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 № 6

по дисциплине «Технология машинного обучения»

Выполнил:
студент группы ИУ5-63Б Кузнецов В.А.
подпись, дата

Проверил:
Гапанюк Ю.Е.
подпись, дата

2024 г.

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:
    a. одну из моделей группы стекинга.
    b. модель многослойного персептрона. По желанию, вместо библиотеки scikit-learn возможно использование библиотек TensorFlow, PyTorch или других аналогичных библиотек.
    c. двумя методами на выбор из семейства МГУА (один из линейных методов COMBI / MULTI + один из нелинейных методов MIA / RIA) с использованием библиотеки gmdh.
    d. **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.

Текст программы:

Основные характеристики датасета

MedInc - медианный доход в районе

HouseAge - средний возраст домов в районе

AveRooms - среднее количество комнат на дом

AveBedrms - среднее количество спален на дом

Population - население района

AveOccup - среднее количество жителей на дом

Latitude - географическая широта района

Longitude - географическая долгота района

MedHouseVal - медианная стоимость домов в районе (целевая переменная)

Подготовка

```
!pip install gmdh
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_c
  and should_run_async(code)
Requirement already satisfied: gmdh in /usr/local/lib/python3.10/dist-packages (1.0.3)
Requirement already satisfied: docstring-inheritance in /usr/local/lib/python3.10/dist-packages (from gmdh) (2.2.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from gmdh) (1.25.2)
```

```
import numpy as np
import pandas as pd
from sklearn.datasets import fetch_california_housing
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score, mean_absolute_error
from sklearn.preprocessing import StandardScaler
from sklearn.neural_network import MLPRegressor
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import StackingRegressor
from sklearn.tree import DecisionTreeRegressor
import gmdh

RANDOM_STATE=123

import warnings
warnings.simplefilter('ignore')


# Загрузка данных
california = fetch_california_housing()
data = pd.DataFrame(data= np.c_[california['data'], california['target']],
                    columns= california['feature_names'] + ['target'])


data.head()
```

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	target
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-122.23	4.526
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-122.22	3.585
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.24	3.521
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	-122.25	3.413
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	-122.25	3.422

```
data.isnull().sum()
```

```
MedInc        0
HouseAge      0
AveRooms      0
AveBedrms     0
Population    0
AveOccup      0
Latitude      0
Longitude     0
```

```
target        0
dtype: int64
```

```
data.describe()
```

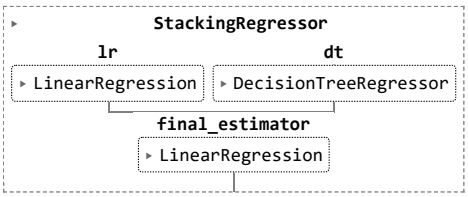
	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	target
count	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000
mean	3.870671	28.639486	5.429000	1.096675	1425.476744	3.070655	35.631861	-119.569704	2.068558
std	1.899822	12.585558	2.474173	0.473911	1132.462122	10.386050	2.135952	2.003532	1.153956
min	0.499900	1.000000	0.846154	0.333333	3.000000	0.692308	32.540000	-124.350000	0.149990
25%	2.563400	18.000000	4.440716	1.006079	787.000000	2.429741	33.930000	-121.800000	1.196000
50%	3.534800	29.000000	5.229129	1.048780	1166.000000	2.818116	34.260000	-118.490000	1.797000
75%	4.743250	37.000000	6.052381	1.099526	1725.000000	3.282261	37.710000	-118.010000	2.647250
max	15.000100	52.000000	141.909091	34.066667	35682.000000	1243.333333	41.950000	-114.310000	5.000010

Разделение на выборки

```
X = data.iloc[:, :-1]
y = data.target


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


# Масштабирование
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Обучение моделей

```
# Стекинг

estimators = [
    ('lr', LinearRegression()),
    ('dt', DecisionTreeRegressor(random_state=RANDOM_STATE))
]

stacking = StackingRegressor(estimators=estimators, final_estimator=LinearRegression())
stacking.fit(X_train_scaled, y_train)
```




```
# Модель многослойного персептрона

mlp = MLPRegressor(hidden_layer_sizes=(100, 50), max_iter=1000, random_state=RANDOM_STATE)
mlp.fit(X_train_scaled, y_train)
```

```
MLPRegressor(hidden_layer_sizes=(100, 50), max_iter=1000, random_state=123)
```

```
# Линейный метод COMBI

combi = gmdh.Combi()
combi.fit(X_train_scaled, y_train)
combi.get_best_polynomial()
```

```
'y = 0.8542*x1 + 0.1137*x2 - 0.321*x3 + 0.3143*x4 - 0.8748*x7 - 0.8305*x8 + 2.0719'
```

```
# Нелинейный метод MIA

mia = gmdh.Mia()
mia.fit(X_train_scaled, y_train)
```

```
<gmdh.gmdh.Mia at 0x7db1f02afb20>
```

Оценка моделей

```
y_pred_stacking = stacking.predict(X_test_scaled)
y_pred_mlp = mlp.predict(X_test_scaled)
y_pred_combi = combi.predict(X_test_scaled)
y_pred_mia = mia.predict(X_test_scaled)
```

```
# MAE
print(f"Stacking: {mean_absolute_error(y_test, y_pred_stacking):.4f}")
print(f"Perceptron: {mean_absolute_error(y_test, y_pred_mlp):.4f}")
print(f"COMBI: {mean_absolute_error(y_test, y_pred_combi):.4f}")
print(f"MIA: {mean_absolute_error(y_test, y_pred_mia):.4f}")
```

```
Stacking: 0.4291
Perceptron: 0.3503
COMBI: 0.5289
MIA: 0.5820
```

```
# R^2
print(f"Stacking: {r2_score(y_test, y_pred_stacking):.4f}")
print(f"Perceptron: {r2_score(y_test, y_pred_mlp):.4f}")
print(f"COMBI: {r2_score(y_test, y_pred_combi):.4f}")
print(f"MIA: {r2_score(y_test, y_pred_mia):.4f}")
```

```
Stacking: 0.7083
Perceptron: 0.8088
COMBI: 0.6058
MIA: 0.5343
```

Вывод

Многослойный перцептрон показал наилучшие результаты среди всех моделей как по MAE, так и по R^2 для данной задачи. Однако и на его обучение ушло больше всего времени.

Стекинг модель с простой архитектурой оказалась на втором месте после персептрона. Если подобрать другие модели и параметры для них она могла бы показать себя лучше.

COMBI и MIA показали результаты хуже. При этом нелинейный метод, способный улавливать более сложные зависимости показал себя хуже линейного.