

**Московский государственный технический
университет им. Н.Э. Баумана.**

Факультет «Информатика и управление»

Кафедра «Системы обработки информации и управления»

Отчет по лабораторной работе №3

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

«Курса «Технологии машинного обучения»»

Выполнила:
студентка группы ИУ5-64
Светашева Ю.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 экранные формы

Jupyter Lab_3_Svetasheva_IU5-64B Last Checkpoint: 4 минуты назад (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Run Code

```
In [1]: #импорт библиотек
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
```

```
In [2]: #Подгружаем DataSet
customers_dataset = pd.read_csv('Mall_Customers.csv', encoding='latin-1')
```

```
In [3]: customers_dataset
```

Out[3]:

	CustomerID	Gender	Age	Annual_Income(k\$)	Spending_Score	Married	City	Number_of_sales
0	1	Male	19	15	39	N	Philadelphia	10
1	2	Male	21	15	81	Y	Philadelphia	11
2	3	Female	20	16	6	N	Philadelphia	8
3	4	Female	23	16	77	Y	Philadelphia	7
4	5	Female	31	17	40	Y	Philadelphia	9
...
63	64	Female	67	48	48	N	New York	44
64	65	Male	63	48	51	Y	New York	45
65	66	Male	89	48	59	Y	New York	48
66	67	Female	48	48	50	N	New York	45
67	68	Female	68	48	48	N	New York	44

68 rows x 8 columns

```
In [5]: #стандартные характеристики датасета
customers_dataset.shape
```

Out[5]: (68, 8)

```
In [6]: customers_dataset.dtypes
```


```
Out[6]: CustomerID      int64
Gender      object
Age         int64
Annual_Income(k$)  int64
Spending_Score  int64
Married      object
City         object
Number_of_sales  int64
dtype: object
```

```
In [ ]:
```











```
In [9]: #Очистка нулевых строк
customers_dataset.isnull().sum()
```

```
Out[9]: CustomerID      0
Gender      0
Age         0
Annual_Income(k$)  0
Spending_Score  0
Married     0
City        0
Number_of_sales  0
dtype: int64
```

```
In [ ]: #Строк с нулевыми значениями не найдено
```

 jupyter Lab_3_Svetasheva_IU5-64B Last Checkpoint: 26 минут назад (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted | Pyth

         Code 

```
In [15]: #Кодирование категориальных признаков
customers_dataset["Gender"] = customers_dataset["Gender"].astype('category')
customers_dataset["Married"] = customers_dataset["Married"].astype('category')
customers_dataset["City"] = customers_dataset["City"].astype('category')

#Назначаем закодированный признакам новым столбцам с помощью accessor
customers_dataset["City_cat"] = customers_dataset["City"].cat.codes
customers_dataset["Married_cat"] = customers_dataset["Married"].cat.codes
customers_dataset["Gender_cat"] = customers_dataset["Gender"].cat.codes
customers_dataset
```

```
Out[15]:
```

	CustomerID	Gender	Age	Annual_Income(k\$)	Spending_Score	Married	City	Number_of_sales	City_cat	Married_cat	Gender_cat
0	1	Male	19	15	39	N	Philadelphia	10	3	0	1
1	2	Male	21	15	81	Y	Philadelphia	11	3	1	1
2	3	Female	20	16	6	N	Philadelphia	8	3	0	0
3	4	Female	23	16	77	Y	Philadelphia	7	3	1	0
4	5	Female	31	17	40	Y	Philadelphia	9	3	1	0
...
63	64	Female	67	48	48	N	New York	44	2	0	0
64	65	Male	63	48	51	Y	New York	45	2	1	1
65	66	Male	89	48	59	Y	New York	48	2	1	1
66	67	Female	48	48	50	N	New York	45	2	0	0
67	68	Female	68	48	48	N	New York	44	2	0	0

68 rows x 11 columns

```
In [29]: #Убираем ненужные столбцы, создаем новый датасет
customers_dataset_cat = customers_dataset.drop(["City", 'Married', 'Gender'], axis=1, inplace = True)
customers_dataset
```

```
Out[29]:
```

	CustomerID	Age	Annual_Income(k\$)	Spending_Score	Number_of_sales	City_cat	Married_cat	Gender_cat
0	1	19	15	39	10	3	0	1
1	2	21	15	81	11	3	1	1
2	3	20	16	6	8	3	0	0
3	4	23	16	77	7	3	1	0
4	5	31	17	40	9	3	1	0
...
63	64	67	48	48	44	2	0	0
64	65	63	48	51	45	2	1	1
65	66	89	48	59	48	2	1	1
66	67	48	48	50	45	2	0	0
67	68	68	48	48	44	2	0	0

68 rows x 8 columns

```
In [35]: #Разделение выборки на обучающую и тестовую
from sklearn.model_selection import train_test_split
y = customers_dataset['Number_of_sales']
X = customers_dataset.drop('Number_of_sales', 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[35]:
```

	CustomerID	Age	Annual_Income(k\$)	Spending_Score	City_cat	Married_cat	Gender_cat
18	19	52	23	29	4	1	1
9	10	30	19	72	3	1	0
36	37	42	34	17	0	0	0
64	65	63	48	51	2	1	1
51	52	33	42	60	1	1	1
23	24	31	25	73	4	1	1
53	54	59	43	60	1	1	1
48	49	29	40	42	1	1	0
55	56	47	43	41	1	0	1
46	47	50	40	55	5	0	0
30	31	60	30	4	0	0	1
52	53	31	43	54	1	1	0
15	16	22	20	79	3	1	1
54	55	50	43	45	1	0	0
7	8	23	18	94	3	1	0


```
In [36]: y_train
```

```
Out[36]: 18    13
          9     10
          36    18
          64    45
          51    30
          23    15
          53    30
          48    31
          55    31
          46    27
          30    20
          52    37
          15     8
          54    32
           7    10
          67    44
          50    36
          33    21
           5     4
          45    31
           1    11
          59    43
           2     8
          22    20
          47    32
          17     7
          26    16
          14     6
          39    25
          29    19
```

```
In [45]: #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[45]:
```

	CustomerID	Age	Annual_Income(k\$)	Spending_Score	City_cat	Married_cat	Gender_cat
count	47.000000	47.000000	47.000000	47.000000	47.000000	47.000000	47.000000
mean	0.510956	0.407529	0.531915	0.496831	0.493617	0.553191	0.468085
std	0.297031	0.316417	0.329406	0.276301	0.327974	0.502538	0.504375
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.276119	0.105769	0.242424	0.313830	0.200000	0.000000	0.000000
50%	0.537313	0.326923	0.575758	0.500000	0.600000	1.000000	0.000000
75%	0.768657	0.644231	0.833333	0.728723	0.800000	1.000000	1.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

 jupyter Lab_3_Svetasheva_IU5-64B Last Checkpoint: час назад (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help

       Run    Code 

```
75% 0.768657 0.644231 0.833333 0.728723
max 1.000000 1.000000 1.000000 1.000000
```

```
In [23]: #Обучение KNN с производным k
```

```
simplefilter('ignore')

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'Оптимизация метрики {cv_model.scoring}: {cv_model.best_score_}')
    print(f'Лучший параметр: {cv_model.best_params_}')
    print('Метрики на тестовом наборе')
    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: 0.8408004377113437
MSE: 34.49757045675413
MAE: 4.73469387755102
```

MAE: 4.73469387755102

```
In [24]: #Кросс валидация
metrics = ['r2', 'neg_mean_squared_error', 'neg_mean_absolute_error']
cv_values = [5, 10]

for cv in cv_values:
    print(f'Результаты кросс-валидации при 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)
```

Результаты кросс-валидации при cv=5

Оптимизация метрики r2: 0.8751043988050631

Лучший параметр: {'n_neighbors': 5}

Метрики на тестовом наборе

R^2: 0.8638676461354932

MSE: 29.49904761904762

MAE: 4.352380952380952

Оптимизация метрики neg_mean_squared_error: -18.308533333333333

Лучший параметр: {'n_neighbors': 5}

Метрики на тестовом наборе

R^2: 0.8638676461354932

MSE: 29.49904761904762

MAE: 4.352380952380952

Оптимизация метрики neg_mean_absolute_error: -3.1373333333333333

Лучший параметр: {'n_neighbors': 5}

Метрики на тестовом наборе

R^2: 0.8638676461354932

MSE: 29.49904761904762

```

Оптимизация метрики neg_mean_absolute_error: -3.137333333333333
Лучший параметр: {'n_neighbors': 5}
Метрики на тестовом наборе
R^2: 0.8638676461354932
MSE: 29.49904761904762
MAE: 4.352380952380952
    
```

Результаты кросс-валидации при cv=10

```

Оптимизация метрики r2: 0.763701265961674
Лучший параметр: {'n_neighbors': 5}
Метрики на тестовом наборе
R^2: 0.8638676461354932
MSE: 29.49904761904762
MAE: 4.352380952380952
    
```

```

Оптимизация метрики neg_mean_squared_error: -19.919599999999996
Лучший параметр: {'n_neighbors': 5}
Метрики на тестовом наборе
R^2: 0.8638676461354932
MSE: 29.49904761904762
MAE: 4.352380952380952
    
```

```

Оптимизация метрики neg_mean_absolute_error: -3.2479999999999998
Лучший параметр: {'n_neighbors': 5}
Метрики на тестовом наборе
R^2: 0.8638676461354932
MSE: 29.49904761904762
MAE: 4.352380952380952
    
```

In [26]: *#Сравнение исходной и оптимальной моделей*

```

print('Basic model\n')
print_metrics(y_test, y_pred_base)
print('_____')
print('\nOptimal model\n')
print_metrics(y_test, y_pred_best)
    
```

Basic model

```

R^2: 0.8408004377113437
MSE: 34.49757045675413
MAE: 4.73469387755102
    
```

Optimal model

```

R^2: 0.8638676461354932
MSE: 29.49904761904762
MAE: 4.352380952380952
    
```

In [27]: *#Визуализация результатов*

```

res = pd.DataFrame({'y_test': y_test, 'y_pred_best': y_pred_best}).sort_values(by='y_test')
res.head()
    
```

Out[27]:

	y_test	y_pred_best
8	1	15.4
11	4	9.4
13	5	8.8
6	6	12.2
16	7	13.8


```
In [28]: plt.figure(figsize=(16, 5))
sns.scatterplot(range(res.shape[0]), res['y_test'], label='actual')
sns.scatterplot(range(res.shape[0]), res['y_pred_best'], label='predicted', alpha=0.6)
plt.ylabel('price')
plt.xlabel('')
plt.title(f'Best KNN model results (k={best_k})')
plt.tick_params(axis='x', bottom=False, labelbottom=False)
plt.show()
```

