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

Cухаревич Д.С.

**Проверила:**

Якимук А. 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.

**Задание по вариантам**

| **№ варианта** | **Выборка** | **Класс** |
| --- | --- | --- |
| 6 | seeds.zip | Последняя колонка |

**Код:**

**import** os  
**import** io  
**import** numpy **as** np  
**import** pandas **as** pd  
**import** matplotlib.pyplot **as** plt  
**from** sklearn.decomposition **import** PCA  
**from** sklearn.preprocessing **import** StandardScaler  
**from** time **import** perf\_counter

fn = **"seeds\_dataset.txt"  
  
if** os.path.exists(fn):  
 **try**:

**except** Exception:  
 **try**:  
 data = pd.read\_csv(fn, header=**None**)  
 **except** Exception:  
 **pass**cols = [**"area"**, **"perimeter"**, **"compactness"**, **"length of kernel"**,  
 **"width of kernel"**, **"asymmetry coefficient"**, **"length of kernel groove"**, **"class"**]  
data.columns = cols[:data.shape[1]]  
  
print(**"Seeds dataset preview:"**)  
print(data.head(15))

X = data.iloc[:, :-1].values  
y = data.iloc[:, -1].astype(int).values

scaler = StandardScaler()  
X\_std = scaler.fit\_transform(X)

t0 = perf\_counter()

cov\_mat = np.cov(X\_std, rowvar=**False**)

eigvals, eigvecs = np.linalg.eig(cov\_mat)

idx = np.argsort(eigvals)[::-1]  
eigvals\_sorted = eigvals[idx].real  
eigvecs\_sorted = eigvecs[:, idx].real

Z2\_manual = X\_std @ eigvecs\_sorted[:, :2]  
Z3\_manual = X\_std @ eigvecs\_sorted[:, :3]

t1 = perf\_counter()  
manual\_time = t1 - t0

t0 = perf\_counter()

pca2 = PCA(n\_components=2)

Z2\_sklearn = pca2.fit\_transform(X\_std)

pca3 = PCA(n\_components=3)

Z3\_sklearn = pca3.fit\_transform(X\_std)

t1 = perf\_counter()

sk\_time = t1 - t0

**def** scatter\_by\_class\_2d(Z, y, title, fname):  
 plt.figure()  
 classes = np.unique(y)  
 markers = [**'o'**, **'^'**, **'s'**, **'D'**, **'P'**, **'X'**]  
 **for** i, c **in** enumerate(classes):  
 mask = y == c  
 plt.scatter(Z[mask, 0], Z[mask, 1], marker=markers[i % len(markers)], label=**f"class {**c**}"**)  
 plt.xlabel(**"PC1"**)  
 plt.ylabel(**"PC2"**)  
 plt.title(title)  
 plt.legend()  
 plt.tight\_layout()  
 plt.savefig(fname, dpi=150)  
 plt.show()  
  
**def** scatter\_by\_class\_3d(Z, y, title, fname):  
 fig = plt.figure()  
 ax = fig.add\_subplot(111, projection=**'3d'**)  
 classes = np.unique(y)  
 markers = [**'o'**, **'^'**, **'s'**, **'D'**, **'P'**, **'X'**]  
 **for** i, c **in** enumerate(classes):  
 mask = y == c  
 ax.scatter(Z[mask, 0], Z[mask, 1], Z[mask, 2], marker=markers[i % len(markers)], label=**f"class {**c**}"**)  
 ax.set\_xlabel(**"PC1"**)  
 ax.set\_ylabel(**"PC2"**)  
 ax.set\_zlabel(**"PC3"**)  
 ax.set\_title(title)  
 ax.legend()  
 plt.tight\_layout()  
 plt.savefig(fname, dpi=150)  
 plt.show()  
  
scatter\_by\_class\_2d(Z2\_manual, y, **"PCA (manual eig) — first 2 PCs"**, **"data/pca\_manual\_2d.png"**)  
scatter\_by\_class\_2d(Z2\_sklearn, y, **"PCA (sklearn) — first 2 PCs"**, **"data/pca\_sklearn\_2d.png"**)  
  
scatter\_by\_class\_3d(Z3\_manual, y, **"PCA (manual eig) — first 3 PCs"**, **"data/pca\_manual\_3d.png"**)  
scatter\_by\_class\_3d(Z3\_sklearn, y, **"PCA (sklearn) — first 3 PCs"**, **"data/pca\_sklearn\_3d.png"**)

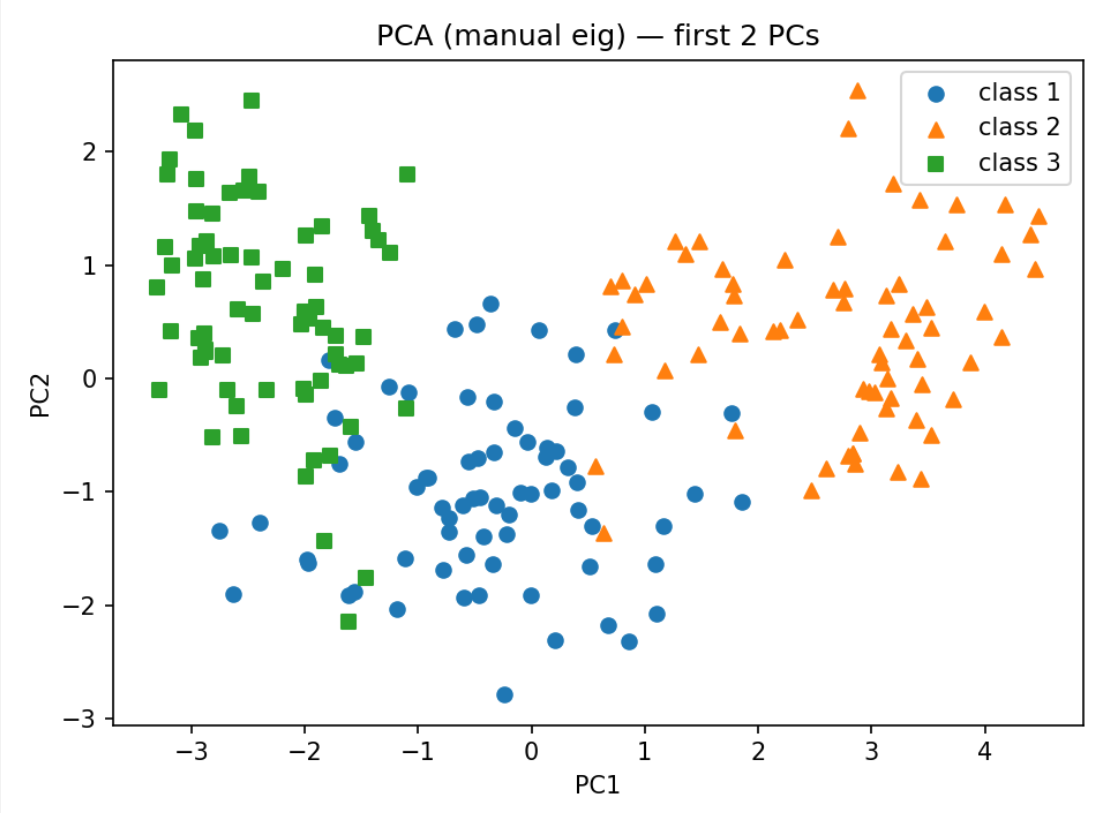
total\_var = eigvals\_sorted.sum().real  
explained\_var\_ratio = eigvals\_sorted / total\_var

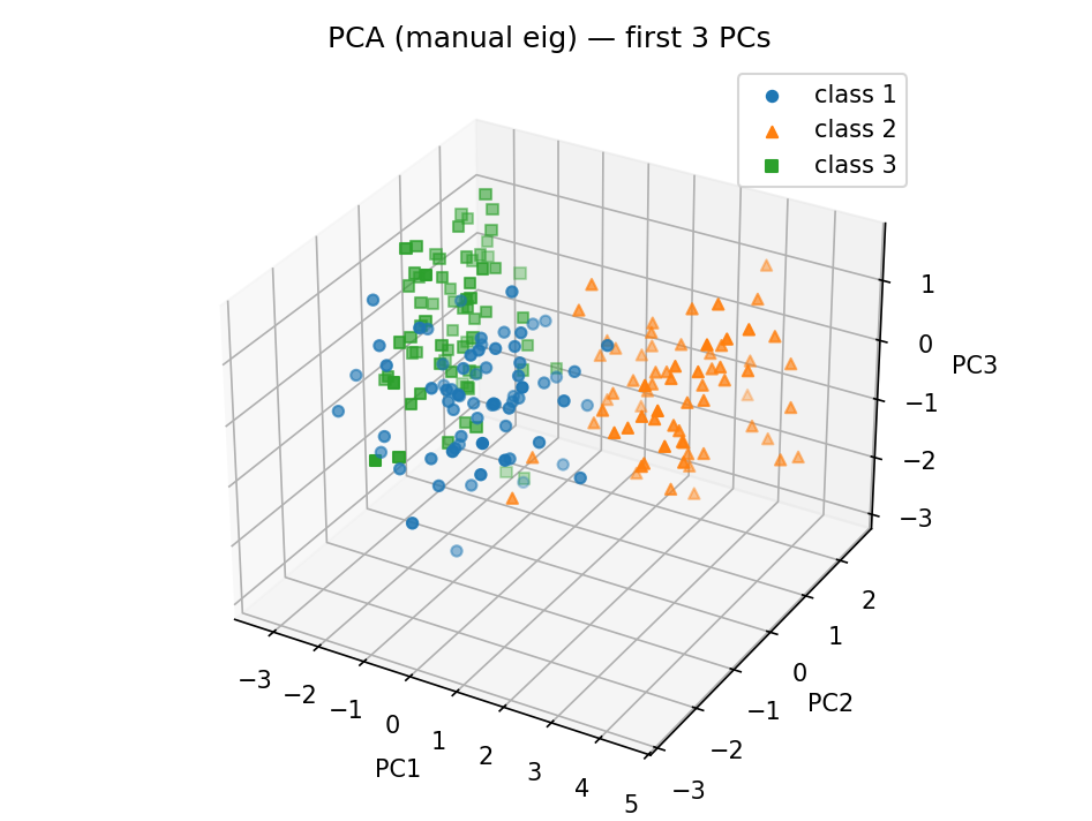
cum\_exp\_2 = explained\_var\_ratio[:2].sum().real  
cum\_exp\_3 = explained\_var\_ratio[:3].sum().real  
loss\_var\_2 = 1.0 - cum\_exp\_2  
loss\_var\_3 = 1.0 - cum\_exp\_3

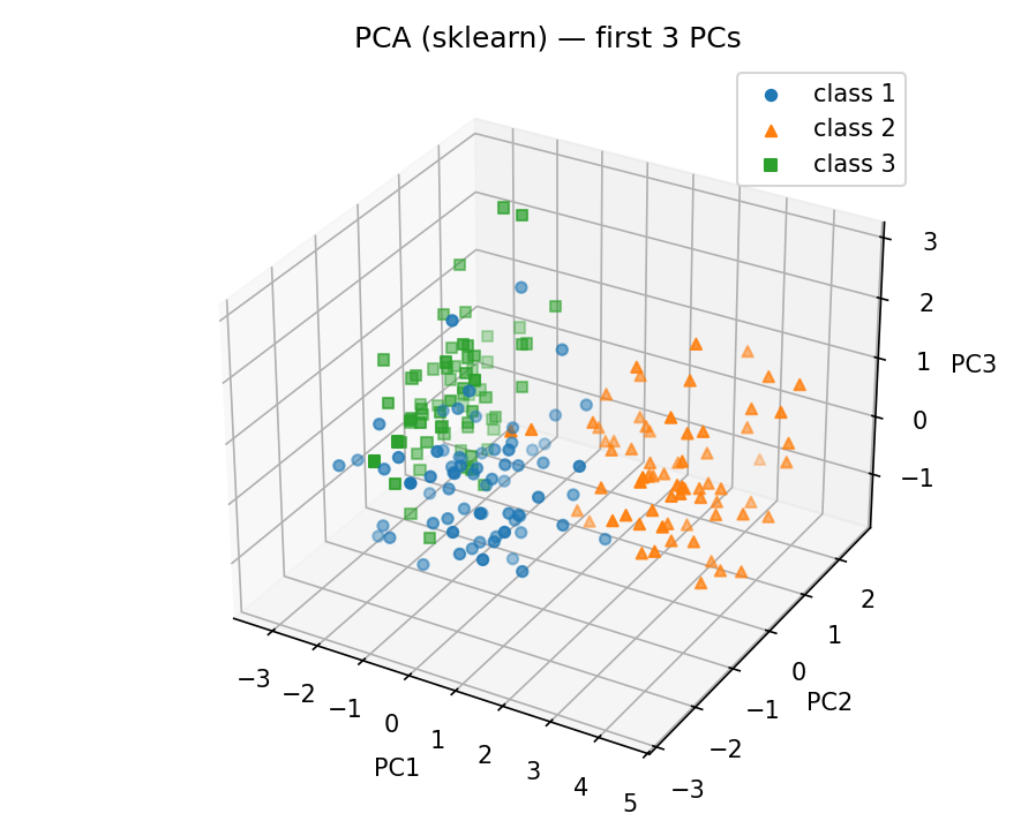
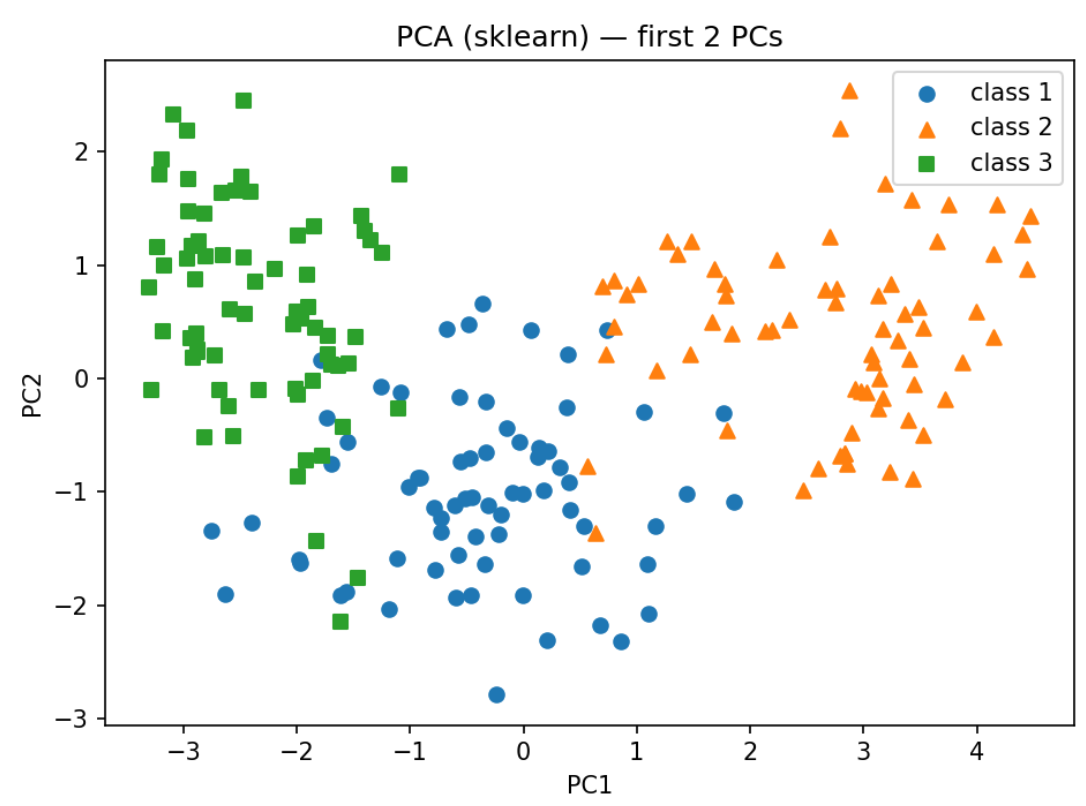
W2 = eigvecs\_sorted[:, :2]  
W3 = eigvecs\_sorted[:, :3]  
  
X\_rec\_2 = (Z2\_manual @ W2.T)  
X\_rec\_3 = (Z3\_manual @ W3.T)

mse\_rec\_2 = np.mean((X\_std - X\_rec\_2)\*\*2)  
mse\_rec\_3 = np.mean((X\_std - X\_rec\_3)\*\*2)

**Вывод:**







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