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 №5

«Ансамбли моделей машинного обучения»

|  |  |
| --- | --- |
| 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 регресии.
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 ансамблевые модели:
   * одну из моделей группы бэггинга (бэггинг или случайный лес или сверхслучайные деревья);
   * одну из моделей группы бустинга;
   * одну из моделей группы стекинга.
5. **(+1 балл на экзамене)** Дополнительно к указанным моделям обучите еще две модели:
   * Модель [многослойного персептрона.](https://scikit-learn.org/stable/modules/neural_networks_supervised.html) По желанию, вместо библиотеки scikit-learn возможно использование

библиотек [TensorFlow](https://www.tensorflow.org/), [PyTorch](https://pytorch.org/) или других аналогичных библиотек.

* + Модель МГУА с использованием библиотеки

- <https://github.com/kvoyager/GmdhPy>(или аналогичных библиотек). Найдите такие параметры запуска модели, при которых она будет по крайней мере не хуже, чем одна из предыдущих ансамблевых

моделей.

1. Оцените качество моделей с помощью одной из подходящих для задачи метрик. Сравните качество полученных моделей.

In [37]:

Лабораторная работа №5: "Ансамбли моделей машинного обучения".

# Загрузка датасета

**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

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

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

**from** sklearn.ensemble **import** RandomForestRegressor, GradientBoostingRegressor

**from** heamy.estimator **import** Regressor

**from** heamy.pipeline **import** ModelsPipeline

**from** heamy.dataset **import** Dataset

**from** sklearn.neural\_network **import** MLPRegressor

**from** gmdhpy **import** gmdh

**from** warnings **import** simplefilter simplefilter('ignore')

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 Memory1\_GB Memory**

1. 1 Apple
2. 2 Apple

MacBook

Pro

Macbook

Air

IPS Panel

Intel Core 2.3 Intel i5

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Ultrabook | 13.3 | 8 | macOS | 1.37 | 1339.69 | Retina ... 2560x1600 Display |
| Ultrabook | 13.3 | 8 | macOS | 1.34 | 898.94 | - ... 1440x900 |
| Notebook | 15.6 | 8 | No OS | 1.86 | 575.00 | Full HD ... 1920x1080 |
| Ultrabook | 15.4 | 16 | macOS | 1.83 | 2537.45 | IPS Panel  Retina ... 2880x1800 Display |
| Ultrabook | 13.3 | 8 | macOS | 1.37 | 1803.60 | IPS Panel  Retina ... 2560x1600 |

Intel Core 1.8 Intel i5

Intel Core

Iris Plus Graphics

640

HD

Graphics 6000

HD

128

128 Flash S

**2** 3 HP 250 G6

MacBook

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

i5 7200U 2.5 Intel

Intel Core

Graphics

620

Radeon Pro

256

Pro

MacBook

i7 2.7 AMD

Intel Core

455 512

Iris Plus

Pro

Display

i5 3.1 Intel

Graphics

650

256

5 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**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | | | | | **Touchscreen** |
| **count** | 1250.000000 | 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 | 0.012000 |
| **std** | 1.416838 | 5.121929 | 0.669436 | 703.965444 | 491.854703 | 283.172078 | 0.502772 | 367.670259 | 411.340426 | 0.228462 ... 0.048951 | 0.108929 |
| **min** | 10.100000 | 2.000000 | 0.690000 | 174.000000 | 1366.000000 | 768.000000 | 0.900000 | 8.000000 | 0.000000 | 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 | 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 | 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 | 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 | 1.000000 |

**ScreenType\_Quad**

**HD+**

**ScreenType\_Quad**

**HD+ /**

**Scre**

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\_Quad**

**HD+ /**

**Touchscreen**

**ScreenType\_Touchscree**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 13.3 | 8 | 1.37 | 1339.69 | 2560 | 1600 | 2.3 | 128 | 0 | 0 ... 0 | 0 |
| **1** | 13.3 | 8 | 1.34 | 898.94 | 1440 | 900 | 1.8 | 128 | 0 | 1 ... 0 | 0 |
| **2** | 15.6 | 8 | 1.86 | 575.00 | 1920 | 1080 | 2.5 | 256 | 0 | 0 ... 0 | 0 |
| **3** | 15.4 | 16 | 1.83 | 2537.45 | 2880 | 1800 | 2.7 | 512 | 0 | 0 ... 0 | 0 |
| **4** | 13.3 | 8 | 1.37 | 1803.60 | 2560 | 1600 | 3.1 | 256 | 0 | 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

Признаки, имеющие максимальную по модулю корреляцию с ценой ноутбука Out[7]: Ram\_GB 0.743141

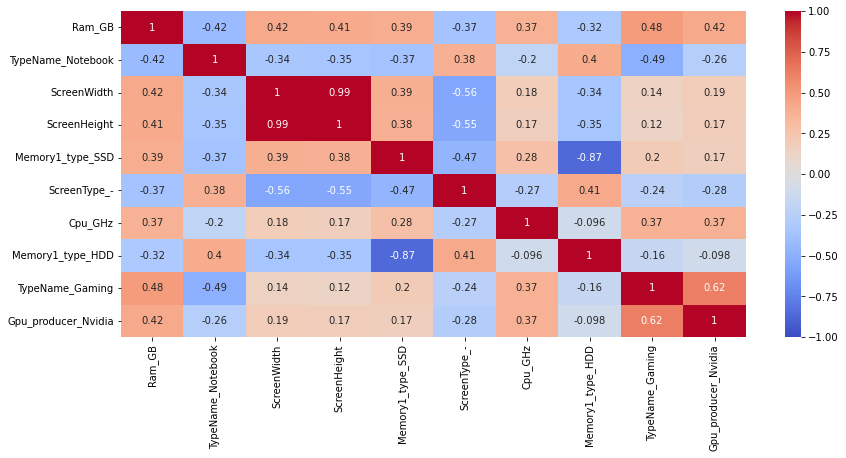
|  |  |
| --- | --- |
| TypeName\_Notebook | 0.555495 |
| ScreenWidth | 0.553660 |
| ScreenHeight | 0.550213 |
| Memory1\_type\_SSD | 0.505318 |
| ScreenType\_- | 0.435191 |
| Cpu\_GHz | 0.431697 |
| Memory1\_type\_HDD | 0.425687 |
| TypeName\_Gaming | 0.377151 |
| Gpu\_producer\_Nvidia | 0.351031 |

Name: Price\_euros, dtype: float64

In [8]:

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

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



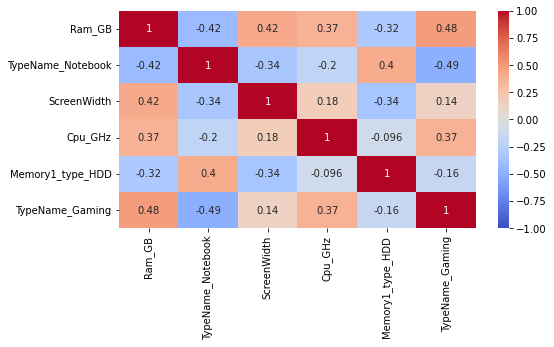
In [9]:

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

In [10]:

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

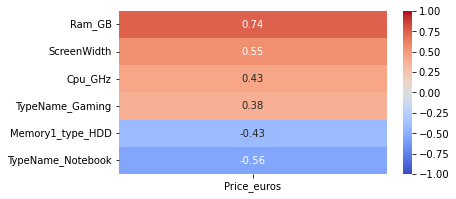
sns**.**heatmap(data[best\_params**.**index]**.**corr(), vmin**=-**1, vmax**=**1, cmap**=**'coolwarm', annot**=True**) plt**.**show()



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, vmax**=**1, cmap**=**'cool plt**.**show()



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)

# Разделение выборки на обучающую и тестовую

In [ ]:

# Масштабирование данных

Метрики

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)

In [39]:

**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)}")

# Модель №1: Случайный лес

In [40]:

print\_metrics(y\_test, RandomForestRegressor(random\_state**=**17)**.**fit(x\_train, y\_train)**.**predict(x\_test))

In [14]:

R^2: 0.6842978189655353

MSE: 136700.47614728688

MAE: 253.06780287849978

## Подбор гиперпараметров

rf **=** RandomForestRegressor(random\_state**=**17)

params **=** {'n\_estimators': [100, 1000], 'criterion': ['squared\_error', 'absolute\_error', 'poisson'], 'max\_features': ['auto', 'sqrt'], 'min\_samples\_leaf': [1, 3, 5]}

grid\_cv **=** GridSearchCV(estimator**=**rf, 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\_)

{'criterion': 'absolute\_error', 'max\_features': 'auto', 'min\_samples\_leaf': 1, 'n\_estimators': 1000}

In [15]:

best\_rf **=** grid\_cv**.**best\_estimator\_ best\_rf**.**fit(x\_train, y\_train)

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

R^2: 0.6898203012827298

MSE: 134309.21625861025

MAE: 252.41492530666685

# Модель №2: Градиентный бустинг

In [41]:

print\_metrics(y\_test, GradientBoostingRegressor(random\_state**=**17)**.**fit(x\_train, y\_train)**.**predict(x\_test))

In [17]:

R^2: 0.6940296765051565

MSE: 132486.53769711652

MAE: 258.8073464450384

## Подбор гиперпараметров

gb **=** GradientBoostingRegressor(random\_state**=**17)

params **=** {'loss': ['squared\_error', 'absolute\_error', 'huber'], 'n\_estimators': [10, 50, 100, 200],

'criterion': ['friedman\_mse', 'squared\_error', 'mse', 'mae'], 'min\_samples\_leaf': [1, 3, 5]} grid\_cv **=** GridSearchCV(estimator**=**gb, cv**=**5, param\_grid**=**params, n\_jobs**=-**1, scoring**=**'r2')

grid\_cv**.**fit(x\_train, y\_train) print(grid\_cv**.**best\_params\_)

{'criterion': 'friedman\_mse', 'loss': 'huber', 'min\_samples\_leaf': 1, 'n\_estimators': 100}

In [18]:

best\_gb **=** grid\_cv**.**best\_estimator\_ best\_gb**.**fit(x\_train, y\_train)

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

R^2: 0.7013333844767404

MSE: 129323.99902194891

MAE: 253.7859718910538

# Модель №3: Стекинг

In [24]:

dataset **=** Dataset(x\_train, y\_train, x\_test)

In [27]:

model\_lr **=** Regressor(dataset**=**dataset, estimator**=**LinearRegression, name**=**'lr') model\_rf **=** Regressor(dataset**=**dataset, estimator**=**RandomForestRegressor,

parameters**=**{'criterion': 'absolute\_error', 'n\_estimators': 1000, 'random\_state': 17}, name**=**'rf') model\_gb **=** Regressor(dataset**=**dataset, estimator**=**GradientBoostingRegressor,

parameters**=**{'loss': 'huber', 'random\_state': 17}, name**=**'rf')

In [28]:

pipeline **=** ModelsPipeline(model\_lr, model\_rf) stack\_ds **=** pipeline**.**stack(k**=**10, seed**=**1)

stacker **=** Regressor(dataset**=**stack\_ds, estimator**=**GradientBoostingRegressor) results **=** stacker**.**validate(k**=**10, scorer**=**mean\_absolute\_error)

Metric: mean\_absolute\_error

Folds accuracy: [191.39424868214826, 223.5971668487191, 216.86376824238184, 272.97770520828004, 275.15017993431206, 225.94857678271197, 236.4669684146994, 268.20677122

61299, 198.22714693578052, 262.2396119721167]

Mean accuracy: 237.10721442472794

Standard Deviation: 29.413190351082335

Variance: 865.1357666290029

In [29]:

y\_pred\_stack **=** stacker**.**predict()

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

R^2: 0.7207185369761542

MSE: 120930.14007496767

MAE: 247.18161038788267

# Модель №4: Многослойный персептрон

In [48]:

print\_metrics(y\_test, MLPRegressor(random\_state**=**17)**.**fit(x\_train, y\_train)**.**predict(x\_test))

In [52]:

R^2: 0.3933464482443907

MSE: 262683.73918006354

MAE: 406.8932580917785

## Подбор гиперпараметров

mlp **=** MLPRegressor(random\_state**=**17)

params **=** {'solver': ['lbfgs', 'sgd', 'adam'], 'hidden\_layer\_sizes': [(100,), (50, 30,), (100, 40,)], 'alpha': [1e-4, 3e-4, 5e-4], 'max\_iter': [500, 1000]}

grid\_cv **=** GridSearchCV(estimator**=**mlp, cv**=**5, param\_grid**=**params, n\_jobs**=-**1, scoring**=**'r2') grid\_cv**.**fit(x\_train, y\_train)

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

{'alpha': 0.0003, 'hidden\_layer\_sizes': (50, 30), 'max\_iter': 500, 'solver': 'lbfgs'}

In [53]:

best\_mlp **=** grid\_cv**.**best\_estimator\_ best\_mlp**.**fit(x\_train, y\_train)

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

R^2: 0.6422646017371612

MSE: 154901.0498344665

MAE: 288.659695272951

# Модель №5: Метод группового учёта аргументов

In [35]:

gm **=** gmdh**.**Regressor(n\_jobs**=-**1)

gm**.**fit(np**.**array(x\_train\_scaled), np**.**array(y\_train)) y\_pred\_gm **=** gm**.**predict(np**.**array(x\_test\_scaled))

print()

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

train layer0 in 0.01 sec train layer1 in 0.05 sec train layer2 in 0.04 sec train layer3 in 0.05 sec train layer4 in 0.04 sec train layer5 in 0.05 sec train layer6 in 0.04 sec train layer7 in 0.04 sec train layer8 in 0.03 sec

R^2: 0.6642449299187112

MSE: 145383.4680475877

MAE: 274.30940411915725

# Сравнение моделей

In [54]:

print("Случайный лес")

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

print("\nГрадиентный бустинг") print\_metrics(y\_test, y\_pred\_gb)

print("\nСтекинг")

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

print("\nМногослойный персептрон") print\_metrics(y\_test, y\_pred\_mlp)

print("\nМетод группового учёта аргументов") print\_metrics(y\_test, y\_pred\_gm)

Случайный лес

R^2: 0.6898203012827298

MSE: 134309.21625861025

MAE: 252.41492530666685

Градиентный бустинг

R^2: 0.7013333844767404

MSE: 129323.99902194891

MAE: 253.7859718910538

Стекинг

R^2: 0.7207185369761542

MSE: 120930.14007496767

MAE: 247.18161038788267

Многослойный персептрон R^2: 0.6422646017371612

MSE: 154901.0498344665

MAE: 288.659695272951

Метод группового учёта аргументов R^2: 0.6642449299187112

MSE: 145383.4680475877

MAE: 274.30940411915725