Практическое задание №2

Задание 1

In [81]:
```
import numpy as np
```

In [33]:
```
data = np.array([
    [4, 2.9],
    [2.5, 1],
    [3.5, 4],
    [2, 2.1]
])
print(data)
```

```
[[4.  2.9]
 [2.5 1. ]
 [3.5 4. ]
 [2.  2.1]]
```

In [29]:
```
def sim_function(xi, xj):
    return pow(np.linalg.norm(xi - xj), 2)
```

In [31]:
```
kernel_matrix = np.zeros(shape=(len(data), len(data)))
for i in range(len(data)):
    for j in range(len(data)):
        kernel_matrix[i][j] = sim_function(data[i], data[j])
print(kernel_matrix)
```

```
[[ 0.    5.86  1.46  4.64]
 [ 5.86  0.   10.    1.46]
 [ 1.46 10.    0.    5.86]
 [ 4.64  1.46  5.86  0.  ]]
```

Задание 2

In [58]:
```
data = np.array([
    [8, 0, 10, 10, 2],
    [-20, -1, -19, -20, 0]
])
print(data)
```

```
[[  8   0  10  10   2]
 [-20  -1 -19 -20   0]]
```

Рассчитайте среднее μ и ковариационную матрицу Σ для матрицы D

In [114]:
```
print('mean:')
print(np.mean(data, axis=1))
print('covariance matrix:')
print(np.cov(data))
```

```
mean:
[  6. -12.]
covariance matrix:
[[ 22.  -47.5]
 [-47.5 110.5]]
```

Рассчитайте собственные числа для матрицы Σ

In [73]:
```
w, v = np.linalg.eigh(np.cov(data))
print('eigenvalues:')
print(w)
```

```
eigenvalues:
[  1.33226359 131.16773641]
```

Какой “внутренний” размер данного набора данных?

In [74]:
```
print('"Inner" size:')
print(data.shape)
```

```
"Inner" size:
(2, 5)
```

Рассчитайте первый главный компонент

In [129]:
```
centered_data = data.T - np.mean(data, axis=1)
print(centered_data)
```

```
[[ 2. -8.]
 [-6. 11.]
 [ 4. -7.]
 [ 4. -8.]
 [-4. 12.]]
```

In [130]:
```
w, v = np.linalg.eigh(np.cov(centered_data.T))
print(w)
print(v)
```

```
[  1.33226359 131.16773641]
[[-0.91696017 -0.39897876]
 [-0.39897876  0.91696017]]
```

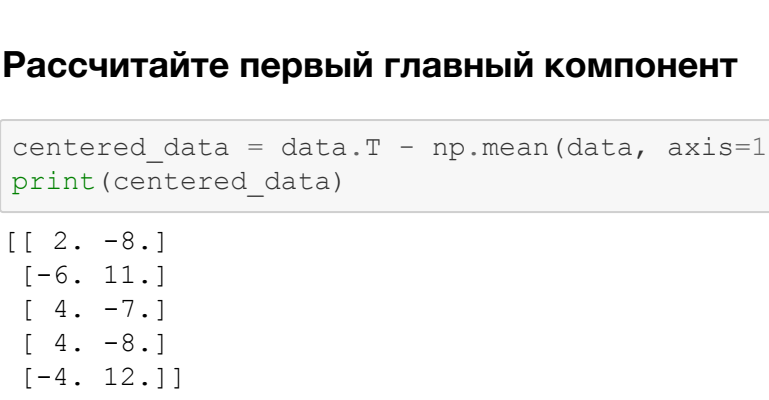
In [132]:
```
PC1 = np.dot(centered_data, -v[:, np.argmax(w)])
print(PC1)
```

```
[  8.13363886 -12.4804344    8.01463621   8.93159638 -12.59943705]
```

Если μ и Σ сверху характеризуют нормальное распределение, из которого были сгенерированы точки, нарисуйте ориентацию / протяженность 2-мерной функции нормальной плотности.

In [157]:
```
X, Y = np.random.multivariate_normal(np.mean(data, axis=1), np.cov(data), 5000).T
plt.plot(X, Y, '.')
```

Out[157]:
```
[<matplotlib.lines.Line2D at 0x11f720c10>]
```



Задание 3

In [172]:
```
from sklearn.decomposition import KernelPCA

data = np.array([
    [4, 2.9],
    [2.5, 1],
    [3.5, 4],
    [2, 2.1]
])
transform = kernel_matrix * (1/2) + np.ones((data.shape[0], data.shape[0])) * 2
transformed_matrix = transform @ kernel_matrix @ transform
PCA1 = KernelPCA(1, kernel="precomputed").fit_transform(transformed_matrix)
print(PCA1)
```

```
[[-2.56357095]
 [ 2.56357095]
 [ 2.56357095]
 [-2.56357095]]
```

In [ ]: