Московский государственный технический университет им. Н.Э. Баумана
Кафедра «Системы обработки информации и управления»

Лабораторная работа №4

по дисциплине

«Методы машинного обучения»

на тему

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

Выполнил:
студент группы ИУ5-61Б
Белоусов Е. А.

____

Москва — 2020 г.

1. Цель

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

2. 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
        import sklearn
        import warnings
        warnings.filterwarnings('ignore')
```

Для данной задачи выберем датасет с красными винами

```
In [2]: data = pd.read_csv('../data/winequality-red.csv')
```

```
In [3]: data.head()
```

```
Out[3]:    fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
        0            7.4              0.70         0.00             1.9      0.076
        1            7.8              0.88         0.00             2.6      0.098
        2            7.8              0.76         0.04             2.3      0.092
        3           11.2              0.28         0.56             1.9      0.075
        4            7.4              0.70         0.00             1.9      0.076

           free sulfur dioxide  total sulfur dioxide  density    pH  sulphates  \
        0                 11.0                  34.0   0.9978  3.51       0.56
        1                 25.0                  67.0   0.9968  3.20       0.68
        2                 15.0                  54.0   0.9970  3.26       0.65
        3                 17.0                  60.0   0.9980  3.16       0.58
        4                 11.0                  34.0   0.9978  3.51       0.56

           alcohol  quality
        0      9.4        5
        1      9.8        5
        2      9.8        5
        3      9.8        6
        4      9.4        5
```

```
In [4]: data.dtypes
```

```
Out[4]: fixed acidity           float64
        volatile acidity        float64
        citric acid             float64
        residual sugar          float64
        chlorides               float64
        free sulfur dioxide     float64
        total sulfur dioxide    float64
        density                 float64
        pH                      float64
        sulphates               float64
        alcohol                 float64
        quality                   int64
        dtype: object

In [5]: data.shape

Out[5]: (1599, 12)

In [6]: wine_target = data['quality']
        del data['quality']

In [7]: wine_target[:10]

Out[7]: 0    5
        1    5
        2    5
        3    6
        4    5
        5    5
        6    5
        7    7
        8    7
        9    5
        Name: quality, dtype: int64

In [8]: data.head()

Out[8]:    fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
        0            7.4              0.70         0.00             1.9      0.076
        1            7.8              0.88         0.00             2.6      0.098
        2            7.8              0.76         0.04             2.3      0.092
        3           11.2              0.28         0.56             1.9      0.075
        4            7.4              0.70         0.00             1.9      0.076

           free sulfur dioxide  total sulfur dioxide  density    pH  sulphates  \
        0                 11.0                  34.0   0.9978  3.51       0.56
        1                 25.0                  67.0   0.9968  3.20       0.68
        2                 15.0                  54.0   0.9970  3.26       0.65
        3                 17.0                  60.0   0.9980  3.16       0.58
        4                 11.0                  34.0   0.9978  3.51       0.56
```

```
   alcohol
0      9.4
1      9.8
2      9.8
3      9.8
4      9.4
```

Разделяем выборку на обучающую и тестовую

```
In [9]: from sklearn.model_selection import train_test_split
        wine_X_train, wine_X_test, wine_y_train, wine_y_test = train_test_split(data, wine_target, test_size=
```

Обучаем модель ближайших соседей для произвольно заданного гиперпараметра k=3 Оценим качество модели с помощбю MAE

```
In [10]: from sklearn.neighbors import KNeighborsClassifier
         from sklearn.metrics import mean_absolute_error
         model_1 = KNeighborsClassifier(n_neighbors=3)
         model_1.fit(wine_X_train, wine_y_train)
         target = model_1.predict(wine_X_test)
         mean_absolute_error(wine_y_test, target)

Out[10]: 0.58333333333333337
```

Строим модель и оценивем ее используя кросс-валидацию

```
In [11]: from sklearn.model_selection import cross_val_score
         scores = cross_val_score(KNeighborsClassifier(n_neighbors=3), data, wine_target, cv=5, scoring='m
         scores, np.mean(scores)

Out[11]: (array([-0.63354037, -0.62305296, -0.63862928, -0.8081761 , -0.66561514]),
          -0.67380277164924562)
```

произведем подбор гиперпараметров используя GridSearch

```
In [12]: from sklearn.model_selection import GridSearchCV
         n_range = np.arange(1, 50)
         turned_parametrs = [{'n_neighbors' : n_range}]
         turned_parametrs

Out[12]: [{'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])}]
```

```
In [13]: %%time
         clf_gs = GridSearchCV(KNeighborsClassifier(), turned_parametrs, cv=5, scoring='mean_absolute_e
         clf_gs.fit(wine_X_train, wine_y_train)

CPU times: user 3.11 s, sys: 0 ns, total: 3.11 s
Wall time: 3.1 s
```

```
In [14]: clf_gs.cv_results_
```

```
Out[14]: {'mean_fit_time': array([ 0.00178742,  0.00129266,  0.00114484,  0.00118623,  0.0011426 ,
                0.00118914,  0.00122261,  0.00116243,  0.00121493,  0.0011888 ,
                0.00119019,  0.00119557,  0.00119338,  0.00124426,  0.00125017,
                0.00116644,  0.00122519,  0.00136981,  0.00136666,  0.00161529,
                0.00120463,  0.00132117,  0.00120082,  0.00119681,  0.00124373,
                0.00121365,  0.00118818,  0.00126786,  0.00120974,  0.0012434 ,
                0.0012044 ,  0.00121074,  0.00135059,  0.00142961,  0.00154557,
                0.00128417,  0.00121279,  0.00123959,  0.00124121,  0.00123434,
                0.00124722,  0.00136862,  0.00139508,  0.00144577,  0.00123138,
                0.00124712,  0.0012464 ,  0.00124855,  0.00130863]),
         'mean_score_time': array([ 0.00229034,  0.00145583,  0.00146494,  0.00153103,  0.00159202,
                0.00173202,  0.00181117,  0.00173898,  0.00178766,  0.00183311,
                0.00187182,  0.0018918 ,  0.0020205 ,  0.0019866 ,  0.00203457,
                0.00206075,  0.00211639,  0.0025104 ,  0.00247817,  0.00284777,
                0.00226197,  0.00235944,  0.00237184,  0.00239234,  0.00263901,
                0.0024394 ,  0.00250964,  0.0027442 ,  0.00256329,  0.0026124 ,
                0.00270681,  0.00267005,  0.00285244,  0.00308123,  0.00323558,
                0.00291739,  0.00284786,  0.00288582,  0.00295944,  0.00297804,
                0.00305953,  0.00310678,  0.0035008 ,  0.00356088,  0.00318384,
                0.00323429,  0.00361018,  0.0033762 ,  0.00357833]),
         'mean_test_score': array([-0.5567471 , -0.63806971, -0.63717605, -0.61215371, -0.57819482,
               -0.56925827, -0.56747096, -0.57730116, -0.58713137, -0.57640751,
               -0.5844504 , -0.5665773 , -0.57640751, -0.56747096, -0.55942806,
               -0.56568365, -0.55406613, -0.55317248, -0.55764075, -0.55317248,
               -0.5567471 , -0.5665773 , -0.56478999, -0.57283289, -0.55495979,
               -0.56210903, -0.56121537, -0.55942806, -0.56300268, -0.56121537,
               -0.55495979, -0.55317248, -0.56032172, -0.55227882, -0.56389634,
               -0.56478999, -0.57730116, -0.57193923, -0.5665773 , -0.55853441,
               -0.56121537, -0.56478999, -0.56032172, -0.5567471 , -0.55853441,
               -0.56032172, -0.56121537, -0.56478999, -0.56032172]),
         'mean_train_score': array([ 0.        , -0.28417315, -0.35634104, -0.39096786, -0.41844008,
               -0.43721685, -0.45686454, -0.46401617, -0.47184218, -0.48009638,
               -0.48590198, -0.48122019, -0.48613619, -0.48702906, -0.4897186 ,
               -0.49439041, -0.49862427, -0.50623029, -0.50443436, -0.51069163,
               -0.51359967, -0.5171776 , -0.5167329 , -0.52008312, -0.51895858,
               -0.52387906, -0.52276146, -0.52788919, -0.52364386, -0.52833187,
               -0.52789792, -0.52923145, -0.52700281, -0.53304234, -0.5372802 ,
               -0.53774134, -0.53839327, -0.53819027, -0.53818428, -0.53885292,
               -0.53952157, -0.54108933, -0.54108759, -0.5408711 , -0.53795832,
               -0.53931058, -0.54021018, -0.53885416, -0.54087033]),
         'param_n_neighbors': masked_array(data = [1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 2
          29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49],
                      mask = [False False False False False False False False False False False False
          False False False False False False False False False False False False
          False False False False False False False False False False False False
          False False False False False False False False False False False False
          False],
                fill_value = ?),
         'params': [{'n_neighbors': 1},
          {'n_neighbors': 2},
          {'n_neighbors': 3},
          {'n_neighbors': 4},
          {'n_neighbors': 5},
          {'n_neighbors': 6},
          {'n_neighbors': 7},
          {'n_neighbors': 8},
          {'n_neighbors': 9},
          {'n_neighbors': 10},
          {'n_neighbors': 11},
          {'n_neighbors': 12},
          {'n_neighbors': 13},
          {'n_neighbors': 14},
          {'n_neighbors': 15},
          {'n_neighbors': 16},
          {'n_neighbors': 17},
          {'n_neighbors': 18},
          {'n_neighbors': 19},
          {'n_neighbors': 20},
          {'n_neighbors': 21},
          {'n_neighbors': 22},
          {'n_neighbors': 23},
          {'n_neighbors': 24},
          {'n_neighbors': 25},
          {'n_neighbors': 26},
          {'n_neighbors': 27},
          {'n_neighbors': 28},
          {'n_neighbors': 29},
          {'n_neighbors': 30},
          {'n_neighbors': 31},
          {'n_neighbors': 32},
          {'n_neighbors': 33},
          {'n_neighbors': 34},
          {'n_neighbors': 35},
          {'n_neighbors': 36},
          {'n_neighbors': 37},
          {'n_neighbors': 38},
          {'n_neighbors': 39},
          {'n_neighbors': 40},
          {'n_neighbors': 41},
          {'n_neighbors': 42},
          {'n_neighbors': 43},
          {'n_neighbors': 44},
          {'n_neighbors': 45},
          {'n_neighbors': 46},
          {'n_neighbors': 47},
          {'n_neighbors': 48},
          {'n_neighbors': 49}],
         'rank_test_score': array([ 8, 49, 48, 47, 44, 37, 35, 42, 46, 40, 45, 32, 40, 35, 14, 31,  5,
                2, 11,  2,  8, 32, 27, 39,  6, 24, 20, 14, 25, 20,  6,  2, 16,  1,
               26, 27, 42, 38, 32, 12, 20, 27, 16,  8, 12, 16, 20, 27, 16], dtype=int32),
         'split0_test_score': array([-0.55752212, -0.64159292, -0.66371681, -0.59292035, -0.56637168,
               -0.53982301, -0.57079646, -0.55752212, -0.59734513, -0.58849558,
               -0.61946903, -0.60619469, -0.61061947, -0.61504425, -0.59292035,
               -0.59734513, -0.57079646, -0.5619469 , -0.53097345, -0.51769912,
               -0.53097345, -0.57079646, -0.56637168, -0.57522124, -0.56637168,
               -0.54867257, -0.57522124, -0.57079646, -0.57522124, -0.57079646,
               -0.57964602, -0.56637168, -0.57964602, -0.55309735, -0.57964602,
               -0.59292035, -0.60176991, -0.61946903, -0.60176991, -0.57964602,
               -0.59292035, -0.59292035, -0.57964602, -0.57964602, -0.57079646,
               -0.56637168, -0.56637168, -0.5840708 , -0.57079646]),
         'split0_train_score': array([-0.        , -0.27995521, -0.3493841 , -0.37849944, -0.40201568,
               -0.43673012, -0.46920493, -0.46584546, -0.47032475, -0.47144457,
               -0.47480403, -0.47144457, -0.47032475, -0.47928331, -0.47592385,
               -0.47816349, -0.46584546, -0.48712206, -0.48600224, -0.48824188,
               -0.49608063, -0.50503919, -0.50279955, -0.49944009, -0.49608063,
               -0.50391937, -0.49048152, -0.49048152, -0.49384099, -0.49496081,
               -0.50503919, -0.49832027, -0.51511758, -0.52407615, -0.51735722,
               -0.52855543, -0.52295633, -0.53191489, -0.53079507, -0.52855543,
               -0.53191489, -0.54199328, -0.5431131 , -0.53639418, -0.52519597,
               -0.53191489, -0.53415454, -0.52967525, -0.53191489]),
         'split1_test_score': array([-0.62831858, -0.63274336, -0.6460177 , -0.61061947, -0.59292035,
               -0.60619469, -0.59292035, -0.60176991, -0.60619469, -0.60619469,
               -0.5840708 , -0.57079646, -0.60619469, -0.58849558, -0.5840708 ,
               -0.59292035, -0.57522124, -0.56637168, -0.58849558, -0.57964602,
               -0.57522124, -0.56637168, -0.57522124, -0.60176991, -0.5840708 ,
               -0.58849558, -0.57522124, -0.57964602, -0.56637168, -0.55752212,
               -0.56637168, -0.57964602, -0.5840708 , -0.58849558, -0.5840708 ,
               -0.59734513, -0.59734513, -0.61061947, -0.60176991, -0.60176991,
               -0.60619469, -0.60619469, -0.59734513, -0.60619469, -0.60619469,
               -0.61504425, -0.61504425, -0.59292035, -0.60619469]),
         'split1_train_score': array([-0.        , -0.27995521, -0.35386338, -0.39529675, -0.42217245,
               -0.42665174, -0.42441209, -0.44680851, -0.46472564, -0.46024636,
               -0.46696529, -0.46696529, -0.48264278, -0.48152296, -0.48824188,
               -0.48264278, -0.49384099, -0.49048152, -0.4837626 , -0.50279955,
               -0.50503919, -0.50615901, -0.50279955, -0.50951848, -0.50727884,
               -0.51847704, -0.52071669, -0.52631579, -0.51735722, -0.51511758,
               -0.51847704, -0.51287794, -0.50391937, -0.51511758, -0.52407615,
               -0.52631579, -0.51175812, -0.52631579, -0.51847704, -0.51847704,
               -0.51735722, -0.51735722, -0.5162374 , -0.52631579, -0.52183651,
               -0.52743561, -0.53303471, -0.51511758, -0.52183651]),
         'split2_test_score': array([-0.50892857, -0.62946429, -0.65625   , -0.60714286, -0.59375   ,
               -0.55357143, -0.55357143, -0.5625    , -0.58035714, -0.55357143,
               -0.55803571, -0.54910714, -0.55357143, -0.53571429, -0.54464286,
               -0.54464286, -0.5625    , -0.55357143, -0.57142857, -0.57142857,
               -0.57589286, -0.57142857, -0.58035714, -0.55803571, -0.54464286,
               -0.57142857, -0.55803571, -0.55357143, -0.55357143, -0.5625    ,
               -0.54910714, -0.5625    , -0.55803571, -0.53125   , -0.54464286,
               -0.53571429, -0.58035714, -0.56696429, -0.5625    , -0.54464286,
               -0.53571429, -0.53571429, -0.53571429, -0.52232143, -0.53125   ,
               -0.52678571, -0.52232143, -0.53125   , -0.53125   ]),
         'split2_train_score': array([-0.        , -0.28379888, -0.36312849, -0.38659218, -0.41564246,
               -0.45251397, -0.46368715, -0.46703911, -0.47486034, -0.49273743,
               -0.49608939, -0.50055866, -0.5027933 , -0.48826816, -0.50837989,
               -0.50167598, -0.50391061, -0.52290503, -0.51843575, -0.50949721,
               -0.51061453, -0.51396648, -0.52625698, -0.53407821, -0.52849162,
               -0.52960894, -0.54189944, -0.53519553, -0.52402235, -0.54078212,
               -0.53854749, -0.55307263, -0.5396648 , -0.54189944, -0.54636872,
               -0.54972067, -0.55865922, -0.54972067, -0.55642458, -0.55977654,
               -0.55530726, -0.55195531, -0.55083799, -0.55307263, -0.55530726,
               -0.55865922, -0.55865922, -0.56759777, -0.56759777]),
         'split3_test_score': array([-0.54054054, -0.62612613, -0.60810811, -0.65315315, -0.6036036 ,
               -0.59459459, -0.58108108, -0.61261261, -0.58108108, -0.59009009,
               -0.58108108, -0.55405405, -0.56306306, -0.56306306, -0.53603604,
               -0.55855856, -0.55855856, -0.55405405, -0.56756757, -0.56756757,
               -0.56306306, -0.58558559, -0.5990991 , -0.61261261, -0.59009009,
               -0.59459459, -0.58108108, -0.57207207, -0.6036036 , -0.59459459,
               -0.58108108, -0.54954955, -0.57207207, -0.57207207, -0.58108108,
               -0.58108108, -0.57657658, -0.56756757, -0.55855856, -0.57207207,
               -0.56756757, -0.57207207, -0.56756757, -0.56306306, -0.56756757,
               -0.57657658, -0.57207207, -0.58108108, -0.58108108]),
         'split3_train_score': array([-0.        , -0.28651059, -0.36566332, -0.41694537, -0.42809365,
               -0.42920847, -0.46265329, -0.46822742, -0.47157191, -0.4960981 ,
               -0.50278707, -0.48494983, -0.47826087, -0.4916388 , -0.49052397,
               -0.5039019 , -0.51282051, -0.51727982, -0.50613155, -0.52173913,
               -0.52508361, -0.52954292, -0.52396878, -0.52619844, -0.52173913,
               -0.51616499, -0.51727982, -0.5328874 , -0.5328874 , -0.53734671,
               -0.52842809, -0.53400223, -0.52842809, -0.53846154, -0.54180602,
               -0.53511706, -0.53623188, -0.53400223, -0.53177258, -0.53511706,
               -0.53957637, -0.53846154, -0.53400223, -0.53846154, -0.53177258,
               -0.53177258, -0.53177258, -0.52842809, -0.53623188]),
         'split4_test_score': array([-0.54751131, -0.66063348, -0.61085973, -0.59728507, -0.53393665,
               -0.5520362 , -0.53846154, -0.5520362 , -0.57013575, -0.54298643,
               -0.57918552, -0.5520362 , -0.54751131, -0.53393665, -0.53846154,
               -0.53393665, -0.50226244, -0.52941176, -0.52941176, -0.52941176,
               -0.53846154, -0.53846154, -0.50226244, -0.5158371 , -0.48868778,
               -0.50678733, -0.5158371 , -0.52036199, -0.5158371 , -0.52036199,
               -0.49773756, -0.50678733, -0.50678733, -0.5158371 , -0.52941176,
               -0.5158371 , -0.52941176, -0.49321267, -0.50678733, -0.49321267,
               -0.50226244, -0.5158371 , -0.52036199, -0.51131222, -0.5158371 ,
               -0.5158371 , -0.52941176, -0.53393665, -0.51131222]),
         'split4_train_score': array([-0.        , -0.29064588, -0.34966592, -0.37750557, -0.42427617,
               -0.44097996, -0.46436526, -0.47216036, -0.47772829, -0.47995546,
               -0.48886414, -0.48218263, -0.49665924, -0.49443207, -0.48552339,
               -0.50556793, -0.51670379, -0.51336303, -0.52783964, -0.5311804 ,
               -0.5311804 , -0.5311804 , -0.52783964, -0.5311804 , -0.54120267,
               -0.55122494, -0.54342984, -0.5545657 , -0.55011136, -0.55345212,
               -0.54899777, -0.54788419, -0.54788419, -0.54565702, -0.55679287,
               -0.54899777, -0.5623608 , -0.54899777, -0.55345212, -0.55233853,
               -0.55345212, -0.55567929, -0.56124722, -0.55011136, -0.55567929,
               -0.5467706 , -0.54342984, -0.55345212, -0.5467706 ]),
         'std_fit_time': array([  4.04459816e-04,   2.13446629e-04,   1.69216064e-05,
                  4.62204040e-05,   1.84718862e-05,   8.07901470e-05,
                  1.09690893e-04,   2.24106362e-05,   9.64835421e-05,
                  2.18937329e-05,   5.33953299e-05,   6.85990538e-05,
                  3.68205367e-05,   6.21603281e-05,   6.76143657e-05,
                  1.16670272e-05,   5.09715057e-05,   1.83298685e-04,
                  3.45268840e-04,   2.88954072e-04,   1.98031568e-05,
                  1.21998635e-04,   4.82616214e-05,   3.09019912e-05,
                  9.23747670e-05,   4.05857056e-05,   1.39238083e-05,
                  9.11308542e-05,   3.02109363e-05,   6.42810093e-05,
                  1.53172387e-05,   1.61811582e-05,   1.60827313e-04,
                  4.48215322e-04,   2.45635182e-04,   9.68077377e-05,
                  2.26391717e-05,   8.46521474e-05,   3.32376365e-05,
                  1.35201624e-05,   7.72681327e-05,   2.44902590e-04,
                  1.29993068e-04,   2.05955016e-04,   3.43042498e-05,
                  7.53395769e-05,   4.56528687e-05,   4.36399533e-05,
                  6.41363527e-05]),
         'std_score_time': array([  4.82873725e-04,   1.00256915e-04,   2.47696595e-05,
                  3.18752712e-05,   6.64009397e-05,   1.22907643e-04,
                  2.38916764e-04,   5.32491551e-05,   3.98508649e-05,
                  6.25668863e-05,   3.62088675e-05,   3.16123257e-05,
                  5.30731967e-05,   1.57461709e-05,   3.71335234e-05,
                  4.01027137e-05,   4.29480216e-05,   7.13759138e-04,
                  5.47696214e-04,   7.72837884e-04,   1.16025367e-05,
                  4.43413435e-05,   2.88068058e-05,   5.88644608e-05,
                  4.82887870e-04,   2.33214810e-05,   4.82841831e-05,
                  5.14847751e-04,   2.65107738e-05,   7.28955049e-05,
                  2.15966882e-04,   3.93374761e-05,   2.22587718e-04,
                  6.46683780e-04,   5.53438393e-04,   1.94011240e-04,
                  5.81310065e-05,   5.11170768e-05,   8.02021673e-05,
                  7.84977956e-05,   9.02551599e-05,   1.52213143e-04,
                  4.87709470e-04,   4.79965311e-04,   5.02164091e-05,
                  8.58474556e-05,   6.02740604e-04,   7.80999585e-05,
                  4.21004671e-04]),
         'std_test_score': array([ 0.03951422,  0.01232525,  0.02313635,  0.02138603,  0.02520717,
                 0.02618631,  0.01935634,  0.02484405,  0.01296001,  0.0238923 ,
                 0.01987711,  0.02132037,  0.02683692,  0.03121213,  0.02425561,
                 0.02549575,  0.02636609,  0.01274455,  0.02347102,  0.02479964,
                 0.01871949,  0.01536613,  0.03281276,  0.03419962,  0.03648306,
                 0.03175331,  0.0237958 ,  0.02118148,  0.02857687,  0.0239226 ,
                 0.03063151,  0.02493364,  0.02799617,  0.02633566,  0.02235968,
                 0.03269609,  0.02563156,  0.04462335,  0.03496263,  0.03721895,
                 0.03789721,  0.03404543,  0.02827103,  0.03539173,  0.03184954,
                 0.03578503,  0.03341468,  0.02647157,  0.03429606]),
         'std_train_score': array([ 0.        ,  0.00407632,  0.00681277,  0.01448732,  0.00915172,
                 0.00921038,  0.01638148,  0.00886172,  0.00439946,  0.01330224,
                 0.01326218,  0.0117267 ,  0.01193667,  0.00579488,  0.01057325,
                 0.01157415,  0.01818614,  0.01458878,  0.01739995,  0.01489658,
                 0.01288466,  0.01120751,  0.01144292,  0.01337278,  0.01583179,
                 0.01592298,  0.01933736,  0.0209371 ,  0.01851404,  0.02076318,
                    0.01529819,  0.02079435,  0.01595301,  0.01156404,  0.01452653,
                    0.00991991,  0.01968529,  0.0094623 ,  0.01449069,  0.01519757,
                    0.01408809,  0.01343035,  0.01531873,  0.00971561,  0.01467019,
                    0.01168379,  0.01010034,  0.01895086,  0.01557672])}
```

```
In [15]: # Лучшая модель
         clf_gs.best_estimator_
```

```
Out[15]: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                    metric_params=None, n_jobs=1, n_neighbors=34, p=2,
                    weights='uniform')
```

```
In [16]: # Лучшее значение метрики
         clf_gs.best_score_
```

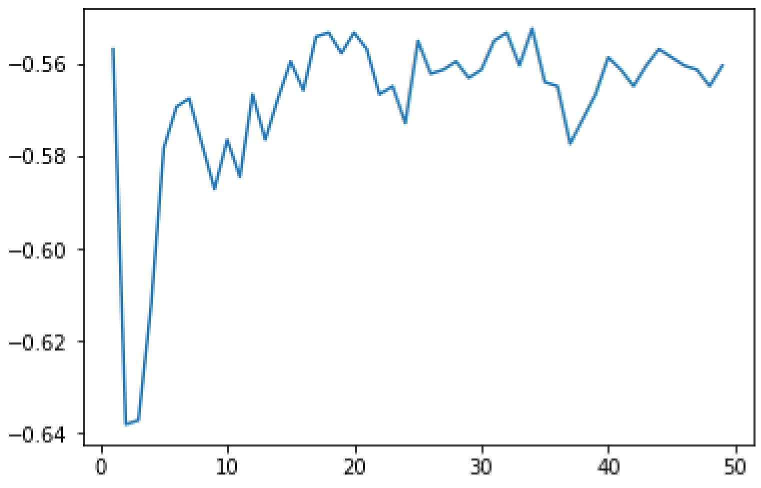
```
Out[16]: -0.55227882037533516
```

```
In [17]: # Лучшее значение параметров
         clf_gs.best_params_
```

```
Out[17]: {'n_neighbors': 34}
```

```
In [18]: %matplotlib inline
         import matplotlib.pyplot as plt
         # Изменение качества на тестовой выборке в зависимости от K-соседей
         plt.plot(n_range, clf_gs.cv_results_['mean_test_score'])
```

```
Out[18]: [<matplotlib.lines.Line2D at 0x7f4e39a42c10>]
```



−0.56
−0.58
−0.60
−0.62
−0.64
0
10
20
30
40
50