Министерство образования Республики Беларусь

Учреждение образования

“Брестский государственный технический университет”

Кафедра интеллектуально-информационных технологий

Интеллектуальный анализ данных

Лабораторная работа №1

PCA

Выполнила:

студентка 4 курса

группы ИИ-24

Алешко А. В.

Проверила:

Андренко К. В.

Брест-2025

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

**Код программы(вариант 1):**

import os

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from mpl\_toolkits.mplot3d import Axes3D # noqa: F401

from sklearn.decomposition import PCA

from sklearn.impute import SimpleImputer

from sklearn.preprocessing import StandardScaler

DATA\_PATH = "D:\универ\универ\ИАД\seeds\_dataset.txt"

OUTPUT\_DIR = "pca\_outputs"

os.makedirs(OUTPUT\_DIR, exist\_ok=True)

def load\_seeds(path):

try:

df = pd.read\_csv(path, sep=None, engine='python', header=None)

except Exception:

df = pd.read\_csv(path, delim\_whitespace=True, header=None)

return df

df = load\_seeds(DATA\_PATH)

print("Исходный размер данных:", df.shape)

n\_cols = df.shape[1]

data = df.iloc[:, :n\_cols-1].copy() # признаки

labels = df.iloc[:, n\_cols-1].copy() # класс

# Обработка пропусков

imputer = SimpleImputer(strategy="mean")

X\_imputed = imputer.fit\_transform(data)

X = pd.DataFrame(X\_imputed, columns=data.columns)

scaler = StandardScaler(with\_mean=True, with\_std=True)

X\_scaled = scaler.fit\_transform(X) # numpy array

X\_mean = scaler.mean\_

X\_std = scaler.scale\_

# Вариант 1: вручную

cov\_matrix = np.cov(X\_scaled, rowvar=False)

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

sort\_idx = np.argsort(eigvals)[::-1]

eigvals\_sorted = eigvals[sort\_idx]

eigvecs\_sorted = eigvecs[:, sort\_idx]

W2 = eigvecs\_sorted[:, :2]

W3 = eigvecs\_sorted[:, :3]

X\_proj\_manual\_2 = X\_scaled.dot(W2)

X\_proj\_manual\_3 = X\_scaled.dot(W3)

# Вариант 2: sklearn PCA

pca2 = PCA(n\_components=2)

X\_proj\_skl\_2 = pca2.fit\_transform(X\_scaled)

pca3 = PCA(n\_components=3)

X\_proj\_skl\_3 = pca3.fit\_transform(X\_scaled)

# Сравнение собственного решения и sklearn

unique\_labels = np.unique(labels)

markers = ['o', 's', '^', 'v', 'P', '\*', 'X', 'D']

def plot\_2d(X2, labels, title, filename):

plt.figure(figsize=(7,6))

for i, lab in enumerate(unique\_labels):

mask = labels == lab

plt.scatter(X2[mask,0], X2[mask,1],

label=str(lab),

marker=markers[i % len(markers)],

s=40)

plt.xlabel("PC1")

plt.ylabel("PC2")

plt.title(title)

plt.legend(title="class")

plt.grid(True, linestyle=':', alpha=0.4)

plt.tight\_layout()

out = os.path.join(OUTPUT\_DIR, filename)

plt.savefig(out, dpi=200)

print("Сохранено:", out)

plt.close()

def plot\_3d(X3, labels, title, filename):

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

ax = fig.add\_subplot(111, projection='3d')

for i, lab in enumerate(unique\_labels):

mask = labels == lab

ax.scatter(X3[mask,0], X3[mask,1], X3[mask,2],

label=str(lab),

marker=markers[i % len(markers)],

s=40)

ax.set\_xlabel("PC1")

ax.set\_ylabel("PC2")

ax.set\_zlabel("PC3")

ax.set\_title(title)

ax.legend(title="class", bbox\_to\_anchor=(1.05, 1))

plt.tight\_layout()

out = os.path.join(OUTPUT\_DIR, filename)

plt.savefig(out, dpi=200, bbox\_inches='tight')

print("Сохранено:", out)

plt.close()

# manual

plot\_2d(X\_proj\_manual\_2, labels.values, "Manual PCA: projection on first 2 PCs", "manual\_pca\_2d.png")

plot\_3d(X\_proj\_manual\_3, labels.values, "Manual PCA: projection on first 3 PCs", "manual\_pca\_3d.png")

# sklearn

plot\_2d(X\_proj\_skl\_2, labels.values, "sklearn PCA: projection on first 2 PCs", "sklearn\_pca\_2d.png")

plot\_3d(X\_proj\_skl\_3, labels.values, "sklearn PCA: projection on first 3 PCs", "sklearn\_pca\_3d.png")

# Вычисление потерь

total\_variance = np.sum(eigvals\_sorted.real)

explained\_by\_2 = np.sum(eigvals\_sorted.real[:2])

explained\_by\_3 = np.sum(eigvals\_sorted.real[:3])

retained\_pct\_2 = explained\_by\_2 / total\_variance \* 100.0

retained\_pct\_3 = explained\_by\_3 / total\_variance \* 100.0

loss\_pct\_2 = 100.0 - retained\_pct\_2

loss\_pct\_3 = 100.0 - retained\_pct\_3

print(f"Суммарная дисперсия (всех компонент): {total\_variance:.6f}")

print(f"Суммарная доля, удержанная 2-мя PC: {retained\_pct\_2:.4f}% => потери {loss\_pct\_2:.4f}%")

print(f"Суммарная доля, удержанная 3-мя PC: {retained\_pct\_3:.4f}% => потери {loss\_pct\_3:.4f}%")

# Реконструкция и MSE

X\_recon\_2 = X\_proj\_manual\_2.dot(W2.T)

X\_recon\_3 = X\_proj\_manual\_3.dot(W3.T)

X\_recon\_2\_orig = scaler.inverse\_transform(X\_recon\_2)

X\_recon\_3\_orig = scaler.inverse\_transform(X\_recon\_3)

X\_orig = scaler.inverse\_transform(X\_scaled)

mse\_2 = np.mean((X\_orig - X\_recon\_2\_orig)\*\*2)

mse\_3 = np.mean((X\_orig - X\_recon\_3\_orig)\*\*2)

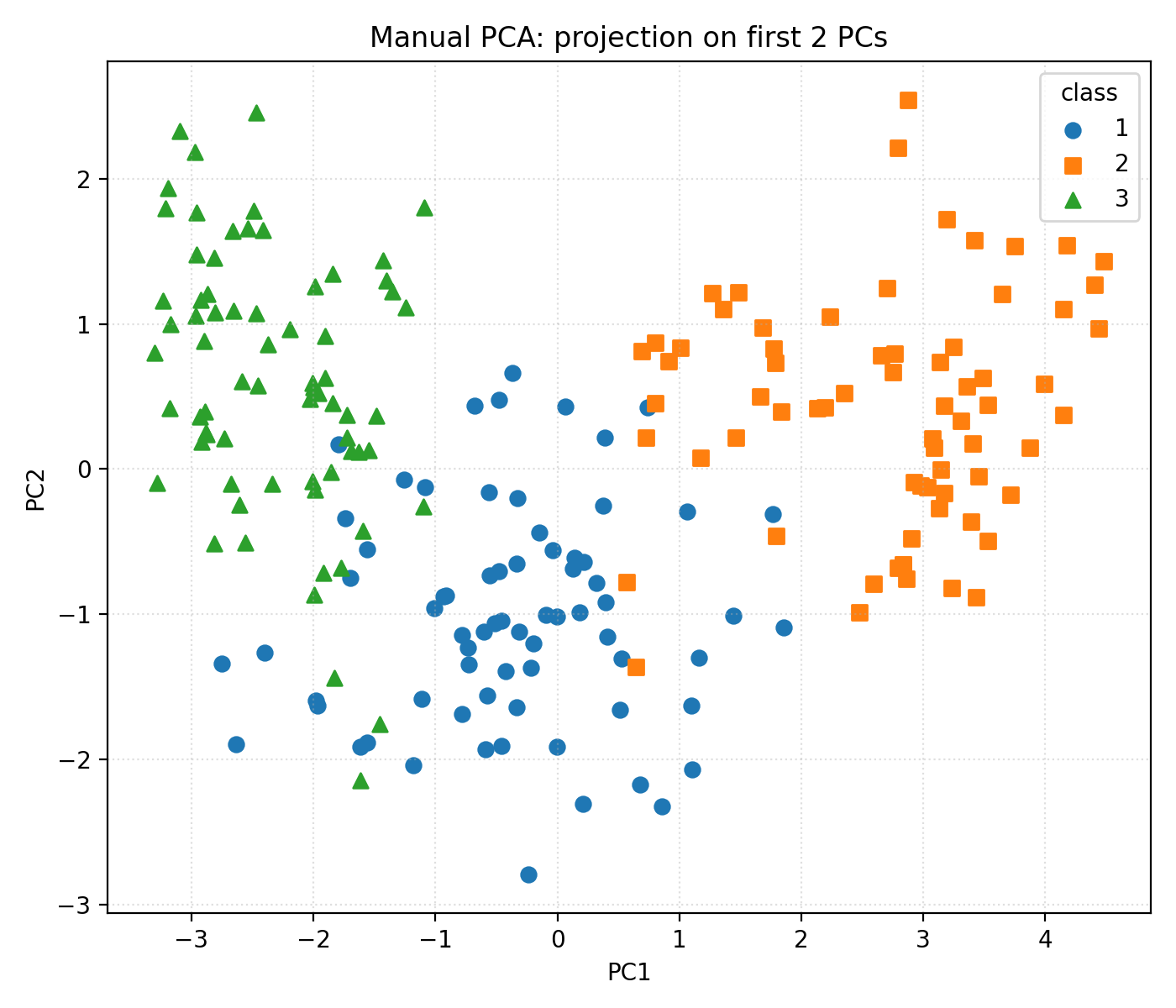
var\_total = np.mean((X\_orig - X\_orig.mean(axis=0))\*\*2)

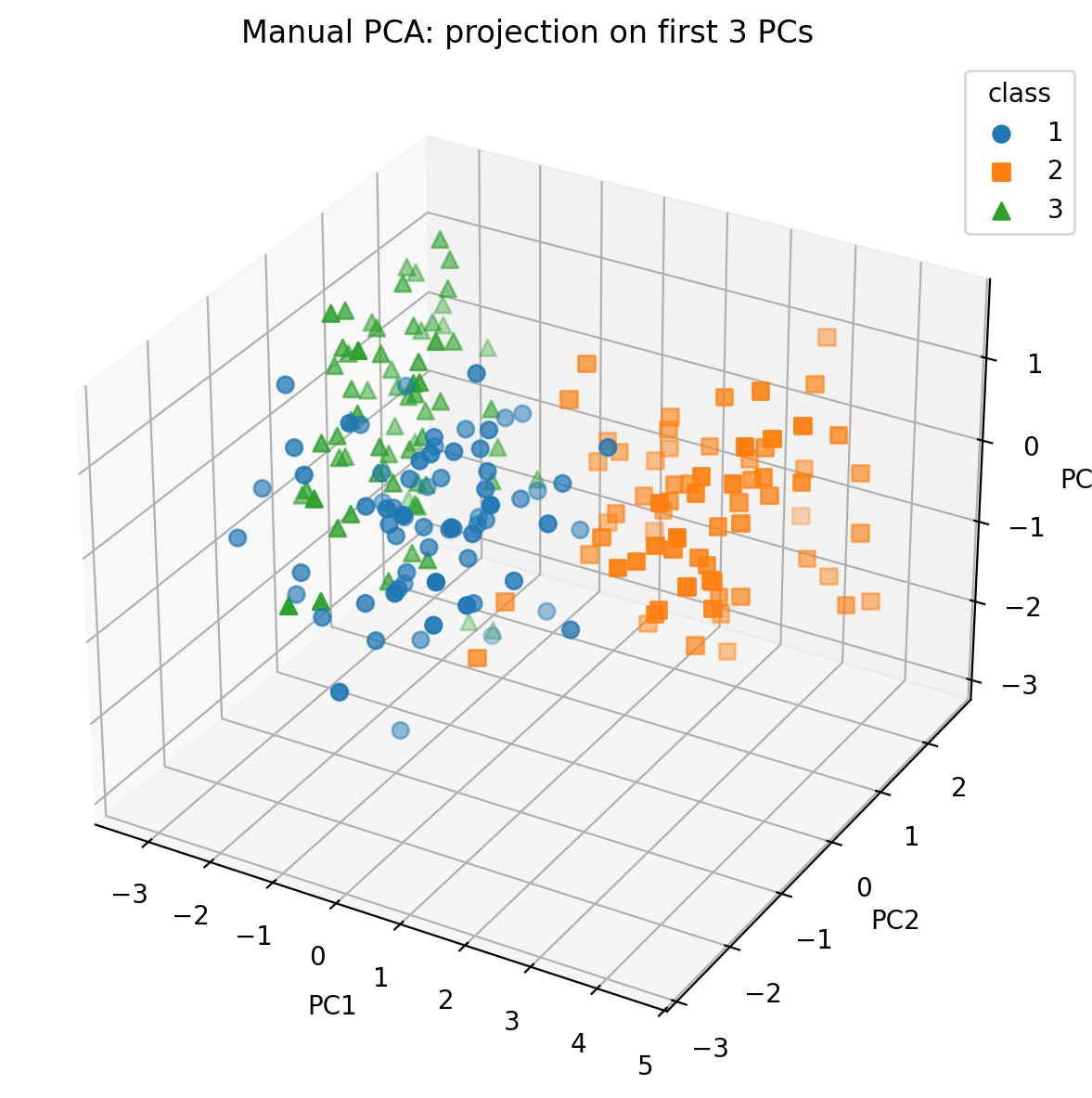
print(f"MSE reconstruction (2-PC): {mse\_2:.6f}")

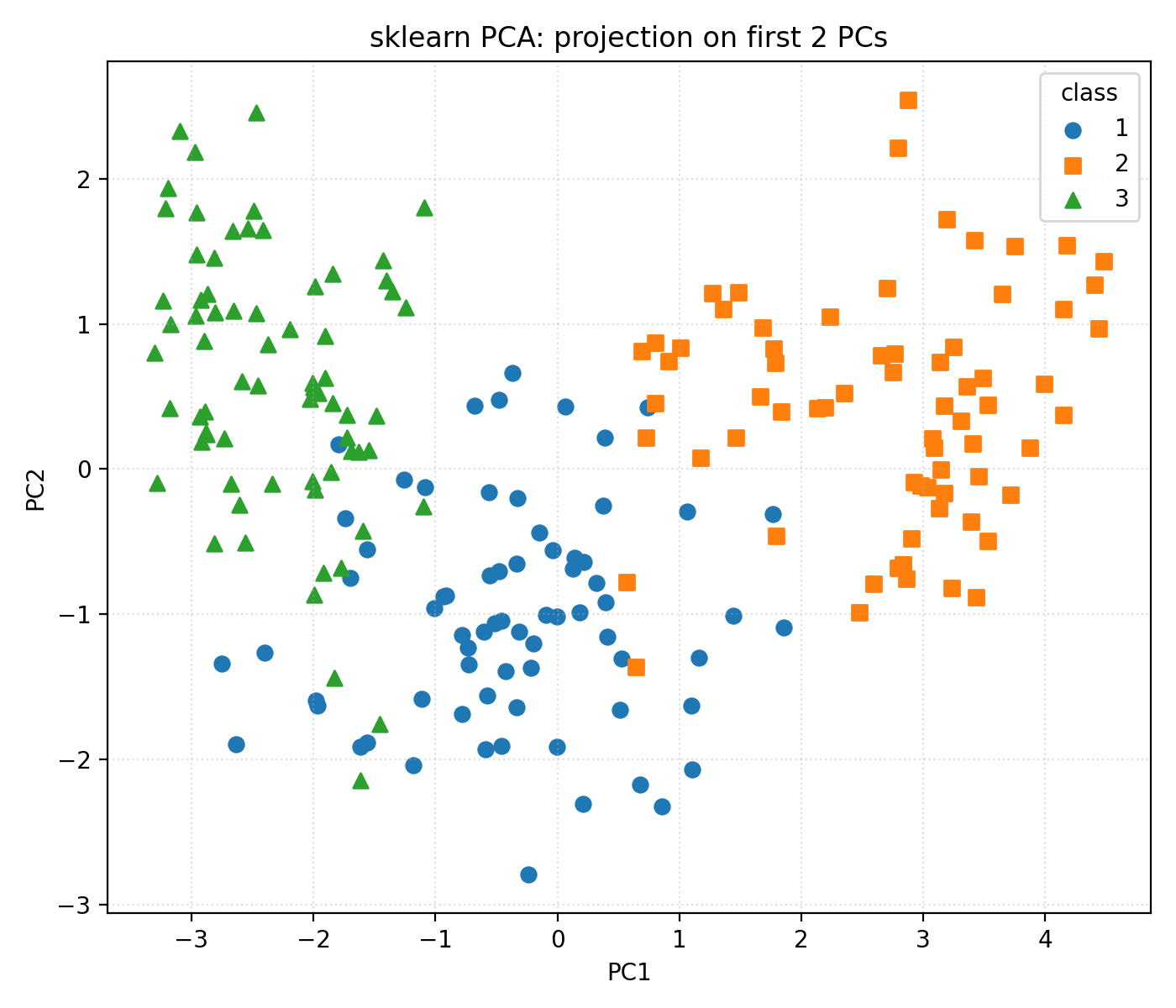
print(f"MSE reconstruction (3-PC): {mse\_3:.6f}")

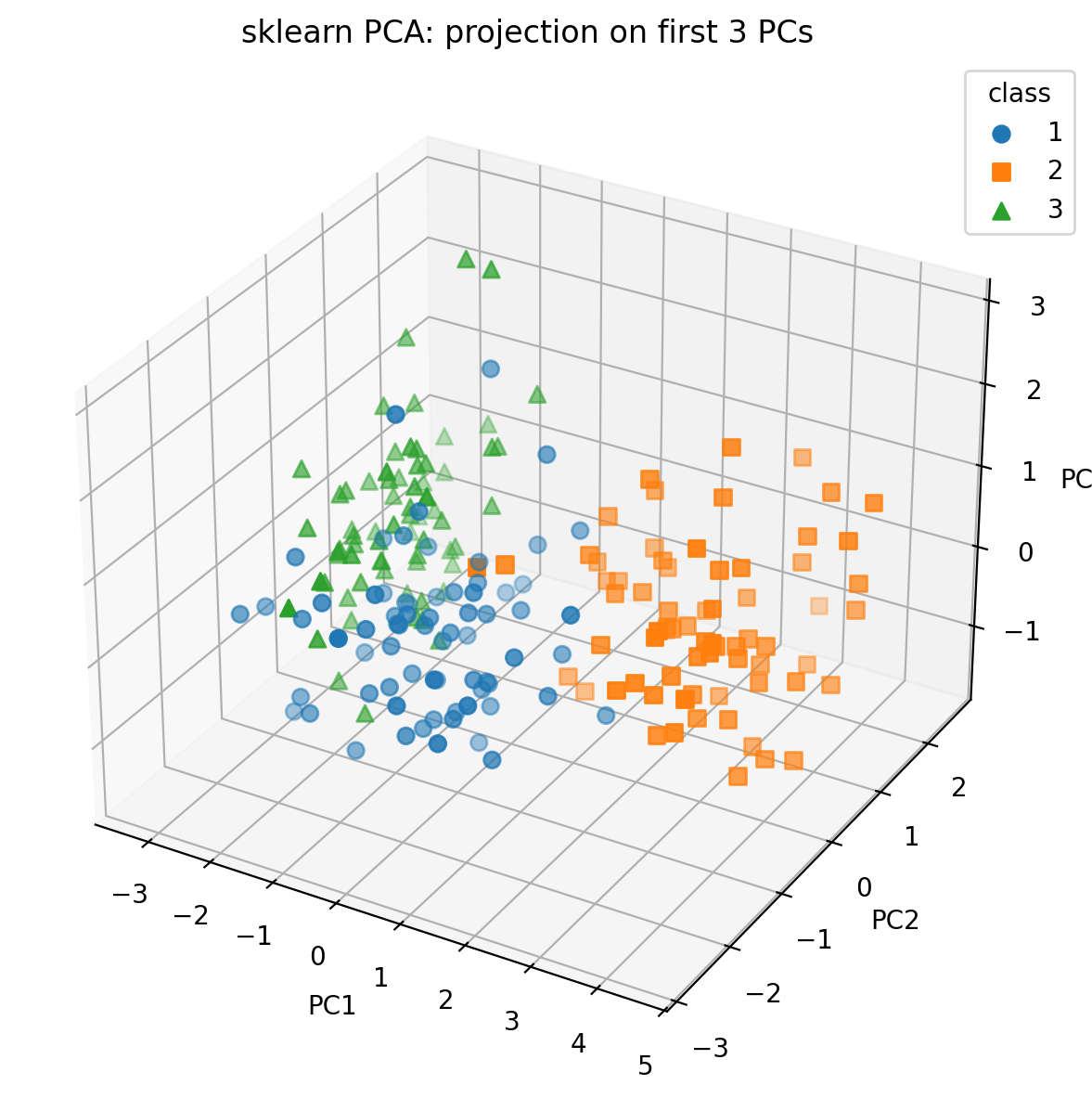
print(f"Отн. потеря дисперсии (2-PC): {mse\_2/var\_total\*100:.4f}%")

print(f"Отн. потеря дисперсии (3-PC): {mse\_3/var\_total\*100:.4f}%")









} **Вывод:** научилась применять метод PCA для осуществления визуализации данных.