Лабораторная работа №4 Сравнение и оценка методов кластеризации

Задача: Сравнить результаты разных методов кластеризации (KMeans, KMeans++, DBSCAN) и определить оптимальное число кластеров

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 [6]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans, DBSCAN
from sklearn.metrics import silhouette_score
```

In [38]:
```
df = pd.read_csv("../online_gaming_behavior_dataset.csv", index_col='PlayerID')
```

In [39]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 40034 entries, 9000 to 49033
Data columns (total 12 columns):
 #   Column                     Non-Null Count  Dtype  
---  ------                     --------------  -----  
 0   Age                        40034 non-null  int64  
 1   Gender                     40034 non-null  object 
 2   Location                   40034 non-null  object 
 3   GameGenre                  40034 non-null  object 
 4   PlayTimeHours              40034 non-null  float64
 5   InGamePurchases            40034 non-null  int64  
 6   GameDifficulty             40034 non-null  object 
 7   SessionsPerWeek            40034 non-null  int64  
 8   AvgSessionDurationMinutes  40034 non-null  int64  
 9   PlayerLevel                40034 non-null  int64  
 10  AchievementsUnlocked       40034 non-null  int64  
 11  EngagementLevel            40034 non-null  object 
dtypes: float64(1), int64(6), object(5)
memory usage: 4.0+ MB
```

Преобразование данных

In [40]:
```
# 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()
scaled_data = scaler.fit_transform(df.drop('EngagementLevel', axis=1)[['PlayTimeHours', "InGamePurchases", "Ses
                                                                       "AvgSessionDurationMinutes", "PlayerLeve
```

In [41]:
```
df.info()
```

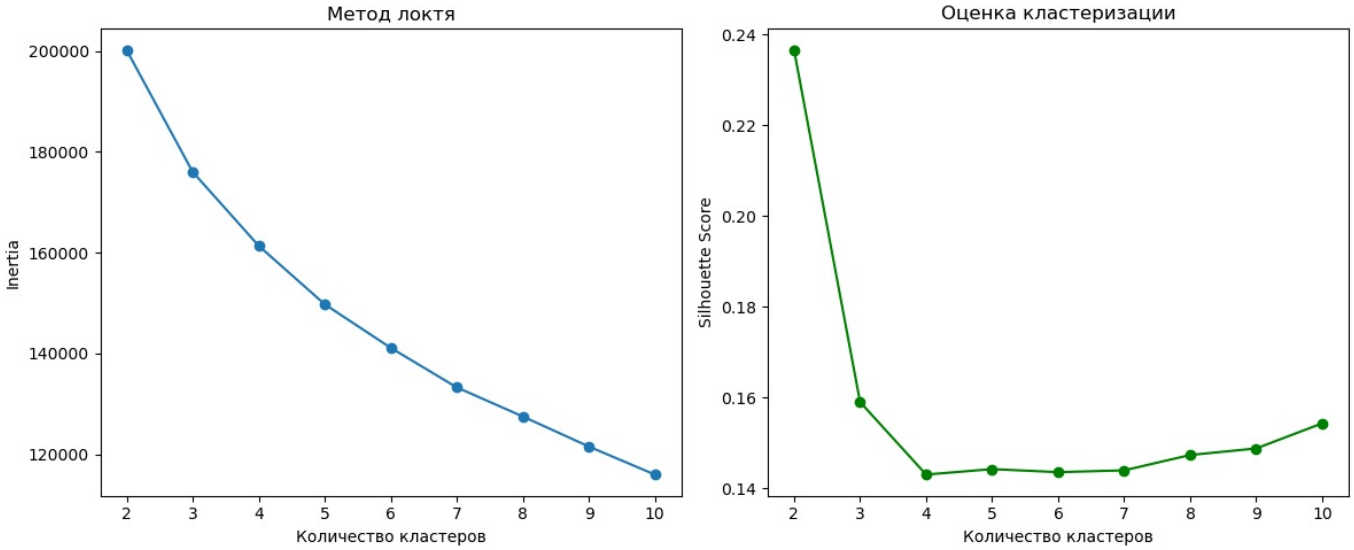
```
<class 'pandas.core.frame.DataFrame'>
Index: 40034 entries, 9000 to 49033
Data columns (total 18 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   Age                        40034 non-null  int64
 1   PlayTimeHours              40034 non-null  float64
 2   InGamePurchases            40034 non-null  int64
 3   SessionsPerWeek            40034 non-null  int64
 4   AvgSessionDurationMinutes  40034 non-null  int64
 5   PlayerLevel                40034 non-null  int64
 6   AchievementsUnlocked       40034 non-null  int64
 7   EngagementLevel            40034 non-null  int64
 8   Gender_Male                40034 non-null  bool
 9   Location_Europe            40034 non-null  bool
 10  Location_Other             40034 non-null  bool
 11  Location_USA               40034 non-null  bool
 12  GameGenre_RPG              40034 non-null  bool
 13  GameGenre_Simulation       40034 non-null  bool
 14  GameGenre_Sports           40034 non-null  bool
 15  GameGenre_Strategy         40034 non-null  bool
 16  GameDifficulty_Hard        40034 non-null  bool
 17  GameDifficulty_Medium      40034 non-null  bool
dtypes: bool(10), float64(1), int64(7)
memory usage: 3.1 MB
```

Основные алгоритмы кластеризации

KMeans

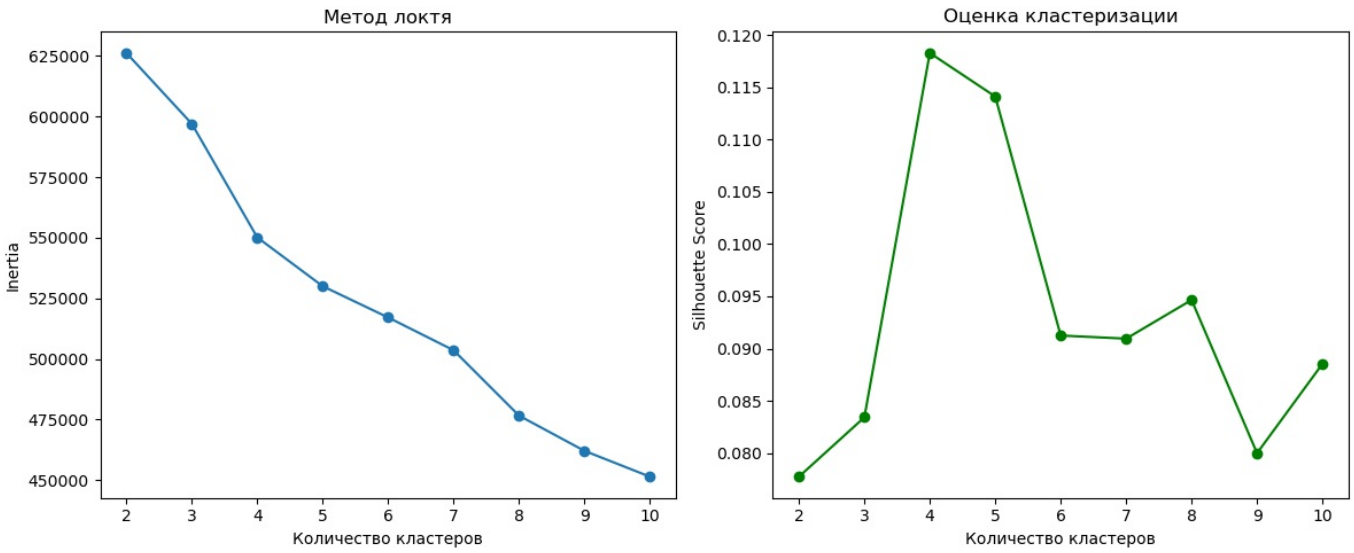
In [43]:
```
# Определение оптимального числа кластеров для KMeans
inertia = []
silhouette_scores = []
cluster_range = range(2, 11)  # перебираем от 2 до 10 кластеров

for k in cluster_range:
    kmeans = KMeans(n_clusters=k, init='random', random_state=42)
    labels = kmeans.fit_predict(scaled_data)
    inertia.append(kmeans.inertia_)
    score = silhouette_score(scaled_data, labels)
    silhouette_scores.append(score)
    print(f"Количество кластеров: {k}, Inertia: {kmeans.inertia_:.2f}, Silhouette Score: {score:.3f}")

# Визуализация метода локтя
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plt.plot(cluster_range, inertia, marker='o')
plt.xlabel('Количество кластеров')
plt.ylabel('Inertia')
plt.title('Метод локтя')

# Визуализация силуэтного коэффициента
plt.subplot(1, 2, 2)
plt.plot(cluster_range, silhouette_scores, marker='o', color='green')
plt.xlabel('Количество кластеров')
plt.ylabel('Silhouette Score')
plt.title('Оценка кластеризации')
plt.tight_layout()
plt.show()
```

```
Количество кластеров: 2, Inertia: 200165.39, Silhouette Score: 0.237
Количество кластеров: 3, Inertia: 176018.15, Silhouette Score: 0.159
Количество кластеров: 4, Inertia: 161371.81, Silhouette Score: 0.143
Количество кластеров: 5, Inertia: 149778.56, Silhouette Score: 0.144
Количество кластеров: 6, Inertia: 141142.04, Silhouette Score: 0.144
Количество кластеров: 7, Inertia: 133308.08, Silhouette Score: 0.144
Количество кластеров: 8, Inertia: 127536.32, Silhouette Score: 0.147
Количество кластеров: 9, Inertia: 121581.11, Silhouette Score: 0.149
Количество кластеров: 10, Inertia: 116021.88, Silhouette Score: 0.154
```



In [ ]:
```python
# Определение оптимального числа кластеров для KMeans++
inertia = []
silhouette_scores = []
cluster_range = range(2, 11)  # перебираем от 2 до 10 кластеров

for k in cluster_range:
    kmeans = KMeans(n_clusters=k, init='k-means++', random_state=42)
    labels = kmeans.fit_predict(scaled_data)
    inertia.append(kmeans.inertia_)
    score = silhouette_score(scaled_data, labels)
    silhouette_scores.append(score)
    print(f"Количество кластеров: {k}, Inertia: {kmeans.inertia_:.2f}, Silhouette Score: {score:.3f}")

# Визуализация метода локтя
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plt.plot(cluster_range, inertia, marker='o')
plt.xlabel('Количество кластеров')
plt.ylabel('Inertia')
plt.title('Метод локтя')

# Визуализация силуэтного коэффициента
plt.subplot(1, 2, 2)
plt.plot(cluster_range, silhouette_scores, marker='o', color='green')
plt.xlabel('Количество кластеров')
plt.ylabel('Silhouette Score')
plt.title('Оценка кластеризации')
plt.tight_layout()
plt.show()
```

```
Количество кластеров: 2, Inertia: 626213.24, Silhouette Score: 0.078
Количество кластеров: 3, Inertia: 596992.06, Silhouette Score: 0.083
Количество кластеров: 4, Inertia: 550140.83, Silhouette Score: 0.118
Количество кластеров: 5, Inertia: 530002.05, Silhouette Score: 0.114
Количество кластеров: 6, Inertia: 517117.52, Silhouette Score: 0.091
Количество кластеров: 7, Inertia: 503588.07, Silhouette Score: 0.091
Количество кластеров: 8, Inertia: 476618.34, Silhouette Score: 0.095
Количество кластеров: 9, Inertia: 462091.56, Silhouette Score: 0.080
Количество кластеров: 10, Inertia: 451438.63, Silhouette Score: 0.089
```



In [44]:
```
optimal_k = 5
kmeans = KMeans(n_clusters=optimal_k, init='k-means++', random_state=42)
kmeans_labels = kmeans.fit_predict(scaled_data)

df['KMeans_Cluster'] = kmeans_labels

print("Распределение по кластерам (KMeans):")
print(df['KMeans_Cluster'].value_counts())
```

```
Распределение по кластерам (KMeans):
KMeans_Cluster
0    8146
4    8041
2    8037
1    7999
3    7811
Name: count, dtype: int64
```

DBSCAN

In [51]:
```
# Подбор параметров для DBSCAN (eps и min_samples)
dbscan = DBSCAN(eps=0.7, min_samples=7)
dbscan_labels = dbscan.fit_predict(scaled_data)

# Добавляем метки DBSCAN в DataFrame
df['DBSCAN_Cluster'] = dbscan_labels

print("Распределение по кластерам (DBSCAN):")
print(df['DBSCAN_Cluster'].value_counts())
```

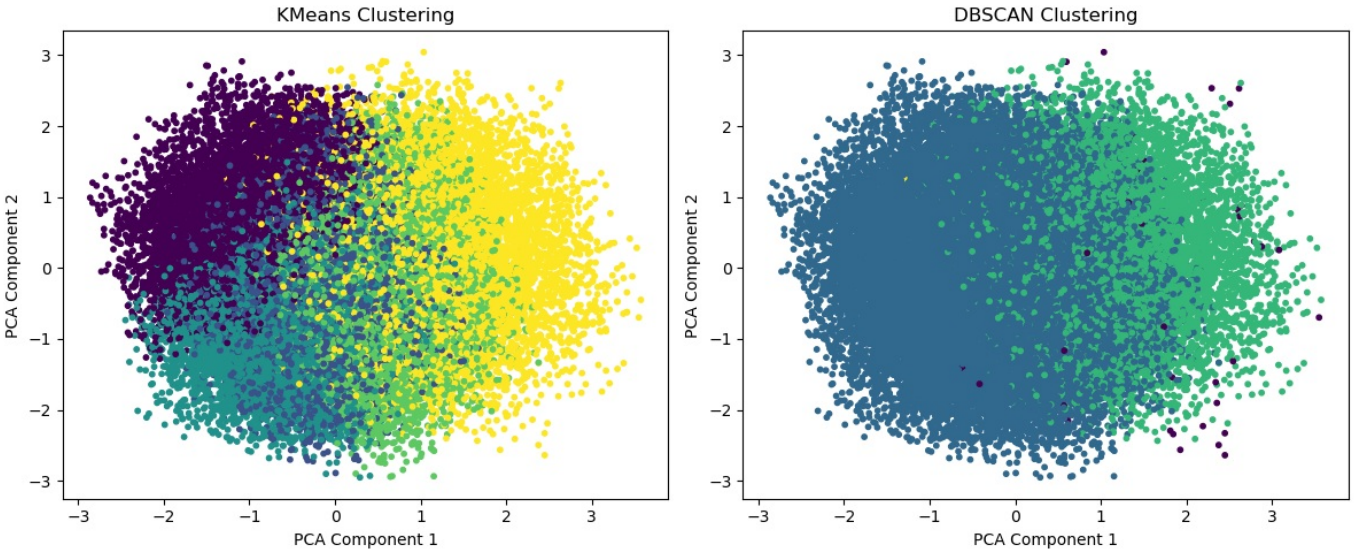
```
Распределение по кластерам (DBSCAN):
DBSCAN_Cluster
 0    31993
 1     7965
-1       72
 2        4
Name: count, dtype: int64
```

Сравнение результатов

In [ ]:
```
# Сравнение результатов кластеризации и визуализация
kmeans_silhouette = silhouette_score(scaled_data, kmeans_labels)
print("Silhouette Score для KMeans:", kmeans_silhouette)
```

```
# Если DBSCAN выделил несколько кластеров (не только шум), можно оценить силуэтный коэффициент
if len(set(dbscan_labels)) > 1 and -1 in dbscan_labels:
    # Убираем шумовые точки (-1) для оценки силуэта
    mask = dbscan_labels != -1
    dbscan_silhouette = silhouette_score(scaled_data[mask], dbscan_labels[mask])
    print("Silhouette Score для DBSCAN (без учета шума):", dbscan_silhouette)
else:
    print("DBSCAN не выделил кластеры или выделил только шум.")

from sklearn.decomposition import PCA

pca = PCA(n_components=2)
reduced_data = pca.fit_transform(scaled_data)

plt.figure(figsize=(12, 5))

# Визуализация KMeans
plt.subplot(1, 2, 1)
plt.scatter(reduced_data[:, 0], reduced_data[:, 1], c=kmeans_labels, cmap='viridis', s=10)
plt.title('KMeans Clustering')
plt.xlabel('PCA Component 1')
plt.ylabel('PCA Component 2')

# Визуализация DBSCAN
plt.subplot(1, 2, 2)
plt.scatter(reduced_data[:, 0], reduced_data[:, 1], c=dbscan_labels, cmap='viridis', s=10)
plt.title('DBSCAN Clustering')
plt.xlabel('PCA Component 1')
plt.ylabel('PCA Component 2')

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

```
Silhouette Score для KMeans: 0.14258348399367266
Silhouette Score для DBSCAN (без учета шума): 0.18162447996910616
```



In [ ]:

Вывод

1. **Оптимальное число кластеров для KMeans:** Метод локтя показал, что оптимальное количество кластеров находится в районе 4-5.

Оценка силуэта подтвердила, что наилучший результат достигается при 5 кластерах (Silhouette Score ≈ 0.12).

2. **Сравнение KMeans и DBSCAN:**
   A. KMeans лучше выделяет кластеры при относительно равномерном распределении точек, но может давать некорректные результаты для кластеров сложной формы.
   B. DBSCAN показал лучший Silhouette Score (0.1816 против 0.1426 у KMeans), но он также выделяет больше шумных точек