Московский государственный технический университет им. Н. Э. Баумана

Курс «Технологии машинного обучения»

Отчёт по лабораторной работе №5

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 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:

- две модели группы бэггинга (бэггинг или случайный лес или сверхслучайные деревья);
- AdaBoost;
- градиентный бустинг.

5. Оцените качество моделей с помощью одной из подходящих для задачи метрик. Сравните качество полученных моделей.

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)

[9]:
```
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
```

10]:
```
df = pd.read_csv('Admission_Predict_Ver1.1.csv')
df.columns = df.columns.str.strip()
df.head()
```

10]:

	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

[11]:
```
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
```

[12]:
```
df.describe().T
```

[12]:

	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

[13]:
```
df = df.drop('Serial No.', axis=1)
```

[14]:
```
df.head()
```

[14]:

	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

[15]:
```
X = df.drop('Chance of Admit', axis=1)
y = df['Chance of Admit']
```

[16]:
```
# 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
```

```python
# 4. Обучение ансамблевых моделей
bagging_model = BaggingRegressor(random_state=52, n_estimators=100, n_jobs=-1)
random_forest_model = RandomForestRegressor(random_state=52, n_estimators=100, n_jobs=-1)
extra_trees_model = ExtraTreesRegressor(random_state=52, n_estimators=100, n_jobs=-1)

adaboost_model = AdaBoostRegressor(random_state=52, n_estimators=100)

gradient_boosting_model = GradientBoostingRegressor(random_state=42, n_estimators=100)

models = {
    "Bagging": bagging_model,
    "Random Forest": random_forest_model,
    "Extra Trees": extra_trees_model,
    "AdaBoost": adaboost_model,
    "Gradient Boosting": gradient_boosting_model
}

results = {}

for name, model in models.items():
    model.fit(X_train, y_train)

    y_pred = model.predict(X_test)

    # Сохранение результатов
    results[name] = y_pred
```

```python
# 5. Оценка и сравнение моделей

# Функция для расчета Somers'D
def calculate_somers_d(y_true, y_pred):
    y_true = np.asarray(y_true)
    y_pred = np.asarray(y_pred)

    n_concordant = 0
    n_discordant = 0
    n_tied_y = 0

    indices = list(combinations(range(len(y_true)), 2))

    for i, j in indices:
        y_true_diff = y_true[i] - y_true[j]
        y_pred_diff = y_pred[i] - y_pred[j]

        if y_true_diff != 0:
            if y_pred_diff != 0:
                if np.sign(y_true_diff) == np.sign(y_pred_diff):
                    n_concordant += 1
                else:
                    n_discordant += 1
        else:
             n_tied_y +=1

    if n_concordant + n_discordant + n_tied_y == 0:
        return 0
    else:
        return (n_concordant - n_discordant) / (n_concordant + n_discordant + n_tied_y)


metrics_results = {}
```

```python
metrics_results = {}

for name, data in results.items():
    y_pred = data

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


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

```python
# Сравнение метрик
metrics_df = pd.DataFrame(metrics_results).T
metrics_df = metrics_df[['R^2', 'MAE', 'MSE', 'RMSE', 'Somers\' D']]
print(metrics_df.round(4))
print("Лучшая модель по R^2, MAE, RMSE: Extra Trees")
print("Лучшая модель по Somers' D: Gradient Boosting")
```

```
                    R^2     MAE     MSE    RMSE  Somers' D
Bagging          0.8090  0.0465  0.0039  0.0625     0.7383
Random Forest    0.8123  0.0462  0.0038  0.0620     0.7391
Extra Trees      0.8251  0.0438  0.0036  0.0598     0.7546
AdaBoost         0.7666  0.0562  0.0048  0.0691     0.7434
Gradient Boosting  0.8236  0.0451  0.0036  0.0601     0.7553
Лучшая модель по R^2, MAE, RMSE: Extra Trees
Лучшая модель по Somers' D: Gradient Boosting
```

В моем случае комбинированные модели проигрывают линейной регрессии, по-моему мнению это происходит из-за того,

что в реальной жизни при приеме в магистратуру общая вступительная оценка студента считается как линейная комбинация показателей