*Государственное образовательное учреждение высшего профессионального образования*

|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | ***«Московский государственный технический университет  имени Н.Э. Баумана»***  ***(МГТУ им. Н.Э. Баумана)*** |

**«Лабораторная работа №4»**

«Технологии машинного обучения»

**ИСПОЛНИТЕЛЬ:**

Студены группы РТ5-61

Курьянов А.И.

**ПРЕПОДАВАТЕЛЬ:**

Гапанюк Ю.Е.

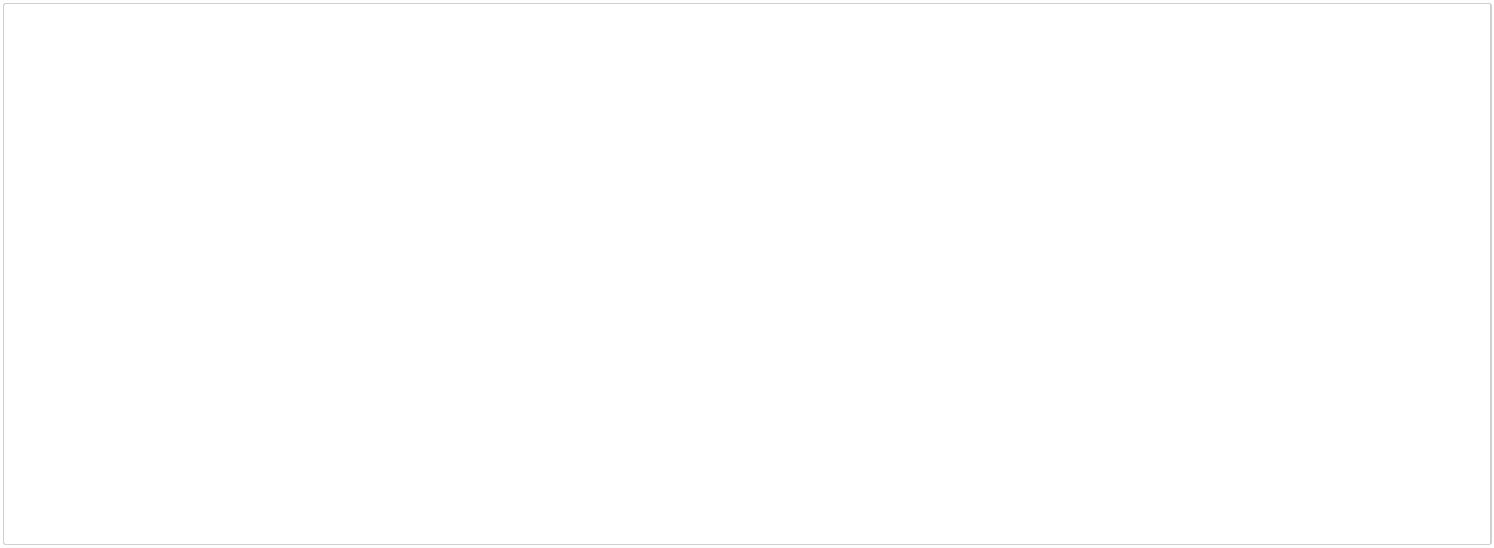
"\_\_"\_\_\_\_\_\_\_\_\_\_\_2020 г.

2020 г.

**Задание**

1. Выберите набор данных (датасет) для решения задачи классификации или регрессии.
2. С использованием 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 [1]:



**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, classification\_report **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**

In [2]:



breast\_cancer = load\_breast\_cancer()

In [3]:



* *Наименования признаков* breast\_cancer.feature\_names

Out[3]:

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 [4]:



type(breast\_cancer.data)

Out[4]:

numpy.ndarray

In [5]:



data = pd.DataFrame(data= np.c\_[breast\_cancer['data'], breast\_cancer['target']], columns= breast\_cancer['feature\_names'].tolist() + ['target'])

In [6]:



data.head()

Out[6]:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **mean** | **mean** | **mean** | **mean** | **mean** | **mean** | **mean** | **mean** | **mean** | **mean** | **worst** | **worst** | **wo** |
|  | **concave** | **fractal ...** |
|  | **radius** | **texture** | **perimeter** | **area** | **smoothness** | **compactness** | **concavity** | **points** | **symmetry** | **dimension** | **texture** | **perimeter** | **ar** |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **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 | 201 |
| **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 | 195 |
| **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 | 170 |
| **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 | 56 |
| **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 | 157 |

5 rows × 31 columns



**Разделение выборки методом train\_test\_split**

In [106]:



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 [107]:



* *Размер обучающей выборки*

X\_train.shape, Y\_train.shape

Out[107]:

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

In [108]:



* *Размер тестовой выборки*

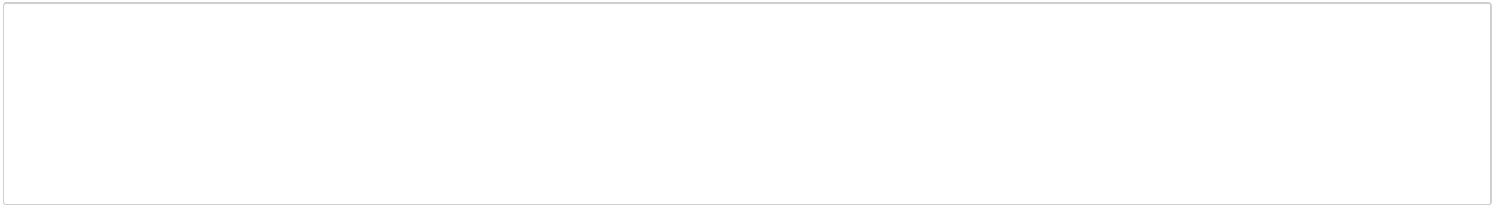
X\_test.shape, Y\_test.shape

Out[108]:

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

**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 классов* cl1\_1 = KNeighborsClassifier(n\_neighbors=3)

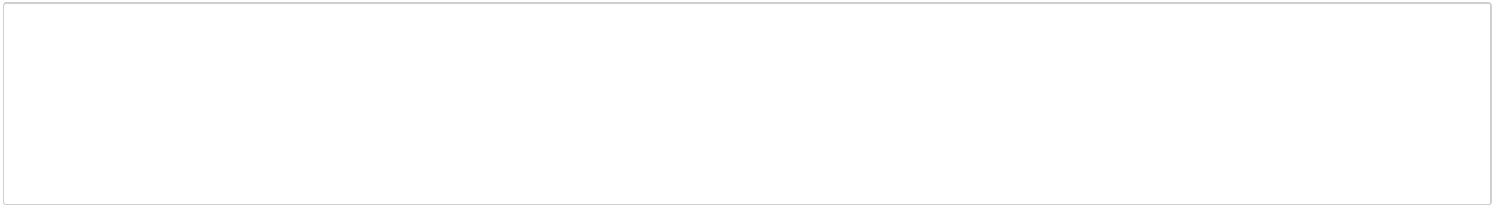
cl1\_1.fit(X\_train, Y\_train) target1\_0 = cl1\_1.predict(X\_train) target1\_1 = cl1\_1.predict(X\_test)

accuracy\_score(Y\_train, target1\_0), accuracy\_score(Y\_test, target1\_1)

Out[109]:

(0.9472361809045227, 0.9239766081871345)

In [110]:



* *8 ближайших соседей*
* *Метрика accuracy вычисляет процент (долю в диапазоне от 0 до 1) 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[110]:

(0.9321608040201005, 0.9415204678362573)

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]:

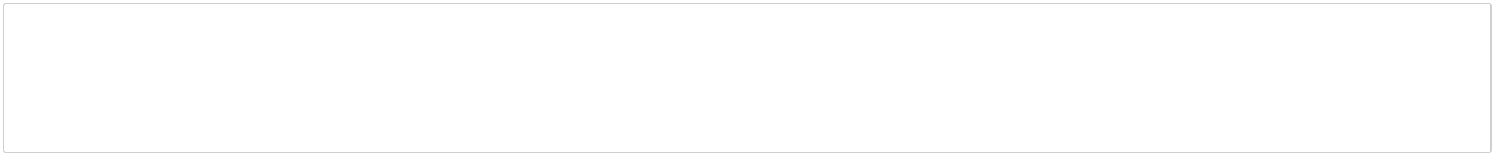


* *Усредненное значение метрики accuracy для 3 фолдов* np.mean(scores)

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,

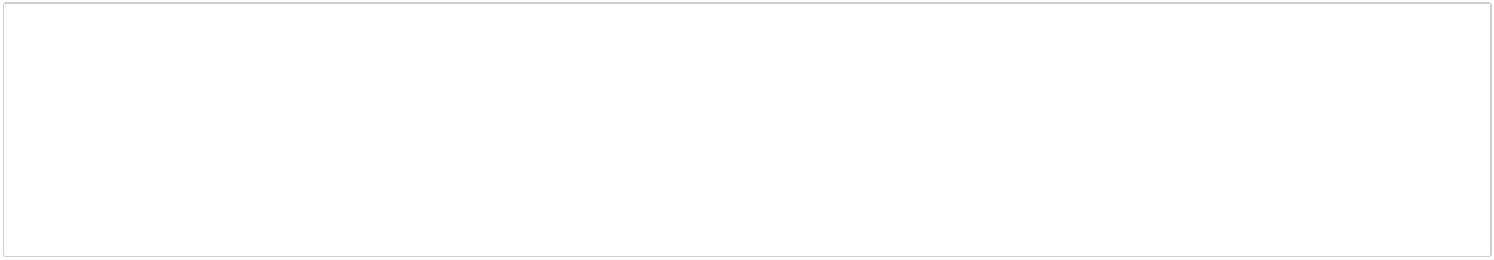
scoring='precision\_weighted')

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, scoring=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])}

**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, scoring='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,

metric='minkowski',

metric\_params=None, n\_jobs=None,

n\_neighbors=5, p=2,

weights='uniform'),

iid='deprecated', 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 [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, 0.01564269, 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], | | | | | | | |  |
| 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([0.96202532, 0.96202532, 0.94936709, 0.93670886, 0.93670886, 0.93670886, 0.94936709, 0.94936709, 0.93670886, 0.93670886]),

'split4\_test\_score': array([0.91139241, 0.91139241, 0.88607595, 0.89873418, 0.87341772, 0.87341772, 0.87341772, 0.86075949, 0.86075949, 0.87341772]),

'mean\_test\_score': array([0.90468354, 0.92218354, 0.92208861, 0.92708861, 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], dtype=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}

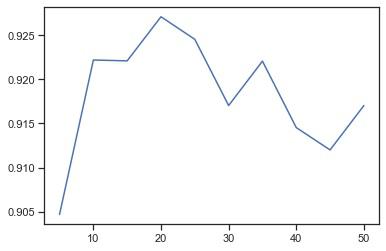
In [124]:



* *Изменение качества на тестовой выборке в зависимости от К-соседей* plt.plot(n\_range, clf\_gs.cv\_results\_['mean\_test\_score'])

Out[124]:

[<matplotlib.lines.Line2D at 0x9c86d50>]



Oптимальный гиперпараметр K = 20

In [ ]:

