

**Московский государственный технический
университет им. Н.Э. Баумана**

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

Курс «Разработка интернет приложений»

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
Васильченко Дарья

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю.Е.

Москва, 2022 г.

Описание задания:

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.

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

Подключение библиотек

```
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')
```

Загрузка датасета

```
In [2]: data = pd.read_csv('cars_price_2.csv')
```

```
In [3]: data.head()
```

```
Out[3]:
```

	Unnamed: 0	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Mileage	Engine	Power	Seats	New_Price	Price
0	0	Maruti Wagon R LXI CNG	Mumbai	2010	72000	CNG	Manual	First	26.6 km/kg	998 CC	58.16 bhp	5.0	NaN	1.75
1	1	Hyundai Creta 1.6 CRDi SX Option	Pune	2015	41000	Diesel	Manual	First	19.67 kmpl	1582 CC	126.2 bhp	5.0	NaN	12.50
2	2	Honda Jazz V	Chennai	2011	46000	Petrol	Manual	First	18.2 kmpl	1199 CC	88.7 bhp	5.0	8.61 Lakh	4.50
3	3	Maruti Ertiga VDI	Chennai	2012	87000	Diesel	Manual	First	20.77 kmpl	1248 CC	88.76 bhp	7.0	NaN	6.00
4	4	Audi A4 New 2.0 TDI Multitronic	Coimbatore	2013	40670	Diesel	Automatic	Second	15.2 kmpl	1968 CC	140.8 bhp	5.0	NaN	17.74

```
In [4]: data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6019 entries, 0 to 6018
Data columns (total 14 columns):
#   Column                Non-Null Count  Dtype  
---  -
0   Unnamed: 0            6019 non-null  int64  
1   Name                  6019 non-null  object  
2   Location              6019 non-null  object  
3   Year                  6019 non-null  int64  
4   Kilometers_Driven     6019 non-null  int64  
5   Fuel_Type             6019 non-null  object  
6   Transmission          6019 non-null  object  
7   Owner_Type            6019 non-null  object  
8   Mileage               6017 non-null  object  
9   Engine                5983 non-null  object  
10  Power                 5983 non-null  object  
11  Seats                 5977 non-null  float64 
12  New_Price             824 non-null   object  
13  Price                 6019 non-null  float64 
dtypes: float64(2), int64(3), object(9)
memory usage: 658.5+ KB
```

```
In [5]: data = data.dropna(subset=['Mileage'], axis=0)
data = data.dropna(subset=['Engine'], axis=0)
data = data.dropna(subset=['Power'], axis=0)
data = data.dropna(subset=['Seats'], axis=0)
data.drop('New_Price', axis=1, inplace=True)
```

```
In [6]: for i in ['Mileage']:
data[i] = data[i].str.replace('km/kg', '')
data[i] = data[i].str.replace('kmpl', '').astype('float')
```

```
In [7]: for i in ['Engine']:
data[i] = data[i].str.replace('CC', '').astype('int')
```

```
In [8]: for i in ['Power']:
data[i] = data[i].str.replace('bhp', '')

data.drop(data[data['Power'] == 'null '].index, inplace=True)
data[data['Power'] == 'null ']

for i in ['Power']:
data[i] = data[i].astype('float')
```

Кодирование категориальных признаков

In [9]:

```
data.info()
```

```

class 'pandas.core.frame.DataFrame'>
Int64Index: 5872 entries, 0 to 6018
Data columns (total 13 columns):
#   Column                Non-Null Count  Dtype  
---  --
0   Unnamed: 0             5872 non-null   int64   
1   Name                   5872 non-null   object  
2   Location               5872 non-null   object  
3   Year                   5872 non-null   int64   
4   Kilometers_Driven      5872 non-null   int64   
5   Fuel_Type              5872 non-null   object  
6   Transmission           5872 non-null   object  
7   Owner_Type             5872 non-null   object  
8   Mileage                5872 non-null   float64  
9   Engine                 5872 non-null   int32    
10  Power                  5872 non-null   float64  
11  Seats                  5872 non-null   float64  
12  Price                  5872 non-null   float64  
dtypes: float64(4), int32(1), int64(3), object(5)
memory usage: 619.3+ KB

```

```
In [10]:
```

```
category_cols = ['Name', 'Location', 'Fuel_Type', 'Transmission', 'Owner_Type']
```

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

Количество уникальных значений

```
Name: 1811
Location: 11
Fuel_Type: 4
Transmission: 2
Owner_Type: 4
```

```
In [11]:
```

```
remove_cols = ['Name']
```

In [12]:

```
for col in remove_cols:
    category_cols.remove(col)
data = pd.get_dummies(data, columns=category_cols)
```

In [13]:

```
data.drop(remove_cols, axis=1, inplace=True)
data.drop(['Unnamed: 0'], axis=1, inplace=True)
data.describe()
```

Out[13]:

	Year	Kilometers_Driven	Mileage	Engine	Power	Seats	Price	Location_Ahmedabad	Location_Bangalore	Location_Chennai	...	Fu
count	5872.000000	5.872000e+03	5872.000000	5872.000000	5872.000000	5872.000000	5872.000000	5872.000000	5872.000000	5872.000000	...	
mean	2013.477691	5.831700e+04	18.277839	1625.745572	113.276894	5.283719	9.603919	0.037466	0.059094	0.081063	...	
std	3.164568	9.216941e+04	4.365657	601.641783	53.881892	0.805081	11.249453	0.189917	0.235821	0.272955	...	
min	1998.000000	1.710000e+02	0.000000	624.000000	34.200000	2.000000	0.440000	0.000000	0.000000	0.000000	...	
25%	2012.000000	3.342250e+04	15.260000	1198.000000	75.000000	5.000000	3.517500	0.000000	0.000000	0.000000	...	
50%	2014.000000	5.260900e+04	18.200000	1495.500000	97.700000	5.000000	5.750000	0.000000	0.000000	0.000000	...	
75%	2016.000000	7.240275e+04	21.100000	1991.000000	138.100000	5.000000	10.000000	0.000000	0.000000	0.000000	...	
max	2019.000000	6.500000e+06	33.540000	5998.000000	560.000000	10.000000	160.000000	1.000000	1.000000	1.000000	...	

8 rows x 28 columns

◀ ▶

Разделение выборки на обучающую и тестовую

In [14]:

```
y = data['Price']
X = data.drop('Price', axis=1)
x train, x test, y train, y test = train test split(X, y, test size=0.3, random state=3)
```

Масштабирование данных

In [15]:

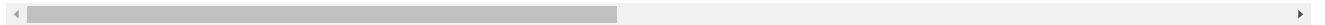
```
scaler = MinMaxScaler().fit(x_train)
x_train = pd.DataFrame(scaler.transform(x_train), columns=x_train.columns)
x_test = pd.DataFrame(scaler.transform(x_test), columns=x_train.columns)
x_train.describe()
```

Out[15]:

[illegible]

	Year	Kilometers_Driven	Mileage	Engine	Power	Seats	Location_Ahmedabad	Location_Bangalore	Location_Chennai	Location_Coimbatore
25%	0.666667	0.005163	0.455874	0.118669	0.079412	0.375000	0.000000	0.000000	0.000000	0.000000
50%	0.761905	0.008007	0.546064	0.180484	0.122461	0.375000	0.000000	0.000000	0.000000	0.000000
75%	0.857143	0.011056	0.629100	0.282613	0.199420	0.375000	0.000000	0.000000	0.000000	0.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

8 rows × 27 columns



Обучение KNN с произвольным k

```
In [16]: 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 [17]: base_k = 6
          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=6

R^2: 0.7618660028183054
MSE: 27.80178102377349
MAE: 2.6051239122209613

Кросс-валидация

```
In [18]: 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: 0.7159716565452815
Лучший параметр: {'n_neighbors': 4}
Метрики на тестовом наборе
R^2: 0.7526887484311768
MSE: 28.873211478433596
MAE: 2.5486975028376846

Оптимизация метрики neg_mean_squared_error: -38.431966855231146
Лучший параметр: {'n_neighbors': 4}
Метрики на тестовом наборе
R^2: 0.7526887484311768
MSE: 28.873211478433596
MAE: 2.5486975028376846

Оптимизация метрики neg_mean_absolute_error: -2.6511885644768856
Лучший параметр: {'n_neighbors': 2}
Метрики на тестовом наборе
R^2: 0.7708511506426075
MSE: 26.75277871736663
MAE: 2.4509818388195233

Результаты кросс-валидации при cv=10

Оптимизация метрики r2: 0.7272574724579125
Лучший параметр: {'n_neighbors': 4}
Метрики на тестовом наборе
R^2: 0.7526887484311768
MSE: 28.873211478433596
MAE: 2.5486975028376846

Оптимизация метрики neg_mean_squared_error: -37.463731710766424
Лучший параметр: {'n_neighbors': 4}
Метрики на тестовом наборе
R^2: 0.7526887484311768
MSE: 28.873211478433596
MAE: 2.5486975028376846

Оптимизация метрики neg_mean_absolute_error: -2.5966374695863745
Лучший параметр: {'n_neighbors': 2}
Метрики на тестовом наборе
R^2: 0.7708511506426075
MSE: 26.75277871736663

MAE: 2.4509818388195233

```
In [19]: metrics = ['r2', 'neg_mean_squared_error', 'neg_mean_absolute_error']

print('Результаты кросс-валидации\n')
for metric in metrics:
    params = {'n_neighbors': range(1, 30)}
    knn_cv = GridSearchCV(KNeighborsRegressor(), params, cv=5, scoring=metric, n_jobs=-1)
    knn_cv.fit(x_train, y_train)
    print_cv_result(knn_cv, x_test, y_test)
```

Результаты кросс-валидации

Оптимизация метрики r2: 0.7159716565452815
Лучший параметр: {'n_neighbors': 4}
Метрики на тестовом наборе
R^2: 0.7526887484311768
MSE: 28.873211478433596
MAE: 2.5486975028376846

Оптимизация метрики neg_mean_squared_error: -38.431966855231146
Лучший параметр: {'n_neighbors': 4}
Метрики на тестовом наборе
R^2: 0.7526887484311768
MSE: 28.873211478433596
MAE: 2.5486975028376846

Оптимизация метрики neg_mean_absolute_error: -2.6511885644768856
Лучший параметр: {'n_neighbors': 2}
Метрики на тестовом наборе
R^2: 0.7708511506426075
MSE: 26.75277871736663
MAE: 2.4509818388195233

```
In [20]: best_k = 2
y_pred_best = KNeighborsRegressor(n_neighbors=best_k).fit(x_train, y_train).predict(x_test)
```

Сравнение исходной и оптимальной моделей

```
In [21]: 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: 0.7618660028183054
MSE: 27.80178102377349
MAE: 2.6051239122209613

Optimal model

R^2: 0.7708511506426075
MSE: 26.75277871736663
MAE: 2.4509818388195233