Министерство образования Республики Беларусь
Учреждение образования
«Брестский государственный технический университет»
Кафедра интеллектуально-информационных технологий

Лабораторная работа №1
По дисциплине «Интеллектуальный анализ данных»

Тема: «PCA»

Выполнила:
студентка 4 курса
группы ИИ-24
Коцуба Е.М.
Проверила:
Андренко К.В.

Брест 2025

Цель работы: научиться применять метод PCA для осуществления визуализации данных.

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.

Вариант 5: выборка wholesale+customers.zip, класс Region.

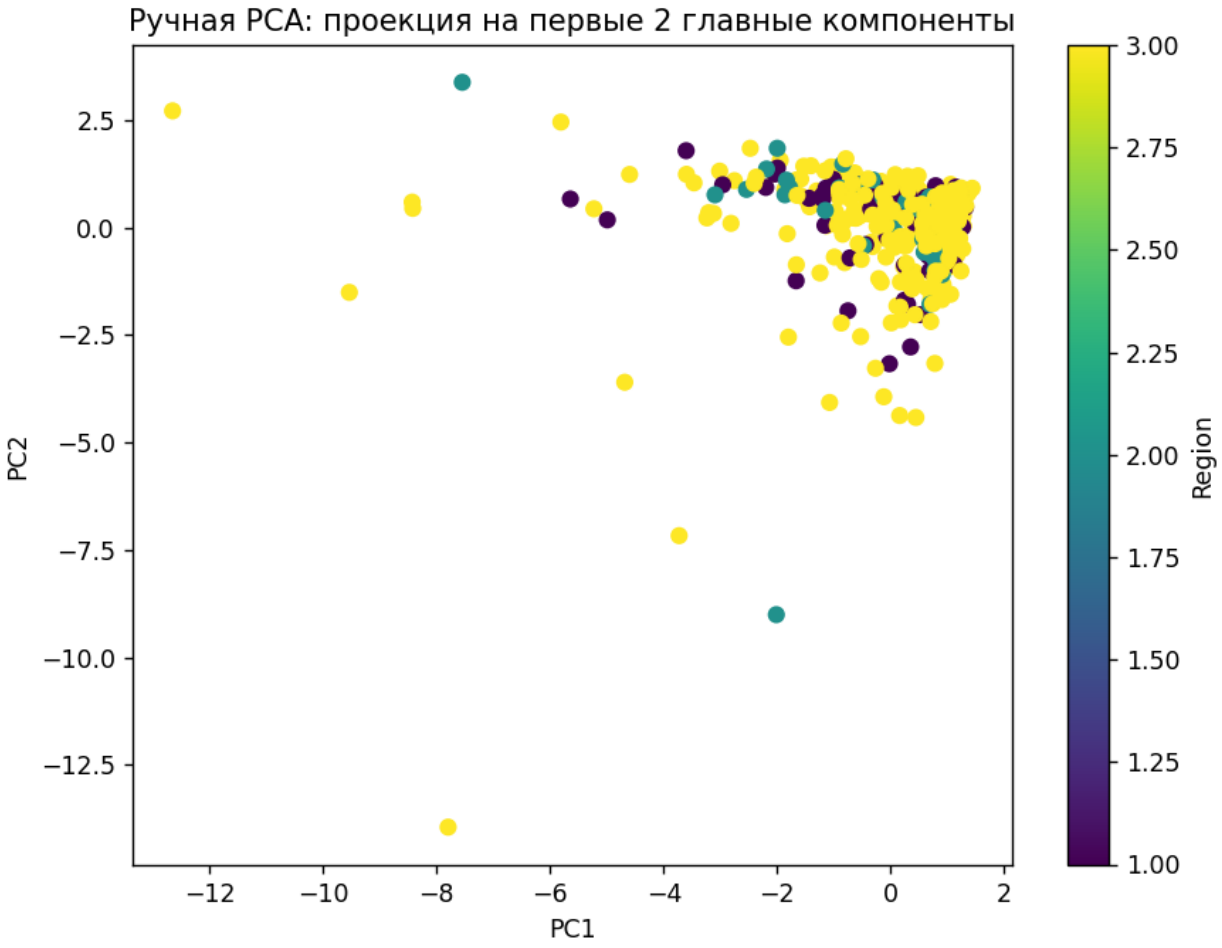
Код программы:

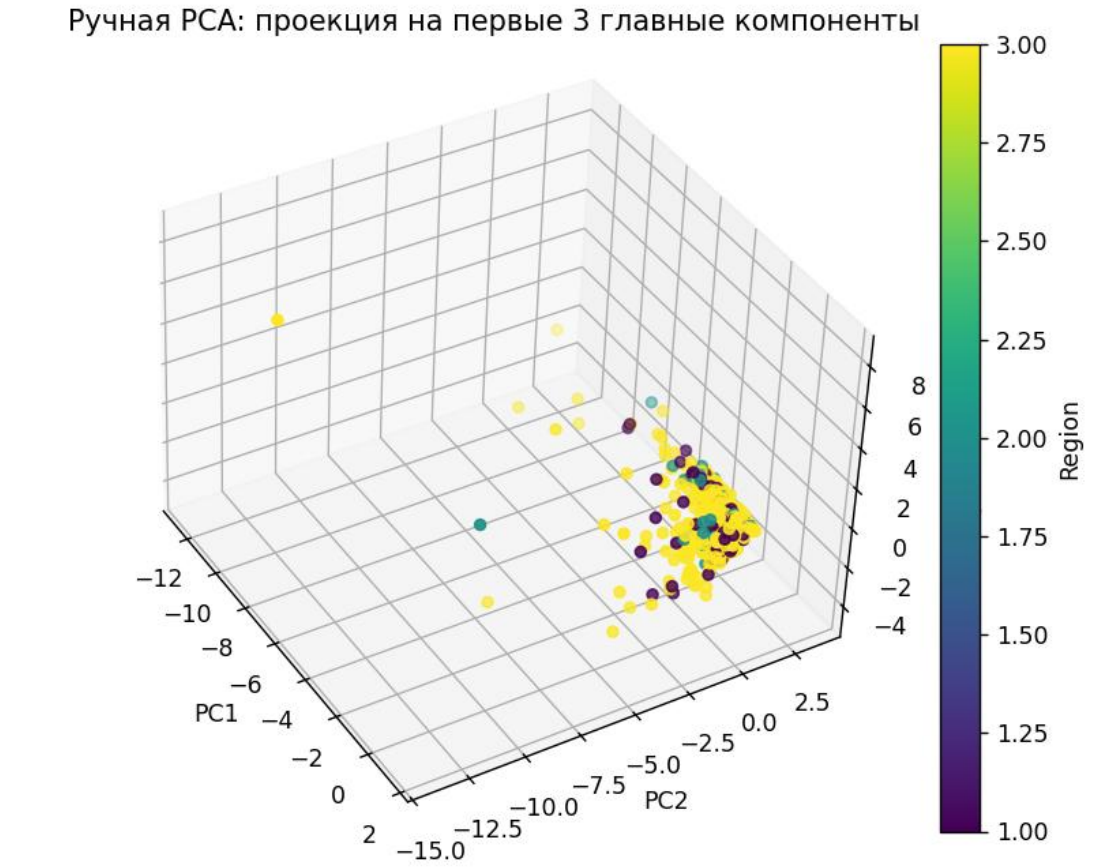
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from mpl_toolkits.mplot3d import Axes3D

df = pd.read_csv('Wholesale customers data.csv')

features = df.iloc[:, 2:].values
labels = df['Region'].values

scaler = StandardScaler()
features_scaled = scaler.fit_transform(features)

# Часть 1: Ручной PCA с numpy.linalg.eig
cov_matrix = np.cov(features_scaled.T)
eigenvalues, eigenvectors = np.linalg.eig(cov_matrix)

idx = eigenvalues.argsort()[::-1]
eigenvalues = eigenvalues[idx]
eigenvectors = eigenvectors[:, idx]

pc_manual_2d = features_scaled.dot(eigenvectors[:, :2])

pc_manual_3d = features_scaled.dot(eigenvectors[:, :3])

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

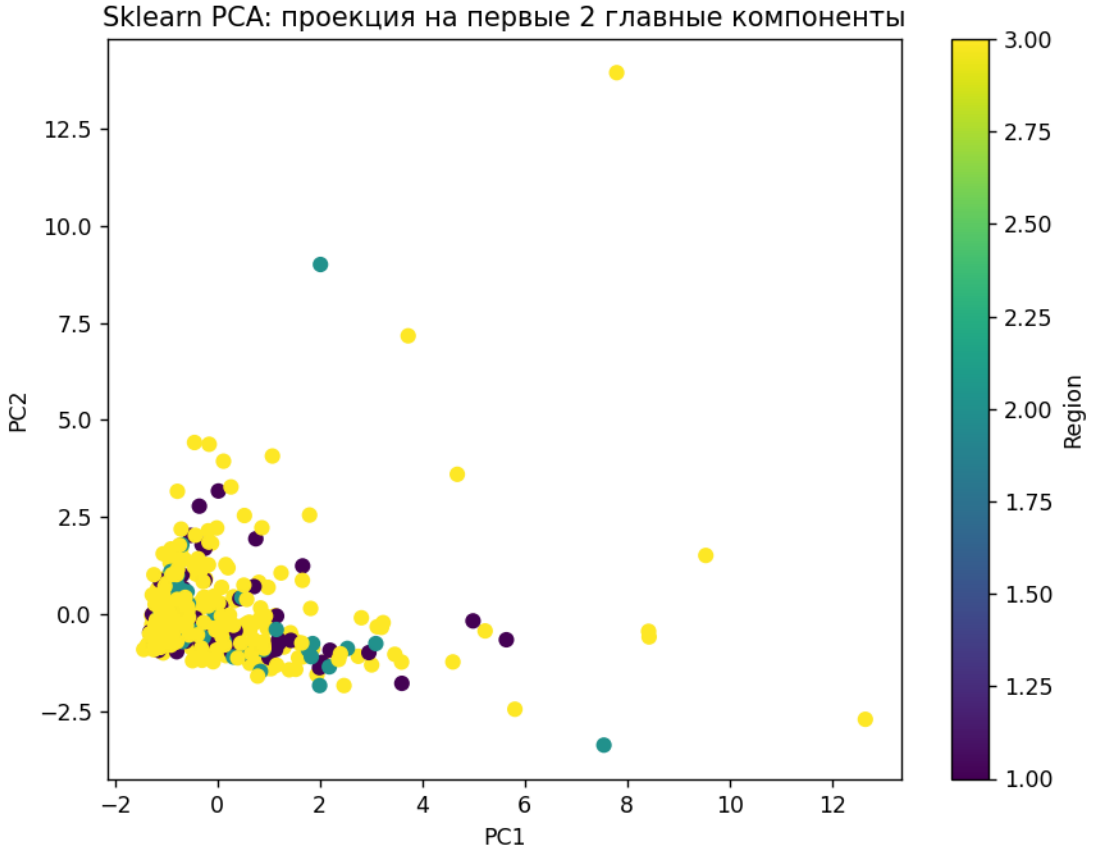
```
scatter = plt.scatter(pc_manual_2d[:, 0], pc_manual_2d[:, 1], c=labels,
cmap='viridis')
plt.title('Ручная PCA: проекция на первые 2 главные компоненты')
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.colorbar(scatter, label='Region')
plt.show()

fig = plt.figure(figsize=(8, 6))
ax = fig.add_subplot(111, projection='3d')
scatter = ax.scatter(pc_manual_3d[:, 0], pc_manual_3d[:, 1], pc_manual_3d[:,
2], c=labels, cmap='viridis')
ax.set_title('Ручная PCA: проекция на первые 3 главные компоненты')
ax.set_xlabel('PC1')
ax.set_ylabel('PC2')
ax.set_zlabel('PC3')
plt.colorbar(scatter, label='Region')
plt.show()

# Часть 2: PCA с sklearn.decomposition.PCA

pca_2d = PCA(n_components=2)
pc_sklearn_2d = pca_2d.fit_transform(features_scaled)

pca_3d = PCA(n_components=3)
pc_sklearn_3d = pca_3d.fit_transform(features_scaled)

plt.figure(figsize=(8, 6))
scatter = plt.scatter(pc_sklearn_2d[:, 0], pc_sklearn_2d[:, 1], c=labels,
cmap='viridis')
plt.title('Sklearn PCA: проекция на первые 2 главные компоненты')
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.colorbar(scatter, label='Region')
plt.show()

fig = plt.figure(figsize=(8, 6))
ax = fig.add_subplot(111, projection='3d')
scatter = ax.scatter(pc_sklearn_3d[:, 0], pc_sklearn_3d[:, 1],
pc_sklearn_3d[:, 2], c=labels, cmap='viridis')
ax.set_title('Sklearn PCA: проекция на первые 3 главные компоненты')
ax.set_xlabel('PC1')
ax.set_ylabel('PC2')
ax.set_zlabel('PC3')
plt.colorbar(scatter, label='Region')
plt.show()

# Часть 3: Вычисление потерь
total_variance = np.sum(eigenvalues)

variance_2d = np.sum(eigenvalues[:2])
loss_2d = 1 - (variance_2d / total_variance)
print(f'Ручное PCA: Потери для 2 компонент: {loss_2d:.4f} (сохраняемая
дисперсия: {variance_2d / total_variance:.4f})')

variance_3d = np.sum(eigenvalues[:3])
loss_3d = 1 - (variance_3d / total_variance)
print(f'Ручное PCA: Потери для 3 компонент: {loss_3d:.4f} (сохраняемая
дисперсия: {variance_3d / total_variance:.4f})')

pca_full = PCA()
pca_full.fit(features_scaled)
explained_variance = pca_full.explained_variance_ratio_
```

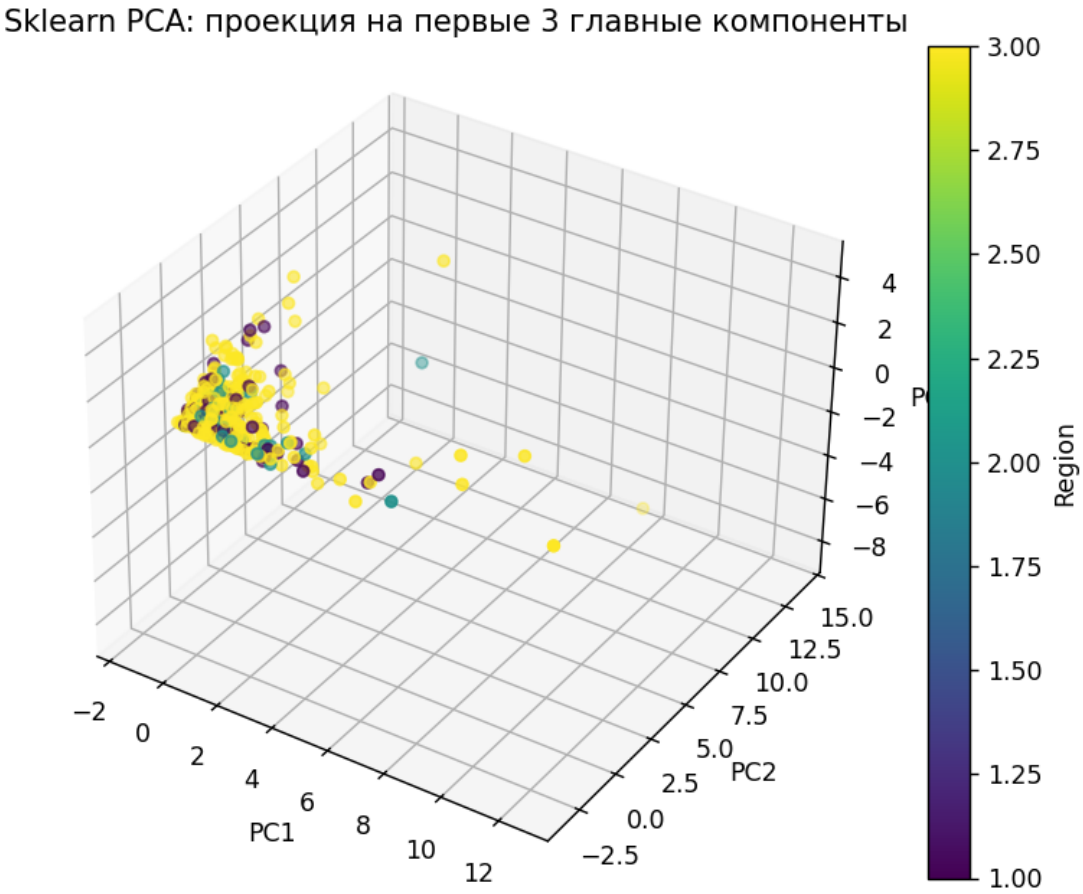
```
sklearn_variance_2d = np.sum(explained_variance[:2])
sklearn_loss_2d = 1 - sklearn_variance_2d
print(f'Sklearn PCA: Потери для 2 компонент: {sklearn_loss_2d:.4f}
(сохраняемая дисперсия: {sklearn_variance_2d:.4f})')

sklearn_variance_3d = np.sum(explained_variance[:3])
sklearn_loss_3d = 1 - sklearn_variance_3d
print(f'Sklearn PCA: Потери для 2 компонент: {sklearn_loss_3d:.4f}
(сохраняемая дисперсия: {sklearn_variance_3d:.4f})')
```

Результат выполнения программы:




Ручная PCA: проекция на первые 3 главные компоненты
PC1
PC2
Region


Sklearn PCA: проекция на первые 2 главные компоненты
PC1
PC2
Region



Ручное PCA: Потери для 2 компонент: 0.2754 (сохраняемая дисперсия: 0.7246)
Ручное PCA: Потери для 3 компонент: 0.1521 (сохраняемая дисперсия: 0.8479)
Sklearn PCA: Потери для 2 компонент: 0.2754 (сохраняемая дисперсия: 0.7246)
Sklearn PCA: Потери для 3 компонент: 0.1521 (сохраняемая дисперсия: 0.8479)

Вывод: оба метода (ручной и sklearn) дают согласованные результаты. Использование sklearn.decomposition.PCA предпочтительнее для практических задач из-за простоты и оптимизации, но ручная реализация полезна для понимания метода. PCA неэффективно для разделения данных по классам (Region) в данном датасете, так как проекции не выявляют четких границ между регионами.