## Понижение размерности

### Цель работы

Применить методы понижения размерности для решения задач машинного обучения.

### Задания для выполнения

1. Сформируйте случайную двумерную область точек
2. Используйте метод главных компонент для понижения размерности
3. Инвертируйте данные полученные после понижения размерности и выведите их на график.
4. Скачайте датасет с цифрами (load\_digits)
5. Используйте метод главных компонент для понижения размерности до 2.
6. Выведите на график полученные значения.
7. Используйте метод главных компонент с количеством компонентов – 64.
8. Постройте график объяснимости модели в зависимости от компонентов.
9. Добавьте шума в датасет. Выведите на экран обычные и зашумленные данные.
10. Найти минимальное количество компонентов необходимое для сохранения хотя бы на 50% информации о датасете.
11. Используйте метод главных компонент для понижения размерности с количеством компонент из пункта 10.
12. Инвертируйте данные полученные после понижения размерности и выведите их на экран.
13. Сделайте вывод о применимости модели

### Методические указания

Для начала работы нам потребуется импортировать необходимые библиотеки:

%matplotlib inline

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns; sns.set()

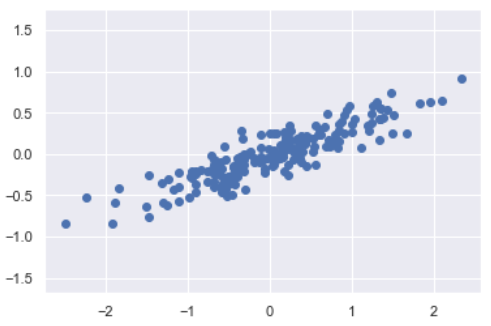
Сформируем случайную двумерную область точек и выведем её на экран:

rng = np.random.RandomState(1)

X = np.dot(rng.rand(2, 2), rng.randn(2,200)).T

plt.scatter(X[:, 0], X[:, 1])

plt.axis('equal');



Создадим и обучим модель метода главных компонент с размерностью 1.

pca = PCA(n\_components=1)

pca.fit(X)

X\_pca = pca.transform(X)

print("original shape: ", X.shape)

print("transformed shape:", X\_pca.shape)

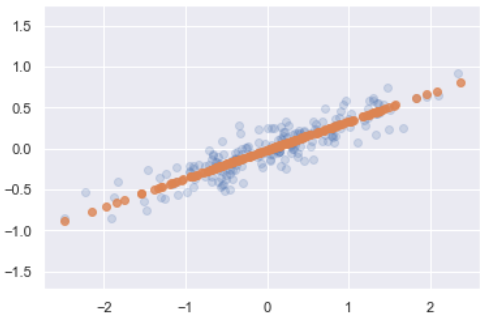
Инвертируем данные полученные после понижения размерности и выведем их на график.

X\_new = pca.inverse\_transform(X\_pca)

plt.scatter(X[:, 0], X[:, 1], alpha=0.2)

plt.scatter(X\_new[:, 0], X\_new[:, 1], alpha=0.8)

plt.axis('equal');



Загрузим датасет load\_digits.

from sklearn.datasets import load\_digits

digits = load\_digits()

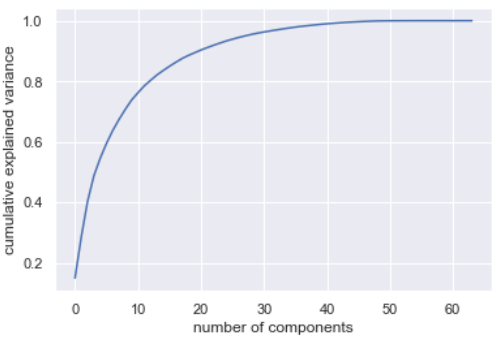
digits.data.shape

Постройте график объяснимости модели в зависимости от компонентов.

plt.plot(np.cumsum(pca.explained\_variance\_ratio\_))

plt.xlabel('number of components')

plt.ylabel('cumulative explained variance');



Вывести на экран данные:

def plot\_digits(data):

fig, axes = plt.subplots(4, 10, figsize=(10, 4),

subplot\_kw={'xticks':[], 'yticks':[]},

gridspec\_kw=dict(hspace=0.1, wspace=0.1))

for i, ax in enumerate(axes.flat):

ax.imshow(data[i].reshape(8, 8),

cmap='binary', interpolation='nearest',

clim=(0, 16))

plot\_digits(digits.data)

Добавить шума в датасет:

np.random.seed(42)

noisy = np.random.normal(digits.data, 4)

plot\_digits(noisy)