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 «Системы обработки информации и управления»

Отчет по лабораторной работе №5

«Ансамбли моделей машинного обучения. Часть 2»

по дисциплине «Технологии машинного обучения»

Выполнил:
студент группы ИУ5-64Б
Стукалов И.Д.
10.05.2024

Проверил:
Гапанюк Ю.Е.

2024 г.

Задание.

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

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:

- 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.

Текст программы

Лабораторная работа 6. Ансамбли моделей машинного обучения. Часть 2.

```
!pip install heamy
```

```
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) (2023.4)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.17.0->heamy) (2024.1)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.17.0->heamy) (1.4.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.17.0->heamy) (3.4.0)
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=4cf9789f88bc55e94f307e714a9c249944d721d065c8565066ed
  Stored in directory: /root/.cache/pip/wheels/e5/e4/9a/bc85119b96421369998ff0f53c0854b57bfb518c460fe8c5de
Successfully built heamy
Installing collected packages: heamy
Successfully installed heamy-0.0.7
```

```
from sklearn.datasets import load_wine, load_iris
from sklearn.model_selection import train_test_split
from heamy.estimator import Regressor, Classifier
from heamy.pipeline import ModelsPipeline
from heamy.dataset import Dataset
from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor, export_graphviz
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, mean_squared_log_error, median_absolute_error, r2_score
```

Чтобы изменить содержимое ячейки, дважды нажмите на нее (или выберите "Ввод")

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

```
# Используем датасет wine
wine = load_wine()
wine_X = wine.data
wine_y = wine.target
```

С использованием метода train_test_split разделите выборку на обучающую и тестовую.

```
wine_X_train, wine_X_test, wine_y_train, wine_y_test = train_test_split(
    wine_X, wine_y, test_size=0.2, random_state=1)
```

С использованием метода train_test_split разделите выборку на обучающую и тестовую.

```
wine_X_train, wine_X_test, wine_y_train, wine_y_test = train_test_split(
    wine_X, wine_y, test_size=0.2, random_state=1)
```

Стекинг

```
# Используем библиотеку heamy
# набор данных
dataset = Dataset(wine_X_train, wine_y_train, wine_X_test)

# модели первого уровня
model_tree = Regressor(dataset=dataset, estimator=DecisionTreeRegressor, name='tree')
model_lr = Regressor(dataset=dataset, estimator=LinearRegression, name='lr')
model_rf = Regressor(dataset=dataset, estimator=RandomForestRegressor, parameters={'n_estimators': 50}, name='rf')
```

```
# Эксперимент 1
# Первый уровень - две модели: дерево и линейная регрессия
# Второй уровень: линейная регрессия

pipeline = ModelsPipeline(model_tree, model_lr)
stack_ds = pipeline.stack(k=10, seed=1)
# модель второго уровня
stacker = Regressor(dataset=stack_ds, estimator=LinearRegression)
results = stacker.validate(k=10,scorer=mean_absolute_error)
```

```
Metric: mean_absolute_error
Folds accuracy: [0.14366357347211464, 0.20629198841959795, 0.1844558548934073, 0.1365196672185216, 0.16278448974557463, 0.18044284300760358, 0.3659790314893313, 0.11904927550480686, 0.13128012492285818, 0.22722949357237662]
Mean accuracy: 0.18576963422461928
Standard Deviation: 0.06840549970778417
Variance: 0.004679312390271661
```

+ Код + Текст

модель многослойного персептрона

```
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score

# Создание датасета
# X, y = make_classification(n_samples=1000, n_features=20, n_informative=15, n_redundant=5, random_state=42)

# Разделение данных на обучающую и тестовую выборки
# X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Определение параметров MLP
mlp_params = {
    'hidden_layer_sizes': (50, 50), # Количество нейронов в скрытых слоях
    'activation': 'relu', # Функция активации
    'solver': 'adam', # Оптимизатор
```

модель многослойного персептрона

```
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score

# Создание датасета
# X, y = make_classification(n_samples=1000, n_features=20, n_informative=15, n_redundant=5, random_state=42)

# Разделение данных на обучающую и тестовую выборки
# X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Определение параметров MLP
mlp_params = {
    'hidden_layer_sizes': (50, 50), # Количество нейронов в скрытых слоях
    'activation': 'relu', # Функция активации
    'solver': 'adam', # Оптимизатор
    'alpha': 0.0001, # Коэффициент регуляризации L2
    'learning_rate': 'constant', # Стратегия изменения скорости обучения
    'learning_rate_init': 0.001, # Начальная скорость обучения
    'max_iter': 200, # Максимальное количество итераций
    'random_state': 42, # Случайное состояние для воспроизводимости
}

# Создание и обучение MLP
mlp = MLPClassifier(**mlp_params)
mlp.fit(wine_X_train, wine_y_train)

# Предсказание на тестовом наборе данных
y_pred = mlp.predict(wine_X_test)

# Оценка качества модели
accuracy = accuracy_score(wine_y_test, y_pred)
print(f'Accuracy: {accuracy}')
```

```
Accuracy: 0.8611111111111112
/usr/local/lib/python3.10/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(
```

МГУА

```
[ ] !pip install git+https://github.com/bauman-team/GMDH.git
    !pip install gmdh
    import gmdh
```

```
Collecting git+https://github.com/bauman-team/GMDH.git
  Cloning https://github.com/bauman-team/GMDH.git to /tmp/pip-req-build-gqfbb5q7
  Running command git clone --filter=blob:none --quiet https://github.com/bauman-team/GMDH.git /tmp/pip-req-build-gqfbb5q7
  Resolved https://github.com/bauman-team/GMDH.git to commit dddc7b9a76b0930d08ac44bd3d7444473552ddb5
  Preparing metadata (setup.py) ... done
Collecting docstring_inheritance (from gmdh==1.0.3)
  Using cached docstring_inheritance-2.2.0-py3-none-any.whl (24 kB)
```

МГУА

```
!pip install git+https://github.com/bauman-team/GMDH.git
!pip install gmdh
import gmdh
```

```
Collecting git+https://github.com/bauman-team/GMDH.git
  Cloning https://github.com/bauman-team/GMDH.git to /tmp/pip-req-build-gqfbb5q7
  Running command git clone --filter=blob:none --quiet https://github.com/bauman-team/GMDH.git /tmp/pip-req-build-gqfbb5q7
  Resolved https://github.com/bauman-team/GMDH.git to commit dddc7b9a76b0930d08ac44bd3d7444473552ddb5
  Preparing metadata (setup.py) ... done
Collecting docstring_inheritance (from gmdh==1.0.3)
  Using cached docstring_inheritance-2.2.0-py3-none-any.whl (24 kB)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from gmdh==1.0.3) (1.25.2)
Building wheels for collected packages: gmdh
  error: subprocess-exited-with-error

  × python setup.py bdist_wheel did not run successfully.
  │ exit code: 1
  ╰─> See above for output.

  note: This error originates from a subprocess, and is likely not a problem with pip.
  Building wheel for gmdh (setup.py) ... error
  ERROR: Failed building wheel for gmdh
  Running setup.py clean for gmdh
Failed to build gmdh
ERROR: Could not build wheels for gmdh, which is required to install pyproject.toml-based projects
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)
  Using cached docstring_inheritance-2.2.0-py3-none-any.whl (24 kB)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from gmdh) (1.25.2)
Installing collected packages: docstring-inheritance, gmdh
Successfully installed docstring-inheritance-2.2.0 gmdh-1.0.3
```

```
[ ] def print_metrics(y_test, y_pred, squared=False):
        print(f"R^2: {r2_score(y_test, y_pred)}")
        crit_name = "MSE" if squared else "RMSE"
        print(f"{crit_name}: {mean_squared_error(y_test, y_pred, squared=squared)}")
        print(f"MAE: {mean_absolute_error(y_test, y_pred)}")
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell` argument and
  and should_run_async(code)
```

```
[ ] combi_model = gmdh.Combi()
    combi_model.fit(wine_X_train, wine_y_train, verbose=1, n_jobs=-1, test_size=0.24, limit=0,
                    criterion=gmdh.Criterion(gmdh.CriterionType.REGULARITY))

    print()
    print(combi_model.get_best_polynomial())
    print()
    y_pred_combi = combi_model.predict(wine_X_test)

    print_metrics(wine_y_test, y_pred_combi)
```

```
  and should_run_async(code)
```

```
combi_model = gmdh.Combi()
combi_model.fit(wine_X_train, wine_y_train, verbose=1, n_jobs=-1, test_size=0.24, limit=0,
                criterion=gmdh.Criterion(gmdh.CriterionType.REGULARITY))
print()
print(combi_model.get_best_polynomial())
print()
y_pred_combi = combi_model.predict(wine_X_test)

print_metrics(wine_y_test, y_pred_combi)
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell` argument and
  and should_run_async(code)
LEVEL 1  [========================] 100% [00m:00s] (13 combinations) error=4.694871
LEVEL 2  [========================] 100% [00m:00s] (78 combinations) error=3.713584
LEVEL 3  [========================] 100% [00m:00s] (286 combinations) error=2.77371
LEVEL 4  [========================] 100% [00m:00s] (715 combinations) error=2.325969
LEVEL 5  [========================] 100% [00m:00s] (1287 combinations) error=2.199924
LEVEL 6  [========================] 100% [00m:00s] (1716 combinations) error=2.169235
LEVEL 7  [========================] 100% [00m:00s] (1716 combinations) error=2.168724
LEVEL 8  [========================] 100% [00m:00s] (1287 combinations) error=2.188381

y = 0.0337*x4 + 0.0187*x6 - 0.3757*x7 - 0.0294*x8 + 0.088*x10 - 0.2374*x11 - 0.001*x13 + 1.554

R^2: 0.86133867931356
RMSE: 0.2931115535963
MAE: 0.2270581109591602
```

```
[ ] ria_model = gmdh.Ria()
    ria_model.fit(wine_X_train, wine_y_train, verbose=1, n_jobs=-1, test_size=0.52, limit=0, k_best=7,
                  criterion=gmdh.Criterion(gmdh.CriterionType.REGULARITY),
                  polynomial_type=gmdh.PolynomialType.LINEAR)
    y_pred_ria = ria_model.predict(wine_X_test)
    print_metrics(wine_y_test, y_pred_ria)
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell` argument and
  and should_run_async(code)
LEVEL 1  [========================] 100% [00m:00s] (78 combinations) error=11.237277
LEVEL 2  [========================] 100% [00m:00s] (91 combinations) error=9.196032
LEVEL 3  [========================] 100% [00m:00s] (91 combinations) error=7.506619
LEVEL 4  [========================] 100% [00m:00s] (91 combinations) error=6.087672
LEVEL 5  [========================] 100% [00m:00s] (91 combinations) error=5.803263
LEVEL 6  [========================] 100% [00m:00s] (91 combinations) error=5.480924
LEVEL 7  [========================] 100% [00m:00s] (91 combinations) error=5.055742
LEVEL 8  [========================] 100% [00m:00s] (91 combinations) error=4.804413
LEVEL 9  [========================] 100% [00m:00s] (91 combinations) error=4.616872
LEVEL 10 [========================] 100% [00m:00s] (91 combinations) error=4.525378
LEVEL 11 [========================] 100% [00m:00s] (91 combinations) error=4.506103
LEVEL 12 [========================] 100% [00m:00s] (91 combinations) error=4.498368
LEVEL 13 [========================] 100% [00m:00s] (91 combinations) error=4.486153
LEVEL 14 [========================] 100% [00m:00s] (91 combinations) error=4.485931
LEVEL 15 [========================] 100% [00m:00s] (91 combinations) error=4.485591
LEVEL 16 [========================] 100% [00m:00s] (91 combinations) error=4.485584
LEVEL 17 [========================] 100% [00m:00s] (91 combinations) error=4.485573
LEVEL 18 [========================] 100% [00m:00s] (91 combinations) error=4.485573
```


```
y = 0.0337*x4 + 0.0187*x6 - 0.3757*x7 - 0.0294*x8 + 0.088*x10 - 0.2374*x11 - 0.001*x13 + 1.554

R^2: 0.86133867931356
RMSE: 0.2931115535963
MAE: 0.2270581109591602
```

```
ria_model = gmdh.Ria()
ria_model.fit(wine_X_train, wine_y_train, verbose=1, n_jobs=-1, test_size=0.52, limit=0, k_best=7,
              criterion=gmdh.Criterion(gmdh.CriterionType.REGULARITY),
              polynomial_type=gmdh.PolynomialType.LINEAR)
y_pred_ria = ria_model.predict(wine_X_test)
print_metrics(wine_y_test, y_pred_ria)
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell` argument and
  and should_run_async(code)
LEVEL 1  [========================] 100% [00m:00s] (78 combinations) error=11.237277
LEVEL 2  [========================] 100% [00m:00s] (91 combinations) error=9.196032
LEVEL 3  [========================] 100% [00m:00s] (91 combinations) error=7.506619
LEVEL 4  [========================] 100% [00m:00s] (91 combinations) error=6.087672
LEVEL 5  [========================] 100% [00m:00s] (91 combinations) error=5.803263
LEVEL 6  [========================] 100% [00m:00s] (91 combinations) error=5.480924
LEVEL 7  [========================] 100% [00m:00s] (91 combinations) error=5.055742
LEVEL 8  [========================] 100% [00m:00s] (91 combinations) error=4.804413
LEVEL 9  [========================] 100% [00m:00s] (91 combinations) error=4.616872
LEVEL 10 [========================] 100% [00m:00s] (91 combinations) error=4.525378
LEVEL 11 [========================] 100% [00m:00s] (91 combinations) error=4.506103
LEVEL 12 [========================] 100% [00m:00s] (91 combinations) error=4.498368
LEVEL 13 [========================] 100% [00m:00s] (91 combinations) error=4.486153
LEVEL 14 [========================] 100% [00m:00s] (91 combinations) error=4.485931
LEVEL 15 [========================] 100% [00m:00s] (91 combinations) error=4.485591
LEVEL 16 [========================] 100% [00m:00s] (91 combinations) error=4.485584
LEVEL 17 [========================] 100% [00m:00s] (91 combinations) error=4.485573
LEVEL 18 [========================] 100% [00m:00s] (91 combinations) error=4.485573
LEVEL 19 [========================] 100% [00m:00s] (91 combinations) error=4.485572
LEVEL 20 [========================] 100% [00m:00s] (91 combinations) error=4.485572
LEVEL 21 [========================] 100% [00m:00s] (91 combinations) error=4.485572
LEVEL 22 [========================] 100% [00m:00s] (91 combinations) error=4.485572
LEVEL 23 [========================] 100% [00m:00s] (91 combinations) error=4.485572
LEVEL 24 [========================] 100% [00m:00s] (91 combinations) error=4.485572
LEVEL 25 [========================] 100% [00m:00s] (91 combinations) error=4.485572
LEVEL 26 [========================] 100% [00m:00s] (91 combinations) error=4.485572
LEVEL 27 [========================] 100% [00m:00s] (91 combinations) error=4.485572
LEVEL 28 [========================] 100% [00m:00s] (91 combinations) error=4.485572
LEVEL 29 [========================] 100% [00m:00s] (91 combinations) error=4.485572
LEVEL 30 [========================] 100% [00m:00s] (91 combinations) error=4.485572
LEVEL 31 [========================] 100% [00m:00s] (91 combinations) error=4.485572
LEVEL 32 [========================] 100% [00m:00s] (91 combinations) error=4.485572
LEVEL 33 [========================] 100% [00m:00s] (91 combinations) error=4.485572
R^2: 0.8894515959071463
RMSE: 0.2617167451586178
MAE: 0.2067247772570705
```