Лабораторная работа №3

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

Задание:

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

Импорт библиотек:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, GridSearchCV, RandomizedSearchCV
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import MinMaxScaler, StandardScaler
from matplotlib import pyplot as plt
import seaborn as sns
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from warnings import simplefilter

simplefilter('ignore')
```

Монтирование Google Drive для получения доступа к данным, лежащим на нем:

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

Загрузка данных:

```
filename = '/content/drive/MyDrive/ford.csv'
```

```
df = pd.read_csv(filename, sep=',')
```

```
df.head()
```

	model	year	price	transmission	mileage	fuelType	tax	mpg	engineSize
0	Fiesta	2017	12000	Automatic	15944	Petrol	150	57.7	1.0
1	Focus	2018	14000	Manual	9083	Petrol	150	57.7	1.0
2	Focus	2017	13000	Manual	12456	Petrol	150	57.7	1.0
3	Fiesta	2019	17500	Manual	10460	Petrol	145	40.3	1.5
4	Fiesta	2019	16500	Automatic	1482	Petrol	145	48.7	1.0

```
df.columns
```

```
Index(['model', 'year', 'price', 'transmission', 'mileage', 'fuelType', 'tax',
       'mpg', 'engineSize'],
      dtype='object')
```

```
df.dtypes
```

```
model            object
year              int64
price             int64
transmission     object
mileage           int64
fuelType         object
tax               int64
mpg             float64
engineSize      float64
dtype: object
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17966 entries, 0 to 17965
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   model         17966 non-null  object 
 1   year          17966 non-null  int64  
 2   price         17966 non-null  int64  
 3   transmission  17966 non-null  object 
 4   mileage       17966 non-null  int64  
 5   fuelType      17966 non-null  object 
 6   tax           17966 non-null  int64  
 7   mpg           17966 non-null  float64
 8   engineSize    17966 non-null  float64
dtypes: float64(2), int64(4), object(3)
memory usage: 1.2+ MB
```

Кодирование категориальных признаков

```
category_cols = ['model', 'transmission', 'fuelType']

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

```
 Количество уникальных значений

 model: 24
 transmission: 3
 fuelType: 5
```

```
df1 = pd.concat([df,
                     pd.get_dummies(df['model'], prefix="model"),
                     pd.get_dummies(df['transmission'], prefix="transmission"),
                     pd.get_dummies(df['fuelType'], prefix="fuelType"),],
                    axis=1)
```

```
df1.drop(['model', 'transmission', 'fuelType'], axis=1, inplace=True)
```

```
df1.head()
```

	year	price	mileage	tax	mpg	engineSize	model_ B-MAX	model_ C-MAX	model_ EcoSport	model_ Edge	...
0	2017	12000	15944	150	57.7	1.0	0	0	0	0	...
1	2018	14000	9083	150	57.7	1.0	0	0	0	0	...
2	2017	13000	12456	150	57.7	1.0	0	0	0	0	...
3	2019	17500	10460	145	40.3	1.5	0	0	0	0	...
4	2019	16500	1482	145	48.7	1.0	0	0	0	0	...

5 rows × 38 columns

Разделение выборки на обучающую и тестовую

```
from sklearn.model_selection import train_test_split
data_X = df1.drop(columns=['mileage'])
data_y = df1['mileage']
data_X_train, data_X_test, data_Y_train, data_Y_test = train_test_split(data_X, data_y, ra
```

Масштабирование данных

```
y = df1['mileage']
X = df1.drop('mileage', axis=1)
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=3)
```

Обучение модели ближайших соседей для произвольно заданного гиперпараметра K

```
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()


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: 0.5048125315929815
MSE: 178463253.4268965
MAE: 9468.331261595547
```

Кросс-валидация

```
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
        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: 0.7168821367909357
Лучший параметр: {'n_neighbors': 7}
Метрики на тестовом наборе
R^2: 0.7150171364568363
MSE: 102706494.49679299
MAE: 6963.915027829314
```

```
Оптимизация метрики neg_mean_squared_error: -109467588.64528474
Лучший параметр: {'n_neighbors': 7}
Метрики на тестовом наборе
R^2: 0.7150171364568363
MSE: 102706494.49679299
MAE: 6963.915027829314

Оптимизация метрики neg_mean_absolute_error: -7185.374764235642
Лучший параметр: {'n_neighbors': 9}
Метрики на тестовом наборе
R^2: 0.7146031649384026
MSE: 102855687.90074442
MAE: 6936.52211915069

Результаты кросс-валидации при cv=10

Оптимизация метрики r2: 0.7206714671726908
Лучший параметр: {'n_neighbors': 6}
Метрики на тестовом наборе
R^2: 0.7129141376879478
MSE: 103464405.44202742
MAE: 6996.241713048855

Оптимизация метрики neg_mean_squared_error: -107667449.08580966
Лучший параметр: {'n_neighbors': 6}
Метрики на тестовом наборе
R^2: 0.7129141376879478
MSE: 103464405.44202742
MAE: 6996.241713048855

Оптимизация метрики neg_mean_absolute_error: -7136.150105680053
Лучший параметр: {'n_neighbors': 8}
Метрики на тестовом наборе
R^2: 0.7153795443492046
MSE: 102575884.38311689
MAE: 6935.565027829313
```

```
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: 0.5115839193958043
Лучший параметр: {'n_neighbors': 7}
Метрики на тестовом наборе
R^2: 0.5052320807403434
MSE: 178312049.8715005
MAE: 9470.486138351444

Оптимизация метрики neg_mean_squared_error: -189060229.5930272
```

```
Лучший параметр: {'n_neighbors': 7}
Метрики на тестовом наборе
R^2: 0.5052320807403434
MSE: 178312049.8715005
MAE: 9470.486138351444

Оптимизация метрики neg_mean_absolute_error: -9654.671580790071
Лучший параметр: {'n_neighbors': 7}
Метрики на тестовом наборе
R^2: 0.5052320807403434
MSE: 178312049.8715005
MAE: 9470.486138351444
```

```
best_k = 2
y_pred_best = KNeighborsRegressor(n_neighbors=best_k).fit(x_train, y_train).predict(x_test
```

Сравнение метрики качества исходной и оптимальной моделей

```
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: 0.5048125315929815
MSE: 178463253.4268965
MAE: 9468.331261595547
_________________________

Optimal model

R^2: 0.45886222198623483
MSE: 195023530.6382189
MAE: 9768.751948051948
```