Федеральное государственное бюджетное образовательное учреждение высшего образования «Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)

Факультет «Информатика и вычислительная техника»
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

Веревкина Диана В.

Подпись и дата:

Проверил:
преподаватель каф. ИУ5

Гапанюк Ю.Е.

Подпись и дата:

Москва, 2022 г.

Цель лабораторной работы

Изучение способов подготовки выборки и подбора гиперпараметров на примере метода ближайших соседей.

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.

Текст программы и результаты ее выполнения

Импорт необходимых библиотек и датасета

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set(style="ticks")

df = pd.read_csv('penguins_size.csv', sep=",")
df
```

	species	island	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_g	sex
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	MALE
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	FEMALE
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	FEMALE
3	Adelie	Torgersen	NaN	NaN	NaN	NaN	NaN
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0	FEMALE
...	...	...	...	...	...	...	...
339	Gentoo	Biscoe	NaN	NaN	NaN	NaN	NaN
340	Gentoo	Biscoe	46.8	14.3	215.0	4850.0	FEMALE
341	Gentoo	Biscoe	50.4	15.7	222.0	5750.0	MALE
342	Gentoo	Biscoe	45.2	14.8	212.0	5200.0	FEMALE
343	Gentoo	Biscoe	49.9	16.1	213.0	5400.0	MALE

Стандартные характеристики датасета

```
# размер набора данных
df.shape
```

```
(344, 7)
```

```
# типы колонок
df.dtypes
```

```
species              object
island               object
culmen_length_mm    float64
culmen_depth_mm     float64
flipper_length_mm   float64
body_mass_g         float64
sex                  object
dtype: object
```

Очистка строк с нулевыми значениями

```
# проверим есть ли пропущенные значения
df.isnull().sum()
```

```
species              0
island               0
culmen_length_mm     2
culmen_depth_mm      2
flipper_length_mm    2
body_mass_g          2
sex                 10
dtype: int64
```

```
# Удаление строк, содержащих пустые значения
df = df.dropna(axis=0, how='any')
(df.shape, df_new.shape)
df
```

	species	island	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_g	sex
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	MALE
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	FEMALE
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	FEMALE
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0	FEMALE
5	Adelie	Torgersen	39.3	20.6	190.0	3650.0	MALE
...	...	...	...	...	...	...	...
338	Gentoo	Biscoe	47.2	13.7	214.0	4925.0	FEMALE
340	Gentoo	Biscoe	46.8	14.3	215.0	4850.0	FEMALE
341	Gentoo	Biscoe	50.4	15.7	222.0	5750.0	MALE
342	Gentoo	Biscoe	45.2	14.8	212.0	5200.0	FEMALE
343	Gentoo	Biscoe	49.9	16.1	213.0	5400.0	MALE

334 rows × 7 columns

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 334 entries, 0 to 343
Data columns (total 7 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   species            334 non-null    object 
 1   island             334 non-null    object 
 2   culmen_length_mm   334 non-null    float64
 3   culmen_depth_mm    334 non-null    float64
 4   flipper_length_mm  334 non-null    float64
 5   body_mass_g        334 non-null    float64
 6   sex                334 non-null    object 
dtypes: float64(4), object(3)
memory usage: 20.9+ KB
```

Кодирование категориальных значений

```
#Кодирование категориальных признаков
df["species"].value_counts()
df["species"] = df["species"].astype('category')

df["island"] = df["island"].astype('category')
df["sex"] = df["sex"].astype('category')

#назначить закодированную переменную новому столбцу с помощью метода доступа (accessor) cat.c
df["species_cat"] = df["species"].cat.codes
df["island_cat"] = df["island"].cat.codes
df["sex_cat"] = df["sex"].cat.codes
df

df_cat = df.drop(['species', 'island', 'sex'], axis=1, inplace=True)
df
```

	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_g	species_cat	island_cat	sex_cat
0	39.1	18.7	181.0	3750.0	0	2	2
1	39.5	17.4	186.0	3800.0	0	2	1
2	40.3	18.0	195.0	3250.0	0	2	1
4	36.7	19.3	193.0	3450.0	0	2	1
5	39.3	20.6	190.0	3650.0	0	2	2

Разделение выборки на обучающую и тестовую

```
#Разделение выборки на обучающую и тестовую
y = df['body_mass_g']
X = df.drop('body_mass_g', axis=1)
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=3)
x_train
```

	culmen_length_mm	culmen_depth_mm	flipper_length_mm	species_cat	island_cat	sex_cat
228	43.3	13.4	209.0	2	0	1
48	36.0	17.9	190.0	0	1	1
5	39.3	20.6	190.0	0	2	2
45	39.6	18.8	190.0	0	1	2
174	43.2	16.6	187.0	1	1	1
...	...	...	...	...	...	...
284	45.8	14.2	219.0	2	0	1

```
y_train
```

```
228    4400.0
48     3450.0
5      3650.0
45     4600.0
174    2900.0
       ...
284    4700.0
```

Масштабирование данных

```
#Масштабирование данных
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()
```

	culmen_length_mm	culmen_depth_mm	flipper_length_mm	species_cat	island_cat	sex_cat
count	233.000000	233.000000	233.000000	233.000000	233.000000	233.000000
mean	0.409151	0.494781	0.497262	0.463519	0.324034	0.736052
std	0.203907	0.238422	0.242362	0.444991	0.352243	0.254419
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.245283	0.320988	0.310345	0.000000	0.000000	0.500000
50%	0.426415	0.518519	0.431034	0.500000	0.500000	0.500000
75%	0.581132	0.679012	0.724138	1.000000	0.500000	1.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

Обучение KNN с произвольным k

```
#Обучение KNN с произвольным k
def print_metrics(y_test, y_pred): #метрики оценивания моделей
    print(f"R^2: {r2_score(y_test, y_pred)}") #Чем ближе R^2 к 1, тем лучше обобщающая способность модели
    print(f"MSE: {mean_squared_error(y_test, y_pred)}") #Средняя Квадратическая Ошибка
    print(f"MAE: {mean_absolute_error(y_test, y_pred)}") #мера схожести предсказаний и правильных значений для каких-либо наблюдений

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 = 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: 0.8745411551160632
MSE: 81616.61446756922
MAE: 224.5049504950495
```

Кросс-валидация

```
#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=-1)
        knn_cv.fit(x_train, y_train)
        print_cv_result(knn_cv, x_test, y_test)
```

```
Результаты кросс-валидации при cv=6

Оптимизация метрики r2: 0.8415321923023411
Лучший параметр: {'n_neighbors': 23}
Метрики на тестовом наборе
R^2: 0.8543978501831682
MSE: 94720.73920530049
MAE: 244.28540680154978

Оптимизация метрики neg_mean_squared_error: -94788.0260457989
Лучший параметр: {'n_neighbors': 23}
Метрики на тестовом наборе
R^2: 0.8543978501831682
MSE: 94720.73920530049
MAE: 244.28540680154978

Оптимизация метрики neg_mean_absolute_error: -242.40475072854937
Лучший параметр: {'n_neighbors': 23}
Метрики на тестовом наборе
R^2: 0.8543978501831682
MSE: 94720.73920530049
MAE: 244.28540680154978
```

```
Результаты кросс-валидации при cv=9

Оптимизация метрики r2: 0.8399379208126817
Лучший параметр: {'n_neighbors': 9}
Метрики на тестовом наборе
R^2: 0.8695487532324968
MSE: 84864.39616183842
MAE: 230.50055005500556

Оптимизация метрики neg_mean_squared_error: -93159.6193415638
Лучший параметр: {'n_neighbors': 9}
Метрики на тестовом наборе
R^2: 0.8695487532324968
MSE: 84864.39616183842
MAE: 230.50055005500556

Оптимизация метрики neg_mean_absolute_error: -243.32431149097818
Лучший параметр: {'n_neighbors': 9}
Метрики на тестовом наборе
R^2: 0.8695487532324968
MSE: 84864.39616183842
MAE: 230.50055005500556
```

```
best_k = 9
y_pred_best = KNeighborsRegressor(n_neighbors=best_k).fit(x_train, y_train).predict(x_test)
```

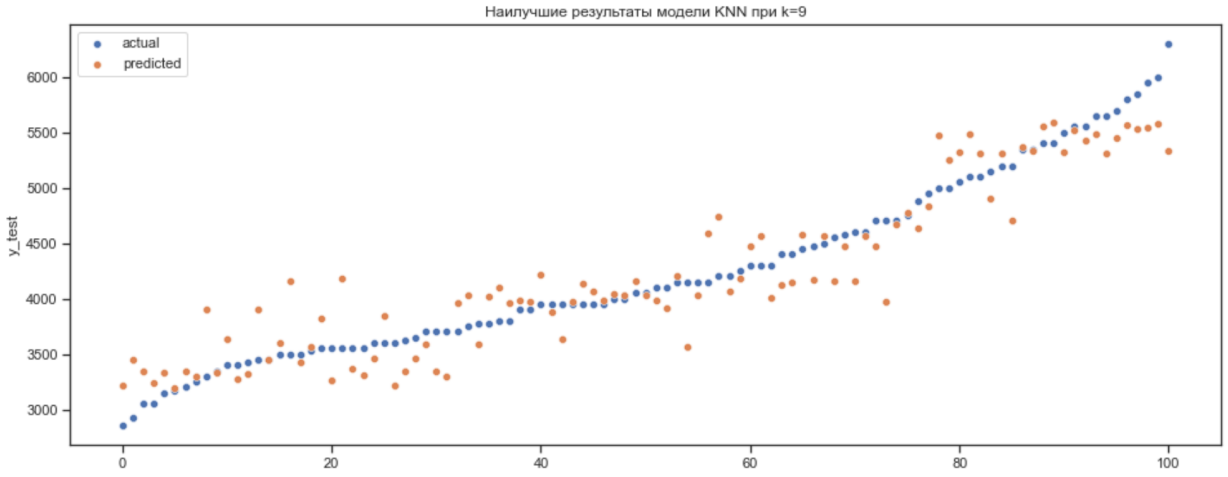
Сравнение исходной и оптимальной моделей

```
#Сравнение исходной и оптимальной моделей
print('Исходная модель\n')
print_metrics(y_test, y_pred_base)
print('\nОптимальная модель\n')
print_metrics(y_test, y_pred_best)
```

```
Исходная модель

R^2: 0.8745411551160632
MSE: 81616.61446756922
MAE: 224.5049504950495

Оптимальная модель

R^2: 0.8695487532324968
MSE: 84864.39616183842
MAE: 230.50055005500556
```

Визуализация

```
#Визуализация
plt.figure(figsize=(16, 6))
sns.scatterplot(range(res.shape[0]), res['y_test'], label='actual')
sns.scatterplot(range(res.shape[0]), res['y_pred_best'], label='predicted')
plt.title(f'Наилучшие результаты модели KNN при k={best_k}')
plt.show()
```



Вывод

В ходе выполнения данной лабораторной работы я повторила язык программирования Python и работу с юпитер тетрадками. Также изучила подготовки выборки и подбора гиперпараметров на примере метода ближайших соседей.