Курс - Библиотеки Python для Data Science: Numpy, Matplotlib, Scikit-learn ¶

Урок 2. Практическое задание 1 (2)

Видеоурок. Вычисления с помощью Numpy. Работа с данными в Pandas.

Тема "Вычисления с помощью Numpy"

Задание 1

Импортируйте библиотеку Numpy и дайте ей псевдоним np.

В [145]:
```
import numpy as np
```

Создайте массив Numpy под названием a размером '5x2', то есть состоящий из 5 строк и 2 столбцов.

Первый столбец должен содержать числа '1, 2, 3, 3, 1,' а второй - числа '6, 8, 11, 10, 7'.

Будем считать, что каждый столбец - это признак, а строка - наблюдение.

В [146]:
```
# Создаём массив a размерности 5x2
a = np.array([[1, 6],
              [2, 8],
              [3, 11],
              [3, 10],
              [1, 7]])

print("\nРазмерность a: {}".format(a.ndim))
print("Форма a: {}".format(a.shape))
print("Количество элементов a: {}".format(a.size))
a
```

```
Размерность a: 2
Форма a: (5, 2)
Количество элементов a: 10
```

Out[146]:
```
array([[ 1,  6],
       [ 2,  8],
       [ 3, 11],
       [ 3, 10],
       [ 1,  7]])
```

Затем найдите среднее значение по каждому признаку, используя метод mean массива Numpy.

Результат запишите в массив mean_a, в нем должно быть 2 элемента.

В [116]:
```
# Находим среднее значение по каждому признаку
mean_a = a.mean(axis=0)
print("\nСреднее значение по признакам (массив mean_a): {}".format(mean_a))
#mean_a
```

```
Среднее значение по признакам (массив mean_a): [2.  8.4]
```

Задание 2

Вычислите массив a_centered, отняв от значений массива “a” средние значения соответствующих признаков, содержащиеся в массиве mean_a.

Вычисление должно производиться в одно действие.

Получившийся массив должен иметь размер 5x2.

В [41]:
```
# Вычисляем массив a_centered
a_centered = a - mean_a # используем Broadcasting
a_centered
```

Out[41]:
```
array([[-1. , -2.4],
       [ 0. , -0.4],
       [ 1. ,  2.6],
       [ 1. ,  1.6],
       [-1. , -1.4]])
```

Задание 3

Найдите скалярное произведение столбцов массива a_centered.

В результате должна получиться величина a_centered_sp.

Затем поделите a_centered_sp на N-1, где N - число наблюдений.

B [142]:
```
# a_centered[0, :] или a_centered[0] # 1-я строка

col_1 = a_centered[:,0] # 1-й столбец массива a_centered
col_2 = a_centered[:,1] # 2-й столбец массива a_centered

# Находим скалярное произведение столбцов массива a_centered.
a_centered_sp = np.dot(col_1, col_2)
print("\nСкалярное произведение столбцов массива:\na_centered_sp = {}".format(a_c

# Делим a_centered_sp на N-1, где N - число наблюдений.
print("\nЧисло наблюдений: {}".format(a.shape[0]))

# Находим ковариацию двух признаков, содержащихся в массиве a
cv_a = a_centered_sp/(a.shape[0] - 1)
print("\nКовариация двух признаков, содержащихся в массиве a:\ncv_a = {}".format(
```

```
Скалярное произведение столбцов массива:
a_centered_sp = 8.0

Число наблюдений: 5

Ковариация двух признаков, содержащихся в массиве a:
cv_a = 2.0
```

Задание 4**

Число, которое мы получили в конце задания 3 является ковариацией двух признаков, содержащихся в массиве “a”.

В задании 3 мы делили сумму произведений центрированных признаков на N-1, а не на N, поэтому полученная нами величина является несмещенной оценкой ковариации.

В этом задании проверьте получившееся число, вычислив ковариацию еще одним способом - с помощью функции np.cov.

В качестве аргумента m функция np.cov должна принимать транспонированный массив “a”.

В получившейся ковариационной матрице (массив Numpy размером 2x2) искомое значение ковариации будет равно элементу в строке с индексом 0 и столбце с индексом 1.

B [143]:
```
print("Проверка:")

# Прямая матрица
print("\nПрямая матрица: \n{}".format(a))

# Транспонированная матрица
at = a.transpose()
print("\nТранспонированная матрица: \n{}".format(at))

# Ковариационная матрица
cov = np.cov(at)
print("\nКовариационная матрица: \n{}".format(cov))

# Искомое значение ковариации
print("\nИскомое значение ковариации:\ncov[0][1] = {}".format(cov[0][1]))

print("\nПроверка: cv_a == cov[0][1]\n{}".format(cv_a == cov[0][1]))
```

```
Проверка:

Прямая матрица:
[[ 1  6]
 [ 2  8]
 [ 3 11]
 [ 3 10]
 [ 1  7]]

Транспонированная матрица:
[[ 1  2  3  3  1]
 [ 6  8 11 10  7]]

Ковариационная матрица:
[[1.  2. ]
 [2.  4.3]]

Искомое значение ковариации:
cov[0][1] = 2.0

Проверка: cv_a == cov[0][1]
True
```