Московский государственный технический университет им. Н.Э. Баумана

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

Курс «Технологии машинного обучения»

Отчет по лабораторной работе №3

«Подготовка обучающей и тестовой выборки, кросс-валидация и подбор гиперпараметров на примере метода ближайших соседей»

Выполнил:
студент группы ИУ5-62Б
Перова Анна

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.

Описание датасета

Датасет `SHOP-SALES.xlsx` содержит информацию о продажах магазина.

Параметры покупки:

- *Date* - дата покупки,
- *Hour* - час совершения покупки,
- *Product* - тип купленного товара,
- *Gender* - пол,
- *Color* - цвет товара,
- *Size* - размер товара,
- *Price* - цена товара без учета НДС,
- *Vat* - НДС,
- *Total* - Цена товара с НДС.

Подключение библиотек для анализа данных

```
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
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
```

Проверка данных

Выведем первые 5 строк датасета для проверки корректного импорта данных:

```
data.head()
```

	Date	HOUR	GENDER	PRODUCT	COLOR	SIZE	SALE CONSULTANT	QUANTITY	PRICE	VAT	TOTAL
0	2020-06-01	10:21:00	MEN	T SHIRT	2160	XS	Mary Taylor	1	36.94	2.96	39.9
1	2020-06-01	10:24:00	WOMEN	SHIRT LONG SLEEVE	2550	40	Kelli Cooley	1	73.98	5.92	79.9
2	2020-06-01	10:26:00	MEN	SHIRT LONG SLEEVE	900	45X	Bradley Saldana	1	73.98	5.92	79.9
3	2020-06-01	10:29:00	WOMEN	KNIT TROUSERS	600	XL	Kelli Cooley	1	46.20	3.70	49.9
4	2020-06-01	10:30:00	MEN	SHIRT LONG SLEEVE	1800	39X	Robert Moran	1	82.32	6.58	88.9

Видим, что данные загружены корректно. Разбиения по строкам и столбцам произведены верно. Проблем с кодировкой не возникло.

Узнаем размер датасета:

```
print(f'Количество записей: {data.shape[0]}\nКоличество параметров: {data.shape[1]}')
```

```
Количество записей: 9999
Количество параметров: 11
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9999 entries, 0 to 9998
Data columns (total 11 columns):
Date                9999 non-null datetime64[ns]
HOUR                9994 non-null object
GENDER              9996 non-null object
PRODUCT             9992 non-null object
COLOR               9999 non-null int64
SIZE                9997 non-null object
SALE CONSULTANT     9999 non-null object
QUANTITY            9999 non-null int64
PRICE               9999 non-null float64
VAT                 9999 non-null float64
TOTAL               9999 non-null float64
dtypes: datetime64[ns](1), float64(3), int64(2), object(5)
memory usage: 859.4+ KB
```

Видим, что в датасете присутствуют данные нескольких типов: вещественные (`float64`),строковые (`object`), дата (`datetime64`). Также узнаём, что в каждом столбце присутствует ровно `9999` значения, следовательно у нас отсутствуют пустые ячейки, что говорит об отсутствии явных пропусков данных в датасете.

Кодирование категорий товаров и времени суток

```
le = LabelEncoder()
data['TIME OF DAY'] = le.fit_transform(data['TIME OF DAY'])
```

```
data['GENDER'] = le.fit_transform(data[ 'GENDER'])
```

```
data['PRODUCT'] = le.fit_transform(data['PRODUCT'])
```

```
data['SALE CONSULTANT'] = le.fit_transform(data['SALE CONSULTANT'])
```

```
data['DAY_OF_WEEK'] = le.fit_transform(data['DAY_OF_WEEK'])
```

```
data['CATEGORY'] = le.fit_transform(data['CATEGORY'])
```

```
data['TIME OF DAY'].unique()
```

```
array([2, 1, 0, 3])
```

```
data.head()
```

	Date	GENDER	PRODUCT	SIZE	SALE CONSULTANT	PRICE	VAT	TOTAL	houres	minutes	TIME OF DAY	DAY_OF_WEEK	CATEGORY
0	2020-06-01	2	25	XS	4	36.94	2.96	39.9	10	21	2	1	7
1	2020-06-01	3	19	40	3	73.98	5.92	79.9	10	24	2	1	7
2	2020-06-01	2	19	45X	0	73.98	5.92	79.9	10	26	2	1	7
3	2020-06-01	3	11	XL	3	46.20	3.70	49.9	10	29	2	1	5
4	2020-06-01	2	19	39X	5	82.32	6.58	88.9	10	30	2	1	7

```
y = data['PRICE']
X = data.drop(['PRICE','Date', 'SIZE'], axis=1)
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=3)
```

```
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()
```

	GENDER	PRODUCT	SALE CONSULTANT	VAT	TOTAL	houres	minutes	TIME OF DAY	DAY_OF_WEEK	CATEGORY
count	6987.000000	6987.000000	6987.000000	6987.000000	6987.000000	6987.000000	6987.000000	6987.000000	6987.000000	6987.000000
mean	0.620390	0.575511	0.499785	0.196925	0.206193	0.460283	0.488988	0.232432	0.497233	0.621497
std	0.323928	0.254893	0.342376	0.099331	0.091569	0.286697	0.296556	0.241165	0.335474	0.124573
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.666667	0.354839	0.200000	0.143203	0.143011	0.230769	0.237288	0.000000	0.166667	0.500000
50%	0.666667	0.612903	0.400000	0.179003	0.178853	0.461538	0.491525	0.333333	0.500000	0.700000
75%	0.666667	0.806452	0.800000	0.250605	0.250538	0.692308	0.745763	0.333333	0.833333	0.700000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

```python
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()
```

```python
base_k = 7
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=7

R^2: 0.6495148830033426
MSE: 190.74089109468164
MAE: 10.183773908895779
```

Снимок экрана

```python
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: 0.6598115681289791
Лучший параметр: {'n_neighbors': 5}
Метрики на тестовом наборе
R^2: 0.656059306313344
MSE: 187.1792872681135
MAE: 10.096415358931552

Оптимизация метрики neg_mean_squared_error: -191.9164066410647
Лучший параметр: {'n_neighbors': 5}
Метрики на тестовом наборе
R^2: 0.656059306313344
MSE: 187.1792872681135
MAE: 10.096415358931552

Оптимизация метрики neg_mean_absolute_error: -10.153557265671484
Лучший параметр: {'n_neighbors': 5}
Метрики на тестовом наборе
R^2: 0.656059306313344
MSE: 187.1792872681135
MAE: 10.096415358931552
```

Снимок экрана

```
Результаты кросс-валидации при cv=10

Оптимизация метрики r2: 0.667016893421113
Лучший параметр: {'n_neighbors': 5}
Метрики на тестовом наборе
R^2: 0.656059306313344
MSE: 187.1792872681135
MAE: 10.096415358931552

Оптимизация метрики neg_mean_squared_error: -187.9485194232596
Лучший параметр: {'n_neighbors': 5}
Метрики на тестовом наборе
R^2: 0.656059306313344
MSE: 187.1792872681135
MAE: 10.096415358931552

Оптимизация метрики neg_mean_absolute_error: -10.01112899311747
Лучший параметр: {'n_neighbors': 5}
Метрики на тестовом наборе
R^2: 0.656059306313344
MSE: 187.1792872681135
MAE: 10.096415358931552
```

Сравнение начальной и лучшей моделей

```
best_k = 5
y_pred_best = KNeighborsRegressor(n_neighbors=best_k).fit(x_train, y_train).predict(x_test)
```

```
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: 0.6495148830033426
MSE: 190.74089109468164
MAE: 10.183773908895779
_____________________

Optimal model

R^2: 0.656059306313344
MSE: 187.1792872681135
MAE: 10.096415358931552
```