Лекция 2

Данные и их представление

Данные

- Данные - это любая информация полученная об объекте/ах исследования в ходе экспериментов, снятия показаний и т. д.
- Существуют разные виды информации:
  - Скалярное значение - значение по какой-то характеристике
  - Сигнал - упорядоченный набор значений характеристики, отображающие изменение
  - Изображения
  - Графы - отображает взаимосвязь различных элементов
  - Текстовая

Свойства данных

- Наличие априорных знаний о данных позволяет лучше решать поставленную задачу
- Знание о наличии ошибок/аномалиях в данных
- Размер выборки - чем больше данных, тем больше информации можно получить. Но данные должны быть репрезентативными.
- Репрезентативные данные описывают максимально возможных количество разнообразных состояний.
- Достоверность данных - насколько данные соответствуют действительности
- Наличие несущественных признаков

Набор данных

- Признак - какая-то характеристика объекта
- Наблюдение - набор значений признаков описывающих объект
- Набор данных - множество наблюдений

Признаки

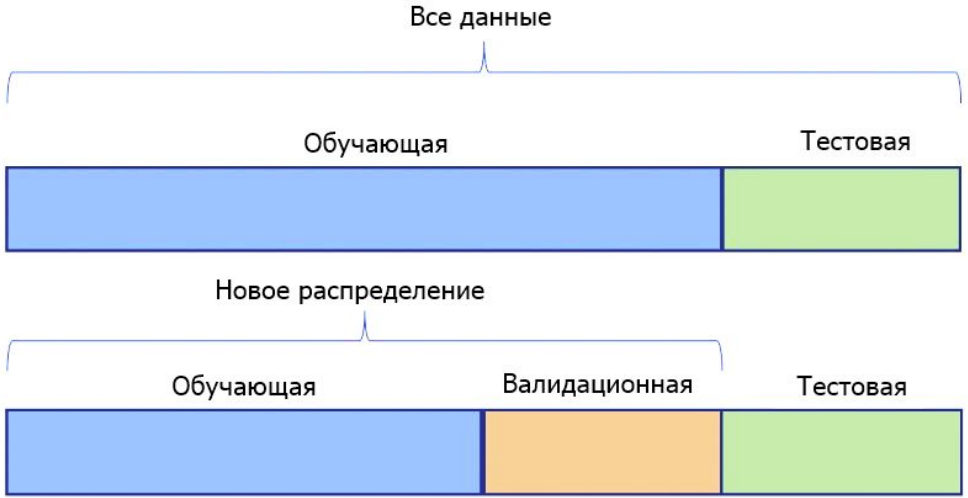
	Имя	Рост, см	Вес, кг
1	Андрей	166	60
2	Борис	177	70
3	Владимир	188	80

Наблюдения

- Набор данных можно рассматривать как множество точек в пространстве признаков, где размерность пространства равна кол-ву признаков

Разбиение на выборки

- Обучающая выборка - на ней метод настраивает свои параметры
- Валидационная выборка - на ней настраиваются гиперпараметры во время обучения
- Тестовая - на ней проверяется проверка финальной обученной модели



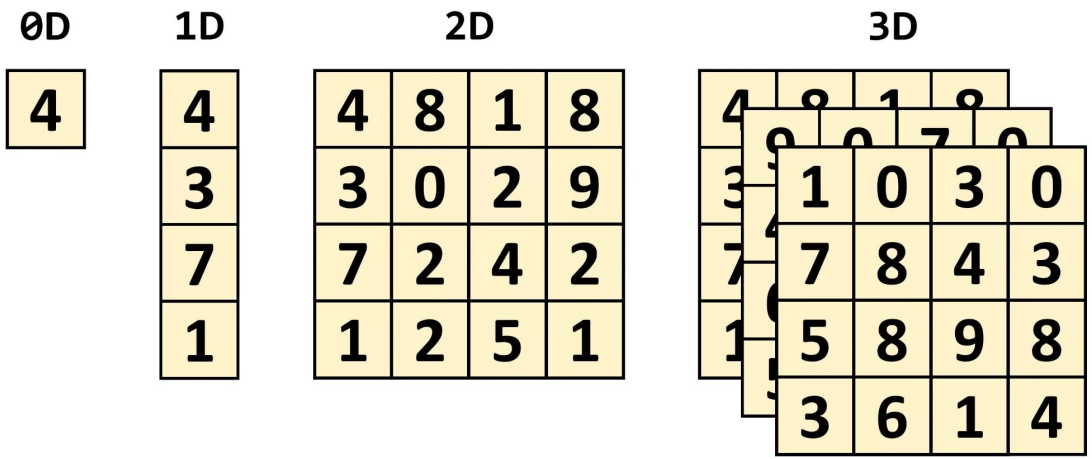
Признаковое описание объектов

- Каждый объект должен набором численных значений, чтобы вычислительное устройство могло его обработать.

- Можно выделить следующие типы признаков:
  - Бинарный - {0, 1} или {-1, 1}
  - Номинальный - конечное множество (обычно целых чисел)
  - Порядковый - конечное множество в котором задан порядок
  - Количественный - множество вещественных чисел

Представление данных

- Почти все современные системы машинного обучения используют тензоры в качестве основной структуры данных
- Одна из важных характеристик тензора - ранг:
  - 0 ранг - скаляр
  - 1 ранг - вектор
  - 2 ранг - матрица



Связь видов информации и тензоров

Ранг	Вид информации
0	Значение одного признака
1	Набор признаков / Одномерный сигнал
2	Одноканальное изображение / Матрица расстояний / Многомерный сигнал
3	Многоканальное изображения / Одноканальное видео
4	Многоканальное видео

Для набора данных, ранг увеличивается на 1 - по этой оси находятся наблюдения

NumPy

- Библиотека NumPy предоставляет удобную работу с тензорами, предоставляя большой спектр математических операций над тензорами
- Для создания тензора используется ф-ция `array`, в которую можно передать список представляющий структуру данных.
- Можно создать тензор без данных используя функции: `arange`, `zeros`, `ones`, `linspace, eye` и функции из подмодуля `random`
- Можно узнать следующие х-ки:
  - `ndim` - ранг тензора
  - `size` - количество элементов
  - `shape` - форма тензора
- Для изменения формы тензора и его ранга используется `reshape`
- Для загрузки тензора из файла используется функция `genfromtxt`

Доступ к элементам

- Для доступа к элементам используются срезы.
- В квадратных скобках указываются значение/диапазоны значений по каждому измерению.

data	
166	60
177	70
188	80

- В примере, первое измерение - наблюдения, второе - признаки
- data[:, 0] -> [166, 177, 188] - получить значений первого признака всех наблюдений
- data[0, :] -> [166, 60] - получить значений всех признаков первого наблюдения
- data[0, 0] -> 166 - получить значение первого признака первого наблюдения
- data[0:2, 1] -> [60, 70] - получить значение первого признака для 1 и 2 наблюдения

Математические операции

- Можно проводить следующие операции:
  - поэлементные операции (+ - * /)
  - тензорное произведение @ или dot
    - dot - более универсальная
    - @ - рекомендуется для матриц
  - x.T - транспонирование
  - применять мат. функции такие как:
    - exp
    - sin
    - ln

```
x = np.array([[1,1,1],[2,2,2],[3,3,3]])
y = np.array([1,2,3])
x + y
```

```
array([[2, 3, 4],
       [3, 4, 5],
       [4, 5, 6]])
```

```
x + y.T
```

```
array([[2, 3, 4],
       [3, 4, 5],
       [4, 5, 6]])
```

```
x + y.reshape(3,1)
```

```
array([[2, 2, 2],
       [4, 4, 4],
       [6, 6, 6]])
```

```
x + np.vstack([y, np.exp(y)])
```

Статистические операции

- Функции `mean`, `std`, `var` позволяют рассчитать среднее, стандартное отклонение и дисперсию.
- По умолчанию они считают учитывая все значения тензора сразу
- Чтобы задать направление расчета, используется параметр `axis`
- `var` по умолчанию считает смещенную оценку, чтобы считать несмещенную, то необходимо установить параметр `ddof = 1`
- Функция `cov` рассчитывает несмещенную ковариационную матрицу. Чтобы рассчитать смещенную ковариационную матрицу, то необходимо установить параметр `bias = True` или `ddof = 0`

Pandas

- Предоставляет модуль для работы с наборами данных.
- Основной структурой является DataFrame (таблица) и Series (ряд)
- Особенностью также является то, что можно хранить данные разного типа данных
- Для создания таблицы, необходимо передать в конструкторе данные. Причем, можно передать словарь, где каждый элемент соответствует своему признаку, и ключ словаря будет названием признака.
- Используя параметры indexes и columns можно задать названия для наблюдений и признаков. *При этом обращение по индексам отключается*
- Для чтения из файла используются методы по типу `read_csv`, `read_excel`, `read_json` и т.д.

Обращение к элементам

- Если отсутствуют имена, то при указании индекса будет возвращен ряд значений по соответствующему признаку. Если есть имена признаков, то вместо индекса необходимо указывать название признака
- Для получения столбца, необходимо использовать `loc` со списком индексов:

```
data.loc[[0, 3, 5]]
```

- Для обращения также можно использовать срезы
- Также можно использовать условия для доступа к элементам
- Методы `head` и `tail` позволяют вывести указанное количество первых или последних строк таблицы. По умолчанию выводится 5 строк.
- Метод `describe` позволяет вывести статистические характеристики по каждому признаку.

Модификация DataFrame

- Если в наборе данных отсутствуют значения, то можно использовать `dropna` для удаления строк, либо `fillna` для заполнения пропущенных значений согласно правилу.
- Метод `drop` позволяет отбросить указанные столбцы.
- Если есть строки дубликаты, то можно получить список дублирующих строк методов `duplicated` или выбросить их методом `drop_duplicates`. Причем, строка считается дублирующей, даже если название индексов разные.
- Для добавления нового признака или наблюдения, достаточно сделать запись по новому индексу

Операции с DataFrame

- Метод `apply` позволяет рассчитать функцию на каждом значении. Задав значение параметру `axis` можно применять функцию к каждому столбцу или строке.
- Метод `filter` позволяет получить подвыборку из таблицы по требуемым индексам. Можно задать следующие параметры:
  - items - название индексов
  - regex - регулярное выражение для индексов
  - like - подстрока, которая содержится в индексе
  - axis - ось по которой рассматриваться индексы (строки/столбцы)

Визуализация данных

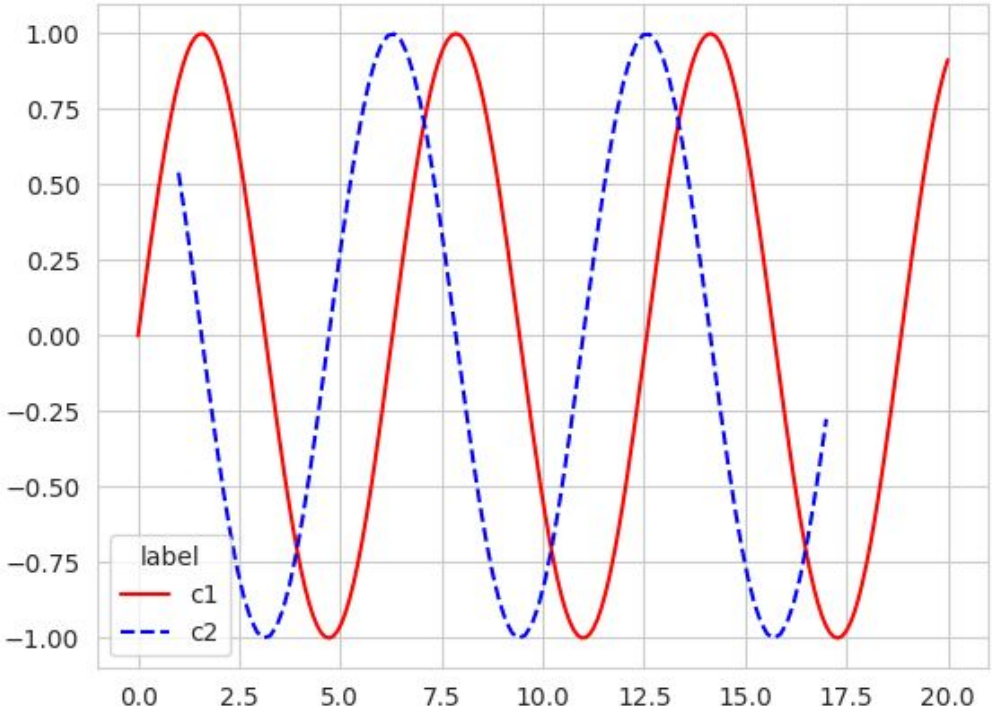
Визуализация данных

- Предварительная визуализация данных позволяет оценить большой объем данных и выбрать подходящие методы для анализа и построения модели.
- Визуализация результатов модели позволяет в компактном и наглядном виде представить и оценить качество модели.
- Основные модули для Python:
  - Matplotlib - удобен для построения простых графиков
  - Seaborn - надстройка над Matplotlib для визуализации наборов данных
  - ggplot2 - предоставляет гибкую настройку для построения сложных графиков

Введение в seaborn: https://seaborn.pydata.org/tutorial/introduction.html
Готовые решения в seaborn: https://seaborn.pydata.org/examples/index.html
Для прочтения: А. Богачев - Графики, которые убеждают всех

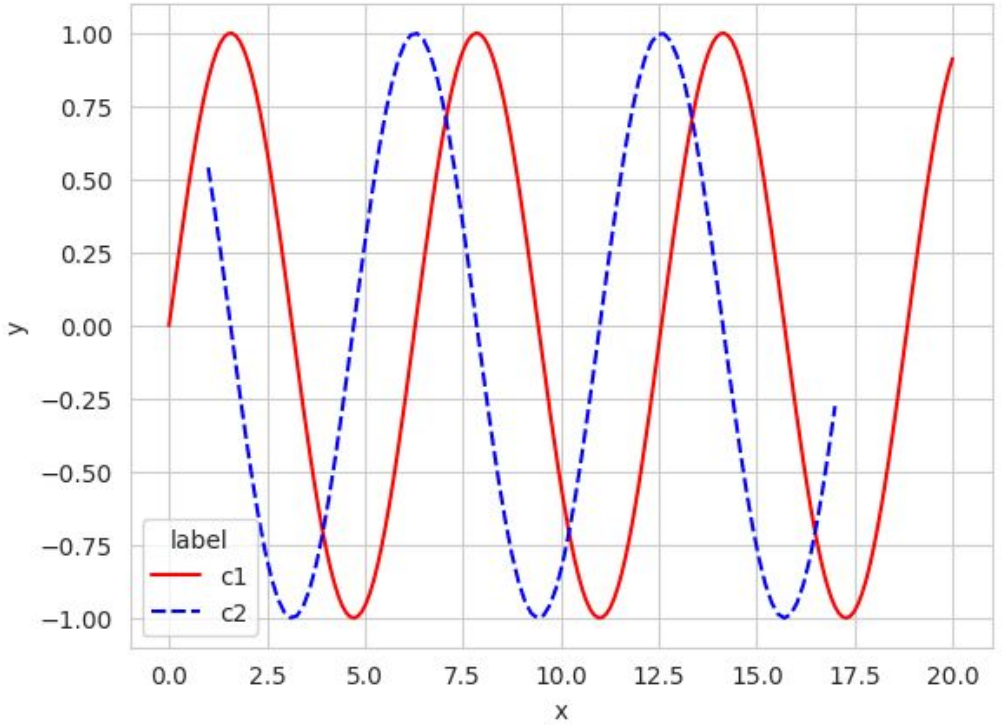
Построение функций с Matplotlib

```
x1 = np.linspace(0,20,200)
y1 = np.sin(x1)
x2 = np.linspace(1,17,100)
y2 = np.cos(x2)

plt.plot(x1,y1,'r-')
plt.plot(x2,y2,'b--')
plt.grid(True)
plt.legend(['c1', 'c2'],
           title = 'label')
plt.show()
```



Построение функций с Seaborn

```
labels = ['c1'] * len(x1)
       + ['c2'] * len(x2)
xy_data = pd.DataFrame(
     {'label': labels,
      'x': np.hstack([x1,x2]),
      'y': np.hstack([y1,y2])})

sns.set_style("whitegrid")
sns.lineplot(xy_data, x = 'x',
             y = 'y',
             style = 'label',
             hue = 'label',
             palette = ['r', 'b'])
```

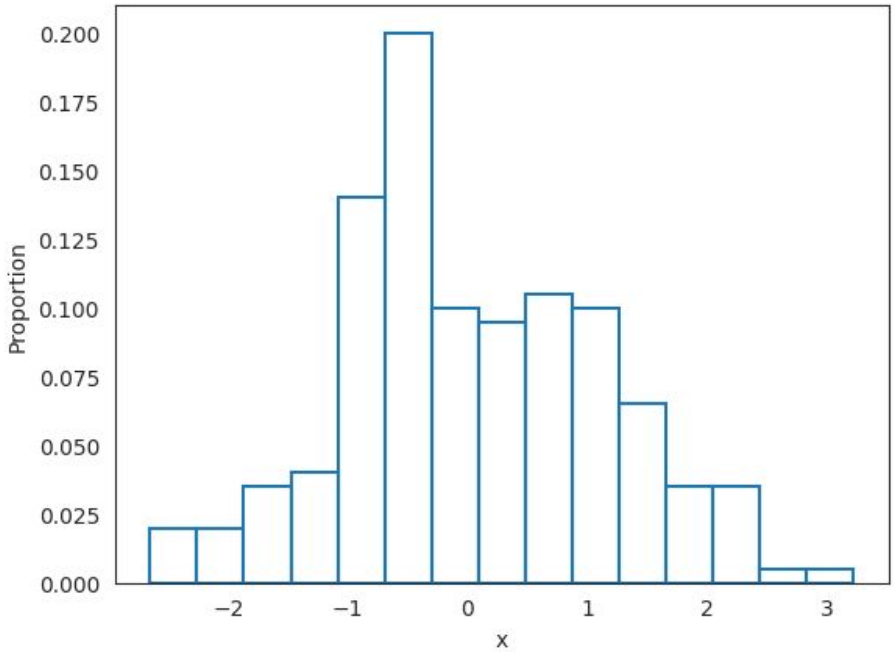


Гистограммы (1)

Гистограмма, столбчатая диаграмма, отображающая количество значений в интервале.

Позволяет оценить распределение и наличие выбросов.

```
sns.histplot(viz_data,
             x = 'x',
            bins = 15,
            fill = False,
            stat = 'proportion')
```

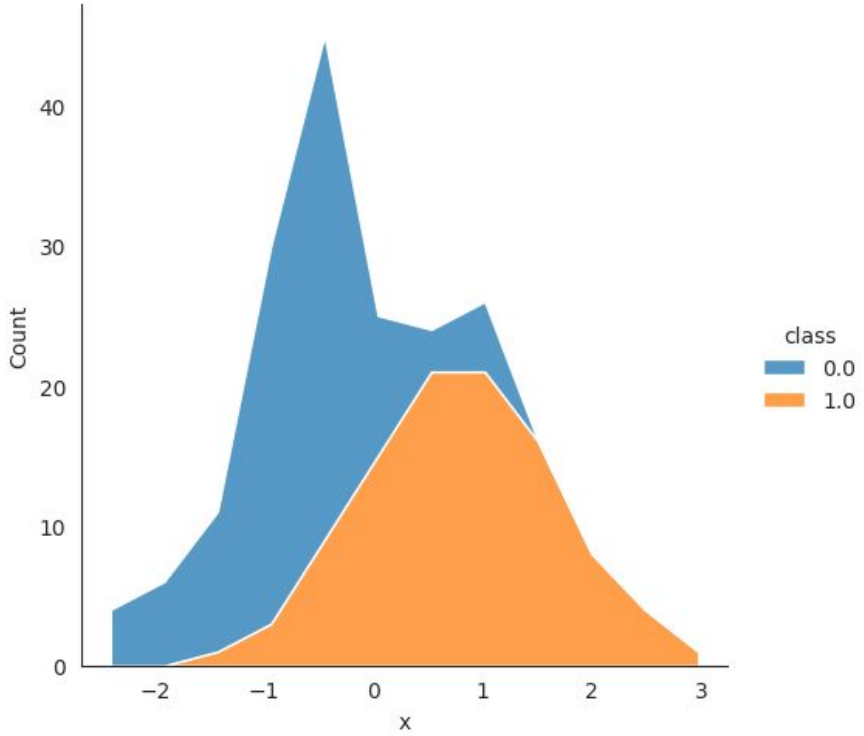


Гистограммы (2)

Разделение по какому-то признаку с накоплением позволяет оценить пропорции на каждом интервале.

displot - позволяет выбирать из разных способов визуализации распределения.

```
sns.displot(viz_data,
            x = 'x',
            kind = 'hist',
            element = 'poly',
            hue = 'class',
            multiple = 'stack')
```

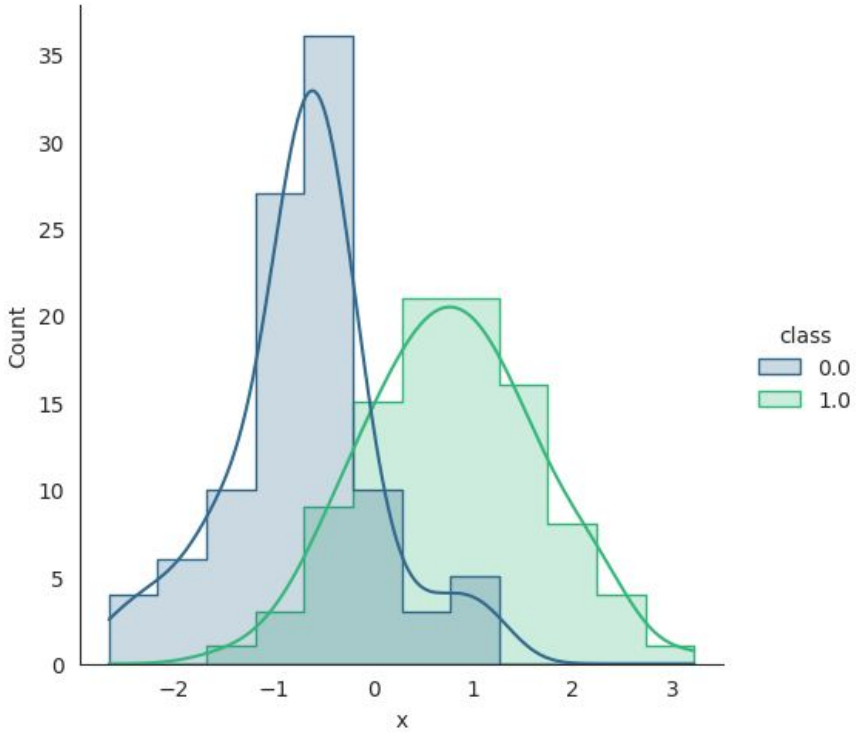


Гистограммы (3)

Разделение по признаку с пересечением позволяет оценить распределение признака по классу. Можно делать оценку о разделимости.

График ядерной оценки плотности (kde) исправляет недостаток гистограмм. Можно построить отдельно (kdeplot).

```
sns.displot(viz_data, x = 'x',
            kind = 'hist',
            element = 'step',
            hue = 'class',
            kde = True,
            palette = 'viridis')
```



Ядерная оценка плотности

В отличие от гистограммы для расчета значений используется операции свертки.

```
sns.set_style('whitegrid')
sns.displot(viz_data,
            x = 'x', y = 'y',
             kind = 'kde',
             col = 'class')
```

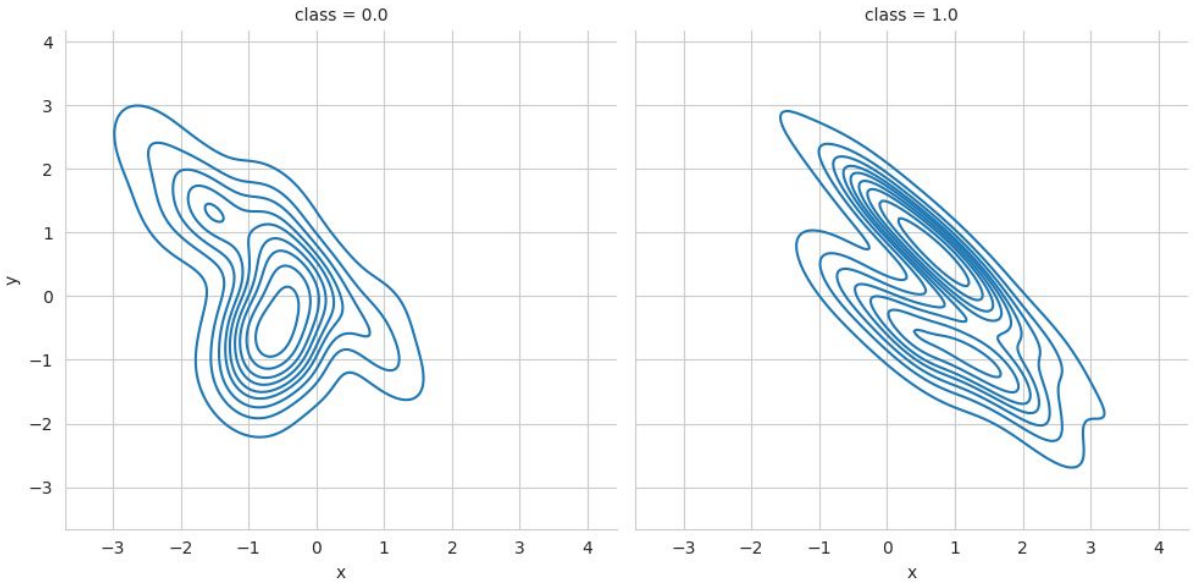
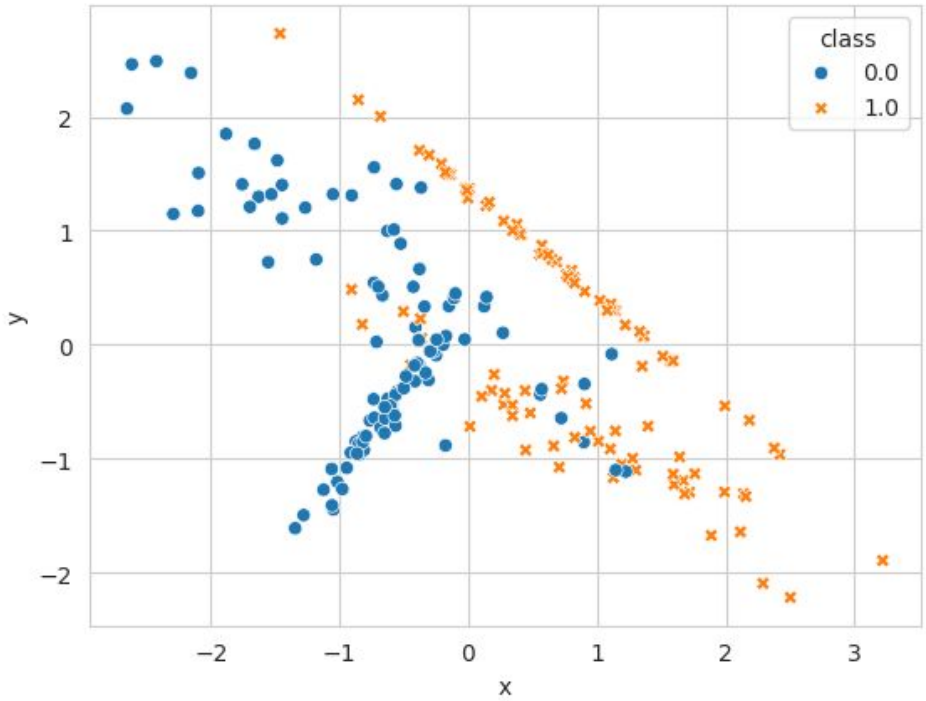


График рассеяния

Позволяет построить каждую точку в пространстве признаков.
Позволяет оценивать плотность, а также наличие выбросов и аномалий

Также можно использовать relplot

```
sns.scatterplot(viz_data,
                x = 'x',
                y = 'y',
                style = 'class',
                hue = 'class')
```



Оценка по категориям

Позволяет оценивать характеристики количественных признаков относительно категориальных.

Например, boxplot (ящик с усами) показывает квартили, среднее, отклонение, и выбросы

```
sns.catplot(viz_data, x = 'x',
            hue = 'class',
            kind = 'box')
```