Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»

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: | Проверил: |
| студент группы ИУ5-63Б | преподаватель каф. ИУ5 |
| Кащеев Максим | Гапанюк Ю.Е. |

Москва, 2022 г.

**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.

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

Лабораторная работа №3: Подготовка обучающей и тестовой выборки, кросс- валидация и подбор гиперпараметров на примере метода ближайших соседей.

# Подключение библиотек

In [1]:

**import** pandas **as** pd

**import** numpy **as** np

**from** sklearn.model\_selection **import** train\_test\_split, GridSearchCV, RandomizedSearchCV

**from** sklearn.neighbors **import** KNeighborsRegressor

**from** sklearn.preprocessing **import** MinMaxScaler, StandardScaler

**from** matplotlib **import** pyplot **as** plt

**import** seaborn **as** sns

**from** sklearn.metrics **import** mean\_absolute\_error, mean\_squared\_error, r2\_score

**from** warnings **import** simplefilter

simplefilter('ignore')

# Загрузка предобработанного в первой лабе датасета

In [2]:

data **=** pd**.**read\_csv('laptop\_price\_preprocessed.csv')

In [3]:

data**.**head()

Out[3]:

**laptop\_ID Company Product**

**TypeName Inches Ram\_GB OpSys Weight\_kg Price\_euros ScreenType ... ScreenRes Cpu\_type Cpu\_GHz Gpu\_producer Gpu\_model Memory**

* 1. 1 Apple

MacBook Ultrabook 13.3 8 macOS 1.37 1339.69

Pro

Macbook

IPS Panel Retina Display

... 2560x1600

Intel Core 2.3 Intel i5

Intel Core

Iris Plus Graphics

640

HD

* 1. 2 Apple

Air Ultrabook 13.3 8 macOS 1.34 898.94 - ... 1440x900

i5 1.8 Intel

Intel Core

Graphics 6000

HD

**2** 3 HP 250 G6 Notebook 15.6 8 No OS 1.86 575.00 Full HD ... 1920x1080

i5 7200U 2.5 Intel

Graphics

620

MacBook Ultrabook 15.4 16 macOS 1.83 2537.45

|  |  |  |
| --- | --- | --- |
| **3** | 4 | Apple |
| **4** | 5 | Apple |

Pro

MacBook Ultrabook 13.3 8 macOS 1.37 1803.60

Pro

IPS Panel Retina Display

IPS Panel Retina Display

... 2880x1800

... 2560x1600

Intel Core 2.7 AMD i7

Intel Core 3.1 Intel i5

Radeon Pro

455

Iris Plus Graphics

650

5 rows × 22 columns

In [4]:

data**.**info()

<class 'pandas.core.frame.DataFrame'> RangeIndex: 1250 entries, 0 to 1249

Data columns (total 22 columns):

# Column Non-Null Count Dtype

1. laptop\_ID 1250 non-null int64
2. Company 1250 non-null object
3. Product 1250 non-null object
4. TypeName 1250 non-null object
5. Inches 1250 non-null float64
6. Ram\_GB 1250 non-null int64
7. OpSys 1250 non-null object
8. Weight\_kg 1250 non-null float64
9. Price\_euros 1250 non-null float64
10. ScreenType 1250 non-null object
11. ScreenWidth 1250 non-null int64
12. ScreenHeight 1250 non-null int64
13. ScreenRes 1250 non-null object
14. Cpu\_type 1250 non-null object
15. Cpu\_GHz 1250 non-null float64
16. Gpu\_producer 1250 non-null object
17. Gpu\_model 1250 non-null object
18. Memory1\_GB 1250 non-null int64
19. Memory1\_type 1250 non-null object
20. Memory2\_GB 1250 non-null int64
21. Memory2\_type 1250 non-null object
22. Memory2 1250 non-null object dtypes: float64(4), int64(6), object(12) memory usage: 215.0+ KB

# Кодирование категориальных признаков

In [5]:

category\_cols **=** ['Memory1\_type', 'Memory2\_type', 'Company', 'Product', 'TypeName', 'OpSys', 'ScreenType', 'Cpu\_type', 'Gpu\_producer', 'Gpu\_model']

print('Количество уникальных значений\n')

**for** col **in** category\_cols:

print(f'{col}: {data[col]**.**unique()**.**size}')

Количество уникальных значений

Memory1\_type: 4

Memory2\_type: 4

Company: 19

Product: 618

TypeName: 6

OpSys: 9

ScreenType: 21

Cpu\_type: 93

Gpu\_producer: 4

Gpu\_model: 110

In [6]:

remove\_cols **=** ['Product']

In [7]:

**for** col **in** remove\_cols:

category\_cols**.**remove(col)

data **=** pd**.**get\_dummies(data, columns**=**category\_cols)

In [8]:

data**.**drop(remove\_cols, axis**=**1, inplace**=True**)

data**.**drop(['laptop\_ID', 'ScreenRes', 'Memory2'], axis**=**1, inplace**=True**) data**.**describe()

Out[8]:

**Inches Ram\_GB Weight\_kg Price\_euros ScreenWidth ScreenHeight Cpu\_GHz Memory1\_GB Memory2\_GB Memory1\_type\_Flash**

**Storage**

**... Gpu\_model\_Radeon**

**R7 M440**

**Gpu\_**

**count** 1250.000000 1250.000000 1250.000000 1250.000000 1250.000000 1250.000000 1250.000000 1250.000000 1250.000000 1250.000000 ... 1250.000000

**mean** 15.034880 8.443200 2.046152 1132.177480 1897.272000 1072.256000 2.303856 447.180800 174.675200 0.055200 ... 0.002400

**std** 1.416838 5.121929 0.669436 703.965444 491.854703 283.172078 0.502772 367.670259 411.340426 0.228462 ... 0.048951

**min** 10.100000 2.000000 0.690000 174.000000 1366.000000 768.000000 0.900000 8.000000 0.000000 0.000000 ... 0.000000

**25%** 14.000000 4.000000 1.500000 600.425000 1600.000000 900.000000 2.000000 256.000000 0.000000 0.000000 ... 0.000000

**50%** 15.600000 8.000000 2.040000 985.000000 1920.000000 1080.000000 2.500000 256.000000 0.000000 0.000000 ... 0.000000

**75%** 15.600000 8.000000 2.310000 1489.747500 1920.000000 1080.000000 2.700000 512.000000 0.000000 0.000000 ... 0.000000

**max** 18.400000 64.000000 4.700000 6099.000000 3840.000000 2160.000000 3.600000 2048.000000 2048.000000 1.000000 ... 1.000000

8 rows × 279 columns

# Разделение выборки на обучающую и тестовую

In [9]:

y **=** data['Price\_euros']

X **=** data**.**drop('Price\_euros', axis**=**1)

x\_train, x\_test, y\_train, y\_test **=** train\_test\_split(X, y, test\_size**=**0.3, random\_state**=**3)

# Масштабирование данных

In [10]:

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[10]:

**Inches Ram\_GB Weight\_kg ScreenWidth ScreenHeight Cpu\_GHz Memory1\_GB Memory2\_GB Memory1\_type\_Flash**

**Storage**

**Memory1\_type\_HDD ... Gpu\_model\_Radeon G**

**R7 M440**

**count** 875.000000 875.000000 875.000000 875.000000 875.000000 875.000000 875.000000 875.000000 875.000000 875.000000 ... 875.000000

**mean** 0.590967 0.103300 0.337162 0.214886 0.218279 0.518493 0.214407 0.084987 0.056000 0.296000 ... 0.002286

**std** 0.173292 0.085202 0.164937 0.203257 0.207356 0.188520 0.180521 0.199047 0.230053 0.456752 ... 0.047782

**min** 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 ... 0.000000

**25%** 0.469880 0.032258 0.201995 0.094584 0.094828 0.407407 0.118110 0.000000 0.000000 0.000000 ... 0.000000

**50%** 0.662651 0.096774 0.336658 0.223929 0.224138 0.592593 0.118110 0.000000 0.000000 0.000000 ... 0.000000

**75%** 0.662651 0.096774 0.401496 0.223929 0.224138 0.666667 0.244094 0.000000 0.000000 1.000000 ... 0.000000

**max** 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 ... 1.000000

8 rows × 278 columns

# Обучение KNN с произвольным k

In [11]:

**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()

In [12]:

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: 0.7722710180524757

MSE: 98607.68197025849

MAE: 218.48083428571428

# Кросс-валидация

In [13]:

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: 0.7726372857710311 Лучший параметр: {'n\_neighbors': 4}

Метрики на тестовом наборе R^2: 0.7841876602937758

MSE: 93447.72183591667

MAE: 209.1155

Оптимизация метрики neg\_mean\_squared\_error: -116355.60149814286 Лучший параметр: {'n\_neighbors': 4}

Метрики на тестовом наборе R^2: 0.7841876602937758

MSE: 93447.72183591667

MAE: 209.1155

Оптимизация метрики neg\_mean\_absolute\_error: -223.9260514285714 Лучший параметр: {'n\_neighbors': 4}

Метрики на тестовом наборе R^2: 0.7841876602937758

MSE: 93447.72183591667

MAE: 209.1155

Результаты кросс-валидации при cv=10

Оптимизация метрики r2: 0.7792314885077024 Лучший параметр: {'n\_neighbors': 4}

Метрики на тестовом наборе R^2: 0.7841876602937758

MSE: 93447.72183591667

MAE: 209.1155

Оптимизация метрики neg\_mean\_squared\_error: -112797.48419457053 Лучший параметр: {'n\_neighbors': 4}

Метрики на тестовом наборе R^2: 0.7841876602937758

MSE: 93447.72183591667

MAE: 209.1155

Оптимизация метрики neg\_mean\_absolute\_error: -217.6273088100836 Лучший параметр: {'n\_neighbors': 4}

Метрики на тестовом наборе R^2: 0.7841876602937758

MSE: 93447.72183591667

MAE: 209.1155

In [14]:

best\_k **=** 4

y\_pred\_best **=** KNeighborsRegressor(n\_neighbors**=**best\_k)**.**fit(x\_train, y\_train)**.**predict(x\_test)

# Сравнение исходной и оптимальной моделей

In [15]:

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: 0.7722710180524757

MSE: 98607.68197025849

MAE: 218.48083428571428

Optimal model

R^2: 0.7841876602937758

MSE: 93447.72183591667

MAE: 209.1155

# Визуализация результатов оптимальной модели

In [16]:

res **=** pd**.**DataFrame({'y\_test': y\_test, 'y\_pred\_best': y\_pred\_best})**.**sort\_values(by**=**'y\_test') res**.**head()

|  |  |  |
| --- | --- | --- |
| Out[16]: | **y\_test** | **y\_pred\_best** |
| **1191** | 174.0 | 292.0000 |
| **1098** | 196.0 | 267.2250 |
| **31** | 199.0 | 313.2475 |
| **1081** | 209.0 | 325.0000 |
| **1243** | 209.0 | 285.9500 |

In [17]:

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()

