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 «РаботасJupyterNotebook, JupyterLab, GoogleColab» по дисциплине «Искусственный интеллект и машинное обучение»

Выполнил:
Гончаров Серафим Ростиславович 2 курс, группа ИВТ-б-о-23-1,09.03.01 «Информатика и вычислительная техника», направленность (профиль) «Автоматизированные системы обработки информации и управления», очная форма обучения

Руководитель практики:
Воронкин Роман Александрович

Ставрополь 2025

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.

Порядок выполнения лабораторной работы:

GoncharovSerafim/Lab6

Задание 1. Обнаружение пропусков в DataFrame.

```
import missingno as msno
import matplotlib.pyplot as plt
msno.matrix(df)
plt.show()
```

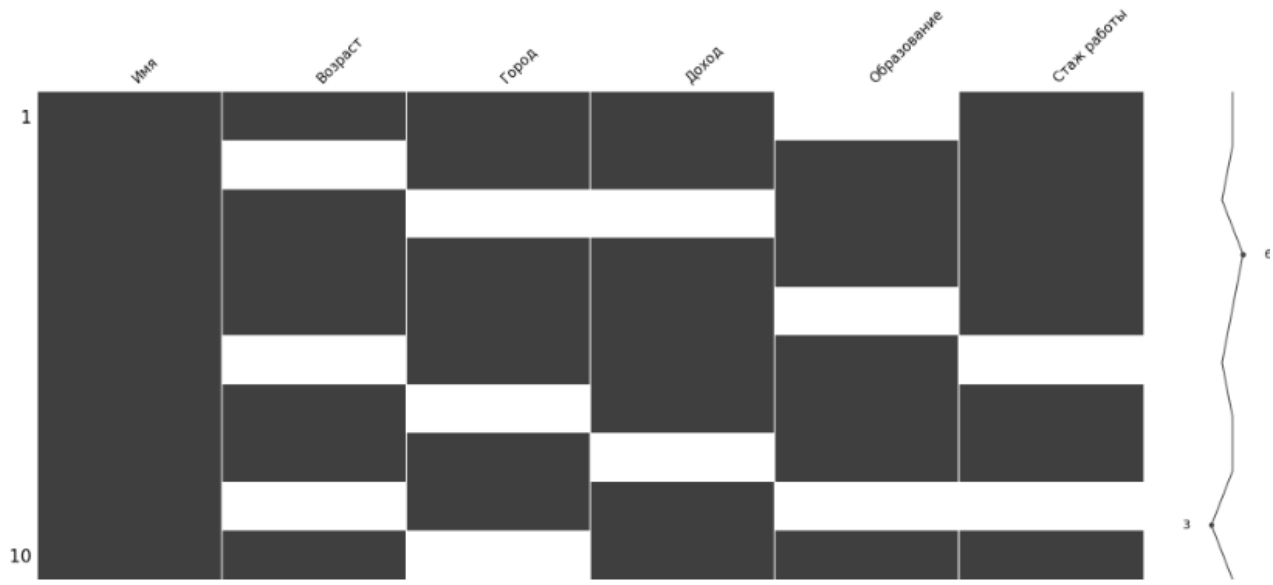


Рисунок 1 – Обнаружение пропусков

Задание 2. Удаление пропущенных значений в DataFrame.

```
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
```

```
df_cleaned = df.dropna()
print(df_cleaned)
```

```
     Имя  Возраст          Город    Доход
0   Анна     25.0          Москва  50000.0
3   Петр     40.0      Новосибирск  70000.0
4  Мария     35.0             СПб  65000.0
```

```
df_cleaned = df.dropna(how="all")
print(df_cleaned)
```

```
       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
```

```
df_cleaned = df.dropna(axis=1)
print(df_cleaned)
```

```
       Имя
0     Анна
1     Иван
```

Рисунок 2 – Удаление пропущенных значений

Задание 3. Заполнение пропусков в DataFrame с помощью .fillna().

```
[33]: df["Возраст"] = df["Возраст"].fillna(30)

[38]: mean_value = df["Доход"].mean()
      df["Доход"] = df["Доход"].fillna(mean_value)

[40]: median_value = df["Доход"].median()
      df["Доход"] = df["Доход"].fillna(median_value)

[42]: mode_value = df["Город"].mode()[0] # mode() возвращает Series
      df["Город"] = df["Город"].fillna(mode_value)

[48]: df["Возраст"] = df["Возраст"].ffill()

[50]: df["Возраст"] = df["Возраст"].bfill()

[54]: df.ffill (axis=0, inplace=True)

[68]: df["Возраст"] = df["Возраст"].ffill(limit=1)
```

Рисунок 3 – Заполнение пропусков

Задание 4. Интерполяция пропущенных значений в DataFrame с помощью .interpolate().

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

```
   день  температура  температура_interp
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
```

```
[72]: df["температура_poly"] = df["температура"].interpolate(method="polynomial", order=2)
```

```
[74]: dates = pd.date_range("2024-01-01", periods=5, freq="D")
      df = pd.DataFrame({
      "дата": dates,
      "уровень воды": [1.2, np.nan, np.nan, 1.8, 2.0]
      })
      df.set_index("дата", inplace=True)
      df["интерполяция"] = df["уровень воды"].interpolate(method="time")
      print(df)
```

```
            уровень воды  интерполяция
дата
2024-01-01           1.2           1.2
2024-01-02           NaN           1.4
2024-01-03           NaN           1.6
2024-01-04           1.8           1.8
2024-01-05           2.0           2.0
```

```
[80]: df = df.interpolate(limit=1, limit_direction="forward")
```

```
[78]: df.interpolate(method="linear", axis=0, inplace=True)
```

Рисунок 4 – Интерполяция пропущенных значений

Задание 5. Обнаружение выбросов в данных.

[86]:
```
Q1 = df["Доход"].quantile(0.25)
Q3 = df["Доход"].quantile(0.75)
IQR = Q3 - Q1
outliers = df[(df["Доход"] < Q1 - 1.5 * IQR) |
(df["Доход"] > Q3 + 1.5 * IQR)]
```

[88]:
```
import matplotlib.pyplot as plt
# Гистограмма (Возраст)
plt.figure(figsize=(6, 4))
plt.hist(df["Возраст"], bins=20, color='cornflowerblue',
edgecolor='black')
plt.title("Гистограмма: Возраст")
plt.xlabel("Возраст")
plt.ylabel("Частота")
plt.grid(True)
plt.tight_layout()
plt.show()
```

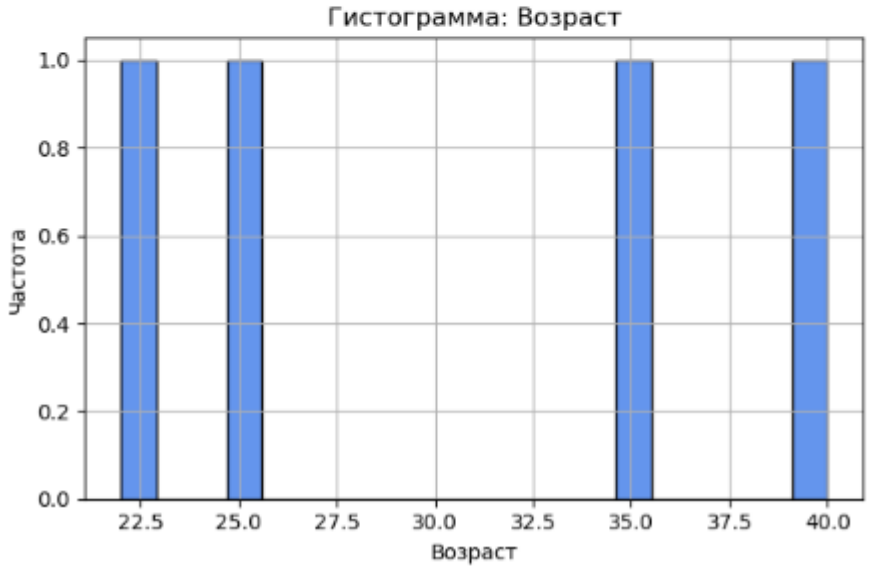


Рисунок 5 – Обнаружение выбросов в данных

Задание 6. Обработка выбросов в pandas.

```
import pandas as pd
import numpy as np
data = {
"Имя": ["Анна", "Иван", "Ольга", "Петр", "Мария", "Дмитрий",
"Елена"],
"Баланс на счете": [50000, 60000, 45000, 70000, 65000, 400000,
450000]
}
df = pd.DataFrame(data)
print(df)
```

```
       Имя  Баланс на счете
0     Анна            50000
1     Иван            60000
2    Ольга            45000
3     Петр            70000
4    Мария            65000
5  Дмитрий           400000
6    Елена           450000
```

```
Q1 = df["Баланс на счете"].quantile(0.25)
Q3 = df["Баланс на счете"].quantile(0.75)
IQR = Q3 - Q1
lower = Q1 - 1.5 * IQR
upper = Q3 + 1.5 * IQR
df_removed = df[(df["Баланс на счете"] >= lower) & (df["Баланс на счете"] <= upper)]
print(df_removed)
```

```
       Имя  Баланс на счете
0     Анна            50000
1     Иван            60000
2    Ольга            45000
3     Петр            70000
4    Мария            65000
5  Дмитрий           400000
6    Елена           450000
```

Рисунок 6 – Обработка выбросов в pandas

Задание 7. Стандартизация признаков.

[112]:
```
import pandas as pd
data = {
"Имя": ["Анна", "Иван", "Ольга", "Петр", "Мария"],
"Возраст": [25, 30, 22, 40, 35],
"Зарплата": [50000, 60000, 45000, 70000, 65000]
}
df = pd.DataFrame(data)
print(df)
```

```
     Имя  Возраст  Зарплата
0   Анна       25     50000
1   Иван       30     60000
2  Ольга       22     45000
3   Петр       40     70000
4  Мария       35     65000
```

[116]:
```
df_standardized = df.copy()
df_standardized["Возраст"] = (df["Возраст"] - df["Возраст"].mean()) / df["Возраст"].std()
df_standardized["Зарплата"] = (df["Зарплата"] - df["Зарплата"].mean()) / df["Зарплата"].std()
print(df_standardized)
```

```
     Имя   Возраст  Зарплата
0   Анна -0.739657 -0.771589
1   Иван -0.054789  0.192897
2  Ольга -1.150577 -1.253831
3   Петр  1.314945  1.157383
4  Мария  0.630078  0.675140
```

[118]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
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 -0.826961 -0.862662
1   Иван -0.061256  0.215666
2  Ольга -1.286384 -1.401826
3   Петр  1.470153  1.293993
4  Мария  0.704448  0.754829
```

Рисунок 7 – Стандартизация признаков

Задание 8. Нормализация признаков.

```
df_normalized = df.copy()
df_normalized["Возраст"] = (df["Возраст"] - df["Возраст"].min()) / (df["Возраст"].max() - df["Возраст"].min())
df_normalized["Зарплата"] = (df["Зарплата"] - df["Зарплата"].min()) / (df["Зарплата"].max() - df["Зарплата"].min())
print(df_normalized)
```

```
     Имя   Возраст  Зарплата
0   Анна  0.166667       0.2
1   Иван  0.444444       0.6
2  Ольга  0.000000       0.0
3   Петр  1.000000       1.0
4  Мария  0.722222       0.8
```

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
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  0.166667       0.2
1   Иван  0.444444       0.6
2  Ольга  0.000000       0.0
3   Петр  1.000000       1.0
4  Мария  0.722222       0.8
```

Рисунок 8 – Нормализация признаков

Задание 9. Робастное масштабирование признаков.

[128]:
```
import pandas as pd
data = {
"Имя": ["Анна", "Иван", "Ольга", "Петр", "Мария", "Дмитрий",
"Елена"],
"Возраст": [25, 30, 22, 40, 35, 120, 5], # выбросы: 120, 5
"Зарплата": [50000, 60000, 45000, 70000, 65000, 1000000, 10000] #выбросы: 1000000, 10000
}

df = pd.DataFrame(data)
print(df)
```

```
       Имя  Возраст  Зарплата
0     Анна       25     50000
1     Иван       30     60000
2    Ольга       22     45000
3     Петр       40     70000
4    Мария       35     65000
5  Дмитрий      120   1000000
6    Елена        5     10000
```

[133]:
```
df_robust = df.copy()
for col in ["Возраст", "Зарплата"]:
    median = df[col].median()
q1 = df[col].quantile(0.25)
q3 = df[col].quantile(0.75)
iqr = q3 - q1
df_robust[col] = (df[col] - median) / iqr
print(df_robust)
```

```
       Имя  Возраст  Зарплата
0     Анна       25     -0.50
1     Иван       30      0.00
2    Ольга       22     -0.75
3     Петр       40      0.50
4    Мария       35      0.25
5  Дмитрий      120     47.00
6    Елена        5     -2.50
```

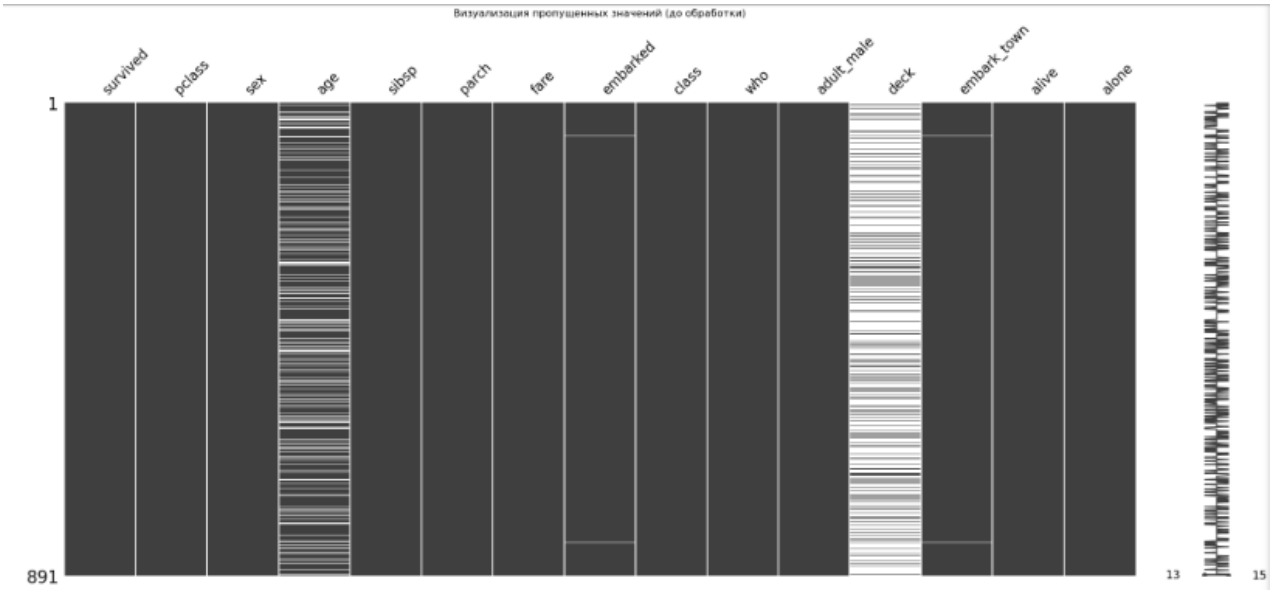
Рисунок 9 – Робастное масштабирование признаков

Задание 10. Задание 1. Обнаружение и обработка пропущенных значений

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()).

[3]:
```
import seaborn as sns
import matplotlib.pyplot as plt
import missingno as msno
import pandas as pd
titanic = sns.load_dataset("titanic")
print("1. Датасет загружен:")
print(titanic.head())
print("\n2. Пропущенные значения ДО обработки:")
print(titanic.isna().sum())
plt.figure(figsize=(10, 5))
msno.matrix(titanic)
plt.title("Визуализация пропущенных значений (до обработки)")
plt.show()
titanic_processed = titanic.copy()
titanic_processed['age'] = titanic_processed['age'].fillna(titanic_processed['age'].mean())
mode_embarked = titanic_processed['embarked'].mode()[0]
titanic_processed['embarked'] = titanic_processed['embarked'].fillna(mode_embarked)
titanic_processed = titanic_processed.drop('deck', axis=1)
print("\n3. Информация ДО обработки:")
titanic.info()
print("\n4. Пропущенные значения ПОСЛЕ обработки:")
print(titanic_processed.isna().sum())
print("\n5. Информация ПОСЛЕ обработки:")
titanic_processed.info()
plt.figure(figsize=(10, 5))
msno.matrix(titanic_processed)
plt.title("Визуализация пропущенных значений (после обработки)")
plt.show()
```

```
1. Датасет загружен:
   survived  pclass     sex   age  sibsp  parch     fare embarked  class  \
0         0       3    male  22.0      1      0   7.2500        S  Third
1         1       1  female  38.0      1      0  71.2833        C  First
2         1       3  female  26.0      0      0   7.9250        S  Third
3         1       1  female  35.0      1      0  53.1000        S  First
4         0       3    male  35.0      0      0   8.0500        S  Third

     who  adult_male deck  embark_town alive  alone
0    man        True  NaN  Southampton    no  False
1  woman       False    C    Cherbourg   yes  False
2  woman       False  NaN  Southampton   yes   True
3  woman       False    C  Southampton   yes  False
4    man        True  NaN  Southampton    no   True

2. Пропущенные значения ДО обработки:
survived         0
pclass           0
sex              0
age            177
sibsp            0
parch            0
fare             0
embarked         2
class            0
who              0
adult_male       0
deck           688
embark_town      2
alive            0
alone            0
dtype: int64
<Figure size 1000x500 with 0 Axes>
```




```
3. Информация ДО обработки:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   survived     891 non-null    int64
 1   pclass       891 non-null    int64
 2   sex          891 non-null    object
 3   age          714 non-null    float64
 4   sibsp        891 non-null    int64
 5   parch        891 non-null    int64
 6   fare         891 non-null    float64
 7   embarked     889 non-null    object
```

```
4. Пропущенные значения ПОСЛЕ обработки:
survived       0
pclass         0
sex            0
age            0
sibsp          0
parch          0
fare           0
embarked       0
class          0
who            0
adult_male     0
embark_town    2
alive          0
alone          0
dtype: int64

5. Информация ПОСЛЕ обработки:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 14 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   survived     891 non-null    int64
 1   pclass       891 non-null    int64
 2   sex          891 non-null    object
 3   age          891 non-null    float64
 4   sibsp        891 non-null    int64
 5   parch        891 non-null    int64
 6   fare         891 non-null    float64
 7   embarked     891 non-null    object
 8   class        891 non-null    category
 9   who          891 non-null    object
 10  adult_male   891 non-null    bool
 11  embark_town  889 non-null    object
 12  alive        891 non-null    object
 13  alone        891 non-null    bool
dtypes: bool(2), category(1), float64(2), int64(4), object(5)
memory usage: 79.4+ KB
<Figure size 1000x500 with 0 Axes>
```

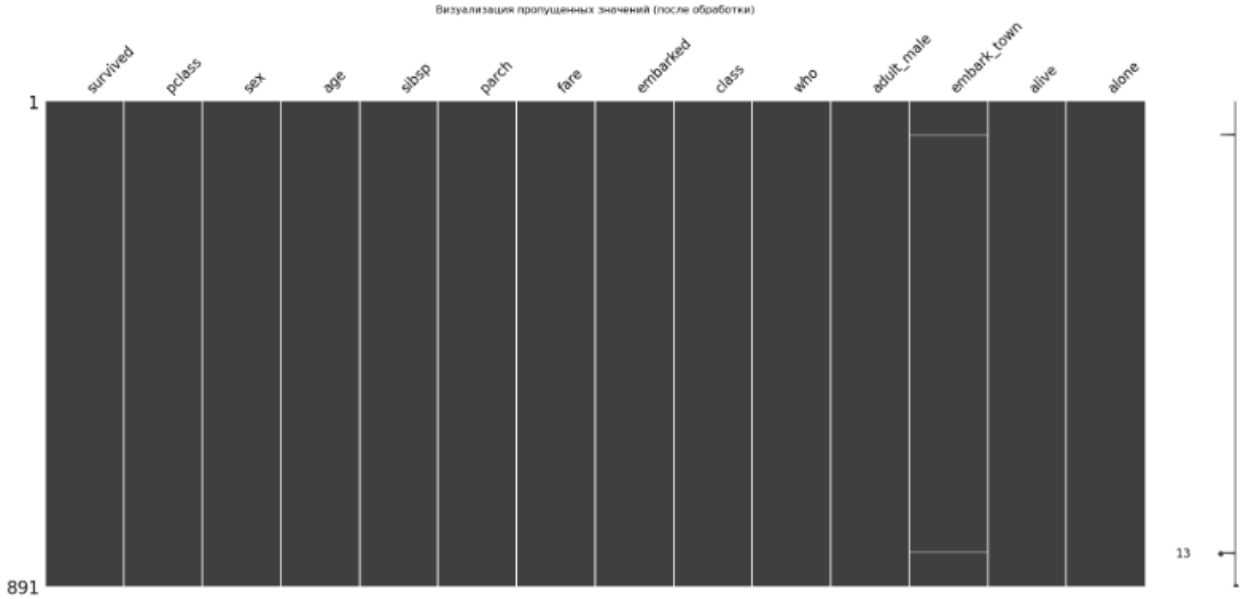


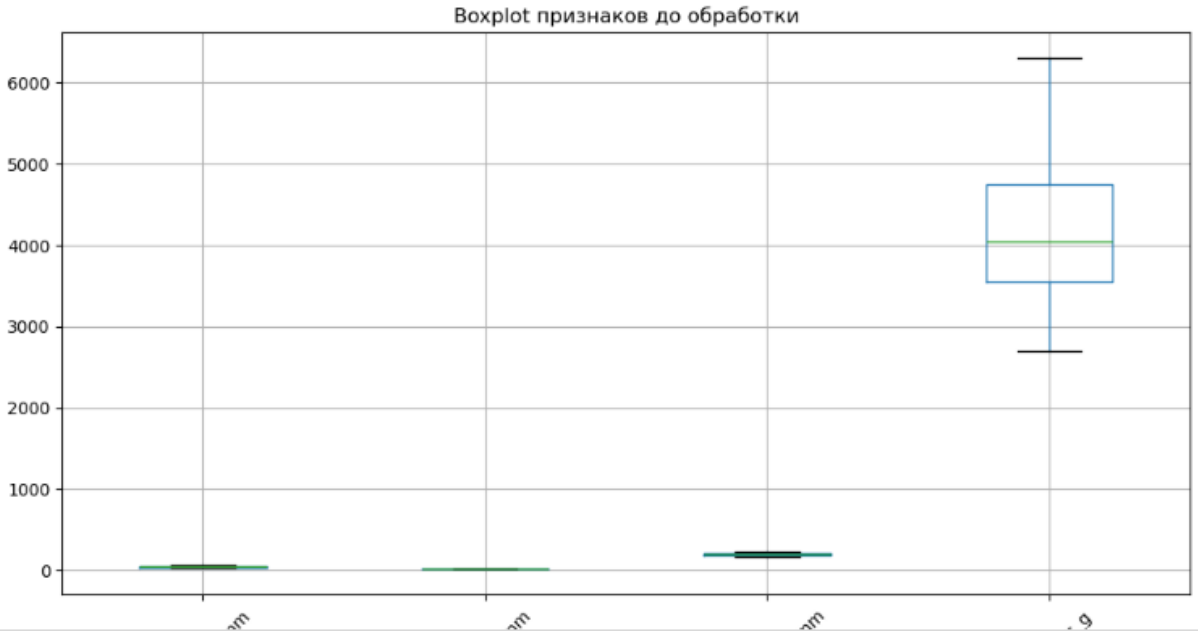
Рисунок 10 – Обнаружение и обработка пропущенных значений

Задание 11. Задание 2. Обнаружение и удаление выбросов

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.

[9]:
```
penguins = sns.load_dataset('penguins')
print("1. Датасет загружен. Размер до обработки:", penguins.shape)
print(penguins.head())
numeric_cols = ['bill_length_mm', 'bill_depth_mm', 'flipper_length_mm', 'body_mass_g']
plt.figure(figsize=(12, 6))
penguins[numeric_cols].boxplot()
plt.title('Boxplot признаков до обработки')
plt.xticks(rotation=45)
plt.show()
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)]
penguins_clean = penguins.copy()
for col in numeric_cols:
    penguins_clean = remove_outliers(penguins_clean, col)
print("\nРазмер датасета до обработки:", penguins.shape)
print("Размер датасета после обработки:", penguins_clean.shape)
print(f"Удалено записей: {len(penguins) - len(penguins_clean)}")
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
sns.boxplot(y=penguins['flipper_length_mm'])
plt.title('До обработки')
plt.subplot(1, 2, 2)
sns.boxplot(y=penguins_clean['flipper_length_mm'])
plt.title('После обработки')
plt.suptitle('Сравнение flipper_length_mm до и после удаления выбросов')
plt.tight_layout()
```

```
1. Датасет загружен. Размер до обработки: (344, 7)
  species     island  bill_length_mm  bill_depth_mm  flipper_length_mm  \
0  Adelie  Torgersen            39.1           18.7              181.0
1  Adelie  Torgersen            39.5           17.4              186.0
2  Adelie  Torgersen            40.3           18.0              195.0
3  Adelie  Torgersen             NaN            NaN                NaN
4  Adelie  Torgersen            36.7           19.3              193.0

   body_mass_g     sex
0       3750.0    Male
1       3800.0  Female
2       3250.0  Female
3          NaN     NaN
4       3450.0  Female
```



```
Размер датасета до обработки: (344, 7)
Размер датасета после обработки: (342, 7)
Удалено записей: 2
```

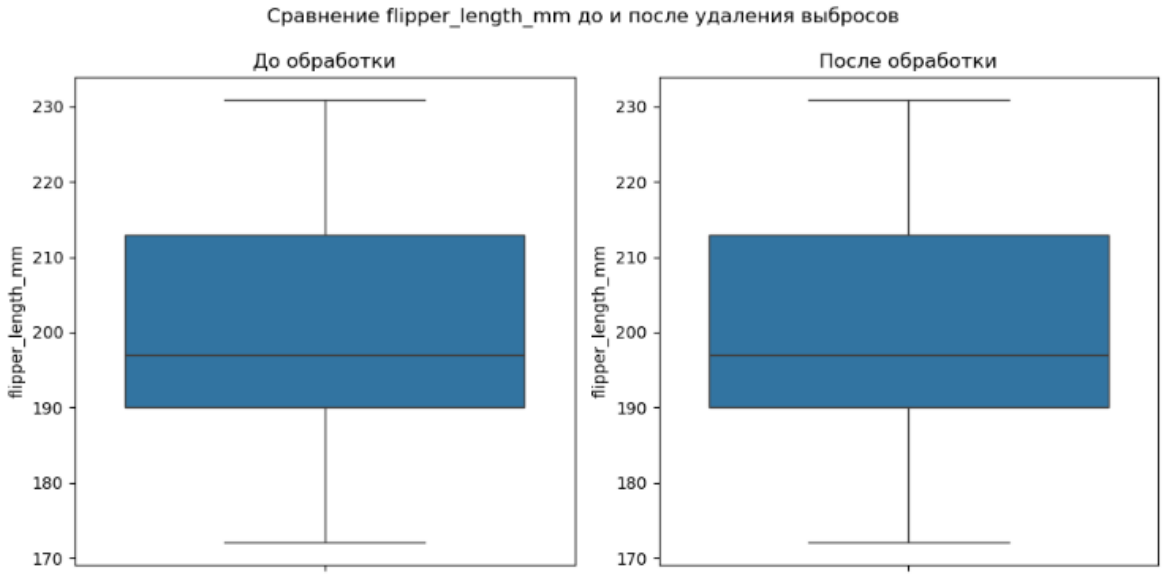


Рисунок 11 – Обнаружение и удаление выбросов

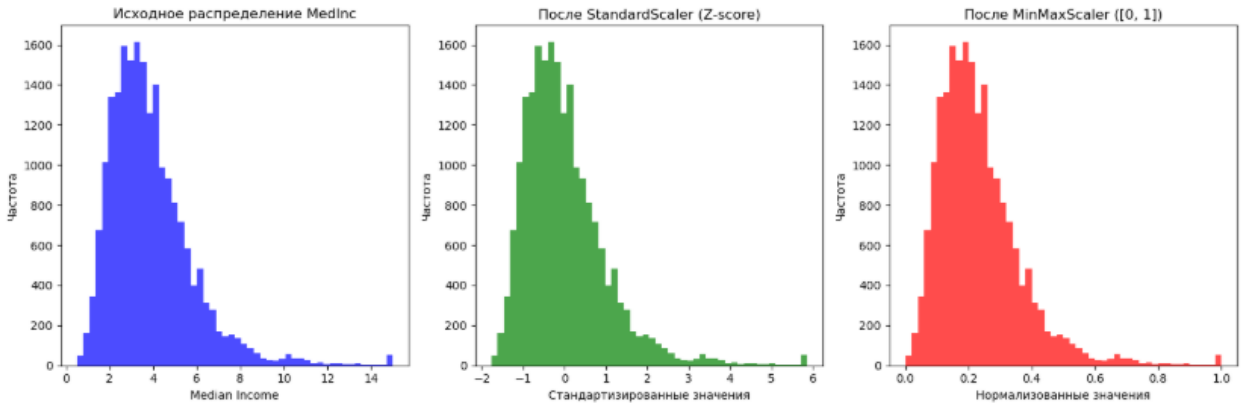
Задание 12. Задание 3. Масштабирование числовых признаков

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 копиитаблицы).
4. Постройте гистограммы распределения признака MedInc до и после масштабирования.
5. Сравните поведение шкал на гистограммах.

```
from sklearn.datasets import fetch_california_housing
from sklearn.preprocessing import StandardScaler, MinMaxScaler

california = fetch_california_housing(as_frame=True)
print("1. Данные загружены. Описание:\n", california.DESCR[:500] + "...")
df = california.frame
print("\n2. Первые 5 строк DataFrame:")
print(df.head())
original_medinc = df['MedInc'].copy()
scaler = StandardScaler()
df_standardized = df.copy()
df_standardized[df.columns] = scaler.fit_transform(df)
minmax_scaler = MinMaxScaler()
df_normalized = df.copy()
df_normalized[df.columns] = minmax_scaler.fit_transform(df)
plt.figure(figsize=(15, 5))
plt.subplot(1, 3, 1)
plt.hist(original_medinc, bins=50, color='blue', alpha=0.7)
plt.title('Исходное распределение MedInc')
plt.xlabel('Median Income')
plt.ylabel('Частота')
plt.subplot(1, 3, 2)
plt.hist(df_standardized['MedInc'], bins=50, color='green', alpha=0.7)
plt.title('После StandardScaler (Z-score)')
plt.xlabel('Стандартизированные значения')
plt.ylabel('Частота')
plt.subplot(1, 3, 3)
plt.hist(df_normalized['MedInc'], bins=50, color='red', alpha=0.7)
plt.title('После MinMaxScaler ([0, 1])')
plt.xlabel('Нормализованные значения')
plt.ylabel('Частота')
```

```
plt.tight_layout()
plt.show()
print("\n5. Сравнение статистик MedInc:")
comparison = pd.DataFrame({
    'Original': original_medinc.describe(),
    'Standardized': df_standardized['MedInc'].describe(),
    'Normalized': df_normalized['MedInc'].describe()
})
print(comparison.round(2))
```

```
1. Данные загружены. Описание:
 .. _california_housing_dataset:

California Housing dataset
--------------------------

**Data Set Characteristics:**

:Number of Instances: 20640

:Number of Attributes: 8 numeric, predictive attributes and the target

:Attribute Information:
    - MedInc        median income in block group
    - HouseAge      median house age in block group
    - AveRooms      average number of rooms per household
    - AveBedrms     average number of bedrooms per household
    - Population    block group popu...

2. Первые 5 строк DataFrame:
   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
```



5. Сравнение статистик MedInc:

	Original	Standardized	Normalized
count	20640.00	20640.00	20640.00
mean	3.87	0.00	0.23
std	1.90	1.00	0.13
min	0.50	-1.77	0.00
25%	2.56	-0.69	0.14
50%	3.53	-0.18	0.21
75%	4.74	0.46	0.29
max	15.00	5.86	1.00

Рисунок 12 – Масштабирование числовых признаков

Задание 13. Задание 4. Кодирование категориальных признаков

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.

```python
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.datasets import fetch_openml

adult = fetch_openml('adult', version=2, as_frame=True)
df = adult.frame[['education', 'marital-status', 'occupation', 'class']].copy()
df = df.rename(columns={'class': 'income'})
print("1. Исходные данные (первые 5 строк):")
print(df.head())
print("\nРазмерность до обработки:", df.shape)
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'
]
education_dict = {v: k for k, v in enumerate(education_order)}
df['education'] = df['education'].map(education_dict)
print("\n2. После Label Encoding education:")
print(df['education'].value_counts().sort_index())
ohe = OneHotEncoder(drop='first', sparse_output=False)  # Явно указываем dense output
encoded_features = ohe.fit_transform(df[['marital-status', 'occupation']])
encoded_df = pd.DataFrame(
    encoded_features,
    columns=ohe.get_feature_names_out(['marital-status', 'occupation']),
    index=df.index
)
df_processed = pd.concat([
    df[['education', 'income']],
    encoded_df
], axis=1)
```

```python
print("\n3. После One-Hot Encoding:")
print(encoded_df.head())
print("\nРазмерность после обработки:", df_processed.shape)
print("\n4. Проверка на дамми-ловушку:")
print("Количество уникальных значений в marital-status:", df['marital-status'].nunique())
print("Количество столбцов после OHE для marital-status:",
      sum(1 for col in encoded_df.columns if 'marital-status' in col))
```

```
1. Исходные данные (первые 5 строк):
      education      marital-status         occupation income
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

Размерность до обработки: (48842, 4)

2. После Label Encoding education:
education
5      1389
6      1812
7       657
1       247
2       509
3       955
4       756
11     1601
10     2061
12     8025
15      594
8     15784
13     2657
0        83
14      834
9     10878
Name: count, dtype: int64
```

```
3. После One-Hot Encoding:
   marital-status_Married-AF-spouse  marital-status_Married-civ-spouse  \
0                               0.0                               0.0
1                               0.0                               1.0
2                               0.0                               1.0
3                               0.0                               1.0
4                               0.0                               0.0

   marital-status_Married-spouse-absent  marital-status_Never-married  \
0                                   0.0                          1.0
1                                   0.0                          0.0
2                                   0.0                          0.0
3                                   0.0                          0.0
4                                   0.0                          1.0

   marital-status_Separated  marital-status_Widowed  occupation_Armed-Forces  \
0                       0.0                     0.0                      0.0
1                       0.0                     0.0                      0.0
2                       0.0                     0.0                      0.0
3                       0.0                     0.0                      0.0
4                       0.0                     0.0                      0.0

   occupation_Craft-repair  occupation_Exec-managerial  \
0                      0.0                         0.0
1                      0.0                         0.0
2                      0.0                         0.0
3                      0.0                         0.0
4                      0.0                         0.0

   occupation_Farming-fishing  occupation_Handlers-cleaners  \
0                         0.0                           0.0
1                         1.0                           0.0
2                         0.0                           0.0
3                         0.0                           0.0
4                         0.0                           0.0

   occupation_Machine-op-inspct  occupation_Other-service  \
0                           1.0                       0.0
1                           0.0                       0.0
2                           0.0                       0.0
3                           1.0                       0.0
4                           0.0                       0.0

   occupation_Priv-house-serv  occupation_Prof-specialty  \
0                         0.0                        0.0
1                         0.0                        0.0
2                         0.0                        0.0
3                         0.0                        0.0
4                         0.0                        0.0

   occupation_Protective-serv  occupation_Sales  occupation_Tech-support  \
0                         0.0               0.0                      0.0
1                         0.0               0.0                      0.0
2                         1.0               0.0                      0.0
3                         0.0               0.0                      0.0
4                         0.0               0.0                      0.0

   occupation_Transport-moving  occupation_nan
0                          0.0            0.0
1                          0.0            0.0
2                          0.0            0.0
3                          0.0            0.0
4                          0.0            1.0

Размерность после обработки: (48842, 22)

4. Проверка на дамми-ловушку:
Количество уникальных значений в marital-status: 7
Количество столбцов после OHE для marital-status: 6
```

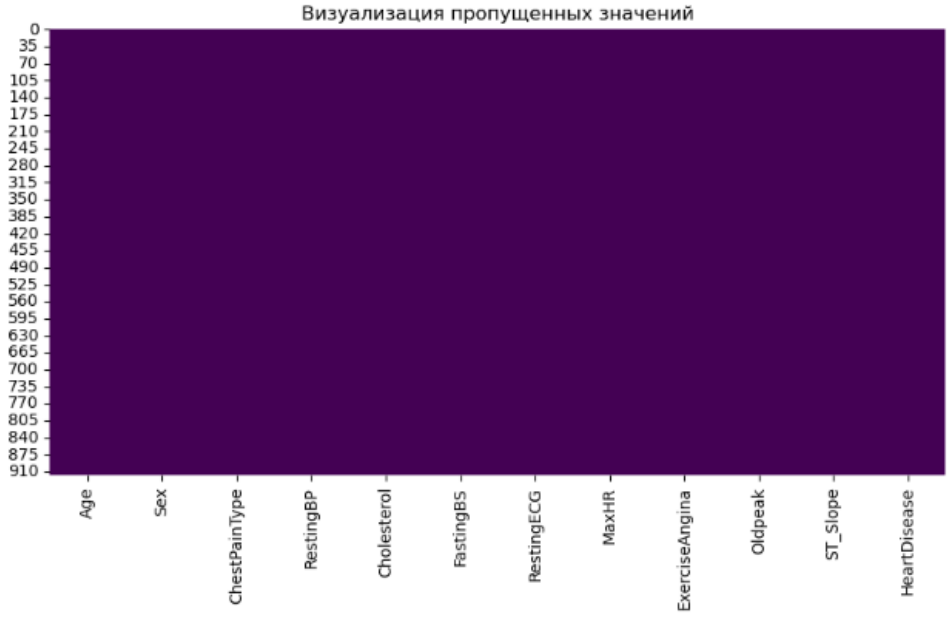
Рисунок 13 – Кодирование категориальных признаков

Задание 14. Задание 5. Комплексный EDA

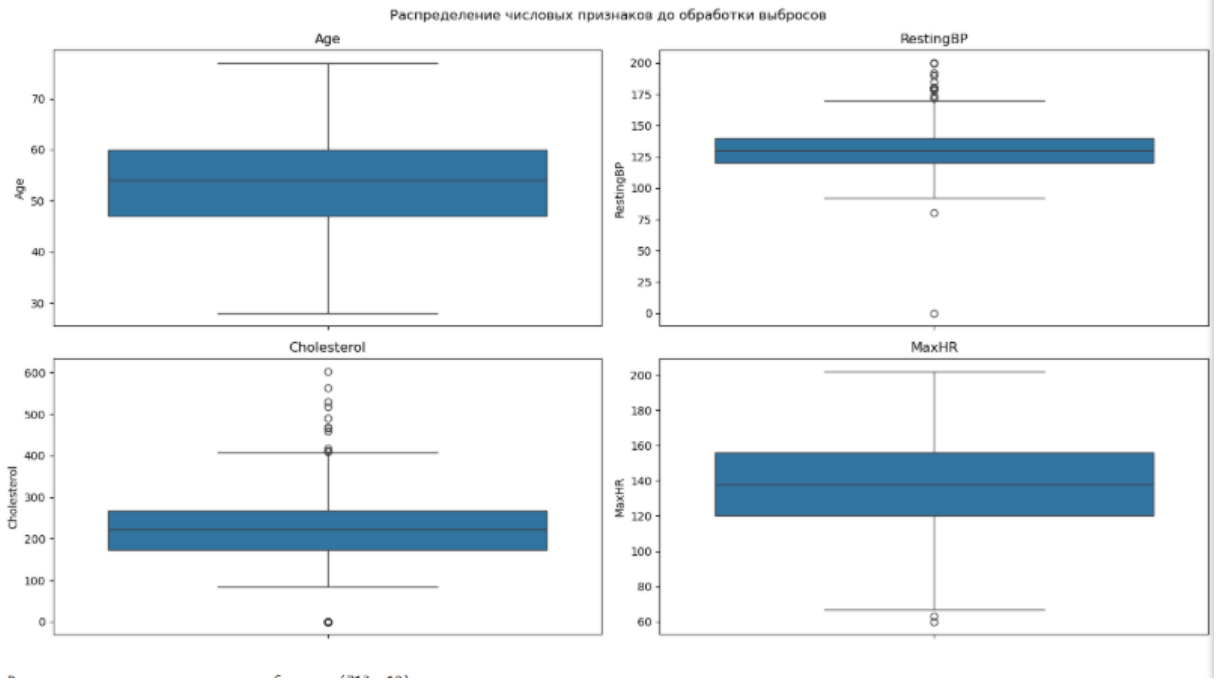
1. Обзор структуры данных (.info() , .describe()).
2. Обнаружение и обработка пропущенных значений.
3. Обнаружение и удаление выбросов по признакам: age , cholesterol ,restingbp , maxhr .
4. Масштабирование числовых признаков.
5. Кодирование категориальных признаков: sex , chestpain , exerciseangina ,restecg .
6. Подготовьте отчёт в виде Jupyter-ноутбука с комментариями к каждому этапу и промежуточными результатами.

```
import seaborn as sns
import matplotlib.pyplot as plt
import missingno as msno
import pandas as pd
from sklearn.datasets import fetch_california_housing
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.datasets import fetch_openml

try:
    df = pd.read_csv('heart.csv')  # Предполагаем, что файл уже загружен
except FileNotFoundError:
    print("Файл не найден. Пожалуйста, загрузите данные с Kaggle и укажите правильный путь.")
    exit()
print("1. ОБЗОР ДАННЫХ")
print("\nИнформация о датасете:")
print(df.info())
print("\nОписательная статистика:")
print(df.describe().transpose())
print("\n2. ОБРАБОТКА ПРОПУЩЕННЫХ ЗНАЧЕНИЙ")
print("\nКоличество пропусков до обработки:")
print(df.isna().sum())
plt.figure(figsize=(10, 5))
sns.heatmap(df.isna(), cbar=False, cmap='viridis')
plt.title("Визуализация пропущенных значений")
plt.show()
print("\nПропуски успешно обработаны")
print("\n3. ОБРАБОТКА ВЫБРОСОВ")
numeric_cols = ['Age', 'RestingBP', 'Cholesterol', 'MaxHR']
plt.figure(figsize=(15, 8))
for i, col in enumerate(numeric_cols, 1):
    plt.subplot(2, 2, i)
    sns.boxplot(y=df[col])
    plt.title(col)
plt.suptitle("Распределение числовых признаков до обработки выбросов")
plt.tight_layout()
plt.show()
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 col in numeric_cols:
    df = remove_outliers(df, col)
print("\nРазмер датасета после удаления выбросов:", df.shape)
print("\n4. МАСШТАБИРОВАНИЕ ЧИСЛОВЫХ ПРИЗНАКОВ")
scaler = StandardScaler()
df[numeric_cols] = scaler.fit_transform(df[numeric_cols])
print("\nПосле масштабирования:")
print(df[numeric_cols].describe().transpose())
print("\n5. КОДИРОВАНИЕ КАТЕГОРИАЛЬНЫХ ПРИЗНАКОВ")
categorical_cols = ['Sex', 'ChestPainType', 'ExerciseAngina', 'RestingECG']
ohe = OneHotEncoder(drop='first', sparse_output=False)
encoded = ohe.fit_transform(df[categorical_cols])
encoded_df = pd.DataFrame(encoded, columns=ohe.get_feature_names_out(categorical_cols))
df_processed = pd.concat([df.drop(categorical_cols, axis=1), encoded_df], axis=1)
print("\nУникальные значения до кодирования:")
for col in categorical_cols:
    print(f"{col}: {df[col].unique()}")
print("\nПервые 5 строк после кодирования:")
print(df_processed.head())
df_processed.to_csv('processed_heart_disease.csv', index=False)
print("\nОбработанные данные сохранены в файл 'processed_heart_disease.csv'")
```

```
1. ОБЗОР ДАННЫХ

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

2. ОБРАБОТКА ПРОПУЩЕННЫХ ЗНАЧЕНИЙ

Количество пропусков до обработки:
Age               0
Sex               0
ChestPainType     0
RestingBP         0
Cholesterol       0
FastingBS         0
RestingECG        0
MaxHR             0
```



```
Пропуски успешно обработаны

3. ОБРАБОТКА ВЫБРОСОВ
```




```
4. МАСШТАБИРОВАНИЕ ЧИСЛОВЫХ ПРИЗНАКОВ

После масштабирования:
             count          mean       std       min       25%       50%  \
Age          713.0 -1.594486e-16  1.000702 -2.598574 -0.713109  0.124875
RestingBP    713.0  3.251256e-16  1.000702 -2.571982 -0.755737 -0.107079
Cholesterol  713.0  6.477599e-17  1.000702 -3.080232 -0.669228 -0.091384
MaxHR        713.0 -1.096209e-16  1.000702 -2.847311 -0.755321 -0.016972

                 75%       max
Age         0.648615  2.534080
RestingBP   0.541580  2.487556
Cholesterol 0.645865  3.076794
MaxHR       0.803417  2.526232

5. КОДИРОВАНИЕ КАТЕГОРИАЛЬНЫХ ПРИЗНАКОВ

Уникальные значения до кодирования:
Sex: ['M' 'F']
ChestPainType: ['ATA' 'NAP' 'ASY' 'TA']
ExerciseAngina: ['N' 'Y']
RestingECG: ['Normal' 'ST' 'LVH']

Первые 5 строк после кодирования:
        Age  RestingBP  Cholesterol  FastingBS     MaxHR  Oldpeak ST_Slope  \
0 -1.341598   0.541580     0.984601        0.0  1.295650      0.0       Up
1 -0.398865   1.838897    -1.187295        0.0  0.639339      1.0     Flat
2 -1.655842  -0.107079     0.865047        0.0 -1.739787      0.0       Up
3 -0.503613   0.411848    -0.509823        0.0 -1.329593      1.5     Flat
4  0.124875   1.190239    -0.888410        0.0 -0.755321      0.0       Up

   HeartDisease  Sex_M  ChestPainType_ATA  ChestPainType_NAP  \
0           0.0    1.0                1.0                0.0
1           1.0    0.0                0.0                1.0
2           0.0    1.0                1.0                0.0
3           1.0    0.0                0.0                0.0
4           0.0    1.0                0.0                1.0

   ChestPainType_TA  ExerciseAngina_Y  RestingECG_Normal  RestingECG_ST
0               0.0               0.0                1.0            0.0
1               0.0               0.0                1.0            0.0
2               0.0               0.0                0.0            1.0
3               0.0               1.0                1.0            0.0
4               0.0               0.0                1.0            0.0
```

Рисунок 14 – Комплексный EDA

Задание 15. Индивидуальное задание

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.

```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler, LabelEncoder, OneHotEncoder
from sklearn.compose import ColumnTransformer

df = pd.read_csv("heart.csv")
print("Первые 5 строк:")
display(df.head())
print("\nИнформация о данных:")
display(df.info())
print("\nОписательная статистика числовых признаков:")
display(df.describe())
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)]
print(f"Размер до удаления выбросов: {df.shape}")
df_clean = df.copy()
for col in ['RestingBP', 'Cholesterol', 'MaxHR']:
    df_clean = remove_outliers(df_clean, col)
print(f"Размер после удаления выбросов: {df_clean.shape}")
numerical_cols = ['Age', 'RestingBP', 'Cholesterol', 'MaxHR', 'Oldpeak']
scaler = StandardScaler()
df_clean[numerical_cols] = scaler.fit_transform(df_clean[numerical_cols])
print("\nПосле масштабирования:")
display(df_clean.head())
categorical_cols = ['Sex', 'ChestPainType', 'RestingECG', 'ExerciseAngina', 'ST_Slope']
df_encoded = pd.get_dummies(df_clean, columns=categorical_cols, drop_first=True)
print("\nПосле One-Hot Encoding:")
display(df_encoded.head())
print("Итоговый датасет (готов для модели):")
display(df_encoded.head())
print(f"\nРазмерность: {df_encoded.shape}")
```

```
print(f"\nРазмерность: {df_encoded.shape}")
sns.countplot(x='HeartDisease', data=df)
plt.title("Распределение HeartDisease")
plt.show()
plt.figure(figsize=(12, 8))
sns.heatmap(df_encoded.corr(), annot=False, cmap='coolwarm')
plt.title("Корреляционная матрица")
plt.show()
```

Первые 5 строк:

	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 данных:
<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
```

Описательная статистика числовых признаков:

	Age	RestingBP	Cholesterol	FastingBS	MaxHR	Oldpeak	HeartDisease
count	918.000000	918.000000	918.000000	918.000000	918.000000	918.000000	918.000000
mean	53.510893	132.396514	198.799564	0.233115	136.809368	0.887364	0.553377
std	9.432617	18.514154	109.384145	0.423046	25.460334	1.066570	0.497414
min	28.000000	0.000000	0.000000	0.000000	60.000000	-2.600000	0.000000
25%	47.000000	120.000000	173.250000	0.000000	120.000000	0.000000	0.000000
50%	54.000000	130.000000	223.000000	0.000000	138.000000	0.600000	1.000000
75%	60.000000	140.000000	267.000000	0.000000	156.000000	1.500000	1.000000
max	77.000000	200.000000	603.000000	1.000000	202.000000	6.200000	1.000000

```
Размер до удаления выбросов: (918, 12)
Размер после удаления выбросов: (713, 12)
```

После масштабирования:

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope	HeartDisease
0	-1.341598	M	ATA	0.541580	0.984601	0	Normal	1.295650	N	-0.838056	Up	0
1	-0.398865	F	NAP	1.838897	-1.187295	0	Normal	0.639339	N	0.103684	Flat	1
2	-1.655842	M	ATA	-0.107079	0.865047	0	ST	-1.739787	N	-0.838056	Up	0
3	-0.503613	F	ASY	0.411848	-0.509823	0	Normal	-1.329593	Y	0.574554	Flat	1
4	0.124875	M	NAP	1.190239	-0.888410	0	Normal	-0.755321	N	-0.838056	Up	0

После One-Hot Encoding:

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

Итоговый датасет (готов для модели):

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

Размерность: (713, 16)

Рисунок 15 – Индивидуальное задание

https://github.com/GoncharovSerafim/

Ответы на контрольные вопросы:

1. Проблемы из-за пропущенных значений

Искажение статистик (среднее, дисперсия).

Ошибки в моделях (например, NaN не обрабатываются алгоритмами).

Уменьшение размера данных при .dropna().

2. Определение пропущенных значений в pandas

df.isna().sum() # Количество пропусков по столбцам

df.isnull().any() # Есть ли хотя бы один пропуск

3. Метод .dropna()

Удаляет строки/столбцы с NaN. Параметры:

axis=0 (строки) или axis=1 (столбцы).

how='any' (любой NaN) или 'all' (все значения NaN).

subset=['столбец'] — проверка только указанных столбцов.

Пример:

df.dropna(axis=0, how='any', subset=['Возраст'])

4. Заполнение средним, медианой, модой

Среднее — подходит для нормального распределения (чувствительно к выбросам).

Медиана — устойчива к выбросам.

Мода — для категориальных данных.

Пример:

df['Возраст'].fillna(df['Возраст'].median(), inplace=True)

5. Метод fillna(method='ffill')

Заполняет пропуски предыдущим значением (forward fill).

Применение: Временные ряды или данные с естественным порядком.

Пример:

df.fillna(method='ffill', inplace=True)

6. Метод .interpolate()

Заполняет пропуски, интерполируя значения (линейно, полиномиально и т.д.).

Отличие от fillna: Учитывает соседние значения, а не просто повторяет их.

Пример:

df['Цена'].interpolate(method='linear', inplace=True)

7. Выбросы и их влияние

Выбросы — аномальные значения, далекие от основного распределения.

Проблемы:

Искажают статистики (среднее, дисперсию).

Влияют на работу моделей (например, линейную регрессию).

8. Метод IQR (Interquartile Range)

Суть: Выбросы — значения за пределами:

Нижняя граница: Q1 - 1.5 * IQR

Верхняя граница: Q3 + 1.5 * IQR

где IQR = Q3 - Q1 (разница между 75% и 25% квантилями).

Пример:

Q1 = df['Возраст'].quantile(0.25)

Q3 = df['Возраст'].quantile(0.75)

IQR = Q3 - Q1

df = df[(df['Возраст'] >= Q1 - 1.5*IQR) & (df['Возраст'] <= Q3 + 1.5*IQR)]

9. Границы IQR и фильтрация

lower_bound = Q1 - 1.5 * IQR

upper_bound = Q3 + 1.5 * IQR

df_clean = df[(df['Признак'] >= lower_bound) & (df['Признак'] <= upper_bound)]

10. Метод .clip()

Ограничивает значения указанными границами:

df['Признак'].clip(lower=lower_bound, upper=upper_bound, inplace=True)

Применение: Альтернатива удалению выбросов.

11. Логарифмическое преобразование

Зачем: Сжимает большие значения, уменьшает влияние выбросов.

Пример:

import numpy as np

df['Зарплата'] = np.log1p(df['Зарплата'])

12. Графические методы для обнаружения выбросов

Boxplot (визуализация IQR).

Scatter plot (точечный график).

Пример:

import seaborn as sns

sns.boxplot(x=df['Возраст'])

13. Осторожность при удалении выбросов

Можно потерять важные аномалии (например, мошеннические операции).

Риск уменьшения размера данных.

14. Зачем масштабировать признаки?

Алгоритмы (KNN, SVM, градиентный спуск) чувствительны к масштабу.

Ускоряет сходимость моделей.

15. Стандартизация vs Нормализация

Стандартизация (StandardScaler): (x - mean) / std → среднее = 0, дисперсия = 1.

Нормализация (MinMaxScaler): (x - min) / (max - min) → диапазон [0, 1].

16. StandardScaler

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
df_scaled = scaler.fit_transform(df[['Признак']])
```

17. MinMaxScaler

Когда использовать: Когда границы значений известны (например, изображения [0, 255]).

Пример:

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
df_scaled = scaler.fit_transform(df[['Признак']])
```

18. RobustScaler

Использует медиану и IQR вместо среднего/стандартного отклонения. Устойчив к выбросам.

Пример:

```
from sklearn.preprocessing import RobustScaler
scaler = RobustScaler()
df_scaled = scaler.fit_transform(df[['Признак']])
```

19. Стандартизация вручную

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

20. Модели, чувствительные к масштабу

Линейная регрессия.

Метод k-ближайших соседей (KNN).

Нейронные сети.

SVM.

21. Преобразование категориальных признаков

Модели работают только с числами. Категории нужно перевести в числовой формат.

22. Порядковый признак

Имеет естественный порядок. Пример: Оценка [1, 2, 3, 4, 5].

23. Номинальный признак

Без порядка. Пример: Цвет ['Красный', 'Синий'].

24. Метод .factorize()

Преобразует категории в числа:

df['Цвет_код'] = pd.factorize(df['Цвет'])[0]

Применение: Для номинальных данных без порядка.

25. Метод .map() для порядковых данных

order = {'Низкий': 0, 'Средний': 1, 'Высокий': 2}

df['Уровень'] = df['Уровень'].map(order)

26. OrdinalEncoder из scikit-learn

Аналог .map(), но для нескольких столбцов:

from sklearn.preprocessing import OrdinalEncoder

encoder = OrdinalEncoder()

df[['Признак']] = encoder.fit_transform(df[['Признак']])

27. One-Hot кодирование

Создает бинарные столбцы для каждой категории. Применение: Для номинальных данных.

Пример:

pd.get_dummies(df, columns=['Цвет'])

28. Как избежать дамми-ловушки

Удалить один столбец (например, drop_first=True в pd.get_dummies).

29. OneHotEncoder vs pd.get_dummies

OneHotEncoder — из sklearn, работает в пайплайнах.

pd.get_dummies — проще, но не сохраняет признаки для новых данных.

30. Target Encoding

Замена категории средним значением целевой переменной.

Риски: Переобучение (утечка информации).

Пример:

df['Город_код'] = df.groupby('Город')['Целевая'].transform('mean')

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