Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования «Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»

(МГТУ им. Н.Э. Баумана)

Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

«Подготовка обучающей и тестовой выборки, кросс-валидация и подбор гиперпараметров на примере метода ближайших соседей»
по курсу «Технологии машинного обучения»
Лабораторная работа №4

Выполнил:
студент группы ИУ5 – 62Б
Гринин О.Е.
подпись,
дата

Проверил:

преподаватель кафедры ИУ5
Гапанюк Ю.Е.
подпись, дата

2020 г.

Подготовка обучающей и тестовой выборки, кросс-валидация и подбор гиперпараметров на примере метода ближайших соседей¶

Цель рабораторной работы¶

Изучение сложных способов подговки выборки и подбора гиперпараметров на примере метода ближайших соседей.

Задание¶

1. Выбрать набор данных (датасет) для решения задачи классификации или регрессии.
2. С использованией метода train_test_split разделите выборку на обучающую и тестовую.
3. Обучите модель ближайших соседей для произвольно заданного гиперпараметра K. Оцените качество модели с помощью подходящих для задачи метрик.
4. Постройте модель и оцените качество модели с использованием кросс-валидации.
5. Произведите подбор гиперпараметра K с использованием GridSearchCV и кросс-валидации.

In [18]:

```
import numpy as np
import pandas as pd
from typing import Dict, Tuple
from scipy import stats
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score, cross_validate
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
from sklearn.neighbors import KNeighborsRegressor, KNeighborsClassifier
from sklearn.metrics import accuracy_score, balanced_accuracy_score
from sklearn.metrics import precision_score, recall_score, f1_score, classification_re
port
from sklearn.metrics import confusion_matrix
from sklearn.metrics import mean_absolute_error, mean_squared_error, mean_squared_log_
error, median_absolute_error, r2_score
from sklearn.metrics import roc_curve, roc_auc_score
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set(style="ticks")
```

Подготовка данных и построение базовых моделей для оценки качества¶

In [20]:

```
breast_cancer = load_breast_cancer()
```

In [21]:

```
# Наименование признаков

breast_cancer.feature_names
```

Out[21]:

```
array(['mean radius', 'mean texture', 'mean perimeter', 'mean area',
       'mean smoothness', 'mean compactness', 'mean concavity',
       'mean concave points', 'mean symmetry', 'mean fractal dimension',
       'radius error', 'texture error', 'perimeter error', 'area error',
       'smoothness error', 'compactness error', 'concavity error',
       'concave points error', 'symmetry error',
       'fractal dimension error', 'worst radius', 'worst texture',
       'worst perimeter', 'worst area', 'worst smoothness',
       'worst compactness', 'worst concavity', 'worst concave points',
       'worst symmetry', 'worst fractal dimension'], dtype='<U23')
```

In [22]:

```
type(breast_cancer.data)
```

Out[22]:

```
numpy.ndarray
```

In [23]:

```
data = pd.DataFrame(data = np.c_[breast_cancer['data'], breast_cancer['target']], colu
mns = breast_cancer['feature_names'].tolist() + ['target'])
```

In [24]:

```
data.head()
```

Out[24]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst texture	worst perimeter	worst area	worst smoothness	worst compactness	worst concavity	worst concave points	worst symmetry	worst fractal dimension	target
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	0.07871	...	17.33	184.60	2019.0	0.1622	0.6656	0.7119	0.2654	0.4601	0.11890	0.0
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667	...	23.41	158.80	1956.0	0.1238	0.1866	0.2416	0.1860	0.2750	0.08902	0.0
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	0.05999	...	25.53	152.50	1709.0	0.1444	0.4245	0.4504	0.2430	0.3613	0.08758	0.0
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	0.09744	...	26.50	98.87	567.7	0.2098	0.8663	0.6869	0.2575	0.6638	0.17300	0.0
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	0.05883	...	16.67	152.20	1575.0	0.1374	0.2050	0.4000	0.1625	0.2364	0.07678	0.0

5 rows × 31 columns

Разделение выборки на обучающую и тестовую¶

In [25]:

```
X_train, X_test, Y_train, Y_test = train_test_split(breast_cancer.data, breast_cancer.
target, test_size = 0.3, random_state = 1)
```

In [26]:

```
# Размер обучающей выборки

X_train.shape, Y_train.shape
```

Out[26]:

```
((398, 30), (398,))
```

In [27]:

```
# Размер тестовой выборки

X_test.shape, Y_test.shape
```

Out[27]:

```
((171, 30), (171,))
```

Обучение модели ближайших соседей для заданного гиперпараметра K¶

In [29]:

```
# 3 ближайших соседа

# Метрика accuracy вычисляет процент правильно определенных классов

cl1_1 = KNeighborsClassifier(n_neighbors=3)

cl1_1.fit(X_train, Y_train)

target1_0 = cl1_1.predict(X_train)

target1_1 = cl1_1.predict(X_test)

accuracy_score(Y_train, target1_0), accuracy_score(Y_test, target1_1)
```

Out[29]:

```
(0.9472361809045227, 0.9239766081871345)
```

In [30]:

```
# 8 ближайших соседа

# Метрика accuracy вычисляет процент правильно определенных классов

cl1_2 = KNeighborsClassifier(n_neighbors=8)

cl1_2.fit(X_train, Y_train)

target2_0 = cl1_2.predict(X_train)

target2_1 = cl1_2.predict(X_test)

accuracy_score(Y_train, target2_0), accuracy_score(Y_test, target2_1)
```

Out[30]:

```
(0.9321608040201005, 0.9415204678362573)
```

Построение модели с использованием кросс-валидации

In [31]:

```
scores = cross_val_score(KNeighborsClassifier(n_neighbors=3), breast_cancer.data, brea
st_cancer.target, cv=3)
```

In [32]:

```
# Значение метрики accuracy для 3 фолдов
scores
```

Out[32]:

```
array([0.89473684, 0.95263158, 0.91534392])
```

In [33]:

```
# Усредненное значение метрики accuracy для 3 фолдов
np.mean(scores)
```

Out[33]:

```
0.9209041121321823
```

In [34]:

```
# использование метрики precision
scores = cross_val_score(KNeighborsClassifier(n_neighbors=3),
                         breast_cancer.data, breast_cancer.target, cv=3,
                         scoring='precision_weighted')
scores, np.mean(scores)
```

Out[34]:

```
(array([0.89654273, 0.9533197 , 0.91504168]), 0.9216347037536606)
```

In [35]:

```
# функция cross_validate позволяет использовать для оценки несколько метрик
scoring = {'precision': 'precision_weighted',
           'jaccard': 'jaccard_weighted',
           'f1': 'f1_weighted'}

scores = cross_validate(KNeighborsClassifier(n_neighbors=3),
                         breast_cancer.data, breast_cancer.target, scoring=scoring,
                         cv=3, return_train_score=True)
scores
```

Out[35]:

```
{'fit_time': array([0.00199342, 0.00099707, 0.00099683]),
 'score_time': array([0.05684829, 0.03191376, 0.07180929]),
 'test_precision': array([0.89654273, 0.9533197 , 0.91504168]),
 'train_precision': array([0.9585625 , 0.95775754, 0.9533197 ]),
 'test_jaccard': array([0.80818208, 0.9091925 , 0.84433622]),
 'train_jaccard': array([0.91863329, 0.91899267, 0.9091925 ]),
 'test_f1': array([0.89287184, 0.95225452, 0.9150832 ]),
 'train_f1': array([0.95744193, 0.95765583, 0.95225452])}
```

Подбор гиперпараметра K с использованием GridSearchCV и кросс-валидации¶

In [36]:

```
n_range = np.array(range(5,55,5))

tuned_parameters = [{'n_neighbors': n_range}]

tuned_parameters
```

Out[36]:

```
[{'n_neighbors': array([ 5, 10, 15, 20, 25, 30, 35, 40, 45, 50])}]
```

In [37]:

```
%%time

clf_gs = GridSearchCV(KNeighborsClassifier(), tuned_parameters, cv=5, scoring='accurac
y')

clf_gs.fit(X_train, Y_train)
```

```
Wall time: 439 ms
C:\Users\miair\Anaconda3\lib\site-packages\sklearn\model_selection\_search.py:814: Dep
recationWarning: The default of the `iid` parameter will change from True to False in
version 0.22 and will be removed in 0.24. This will change numeric results when test-s
et sizes are unequal.
  DeprecationWarning)
```

Out[37]:

```
GridSearchCV(cv=5, error_score='raise-deprecating',
             estimator=KNeighborsClassifier(algorithm='auto', leaf_size=30,
                                            metric='minkowski',
                                            metric_params=None, n_jobs=None,
                                            n_neighbors=5, p=2,
                                            weights='uniform'),
             iid='warn', n_jobs=None,
             param_grid=[{'n_neighbors': array([ 5, 10, 15, 20, 25, 30, 35, 40, 45, 50
])}],
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring='accuracy', verbose=0)
```

In [38]:

```
clf_gs.cv_results_
```

Out[38]:

```
{'mean_fit_time': array([0.00219526, 0.00119758, 0.00119772, 0.00099769, 0.00099721,
        0.00099802, 0.00099797, 0.00059857, 0.00099797, 0.00099716]),
 'std_fit_time': array([2.39400872e-03, 3.99112787e-04, 3.98803025e-04, 4.42200589e-07
,
        4.15696997e-07, 7.07263802e-07, 9.70220087e-07, 4.88733398e-04,
        6.31052615e-04, 9.36836372e-07]),
 'mean_score_time': array([0.00558429, 0.00638208, 0.00438714, 0.00558529, 0.00558486,
        0.00737967, 0.00498548, 0.00478649, 0.00658183, 0.01017289]),
 'std_score_time': array([0.00135295, 0.00119687, 0.00048881, 0.0019541 , 0.00135255,
        0.00325273, 0.00063045, 0.00039911, 0.00195491, 0.00116335]),
 'param_n_neighbors': masked_array(data=[5, 10, 15, 20, 25, 30, 35, 40, 45, 50],
              mask=[False, False, False, False, False, False, False, False,
                    False, False],
        fill_value='?',
             dtype=object),
 'params': [{'n_neighbors': 5},
  {'n_neighbors': 10},
  {'n_neighbors': 15},
  {'n_neighbors': 20},
  {'n_neighbors': 25},
  {'n_neighbors': 30},
  {'n_neighbors': 35},
```

```
  {'n_neighbors': 40},
  {'n_neighbors': 45},
  {'n_neighbors': 50}],
 'split0_test_score': array([0.8625, 0.925 , 0.9   , 0.9375, 0.9375, 0.9   , 0.9   , 0
.8875,
        0.8875, 0.9   ]),
 'split1_test_score': array([0.875 , 0.8875, 0.9125, 0.9   , 0.9125, 0.9125, 0.925 , 0
.9125,
        0.9125, 0.9125]),
 'split2_test_score': array([0.9125, 0.925 , 0.9625, 0.9625, 0.9625, 0.9625, 0.9625, 0
.9625,
        0.9625, 0.9625]),
 'split3_test_score': array([0.9625, 0.9625, 0.95  , 0.9375, 0.9375, 0.9375, 0.95  , 0
.95  ,
        0.9375, 0.9375]),
 'split4_test_score': array([0.91025641, 0.91025641, 0.88461538, 0.8974359 , 0.8717948
7,
        0.87179487, 0.87179487, 0.85897436, 0.85897436, 0.87179487]),
 'mean_test_score': array([0.90452261, 0.92211055, 0.92211055, 0.92713568, 0.92462312,
        0.91708543, 0.92211055, 0.91457286, 0.9120603 , 0.91708543]),
 'std_test_score': array([0.03500052, 0.02448629, 0.02960751, 0.02480818, 0.0305208 ,
        0.03103614, 0.03286864, 0.03831434, 0.03626371, 0.03103614]),
 'rank_test_score': array([10,  3,  3,  1,  2,  6,  3,  8,  9,  6])}
```

In [39]:

```
# Лучшая модель
clf_gs.best_estimator_
```

Out[39]:

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=20, p=2,
                     weights='uniform')
```

In [40]:

```
# Лучшее значение метрики
clf_gs.best_score_
```

Out[40]:

```
0.9271356783919598
```

In [41]:

```
# Лучшее значение параметров
clf_gs.best_params_
```

Out[41]:

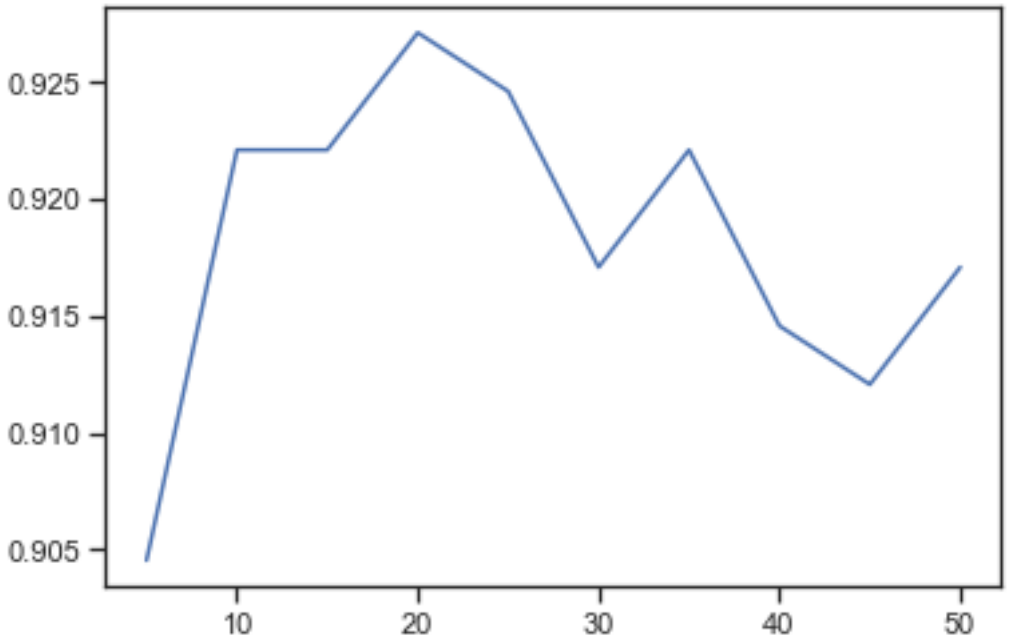
```
{'n_neighbors': 20}
```

In [42]:

```
# Изменение качества на тестовой выборке в зависимости от К-соседей
plt.plot(n_range, clf_gs.cv_results_['mean_test_score'])
```

Out[42]:

```
[<matplotlib.lines.Line2D at 0x20fd66888c8>]
```



Оптимальный гиперпараметр K = 20