



**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 вычислительная техника»  
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Технологии машинного обучения»

Отчет по лабораторной работе №3  
«Подготовка обучающей и тестовой выборки, кросс-валидация и подбор  
гиперпараметров на примере метода ближайших соседей»

Выполнил:  
студент группы ИУ5-63Б

Баркалова И.В.

Подпись и дата:

Проверил:  
преподаватель каф.  
ИУ5

Гапанюк Ю.Е.

Подпись и дата:

Москва, 2022 г.

## Цель лабораторной работы

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

## Описание задания

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.

# Текст программы и результаты ее выполнения

```
In [13]: import numpy as np
import pandas as pd
from sklearn.datasets import *
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set(style = "ticks")
```

```
In [14]: wine = load_wine()
```

```
In [15]: wine['data'].shape
```

```
Out[15]: (178, 13)
```

```
In [16]: for x in wine:
print(x)
```

```
data
target
frame
target_names
DESCR
feature_names
```

```
In [17]: data = pd.DataFrame(data= np.c_[wine['data'], wine['target']],
columns= wine['feature_names'] + ['target'])
```

```
In [18]: data
```

```
Out[18]:
```

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins	color_intensity	hue	od280/od310
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06	0.28	2.29	5.64	1.04	
1	13.20	1.78	2.14	11.2	100.0	2.65	2.76	0.26	1.28	4.38	1.05	
2	13.16	2.36	2.67	18.6	101.0	2.80	3.24	0.30	2.81	5.68	1.03	
3	14.37	1.95	2.50	16.8	113.0	3.85	3.49	0.24	2.18	7.80	0.86	
4	13.24	2.59	2.87	21.0	118.0	2.80	2.69	0.39	1.82	4.32	1.04	
...	...	...	...	...	...	...	...	...	...	...	...	...
173	13.71	5.65	2.45	20.5	95.0	1.68	0.61	0.52	1.06	7.70	0.64	
174	13.40	3.91	2.48	23.0	102.0	1.80	0.75	0.43	1.41	7.30	0.70	
175	13.27	4.28	2.26	20.0	120.0	1.59	0.69	0.43	1.35	10.20	0.59	
176	13.17	2.59	2.37	20.0	120.0	1.65	0.68	0.53	1.46	9.30	0.60	
177	14.13	4.10	2.74	24.5	96.0	2.05	0.76	0.56	1.35	9.20	0.61	

178 rows x 14 columns

```
In [19]: data.dtypes
```

```
Out[19]: alcohol          float64
malic_acid              float64
ash                    float64
alcalinity_of_ash      float64
magnesium              float64
total_phenols          float64
flavanoids             float64
nonflavanoid_phenols   float64
proanthocyanins        float64
color_intensity        float64
hue                   float64
od280/od315_of_diluted_wines float64
proline                float64
target                 float64
dtype: object
```

```
In [20]: data1 = pd.read_csv('penguins_size.csv', sep=',')
```

```
In [21]: data1
```

```
Out[21]:
```

	species	island	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_g	sex
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	MALE
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	FEMALE
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	FEMALE
3	Adelie	Torgersen	NaN	NaN	NaN	NaN	NaN
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0	FEMALE
...	...	...	...	...	...	...	...
339	Gentoo	Biscoe	NaN	NaN	NaN	NaN	NaN
340	Gentoo	Biscoe	46.8	14.3	215.0	4850.0	FEMALE
341	Gentoo	Biscoe	50.4	15.7	222.0	5750.0	MALE
342	Gentoo	Biscoe	45.2	14.8	212.0	5200.0	FEMALE
343	Gentoo	Biscoe	49.9	16.1	213.0	5400.0	MALE

344 rows x 7 columns

```
In [22]: data1.dtypes
```

```
Out[22]: species          object
island                  object
culmen_length_mm        float64
culmen_depth_mm         float64
flipper_length_mm        float64
body_mass_g             float64
sex                     object
dtype: object
```

```
In [24]: data1 = data1.dropna(axis=0, how='any')
data1.shape
data1
```

```
Out[24]:
```

	species	island	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_g	sex
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	MALE
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	FEMALE
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	FEMALE
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0	FEMALE
5	Adelie	Torgersen	39.3	20.6	190.0	3650.0	MALE
...	...	...	...	...	...	...	...
338	Gentoo	Biscoe	47.2	13.7	214.0	4925.0	FEMALE
340	Gentoo	Biscoe	46.8	14.3	215.0	4850.0	FEMALE
341	Gentoo	Biscoe	50.4	15.7	222.0	5750.0	MALE
342	Gentoo	Biscoe	45.2	14.8	212.0	5200.0	FEMALE
343	Gentoo	Biscoe	49.9	16.1	213.0	5400.0	MALE

334 rows x 7 columns

```
In [25]: data1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 334 entries, 0 to 343
Data columns (total 7 columns):
#   Column              Non-Null Count  Dtype
---  -
0   species              334 non-null   object
1   island               334 non-null   object
2   culmen_length_mm     334 non-null   float64
3   culmen_depth_mm      334 non-null   float64
4   flipper_length_mm    334 non-null   float64
5   body_mass_g          334 non-null   float64
6   sex                  334 non-null   object
dtypes: float64(4), object(3)
memory usage: 20.9+ KB
```

```
In [26]: #Кодирование категориальных признаков
data1["species"].value_counts()
data1["species"] = data1["species"].astype('category')

data1["island"] = data1["island"].astype('category')
data1["sex"] = data1["sex"].astype('category')

#Назначить закодированную переменную новому столбцу с помощью метода доступа
data1["species_cat"] = data1["species"].cat.codes
data1["island_cat"] = data1["island"].cat.codes
data1["sex_cat"] = data1["sex"].cat.codes
data1

data1_cat = data1.drop(['species', 'island', 'sex'], axis=1, inplace=True)
data1
```

Out [26]:

	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_g	species_cat	island_cat	sex_cat
0	39.1	18.7	181.0	3750.0	0	2	2
1	39.5	17.4	186.0	3800.0	0	2	1
2	40.3	18.0	195.0	3250.0	0	2	1
4	36.7	19.3	193.0	3450.0	0	2	1
5	39.3	20.6	190.0	3650.0	0	2	2
...	...	...	...	...	...	...	...
338	47.2	13.7	214.0	4925.0	2	0	1
340	46.8	14.3	215.0	4850.0	2	0	1
341	50.4	15.7	222.0	5750.0	2	0	2
342	45.2	14.8	212.0	5200.0	2	0	1
343	49.9	16.1	213.0	5400.0	2	0	2

334 rows x 7 columns

```
In [31]: #разделение выборки
from sklearn.model_selection import train_test_split
y = data1['body_mass_g']
X = data1.drop('body_mass_g', axis=1)
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=3)
x_train
```

Out [31]:

	culmen_length_mm	culmen_depth_mm	flipper_length_mm	species_cat	island_cat	sex_cat
228	43.3	13.4	209.0	2	0	1
48	36.0	17.9	190.0	0	1	1
5	39.3	20.6	190.0	0	2	2
45	39.6	18.8	190.0	0	1	2
174	43.2	16.6	187.0	1	1	1
...	...	...	...	...	...	...
284	45.8	14.2	219.0	2	0	1
263	49.6	15.0	216.0	2	0	2
137	40.2	20.1	200.0	0	1	2
256	42.6	13.7	213.0	2	0	1
158	46.1	18.2	178.0	1	1	1

233 rows x 6 columns

```
In [32]: y_train
```

```
Out[32]: 228    4400.0
         48     3450.0
          5     3650.0
          45    4600.0
          174   2900.0
          ...
          284    4700.0
          263    4750.0
          137    3975.0
          256    4950.0
          158    3250.0
         Name: body_mass_g, Length: 233, dtype: float64
```

```
In [35]: #Масштабирование данных
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler().fit(x_train)
x_train = pd.DataFrame(scaler.transform(x_train), columns = x_train.columns)
x_test = pd.DataFrame(scaler.transform(x_test), columns = x_train.columns)
x_train.describe()
```

```
Out[35]:
```

	culmen_length_mm	culmen_depth_mm	flipper_length_mm	species_cat	island_cat	sex_cat
count	233.000000	233.000000	233.000000	233.000000	233.000000	233.000000
mean	0.409151	0.494781	0.497262	0.463519	0.324034	0.736052
std	0.203907	0.238422	0.242362	0.444991	0.352243	0.254419
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.245283	0.320988	0.310345	0.000000	0.000000	0.500000
50%	0.426415	0.518519	0.431034	0.500000	0.500000	0.500000
75%	0.581132	0.679012	0.724138	1.000000	0.500000	1.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

```
In [40]: from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
def print_metrics(y_test, y_pred):
    print(f'R^2: {r2_score(y_test, y_pred)}')
    print(f'MSE: {mean_squared_error(y_test, y_pred)}')
    print(f'MAE: {mean_absolute_error(y_test, y_pred)}')

def print_cv_result(cv_model, x_test, y_test):
    print(f'Optimizacia metrici {cv_model.scoring}: {cv_model.best_score_}')
    print(f'Lyshiy parametr {cv_model.best_params_}')
    print(f'Metricic na testovom nabore')
    print_metrics(y_test, cv_model.predict(x_test))
    print()

base_k = 7
base_knn = KNeighborsRegressor(n_neighbors=base_k)
base_knn.fit(x_train, y_train)
y_pred_base=base_knn.predict(x_test)
print(f'Test metrics for knn with k={base_k}\n')
print_metrics(y_test, y_pred_base)
```

Test metrics for knn with k=7

R^2: 0.8745411551160632  
MSE: 81616.61446756922  
MAE: 224.5049504950495

```
In [43]: from sklearn.model_selection import GridSearchCV
metrics = ['r2', 'neg_mean_squared_error', 'neg_mean_absolute_error']
cv_values=[5, 10]

for cv in cv_values:
    print(f'Rezultati cross validation for cv={cv}\n')
    for metric in metrics:
        params = {'n_neighbors': range(1, 30)}
        knn_cv = GridSearchCV(KNeighborsRegressor(), params, cv=cv, scoring=metric, n_jobs=-1)
        knn_cv.fit(x_train, y_train)
        print_cv_result(knn_cv, x_test, y_test)
```

Rezultati cross validation for cv=5

Optimizacia metrice r2: 0.848412080093014  
 Lyshiy parametr {'n\_neighbors': 23}  
 Metritic na testovom nabore  
 R^2: 0.8543978501831682  
 MSE: 94720.73920530049  
 MAE: 244.28540680154978

Optimizacia metrice neg\_mean\_squared\_error: -92214.18580342011  
 Lyshiy parametr {'n\_neighbors': 23}  
 Metritic na testovom nabore  
 R^2: 0.8543978501831682  
 MSE: 94720.73920530049  
 MAE: 244.28540680154978

Optimizacia metrice neg\_mean\_absolute\_error: -240.6154727908941  
 Lyshiy parametr {'n\_neighbors': 23}  
 Metritic na testovom nabore  
 R^2: 0.8543978501831682  
 MSE: 94720.73920530049  
 MAE: 244.28540680154978

Rezultati cross validation for cv=10

Optimizacia metrice r2: 0.8378099672548694  
 Lyshiy parametr {'n\_neighbors': 27}  
 Metritic na testovom nabore  
 R^2: 0.8535769986835144  
 MSE: 95254.73998017087  
 MAE: 240.2273560689402

Optimizacia metrice neg\_mean\_squared\_error: -92791.22790178127  
 Lyshiy parametr {'n\_neighbors': 27}  
 Metritic na testovom nabore  
 R^2: 0.8535769986835144  
 MSE: 95254.73998017087  
 MAE: 240.2273560689402

Optimizacia metrice neg\_mean\_absolute\_error: -240.6379159957058  
 Lyshiy parametr {'n\_neighbors': 27}  
 Metritic na testovom nabore  
 R^2: 0.8535769986835144  
 MSE: 95254.73998017087  
 MAE: 240.2273560689402

```
In [45]: best_k = 9
y_pred_best = KNeighborsRegressor(n_neighbors=best_k).fit(x_train, y_train).predict(x_test)
```

```
In [46]: #Сравнение исходной и оптимальной модели
print('Исходная модель\n')
print_metrics(y_test, y_pred_base)
print('\nОптимальная модель\n')
print_metrics(y_test, y_pred_best)
```

Исходная модель

R^2: 0.8745411551160632  
 MSE: 81616.61446756922  
 MAE: 224.5049504950495

Оптимальная модель

R^2: 0.8695487532324968  
 MSE: 84864.39616183842  
 MAE: 230.50055005500556