Московский государственный технический университет им. Н. Э. Баумана

Курс «Технологии машинного обучения»

Отчёт по лабораторной работе №6

Выполнил:
Каятский П. Е.
группа ИУ5-64Б

Проверил:
Гапанюк Ю.Е.

Дата: 19.04.25

Подпись:

Дата:

Подпись:

Москва, 2025 г.

Цель лабораторной работы: изучение ансамблей моделей машинного обучения.

Задание:

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 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.
    - **В настоящее время библиотека МГУА не позволяет решать задачу классификации !!!**
5. Оцените качество моделей с помощью одной из подходящих для задачи метрик. Сравните качество полученных моделей.
6. В телегамм-канале потока ИУ5 в теме **ТМО_МГУА** напишите обратную связь по использованию библиотеки gmdh:
    - обнаруженные баги с приложением скриншотов ошибок, **за каждый найденный баг +1 балл на экзамене**;
    - опечатки в документации или учебном пособии МГУА;
    - возникшие вопросы или трудности при установке и использовании библиотеки;
    - любая другая информация (критика, предложения по улучшению и тд).

7. Справочные материалы по МГУА:
    - Видеозапись доклада.
    - Учебное пособие по МГУА (предварительная версия).
    - Примеры использования библиотеки.

Ход выполнения:

```
Add Code Cell | Add Markdown Cell

[1]
from sklearn.datasets import load_diabetes
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

data = load_diabetes()
X, y = data.data, data.target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
Executed at 2025.04.19 10:04:27 in 3s 339ms

[2]
from sklearn.ensemble import StackingRegressor
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import GradientBoostingRegressor

estimators = [
    ('dt', DecisionTreeRegressor(random_state=42)),
    ('boosting', GradientBoostingRegressor(random_state=42))
]

stacking = StackingRegressor(estimators=estimators, final_estimator=LinearRegression())
stacking.fit(X_train, y_train)
Executed at 2025.04.19 10:04:35 in 3s 843ms

StackingRegressor(estimators=[('dt', DecisionTreeRegressor(random_state=42)),
                              ('boosting',
                               GradientBoostingRegressor(random_state=42))],
                  final_estimator=LinearRegression())

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

3]
from sklearn.neural_network import MLPRegressor

mlp = MLPRegressor(hidden_layer_sizes=(100, 50), max_iter=500, random_state=42)
mlp.fit(X_train, y_train)
Executed at 2025.04.19 10:04:40 in 1s 444ms

C:\Users\kayat\AppData\Local\Programs\Python\Python39\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py

Web: 1,039x86 px

_]
from gmdh import Combi, Mia

combi = Combi()
combi.fit(X_train, y_train)
y_pred_combi = combi.predict(X_test)

mia = Mia()
mia.fit(X_train, y_train)
y_pred_mia = mia.predict(X_test)
```

[20]

```python
from sklearn.metrics import mean_squared_error, r2_score

y_pred_stack = stacking.predict(X_test)
print(f"Stacking: MSE = {mean_squared_error(y_test, y_pred_stack):.4f}, R² = {r2_score(y_test, y_pred_stack):.4f}")

y_pred_mlp = mlp.predict(X_test)
print(f"MLP: MSE = {mean_squared_error(y_test, y_pred_mlp):.4f}, R² = {r2_score(y_test, y_pred_mlp):.4f}")

print(f"COMBI: MSE = {mean_squared_error(y_test, y_pred_combi):.4f}, R² = {r2_score(y_test, y_pred_combi):.4f}")

print(f"MIA: MSE = {mean_squared_error(y_test, y_pred_mia):.4f}, R² = {r2_score(y_test, y_pred_mia):.4f}")
```

```
Stacking: MSE = 3066.0214, R² = 0.4320
MLP: MSE = 3026.0175, R² = 0.4395
COMBI: MSE = 2943.8537, R² = 0.4547
MIA: MSE = 2748.4422, R² = 0.4909
```