Московский государственный технический университет им. Н. Э. Баумана

Курс «Технологии машинного обучения»

Отчёт по лабораторной работе №6

Выполнил:
Файзуллин К. Х.
группа ИУ5-64Б

Проверил:
Гапанюк Ю.Е.

Дата: 21.06.25

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.

Ход выполнения:

```
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)
```

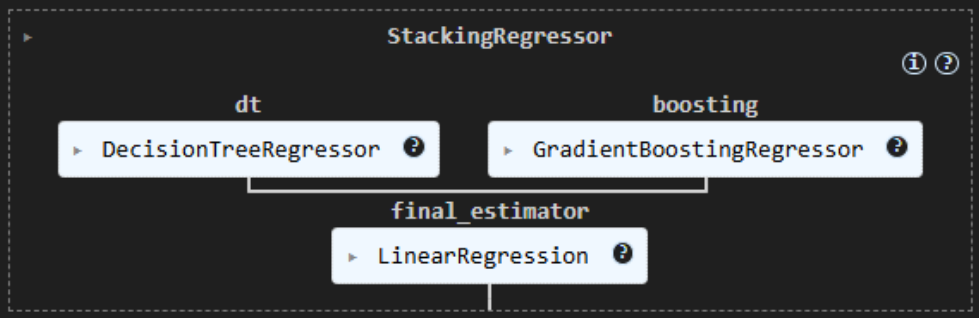
✓ 0.8s

```python
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)
```

✓ 0.5s




```python
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)
```

✓ 0.5s

MLPRegressor

▼ Parameters

loss	'squared_error'
hidden_layer_sizes	(100, ...)
activation	'relu'
solver	'adam'
alpha	0.0001
batch_size	'auto'
learning_rate	'constant'
learning_rate_init	0.001
power_t	0.5
max_iter	500
shuffle	True
random_state	42
tol	0.0001
verbose	False
warm_start	False
momentum	0.9
nesterovs_momentum	True
early_stopping	False
validation_fraction	0.1
beta_1	0.9
beta_2	0.999
epsilon	1e-08
n_iter_no_change	10
max_fun	15000

```python
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)
```

```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
```