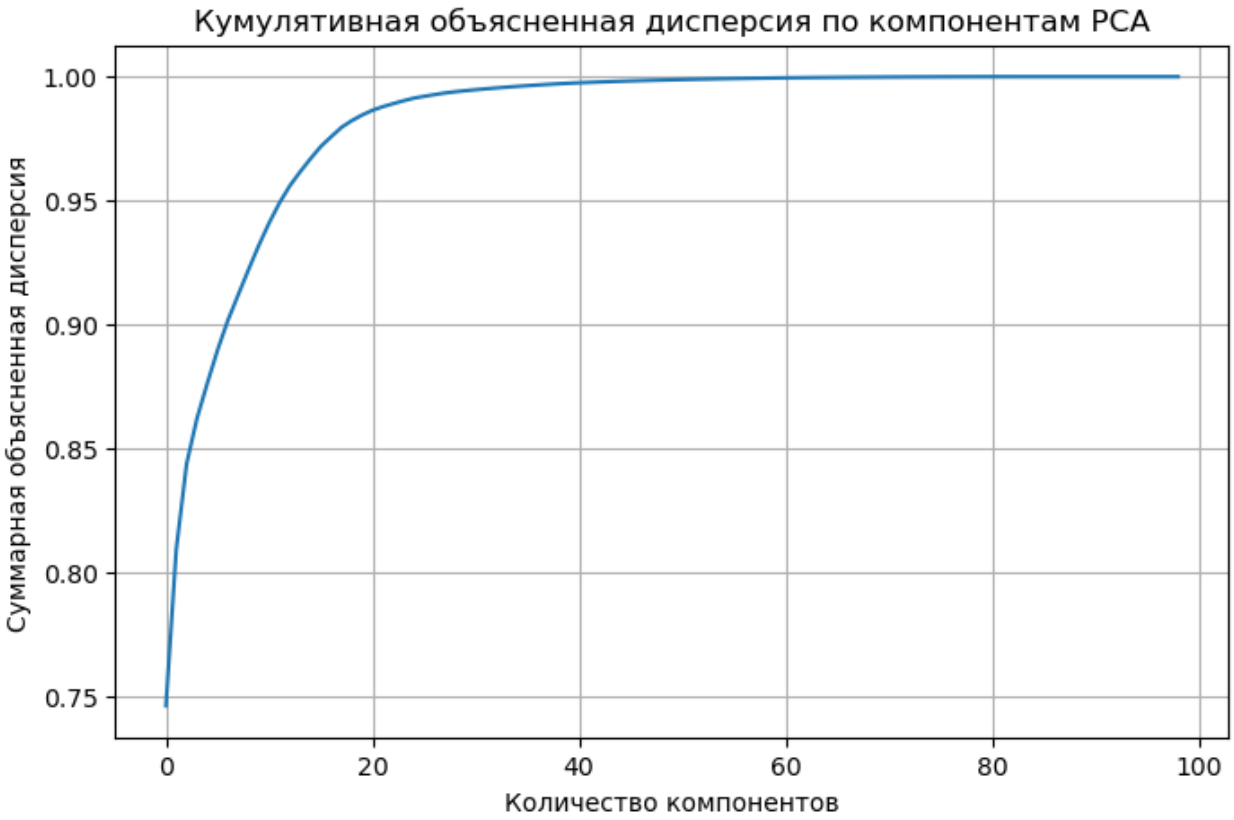
In [2]:

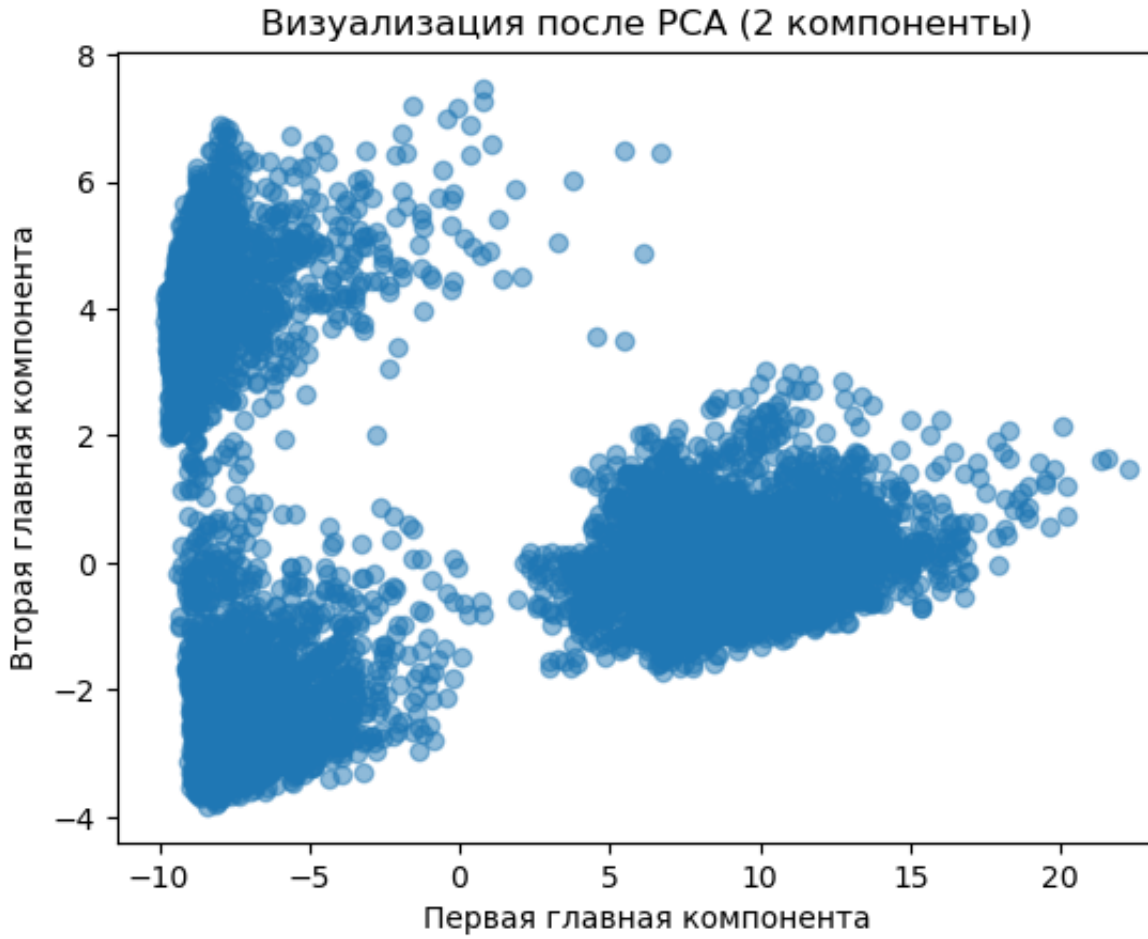
```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.feature_selection import VarianceThreshold

# Шаг 1: Загрузка данных
X = np.loadtxt('X_train.txt')

# Шаг 2: Осмотр данных
print("Количество признаков:", X.shape[1])
print("Пропуски в данных:", np.isnan(X).any())
print("Масштабы признаков различаются:", np.std(X, axis=0).var())

# Шаг 3: Удаление признаков с низкой вариативностью
sel = VarianceThreshold(threshold=(.8 * (1 - .8)))
X_sel = sel.fit_transform(X)
print("Признаков после удаления:", X_sel.shape[1])

# Шаг 4: Масштабирование признаков
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_sel)

# Шаг 5: Применение PCA
pca = PCA().fit(X_scaled)
plt.figure(figsize=(8, 5))
plt.plot(np.cumsum(pca.explained_variance_ratio_))
plt.xlabel('Количество компонентов')
plt.ylabel('Суммарная объясненная дисперсия')
plt.title('Кумулятивная объясненная дисперсия по компонентам PCA')
plt.grid(True)
plt.show()

# Определение минимального количества главных компонент для объяснения хо
cumulative_variance = np.cumsum(pca.explained_variance_ratio_)
n_components = np.where(cumulative_variance >= 0.9)[0][0] + 1
print(f"Необходимое количество компонент для 90% дисперсии: {n_components

# Применение PCA с двумя компонентами для визуализации
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)
plt.scatter(X_pca[:, 0], X_pca[:, 1], alpha=0.5)
plt.xlabel('Первая главная компонента')
plt.ylabel('Вторая главная компонента')
plt.title('Визуализация после PCA (2 компоненты)')
plt.show()

# Шаг 6: Кластеризация с помощью метода KMeans
inertias = []
for k in range(1, 11):
    # Указываем значение n_init явно, чтобы избежать предупреждений
    kmeans = KMeans(n_clusters=k, random_state=1, n_init=10).fit(X_pca)
    inertias.append(kmeans.inertia_)
```

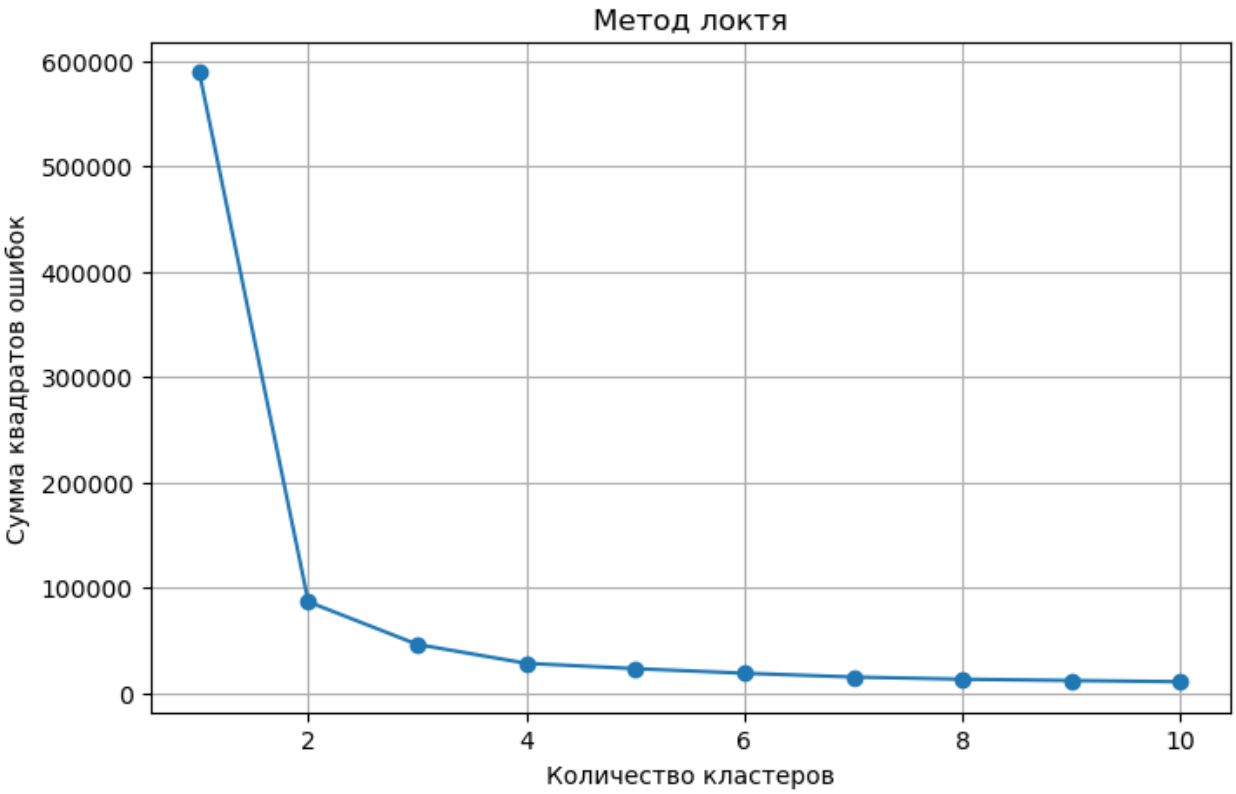
```
plt.figure(figsize=(8, 5))
plt.plot(range(1, 11), inertias, marker='o')
plt.xlabel('Количество кластеров')
plt.ylabel('Сумма квадратов ошибок')
plt.title('Метод локтя')
plt.grid(True)
plt.show()

# Выбор оптимального количества кластеров и визуализация
optimal_k = 5  # предположим, что оптимальное количество кластеров равно
kmeans = KMeans(n_clusters=optimal_k, random_state=1, n_init=10)
clusters = kmeans.fit_predict(X_pca)
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=clusters, cmap='viridis', alpha=0
plt.xlabel('Первая главная компонента')
plt.ylabel('Вторая главная компонента')
plt.title('Кластеризация после PCA')
plt.show()

# Шаг 7: Загрузка и сравнение с реальными метками
y = np.loadtxt('y_train.txt').astype(int)
activity_labels = np.loadtxt('activity_labels.txt', dtype=str)

# Сравнение кластеров с реальными метками (примерный подход)
from sklearn.metrics import confusion_matrix
import seaborn as sns

cm = confusion_matrix(y, clusters + 1)  # предположим, что кластеры начин
plt.figure(figsize=(10, 7))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Предсказанные кластеры')
plt.ylabel('Истинные метки')
plt.title('Матрица путаницы')
plt.show()
```

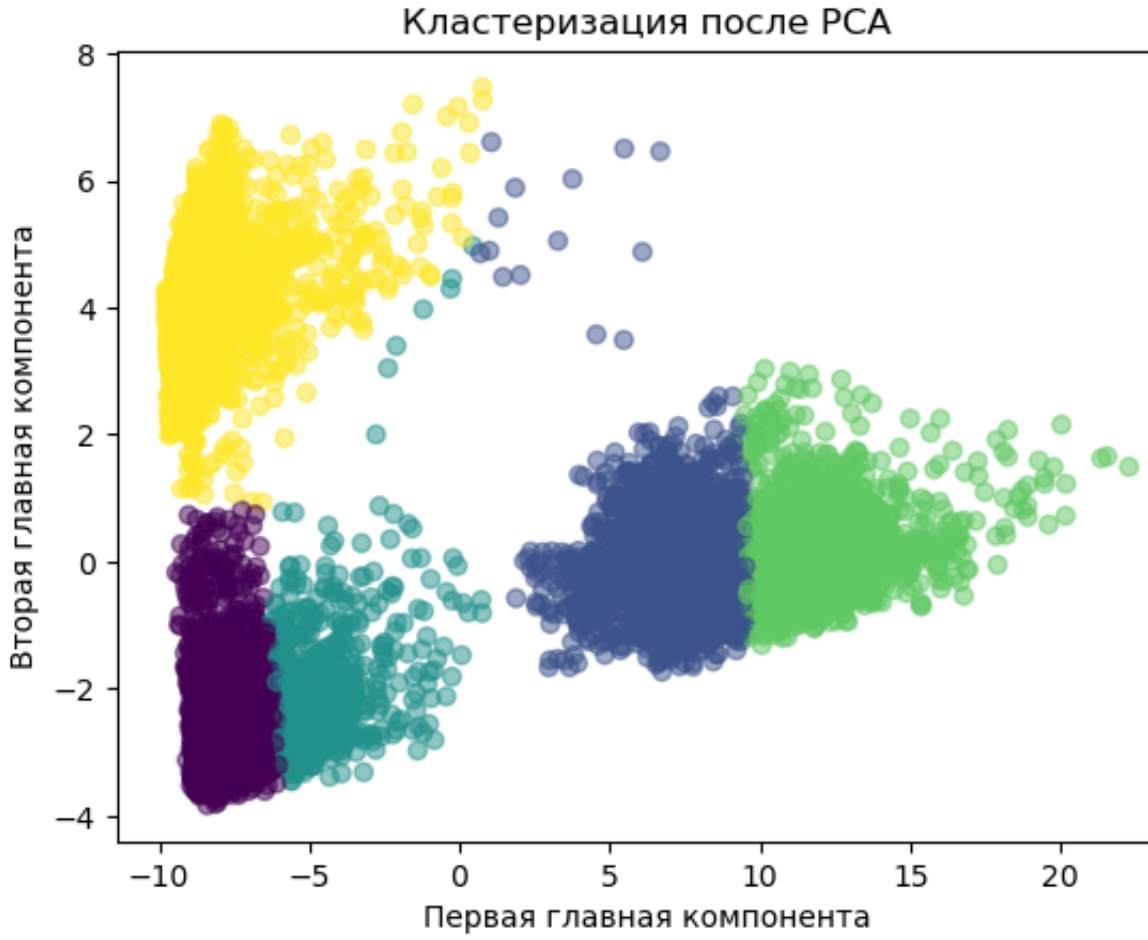
```
Количество признаков: 561
Пропуски в данных: False
Масштабы признаков различаются: 0.018747821292012665
Признаков после удаления: 99
```

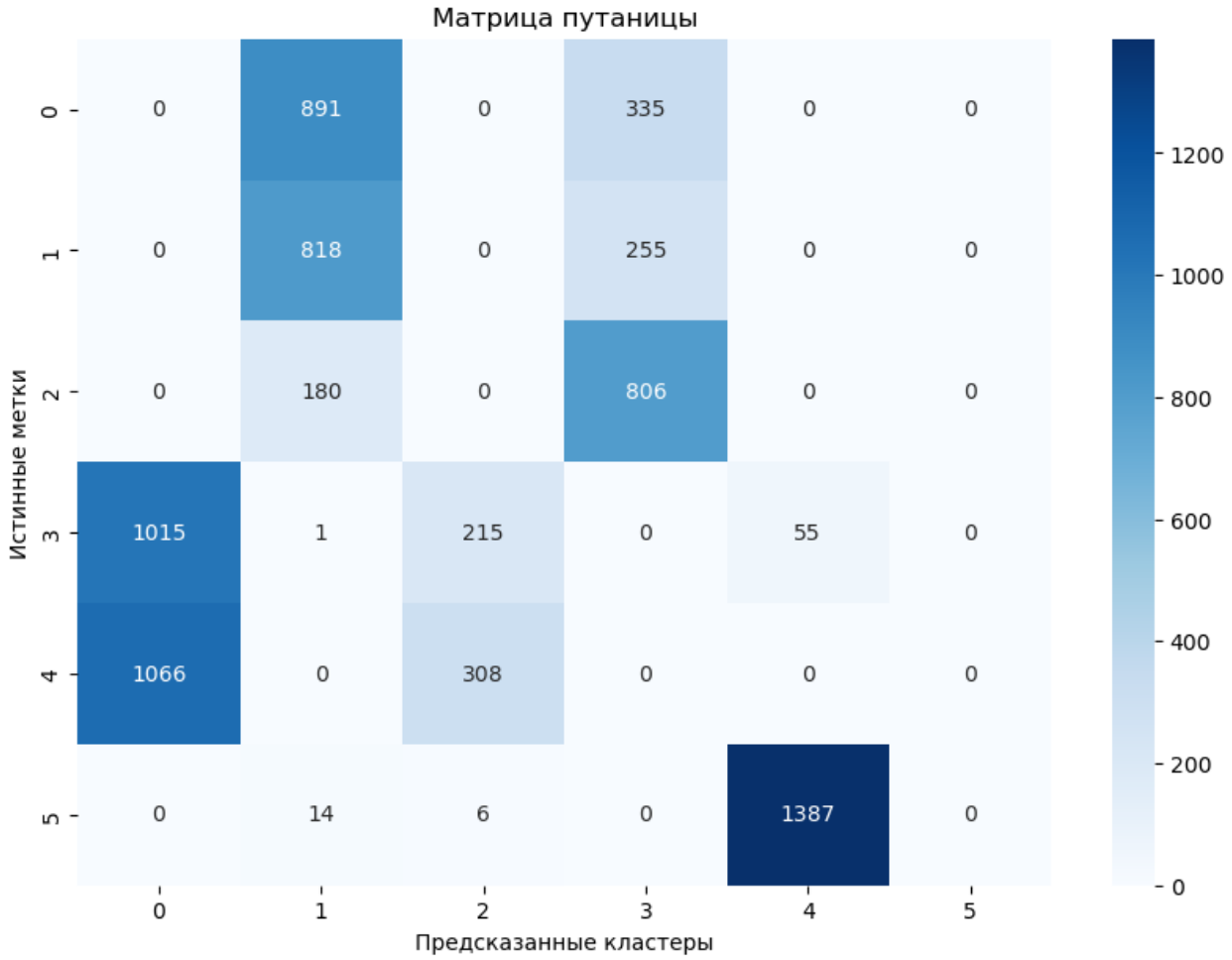


Необходимое количество компонент для 90% дисперсии: 7




Метод локтя
Сумма квадратов ошибок
Количество кластеров
600000
500000
400000
300000
200000
100000
0
2
4
6
8
10


Кластеризация после PCA
Вторая главная компонента
Первая главная компонента
8
6
4
2
0
−2
−4
−10
−5
0
5
10
15
20


Матрица путаницы
Истинные метки
Предсказанные кластеры
0 891 0 335 0 0
0 818 0 255 0 0
0 180 0 806 0 0
1015 1 215 0 55 0
1066 0 308 0 0 0
0 14 6 0 1387 0
0 1 2 3 4 5
0 200 400 600 800 1000 1200

In [ ]: