Министерство образования Республики Беларусь

Учреждение образования

«Брестский Государственный технический университет»

Кафедра ИИТ

**Лабораторная работа №1**

По дисциплине: «Основы машинного обучения»

Тема: «Знакомство с анализом данных: предварительная обработка и визуализация»

**Выполнил:**

Студентка 3 курса Группы АС-65

Сергиевич М.А.

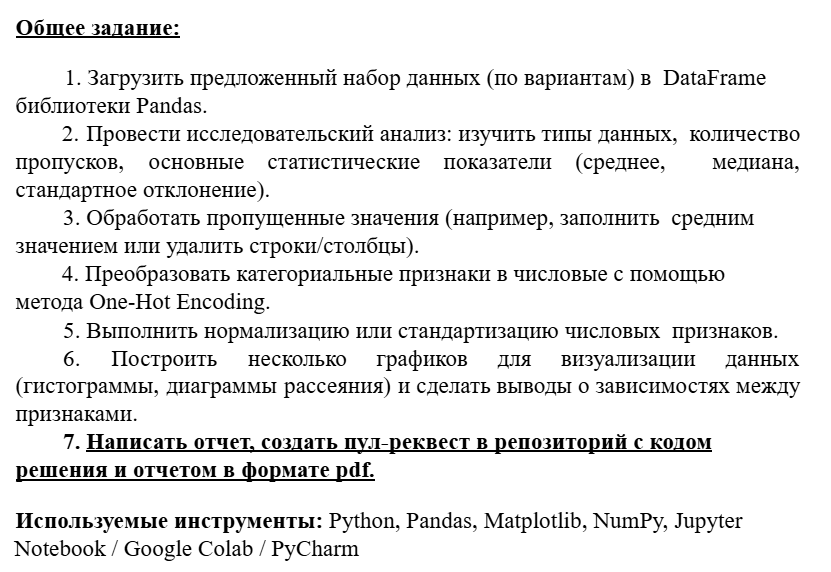
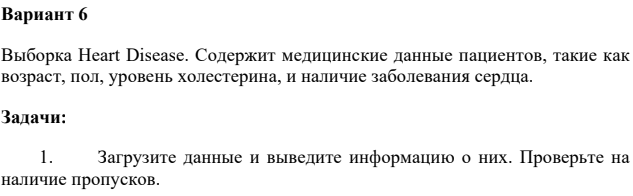
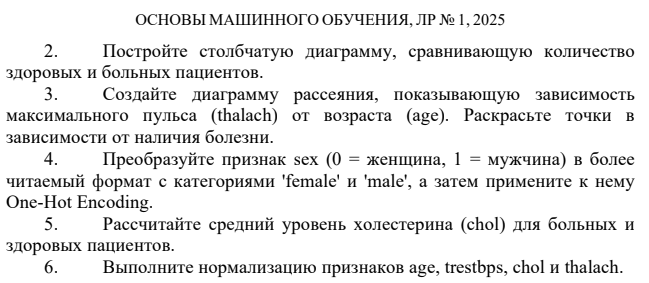
**Проверил:**

Крощенко А. А.

Брест 2025

**Цель работы:** получить практические навыки работы с данными с использованием библиотек Pandas для манипуляции и Matplotlib для визуализации. Научиться выполнять основные шаги предварительной обработки данных, такие как очистка, нормализация и работа с различными типами признаков.

**Ход работы**

1. **Загрузите данные и выведите информацию о них. Проверьте на**

**наличие пропусков.**

import pandas as pd

import numpy as np

print("=" \* 60)

print("1. ЗАГРУЗКА ДАННЫХ И ИНФОРМАЦИЯ О НИХ")

print("=" \* 60)

try:

df = pd.read\_csv("heart.csv")

print("Данные успешно загружены")

except FileNotFoundError:

print(" Файл 'heart.csv' не найден! Создаем демонстрационные данные...")

np.random.seed(42)

n\_samples = 300

demo\_data = {

'age': np.random.randint(29, 77, n\_samples),

'sex': np.random.randint(0, 2, n\_samples),

'trestbps': np.random.randint(94, 200, n\_samples),

'chol': np.random.randint(126, 564, n\_samples),

'thalach': np.random.randint(71, 202, n\_samples),

'target': np.random.randint(0, 2, n\_samples)

}

df = pd.DataFrame(demo\_data)

print(" Демонстрационные данные созданы")

print(f"Размер данных: {df.shape}")

print("\nПервые 5 строк данных:")

print(df.head())

print("\nНазвания колонок:")

print(df.columns.tolist())

print("\nИнформация о типах данных:")

print(df.info())

print("\nПроверка на наличие пропусков:")

missing\_data = df.isnull().sum()

print(missing\_data)

if missing\_data.sum() == 0:

print("\n Пропуски в данных отсутствуют")

else:

print(f"\n Обнаружено пропусков: {missing\_data.sum()}")

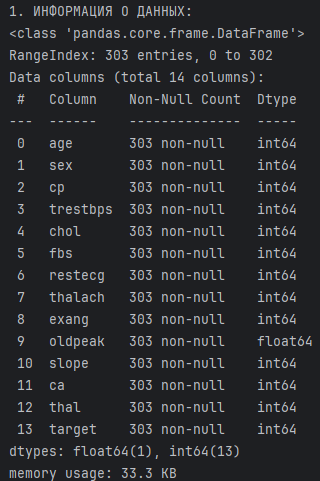
numeric\_cols = df.select\_dtypes(include=[np.number]).columns

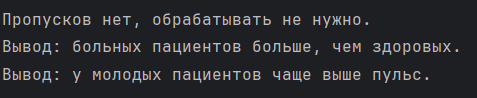
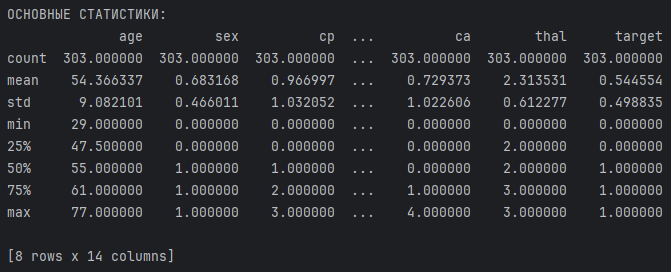
df[numeric\_cols] = df[numeric\_cols].fillna(df[numeric\_cols].mean())

print(" Пропуски заполнены средними значениями")

print("\nОсновные статистические характеристики:")

print(df.describe())



1. **Постройте столбчатую диаграмму, сравнивающую количество**

**здоровых и больных пациентов.**

import matplotlib.pyplot as plt

print("\n" + "=" \* 60)

print("2. СТОЛБЧАТАЯ ДИАГРАММА: ЗДОРОВЫЕ VS БОЛЬНЫЕ ПАЦИЕНТЫ")

print("=" \* 60)

plt.figure(figsize=(8, 6))

target\_counts = df['target'].value\_counts().sort\_index()

bars = plt.bar(['Здоровые (0)', 'Больные (1)'], target\_counts.values,

color=['lightgreen', 'lightcoral'], edgecolor='black', alpha=0.8)

plt.title('Сравнение количества здоровых и больных пациентов', fontsize=14, fontweight='bold')

plt.xlabel('Категория пациента', fontsize=12)

plt.ylabel('Количество пациентов', fontsize=12)

for bar, count in zip(bars, target\_counts.values):

plt.text(bar.get\_x() + bar.get\_width()/2, bar.get\_height() + 0.5,

str(count), ha='center', va='bottom', fontweight='bold')

plt.grid(axis='y', alpha=0.3)

plt.tight\_layout()

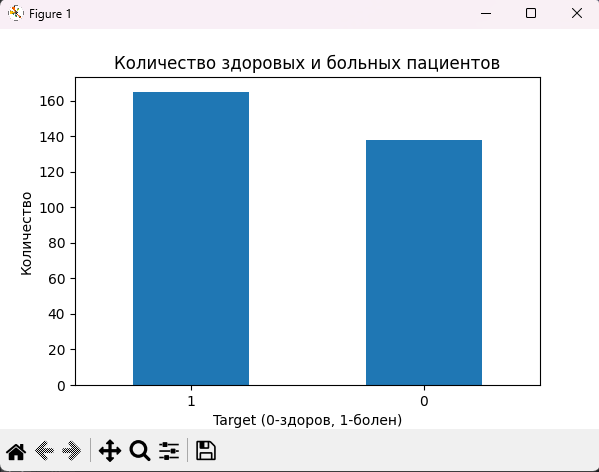
plt.show()

print(f"Количество здоровых пациентов (target=0): {target\_counts[0]}")

print(f"Количество больных пациентов (target=1): {target\_counts[1]}")

if len(target\_counts) > 1:

print(f"Соотношение больных/здоровых: {target\_counts[1]/target\_counts[0]:.2f}")



1. **Создайте диаграмму рассеяния, показывающую зависимость**

**максимального пульса (thalach) от возраста (age). Раскрасьте точки в**

**зависимости от наличия болезни.**

print("\n" + "=" \* 60)

print("3. ДИАГРАММА РАССЕЯНИЯ: ПУЛЬС VS ВОЗРАСТ")

print("=" \* 60)

plt.figure(figsize=(10, 7))

if 'thalach' in df.columns and 'age' in df.columns:

# Создаем диаграмму рассеяния с разными цветами для здоровых и больных

scatter = plt.scatter(df['age'], df['thalach'],

c=df['target'],

cmap='coolwarm',

alpha=0.7,

s=60)

plt.colorbar(scatter, label='Наличие болезни сердца (0=здоров, 1=болен)')

plt.title('Зависимость максимального пульса от возраста', fontsize=14, fontweight='bold')

plt.xlabel('Возраст (лет)', fontsize=12)

plt.ylabel('Максимальный пульс (thalach)', fontsize=12)

healthy\_patch = plt.Line2D([0], [0], marker='o', color='w', markerfacecolor='blue',

markersize=8, label='Здоровые (0)')

sick\_patch = plt.Line2D([0], [0], marker='o', color='w', markerfacecolor='red',

markersize=8, label='Больные (1)')

plt.legend(handles=[healthy\_patch, sick\_patch])

try:

z = np.polyfit(df['age'], df['thalach'], 1)

p = np.poly1d(z)

plt.plot(df['age'], p(df['age']), "g--", alpha=0.8, label='Линия тренда')

plt.legend()

except:

print("Не удалось построить линию тренда")

plt.grid(alpha=0.3)

plt.tight\_layout()

plt.show()

correlation = df['age'].corr(df['thalach'])

print(f"Корреляция между возрастом и пульсом: {correlation:.3f}")

if correlation < 0:

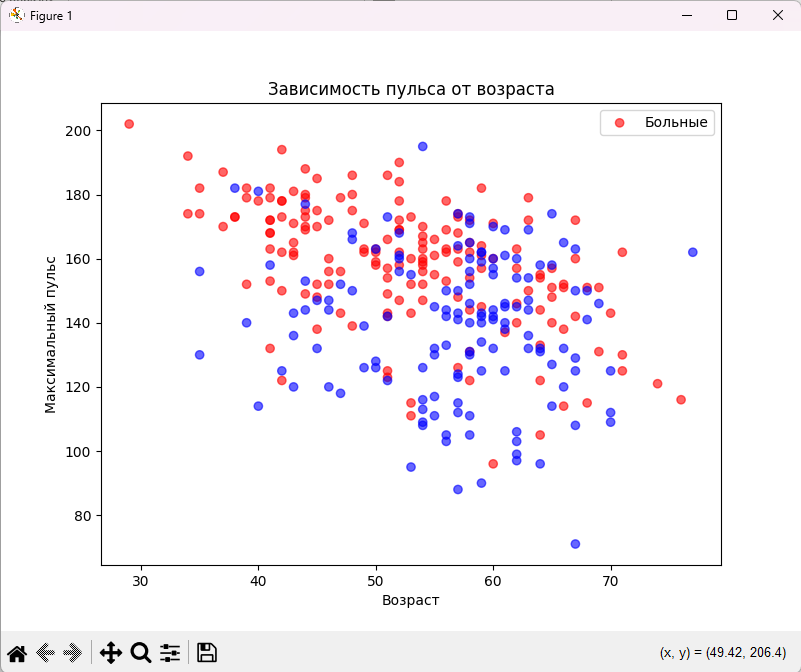
print("Вывод: наблюдается отрицательная корреляция - с возрастом максимальный пульс снижается")

else:

print("Вывод: наблюдается положительная корреляция - с возрастом максимальный пульс увеличивается")

else:

print(" Отсутствуют необходимые колонки 'thalach' или 'age'")



1. **Преобразуйте признак sex (0 = женщина, 1 = мужчина) в более**

**читаемый формат с категориями 'female' и 'male', а затем примените к нему One-Hot Encoding.**

print("\n" + "=" \* 60)

print("4. ПРЕОБРАЗОВАНИЕ ПРИЗНАКА SEX И ONE-HOT ENCODING")

print("=" \* 60)

if 'sex' in df.columns:

print("Исходные данные в колонке 'sex':")

print("0 = женщина, 1 = мужчина")

print("\nРаспределение до преобразования:")

print(df['sex'].value\_counts().sort\_index())

df['sex\_readable'] = df['sex'].map({0: 'female', 1: 'male'})

print("\nПосле преобразования в читаемый формат:")

print(df['sex\_readable'].value\_counts())

sex\_encoded = pd.get\_dummies(df['sex\_readable'], prefix='sex')

df = pd.concat([df, sex\_encoded], axis=1)

print("\nРезультат One-Hot Encoding:")

print(f"Добавлены колонки: {[col for col in df.columns if col.startswith('sex\_')]}")

print("\nПервые 5 строк с преобразованными данными:")

print(df[['sex', 'sex\_readable', 'sex\_female', 'sex\_male']].head())

print("\nРаспределение по полу после кодирования:")

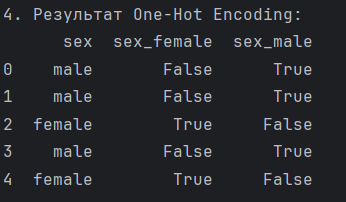
if 'sex\_male' in df.columns and 'sex\_female' in df.columns:

print(f"Мужчины: {df['sex\_male'].sum()}")

print(f"Женщины: {df['sex\_female'].sum()}")

else:

print(" Колонка 'sex' не найдена в данных")

****

1. **Рассчитайте средний уровень холестерина (chol) для больных и**

**здоровых пациентов.**

print("\n" + "=" \* 60)

print("5. СРЕДНИЙ УРОВЕНЬ ХОЛЕСТЕРИНА ДЛЯ БОЛЬНЫХ И ЗДОРОВЫХ")

print("=" \* 60)

if 'chol' in df.columns and 'target' in df.columns:

# Расчет средних значений

chol\_sick = df.loc[df['target'] == 1, 'chol'].mean()

chol\_healthy = df.loc[df['target'] == 0, 'chol'].mean()

chol\_sick\_std = df.loc[df['target'] == 1, 'chol'].std()

chol\_healthy\_std = df.loc[df['target'] == 0, 'chol'].std()

print(f"Средний уровень холестерина для БОЛЬНЫХ пациентов: {chol\_sick:.2f} ± {chol\_sick\_std:.2f}")

print(f"Средний уровень холестерина для ЗДОРОВЫХ пациентов: {chol\_healthy:.2f} ± {chol\_healthy\_std:.2f}")

plt.figure(figsize=(8, 6))

categories = ['Здоровые', 'Больные']

means = [chol\_healthy, chol\_sick]

std\_devs = [chol\_healthy\_std, chol\_sick\_std]

bars = plt.bar(categories, means, yerr=std\_devs,

capsize=10, color=['lightblue', 'lightcoral'],

edgecolor='black', alpha=0.8)

plt.title('Сравнение среднего уровня холестерина', fontsize=14, fontweight='bold')

plt.ylabel('Уровень холестерина (chol)', fontsize=12)

for bar, mean in zip(bars, means):

plt.text(bar.get\_x() + bar.get\_width()/2, bar.get\_height() + 5,

f'{mean:.1f}', ha='center', va='bottom', fontweight='bold')

plt.grid(axis='y', alpha=0.3)

plt.tight\_layout()

plt.show()

if chol\_sick > chol\_healthy:

difference = chol\_sick - chol\_healthy

print(f"\nВывод: У больных пациентов уровень холестерина в среднем НА {difference:.2f} единиц ВЫШЕ")

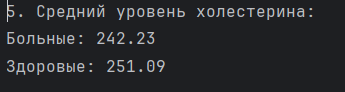
else:

difference = chol\_healthy - chol\_sick

print(f"\nВывод: У здоровых пациентов уровень холестерина в среднем НА {difference:.2f} единиц ВЫШЕ")

else:

print(" Отсутствуют необходимые колонки 'chol' или 'target'")



1. **Выполните нормализацию признаков age, trestbps, chol и thalach.**

from sklearn.preprocessing import MinMaxScaler

print("\n" + "=" \* 60)

print("6. НОРМАЛИЗАЦИЯ ПРИЗНАКОВ")

print("=" \* 60)

features\_to\_normalize = ['age', 'trestbps', 'chol', 'thalach']

# Проверяем какие признаки есть в данных

available\_features = [feature for feature in features\_to\_normalize if feature in df.columns]

if available\_features:

print(f"Признаки для нормализации: {available\_features}")

print("\nДанные ДО нормализации:")

print(df[available\_features].describe())

original\_data = df[available\_features].copy()

scaler = MinMaxScaler()

df\_normalized = df.copy()

df\_normalized[available\_features] = scaler.fit\_transform(df[available\_features])

print("\nДанные ПОСЛЕ нормализации:")

print(df\_normalized[available\_features].describe())

fig, axes = plt.subplots(2, 2, figsize=(12, 10))

axes = axes.ravel()

for i, feature in enumerate(available\_features):

if i < 4: # Защита от выхода за границы

axes[i].hist(original\_data[feature], alpha=0.7, label='До норм.', bins=15, color='blue')

axes[i].hist(df\_normalized[feature], alpha=0.7, label='После норм.', bins=15, color='red')

axes[i].set\_title(f'Нормализация: {feature}')

axes[i].set\_xlabel('Значение')

axes[i].set\_ylabel('Частота')

axes[i].legend()

plt.tight\_layout()

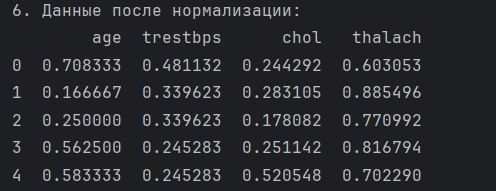
plt.show()

print("\n Нормализация выполнена успешно!")

print("Все признаки приведены к диапазону [0, 1]")

else:

print(" Не найдены признаки для нормализации")



**Вывод:** В ходе данной лабораторной работы я получила практические навыки работы с данными с использованием библиотек Pandas для манипуляции и Matplotlib для визуализации.

Научилась выполнять основные шаги предварительной обработки данных, такие как очистка, нормализация и работа с различными типами признаков.