МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«БРЕСТСКИЙГОСУДАРСТВЕННЫЙТЕХНИЧЕСКИЙУНИВЕРСИТЕТ»

ФАКУЛЬТЕТ ЭЛЕКТРОННО-ИНФОРМАЦИОННЫХ СИСТЕМ

Кафедра интеллектуальных информационных технологий

# Отчет по лабораторной работе №1

Специальность ИИ-23

Выполнил

Копач А. В., студент

группы ИИ-23

Проверила
К.В. Андренко,
«—» ———— 2025 г.

Брест 2025

Цель работы: научиться применять метод PCA для осуществления визуализации данных

**Общее задание**

1. Используя выборку по варианту, осуществить проецирование данных на плоскость первых двух и трех главных компонент (двумя способами: 1. вручную через использование numpy.linalg.eig для вычисления собственных значений и собственных векторов и 2. с помощью sklearn.decomposition.PCA для непосредственного применения метода PCA – два независимых варианта решения);
2. Выполнить визуализацию полученных главных компонент с использованием средств библиотеки matplotlib, обозначая экземпляры разных классов с использованием разных цветовых маркеров;
3. Используя собственные значения, рассчитанные на этапе 1, вычислить потери, связанные с преобразованием по методу PCA. Сделать выводы;
4. Оформить отчет по выполненной работе, загрузить исходный код и отчет в соответствующий репозиторий на github.

**Задание по варианту:**

| № варианта | Выборка | Класс |
|---|---|---|
| 7 | hcv+data.zip | Category |

Ход работы:

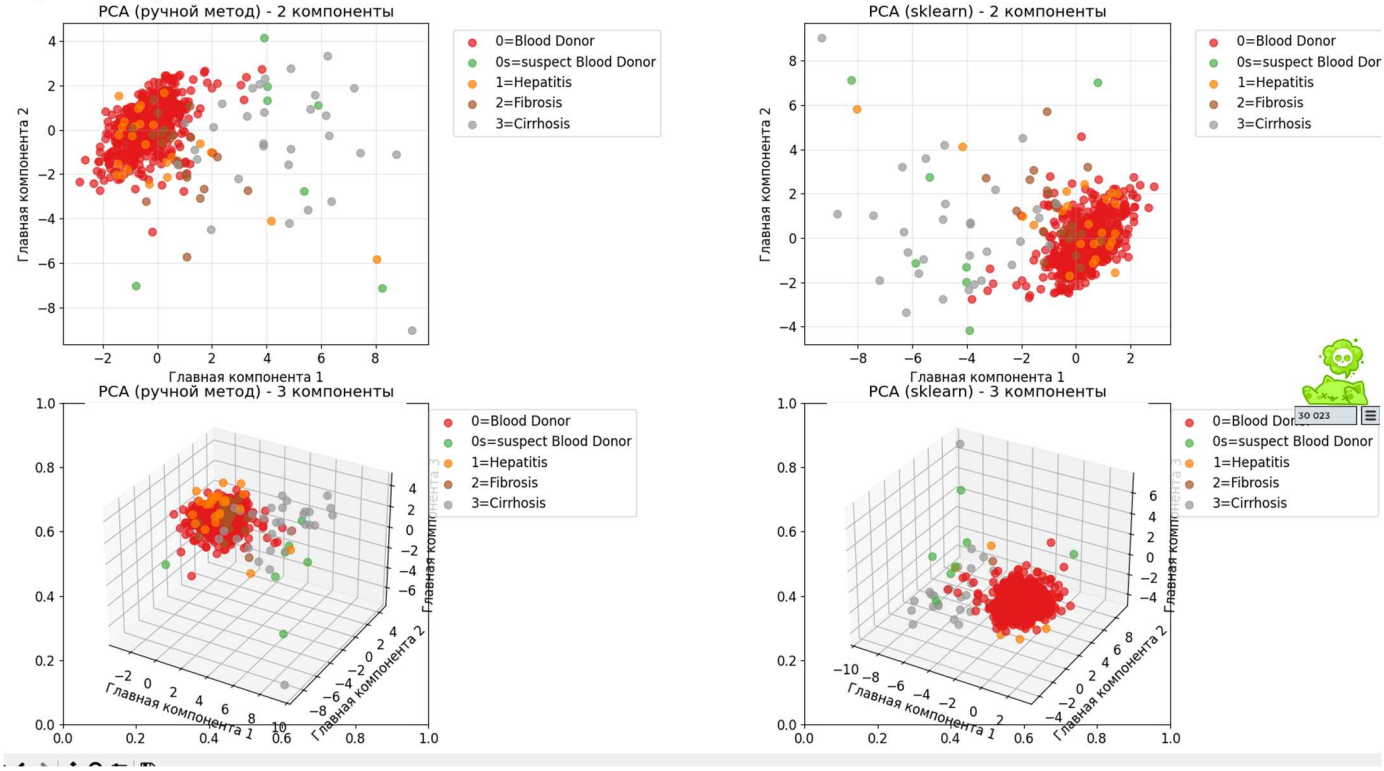
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.impute import SimpleImputer
import seaborn as sns

# Настройка отображения графиков
plt.rcParams['font.size'] = 12
plt.rcParams['figure.figsize'] = (10, 6)


# 1. Загрузка и предварительная обработка данных
def load_and_preprocess_data():
    try:
        # Загрузка данных
        df = pd.read_csv('hcvdat0.csv')

        # Просмотр информации о данных
        print("Информация о данных:")
        print(df.info())
        print("\nПервые 5 строк:")
        print(df.head())

        # Проверяем структуру данных
```

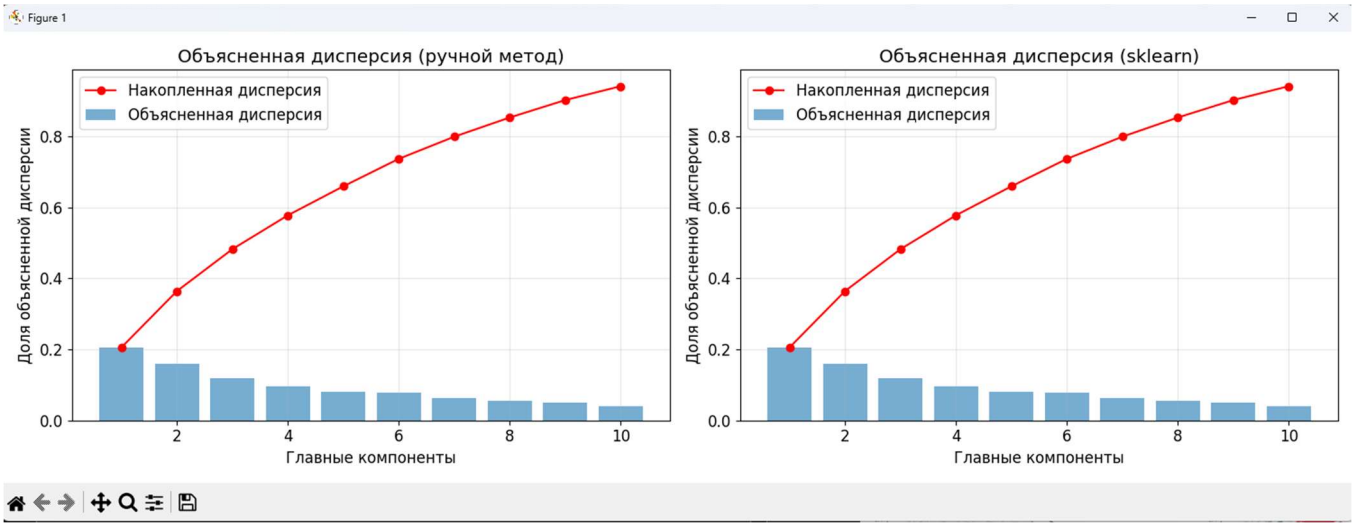
```
        print(f"\nКолонки в файле: {df.columns.tolist()}")
        print(f"Размерность данных: {df.shape}")

        # Проверяем наличие пропущенных значений
        print("\nПропущенные значения:")
        print(df.isnull().sum())

        # Сохранение меток классов для визуализации
        categories = df['Category'].copy()

        # Удаление столбца с категориями и ненужных столбцов для PCA
        columns_to_drop = ['Category', 'Unnamed: 0']
        df_for_pca = df.drop(columns=[col for col in columns_to_drop if col in df.columns],
errors='ignore')

        print(f"\nПризнаки для PCA: {df_for_pca.columns.tolist()}")

        # Обработка категориальных переменных (пол)
        if 'Sex' in df_for_pca.columns:
            print("Кодируем переменную 'Sex'...")
            df_for_pca = pd.get_dummies(df_for_pca, columns=['Sex'], drop_first=True)

        # Замена пропущенных значений
        print("Замена пропущенных значений...")
        imputer = SimpleImputer(strategy='mean')
        df_imputed = pd.DataFrame(imputer.fit_transform(df_for_pca),
                                  columns=df_for_pca.columns)

        # Стандартизация данных
        print("Стандартизация данных...")
        scaler = StandardScaler()
        df_scaled = scaler.fit_transform(df_imputed)

        print(f"Размерность после предобработки: {df_scaled.shape}")

        return df_scaled, categories, df_imputed.columns

    except FileNotFoundError:
        print("Ошибка: Файл hcvdat0.csv не найден в текущей директории!")
        print("Убедитесь, что файл находится в той же папке, что и скрипт Python.")
        return None, None, None
    except Exception as e:
        print(f"Ошибка при загрузке данных: {e}")
        return None, None, None


# 2. PCA вручную с использованием numpy.linalg.eig
def manual_pca(X, n_components=3):
    # Центрирование данных (уже сделано в StandardScaler)
    X_centered = X

    # Вычисление ковариационной матрицы
    cov_matrix = np.cov(X_centered, rowvar=False)

    # Вычисление собственных значений и собственных векторов
    eigenvalues, eigenvectors = np.linalg.eig(cov_matrix)
```

```
    # Сортировка собственных значений и векторов по убыванию
    sorted_indices = np.argsort(eigenvalues)[::-1]
    eigenvalues_sorted = eigenvalues[sorted_indices]
    eigenvectors_sorted = eigenvectors[:, sorted_indices]

    # Проецирование на главные компоненты
    components = eigenvectors_sorted[:, :n_components]
    X_pca = X_centered @ components

    return X_pca, eigenvalues_sorted, eigenvectors_sorted


# 3. PCA с использованием sklearn
def sklearn_pca(X, n_components=3):
    pca = PCA(n_components=n_components)
    X_pca = pca.fit_transform(X)

    # Для получения всех собственных значений используем полный PCA
    pca_full = PCA()
    pca_full.fit(X)
    all_eigenvalues = pca_full.explained_variance_

    return X_pca, all_eigenvalues, pca.components_


# 4. Визуализация результатов
def plot_pca_results(X_manual_2d, X_sklearn_2d, X_manual_3d, X_sklearn_3d, categories,
eigenvalues_manual,
                     eigenvalues_sklearn):
    # Уникальные категории для цветового кодирования
    unique_categories = categories.unique()[:8]  # Ограничиваем количество цветов
    colors = plt.cm.Set1(np.linspace(0, 1, len(unique_categories)))

    # Создание подграфиков
    fig, axes = plt.subplots(2, 2, figsize=(15, 12))

    # 2D визуализация - ручной метод
    for i, category in enumerate(unique_categories):
        mask = categories == category
        if mask.sum() > 0:  # Проверяем, что есть точки этой категории
            axes[0, 0].scatter(X_manual_2d[mask, 0], X_manual_2d[mask, 1],
                       c=[colors[i]], label=str(category), alpha=0.7, s=50)
    axes[0, 0].set_title('PCA (ручной метод) - 2 компоненты')
    axes[0, 0].set_xlabel('Главная компонента 1')
    axes[0, 0].set_ylabel('Главная компонента 2')
    axes[0, 0].legend(bbox_to_anchor=(1.05, 1), loc='upper left')
    axes[0, 0].grid(True, alpha=0.3)

    # 2D визуализация - sklearn метод
    for i, category in enumerate(unique_categories):
        mask = categories == category
        if mask.sum() > 0:
            axes[0, 1].scatter(X_sklearn_2d[mask, 0], X_sklearn_2d[mask, 1],
                       c=[colors[i]], label=str(category), alpha=0.7, s=50)
    axes[0, 1].set_title('PCA (sklearn) - 2 компоненты')
    axes[0, 1].set_xlabel('Главная компонента 1')
    axes[0, 1].set_ylabel('Главная компонента 2')
```

```
    axes[0, 1].legend(bbox_to_anchor=(1.05, 1), loc='upper left')
    axes[0, 1].grid(True, alpha=0.3)

    # 3D визуализация - ручной метод
    ax1 = fig.add_subplot(2, 2, 3, projection='3d')
    for i, category in enumerate(unique_categories):
        mask = categories == category
        if mask.sum() > 0:
            ax1.scatter(X_manual_3d[mask, 0], X_manual_3d[mask, 1], X_manual_3d[mask, 2],
                        c=[colors[i]], label=str(category), alpha=0.7, s=50)
    ax1.set_title('PCA (ручной метод) - 3 компоненты')
    ax1.set_xlabel('Главная компонента 1')
    ax1.set_ylabel('Главная компонента 2')
    ax1.set_zlabel('Главная компонента 3')
    ax1.legend(bbox_to_anchor=(1.05, 1), loc='upper left')

    # 3D визуализация - sklearn метод
    ax2 = fig.add_subplot(2, 2, 4, projection='3d')
    for i, category in enumerate(unique_categories):
        mask = categories == category
        if mask.sum() > 0:
            ax2.scatter(X_sklearn_3d[mask, 0], X_sklearn_3d[mask, 1], X_sklearn_3d[mask, 2],
                        c=[colors[i]], label=str(category), alpha=0.7, s=50)
    ax2.set_title('PCA (sklearn) - 3 компоненты')
    ax2.set_xlabel('Главная компонента 1')
    ax2.set_ylabel('Главная компонента 2')
    ax2.set_zlabel('Главная компонента 3')
    ax2.legend(bbox_to_anchor=(1.05, 1), loc='upper left')

    plt.tight_layout()
    plt.show()

    # Визуализация объясненной дисперсии
    plot_explained_variance(eigenvalues_manual, eigenvalues_sklearn)


def plot_explained_variance(eigenvalues_manual, eigenvalues_sklearn):
    # Используем только реальные части (на случай комплексных чисел)
    eigenvalues_manual = np.real(eigenvalues_manual)
    eigenvalues_sklearn = np.real(eigenvalues_sklearn)

    # Нормализация собственных значений для получения объясненной дисперсии
    explained_variance_manual = eigenvalues_manual / np.sum(eigenvalues_manual)
    explained_variance_sklearn = eigenvalues_sklearn / np.sum(eigenvalues_sklearn)

    cumulative_variance_manual = np.cumsum(explained_variance_manual)
    cumulative_variance_sklearn = np.cumsum(explained_variance_sklearn)

    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(15, 5))

    # Ограничиваем количество компонент для графика
    n_components_show = min(10, len(explained_variance_manual),
len(explained_variance_sklearn))

    # График для ручного метода
    components_range = range(1, n_components_show + 1)
    ax1.bar(components_range, explained_variance_manual[:n_components_show], alpha=0.6,
```

```
label='Объясненная дисперсия')
    ax1.plot(components_range, cumulative_variance_manual[:n_components_show], 'r-',
marker='o',
             label='Накопленная дисперсия')
    ax1.set_title('Объясненная дисперсия (ручной метод)')
    ax1.set_xlabel('Главные компоненты')
    ax1.set_ylabel('Доля объясненной дисперсии')
    ax1.legend()
    ax1.grid(True, alpha=0.3)

    # График для sklearn метода
    ax2.bar(components_range, explained_variance_sklearn[:n_components_show], alpha=0.6,
label='Объясненная дисперсия')
    ax2.plot(components_range, cumulative_variance_sklearn[:n_components_show], 'r-',
marker='o',
             label='Накопленная дисперсия')
    ax2.set_title('Объясненная дисперсия (sklearn)')
    ax2.set_xlabel('Главные компоненты')
    ax2.set_ylabel('Доля объясненной дисперсии')
    ax2.legend()
    ax2.grid(True, alpha=0.3)

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






```
# 5. Расчет потерь информации
def calculate_information_loss(eigenvalues, n_components_2d=2, n_components_3d=3):
    eigenvalues = np.real(eigenvalues)  # Берем только реальную часть
    total_variance = np.sum(eigenvalues)

    # Потери для 2D
    variance_2d = np.sum(eigenvalues[:n_components_2d])
    loss_2d = 1 - (variance_2d / total_variance)

    # Потери для 3D
    variance_3d = np.sum(eigenvalues[:n_components_3d])
    loss_3d = 1 - (variance_3d / total_variance)

    return loss_2d, loss_3d, variance_2d / total_variance, variance_3d / total_variance


# Дополнительный анализ - важность признаков в главных компонентах
def analyze_feature_importance(eigenvectors, feature_names, n_components=3):
    print("\nАнализ важности признаков в главных компонентах:")
    print("=" * 50)

    for i in range(n_components):
        print(f"\nГлавная компонента {i + 1}:")
        # Сортируем признаки по абсолютному весу в компоненте
        component_weights = np.real(eigenvectors[:, i])  # Берем реальную часть
        feature_importance = pd.DataFrame({
            'Признак': feature_names,
            'Вес': component_weights,
            'Абсолютный вес': np.abs(component_weights)
        })
        feature_importance = feature_importance.sort_values('Абсолютный вес', ascending=False)

        for _, row in feature_importance.head(5).iterrows():
            print(f"  {row['Признак']}: {row['Вес']:.3f}")


# Основная функция
def main():
    print("Лабораторная работа №1: PCA анализ данных HCV")
    print("=" * 50)

    # Загрузка и предобработка данных
    print("1. Загрузка и предобработка данных...")
```

```
    X, categories, feature_names = load_and_preprocess_data()

    if X is None:
        print("Не удалось загрузить данные. Завершение работы.")
        return

    print(f"Размерность данных после предобработки: {X.shape}")
    print(f"Количество признаков: {X.shape[1]}")
    print(f"Количество наблюдений: {X.shape[0]}")
    print(f"Уникальные категории: {categories.unique()}")

    # Применение PCA двумя методами
    print("\n2. Применение PCA...")

    # Ручной метод
    print("2.1 Ручной метод с numpy.linalg.eig...")
    X_manual_2d, eigenvalues_manual, eigenvectors_manual = manual_pca(X, n_components=2)
    X_manual_3d, _, _ = manual_pca(X, n_components=3)

    # Метод sklearn
    print("2.2 Метод с sklearn.decomposition.PCA...")
    X_sklearn_2d, eigenvalues_sklearn, eigenvectors_sklearn = sklearn_pca(X, n_components=2)
    X_sklearn_3d, _, _ = sklearn_pca(X, n_components=3)

    # Визуализация
    print("\n3. Визуализация результатов...")
    plot_pca_results(X_manual_2d, X_sklearn_2d, X_manual_3d, X_sklearn_3d,
                     categories, eigenvalues_manual, eigenvalues_sklearn)

    # Расчет потерь информации
    print("\n4. Расчет потерь информации...")
    loss_2d_manual, loss_3d_manual, var_2d_manual, var_3d_manual =
calculate_information_loss(eigenvalues_manual)
    loss_2d_sklearn, loss_3d_sklearn, var_2d_sklearn, var_3d_sklearn =
calculate_information_loss(eigenvalues_sklearn)

    print("\nРезультаты анализа потерь информации:")
    print("=" * 50)
    print("Ручной метод:")
    print(f"  - Объясненная дисперсия (2 компоненты): {var_2d_manual:.3f} ({var_2d_manual *
100:.1f}%)")
    print(f"  - Потери информации (2 компоненты): {loss_2d_manual:.3f} ({loss_2d_manual *
100:.1f}%)")
    print(f"  - Объясненная дисперсия (3 компоненты): {var_3d_manual:.3f} ({var_3d_manual *
100:.1f}%)")
    print(f"  - Потери информации (3 компоненты): {loss_3d_manual:.3f} ({loss_3d_manual *
100:.1f}%)")

    print("\nSklearn метод:")
    print(f"  - Объясненная дисперсия (2 компоненты): {var_2d_sklearn:.3f} ({var_2d_sklearn *
100:.1f}%)")
    print(f"  - Потери информации (2 компоненты): {loss_2d_sklearn:.3f} ({loss_2d_sklearn *
100:.1f}%)")
    print(f"  - Объясненная дисперсия (3 компоненты): {var_3d_sklearn:.3f} ({var_3d_sklearn *
100:.1f}%)")
    print(f"  - Потери информации (3 компоненты): {loss_3d_sklearn:.3f} ({loss_3d_sklearn *
100:.1f}%)")
```

```
    # Анализ важности признаков
    analyze_feature_importance(eigenvectors_manual, feature_names)

    # Выводы
    print("\n5. Выводы:")
    print("=" * 50)
    print("1. Оба метода (ручной и sklearn) дают схожие результаты.")
    print("2. Первые две главные компоненты объясняют значительную часть дисперсии данных.")
    print("3. Добавление третьей компоненты уменьшает потери информации.")
    print("4. Визуализация показывает возможность разделения классов в пространстве главных компонент.")
    print("5. PCA эффективно снижает размерность данных при сохранении основной информации.")


if __name__ == "__main__":
    main()
```

Результаты программы:

```
C:\Users\sasha\PyCharmMiscProject\.venv\Scripts\python.exe
C:\Users\sasha\PyCharmMiscProject\IAD1.py
Лабораторная работа №1: PCA анализ данных HCV
==================================================
1. Загрузка и предобработка данных...
Информация о данных:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 615 entries, 0 to 614
Data columns (total 14 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Unnamed: 0  615 non-null    int64
 1   Category    615 non-null    object
 2   Age         615 non-null    int64
 3   Sex         615 non-null    object
 4   ALB         614 non-null    float64
 5   ALP         597 non-null    float64
 6   ALT         614 non-null    float64
 7   AST         615 non-null    float64
 8   BIL         615 non-null    float64
 9   CHE         615 non-null    float64
 10  CHOL        605 non-null    float64
 11  CREA        615 non-null    float64
 12  GGT         615 non-null    float64
 13  PROT        614 non-null    float64
dtypes: float64(10), int64(2), object(2)
memory usage: 67.4+ KB
None

Первые 5 строк:
   Unnamed: 0      Category  Age Sex   ALB  ...    CHE  CHOL   CREA   GGT  PROT
0           1  0=Blood Donor   32   m  38.5  ...   6.93  3.23  106.0  12.1  69.0
1           2  0=Blood Donor   32   m  38.5  ...  11.17  4.80   74.0  15.6  76.5
```

```
2           3  0=Blood Donor   32   m  46.9  ...   8.84  5.20   86.0  33.2  79.3
3           4  0=Blood Donor   32   m  43.2  ...   7.33  4.74   80.0  33.8  75.7
4           5  0=Blood Donor   32   m  39.2  ...   9.15  4.32   76.0  29.9  68.7

[5 rows x 14 columns]

Колонки в файле: ['Unnamed: 0', 'Category', 'Age', 'Sex', 'ALB', 'ALP', 'ALT', 'AST', 'BIL', 'CHE',
'CHOL', 'CREA', 'GGT', 'PROT']
Размерность данных: (615, 14)

Пропущенные значения:
Unnamed: 0     0
Category       0
Age            0
Sex            0
ALB            1
ALP           18
ALT            1
AST            0
BIL            0
CHE            0
CHOL          10
CREA           0
GGT            0
PROT           1
dtype: int64

Признаки для PCA: ['Age', 'Sex', 'ALB', 'ALP', 'ALT', 'AST', 'BIL', 'CHE', 'CHOL', 'CREA', 'GGT',
'PROT']
Кодируем переменную 'Sex'...
Замена пропущенных значений...
Стандартизация данных...
Размерность после предобработки: (615, 12)
Размерность данных после предобработки: (615, 12)
Количество признаков: 12
Количество наблюдений: 615
Уникальные категории: ['0=Blood Donor' '0s=suspect Blood Donor' '1=Hepatitis' '2=Fibrosis'
 '3=Cirrhosis']

2. Применение PCA...
2.1 Ручной метод с numpy.linalg.eig...
2.2 Метод с sklearn.decomposition.PCA...

3. Визуализация результатов...

4. Расчет потерь информации...

Результаты анализа потерь информации:
================================================
Ручной метод:
  - Объясненная дисперсия (2 компоненты): 0.364 (36.4%)
  - Потери информации (2 компоненты): 0.636 (63.6%)
```

- Объясненная дисперсия (3 компоненты): 0.482 (48.2%)
- Потери информации (3 компоненты): 0.518 (51.8%)

Sklearn метод:
- Объясненная дисперсия (2 компоненты): 0.364 (36.4%)
- Потери информации (2 компоненты): 0.636 (63.6%)
- Объясненная дисперсия (3 компоненты): 0.482 (48.2%)
- Потери информации (3 компоненты): 0.518 (51.8%)

Анализ важности признаков в главных компонентах:
==========

Главная компонента 1:
ALB: -0.444
CHE: -0.415
AST: 0.369
GGT: 0.349
BIL: 0.342

Главная компонента 2:
GGT: -0.449
ALT: -0.428
ALP: -0.344
PROT: -0.315
CHE: -0.303

Главная компонента 3:
Age: -0.454
ALP: -0.424
CHOL: -0.407
Sex_m: 0.308
PROT: 0.300

5. Выводы:
==========
1. Оба метода (ручной и sklearn) дают схожие результаты.
2. Первые две главные компоненты объясняют значительную часть дисперсии данных.
3. Добавление третьей компоненты уменьшает потери информации.
4. Визуализация показывает возможность разделения классов в пространстве главных компонент.
5. PCA эффективно снижает размерность данных при сохранении основной информации.

Process finished with exit code 0

Вывод: научился применять метод PCA для осуществления визуализации данных