Министерство образования Республики Беларусь

Учреждение образования

«Брестский Государственный технический университет»

Кафедра ИИТ

Отчет по лабораторной работе 1

Специальность ИИ-23

Выполнил:

Тутина Е.Д.

Студент группы ИИ-23

Проверил:

Андренко К. В.
Преподаватель-стажёр
Кафедры ИИТ,
«___» _____ 2025

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.

Вариант: 11
Выборка : seeds.zip
Класс: последняя колонка

Код программы:

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

DATA_URL = "https://archive.ics.uci.edu/ml/machine-learning-databases/00236/seeds_dataset.txt"
OUTDIR = "seeds_pca_results"
os.makedirs(OUTDIR, exist_ok=True)

col_names = [
   'area', 'perimeter', 'compactness',
   'length_kernel', 'width_kernel',
   'asymmetry_coef', 'length_groove', 'class'
]
```

```
df = pd.read_csv(DATA_URL, sep='\s+', header=None, names=col_names)

print("Dataset shape:", df.shape)
print(df.head())
print("\nClass distribution:\n", df['class'].value_counts())

X = df.drop(columns=['class']).copy()
y = df['class'].copy()

print("\nMissing values per column:\n", X.isna().sum())

imputer = SimpleImputer(strategy='median')
X_imputed = imputer.fit_transform(X)
X = pd.DataFrame(X_imputed, columns=X.columns)

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

X_centered = X_scaled - np.mean(X_scaled, axis=0)

cov = np.cov(X_centered, rowvar=False)

eig_vals, eig_vecs = np.linalg.eig(cov)

idx = np.argsort(eig_vals)[::-1]
eig_vals_sorted = eig_vals[idx].real
eig_vecs_sorted = eig_vecs[:, idx].real

PC_manual_2 = X_centered.dot(eig_vecs_sorted[:, :2])
PC_manual_3 = X_centered.dot(eig_vecs_sorted[:, :3])

total_variance = eig_vals_sorted.sum()
explained_variance_ratio_manual = eig_vals_sorted / total_variance
cumulative_explained = np.cumsum(explained_variance_ratio_manual)

print("\nEigenvalues (sorted):\n", eig_vals_sorted)
print("Explained variance ratio (manual):\n", explained_variance_ratio_manual)
print("Cumulative explained (manual):\n", cumulative_explained)

pca = PCA(n_components=7)
scores_sklearn = pca.fit_transform(X_scaled)
explained_ratio_sklearn = pca.explained_variance_ratio_
cum_explained_sklearn = np.cumsum(explained_ratio_sklearn)

print("\nExplained variance ratio (sklearn):\n", explained_ratio_sklearn)
print("Cumulative explained (sklearn):\n", cum_explained_sklearn)

PC_sklearn_2 = scores_sklearn[:, :2]
```

```
PC_sklearn_3 = scores_sklearn[:, :3]

import matplotlib
matplotlib.use('Agg')

def plot_2d(pc_scores, labels, title, outpath):
    plt.figure(figsize=(7,6))
    classes = np.unique(labels)
    markers = ['o', '^', 's']
    colors = ['tab:blue', 'tab:orange', 'tab:green']
    for i, cls in enumerate(classes):
        mask = labels == cls
        plt.scatter(pc_scores[mask,0], pc_scores[mask,1],
                    label=str(cls), marker=markers[i%len(markers)], alpha=0.8)
    plt.xlabel('PC1'); plt.ylabel('PC2'); plt.title(title)
    plt.legend(title='class')
    plt.grid(True)
    plt.tight_layout()
    plt.savefig(outpath, dpi=150)
    plt.close()

plot_2d(PC_manual_2, y.values, "PCA Manual (first 2 PCs)", os.path.join(OUTDIR, "manual_pca_2d.png"))
plot_2d(PC_sklearn_2, y.values, "PCA sklearn (first 2 PCs)", os.path.join(OUTDIR, "sklearn_pca_2d.png"))

def plot_3d(pc_scores, labels, title, outpath):
    fig = plt.figure(figsize=(8,6))
    ax = fig.add_subplot(111, projection='3d')
    classes = np.unique(labels)
    markers = ['o', '^', 's']
    colors = ['tab:blue', 'tab:orange', 'tab:green']
    for i, cls in enumerate(classes):
        mask = labels == cls
        ax.scatter(pc_scores[mask,0], pc_scores[mask,1], pc_scores[mask,2],
                   label=str(cls), marker=markers[i%len(markers)], alpha=0.8)
    ax.set_xlabel('PC1'); ax.set_ylabel('PC2'); ax.set_zlabel('PC3')
    ax.set_title(title)
    ax.legend(title='class')
    plt.tight_layout()
    plt.savefig(outpath, dpi=150)
    plt.close()

plot_3d(PC_manual_3, y.values, "PCA Manual (first 3 PCs)", os.path.join(OUTDIR, "manual_pca_3d.png"))
plot_3d(PC_sklearn_3, y.values, "PCA sklearn (first 3 PCs)", os.path.join(OUTDIR, "sklearn_pca_3d.png"))

def variance_loss_by_k(eigvals_sorted, k):
    total = eigvals_sorted.sum()
    lost = eigvals_sorted[k:].sum()
    return lost / total
```

```
for k in (1,2,3,4,5,6,7):
    lost_frac = variance_loss_by_k(eig_vals_sorted, k)
    print(f"Using {k} components -> lost variance fraction = {lost_frac:.6f} ({lost_frac*100:.3f}%)")

def reconstruction_mse(X_scaled, pca_full, k):
    components_k = pca_full.components_[:k]
    mean = pca_full.mean_
    scores = (X_scaled - mean).dot(components_k.T)
    X_rec = scores.dot(components_k) + mean
    mse = np.mean((X_scaled - X_rec)**2)
    return mse

for k in (1,2,3,4,5,6,7):
    mse = reconstruction_mse(X_scaled, pca, k)
    print(f"Reconstruction MSE with {k} components: {mse:.6f}")

results = {
    "eig_vals_sorted": eig_vals_sorted.tolist(),
    "explained_variance_ratio_manual": explained_variance_ratio_manual.tolist(),
    "explained_variance_ratio_sklearn": explained_ratio_sklearn.tolist(),
    "cumulative_manual": cumulative_explained.tolist(),
    "cumulative_sklearn": cum_explained_sklearn.tolist()
}
with open(os.path.join(OUTDIR, "pca_results_summary.json"), "w") as f:
    import json
    json.dump(results, f, indent=2)

print("\nPlots and summary saved to:", OUTDIR)
```

Результат работы программы:

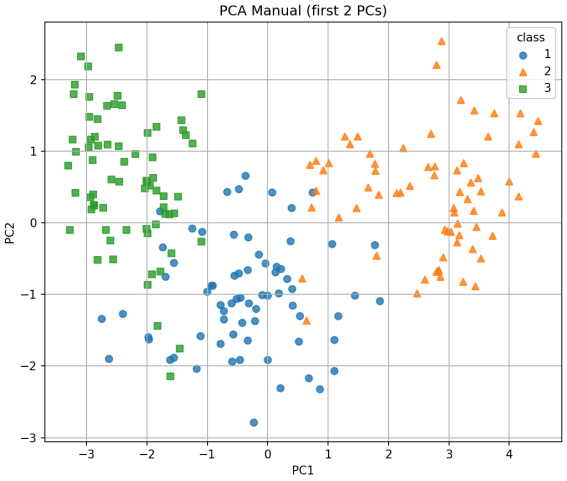
```
Dataset shape: (210, 8)
    area  perimeter  compactness  ...  asymmetry_coef  length_groove  class
0  15.26      14.84       0.8710  ...           2.221          5.220      1
1  14.88      14.57       0.8811  ...           1.018          4.956      1
2  14.29      14.09       0.9050  ...           2.699          4.825      1
3  13.84      13.94       0.8955  ...           2.259          4.805      1
4  16.14      14.99       0.9034  ...           1.355          5.175      1

[5 rows x 8 columns]

Class distribution:
 class
```

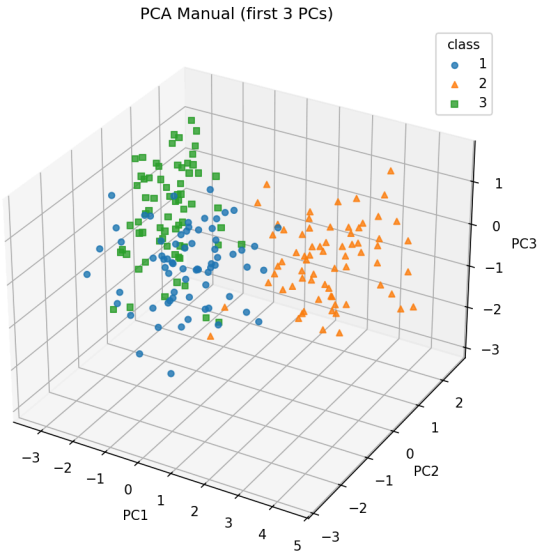
```
1    70
2    70
3    70
Name: count, dtype: int64


Missing values per column:
 area              0
perimeter         0
compactness       0
length_kernel     0
width_kernel      0
asymmetry_coef    0
length_groove     0
dtype: int64


Eigenvalues (sorted):
 [5.05527392e+00 1.20330286e+00 6.81247474e-01 6.86915798e-02
 1.88031478e-02 5.35755786e-03 8.16283865e-04]
Explained variance ratio (manual):
 [7.18743027e-01 1.71081835e-01 9.68576341e-02 9.76635386e-03
 2.67337271e-03 7.61720812e-04 1.16056686e-04]
Cumulative explained (manual):
 [0.71874303 0.88982486 0.9866825  0.99644885 0.99912222 0.99988394
 1.        ]


Explained variance ratio (sklearn):
 [7.18743027e-01 1.71081835e-01 9.68576341e-02 9.76635386e-03
 2.67337271e-03 7.61720812e-04 1.16056686e-04]
Cumulative explained (sklearn):
 [0.71874303 0.88982486 0.9866825  0.99644885 0.99912222 0.99988394
 1.        ]
Using 1 components -> lost variance fraction = 0.281257 (28.126%)
Using 2 components -> lost variance fraction = 0.110175 (11.018%)
Using 3 components -> lost variance fraction = 0.013318 (1.332%)
Using 4 components -> lost variance fraction = 0.003551 (0.355%)
Using 5 components -> lost variance fraction = 0.000878 (0.088%)
Using 6 components -> lost variance fraction = 0.000116 (0.012%)
Using 7 components -> lost variance fraction = 0.000000 (0.000%)
Reconstruction MSE with 1 components: 0.281257
Reconstruction MSE with 2 components: 0.110175
```

```
Reconstruction MSE with 3 components: 0.013318
Reconstruction MSE with 4 components: 0.003551
Reconstruction MSE with 5 components: 0.000878
Reconstruction MSE with 6 components: 0.000116
Reconstruction MSE with 7 components: 0.000000
```

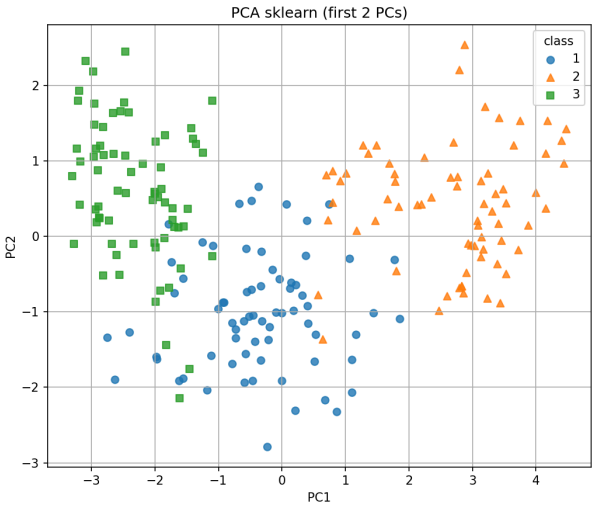
Проецирование на плоскость :

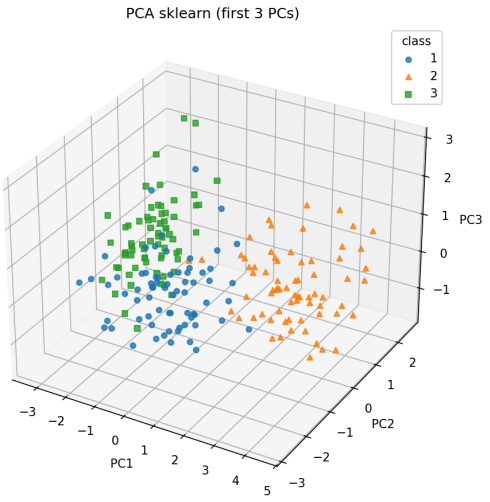
1. вручную





2. с помощью sklearn.decomposition.PCA





Вывод: научился применять метод PCA для осуществления визуализации данных.