МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА - Российский технологический университет»

РТУ МИРЭА

Институт Информационных Технологий
Кафедра прикладной математики

ОТЧЁТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 6

«Кластеризация»

по дисциплине

«Технологии и инструментарий анализа больших данных»

Выполнил студент группы Лазарев А. В.

ИВБО-03-21

Принял преподаватель кафедры прикладной математики Тетерин Н.Н.

Практическая работа выполнена « __ » _____ 2024 г.

«Зачтено» « __ » _____ 2024 г.

Москва 2024

СОДЕРЖАНИЕ

1 РЕШЕНИЕ ЗАДАЧ

1.1 Задача №1

Решение программы представлено на Рисунке 1.1.

```
[ ] url = 'https://raw.githubusercontent.com/InspectorJelly/BigDataMirea/refs/heads/main/datasets/data_Statistics.csv'
    data = pd.read_csv(url)
```

Рисунок 1.1 – Программа

1.2 Задача №2

Решение и результат программы представлены на Рисунке 1.2, 1.3.

```
numerical_features = [
    'Placed', 'Eliminations', 'Assists', 'Revives', 'Accuracy', 'Hits',
    'Head Shots', 'Distance Traveled', 'Materials Gathered',
    'Materials Used', 'Damage Taken', 'Damage to Players',
    'Damage to Structures'
]
data_numeric = data[numerical_features]

scaler = StandardScaler()
data_scaled = scaler.fit_transform(data_numeric)

wcss = []  # Within-Cluster Sum of Squares
silhouette_scores = []  # Коэффициент силуэта

k_values = range(2, 11)  # Проверяем значения от 2 до 10 кластеров

for k in k_values:
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(data_scaled)
    wcss.append(kmeans.inertia_)

    labels = kmeans.labels_
    silhouette = silhouette_score(data_scaled, labels)
    silhouette_scores.append(silhouette)

plt.figure(figsize=(10, 5))
plt.plot(k_values, wcss, marker='o', label='WCSS (Rule of Elbow)')
plt.title('Rule of Elbow')
plt.xlabel('Number of clusters (k)')
plt.ylabel('WCSS')
plt.legend()
plt.show()

plt.figure(figsize=(10, 5))
plt.plot(k_values, silhouette_scores, marker='o', label='Silhouette Score')
plt.title('Silhouette Score Analysis')
plt.xlabel('Number of clusters (k)')
plt.ylabel('Silhouette Score')
plt.legend()
plt.show()

optimal_k = k_values[np.argmax(silhouette_scores)]
print(f"Оптимальное количество кластеров по коэффициенту силуэта: {optimal_k}")

final_kmeans = KMeans(n_clusters=optimal_k, random_state=42)
final_kmeans.fit(data_scaled)
data['Cluster'] = final_kmeans.labels_

data.to_csv('clustered_data.csv', index=False)
print("Результаты кластеризации сохранены в 'clustered_data.csv'.")
```

Рисунок 1.2 – Программа

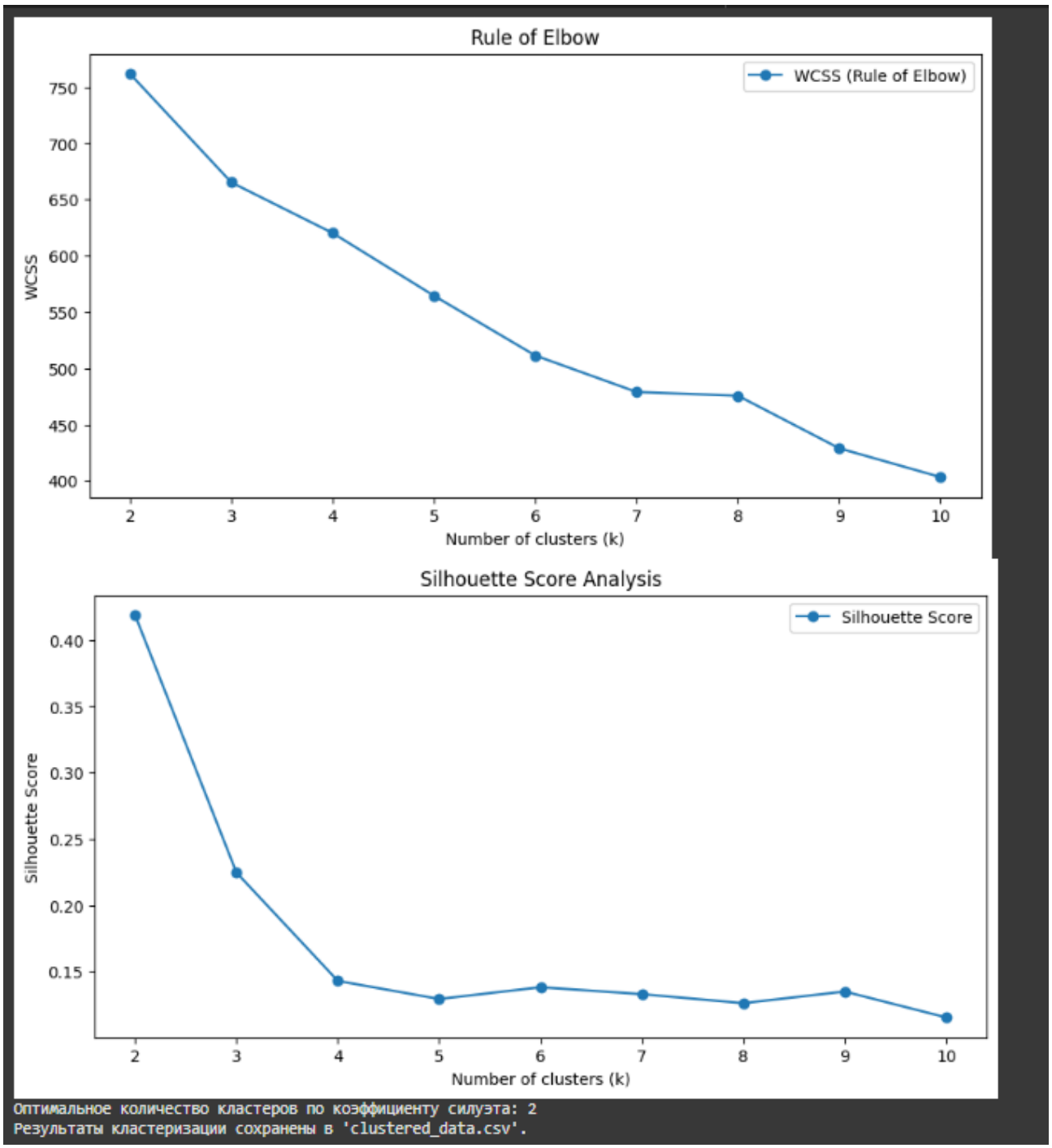

Рисунок 1.3 – Результат выполнения программы

1.3 Задача №3

Решение и результат программы представлены на Рисунке 1.4, 1.5.

```python
numerical_features = [
    'Placed', 'Eliminations', 'Assists', 'Revives', 'Accuracy', 'Hits',
    'Head Shots', 'Distance Traveled', 'Materials Gathered',
    'Materials Used', 'Damage Taken', 'Damage to Players',
    'Damage to Structures'
]
data_numeric = data[numerical_features]

scaler = StandardScaler()
data_scaled = scaler.fit_transform(data_numeric)

linked = linkage(data_scaled, method='ward')  # Метод "ward" минимизирует дисперсию внутри кластеров

plt.figure(figsize=(15, 7))
dendrogram(linked, orientation='top', distance_sort='descending', show_leaf_counts=True)
plt.title('Дендрограмма')
plt.xlabel('Образцы данных')
plt.ylabel('Расстояние')
plt.show()

num_clusters = 4
clusters = fcluster(linked, num_clusters, criterion='maxclust')

data['Cluster'] = clusters
```

Рисунок 1.4 – Программа

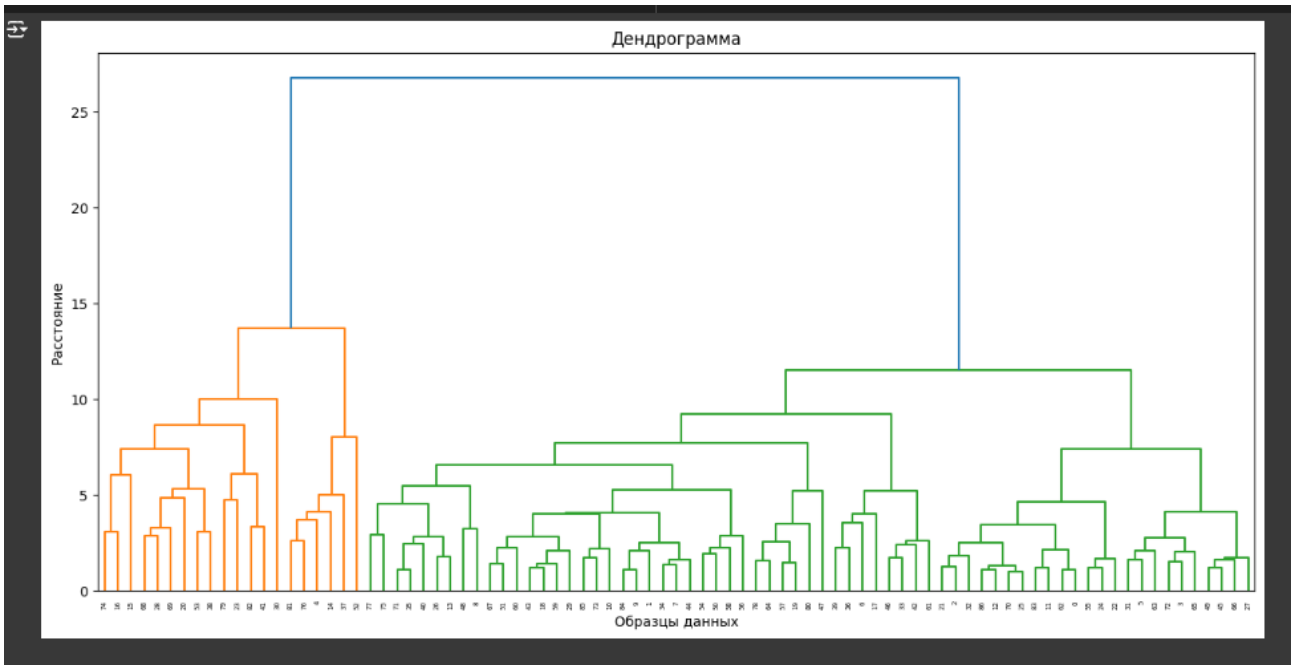

Рисунок 1.5 – Результат выполнения программы

1.4 Задача №4

Решение и результат программы представлены на Рисунках 1.6, 1.7.

```
numerical_features = [
    'Placed', 'Eliminations', 'Assists', 'Revives', 'Accuracy', 'Hits',
    'Head Shots', 'Distance Traveled', 'Materials Gathered',
    'Materials Used', 'Damage Taken', 'Damage to Players',
    'Damage to Structures'
]
data_numeric = data[numerical_features]

scaler = StandardScaler()
data_scaled = scaler.fit_transform(data_numeric)

eps = 1.5  # Максимальное расстояние между соседними точками
min_samples = 5  # Минимальное количество точек для образования кластера

dbscan = DBSCAN(eps=eps, min_samples=min_samples)
data['Cluster'] = dbscan.fit_predict(data_scaled)

n_clusters = len(set(data['Cluster'])) - (1 if -1 in data['Cluster'] else 0)
n_noise = list(data['Cluster']).count(-1)

print(f"Количество кластеров: {n_clusters}")
print(f"Количество шумовых точек: {n_noise}")

from sklearn.decomposition import PCA

pca = PCA(n_components=2)
data_pca = pca.fit_transform(data_scaled)

plt.figure(figsize=(10, 7))
plt.scatter(data_pca[:, 0], data_pca[:, 1], c=data['Cluster'], cmap='viridis', s=50, alpha=0.7)
plt.title('DBSCAN Clustering')
plt.xlabel('PCA Component 1')
plt.ylabel('PCA Component 2')
plt.colorbar(label='Cluster')
plt.show()
```

Рисунок 1.6 – Программа

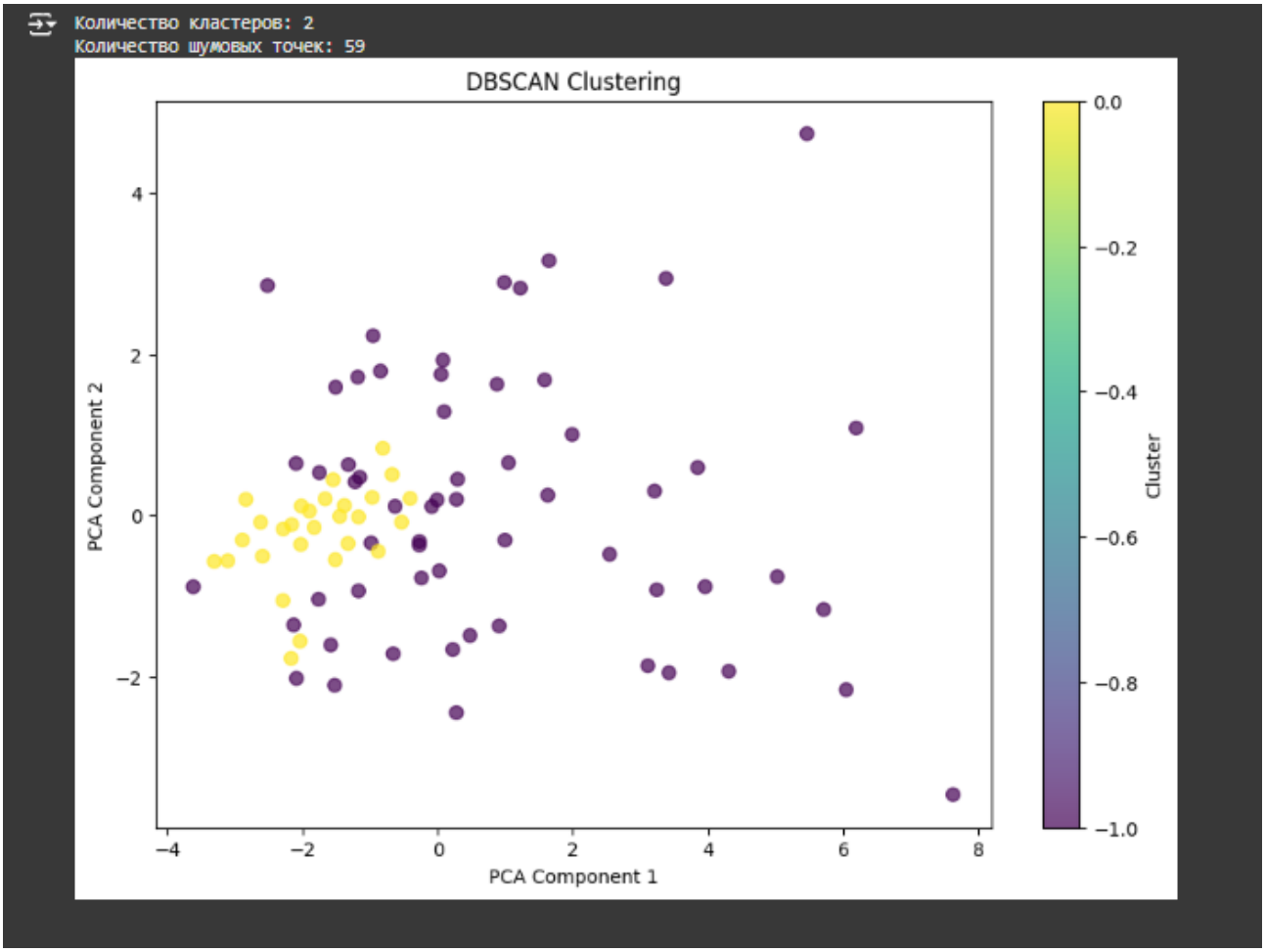

Рисунок 1.7 – Результат выполнения программы

1.5 Задача №5

Решение и результат программы представлены на Рисунке 1.8, 1.9, 1.10.

```
numerical_features = [
    'Placed', 'Eliminations', 'Assists', 'Revives', 'Accuracy', 'Hits',
    'Head Shots', 'Distance Traveled', 'Materials Gathered',
    'Materials Used', 'Damage Taken', 'Damage to Players',
    'Damage to Structures'
]
data_numeric = data[numerical_features]
clusters = data['Cluster']

tsne = TSNE(n_components=2, perplexity=30, random_state=42)
data_tsne = tsne.fit_transform(data_numeric)

plt.figure(figsize=(10, 7))
plt.scatter(data_tsne[:, 0], data_tsne[:, 1], c=clusters, cmap='viridis', s=50, alpha=0.7)
plt.title('t-SNE Visualization of Clusters')
plt.xlabel('t-SNE Component 1')
plt.ylabel('t-SNE Component 2')
plt.colorbar(label='Cluster')
plt.show()

tsne_3d = TSNE(n_components=3, perplexity=30, random_state=42)
data_tsne_3d = tsne_3d.fit_transform(data_numeric)

fig = plt.figure(figsize=(10, 7))
ax = fig.add_subplot(111, projection='3d')
scatter = ax.scatter(data_tsne_3d[:, 0], data_tsne_3d[:, 1], data_tsne_3d[:, 2],
                     c=clusters, cmap='viridis', s=50, alpha=0.7)
plt.title('3D t-SNE Visualization of Clusters')
plt.colorbar(scatter, label='Cluster')
plt.show()
```

Рисунок 1.8 – Программа

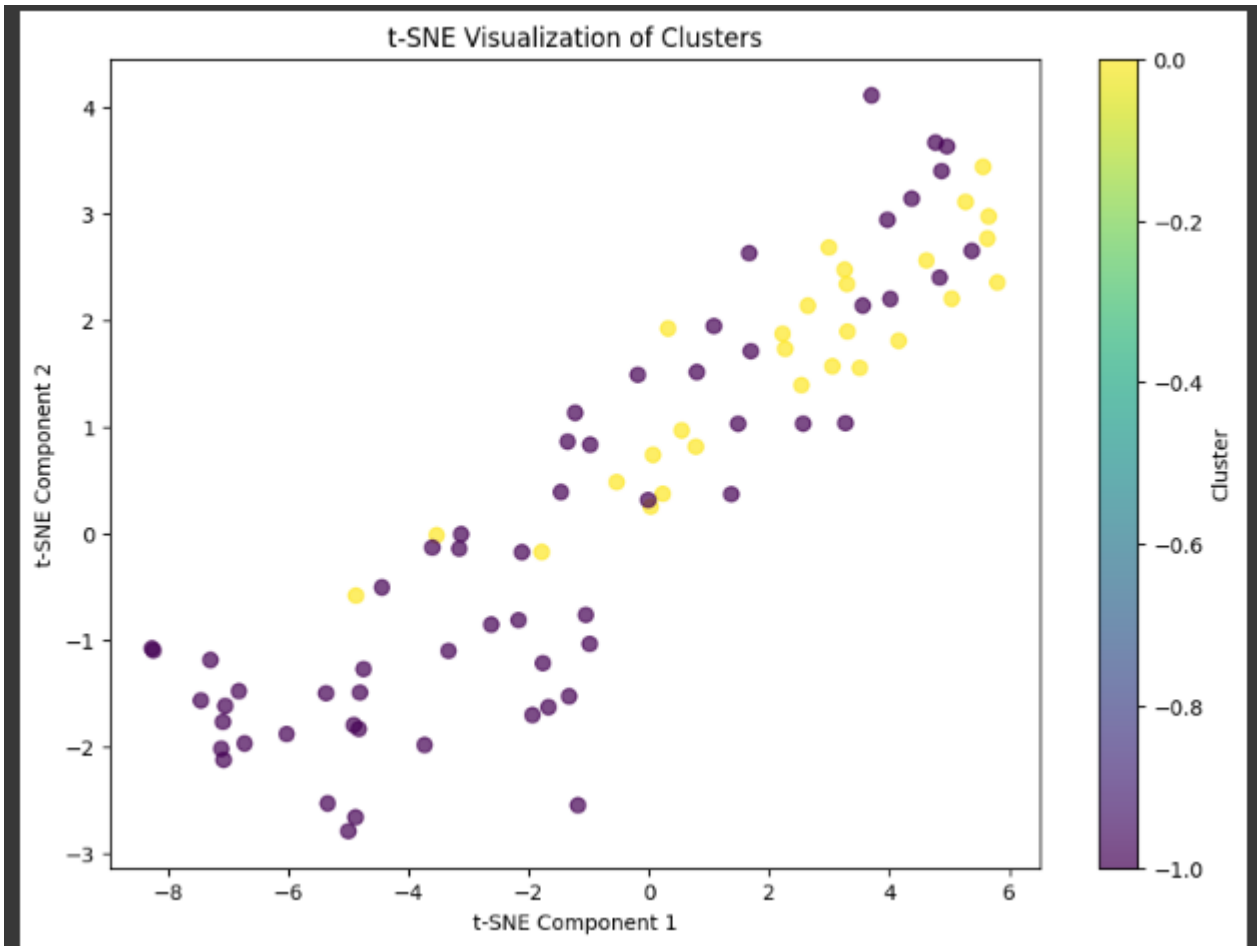

Рисунок 1.9 – Результат выполнения программы в 2D

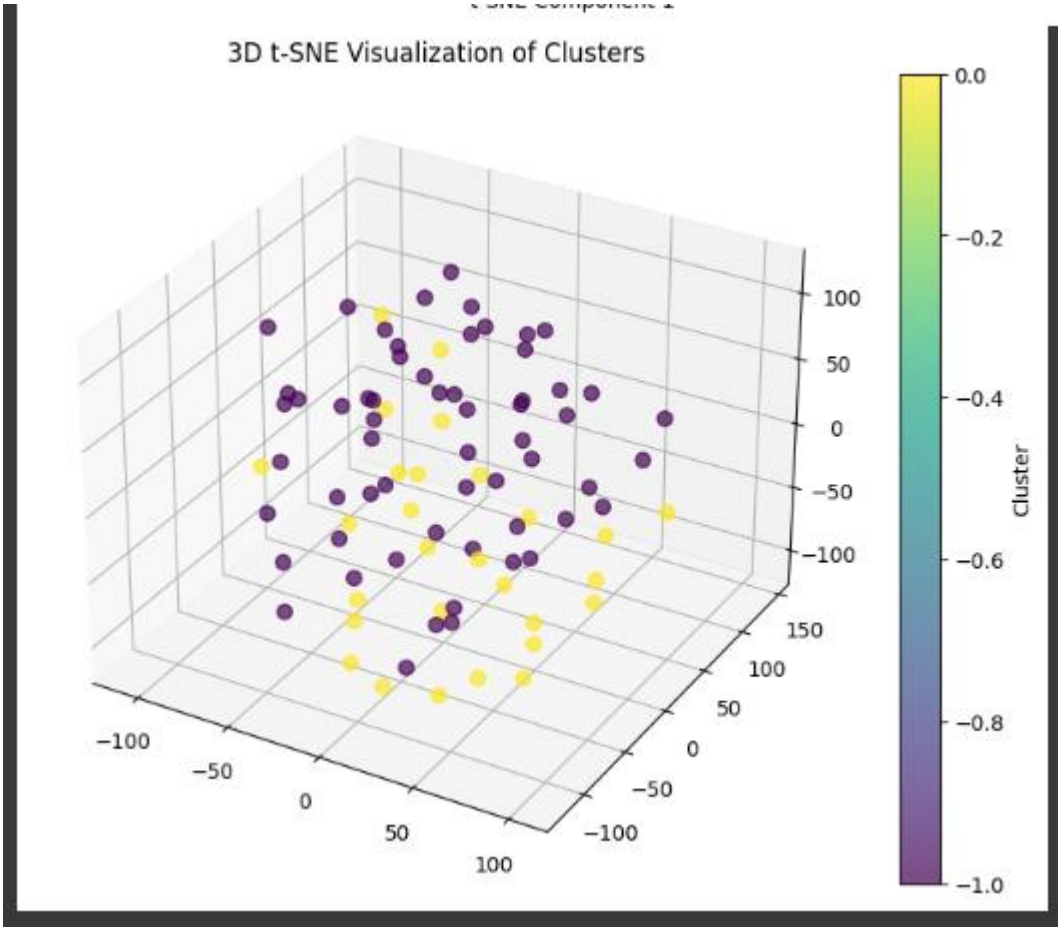

Рисунок 1.10 – Результат выполнения программы в 3D