МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«БРЕСТСКИЙГОСУДАРСТВЕННЫЙТЕХНИЧЕСКИЙУНИВЕРСИТЕТ» ФАКУЛЬТЕТ

ЭЛЕКТРОННО-ИНФОРМАЦИОННЫХ СИСТЕМ

Кафедра интеллектуальных информационных технологий

# Отчет по лабораторной работе №2

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 автоэнкодеры для осуществления визуализации данных и их анализа.

**Задания:**

1. Используя выборку по варианту, осуществить проецирование данных на плоскость первых двух и трех главных компонент с использованием нейросетевой модели автоэнкодера (с двумя и тремя нейронами в среднем слое);

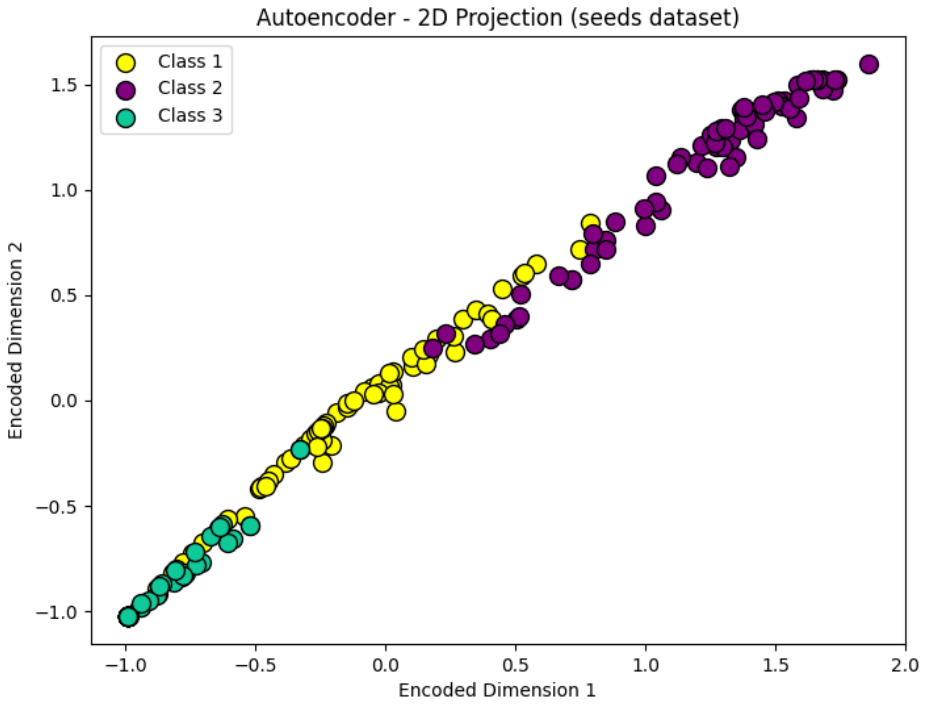
2. Выполнить визуализацию полученных главных компонент с использованием средств библиотеки matplotlib, обозначая экземпляры разных классов с использованием разных цветовых маркеров;

3. Реализовать метод t-SNE для визуализации данных (использовать также 2 и 3 компонента), построить соответствующую визуализацию;

**Реализованный автоэнкодера с двумя скрытыми слоями и выходом на 2 нейрона**

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.neural_network import MLPRegressor

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)

autoencoder = MLPRegressor(hidden_layer_sizes=(6, 4), activation='relu', solver='adam',
max_iter=1000)

autoencoder.fit(X_scaled, X_scaled)

X_encoded_2D = autoencoder.predict(X_scaled)
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_encoded_2D[y == class_value, 0], X_encoded_2D[y == class_value, 1],
                c=colors[class_value], label=class_names[class_value], edgecolor='k', s=100)
plt.title('Autoencoder - 2D Projection (seeds dataset)')
plt.xlabel('Encoded Dimension 1')
plt.ylabel('Encoded Dimension 2')
plt.legend()
```

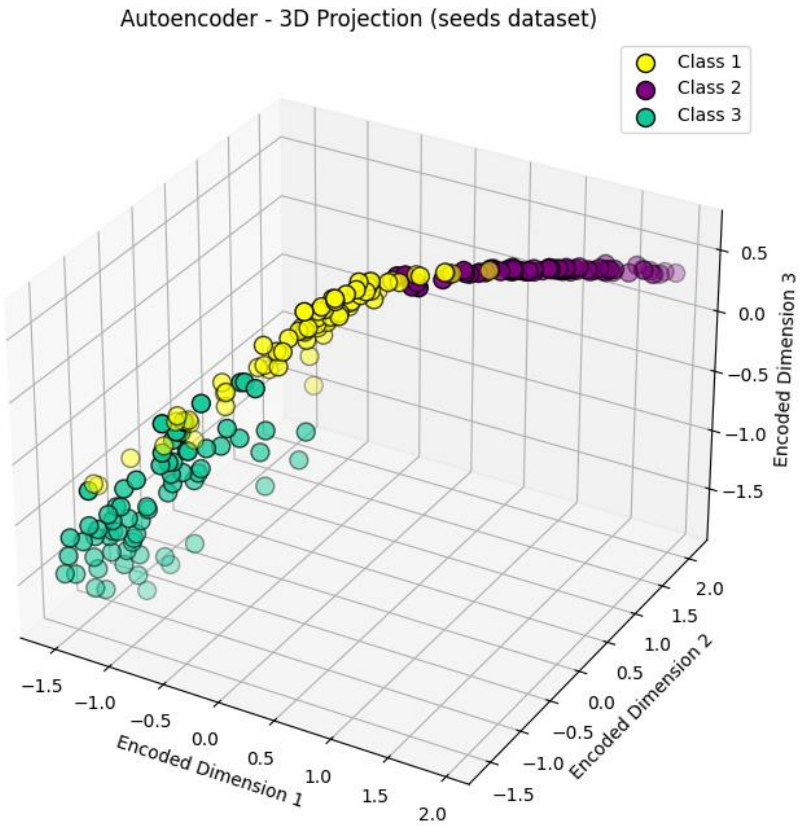
```
plt.show()
```




**Реализованный автоэнкодера с двумя скрытыми слоями и выходом на 3 нейрона**

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.neural_network import MLPRegressor

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)

autoencoder_3d = MLPRegressor(hidden_layer_sizes=(6, 4), activation='relu', solver='adam',
max_iter=1000)

autoencoder_3d.fit(X_scaled, X_scaled)

X_encoded_3D = autoencoder_3d.predict(X_scaled)

fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111, projection='3d')
colors = {1: 'yellow', 2: 'purple', 3: '#10C999'}
class_names = {1: 'Class 1', 2: 'Class 2', 3: 'Class 3'}

for class_value in set(y):
```

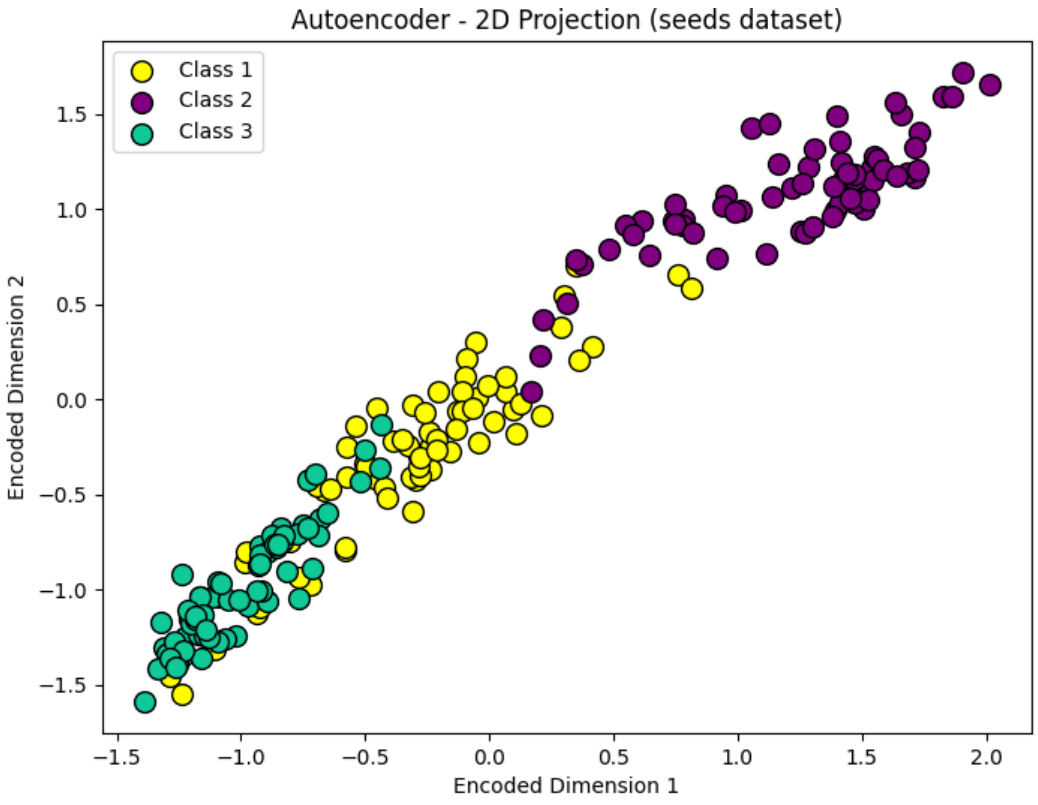
```
    ax.scatter(X_encoded_3D[y == class_value, 0], X_encoded_3D[y == class_value, 1],
X_encoded_3D[y == class_value, 2],
               c=colors[class_value], label=class_names[class_value], edgecolor='k', s=100)

ax.set_title('Autoencoder - 3D Projection (seeds dataset)')
ax.set_xlabel('Encoded Dimension 1')
ax.set_ylabel('Encoded Dimension 2')
ax.set_zlabel('Encoded Dimension 3')
plt.legend()
plt.show()
```




**Реализованный автоэнкодера с тремя скрытыми слоями и выходом на 2 нейрона**

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.neural_network import MLPRegressor

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)
```

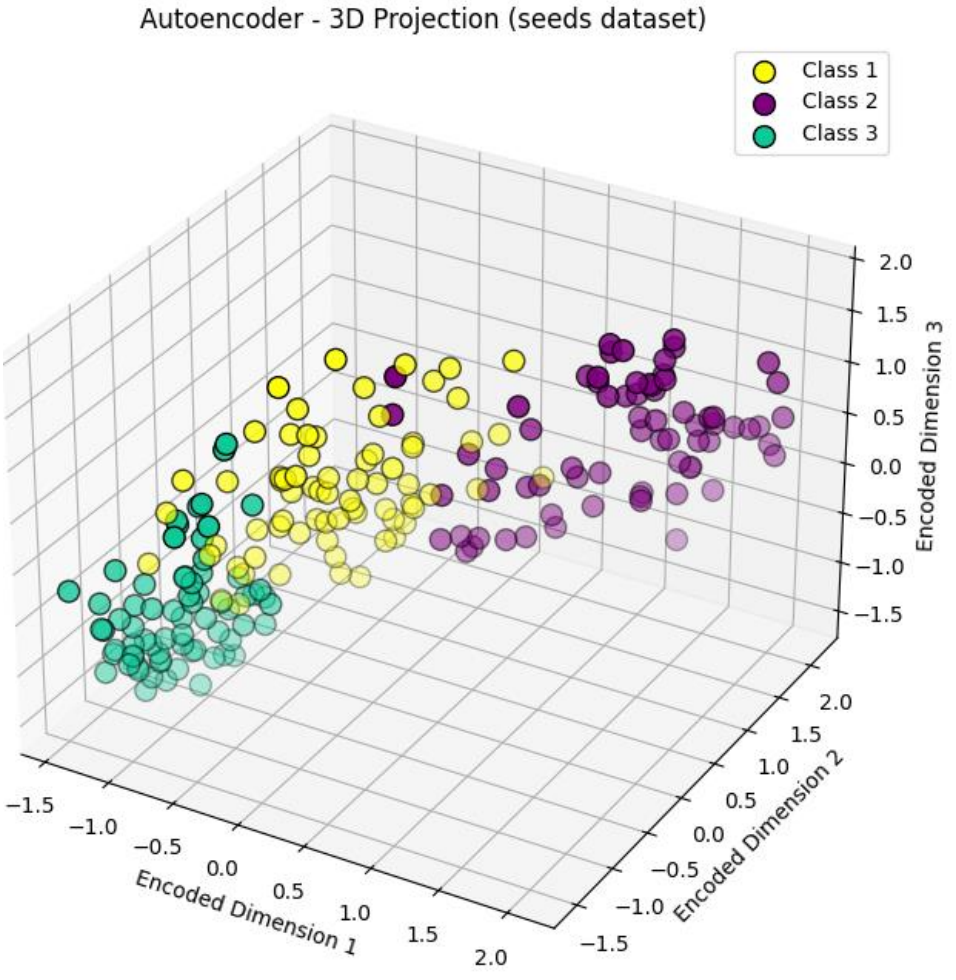
```
autoencoder = MLPRegressor(hidden_layer_sizes=(8, 6, 4), activation='relu', solver='adam',
max_iter=1000)

autoencoder.fit(X_scaled, X_scaled)

X_encoded_2D = autoencoder.predict(X_scaled)

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_encoded_2D[y == class_value, 0], X_encoded_2D[y == class_value, 1],
                c=colors[class_value], label=class_names[class_value], edgecolor='k', s=100)
plt.title('Autoencoder - 2D Projection (seeds dataset)')
plt.xlabel('Encoded Dimension 1')
plt.ylabel('Encoded Dimension 2')
plt.legend()
plt.show()
```




**Реализованный автоэнкодера с тремя скрытыми слоями и выходом на 3 нейрона**

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.neural_network import MLPRegressor

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
```

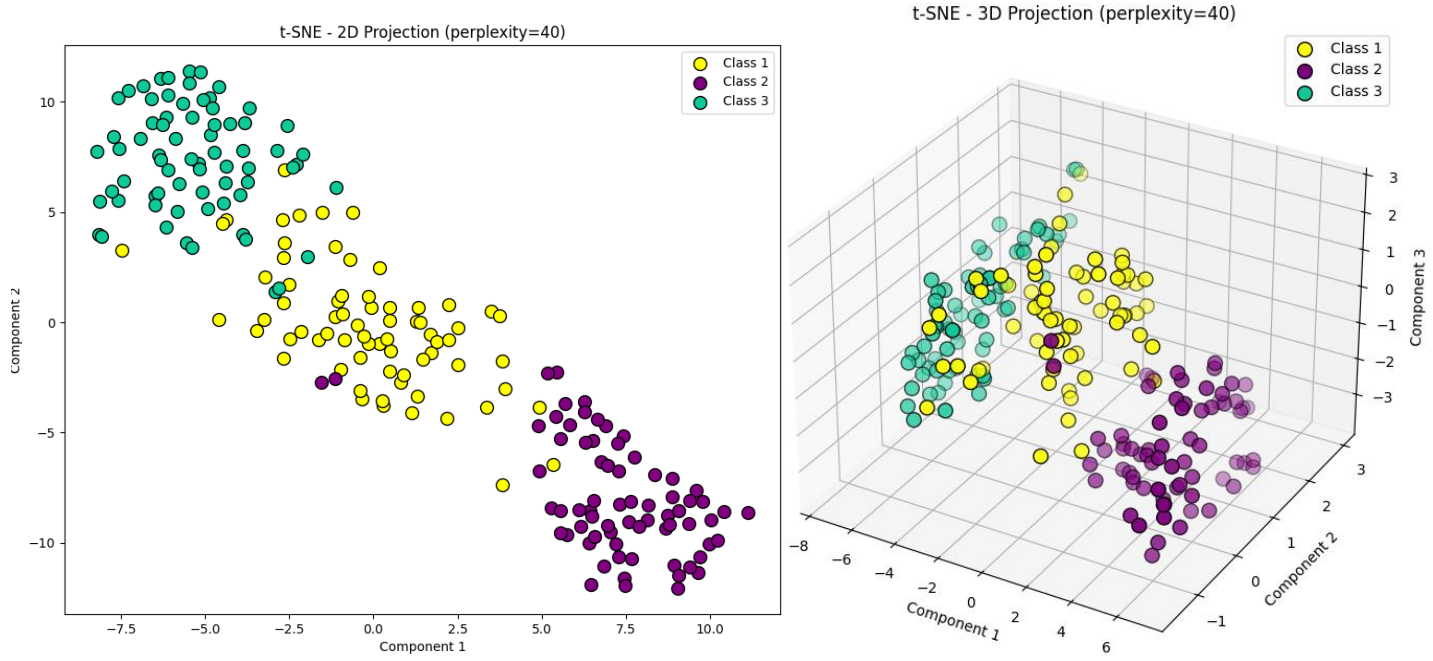
```
X, y = prepare_data()
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

autoencoder_3d = MLPRegressor(hidden_layer_sizes=(8, 6, 4), activation='relu', solver='adam',
max_iter=1000)

autoencoder_3d.fit(X_scaled, X_scaled)

X_encoded_3D = autoencoder_3d.predict(X_scaled)

fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111, projection='3d')
colors = {1: 'yellow', 2: 'purple', 3: '#10C999'}
class_names = {1: 'Class 1', 2: 'Class 2', 3: 'Class 3'}

for class_value in set(y):
    ax.scatter(X_encoded_3D[y == class_value, 0], X_encoded_3D[y == class_value, 1],
X_encoded_3D[y == class_value, 2],
               c=colors[class_value], label=class_names[class_value], edgecolor='k', s=100)

ax.set_title('Autoencoder - 3D Projection (seeds dataset)')
ax.set_xlabel('Encoded Dimension 1')
ax.set_ylabel('Encoded Dimension 2')
ax.set_zlabel('Encoded Dimension 3')
plt.legend()
plt.show()
```



**Реализованный метод t-SNE**

```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.manifold import TSNE
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)

def visualize_tsne_2d(X_scaled, y, perplexity):
    plt.figure(figsize=(10, 8))

    tsne = TSNE(n_components=2, perplexity=perplexity, init='pca', random_state=42)
    X_tsne_2D = tsne.fit_transform(X_scaled)

    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_tsne_2D[y == class_value, 0], X_tsne_2D[y == class_value, 1],
                    c=colors[class_value], label=class_names[class_value], edgecolor='k',
s=100)

    plt.title(f't-SNE - 2D Projection (perplexity={perplexity})')
    plt.xlabel('Component 1')
    plt.ylabel('Component 2')
    plt.legend()
    plt.show()

perplexity = 40
visualize_tsne_2d(X_scaled, y, perplexity)

def visualize_tsne_3d(X_scaled, y, perplexity):
    fig = plt.figure(figsize=(10, 8))

    tsne = TSNE(n_components=3, perplexity=perplexity, init='pca', random_state=42)
    X_tsne_3D = tsne.fit_transform(X_scaled)

    ax = fig.add_subplot(111, projection='3d')
    colors = {1: 'yellow', 2: 'purple', 3: '#10C999'}
    class_names = {1: 'Class 1', 2: 'Class 2', 3: 'Class 3'}
    for class_value in set(y):
        ax.scatter(X_tsne_3D[y == class_value, 0], X_tsne_3D[y == class_value, 1],
X_tsne_3D[y == class_value, 2],
                   c=colors[class_value], label=class_names[class_value], edgecolor='k',
s=100)
```

```
    ax.set_title(f't-SNE - 3D Projection (perplexity={perplexity})')
    ax.set_xlabel('Component 1')
    ax.set_ylabel('Component 2')
    ax.set_zlabel('Component 3')
    ax.legend()
    plt.show()

visualize_tsne_3d(X_scaled, y, perplexity)
```




**Вывод**: на практике научилась использовать метод t-SNE и применять автоэнкодер для визуализации данных.