Домашнее задание №2

Визуальный анализ данных о сердечно-сосудистых заболеваниях

В задании предлагается с помощью визуального анализа ответить на несколько вопросов по данным о сердечно-сосудистых заболеваниях. Данные использовались в соревновании ML Boot Camp 5 (в репозитории в папке data есть текстовик с ссылкой, выгрузите в эту же папку весь архив, чтобы код работал).

Заполните код в клетках (где написано "Ваш код здесь") и ответьте на вопросы в веб-форме. Код отправлять никуда не нужно.

В соревновании предлагалось определить наличие/отсутствие сердечно-сосудистых заболеваний (ССЗ) по результатам осмотра пациента.

Описание данных.

Датасет сформирован из реальных клинических анализов, и в нём используются признаки, которые можно разбить на 3 группы:

Объективные признаки:

- Возраст (age)
- Рост (height)
- Вес (weight)
- Пол (gender)

Результаты измерения:

- Артериальное давление верхнее и нижнее (ap_hi, ap_lo)
- Холестерин (cholesterol)
- Глюкоза (gluc)

Субъективные признаки (со слов пациентов):

- Курение (smoke)
- Употребление алкоголя (alco)
- Физическая активность (active)

Целевой признак (который интересно будет прогнозировать):

- Наличие сердечно-сосудистых заболеваний по результатам классического врачебного осмотра (cardio)

Возраст дан в днях. Значения показателей холестерина и глюкозы представлены одним из трех классов: норма, выше нормы, значительно выше нормы. Значения субъективных признаков — бинарны.

Все показатели даны на момент осмотра.

```
# подгружаем все нужные пакеты
import pandas as pd
import numpy as np
# игнорируем warnings
import warnings
warnings.filterwarnings("ignore")
import seaborn as sns
import matplotlib
import matplotlib.pyplot as plt
import matplotlib.ticker
%matplotlib inline
# настройка внешнего вида графиков в seaborn
sns.set_context(
 "notebook",
 font_scale = 1.5,
 rc = {
 "figure.figsize" : (12, 9),
 "axes.titlesize" : 18
 }
)
```

В рамках задания для простоты будем работать только с обучающей выборкой. Чистить данные от выбросов и ошибок в данных НЕ нужно, кроме тех случаев, где об этом явно указано. Все визуализации рекомендуем производить с помощью библиотеки `Seaborn`.

Проведем небольшой EDA

```
train = pd.read_csv(r"C:\Users\Елена\Downloads\data\
mlbootcamp5_train.csv", sep=';',
                    index_col='id')

print('Размер датасета: ', train.shape)
train.head()
```

```
Размер датасета:  (70000, 12)
```

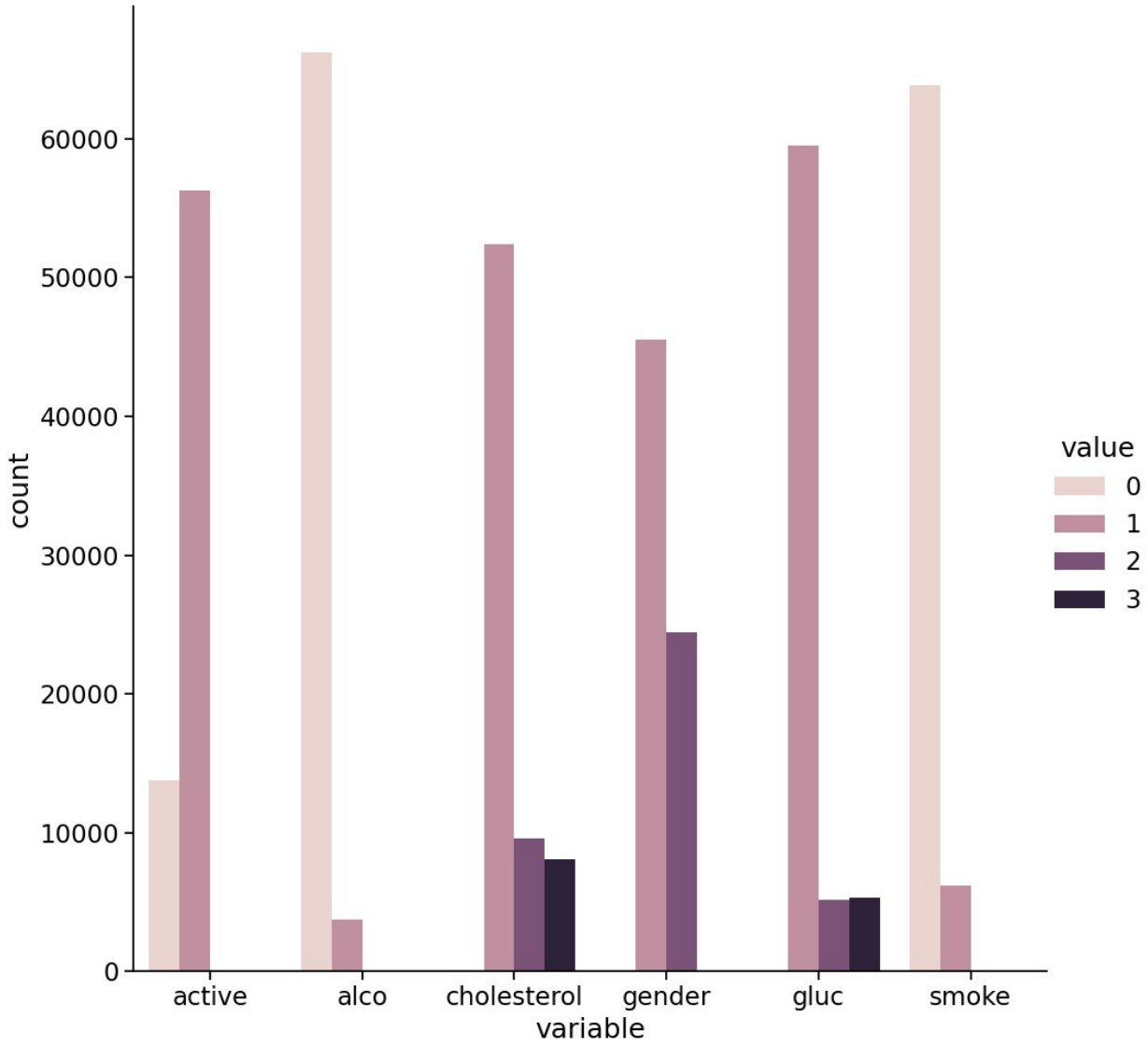
id	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke
0	18393	2	168	62.0	110	80	1	1	0
1	20228	1	156	85.0	140	90	3	1	0
2	18857	1	165	64.0	130	70	3	1	

```
0
3   17623       2     169    82.0    150    100            1     1
0
4   17474       1     156    56.0    100     60            1     1
0

    alco  active  cardio
id
0      0       1       0
1      0       1       1
2      0       0       1
3      0       1       1
4      0       0       0
```

Для начала всегда неплохо бы посмотреть на значения, которые принимают переменные.

Переведем данные в "Long Format"-представление и отрисуем с помощью factorplot количество значений, которые принимают категориальные переменные.

```
train_uniques = pd.melt(frame=train, value_vars=['gender',
'cholesterol',
                                                 'gluc', 'smoke',
'alco',
                                                 'active'],
                        id_vars=['cardio'])
train_uniques = pd.DataFrame(train_uniques.groupby(['variable',
'value'])
                             ['value'].count()) \
    .sort_index(level=[0, 1]) \
    .rename(columns={'value': 'count'}) \
    .reset_index()

sns.factorplot(x='variable', y='count', hue='value',
            data=train_uniques, kind='bar', height=10);
```

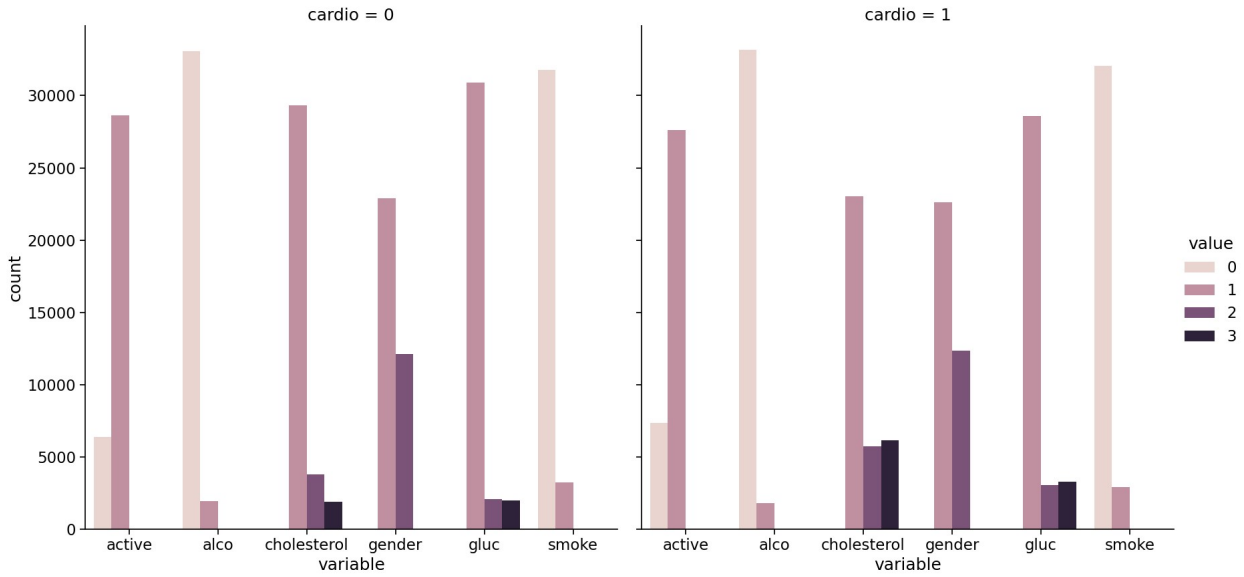


Видим, что классы целевой переменной `cardio` сбалансированы, отлично!

Можно также разбить элементы обучающей выборки по значениям целевой переменной: иногда на таких графиках можно сразу увидеть самый значимый признак.

```
train_uniques = pd.melt(frame=train, value_vars=['gender',
'cholesterol',
                                                 'gluc', 'smoke',
'alco',
                                                 'active'],
                        id_vars=['cardio'])
train_uniques = pd.DataFrame(train_uniques.groupby(['variable',
'value',
                                                    'cardio'])
['value'].count()) \
    .sort_index(level=[0, 1]) \
```

```
    .rename(columns={'value': 'count'}) \
    .reset_index()
sns.factorplot(x='variable', y='count', hue='value',
               col='cardio', data=train_uniques, kind='bar', height=9);
```



Видим, что в зависимости от целевой переменной сильно меняется распределение холестерина и глюкозы. Совпадение?

Немного статистики по уникальным значениям признаков.

```
for c in train.columns:
    n = train[c].nunique()
    print(c)

    if n <= 3:
        print(n, sorted(train[c].value_counts().to_dict().items()))
    else:
        print(n)
    print(10 * '-')
```

```
age
8076
----------
gender
2 [(1, 45530), (2, 24470)]
----------
height
109
----------
weight
287
----------
```

```
ap_hi
153
----------
ap_lo
157
----------
cholesterol
3 [(1, 52385), (2, 9549), (3, 8066)]
----------
gluc
3 [(1, 59479), (2, 5190), (3, 5331)]
----------
smoke
2 [(0, 63831), (1, 6169)]
----------
alco
2 [(0, 66236), (1, 3764)]
----------
active
2 [(0, 13739), (1, 56261)]
----------
cardio
2 [(0, 35021), (1, 34979)]
----------
```

Итого:

- Пять количественных признаков (без id)
- Семь категориальных
- 70000 объектов

---

1. Визуализируем корреляционную матрицу

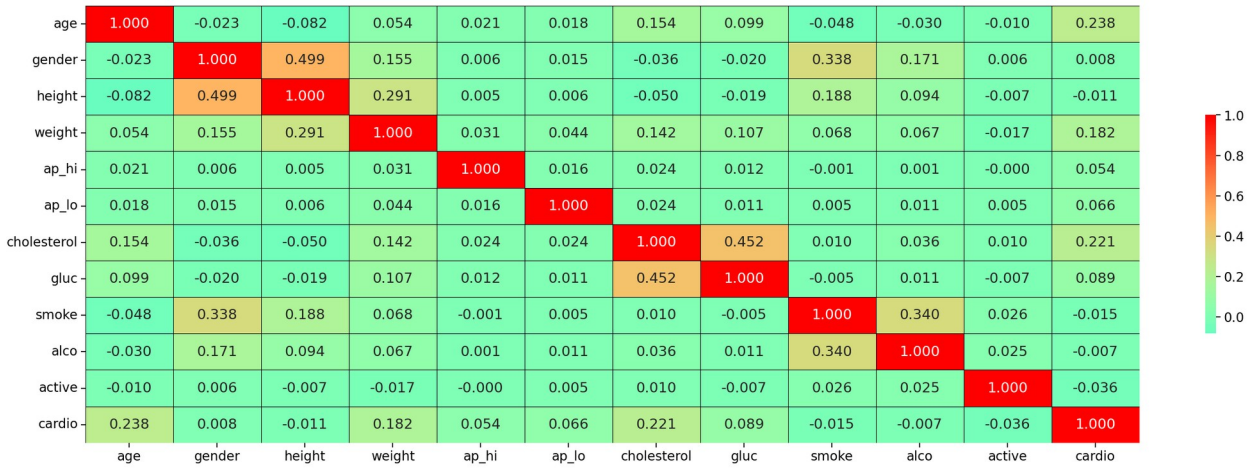
Для того чтобы лучше понять признаки в датасете, можно посчитать матрицу коэффициентов корреляции между признаками.

Постройте heatmap корреляционной матрицы. Матрица формируется средствами `Pandas`, со стандартным значением параметров.

1. Какие два признака больше всего коррелируют (по Пирсону) с признаком `height` ?

- Gluc, Cholesterol
- Weight, Alco
- Smoke, Alco
- Weight, Gender

```
corr_matrix = train.corr(method='pearson')
plt.figure(figsize=(30, 10))
sns.heatmap(corr_matrix, vmax=1, center=0, fmt='.3f', annot=True,
linewidths=.6, linecolor = 'black', cbar_kws={"shrink": .5},
cmap='rainbow')
plt.show()
hight_corr = corr_matrix['height'].sort_values(ascending=False)
hight_corr
```




```
height          1.000000
gender          0.499033
weight          0.290968
smoke           0.187989
alco            0.094419
ap_lo           0.006150
ap_hi           0.005488
active         -0.006570
cardio         -0.010821
gluc           -0.018595
cholesterol    -0.050226
age            -0.081515
Name: height, dtype: float64
```

---

2. Распределение роста для мужчин и женщин

Как мы увидели, в процессе исследования уникальных значений пол кодируется значениями 1 и 2, расшифровка изначально не была нам дана в описании данных, но мы догадались, кто есть кто, посчитав средние значения роста (или веса) при разных значениях признака `gender`. Теперь сделаем то же самое, но графически.

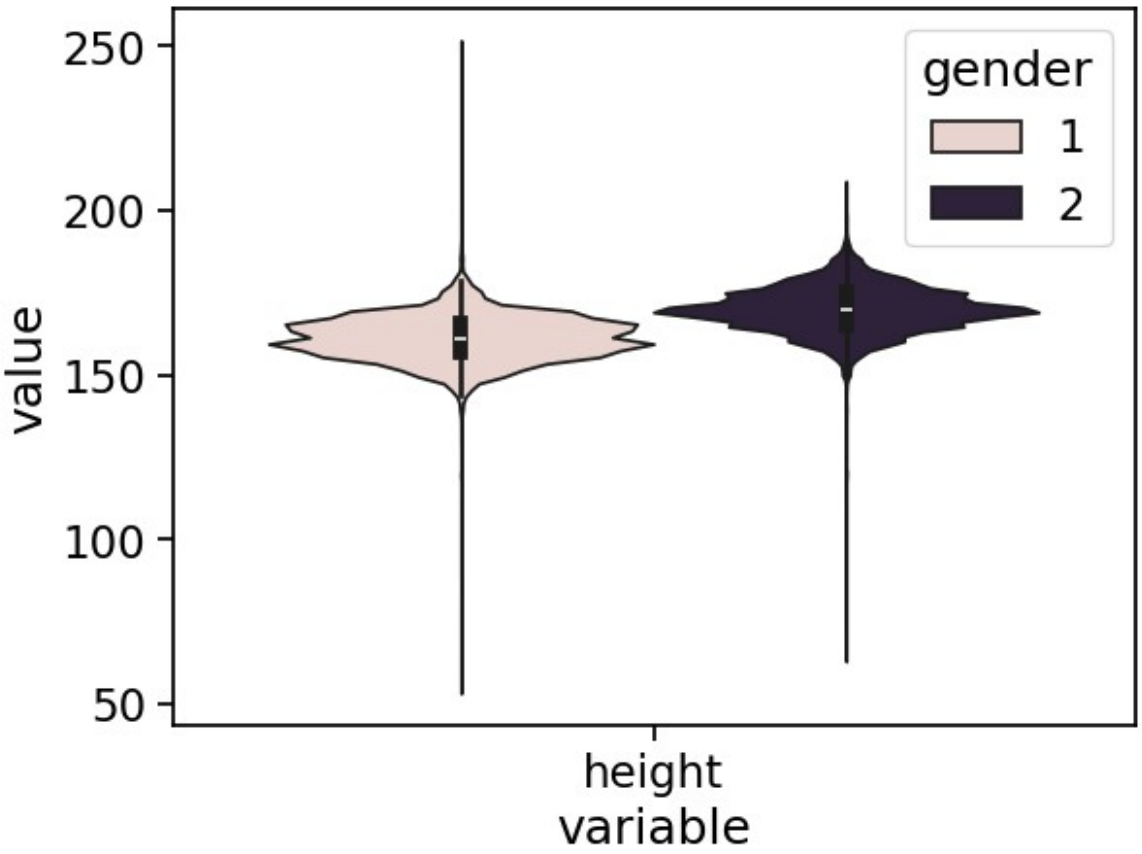
Постройте violinplot для роста и пола. Используйте:

- hue – для разбивки по полу
- scale – для оценки количества каждого из полов

Для корректной отрисовки, преобразуйте DataFrame в "Long Format"-представление с помощью функции melt в pandas. еще один пример

```
train_melted = pd.melt(train, id_vars=['gender'], value_vars
=['height'])
sns.violinplot(x='variable', y = 'value', hue = 'gender',
data=train_melted, scale= 'count')
```

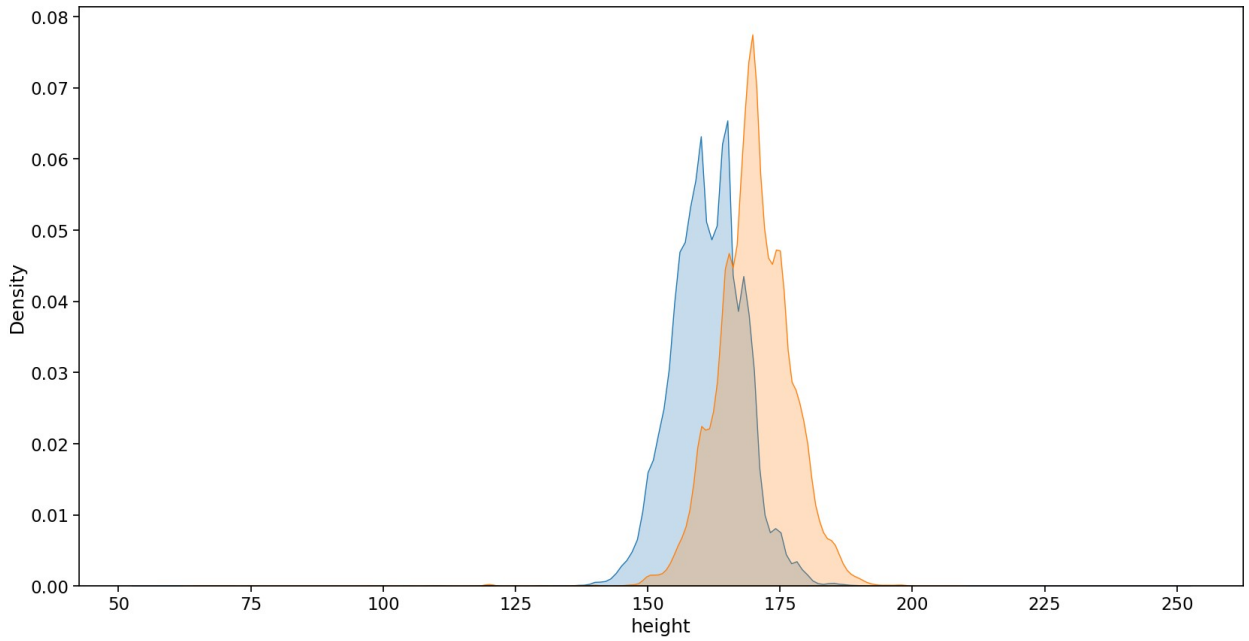
```
<Axes: xlabel='variable', ylabel='value'>
```



Постройте на одном графике два отдельных kdeplot роста, отдельно для мужчин и женщин. На нем разница будет более наглядной, но нельзя будет оценить количество мужчин/женщин.

```
plt.figure(figsize=(20, 10))
sns.kdeplot(data=train[train['gender'] == 1]['height'], shade = True)
sns.kdeplot(data=train[train['gender'] == 2]['height'], shade = True)
```

```
<Axes: xlabel='height', ylabel='Density'>
```



3. Ранговая корреляция

В большинстве случаев достаточно воспользоваться линейным коэффициентом корреляции *Пирсона* для выявления закономерностей в данных, но мы пойдём чуть дальше и используем ранговую корреляцию, которая поможет нам выявить пары, в которых меньший ранг из вариационного ряда одного признака всегда предшествует большему другого (или наоборот, в случае отрицательной корреляции).

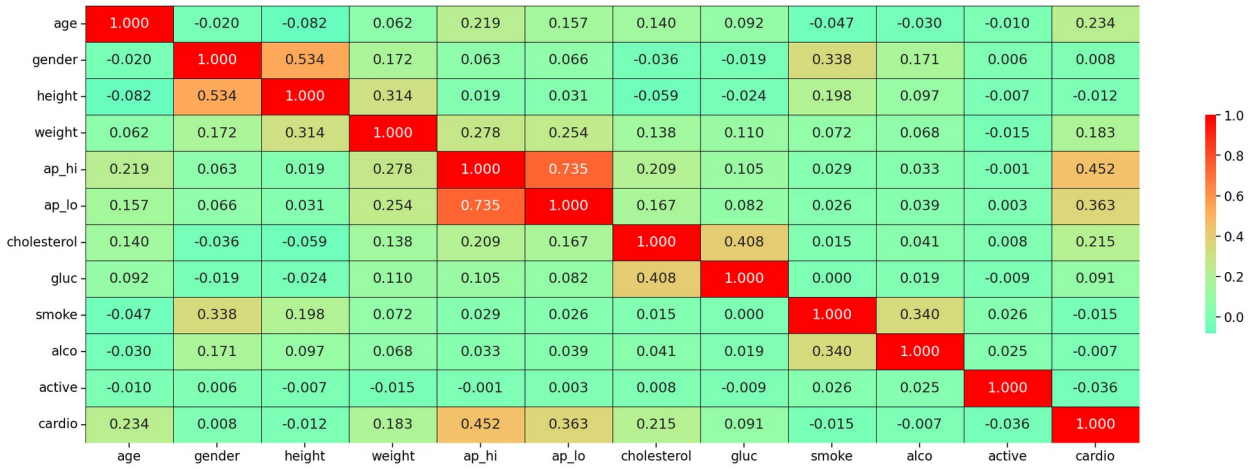
Постройте корреляционную матрицу, используя коэффициент Спирмена

3.1 Какие признаки теперь больше всего коррелируют (по Спирмену) друг с другом?

- Height, Weight
- Age, Weight
- Ap_hi, Ap_lo
- Cholesterol, Gluc
- Cardio, Cholesterol
- Smoke, Alco

```
corr_matrix1 = train.corr(method='spearman')
plt.figure(figsize=(30, 10))
sns.heatmap(corr_matrix1, vmax=1, center=0, fmt='.3f', annot=True,
linewidths=.6, linecolor = 'black', cbar_kws={"shrink":
```

```
.5},cmap='rainbow')
plt.show()
hight_corr1 = corr_matrix1['height'].sort_values(ascending=False)
```



3.2 Почему мы получили такое большое (относительно) значение ранговой корреляции у этих признаков?

- Неточности в данных (ошибки при сборе данных)
- Связь ошибочна, переменные никак не должны быть связаны друг с другом
- Природа данных

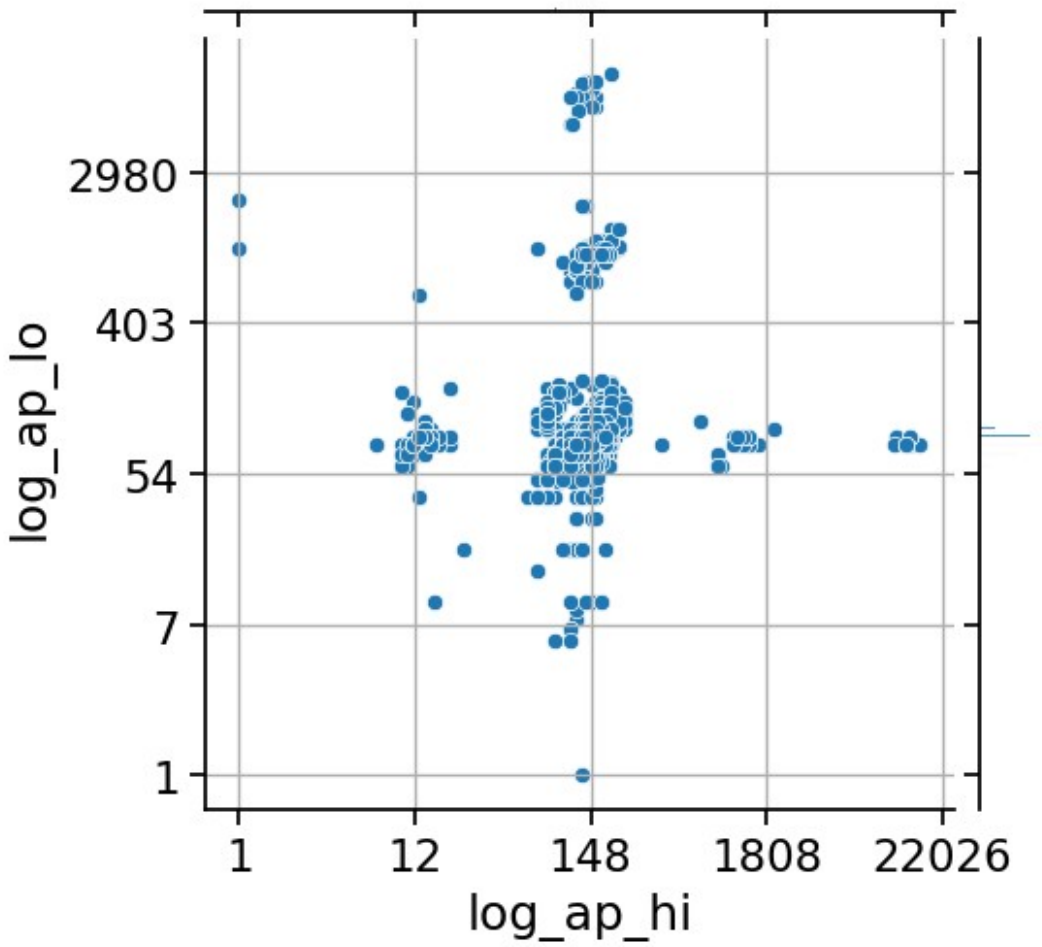
---

4. Совместное распределение признаков

Постройте совместный график распределения jointplot двух наиболее коррелирующих между собой признаков (по Спирмену).

Кажется, наш график получился неинформативным из-за выбросов в значениях. Постройте тот же график, но с логарифмической шкалой (чтобы не получать OverflowError необходимо отфильтровать значения меньше либо равные нулю).

```
filtered_data = train[(train['ap_hi'] > 0) & (train['ap_lo'] > 0)]
filtered_data['log_ap_hi'] = np.log(filtered_data['ap_hi'])
filtered_data['log_ap_lo'] = np.log(filtered_data['ap_lo'])
g = sns.jointplot(x = 'log_ap_hi', y = 'log_ap_lo', data =
filtered_data, kind = 'scatter')
g.ax_joint.grid(True)
g.ax_joint.yaxis.set_major_formatter(matplotlib.ticker.FuncFormatter(l
ambda x, pos: str(round(int(np.exp(x))))))
g.ax_joint.xaxis.set_major_formatter(matplotlib.ticker.FuncFormatter(l
ambda x, pos: str(round(int(np.exp(x))))))
```

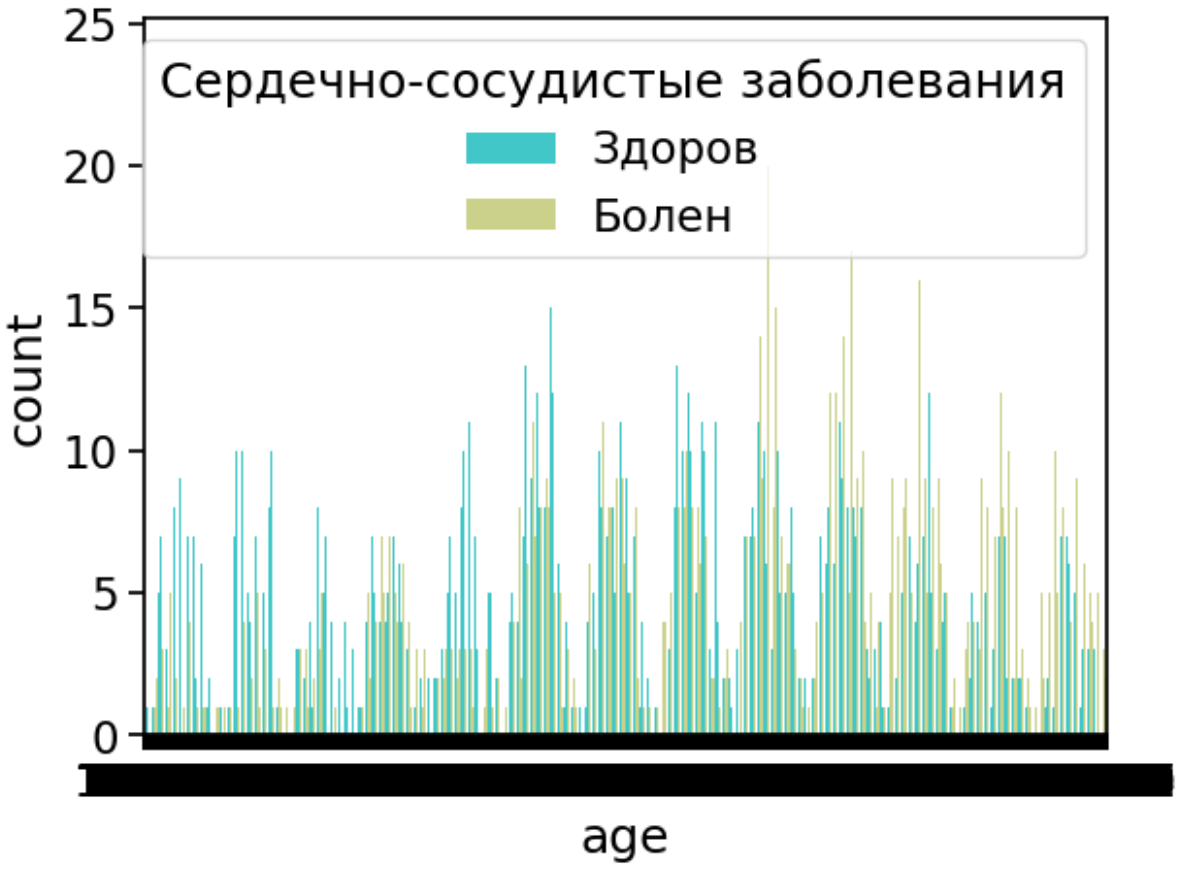


4.1 Сколько чётко выраженных кластеров получилось на совместном графике выбранных признаков, с логарифмической шкалой? Под кластером в данной задаче понимается плотное скопление точек, в окрестности которого пренебрежительно мало одиночных наблюдений и которое визуально отделимо от других кластеров.

- 1
- 2
- 3
- больше трёх

```
sns.countplot(x="age", hue='cardio', data=train, palette="rainbow")
plt.legend(title='Сердечно-сосудистые заболевания', loc='upper right',
labels=['Здоров','Болен'])
```

```
<matplotlib.legend.Legend at 0x24b70844610>
```



---

5. Возраст

Посчитаем, сколько полных лет было респондентам на момент их занесения в базу.

```
train['age_years'] = (train['age'] // 365.25).astype(int)
```

Постройте Countplot, где на оси абсцисс будет отмечен возраст, на оси ординат – количество. Каждое значение возраста должно иметь два столбца, соответствующих количеству человек каждого класса **cardio** (здоров/болен) данного возраста.

5. В каком возрасте количество пациентов с ССЗ впервые становится больше, чем здоровых?

- 44
- 49
- 53

- 62

```
df_age = train.groupby(['age_years',
'cardio']).size().unstack(fill_value=0)
df_age
```

cardio	0	1
age_years		
29	3	0
30	1	0
39	1430	450
40	1194	330
41	1416	588
42	915	400
43	1350	787
44	907	501
45	1317	897
46	906	593
47	1195	1108
48	928	777
49	2131	1533
50	1806	1165
51	1897	1704
52	1634	1409
53	2061	2105
54	1712	1597
55	1890	2335
56	1541	1766
57	1703	2284
58	1401	1709
59	1468	2376
60	1243	1690
61	936	1997
62	657	1334
63	805	2132
64	574	1412