

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

Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Технологии машинного обучения»

Лабораторная работа №5

Выполнила:

студентка ИУ5-62Б
Федосеева Елизавета

Проверил:

преподаватель каф. ИУ5
Гапанюк Ю.Е.

Москва, 2022 г.

Задание:

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 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](#). По желанию, вместо библиотеки `scikit-learn` возможно использование библиотек [TensorFlow](#), [PyTorch](#) 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.
6. Оцените качество моделей с помощью одной из подходящих для задачи метрик. Сравните качество полученных моделей.

Лабораторная работа №5: Ансамбли моделей машинного обучения.

0) Библиотеки, загрузка датасета, кодирование категориальных признаков

```
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
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]: df = pd.read_csv('student-mat.csv')
```

```
In [3]: df.head()
```

```
Out[3]:
```

| | school | sex | age | address | famsize | Pstatus | Medu | Fedu | Mjob | Fjob | ... | famrel | freetime | goout | Dalc | Walc | health | absences | G1 |
|---|--------|-----|-----|---------|---------|---------|------|------|---------|----------|-----|--------|----------|-------|------|------|--------|----------|----|
| 0 | GP | F | 18 | U | GT3 | A | 4 | 4 | at_home | teacher | ... | 4 | 3 | 4 | 1 | 1 | 3 | 6 | 5 |
| 1 | GP | F | 17 | U | GT3 | T | 1 | 1 | at_home | other | ... | 5 | 3 | 3 | 1 | 1 | 3 | 4 | 5 |
| 2 | GP | F | 15 | U | LE3 | T | 1 | 1 | at_home | other | ... | 4 | 3 | 2 | 2 | 3 | 3 | 10 | 7 |
| 3 | GP | F | 15 | U | GT3 | T | 4 | 2 | health | services | ... | 3 | 2 | 2 | 1 | 1 | 5 | 2 | 15 |
| 4 | GP | F | 16 | U | GT3 | T | 3 | 3 | other | other | ... | 4 | 3 | 2 | 1 | 2 | 5 | 4 | 6 |

5 rows × 33 columns

```
In [4]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 395 entries, 0 to 394
Data columns (total 33 columns):
 #   Column          Non-Null Count  Dtype
---  -
 0   school          395 non-null    object
 1   sex             395 non-null    object
 2   age            395 non-null    int64
 3   address        395 non-null    object
 4   famsize        395 non-null    object
 5   Pstatus        395 non-null    object
 6   Medu           395 non-null    int64
 7   Fedu           395 non-null    int64
 8   Mjob           395 non-null    object
 9   Fjob           395 non-null    object
10   reason         395 non-null    object
11   guardian       395 non-null    object
12   traveltime     395 non-null    int64
13   studytime      395 non-null    int64
14   failures       395 non-null    int64
15   schoolsup      395 non-null    object
16   famsup         395 non-null    object
17   paid           395 non-null    object
18   activities     395 non-null    object
19   nursery        395 non-null    object
20   higher         395 non-null    object
21   internet       395 non-null    object
22   romantic       395 non-null    object
23   famrel         395 non-null    int64
24   freetime       395 non-null    int64
25   goout          395 non-null    int64
26   Dalc           395 non-null    int64
27   Walc           395 non-null    int64
28   health         395 non-null    int64
```

```
29 absences      395 non-null    int64
30 G1             395 non-null    int64
31 G2             395 non-null    int64
32 G3             395 non-null    int64
dtypes: int64(16), object(17)
memory usage: 102.0+ KB
```

Определим категориальные признаки и закодируем их.

```
In [5]: category_cols = ['school', 'sex', 'address', 'famsize', 'Pstatus', 'Mjob', 'Fjob', 'reason', 'guardian', 'schools
                        'paid', 'activities', 'nursery', 'higher', 'internet', 'romantic']

print('Количество уникальных значений\n')
for col in category_cols:
    print(f'{col}: {df[col].unique().size}')
```

Количество уникальных значений

```
school: 2
sex: 2
address: 2
famsize: 2
Pstatus: 2
Mjob: 5
Fjob: 5
reason: 4
guardian: 3
schoolsup: 2
famsup: 2
paid: 2
activities: 2
nursery: 2
higher: 2
internet: 2
romantic: 2
```

```
In [6]: df = pd.get_dummies(df, columns=category_cols)
```

1) Разделение выборки на обучающую и на тестовую

Для начала проведем корреляционный анализ, чтобы выявить признаки, имеющие наибольшее значение для прогнозирования успеваемости.

```
In [7]: print('Признаки, имеющие максимальную по модулю корреляцию с итоговой оценкой')
best_params = df.corr()['G3'].map(abs).sort_values(ascending=False)[1:]
best_params = best_params[best_params.values > 0.3]
best_params
```

Признаки, имеющие максимальную по модулю корреляцию с итоговой оценкой

```
Out[7]: G2          0.904868
G1          0.801468
failures    0.360415
Name: G3, dtype: float64
```

Признаков вышло мало, поэтому сразу разделим выборки.

```
In [8]: y = df['G3']
X = df[best_params.index]
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=3)
```

2) Масштабирование данных

```
In [9]: 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)
```

3) Метрики

```
In [10]: 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 [11]: print_metrics(y_test, RandomForestRegressor(random_state=17).fit(x_train, y_train).predict(x_test))

R^2: 0.857222085513676
MSE: 3.0365185332968876
MAE: 1.108120080570278
```

Подбор гиперпараметров

```
In [12]: 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': 'poisson', 'max_features': 'auto', 'min_samples_leaf': 5, 'n_estimators': 100}
```

```
In [13]: 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: 0.7925110266725222
MSE: 4.412756099081269
MAE: 1.3328462761153206
```

Модель №2: Градиентный бустинг

```
In [14]: print_metrics(y_test, GradientBoostingRegressor(random_state=17).fit(x_train, y_train).predict(x_test))

R^2: 0.8579452311033859
MSE: 3.0211390889806844
MAE: 1.1456496997930061
```

Подбор гиперпараметров

```
In [15]: 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': 'mae', 'loss': 'huber', 'min_samples_leaf': 3, 'n_estimators': 50}
```

```
In [16]: 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: 0.839519389707239
MSE: 3.4130092818762945
MAE: 1.0444923449882149
```

Модель №3: Стекинг

```
In [17]: dataset = Dataset(x_train, y_train, x_test)
```

```
In [18]: model_lr = Regressor(dataset=dataset, estimator=LinearRegression, name='lr')
model_rf = Regressor(dataset=dataset, estimator=RandomForepipeline = ModelsPipeline(model_lr, model_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, 268.2067712261299, 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)
```

R²: 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))
```

R²: 0.3933464482443907

MSE: 262683.73918006354

MAE: 406.8932580917785

Подбор гиперпараметров

```
In [52]: 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)
```

R²: 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
R²: 0.6898203012827298
MSE: 134309.21625861025
MAE: 252.41492530666685

Градиентный бустинг
R²: 0.7013333844767404
MSE: 129323.99902194891
MAE: 253.7859718910538

Стекинг
R²: 0.7207185369761542
MSE: 120930.14007496767
MAE: 247.18161038788267

Многослойный персептрон
R²: 0.6422646017371612
MSE: 154901.0498344665
MAE: 288.659695272951

Метод группового учёта аргументов
R²: 0.6642449299187112
MSE: 145383.4680475877
MAE: 274.30940411915725