Министерство образования Республики Беларусь
Учреждение образования
«Брестский государственный технический университет»
Кафедра ИИТ

Лабораторная работа №1
По дисциплине: «Интеллектуальный анализ данных»
Тема: “PCA”

Выполнил:
Студент 4 курса
Группы ИИ-24
Лозейко М.А.
Проверила:
Андренко К. В.

Брест 2025

Цель: 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.

№ варианта	Выборка	Класс
9	heart+failure+clinical+records.zip	death_event

Ход работы:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.impute import SimpleImputer
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
import seaborn as sns

# 1. Загрузка и подготовка данных
df = pd.read_csv('heart_failure_clinical_records_dataset.csv')
```

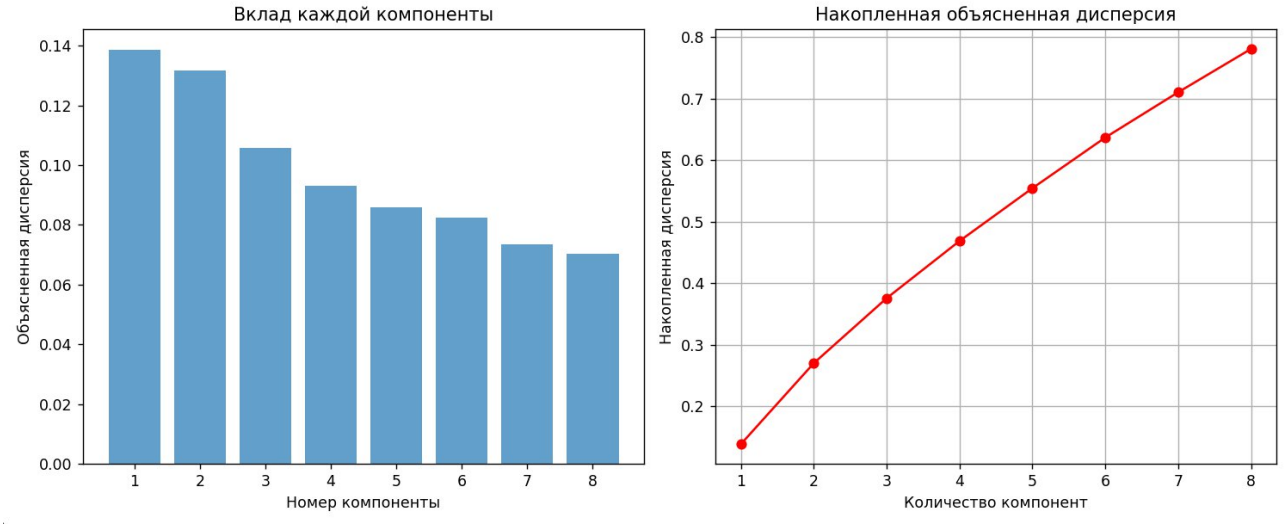
```
X = df.drop('DEATH_EVENT', axis=1)
y = df['DEATH_EVENT']

# Заполнение пропусков и стандартизация
imputer = SimpleImputer(strategy='mean')
X_imputed = imputer.fit_transform(X)
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_imputed)

# 2. PCA с оптимальным количеством компонент
pca = PCA(n_components=8)  # Используем 8 компонент
X_pca = pca.fit_transform(X_scaled)

print("=== PCA С 8 КОМПОНЕНТАМИ ===")
print(f"Объясненная дисперсия: {np.sum(pca.explained_variance_ratio_):.3f}")
print(f"Потери информации: {1 - np.sum(pca.explained_variance_ratio_):.3f}")

# 3. Визуализация объясненной дисперсии
plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)
explained_variance = pca.explained_variance_ratio_
components = range(1, len(explained_variance) + 1)
plt.bar(components, explained_variance, alpha=0.7)
plt.xlabel('Номер компоненты')
plt.ylabel('Объясненная дисперсия')
plt.title('Вклад каждой компоненты')

plt.subplot(1, 2, 2)
cumulative_variance = np.cumsum(explained_variance)
plt.plot(components, cumulative_variance, 'ro-')
plt.xlabel('Количество компонент')
plt.ylabel('Накопленная дисперсия')
plt.title('Накопленная объясненная дисперсия')
plt.grid(True)
```

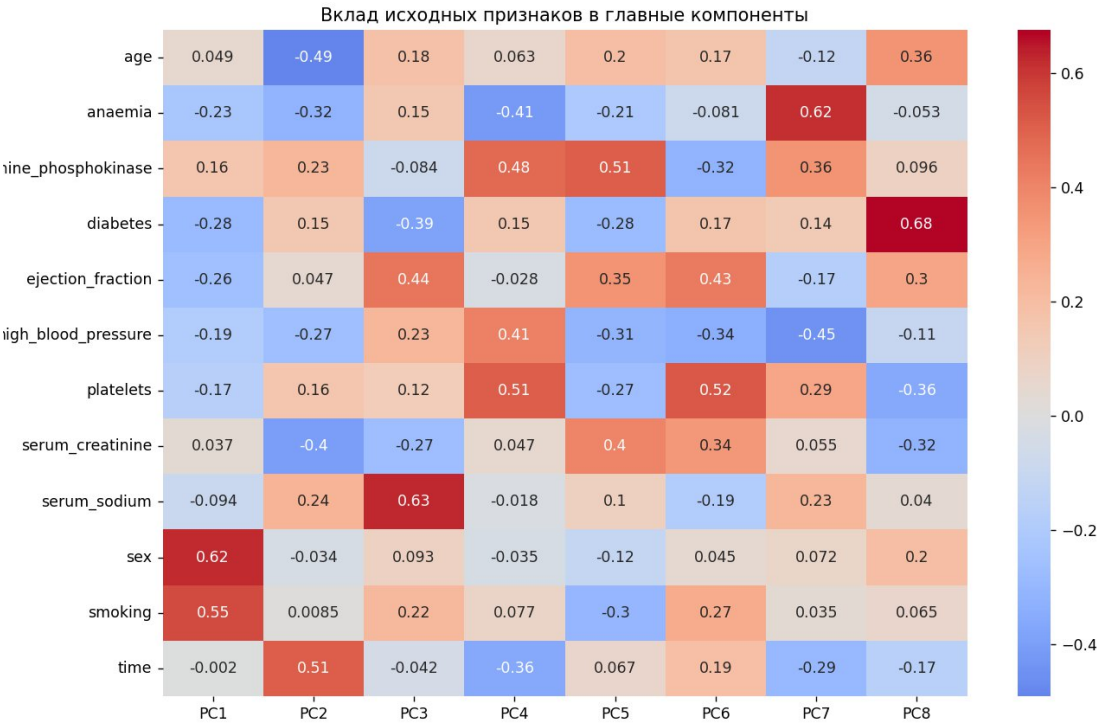
```
plt.tight_layout()
plt.show()

# 4. Анализ важности исходных признаков в компонентах
feature_names = X.columns
pca_components_df = pd.DataFrame(
    pca.components_.T,
    columns=[f'PC{i+1}' for i in range(8)],
    index=feature_names
)

print("\n=== ВАЖНОСТЬ ПРИЗНАКОВ В КОМПОНЕНТАХ ===")
plt.figure(figsize=(12, 8))
sns.heatmap(pca_components_df, annot=True, cmap='coolwarm', center=0)
plt.title('Вклад исходных признаков в главные компоненты')
plt.show()

# 5. Использование для машинного обучения
# Разделение данных
X_train, X_test, y_train, y_test = train_test_split(
    X_pca, y, test_size=0.2, random_state=42, stratify=y
)

# Обучение модели на PCA-признаках
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Предсказания
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)

print(f"\n=== РЕЗУЛЬТАТЫ МОДЕЛИРОВАНИЯ ===")
print(f"Точность модели на PCA-признаках: {accuracy:.3f}")
print("\nОтчет классификации:")
print(classification_report(y_test, y_pred))
```

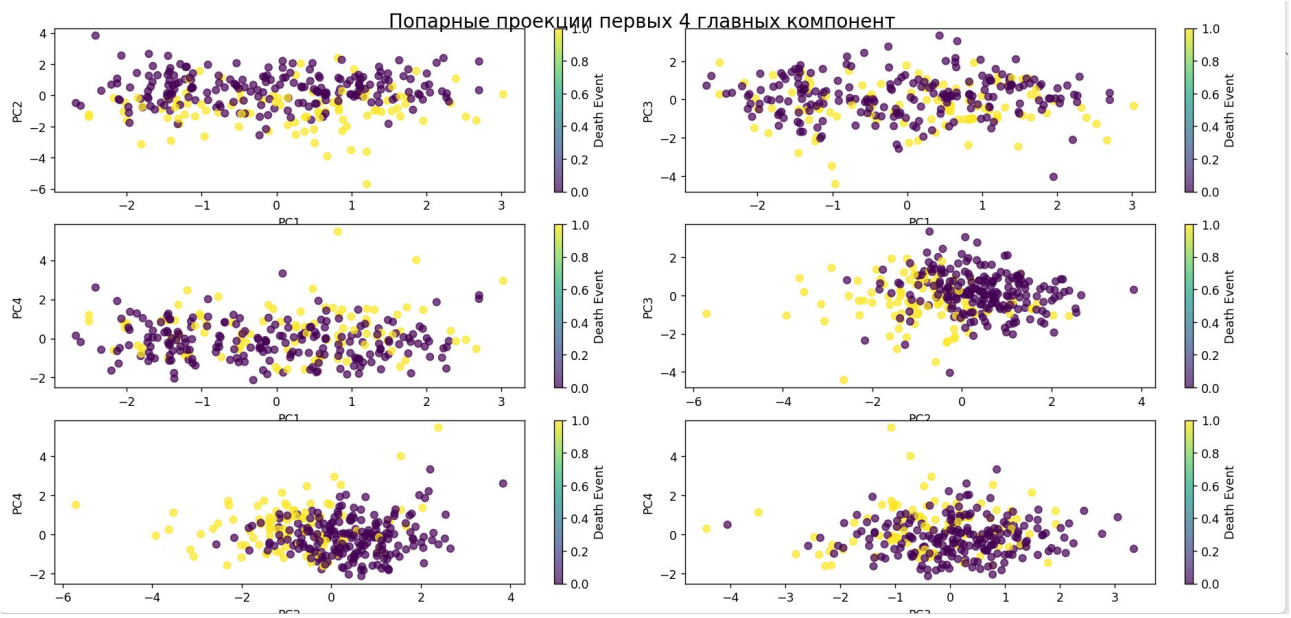
```
# 6. Сравнение с исходными признаками
X_train_orig, X_test_orig, y_train_orig, y_test_orig = train_test_split(
    X_scaled, y, test_size=0.2, random_state=42, stratify=y
)

model_orig = RandomForestClassifier(n_estimators=100, random_state=42)
model_orig.fit(X_train_orig, y_train_orig)
y_pred_orig = model_orig.predict(X_test_orig)
accuracy_orig = accuracy_score(y_test_orig, y_pred_orig)

print(f"\nТочность на исходных признаках: {accuracy_orig:.3f}")
print("Разница в точности:", abs(accuracy - accuracy_orig))

# 7. Визуализация первых 4 компонент попарно
pca_4d = PCA(n_components=4)
X_pca_4d = pca_4d.fit_transform(X_scaled)

plt.figure(figsize=(15, 12))
for i in range(4):
    for j in range(i+1, 4):
        plt.subplot(3, 2, (i*2 + j) if i < 2 else (i+j+1))
        scatter = plt.scatter(X_pca_4d[:, i], X_pca_4d[:, j], c=y,
                              cmap='viridis', alpha=0.7)
        plt.xlabel(f'PC{i+1}')
        plt.ylabel(f'PC{j+1}')
        plt.colorbar(scatter, label='Death Event')

plt.suptitle('Попарные проекции первых 4 главных компонент', fontsize=16)
plt.tight_layout()
plt.show()

# 8. Анализ наиболее важных компонент
print("\n=== АНАЛИЗ КОМПОНЕНТ ===")
for i in range(4):  # Анализируем первые 4 самые важные компоненты
    print(f"\nКомпонента {i+1} (объясняет {explained_variance[i]:.3f} дисперсии):")
```

```
    component_weights = pca_components_df[f'PC{i+1}'].sort_values(key=abs, ascending=False)
    print("Самые важные признаки:")
    for feature, weight in component_weights.head(3).items():
        print(f"  {feature}: {weight:.3f}")
```

Вывод кода:







Вывод: Я научился применять метод PCA для осуществления визуализации данных.