МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«БРЕСТСКИЙГОСУДАРСТВЕННЫЙТЕХНИЧЕСКИЙУНИВЕРСИТЕТ» ФАКУЛЬТЕТ

ЭЛЕКТРОННО-ИНФОРМАЦИОННЫХ СИСТЕМ

Кафедра интеллектуальных информационных технологий

Отчет по лабораторной работе №1

Специальность ИИ-22

Выполнила
Леваневская Н.И.
студентка группы ИИ-22

Проверил
А.А. Крощенко,
ст. преп.
кафедры ИИТ,
«—» ———— 2024 г.

Брест 2024

Вариант 11

Цель работы: научиться применять метод PCA для осуществления визуализации данных

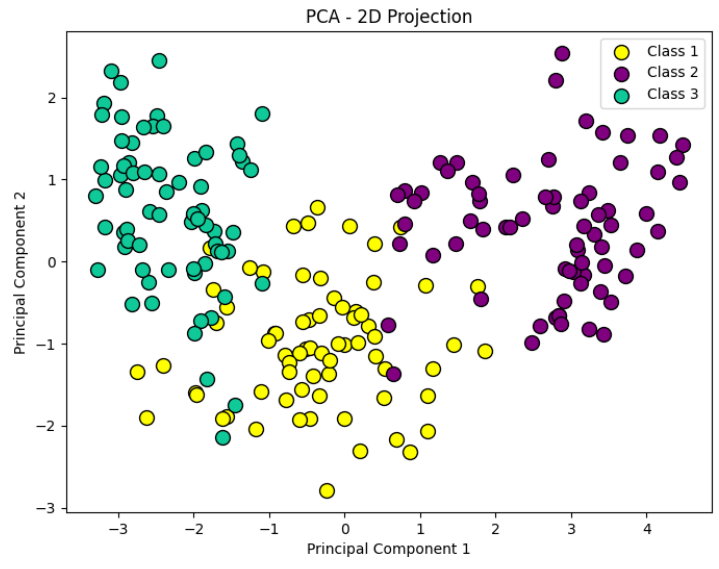
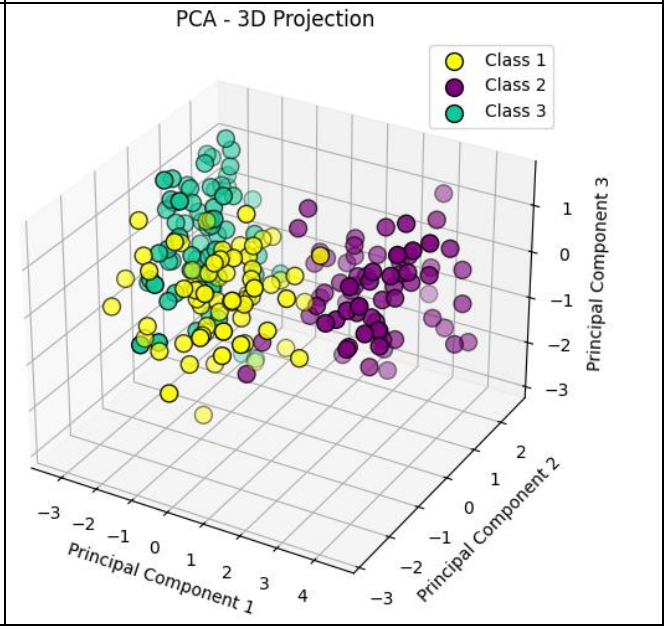
Задания:

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;

Реализованный PCA метод

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

def prepare_data():
    df = pd.read_csv("seeds.csv")
    feature_cols = ['V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7']
    target_col = 'V8'
    X = df[feature_cols].values
    y = df[target_col].values
    return X, y

X, y = prepare_data()
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

def pca_method(data, n_components=2):
    cov_matrix = np.cov(data.T)
    eigen_values, eigen_vectors = np.linalg.eig(cov_matrix)
    sort_index = np.argsort(-eigen_values)
    eigen_vectors = eigen_vectors[:, sort_index]
    return np.dot(data, eigen_vectors[:, :n_components]), eigen_values[sort_index]

X_pca_2D, eigen_values_2D = pca_method(X_scaled, n_components=2)
explained_variance_ratio_2D = eigen_values_2D[:2] / np.sum(eigen_values_2D)
loss_variance_2D = 1 - explained_variance_ratio_2D.sum()

colors = {1: 'yellow', 2: 'purple', 3: '#10C999'}
class_names = {1: 'Class 1', 2: 'Class 2', 3: 'Class 3'}
```

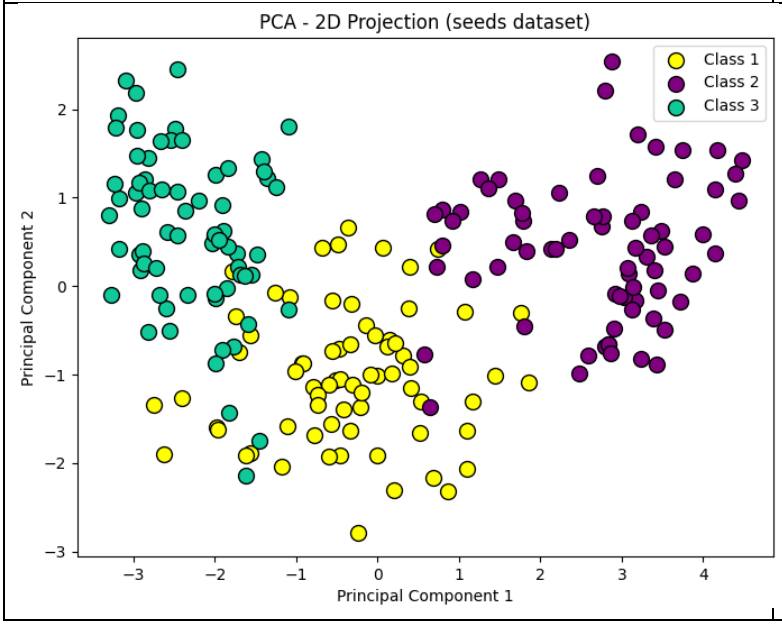
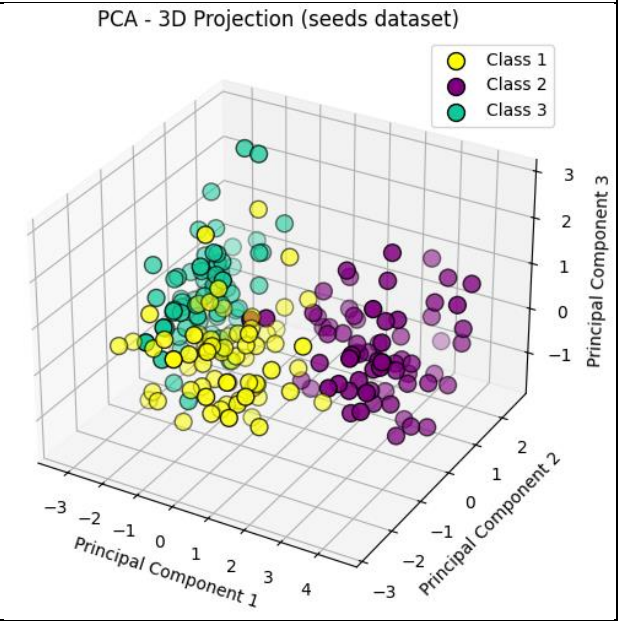
```
y_colors = [colors[int(label)] for label in y]

plt.figure(figsize=(8, 6))
for class_value in np.unique(y):
    plt.scatter(X_pca_2D[y == class_value, 0], X_pca_2D[y == class_value, 1],
                c=colors[class_value], label=class_names[class_value], edgecolor='k',
s=100)
plt.title('PCA - 2D Projection')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.legend()
plt.show()

X_pca_3D, eigen_values_3D = pca_method(X_scaled, n_components=3)
explained_variance_ratio_3D = eigen_values_3D[:3] / np.sum(eigen_values_3D)
loss_variance_3D = 1 - explained_variance_ratio_3D.sum()

fig = plt.figure(figsize=(8, 6))
ax = fig.add_subplot(111, projection='3d')
for class_value in np.unique(y):
    ax.scatter(X_pca_3D[y == class_value, 0], X_pca_3D[y == class_value, 1], X_pca_3D[y ==
class_value, 2],
               c=colors[class_value], label=class_names[class_value], edgecolor='k',
s=100)
plt.title('PCA - 3D Projection')
ax.set_xlabel('Principal Component 1')
ax.set_ylabel('Principal Component 2')
ax.set_zlabel('Principal Component 3')
plt.legend()
plt.show()

print(f'Loss Variance (2D PCA): {loss_variance_2D}')
print(f'Loss Variance (3D PCA): {loss_variance_3D}')
```

Loss Variance (2D PCA): 0.11017513815087643	**Loss Variance (3D PCA): 0.013317504067955399**
	

PCA с использованием scikit-learn

```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
import pandas as pd
import matplotlib.pyplot as plt

def prepare_data():
    df = pd.read_csv("seeds.csv")
    feature_cols = ['V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7']
    target_col = 'V8'
    X = df[feature_cols].values
    y = df[target_col].values
    return X, y

X, y = prepare_data()
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

pca_2d = PCA(n_components=2)
X_reduced_2D = pca_2d.fit_transform(X_scaled)
explained_variance_ratio_2D = pca_2d.explained_variance_ratio_
loss_variance_2D = 1 - explained_variance_ratio_2D.sum()

plt.figure(figsize=(8, 6))
colors = {1: 'yellow', 2: 'purple', 3: '#10C999'}
class_names = {1: 'Class 1', 2: 'Class 2', 3: 'Class 3'}
for class_value in set(y):
    plt.scatter(X_reduced_2D[y == class_value, 0], X_reduced_2D[y == class_value, 1],
                c=colors[class_value], label=class_names[class_value], edgecolor='k',
s=100)
plt.title('PCA - 2D Projection (seeds dataset)')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.legend()
plt.show()

pca_3d = PCA(n_components=3)
X_reduced_3D = pca_3d.fit_transform(X_scaled)
explained_variance_ratio_3D = pca_3d.explained_variance_ratio_
loss_variance_3D = 1 - explained_variance_ratio_3D.sum()

fig = plt.figure(figsize=(8, 6))
ax = fig.add_subplot(111, projection='3d')
for class_value in set(y):
    ax.scatter(X_reduced_3D[y == class_value, 0], X_reduced_3D[y == class_value, 1],
X_reduced_3D[y == class_value, 2],
               c=colors[class_value], label=class_names[class_value], edgecolor='k',
s=100)
plt.title('PCA - 3D Projection (seeds dataset)')
ax.set_xlabel('Principal Component 1')
ax.set_ylabel('Principal Component 2')
ax.set_zlabel('Principal Component 3')
plt.legend()
plt.show()
```

```
print(f'Loss Variance (2D PCA): {loss_variance_2D}')
print(f'Loss Variance (3D PCA): {loss_variance_3D}')
```

Loss Variance (2D PCA): 0.1101751381508762	**Loss Variance (3D PCA): 0.013317504067955288**
	

Вывод: на практике научилась реализовать метод PCA, а также использовать его через библиотеку sklearn.