**Задание 2. Ручное уменьшение размерности признаков**

В задании вам дан набор данных. Геометрическая интерпретация этого набора - точки в многомерном пространстве (размерности D). Базис этого пространства это вектора, где один из признаков 1, а остальные 0. Таких векторов D штук. А сами данные это проекции точек на оси координат, заданные данным базисом. В задании вам необходимо произвести линейное преобразование данного пространства, т.е. перейти к новой системе координат, отвечающей определенным требованиям, что эквивалентно умножению массива данных на матрицу W. Матрицу W получим в результате использования метода fit() объекта sklearn.decomposition.PCA(). После чего вы получите набор данных той же размерности, что и исходный (D), но в другой системе координат (используя метод transform). Таким образом, никаких циклов использовать не надо. Все что надо сделать:

1. Создать экземпляр класса sklearn.decomposition.PCA с параметрами по умолчанию.
2. Использовать метод fit полученного объекта с набором данных в качестве аргумента.
3. Использовать метод transform для получения отображения данных в новое пространство.

Теперь необходимо определить дисперсии распределения данных вдоль осей координат нового пространства. Проекции данных на оси координат нового пространства это и есть столбцы полученной в результате применения метода transform матрицы. Таким образом, осталось только посчитать эти дисперсии. Например, использовать функцию np.std, передав ей в качестве аргументов полученный массив данных и axis=0.

Суть задания заключается в том, что бы данные размерности D преобразовать в данные меньшей размерности d, при этом потеряв минимум информации. Для чего предлагается проанализировать дисперсии признаков по всем компонентам, после отображения данных в пространство с определенными свойствами (указаны в начале тетрадки).

Иными словами, сначала вам предлагается найти векторы (D штук) с помощью класса sklearn.decomposition.PCA, которые послужат базисом нового пространства. Для чего необходимо создать экземпляр данного класса с параметрами по умолчанию и использовать метод fit() с предлагаемыми данными в качестве аргумента. Затем отобразить данные в новое пространство такой же размерности (D) с помощью метода transform() и найти дисперсии проекций предложенных данных на вектора базиса данного пространства. Собственно проекции вы получили когда отобразили данные методом transform(). Эти проекции и есть координаты объектов в новом пространстве. Таким образом, осталось просто посчитать их, например, воспользовавшись методом std(axis=0). После чего, следуя описанному в задании алгоритму найти d, то есть количество компонент, дисперсия вдоль которых значительно выше, чем вдоль остальных (пик в разностях соответствует скачку в ряду упорядоченных дисперсий). Это и будет искомая размерность пространства.

Построить модель PCA с D главными компонентами по этим данным. (Понятно)

Спроецировать данные на главные компоненты. (найти собственные вектора???)

Оценить их дисперсию вдоль главных компонент. (найти собственные значения???)

Отсортировать дисперсии в порядке убывания и получить их попарные разности: λ(i−1)−λ(i). (отсортировать собственные значения по убыванию и найти попарные разности???)

{kolichestvo = np.arange(80)

for n in kolichestvo:

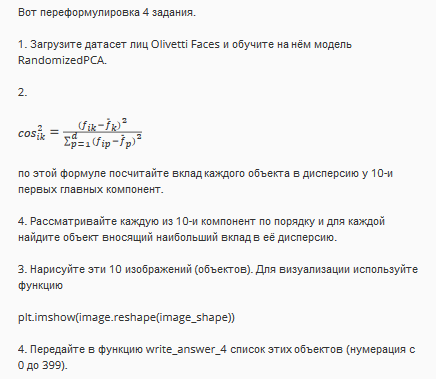
n = n + 1

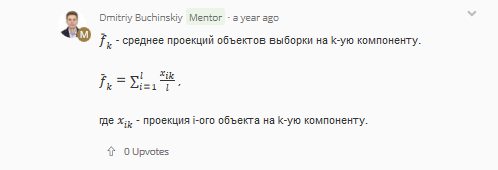
model = PCA(n\_components=n, svd\_solver='full').fit(data)

scores = cross\_val\_score(model, data, cv=3)

print(scores.mean())}

Задание 4:





model = RandomizedPCA()

model.fit(data)

components10 = model.transform(data)[:, :10]

components10 \*\* 2

cos\_list = np.empty(components10.shape)

for i in range(components10.shape[0]):

for j in range(components10.shape[1]):

cos\_list[i][j] = components10[i][j] / np.sum(components10[i])

# numbers[i][j] = components10[i][j] / (np.sum(components10[i]) - components10[i][j])

answers = [l.tolist().index(np.max(l)) for l in cos\_list.T]

Получив матрицу из 10 главных компонент (далее ГК), хотим посчитать какой вклад дает каждый элемент матрицы в дисперсию ГК. Для каждого элемента делаем одно и то же: выбираем для какой ГК будем считать расстояние (например 1-ой), выбираем вклад какого элемента будем считать (например тоже 1-ый), берем элемент из 1-ой строки и 1-го столбца матрицы, возводим в квадрат и делим на сумму квадратов элементов, стоящих в 1-ой строке. Повторяем такую операцию для всех элементов матрицы и выбираем максимальные.

в задании надо

1) посчитать квадрат косинуса угла между каждым из 400 изображений и каждой из 10 компонент (всего 4000 косинусов)

2) для каждой из 10 компонент найти изображение с максимальным по модулю косинусом

3) скормить порядковые номера этих изображений в датасете (10 целых чисел в отрезке от 0 до 399, через пробел, без переноса строки) грейдеру.

model = PCA(n\_components=x)

model.fit(data)

a = cv\_score(model, data)

Задание 3

Корреляцию в 3 задании считать с помощью numpy.corrcoef, а не np.correlate

**модуль центрированной проекции объекта на компоненту**: берем данные после преобразования.

берем от них модуль.

потом нормируем строчку, чтобы сумма по строке равнялась 1.

потом в каждой столбце находим индекс с наибольшим значением.

в ответ записываем 10 чисел. номера строк

Признаков 4. Компонент 2. Надо посчитать корреляцию каждого признака с каждой компонентой. Итого 8 корреляций. Дальше для каждого признака надо сравнить его корреляцию с первой компонентой с его же корреляцией со второй (по модулю) и добавить номер этого признака либо в список для первой компоненты, либо в список для второй. И вот таким образом 4 признака надо распределить в два списка.

То есть ваш ответ - это два списка, по которым вы распределили номера 1,2,3,4. Например list1 = [1,2,3,4], list2 =[], после чего вы делаете write\_answer\_3(list1,list2)

##### v, W\_true = np.linalg.eig(C)

##### mu = np.zeros(2)

##### C = np.array([[3,1],[1,2]])

##### data = np.random.multivariate\_normal(mu, C, size=50)

##### plt.scatter(data[:,0], data[:,1])

##### # построим истинные компоненты, вдоль которых максимальна дисперсия данных

##### plt.plot(data[:,0], (W\_true[0,0]/W\_true[0,1])\*data[:,0], color="g")

##### plt.plot(data[:,0], (W\_true[1,0]/W\_true[1,1])\*data[:,0], color="g")

Задание 1: Выполняя задание 1 недели 2, обнаружил, что cross\_val\_score выдает в ответе вектор отрицательных чисел. Это меня порядком насторожило, ведь насколько я понимаю ответом должен быть вектор неотрицательных вероятностей!

Решение задания(d\_opt) я нашел и оно было принято грейдером, но d\_opt я выбирал из наибольшего отрицательного means ответов cross\_val\_score.

Помогите разобраться почему cross\_val\_score выдает отрицательные ответы, валидно ли это?