Московский государственный технический университет им. Н. Э. Баумана

Курс «Технологии машинного обучения»

Отчёт по лабораторной работе №3

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

Проверил:
Гапанюк Ю.Е.

Дата:

Подпись:

Дата:

Подпись:

Москва, 2025 г.

Цель лабораторной работы

Цель: изучение способов подготовки выборки и подбора гиперпараметров на примере метода ближайших соседей.

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. С использованием метода train_test_split разделите выборку на обучающую и тестовую.
4. Обучите модель ближайших соседей для произвольно заданного гиперпараметра K. Оцените качество модели с помощью подходящих для задачи метрик.
5. Произведите подбор гиперпараметра K с использованием GridSearchCV и RandomizedSearchCV и кросс-валидации, оцените качество оптимальной модели. Используйте не менее двух стратегий кросс-валидации.
6. Сравните метрики качества исходной и оптимальной 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)

[29]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, KFold, LeaveOneOut, GridSearchCV, RandomizedSearchCV
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
import matplotlib.pyplot as plt
import seaborn as sns
```

[30]:
```
df = pd.read_csv('Admission_Predict_Ver1.1.csv')
df.columns = df.columns.str.strip()
df.head()
```

[30]:

	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

[31]:
```
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
```

[32]:
```
df.describe().T
```

[32]:

	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

[33]:
```
df = df.drop('Serial No.', axis=1)
```

[34]:
```
df.head()
```

[34]:

	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

[35]:
```
X = df.drop('Chance of Admit', axis=1)
y = df['Chance of Admit']
```

[36]:
```
# 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
```

[37]:
```
# 4. Обучение базовой KNN и оценка качества
k_base = 7
knn_base = KNeighborsRegressor(n_neighbors=k_base)
knn_base.fit(X_train, y_train)
y_pred_base = knn_base.predict(X_test)

# Оценка качества
r2_base = r2_score(y_test, y_pred_base)
mse_base = mean_squared_error(y_test, y_pred_base)
rmse_base = np.sqrt(mse_base)
mae_base = mean_absolute_error(y_test, y_pred_base)

print("r2_base: ", r2_base)
print("mse_base: ", mse_base)
print("rmse_base: ", rmse_base)
print("mae_base: ", mae_base)
```

```
r2_base:  0.7906582679995813
mse_base:  0.004285142857142855
rmse_base:  0.06546100256750469
mae_base:  0.04983999999999999
```

[38]:
```python
# 5. Подбор гиперпараметра K
k_range = np.arange(1, 31)
param_search_grid = {'n_neighbors': k_range}

# Стратегии кросс-валидации
cv_k5 = KFold(n_splits=5, shuffle=True, random_state=52)
cv_k10 = KFold(n_splits=10, shuffle=True, random_state=52)
cv_loo = LeaveOneOut() # Оставим закомментированным, т.к. может быть долго

cv_strategies = {'KFold-5': cv_k5, 'KFold-10': cv_k10, 'LOO': cv_loo}
results = {}

# Используем GridSearchCV
print('Используем GridSearchCV\n')
for name, cv_method in cv_strategies.items():
    print(f"\n-- GridSearchCV, CV: {name} --")
    current_scoring = 'r2' if name != 'LOO' else 'neg_mean_squared_error' # добавили из-за loo
    search = GridSearchCV(KNeighborsRegressor(), param_search_grid, cv=cv_method, scoring=current_scoring, n_jobs=-1)
    search.fit(X_train, y_train)

    best_k = search.best_params_['n_neighbors']
    best_score_cv = search.best_score_

    # Оценка модели
    best_model = search.best_estimator_
    y_pred_best = best_model.predict(X_test)
    r2_test = r2_score(y_test, y_pred_best)
    mse_test = mean_squared_error(y_test, y_pred_best)
    rmse_test = np.sqrt(mean_squared_error(y_test, y_pred_best))
    mae_test = mean_absolute_error(y_test, y_pred_best)

    results[f'GridSearch_{name}'] = {'k': best_k, 'r2_cv': best_score_cv, 'r2_test': r2_test, 'mse_test': mse_test, 'rmse_test': rmse_test, 'mae_test': mae_test}

    print("Лучший K: ", best_k)
    print(f"Лучший {current_scoring}: ", best_score_cv)
    print("R2: ", r2_test)
    print("RMSE: ", rmse_test)
    print("MAE: ", mae_test)
```

```python
# Используем RandomizedSearchCV
print('Используем RandomizedSearchCV\n')
n_iterations = 20
param_random_dist = {'n_neighbors': k_range}

for name, cv_method in cv_strategies.items():
    print(f"\n-- RandomizedSearchCV, CV: {name} --")
    current_scoring = 'r2' if name != 'LOO' else 'neg_mean_squared_error' # добавили из-за loo
    search = RandomizedSearchCV(KNeighborsRegressor(), param_random_dist, n_iter=n_iterations,
                                cv=cv_method, scoring=current_scoring, random_state=42, n_jobs=-1)
    search.fit(X_train, y_train)

    best_k = search.best_params_['n_neighbors']
    best_score_cv = search.best_score_

    # Оценка модели
    best_model = search.best_estimator_
    y_pred_best = best_model.predict(X_test)
    r2_test = r2_score(y_test, y_pred_best)
    mse_test = mean_squared_error(y_test, y_pred_best)
    rmse_test = np.sqrt(mean_squared_error(y_test, y_pred_best))
    mae_test = mean_absolute_error(y_test, y_pred_best)

    results[f'RandomSearch_{name}'] = {'k': best_k, 'r2_cv': best_score_cv, 'r2_test': r2_test, 'mse_test': mse_test, 'rmse_test': rmse_test, 'mae_test': mae_test}

    print("Лучший K: ", best_k)
    print(f"Лучший {current_scoring}: ", best_score_cv)
    print("R2: ", r2_test)
    print("RMSE: ", rmse_test)
    print("MAE: ", mae_test)
```

```
Используем GridSearchCV


-- GridSearchCV, CV: KFold-5 --
Лучший K:  11
Лучший r2:  0.7213828555453693
R2:  0.7887916366522952
RMSE:  0.06575220198682095
MAE:  0.050116363636363646

-- GridSearchCV, CV: KFold-10 --
Лучший K:  15
Лучший r2:  0.7119321920518149
R2:  0.7948854877183504
RMSE:  0.06479670773529572
MAE:  0.04939733333333332

-- GridSearchCV, CV: LOO --
Лучший K:  19
Лучший neg_mean_squared_error:  -0.00550940055401662
R2:  0.7918916582159906
RMSE:  0.06526787758825428
MAE:  0.05068631578947369
Используем RandomizedSearchCV


-- RandomizedSearchCV, CV: KFold-5 --
Лучший K:  16
Лучший r2:  0.7211137580803942
R2:  0.7931726229967958
RMSE:  0.0650666965505396
MAE:  0.04986999999999999

-- RandomizedSearchCV, CV: KFold-10 --
Лучший K:  17
Лучший r2:  0.7104187162752831
R2:  0.7914582870529977
RMSE:  0.06533580015586339
MAE:  0.05062588235294117
```

```
-- GridSearchCV, CV: LOO --
Лучший K:  19
Лучший neg_mean_squared_error:  -0.00550940055401662
R2:  0.7918916582159906
RMSE:  0.06526787758825428
MAE:  0.05068631578947369
Используем RandomizedSearchCV


-- RandomizedSearchCV, CV: KFold-5 --
Лучший K:  16
Лучший r2:  0.7211137580803942
R2:  0.7931726229967958
RMSE:  0.0650666965505396
MAE:  0.04986999999999999

-- RandomizedSearchCV, CV: KFold-10 --
Лучший K:  17
Лучший r2:  0.7104187162752831
R2:  0.7914582870529977
RMSE:  0.06533580015586339
MAE:  0.05062588235294117

-- RandomizedSearchCV, CV: LOO --
Лучший K:  16
Лучший neg_mean_squared_error:  -0.005511078125
R2:  0.7931726229967958
RMSE:  0.0650666965505396
MAE:  0.04986999999999999
```

[42]:
```python
# 6. Итоговое сравнение метрик
base_metrics = {
    'k': k_base,
    'r2_test': r2_base,
    'rmse_test': rmse_base,
    'mae_test': mae_base
}
best_metrics_overall = base_metrics.copy()
best_model_origin = "Базовая модель"


for name, metrics_found in results.items():
    if metrics_found['r2_test'] > best_metrics_overall['r2_test']:
        best_metrics_overall = metrics_found.copy()
        best_model_origin = name

print("Результаты базовой модели")
print("r2:   ", r2_base)
print("mse:  ", mse_base)
print("rmse: ", rmse_base)
print("mae:  ", mae_base)

print(f"\nРезультаты лучшей модели: {best_model_origin}")
print("r2:   ", best_metrics_overall['r2_test'])
print("mse:  ", best_metrics_overall['mse_test'])
print("rmse: ", best_metrics_overall['rmse_test'])
print("mae:  ", best_metrics_overall['mae_test'])
```

```
Результаты базовой модели
r2:    0.7906582679995813
mse:   0.004285142857142855
rmse:  0.06546100256750469
mae:   0.04983999999999999

Результаты лучшей модели: GridSearch_KFold-10
r2:    0.7948854877183504
mse:   0.004198613333333332
rmse:  0.06479670773529572
mae:   0.04939733333333332
```

Вывод: смогли немного улучшить предсказательную способность модели

Незначительность улучшения может быть связана с малым количеством данных