Системы искусственного интеллекта

Лекция 3

Контроль качества алгоритмов.

Сравнение по качеству

Запорожцев Иван Федорович
zaporozhtsev.if.work@gmail.com

Контроль качества: базовые идеи

Отложенный контроль

held-out | validation data

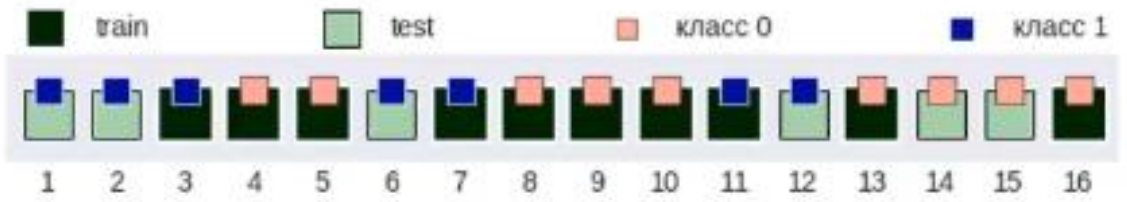
Разбить выборку на две части:
обучающую и тестовую (контрольную)

Выборку делим на две части:

- **обучение** – здесь обучение алгоритма
- **отложенный контроль** – здесь оценка качества / выбор алгоритма с наименьшей ошибкой

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(x, y,
test_size=0.33, random_state=41)
```




```
X_train, X_test, y_train, y_test = train_test_split(x, y,
test_size=0.33, shuffle=False)
```

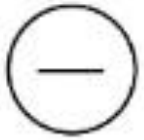
Контроль качества: базовые идеи

Обычно делят 80% / 20%

Больше тест – надёжнее оценка,
больше обучение – алгоритм похож на финальный

Оценка ошибки зависит от конкретной выбранной отложенной выборки

Часто сильно меняется при другом выборе



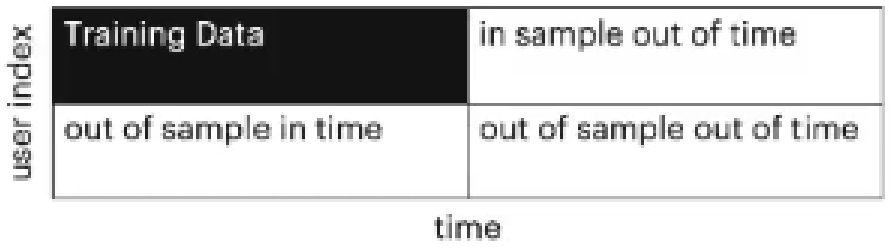
Если переобучить алгоритм для всех данных, то мы не знаем оценку его ошибки

В каком-то смысле, неустранимый недостаток)

Контроль качества: базовые идеи

Тест/валидация моделирует реальную работу алгоритма!

- деление по группам
- предсказание будущего
- стратификация
  `stratify=None`



Тест/валидация должен быть случайным

`shuffle=True`

(или специально подготовленным)

- убирать дубликаты

Нельзя явно или неявно использовать метки объектов, на которых оцениваешь ошибку (качество)

- делать корректно селекцию признаков / MTE

Контроль качества: базовые идеи

Способы контроля

«Many sources instead classify holdout as a type of simple validation, rather than a simple or degenerate form of cross-validation»

en.wikipedia.org

- **Отложенный контроль**
  Held-out data, hold-out set
- **Скользящий контроль**
  Cross-validation – иногда так называют «всё»
- **Бутстреп**
  Bootstrap
- **Контроль по времени**
  Out-of-time-контроль

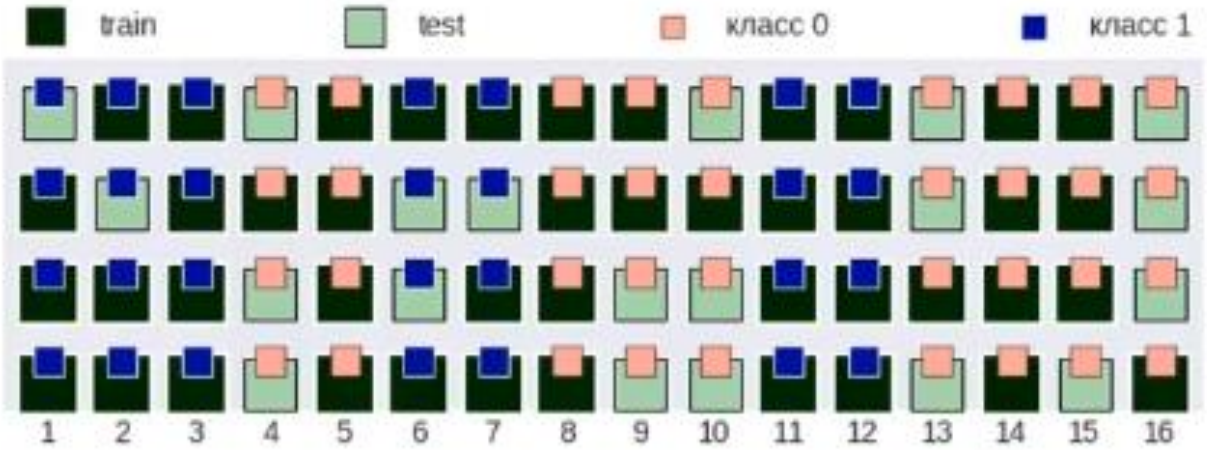
Дальше подробно расскажем про разные способы контроля...

Контроль качества: базовые идеи

Random Subsampling Cross-Validation

k раз случайно выбираем отложенный контроль, усредняем ошибки на всех отложенных выборках

```
sklearn.model_selection.ShuffleSplit(n_splits=4,
                                      test_size=0.3,
                                      train_size=None,
                                      random_state=None)
```

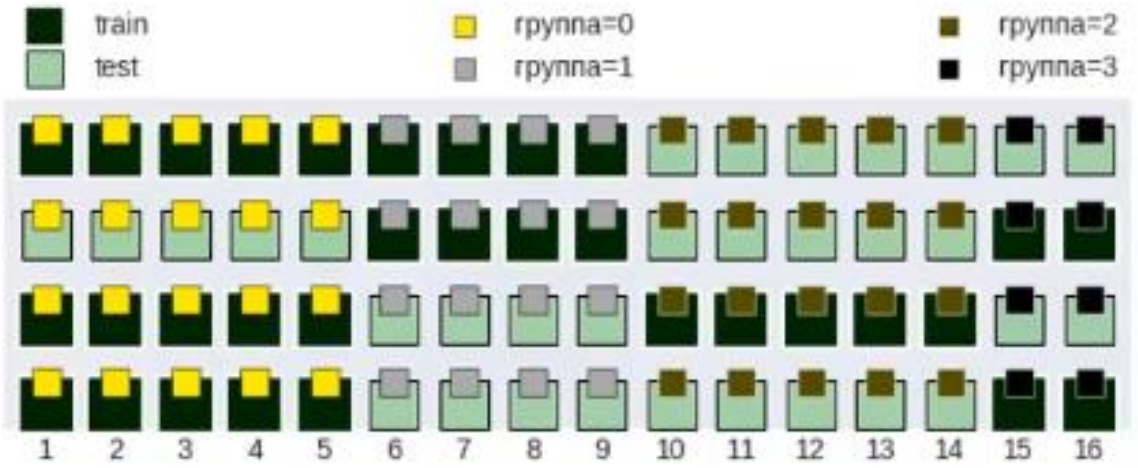


Контроль качества: базовые идеи

Random Subsampling Cross-Validation

Без разбиения групп

```
sklearn.model_selection.GroupShuffleSplit(n_splits=4,
                                          test_size=0.3,
                                          train_size=None,
                                          random_state=None)
for t, (itrain, itest) in enumerate(cv.split(x,
groups=g)):
    ...
```

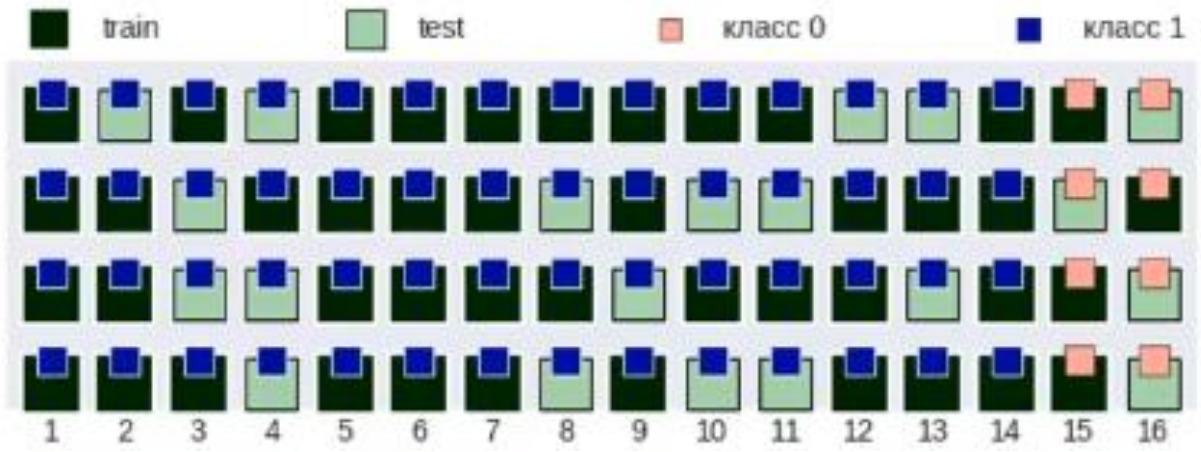


Контроль качества: базовые идеи

Random Subsampling Cross-Validation

Сохраняя пропорции классов

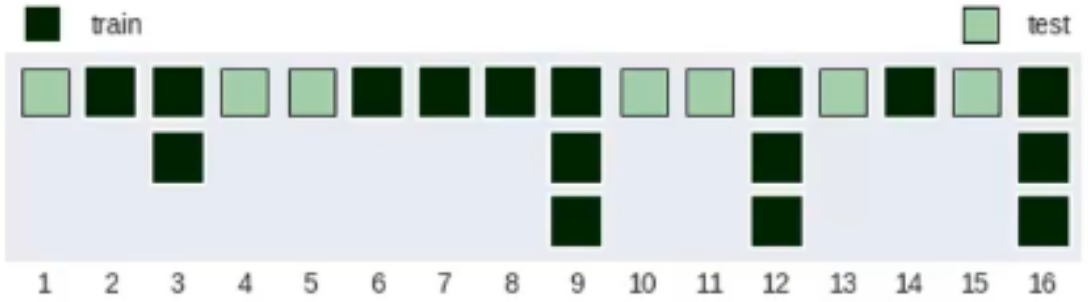
```
sklearn.model_selection.StratifiedShuffleSplit(n_splits=4,
                                               test_size=0.3,
                                               train_size=None,
                                               random_state=None)
for t, (itrain, itest) in enumerate(cv.split(x, groups=y)):
    ...
```



Контроль качества. Бутстрэп

С помощью выбора с возвращением формируется подвыборка полного объёма m, на которой производится обучение модели

На остальных объектах (которые не попали в обучение) – контроль

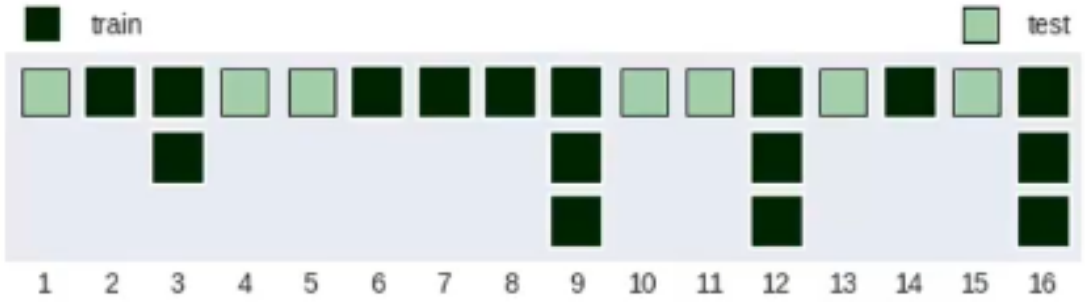



```
i_train = [9, 16, 14, 9, 7, 12, 3, 12, 9, 8, 3, 2, 16, 12,
6, 16]
i_test =  [1, 4, 5, 10, 11, 13, 15]
```

Контроль качества. Бутстрэп

С помощью выбора с возвращением формируется подвыборка полного объёма m, на которой производится обучение модели

На остальных объектах (которые не попали в обучение) – контроль




```
i_train = [9, 16, 14, 9, 7, 12, 3, 12, 9, 8, 3, 2, 16, 12,
6, 16]
i_test =  [1, 4, 5, 10, 11, 13, 15]
```

Какова вероятность, что объект не попадёт в трэйн:

$$\left(1-\frac{1}{m}\right)^m \approx e^{-1} \approx 0.37 = 37\% \text{ выборки}$$

$$\lim_{m\to\infty}\left(1+\frac{1}{m}\right)^m = e$$

$$\lim_{m\to\infty}\left(1-\frac{1}{m}\right)^m = \left\{\begin{matrix}замена\\ t:=-m\end{matrix}\right\} = \lim_{t\to\infty}\left(1+\frac{1}{t}\right)^{-t} =$$

$$= \lim_{t\to\infty}\left(\underbrace{\left(1+\frac{1}{t}\right)^t}_{\to e \text{ при } t\to\infty}\right)^{-1} = e^{-1}$$

модель учится на выборке того же объёма, что и итоговая (которую мы обучим по всей выборке)

- использует не все данные
- есть дубликаты

с точки зрения распределения бутстреп-выборка похожа на исходную

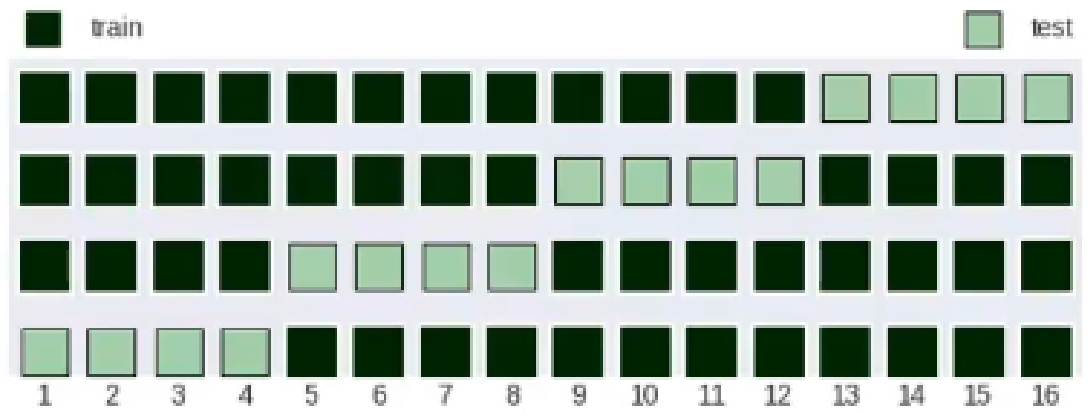
Перекрёстная проверка по фолдам

k-fold cross-validation

- Разделить выборку на k примерно равных частей (обычно k=10)
- Цикл по i = 1...k
  - i-я часть – для теста,
    объединение остальных – для обучения
- Усреднить k ошибок, вычисленных на разных итерациях цикла на валидациях

можно использовать дисперсию для оценки доверия к полученному качеству

```
sklearn.model_selection.KFold(n_splits=4,
                              shuffle=False,
                              random_state=None)
```

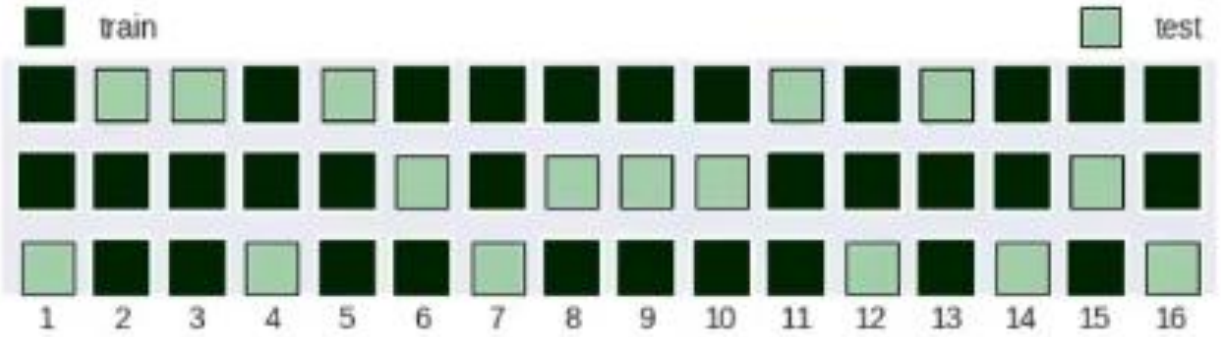


Перекрёстная проверка по фолдам

k-fold cross-validation

С перемешиванием

```
sklearn.model_selection.KFold(n_splits=3, shuffle=True,
random_state=None)
```

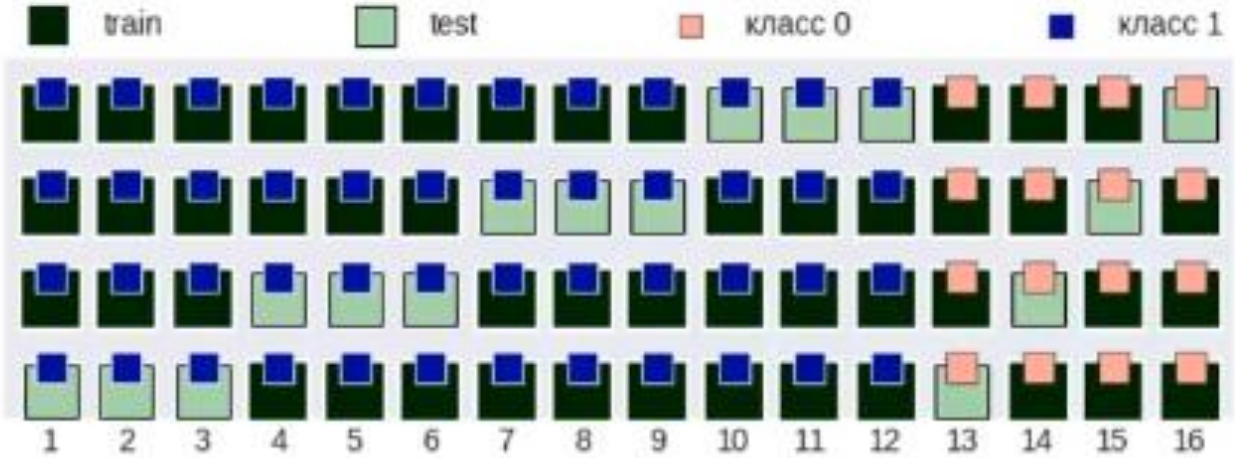


k-fold CV = k-fold cross-validation

Перекрёстная проверка по фолдам

Сохранение пропорций классов

```
sklearn.model_selection.StratifiedKFold(n_splits=4,
                                        shuffle=False,
                                        random_state=None)
```



перемешиваем:

shuffle=True

Перекрёстная проверка по фолдам

Не разбиваем группы

```
sklearn.model_selection.GroupKFold(n_splits=4)


from sklearn.model_selection import LeaveOneGroupOut
```

есть **`sklearn.model_selection.PredefinedSplit`**
разбиение индуцированное группами

Контроль по времени

Out-of-time-контроль

— Часто не получится сделать много контролей

Слишком маленькая предыстория

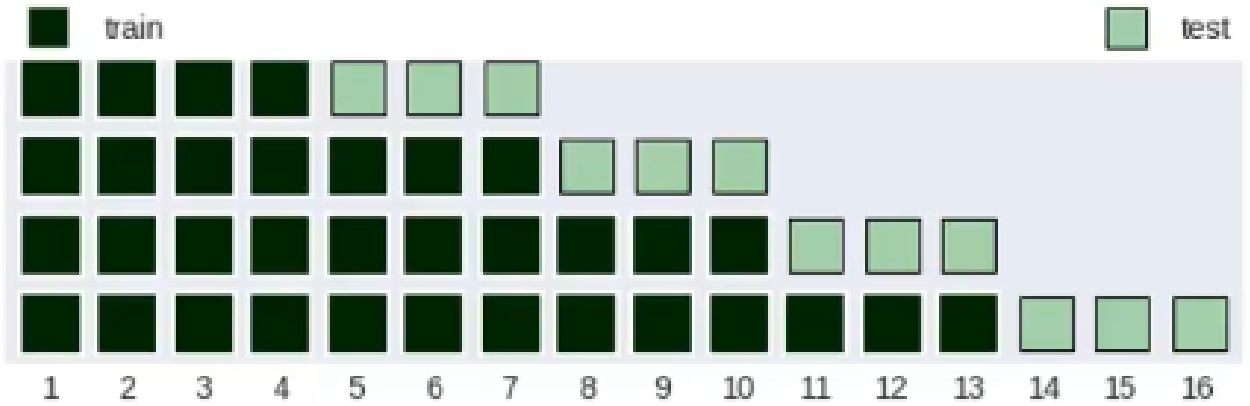
\+ Можно организовать «под контекст»

На следующий день, неделю, месяц

TimeSeriesSplit: разбиения временных рядов

(Time series cross-validation)

```
sklearn.model_selection.TimeSeriesSplit(n_splits=4, # сколько делить
                                        max_train_size=None, # сколько делить
                                        test_size=None, # n_samples // (n_splits + 1)
                                                        # если gap=0
                                        gap=0) # сколько "откусывать" в конце
                                                # по умолчанию не используется
```



Схемы с повторениями

«Сделать несколько раз»

```
model_selection.RepeatedKFold (n_splits=2, n_repeats=2, random_state=0)
model_selection.RepeatedStratifiedKFold(n_splits=2, n_repeats=2, random_state=0)
```

```
import numpy as np
from sklearn.model_selection import RepeatedStratifiedKFold
X = np.array([[1, 2], [3, 4], [1, 2], [3, 4]])
y = np.array([0, 0, 1, 1])
rskf = RepeatedStratifiedKFold(n_splits=2, n_repeats=2,
                               random_state=36851234)
for train_index, test_index in rskf.split(X, y):
    print( "TRAIN:", train_index, "TEST:", test_index)
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]

    TRAIN: [1 2] TEST: [0 3]
    TRAIN: [0 3] TEST: [1 2]
    TRAIN: [1 3] TEST: [0 2]
    TRAIN: [0 2] TEST: [1 3]
```

Проблема валидации и тестирования

Мы говорили про проблему контроля качества алгоритма.

Но чтобы его контролировать, надо выбрать алгоритм → проблема выбора алгоритма

Model Selection

Так назывался основной модуль в представленном коде

Выбор модели в широком смысле – «пайплайна» (выбирается с наименьшей ошибкой, см. дальше):

- выбор модели алгоритмов
- выбор значения гиперпараметров
- выбор признаков
- выбор способа предобработки данных

Модификация основной идеи для выбора и тестирования

Обучающая выборка – Training Set

обучение модели (настройка её параметров)

Валидационная выборка – Validation Set

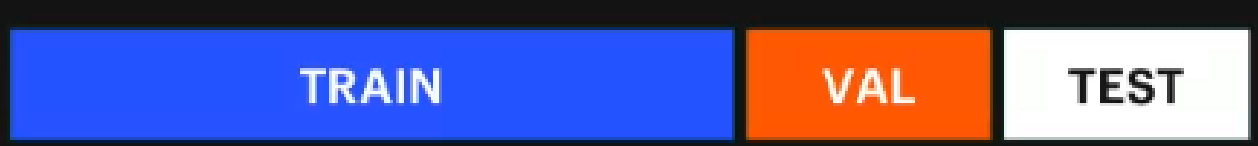
выбор пайплайна (модели / гиперпараметров / признаков)

иногда: локальный контроль

Тестовая выборка – Test Set

оценка качества алгоритма

иногда: итоговая оценка



Но можно проводить и более сложные схемы

k-Fold CV (валидация) + hold out (итоговый тест)

В редких случаях доверяют результатам валидации

```python
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import ShuffleSplit
from sklearn.linear_model import LogisticRegression

logreg = LogisticRegression()
cv = ShuffleSplit(n_splits=3, test_size=0.1,
                  train_size=None, random_state=None)
cross_val_score(logreg, X, y, cv=cv)
```

У этих функций много параметров...
Они (функции) «понимают» друг друга

Если не указываем скорер – используется встроенный (в модель)

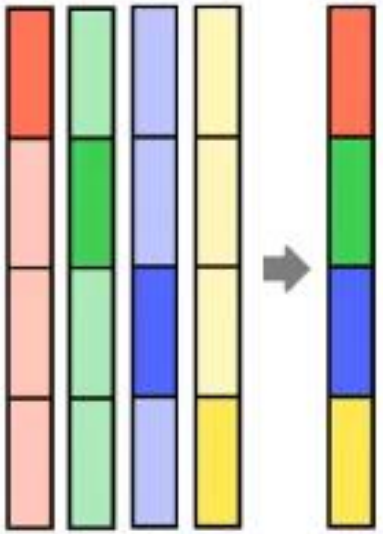
Есть аналогичная функция, которая сохраняет ещё время обучения и работы:

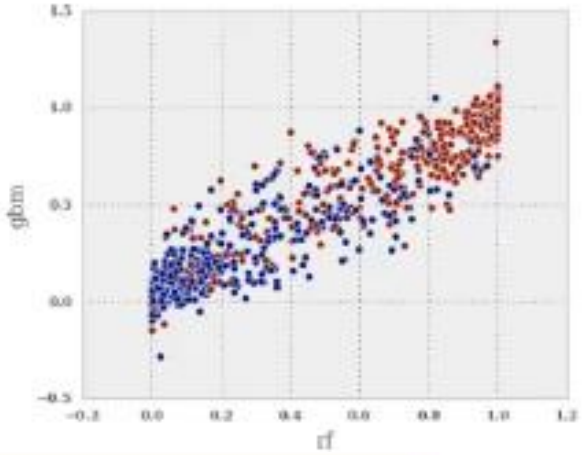
`sklearn.model_selection.cross_validate`

Перестановочный тест для оценки неслучайности результата:

`sklearn.model_selection.permutation_test_score`

На какие объектах обучение хуже всего проходит?





Ответы алгоритма с помощью выбранного контроля: минутка кода

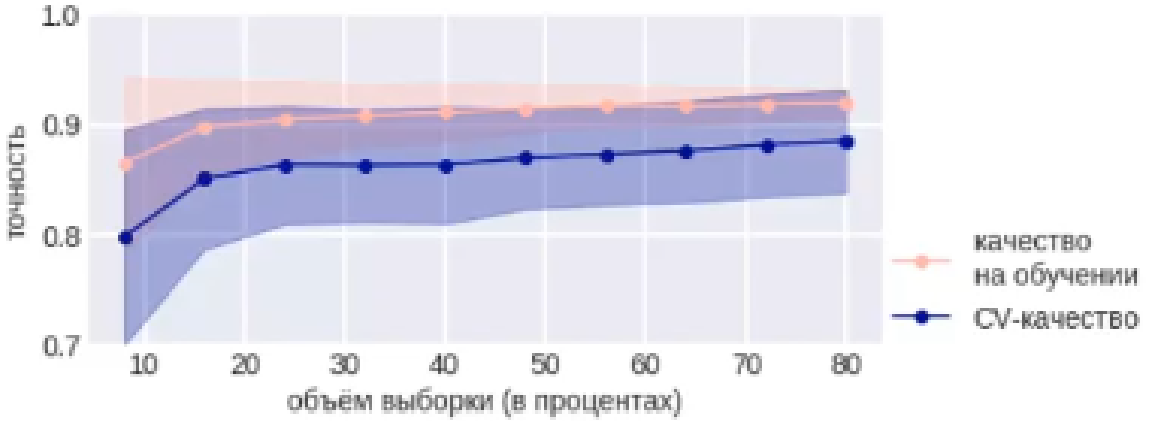
```
from sklearn.model_selection import cross_val_predict
from sklearn.model_selection import KFold
cv = KFold(n_splits=10, shuffle=True, random_state=1)

# оценим rf на CV
a_rf = cross_val_predict(rf, X, y, cv=cv)
# оценим gbm на CV
a_gbm = cross_val_predict(gbm, X, y, cv=cv)

plt.scatter(a_rf, a_gbm, c=y)
plt.xlabel('rf')
plt.ylabel('gbm')
```

Переобучение и избыточность данных

- Делим данные на обучение и контроль (м.б. очень много раз)
- Обучаемся на k% от обучающей выборки для разных k
- Строим графики ошибок/качества на train/CV от k




```
model_selection.learning_curve

    train_sizes, train_scores, test_scores =
learning_curve(estimator, X, y, cv=cv,

n_jobs=n_jobs, train_sizes=train_sizes)
```

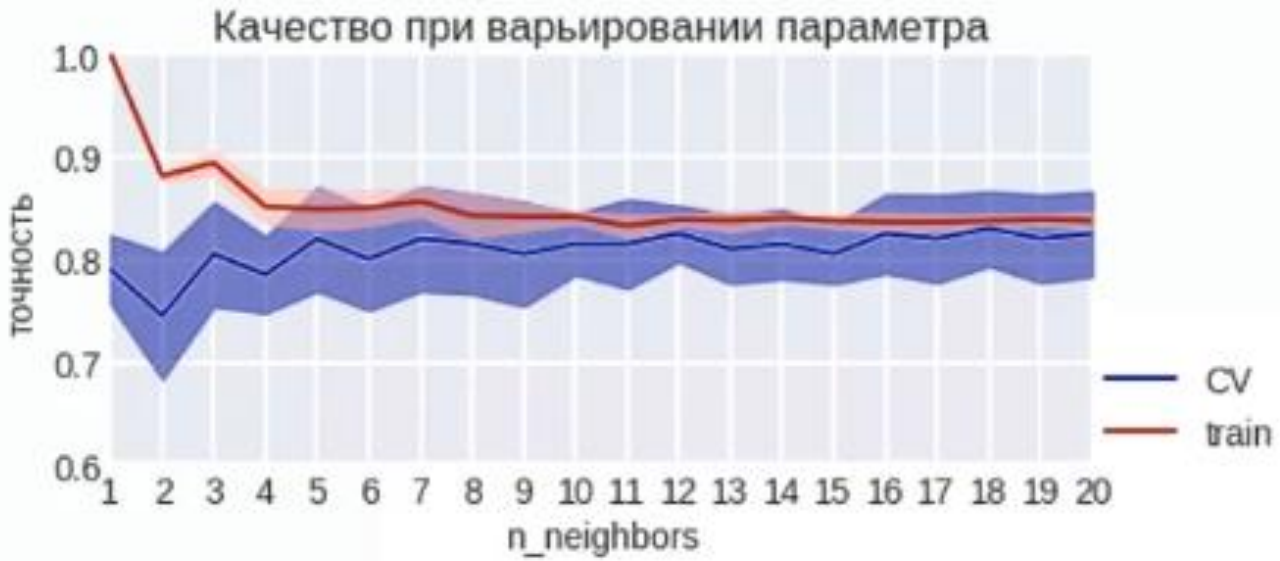
Есть зазор между обучением и CV

Тонкость: 100% – вся выборка, но здесь `test_size=0.2`

Качество от параметров

Валидационная кривая (Validation Curve) показывает зависимость качества / ошибки при выбранной схеме контроля от значений гиперпараметров

sklearn.model_selection.validation_curve



Перебор значений гиперпараметров

Делим данные на обучение и контроль (м.б. очень много раз)
При разных значениях параметров обучаемся и проверяем качество

```
from sklearn.model_selection import GridSearchCV
parameters = {'metric':('euclidean', 'manhattan', 'chebyshev'),
              'n_neighbors':[1, 3, 5, 7, 9, 11] , scoring='roc_auc'}
clf = GridSearchCV(estimator, parameters, cv=5)
clf.fit(X, y)
clf.cv_results_['mean_test_score']
```

	k=1	*k*=3	*k*=5	*k*=7	*k*=9	*k*=11
euclidean	76.0	77.0	79.0	78.5	80.5	82.5
manhattan	74.0	74.0	79.0	79.5	80.5	81.0
chebyshev	76.5	78.5	80.0	80.0	81.0	81.5

Есть также случайный поиск

`model_selection.RandomizedSearchCV`

Тут есть «число итераций», можно передавать распределения параметров

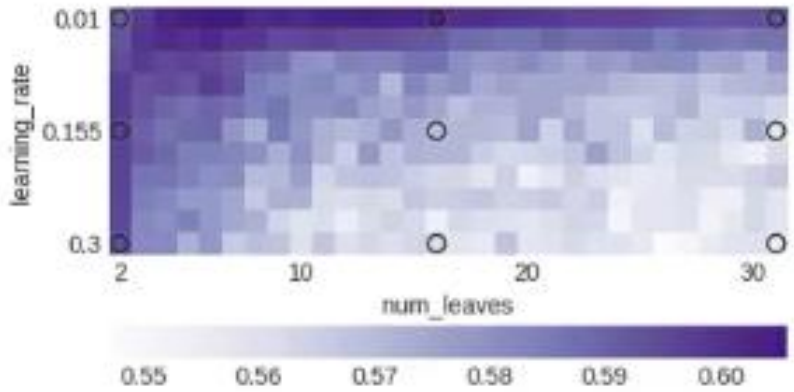
Перебор параметров

Случайный поиск считают предпочтительным

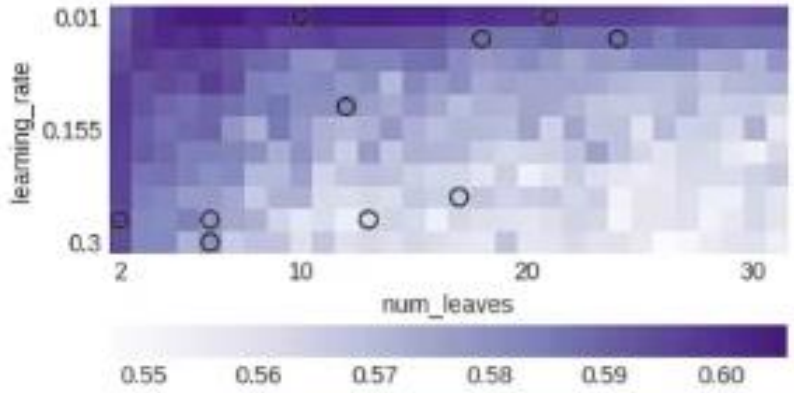
```
import lightgbm as lgb
model = lgb.LGBMClassifier(n_estimators=100, subsample=0.75,
colsample_bytree=0.75)

from sklearn.model_selection import GridSearchCV
parameters = {'num_leaves': np.arange(2, 32),
              'learning_rate':np.linspace(0.01, 0.3, 11)}
clf = GridSearchCV(model, parameters, cv=5,
scoring='roc_auc')
clf.fit(X, y)
```

GridSearchCV



RandomizedSearchCV



Абстракции sklearn | Пайплайны

```
model.get_params()
{'boosting_type': 'gbdt',
 'class_weight': None,
 'colsample_bytree': 0.75,
 'importance_type': 'split',
 'learning_rate': 0.1,
 'max_depth': -1,
 'min_child_samples': 20,
 'min_child_weight': 0.001,
 'min_split_gain': 0.0,
 'n_estimators': 100,
 'n_jobs': -1,
 'num_leaves': 31,
 'objective': None,
 'random_state': None,
 'reg_alpha': 0.0,
 'reg_lambda': 0.0,
 'silent': True,
 'subsample': 0.75,
 'subsample_for_bin': 200000,
 'subsample_freq': 0}
```

- Не забывайте указать метрику качества (а лучше несколько) **score**
- Распараллеливание **n_jobs=-1**
- Можно сделать вычисления устойчивым к ошибкам **error_score=0**
- Оптимизировать целый пайплайн!

Спасибо за внимание!

Запорожцев Иван Федорович
zaporozhtsev.if.work@gmail.com