Лабораторая работа №5

Задача: Провести полный анализ для сегментации датасета

Описание датасета "Predict Online Gaming Behavior Dataset"

Этот набор данных фиксирует комплексные метрики и демографию, связанные с поведением игроков в онлайн-игровых средах. Он включает такие переменные, как демография игроков, детали, характерные для игры, метрики вовлеченности и целевую переменную, отражающую удержание игроков.

Переменная	Описание
PlayerID	Уникальный идентификатор для каждого игрока.
Age	Возраст игрока.
Gender	Пол игрока.
Location	Географическое местоположение игрока.
GameGenre	Жанр игры, в которой участвует игрок.
PlayTimeHours	Среднее количество часов, проведенных за игрой за одну сессию.
InGamePurchases	Признак того, делает ли игрок внутриигровые покупки (0 — Нет, 1 — Да).
GameDifficulty	Уровень сложности игры.
SessionsPerWeek	Количество игровых сессий в неделю.
AvgSessionDurationMinutes	Средняя продолжительность каждой игровой сессии в минутах.
PlayerLevel	Текущий уровень игрока в игре.
AchievementsUnlocked	Количество достижений, разблокированных игроком.
EngagementLevel	Категоризированный уровень вовлеченности, отражающий удержание игроков ('Высокий', 'Средний', 'Низкий').

Целевая переменная — EngagementLevel — указывает на уровень вовлеченности игрока и категоризируется как 'Высокий', 'Средний' или 'Низкий'.

In [17]:
```
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import silhouette_score, calinski_harabasz_score, davies_bouldin_score
```

In [ ]:
```
df = pd.read_csv('../online_gaming_behavior_dataset.csv')
```

In [20]:
```
df.head()
```

Out[20]:

	PlayerID	Age	Gender	Location	GameGenre	PlayTimeHours	InGamePurchases	GameDifficulty	SessionsPerWeek	AvgSessior
0	9000	43	Male	Other	Strategy	16.271119	0	Medium	6	
1	9001	29	Female	USA	Strategy	5.525961	0	Medium	5	
2	9002	22	Female	USA	Sports	8.223755	0	Easy	16	
3	9003	35	Male	USA	Action	5.265351	1	Easy	9	
4	9004	33	Male	Europe	Action	15.531945	0	Medium	2	

Преобразование данных

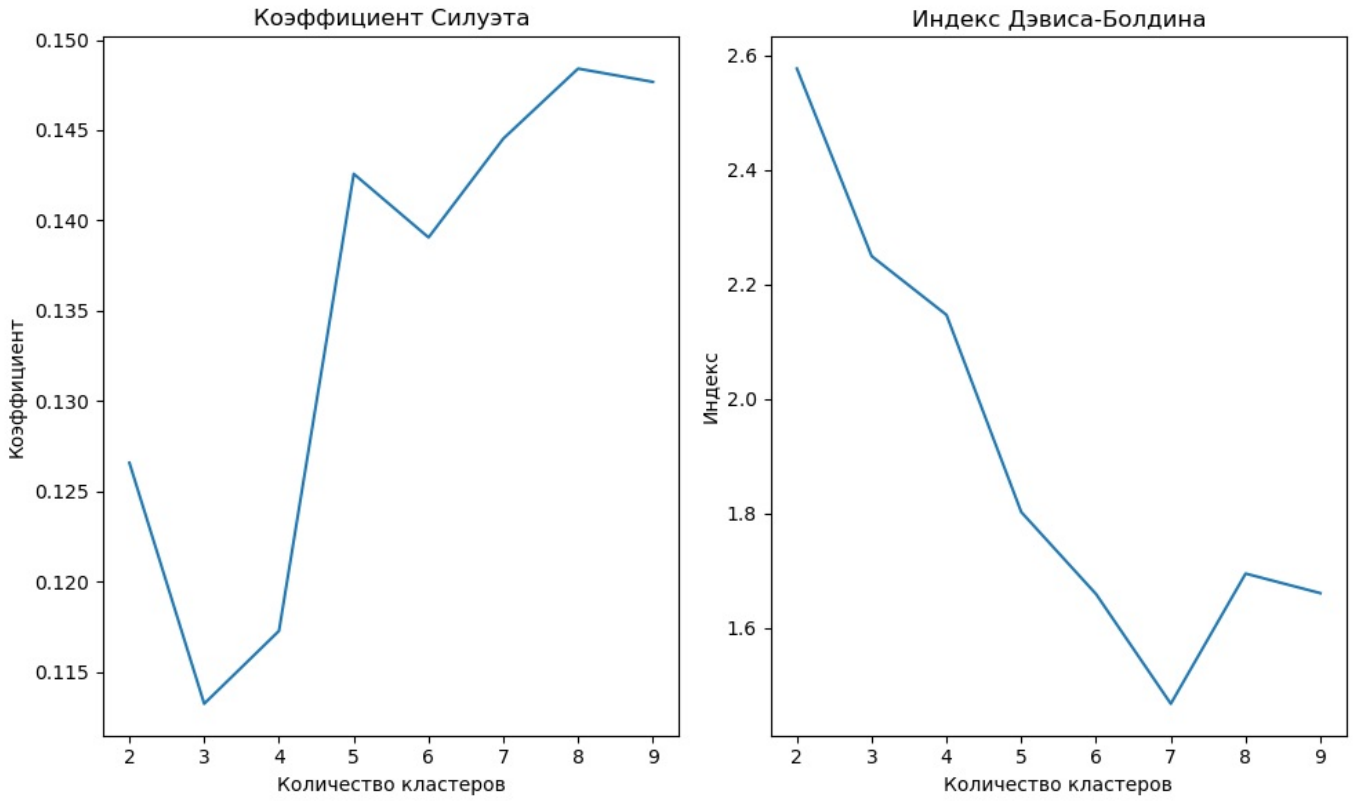
In [21]:
```
# Преобразование категориальных переменных
df = pd.get_dummies(df, columns=['Gender', 'Location', 'GameGenre', "GameDifficulty"], drop_first=True)
# Преобразование целевой переменной в числовой формат
engagement_map = {'Low': 0, 'Medium': 1, 'High': 2}
df['EngagementLevel'] = df['EngagementLevel'].map(engagement_map)
# Масштабирование данных
scaler = StandardScaler()
data_scaled = scaler.fit_transform(df.drop('EngagementLevel', axis=1)[['PlayTimeHours', "InGamePurchases", "Ses
                                                                       "AvgSessionDurationMinutes", "PlayerLeve
```

In [28]:
```
scaler = StandardScaler()
data_scaled_with_cat = scaler.fit_transform(df.drop('EngagementLevel', axis=1))
```

Оценка качества

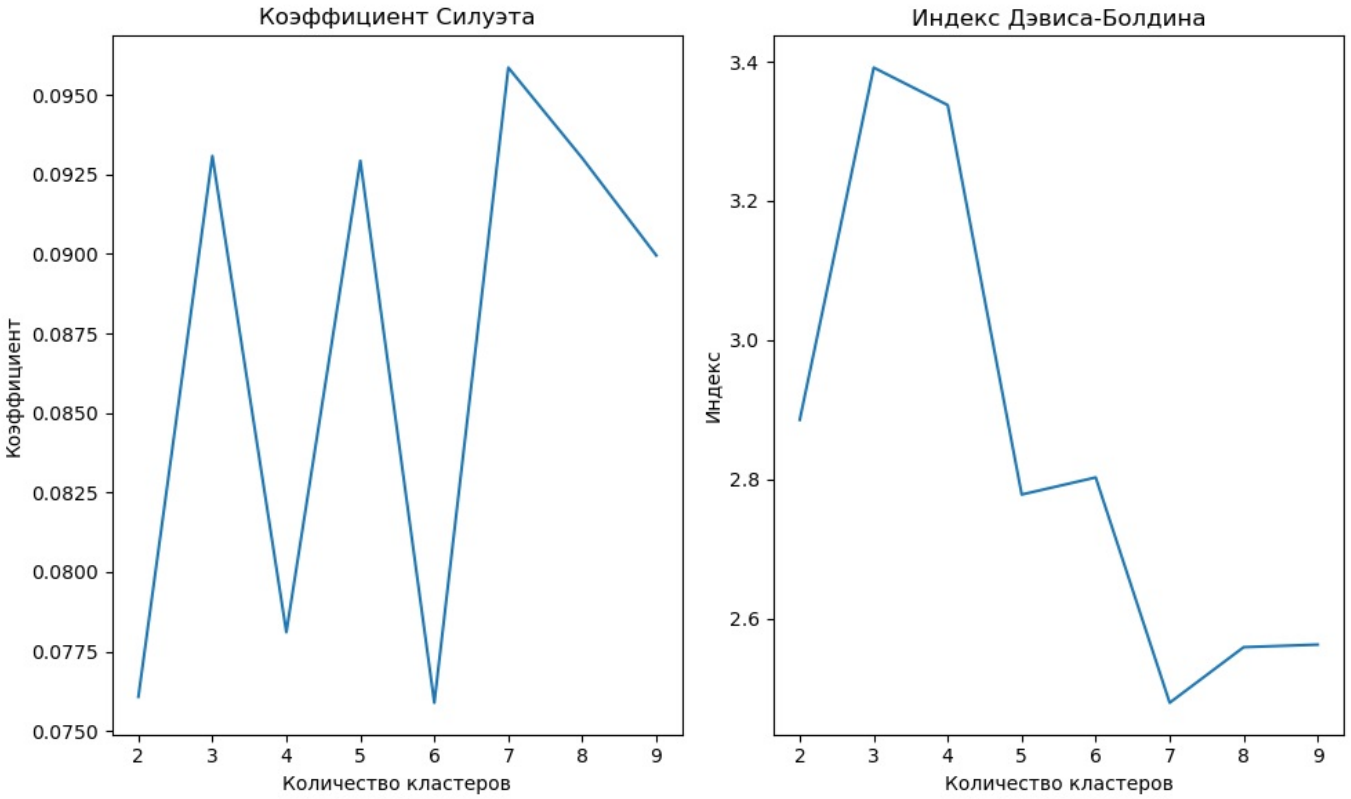
Будет использоваться алгоритм K-Means для сегментации данных. Для определения оптимального количества кластеров будем использовать коэффициент силуэта и индекс дэвиса-болдина

In [31]:
```
# Определение оптимального количества кластеров
silhouette_scores = []
calinski_harabasz_scores = []
davies_bouldin_scores = []

for k in range(2, 10):
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(data_scaled)

    silhouette = silhouette_score(data_scaled, kmeans.labels_)
    calinski_harabasz = calinski_harabasz_score(data_scaled, kmeans.labels_)
    davies_bouldin = davies_bouldin_score(data_scaled, kmeans.labels_)

    silhouette_scores.append(silhouette)
    calinski_harabasz_scores.append(calinski_harabasz)
    davies_bouldin_scores.append(davies_bouldin)

# Визуализация результатов
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))

plt.subplot(1, 2, 1)
plt.plot(range(2, 10), silhouette_scores)
plt.title('Коэффициент Силуэта')
plt.xlabel('Количество кластеров')
plt.ylabel('Коэффициент')


plt.subplot(1, 2, 2)
plt.plot(range(2, 10), davies_bouldin_scores)
plt.title('Индекс Дэвиса-Болдина')
plt.xlabel('Количество кластеров')
plt.ylabel('Индекс')

plt.tight_layout()
plt.show()
```



In [30]:
```
# Определение оптимального количества кластеров
silhouette_scores = []
calinski_harabasz_scores = []
davies_bouldin_scores = []

for k in range(2, 10):
    kmeans = KMeans(n_clusters=k, random_state=42)
```

```
    kmeans.fit(data_scaled_with_cat)

    silhouette = silhouette_score(data_scaled_with_cat, kmeans.labels_)
    calinski_harabasz = calinski_harabasz_score(data_scaled_with_cat, kmeans.labels_)
    davies_bouldin = davies_bouldin_score(data_scaled_with_cat, kmeans.labels_)
    silhouette_scores.append(silhouette)
    calinski_harabasz_scores.append(calinski_harabasz)
    davies_bouldin_scores.append(davies_bouldin)

# Визуализация результатов
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))

plt.subplot(1, 2, 1)
plt.plot(range(2, 10), silhouette_scores)
plt.title('Коэффициент Силуэта')
plt.xlabel('Количество кластеров')
plt.ylabel('Коэффициент')


plt.subplot(1, 2, 2)
plt.plot(range(2, 10), davies_bouldin_scores)
plt.title('Индекс Дэвиса-Болдина')
plt.xlabel('Количество кластеров')
plt.ylabel('Индекс')

plt.tight_layout()
plt.show()
```



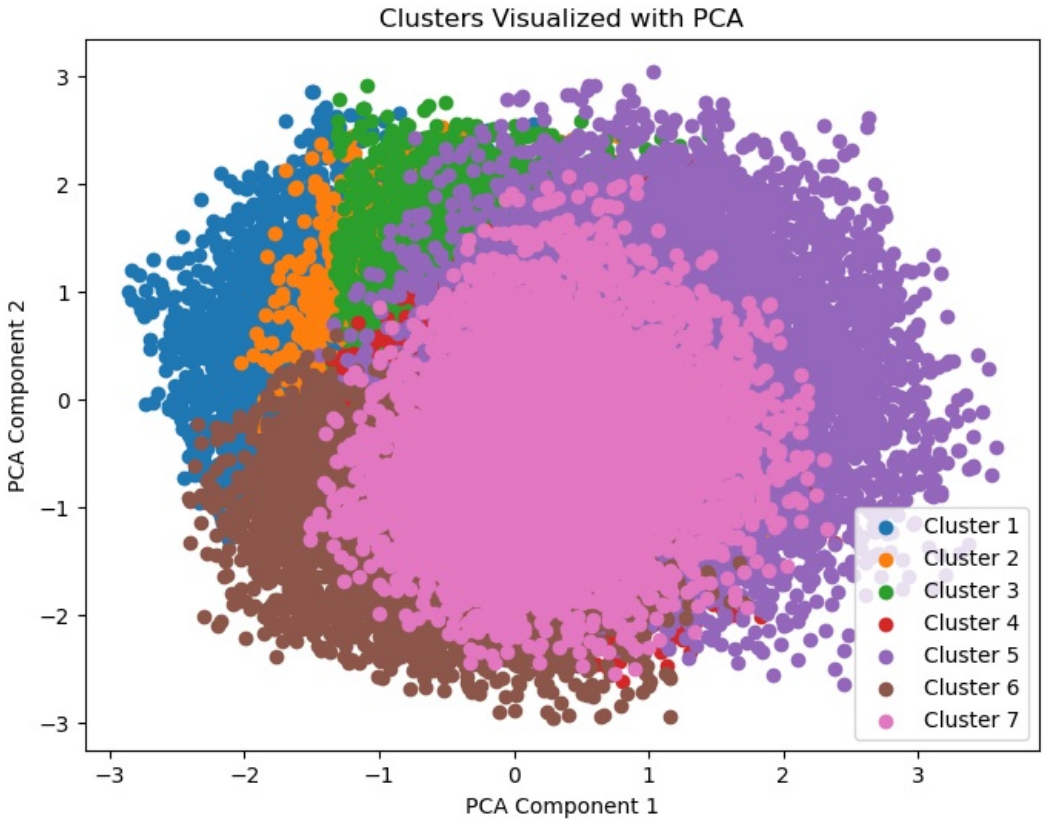
Мы видим, что если включать категориальные переменные в данные, то у нас падают значение наших коэффициентов, поэтому далее будем рассматривать данные без категориальных перменных

In [ ]:
```
# Выбор оптимального количества кластеров на основе графиков
optimal_k = 7

kmeans_optimal = KMeans(n_clusters=optimal_k, random_state=42)
kmeans_optimal.fit(data_scaled)


cluster_labels = kmeans_optimal.labels_

for i in range(optimal_k):
    cluster_data = df[cluster_labels == i]
    print(f"Кластер {i+1}:")
    display(cluster_data.describe())
    print()
```

Визуализация результатов

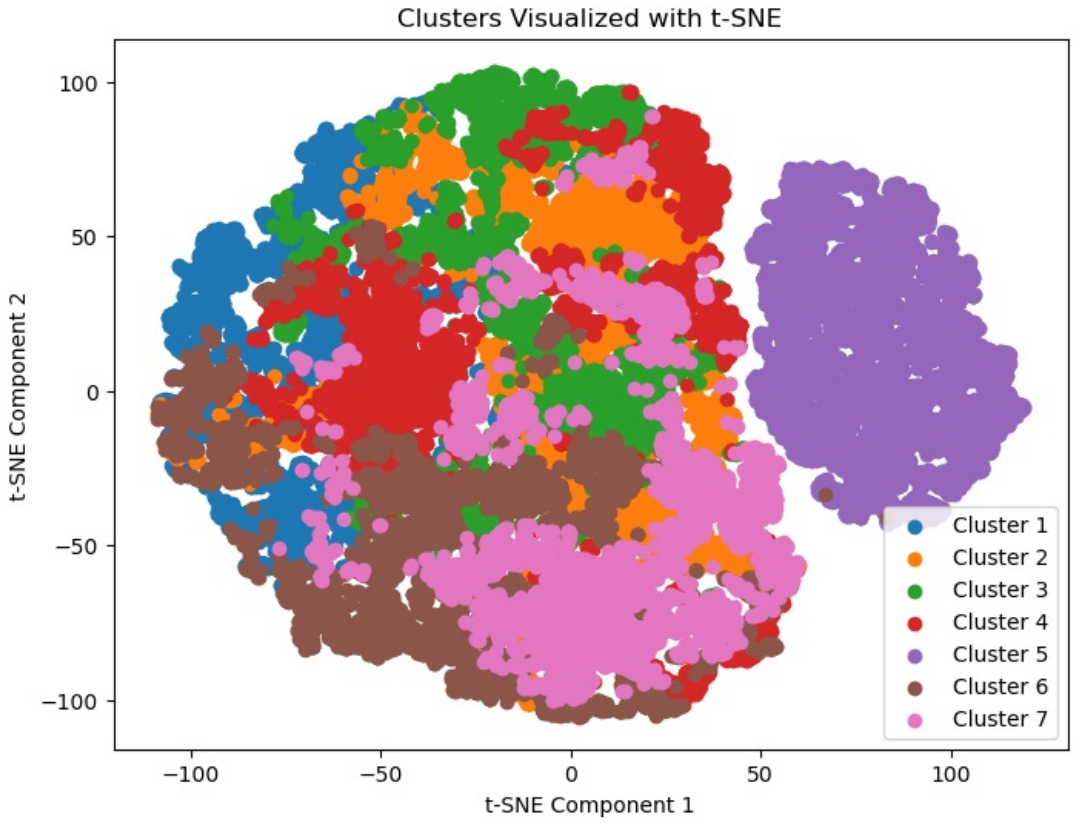
PCA

In [ ]:
```
from sklearn.decomposition import PCA

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

plt.figure(figsize=(8, 6))
for cluster in range(optimal_k):
    plt.scatter(data_pca[cluster_labels == cluster, 0],
                data_pca[cluster_labels == cluster, 1],
                label=f'Cluster {cluster + 1}')

plt.title('Clusters Visualized with PCA')
plt.xlabel('PCA Component 1')
plt.ylabel('PCA Component 2')
plt.legend()
plt.show()
```



T-SNE

In [ ]:
```
from sklearn.manifold import TSNE

tsne = TSNE(n_components=2, random_state=42)
data_tsne = tsne.fit_transform(data_scaled)

plt.figure(figsize=(8, 6))
for cluster in range(optimal_k):
    plt.scatter(data_tsne[cluster_labels == cluster, 0],
                data_tsne[cluster_labels == cluster, 1],
                label=f'Cluster {cluster + 1}')

plt.title('Clusters Visualized with t-SNE')
plt.xlabel('t-SNE Component 1')
plt.ylabel('t-SNE Component 2')
plt.legend()
plt.show()
```



In [ ]: