

Министерство науки и высшего образования Российской Федерации  
Федеральное государственное автономное образовательное учреждение  
высшего образования  
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт перспективной инженерии  
Департамент цифровых, робототехнических систем и электроники

**ОТЧЕТ**  
**ПО ЛАБОРАТОРНОЙ РАБОТЕ №6**  
**дисциплины**  
**«Искусственный интеллект и машинное обучение»**  
**Вариант 9**

Выполнил:  
Кравчук Мирослав Витальевич  
2 курс, группа ИТС-б-о-23-1,  
11.03.02 «Инфокоммуникационные  
технологии и системы связи», очная  
форма обучения

---

(подпись)

Проверил:  
Доцент департамента цифровых,  
робототехнических систем и электроники  
Воронкин Р.А.

---

(подпись)

Отчет защищен с оценкой \_\_\_\_\_ Дата защиты \_\_\_\_\_

Ставрополь, 2025 г.

**Тема работы:** Основные этапы исследовательского анализа данных

**Цель работы:** научиться применять методы обработки данных в pandas.DataFrame , необходимые для разведочного анализа данных (EDA),включая работу с пропусками, выбросами, масштабирование и кодирование категориальных признаков.

Ссылка на git репозиторий: <https://github.com/miron2314/DLab-6.git>

**Порядок выполнения работы:**

1.Проработал примеры.

```
import pandas as pd
data = {
    "Имя": ["Анна", "Иван", "Ольга", None, "Мария"],
    "Возраст": [25, 30, None, 40, 35],
    "Город": ["Москва", None, "Казань", "Новосибирск", "СПб"]
}
df = pd.DataFrame(data)
print(df)
```

	Имя	Возраст	Город
0	Анна	25.0	Москва
1	Иван	30.0	None
2	Ольга	NaN	Казань
3	None	40.0	Новосибирск
4	Мария	35.0	СПб

Рисунок 1. Обнаружение пропусков в DataFrame. Метод .isna()

```
import pandas as pd
import numpy as np
data = {
    "Имя": ["Анна", "Иван", "Ольга", "Петр", "Мария", "Дмитрий"],
    "Возраст": [25, np.nan, 22, 40, 35, np.nan],
    "Город": ["Москва", "СПб", np.nan, "Новосибирск", "СПб", "Екатеринбург"],
    "Доход": [50000, 60000, np.nan, 70000, 65000, 55000]
}
df = pd.DataFrame(data)
print(df)
```


	Имя	Возраст	Город	Доход
0	Анна	25.0	Москва	50000.0
1	Иван	NaN	СПб	60000.0
2	Ольга	22.0	NaN	NaN
3	Петр	40.0	Новосибирск	70000.0
4	Мария	35.0	СПб	65000.0
5	Дмитрий	NaN	Екатеринбург	55000.0

Рисунок 2. Обнаружение пропусков в DataFrame. Метод. notna()

```

import pandas as pd
import numpy as np
# Генерируем DataFrame с пропусками
np.random.seed(42)
data = {
    "Имя": ["Анна", "Иван", "Ольга", "Петр", "Мария", "Дмитрий",
            "Елена", "Сергей", "Алина", "Артем"],
    "Возраст": [25, np.nan, 22, 40, 35, np.nan, 28, 31, np.nan, 29],
    "Город": ["Москва", "СПб", np.nan, "Новосибирск", "СПб",
              "Екатеринбург", np.nan, "Казань", "Томск", np.nan],
    "Доход": [50000, 60000, np.nan, 70000, 65000, 55000, 48000,
              np.nan, 52000, 58000],
    "Образование": [np.nan, "Высшее", "Среднее", "Высшее", np.nan,
                    "Высшее", "Среднее", "Высшее", np.nan, "Среднее"],
    "Стаж работы": [3, 8, 1, 15, 10, np.nan, 5, 7, np.nan, 2]
}
df = pd.DataFrame(data)
print(df)

```



	Имя	Возраст	Город	Доход	Образование	Стаж работы
0	Анна	25.0	Москва	50000.0	NaN	3.0
1	Иван	NaN	СПб	60000.0	Высшее	8.0
2	Ольга	22.0	NaN	NaN	Среднее	1.0
3	Петр	40.0	Новосибирск	70000.0	Высшее	15.0
4	Мария	35.0	СПб	65000.0	NaN	10.0
5	Дмитрий	NaN	Екатеринбург	55000.0	Высшее	NaN
6	Елена	28.0	NaN	48000.0	Среднее	5.0
7	Сергей	31.0	Казань	NaN	Высшее	7.0
8	Алина	NaN	Томск	52000.0	NaN	NaN
9	Артем	29.0	NaN	58000.0	Среднее	2.0

Рисунок 3. Визуализация пропусков с missingno

```
import pandas as pd
import numpy as np
data = {
    "Имя": ["Анна", "Иван", "Ольга", "Петр", "Мария", "Дмитрий"],
    "Возраст": [25, np.nan, 22, 40, 35, np.nan],
    "Город": ["Москва", "СПб", np.nan, "Новосибирск", "СПб",
    "Екатеринбург"],
    "Доход": [50000, 60000, np.nan, 70000, 65000, 55000]
}
df = pd.DataFrame(data)
print(df)
```

	Имя	Возраст	Город	Доход
0	Анна	25.0	Москва	50000.0
1	Иван	NaN	СПб	60000.0
2	Ольга	22.0	NaN	NaN
3	Петр	40.0	Новосибирск	70000.0
4	Мария	35.0	СПб	65000.0
5	Дмитрий	NaN	Екатеринбург	55000.0

Рисунок 4. Удаление строк с пропусками ( dropna() )

```
import pandas as pd
import numpy as np
df = pd.DataFrame({
    "день": [1, 2, 3, 4, 5],
    "температура": [20.0, np.nan, np.nan, 24.0, 25.0]
})
df["температура_интерп"] = df["температура"].interpolate()
print(df)
```

	день	температура	температура_интерп
0	1	20.0	20.000000
1	2	NaN	21.333333
2	3	NaN	22.666667
3	4	24.0	24.000000
4	5	25.0	25.000000

Рисунок 5. Линейная интерполяция (по умолчанию)

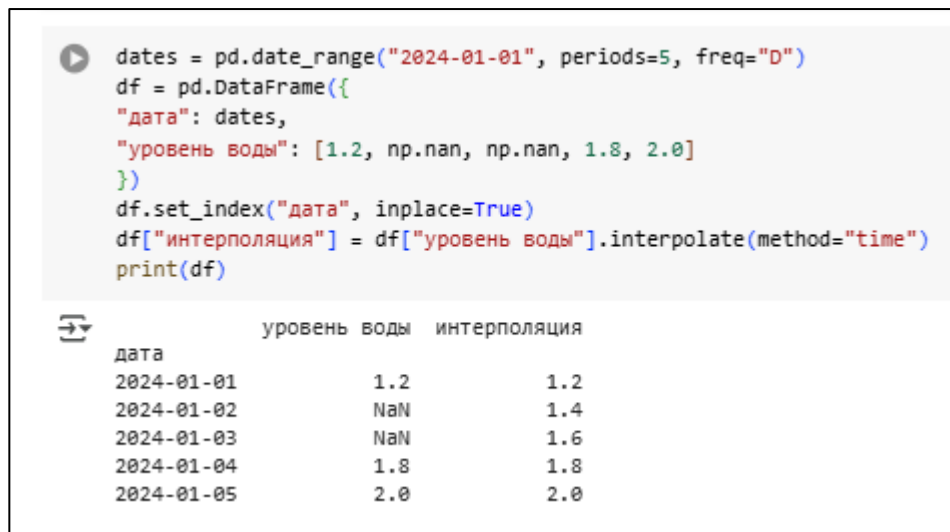


Рисунок 6. Интерполяция временных рядов ( method='time' )

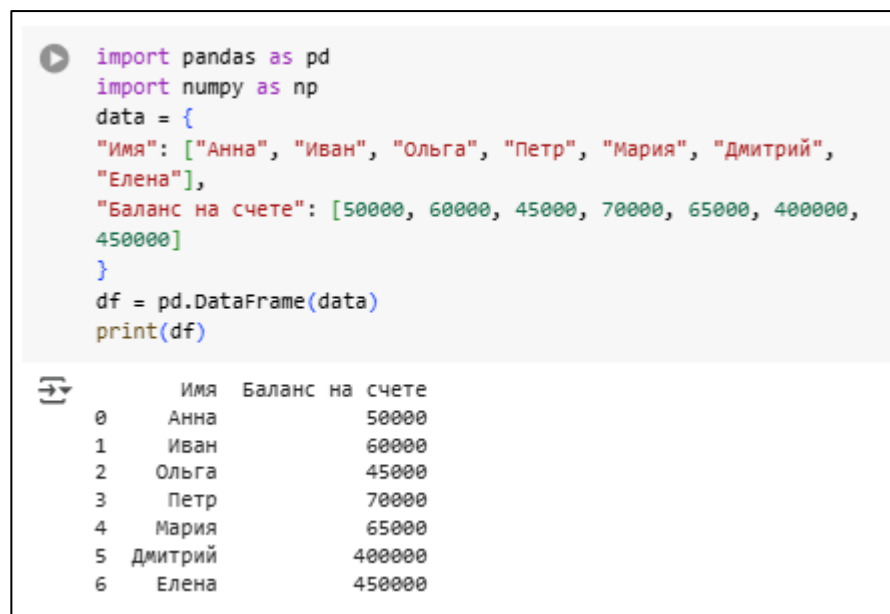


Рисунок 7. Обработка выбросов в pandas

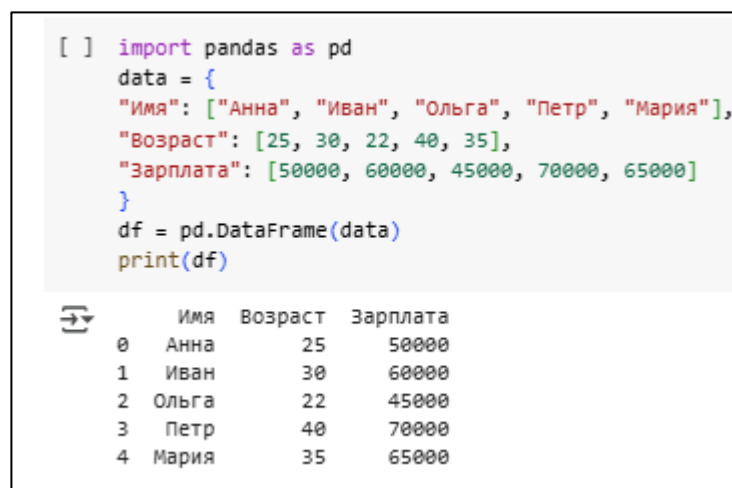


Рисунок 8. Стандартизация признаков

```
from sklearn.preprocessing import RobustScaler
scaler = RobustScaler()
scaled_values = scaler.fit_transform(df[["Возраст", "Зарплата"]])
df_scaled = df.copy()
df_scaled[["Возраст", "Зарплата"]] = scaled_values
print(df_scaled)
```

	Имя	Возраст	Зарплата
0	Анна	-0.5	-0.666667
1	Иван	0.0	0.000000
2	Ольга	-0.8	-1.000000
3	Петр	1.0	0.666667
4	Мария	0.5	0.333333

Рисунок 9. Стандартизация с использованием StandardScaler

```
import pandas as pd
data = {
    "Имя": ["Анна", "Иван", "Ольга", "Петр", "Мария"],
    "Образование": ["среднее", "высшее", "начальное", "высшее", "среднее"]
}
df = pd.DataFrame(data)
print(df)
```

	Имя	Образование
0	Анна	среднее
1	Иван	высшее
2	Ольга	начальное
3	Петр	высшее
4	Мария	среднее

Рисунок 10. Нормализация с использованием MinMaxScaler

```
ordered_mapping = {
    "начальное": 0,
    "среднее": 1,
    "высшее": 2
}
df_ordered = df.copy()
df_ordered["Образование"] = df["Образование"].map(ordered_mapping)
print(df_ordered)
```

	Имя	Образование
0	Анна	1
1	Иван	2
2	Ольга	0
3	Петр	2
4	Мария	1

Рисунок 11. Label Encoding с использованием pandas.map()

```
import pandas as pd
data = {
    "Имя": ["Анна", "Иван", "Ольга", "Петр", "Мария"],
    "Город": ["Москва", "СПб", "Казань", "Москва", "Казань"]
}
df = pd.DataFrame(data)
print(df)
```

	Имя	Город
0	Анна	Москва
1	Иван	СПб
2	Ольга	Казань
3	Петр	Москва
4	Мария	Казань

Рисунок 12. One-Hot Encoding

2.Выполнил практические задания

### Задание 1. Обнаружение и обработка пропущенных значений

Датасет: titanic (пассажиры Титаника)

Источник: `seaborn.load_dataset("titanic")`

#### Инструкции:

1. Загрузите датасет titanic
2. Определите количество пропущенных значений в каждом столбце.
3. Визуализируйте пропуски с помощью библиотеки missingno .
4. Заполните пропущенные значения:  
признак age — средним значением;  
признак embarked — наиболее частым значением;  
признак deck — удалите.
5. Отобразите информацию о таблице до и после обработки ( `.info()` , `.isna().sum()` ).

#### Листинг кода:

```
import pandas as pd
import seaborn as sns
import missingno as msno
import matplotlib.pyplot as plt
titanic = sns.load_dataset("titanic")
```

```

missing_values = titanic.isna().sum()
print("Количество пропущенных значений в каждом столбце:")
print(missing_values)
plt.figure(figsize=(10, 5))
msno.matrix(titanic)
plt.title('Пропущенные значения в датасете Titanic')
plt.show()
titanic['age'].fillna(titanic['age'].mean(), inplace=True)
titanic['embarked'].fillna(titanic['embarked'].mode()[0], inplace=True)
if 'deck' in titanic.columns:
    titanic.drop(columns=['deck'], inplace=True)
print("\nИнформация о таблице до обработки:")
print(titanic.info())
print("\nКоличество пропущенных значений после обработки:")
print(titanic.isna().sum())

```

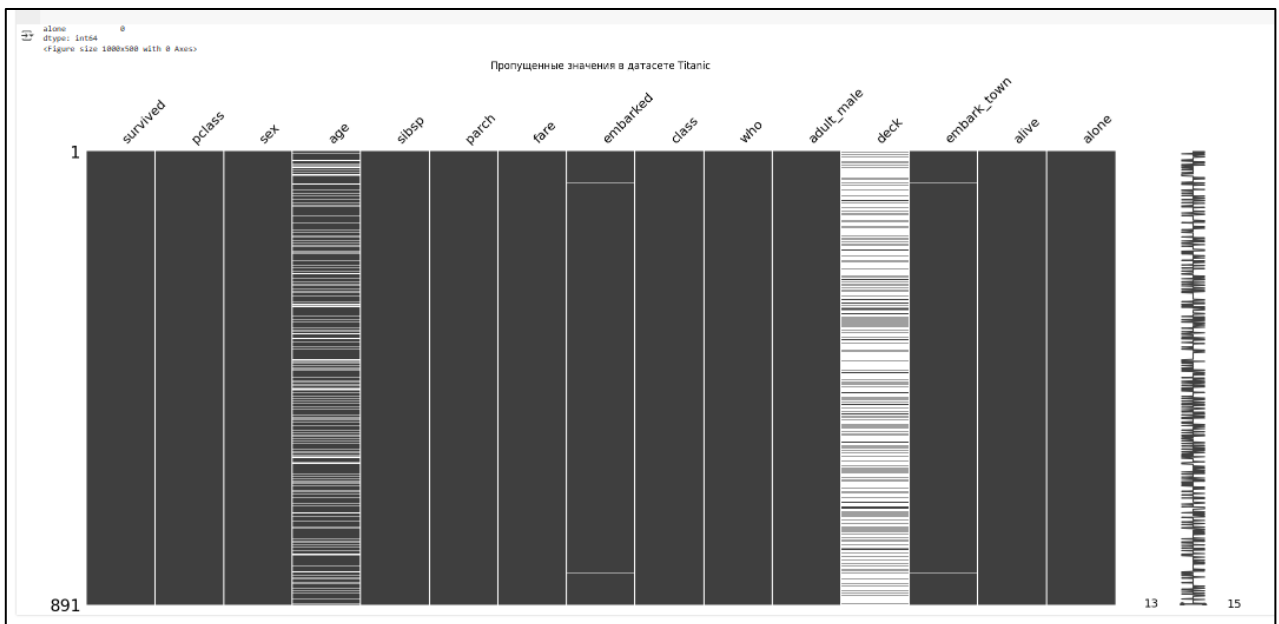


Рисунок 13. Практическое задание 1

## Задание 2. Обнаружение и удаление выбросов

**Датасет:** penguins (описание антарктических пингвинов)

**Источник:** seaborn.load\_dataset("penguins")

### Инструкции:

1. Загрузите датасет penguins.
2. Постройте boxplot-графики для признаков bill\_length\_mm , bill\_depth\_mm , flipper\_length\_mm , body\_mass\_g .
3. Используя метод межквартильного размаха (IQR), выявите и удалите выбросы по каждому из указанных признаков.



4. Сравните размеры датасета до и после фильтрации.
5. Постройте boxplot-график до и после удаления выбросов для одного из признаков.

### Листинг кода:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
penguins = sns.load_dataset("penguins")
features = ['bill_length_mm', 'bill_depth_mm', 'flipper_length_mm', 'body_mass_g']
plt.figure(figsize=(15, 10))
for i, feature in enumerate(features):
    plt.subplot(2, 2, i + 1)
    sns.boxplot(y=penguins[feature])
    plt.title(f'Boxplot для {feature}')
plt.tight_layout()
plt.show()

def remove_outliers(data, feature):
    Q1 = data[feature].quantile(0.25)
    Q3 = data[feature].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    return data[(data[feature] >= lower_bound) & (data[feature] <= upper_bound)]

for feature in features:
    penguins = remove_outliers(penguins, feature)
print(f'Размер датасета до удаления выбросов: {len(sns.load_dataset("penguins"))}')
print(f'Размер датасета после удаления выбросов: {len(penguins)}')
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
sns.boxplot(y=sns.load_dataset("penguins")['bill_length_mm'])
plt.title('Boxplot до удаления выбросов')
plt.subplot(1, 2, 2)
sns.boxplot(y=penguins['bill_length_mm'])
plt.title('Boxplot после удаления выбросов')
plt.tight_layout()
plt.show()
```

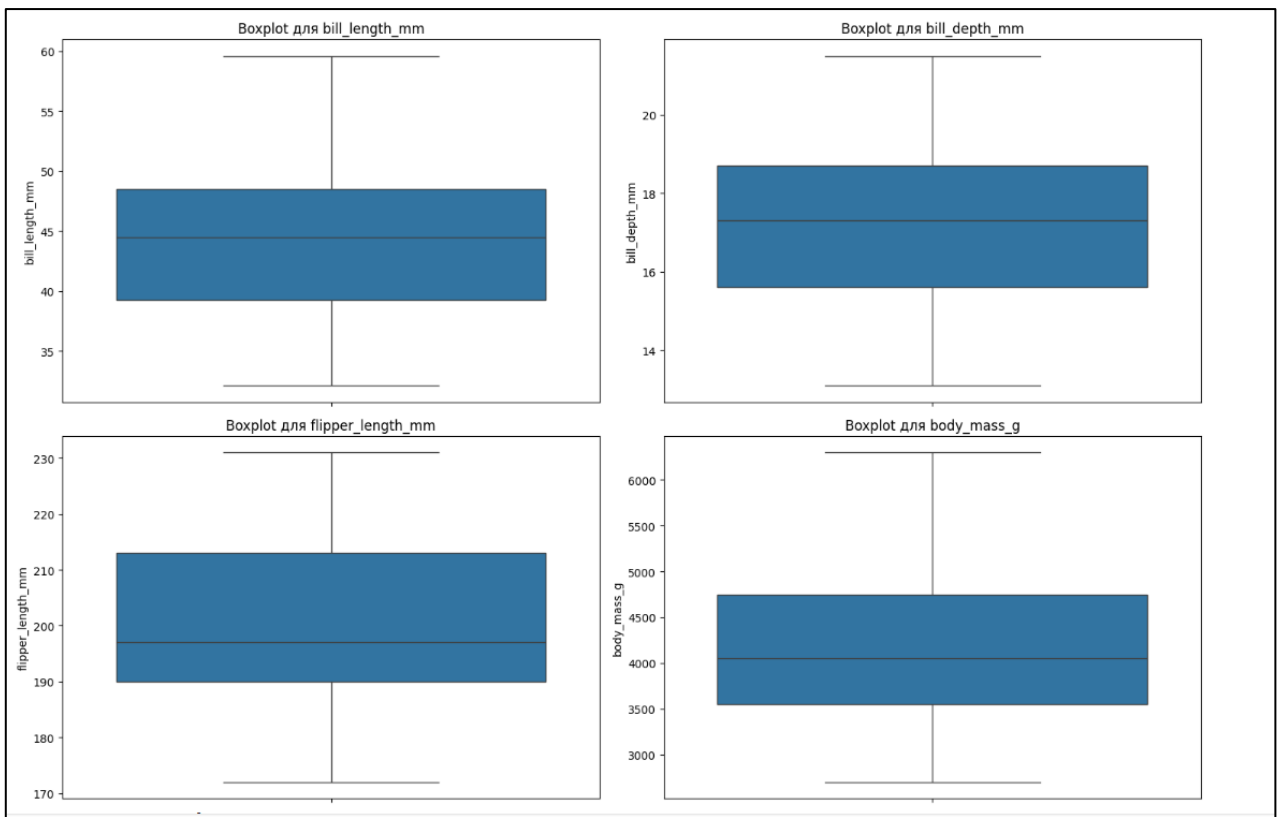


Рисунок 14. Практическое задание 2

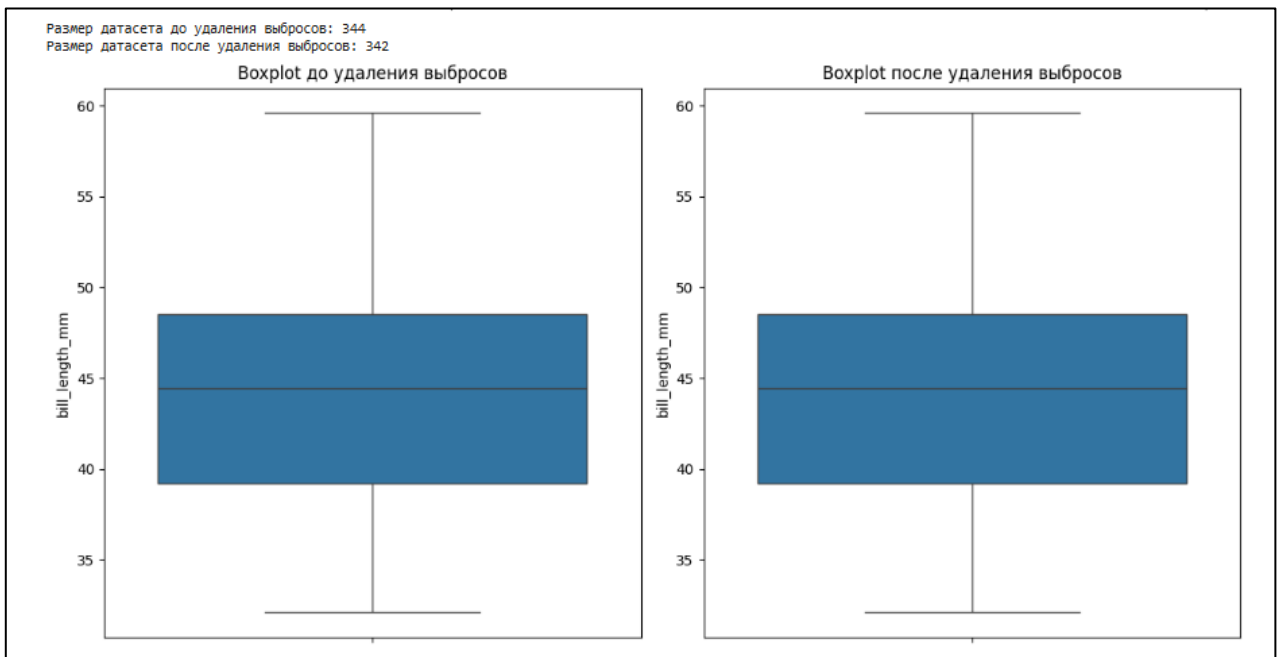


Рисунок 15. Практическое задание 2

### Задание 3. Масштабирование числовых признаков

Датасет: california housing

**Источник:** `from sklearn.datasets import fetch_california_housing`

**Инструкции:**

1. Загрузите данные с помощью `fetch_california_housing(as_frame=True)`.
2. Преобразуйте данные в `pandas.DataFrame`.
3. Выполните:  
стандартизацию признаков с помощью `StandardScaler`; нормализацию в диапазон `[0, 1]` с помощью `MinMaxScaler` (на копии таблицы).
4. Постройте гистограммы распределения признака `MedInc` до и после масштабирования.
5. Сравните поведение шкал на гистограммах.

**Листинг кода:**

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import fetch_california_housing
from sklearn.preprocessing import StandardScaler, MinMaxScaler
california_housing = fetch_california_housing(as_frame=True)
df = california_housing.frame
scaler = StandardScaler()
df['MedInc_standardized'] = scaler.fit_transform(df[['MedInc']])
minmax_scaler = MinMaxScaler()
df['MedInc_normalized'] = minmax_scaler.fit_transform(df[['MedInc']])
plt.figure(figsize=(12, 6))
plt.subplot(1, 3, 1)
plt.hist(df['MedInc'], bins=30, color='blue', alpha=0.7)
plt.title('Оригинальный MedInc')
plt.xlabel('MedInc')
plt.ylabel('Частота')
plt.subplot(1, 3, 2)
plt.hist(df['MedInc_standardized'], bins=30, color='green', alpha=0.7)
plt.title('Стандартизированный MedInc')
plt.xlabel('MedInc (стандартизированный)')
plt.ylabel('Частота')
plt.subplot(1, 3, 3)
plt.hist(df['MedInc_normalized'], bins=30, color='orange', alpha=0.7)
plt.title('Нормализованный MedInc')
plt.xlabel('MedInc (нормализованный)')
plt.ylabel('Частота')
plt.tight_layout()
plt.show()
```

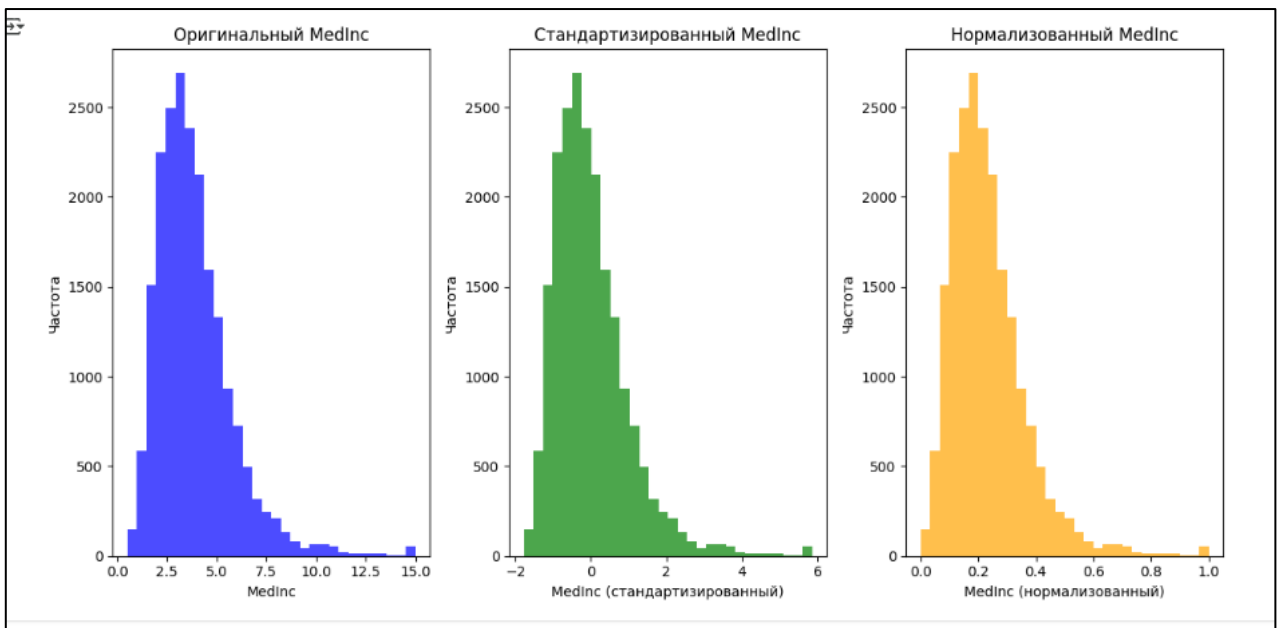


Рисунок 16. Практическое задание 3

#### Задание 4. Кодирование категориальных признаков

**Датасет:** adult (перепись населения США, income dataset)

**Источник:** <https://archive.ics.uci.edu/ml/datasets/adult>

**Или через библиотеку** `sklearn.datasets.fetch_openml("adult")`

#### Инструкции:

1. Загрузите данные и отберите признаки: категориальные: education, marital-status, occupation; целевой признак: income.
2. Проведите Label Encoding для признака education, предполагая, что уровни образования упорядочены.
3. Примените One-Hot Encoding к признакам marital-status и occupation.
4. Проверьте итоговую размерность таблицы до и после кодирования.
5. Убедитесь, что в one-hot-кодировании не присутствует дамми-ловушка

#### Листинг кода:

```
import pandas as pd
from sklearn.datasets import fetch_openml
from sklearn.preprocessing import LabelEncoder
adult_data = fetch_openml("adult", version=2, as_frame=True)
df = adult_data.frame
print("Доступные столбцы в датасете:")
print(df.columns.tolist())
print("\nПервые 5 строк DataFrame:")
```

```

print(df.head())
try:
    df = df[['education', 'marital-status', 'occupation', 'class']]
except KeyError as e:
    print(f"Ошибка: {e}. Проверьте названия столбцов.")
education_ordered = [
    'Preschool', '1st-4th', '5th-6th', '7th-8th', '9th',
    '10th', '11th', '12th', 'HS-grad', 'Some-college',
    'Bachelors', 'Masters', 'Doctorate'
]
label_encoder = LabelEncoder()
df['education_encoded'] = label_encoder.fit_transform(df['education'])
df = pd.get_dummies(df, columns=['marital-status', 'occupation'], drop_first=True)
print("Размерность до кодирования:", df.shape[0], "строк и", df.shape[1], "столбцов")
print("Размерность после кодирования:", df.shape[0], "строк и", df.shape[1], "столбцов")
dummy_columns = [col for col in df.columns if 'marital-status_' in col or 'occupation_' in col]
print("Проверка на дамми-ловушку:")
print(df[dummy_columns].sum(axis=1).value_counts())

```

Доступные столбцы в датасете:  
['age', 'workclass', 'fnlwgt', 'education', 'education-num', 'marital-status', 'occupation', 'relationship', 'race', 'sex', 'capital-gain', 'capital-loss', 'hours-per-week', 'native-country', 'class']

Первые 5 строк DataFrame:

	age	workclass	fnlwgt	education	education-num	marital-status
0	25	Private	226802	11th	7	Never-married
1	38	Private	89814	HS-grad	9	Married-civ-spouse
2	28	Local-gov	336951	Assoc-acdm	12	Married-civ-spouse
3	44	Private	160323	Some-college	10	Married-civ-spouse
4	18	NaN	103497	Some-college	10	Never-married

occupation relationship race sex capital-gain capital-loss \

	Machine-op-inspct	Own-child	Black	Male	0	0
0	Machine-op-inspct	Own-child	Black	Male	0	0
1	Farming-fishing	Husband	White	Male	0	0
2	Protective-serv	Husband	White	Male	0	0
3	Machine-op-inspct	Husband	Black	Male	7688	0
4	NaN	Own-child	White	Female	0	0

hours-per-week native-country class

	hours-per-week	native-country	class
0	40	United-States	<=50K
1	50	United-States	<=50K
2	40	United-States	>50K
3	40	United-States	>50K
4	30	United-States	<=50K

Размерность до кодирования: 48842 строк и 22 столбцов  
Размерность после кодирования: 48842 строк и 22 столбцов  
Проверка на дамми-ловушку:

	35251	12129	0
2	35251	12129	0
1	12129	0	1462
0	1462	0	0

Name: count, dtype: int64  
<ipython-input-6-6512a695d97>:31: SettingWithCopyWarning:  
A value is trying to be set on a copy of a slice from a DataFrame.  
Try using .loc[row\_indexer,col\_indexer] = value instead  
See the caveats in the documentation: [https://pandas.pydata.org/pandas-docs/stable/user\\_guide/indexing.html#returning-a-view-versus-a-copy](https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)  
df['education\_encoded'] = label\_encoder.fit\_transform(df['education'])

Рисунок 17. Практическое задание 4

## Задание 5. Комплексный EDA

**Датасет: heart disease (заболевания сердца)**

**Источник:**<https://www.kaggle.com/datasets/fedesoriano/heart-failure-prediction>

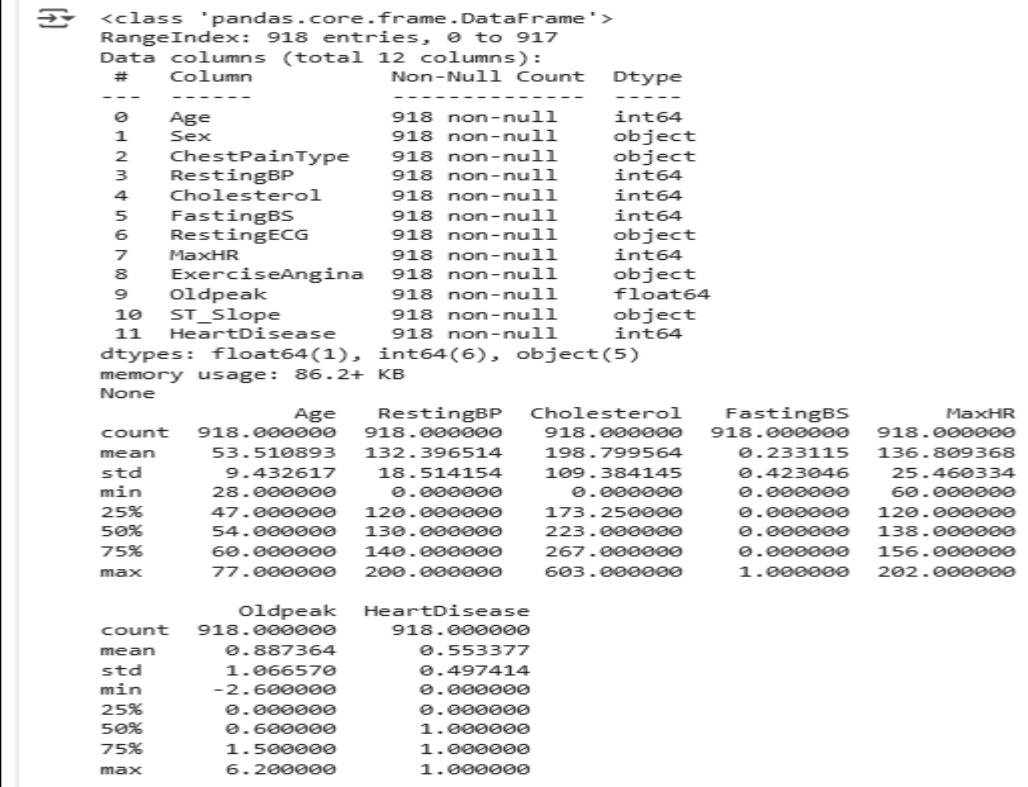
1. Обзор структуры данных ( .info() , .describe() ).
2. Обнаружение и обработка пропущенных значений.

3. Обнаружение и удаление выбросов по признакам: age , cholesterol , restingbp , maxhr .
4. Масштабирование числовых признаков.
5. Кодирование категориальных признаков: sex , chestpain , exerciseangina ,restecg .
6. Подготовьте отчёт в виде Jupyter-ноутбука с комментариями к каждому этапу и промежуточными результатами.

## 1. Обзор структуры данных

### Листинг кода:

```
# Загрузка данных
heart_disease_df = pd.read_csv('heart.csv')
# Обзор структуры
print(heart_disease_df.info())
# Обзор статистики
print(heart_disease_df.describe())
```



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 918 entries, 0 to 917
Data columns (total 12 columns):
#   Column                Non-Null Count  Dtype
---  -
0   Age                   918 non-null   int64
1   Sex                   918 non-null   object
2   ChestPainType         918 non-null   object
3   RestingBP             918 non-null   int64
4   Cholesterol            918 non-null   int64
5   FastingBS             918 non-null   int64
6   RestingECG            918 non-null   object
7   MaxHR                 918 non-null   int64
8   ExerciseAngina        918 non-null   object
9   Oldpeak               918 non-null   float64
10  ST_Slope              918 non-null   object
11  HeartDisease           918 non-null   int64
dtypes: float64(1), int64(6), object(5)
memory usage: 86.2+ KB
None
```

	Age	RestingBP	Cholesterol	FastingBS	MaxHR
count	918.000000	918.000000	918.000000	918.000000	918.000000
mean	53.510893	132.396514	198.799564	0.233115	136.809368
std	9.432617	18.514154	109.384145	0.423046	25.460334
min	28.000000	0.000000	0.000000	0.000000	60.000000
25%	47.000000	120.000000	173.250000	0.000000	120.000000
50%	54.000000	130.000000	223.000000	0.000000	138.000000
75%	60.000000	140.000000	267.000000	0.000000	156.000000
max	77.000000	200.000000	603.000000	1.000000	202.000000

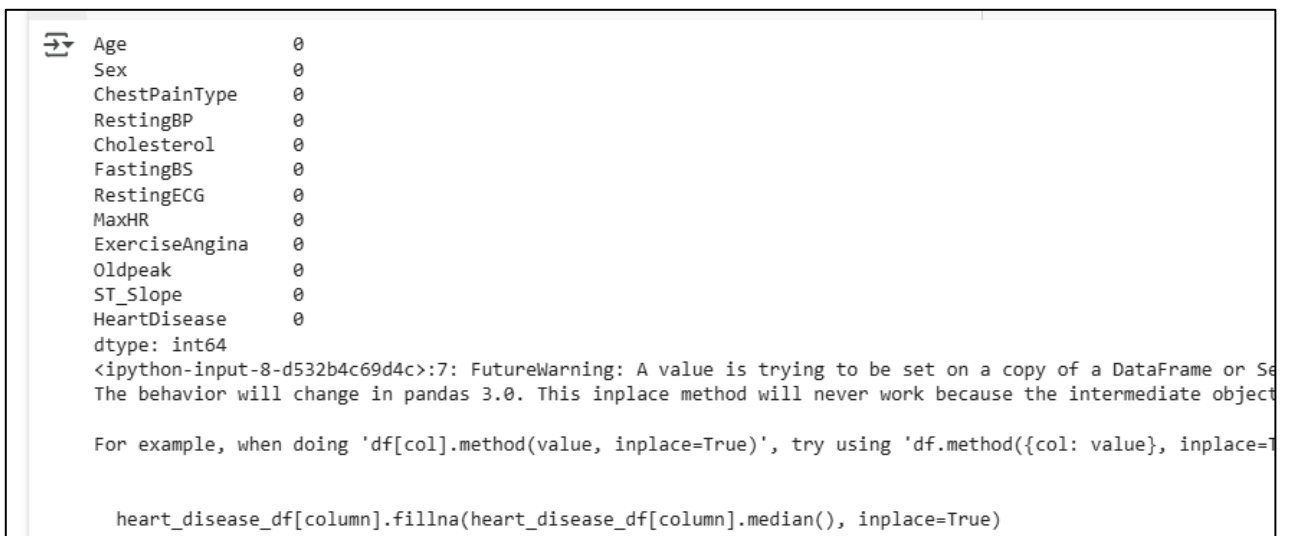
	Oldpeak	HeartDisease
count	918.000000	918.000000
mean	0.887364	0.553377
std	1.066570	0.497414
min	-2.600000	0.000000
25%	0.000000	0.000000
50%	0.600000	1.000000
75%	1.500000	1.000000
max	6.200000	1.000000

Рисунок 18. Обзор структуры данных

## 2. Обнаружение и обработка пропущенных значений

### Листинг кода:

```
print(heart_disease_df.isnull().sum())
for column in ['RestingBP', 'Cholesterol', 'MaxHR']:
    heart_disease_df[column].fillna(heart_disease_df[column].median(), inplace=True)
heart_disease_df.dropna(subset=['ChestPainType', 'ExerciseAngina', 'RestingECG'], inplace=True)
```



```
Age      0
Sex      0
ChestPainType  0
RestingBP  0
Cholesterol  0
FastingBS  0
RestingECG  0
MaxHR     0
ExerciseAngina  0
Oldpeak   0
ST_Slope  0
HeartDisease  0
dtype: int64
<ipython-input-8-d532b4c69d4c>:7: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object
For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)'

heart_disease_df[column].fillna(heart_disease_df[column].median(), inplace=True)
```

Рисунок 19. Обнаружение и обработка пропущенных значений

## 3. Обнаружение и удаление выбросов

### Листинг кода:

```
def remove_outliers(df, column):
    q1 = df[column].quantile(0.25)
    q3 = df[column].quantile(0.75)
    iqr = q3 - q1
    lower_bound = q1 - 1.5 * iqr
    upper_bound = q3 + 1.5 * iqr
    return df[(df[column] >= lower_bound) & (df[column] <= upper_bound)]
for feature in ['Age', 'Cholesterol', 'RestingBP', 'MaxHR']:
    heart_disease_df = remove_outliers(heart_disease_df, feature)
```

```

✓ [18] def remove_outliers(df, column):
0     q1 = df[column].quantile(0.25)
сек.   q3 = df[column].quantile(0.75)
        iqr = q3 - q1
        lower_bound = q1 - 1.5 * iqr
        upper_bound = q3 + 1.5 * iqr
        return df[(df[column] >= lower_bound) & (df[column] <= upper_bound)]
    for feature in ['Age', 'Cholesterol', 'RestingBP', 'MaxHR']:
        heart_disease_df = remove_outliers(heart_disease_df, feature)

```

Рисунок 20. Обнаружение и удаление выбросов

#### 4. Масштабирование числовых признаков

##### Листинг кода:

```

from sklearn.preprocessing import StandardScaler

# Числовые признаки для масштабирования
numeric_features = ['Age', 'RestingBP', 'Cholesterol', 'MaxHR']
scaler = StandardScaler()
heart_disease_df[numeric_features] = scaler.fit_transform(heart_disease_df[numeric_features])

```

```

▶ from sklearn.preprocessing import StandardScaler

# Числовые признаки для масштабирования
numeric_features = ['Age', 'RestingBP', 'Cholesterol', 'MaxHR']
scaler = StandardScaler()
heart_disease_df[numeric_features] = scaler.fit_transform(heart_disease_df[numeric_features])

```

Рисунок 21. Масштабирование числовых признаков

#### 5. Кодирование категориальных признаков

##### Листинг кода:

```

heart_disease_df = pd.get_dummies(
    heart_disease_df,
    columns=['Sex', 'ChestPainType', 'ExerciseAngina', 'RestingECG'],
    drop_first=True
)

```



## 5. Кодирование категориальных признаков

```
heart_disease_df = pd.get_dummies(
    heart_disease_df,
    columns=['Sex', 'ChestPainType', 'ExerciseAngina', 'RestingECG'],
    drop_first=True
)
```

Рисунок 22. Кодирование категориальных признаков

3. Вывел итоговый датасет.

```
print(heart_disease_df.head())
```

	Age	RestingBP	Cholesterol	FastingBS	MaxHR	Oldpeak	ST_Slope	\
0	-1.343776	0.539269	0.962124	0	1.296933	0.0	Up	
1	-0.400479	1.836853	-1.180312	0	0.640055	1.0	Flat	
2	-1.658208	-0.109523	0.844191	0	-1.741130	0.0	Up	
3	-0.505290	0.409510	-0.512029	0	-1.330581	1.5	Flat	
4	0.123574	1.188061	-0.885481	0	-0.755812	0.0	Up	

	HeartDisease	Sex_M	ChestPainType_ATA	ChestPainType_NAP	\
0	0	True	True	False	
1	1	False	False	True	
2	0	True	True	False	
3	1	False	False	False	
4	0	True	False	True	

	ChestPainType_TA	ExerciseAngina_Y	RestingECG_Normal	RestingECG_ST
0	False	False	True	False
1	False	False	True	False
2	False	False	False	True
3	False	True	True	False
4	False	False	True	False

Рисунок 23. Итоговый датасет

4. Выполнил индивидуальное задание.

### Индивидуальное задание

**Цель:** выполнить полноценный исследовательский анализ данных (EDA), применяя методы выявления и обработки пропусков, выбросов, масштабирования числовых признаков и кодирования категориальных переменных.

**Условия выполнения:**

Выберите реальный табличный датасет со структурой не менее 8 признаков и целевым признаком, подходящим для задач регрессии или классификации. Возможные варианты: датасеты с платформ Kaggle, UCI, OpenML; встроенные датасеты из библиотек sklearn , seaborn ; или запросите рекомендованный набор у преподавателя.

Убедитесь, что в датасете присутствуют: числовые признаки, категориальные признаки, целевой столбец (предсказуемый признак), необязательно: пропущенные значения или выбросы.

Требования к выполнению:

Выполните последовательные шаги исследовательского анализа:

#### 1. Обзор структуры данных

Загрузите датасет.

Выведите общую информацию ( .info() , .describe() ).

Опишите: сколько признаков, каких типов, какова структура целевого признака

#### 2. Обнаружение и обработка пропусков

Определите, есть ли пропущенные значения.

Обоснуйте выбранный способ их устранения (удаление, заполнение средним/модой и т.д.).

Примените выбранный способ

#### 3. Обнаружение и удаление выбросов

Выберите 3–5 числовых признаков.

Используя метод IQR, удалите выбросы.

Сравните объём данных до и после очистки.

#### 4. Масштабирование числовых признаков

Выполните стандартизацию (z-преобразование) с помощью StandardScaler .

Объясните, зачем выполняется масштабирование

#### 5. Кодирование категориальных признаков

Выполните:

Label Encoding для порядковых признаков (при наличии);

One-Hot Encoding для номинальных признаков.

Проверьте, исключена ли дамми-ловушка.

## 6. Финальный набор данных

Убедитесь, что датасет не содержит пропусков, выбросов, категориальных данных в строковом виде.

Признаки приведены к числовому виду, масштабированы.

Представьте итоговый DataFrame, готовый к использованию в моделях

### ✓ 1. Обзор структуры данных

```
!pip install -q kaggle
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
!mkdir -p ~/.kaggle
!echo '{"username":"yourusername","key":"yourkey"}' > ~/.kaggle/kaggle.json # Замените на свои данные
!chmod 600 ~/.kaggle/kaggle.json
!kaggle datasets download -d fedesoriano/heart-failure-prediction
!unzip -q heart-failure-prediction.zip
df = pd.read_csv('heart.csv')
print("Исходный размер данных:", df.shape)
print("\nПервые 5 строк данных:")
display(df.head())
```

Dataset URL: <https://www.kaggle.com/datasets/fedesoriano/heart-failure-prediction>  
License(s): ODbL-1.0  
heart-failure-prediction.zip: Skipping, found more recently modified local copy (use --force to force download)  
replace heart.csv? [y]es, [n]o, [A]ll, [N]one, [r]ename: Исходный размер данных: (918, 12)

Первые 5 строк данных:

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope	HeartDisease
0	40	M	ATA	140	289	0	Normal	172	N	0.0	Up	0
1	49	F	NAP	160	180	0	Normal	156	N	1.0	Flat	1
2	37	M	ATA	130	283	0	ST	98	N	0.0	Up	0
3	48	F	ASY	138	214	0	Normal	108	Y	1.5	Flat	1
4	54	M	NAP	150	195	0	Normal	122	N	0.0	Up	0

Рисунок 24. Обзор структуры данных

### Листинг кода:

```
print("\n=== Информация о датасете ===")
df.info()
print("\n=== Статистическое описание ===")
display(df.describe(include='all').T)
print("\n=== Распределение целевой переменной ===")
print(df['HeartDisease'].value_counts(normalize=True))
```

```

=== Информация о датасете ===
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 918 entries, 0 to 917
Data columns (total 12 columns):
#   Column                Non-Null Count  Dtype
---  -
0   Age                    918 non-null    int64
1   Sex                    918 non-null    object
2   ChestPainType          918 non-null    object
3   RestingBP              918 non-null    int64
4   Cholesterol             918 non-null    int64
5   FastingBS              918 non-null    int64
6   RestingECG             918 non-null    object
7   MaxHR                  918 non-null    int64
8   ExerciseAngina         918 non-null    object
9   Oldpeak                918 non-null    float64
10  ST_Slope               918 non-null    object
11  HeartDisease           918 non-null    int64
dtypes: float64(1), int64(6), object(5)
memory usage: 86.2+ KB

=== Статистическое описание ===

```

	count	unique	top	freq	mean	std	min	25%	50%	75%	max
Age	918.0	NaN	NaN	NaN	53.510893	9.432617	28.0	47.0	54.0	60.0	77.0
Sex	918	2	M	725	NaN	NaN	NaN	NaN	NaN	NaN	NaN
ChestPainType	918	4	ASY	496	NaN	NaN	NaN	NaN	NaN	NaN	NaN
RestingBP	918.0	NaN	NaN	NaN	132.396514	18.514154	0.0	120.0	130.0	140.0	200.0
Cholesterol	918.0	NaN	NaN	NaN	198.799564	109.384145	0.0	173.25	223.0	267.0	603.0
FastingBS	918.0	NaN	NaN	NaN	0.233115	0.423046	0.0	0.0	0.0	0.0	1.0
RestingECG	918	3	Normal	552	NaN	NaN	NaN	NaN	NaN	NaN	NaN
MaxHR	918.0	NaN	NaN	NaN	136.809368	25.460334	60.0	120.0	138.0	156.0	202.0
ExerciseAngina	918	2	N	547	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Oldpeak	918.0	NaN	NaN	NaN	0.887364	1.06657	-2.6	0.0	0.6	1.5	6.2
ST_Slope	918	3	Flat	460	NaN	NaN	NaN	NaN	NaN	NaN	NaN
HeartDisease	918.0	NaN	NaN	NaN	0.553377	0.497414	0.0	0.0	1.0	1.0	1.0

```

=== Распределение целевой переменной ===
HeartDisease
1    0.553377
0    0.446623
Name: proportion, dtype: float64

```

Рисунок 25. Обзор структуры данных

## 2. Обнаружение и обработка пропущенных значений

```
print("\n=== Пропущенные значения ===")  
print(df.isnull().sum())
```



```
=== Пропущенные значения ===  
Age          0  
Sex          0  
ChestPainType 0  
RestingBP    0  
Cholesterol  0  
FastingBS    0  
RestingECG   0  
MaxHR        0  
ExerciseAngina 0  
Oldpeak      0  
ST_Slope     0  
HeartDisease  0  
dtype: int64
```

Рисунок 26. Обнаружение и обработка пропущенных значений

## 3. Обнаружение и удаление выбросов

### Листинг кода:

```
# Визуализация выбросов  
plt.figure(figsize=(15, 10))  
features = ['Age', 'RestingBP', 'Cholesterol', 'MaxHR']  
  
for i, feature in enumerate(features, 1):  
    plt.subplot(2, 2, i)  
    sns.boxplot(x=df[feature])  
    plt.title(f'Boxplot для {feature}')  
plt.tight_layout()  
plt.show()  
  
# Удаление выбросов методом IQR  
def remove_outliers(df, columns):  
    for col in columns:  
        Q1 = df[col].quantile(0.25)  
        Q3 = df[col].quantile(0.75)  
        IQR = Q3 - Q1  
        lower_bound = Q1 - 1.5 * IQR  
        upper_bound = Q3 + 1.5 * IQR  
        df = df[(df[col] >= lower_bound) & (df[col] <= upper_bound)]  
    return df  
  
df_clean = remove_outliers(df, features)  
print(f"\nУдалено записей: {len(df) - len(df_clean)}")  
print(f"Новый размер данных: {df_clean.shape}")
```

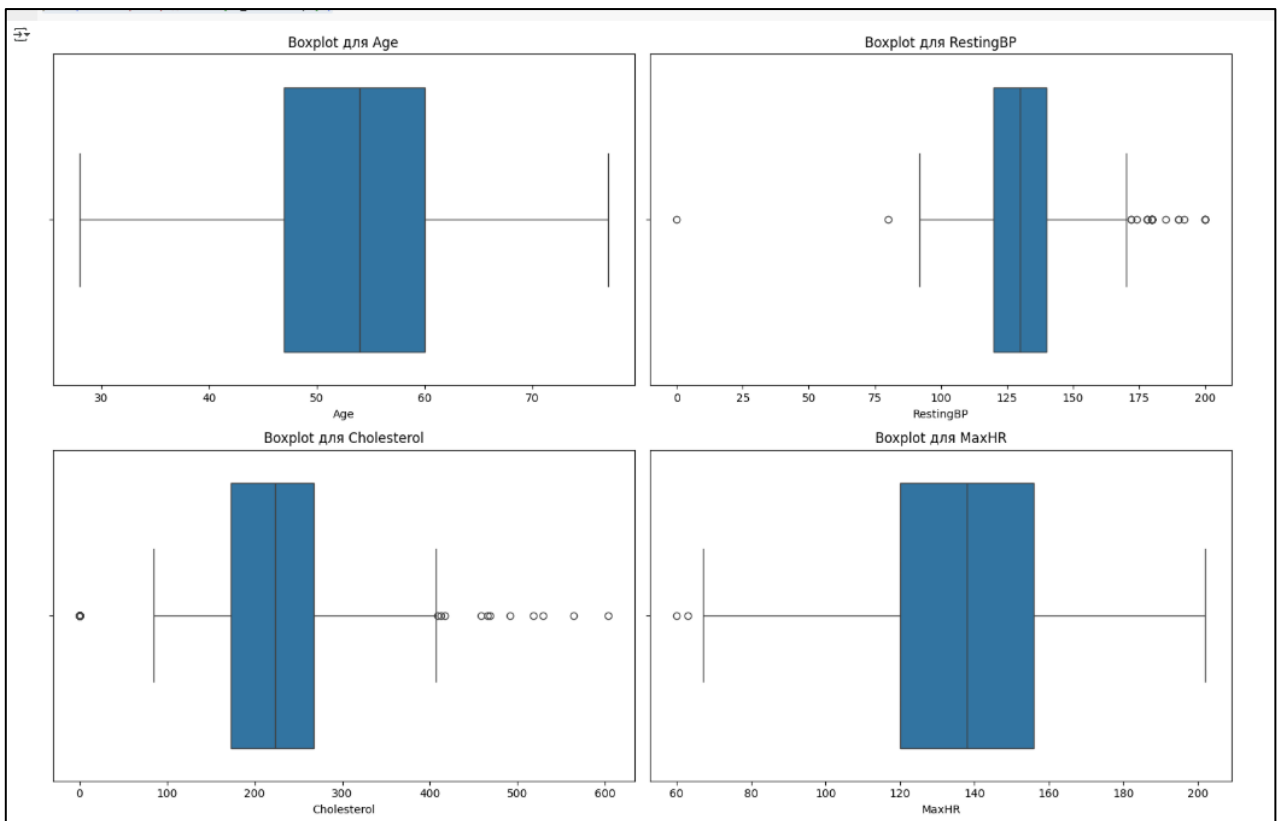


Рисунок 27. Обнаружение и удаление выбросов

## 4. Масштабирование числовых признаков

### Листинг кода:

```
# Обоснование: Стандартизация улучшает работу алгоритмов ML
numerical_features = ['Age', 'RestingBP', 'Cholesterol', 'MaxHR', 'Oldpeak']
scaler = StandardScaler()
df_clean[numerical_features] = scaler.fit_transform(df_clean[numerical_features])

# Визуализация после масштабирования
plt.figure(figsize=(15, 10))
for i, feature in enumerate(numerical_features, 1):
    plt.subplot(2, 3, i)
    sns.histplot(df_clean[feature], kde=True)
    plt.title(f'Распределение {feature} после масштабирования')
plt.tight_layout()
plt.show()
```

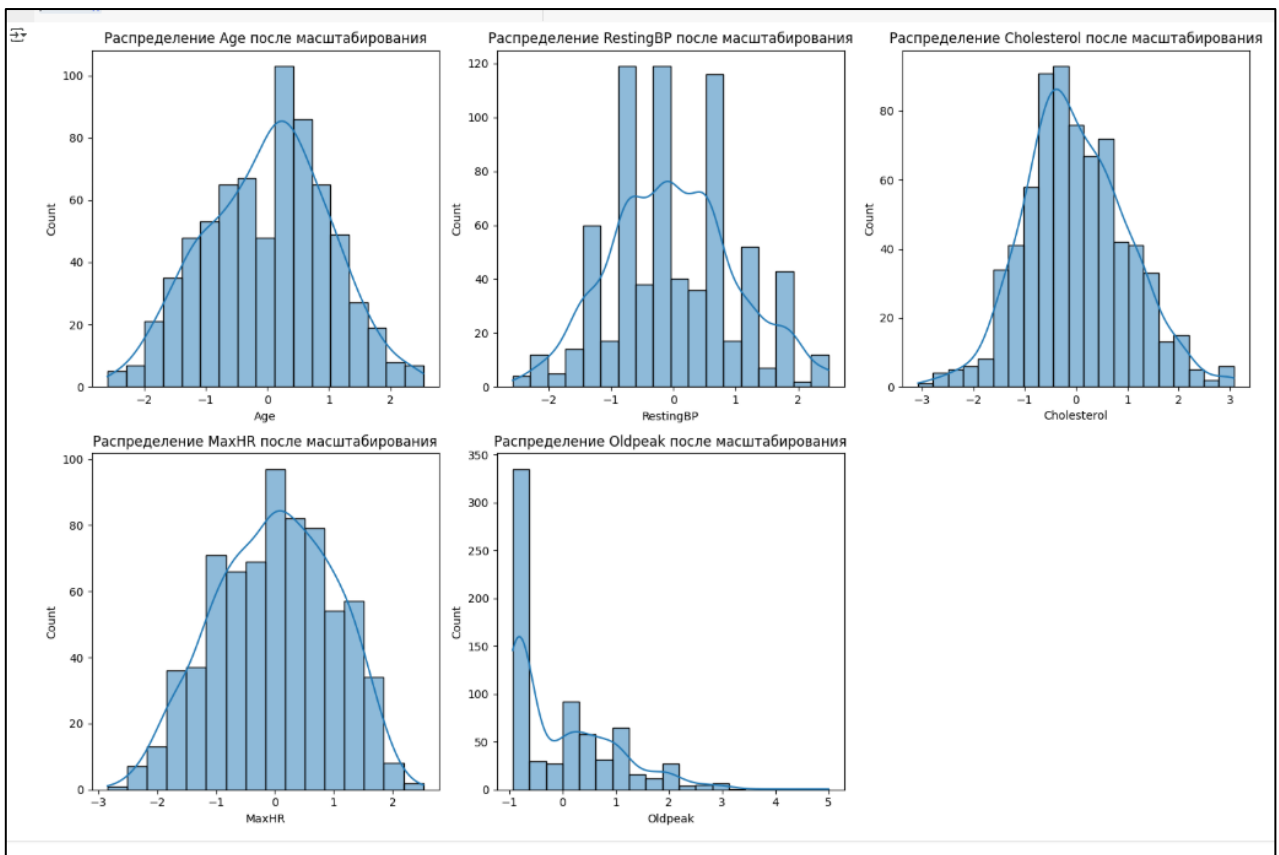


Рисунок 28. Масштабирование числовых признаков

## 5. Кодирование категориальных признаков

```
# Преобразование категориальных признаков
# Бинарные признаки
df_clean['Sex'] = df_clean['Sex'].map({'F': 0, 'M': 1})
df_clean['ExerciseAngina'] = df_clean['ExerciseAngina'].map({'N': 0, 'Y': 1})

# Порядковый признак (Label Encoding)
st_slope_mapping = {'Flat': 1, 'Up': 2, 'Down': 0}
df_clean['ST_Slope'] = df_clean['ST_Slope'].map(st_slope_mapping)

# One-Hot Encoding для номинальных признаков
nominal_features = ['ChestPainType', 'RestingECG']
df_clean = pd.get_dummies(df_clean, columns=nominal_features, drop_first=True)

# Проверка исключения дамми-ловушки
print("\nПроверка дамми-ловушки:")
print("ChestPainType features:", [col for col in df_clean.columns if 'ChestPainType' in col])
print("RestingECG features:", [col for col in df_clean.columns if 'RestingECG' in col])
```

Проверка дамми-ловушки:  
 ChestPainType features: ['ChestPainType\_ATA', 'ChestPainType\_NAP', 'ChestPainType\_TA']  
 RestingECG features: ['RestingECG\_Normal', 'RestingECG\_ST']

Рисунок 29. Кодирование категориальных признаков

## 6. Финальный набор данных

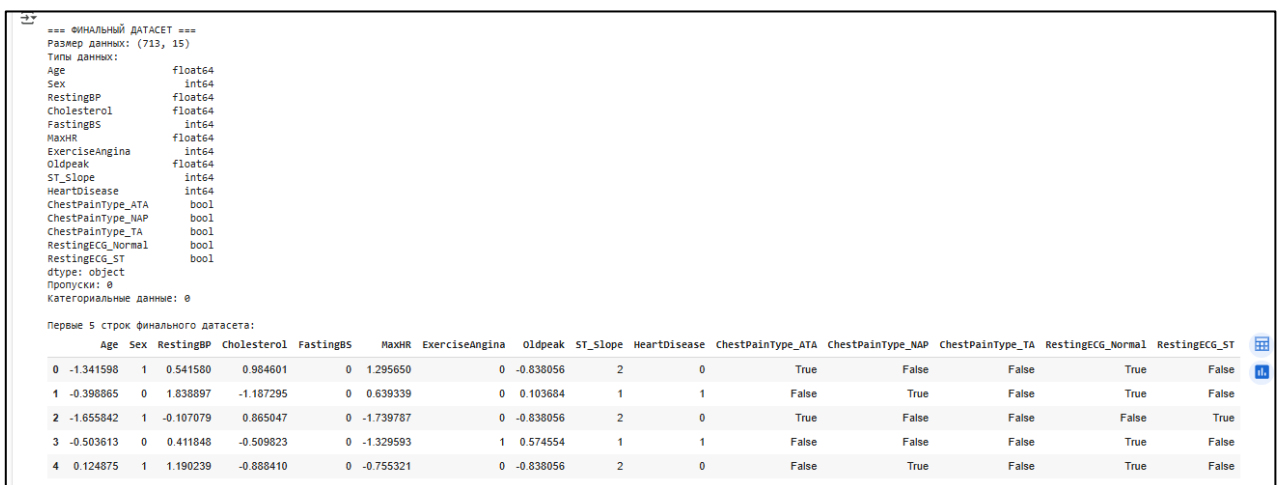
### Листинг кода:

```
# Проверка финального датасета
print("\n=== ФИНАЛЬНЫЙ ДАТАСЕТ ===")
print(f"Размер данных: {df_clean.shape}")
print(f"Типы данных:\n{df_clean.dtypes}")
print(f"Пропуски: {df_clean.isnull().sum().sum()}")
print(f"Категориальные данные: {sum(df_clean.dtypes == 'object')}")

# Первые 5 строк
print("\nПервые 5 строк финального датасета:")
display(df_clean.head())

# Визуализация корреляций
plt.figure(figsize=(16, 12))
corr = df_clean.corr()
sns.heatmap(corr, annot=False, cmap='coolwarm',
            mask=np.triu(np.ones_like(corr, dtype=bool)))
plt.title('Матрица корреляций финального датасета')
plt.show()

# Сохранение результата
df_clean.to_csv('heart_disease_processed.csv', index=False)
```



The screenshot shows a Jupyter Notebook interface. The top part displays the output of the code from the previous block, summarizing the dataset: 713 rows, 15 columns, with various data types and no missing values. Below this, the first 5 rows of the dataset are displayed in a table format.

	Age	Sex	RestingBP	Cholesterol	FastingBS	MaxHR	ExerciseAngina	Oldpeak	ST_Slope	HeartDisease	ChestPainType_ATA	ChestPainType_NAP	ChestPainType_TA	RestingECG_Normal	RestingECG_ST
0	-1.341598	1	0.541580	0.984601	0	1.295650	0	-0.838056	2	0	True	False	False	True	False
1	-0.398865	0	1.838897	-1.187295	0	0.639339	0	0.103684	1	1	False	True	False	True	False
2	-1.655842	1	-0.107079	0.865047	0	-1.739787	0	-0.838056	2	0	True	False	False	False	True
3	-0.503613	0	0.411848	-0.509823	0	-1.329593	1	0.574554	1	1	False	False	False	True	False
4	0.124875	1	1.190239	-0.888410	0	-0.755321	0	-0.838056	2	0	False	True	False	True	False

Рисунок 30. Финальный сбор данных



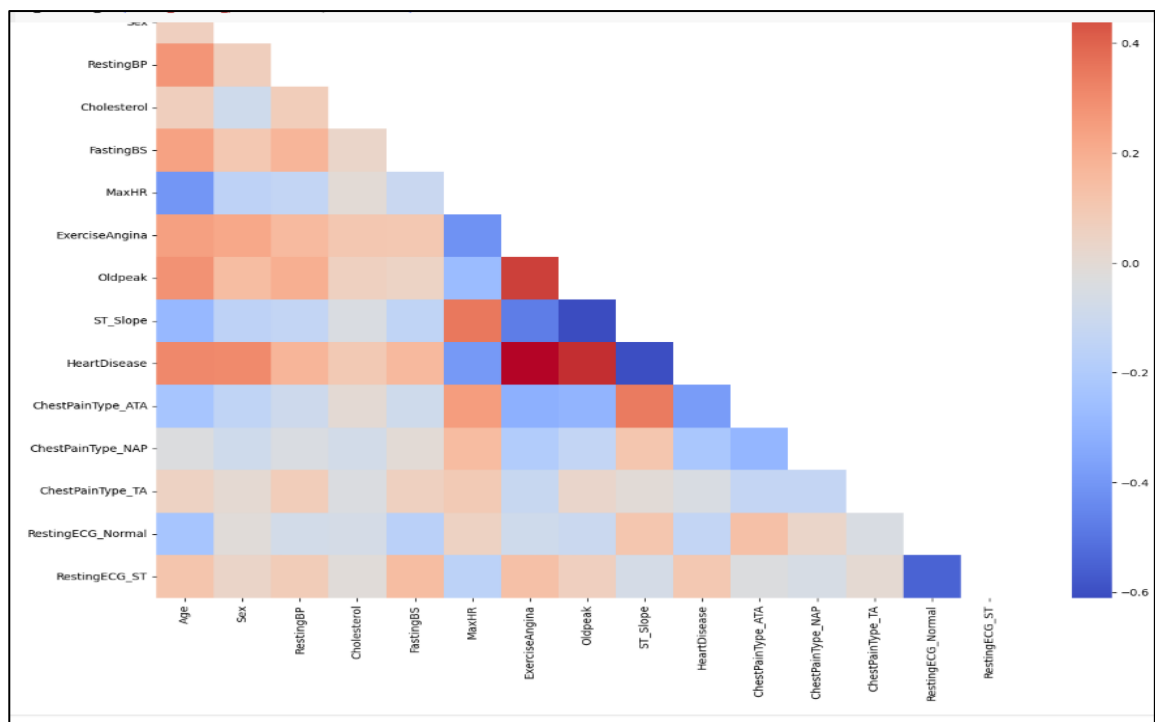


Рисунок 31. Финальный сбор данных

**Вывод:** в ходе выполнения лабораторной работы научился применять методы обработки данных в `pandas.DataFrame`, необходимые для разведочного анализа данных (EDA), включая работу с пропусками, выбросами, масштабирование и кодирование категориальных признаков

### Ответы на контрольные вопросы:

#### 1. Какие типы проблем могут возникнуть из-за пропущенных значений в данных?

Проблемы, возникающие из-за пропущенных значений:

- Искажение результатов: Анализ и выводы могут стать некорректными.
- Уменьшение объема выборки: Удаляя строки с пропуском, уменьшается размер обучающей выборки.
- Неправильная оценка моделей: Модели могут выдавать неверные прогнозы из-за неполных данных.

#### 2. Как с помощью методов `pandas` определить наличие пропущенных значений?

Методы `pandas` для проверки наличия пропущенных значений:

`df.isnull().sum()` или `df.info()`

### **3. Что делает метод `.dropna()` и какие параметры он принимает?**

Метод `dropna` удаляет строки или столбцы, содержащие NaN-значения.

Основные параметры:

`axis`: Определяет ось удаления (0 — строки, 1 — столбцы).

`how`: Критерий удаления строк/столбцов («any» — удаление при любом NaN, «all» — только если все значения NaN).

`thresh`: Минимальное число допустимых ненулевых значений.

`subset`: Столбцы, по которым проверяется наличие NaN.

### **4. Чем различаются подходы заполнения пропусков средним, медианой и модой?**

Среднее значение подвержено влиянию крайних точек (выбросов), медиана устойчивее к ним, а мода выбирается из наиболее часто встречающихся значений. Среднее лучше всего применять для симметричных распределений, медиана — для асимметричных, мода полезна для категориальных переменных.

### **5. Как работает метод `fillna(method='ffill')` и в каких случаях он применим?**

Метод `fillna(method='ffill')` заменяет пропуски значениями предыдущей записи (forward-filling). Применимо в временных рядах или последовательностях, где отсутствие значительных изменений между соседними элементами считается нормальным.

### **6. Какую задачу решает метод `interpolate()` и чем он отличается от `fillna()`?**

Метод `interpolate()` осуществляет интерполяцию пропущенных значений, используя различные алгоритмы (линейная, полиномиальная и др.). Основное отличие от `fillna()` заключается в попытке предсказания пропущенного значения на основе соседних наблюдений. Выявление и обработка выбросов

### **7. Что такое выбросы и почему они могут исказить результаты анализа?**

Выбросы — это наблюдения, значительно отличающиеся от остальных данных. Они приводят к смещению среднего, увеличению дисперсии и ошибкам в оценках корреляций и регрессий.

## **8. В чём суть метода межквартильного размаха (IQR) и как он используется для обнаружения выбросов?**

IQR (Interquartile Range) рассчитывается как разница между третьим квартилем и первым квартилем:

Выбросы определяются следующим образом:

Нижняя граница:  $Q1 - 1.5 * IQR$

Верхняя граница:  $Q3 + 1.5 * IQR$

Любое наблюдение вне указанных границ классифицируется как выброс.

## **9. Как вычислить границы IQR и применить их в фильтрации?**

Пример расчета границ и фильтрации:

```
q1 = df['column'].quantile(0.25)
```

```
q3 = df['column'].quantile(0.75)
```

```
iqr = q3 - q1
```

```
lower_bound = q1 - 1.5 * iqr
```

```
upper_bound = q3 + 1.5 * iqr
```

```
filtered_df = df[(df['column'] >= lower_bound) & (df['column'] <= upper_bound)]
```

## **10. Что делает метод .clip() и как его можно использовать для обработки выбросов?**

Метод .clip(lower, upper) ограничивает значения снизу и сверху указанными пределами. Используется для устранения влияния экстремально больших или малых значений.

## **11. Зачем может потребоваться логарифмическое преобразование числовых признаков?**

Логарифмическое преобразование помогает стабилизировать дисперсию, уменьшить влияние выбросов и сделать распределение ближе к нормальному виду, улучшая производительность многих моделей машинного

обучения.

**12. Какие графические методы позволяют обнаружить выбросы (укажи не менее двух)?**

Для выявления выбросов используют:

Ящик с усами (boxplot): визуализирует распределение данных и выделяет потенциальные выбросы.

Диаграмму рассеяния (scatter plot): позволяет визуально оценить отклонения отдельных точек относительно общей массы данных.

**13. Почему важно быть осторожным при удалении выбросов из обучающих данных?**

Удаление выбросов может привести к потере важной информации и снижению точности модели. Важно учитывать природу данных и понимать причины появления аномалий.

Масштабирование и преобразование признаков

**14. Зачем необходимо масштабирование признаков перед обучением моделей?**

Масштабирование обеспечивает сопоставимость различных признаков друг с другом, улучшает сходство градиентных оптимизационных процессов и повышает точность некоторых моделей (например, SVM, линейной регрессии, kNN).

**15. Чем отличается стандартизация от нормализации?**

Стандартизация нормализует данные путем приведения к нулевому среднему и единичной дисперсии:

$$z = \frac{x - \mu}{\sigma}$$

Нормализация (MinMax Scaler) сжимает диапазон данных к интервалу [0, 1]:

$$x' = \frac{x - x_{\min}}{x_{\max} - x_{\min}}$$

**16. Что делает StandardScaler и как рассчитываются преобразованные значения?**

Класс StandardScaler реализует стандартизацию, приводящую каждое значение признака к следующему виду:

$$z = \frac{x_i - \bar{x}}{\sigma_x}$$

где  $\bar{x}$  — среднее значение,  $\sigma$  — стандартное отклонение.

**17. Как работает MinMaxScaler и когда его использование предпочтительно?**

MinMaxScaler превращает каждый признак в диапазоне [0, 1] по формуле:

$$x' = \frac{x - x_{\min}}{x_{\max} - x_{\min}}$$

Предпочтителен для методов, зависящих от расстояний (kNN, кластеризации), когда важны абсолютные величины признаков.

**18. В чём преимущества RobustScaler при наличии выбросов?**

RobustScaler масштабирует признаки, основываясь на квантилях, что снижает чувствительность к выбросам. Формула:

$$x' = \frac{x - \text{median}(x)}{IQR}$$

где  $\text{median}(x)$  — медиана,  $IQR$  — межквартильный размах.

**19. Как реализовать стандартизацию с помощью .mean() и .std() вручную в pandas?**

Реализация стандартной нормализации вручную:

```
scaled_column = (df['column'] - df['column'].mean()) / df['column'].std()
```

**20. Какие типы моделей наиболее чувствительны к масштабу признаков?**

Модели, сильно зависимые от масштаба входных данных:

Линейная регрессия

Метод ближайших соседей (KNN)

Нейронные сети

Методы опорных векторов (SVM)

Преобразование категориальных признаков

**21. Почему необходимо преобразовывать категориальные признаки перед обучением модели?**

Категоричные признаки нельзя сравнивать количественно (например, цвета, профессии). Для моделирования необходимы преобразования в числовой вид, такие как один-кодовые схемы или факторизованные представления.

**22. Что такое порядковый признак? Приведи пример.**

Порядковые признаки имеют упорядоченность, например, оценки ("низкий", "средний", "высокий"). Эти категории имеют смысл порядка, хотя расстояние между ними может быть неоднородным.

**23. Что такое номинальный признак? Приведи пример.**

Номинальный признак представляет собой категорию без внутреннего порядка, например, страна проживания или название продукта.

**24. Как работает метод `.factorize()` и для каких случаев он подходит?**

`.factorize()` присваивает каждой уникальной категории уникальное целое число. Подходит для быстрого преобразования небольшого количества уникальных категорий, особенно для простых задач классификации.

**25. Как применить метод `.map()` для кодирования категориальных признаков с известным порядком?**

Использование `.map()` для преобразования признаков с заранее заданным порядком:

```
mapping_dict = {'low': 1, 'medium': 2, 'high': 3}
df['column'] = df['column'].map(mapping_dict)
```

## **26. Что делает класс `OrdinalEncoder` из `scikit-learn`?**

`OrdinalEncoder` преобразует категориальные признаки в целочисленные, сохраняя порядок классов. Полезен, когда существует естественный порядок среди категорий.

## **27. В чём суть `one-hot` кодирования и когда оно применяется?**

`One-hot` кодирование создает бинарные признаки для каждого уникального уровня категориальной переменной. Применяется, когда категориальные признаки являются номинальными и отсутствует очевидный порядок.

## **28. Как избежать дамми-ловушки при `one-hot` кодировании?**

Дамми-ловушка возникает при полной мультиколлинеарности в матрице дизайна. Избегается путём исключения одной колонки (`dummy variable trap`).

## **29. Как работает `OneHotEncoder` из `scikit-learn` и чем он отличается от `pd.get_dummies()`?**

`OneHotEncoder` из `scikit-learn` преобразует категориальные признаки в `one-hot` кодированные массивы, поддерживая скалярность объектов. Отличия от `pd.get_dummies()` включают возможность задания параметров (например, `drop_first=True`) и совместимость с конвейерами `sklearn`.

## **30. В чём суть метода `target encoding` и какие риски он в себя несет?**

`Target encoding` заменяет каждую категорию средней целевой переменной (`target`). Риски связаны с переобучением, поскольку информация о целевой переменной может повлиять на процесс обучения, вызывая утечку данных.