`

Министерство образования Республики Беларусь

Учреждение образования

«Брестский Государственный технический университет»

Кафедра ИИТ

Отчет по лабораторной работе 1

Специальность ИИ-23

Выполнил:

Макаревич Н.Р.

Студент группы ИИ-23

Проверил:

Андрейко К. В.
Преподаватель стажер
Кафедры ИИТ,
«___» _____ 2025 г.

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
1	seeds.zip	Последняя колонка

Код программы:

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

```
df_seeds = pd.read_csv('/content/seeds_dataset.txt', sep='\s+', header=None)
```

```
scaler = StandardScaler()
scaled_seeds = scaler.fit_transform(df_seeds.iloc[:, :-1])
```

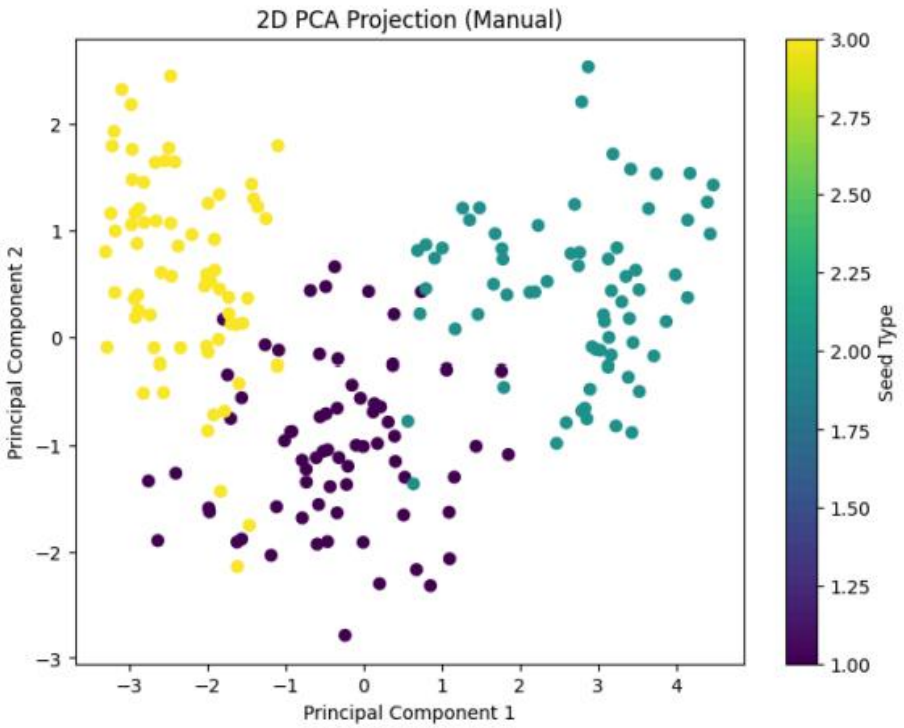
```
cov_matrix = np.cov(scaled_seeds, rowvar=False)
```

```
eigenvalues, eigenvectors = np.linalg.eig(cov_matrix)
```

```
sorted_indices = np.argsort(eigenvalues)[::-1]
top_2_eigenvectors = eigenvectors[:, sorted_indices[:2]]
scaled_seeds_2d = np.dot(scaled_seeds, top_2_eigenvectors)
```

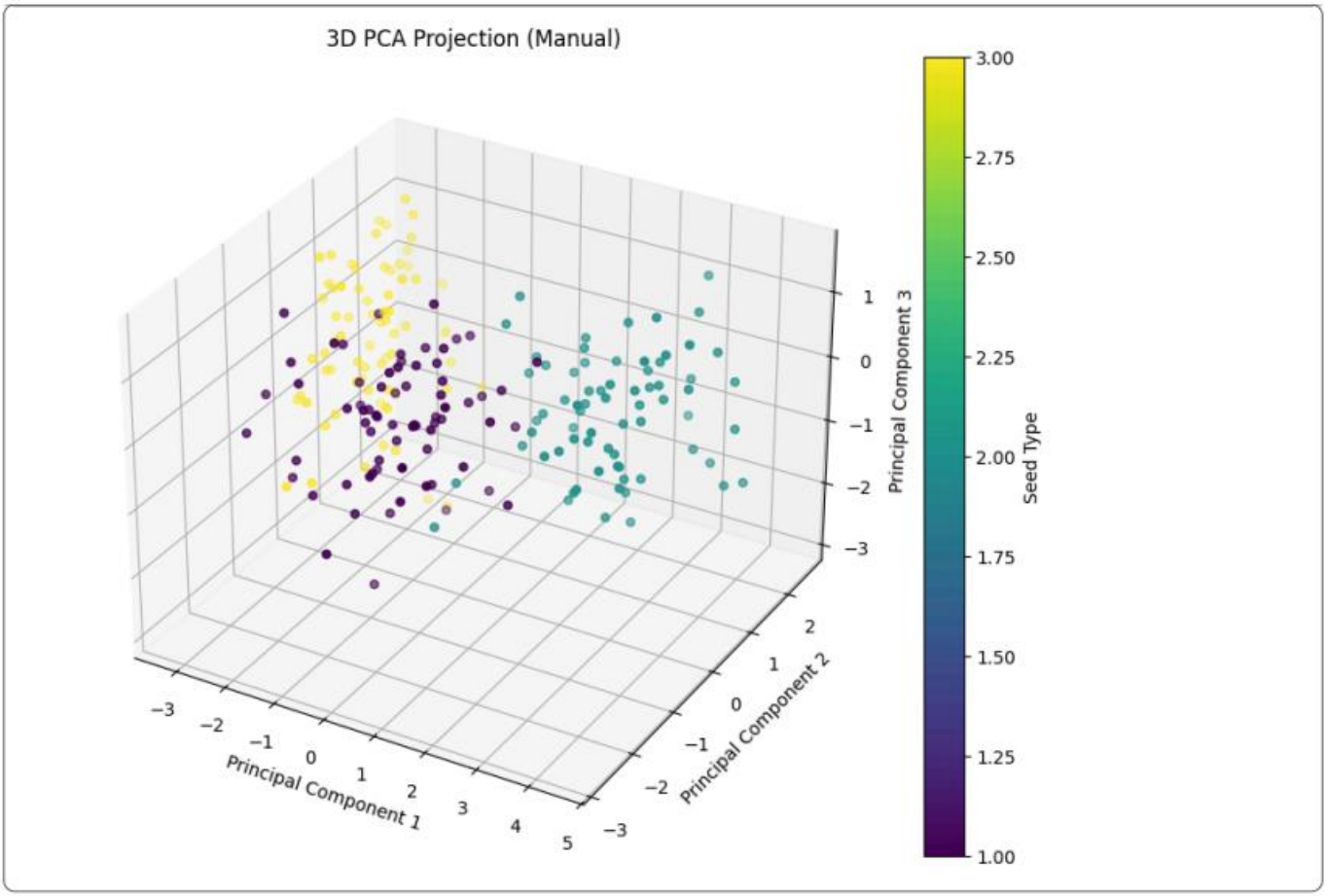
`

```
target = df_seeds.iloc[:, -1]

plt.figure(figsize=(8, 6))
scatter = plt.scatter(scaled_seeds_2d[:, 0], scaled_seeds_2d[:, 1], c=target, cmap='viridis')
plt.title('2D PCA Projection (Manual)')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.colorbar(scatter, label='Seed Type')
plt.show()
```




```
sorted_indices = np.argsort(eigenvalues)[::-1]
top_3_eigenvectors = eigenvectors[:, sorted_indices[:3]]
scaled_seeds_3d = np.dot(scaled_seeds, top_3_eigenvectors)

fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111, projection='3d')
scatter = ax.scatter(scaled_seeds_3d[:, 0], scaled_seeds_3d[:, 1], scaled_seeds_3d[:, 2], c=target, cmap='viridis')
ax.set_title('3D PCA Projection (Manual)')
ax.set_xlabel('Principal Component 1')
ax.set_ylabel('Principal Component 2')
ax.set_zlabel('Principal Component 3')
fig.colorbar(scatter, label='Seed Type')
plt.show()
```

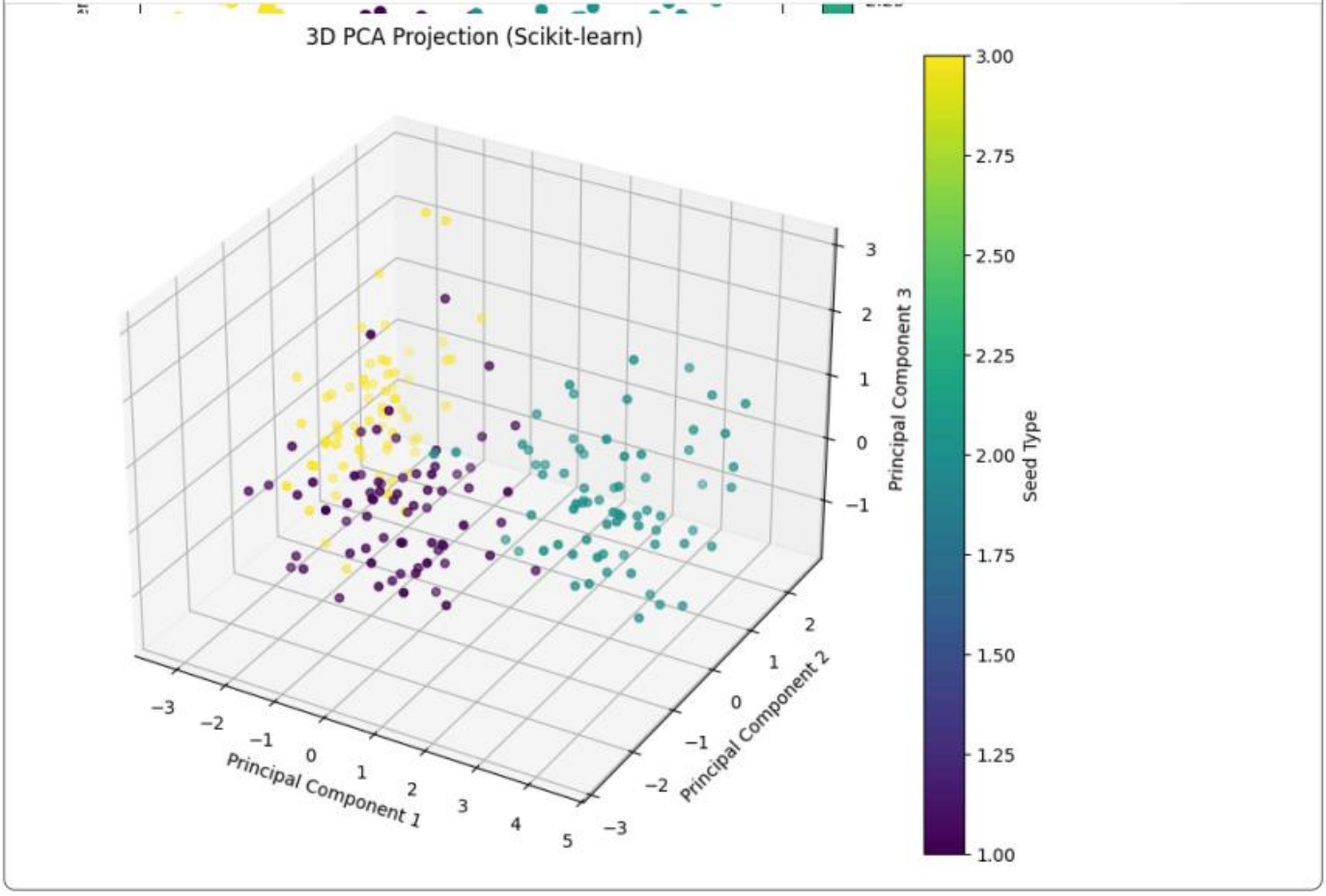
`




```
from sklearn.decomposition import PCA

pca_2d_sklearn = PCA(n_components=2)
scaled_seeds_2d_sklearn = pca_2d_sklearn.fit_transform(scaled_seeds)

plt.figure(figsize=(8, 6))
scatter = plt.scatter(scaled_seeds_2d_sklearn[:, 0], scaled_seeds_2d_sklearn[:, 1], c=target, cmap='viridis')
plt.title('2D PCA Projection (Scikit-learn)')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.colorbar(scatter, label='Seed Type')
plt.show()

pca_3d_sklearn = PCA(n_components=3)
scaled_seeds_3d_sklearn = pca_3d_sklearn.fit_transform(scaled_seeds)
```

`

```
fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111, projection='3d')
scatter = ax.scatter(scaled_seeds_3d_sklearn[:, 0], scaled_seeds_3d_sklearn[:, 1], scaled_seeds_3d_sklearn[:, 2], c=target, cr
ax.set_title('3D PCA Projection (Scikit-learn)')
ax.set_xlabel('Principal Component 1')
ax.set_ylabel('Principal Component 2')
ax.set_zlabel('Principal Component 3')
fig.colorbar(scatter, label='Seed Type')
plt.show()
```




```
total_var = np.sum(eigenvalues)

sorted_eigenvalues = np.sort(eigenvalues)[::-1]
explained_var_ratio = np.cumsum(sorted_eigenvalues) / total_var

loss_2d = 1 - explained_var_ratio[1]
loss_3d = 1 - explained_var_ratio[2]

print("Потери при 2D проекции:", loss_2d)
print("Потери при 3D проекции:", loss_3d)
```

```
Потери при 2D проекции: 0.11017513815087676
Потери при 3D проекции: 0.013317504067955621
```

Вывод: научился применять метод PCA для осуществления визуализации данных.