Московский государственный технический

университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Технологии машинного обучения»

Лабораторная работа №3

Выполнила:
студентка ИУ5-62Б
Федосеева Елизавета

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю.Е.

Москва, 2022 г.

Задание:

1. Выберите набор данных (датасет) для решения задачи классификации или регрессии.
2. С использованием метода train_test_split разделите выборку на обучающую и тестовую.
3. Обучите модель ближайших соседей для произвольно заданного гиперпараметра K. Оцените качество модели с помощью подходящих для задачи метрик.
4. Произведите подбор гиперпараметра K с использованием GridSearchCV и/или RandomizedSearchCV и кросс-валидации, оцените качество оптимальной модели. Желательно использование нескольких стратегий кросс-валидации.
5. Сравните метрики качества исходной и оптимальной моделей.

Лабораторая работа №3: Подготовка обучающей и тестовой выборки, кросс-валидация и подбор гиперпараметров на примере метода ближайших соседей.

0) Подключение библиотек, загрузка и очистка датасета, кодирование категориальных признаков

In [1]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, GridSearchCV, RandomizedSearchCV
from sklearn.neighbors import KNeighborsRegressor
from sklearn.preprocessing import MinMaxScaler, StandardScaler
from matplotlib import pyplot as plt
import seaborn as sns
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from warnings import simplefilter

simplefilter('ignore')
```

In [2]:
```
df = pd.read_csv('student-mat.csv')
```

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 395 entries, 0 to 394
Data columns (total 33 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   school      395 non-null    object
 1   sex         395 non-null    object
 2   age         395 non-null    int64 
 3   address     395 non-null    object
 4   famsize     395 non-null    object
 5   Pstatus     395 non-null    object
 6   Medu        395 non-null    int64 
 7   Fedu        395 non-null    int64 
 8   Mjob        395 non-null    object
 9   Fjob        395 non-null    object
 10  reason      395 non-null    object
 11  guardian    395 non-null    object
 12  traveltime  395 non-null    int64 
 13  studytime   395 non-null    int64 
 14  failures    395 non-null    int64 
 15  schoolsup   395 non-null    object
 16  famsup      395 non-null    object
 17  paid        395 non-null    object
 18  activities  395 non-null    object
 19  nursery     395 non-null    object
 20  higher      395 non-null    object
 21  internet    395 non-null    object
 22  romantic    395 non-null    object
 23  famrel      395 non-null    int64 
 24  freetime    395 non-null    int64 
 25  goout       395 non-null    int64 
 26  Dalc        395 non-null    int64 
 27  Walc        395 non-null    int64 
 28  health      395 non-null    int64 
 29  absences    395 non-null    int64 
 30  G1          395 non-null    int64 
 31  G2          395 non-null    int64 
 32  G3          395 non-null    int64 
dtypes: int64(16), object(17)
memory usage: 102.0+ KB
```

In [4]:
```
df.head()
```

Out[4]:

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	famrel	freetime	goout	Dalc	Walc	health	absences	G1
0	GP	F	18	U	GT3	A	4	4	at_home	teacher	...	4	3	4	1	1	3	6	5
1	GP	F	17	U	GT3	T	1	1	at_home	other	...	5	3	3	1	1	3	4	5
2	GP	F	15	U	LE3	T	1	1	at_home	other	...	4	3	2	2	3	3	10	7
3	GP	F	15	U	GT3	T	4	2	health	services	...	3	2	2	1	1	5	2	15

4	GP	F	16	U	GT3	T	3	3	other	other	...	4	3	2	1	2	5	4	6

5 rows × 33 columns

Проверим наличие пустых значений в датасете:

In [5]:
```
df.isnull().sum()
```

Out[5]:
```
school        0
sex           0
age           0
address       0
famsize       0
Pstatus       0
Medu          0
Fedu          0
Mjob          0
Fjob          0
reason        0
guardian      0
traveltime    0
studytime     0
failures      0
schoolsup     0
famsup        0
paid          0
activities    0
nursery       0
higher        0
internet      0
romantic      0
famrel        0
freetime      0
goout         0
Dalc          0
Walc          0
health        0
absences      0
G1            0
G2            0
G3            0
dtype: int64
```

Определим категориальные признаки и закодируем их.

In [6]:
```
category_cols = ['school', 'sex', 'address', 'famsize', 'Pstatus', 'Mjob', 'Fjob', 'reason', 'guardian', 'schools
                 'paid', 'activities', 'nursery', 'higher', 'internet', 'romantic']

print('Количество уникальных значений\n')
for col in category_cols:
    print(f'{col}: {df[col].unique().size}')
```

```
Количество уникальных значений

school: 2
sex: 2
address: 2
famsize: 2
Pstatus: 2
Mjob: 5
Fjob: 5
reason: 4
guardian: 3
schoolsup: 2
famsup: 2
paid: 2
activities: 2
nursery: 2
higher: 2
internet: 2
romantic: 2
```

In [16]:
```
df = pd.get_dummies(df, columns=category_cols)
```

In [17]:
```
df.head()
```

Out[17]:

	age	Medu	Fedu	traveltime	studytime	failures	famrel	freetime	goout	Dalc	...	activities_no	activities_yes	nursery_no	nursery_yes	hi
0	18	4	4	2	2	0	4	3	4	1	...	1	0	0	1	
1	17	1	1	1	2	0	5	3	3	1	...	1	0	1	0	
2	15	1	1	1	2	3	4	3	2	2	...	1	0	0	1	
3	15	4	2	1	3	0	3	2	2	1	...	0	1	0	1	
4	16	3	3	1	2	0	4	3	2	1	...	1	0	0	1	

5 rows × 102 columns

In [18]:
```
df.describe()
```

Out[18]:

	age	Medu	Fedu	traveltime	studytime	failures	famrel	freetime	goout	Dalc	...	activities_n
count	395.000000	395.000000	395.000000	395.000000	395.000000	395.000000	395.000000	395.000000	395.000000	395.000000	...	395.00000
mean	16.696203	2.749367	2.521519	1.448101	2.035443	0.334177	3.944304	3.235443	3.108861	1.481013	...	0.49113
std	1.276043	1.094735	1.088201	0.697505	0.839240	0.743651	0.896659	0.998862	1.113278	0.890741	...	0.50055
min	15.000000	0.000000	0.000000	1.000000	1.000000	0.000000	1.000000	1.000000	1.000000	1.000000	...	0.00000
25%	16.000000	2.000000	2.000000	1.000000	1.000000	0.000000	4.000000	3.000000	2.000000	1.000000	...	0.00000
50%	17.000000	3.000000	2.000000	1.000000	2.000000	0.000000	4.000000	3.000000	3.000000	1.000000	...	0.00000
75%	18.000000	4.000000	3.000000	2.000000	2.000000	0.000000	5.000000	4.000000	4.000000	2.000000	...	1.00000
max	22.000000	4.000000	4.000000	4.000000	4.000000	3.000000	5.000000	5.000000	5.000000	5.000000	...	1.00000

8 rows × 102 columns

1) Разделение выборки на обучающую и на тестовую

In [19]:
```
y = df['G3']
X = df.drop('G3', axis=1)
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=3)
x_train
```

Out[19]:

	age	Medu	Fedu	traveltime	studytime	failures	famrel	freetime	goout	Dalc	...	activities_no	activities_yes	nursery_no	nursery_yes
65	16	4	3	3	2	0	5	4	3	1	...	0	1	0	1
248	18	3	3	1	2	1	4	3	3	1	...	1	0	0	1
186	16	1	2	1	1	0	3	3	3	1	...	0	1	0	1
366	18	4	4	2	3	0	4	2	2	2	...	1	0	0	1
70	16	3	1	2	4	0	4	3	2	1	...	1	0	0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
256	17	4	2	1	4	0	4	2	3	1	...	0	1	0	1
131	15	1	1	3	1	0	4	3	3	1	...	0	1	1	0
249	16	0	2	1	1	0	4	3	2	2	...	1	0	1	0
152	15	3	3	2	3	2	4	2	1	2	...	0	1	0	1
362	18	3	3	2	2	0	4	3	2	1	...	1	0	0	1

276 rows × 101 columns

Масштабирование данных

In [21]:
```
scaler = MinMaxScaler().fit(x_train)
x_train = pd.DataFrame(scaler.transform(x_train), columns=x_train.columns)
x_test = pd.DataFrame(scaler.transform(x_test), columns=x_train.columns)
x_train.describe()
```

Out[21]:

	age	Medu	Fedu	traveltime	studytime	failures	famrel	freetime	goout	Dalc	...	activities_n
count	276.000000	276.000000	276.000000	276.000000	276.000000	276.000000	276.000000	276.000000	276.000000	276.000000	...	276.00000
mean	0.280193	0.683877	0.620471	0.164251	0.332126	0.109903	0.735507	0.555254	0.525362	0.120471	...	0.51449
std	0.209362	0.276213	0.269220	0.240358	0.273397	0.243706	0.216938	0.255627	0.276816	0.214746	...	0.50069

min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.00000
25%	0.166667	0.500000	0.500000	0.000000	0.000000	0.000000	0.750000	0.500000	0.250000	0.000000	...	0.00000
50%	0.333333	0.750000	0.500000	0.000000	0.333333	0.000000	0.750000	0.500000	0.500000	0.000000	...	1.00000
75%	0.500000	1.000000	0.750000	0.333333	0.333333	0.000000	1.000000	0.750000	0.750000	0.250000	...	1.00000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	...	1.00000

8 rows × 101 columns

3) Обучение KNN с произвольным k

In [22]:
```
def print_metrics(y_test, y_pred):
    print(f"R^2: {r2_score(y_test, y_pred)}")
    print(f"MSE: {mean_squared_error(y_test, y_pred)}")
    print(f"MAE: {mean_absolute_error(y_test, y_pred)}")

def print_cv_result(cv_model, x_test, y_test):
    print(f'Оптимизация метрики {cv_model.scoring}: {cv_model.best_score_}')
    print(f'Лучший параметр: {cv_model.best_params_}')
    print('Метрики на тестовом наборе')
    print_metrics(y_test, cv_model.predict(x_test))
    print()
```

In [24]:
```
base_k = 8
base_knn = KNeighborsRegressor(n_neighbors=base_k)
base_knn.fit(x_train, y_train)
y_pred_base = base_knn.predict(x_test)
print(f'Test metrics for KNN with k={base_k}\n')
print_metrics(y_test, y_pred_base)
```

```
Test metrics for KNN with k=8

R^2: 0.09924947413071761
MSE: 19.156643907563026
MAE: 3.3098739495798317
```

4) Кросс-валидация

In [25]:
```
metrics = ['r2', 'neg_mean_squared_error', 'neg_mean_absolute_error']
cv_values = [5, 10]

for cv in cv_values:
    print(f'Результаты кросс-валидации при cv={cv}\n')
    for metric in metrics:
        params = {'n_neighbors': range(1, 30)}
        knn_cv = GridSearchCV(KNeighborsRegressor(), params, cv=cv, scoring=metric, n_jobs=-1)
        knn_cv.fit(x_train, y_train)
        print_cv_result(knn_cv, x_test, y_test)
```

```
Результаты кросс-валидации при cv=5

Оптимизация метрики r2: 0.12675188436118678
Лучший параметр: {'n_neighbors': 9}
Метрики на тестовом наборе
R^2: 0.11830778342492743
MSE: 18.751322751322746
MAE: 3.303454715219421

Оптимизация метрики neg_mean_squared_error: -17.854033990700653
Лучший параметр: {'n_neighbors': 9}
Метрики на тестовом наборе
R^2: 0.11830778342492743
MSE: 18.751322751322746
MAE: 3.303454715219421

Оптимизация метрики neg_mean_absolute_error: -3.166931216931217
Лучший параметр: {'n_neighbors': 27}
Метрики на тестовом наборе
R^2: 0.1279507568047641
MSE: 18.546241541884243
MAE: 3.318082788671025

Результаты кросс-валидации при cv=10

Оптимизация метрики r2: 0.11137702815765167
Лучший параметр: {'n_neighbors': 26}
```

```
Метрики на тестовом наборе
R^2: 0.14156097133621293
MSE: 18.256787330316744
MAE: 3.2860374919198443

Оптимизация метрики neg_mean_squared_error: -18.218296222810107
Лучший параметр: {'n_neighbors': 12}
Метрики на тестовом наборе
R^2: 0.15446659616338154
MSE: 17.982317927170868
MAE: 3.2710084033613445

Оптимизация метрики neg_mean_absolute_error: -3.1924055829228246
Лучший параметр: {'n_neighbors': 29}
Метрики на тестовом наборе
R^2: 0.12319996493398677
MSE: 18.64727864986661
MAE: 3.325702694871052
```

In [27]:

```
best_k = 29
y_pred_best = KNeighborsRegressor(n_neighbors=best_k).fit(x_train, y_train).predict(x_test)
```

5) Сравнение исходной и оптимальной моделей

In [28]:

```
print('Basic model\n')
print_metrics(y_test, y_pred_base)
print('_________________________')
print('\nOptimal model\n')
print_metrics(y_test, y_pred_best)
```

```
Basic model

R^2: 0.09924947413071761
MSE: 19.156643907563026
MAE: 3.3098739495798317
_________________________

Optimal model

R^2: 0.12319996493398677
MSE: 18.64727864986661
MAE: 3.325702694871052
```