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 Системы обработки информации и управления (ИУ5)

ОТЧЁТ
по лабораторной работе №4

По курсу: «Технологии машинного обучения»

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

Выполнил: студент группы ИУ5-64Б _____ Евсюков Н.М.
(Подпись, дата) (Ф.И.О.)

Проверил: _____ Гапанюк Ю.Е.
(Подпись, дата) (Ф.И.О.)

2020 г.

1. Цель работы

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

2. Описание задания

- Выбрать набор данных (датасет) для решения задачи классификации или регрессии
- С использованием метода train_test_split разделить выборку на обучающую и тестовую
- Обучить модель ближайших соседей для произвольно заданного гиперпараметра K
- Оценить качество модели с помощью подходящих для задачи метрик
- Построить модель и оценить качество модели с использованием кросс-валидации
- Произвести подбор гиперпараметра K с использованием GridSearchCV и кросс-валидации
- Сформировать отчет и разместить его на своем репозитории GitHub

In [91]:
```
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
plot_confusion_matrix
from sklearn.metrics import precision_score, recall_score, f1_score,
cla ssification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import mean_absolute_error, mean_squared_error,
mea n_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
")
```

3. Подготовка данных и построение базовых моделей для оценки качества

In [100]:
```
breast_cancer = load_breast_cancer()
```

In [97]:
```
#Наименования признаков
breast_cancer.feature_names
```

Out[97]:
```
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 [103]:
```
type(breast_cancer.data)
```

Out[103]:

numpy.ndarray

In [104]:
```
data= pd.DataFrame(data= np.c_[breast_cancer['data'],
breast_cancer['ta rget']],
                   columns= breast_cancer['feature_names'].tolist() +
['target'])
```

In [105]:
```
data.head()
```

Out[105]:

concave s

5 rows × 31 columns

4. Разделение выборки на обучающую и тестовую

In [106]:
```
X_train, X_test, Y_train, Y_test = train_test_split(
    breast_cancer.data, breast_cancer.target, test_size=0.3,
    random_stat
e=1)
```

In [107]:
```
# Размер обучающей выборки
X_train.shape, Y_train.shape
```

Out[107]: ((398, 30), (398,))

In [108]:
```
# Размер тестовой выборки
X_test.shape, Y_test.shape
```

Out[108]: ((171, 30), (171,))

5. Обучение модели ближайших соседей для заданного гиперпараметра K

In [109]:
```
# 3 ближайших соседа
# Метрика accuracy вычисляет процент (долю в диапазоне от 0 до 1) правильно определенных
класс ов
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[109]: (0.9472361809045227, 0.9239766081871345)

In [110]:
```
# 8 ближайших соседей
# Метрика accuracy вычисляет процент (долю в диапазоне от 0 до 1) правильно определенных
класс ов
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[110]: (0.9321608040201005, 0.9415204678362573)

6. Построение модели с использованием кросс-валидации

In [111]:
```
scores = cross_val_score(KNeighborsClassifier(n_neighbors=3),
                         breast_cancer.data, breast_cancer.target,
                         cv=3)
```

In [112]:
```
# Значение метрики accuracy для 3 фолдов
scores
```

Out[112]: array([0.89473684, 0.95263158, 0.91534392])

In [113]:
```
# Усредненное значение метрики accuracy для 3 фолдов
np.mean(scores)
```

Out[113]: 0.9209041121321823

In [114]:
```
# 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[114]:
```
(array([0.89654273, 0.9533197 , 0.91504168]), 0.9216347037536606)
```

In [116]:
```
# 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, scorin

g=scoring,                                      cv=3, return_train_score=True)

scores
```

Out[116]:
```
{'fit_time': array([0., 0., 0.]),
 'score_time': array([0.03152204, 0.01564574, 0.03126574]),
 '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])}
```

7. Подбор гиперпараметра K с использованием GridSearchCV и кросс-валидации

In [118]:
```
n_range = np.array(range(5,55,5))
tuned_parameters = [{'n_neighbors':
n_range}] tuned_parameters
```

Out[118]:
```
[{'n_neighbors': array([ 5, 10, 15, 20, 25, 30, 35, 40, 45, 50])}]
```

In [119]:
```
%%time
clf_gs = GridSearchCV(KNeighborsClassifier(), tuned_parameters, cv=5,
sc oring='accuracy')
clf_gs.fit(X_train, Y_train)
```

```
Wall time: 686 ms
```

Out[119]:
```
GridSearchCV(cv=5, error_score=nan,
             estimator=KNeighborsClassifier(algorithm='auto', leaf_size=



30,

                                                iid='deprecated',
                                                n_jobs=None,

one,
```

```
m e                    n k o w s k i ' ,          s=5, p=2,
t r                    metric_params=N            weights='unifor
i c                    one, n_jobs=N              m '),
= '
m i                    n_neighbor

                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=Fal
       se,
                scoring='accuracy', verbose=0)
```

In [120]:
```
clf_gs.cv_results_
```

Out[120]:
```
{'mean_fit_time': array([0.00231314, 0.00184054, 0.00312042, 0.01037116,
        0.00315456,
               0.0062571 , 0.0031249 , 0.00624986, 0.        , 0.        ]),
        'std_fit_time': array([0.00079557, 0.00119532, 0.00624084, 0.00866432,
       0.00630913,
               0.00766336, 0.00624981, 0.00765448, 0.        , 0.        ]),
        'mean_score_time': array([0.01362453, 0.00723748, 0.01249657, 0.0156426
       9, 0.01246901,
               0.0062501 , 0.00624762, 0.00312676, 0.00937333, 0.00625267]),
        'std_score_time': array([0.01524153, 0.00502408, 0.00624831, 0.0098877
       , 0.01167582,
               0.00765478, 0.00765174, 0.00625353, 0.00765331, 0.00765793]),
        'param_n_neighbors': masked_array(data=[5, 10, 15, 20, 25, 30, 35, 40,
       45, 50],

                                          se,                mask=[False, False, False,
                                                             False, False, False, False,
                                                             Fal

                                                                   F

                                                a
                                                l
                                                s
                                                e
                                                ,
                                                F
                                                a
                                                l
                                                s
                                                e
                                                ]
                                                ,
                                                f
                                                i
                                                l
                                                l
                                                _
                                                v
                                                a
                                                l
                                                u
                                                e
                                                =
                                                '
```

```
    '                                                    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, 0.9
            125, 0.925 , 0.9125,
                    0.9125, 0.9125]),
             'split2_test_score': array([0.9125, 0.925 , 0.9625, 0.9625, 0.9625, 0.9
            625, 0.9625, 0.9625,
                    0.9625, 0.9625]),
             'split3_test_score': array([0.96202532, 0.96202532, 0.94936709, 0.93670
            886, 0.93670886,
                    0.93670886, 0.94936709, 0.94936709, 0.93670886, 0.93670886]),
            'split4_test_score': array([0.91139241, 0.91139241, 0.88607595, 0.89873
                 418, 0.87341772,
                 0.87341772, 0.87341772, 0.86075949, 0.86075949, 0.87341772]),
            'mean_test_score': array([0.90468354, 0.92218354, 0.92208861, 0.9270886
                 1, 0.92452532,
                 0.91702532, 0.92205696, 0.91452532, 0.91199367, 0.91702532]),
            'std_test_score': array([0.0347987 , 0.02417697, 0.02916832, 0.02446499
            , 0.03005146,
            0.03055274, 0.03238033, 0.03779088, 0.03574036, 0.03055274]),
                                 'rank_test_score': array([10,   3,  4,    1,    2,
6,                               5,  8,  9,  6], dtyp e=int32)}
```

In [121]:
```
# Лучшая модель
clf_gs.best_estimator_
```

Out[121]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=20, p=
          2,
                     weights='uniform')
```

In [122]:
```
# Лучшее значение метрики
clf_gs.best_score_
```

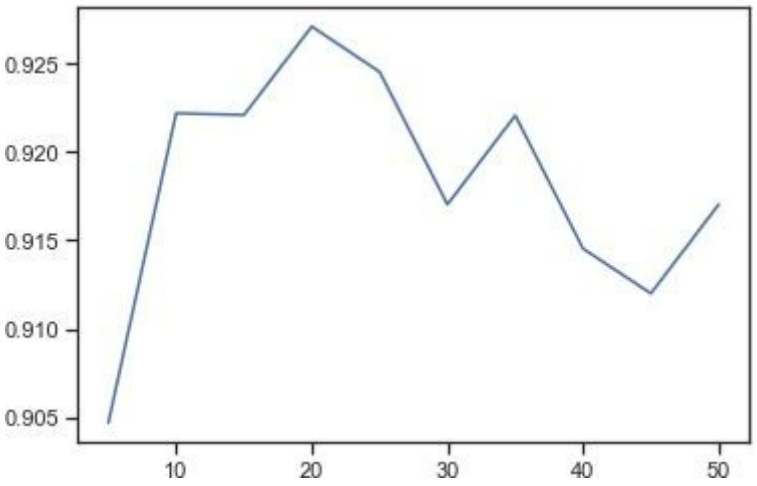
Out[122]:
```
0.9270886075949367
```

In [123]:
```
# Лучшее значение параметров
clf_gs.best_params_
```

Out[123]:
```
{'n_neighbors': 20}
```

In [124]:
```
# Изменение качества на тестовой выборке в зависимости от K-соседей
plt.plot(n_range, clf_gs.cv_results_['mean_test_score'])
```

Out[124]: [<matplotlib.lines.Line2D at 0x9c86d50>]



	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809

Оптимальный гиперпараметр K = 20