Защищено:
Гапанюк Ю.Е.

"__"_____2023 г.

Демонстрация:
Гапанюк Ю.Е.

"__"_____2023 г.

Отчет по лабораторной работе № 3 по курсу Технологии машинного обучения ГУИМЦ

Тема работы: " Подготовка обучающей и тестовой выборки, кросс-валидация и подбор гиперпараметров на примере метода ближайших соседей."

8
(количество листов)
Вариант № **2**

ИСПОЛНИТЕЛЬ:

студент группы ИУ5Ц-84Б _____
(подпись)

Распашнов А.А.

"__"_____2023 г.

Москва, МГТУ - 2023

Цель лабораторной работы

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

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. Произведите подбор гиперпараметра K с использованием GridSearchCV и/или RandomizedSearchCV и кросс-валидации, оцените качество оптимальной модели. Желательно использование нескольких стратегий кросс-валидации.
5. Сравните метрики качества исходной и оптимальной моделей.

Ход выполнения работы

Текстовое описание набора данных

В качестве набора данных используется dataset рейтингов университетов мира на основании трёх рейтингов. Датасет доступен по адресу: https://www.kaggle.com/c/titanic/data?select=train.csv

Из набора данных будет рассматриваться только файл train.csv

ЛР3

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.impute import SimpleImputer
from sklearn.model_selection import train_test_split
```

In [2]:
```
import numpy as np
import pandas as pd
from typing import Dict, Tuple
from scipy import stats
from sklearn.datasets import load_iris, load_boston
from sklearn.model_selection import train_test_split
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_err
from sklearn.metrics import roc_curve, roc_auc_score
import seaborn as sns
from sklearn.model_selection import learning_curve
import matplotlib.pyplot as plt
%matplotlib inline
sns.set(style="ticks")
```

In [3]:
```
from sklearn.model_selection import KFold, RepeatedKFold, LeaveOneOut, LeavePOut, ShuffleSplit, StratifiedKFold
```

In [4]:
```
from sklearn.model_selection import train_test_split
```

In [5]:
```
# чтение обучающей выборки
data = pd.read_csv('train.csv')
```

In [6]:
```
# уберем непонятный для нас параметр, чтобы он не помешал в будущем
data.drop(['Name','Sex','Ticket','Embarked','Cabin','PassengerId', 'Parch'], axis = 1, inplace = True)
```

In [7]:
```
data
```

Out[7]:

	Survived	Pclass	Age	SibSp	Fare
0	0	3	22.0	1	7.2500
1	1	1	38.0	1	71.2833
2	1	3	26.0	0	7.9250
3	1	1	35.0	1	53.1000
4	0	3	35.0	0	8.0500
...	...	...	...	...	...
886	0	2	27.0	0	13.0000
887	1	1	19.0	0	30.0000
888	0	3	NaN	1	23.4500
889	1	1	26.0	0	30.0000
890	0	3	32.0	0	7.7500

891 rows × 5 columns

In [8]:
```
data = data.fillna(1)
data.head()
```

Out[8]:

	Survived	Pclass	Age	SibSp	Fare
0	0	3	22.0	1	7.2500
1	1	1	38.0	1	71.2833
2	1	3	26.0	0	7.9250

3	1	1	35.0	1	53.1000
4	0	3	35.0	0	8.0500

In [9]:
```
parts = np.split(data, [4,5], axis=1)
X = parts[0]
Y = parts[1]
print('Входные данные:\n\n', X.head(), '\n\nВыходные данные:\n\n', Y.head())
```

```
Входные данные:

    Survived  Pclass   Age  SibSp
0        0.0     3.0  22.0    1.0
1        1.0     1.0  38.0    1.0
2        1.0     3.0  26.0    0.0
3        1.0     1.0  35.0    1.0
4        0.0     3.0  35.0    0.0

Выходные данные:

       Fare
0   7.2500
1  71.2833
2   7.9250
3  53.1000
4   8.0500
```

Разделение выборки

In [10]:
```
X_train,  X_test,  Y_train,  Y_test = train_test_split(X,  Y, test_size=0.03)
```

In [11]:
```
print('Входные параметры обучающей выборки:\n\n',X_train.head(), \
      '\n\nВходные параметры тестовой выборки:\n\n', X_test.head(), \
      '\n\nВыходные параметры обучающей выборки:\n\n', Y_train.head(), \
      '\n\nВыходные параметры тестовой выборки:\n\n', Y_test.head())
```

```
Входные параметры обучающей выборки:

      Survived  Pclass   Age SibSp
861       0.0     2.0  21.0    1.0
557       0.0     1.0   1.0    0.0
356       1.0     1.0  22.0    0.0
178       0.0     2.0  30.0    0.0
72        0.0     2.0  21.0    0.0

Входные параметры тестовой выборки:

      Survived  Pclass   Age  SibSp
829       1.0     1.0  62.0    0.0
451       0.0     3.0   1.0    1.0
454       0.0     3.0   1.0    0.0
781       1.0     1.0  17.0    1.0
571       1.0     1.0  53.0    2.0

Выходные параметры обучающей выборки:

         Fare
861   11.500
557  227.525
356   55.000
178   13.000
72    73.500

Выходные параметры тестовой выборки:

         Fare
829  80.0000
451  19.9667
454   8.0500
781  57.0000
571  51.4792
```

In [14]:
```
# Проверим правильность разделения выборки на тестовую и обучающую. Посмотрим на размеры матриц.
print(X_train.shape)
print(X_test.shape)
print(Y_train.shape)
```

```
print(Y_test.shape)
```

```
(864, 4)
(27, 4)
(864, 1)
(27, 1)
```

Модель ближайших соседей для произвольного гиперпараметра K

In [15]:
```
from sklearn.neighbors import KNeighborsRegressor
```

In [16]:
```
# Решение задачи регрессии методом 2, 5 и 10 ближайших соседей
Regressor_2NN = KNeighborsRegressor(n_neighbors = 2)
Regressor_5NN = KNeighborsRegressor(n_neighbors = 5)
Regressor_10NN = KNeighborsRegressor(n_neighbors = 10)
print('Пример модели:\n\n', Regressor_10NN)
```

```
Пример модели:

 KNeighborsRegressor(n_neighbors=10)
```

In [17]:
```
Regressor_2NN.fit(X_train, Y_train)
Regressor_5NN.fit(X_train, Y_train)
Regressor_10NN.fit(X_train, Y_train)
target_2NN = Regressor_2NN.predict(X_test)
target_5NN = Regressor_5NN.predict(X_test)
target_10NN = Regressor_10NN.predict(X_test)
print('Пример предсказанных значений:\n\n', target_10NN[:5], '\n ...')
```

```
Пример предсказанных значений:

 [[14.73625]
 [ 8.02   ]
 [15.27916]
 [25.01041]
 [ 9.80999]]
 ...
```

Оценка качества регрессии (Метрики качества)

In [18]:
```
from sklearn.metrics import mean_absolute_error, mean_squared_error, median_absolute_error, r2_score , accuracy
```

In [19]:
```
# Оценка средней абсолютной ошибки
print('Средняя абсолютная ошибка для 2 ближайших соседей:',mean_absolute_error(Y_test,
target_2NN))
print('Средняя абсолютная ошибка для 5 ближайших соседей:',mean_absolute_error(Y_test,
target_5NN))
print('Средняя абсолютная ошибка для 10 ближайших соседей:',mean_absolute_error(Y_test,
target_10NN))
```

```
Средняя абсолютная ошибка для 2 ближайших соседей: 22.111566666666672
Средняя абсолютная ошибка для 5 ближайших соседей: 22.83663333333333
Средняя абсолютная ошибка для 10 ближайших соседей: 23.232667407407412
```

In [20]:
```
# Оценка средней квадратичной ошибки
print('Средняя квадратичная ошибка для 2 ближайших соседей:',mean_squared_error(Y_test,
target_2NN))
print('Средняя квадратичная ошибка для 5 ближайших соседей:',mean_squared_error(Y_test,
target_5NN))
print('Средняя квадратичная ошибка для 10 ближайших соседей:',mean_squared_error(Y_test
, target_10NN))
```

```
Средняя квадратичная ошибка для 2 ближайших соседей: 1662.2206168757414
Средняя квадратичная ошибка для 5 ближайших соседей: 2025.9715158590964
Средняя квадратичная ошибка для 10 ближайших соседей: 1991.3278515739114
```

In [21]:
```

```

```
# Оценка коэффициента детерминации

print('Коэффициент детерминации для 2 ближайших соседей:',r2_score(Y_test, target_2NN))
print('Коэффициент детерминации для 5 ближайших соседей:',r2_score(Y_test, target_5NN))
print('Коэффициент детерминации для 10 ближайших соседей:',r2_score(Y_test, target_10NN
))
```

```
Коэффициент детерминации для 2 ближайших соседей: 0.5017857093074095
Коэффициент детерминации для 5 ближайших соседей: 0.39275932960432836
Коэффициент детерминации для 10 ближайших соседей: 0.4031430105992597
```

```
## Grid Search (решетчатый поиск)
```

```
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
```

Подбор гиперпараметров

GridSearch через среднюю квадратичную ошибку

Рассмотрим все количества ближайших соседей от 1 до 100, чтобы найти лучший результат. Возьмем 10 фолдов.

```
from sklearn.model_selection import GridSearchCV
n_range = np.array(range(1, 101, 1))
tuned_parameters = [{'n_neighbors': n_range}]
gs = GridSearchCV(KNeighborsRegressor(), tuned_parameters, cv=10, scoring='neg_mean_squared_error')
gs.fit(X_train, Y_train)
```

Out[24]:

```
GridSearchCV(cv=10, estimator=KNeighborsRegressor(),
             param grid=[{'n neighbors': array([  1,   2,   3,   4,   5,   6,   7,   8,   9,  10,  11,  12,  13,
        14,  15,  16,  17,  18,  19,  20,  21,  22,  23,  24,  25,  26,
        27,  28,  29,  30,  31,  32,  33,  34,  35,  36,  37,  38,  39,
        40,  41,  42,  43,  44,  45,  46,  47,  48,  49,  50,  51,  52,
        53,  54,  55,  56,  57,  58,  59,  60,  61,  62,  63,  64,  65,
        66,  67,  68,  69,  70,  71,  72,  73,  74,  75,  76,  77,  78,
        79,  80,  81,  82,  83,  84,  85,  86,  87,  88,  89,  90,  91,
        92,  93,  94,  95,  96,  97,  98,  99, 100])}],
             scoring='neg mean squared error')
```

```
print('Лучшая модель:\n\n', gs.best_estimator_)
print('\nЛучшее число ближайших соседей:\n\n',gs.best_params_)
print('\nЛучшее значение средней квадратичной ошибки:\n\n',gs.best_score_)
```

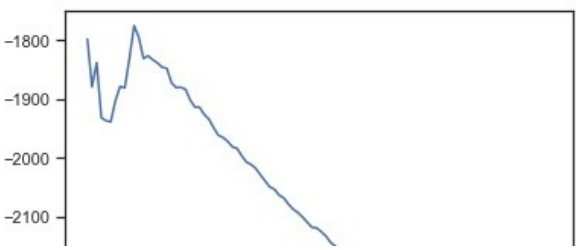
```
Лучшая модель:

 KNeighborsRegressor(n_neighbors=11)

Лучшее число ближайших соседей:

 {'n_neighbors': 11}

Лучшее значение средней квадратичной ошибки:

 -1774.7300745209031
```

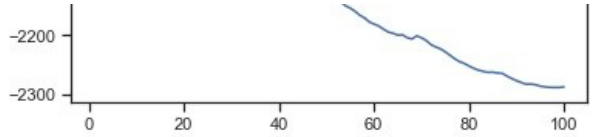
```
 print('Изменение качества тестовой выборки в зависимости от кол-ва соседей:\n')
plt.plot(n_range, gs.cv_results_['mean_test_score'])
```

```
Изменение качества тестовой выборки в зависимости от кол-ва соседей:
```

Out[26]:

```
[<matplotlib.lines.Line2D at 0x7fb97d5d9610>]
```





GridSearch через коэффициент детерминации

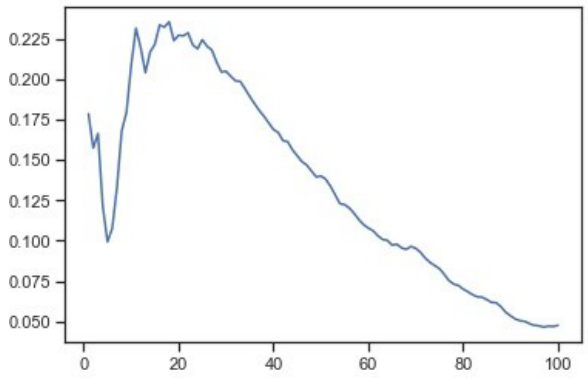
In [27]:

```
gs_det = GridSearchCV(KNeighborsRegressor(), tuned_parameters, cv=10, scoring='r2')
gs_det.fit(X_train, Y_train)
print('Лучшая модель:\n\n', gs_det.best_estimator_)
print('\nЛучшее число ближайших соседей:\n\n',gs_det.best_params_)
print('\nЛучшее значение коэффициента детерминации:\n\n',gs_det.best_score_)
print('\nИзменение качества тестовой выборки в зависимости от кол-ва соседей:\n')
plt.plot(n_range, gs_det.cv_results_['mean_test_score'])
```

```
Лучшая модель:

 KNeighborsRegressor(n_neighbors=18)

Лучшее число ближайших соседей:

 {'n_neighbors': 18}

Лучшее значение коэффициента детерминации:

 0.23532902602360656

Изменение качества тестовой выборки в зависимости от кол-ва соседей:
```

Out[27]:

```
[<matplotlib.lines.Line2D at 0x7fb977ec3040>]
```



Кросс-валидация

In [29]:

```
from sklearn.model_selection import cross_val_score
scores_2NN = cross_val_score(KNeighborsRegressor(n_neighbors = 2), X, Y, cv=5, scoring= 'r2')
scores_5NN = cross_val_score(KNeighborsRegressor(n_neighbors = 5), X, Y, cv=5, scoring= 'r2')
scores_10NN = cross_val_score(KNeighborsRegressor(n_neighbors = 10), X, Y, cv=5, scoring = 'r2')
scores_50NN = cross_val_score(KNeighborsRegressor(n_neighbors = 50), X, Y, cv=5, scoring = 'r2')
scores_100NN = cross_val_score(KNeighborsRegressor(n_neighbors = 100), X, Y, cv=5, scoring = 'r2')
print('Пример значений коэф. детерминации для 5 фолдов для метода 10 ближайших соседей: \n', scores_10NN, '\n\n'
print('Усредненное значение коэффициента детерминации для:\n')
print('- 2 ближайших соседей:', np.mean(scores_2NN), '\n')
print('- 5 ближайших соседей:', np.mean(scores_5NN), '\n')
print('- 10 ближайших соседей:', np.mean(scores_10NN), '\n')
print('- 50 ближайших соседей:', np.mean(scores_50NN), '\n')
print('- 100 ближайших соседей:', np.mean(scores_100NN), '\n')
```

```
Пример значений коэф. детерминации для 5 фолдов для метода 10 ближайших соседей:
 [0.34166201 0.38655715 0.14117213 0.28452217 0.2883947 ]


Усредненное значение коэффициента детерминации для:

- 2 ближайших соседей: 0.18694561138232885

- 5 ближайших соседей: 0.23548126907370337
```

- 10 ближайших соседей: 0.28846163209364245

- 50 ближайших соседей: 0.13534843218545478

- 100 ближайших соседей: 0.05880772437701802