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 3

Выполнил:
Борцов Богдан Михайлович
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:
Доцент департамента цифровых, робототехнических систем и электроники Воронкин Р.А.

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2025 г.

Тема: Основные этапы исследовательского анализа данных

Цель: 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/REPONCFU/ai-jlab6

Порядок выполнения работы:

1. Задание 1:

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

# Загрузка датасета
df = sns.load_dataset("titanic")

# Определение количества пропущенных значений
print("Количество пропущенных значений до обработки:")
print(df.isna().sum())

# Визуализация пропусков
msno.matrix(df)
plt.show()

# Заполнение пропусков
df['age'] = df['age'].fillna(df['age'].mean())  # Среднее значение
df['embarked'] = df['embarked'].fillna(df['embarked'].mode()[0])  # Наиболее частое значение
df = df.drop(columns=['deck'])  # Удаление столбца

# Проверка после обработки
print("\nКоличество пропущенных значений после обработки:")
print(df.isna().sum())

# Общая информация
print("\nИнформация о таблице после обработки:")
print(df.info())
```

Рисунок 1. Листинг программы задание 1

2. Задание 2:

```python
# Загрузка датасета
df = sns.load_dataset("penguins")

# Построение boxplot для указанных признаков
numeric_cols = ['bill_length_mm', 'bill_depth_mm', 'flipper_length_mm', 'body_mass_g']
for col in numeric_cols:
    plt.figure(figsize=(6, 4))
    sns.boxplot(x=df[col])
    plt.title(f"Boxplot для {col}")
    plt.show()

# Удаление выбросов с использованием IQR
def remove_outliers(df, column):
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower = Q1 - 1.5 * IQR
    upper = Q3 + 1.5 * IQR
    return df[(df[column] >= lower) & (df[column] <= upper)]

original_size = df.shape[0]
for col in numeric_cols:
    df = remove_outliers(df, col)
new_size = df.shape[0]

# Сравнение размеров датасета
print(f"Размер датасета до удаления выбросов: {original_size}")
print(f"Размер датасета после удаления выбросов: {new_size}")

# Boxplot после удаления выбросов
plt.figure(figsize=(6, 4))
sns.boxplot(x=df['bill_length_mm'])
plt.title("Boxplot для bill_length_mm после удаления выбросов")
plt.show()
```

Рисунок 2. Листинг программы задание 2

3. Задание 3:

```python
# Загрузка данных
data = fetch_california_housing(as_frame=True)
df = data.frame

# Стандартизация
scaler_standard = StandardScaler()
df_standardized = df.copy()
df_standardized[df.columns] = scaler_standard.fit_transform(df)

# Нормализация
scaler_minmax = MinMaxScaler()
df_normalized = df.copy()
df_normalized[df.columns] = scaler_minmax.fit_transform(df)

# Гистограммы до и после масштабирования
plt.figure(figsize=(12, 5))
plt.subplot(1, 3, 1)
plt.hist(df['MedInc'], bins=20, color='blue', alpha=0.7)
plt.title("До масштабирования")

plt.subplot(1, 3, 2)
plt.hist(df_standardized['MedInc'], bins=20, color='green', alpha=0.7)
plt.title("После StandardScaler")

plt.subplot(1, 3, 3)
plt.hist(df_normalized['MedInc'], bins=20, color='red', alpha=0.7)
plt.title("После MinMaxScaler")

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

Рисунок 3. Листинг программы задание 3

4. Задание 4:

```python
import pandas as pd
from sklearn.preprocessing import LabelEncoder, OneHotEncoder

# Загрузка данных

columns = ['age', 'workclass', 'fnlwgt', 'education', 'education-num', 'marital-status',
           'occupation', 'relationship', 'race', 'sex', 'capital-gain', 'capital-loss',
           'hours-per-week', 'native-country', 'income']

df = pd.read_csv(file_path, header=None, names=columns, na_values=' ?', skipinitialspace=True)

# Выбор нужных признаков
categorical_features = ['education', 'marital-status', 'occupation']
target_feature = 'income'

# Просмотр информации о данных перед обработкой
print("Информация о данных перед обработкой:")
print(df[categorical_features + [target_feature]].info())
print("\nПервые 5 строк данных:")
print(df[categorical_features + [target_feature]].head())

# 1. Label Encoding для признака education (предполагаем порядок)
# Создаем порядок уровней образования (от низшего к высшему)
education_order = [
    'Preschool', '1st-4th', '5th-6th', '7th-8th', '9th', '10th', '11th', '12th',
    'HS-grad', 'Some-college', 'Assoc-voc', 'Assoc-acdm', 'Bachelors', 'Masters',
    'Prof-school', 'Doctorate'
]

# Создаем словарь для соответствия
education_mapping = {v: i for i, v in enumerate(education_order)}

# Применяем Label Encoding
df['education_encoded'] = df['education'].map(education_mapping)

# Проверяем результат
print("\nРезультат Label Encoding для education:")
print(df[['education', 'education_encoded']].head(10))

# 2. One-Hot Encoding для marital-status и occupation
# Сначала проверим наличие пропущенных значений
print("\nКоличество пропущенных значений:")
print(df[['marital-status', 'occupation']].isna().sum())

# Заполним пропуски в occupation модой
df['occupation'].fillna(df['occupation'].mode()[0], inplace=True)

# Применяем One-Hot Encoding с исключением одного столбца (избегаем дамми-ловушку)
df_encoded = pd.get_dummies(df, columns=['marital-status', 'occupation'], drop_first=True)

# Проверяем результат
print("\nСтолбцы после One-Hot Encoding:")
print(df_encoded.filter(regex='marital-status|occupation').columns)

# Проверяем итоговую размерность таблицы
print("\nРазмерность таблицы до кодирования:", df.shape)
print("Размерность таблицы после кодирования:", df_encoded.shape)

# Проверяем, что нет дамми-ловушки (один столбец удален для каждой категории)
print("\nПроверка на дамми-ловушку:")
print("Уникальные значения marital-status:", df['marital-status'].nunique())
print("Количество столбцов после кодирования:",
      len(df_encoded.filter(regex='marital-status').columns))
print("Уникальные значения occupation:", df['occupation'].nunique())
print("Количество столбцов после кодирования:",
      len(df_encoded.filter(regex='occupation').columns))

# Сохраняем обработанные данные
df_encoded.to_csv(output_path, index=False)
print(f"\nОбработанные данные сохранены в: {output_path}")
```

```
Информация о данных перед обработкой:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 4 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   education       32561 non-null  object
 1   marital-status  32561 non-null  object
 2   occupation      32561 non-null  object
 3   income          32561 non-null  object
dtypes: object(4)
memory usage: 1017.7+ KB
None

Первые 5 строк данных:
   education      marital-status         occupation income
0  Bachelors       Never-married       Adm-clerical  <=50K
1  Bachelors  Married-civ-spouse    Exec-managerial  <=50K
2    HS-grad            Divorced  Handlers-cleaners  <=50K
3       11th  Married-civ-spouse  Handlers-cleaners  <=50K
4  Bachelors  Married-civ-spouse     Prof-specialty  <=50K

Результат Label Encoding для education:
   education  education_encoded
0  Bachelors                12
1  Bachelors                12
2    HS-grad                 8
3       11th                 6
4  Bachelors                12
5    Masters                13
6        9th                 4
7    HS-grad                 8
8    Masters                13
9  Bachelors                12

Количество пропущенных значений:
marital-status    0
occupation        0
dtype: int64

Столбцы после One-Hot Encoding:
Index(['marital-status_Married-AF-spouse', 'marital-status_Married-civ-spouse',
       'marital-status_Married-spouse-absent', 'marital-status_Never-married',
       'marital-status_Separated', 'marital-status_Widowed',
       'occupation_Adm-clerical', 'occupation_Armed-Forces',
       'occupation_Craft-repair', 'occupation_Exec-managerial',
       'occupation_Farming-fishing', 'occupation_Handlers-cleaners',
       'occupation_Machine-op-inspct', 'occupation_Other-service',
       'occupation_Priv-house-serv', 'occupation_Prof-specialty',
       'occupation_Protective-serv', 'occupation_Sales',
       'occupation_Tech-support', 'occupation_Transport-moving'],
      dtype='object')

Размерность таблицы до кодирования: (32561, 16)
Размерность таблицы после кодирования: (32561, 34)

Проверка на дамми-ловушку:
Уникальные значения marital-status: 7
Количество столбцов после кодирования: 6
Уникальные значения occupation: 15
Количество столбцов после кодирования: 14
```

Рисунок 4. Листинг программы задание 4

5. Задание 5:

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler, MinMaxScaler, OrdinalEncoder, OneHotEncoder
import missingno as msno

# Загрузка данных
df = pd.read_csv(file_path)

# Первые 5 строк датасета
print("Первые 5 строк датасета:")
display(df.head())

# Общая информация о данных
print("\nИнформация о датасете:")
display(df.info())

# Описательная статистика
print("\nОписательная статистика:")
display(df.describe().T)

# Проверка на пропущенные значения
print("Количество пропущенных значений в каждом столбце:")
display(df.isna().sum())

# Визуализация пропусков
msno.matrix(df)
plt.title('Матрица пропущенных значений')
plt.show()

# Выбор числовых признаков для анализа выбросов
numeric_cols = ['Age', 'RestingBP', 'Cholesterol', 'MaxHR', 'Oldpeak']

# Функция для удаления 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
    return data[(data[column] >= lower) & (data[column] <= upper)]

# Построение boxplot до удаления выбросов
plt.figure(figsize=(15, 8))
for i, col in enumerate(numeric_cols, 1):
    plt.subplot(2, 3, i)
    sns.boxplot(y=df[col])
    plt.title(f'Boxplot для {col} (до обработки)')
plt.tight_layout()
plt.show()

# Удаление выбросов
original_size = len(df)
for col in numeric_cols:
    df = remove_outliers_iqr(df, col)
new_size = len(df)

# Построение boxplot после удаления выбросов
plt.figure(figsize=(15, 8))
for i, col in enumerate(numeric_cols, 1):
    plt.subplot(2, 3, i)
    sns.boxplot(y=df[col])
    plt.title(f'Boxplot для {col} (после обработки)')
plt.tight_layout()
plt.show()

print(f"Размер датасета до удаления выбросов: {original_size}")
print(f"Размер датасета после удаления выбросов: {new_size}")
print(f"Удалено записей: {original_size - new_size} ({((original_size - new_size)/original_size)*100:.2f}%)")

# Создаем копию датасета для масштабирования
df_scaled = df.copy()

# Стандартизация (Z-преобразование)
scaler = StandardScaler()
df_scaled[numeric_cols] = scaler.fit_transform(df_scaled[numeric_cols])
```

```python
# Визуализация распределения до и после масштабирования
plt.figure(figsize=(15, 6))

# До масштабирования
plt.subplot(1, 2, 1)
sns.histplot(df['Age'], kde=True)
plt.title('Распределение Age до масштабирования')

# После масштабирования
plt.subplot(1, 2, 2)
sns.histplot(df_scaled['Age'], kde=True)
plt.title('Распределение Age после StandardScaler')

plt.tight_layout()
plt.show()

# Проверка среднего и стандартного отклонения после масштабирования
print("\nСредние значения после StandardScaler:")
display(df_scaled[numeric_cols].mean())

print("\nСтандартные отклонения после StandardScaler:")
display(df_scaled[numeric_cols].std())

# Определим порядковые и номинальные признаки
ordinal_features = ['ST_Slope']  # Упорядоченный признак: Down, Flat, Up
nominal_features = ['Sex', 'ChestPainType', 'RestingECG', 'ExerciseAngina']  # Номинальные признаки

# Label Encoding для порядкового признака
ordinal_mapping = {
    'Down': 0,
    'Flat': 1,
    'Up': 2
}
df_scaled['ST_Slope'] = df_scaled['ST_Slope'].map(ordinal_mapping)

# One-Hot Encoding для номинальных признаков
# Используем drop='first' для избежания дамми-ловушки
encoder = OneHotEncoder(drop='first', sparse_output=False)
encoded_nominal = encoder.fit_transform(df_scaled[nominal_features])
encoded_df = pd.DataFrame(encoded_nominal, columns=encoder.get_feature_names_out(nominal_features))

# Объединяем закодированные признаки с основным датасетом
df_final = pd.concat([df_scaled.drop(nominal_features, axis=1), encoded_df], axis=1)

# Проверка размерности до и после кодирования
print(f"Размерность до кодирования: {df_scaled.shape}")
print(f"Размерность после кодирования: {df_final.shape}")

# Просмотр итогового датасета
print("\nПервые 5 строк итогового датасета:")
display(df_final.head())

# Проверка итогового датасета
print("Информация об итоговом датасете:")
display(df_final.info())

# Сохранение обработанного датасета
df_final.to_csv(output_path, index=False)
print(f"\nОбработанный датасет сохранен по пути: {output_path}")
```

Первые 5 строк датасета:

	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

Информация о датасете:

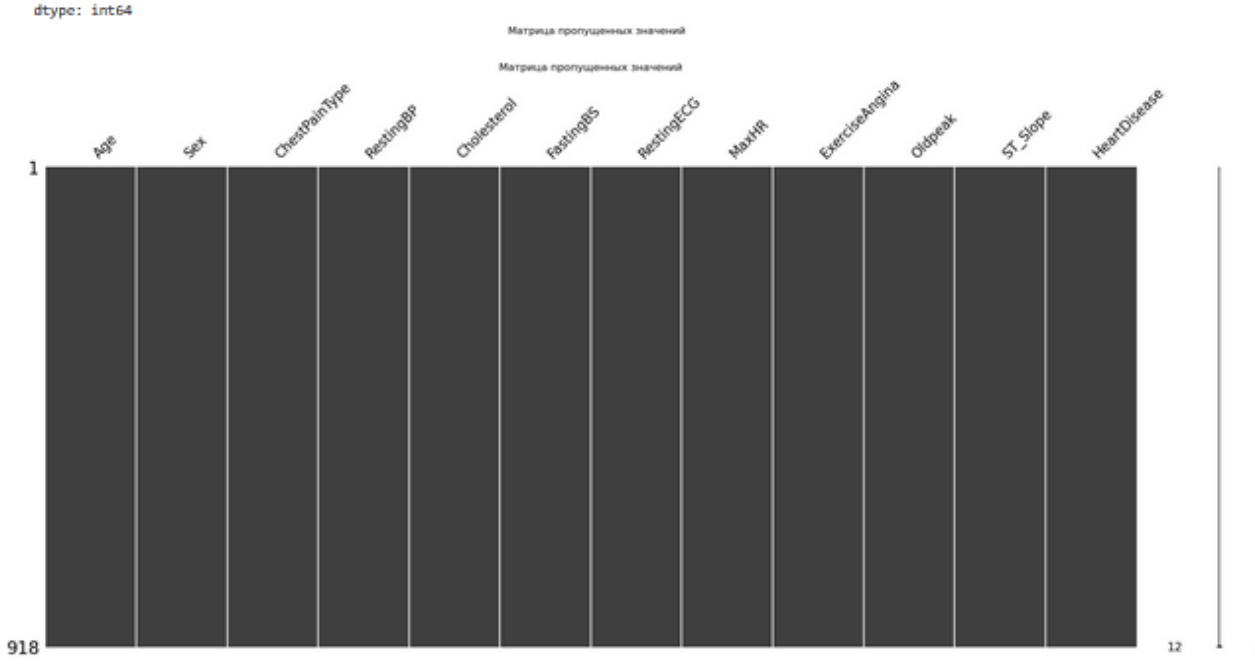
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 918 entries, 0 to 917
Data columns (total 12 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Age             918 non-null    int64  
 1   Sex             918 non-null    object 
 2   ChestPainType   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
None
```

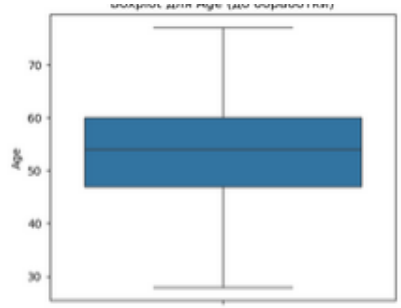
Описательная статистика:

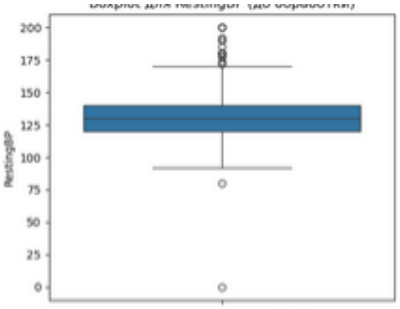
	count	mean	std	min	25%	50%	75%	max
Age	918.0	53.510893	9.432617	28.0	47.00	54.0	60.0	77.0
RestingBP	918.0	132.396514	18.514154	0.0	120.00	130.0	140.0	200.0
Cholesterol	918.0	198.799564	109.384145	0.0	173.25	223.0	267.0	603.0
FastingBS	918.0	0.233115	0.423046	0.0	0.00	0.0	0.0	1.0
MaxHR	918.0	136.809368	25.460334	60.0	120.00	138.0	156.0	202.0
Oldpeak	918.0	0.887364	1.066570	-2.6	0.00	0.6	1.5	6.2
HeartDisease	918.0	0.553377	0.497414	0.0	0.00	1.0	1.0	1.0

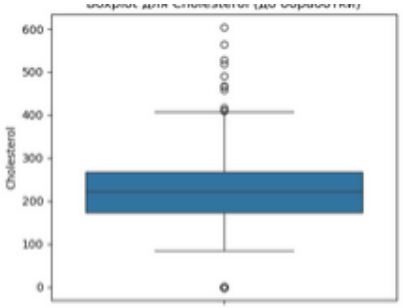
Количество пропущенных значений в каждом столбце:

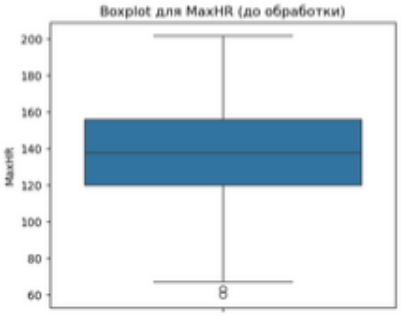
```
Age               0
Sex               0
ChestPainType     0
RestingBP         0
Cholesterol       0
FastingBS         0
RestingECG        0
MaxHR             0
ExerciseAngina    0
Oldpeak           0
ST_Slope          0
HeartDisease      0
dtype: int64
```

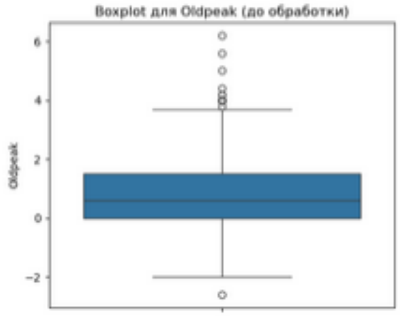


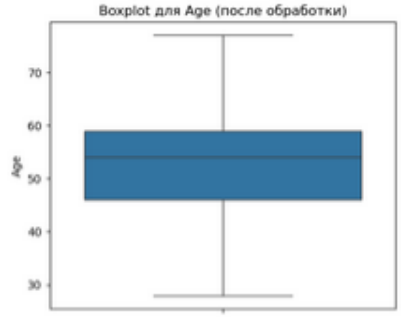


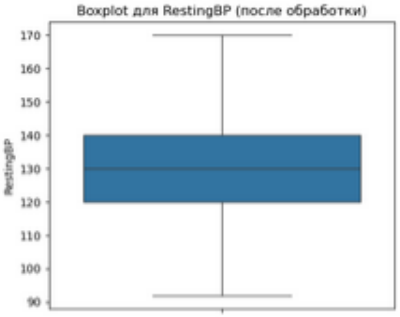


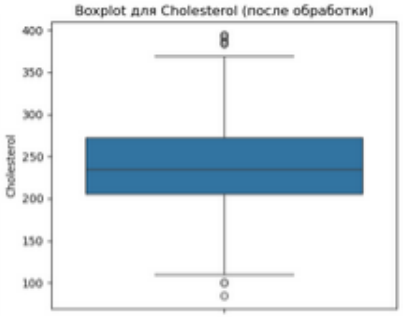


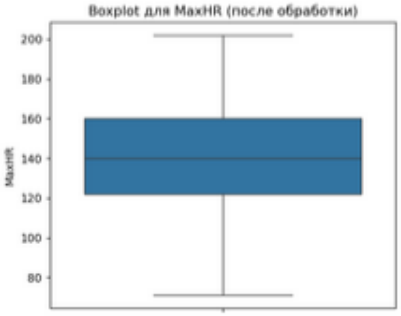


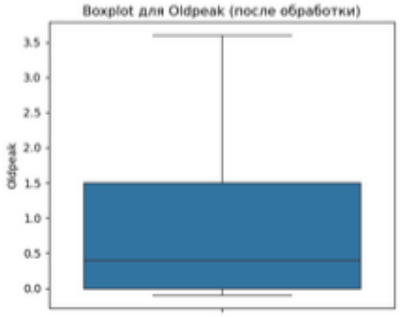




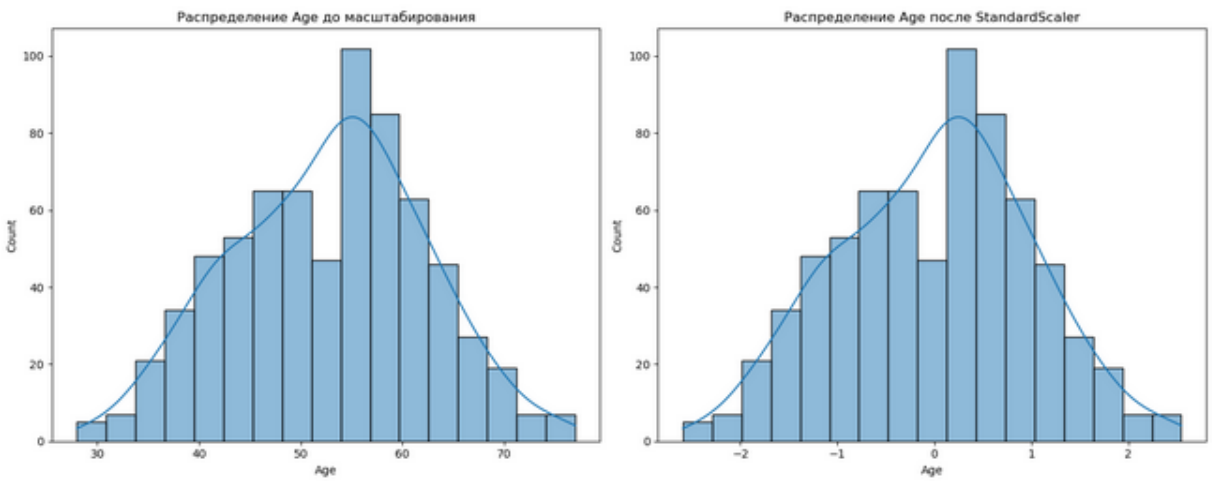









```
Размер датасета до удаления выбросов: 918
Размер датасета после удаления выбросов: 701
Удалено записей: 217 (23.64%)
```




```
Средние значения после StandardScaler:
Age           -4.054452e-17
RestingBP      7.880841e-16
Cholesterol    2.458812e-16
MaxHR          4.257175e-16
Oldpeak       -1.013613e-16
dtype: float64

Стандартные отклонения после StandardScaler:
Age            1.000714
RestingBP      1.000714
Cholesterol    1.000714
MaxHR          1.000714
Oldpeak        1.000714
dtype: float64
Размерность до кодирования: (701, 12)
Размерность после кодирования: (903, 15)

Первые 5 строк итогового датасета:
```

	Age	RestingBP	Cholesterol	FastingBS	MaxHR	Oldpeak	ST_Slope	HeartDisease	Sex_M	ChestPainType_ATA	ChestPainType_NAP	ChestPainType_TA	R
0	-1.334331	0.548422	0.984547	0.0	1.290461	-0.864955	2.0	0.0	1.0	1.0	0.0	0.0	
1	-0.390470	1.845686	-1.182352	0.0	0.633456	0.178110	1.0	1.0	0.0	0.0	1.0	0.0	
2	-1.648952	-0.100209	0.865268	0.0	-1.748185	-0.864955	2.0	0.0	1.0	1.0	0.0	0.0	
3	-0.495344	0.418696	-0.506439	0.0	-1.337557	0.699642	1.0	1.0	0.0	0.0	0.0	0.0	
4	0.133897	1.197054	-0.884155	0.0	-0.762678	-0.864955	2.0	0.0	1.0	0.0	1.0	0.0	

```
Информация об итоговом датасете:
<class 'pandas.core.frame.DataFrame'>
Index: 903 entries, 0 to 673
Data columns (total 15 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Age                 701 non-null    float64
 1   RestingBP           701 non-null    float64
 2   Cholesterol         701 non-null    float64
 3   FastingBS           701 non-null    float64
 4   MaxHR               701 non-null    float64
 5   Oldpeak             701 non-null    float64
 6   ST_Slope            701 non-null    float64
 7   HeartDisease        701 non-null    float64
 8   Sex_M               701 non-null    float64
 9   ChestPainType_ATA   701 non-null    float64
 10  ChestPainType_NAP   701 non-null    float64
 11  ChestPainType_TA    701 non-null    float64
 12  RestingECG_Normal   701 non-null    float64
 13  RestingECG_ST       701 non-null    float64
 14  ExerciseAngina_Y    701 non-null    float64
dtypes: float64(15)
memory usage: 112.9 KB
```

Рисунок 5. Листинг программы задание 5

6. Для выполнения индивидуального задания был выбран Adult Income Dataset:

1. Обзор структуры данных

[29]:
```
import pandas as pd

column_names = [
    "age", "workclass", "fnlwgt", "education", "education-num",
    "marital-status", "occupation", "relationship", "race", "sex",
    "capital-gain", "capital-loss", "hours-per-week", "native-country", "income"
]

df = pd.read_csv("adult.data.csv", header=None, names=column_names, na_values=" ?", skipinitialspace=True)
```

[30]:
```
print(df.info())
print(df.describe(include="all"))
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             32561 non-null  int64
 1   workclass       32561 non-null  object
 2   fnlwgt          32561 non-null  int64
 3   education       32561 non-null  object
 4   education-num   32561 non-null  int64
 5   marital-status  32561 non-null  object
 6   occupation      32561 non-null  object
 7   relationship    32561 non-null  object
 8   race            32561 non-null  object
 9   sex             32561 non-null  object
 10  capital-gain    32561 non-null  int64
 11  capital-loss    32561 non-null  int64
 12  hours-per-week  32561 non-null  int64
 13  native-country  32561 non-null  object
 14  income          32561 non-null  object
dtypes: int64(6), object(9)
memory usage: 3.7+ MB
None
                age workclass        fnlwgt education  education-num  \
count   32561.000000     32561  3.256100e+04     32561   32561.000000
unique           NaN         9           NaN        16            NaN
top              NaN   Private           NaN   HS-grad            NaN
freq             NaN     22696           NaN     10501            NaN
mean       38.581647       NaN  1.897784e+05       NaN      10.080679
std        13.640433       NaN  1.055500e+05       NaN       2.572720
min        17.000000       NaN  1.228500e+04       NaN       1.000000
25%        28.000000       NaN  1.178270e+05       NaN       9.000000
50%        37.000000       NaN  1.783560e+05       NaN      10.000000
75%        48.000000       NaN  2.370510e+05       NaN      12.000000
max        90.000000       NaN  1.484705e+06       NaN      16.000000
```

	marital-status	occupation	relationship	race	sex	\
count	32561	32561	32561	32561	32561	
unique	7	15	6	5	2	
top	Married-civ-spouse	Prof-specialty	Husband	White	Male	
freq	14976	4140	13193	27816	21790	
mean	NaN	NaN	NaN	NaN	NaN	
std	NaN	NaN	NaN	NaN	NaN	
min	NaN	NaN	NaN	NaN	NaN	
25%	NaN	NaN	NaN	NaN	NaN	
50%	NaN	NaN	NaN	NaN	NaN	
75%	NaN	NaN	NaN	NaN	NaN	
max	NaN	NaN	NaN	NaN	NaN	

	capital-gain	capital-loss	hours-per-week	native-country	income
count	32561.000000	32561.000000	32561.000000	32561	32561
unique	NaN	NaN	NaN	42	2
top	NaN	NaN	NaN	United-States	<=50K
freq	NaN	NaN	NaN	29170	24720
mean	1077.648844	87.303830	40.437456	NaN	NaN
std	7385.292085	402.960219	12.347429	NaN	NaN
min	0.000000	0.000000	1.000000	NaN	NaN
25%	0.000000	0.000000	40.000000	NaN	NaN
50%	0.000000	0.000000	40.000000	NaN	NaN
75%	0.000000	0.000000	45.000000	NaN	NaN
max	99999.000000	4356.000000	99.000000	NaN	NaN

2. Обработка пропущенных значений

```
for column in ["workclass", "occupation", "native-country"]:
    mode_value = df[column].mode()[0]
    df[column] = df[column].fillna(mode_value)
```

3. Обнаружение и удаление выбросов

```
for column in ["age", "hours-per-week"]:
    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
    df = df[(df[column] >= lower_bound) & (df[column] <= upper_bound)]
```

4. Масштабирование числовых признаков

```
for column in ["age", "education-num", "hours-per-week"]:
    mean = df[column].mean()
    std = df[column].std()
    df[column] = (df[column] - mean) / std
```

5. Кодирование категориальных признаков

```
education_order = {
    "Preschool": 0,
    "1st-4th": 1,
    "5th-6th": 2,
    "7th-8th": 3,
    "9th": 4,
    "10th": 5,
    "11th": 6,
    "12th": 7,
    "HS-grad": 8,
    "Some-college": 9,
    "Assoc-voc": 10,
    "Assoc-acdm": 11,
    "Bachelors": 12,
    "Masters": 13,
    "Prof-school": 14,
    "Doctorate": 15
}
df["education"] = df["education"].map(education_order)
```

```
df = pd.get_dummies(df, columns=["workclass"], drop_first=True)
```

6. Финальный обзор данных

```
print(df.info())
print(df.head())
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 23499 entries, 0 to 32560
Data columns (total 22 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   age                          23499 non-null  float64
 1   fnlwgt                       23499 non-null  int64
 2   education                    0 non-null      float64
 3   education-num                23499 non-null  float64
 4   marital-status               23499 non-null  object
 5   occupation                   23499 non-null  object
 6   relationship                 23499 non-null  object
 7   race                         23499 non-null  object
 8   sex                          23499 non-null  object
 9   capital-gain                 23499 non-null  int64
 10  capital-loss                 23499 non-null  int64
 11  hours-per-week               23499 non-null  float64
 12  native-country               23499 non-null  object
 13  income                       23499 non-null  object
 14  workclass_Federal-gov        23499 non-null  bool
 15  workclass_Local-gov          23499 non-null  bool
 16  workclass_Never-worked       23499 non-null  bool
 17  workclass_Private            23499 non-null  bool
 18  workclass_Self-emp-inc       23499 non-null  bool
 19  workclass_Self-emp-not-inc   23499 non-null  bool
 20  workclass_State-gov          23499 non-null  bool
 21  workclass_Without-pay        23499 non-null  bool
dtypes: bool(8), float64(4), int64(3), object(7)
memory usage: 2.9+ MB
None
        age  fnlwgt  education  education-num      marital-status  \
0  0.021298   77516        NaN       1.139338       Never-married
2 -0.060734  215646        NaN      -0.436072            Divorced
3  1.169747  234721        NaN      -1.223777  Married-civ-spouse
4 -0.881055  338409        NaN       1.139338  Married-civ-spouse
5 -0.142766  284582        NaN       1.533190  Married-civ-spouse

          occupation   relationship   race     sex  capital-gain  ...  \
0       Adm-clerical  Not-in-family  White    Male          2174  ...
2  Handlers-cleaners  Not-in-family  White    Male             0  ...
3  Handlers-cleaners        Husband  Black    Male             0  ...
4     Prof-specialty           Wife  Black  Female             0  ...
5    Exec-managerial           Wife  White  Female             0  ...

   native-country  income workclass_Federal-gov workclass_Local-gov  \
0   United-States   <=50K                False               False
2   United-States   <=50K                False               False
3   United-States   <=50K                False               False
4            Cuba   <=50K                False               False
5   United-States   <=50K                False               False

   workclass_Never-worked  workclass_Private  workclass_Self-emp-inc  \
0                   False              False                   False
2                   False               True                   False
3                   False               True                   False
4                   False               True                   False
5                   False               True                   False

   workclass_Self-emp-not-inc  workclass_State-gov  workclass_Without-pay
0                       False                True                  False
2                       False               False                  False
3                       False               False                  False
4                       False               False                  False
5                       False               False                  False

[5 rows x 22 columns]
```

Рисунок 6. Листинг программы индивидуального задания

Ответы на контрольные вопросы:

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

Пропущенные значения могут искажать статистические показатели, приводить к ошибкам в моделях машинного обучения и снижать мощность выборки.

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

В pandas можно использовать методы `df.isna().sum()` для подсчета пропусков по столбцам и `df.isna().any()` для проверки их наличия.

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

Метод `.dropna()` удаляет строки или столбцы с пропущенными значениями. Основные параметры:

- `axis=0` (строки) или `axis=1` (столбцы)

- `how='any'` (удалить, если есть хотя бы один пропуск) или `'all'` (если все значения пропущены)

- `subset` для указания столбцов

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

- Среднее подходит для нормального распределения, но чувствительно к выбросам

- Медиана устойчива к выбросам, хороша для асимметричных данных

- Мода используется для категориальных данных

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

`fillna(method='ffill')` заполняет пропуски предыдущим известным значением. Применяется в временных рядах и данных с естественным порядком.

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

`interpolate()` вычисляет промежуточные значения между известными точками (линейная, полиномиальная интерполяция), тогда как `fillna()` просто заменяет пропуски фиксированными значениями.

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

Выбросы - аномальные значения, значительно отличающиеся от основной массы данных. Они искажают статистические показатели и работу моделей.

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

Метод IQR (межквартильный размах):

- IQR = Q3 (75-й перцентиль) - Q1 (25-й перцентиль)
- Границы выбросов: Q1 - 1.5*IQR (нижняя), Q3 + 1.5*IQR (верхняя)

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

```
Q1 = df['column'].quantile(0.25)
Q3 = df['column'].quantile(0.75)
IQR = Q3 - Q1
df = df[(df['column'] >= Q1-1.5*IQR) & (df['column'] <= Q3+1.5*IQR)]
```

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

Метод `.clip()` ограничивает значения заданными границами, заменяя выбросы на пороговые значения.

11. Зачем может потребоваться логарифмическое преобразование числовых признаков?

Логарифмическое преобразование (`np.log1p()`) уменьшает асимметрию распределения и сжимает диапазон больших значений.

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

Для обнаружения выбросов используют:

- Boxplot (ящик с усами)

- Точечные диаграммы (scatter plot)

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

Удаление выбросов требует осторожности, так как они могут содержать важную информацию о редких, но значимых событиях.

14. Зачем необходимо масштабирование признаков перед обучением моделей?

Масштабирование признаков необходимо для:

- Алгоритмов, чувствительных к масштабу данных (KNN, SVM, нейросети)

- Ускорения сходимости градиентного спуска

15. Чем отличается стандартизация от нормализации?

Отличия:

- Стандартизация: (x - mean)/std, диапазон ≈[-3,3], сохраняет выбросы

- Нормализация: (x - min)/(max - min), диапазон [0,1], чувствительна к выбросам

16. Что делает StandardScaler и как рассчитываются преобразованные значения?

`StandardScaler` преобразует данные к среднему=0 и std=1. Формула: (x - μ)/σ

17. Как работает MinMaxScaler и когда его использование предпочтительно?

`MinMaxScaler` сжимает данные в диапазон [0,1]. Подходит, когда важны границы значений.

18. В чём преимущества RobustScaler при наличии выбросов?

`RobustScaler` использует медиану и IQR, устойчив к выбросам.

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

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

20. Какие типы моделей наиболее чувствительны к масштабу признаков?

Наиболее чувствительны к масштабу: KNN, SVM, линейные модели, нейросети.

21. Почему необходимо преобразовывать категориальные признаки перед обучением модели?

Категориальные признаки преобразуют в числовые, так как большинство алгоритмов работают только с числами.

22. Что такое порядковый признак? Приведите пример.

Порядковый признак - категории с естественным порядком (например, "низкий", "средний", "высокий").

23. Что такое номинальный признак? Приведите пример.

Номинальный признак - категории без порядка (например, цвета, названия городов).

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

Метод `.factorize()` присваивает категориям числовые коды (0,1,2...). Подходит для порядковых данных.

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

mapping = {'низкий':0, 'средний':1, 'высокий':2}

df['column'] = df['column'].map(mapping)

26. Что делает класс OrdinalEncoder из scikit-learn?

`OrdinalEncoder` из sklearn аналогичен `.factorize()`, но работает с несколькими столбцами.

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

One-hot кодирование создает отдельный бинарный столбец для каждой категории. Применяется для номинальных признаков.

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

Чтобы избежать дамми-ловушки, один из столбцов удаляют (`drop_first=True` в `pd.get_dummies()`).

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

`OneHotEncoder` из sklearn интегрируется в pipeline, а `pd.get_dummies()` проще в использовании.

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

Target encoding заменяет категории средним значением целевой переменной. Риски: утечка данных и переобучение на редких категориях.

Вывод: В ходе лабораторной работы были изучены и применены основные этапы исследовательского анализа данных (EDA), включая обнаружение и обработку пропущенных значений, выявление и устранение выбросов, масштабирование числовых признаков и кодирование категориальных переменных. Практические задания позволили закрепить навыки работы с библиотекой pandas, для подготовки данных к дальнейшему анализу и моделированию. Результатом работы стало освоение универсальных методов EDA, которые могут быть применены к любым структурированным данным для повышения качества анализа и прогнозирования.