как "sensitive" классификатор к обнаружению положительных классу?
  - Также из:  $\frac{TP}{(TN+FP)} = 0.907$  Positive Rate (TPR) или "Recall"
- Specificity:
  - Как "specific" (или "selective") классификатор в предсказании позитивного класса?
  - False Positive Rate (FPR) = (1 – Specificity)

# Метрики из Confusion matrix

- Prevalence:  $\frac{(TN+FN)}{(TN+FP+FN+FP)} = 0.859$ 
  - Какой процент позитивных(1) значений у нас?
  - Более высокая Prevalence подразумевает, что вы можете получить более высокую Precision даже при угадывании.
- Detection Rate:  $\frac{TN}{(TN+FP+FN+FP)} = 0.615$ 
  - Правильно предсказанные 1 во всем сете
  - Если у нас хорошая точность модели, при среднем Detection Rate, как будет меняться точность если в выборке будет больше единиц.

# Метрики из Confusion matrix

- Balanced Accuracy:  $\frac{Sensitivity+Specificity}{2} = 0.574$ 
  - Процент правильных предсказаний среди 0 и 1
  - Подходит, если ваша выборка несбалансированная
- Precision:  $\frac{TP}{(TP+FP)} = 0.556$ 
  - Когда мы предсказываем 1, как часто это верно
- F1-score:  $\frac{2*Precision*Recall}{(Precision+Recall)} = 0.337$ 
  - Это комбинация Precision и Recall
  - Подходит, если есть дисбаланс в выборке
  - В более общем виде:  $F_\beta = (1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}}$

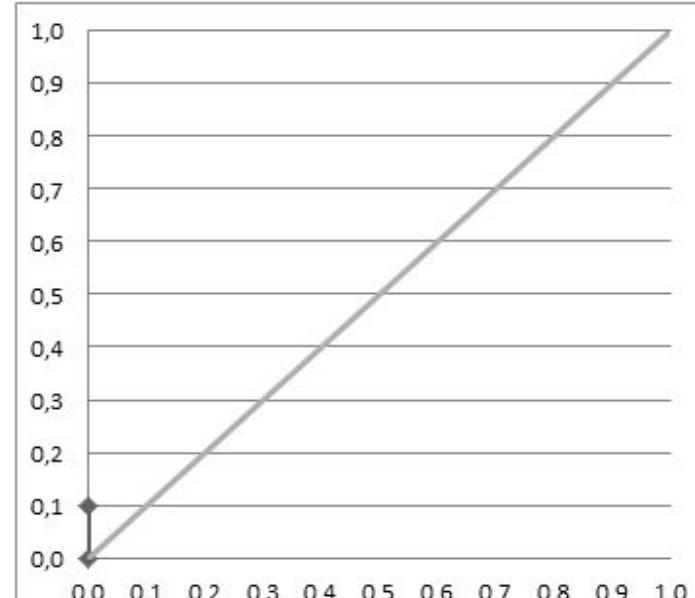
# ROC

Учитывая набор данных и классификатор, который может ранжировать:

- Упорядочить тестовые примеры по шкале от самой высокой до самой низкой
- Начинаем в  $(0, 0)$  и с максимума вероятности
- Для каждого примера  $x$  в упорядоченном множестве
  - Если  $x$  положительный( $=1$ ), двигаемся  $1/\text{pos}$  вверх
  - Если  $x$  негативный( $=0$ ), двигаемся  $1/\text{neg}$  вправо

Где  $\text{pos}$  и  $\text{neg}$  доли положительных и отрицательных примеров.

#	C	Score
1	P	0,9
2	P	0,8
3	N	0,7
4	P	0,6
5	P	0,55
6	P	0,54
7	N	0,53
8	N	0,52
9	P	0,51
10	N	0,505
11	P	0,4
12	N	0,39
13	P	0,38
14	N	0,37
15	N	0,36
16	N	0,35
17	P	0,34
18	N	0,33
19	P	0,3
20	N	0,1

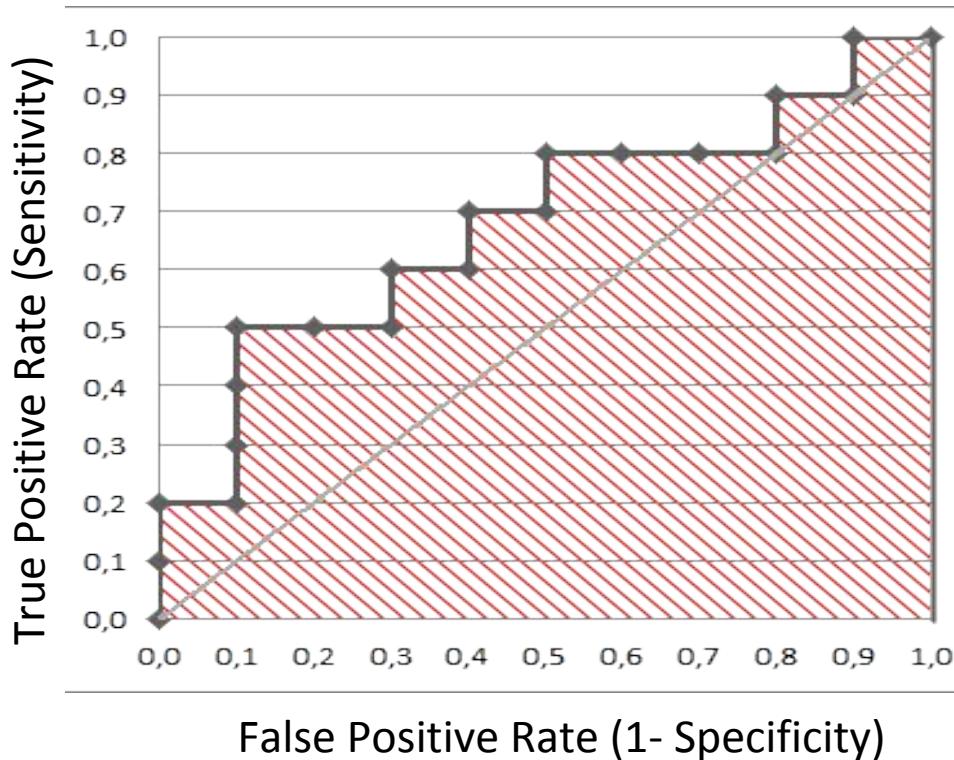


False Positive Rate ( $1 - \text{Specificity}$ )

True Positive Rate (Sensitivity)

# AUC ROC

- Только для бинарных задач
- Зависит от порядка предсказаний, а не абсолютных значений
- Для бейзлайна должно быть 0.5
- Для лучшего решения примерно 1



# ROC Пример

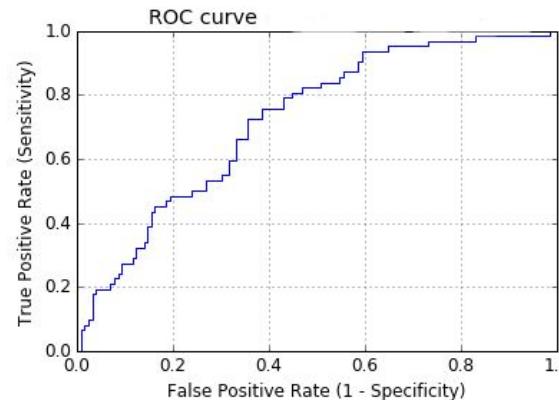
- ROC может помочь выбрать правильный порог (threshold), который будет выравнивать вашу sensitivity и specificity для вашего примера.
- Вы не можете видеть пороги, используемые для генерации кривой на самой кривой ROC.

In [59]:

```
# Важно: первый аргумент это исходные значения, второй это предсказываемые вероятности.

# мы передаем y_test и y_pred_prob
# Мы не используем y_pred_class, потому он даст неправильный результат
# roc_curve возвращает 3 объекта fpr, tpr, thresholds
# fpr: false positive rate
# tpr: true positive rate
fpr, tpr, thresholds = metrics.roc_curve(y_test, y_pred_prob)

plt.plot(fpr, tpr)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.0])
plt.rcParams['font.size'] = 12
plt.title('ROC curve')
plt.xlabel('False Positive Rate (1 - Specificity)')
plt.ylabel('True Positive Rate (Sensitivity)')
plt.grid(True)
```



# ROC Пример

```
In [63]: # Определяем функцию которая принимает порог и выводит sensitivity и specificity
def evaluate_threshold(threshold):
    print('Sensitivity:', tpr[thresholds > threshold][-1])
    print('Specificity:', 1 - fpr[thresholds > threshold][-1])
```

```
In [64]: evaluate_threshold(0.5)
```

```
Sensitivity: 0.241935483871
Specificity: 0.907692307692
```

```
In [65]: evaluate_threshold(0.3)
```

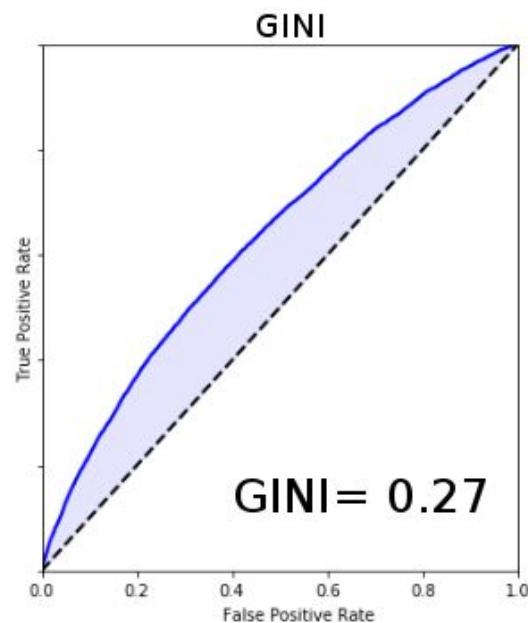
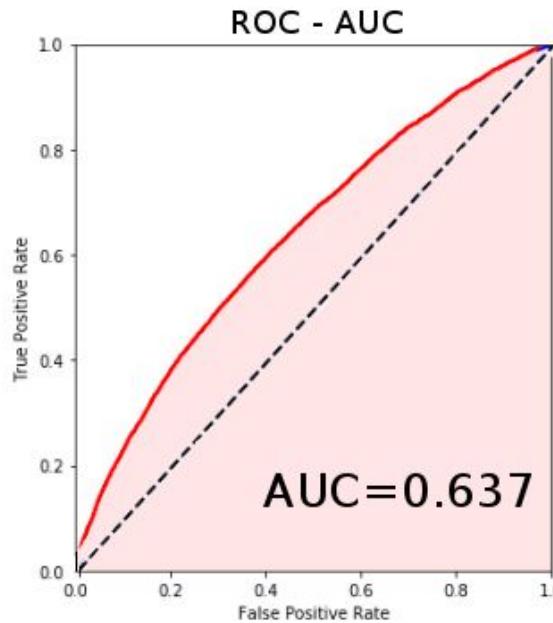
```
Sensitivity: 0.725806451613
Specificity: 0.615384615385
```

# AUC ROC Пример

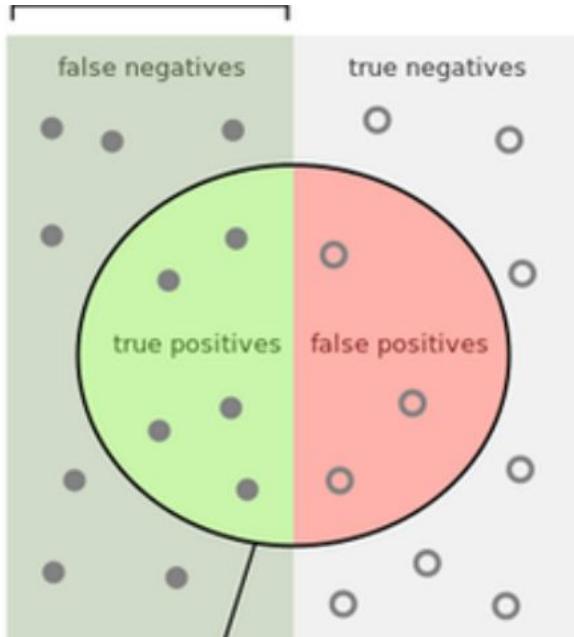
```
In [67]: # Важно: первый аргумент это исходные значения, второй это предсказываемые вероятности.  
print(metrics.roc_auc_score(y_test, y_pred_prob))  
0.724565756824
```

- AUC полезен как единственная оценка для классификатора
- Если вы случайно выбрали одно положительное и одно отрицательное наблюдение, AUC представляет вероятность того, что ваш классификатор назначит более высокую предсказанную вероятность положительному наблюдению
- AUC полезен даже при высоком уровне дисбаланса (в отличие от accuracy)
- Любое константное предсказание приводит к тому же значению AUC (например – 0.5)

# AUC ROC - Gini



# Precision / recall



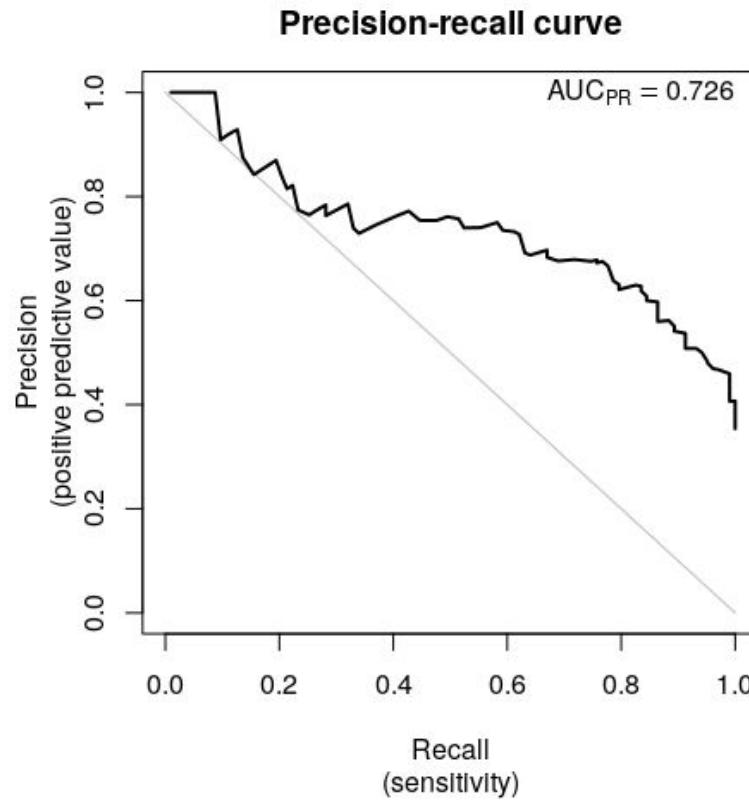
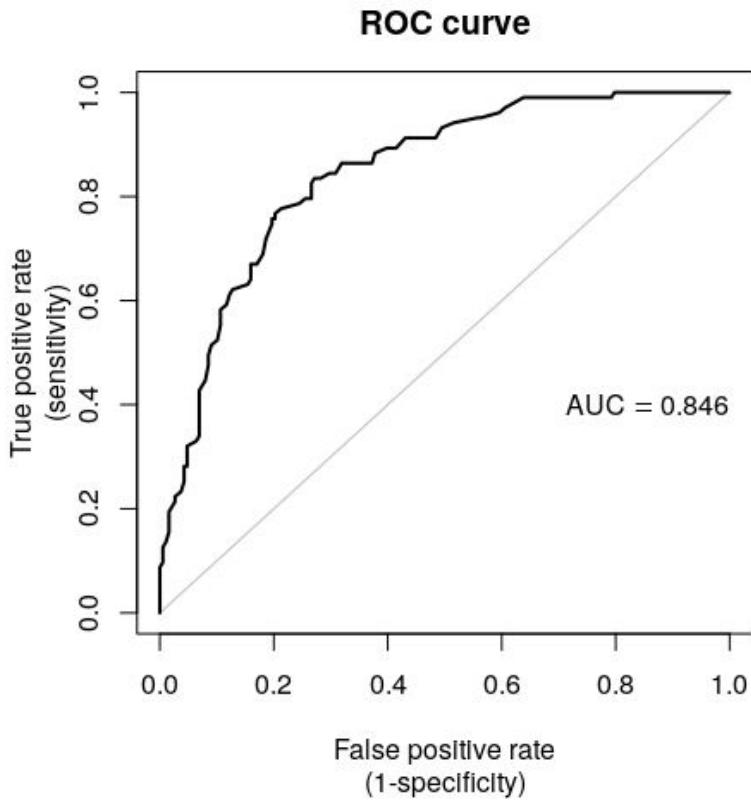
How many selected items are relevant?

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

How many relevant items are selected?

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

# Precision recall curve

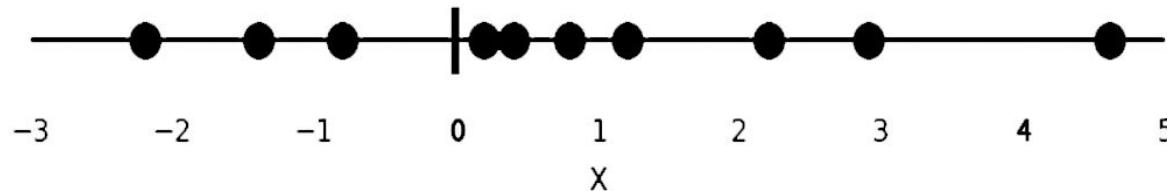


# LogLoss

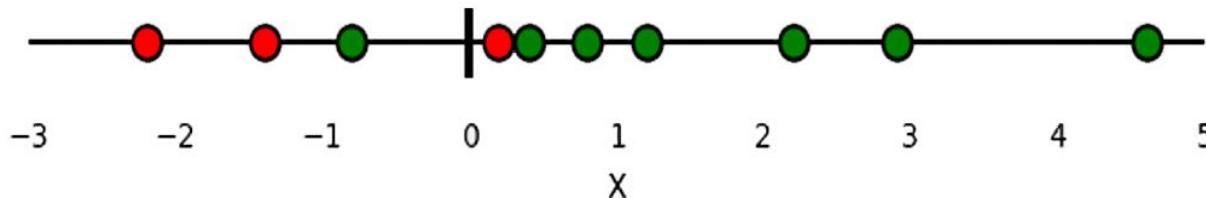
- Binary:  $\text{LogLoss} = -\frac{1}{N} \sum_{i=1}^N [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$

# LogLoss пример

У нас есть 10 значений признака  $x$ :

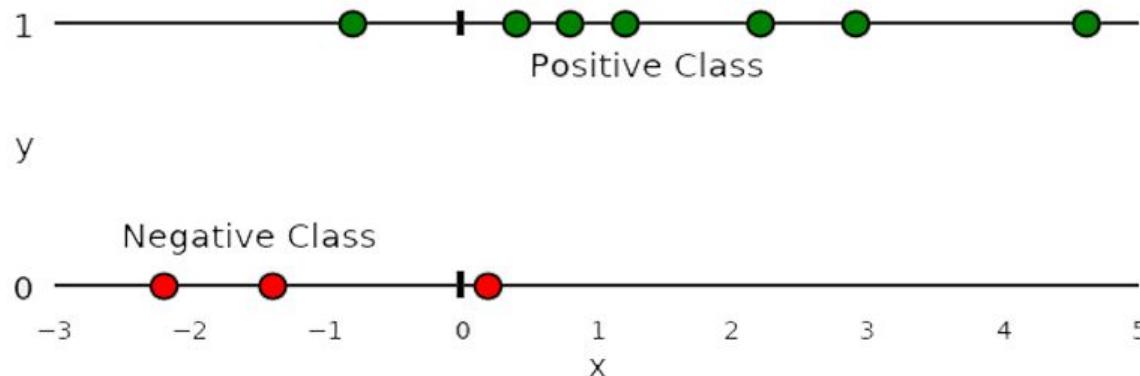


Давайте обозначим некоторые цветами (красный и зеленый):



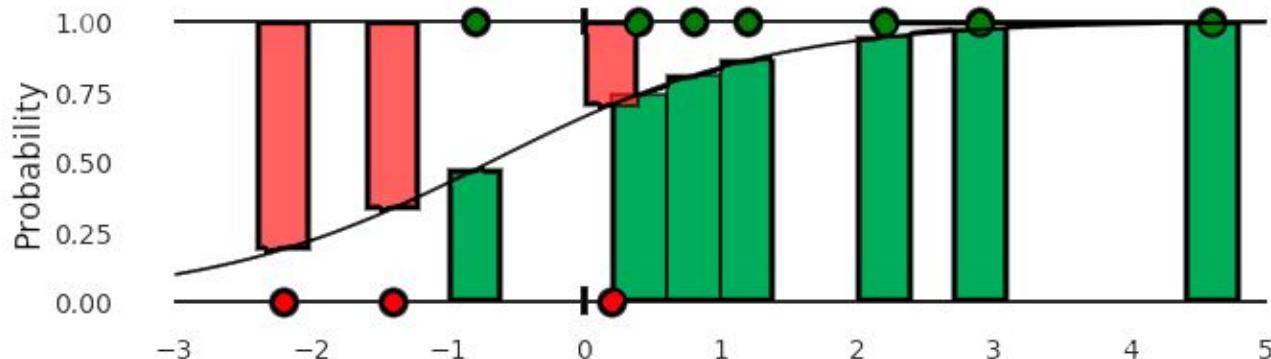
# LogLoss пример

Давайте разделим на зеленые и красные:



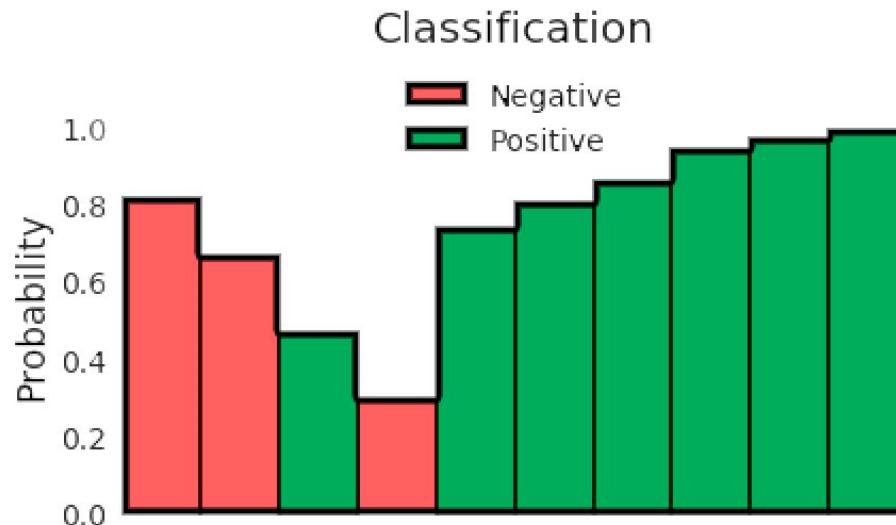
# LogLoss пример

Обучив логистическую регрессию получаем



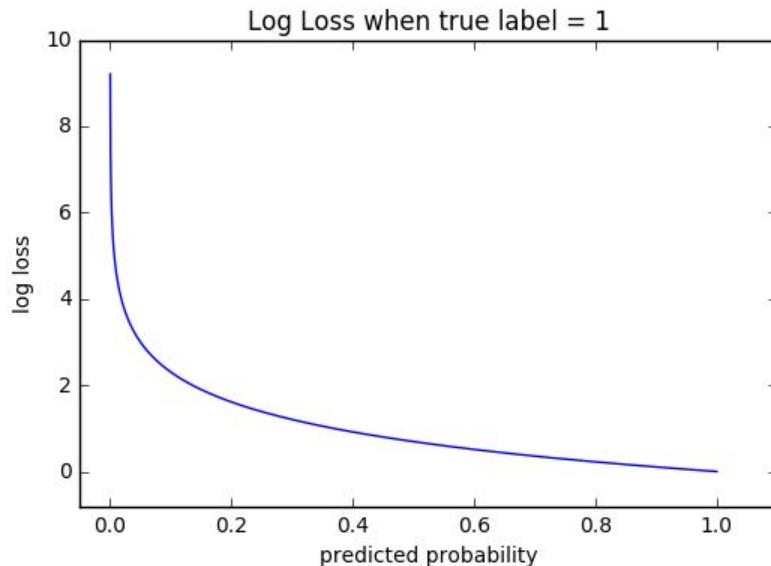
# LogLoss пример

Получили наши вероятности:



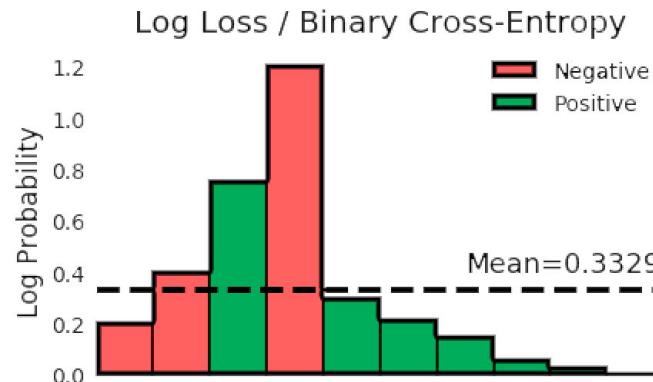
# LogLoss пример

График ниже дает нам ясную картину – если прогнозируемая вероятность исходного класса становится ближе к нулю, потеря возрастает экспоненциально:



# LogLoss пример

Давайте возьмем (отрицательный) логарифм вероятностей - это соответствующий loss каждой точки. После усреднения получим:



Мы успешно посчитали **binary cross-entropy / log loss = 0.3329!**

# LogLoss

- **Binary:**  $\text{LogLoss} = \frac{1}{N} \sum_{i=1}^N [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$   $y_i \in \mathbb{R}$ ,  $\hat{y}_i \in \mathbb{R}$
- **Multiclass:**  $\text{LogLoss} = -\frac{1}{N} \sum_{i=1}^N \sum_{l=1}^L y_{i,l} \log(\hat{y}_{i,l})$   $y_i \in \mathbb{R}^L$ ,  $\hat{y}_i \in \mathbb{R}^L$
- **На практике:**  $\text{LogLoss} = -\frac{1}{N} \sum_{i=1}^N \sum_{l=1}^L y_{i,l} \log(\min(\max(\hat{y}_{i,l}, 10^{-15}), 1 - 10^{-15}))$

# LogLoss

- Можно использовать только, если ваш классификатор выдаёт вероятности
- Поможет сильно наказывать неверные значения
- Лучшая константа:

$$\text{LogLoss} = -\frac{1}{N} \sum_{i=1}^N y_i \log(\alpha) + (1 - y_i) \log(1 - \alpha)$$

# Регрессия

# План

- Обозначения
- MSE, RMSE, R-squared
- MAE
- (R)MSPE, MAPE
- (R)MSLE

# Обозначения

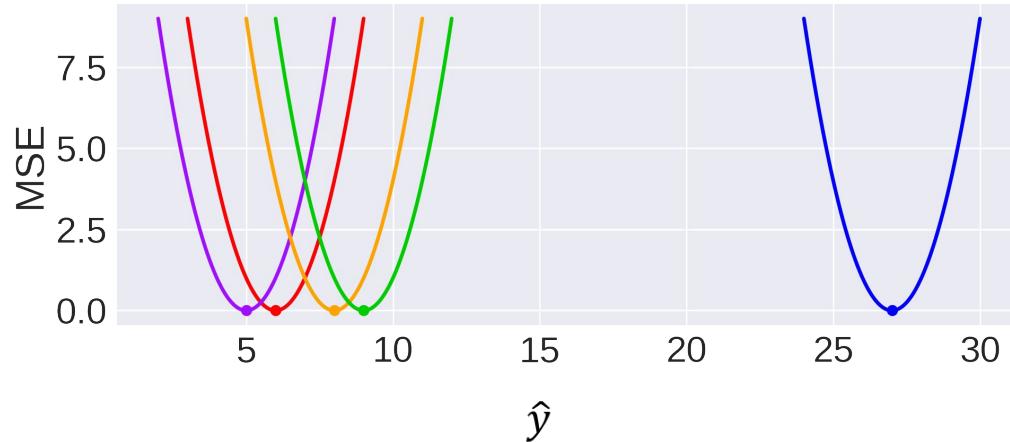
- $N$  – число объектов
- $y \in \mathbb{R}^N$  – значения target
- $\hat{y} \in \mathbb{R}^N$  – предсказания
- $\hat{y}_i \in \mathbb{R}$  – предсказания для i-го объекта
- $y_i \in \mathbb{R}$  – target для i-го объекта

# MSE: Mean Squared Error

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2$$

Data:

X	Y
...	5
...	9
...	8
...	6
...	27



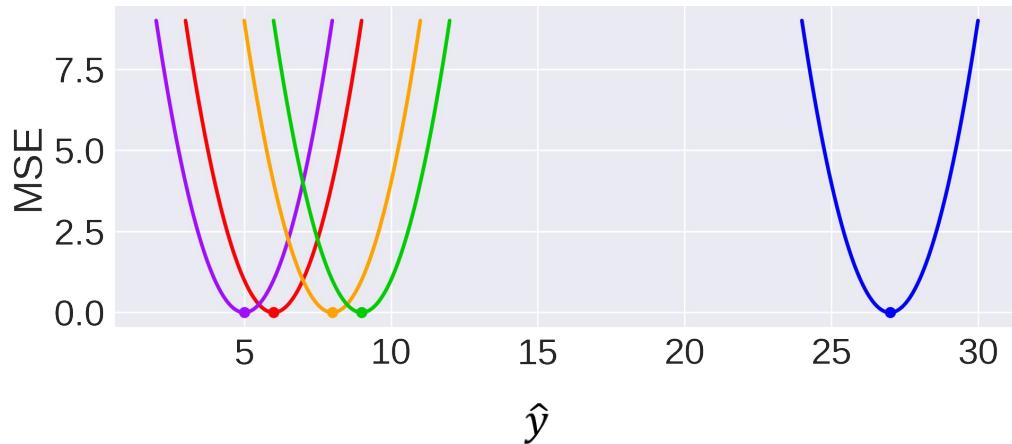
# MSE: оптимальная константа

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^N (y_i - \alpha)^2$$

Data:

X	Y
...	5
...	9
...	8
...	6
...	27

Лучшая константа: ?



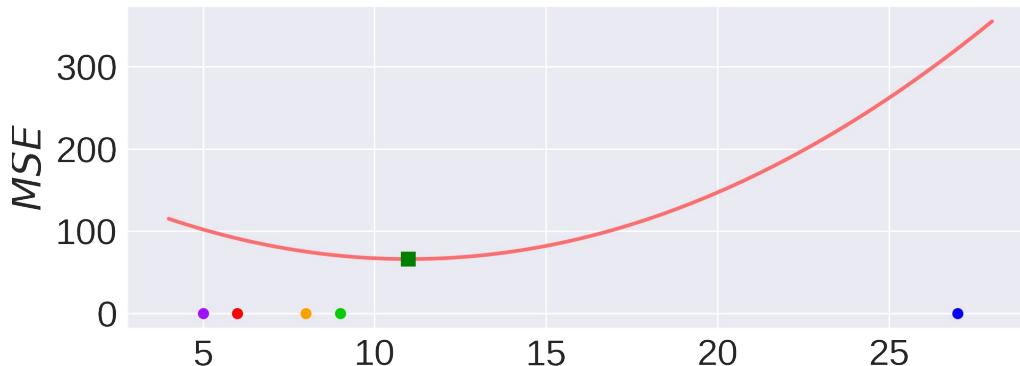
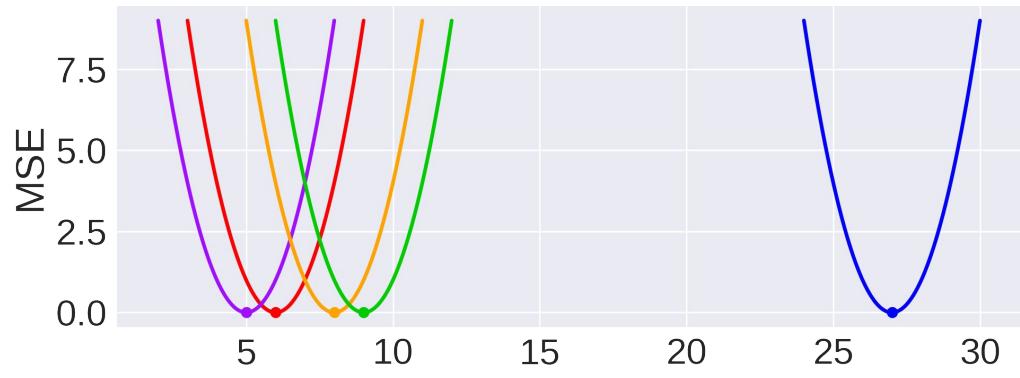
# MSE: оптимальная константа

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^N (y_i - \alpha)^2$$

Data:

X	Y
...	5
...	9
...	8
...	6
...	27

Лучшая константа: target mean



# MSE заметки: RMSE (Root mean squared error)

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2$$

Root mean square error:

- $\text{RMSE} = \sqrt{\frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2} = \sqrt{\text{MSE}}$

$$\text{MSE}(a) > \text{MSE}(b) \iff \text{RMSE}(a) > \text{RMSE}(b)$$

# MSE заметки: RMSE (Root mean squared error)

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2$$

Root mean square error:

- $\text{RMSE} = \sqrt{\frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2} = \sqrt{\text{MSE}}$

$$\text{MSE}(a) > \text{MSE}(b) \iff \text{RMSE}(a) > \text{RMSE}(b)$$

- $$\frac{\partial \text{RMSE}}{\partial \hat{y}_i} = \frac{1}{2\sqrt{\text{MSE}}} \frac{\partial \text{MSE}}{\partial \hat{y}_i}$$

# MSE заметки: RMSE (Root mean squared error)

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2$$

R-squared:

$$R^2 = 1 - \frac{\frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2}{\frac{1}{N} \sum_{i=1}^N (y_i - \bar{y})^2} = 1 - \frac{\text{MSE}}{\frac{1}{N} \sum_{i=1}^N (y_i - \bar{y})^2}$$

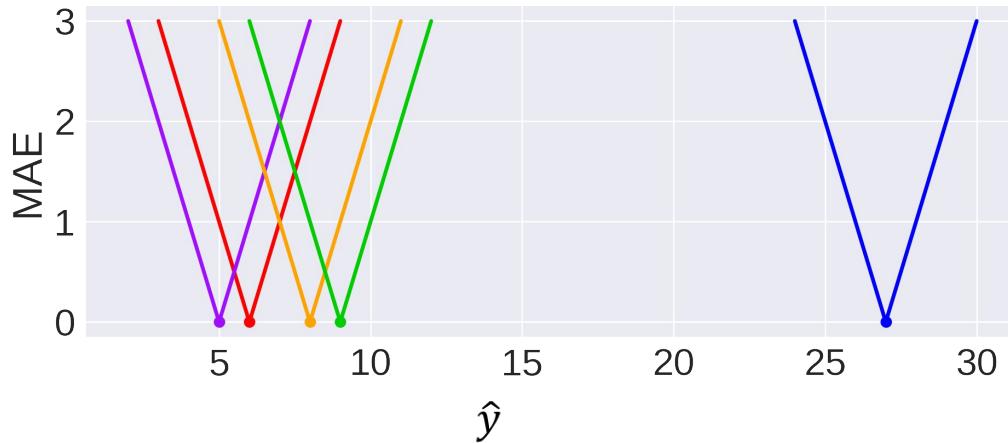
$$\bar{y} = \frac{1}{N} \sum_{i=1}^N y_i$$

# MAE: Mean Absolute Error

$$\text{MAE} = \frac{1}{N} \sum_{i=1}^N |y_i - \hat{y}_i|$$

Data:

X	Y
-1	5
1	9
-2	8
3	6
3	27



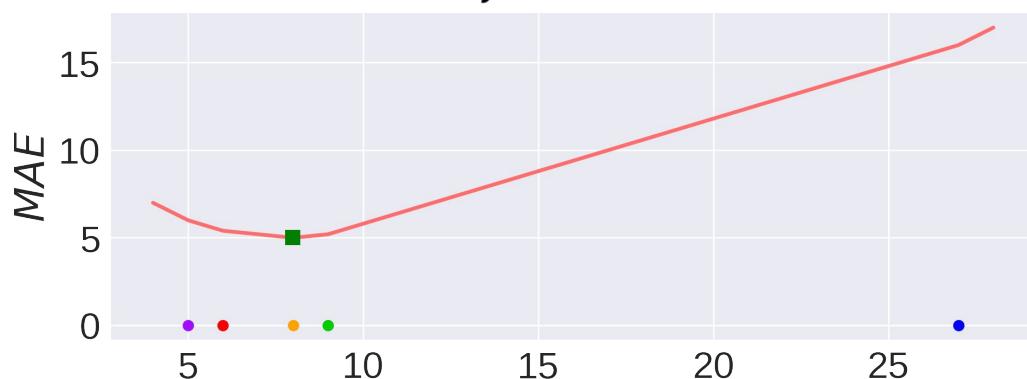
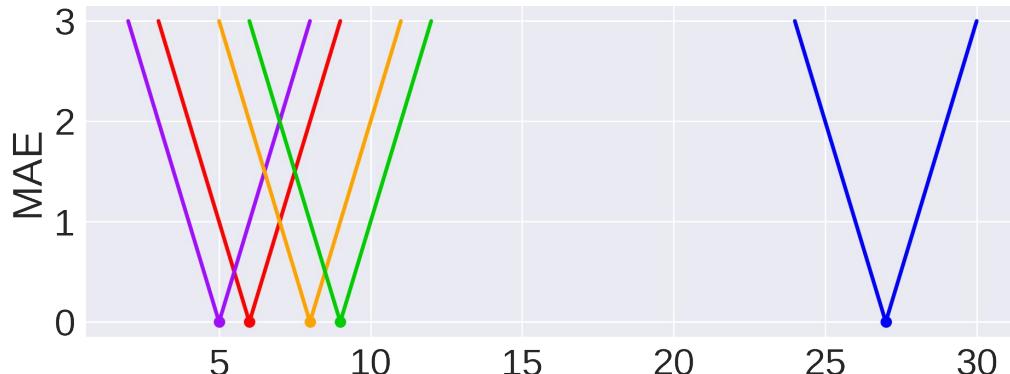
# MAE: Mean Absolute Error

$$\text{MAE} = \frac{1}{N} \sum_{i=1}^N |y_i - \alpha|$$

Data:

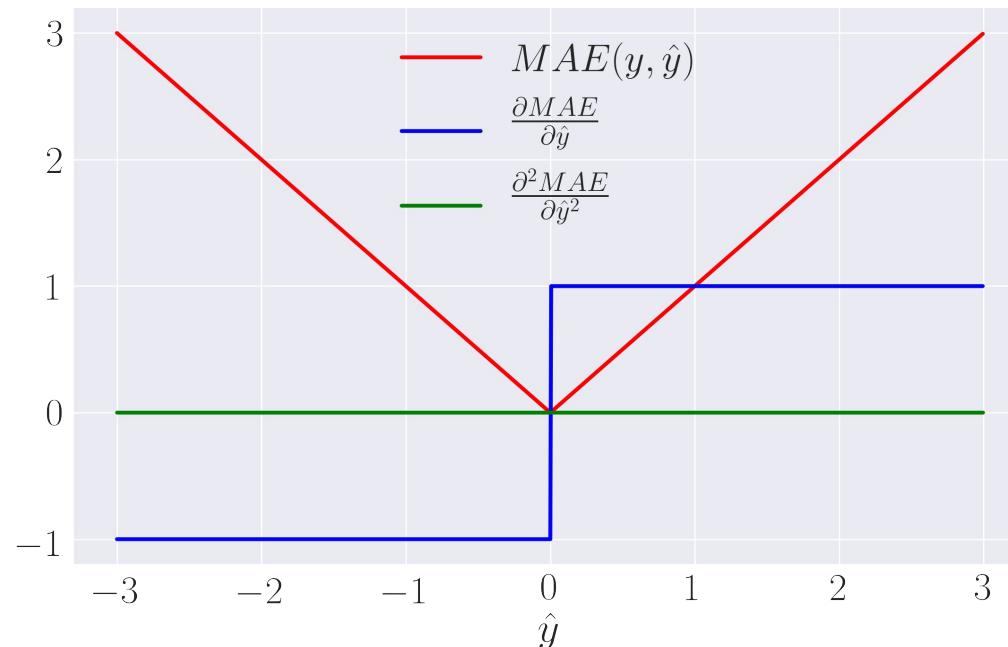
X	Y
-1	5
1	9
-2	8
3	6
3	27

Лучшая константа: target median



# МАЕ: производные

$$\text{MAE} = \frac{1}{N} \sum_{i=1}^N |y_i - \hat{y}_i|$$



# MAE vs MSE

- Есть выбросы (outliers) в данных?
  - Используйте MAE
- Это точно выбросы (outliers)?
  - Используйте MAE
- Или это обычные значения?
  - Используйте MSE

# Из MSE и MAE в MSPE и MAPE

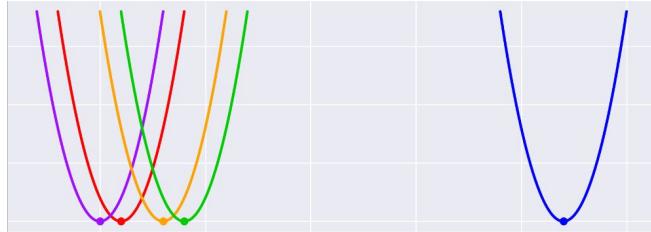
- Магазин 1: предсказано 9, продано 10, MSE = 1
  - Магазин 2: предсказано 999, продано 1000, MSE = 1
- 
- Магазин 1: предсказано 9, продано 10, MSE = 1
  - Магазин 2: предсказано 900, продано 1000, MSE = 1000
- 
- Магазин 1: предсказано 9, продано 10, relative\_metric = 1
  - Магазин 2: предсказано 900, продано 1000, relative\_metric = 1

\* MAPE - Mean absolute percentage error

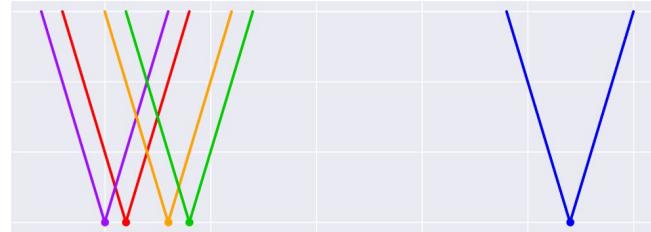
\* MSPE - Mean squared percentage error

# Из MSE и MAE в MSPE и MAPE

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2$$



$$\text{MAE} = \frac{1}{N} \sum_{i=1}^N |y_i - \hat{y}_i|$$



$$\text{MSPE} = \frac{100\%}{N} \sum_{i=1}^N \left( \frac{y_i - \hat{y}_i}{y_i} \right)^2$$



$$\text{MAPE} = \frac{100\%}{N} \sum_{i=1}^N \left| \frac{y_i - \hat{y}_i}{y_i} \right|$$



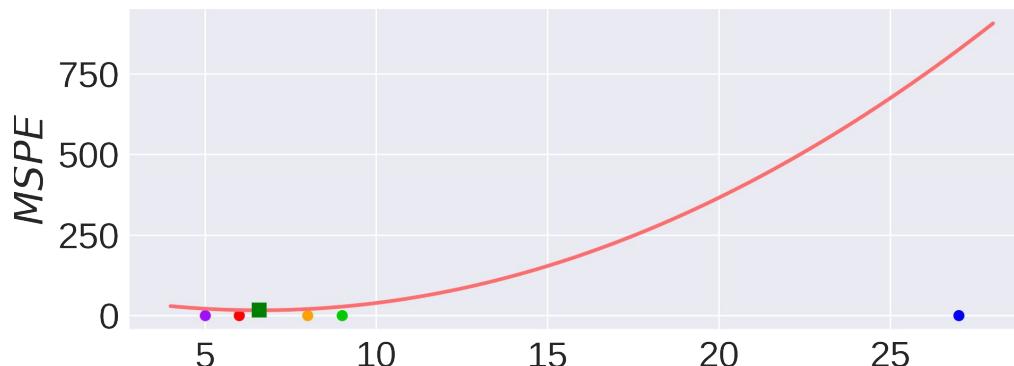
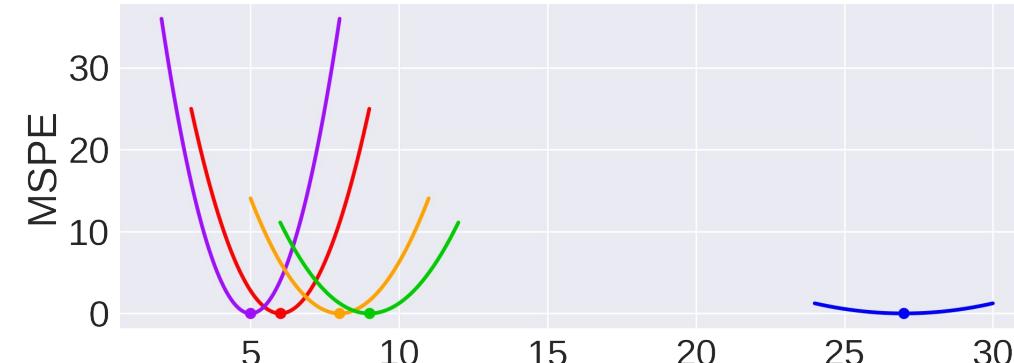
# MSPE: константа

$$MSPE = \frac{100\%}{N} \sum_{i=1}^N \left( \frac{y_i - \alpha}{y_i} \right)^2$$

Лучшая константа:  
weighted target mean

Data:

X	Y
-1	4
1	3
-2	6
3	7
3	25



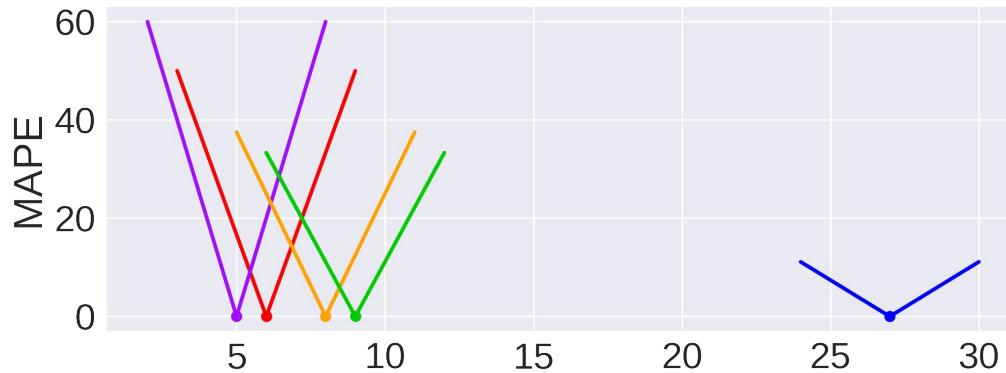
# MAPE: константа

$$\text{MAPE} = \frac{100\%}{N} \sum_{i=1}^N \left| \frac{y_i - \alpha}{y_i} \right|$$

Лучшая константа:  
?

Data:

X	Y
-1	4
1	3
-2	6
3	7
3	25



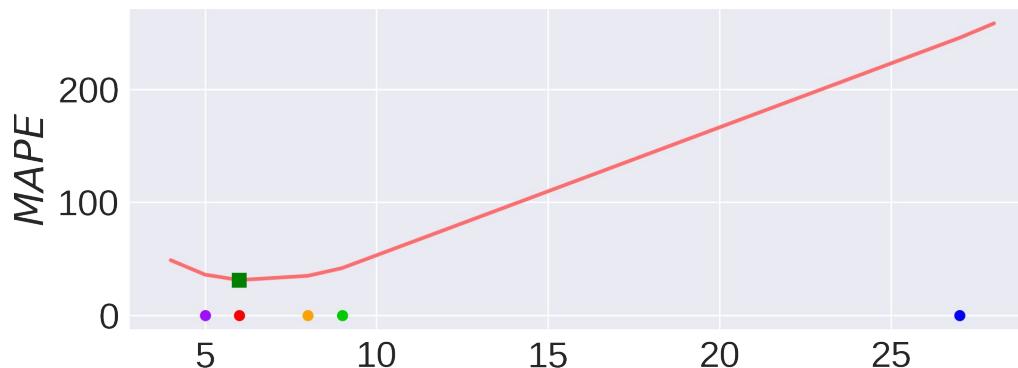
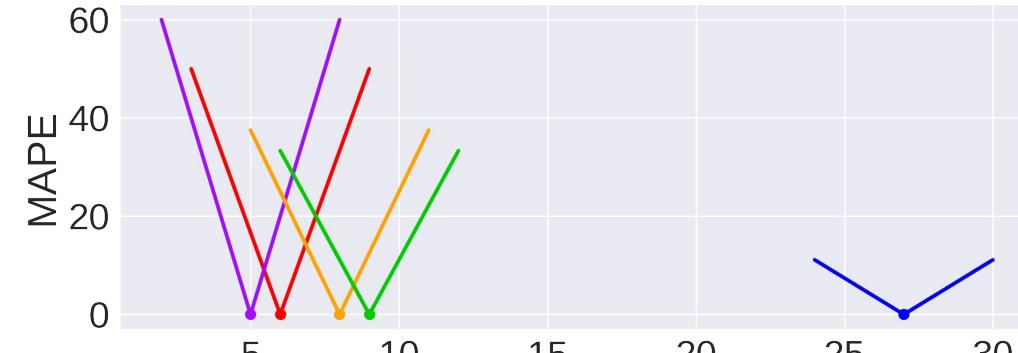
# MAPE: константа

$$\text{MAPE} = \frac{100\%}{N} \sum_{i=1}^N \left| \frac{y_i - \alpha}{y_i} \right|$$

Лучшая константа:  
weighted target median

Data:

X	Y
-1	4
1	3
-2	6
3	7
3	25



# (R)MSLE: Root Mean Square Logarithmic Error

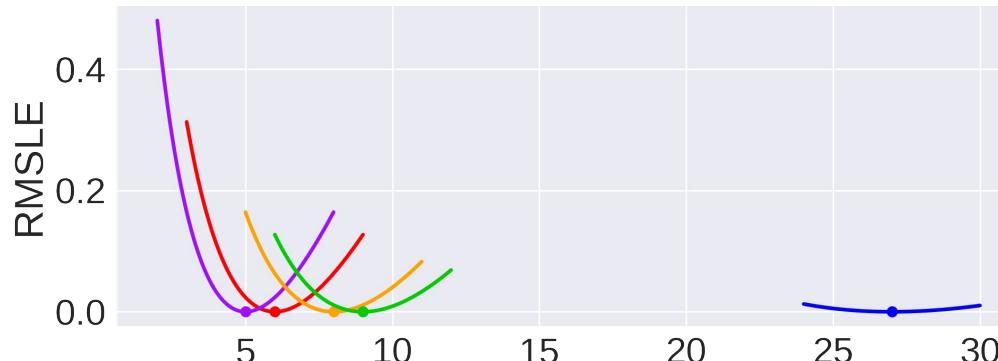
$$\begin{aligned}\text{RMSLE} &= \sqrt{\frac{1}{N} \sum_{i=1}^N (\log(y_i + 1) - \log(\hat{y}_i + 1))^2} = \\ &= RMSE(\log(y_i + 1), \log(\hat{y}_i + 1)) = \\ &= \sqrt{MSE(\log(y_i + 1), \log(\hat{y}_i + 1))}\end{aligned}$$

# (R)MSLE: Root Mean Square Logarithmic Error

Data:

X	Y
-1	4
1	3
-2	6
3	7
3	25

$$\text{RMSLE} = \sqrt{\frac{1}{N} \sum_{i=1}^N (\log(y_i + 1) - \log(\hat{y}_i + 1))^2}$$



# (R)MSLE: константа

$$\begin{aligned}\text{RMSLE} &= \sqrt{\frac{1}{N} \sum_{i=1}^N (\log(y_i + 1) - \log(\alpha + 1))^2} = \\ &= RMSE(\log(y_i + 1), \log(\alpha + 1)) = \\ &= \sqrt{MSE(\log(y_i + 1), \log(\alpha + 1))}\end{aligned}$$

# (R)MSLE: константа

$$\begin{aligned}\text{RMSLE} &= \sqrt{\frac{1}{N} \sum_{i=1}^N (\log(y_i + 1) - \log(\alpha + 1))^2} = \\ &= RMSE(\log(y_i + 1), \log(\alpha + 1)) = \\ &= \sqrt{MSE(\log(y_i + 1), \log(\alpha + 1))}\end{aligned}$$

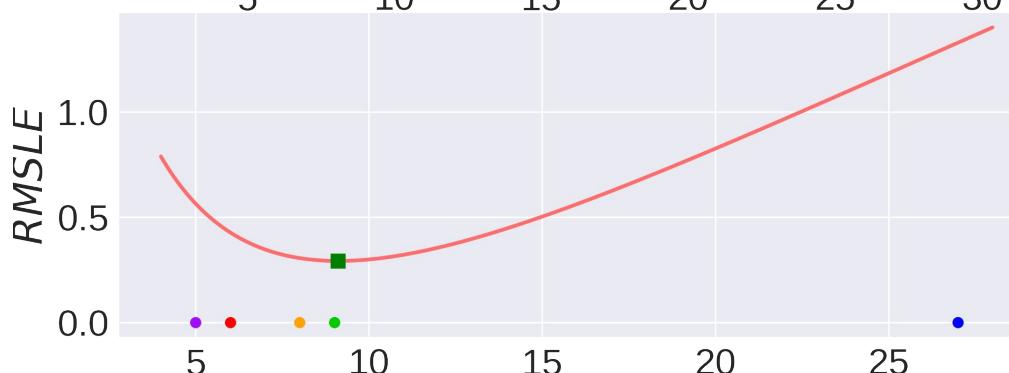
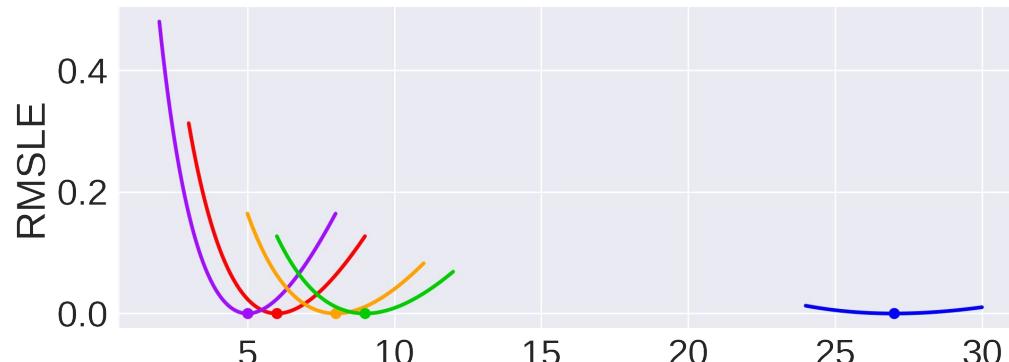
- Лучшая константа в логарифмическом пространстве это mean target value
- Чтобы получить ответ нужно взять экспоненту

# (R)MSLE: константа

$$\text{RMSLE} = \sqrt{\frac{1}{N} \sum_{i=1}^N (\log(y_i + 1) - \log(\alpha + 1))^2}$$

Data:

X	Y
-1	4
1	3
-2	6
3	7
3	25



# Сравнение результатов

Metric	Constant
MSE	11
RMSLE	9.11
MAE	8
MSPE	6.6
MAPE	6

# ИТОГИ

Метрики, чувствительные к относительным ошибкам:

- (R)MSPE
  - Взвешенная версия MSE
- MAPE
  - Взвешенная версия MAE
- (R)MSLE
  - MSE в логарифмическом пространстве

# Оптимизация метрик

# План

- Loss или Metric
- Подходы к оптимизации target metric
- Оптимизация регрессионных метрик
- Оптимизация метрик классификации

# Loss vs Metric

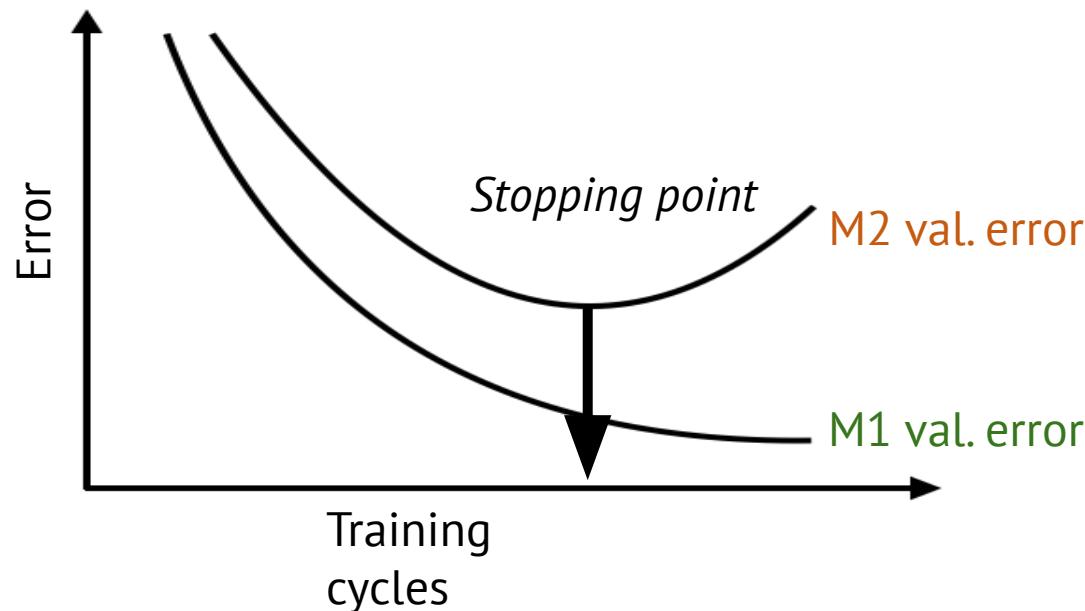
- Target metric это то, что мы хотим оптимизировать
- Optimization loss это то, что модель оптимизирует

# Подходы к оптимизации target metric

- Просто запускай нужную модель!
  - MSE, Logloss
- Препроцессинг train и оптимизация других метрик:
  - MSPE, MAPE, RMSLE, ...
- Оптимизация других метрик, постпроцессинг предсказаний:
  - Accuracy, Карра
- Написать свою loss function:
  - Любую, если сможешь :)
- Оптимизируй другую метрику, используй ранний выход
  - Любая

# Подходы к оптимизации target metric

- Оптимизируем метрику M1, отслеживаем метрику M2
  - Выходим когда M2 лучшая



# RMSE, MSE, R-squared

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2$$

Как ты ее оптимизируешь?

Просто обучи правильную модель!

$$\text{RMSE} = \sqrt{\text{MSE}} \quad R^2 = 1 - \frac{\text{MSE}}{\frac{1}{N} \sum_{i=1}^N (y_i - \bar{y})^2}$$

# MAE

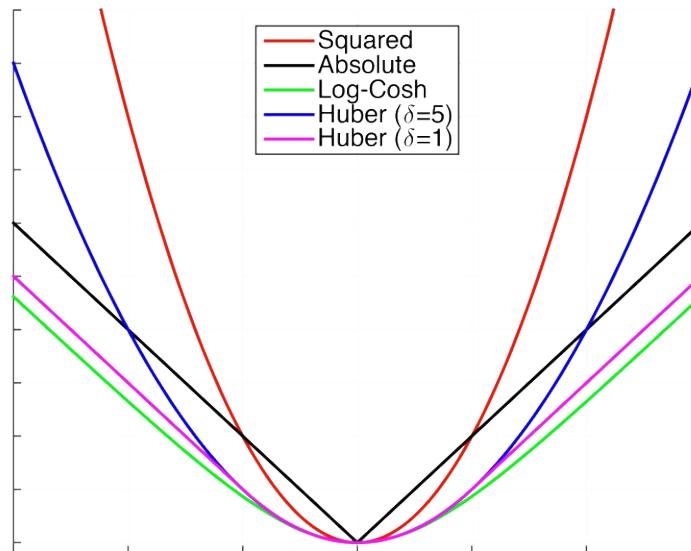
$$\text{MAE} = \frac{1}{N} \sum_{i=1}^N |y_i - \hat{y}_i|$$

Как ты ее оптимизируешь?

Просто обучи правильную модель!

# MAE: оптимальная константа

$$\text{MAE} = \frac{1}{N} \sum_{i=1}^N |y_i - \hat{y}_i|$$



# MSPE и MAPE

$$\text{MSPE} = \frac{100\%}{N} \sum_{i=1}^N \left( \frac{y_i - \hat{y}_i}{y_i} \right)^2 \quad \text{MAPE} = \frac{100\%}{N} \sum_{i=1}^N \left| \frac{y_i - \hat{y}_i}{y_i} \right|$$

Как ты их оптимизируешь?

Просто ~~fit правильную модель!~~

# MSPE (MAPE) как weighted MSE (MAE)

Семпл весов

$$\text{MSPE} = \frac{100\%}{N} \sum_{i=1}^N \left( \frac{y_i - \hat{y}_i}{y_i} \right)^2 \quad \left| \quad w_i = \frac{1/y_i^2}{\sum_{i=1}^N 1/y_i^2} \right.$$

---

$$\text{MAPE} = \frac{100\%}{N} \sum_{i=1}^N \left| \frac{y_i - \hat{y}_i}{y_i} \right| \quad \left| \quad w_i = \frac{1/y_i}{\sum_{i=1}^N 1/y_i} \right.$$

# MSPE (MAPE)

- Используйте веса для сэмпла ('sample weights')
  - И используйте MSE (MAE)
- Ресэмпл train set
  - df.sample(weights=sample\_weights)
  - И используй любую модель оптимизируя MSE (MAE)
  - Обычно нужно много раз использовать ресэмпл и усреднение

# RMSLE

$$\begin{aligned}\text{RMSLE} &= \sqrt{\frac{1}{N} \sum_{i=1}^N (\log(y_i + 1) - \log(\hat{y}_i + 1))^2} = \\ &= \sqrt{MSE(\log(y_i + 1), \log(\hat{y}_i + 1))}\end{aligned}$$

Train:

- Изменение target:

$$z_i = \log(y_i + 1)$$

- Обучение модели с MSE loss

Test:

- Изменяйте предсказания обратно:  
 $\hat{y}_i = \exp(\hat{z}_i) - 1$

# Logloss

$$\text{LogLoss} = -\frac{1}{N} \sum_{i=1}^N y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)$$

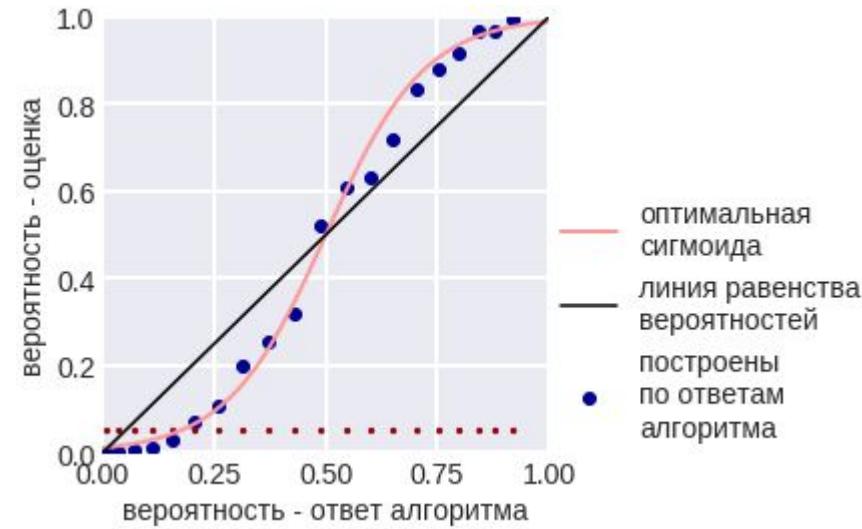
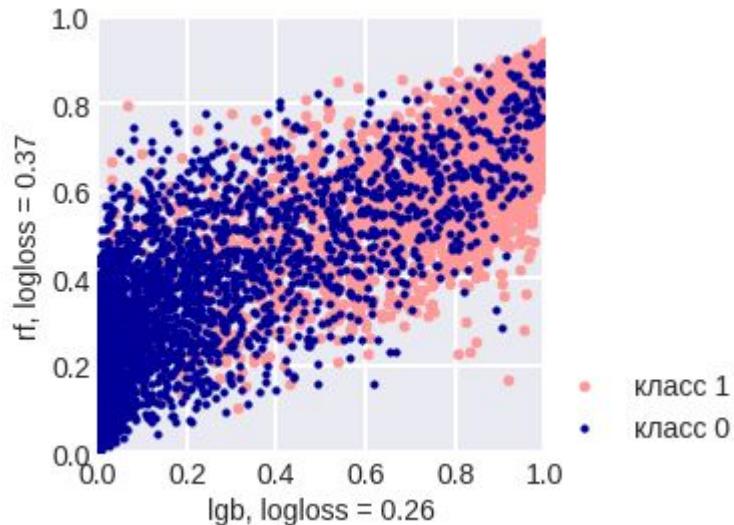
Как мы оптимизируем её?

Просто обучи нужную модель!  
(или откалибруй чужие)

# Logloss

- Правильные вероятности:
  - Возьми все объекты с счетом примерно ~ 0.8
    - 80% их - class 1
    - 20% их - class 0
- Неправильные вероятности:
  - Возьми все объекты с счетом примерно ~ 0.8
    - 50% их - class 1
    - 50% их - class 0

# Калибровка LogLoss



# Проверка распределения на основе logloss

- Засыпаем константное предсказание р

$$r = \frac{\log loss + \log(1 - p)}{\log\left(\frac{1-p}{p}\right)}$$

p=0.369 (const), public logloss=0.554 => r=0.174 (доля 1 в public leaderboard)

<https://www.kaggle.com/davidthaler/how-many-1-s-are-in-the-public-lb/notebook>

- Далее можно подкорректировать вероятности

<https://swarbrickjones.wordpress.com/2017/03/28/cross-entropy-and-training-test-class-imbalance/>

<https://www.kaggle.com/c/quora-question-pairs/discussion/31179>

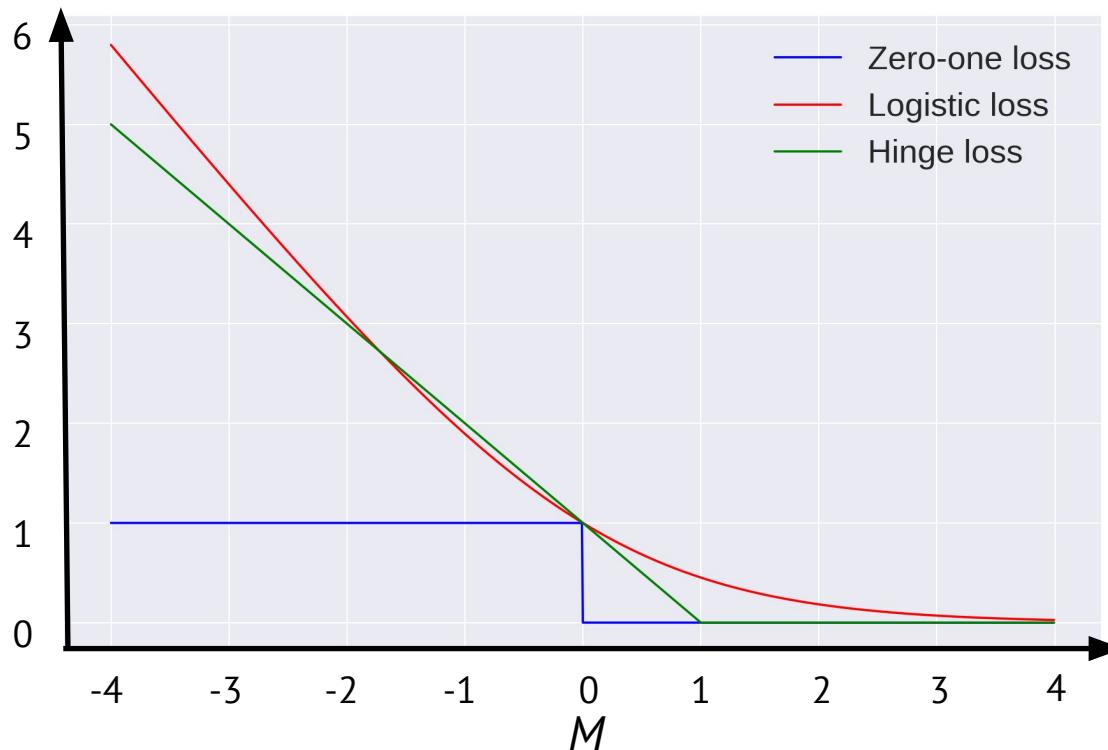
# Accuracy

$$\text{Accuracy} = \frac{1}{N} \sum_{i=1}^N [\hat{y}_i = y_i]$$

Как мы оптимизируем её?

Обучи модель на любой метрике и  
настрой трешхолд

# Accuracy



# Accuracy

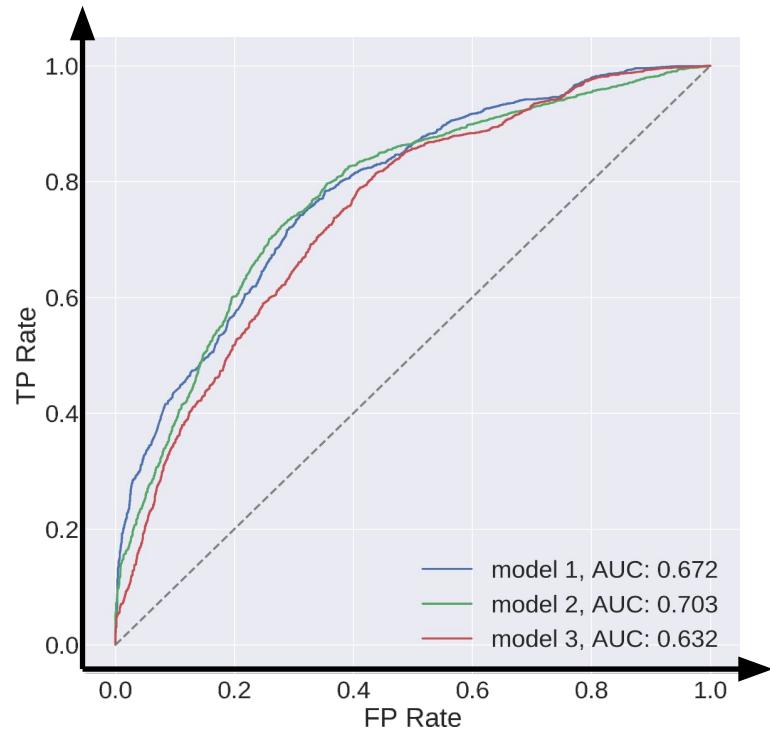
$$\text{Accuracy} = \frac{1}{N} \sum_{i=1}^N [[f(x) > b] = y_i]$$

$$b = 0.5 \quad \Rightarrow \quad \text{Accuracy} = \frac{6}{7}$$

$$b = 0.7 \quad \Rightarrow \quad \text{Accuracy} = 1$$

# AUC (ROC)

Как мы оптимизируем ее?  
Запусти правильную модель  
(или просто оптимизируй logloss)

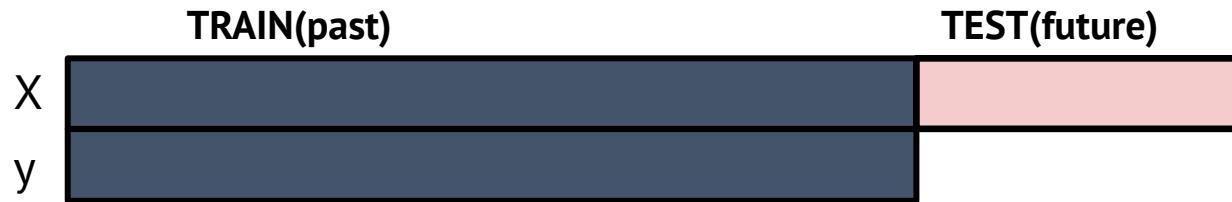


# Валидация

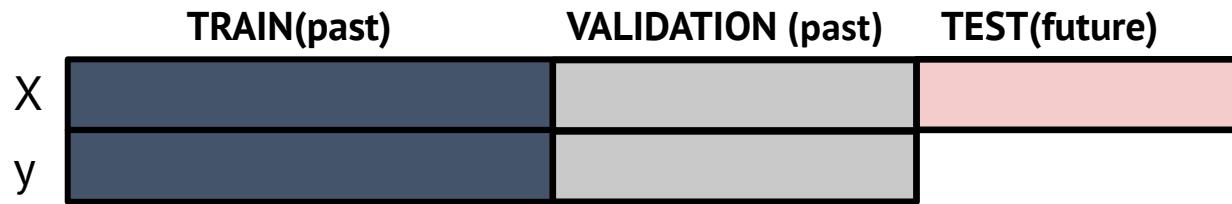
# План

- Мотивация
- Стратегии валидации
- Выбор числа и способа разбиений
- Анализ причин способов избежать валидационных проблем

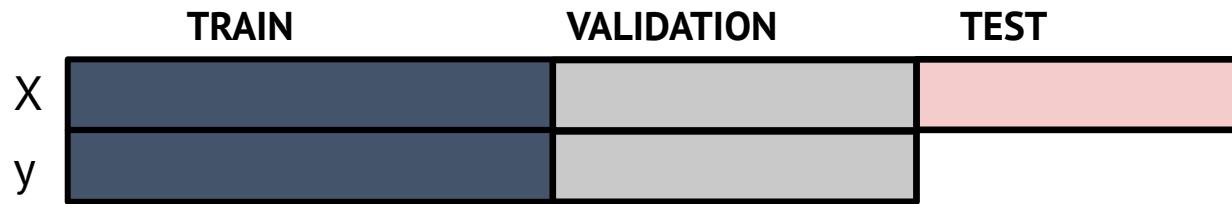
# Валидация - пример



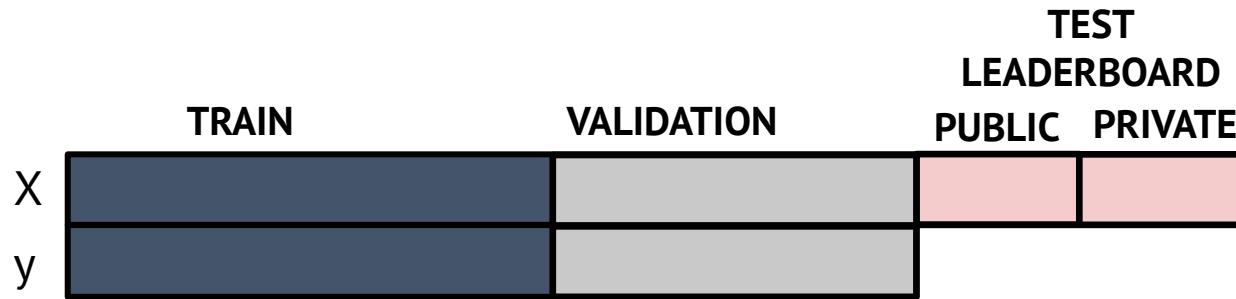
# Валидация - пример



# Валидация - пример



# Валидация – пример (соревнования)



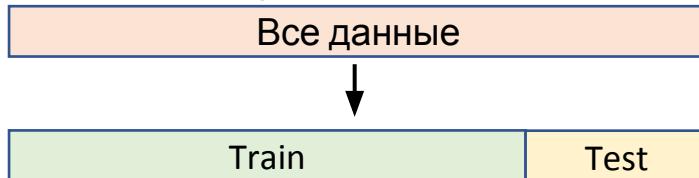
# Мотивация

- Хорошая валидация не гарантирует успех
- Но плохая валидация почти гарантированно приводит к провалу
- Хорошая валидация должна повторять разделение на train/test в данных
- Хорошая валидация должна коррелировать с Public LB (не обязательно повторять, но меняться синхронно)

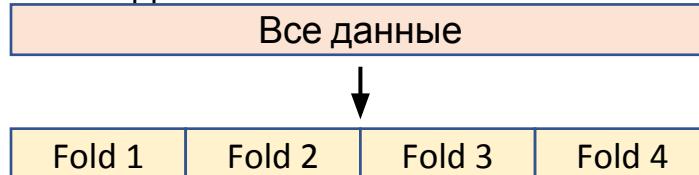
# Стратегии валидации

- По количеству разбиений

- Одиночное разбиение (holdout)



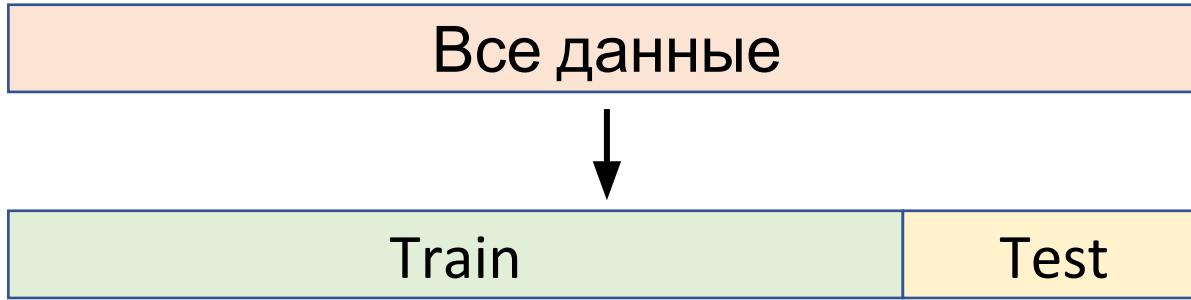
- Фолды



- Leave-one-out

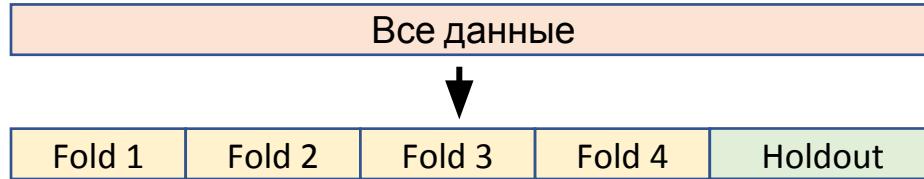


# Holdout



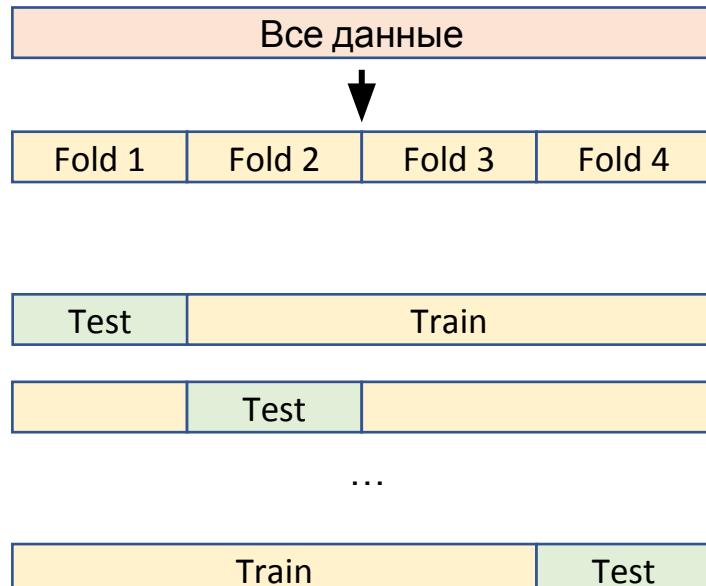
# Holdout – pros/cons

- + Хорошее начало для бейзлайна
- + Когда не хватает вычислительных мощностей
- + Использовать совместно с другой стратегией (фолды) для финальной проверки



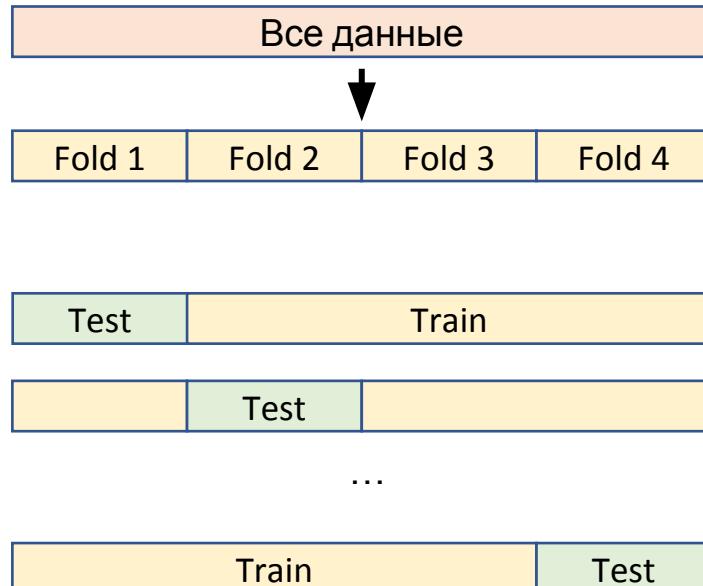
- Не охватывает все данные – может выучиться на неверный паттерн

# Разбиение на n\_splits



Усредняем по  
выбранному  
**n\_splits**  
(обычно 3-10)

# Разбиение leave one out



# Разбиение на фолды - sklearn

In [26]:

```
1 from sklearn.model_selection import KFold
2 from sklearn.model_selection import LeaveOneOut
3
4 loo = LeaveOneOut().split(X)
5 kf = KFold(n_splits=5, shuffle=True, random_state=132).split(X)
6 # .split() - возвращает генератор, после 1-го прохода он становится пуст
7 # можно обернуть это в список
8 loo = list(LeaveOneOut().split(X))
9 kf = list(KFold(n_splits=5, shuffle=True, random_state=132).split(X))
10
11 i_fold = 0
12 kf[i_fold] # tuple массивов индексов (train_ind, test_ind)
```

Out[26]: (array([ 0, 1, 2, 3, 4, 5, 6, 7, 8, 10, 12, 13, 15, 16, 17, 18]),  
 array([ 9, 11, 14, 19]))

# Фолды, LOO – pros/cons

- + Охватывают все данные
- + Можно сразу усреднить  $n_{splits}$  предсказаний и получить более качественную модель
- Вычислительное время пропорционально числу разбиений
- Некоторые разбиения могут получиться меньше/хуже других (когда?)

# Стратификация

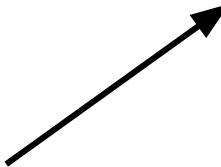
Объекты и их таргеты

0	1	0	0	1	1	1	0
---	---	---	---	---	---	---	---

# Стратификация

Объекты и их таргеты

0	1	0	0	1	1	1	0
0	1	0	0	1	1	1	0
0.5		0		1		0.5	

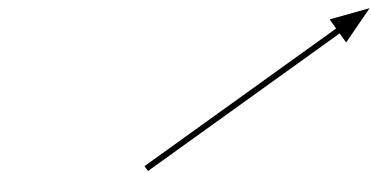


Получилось не очень  
равномерно.  
Оранжевый/зеленый  
фолд по факту  
бесполезен

# Стратификация

Объекты и их таргеты

0	1	0	0	1	1	1	0
0	1	0	0	1	1	1	0
0.5		0			1		0.5
0	1	0	0	1	1	1	0
0.5		0.5	0.5	0.5	0.5	0.5	



Получилось  
равномерно

# Стратификация

Объекты и их таргеты

0	1	0	0	1	1	1	0
0	1	0	0	1	1	1	0
0.5		0			1		0.5
0	1	0	0	1	1	1	0
0.5	0.5	0.5	0.5	0.5	0.5		

Стратификация уместна:

- Небольшие выборки
- Несбалансированные выборки
- Многоклассовая классификация

# Стратификация - sklearn

## Было

```
In [29]: 1 from sklearn.model_selection import train_test_split
2 from sklearn.model_selection import KFold
3
4 X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.6, test_size=0.3,
5                                         shuffle=True, random_state=117)
6
7 kf = list(KFold(n_splits=5, shuffle=True, random_state=132).split(X))
```

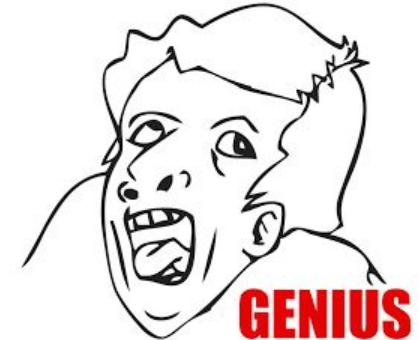
## Стало

```
In [30]: 1 from sklearn.model_selection import train_test_split
2 from sklearn.model_selection import StratifiedKFold
3
4 X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.6, test_size=0.3,
5                                         shuffle=True, random_state=117,
6                                         stratify=y)
7
8 skf = list(StratifiedKFold(n_splits=5, shuffle=True, random_state=132).split(X, y))
```

# Стратегии валидации

Что общего у данных проблем?

- Предсказывание продаж по данным следующего месяца
- Предсказывание вероятности угона машины, обучаясь на следующем полисе где ее не угнали
- Предсказывание вероятности что  $A==B$ , зная что  $A==C$ ,  $C==B$



# Стратегии валидации

Что общего у данных проблем?

- Предсказывание продаж по данным следующего месяца
- Предсказывание вероятности угона машины, обучаясь на следующем полисе где ее не угнали
- Предсказывание вероятности что  $A==B$ , зная что  $A==C$ ,  $C==B$

Некорректное использование информации из обучения в валидацию!

**Основная задача – воспроизвести разбиение на Train/Test от организаторов.**

(и убедиться что ваша валидация коррелирует с LB)

Если в Train/Test время не перемешано, то можно тоже не мешать 😊

# Пример – задача Hearthstone



# Hearthstone - проблема

- Данные – JSON описание текущего состояния матча
- Нужно предсказать вероятность победы
- StratifiedKFold: 0.9+ ROC-AUC , 10000+ итераций и продолжал расти
- Public LB: 0.83 ROC-AUC 😞

В чем причина?



```
{'decision': 0,  
'gamestate_id': 2886512,  
'opponent': {'hero': {'armor': 0,  
'attack': 1,  
'hero_card_id': 754,  
'hp': 29,  
'special_skill_used': True,  
'weapon_durability': 2},  
'played_cards': [],  
'stats': {'crystals_all': 7,  
'crystals_current': 0,  
'deck_count': 17,  
'fatigue_damage': 0,  
'hand_count': 6,  
'played_minions_count': 0,  
'spell_dmg_bonus': 0}},  
'player': {'hand': [{"attack": 3,  
'charge': False,  
'crystals_cost': 4,  
'forgetful': False,  
'freezing': False,  
'gamestate_id': 2886512,  
'hero': 494,  
'hp': 5,  
'id': 198,  
'name': "Sen'jin Shieldmasta",  
'poisonous': False,  
'shield': False,  
'stealth': False,  
'taunt': True,  
'turn': 8,  
'type': 'MINION',  
'windfury': False}},
```

# Hearthstone - причина

- В обучающих данных присутствовали несколько раз одни и те же матчи. На LB были другие матчи
- Необходимо было построить валидацию, разделяющую матчи
- Один из способов – разделить фолды по парам **герой\_игрока-герой\_оппонента** (герой является константой в течение матча)

	winrate									
hero2	druid	hunter	mage	paladin	priest	rogue	shaman	warlock	warrior	
hero1										
druid	0.481535	0.493059	0.499659	0.495676	0.493580	0.498933	0.481967	0.509310	0.500472	
hunter	0.511360	0.484766	0.501777	0.502375	0.498902	0.497866	0.486868	0.493590	0.524871	
mage	0.472710	0.490263	0.478505	0.500101	0.492019	0.495812	0.483165	0.480531	0.462521	
paladin	0.481438	0.489381	0.509378	0.497768	0.513639	0.488758	0.485734	0.489486	0.476657	
priest	0.510035	0.478049	0.506536	0.505544	0.511002	0.503174	0.498279	0.497251	0.485901	
rogue	0.480907	0.481082	0.499254	0.484115	0.481939	0.484539	0.482277	0.496418	0.476980	
shaman	0.501467	0.493579	0.514331	0.502595	0.496285	0.504875	0.497927	0.510762	0.504431	
warlock	0.492162	0.506332	0.510062	0.491555	0.494365	0.506532	0.496003	0.505349	0.482116	
warrior	0.490658	0.493252	0.520759	0.497189	0.521228	0.501505	0.501851	0.480517	0.521044	

# Стратегии валидации

- По типу разбиений

- Случайное

Если наши данные представляют независимые наблюдения, не привязанные ко времени, id.

- По времени

Есть временная зависимость

- По id

Есть группы объектов, которые не пересекаются

- Комбинирование id, время, пары объектов

Предсказание отклика **новых** клиентов в будущем

Анализ дубликатов объектов

# Валидация по времени

Fold1	Fold2	Fold3	Fold4	Fold5	Fold6
	Train		Validation		
		Train		Validation	
			Train		Validation

Как  
вариант

Fold1	Fold2	Fold3	Fold4	Fold5	Fold6
	Train		Validation1	Validation2	Validation3
		Train		Validation1	Validation2
			Train		Validation1

# Валидация по времени - sklearn

```
1 from sklearn.model_selection import TimeSeriesSplit  
2  
3 tss = list(TimeSeriesSplit(n_splits=5).split(X))  
4 tss[0]  
  
(array([0, 1, 2, 3, 4]), array([5, 6, 7]))
```

# Валидация по id

- Разные id могут обозначать разные группы клиентов
- Разные объекты могут описывать разные состояния одного и того же объекта

# Валидация по id - sklearn

```
1 from sklearn.model_selection import GroupKFold
2 from sklearn.model_selection import LeaveOneGroupOut
3 gkf = list(GroupKFold(n_splits=2).split(X, y, groups=groups))
4
5 # параметр y здесь не участвует в стратификации и по факту бесполезен
6 print(groups[gkf[0][0]], groups[gkf[0][1]])
7 print(y[gkf[0][0]], y[gkf[0][1]], end='\n----\n')
8
9 logo = list(LeaveOneGroupOut().split(X, y, groups=groups))
10
11 # параметр y здесь не участвует в стратификации и по факту бесполезен
12 print(len(logo), np.unique(groups))
13 print(groups[logo[0][0]], groups[logo[0][1]])
14 print(y[gkf[0][0]], y[gkf[0][1]])
```

```
[0 2 0 2 0 2 0 2 0 2] [1 3 1 3 1 3 1 3 1 3]
[1 1 1 1 1 1 1 1 1] [0 0 0 0 0 0 0 0 0 0]
-----
4 [0 1 2 3]
[1 2 3 1 2 3 1 2 3 1 2 3 1 2 3] [0 0 0 0 0]
[1 1 1 1 1 1 1 1 1] [0 0 0 0 0 0 0 0 0]
```

Если хотим и по группам и стратифицированно – придется заморочиться

# Более сложные стратегии валидации

Quora Question Pairs



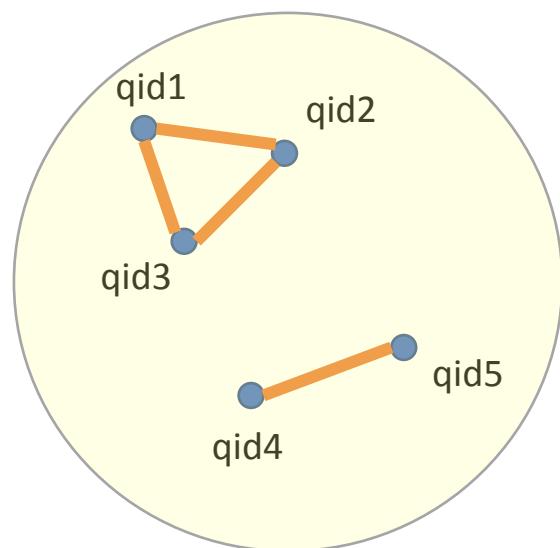
	<b>id</b>	<b>qid1</b>	<b>qid2</b>	<b>question1</b>	<b>question2</b>	<b>is_duplicate</b>
<b>0</b>	0	1	2	What is the step by step guide to invest in share market in india?	What is the step by step guide to invest in share market?	0
<b>1</b>	1	3	4	What is the story of Kohinoor (Koh-i-Noor) Diamond?	What would happen if the Indian government stole the Kohinoor (Koh-i-Noor) diamond back?	0
<b>2</b>	2	5	6	How can I increase the speed of my internet connection while using a VPN?	How can Internet speed be increased by hacking through DNS?	0
<b>3</b>	3	7	8	Why am I mentally very lonely? How can I solve it?	Find the remainder when $23^{24}$ is divided by 24,23?	0
<b>4</b>	4	9	10	Which one dissolve in water quickly sugar, salt, methane and carbon di oxide?	Which fish would survive in salt water?	0
<b>5</b>	5	11	12	Astrology: I am a Capricorn Sun Cap moon and cap rising...what does that say about me?	I'm a triple Capricorn (Sun, Moon and ascendant in Capricorn) What does this say about me?	1

# Более сложные стратегии валидации

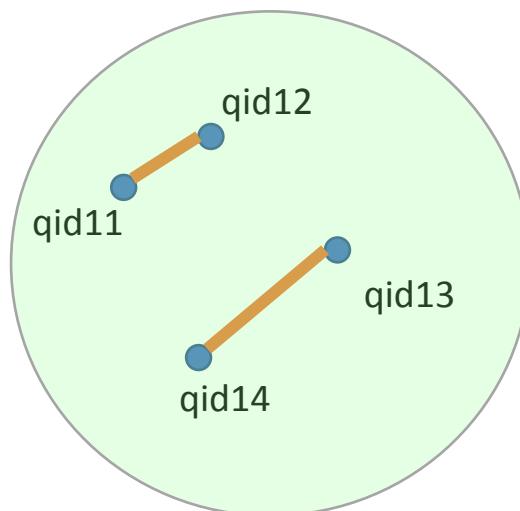
Quora Question Pairs

# Quora

Train



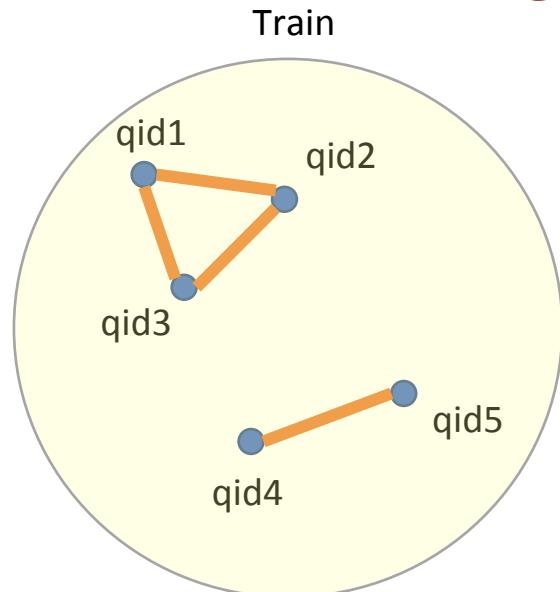
Test



# Валидация по компонентам связности

Quora Question Pairs

# Quora



Разделение 1

Fold1:

qid1 – qid2

qid1 – qid3

...

Fold2:

qid2 – qid3

qid4 – qid5

...

Разделение 2

Fold1:

qid1 – qid2

qid1 – qid3

...

Fold2:

qid4 – qid5

...

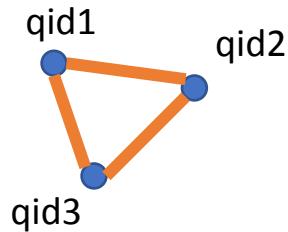
Что  
лучше?

# Более сложные стратегии валидации

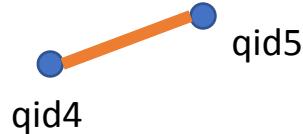
Quora Question Pairs



Разделение по компонентам  
связности



Fold1



Fold2

...

# Проблемы валидации

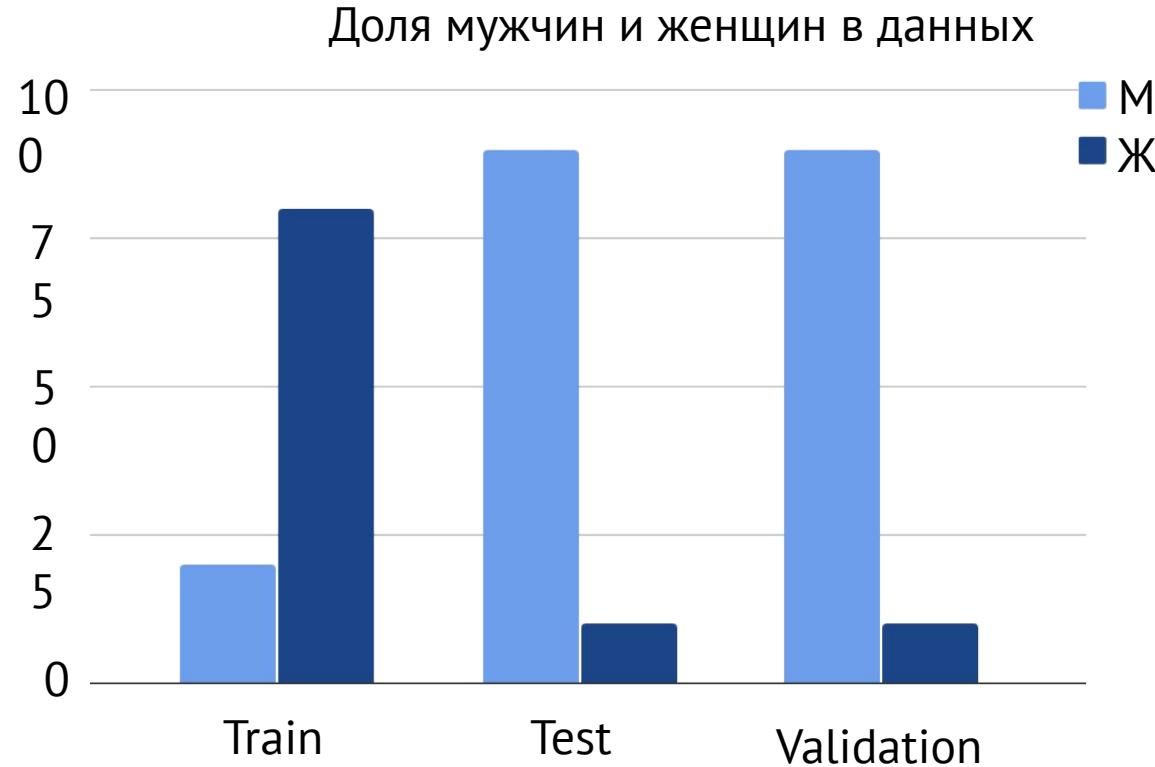
Основные причины проблем валидации:

- Недостаток данных в public leaderboard
- Некорректное разделение на train/test
- Разные распределения в train/test

# Разные распределения признаков



# Разные распределения признаков



# Leaderboard Shuffle

Вы можете оказаться здесь

In the money								
#	△pub	Team Name	Kernel	Team Members	Score	Entries	Last	
1	▲ 3	Shize & Nir		 	0.82907	298	3y	
2	▲ 848	kg_joi		 	0.82906	35	3y	
3	▲ 6	#1 Leustagos		  +8	0.82853	227	3y	
4	▲ 1	Why so noise?		 	0.82846	195	3y	
5	▲ 2024	Michael Hartman		 	0.82834	22	3y	
6	▲ 229	Noah Xiao @ Accenture		 	0.82834	44	3y	
7	▲ 4	Rolling Stones (Can't Get No [...]		 	0.82829	198	3y	
8	▲ 718	Matt Motoki		 	0.82824	30	3y	
9	▲ 4	no one		 	0.82823	54	3y	
10	▲ 1240	Bang Nguyen		 	0.82818	35	3y	

# Leaderboard Shuffle

А можете здесь (это одно и то же соревнование)

1600	▼ 928	caiomsoouza		0.82547	33	3y
1601	▼ 917	Booster		0.82547	14	3y
1602	▼ 917	back22010		0.82547	11	3y
1603	▼ 916	Anorexic Hippo		0.82547	25	3y
1604	▼ 1361	جانے کب ہوں گے کم، بینک والوں کے غم		0.82547	9	3y
1605	▼ 967	AnonSci		0.82547	25	3y
1606	▼ 1155	vettipaiyan	 FREE BUGS	0.82547	15	3y
1607	▼ 917	paulperry		0.82547	11	3y
1608	▼ 1390	fallLeaf		0.82547	62	3y
1609	▼ 916	ArjoonnSharma		0.82547	34	3y
1610	▼ 915	__=.=		0.82547	21	3y
1611	▼ 915	Nilesh Kadam		0.82547	9	3y
1612	▼ 1151	ShankerDS		0.82547	15	3y

# Leaderboard Shuffle

Вы можете ожидать шаффл если:

- Рандом
- Мало данных
- Разные распределения на public/private

# Валидация - заключение

- При большой дисперсии на этапе валидации нужно принимать дополнительные меры:
  - Усреднение предсказания разных фолдов
  - Обучение модели на одном разбиении – оценка на другом
- Если метрика сабмита не совпадает с метрикой на этапе валидации:
  - Убедиться достаточно ли данных в public LB
  - Проверить модель на предмет переобучения
  - Убедиться что мы выбрали верную стратегию разбиения
  - Проверить совпадают ли распределения таргета/признаков на train/test
- Быть готовым к шаффлу в случае:
  - Рандома (т.е. всегда 😊)
  - Небольшого количества данных
  - Разные распределения public/private

# ИТОГИ

- Неправильный подход к метрике и валидации – верный способ испортить даже самые лучшие признаки
- Метрики
  - Классификация
    - Accuracy
    - Метрики из Confusion Matrix
    - Roc Auc
    - Log Loss
  - Регрессия
    - MSE, RMSE, R-squared
    - MAE
    - (R)MSPE, MAPE
    - (R)MSLE
  - Оптимизация метрик
- Валидация
  - Стратегии валидации
  - Выбор числа и способа разбиений
  - Анализ причин способов избежать валидационных проблем

# Домашнее задание

- Представьте что классы представлены двумя множествами, изображенные треугольниками. В том месте где треугольники пересекаются присутствуют объекты обоих классов. Вертикальная прямая отображает работу классификатора (левее - в красный класс, правее - в синий). Меняя положение прямой по горизонтальной оси мы меняем threshold.
- Построить графики ROC, PR, precision(threshold), recall(threshold)

