Подготовка обучающей и тестовой выборки. Кросс-валидация и подбор гиперпараметров на примере метода ближайших соседей.

1. Цель работы

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

---

2. Подготовка данных и библиотек

In [2]:
```python
from operator import itemgetter
import matplotlib. pyplot as plt
import matplotlib. ticker as ticker
import numpy as np
import pandas as pd
import pandas_profiling
import math
from simple_kNN import *
from sklearn.datasets import *
from typing import Dict, Tuple
from scipy import stats
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
from sklearn.model_selection import cross_val_score
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
from sklearn.metrics import confusion_matrix
from sklearn.metrics import mean_absolute_error, mean_squared_error
from sklearn.metrics import roc_curve, roc_auc_score, classification_report
import seaborn as sns
import matplotlib. pyplot as plt
%matplotlib inline
sns.set(style="ticks")
```

In [3]:
```python
# Загрузка датасета из скалерна
data = load_boston()
# Конвертирование массива в датафрейм
boston_df = pd.DataFrame(data=np.c_[data['data']], columns=data.feature_names)
boston_df['target'] = data['target']
```

In [4]:
```python
# Столбцы
data.feature_names
```

Out[4]:
```
array(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD',
       'TAX', 'PTRATIO', 'B', 'LSTAT'], dtype='<U7')
```

In [5]:
```python
# Информация о датасете
boston_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   CRIM     506 non-null    float64
 1   ZN       506 non-null    float64
 2   INDUS    506 non-null    float64
 3   CHAS     506 non-null    float64
 4   NOX      506 non-null    float64
 5   RM       506 non-null    float64
 6   AGE      506 non-null    float64
 7   DIS      506 non-null    float64
 8   RAD      506 non-null    float64
 9   TAX      506 non-null    float64
 10  PTRATIO  506 non-null    float64
 11  B        506 non-null    float64
 12  LSTAT    506 non-null    float64
 13  target   506 non-null    float64
dtypes: float64(14)
memory usage: 55.5 KB
```

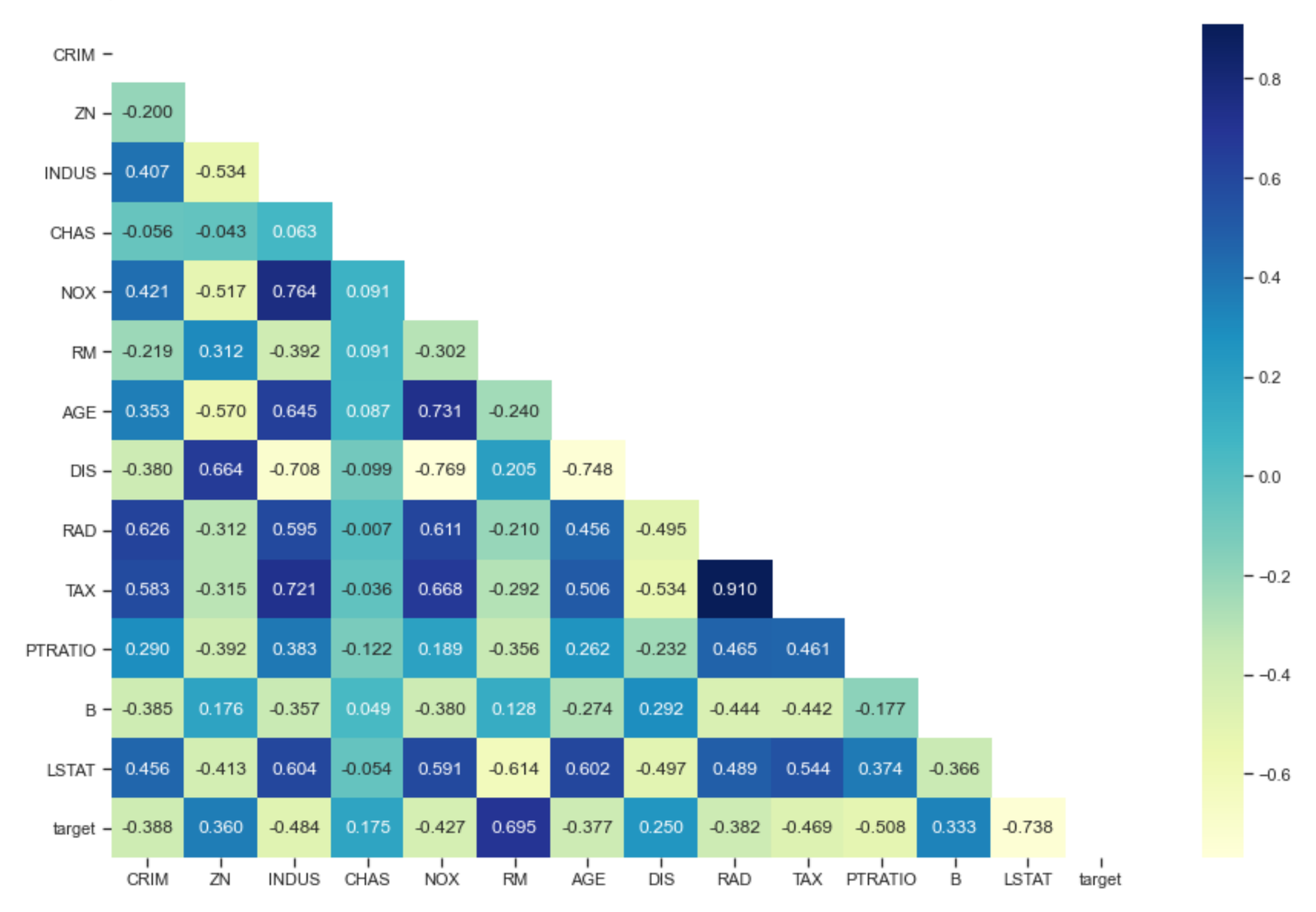
In [6]:
```python
boston_df.describe()
```

Out[6]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	target
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.574901	3.795043	9.549407	408.237154	18.455534	356.674032	12.653063	22.532806
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861	2.105710	8.707259	168.537116	2.164946	91.294864	7.141062	9.197104
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.129600	1.000000	187.000000	12.600000	0.320000	1.730000	5.000000
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000	2.100175	4.000000	279.000000	17.400000	375.377500	6.950000	17.025000
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000	3.207450	5.000000	330.000000	19.050000	391.440000	11.360000	21.200000
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000	5.188425	24.000000	666.000000	20.200000	396.225000	16.955000	25.000000
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.126500	24.000000	711.000000	22.000000	396.900000	37.970000	50.000000

In [7]:
```python
fig, ax = plt.subplots(figsize = (15,10))
mask = np.zeros_like(boston_df.corr(), dtype=np.bool)
mask[np.triu_indices_from(mask)] = True
sns.heatmap(boston_df.corr(), cmap='YlGnBu', mask=mask, annot=True, fmt='.3f')
```

Out[7]:
```
<AxesSubplot:>
```



In [8]:
```python
pandas_profiling.ProfileReport(boston_df)
```

Out[8]:

3. Разделение датафрейма на обучающую и тестовую выборку

In [9]:
```python
boston_x_train, boston_x_test, boston_y_train, boston_y_test = train_test_split(data.data, data.target, test_size=0.25, random_state=1)
```

In [10]:
```python
print(f'boston_X_train\tboston_y_train\nСтроки : {boston_x_train.shape[0]}\tСтроки : {boston_y_train.shape[0]}\nСтолбцы : {boston_x_train.shape[1]}\tСтолбцы : 1')
```

```
boston_X_train  boston_y_train
Строки : 379    Строки : 379
Столбцы : 13    Столбцы : 1
```

In [11]:
```python
print(f'boston_X_test\tboston_y_test\nСтроки : {boston_x_test.shape[0]}\tСтроки : {boston_y_test.shape[0]}\nСтолбцы : {boston_x_test.shape[1]}\tСтолбцы : 1')
```

```
boston_X_test   boston_y_test
Строки : 127    Строки : 127
Столбцы : 13    Столбцы : 1
```

In [12]:
```python
np.unique(boston_y_train)
```

Out[12]:
```
array([ 5. ,  5.6,  7. ,  7.2,  7.4,  7.5,  8.3,  8.4,  8.5,  8.7,  8.8,
        9.5,  9.6,  9.7, 10.2, 10.4, 10.5, 10.8, 10.9, 11.3, 11.7, 11.8,
       11.9, 12. , 12.1, 12.3, 12.5, 12.7, 12.8, 13.1, 13.2, 13.3, 13.4,
       13.5, 13.6, 13.8, 13.9, 14. , 14.1, 14.2, 14.3, 14.4, 14.5, 14.8,
       14.9, 15. , 15.1, 15.2, 15.3, 15.4, 15.6, 15.7, 16. , 16.1, 16.2,
       16.3, 16.4, 16.5, 16.6, 16.7, 16.8, 17. , 17.1, 17.2, 17.3, 17.4,
       17.5, 17.6, 17.7, 17.8, 17.9, 18.2, 18.3, 18.4, 18.5, 18.6, 18.7,
       18.8, 18.9, 19. , 19.1, 19.2, 19.3, 19.4, 19.5, 19.6, 19.7, 19.8,
       19.9, 20. , 20.1, 20.2, 20.3, 20.4, 20.5, 20.6, 20.7, 20.8, 20.9,
       21. , 21.1, 21.2, 21.4, 21.5, 21.6, 21.7, 21.8, 21.9, 22. , 22.2,
       22.3, 22.4, 22.5, 22.6, 22.7, 22.8, 22.9, 23. , 23.1, 23.2, 23.3,
       23.4, 23.6, 23.7, 23.8, 23.9, 24. , 24.1, 24.2, 24.3, 24.4, 24.5,
       24.6, 24.7, 24.8, 25. , 25.1, 25.2, 26.2, 26.4, 26.6, 26.7, 27.1,
       27.5, 28. , 28.1, 28.4, 28.5, 28.6, 28.7, 29. , 29.1, 29.4, 29.6,
       29.8, 29.9, 30.1, 30.3, 30.5, 30.7, 30.8, 31. , 31.1, 31.5, 31.6,
       31.7, 32. , 32.2, 32.4, 32.5, 32.7, 32.9, 33. , 33.1, 33.2, 33.3,
       33.4, 33.8, 34.6, 34.7, 34.9, 35.1, 35.2, 35.4, 36.1, 36.2, 36.5,
       37. , 37.6, 37.9, 38.7, 39.8, 41.3, 42.3, 42.8, 43.8, 44. , 45.4,
       46. , 46.7, 48.5, 48.8, 50. ])
```

In [13]:
```python
print(f'Размер boston_y_test : {len(boston_y_test)}')
print(boston_y_test)
```

```
Размер boston_y_test : 127
[28.2 23.9 16.6 22.  20.8 23.  27.9 14.5 21.5 22.6 23.7 31.2 19.3 19.4
 19.4 27.9 13.9 50.  24.1 14.6 16.2 15.6 23.8 25.  23.5  8.3 13.5 17.5
 43.1 11.5 24.1 18.5 50.  12.6 19.8 24.5 14.9 36.2 11.9 19.1 22.6 20.7
 30.1 13.3 14.6  8.4 50.  12.7 25.  18.6 29.8 22.2 28.7 23.8  8.1 22.2
  6.3 22.1 17.5 48.3 16.7 26.6  8.5 14.5 23.7 37.2 41.7 16.5 21.7 22.7
 23.  10.5 21.9 21.  20.4 21.8 50.  22.  23.3 37.3 18.  19.2 34.9 13.4
 22.9 22.5 13.  24.6 18.3 18.1 23.9 50.  13.6 22.9 10.9 18.9 22.4 22.9
 44.8 21.7 10.2 15.4 25.3 23.3  7.2 21.2 11.7 27.  29.6 26.5 43.5 23.6
 11.  33.4 36.  36.4 19.  20.2 34.9 50.  19.3 14.9 26.6 19.9 24.8 21.2
 23.9]
```

4. Построение базовой модели на основе ближайших соседей

In [16]:
```python
# 2 ближайших соседа
cl1_1 = KNeighborsRegressor(n_neighbors=2)
cl1_1.fit(boston_x_train, boston_y_train)
target1_1 = cl1_1.predict(boston_x_test)
len(target1_1), target1_1
```

Out[16]:
```
(127,
 array([26.4 , 28.4 , 18.  , 19.15, 33.05, 17.3 , 29.65, 15.1 , 22.35,
        25.05, 24.1 , 27.65, 22.  , 33.95, 17.6 , 16.65, 12.  , 34.65,
        23.95, 12.6 , 20.95, 18.8 , 22.2 , 21.6 , 30.25, 11.05, 13.55,
        17.3 , 42.65, 13.7 , 33.4 , 19.05, 41.95, 15.1 , 21.1 , 21.05,
        15.7 , 26.05, 11.15, 22.9 , 22.2 , 21.4 , 22.2 , 19.  , 17.8 ,
        12.95, 50.  , 13.95, 21.6 , 14.35, 21.15, 22.65, 18.6 , 17.2 ,
        12.1 , 21.3 , 15.55, 20.45, 18.  , 28.  , 15.7 , 28.45, 14.1 ,
        14.2 , 18.9 , 43.1 , 28.3 , 22.  , 24.65, 32.8 , 26.65,  9.35,
        15.7 , 17.4 , 19.75, 20.9 , 20.95, 22.75, 45.65, 29.65, 17.45,
        20.65, 31.25, 13.  , 24.3 , 24.8 , 15.7 , 23.65, 18.6 , 17.45,
        22.5 , 42.65, 17.65, 20.65, 12.3 , 16.95, 21.  , 22.85, 26.3 ,
        21.3 , 16.25, 15.1 , 24.95, 26.4 , 11.75, 21.1 , 13.45, 18.8 ,
        38.  , 22.5 , 22.85, 22.35, 15.2 , 36.9 , 39.45, 30.15, 20.05,
        14.35, 29.65, 28.3 , 20.85, 13.05, 24.95, 15.3 , 21.55, 18.6 ,
        25.6 ]))
```

In [17]:
```python
# 10 ближайших соседа
cl1_2 = KNeighborsRegressor(n_neighbors=10)
cl1_2.fit(boston_x_train, boston_y_train)
target1_2 = cl1_2.predict(boston_x_test)
len(target1_2), target1_2
```

Out[17]:
```
(127,
 array([31.08, 29.37, 21.51, 26.99, 24.53, 16.73, 35.35, 16.9 , 21.33,
        25.37, 21.75, 29.37, 18.03, 26.44, 20.25, 20.77, 18.89, 26.81,
        26.44, 20.73, 22.08, 17.12, 20.11, 26.21, 26.86, 12.29, 17.34,
        20.79, 29.31, 12.67, 29.09, 20.94, 36.9 , 15.86, 21.16, 21.3 ,
        18.79, 22.76, 18.41, 21.04, 25.  , 21.71, 25.25, 18.19, 17.54,
        12.22, 30.12, 12.78, 17.74, 16.06, 22.43, 23.  , 25.9 , 18.15,
        17.97, 22.73, 20.77, 22.02, 21.18, 26.2 , 18.76, 22.32, 13.96,
        11.74, 26.27, 32.34, 26.77, 20.31, 23.88, 23.83, 26.58,  9.67,
        20.02, 23.18, 19.91, 21.68, 20.37, 23.43, 27.43, 35.35, 16.86,
        18.56, 25.83, 12.87, 23.85, 28.28, 17.93, 27.74, 20.16, 16.86,
        23.15, 29.99, 15.59, 22.68, 12.83, 21.73, 24.12, 24.44, 28.14,
        20.89, 13.22, 14.73, 22.27, 24.21, 10.81, 20.89, 12.29, 24.21,
        31.95, 20.14, 25.71, 23.39, 12.87, 31.08, 29.55, 25.76, 21.45,
        22.04, 33.72, 28.83, 25.  , 12.29, 22.27, 25.13, 23.74, 20.49,
        26.9 ]))
```

5. Метрики качества классификации

In [18]:
```python
cl1_1.score(boston_x_train, boston_y_train)
```

Out[18]:
```
0.8566879344566602
```

In [19]:
```python
cl1_2.score(boston_x_train, boston_y_train)
```

Out[19]:
```
0.5342300718439038
```

6. Произведите подбор гиперпараметра К с использованием GridSearchCV

In [21]:
```python
param_grid = {'n_neighbors' : np.arange(1, 25)}
grid = GridSearchCV(KNeighborsRegressor(), param_grid, cv=3)
grid.fit(boston_x_train, boston_y_train)
print(f'Подобранный параметр: {grid.best_params_}')
print(f'Оценка при подобранном параметре: {grid.best_score_}')
```

```
Подобранный параметр: {'n_neighbors': 7}
Оценка при подобранном параметре: 0.3376584947980437
```

7. Кросс валидация

In [22]:
```python
DEFAULT_FOLDS = 3
DEFAULT_RANGES = np.arange(1, 25)
```

In [23]:
```python
cv1 = cross_val_score(KNeighborsRegressor(n_neighbors=5), data.data, data.target, cv=DEFAULT_FOLDS, scoring='max_error')
cv1
```

Out[23]:
```
array([-31.72, -31.66, -31.66])
```

In [24]:
```python
from sklearn.model_selection import *
```

In [25]:
```python
def knn_cross_strtegies(mass_strategies, mass_matrics, k_range, cv, x, y):
    k_strategies_scores = {}
    for strat in mass_strategies:
        strategia = strat(cv) # n_splits
        k_scores = knn_cross_val_score(mass_matrics, k_range, strategia, x, y)
        k_strategies_scores[strat.__name__] = k_scores

    return k_strategies_scores

def knn_cross_val_score(mass_matrics, k_range, cv, x, y):
    k_scores = {}
    for metric in mass_matrics:
        for k in k_range:
            knn = KNeighborsRegressor(n_neighbors=k)
            scores = cross_val_score(knn, x, y, cv=cv, scoring=metric)
            if metric not in k_scores:
                k_scores.update({metric : []})

            k_scores[metric].append(scores.mean())
    return k_scores

def plot_knn_cross_val_score(k_scores, cv):
    columns=len(k_scores.items())
    fig = plt.figure(figsize=(10*columns, 10))
    rows=1
    index = 1
    for stratName, metrics in k_scores.items():
        ax = fig.add_subplot(rows, columns, index)
        for name, scores in metrics.items():
            ax.plot(np.arange(1, len(scores)+1), scores)
        ax.set_title(stratName)
        ax.legend(metrics.keys(), loc='lower right')
        index+=1
```

In [26]:
```python
mass_metrics = ['neg_mean_absolute_error', 'neg_mean_squared_error', 'max_error']
strategies = [KFold, RepeatedKFold, ShuffleSplit]
```

In [27]:
```python
scores = knn_cross_strtegies(strategies, mass_metrics, DEFAULT_RANGES, DEFAULT_FOLDS, data.data, data.target)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:67: FutureWarning: Pass n_splits=3 as keyword args. From version 0.25 passing these as positional arguments will result in an error
  warnings.warn("Pass {} as keyword args. From version 0.25 "
```

7. Сравнение моделей

In [28]:
```python
plot_knn_cross_val_score(scores, DEFAULT_FOLDS)
```