Московский государственный технический университет им. Н. Э. Баумана

Курс «Технологии машинного обучения»

Отчёт по лабораторной работе №6

Выполнил:
Лупарев С. В.
группа ИУ5-63Б

Проверил:
Гапанюк Ю.Е.

Дата:

Подпись:

Дата:

Подпись:

Москва, 2025 г.

Цель лабораторной работы

Цель: изучение ансамблей моделей машинного обучения.

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. 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).

Код программы и экранные формы

The parameters included are :

GRE Scores (out of 340)

TOEFL Scores (out of 120)

University Rating (out of 5)

Statement of Purpose and Letter of Recommendation Strength (out of 5)

Undergraduate GPA (out of 10)

Research Experience (either 0 or 1)

Chance of Admit (ranging from 0 to 1)

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import BaggingRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import ExtraTreesRegressor
from sklearn.ensemble import AdaBoostRegressor
from sklearn.ensemble import GradientBoostingRegressor
# Метрики
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
import math
from itertools import combinations
from sklearn.ensemble import StackingRegressor, RandomForestRegressor
from sklearn.linear_model import Ridge
from sklearn.svm import SVR
from sklearn.neural_network import MLPRegressor
from sklearn.preprocessing import StandardScaler
from gmdh import Combi, Ria, CriterionType, Criterion
import gmdh
```

```
df = pd.read_csv('Admission_Predict_Ver1.1.csv')
df.columns = df.columns.str.strip()
df.head()
```

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
0	1	337	118	4	4.5	4.5	9.65	1	0.92
1	2	324	107	4	4.0	4.5	8.87	1	0.76
2	3	316	104	3	3.0	3.5	8.00	1	0.72
3	4	322	110	3	3.5	2.5	8.67	1	0.80
4	5	314	103	2	2.0	3.0	8.21	0	0.65

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Serial No.         500 non-null    int64  
 1   GRE Score          500 non-null    int64  
 2   TOEFL Score        500 non-null    int64  
 3   University Rating  500 non-null    int64  
 4   SOP                500 non-null    float64
 5   LOR                500 non-null    float64
 6   CGPA               500 non-null    float64
 7   Research           500 non-null    int64  
 8   Chance of Admit    500 non-null    float64
dtypes: float64(4), int64(5)
memory usage: 35.3 KB
```

```
df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
Serial No.	500.0	250.50000	144.481833	1.00	125.7500	250.50	375.25	500.00
GRE Score	500.0	316.47200	11.295148	290.00	308.0000	317.00	325.00	340.00
TOEFL Score	500.0	107.19200	6.081868	92.00	103.0000	107.00	112.00	120.00
University Rating	500.0	3.11400	1.143512	1.00	2.0000	3.00	4.00	5.00
SOP	500.0	3.37400	0.991004	1.00	2.5000	3.50	4.00	5.00
LOR	500.0	3.48400	0.925450	1.00	3.0000	3.50	4.00	5.00
CGPA	500.0	8.57644	0.604813	6.80	8.1275	8.56	9.04	9.92
Research	500.0	0.56000	0.496884	0.00	0.0000	1.00	1.00	1.00
Chance of Admit	500.0	0.72174	0.141140	0.34	0.6300	0.72	0.82	0.97

```
df = df.drop('Serial No.', axis=1)
```

```
df.head()
```

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
0	337	118	4	4.5	4.5	9.65	1	0.92
1	324	107	4	4.0	4.5	8.87	1	0.76
2	316	104	3	3.0	3.5	8.00	1	0.72
3	322	110	3	3.5	2.5	8.67	1	0.80
4	314	103	2	2.0	3.0	8.21	0	0.65

```
X = df.drop('Chance of Admit', axis=1)
y = df['Chance of Admit']
```

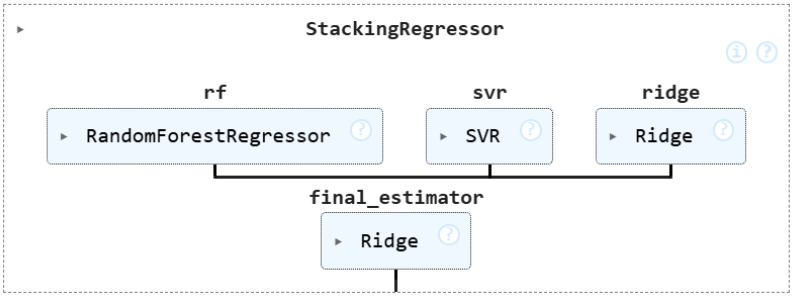
```
# 3. Разделение на train и test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=52)
print(X_train.shape[0])
print(X_test.shape[0])
```

```
375
125
```

```
X_train_np = X_train.values
X_test_np = X_test.values
y_train_np = y_train.values
y_test_np = y_test.values

# 4.1. Модель Стекинга
from sklearn.ensemble import StackingRegressor, RandomForestRegressor
from sklearn.linear_model import Ridge
from sklearn.svm import SVR


estimators = [
    ('rf', RandomForestRegressor(n_estimators=50, random_state=52, n_jobs=-1)), # Уменьшим n_estimators для скорости
    ('svr', SVR(kernel='rbf')), # Нелинейная модель
    ('ridge', Ridge(random_state=52)) # Линейная модель
]
stacking_model = StackingRegressor(
    estimators=estimators,
    final_estimator=Ridge(random_state=52),
    cv=5,
    n_jobs=-1
)
stacking_model.fit(X_train, y_train)
```




```
# 4.2. Многослойный Перцептрон
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train_np)
X_test_scaled = scaler.transform(X_test_np)

mlp_model = MLPRegressor(
    hidden_layer_sizes=(100, 50),
    activation='relu',
    solver='adam',
    max_iter=1000,
    random_state=42,
    alpha=0.0001,
    early_stopping=True,
    validation_fraction=0.1,
    n_iter_no_change=10,
    verbose=False
)
mlp_model.fit(X_train_scaled, y_train_np)
```

MLPRegressor

```
MLPRegressor(early_stopping=True, hidden_layer_sizes=(100, 50), max_iter=1000,
             random_state=42)
```

```python
# 4.3. COMBI
#X_train2, X_test2, y_train2, y_test2 = gmdh.split_data(X, y, test_size=0.25, random_state=52)
combi_model = Combi()
combi_model.fit(X_train_np, y_train_np, verbose=1)
# combi_model.fit(X_train2, y_train2, verbose=1)
# combi_model.fit(X_train_np[:3], y_train_np[:3])
print(f"Лучший полином COMBI: {combi_model.get_best_polynomial()}")
```

```
LEVEL 1  [=========================] 100% [00m:00s] (7 combinations) error=0.900323
LEVEL 2  [=========================] 100% [00m:00s] (21 combinations) error=0.825566
LEVEL 3  [=========================] 100% [00m:00s] (35 combinations) error=0.776935
LEVEL 4  [=========================] 100% [00m:00s] (35 combinations) error=0.772056
LEVEL 5  [=========================] 100% [00m:00s] (21 combinations) error=0.773736
Лучший полином COMBI: y = 0.0124*x3 + 0.0167*x5 + 0.1656*x6 + 0.0258*x7 - 0.8127
```

```python
# 4.3. RIA
ria_model = Ria()
reg_criterion = Criterion(criterion_type=CriterionType.REGULARITY)
ria_model.fit(X_train_np, y_train_np, criterion=reg_criterion, k_best=5, p_average=3, verbose=1)
print(f"Лучший полином RIA: {ria_model.get_best_polynomial()}")
```

```
LEVEL 1  [=========================] 100% [00m:00s] (21 combinations) error=0.839272
LEVEL 2  [=========================] 100% [00m:00s] (35 combinations) error=0.785363
LEVEL 3  [=========================] 100% [00m:00s] (35 combinations) error=0.771502
LEVEL 4  [=========================] 100% [00m:00s] (35 combinations) error=0.769948
LEVEL 5  [=========================] 100% [00m:00s] (35 combinations) error=0.768699
LEVEL 6  [=========================] 100% [00m:00s] (35 combinations) error=0.76869
LEVEL 7  [=========================] 100% [00m:00s] (35 combinations) error=0.768691
Лучший полином RIA: f1 = 0.1052*x6 + 0.0852*x7 - 0.0067*x6*x7 + 0.0057*x6^2 - 0.615
f2 = 0.0262*x5 + 0.8345*f1 - 0.0637*x5*f1 + 0.0057*x5^2 + 0.2019*f1^2 + 0.0103
f3 = 1.1117*f2 + 0.048*x7*f2 - 0.0313*x7^2 - 0.1014*f2^2 - 0.0299
f4 = 0.0207*x1 + 0.7454*f3 - 6.47967e-05*x1*f3 - 2.75421e-05*x1^2 + 0.0272*f3^2 - 3.5936
f5 = - 0.0216*x5 + 1.081*f4 + 0.0249*x5*f4 + 0.0015*x5^2 - 0.1393*f4^2 + 0.0076
y = - 0.0005*x5 + 0.9989*f5 + 0.003*x5*f5 - 0.0002*x5^2 - 0.0064*f5^2 + 0.0014
```

```python
# 5. Оценка качества моделей
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error

models_to_evaluate = {
    "Stacking": {'model': stacking_model, 'scaled': False},
    "MLP": {'model': mlp_model, 'scaled': True},
    "GMDH_COMBI" : {'model': combi_model, 'scaled': False},
    "GMDH_RIA" : {'model': ria_model, 'scaled': False}
}

metrics_results = {}

for name, info in models_to_evaluate.items():
    model = info['model']
    is_scaled = info['scaled']
    X_test_current = X_test_scaled if is_scaled else X_test_np
    y_pred = model.predict(X_test_current)

    # Расчет метрик
    r2 = r2_score(y_test_np, y_pred)
    mae = mean_absolute_error(y_test_np, y_pred)
    mse = mean_squared_error(y_test_np, y_pred)
    rmse = math.sqrt(mse)

    metrics_results[name] = {
        'R^2': r2,
        'MAE': mae,
        'MSE': mse,
        'RMSE': rmse
    }


metrics_df = pd.DataFrame(metrics_results).T
metrics_df = metrics_df[['R^2', 'MAE', 'MSE', 'RMSE']]
print(metrics_df.round(4))
```

```
               R^2     MAE     MSE    RMSE
Stacking    0.8511  0.0421  0.0030  0.0552
MLP         0.7696  0.0536  0.0047  0.0687
GMDH_COMBI  0.8415  0.0443  0.0032  0.0570
GMDH_RIA    0.8542  0.0426  0.0030  0.0546
```