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 6

Выполнил:
Якушенко Антон Андреевич
2 курс, группа ИТС-б-о-23-1,
11.03.02 «Инфокоммуникационные технологии и системы связи»,
направленность (профиль) «Инфокоммуникационные системы и сети», очная форма обучения

_____
(подпись)

Проверил:
Ассистент департамента цифровых, робототехнических систем и электроники Хацукова А.И

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2025 г.

ТЕМА: 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 репозиторий: https://github.com/Yakush766/LB6.git

Порядок выполнения работы:

1. Обнаружение и обработка пропущенных значений.

```python
import pandas as pd
import missingno as msno

# 1. Загрузка датасета titanic
titanic = pd.read_csv('titanic.csv')  # Укажите путь к файлу с датасетом

# 5. Отобразим информацию о таблице до обработки
print("Информация до обработки:")
print(titanic.info())
print("Количество пропущенных значений до обработки:")
print(titanic.isna().sum())

# 2. Определим количество пропущенных значений в каждом столбце
missing_counts = titanic.isna().sum()
print("\nКоличество пропущенных значений по столбцам:")
print(missing_counts)

# 3. Визуализируем пропуски с помощью библиотеки missingno
msno.matrix(titanic)

# 4. Заполнение пропущенных значений
# - признак age — средним значением
titanic['Age'].fillna(titanic['Age'].mean(), inplace=True)

# - признак embarked — наиболее частым значением
most_common_embarked = titanic['Embarked'].mode()[0]
titanic['Embarked'].fillna(most_common_embarked, inplace=True)

# - признак deck — удалить
# Предположим, что deck - это первый символ из столбца Cabin
# Добавим столбец deck, если нет, иначе удалим его
if 'Deck' in titanic.columns:
    titanic.drop(columns=['Deck'], inplace=True)
else:
    # Создаем столбец deck из Cabin (первый символ)
    titanic['Deck'] = titanic['Cabin'].str[0]
    # Удаляем столбец deck
    titanic.drop(columns=['Deck'], inplace=True)

# 5. Отобразим информацию о таблице после обработки
print("\nИнформация после обработки:")
print(titanic.info())
print("Количество пропущенных значений после обработки:")
print(titanic.isna().sum())
```

Рисунок 1. Код для выполнения программы

```
Информация до обработки:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
None
Количество пропущенных значений до обработки:
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64

Количество пропущенных значений по столбцам:
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64

Информация после обработки:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          891 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     891 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
None
Количество пропущенных значений после обработки:
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age              0
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         0
dtype: int64
```

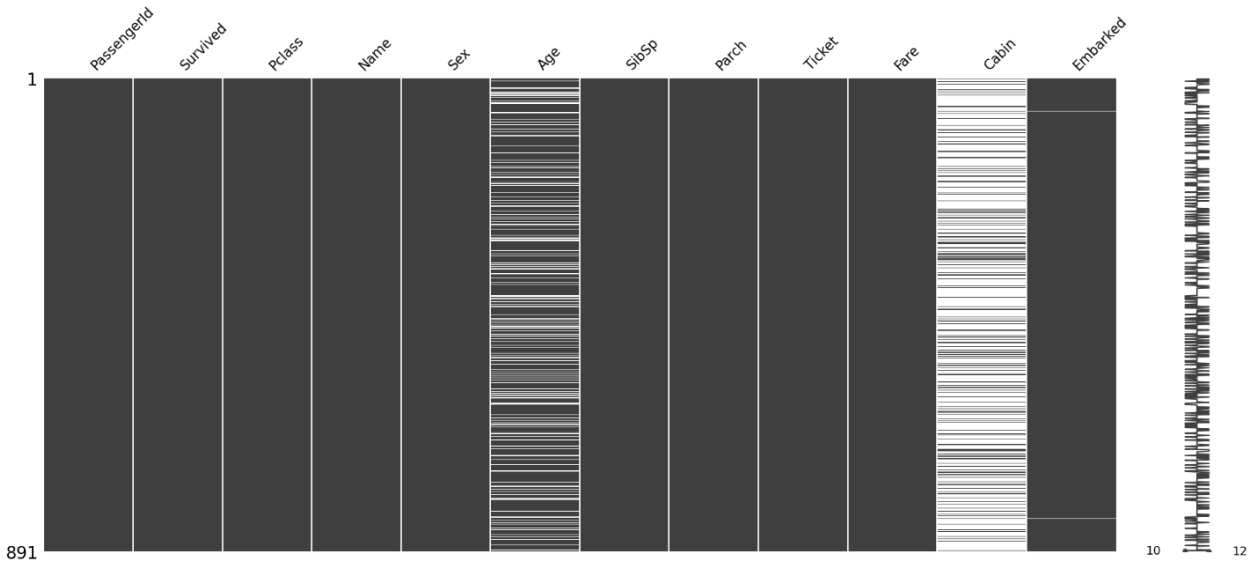


Рисунок 2. Результат

2. Обнаружение и удаление выбросов.

```python
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# 1. Загрузка датасета penguins
penguins = pd.read_csv('penguins.csv')  # Укажите путь к файлу с датасетом

# 4. Размеры датасета до фильтрации
print("Размер датасета до фильтрации:", penguins.shape)

# 2. Построение boxplot-графиков для указанных признаков
features = ['bill_length_mm', 'bill_depth_mm', 'flipper_length_mm', 'body_mass_g']
for feature in features:
    plt.figure(figsize=(6, 4))
    sns.boxplot(x=penguins[feature])
    plt.title(f'Boxplot для {feature}')
    plt.show()

# 3. Выявление и удаление выбросов с помощью IQR для каждого из признаков
def remove_outliers_iqr(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)]

penguins_filtered = penguins.copy()
for feature in features:
    penguins_filtered = remove_outliers_iqr(penguins_filtered, feature)

# 4. Размеры датасета после фильтрации
print("Размер датасета после фильтрации:", penguins_filtered.shape)

# 5. Построение boxplot до и после удаления выбросов для одного признака, например, 'body_mass_g'
plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)
sns.boxplot(x=penguins['body_mass_g'])
plt.title('До удаления выбросов (body_mass_g)')

plt.subplot(1, 2, 2)
sns.boxplot(x=penguins_filtered['body_mass_g'])
plt.title('После удаления выбросов (body_mass_g)')

plt.tight_layout()
plt.show()
```

```
Размер датасета до фильтрации: (344, 9)
```

Рисунок 3. Код для выполнения программы

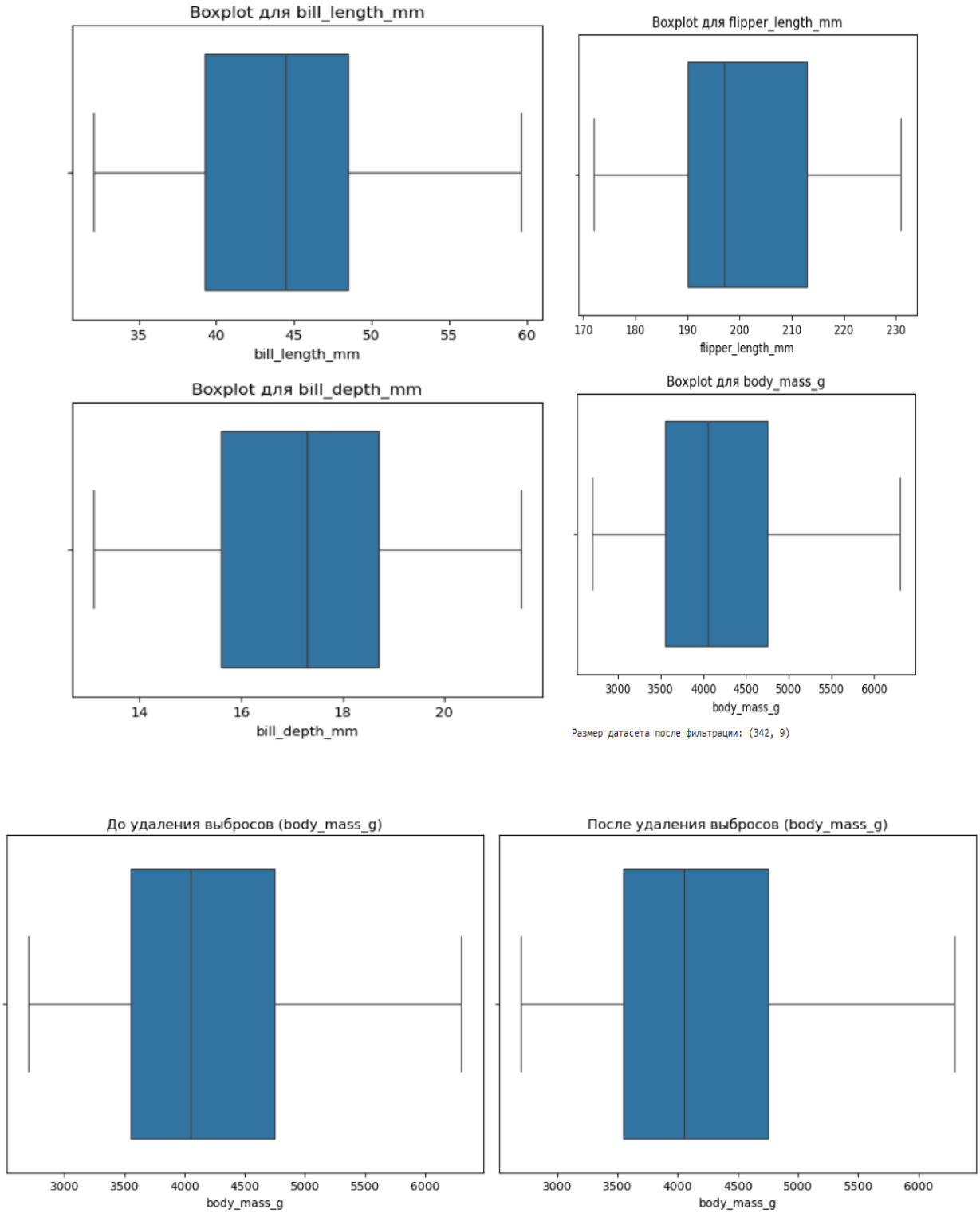


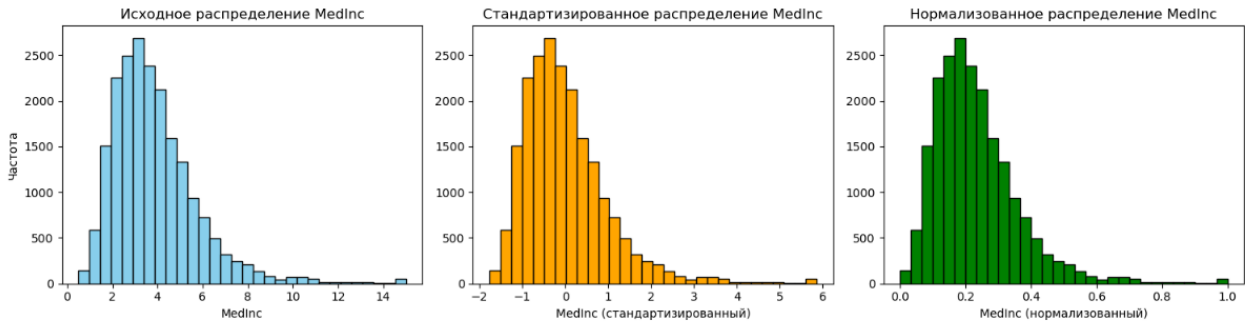
Рисунок 4. Выполненная программа

3. Масштабирование числовых признаков.

```python
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import fetch_california_housing
from sklearn.preprocessing import StandardScaler, MinMaxScaler

# 1. Загрузка данных с помощью fetch_california_housing
data = fetch_california_housing(as_frame=True)
df = data.frame  # 2. Преобразование в pandas.DataFrame (уже сделано)

# Выведем первые строки для ознакомления
print(df.head())

# 3. Стандартизация и нормализация признаков
scaler_standard = StandardScaler()
scaler_minmax = MinMaxScaler()

# Копия таблицы для нормализации
df_minmax = df.copy()

# Стандартизация всех признаков
df_standard = pd.DataFrame(scaler_standard.fit_transform(df), columns=df.columns)

# Нормализация всех признаков (на копии)
df_minmax[df.columns] = scaler_minmax.fit_transform(df_minmax[df.columns])

# 4. Построение гистограмм для признака MedInc (median income) до и после масштабирования
plt.figure(figsize=(15, 4))

plt.subplot(1, 3, 1)
plt.hist(df['MedInc'], bins=30, color='skyblue', edgecolor='black')
plt.title('Исходное распределение MedInc')
plt.xlabel('MedInc')
plt.ylabel('Частота')

plt.subplot(1, 3, 2)
plt.hist(df_standard['MedInc'], bins=30, color='orange', edgecolor='black')
plt.title('Стандартизированное распределение MedInc')
plt.xlabel('MedInc (стандартизированный)')

plt.subplot(1, 3, 3)
plt.hist(df_minmax['MedInc'], bins=30, color='green', edgecolor='black')
plt.title('Нормализованное распределение MedInc')
plt.xlabel('MedInc (нормализованный)')

plt.tight_layout()
plt.show()

# 5. Сравнение поведения шкал на гистограммах:
print("""
- Исходное распределение показывает реальные значения признака MedInc.
- Стандартизация (StandardScaler) приводит данные к среднему 0 и стандартному отклонению 1,
  поэтому гистограмма центрирована около 0 с симметричным распределением.
- Нормализация (MinMaxScaler) масштабирует значения в диапазон [0, 1],
  поэтому гистограмма сдвинута и сжата в этом интервале.
""")
```

Рисунок 5. Код для выполнения программы

```
   MedInc  HouseAge  AveRooms  AveBedrms  Population  AveOccup  Latitude  \
0  8.3252      41.0  6.984127   1.023810       322.0  2.555556     37.88
1  8.3014      21.0  6.238137   0.971880      2401.0  2.109842     37.86
2  7.2574      52.0  8.288136   1.073446       496.0  2.802260     37.85
3  5.6431      52.0  5.817352   1.073059       558.0  2.547945     37.85
4  3.8462      52.0  6.281853   1.081081       565.0  2.181467     37.85

   Longitude  MedHouseVal
0    -122.23        4.526
1    -122.22        3.585
2    -122.24        3.521
3    -122.25        3.413
4    -122.25        3.422
```




```
- Исходное распределение показывает реальные значения признака MedInc.
- Стандартизация (StandardScaler) приводит данные к среднему 0 и стандартному отклонению 1,
  поэтому гистограмма центрирована около 0 с симметричным распределением.
- Нормализация (MinMaxScaler) масштабирует значения в диапазон [0, 1],
  поэтому гистограмма сдвинута и сжата в этом интервале.
```

Рисунок 6. Выполненная программа

4. Кодирование категориальных признаков.

```python
import pandas as pd
from sklearn.datasets import fetch_openml
from sklearn.preprocessing import LabelEncoder

# Загружаем датасет adult через sklearn
adult = fetch_openml("adult", version=2, as_frame=True)
df = adult.frame  # Получаем DataFrame напрямую

print("Все столбцы датасета:")
print(df.columns.tolist(), end="\n\n")

# В датасете целевой признак называется 'class', а не 'income'
# Отбираем нужные признаки с правильным именем целевого признака
df_selected = df[['education', 'marital-status', 'occupation', 'class']]

print("Исходные данные:")
print(df_selected.head(), end="\n\n")

# 2. Label Encoding для 'education'
le_education = LabelEncoder()
df_selected['education_encoded'] = le_education.fit_transform(df_selected['education'])

print("После Label Encoding признака 'education':")
print(df_selected[['education', 'education_encoded']], end="\n\n")

# 3. One-Hot Encoding для 'marital-status' и 'occupation'
df_encoded = pd.get_dummies(df_selected, columns=['marital-status', 'occupation'], drop_first=False)

print("После One-Hot Encoding признаков 'marital-status' и 'occupation':")
print(df_encoded.head(), end="\n\n")

# 4. Проверка размерности таблицы до и после кодирования
print("Размерность до кодирования:", df_selected.shape)
print("Размерность после кодирования:", df_encoded.shape, end="\n\n")

# 5. Проверка дамми-ловушки
marital_cols = [col for col in df_encoded.columns if col.startswith('marital-status_')]
print("Корреляция между one-hot признаками 'marital-status':")
print(df_encoded[marital_cols].corr(), end="\n\n")

print("Если матрица корреляции содержит 1 между всеми столбцами, значит дамми-ловушка присутствует.")
print("Чтобы избежать дамми-ловушки, можно использовать drop_first=True в pd.get_dummies.")
```

Рисунок 7. Код для выполнения задания

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

Исходные данные:
      education       marital-status         occupation  class
0          11th        Never-married  Machine-op-inspct  <=50K
1       HS-grad  Married-civ-spouse     Farming-fishing  <=50K
2     Assoc-acdm  Married-civ-spouse     Protective-serv   >50K
3   Some-college  Married-civ-spouse  Machine-op-inspct   >50K
4   Some-college        Never-married                NaN  <=50K

После Label Encoding признака 'education':
           education  education_encoded
0               11th                 1
1            HS-grad                11
2          Assoc-acdm                 7
3        Some-college                15
4        Some-college                15
...               ...               ...
48837     Assoc-acdm                 7
48838        HS-grad                11
48839        HS-grad                11
48840        HS-grad                11
48841        HS-grad                11

[48842 rows x 2 columns]

После One-Hot Encoding признаков 'marital-status' и 'occupation':
      education  class  education_encoded  marital-status_Divorced  \
0          11th  <=50K                  1                    False
1       HS-grad  <=50K                 11                    False
2     Assoc-acdm   >50K                  7                    False
3   Some-college   >50K                 15                    False
4   Some-college  <=50K                 15                    False

   marital-status_Married-AF-spouse  marital-status_Married-civ-spouse  \
0                             False                              False
1                             False                               True
2                             False                               True
3                             False                               True
4                             False                              False

   marital-status_Married-spouse-absent  marital-status_Never-married  \
0                                 False                         True
1                                 False                        False
2                                 False                        False
3                                 False                        False
4                                 False                         True

   marital-status_Separated  marital-status_Widowed  ...  \
0                     False                   False  ...
1                     False                   False  ...
2                     False                   False  ...
3                     False                   False  ...
4                     False                   False  ...
```

Рисунок 8. Выполненная программа

5. Комплексный EDA.

```python
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler, LabelEncoder

# Загрузка данных
df = pd.read_csv('heart.csv')

# Приводим названия столбцов к нижнему регистру и убираем пробелы
df.columns = df.columns.str.strip().str.lower()

# Списки признаков по типу (с учётом приведённых имён столбцов)
num_cols = ['age', 'cholesterol', 'restingbp', 'maxhr', 'oldpeak']
cat_cols = ['sex', 'chestpaintype', 'exerciseangina', 'restingecg', 'st_slope']

# Обработка пропущенных значений
for col in num_cols:
    if col in df.columns:
        if df[col].isnull().sum() > 0:
            median_val = df[col].median()
            df[col].fillna(median_val, inplace=True)
    else:
        print(f"Внимание: числовой столбец '{col}' отсутствует в данных!")

for col in cat_cols:
    if col in df.columns:
        if df[col].isnull().sum() > 0:
            mode_val = df[col].mode()[0]
            df[col].fillna(mode_val, inplace=True)
    else:
        print(f"Внимание: категориальный столбец '{col}' отсутствует в данных!")

# Удаление выбросов по методу IQR
def remove_outliers_iqr(data, column):
    Q1 = data[column].quantile(0.25)
    Q3 = data[column].quantile(0.75)
    IQR = Q3 - Q1
    lower = Q1 - 1.5 * IQR
    upper = Q3 + 1.5 * IQR
    print(f"{column}: удаляется {(data[column] < lower).sum() + (data[column] > upper).sum()} выбросов")
    return data[(data[column] >= lower) & (data[column] <= upper)]

for col in ['age', 'cholesterol', 'restingbp', 'maxhr']:
    if col in df.columns:
        df = remove_outliers_iqr(df, col)
    else:
        print(f"Внимание: столбец '{col}' отсутствует, пропускаем удаление выбросов по нему.")

# Масштабирование числовых признаков
scaler = StandardScaler()
df[num_cols] = scaler.fit_transform(df[num_cols])

# Кодирование категориальных признаков
# Кодируем sex (предполагается бинарный)
if 'sex' in df.columns:
    le = LabelEncoder()
    df['sex'] = le.fit_transform(df['sex'])
else:
    print("Внимание: столбец 'sex' отсутствует, кодирование пропущено.")

# One-hot кодирование для остальных категориальных признаков
to_onehot = [col for col in cat_cols if col != 'sex' and col in df.columns]
df = pd.get_dummies(df, columns=to_onehot, drop_first=True)

print("\nОбработка завершена. Итоговые данные:")
print(df.info())
print(df.head())
```

Рисунок 9. Код для выполнения задания

```
age: удаляется 0 выбросов
cholesterol: удаляется 183 выбросов
restingbp: удаляется 20 выбросов
maxhr: удаляется 0 выбросов

Обработка завершена. Итоговые данные:
<class 'pandas.core.frame.DataFrame'>
Index: 715 entries, 0 to 917
Data columns (total 16 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   age                 715 non-null    float64
 1   sex                 715 non-null    int32
 2   restingbp           715 non-null    float64
 3   cholesterol         715 non-null    float64
 4   fastingbs           715 non-null    int64
 5   maxhr               715 non-null    float64
 6   oldpeak             715 non-null    float64
 7   heartdisease        715 non-null    int64
 8   chestpaintype_ATA   715 non-null    bool
 9   chestpaintype_NAP   715 non-null    bool
 10  chestpaintype_TA    715 non-null    bool
 11  exerciseangina_Y    715 non-null    bool
 12  restingecg_Normal   715 non-null    bool
 13  restingecg_ST       715 non-null    bool
 14  st_slope_Flat       715 non-null    bool
 15  st_slope_Up         715 non-null    bool
dtypes: bool(8), float64(5), int32(1), int64(2)
memory usage: 53.1 KB
None
        age  sex  restingbp  cholesterol  fastingbs     maxhr   oldpeak  \
0 -1.343776    1   0.539269     0.962124          0  1.296933 -0.839138
1 -0.400479    0   1.836853    -1.180312          0  0.640055  0.097605
2 -1.658208    1  -0.109523     0.844191          0 -1.741130 -0.839138
3 -0.505290    0   0.409510    -0.512029          0 -1.330581  0.565976
4  0.123574    1   1.188061    -0.885481          0 -0.755812 -0.839138

   heartdisease  chestpaintype_ATA  chestpaintype_NAP  chestpaintype_TA  \
0             0               True              False             False
1             1              False               True             False
2             0               True              False             False
3             1              False              False             False
4             0              False               True             False

   exerciseangina_Y  restingecg_Normal  restingecg_ST  st_slope_Flat  \
0             False               True          False          False
1             False               True          False           True
2             False              False           True          False
3              True               True          False           True
4             False               True          False          False

   st_slope_Up
0         True
1        False
2         True
3        False
4         True
```

Рисунок 10. Выполненная программа

Индивидуальное задание:

```python
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler, LabelEncoder

# === 1. Обзор структуры данных ===
df = pd.read_csv("test.csv")  # Загрузка датасета
print(df.info())              # Общая информация
print(df.describe())          # Статистика по числовым признакам

# === 2. Обнаружение и обработка пропусков ===
print(df.isnull().sum())      # Проверка на пропуски
df.fillna(df.median(numeric_only=True), inplace=True)  # Заполнение медианой (если бы были)

# === 3. Обнаружение и удаление выбросов ===
selected_features = ['battery_power', 'int_memory', 'px_height', 'px_width', 'ram']
for col in selected_features:
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    lower = Q1 - 1.5 * IQR
    upper = Q3 + 1.5 * IQR
    df = df[(df[col] >= lower) & (df[col] <= upper)]  # Удаление выбросов

# === 4. Масштабирование числовых признаков ===
scaler = StandardScaler()
numeric_features = df.select_dtypes(include=np.number).drop(columns=['id']).columns
df[numeric_features] = scaler.fit_transform(df[numeric_features])

# === 5. Кодирование категориальных признаков ===
# В данном случае категориальных признаков нет, но на случай их появления:
categorical_features = df.select_dtypes(include='object').columns
label_encoders = {}
for col in categorical_features:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    label_encoders[col] = le

# === 6. Финальный набор данных ===
print(df.info())        # Проверка после всех преобразований
print(df.shape)         # Размерность итогового набора
```

Рисунок 11. Полный код для выполнения каждого пункта задания

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 21 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   id              1000 non-null   int64  
 1   battery_power   1000 non-null   int64  
 2   blue            1000 non-null   int64  
 3   clock_speed     1000 non-null   float64
 4   dual_sim        1000 non-null   int64  
 5   fc              1000 non-null   int64  
 6   four_g          1000 non-null   int64  
 7   int_memory      1000 non-null   int64  
 8   m_dep           1000 non-null   float64
 9   mobile_wt       1000 non-null   int64  
 10  n_cores         1000 non-null   int64  
 11  pc              1000 non-null   int64  
 12  px_height       1000 non-null   int64  
 13  px_width        1000 non-null   int64  
 14  ram             1000 non-null   int64  
 15  sc_h            1000 non-null   int64  
 16  sc_w            1000 non-null   int64  
 17  talk_time       1000 non-null   int64  
 18  three_g         1000 non-null   int64  
 19  touch_screen    1000 non-null   int64  
 20  wifi            1000 non-null   int64  
dtypes: float64(2), int64(19)
memory usage: 164.2 KB
None
```

	id	battery_power	blue	clock_speed	dual_sim	...
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	
mean	500.500000	1248.510000	0.516000	1.540900	0.517000	
std	288.819436	432.458227	0.499994	0.829268	0.499961	
min	1.000000	500.000000	0.000000	0.500000	0.000000	
25%	250.750000	895.000000	0.000000	0.700000	0.000000	
50%	500.500000	1246.500000	1.000000	1.500000	1.000000	
75%	750.250000	1629.250000	1.000000	2.300000	1.000000	
max	1000.000000	1999.000000	1.000000	3.000000	1.000000	

	fc	four_g	int_memory	m_dep	mobile_wt	...
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.00000	...
mean	4.593000	0.487000	33.652000	0.517500	139.51100	...
std	4.463325	0.500081	18.128694	0.280861	34.85155	...
min	0.000000	0.000000	2.000000	0.100000	80.00000	...
25%	1.000000	0.000000	18.000000	0.300000	109.75000	...
50%	3.000000	0.000000	34.500000	0.500000	139.00000	...
75%	7.000000	1.000000	49.000000	0.800000	170.00000	...
max	19.000000	1.000000	64.000000	1.000000	200.00000	...

	pc	px_height	px_width	ram	sc_h
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000
mean	10.054000	627.121000	1239.774000	2138.998000	11.995000
std	6.095099	432.929699	439.670981	1088.092278	4.320607
min	0.000000	0.000000	501.000000	263.000000	5.000000
25%	5.000000	263.750000	831.750000	1237.250000	8.000000
50%	10.000000	564.500000	1250.000000	2153.500000	12.000000
75%	16.000000	903.000000	1637.750000	3065.500000	16.000000
max	20.000000	1907.000000	1998.000000	3989.000000	19.000000

	sc_w	talk_time	three_g	touch_screen	wifi
count	1000.000000	1000.000000	1000.000000	1000.00000	1000.000000
mean	5.316000	11.085000	0.756000	0.50000	0.507000
std	4.240062	5.497636	0.429708	0.50025	0.500201
min	0.000000	2.000000	0.000000	0.00000	0.000000
25%	2.000000	6.750000	1.000000	0.00000	0.000000
50%	5.000000	11.000000	1.000000	0.50000	1.000000
75%	8.000000	16.000000	1.000000	1.00000	1.000000
max	18.000000	20.000000	1.000000	1.00000	1.000000

```
[8 rows x 21 columns]
id              0
battery_power   0
blue            0
clock_speed     0
dual_sim        0
fc              0
four_g          0
int_memory      0
m_dep           0
mobile_wt       0
n_cores         0
pc              0
px_height       0
px_width        0
ram             0
sc_h            0
sc_w            0
talk_time       0
three_g         0
touch_screen    0
wifi            0
dtype: int64
<class 'pandas.core.frame.DataFrame'>
Index: 998 entries, 0 to 999
Data columns (total 21 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   id              998 non-null    int64  
 1   battery_power   998 non-null    float64
 2   blue            998 non-null    float64
 3   clock_speed     998 non-null    float64
 4   dual_sim        998 non-null    float64
 5   fc              998 non-null    float64
 6   four_g          998 non-null    float64
 7   int_memory      998 non-null    float64
 8   m_dep           998 non-null    float64
 9   mobile_wt       998 non-null    float64
 10  n_cores         998 non-null    float64
 11  pc              998 non-null    float64
 12  px_height       998 non-null    float64
 13  px_width        998 non-null    float64
 14  ram             998 non-null    float64
 15  sc_h            998 non-null    float64
 16  sc_w            998 non-null    float64
 17  talk_time       998 non-null    float64
 18  three_g         998 non-null    float64
 19  touch_screen    998 non-null    float64
 20  wifi            998 non-null    float64
dtypes: float64(20), int64(1)
memory usage: 171.5 KB
None
(998, 21)
```

Рисунок 12. Выполненная программа

Ответы на контрольные вопросы:

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

Снижение качества модели, искажение статистик, невозможность обучения моделей, ошибки при визуализации.

2. Как с помощью метода .isna() определить наличие пропущенных значений?

.isna() возвращает True для всех пропущенных значений; можно использовать .sum() для подсчета.

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

Удаляет строки/столбцы с пропущенными значениями. Параметры: axis, how, thresh, subset.

4. Чем различаются способы заполнения пропусков средним, медианой и модой?

Среднее чувствительно к выбросам, медиана устойчива, мода — для категориальных данных.

5. Как работает метод fillna(method='ffill') и в каких случаях он применяется?

Копирует предыдущее значение в ячейки с NaN — используется при временных рядах.

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

Интерполяция оценивает промежуточные значения; fillna просто подставляет существующее.

7. Что такое выбросы и почему они могут искажать результаты анализа?

Аномальные значения, сильно отличающиеся от других. Искажают средние, влияют на модели.

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

IQR = Q3 - Q1; выбросы — значения ниже Q1 - 1.5*IQR или выше Q3 + 1.5*IQR.

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

Q1 = df[col].quantile(0.25), Q3 = df[col].quantile(0.75); фильтрация по диапазону [Q1 - 1.5*IQR, Q3 + 1.5*IQR].

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

Ограничивает значения в колонке указанным интервалом (например, по IQR).

11. Зачем может потребоваться логарифмическое преобразование числовых признаков?

Для устранения смещения распределения, уменьшения влияния выбросов.

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

Boxplot, гистограммы, scatter plot, z-оценка.

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

Можно потерять важные данные или уменьшить разнообразие обучающей выборки.

14. Зачем необходимо масштабирование признаков перед обучением моделей?

Многие алгоритмы чувствительны к масштабу признаков — без масштабирования они работают некорректно.

15. Чем отличается стандартизация от нормализации?

Стандартизация: приведение к среднему 0 и отклонению 1. Нормализация: масштабирование в диапазон [0,1].

16. Что делает StandardScaler и как рассчитываются преобразованные значения?

Вычисляет (x - mean) / std — стандартное масштабирование.

17. Как работает MinMaxScaler и когда его использование предпочтительно?

Масштабирует в диапазон [0,1]. Подходит, если нет выбросов.

18. В чём преимущество RobustScaler при наличии выбросов?

Использует медиану и IQR, устойчив к выбросам.

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

(df - df.mean()) / df.std()

20. Почему необходимо преобразовывать категориальные признаки перед обучением моделей?

Модели не работают с текстами — необходимо числовое представление.

21. Что такое порядковый признак? Приведите пример.

Категория с порядком (например, «низкий», «средний», «высокий»).

22. Что такое номинальный признак? Приведите пример.

Категория без порядка (например, «красный», «синий», «зелёный»).

23. Как работает метод .factorize() и для каких случаев он подходит?

Кодирует уникальные категории числами. Подходит для порядковых категорий.

24. Как применять OneHotEncoding для кодирования категориальных признаков с помощью pandas?

С помощью pd.get_dummies(data, columns=['категория']).

25. Чем отличаются OneHotEncoder и OrdinalEncoder из scikit-learn?

OneHot — для номинальных признаков, Ordinal — для порядковых.

26. Что такое дата-фрейм после one-hot кодирования?

Таблица с бинарными колонками для каждой категории.

27. Как влияет увеличение размерности при one-hot кодировании?

Может привести к разреженной матрице и замедлению обучения моделей.

28. В чём суть one-hot кодирования и когда оно применяется? One-hot кодирование — это способ представления категориальных признаков

в виде бинарных векторов. Каждая уникальная категория становится отдельным столбцом, где для строки с этой категорией значение — 1, для остальных—0.

Применяется: Когда категории не имеют порядка (номинальные признаки).Перед обучением моделей, которые требуют числового ввода (например, линейные модели, деревья, нейросети и др.).

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

OneHotEncoder — часть scikit-learn, работает с массивами и пайплайнами. Требует предварительного преобразования категорий в числовой формат или используется с ColumnTransformer. Поддерживает сохранение структуры при трансформации новых данных. pd.get_dummies() — функция pandas, быстро и удобно создаёт дамми-переменные, но не подходит для применения на новых данных без повторного вызова.

30. В чём суть метода target encoding и какие риски он в себе несёт?

Target encoding (или mean encoding) — это замена категориального признака на среднее значение целевой переменной для каждой категории. Пример: если в категории A средний target = 0.8, а в B = 0.2, то A → 0.8, B → 0.2.

Риски:

- Может привести к **утечке данных (data leakage)** при обучении модели, особенно если целевые значения из теста попали в расчёт.
- Может **переобучиться** на редкие категории.

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