Защищено: Гапанюк Ю.Е.

" " 2023 г.

Демонстрация: Гапанюк Ю.Е.

" " 2023 г.

**Отчет по лабораторной работе № 3 по курсу Технологии машинного обучения**

**ГУИМЦ**

**Тема работы: " Подготовка обучающей и тестовой выборки, кросс-валидация и подбор гиперпараметров на примере метода ближайших соседей."**

8

(количество листов) Вариант № **2**

ИСПОЛНИТЕЛЬ:

студент группы ИУ5Ц-84Б

(подпись)

Распашнов А.А.

" " 2023 г.

Москва, МГТУ - 2023

# **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.

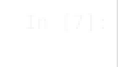
# **Ход выполнения работы**

## **Текстовое описание набора данных**

В качестве набора данных используется dataset рейтингов университетов мира на основании трёх рейтингов. Датасет доступен по адресу: https://www.kaggle.com/c/titanic/data?select=train.csv

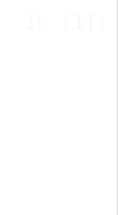
Из набора данных будет рассматриваться только файл train.csv

# ЛР3



891 rows × 5 columns

data



**import** numpy **as** np **import** pandas **as** pd **import** seaborn **as** sns

**import** matplotlib.pyplot **as** plt

**%matplotlib** inline

**from** sklearn.impute **import** SimpleImputer

**from** sklearn.model\_selection **import** train\_test\_split



**import** numpy **as** np

**import** pandas **as** pd

**from** typing **import** Dict, Tuple

**from** scipy **import** stats

**from** sklearn.datasets **import** load\_iris, load\_boston

**from** sklearn.model\_selection **import** train\_test\_split

**from** sklearn.neighbors **import** KNeighborsRegressor, KNeighborsClassifier **from** sklearn.metrics **import** accuracy\_score, balanced\_accuracy\_score **from** sklearn.metrics **import** plot\_confusion\_matrix

**from** sklearn.metrics **import** precision\_score, recall\_score, f1\_score, classification\_report

**from** sklearn.metrics **import** confusion\_matrix

**from** sklearn.metrics **import** mean\_absolute\_error, mean\_squared\_error, mean\_squared\_log\_error, median\_absolute\_err

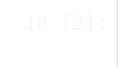
**from** sklearn.metrics **import** roc\_curve, roc\_auc\_score

**import** seaborn **as** sns

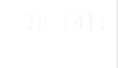
**from** sklearn.model\_selection **import** learning\_curve

**import** matplotlib.pyplot **as** plt

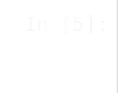
**%matplotlib** inline sns**.**set(style**=**"ticks")



**from** sklearn.model\_selection **import** KFold, RepeatedKFold, LeaveOneOut, LeavePOut, ShuffleSplit, StratifiedKFold

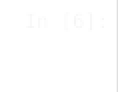


**from** sklearn.model\_selection **import** train\_test\_split



*# чтение обучающей выборки*

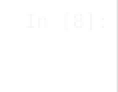
data **=** pd**.**read\_csv('train.csv')



*# уберем непонятный для нас параметр, чтобы он не помешал в будущем*

data**.**drop(['Name','Sex','Ticket','Embarked','Cabin','PassengerId', 'Parch'], axis **=** 1, inplace **= True**)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Survived** | **Pclass** | **Age** | **SibSp** | **Fare** |
| **0** | 0 | 3 | 22.0 | 1 | 7.2500 |
| **1** | 1 | 1 | 38.0 | 1 | 71.2833 |
| **2** | 1 | 3 | 26.0 | 0 | 7.9250 |
| **3** | 1 | 1 | 35.0 | 1 | 53.1000 |
| **4** | 0 | 3 | 35.0 | 0 | 8.0500 |
| **...** | ... | ... | ... | ... | ... |
| **886** | 0 | 2 | 27.0 | 0 | 13.0000 |
| **887** | 1 | 1 | 19.0 | 0 | 30.0000 |
| **888** | 0 | 3 | NaN | 1 | 23.4500 |
| **889** | 1 | 1 | 26.0 | 0 | 30.0000 |
| **890** | 0 | 3 | 32.0 | 0 | 7.7500 |



data **=** data**.**fillna(1) data**.**head()

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Survived** | **Pclass** | **Age** | **SibSp** | **Fare** |
| **0** 0 | 3 | 22.0 | 1 | 7.2500 |
| **1** 1 | 1 | 38.0 | 1 | 71.2833 |
| **2** 1 | 3 | 26.0 | 0 | 7.9250 |

|  |  |  |
| --- | --- | --- |
| **3** | 1 | 1 35.0 1 53.1000 |
| **4** | 0 | 3 35.0 0 8.0500 |



parts **=** np**.**split(data, [4,5], axis**=**1) X **=** parts[0]

Y **=** parts[1]

print('Входные данные:\n\n', X**.**head(), '\n\nВыходные данные:\n\n', Y**.**head())

Входные данные:

Survived Pclass Age SibSp

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 0.0 | 3.0 22.0 | 1.0 |
| 1 | 1.0 | 1.0 38.0 | 1.0 |
| 2 | 1.0 | 3.0 26.0 | 0.0 |
| 3 | 1.0 | 1.0 35.0 | 1.0 |
| 4 | 0.0 | 3.0 35.0 | 0.0 |

Выходные данные: Fare

0 7.2500

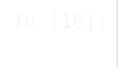
1 71.2833

2 7.9250

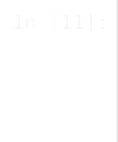
3 53.1000

4 8.0500

# Разделение выборки



X\_train, X\_test, Y\_train, Y\_test **=** train\_test\_split(X, Y, test\_size**=**0.03)



print('Входные параметры обучающей выборки:\n\n',X\_train**.**head(), \ '\n\nВходные параметры тестовой выборки:\n\n', X\_test**.**head(), \ '\n\nВыходные параметры обучающей выборки:\n\n', Y\_train**.**head(), \ '\n\nВыходные параметры тестовой выборки:\n\n', Y\_test**.**head())

Входные параметры обучающей выборки:

Survived Pclass Age SibSp 861 0.0 2.0 21.0 1.0

557 0.0 1.0 1.0 0.0

356 1.0 1.0 22.0 0.0

178 0.0 2.0 30.0 0.0

72 0.0 2.0 21.0 0.0

Входные параметры тестовой выборки:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Survived | Pclass | Age | SibSp |
| 829 | 1.0 | 1.0 | 62.0 | 0.0 |
| 451 | 0.0 | 3.0 | 1.0 | 1.0 |
| 454 | 0.0 | 3.0 | 1.0 | 0.0 |
| 781 | 1.0 | 1.0 | 17.0 | 1.0 |
| 571 | 1.0 | 1.0 | 53.0 | 2.0 |

Выходные параметры обучающей выборки: Fare

861 11.500

557 227.525

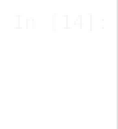
356 55.000

178 13.000

72 73.500

Выходные параметры тестовой выборки: Fare

|  |  |
| --- | --- |
| 829 | 80.0000 |
| 451 | 19.9667 |
| 454 | 8.0500 |
| 781 | 57.0000 |
| 571 | 51.4792 |



*# Проверим правильность разделения выборки на тестовую и обучающую. Посмотрим на размеры матриц.*

print(X\_train**.**shape) print(X\_test**.**shape) print(Y\_train**.**shape)

print(Y\_test**.**shape)

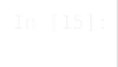
(864, 4)

(27, 4)

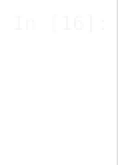
(864, 1)

(27, 1)

# Модель ближайших соседей для произвольного гиперпараметра K

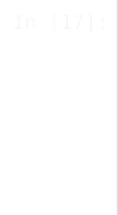


**from** sklearn.neighbors **import** KNeighborsRegressor



*# Решение задачи регрессии методом 2, 5 и 10 ближайших соседей* Regressor\_2NN **=** KNeighborsRegressor(n\_neighbors **=** 2) Regressor\_5NN **=** KNeighborsRegressor(n\_neighbors **=** 5) Regressor\_10NN **=** KNeighborsRegressor(n\_neighbors **=** 10) print('Пример модели:\n\n', Regressor\_10NN)

Пример модели: KNeighborsRegressor(n\_neighbors=10)



Regressor\_2NN**.**fit(X\_train, Y\_train) Regressor\_5NN**.**fit(X\_train, Y\_train) Regressor\_10NN**.**fit(X\_train, Y\_train) target\_2NN **=** Regressor\_2NN**.**predict(X\_test) target\_5NN **=** Regressor\_5NN**.**predict(X\_test) target\_10NN **=** Regressor\_10NN**.**predict(X\_test)

print('Пример предсказанных значений:\n\n', target\_10NN[:5], '\n ...')

Пример предсказанных значений:

[[14.73625]

[ 8.02 ]

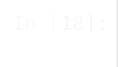
[15.27916]

[25.01041]

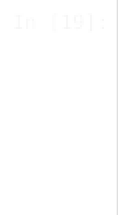
[ 9.80999]]

...

# Оценка качества регрессии (Метрики качества)



**from** sklearn.metrics **import** mean\_absolute\_error, mean\_squared\_error, median\_absolute\_error, r2\_score , accuracy



*# Оценка средней абсолютной ошибки*

print('Средняя абсолютная ошибка для 2 ближайших соседей:',mean\_absolute\_error(Y\_test, target\_2NN))

print('Средняя абсолютная ошибка для 5 ближайших соседей:',mean\_absolute\_error(Y\_test, target\_5NN))

print('Средняя абсолютная ошибка для 10 ближайших соседей:',mean\_absolute\_error(Y\_test, target\_10NN))

Средняя абсолютная ошибка для 2 ближайших соседей: 22.111566666666672 Средняя абсолютная ошибка для 5 ближайших соседей: 22.83663333333333 Средняя абсолютная ошибка для 10 ближайших соседей: 23.232667407407412



*# Оценка средней квадратичной ошибки*

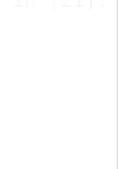
print('Средняя квадратичная ошибка для 2 ближайших соседей:',mean\_squared\_error(Y\_test, target\_2NN))

print('Средняя квадратичная ошибка для 5 ближайших соседей:',mean\_squared\_error(Y\_test, target\_5NN))

print('Средняя квадратичная ошибка для 10 ближайших соседей:',mean\_squared\_error(Y\_test

, target\_10NN))

Средняя квадратичная ошибка для 2 ближайших соседей: 1662.2206168757414 Средняя квадратичная ошибка для 5 ближайших соседей: 2025.9715158590964 Средняя квадратичная ошибка для 10 ближайших соседей: 1991.3278515739114

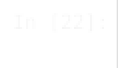


*# Оценка коэффициента детерминации*

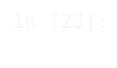
print('Коэффициент детерминации для 2 ближайших соседей:',r2\_score(Y\_test, target\_2NN)) print('Коэффициент детерминации для 5 ближайших соседей:',r2\_score(Y\_test, target\_5NN)) print('Коэффициент детерминации для 10 ближайших соседей:',r2\_score(Y\_test, target\_10NN

))

Коэффициент детерминации для 2 ближайших соседей: 0.5017857093074095 Коэффициент детерминации для 5 ближайших соседей: 0.39275932960432836 Коэффициент детерминации для 10 ближайших соседей: 0.4031430105992597



*## Grid Search (решетчатый поиск)*



**from** sklearn.model\_selection **import** GridSearchCV, RandomizedSearchCV

# Подбор гиперпараметров

GridSearch через среднюю квадратичную ошибку

Рассмотрим все количества ближайших соседей от 1 до 100, чтобы найти лучший результат. Возьмем 10 фолдов.



GridSearchCV(cv=10, estimator=KNeighborsRegressor(),

param\_grid=[{'n\_neighbors': array([ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,

92, 93, 94, 95, 96, 97, 98, 99, 100])}],

scoring='neg\_mean\_squared\_error')

**from** sklearn.model\_selection **import** GridSearchCV n\_range **=** np**.**array(range(1, 101, 1)) tuned\_parameters **=** [{'n\_neighbors': n\_range}]

gs **=** GridSearchCV(KNeighborsRegressor(), tuned\_parameters, cv**=**10, scoring**=**'neg\_mean\_squared\_error') gs**.**fit(X\_train, Y\_train)

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 14, | 15, | 16, | 17, | 18, | 19, | 20, | 21, | 22, | 23, | 24, | 25, | 26, |
| 27, | 28, | 29, | 30, | 31, | 32, | 33, | 34, | 35, | 36, | 37, | 38, | 39, |
| 40, | 41, | 42, | 43, | 44, | 45, | 46, | 47, | 48, | 49, | 50, | 51, | 52, |
| 53, | 54, | 55, | 56, | 57, | 58, | 59, | 60, | 61, | 62, | 63, | 64, | 65, |
| 66, | 67, | 68, | 69, | 70, | 71, | 72, | 73, | 74, | 75, | 76, | 77, | 78, |
| 79, | 80, | 81, | 82, | 83, | 84, | 85, | 86, | 87, | 88, | 89, | 90, | 91, |



print('Лучшая модель:\n\n', gs**.**best\_estimator\_) print('\nЛучшее число ближайших соседей:\n\n',gs**.**best\_params\_)

print('\nЛучшее значение средней квадратичной ошибки:\n\n',gs**.**best\_score\_)

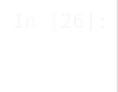
Лучшая модель: KNeighborsRegressor(n\_neighbors=11)

Лучшее число ближайших соседей:

{'n\_neighbors': 11}

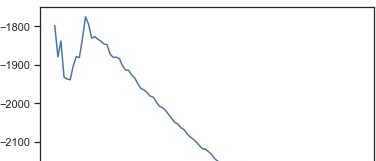
Лучшее значение средней квадратичной ошибки:

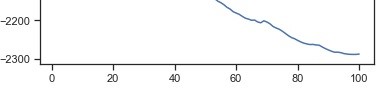
-1774.7300745209031



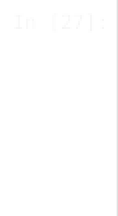
print('Изменение качества тестовой выборки в зависимости от кол-ва соседей:\n') plt**.**plot(n\_range, gs**.**cv\_results\_['mean\_test\_score'])

Изменение качества тестовой выборки в зависимости от кол-ва соседей: [<matplotlib.lines.Line2D at 0x7fb97d5d9610>]





# GridSearch через коэффициент детерминации



gs\_det **=** GridSearchCV(KNeighborsRegressor(), tuned\_parameters, cv**=**10, scoring**=**'r2') gs\_det**.**fit(X\_train, Y\_train)

print('Лучшая модель:\n\n', gs\_det**.**best\_estimator\_) print('\nЛучшее число ближайших соседей:\n\n',gs\_det**.**best\_params\_)

print('\nЛучшее значение коэффициента детерминации:\n\n',gs\_det**.**best\_score\_) print('\nИзменение качества тестовой выборки в зависимости от кол-ва соседей:\n') plt**.**plot(n\_range, gs\_det**.**cv\_results\_['mean\_test\_score'])

Лучшая модель: KNeighborsRegressor(n\_neighbors=18)

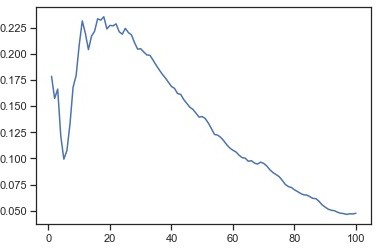
Лучшее число ближайших соседей:

{'n\_neighbors': 18}

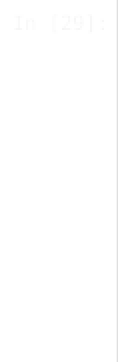
Лучшее значение коэффициента детерминации: 0.23532902602360656

Изменение качества тестовой выборки в зависимости от кол-ва соседей:

[<matplotlib.lines.Line2D at 0x7fb977ec3040>]



# Кросс-валидация



**from** sklearn.model\_selection **import** cross\_val\_score

scores\_2NN **=** cross\_val\_score(KNeighborsRegressor(n\_neighbors **=** 2), X, Y, cv**=**5, scoring**=** 'r2') scores\_5NN **=** cross\_val\_score(KNeighborsRegressor(n\_neighbors **=** 5), X, Y, cv**=**5, scoring**=** 'r2') scores\_10NN **=** cross\_val\_score(KNeighborsRegressor(n\_neighbors **=** 10), X, Y, cv**=**5, scoring **=** 'r2') scores\_50NN **=** cross\_val\_score(KNeighborsRegressor(n\_neighbors **=** 50), X, Y, cv**=**5, scoring **=** 'r2') scores\_100NN **=** cross\_val\_score(KNeighborsRegressor(n\_neighbors **=** 100), X, Y, cv**=**5, scoring **=** 'r2')

print('Пример значений коэф. детерминации для 5 фолдов для метода 10 ближайших соседей: \n', scores\_10NN, '\n\n' print('Усредненное значение коэффициента детерминации для:\n')

print('- 2 ближайших соседей:', np**.**mean(scores\_2NN), '\n') print('- 5 ближайших соседей:', np**.**mean(scores\_5NN), '\n') print('- 10 ближайших соседей:', np**.**mean(scores\_10NN), '\n') print('- 50 ближайших соседей:', np**.**mean(scores\_50NN), '\n') print('- 100 ближайших соседей:', np**.**mean(scores\_100NN), '\n')

Пример значений коэф. детерминации для 5 фолдов для метода 10 ближайших соседей: [0.34166201 0.38655715 0.14117213 0.28452217 0.2883947 ]

Усредненное значение коэффициента детерминации для:

- 2 ближайших соседей: 0.18694561138232885

- 5 ближайших соседей: 0.23548126907370337

- 10 ближайших соседей: 0.28846163209364245

- 50 ближайших соседей: 0.13534843218545478

- 100 ближайших соседей: 0.05880772437701802