



**Министерство науки и высшего образования
Российской Федерации 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 системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»**

Курс «Технологии машинного обучения»

Лабораторная работа № 5-6

Выполнил:
студент группы ИУ5-63Б
Попов С. Д.

Проверил:
Гапанюк Ю. Е.

2024 г.

✓ Лабораторная работа

✓ Выберите набор данных (датасет) для решения задачи классификации или регрессии.

В случае необходимости проведите удаление или заполнение пропусков и кодирование категориальных признаков. С использованием метода `train_test_split` разделите выборку на обучающую и тестовую. Обучите следующие ансамблевые модели:

- две модели группы бэггинга (бэггинг или случайный лес или сверхслучайные деревья);
- AdaBoost;
- градиентный бустинг.
- одну из моделей группы стекинга.
- модель многослойного персептрона. По желанию, вместо библиотеки `scikit-learn` возможно использование библиотек TensorFlow, PyTorch или других аналогичных библиотек.
- двумя методами на выбор из семейства МГУА (один из линейных методов COMBI / MULTI + один из нелинейных методов MIA / RIA) с использованием библиотеки `gmdh`.

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

```
!pip install heamy
!pip install gmdh
```


```
Collecting heamy
  Downloading heamy-0.0.7.tar.gz (30 kB)
  Preparing metadata (setup.py) ... done
Requirement already satisfied: scikit-learn>=0.17.0 in /usr/local/lib/python3.10/dist-packages (from heamy) (1.2.2)
Requirement already satisfied: pandas>=0.17.0 in /usr/local/lib/python3.10/dist-packages (from heamy) (2.0.3)
Requirement already satisfied: six>=1.10.0 in /usr/local/lib/python3.10/dist-packages (from heamy) (1.16.0)
Requirement already satisfied: scipy>=0.16.0 in /usr/local/lib/python3.10/dist-packages (from heamy) (1.11.4)
Requirement already satisfied: numpy>=1.7.0 in /usr/local/lib/python3.10/dist-packages (from heamy) (1.25.2)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.17.0->heamy) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.17.0->heamy) (2020.1)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.17.0->heamy) (2022.1)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.17.0->heamy) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.17.0->heamy) (3.2.0)
Building wheels for collected packages: heamy
  Building wheel for heamy (setup.py) ... done
  Created wheel for heamy: filename=heamy-0.0.7-py2.py3-none-any.whl size=15347 sha256=438912c8ffaabef6240cf080ff7bd7e06
  Stored in directory: /root/.cache/pip/wheels/e5/e4/9a/bc85119b96421369998ff0f53c0854b57bfb518c460fe8c5de
Successfully built heamy
Installing collected packages: heamy
Successfully installed heamy-0.0.7
Collecting gmdh
  Downloading gmdh-1.0.3-cp310-cp310-manylinux1_x86_64.whl (875 kB)
  875.3/875.3 kB 4.9 MB/s eta 0:00:00
Collecting docstring-inheritance (from gmdh)
  Downloading docstring_inheritance-2.2.0-py3-none-any.whl (24 kB)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from docstring-inheritance (from gmdh)) (1.25.2)
Installing collected packages: docstring-inheritance, gmdh
Successfully installed docstring-inheritance-2.2.0 gmdh-1.0.3
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import gmdh
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error
from sklearn.datasets import *
from heamy.dataset import Dataset
from heamy.estimator import Regressor, Classifier
from heamy.pipeline import ModelsPipeline
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier, StackingClassifier, GradientBoostingClassifier
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error, r2_score
```

```
<frozen importlib._bootstrap>:914: ImportWarning: _PyDrive2ImportHook.find_spec() not found; falling back to find_module
<frozen importlib._bootstrap>:914: ImportWarning: _PyDriveImportHook.find_spec() not found; falling back to find_module
<frozen importlib._bootstrap>:914: ImportWarning: _GenerativeAIImportHook.find_spec() not found; falling back to find_module
<frozen importlib._bootstrap>:914: ImportWarning: _OpenCVImportHook.find_spec() not found; falling back to find_module
<frozen importlib._bootstrap>:914: ImportWarning: _APICoreClientInfoImportHook.find_spec() not found; falling back to find_module
<frozen importlib._bootstrap>:914: ImportWarning: _BokehImportHook.find_spec() not found; falling back to find_module
<frozen importlib._bootstrap>:914: ImportWarning: _AltairImportHook.find_spec() not found; falling back to find_module
<frozen importlib._bootstrap>:914: ImportWarning: _PyDrive2ImportHook.find_spec() not found; falling back to find_module
<frozen importlib._bootstrap>:914: ImportWarning: _PyDriveImportHook.find_spec() not found; falling back to find_module
```


```
<frozen importlib._bootstrap>:914: ImportWarning: _GenerativeAIImportHook.find_spec() not found; falling back to find_mo
<frozen importlib._bootstrap>:914: ImportWarning: _OpenCVImportHook.find_spec() not found; falling back to find_module()
<frozen importlib._bootstrap>:914: ImportWarning: APICoreClientInfoImportHook.find_spec() not found; falling back to fin
<frozen importlib._bootstrap>:914: ImportWarning: _BokehImportHook.find_spec() not found; falling back to find_module()
<frozen importlib._bootstrap>:914: ImportWarning: _AltairImportHook.find_spec() not found; falling back to find_module()
<frozen importlib._bootstrap>:914: ImportWarning: _PyDrive2ImportHook.find_spec() not found; falling back to find_module
<frozen importlib._bootstrap>:914: ImportWarning: _PyDriveImportHook.find_spec() not found; falling back to find_module(
<frozen importlib._bootstrap>:914: ImportWarning: _GenerativeAIImportHook.find_spec() not found; falling back to find_mo
<frozen importlib._bootstrap>:914: ImportWarning: _OpenCVImportHook.find_spec() not found; falling back to find_module()
<frozen importlib._bootstrap>:914: ImportWarning: APICoreClientInfoImportHook.find_spec() not found; falling back to fin
<frozen importlib._bootstrap>:914: ImportWarning: _BokehImportHook.find_spec() not found; falling back to find_module()
<frozen importlib._bootstrap>:914: ImportWarning: _AltairImportHook.find_spec() not found; falling back to find_module()
<frozen importlib._bootstrap>:914: ImportWarning: _PyDrive2ImportHook.find_spec() not found; falling back to find_module
<frozen importlib._bootstrap>:914: ImportWarning: _PyDriveImportHook.find_spec() not found; falling back to find_module(
<frozen importlib._bootstrap>:914: ImportWarning: _GenerativeAIImportHook.find_spec() not found; falling back to find_mo
<frozen importlib._bootstrap>:914: ImportWarning: _OpenCVImportHook.find_spec() not found; falling back to find_module()
<frozen importlib._bootstrap>:914: ImportWarning: APICoreClientInfoImportHook.find_spec() not found; falling back to fin
<frozen importlib._bootstrap>:914: ImportWarning: _BokehImportHook.find_spec() not found; falling back to find_module()
<frozen importlib._bootstrap>:914: ImportWarning: _AltairImportHook.find_spec() not found; falling back to find_module()
<frozen importlib._bootstrap>:914: ImportWarning: _PyDrive2ImportHook.find_spec() not found; falling back to find_module
<frozen importlib._bootstrap>:914: ImportWarning: _PyDriveImportHook.find_spec() not found; falling back to find_module(
<frozen importlib._bootstrap>:914: ImportWarning: _GenerativeAIImportHook.find_spec() not found; falling back to find_mo
<frozen importlib._bootstrap>:914: ImportWarning: _OpenCVImportHook.find_spec() not found; falling back to find_module()
<frozen importlib._bootstrap>:914: ImportWarning: APICoreClientInfoImportHook.find_spec() not found; falling back to fin
<frozen importlib._bootstrap>:914: ImportWarning: _BokehImportHook.find_spec() not found; falling back to find_module()
<frozen importlib._bootstrap>:914: ImportWarning: _AltairImportHook.find_spec() not found; falling back to find_module()
<frozen importlib._bootstrap>:914: ImportWarning: _PyDrive2ImportHook.find_spec() not found; falling back to find_module
<frozen importlib._bootstrap>:914: ImportWarning: _PyDriveImportHook.find_spec() not found; falling back to find_module(
<frozen importlib._bootstrap>:914: ImportWarning: _GenerativeAIImportHook.find_spec() not found; falling back to find_mo
<frozen importlib._bootstrap>:914: ImportWarning: _OpenCVImportHook.find_spec() not found; falling back to find_module()
<frozen importlib._bootstrap>:914: ImportWarning: APICoreClientInfoImportHook.find_spec() not found; falling back to fin
<frozen importlib._bootstrap>:914: ImportWarning: _BokehImportHook.find_spec() not found; falling back to find_module()
<frozen importlib._bootstrap>:914: ImportWarning: _AltairImportHook.find_spec() not found; falling back to find_module()
<frozen importlib._bootstrap>:914: ImportWarning: _PyDrive2ImportHook.find_spec() not found; falling back to find_module
<frozen importlib._bootstrap>:914: ImportWarning: _PyDriveImportHook.find_spec() not found; falling back to find_module(
```

```
import warnings
warnings.filterwarnings('ignore')
```

```
 /usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call
and should_run_async(code)
```

```
df = pd.read_csv('cleaned_all_phones.csv')
```


```
df.head()
```



	phone_name	brand	os	inches	resolution	battery	battery_type	ram(GB)
0	Y6II Compact	Huawei	Android 5.1	5.0	720x1280	2200	Li-Po	
1	K20 plus	LG	Android 7.0	5.3	720x1280	2700	Li-Ion	
2	P8 Lite (2017)	Huawei	Android 7.0	5.2	1080x1920	3000	Li-Ion	
3	Redmi Note 4	Xiaomi	Android 6.0	5.5	1080x1920	4100	Li-Po	
4	P10	Huawei	Android 7.0	5.1	1080x1920	3200	Li-Ion	

5 rows × 22 columns


```
df.tail()
```



	phone_name	brand	os	inches	resolution	battery	battery_type	ram
1507	vivo Y77t	Vivo	Android 13	6.64	1080x2388	5000	Li-Po	
1508	11x	Realme	Android 13	6.72	1080x2400	5000	Li-Po	
1509	GT5	Realme	Android 13	6.74	1240x2772	5240	Li-Po	
1510	GT5 240W	Realme	Android 13	6.74	1240x2772	4600	Li-Po	
1511	vivo iQOO Z7 Pro	Vivo	Android 13	6.78	1080x2400	4600	Li-Po	


5 rows x 22 columns

df.shape




(1512, 22)

df.info()



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1512 entries, 0 to 1511
Data columns (total 22 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   phone_name                            1512 non-null   object
1   brand                                1512 non-null   object
2   os                                    1512 non-null   object
3   inches                               1512 non-null   float64
4   resolution                            1512 non-null   object
5   battery                              1512 non-null   int64
6   battery_type                          1512 non-null   object
7   ram(GB)                              1512 non-null   int64
8   announcement_date                    1512 non-null   object
9   weight(g)                            1512 non-null   float64
10  storage(GB)                          1512 non-null   int64
11  video_720p                           1512 non-null   bool
12  video_1080p                           1512 non-null   bool
13  video_4K                              1512 non-null   bool
14  video_8K                              1512 non-null   bool
15  video_30fps                           1512 non-null   bool
16  video_60fps                           1512 non-null   bool
17  video_120fps                           1512 non-null   bool
18  video_240fps                           1512 non-null   bool
19  video_480fps                           1512 non-null   bool
20  video_960fps                           1512 non-null   bool
21  price(USD)                            1512 non-null   float64
dtypes: bool(10), float64(3), int64(3), object(6)
memory usage: 156.6+ KB
```

df.describe()




	inches	battery	ram(GB)	weight(g)	storage(GB)	price(USD)
count	1512.000000	1512.000000	1512.000000	1512.000000	1512.000000	1512.000000
mean	6.422460	4389.798942	6.683862	187.636243	109.164683	337.847036
std	0.477043	784.607022	2.701433	26.200115	74.436484	266.740821
min	3.800000	1821.000000	1.000000	130.000000	1.000000	40.000000
25%	6.300000	4000.000000	4.000000	175.000000	64.000000	179.997500
50%	6.500000	4500.000000	8.000000	187.000000	128.000000	260.000000
75%	6.670000	5000.000000	8.000000	197.250000	128.000000	400.000000
max	10.400000	7250.000000	24.000000	500.000000	512.000000	2300.000000

df.duplicated().sum()



0

df.isna().sum()



phone_name	0
brand	0
os	0
inches	0

```

resolution      0
battery         0
battery_type    0
ram(GB)         0
announcement_date 0
weight(g)       0
storage(GB)     0
video_720p      0
video_1080p     0
video_4K        0
video_8K        0
video_30fps     0
video_60fps     0
video_120fps    0
video_240fps    0
video_480fps    0
video_960fps    0
price(USD)      0
dtype: int64

```

```
df.columns
```

```

Index(['phone_name', 'brand', 'os', 'inches', 'resolution', 'battery',
      'battery_type', 'ram(GB)', 'announcement_date', 'weight(g)',
      'storage(GB)', 'video_720p', 'video_1080p', 'video_4K', 'video_8K',
      'video_30fps', 'video_60fps', 'video_120fps', 'video_240fps',
      'video_480fps', 'video_960fps', 'price(USD)'],
      dtype='object')

```

✓ Преобразование данных

```

df['width'] = [int(i.split('x')[0]) for i in df['resolution']]
df['height'] = [int(i.split('x')[1]) for i in df['resolution']]

```

```

from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['brand'] = le.fit_transform(df['brand'])
df['battery_type'] = le.fit_transform(df['battery_type'])
df['os'] = le.fit_transform(df['os'])

```

```

bool_col = [col for col in df.columns if df[col].dtype == 'bool']
df[bool_col] = df[bool_col].astype(int)

```

```

df['announcement_date'] = pd.to_datetime(df['announcement_date'])
df['year'] = df['announcement_date'].dt.year

```

```

camera = [x for x in df.columns if 'video' in x]
df['camera_score'] = df[camera].sum(axis=1)

```

```
df.drop(bool_col, axis = 1, inplace=True)
```

```
df.columns
```

```

Index(['phone_name', 'brand', 'os', 'inches', 'resolution', 'battery',
      'battery_type', 'ram(GB)', 'announcement_date', 'weight(g)',
      'storage(GB)', 'price(USD)', 'width', 'height', 'year', 'camera_score'],
      dtype='object')

```

```
df = df.drop(['phone_name', 'announcement_date', 'resolution'], axis = 1)
```

```
df.info()
```

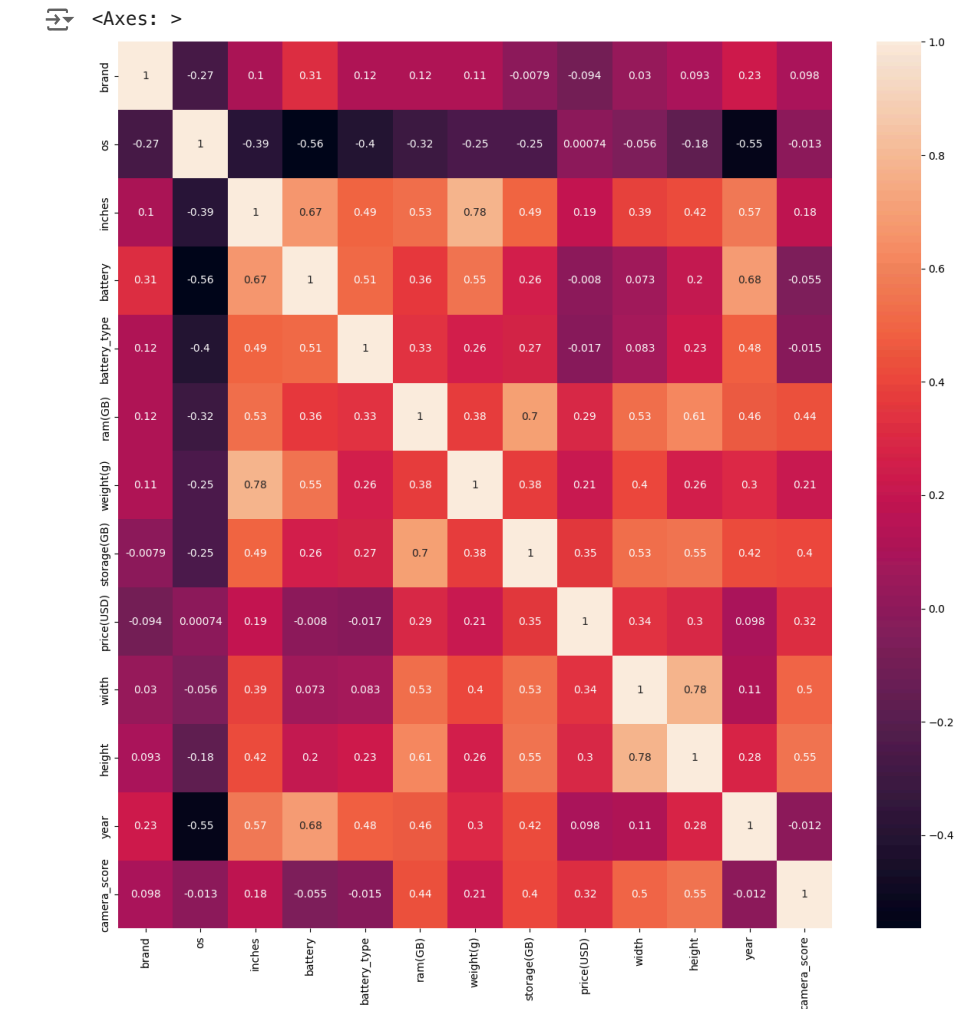
```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1512 entries, 0 to 1511
Data columns (total 13 columns):
#   Column          Non-Null Count  Dtype
---  ---
0   brand           1512 non-null  int64
1   os              1512 non-null  int64
2   inches          1512 non-null  float64
3   battery         1512 non-null  int64
4   battery_type    1512 non-null  int64
5   ram(GB)         1512 non-null  int64
6   weight(g)       1512 non-null  float64
7   storage(GB)     1512 non-null  int64
8   price(USD)      1512 non-null  float64
9   width           1512 non-null  int64

```

```
10 height          1512 non-null  int64
11 year            1512 non-null  int32
12 camera_score    1512 non-null  int64
dtypes: float64(3), int32(1), int64(9)
memory usage: 147.8 KB
```

```
plt.figure(figsize = (15,15))
sns.heatmap(df.corr(), annot = True)
```



✓ Масштабирование данных

```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
scaled_data = scaler.fit_transform(df)
df = pd.DataFrame(scaled_data, columns=df.columns)

X = df.drop(['price(USD)'], axis = 1)
y = df['price(USD)']
```

✓ Разделение выборки на обучающую и тестовую

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)

X_train.shape, y_train.shape, X_test.shape, y_test.shape
```

↗ ((1209, 12), (1209,), (303, 12), (303,))

✓ Обучение ансамблевых моделей

✓ Модель бэггинга

```
from sklearn.ensemble import BaggingRegressor

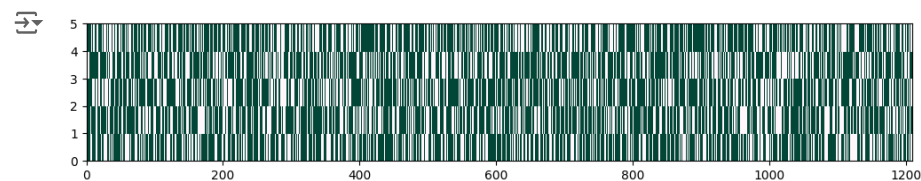
bagging_model = BaggingRegressor(n_estimators=5, oob_score=True, random_state=10)
bagging_model.fit(X_train, y_train)

bagging_y = bagging_model.predict(X_test)

bin_array = np.zeros((5, X_train.shape[0]))
for i in range(5):
    for j in bagging_model.estimators_samples_[i]:
        bin_array[i][j] = 1
bin_array
```

↗ array([[0., 0., 0., ..., 1., 1., 0.],
[0., 1., 0., ..., 0., 1., 1.],
[1., 1., 1., ..., 1., 0., 0.],
[1., 0., 1., ..., 1., 1., 1.],
[1., 1., 1., ..., 1., 1., 0.]])

```
fig, ax = plt.subplots(figsize=(12,2))
ax.pcolor(bin_array, cmap='PuBuGn')
plt.show()
```



```
for i in range(5):
    cur_data = bin_array[i]
    len_cur_data = len(cur_data)
    sum_cur_data = sum(cur_data)
    (len(bin_array[0]) - sum(bin_array[0])) / len(bin_array[0])
    oob_i = (len_cur_data - sum_cur_data) / len_cur_data
    print('Для модели {} размер OOB составляет {}'.format(i+1, round(oob_i, 4)*100.0))
```

↗ Для модели 1 размер OOB составляет 36.15%
Для модели 2 размер OOB составляет 36.059999999999995%
Для модели 3 размер OOB составляет 36.89%
Для модели 4 размер OOB составляет 37.8%
Для модели 5 размер OOB составляет 37.3%

✓ Случайный лес

```
# 10 деревьев
tree = RandomForestRegressor(n_estimators=10, random_state=12)
tree.fit(X_train, y_train)
tree_y = tree.predict(X_test)
```

tree_y

```
array([0.10176991, 0.2591085 , 0.05561593, 0.19871504, 0.14159292,
       0.08080177, 0.10339233, 0.35619469, 0.09459646, 0.17923319,
       0.11712035, 0.19685841, 0.21578761, 0.14050885, 0.09756637,
       0.08584071, 0.11147566, 0.08453142, 0.26202389, 0.08601062,
       0.04324726, 0.10297699, 0.23807788, 0.0880531 , 0.38230088,
       0.30313009, 0.08893805, 0.02763274, 0.14610566, 0.09175522,
       0.15221239, 0.09159292, 0.12538301, 0.10486726, 0.10474336,
       0.18858354, 0.18650442, 0.14779159, 0.1135469 , 0.15115472,
       0.15588407, 0.11712035, 0.17208053, 0.06548673, 0.11451372,
       0.04061947, 0.32364779, 0.28821504, 0.30353982, 0.08038348,
       0.12141593, 0.08893805, 0.22787611, 0.26150442, 0.16497788,
       0.14291549, 0.09140118, 0.09335752, 0.07710575, 0.06016947,
       0.17787611, 0.09557522, 0.18858354, 0.36411504, 0.14773894,
       0.27396903, 0.12131814, 0.17377301, 0.27402655, 0.06769912,
       0.16566394, 0.02827743, 0.33141593, 0.05584071, 0.10375133,
       0.13185841, 0.12365652, 0.19110619, 0.15324867, 0.12256637,
       0.22581106, 0.07163274, 0.09132743, 0.06769912, 0.09026549,
       0.04469027, 0.11688053, 0.05973451, 0.11283186, 0.19513274,
       0.15 , 0.50176991, 0.07250619, 0.02857699, 0.2000385 ,
       0.17310841, 0.14233053, 0.13244653, 0.12616504, 0.13853783,
       0.14032212, 0.08119469, 0.12166991, 0.05641745, 0.20012788,
       0.05320531, 0.20112832, 0.0857194 , 0.14469027, 0.04955752,
       0.10499749, 0.2000385 , 0.10575221, 0.06371681, 0.09047699,
       0.38761062, 0.23008805, 0.20229513, 0.06530973, 0.18225664,
       0.07655605, 0.12220743, 0.07345133, 0.05978451, 0.04930389,
       0.3854351 , 0.34245442, 0.08938053, 0.09380531, 0.11747788,
       0.10336168, 0.09911504, 0.15767982, 0.08561947, 0.06039823,
       0.04895499, 0.19557522, 0.06412248, 0.14115044, 0.16575208,
       0.15457788, 0.26349558, 0.05728496, 0.37234513, 0.20707965,
       0.02876106, 0.08849558, 0.15562965, 0.01700619, 0.17699115,
       0.12079646, 0.1659292 , 0.1141141 , 0.0941146 , 0.10022736,
       0.33592867, 0.06279808, 0.10349558, 0.14115044, 0.10929204,
       0.14836667, 0.08982257, 0.50486726, 0.0960177 , 0.03802313,
       0.09229504, 0.06761805, 0.04055434, 0.12079646, 0.14292035,
       0.16163717, 0.07710575, 0.16401858, 0.14376106, 0.13850084,
       0.16057367, 0.1 , 0.12131814, 0.20707965, 0.08205611,
       0.12345133, 0.15728168, 0.04729833, 0.02337876, 0.27145708,
       0.18440133, 0.11495115, 0.08761062, 0.0829154 , 0.12644735,
       0.10987739, 0.19915487, 0.02119602, 0.06457876, 0.17437788,
       0.10416681, 0.08390466, 0.20256549, 0.08400531, 0.11946903,
       0.11844425, 0.05796407, 0.03973451, 0.03802313, 0.18903973,
       0.11353938, 0.09690265, 0.08938584, 0.11599558, 0.16327434,
       0.21725664, 0.16007876, 0.1874073 , 0.27181361, 0.19043274,
       0.08663009, 0.17989947, 0.14970487, 0.04079189, 0.2610177 ,
       0.14911504, 0.08837522, 0.1135469 , 0.05097345, 0.06906062,
       0.1168385 , 0.12300885, 0.13565324, 0.15940265, 0.13424912,
       0.06398894, 0.13053097, 0.26884558, 0.16222071, 0.1473885 ,
       0.09675708, 0.12920354, 0.08061858, 0.04709265, 0.10176991,
       0.0877985 , 0.34645442, 0.07035398, 0.1948177 , 0.29527434,
       0.23118221, 0.17890855, 0.07431416, 0.13274115, 0.37871681,
       0.03753673, 0.15 , 0.0804615 , 0.10336168, 0.23265398,
       0.13353929, 0.20676991, 0.07816018, 0.18262071, 0.5079646 ,
       0.05708496, 0.2102208 , 0.08128628, 0.10708451, 0.04469027,
       0.09867212, 0.19263805, 0.18150301, 0.12743363, 0.10268982,
       0.1380531 , 0.19424779, 0.03035398, 0.09778761, 0.09574867,
       0.26957655, 0.12973451, 0.07875389, 0.05740301, 0.12075168,
       0.05740301, 0.28404867, 0.05522124, 0.09593186, 0.08717115,
       0.1311782 , 0.26442434, 0.19238938, 0.17566372, 0.08980885,
```

✓ Ada boosting

```
from sklearn.ensemble import AdaBoostRegressor
ada_boost_model = AdaBoostRegressor(random_state=42)
ada_boost_model.fit(X_train, y_train)
predictions = ada_boost_model.predict(X_test)
predictions
```

```
array([0.16756232, 0.1859162 , 0.14925959, 0.26920256, 0.14478193,
       0.17883235, 0.14478193, 0.47063134, 0.17883235, 0.17883235,
       0.17883235, 0.17883235, 0.17883235, 0.14392564, 0.14925959,
       0.17883235, 0.14925959, 0.25469012, 0.17883235, 0.14925959,
       0.15760683, 0.18824637, 0.13246773, 0.25469012, 0.33385408,
       0.13521878, 0.13521878, 0.18824637, 0.17883235, 0.2616506 ,
       0.14478193, 0.16756232, 0.13061342, 0.17883235,
```



```

0.17395657, 0.16756232, 0.16756232, 0.22073009, 0.16756232,
0.14869284, 0.17883235, 0.18824637, 0.19664357, 0.17883235,
0.14478193, 0.18824637, 0.26920256, 0.26240248, 0.11645297,
0.14925959, 0.14478193, 0.14478193, 0.17883235, 0.2616506 ,
0.13521878, 0.14925959, 0.16756232, 0.14925959, 0.14392564,
0.17395657, 0.12097364, 0.17395657, 0.2616506 , 0.17395657,
0.18824637, 0.17883235, 0.16756232, 0.30167762, 0.12097364,
0.17883235, 0.14925959, 0.26920256, 0.14925959, 0.16756232,
0.14925959, 0.17883235, 0.18824637, 0.17883235, 0.14869284,
0.21700813, 0.2616506 , 0.16756232, 0.12097364, 0.25469012,
0.14478193, 0.16756232, 0.099378 , 0.099378 , 0.14869284,
0.17883235, 0.39222966, 0.16756232, 0.14925959, 0.33385408,
0.16756232, 0.17883235, 0.17883235, 0.17883235, 0.17883235,
0.1859162 , 0.14392564, 0.16756232, 0.14925959, 0.17883235,
0.14925959, 0.26920256, 0.14925959, 0.099378 , 0.099378 ,
0.16756232, 0.33385408, 0.12097364, 0.099378 , 0.17883235,
0.25469012, 0.22073009, 0.25469012, 0.17883235, 0.18824637,
0.16756232, 0.14925959, 0.12097364, 0.14925959, 0.14925959,
0.54160177, 0.34432782, 0.14478193, 0.25469012, 0.14925959,
0.17883235, 0.17883235, 0.17883235, 0.16946697, 0.16234584,
0.14925959, 0.17883235, 0.14925959, 0.13246773, 0.17883235,
0.16756232, 0.18824637, 0.14925959, 0.34159646, 0.17395657,
0.13521878, 0.14478193, 0.16756232, 0.14925959, 0.1859162 ,
0.14869284, 0.17395657, 0.14925959, 0.17883235, 0.17883235,
0.1859162 , 0.14925959, 0.14478193, 0.14478193, 0.14478193,
0.16756232, 0.16756232, 0.54160177, 0.14869284, 0.14478193,
0.12097364, 0.14869284, 0.14925959, 0.14869284, 0.14869284,
0.16756232, 0.14925959, 0.17883235, 0.16756232, 0.14925959,
0.17883235, 0.17883235, 0.17883235, 0.14869284, 0.14925959,
0.14478193, 0.17883235, 0.14925959, 0.14925959, 0.17395657,
0.17883235, 0.17883235, 0.12097364, 0.16756232, 0.16234584,
0.17883235, 0.1859162 , 0.14925959, 0.1859162 , 0.14925959,
0.14478193, 0.17883235, 0.18824637, 0.16234584, 0.12097364,
0.14925959, 0.17883235, 0.14478193, 0.14478193, 0.18824637,
0.16756232, 0.17883235, 0.14925959, 0.17883235, 0.17883235,
0.14869284, 0.17883235, 0.16756232, 0.16756232, 0.17395657,
0.17883235, 0.17883235, 0.16756232, 0.14925959, 0.2616506 ,
0.17395657, 0.14925959, 0.22073009, 0.14478193, 0.14925959,
0.16756232, 0.17883235, 0.16756232, 0.13521878, 0.17883235,
0.099378 , 0.099378 , 0.17395657, 0.16756232, 0.17883235,
0.14925959, 0.13521878, 0.14925959, 0.14925959, 0.16756232,
0.16756232, 0.30397909, 0.17883235, 0.26920256, 0.17883235,
0.18824637, 0.1859162 , 0.099378 , 0.17883235, 0.25469012,
0.14925959, 0.17883235, 0.14925959, 0.17883235, 0.17395657,
0.17883235, 0.17883235, 0.14925959, 0.17883235, 0.54160177,
0.14925959, 0.34159646, 0.14925959, 0.16756232, 0.13521878,
0.16946697, 0.1859162 , 0.18824637, 0.17883235, 0.17883235,
0.14478193, 0.2616506 , 0.12097364, 0.14925959, 0.13061342,
0.19191669, 0.16234584, 0.14925959, 0.14925959, 0.17883235,
0.14925959, 0.26920256, 0.14478193, 0.16756232, 0.14925959,
0 17883235 0 18824637 0 17883235 0 2616506 0 16756232

```

✓ Модель градиентного бустинга

```
from sklearn.ensemble import GradientBoostingRegressor
```

```

gradient_model = GradientBoostingRegressor(n_estimators=5)
gradient_model.fit(X_train, y_train)
y_gradientboosting = gradient_model.predict(X_test)
y_gradientboosting

```

```

array([0.13511246, 0.17233101, 0.11277814, 0.18850368, 0.13255121,
0.12122442, 0.11866317, 0.12122442, 0.13511246, 0.13511246,
0.13511246, 0.13511246, 0.13511246, 0.11866317,
0.12949012, 0.13511246, 0.12949012, 0.15173054, 0.12122442,
0.11866317, 0.12014078, 0.15173054, 0.11866317, 0.18850368,
0.28359385, 0.11866317, 0.11866317, 0.15173054, 0.12122442,
0.13511246, 0.13255121, 0.12122442, 0.13255121, 0.13511246,
0.11866317, 0.13255121, 0.13511246, 0.12122442, 0.12122442,
0.12723572, 0.13511246, 0.15173054, 0.13255121, 0.13511246,
0.10475473, 0.21019676, 0.14315799, 0.175785 , 0.11866317,
0.12949012, 0.11866317, 0.13255121, 0.12122442, 0.13511246,
0.13255121, 0.11866317, 0.13511246, 0.0988697 , 0.11277814,
0.17687004, 0.11866317, 0.11866317, 0.12949012, 0.13255121,
0.21019676, 0.11840048, 0.13511246, 0.20638317, 0.11866317,
0.13511246, 0.11866317, 0.16867485, 0.11277814, 0.12122442,
0.13255121, 0.12122442, 0.15173054, 0.13511246, 0.14315799,
0.15699616, 0.11866317, 0.12122442, 0.11866317, 0.15405581,
0.10475473, 0.13511246, 0.11866317, 0.11866317, 0.19594753,
0.13511246, 0.23390039, 0.11866317, 0.11866317, 0.20638317,
0.13511246, 0.13511246, 0.13511246, 0.13511246, 0.13511246,
0.13511246, 0.11866317, 0.13511246, 0.0988697 , 0.13511246,
0.11277814, 0.15774704, 0.12949012, 0.11866317, 0.11866317,
0.13511246, 0.20638317, 0.11866317, 0.11866317, 0.12122442,
0.18850368, 0.15173054, 0.15173054, 0.10731599, 0.1400969 ,

```

```

0.13511246, 0.11866317, 0.11866317, 0.11277814, 0.10475473,
0.12122442, 0.20638317, 0.10475473, 0.14315799, 0.11866317,
0.12122442, 0.12122442, 0.13511246, 0.13742704, 0.11840048,
0.11866317, 0.13511246, 0.11277814, 0.13255121, 0.13511246,
0.12122442, 0.16657322, 0.11866317, 0.20638317, 0.15800067,
0.11866317, 0.11866317, 0.13511246, 0.0988697, 0.13511246,
0.14315799, 0.12949012, 0.11866317, 0.13511246, 0.11840048,
0.13511246, 0.10475473, 0.11866317, 0.13255121, 0.11866317,
0.12122442, 0.13511246, 0.13672193, 0.14315799, 0.10475473,
0.11866317, 0.10475473, 0.0988697, 0.14315799, 0.13511246,
0.13511246, 0.0988697, 0.13511246, 0.12122442, 0.11866317,
0.13511246, 0.12122442, 0.11840048, 0.12122442, 0.11866317,
0.13255121, 0.13511246, 0.0988697, 0.11866317, 0.13511246,
0.13511246, 0.13511246, 0.11866317, 0.12122442, 0.13511246,
0.13511246, 0.12122442, 0.10169365, 0.10449204, 0.11866317,
0.11866317, 0.12122442, 0.16657322, 0.10731599, 0.11866317,
0.11866317, 0.12122442, 0.10475473, 0.10475473, 0.15173054,
0.13511246, 0.13511246, 0.11277814, 0.13511246, 0.13511246,
0.13511246, 0.13511246, 0.13511246, 0.13511246, 0.13255121,
0.13511246, 0.13511246, 0.13511246, 0.11277814, 0.15173054,
0.13511246, 0.11866317, 0.12122442, 0.11866317, 0.11866317,
0.13511246, 0.13511246, 0.13511246, 0.13511246, 0.13511246,
0.10475473, 0.11866317, 0.13511246, 0.11866317, 0.13511246,
0.13255121, 0.13255121, 0.11866317, 0.0988697, 0.13511246,
0.11840048, 0.19164992, 0.12122442, 0.15173054, 0.13511246,
0.18850368, 0.13511246, 0.11866317, 0.13511246, 0.205448,
0.11866317, 0.13511246, 0.10169365, 0.12122442, 0.13511246,
0.13511246, 0.13511246, 0.11277814, 0.13511246, 0.13672193,
0.11866317, 0.13511246, 0.10169365, 0.13511246, 0.11866317,
0.13747281, 0.13672193, 0.15173054, 0.15173054, 0.12122442,
0.13255121, 0.11866317, 0.11866317, 0.11277814, 0.13255121,
0.16867485, 0.11277814, 0.10475473, 0.0988697, 0.13511246,
0.0988697, 0.15173054, 0.11866317, 0.13511246, 0.11866317,
0.13255121, 0.13255121, 0.11866317, 0.13255121, 0.13255121

```

✓ Стекинг

```
dataset = Dataset(X_train, y_train, X_test)
```

```
model_rf = Regressor(dataset=dataset, estimator=RandomForestRegressor, parameters={'n_estimators': 10}, name='rf')
model_lr = Regressor(dataset=dataset, estimator=LinearRegression, parameters={}, name='lr')
```

```
pipeline = ModelsPipeline(model_rf, model_lr)
stack_ds = pipeline.stack(k=15, seed=111)
```

```
stacker = Regressor(dataset=stack_ds, estimator=DecisionTreeRegressor)
stacker_y = stacker.predict()
results = stacker.validate(k=15, scorer=mean_squared_error)
```

```

Metric: mean_squared_error
Folds accuracy: [0.017915427802896483, 0.021731769122982064, 0.013347832485804257, 0.017442330518269072, 0.0290147083254
Mean accuracy: 0.020128072334420894
Standard Deviation: 0.005419890362223638
Variance: 2.9375211538524682e-05

```

✓ Модель многослойного персептрона

```
from sklearn.neural_network import MLPRegressor
```

```

mlp = MLPRegressor(hidden_layer_sizes=(100, 50), # Структура скрытых слоев
                    activation='relu',           # Функция активации
                    solver='adam',               # Оптимизатор
                    max_iter=1000,              # Максимальное число итераций
                    random_state=42)

```

```
# Обучение модели
mlp.fit(X_train, y_train)
```

```
nn_y = mlp.predict(X_test)
nn_y
```

```

array([0.11538884, 0.1475732, 0.09323403, 0.22812901, 0.10920988,
0.13014394, 0.08178939, 0.11760709, 0.10610423, 0.11396791,
0.13423277, 0.16312752, 0.12729333, 0.12887886, 0.07471095,
0.0960732, 0.10433174, 0.08557473, 0.15507126, 0.10988527,
0.06104875, 0.10879311, 0.24910293, 0.10433499, 0.28092765,
0.23801488, 0.04902827, 0.06365788, 0.16587253, 0.09987761,
0.12508277, 0.05000829, 0.10445032, 0.09697995, 0.10331713,
0.05920072, 0.07695417, 0.11933005, 0.1128849, 0.09731224,
0.07838016, 0.13423277, 0.21531335, 0.08918128, 0.10878526,
0.05755499, 0.426858, 0.15757394, 0.26443565, 0.09275872,
0.11582431, 0.11065747, 0.10539497, 0.09496359, 0.15870614,

```

```

0.18636052, 0.09057254, 0.11070021, 0.09295011, 0.05394476,
0.11573772, 0.08166461, 0.05920072, 0.11768173, 0.11024979,
0.38750266, 0.07822764, 0.11128127, 0.24819281, 0.09657004,
0.11785008, 0.06577801, 0.20268041, 0.0654479 , 0.10149517,
0.0748529 , 0.09161444, 0.22952322, 0.20660413, 0.15227305,
0.16650595, 0.09842326, 0.10815743, 0.09657004, 0.05843818,
0.07857027, 0.16035054, 0.08934942, 0.08197133, 0.11695724,
0.1499057 , 0.27365063, 0.09579997, 0.05489506, 0.22235218,
0.10428457, 0.14893687, 0.12184264, 0.12823993, 0.12148396,
0.12133107, 0.08038515, 0.11647827, 0.08508838, 0.20314722,
0.06488663, 0.15596037, 0.09180963, 0.08795227, 0.07458806,
0.11532824, 0.22235218, 0.08483764, 0.0663059 , 0.09709347,
0.17997262, 0.17822098, 0.14430179, 0.09157508, 0.14861832,
0.11597013, 0.07500735, 0.09293246, 0.09486835, 0.05200361,
0.12717025, 0.21678705, 0.11396098, 0.1217402 , 0.05218756,
0.09204128, 0.1251515 , 0.17177759, 0.03124266, 0.08636616,
0.09969585, 0.1564211 , 0.07343659, 0.08114427, 0.11951358,
0.09939511, 0.21067611, 0.09008146, 0.35425679, 0.12179877,
0.04827865, 0.10349031, 0.12395861, 0.04498454, 0.12854828,
0.13708183, 0.11929649, 0.04565419, 0.11000646, 0.10592178,
0.10058381, 0.08051355, 0.04612547, 0.08741956, 0.11507196,
0.08094913, 0.11448191, 0.19029131, 0.13929413, 0.05250748,
0.06689683, 0.08145896, 0.0409143 , 0.12124261, 0.10242368,
0.11621705, 0.09295011, 0.12686604, 0.10350314, 0.05857901,
0.12482037, 0.09412937, 0.07822764, 0.110147 , 0.07602304,
0.10634102, 0.16626857, 0.08492951, 0.05710912, 0.10972523,
0.18069666, 0.11418766, 0.06399392, 0.09958021, 0.12273068,
0.111828 , 0.11066595, 0.04494416, 0.09107661, 0.04948704,
0.10773866, 0.10147352, 0.16933085, 0.08453812, 0.09651596,
0.09716796, 0.0923456 , 0.06067951, 0.05250748, 0.13565747,
0.14196901, 0.14378271, 0.11339027, 0.09044682, 0.12121381,
0.13192427, 0.12687243, 0.12229883, 0.22281299, 0.10515386,
0.12448383, 0.13118761, 0.11602733, 0.07547219, 0.18167834,
0.15204711, 0.09183976, 0.1128849 , 0.07556754, 0.08525186,
0.16016537, 0.1513833 , 0.11687761, 0.09458125, 0.12406388,
0.03242133, 0.06967397, 0.10409459, 0.08052159, 0.10291004,
0.1044202 , 0.12336455, 0.05937816, 0.08541059, 0.11610394,
0.1077498 , 0.24210172, 0.10620578, 0.21862241, 0.2507288 ,
0.22682944, 0.1097919 , 0.0829525 , 0.14970095, 0.29216552,
0.05741756, 0.1499057 , 0.07305057, 0.09204128, 0.11035619,
0.1094373 , 0.22946625, 0.06345325, 0.08912573, 0.18433933,
0.0971331 , 0.22367556, 0.07646124, 0.10870699, 0.1121786 ,
0.1406179 , 0.18282943, 0.23367718, 0.15461364, 0.0706927 ,
0.08988797, 0.07706828, 0.06979774, 0.10217188, 0.10503668,
0.17257081, 0.11849973, 0.08354512, 0.07193773, 0.10282381,
0.07193773, 0.16796394, 0.0690563 , 0.11343235, 0.09616962,
0.08981784, 0.1504968 , 0.18185462, 0.13083517, 0.09121092,

```

```
from gmdh import Multi, split_data
```

```
model = Multi()
x_train, x_test, y_train, y_test = split_data(X, y, test_size=0.33)
```

```
model.fit(x_train, y_train, k_best=2, test_size=0.3)
```

```
y_predicted = model.predict(X_test)
y_predicted
```

```

array([0.14208213, 0.15213179, 0.11057812, 0.21294061, 0.11326468,
0.18122046, 0.08545414, 0.13698165, 0.14320159, 0.14481173,
0.14500974, 0.15271739, 0.15417166, 0.14575538, 0.08192499,
0.1245362 , 0.13336794, 0.12275594, 0.17018991, 0.14239416,
0.07194184, 0.11098339, 0.2083537 , 0.11256565, 0.22848628,
0.22135126, 0.07678513, 0.07882243, 0.18729172, 0.13011019,
0.14333139, 0.08402621, 0.13332594, 0.12947767, 0.1326047 ,
0.08911677, 0.11578084, 0.14635501, 0.14036831, 0.1316984 ,
0.12951168, 0.14500974, 0.19079406, 0.12509798, 0.13163968,
0.06014699, 0.20485126, 0.19691121, 0.21110545, 0.09127589,
0.12908191, 0.11872863, 0.14331796, 0.1111788 , 0.19870576,
0.16407523, 0.08942233, 0.13865871, 0.10409659, 0.09450742,
0.17633032, 0.13255134, 0.08911677, 0.12571356, 0.12078037,
0.19370552, 0.11453216, 0.14269018, 0.21274692, 0.08987196,
0.14011519, 0.07178486, 0.2392642 , 0.07047723, 0.13145518,
0.11204668, 0.12427387, 0.20200727, 0.16611821, 0.17306605,
0.16032577, 0.09424487, 0.13363677, 0.08987196, 0.11210499,
0.06352499, 0.18404729, 0.08452408, 0.11264963, 0.12707816,
0.14971526, 0.19519468, 0.11008545, 0.07129843, 0.21367027,
0.13979981, 0.16423087, 0.142288 , 0.14164306, 0.15077137,
0.14014441, 0.09193736, 0.14354145, 0.10440081, 0.17224424,
0.07409544, 0.15767965, 0.12585225, 0.09337711, 0.06672098,
0.14484908, 0.21367027, 0.08594463, 0.08024497, 0.13089564,
0.1884368 , 0.16642685, 0.16994669, 0.1224188 , 0.1147659 ,
0.14387621, 0.07194184, 0.09467171, 0.0998361 , 0.07489993,
0.1375897 , 0.1738458 , 0.11710169, 0.16456454, 0.0746844 ,
0.12427652, 0.14452877, 0.16431892, 0.13868911, 0.11116841,
0.11052441, 0.18430581, 0.08909712, 0.12619687, 0.14011519,
0.13145518, 0.2016658 , 0.10961697, 0.29093672, 0.13938798,
0.08883479, 0.10147025, 0.1429612 , 0.06403787, 0.14493334,

```

```

0.16268885, 0.14909951, 0.0656449 , 0.13429598, 0.1333161 ,
0.13311695, 0.10202025, 0.07647928, 0.09395893, 0.10867471,
0.10364631, 0.14715229, 0.16283633, 0.15173557, 0.06123221,
0.06867783, 0.10146645, 0.06659009, 0.17503878, 0.13606144,
0.14603511, 0.10409659, 0.14185536, 0.13308272, 0.08374318,
0.15220061, 0.12242666, 0.11453216, 0.14739629, 0.07194184,
0.11352265, 0.19127154, 0.10410614, 0.07081199, 0.13914318,
0.18434555, 0.13737064, 0.1153479 , 0.13051237, 0.14395289,
0.13393115, 0.16285892, 0.06382274, 0.12562947, 0.07431957,
0.08140247, 0.13067076, 0.1824877 , 0.12204499, 0.10628779,
0.11524497, 0.12524675, 0.05514081, 0.06123221, 0.16106344,
0.15690347, 0.15075592, 0.12175454, 0.1392487 , 0.14743406,
0.13767877, 0.14076144, 0.14142916, 0.17805237, 0.17123057,
0.14290333, 0.19793271, 0.14798086, 0.08289172, 0.23978828,
0.14939564, 0.11724043, 0.14036831, 0.08341685, 0.08557243,
0.18273966, 0.15110703, 0.14545713, 0.07893899, 0.15017948,
0.04619133, 0.07565283, 0.1349623 , 0.12970905, 0.13404103,
0.12392815, 0.13248939, 0.07991521, 0.10452242, 0.14387621,
0.1266652 , 0.17555931, 0.14068454, 0.21233257, 0.21966616,
0.20937512, 0.13469787, 0.08833144, 0.17341741, 0.24639154,
0.07951672, 0.14971526, 0.10335795, 0.12427652, 0.13877835,
0.14159287, 0.19514062, 0.07262211, 0.14828798, 0.16283633,
0.11127771, 0.19021998, 0.10335795, 0.14370967, 0.08960123,
0.13087995, 0.19144505, 0.20441554, 0.18518863, 0.11407599,
0.13670735, 0.0660013 , 0.06851634, 0.11971725, 0.11892017,
0.17755042, 0.12297064, 0.10311474, 0.10179335, 0.1332015 ,
0.10179335, 0.17026461, 0.06496807, 0.14788194, 0.11102069,
0.1332015 , 0.16320008, 0.15000000, 0.16757600, 0.13320008

```

```
from gmdh import Mia
```

```

mia_model = Mia()
mia_model.fit(x_train, y_train, k_best=5, p_average=3)

```

```

y_mia = mia_model.predict(X_test)
y_mia

```



```

0.12126872, 0.15739188, 0.12138252, 0.21974952, 0.1437926 ,
0.08574617, 0.11747657, 0.19723418, 0.11587192, 0.25088256,
0.33463973, 0.08562167, 0.08562167, 0.18896404, 0.14152684,
0.16035437, 0.1122179 , 0.14376598, 0.13219958, 0.16105563,
0.1040853 , 0.12064946, 0.16299098, 0.1437812 , 0.14102891,
0.13976987, 0.16482614, 0.21688645, 0.12041645, 0.15501119,
0.08574844, 0.18442375, 0.17798824, 0.21632034, 0.08527284,
0.1209401 , 0.11589481, 0.13302232, 0.11006675, 0.17463837,
0.13497775, 0.10411045, 0.15675786, 0.11702471, 0.1041211 ,
0.14494979, 0.12484199, 0.1040853 , 0.12078691, 0.11926819,
0.16523433, 0.11052362, 0.15999851, 0.30774505, 0.09443733,
0.16616816, 0.08578454, 0.21877254, 0.08579442, 0.14071088,
0.12019544, 0.1423262 , 0.18310614, 0.1643798 , 0.13882163,
0.17223735, 0.0852821 , 0.14416395, 0.09443733, 0.11142644,
0.09050934, 0.16429734, 0.09444354, 0.10999438, 0.12800016,
0.15793793, 0.206068 , 0.11618405, 0.0857764 , 0.3091745 ,
0.15899454, 0.16317722, 0.15930468, 0.16153994, 0.16402871,
0.15533822, 0.104105 , 0.16131944, 0.11696712, 0.16316234,
0.08580834, 0.18175991, 0.1205268 , 0.06209487, 0.08570927,
0.16017894, 0.3091745 , 0.09444147, 0.09443644, 0.14050196,
0.2185989 , 0.18165209, 0.21689169, 0.14336573, -0.00995588,
0.15881371, 0.08574617, 0.10080561, 0.08497152, 0.10412437,
0.14140111, 0.19227421, 0.12462393, 0.10482502, 0.10409951,
0.14191762, 0.15155549, 0.17245061, 0.11143697, 0.12763639,
0.11650635, 0.17013729, 0.10413118, 0.13263546, 0.16616816,
0.14071088, 0.18090494, 0.11628954, 0.23266555, 0.13297006,
0.10404767, 0.10873754, 0.16092831, 0.08579141, 0.16017894,
0.10831974, 0.12946173, 0.08578454, 0.16011512, 0.14481241,
0.15877801, 0.11626195, 0.0857189 , 0.11044878, 0.08527948,
0.11037388, 0.16437037, 0.17034547, 0.09403452, 0.08576517,
0.08570112, 0.11639525, 0.0858133 , 0.12701052, 0.1543229 ,
0.15897456, 0.11702471, 0.16904991, 0.14336573, 0.10411067,
0.16320008, 0.13968547, 0.11052362, 0.13764917, 0.08574617,
0.12078143, 0.17340889, 0.11693489, 0.08576683, 0.16271117,
0.17790221, 0.16277849, 0.124524 , 0.14222955, 0.16234439,
0.15943525, 0.12506827, 0.08578812, 0.14698252, 0.10409402,
0.10080747, 0.14376598, 0.17751175, 0.14204973, 0.08520939,
0.11650635, 0.14389082, 0.07162622, 0.08576517, 0.18528137,
0.16849024, 0.15817123, 0.11657985, 0.16500657, 0.16035437,
0.16283607, 0.15694775, 0.16151908, 0.16692319, 0.13637818,
0.15762768, 0.17316149, 0.16192641, 0.10412955, 0.22156972,
0.15381497, 0.11642423, 0.1437812 , 0.09444396, 0.10411045,
0.16586888, 0.15875835, 0.16112252, 0.08570884, 0.17270925,
0.07132141, 0.09444363, 0.15999851, 0.12598949, 0.16032443,
0.12107588, 0.12030567, 0.08578637, 0.1170128 , 0.15881371,
0.13428453, 0.16686175, 0.14091801, 0.27455851, 0.17844207,
0.21065266, 0.1617855 , 0.09643824, 0.16055858, 0.24358203,
0.10411045, 0.15793793, 0.11661236, 0.14191762, 0.16218302,
0.16320008, 0.1680298 , 0.08579957, 0.17071183, 0.17034547,
0.11656236, 0.1685697 , 0.11661236, 0.16192641, 0.09443103,

```

```
0.11004112, 0.22209031, 0.00311232, 0.10304111, 0.11041219,
0.14222955, 0.17197698, 0.16938725, 0.16139417, 0.11046053,
0.11891289, 0.10411637, 0.07129832, 0.11034979, 0.08573522,
0.14071088, 0.09444005, 0.08578812, 0.19695382, 0.17067102,
0.11938378, 0.16315762, 0.1041211 ])
```

✓ Оценка моделей

```
results_metrics = [mean_squared_error(y_test, bagging_y), mean_squared_error(y_test, tree_y), mean_squared_error(y_test, pred
model_list = ['bagging', 'random_forest', 'ada-boosting', 'gradient-boosting', 'stacker', 'nn_y', 'multi', 'mia']
```

```
sorted_el = list(sorted(list(zip(model_list, results_metrics)), key=lambda x: -x[1]))
results_metrics = list(map(lambda x: x[1], sorted_el))
model_list = list(map(lambda x: x[0], sorted_el))
```

```
fig, ax = plt.subplots(figsize=(20,20))
pos = np.arange(len(model_list))
rects = ax.barh(pos, results_metrics,
                 align='center',
                 height=0.5,
                 tick_label=model_list)
ax.set_title('mean_squared')
for a, b in zip(pos, results_metrics):
    plt.text(max(results_metrics)/2, a-0.05, str(round(b,6)), color='black')
plt.show()
```

