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 №4

«Линейные модели, SVM и деревья решений»

|  |  |
| --- | --- |
| 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. В случае необходимости проведите удаление или заполнение пропусков и кодирование категориальных признаков.
3. С использованием метода train\_test\_split разделите выборку на обучающую и тестовую.
4. Обучите следующие модели:
   * одну из линейных моделей (линейную или полиномиальную регрессию при решении задачи регрессии, логистическую регрессию при решении задачи классификации);
   * SVM;
   * дерево решений.
5. Оцените качество моделей с помощью двух подходящих для задачи метрик. Сравните качество полученных моделей.
6. Постройте график, показывающий важность признаков в дереве решений.
7. Визуализируйте дерево решений или выведите правила дерева решений в текстовом виде.

Лабораторная работа №4: "Линейные модели, SVM и деревья решений".

# Загрузка датасета

In [1]:

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

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

**import** matplotlib.pyplot **as** plt

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

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

**from** sklearn.linear\_model **import** LinearRegression, Lasso, Ridge

**from** sklearn.tree **import** DecisionTreeRegressor, export\_graphviz, export\_text

**from** sklearn.svm **import** SVR

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

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

**from** IPython.display **import** Image

**from** IPython.core.display **import** HTML

In [2]:

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

Out[2]:

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

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

|  |  |  |
| --- | --- | --- |
| 1.8 | Intel | HD  Graphics |
|  |  | 6000 |
| 2.5 | Intel | HD  Graphics |
|  |  | 620 |

Iris Plus Graphics

640

1. 2 Apple

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

i5

Intel Core

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

i5 7200U

1. 4 Apple
2. 5 Apple

MacBook Ultrabook 15.4 16 macOS 1.83 2537.45

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

1. rows × 22 columns

# Кодирование категориальных признаков

In [3]:

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 [4]:

remove\_cols **=** ['Product', 'Gpu\_model', 'Cpu\_type']

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

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

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

In [5]:

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

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

Out[5]:

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

**Storage**

**... ScreenType\_Quad**

**HD+**

**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 × 76 columns

In [6]:

data**.**head()

Out[6]:

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

**Memory1\_type\_Flash ...**

**Storage**

**ScreenType\_Quad**

**HD+**

**ScreenType\_Qua**

**HD+**

**Touchscree**

**0** 13.3 8 1.37 1339.69 2560 1600 2.3 128 0 0 ... 0

**1** 13.3 8 1.34 898.94 1440 900 1.8 128 0 1 ... 0

**2** 15.6 8 1.86 575.00 1920 1080 2.5 256 0 0 ... 0

**3** 15.4 16 1.83 2537.45 2880 1800 2.7 512 0 0 ... 0

**4** 13.3 8 1.37 1803.60 2560 1600 3.1 256 0 0 ... 0

5 rows × 76 columns

# Корреляционный анализ

In [7]:

print('Признаки, имеющие максимальную по модулю корреляцию с ценой ноутбука')

best\_params **=** data**.**corr()['Price\_euros']**.**map(abs)**.**sort\_values(ascending**=False**)[1:] best\_params **=** best\_params[best\_params**.**values **>** 0.35]

best\_params

Out[7]:

In [8]:

Признаки, имеющие максимальную по модулю корреляцию с ценой ноутбука Ram\_GB 0.743141

TypeName\_Notebook 0.555495

ScreenWidth 0.553660

ScreenHeight 0.550213

Memory1\_type\_SSD 0.505318

ScreenType\_- 0.435191

Cpu\_GHz 0.431697

Memory1\_type\_HDD 0.425687

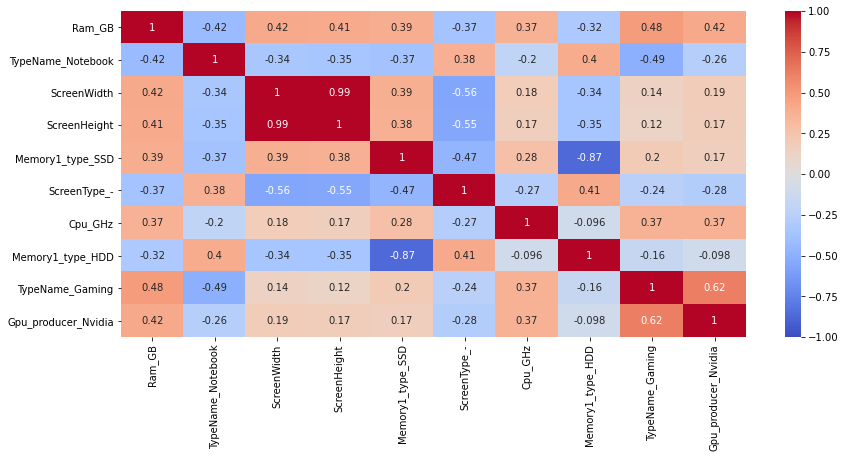
TypeName\_Gaming 0.377151

Gpu\_producer\_Nvidia 0.351031

Name: Price\_euros, dtype: float64

plt**.**figure(figsize**=**(14, 6))

sns**.**heatmap(data[best\_params**.**index]**.**corr(), vmin**=-**1, vmax**=**1, cmap**=**'coolwarm', annot**=True**) plt**.**show()

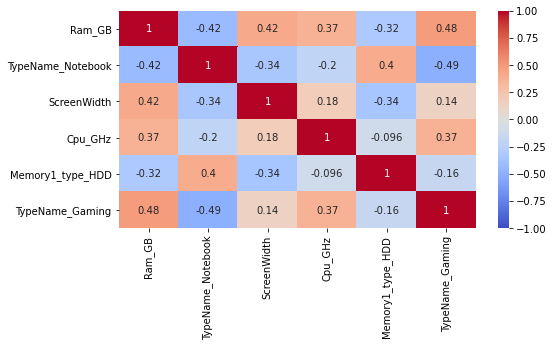


In [9]: best\_params **=** best\_params**.**drop(['ScreenHeight', 'Memory1\_type\_SSD', 'ScreenType\_-', 'Gpu\_producer\_Nvidia']) *#, 'Gpu\_producer\_Nvidia', , 'TypeName\_*

In [10]:

plt**.**figure(figsize**=**(8, 4))

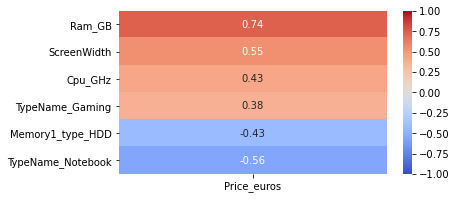
sns**.**heatmap(data[best\_params**.**index]**.**corr(), vmin**=-**1, vmax**=**1, cmap**=**'coolwarm', annot**=True**) plt**.**show()



In [11]:

plt**.**figure(figsize**=**(6, 3))

sns**.**heatmap(pd**.**DataFrame(data[np**.**append(best\_params**.**index**.**values, 'Price\_euros')]**.**corr()['Price\_euros']**.**sort\_values(ascending**=False**)[1:]), vmin**=-**1 plt**.**show()



# Разделение выборки на обучающую и тестовую

In [12]:

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

X **=** data[best\_params**.**index]

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

# Линейная регрессия

In [13]:

**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)}")

In [14]:

linear\_model **=** LinearRegression() linear\_model**.**fit(x\_train, y\_train)

y\_pred\_linear **=** linear\_model**.**predict(x\_test)

print\_metrics(y\_test, y\_pred\_linear)

R^2: 0.6770754057292072

MSE: 139827.8138334042

MAE: 275.17093485469525

# Пополиномиальная регрессия

In [15]:

poly\_model **=** PolynomialFeatures(degree**=**3)

x\_train\_poly **=** poly\_model**.**fit\_transform(x\_train) x\_test\_poly **=** poly\_model**.**fit\_transform(x\_test)

linear\_model **=** LinearRegression()

linear\_model**.**fit(x\_train\_poly, y\_train)

y\_pred\_poly **=** linear\_model**.**predict(x\_test\_poly) print\_metrics(y\_test, y\_pred\_poly)

R^2: 0.679477609592945

MSE: 138787.64866603096

MAE: 264.41254443760585

# SVM

In [16]:

scaler **=** StandardScaler()**.**fit(x\_train)

x\_train\_scaled **=** pd**.**DataFrame(scaler**.**transform(x\_train), columns**=**x\_train**.**columns) x\_test\_scaled **=** pd**.**DataFrame(scaler**.**transform(x\_test), columns**=**x\_train**.**columns)

x\_train\_scaled**.**describe()

Out[16]:

In [17]:

**Ram\_GB TypeName\_Notebook ScreenWidth Cpu\_GHz Memory1\_type\_HDD TypeName\_Gaming count** 8.750000e+02 8.750000e+02 8.750000e+02 8.750000e+02 8.750000e+02 8.750000e+02 **mean** 5.468641e-17 1.715453e-16 8.263231e-18 -6.127302e-16 -4.491645e-17 -9.478133e-17 **std** 1.000572e+00 1.000572e+00 1.000572e+00 1.000572e+00 1.000572e+00 1.000572e+00

**min** -1.213107e+00 -1.136035e+00 -1.057818e+00 -2.751903e+00 -6.484247e-01 -4.401414e-01

**25%** -8.342815e-01 -1.136035e+00 -5.922109e-01 -5.895875e-01 -6.484247e-01 -4.401414e-01

**50%** -7.663095e-02 8.802544e-01 4.451633e-02 3.932830e-01 -6.484247e-01 -4.401414e-01

**75%** -7.663095e-02 8.802544e-01 4.451633e-02 7.864312e-01 1.542199e+00 -4.401414e-01

**max** 1.053048e+01 8.802544e-01 3.864880e+00 2.555598e+00 1.542199e+00 2.271997e+00

{'C': 11.0}

params **=** {'C': np**.**concatenate([np**.**arange(0.1, 2, 0.1), np**.**arange(2, 15, 1)])} svm\_model **=** SVR(kernel**=**'linear')

grid\_cv **=** GridSearchCV(estimator**=**svm\_model, param\_grid**=**params, cv**=**10, n\_jobs**=-**1, scoring**=**'r2') grid\_cv**.**fit(x\_train\_scaled, y\_train)

print(grid\_cv**.**best\_params\_)

In [18]:

best\_svm\_model **=** grid\_cv**.**best\_estimator\_

best\_svm\_model **=** SVR(kernel**=**'linear', C**=**11) best\_svm\_model**.**fit(x\_train\_scaled, y\_train)

y\_pred\_svm **=** best\_svm\_model**.**predict(x\_test\_scaled) print\_metrics(y\_test, y\_pred\_svm)

R^2: 0.6670420012232607

MSE: 144172.32348755666

MAE: 272.96301929727315

# Дерево решений

In [19]:

params **=** {'min\_samples\_leaf': range(3, 30)} tree **=** DecisionTreeRegressor(random\_state**=**3)

grid\_cv **=** GridSearchCV(estimator**=**tree, cv**=**5, param\_grid**=**params, n\_jobs**=-**1, scoring**=**'neg\_mean\_absolute\_error') grid\_cv**.**fit(x\_train, y\_train)

print(grid\_cv**.**best\_params\_)

{'min\_samples\_leaf': 5}

In [20]:

best\_tree **=** grid\_cv**.**best\_estimator\_

best\_tree**.**fit(x\_train, y\_train)

y\_pred\_tree **=** best\_tree**.**predict(x\_test) print\_metrics(y\_test, y\_pred\_tree)

R^2: 0.6842074880923352

MSE: 136739.5898250473

MAE: 258.4878614139572

In [21]:

importances **=** pd**.**DataFrame(data**=**zip(x\_train**.**columns, best\_tree**.**feature\_importances\_), columns**=**['Признак', 'Важность']) print('Важность признаков в дереве решений\n')

**for** row **in** importances**.**sort\_values(by**=**'Важность', ascending**=False**)**.**values: print(f'{row[0]}: {round(row[1], 3)}')

Важность признаков в дереве решений

Ram\_GB: 0.746

Cpu\_GHz: 0.121

TypeName\_Notebook: 0.073

ScreenWidth: 0.03

Memory1\_type\_HDD: 0.017

TypeName\_Gaming: 0.014

In [22]:

plt**.**figure(figsize**=**(12, 4))

sns**.**barplot(data**=**importances**.**sort\_values(by**=**'Важность', ascending**=False**), y**=**'Признак', x**=**'Важность', orient**=**'h', ) plt**.**title('Важность признаков в дереве решений')

plt**.**show()



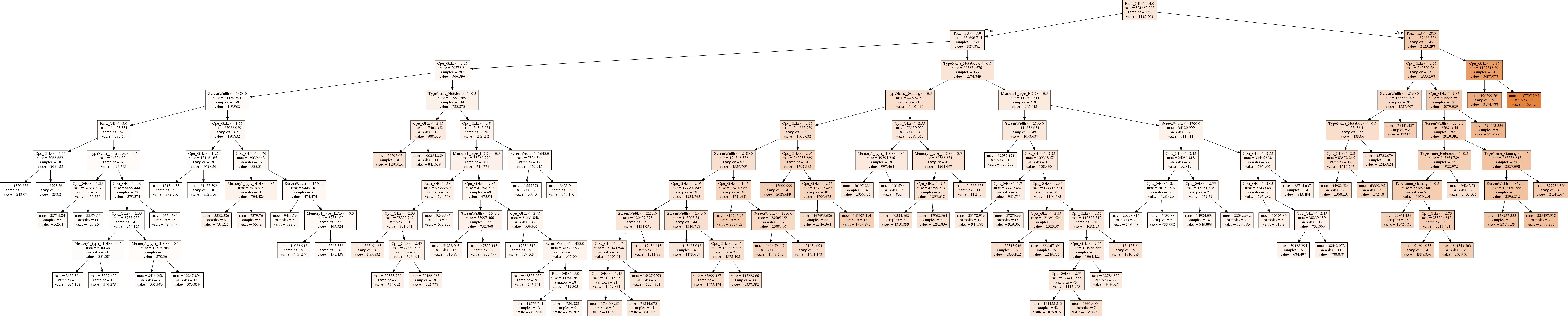
In [23]:

export\_graphviz(best\_tree, feature\_names**=**best\_params**.**index, filled**=True**, out\_file**=**'tree.dot')

**!**dot -Tpng tree.dot -o tree.png Image(filename**=**'tree.png')

Out[23]:

In [24]:



# Сравнение моделей

print('Линейная регрессия')

print\_metrics(y\_test, y\_pred\_linear)

print('\nПолиномиальная регрессия') print\_metrics(y\_test, y\_pred\_poly)

print('\nМетод опорных векторов') print\_metrics(y\_test, y\_pred\_svm)

print('\nДерево решений')

print\_metrics(y\_test, y\_pred\_tree)

Линейная регрессия

R^2: 0.6770754057292072

MSE: 139827.8138334042

MAE: 275.17093485469525

Полиномиальная регрессия R^2: 0.679477609592945

MSE: 138787.64866603096

MAE: 264.41254443760585

Метод опорных векторов R^2: 0.6670420012232607

MSE: 144172.32348755666

MAE: 272.96301929727315

Дерево решений

R^2: 0.6842074880923352

MSE: 136739.5898250473

MAE: 258.4878614139572