

Министерство образования и науки Российской Федерации  
Федеральное государственное автономное образовательное учреждение  
высшего образования  
«Самарский национальный исследовательский университет имени академика С.П. Королева»

Лабораторная работа №2

Метод главных компонент  
(Principal Component Analysis)

*Методические указания  
к лабораторным работам по курсу «Большие данные»*

Самара 2020

Составитель В.В. Жидченко

УДК 681.3.016

**Метод главных компонент (Principal Component Analysis):**

Метод. указания к лабораторным занятиям / Самар. ун-т; Сост. В.В. Жидченко.  
Самара, 2020. 11 с.

В методических указаниях рассматриваются основы метода главных компонент, широко применяющегося при анализе “больших данных” для снижения размерности задачи. Используются возможности библиотек языка Python: numpy, matplotlib, sklearn, pandas. Методические указания подготовлены на кафедре программных систем.

# Работа с языком программирования Python в web-среде. Система Google Colab

В данной работе используются возможности системы Google Colab для создания и запуска программ на языке Python. Указанная система предоставляет вычислительные ресурсы и интерактивный web-интерфейс, работающий по технологии Jupyter Notebook. Система предоставляет виртуальные машины с предустановленными популярными библиотеками Python, поэтому она позволяет познакомиться с различными методами обработки данных без установки программного обеспечения на личный компьютер. Для работы с системой необходимо авторизоваться с помощью аккаунта Google.

## Задание 1. Знакомство с системой Colab и реализацией PCA в Python

1) В окне браузера откройте систему Google Colab по ссылке:

<https://colab.research.google.com/>

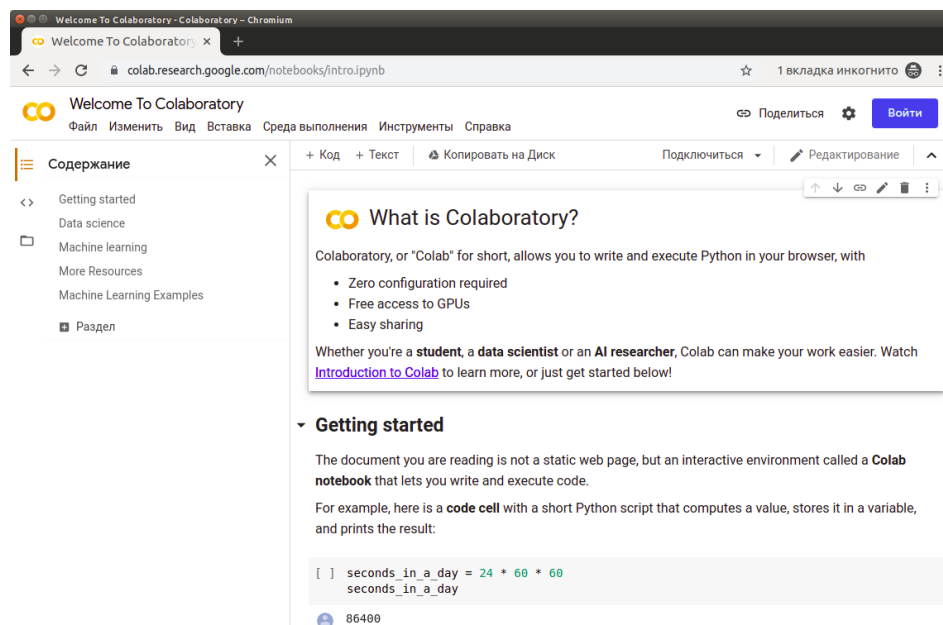


Рис. 1 Главное окно системы Google Colab

2) Выберите меню «Файл» → «Создать блокнот» и авторизуйтесь с аккаунтом Google. Откроется интерфейс редактирования нового блокнота (Notebook). Исходный код на Python вводится в поля ввода, называемые ячейками. Слева от ячейки находится кнопка запуска введенного в ячейку кода на исполнение. Можно создать несколько ячеек и запускать их в произвольном порядке. Переменные, созданные в одной ячейке, видны в других ячейках,

запускаемых после нее. Это позволяет вводить код постепенно, запуская каждый фрагмент и анализируя результат.

3) Введите следующий текст в ячейку:

```
import numpy as np #Подключение библиотеки (модуля) NumPy под локальным именем np
import matplotlib.pyplot as plt #Подключение модуля pyplot библиотеки matplotlib под локальным
именем plt
rng = np.random.RandomState(1020304) #Инициализация генератора псевдослучайных чисел
X = np.dot(rng.rand(2, 2), rng.randn(2, 200)).T #Генерация матрицы случайных чисел размером 200x2
print("X.shape: ", X.shape) #Печать размерности сгенерированной матрицы
print("X: ", X[0:5]) #Печать первых 5 строк матрицы
plt.scatter(X[:, 0], X[:, 1]) #Графический вывод точек, представленных матрицей
plt.axis('square'); #Равный масштаб графика по обеим осям
for i in range(X.shape[1]): #Вычисление и печать среднего значения
    print("X[:,", i, "].mean: ", X[:, i].mean()) #данных в столбцах матрицы
```

4) Запустите введенный код на исполнение, нажав кнопку запуска слева от ячейки или нажав комбинацию клавиш "Ctrl"- "Enter".

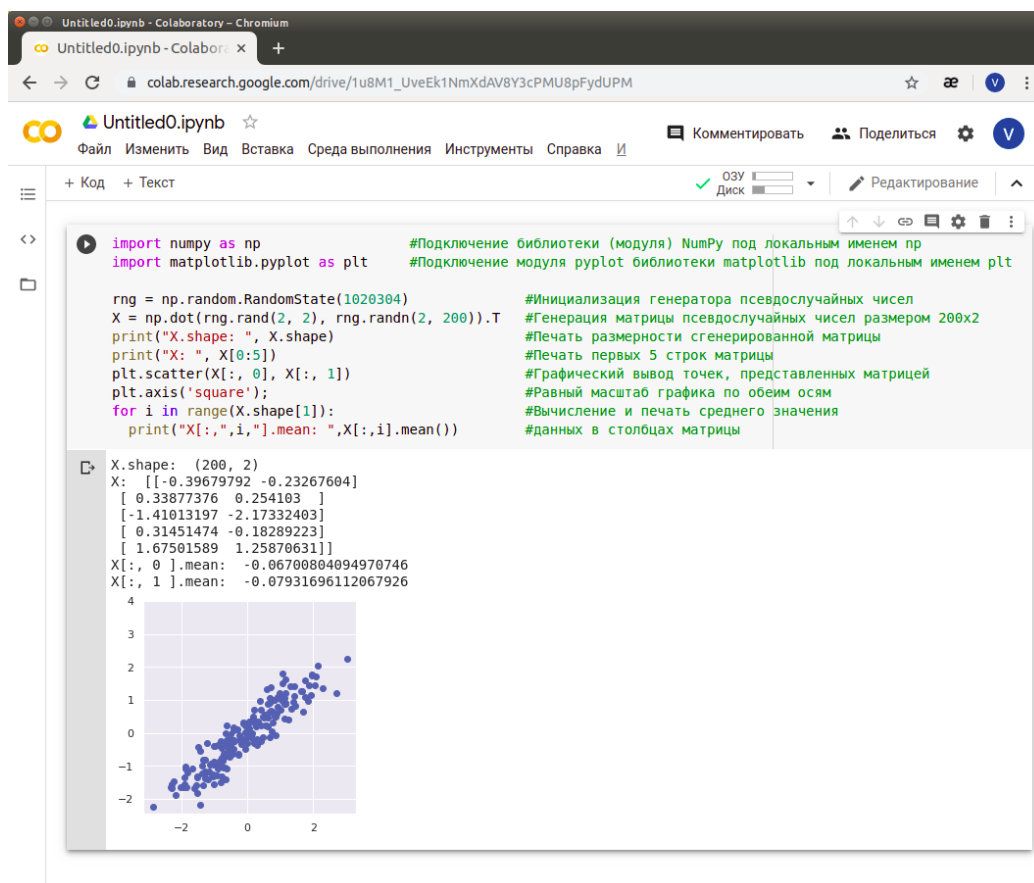


Рис. 2 Запуск ячейки на исполнение

5) Создайте новую ячейку, нажав «+ Код» сверху окна. Введите следующий код в новую ячейку:

```
mean_vec = [ X[:,0].mean(), X[:,1].mean() ] #Создание вектора среднего столбцов матрицы X
cov_mat = (X - mean_vec).T.dot((X - mean_vec)) / (X.shape[0]-1) #Вычисление ковариационной матрицы
```

```

print('Covariance matrix: \n%s' %cov_mat) #Печать ковариационной матрицы
print('NumPy covariance matrix: \n%s' %np.cov(X.T)) #Вычисление и печать ковариационной матрицы в
numpy

%time eig_vals, eig_vecs = np.linalg.eig(cov_mat) #Вычисление собственных векторов и значений в
numpy
print('Eigenvectors.shape: ', eig_vecs.shape) #Печать размерности матрицы собственных векторов
print('Eigenvectors: ', eig_vecs) #Печать матрицы собственных векторов
print('\nEigenvalues.shape: ', eig_vals.shape) #Печать размерности матрицы собственных значений
print('Eigenvalues: ', eig_vals) #Печать матрицы собственных значений

```

Запустите код ячейки на исполнение. Убедитесь, что ковариационная матрица, вычисленная по явной формуле, совпадает с матрицей, вычисленной с помощью функции библиотеки NumPy.

Команда `%time` является внутренней командой Jupyter Notebook (Built-in magic command), которая позволяет измерить длительность выполнения строки кода.

```

mean_vec = [ X[:,0].mean(), X[:,1].mean() ]
cov_mat = (X - mean_vec).T.dot((X - mean_vec)) / (X.shape[0]-1)
print('Covariance matrix: \n%s' %cov_mat)
print('NumPy covariance matrix: \n%s' %np.cov(X.T))

#Perform eigendecomposition on covariance matrix
%time eig_vals, eig_vecs = np.linalg.eig(cov_mat)
print('Eigenvectors.shape: ', eig_vecs.shape)
print('Eigenvectors: ', eig_vecs)
print('\nEigenvalues.shape: ', eig_vals.shape)
print('Eigenvalues: ', eig_vals)

```

ОЗУ  
Диск

```

#Создание вектора среднего столбцов матрицы X
#Вычисление ковариационной матрицы
#Печать ковариационной матрицы
#Вычисление и печать ковариационной матрицы в numpy
#Вычисление собственных векторов и значений в numpy
#Печать размерности матрицы собственных векторов
#Печать матрицы собственных векторов
#Печать размерности матрицы собственных значений
#Печать матрицы собственных значений

```

```

Covariance matrix:
[[1.2401979  0.98966384]
 [0.98966384 0.92263781]]
NumPy covariance matrix:
[[1.2401979  0.98966384]
 [0.98966384 0.92263781]]
CPU times: user 550 µs, sys: 966 µs, total: 1.52 ms
Wall time: 25.5 ms
Eigenvectors.shape: (2, 2)
Eigenvectors: [[ 0.76105601 -0.64868617]
 [ 0.64868617  0.76105601]]

Eigenvalues.shape: (2,)
Eigenvalues: [2.08373797 0.07909774]

```

Рис. 3 Пример вывода результатов работы кода в ячейке

## 6) Создайте новую ячейку и введите следующий код:

```

from sklearn.decomposition import PCA #Подключение модуля из библиотеки sklearn под лок. именем PCA

pca = PCA(n_components=2) #Создание объекта для выполнения Principal Component Analysis
pca.fit(X) #Выполнение PCA - вычисление двух главных компонент
print("components: \n",pca.components_) #Печать вычисленных компонент

```

```

print("explained_variance: \n",pca.explained_variance_) #Печать дисперсии, "объясненной" каждой
компонентой
print("explained_variance_ratio_: \n",pca.explained_variance_ratio_) #Печать доли "объясненной"
дисперсии каждой компоненты
print("pca.mean_: ", pca.mean_) #Печать вектора средних значений по столбцам матрицы данных

def draw_vector(v0, v1, textstr, vectorcolor, ax=None): #Пользовательская функция для рисования
векторов
    ax = ax or plt.gca()
    arrowprops=dict(arrowstyle='-', linewidth=2, shrinkA=0, shrinkB=0, color=vectorcolor)
    ax.annotate(textstr, v1, v0, arrowprops=arrowprops)

i=1 #Инициализация переменной i (цвет отображаемого вектора)
plt.figure(figsize=(6, 6), dpi=80) #Создание объекта рисунка с графиком
plt.scatter(X[:, 0], X[:, 1], alpha=0.5) #Графическое отображение матрицы исходных данных
for length, vector in zip(pca.explained_variance_, pca.components_): #Цикл по главным компонентам
    print("length: ", length) #Печать длины вектора главной компоненты
    print("vector: ", vector) #Печать вектора главной компоненты
    v = vector * 2.7 * np.sqrt(length) #Вычисление длины вектора, пропорциональной объясненной
    дисперсии
    draw_vector(pca.mean_ - v, pca.mean_ + v, i, (0.5*i,0,0)) #Рисование вектора главной
    компоненты
    i+=1 #Изменение цвета очередного вектора
plt.axis("square"); #Равный масштаб графика по обеим осям
#plt.axes().set(xlim=(-4, 4), ylim=(-4, 4)) #Определение границ по осям графика (при необходимости)

```

Запустите код ячейки на исполнение. Убедитесь, что векторы главных компонент, вычисленные с помощью метода класса `sklearn.decomposition.PCA` совпадают с точностью до знака с векторами главных компонент, вычисленными ранее с помощью разложения на собственные векторы. Убедитесь с помощью графика, что векторы главных компонент определяют направления наибольшего разброса данных. Проанализируйте долю объясненной дисперсии (PVE) для каждой главной компоненты.

б) Замените строку 4 в первой ячейке на следующую:

```

X = np.dot([[0.7,-0.7],[0.7,0.7]], rng.randn(2, 200)).T #Генерация матрицы псевдослучайных чисел размером 200x2

```

Вместо произвольного преобразования координат каждой точки, как в первом случае, данная строка выполняет поворот. В результате изменяется распределение координат точек данных. Повторите вычисления в остальных ячейках. Как изменилась длина векторов, соответствующих главным компонентам? Объясните получаемые собственные значения и долю объясненной дисперсии для главных компонент.

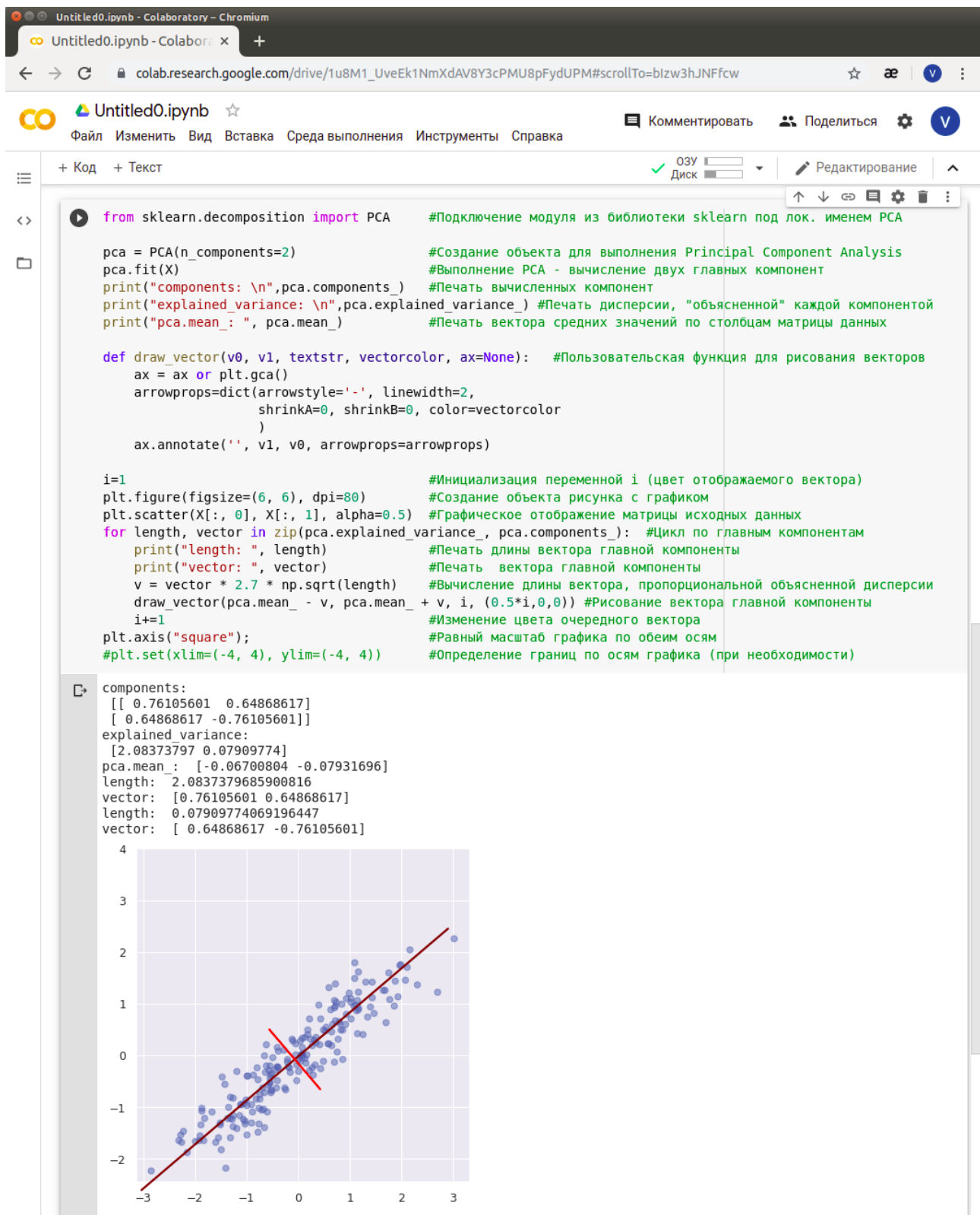


Рис. 4 Вычисление и отображение главных компонент

## Задание 2. Анализ набора статистических данных с помощью метода главных компонент

1) Загрузите тестовый набор статистических данных «USArrests» по ссылке:

<https://forge.scilab.org/index.php/p/rdataset/source/tree/master/csv/datasets/USArrests.csv>

Этот набор данных содержит количество арестов в различных штатах США за некоторый год в расчете на 100000 жителей штата. Данные сгруппированы по видам преступлений. Всего представлено 3 вида преступлений и 50 штатов. Для каждого штата указана также доля населения, проживающего в городских населенных пунктах (в процентах). Таким образом, количество выборок (samples) в этом наборе равно 50, каждая выборка характеризуется четырьмя признаками (features). Таблица данных содержит 50 строк и 4 столбца. Каждую выборку (единицу данных) можно представить точкой в четырехмерном пространстве признаков. Координатами точки по каждой из четырех осей будет значение соответствующего признака.

Сохраните файл USArrests.csv на Google Drive под тем же аккаунтом, под которым Вы работаете в Colab.

2) Создайте новый блокнот в Colab. Загрузите данные из файла USArrests.csv с помощью следующего кода:

```
#Load dependencies
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from matplotlib import*
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA as sklearnPCA
from google.colab import drive

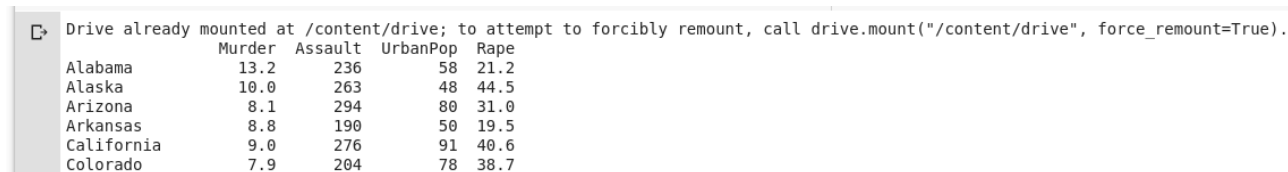
drive.mount('/content/drive')
Dset = pd.read_csv('/content/drive/My Drive/USArrests.csv', index_col=0)
print(Dset)
```

Система Colab предложит перейти по ссылке в Ваш Google аккаунт и подтвердить доступ к файлу. Скопируйте сгенерированный Authorization code и



вставьте его в поле «Enter your authorization code:» После этого откроется доступ к файлу из Colab.

В результате выполнения кода Вы увидите содержимое файла:



Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force\_remount=True).

	Murder	Assault	UrbanPop	Rape
Alabama	13.2	236	58	21.2
Alaska	10.0	263	48	44.5
Arizona	8.1	294	80	31.0
Arkansas	8.8	190	50	19.5
California	9.0	276	91	40.6
Colorado	7.9	204	78	38.7

Рис. 5 Вывод содержимого файла с данными

3) Создайте новую ячейку и введите в нее следующий код:

```
for (columnName, columnData) in Dset.iteritems():
    print("\n",columnName,"\n" column mean and standard deviation: ", columnData.mean(),
columnData.std())

X_std = StandardScaler().fit_transform(Dset)

print("\nStandardized:", X_std.shape)
for i in range(X_std.shape[1]):
    print("\n",Dset.columns[i],"\n" column mean and std. dev.: ", X_std[:,i].mean(), X_std[:,i].std())

mean_vec = np.mean(X_std, axis=0)
print("mean_vec: ", mean_vec)

cov_mat = np.cov(X_std.T)
print('cov_mat.shape: ', cov_mat.shape)
print('cov_mat: \n', cov_mat)
#Perform eigendecomposition on covariance matrix
eig_vals, eig_vecs = np.linalg.eig(cov_mat)
print('Eigenvectors ', eig_vecs.shape, ":")
print(eig_vecs.T)
print('Eigenvalues ', eig_vals.shape, ":")
print(eig_vals)

#Explained variance
pca = sklearnPCA().fit(X_std)
print("sklearn PCA components: ", pca.components_)
print("sklearn PCA explained_variance: ", pca.explained_variance_)
plt.plot(np.cumsum(pca.explained_variance_ratio_))
plt.xlabel('number of components')
plt.ylabel('cumulative explained variance')
plt.show()
```

Убедитесь, что метод `StandardScaler().fit_transform(Dset)` библиотеки `sklearn` приводит набор данных к нормализованной форме с нулевым средним значением по столбцам данных и стандартным отклонением 1.

Убедитесь, что главные компоненты, вычисленные методом `fit()` класса PCA библиотеки `sklearn` совпадают с собственными векторами ковариационной матрицы, а объясненная дисперсия главных компонент совпадает с собственными значениями матрицы.

Проанализируйте построенный график суммарной объясненной дисперсии главных компонент. Какую долю дисперсии объясняют суммарно первые две главные компоненты?

4) Создайте новую ячейку и введите в нее следующий код:

```
fig = plt.figure(figsize=(10, 10), dpi=80)
ax = plt.axes()
X_pca = pca.transform(X_std)
ax.scatter(X_pca[:,0], X_pca[:,1], alpha=0.4)
f1_pca = pca.transform([[1,0,0,0],[0,1,0,0],[0,0,1,0],[0,0,0,1]])
ax.quiver([pca.mean_[0]], [pca.mean_[1]], [f1_pca[0,0]], [f1_pca[0,1]], scale=3)
ax.annotate('Murder', xy=(0, 0), xytext=(f1_pca[0,0]*np.sqrt(3), f1_pca[0,1]*np.sqrt(3)),
           fontsize=16)
ax.quiver([pca.mean_[0]], [pca.mean_[1]], [f1_pca[1,0]], [f1_pca[1,1]], scale=3)
ax.annotate('Assault', xy=(0, 0), xytext=(f1_pca[1,0]*np.sqrt(3), f1_pca[1,1]*np.sqrt(3)),
           fontsize=16)
ax.quiver([pca.mean_[0]], [pca.mean_[1]], [f1_pca[2,0]], [f1_pca[2,1]], scale=3)
ax.annotate('UrbanPop', xy=(0, 0), xytext=(f1_pca[2,0]*np.sqrt(3), f1_pca[2,1]*np.sqrt(3)),
           fontsize=16)
ax.quiver([pca.mean_[0]], [pca.mean_[1]], [f1_pca[3,0]], [f1_pca[3,1]], scale=3)
ax.annotate('Rape', xy=(0, 0), xytext=(f1_pca[3,0]*np.sqrt(3), f1_pca[3,1]*np.sqrt(3)), fontsize=16)
for i in range(X_pca.shape[0]):
    ax.annotate(Dset.index[i], xy=(0,0), xytext=(X_pca[i,0], X_pca[i,1]))
```

Запустите ячейку на выполнение. Данный код строит проекцию точек данных из четырехмерного пространства исходных признаков на плоскость, описываемую первыми двумя главными компонентами. Черные стрелки — это проекции исходных осей данных на ту же плоскость.

Каким исходным признакам придает больший вес первая главная компонента? Какому признаку придает больший вес вторая компонента? Что можно сказать о коррелированности соответствующих признаков? Если оси исходных признаков сонаправлены в пространстве главных компонент, это означает, что исходные признаки коррелированы. Что можно сказать о зависимости между количеством арестов по различным видам преступлений в

одном и том же штате? Что можно сказать о зависимости между количеством арестов и уровне урбанизации штата?

Что можно сказать о количестве арестов и уровне урбанизации в штатах, расположенных ближе к центру графика (они имеют высокий, низкий, средний уровень по соответствующим показателям)?

Составьте краткий отчет по результатам работы. В отчет необходимо включить Ваши скриншоты с исходным текстом и выводом программы, а также привести ответы на приведенные в задании вопросы.