Государственное бюджетное общеобразовательное учреждение города Москвы «Школа №1359 им. Авиаконструктора М.Л. Миля»

**Анализ биометрических показателей пациентов с использованием компьютерных методов и предсказание возможности сердечного приступа**

Выполнил:

Мифтахов Максим Ренатович, 10 класс

Научный руководитель:

Шиншинова Ирина Олеговна

Москва, 2024

Оглавление

[Введение 3](#_Toc158122046)

[Данные 5](#_Toc158122047)

[Формализация проблемы 6](#_Toc158122048)

[Импорт библиотек и предобработка 7](#_Toc158122049)

[Исследование и визуализация данных 10](#_Toc158122050)

[Глубокий анализ признаков и формирование выборок 15](#_Toc158122051)

[Нормализация данных 19](#_Toc158122052)

[Используемые модели 20](#_Toc158122053)

[Выводы 24](#_Toc158122054)

# Введение

Проблема анализа биометрических показателей пациентов с использованием компьютерных методов и предсказание возможности сердечного приступа является актуальной в наши дни по нескольким причинам.

Во-первых, сердечные заболевания остаются одной из основных причин смерти во многих странах мира, и раннее выявление риска сердечного приступа может способствовать более эффективному лечению.

Во-вторых, использование компьютерных методов для анализа биометрических данных пациентов позволяет проводить более точный и объективный анализ, что может улучшить способность предсказания возможности сердечного приступа.

Наконец, развитие технологий и методов машинного обучения позволяет создавать более точные модели прогнозирования сердечных заболеваний на основе биометрических данных, что открывает новые возможности для раннего выявления риска сердечного приступа и предупреждения заболеваний.

Таким образом, анализ биометрических показателей пациентов с применением компьютерных методов и предсказание возможности сердечного приступа остаются актуальными и могут иметь значительное влияние на улучшение диагностики и лечения сердечных заболеваний.

Сердечный приступ, или инфаркт миокарда, происходит, когда по какой-либо причине артерия, отвечающая за подачу крови к сердечной мышце, становится заблокирована, что приводит к отсутствию кислорода и питательных веществ для части сердечной мышцы. Это приводит к некрозу ткани и повреждению функционирующей мышечной ткани сердца.

Сердечный приступ является серьезным и опасным состоянием, которое требует немедленной медицинской помощи. Безболезненность, чувство давления, жжения или сдавливания в области груди, а также необычные боли в шее, спине, плечах, руках или животе могут быть признаками инфаркта. Однако у некоторых людей сильные боли отсутствуют или являются менее явными.

Незамедлительное лечение инфаркта может помочь восстановить нормальное кровоснабжение сердца и предотвратить серьезные и даже фатальные осложнения, такие как аритмия, сердечная недостаточность или разрушение сердечной мышцы.

Сердечный приступ опасен, так как может приводить к смертельным последствиям и угрожать здоровью и жизни человека. Поэтому важно принимать меры для предотвращения сердечных заболеваний и заботиться о своем здоровье, включая здоровый образ жизни, регулярные медицинские осмотры и контроль факторов риска.

Выполняя текущую работу, мы сможем лучше понять, что влияет на возможность сердечного приступа, а также составим модель, которая будет строить прогноз.

**Цель**

Изучить данные с биометрическими показателями пациентов и с использованием компьютерных методов и составить модель предсказания возможности сердечного приступа.

**Задачи**

Задачи текущей работы:

1. Проанализировать данные с биометрическими показателями пациентов: выявить основные зависимости и закономерности.
2. Составить модели для предсказания возможности сердечного приступа, оценить их эффективность.
3. Улучшить знания синтаксиса python и библиотек для анализа данных.
4. Проанализировать результаты и сформировать план дальнейшего исследования вопроса.

# Данные

Данные взяты из открытого доступа с сайта Kaggle.com. Kaggle.com - это платформа для анализа данных и машинного обучения. Она была создана в 2010 году и с тех пор стала одним из самых популярных мест для соревнования по машинному обучению и анализу данных. На это сайте пользователи могут загружать свои данные, создавать модели машинного обучения и соревноваться с другими участниками. Также на сайте есть множество задач для решения, включая задачи по распознаванию изображений, обработке естественного языка и многим другим. Платформа является отличным местом для обучения и развития навыков в области анализа данных и машинного обучения. Большая часть таблиц для анализа данных хранится в формате csv. В этом формате каждая строка таблицы представляет собой строку файла, а каждый столбец - поле в строке. Данные в каждом поле разделены запятыми, а в конце строки ставится символ перевода строки. CSV-файлы обычно используются для обмена данными между различными приложениями и базами данных.

Таблица heart.csv, которую мы будем использовать, имеет следующие параметры:

1. **Age:** возраст пациента
2. **Sex:** пол пациента (0 – женский, 1 – мужской)
3. **Exng :** стенокардия, вызванная физическими упражнениями (1 – да, 0 – нет). Стенокардия – это клиническая форма ишемической болезни сердца.
4. **Ca (caa):** количество магистральных сосудов (0 – 3)
5. **Cp:** тип боли в груди
   1. 0: типовая стенокардия
   2. 1: нетиповая стенокардия
   3. 2: боль, не связанная с стенокардия
   4. 3: бессимптомная
6. **Trtbps:** давление крови в покое (в мм. ртутного столба)
7. **Chol:** холестерин в мг/дл.
8. **Fbs:** уровень сахара в крови натощак (больше 120 мг/дл) (1 – да, 0 – нет)
9. **Restecg:** результат электрокардиограммы в покое:
   1. 0: нормально
   2. 1: наличие аномальных ST-T волн
   3. 2: показывает возможность гипертрофии левого желудочка
10. **Thalachh:** максимальная частота сердечных сокращений
11. **Thall:** вариабельность сердечного ритма
12. **Oldpeak:** предыдущий пик
13. **Output:** 0 – минимальный шанс сердечного приступа, 1 – максимальный шанс сердечного приступа

# Формализация проблемы

При решении проблем машинного обучения, как правило, существует два типа данных (и модели машинного обучения):¶

* Контролируемые данные (Supervised data): всегда есть одна или несколько целевых переменных, связанных с ними.
* Неконтролируемые данные (Unsupervised data): не имеют целевой переменной.

Контролируемую проблему значительно легче решить, чем неконтролируемую. Проблема, в которой мы пытаемся предсказывать возможность сердечного приступа - контролируемая. Например, если проблема заключается в прогнозировании цен на жилье, учитывая исторические цены на жилье, с такими функциями, как наличие больницы, школы или супермаркета, расстояние до ближайшего общественного транспорта и т. д., является контролируемой проблемой. Аналогичным образом, когда нам предоставляются изображения кошек и собак, и мы заранее знаем, какие из них являются кошками, а какие - собаками, и, если задача заключается в создании модели, которая предсказывает, является ли предоставленное изображение кошки или собаки, проблема считается контролируемой.

В этом наборе данных, у нас есть проблема управляемого машинного обучения для прогнозирования сердечной недостаточности.

Сердечно-сосудистые заболевания (СЗС) являются причиной смерти номер 1 во всем мире, унося, по оценкам ученых, 17,9 миллиона жизней каждый год, что составляет 31% всех смертей во всем мире. Четыре из 5 смертей от СЗС происходят из-за сердечных приступов и инсультов, и одна треть этих смертей происходит преждевременно у людей в возрасте до 70 лет. Сердечная недостаточность является распространенной патологией, вызванной сердечно-сосудистыми заболеваниями.

Люди с сердечно-сосудистыми заболеваниями или с высоким сердечно-сосудистым риском (из-за наличия одного или нескольких факторов риска, таких как гипертония, диабет, гиперлипидемия или уже установленное заболевание) нуждаются в раннем обнаружении такой проблемы, чтобы вовремя начать лечение и предотвратить летальный исход.

Для анализа данных будем использовать python. Начнем решение поставленной задачи с импорта данных и их анализа.

# Импорт библиотек и предобработка

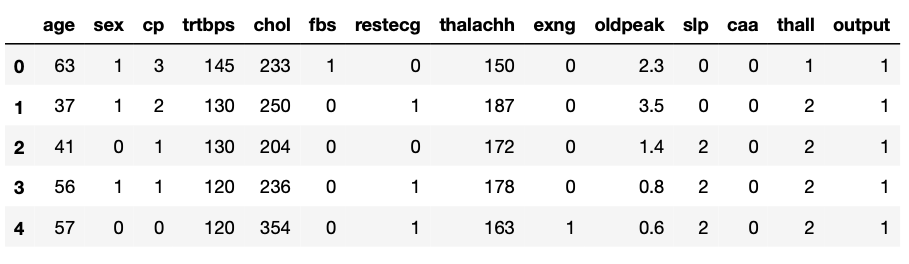
Начинаем с импорта библиотек, которые мы будем использовать

import os  
import joblib  
import numpy as np  
import pandas as pd  
import warnings  
  
import matplotlib  
import matplotlib.pyplot as plt  
from matplotlib import ticker  
import seaborn as sns  
  
# setting up options  
pd.set\_option('display.max\_rows', None)  
pd.set\_option('display.max\_columns', None)  
pd.set\_option('float\_format', '{:f}'.format)  
warnings.filterwarnings('ignore')  
  
import warnings as wr  
wr.filterwarnings("ignore") #to ignore the warnings

Считываем таблицу и выводим первые 5 ее строк:

df = pd.read\_csv("heart.csv")  
df.head()

Вывод:



**Наблюдение:** можно видеть, что все столбцы имеют тип данных int или float.Output - результат для прогнозирования (предрасположен или нет человек к сердечному приступу)

Посчитаем количество строк и столбцов в матрице:

print('Количество рядов : ',df.shape[0], ',количество столбцов: ',df.shape[1])

Количество рядов : 303, количество столбцов: 14

Выведем информацию о таблице:

df.info()

Получаем следующий вывод:

<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 303 entries, 0 to 302  
Data columns (total 14 columns):  
 # Column Non-Null Count Dtype   
--- ------ -------------- -----   
 0 age 303 non-null int64   
 1 sex 303 non-null int64   
 2 cp 303 non-null int64   
 3 trtbps 303 non-null int64   
 4 chol 303 non-null int64   
 5 fbs 303 non-null int64   
 6 restecg 303 non-null int64   
 7 thalachh 303 non-null int64   
 8 exng 303 non-null int64   
 9 oldpeak 303 non-null float64  
 10 slp 303 non-null int64   
 11 caa 303 non-null int64   
 12 thall 303 non-null int64   
 13 output 303 non-null int64   
dtypes: float64(1), int64(13)  
memory usage: 33.3 KB

**Наблюдение:** видим, что во всем наборе данных нет отсутствующих строк. поэтому нам не нужно проводить дополнительное заполнение / удаление. Все столбцы, кроме oldpeak (float), имеют тип данных int.

Дополнительно рассмотрим, есть ли в нашей таблицы дубликаты, чтобы не учитывать их несколько раз:

df.duplicated().sum()

В таблице нашелся один дубликат, исключим эту строку.

df.drop\_duplicates(inplace=True)

Посмотрим на статистические показатели параметров:

df.describe().T

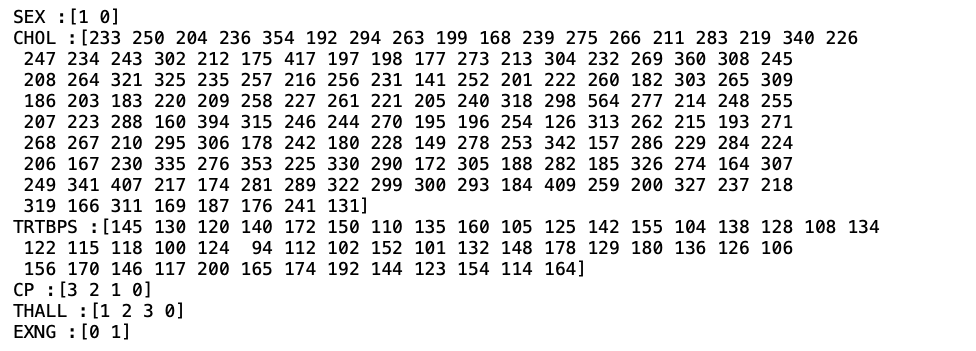
****

Наблюдения:

* Среднее кровяное давление (trtbps) у пациентов составляет 130, в то время как максимальное значение доходит до 200.
* Средняя частота сердечных сокращений (thalachh) среди участников исследования составляет 150, в то время как в целом она колеблется от 133 до 202
* Возраст группы колеблется от 29 до 77 лет, а средний возраст составляет 55,5.

Посмотрим на уникальные значения некоторых параметров:

list\_col=['sex','chol','trtbps','cp','thall','exng']  
  
for col in list\_col:   
 print('{} :{} ' . format(col.upper(),df[col].unique()))



Наблюдения:

* Самый высокий уровень холестерина (chol) - 564, а самый низкий - 126.
* Кровяное давление в покое у людей варьируется от 94 до 200.

# Исследование и визуализация данных

Первый вопрос, который возникает - зачем мы анализируем данные, а не сразу строим модель? Это делаем для того, чтобы:

1. Лучше понимать данные и очистить их от выбросов в случае необходимости.
2. Определить связь между признаками и целевым показателем.
3. Оценить важные и неважные переменные для анализа.
4. Обработка отсутствующих значений или значений, которые заполнены некорректно.
5. Получить больше выводов о данных и использовать их для построения модели.

Процесс обработки и анализа данных отнимает много времени, но позволяет сделать выборку более качественной, что положительно влияет на построение модели и дальнейшее исследование.

**Визуализация данных:**

Построим гистограмму распределения пола:

plt.figure(figsize=(12,6))  
ax=plt.axes()  
ax.set\_facecolor("green")  
p = sns.countplot(data=df, x="sex", palette='pastel')

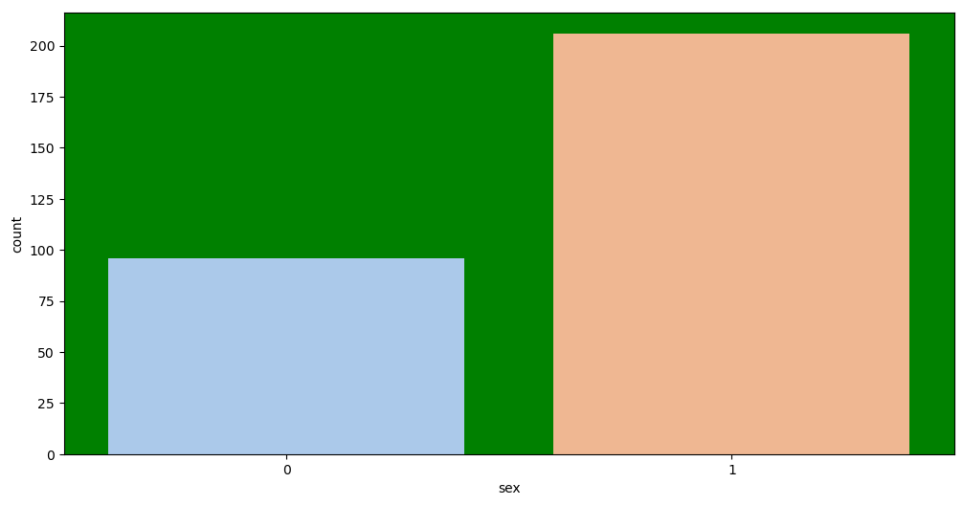


Figure 1

Наблюдение:

* В таблице есть данные 96 женщинах и 206 мужчинах, перевес в пользу вторых более чем в два раза.

Рассмотрим распределение по типу боли в грудной клетке:

ax=plt.axis()  
sns.countplot(x='cp', data=df, palette='pastel')  
pass

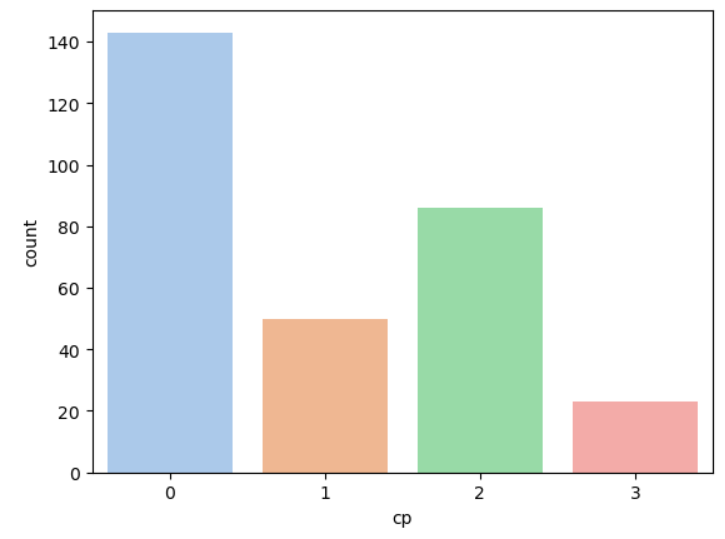


Figure 2

**Наблюдение:**

Людей категории 0 (типовая стенокардия) больше всего в таблице, в то время как количество людей бессимптомными болями меньше всего. Значит, в большинстве случаев получается точно определить тип грудной боли.

Рассмотрим также распределение параметра fbs (уровень сахара в крови):

sns.countplot(x='fbs', data=df, palette='pastel')  
pass

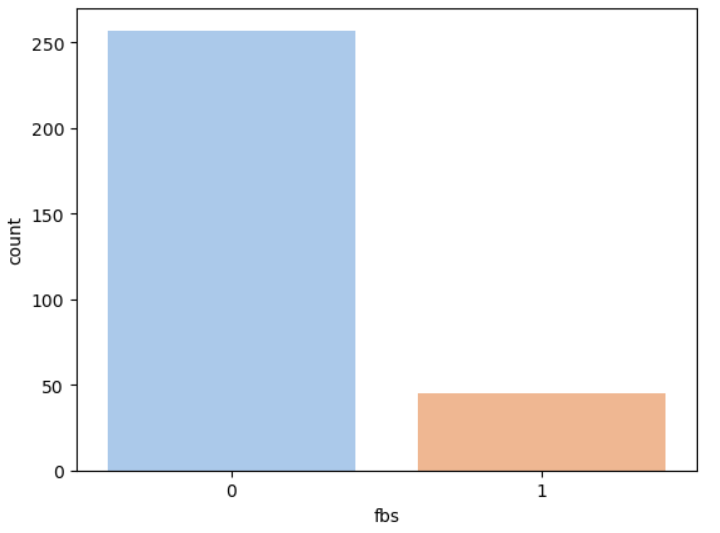


Figure 3

Наблюдение: Люди первой категории составляют менее 25% всех пациентов. То есть у большей части пациентов уровень сахара в крови в пределах нормы для возраста.

Построим аналогичную гистограмму для параметра rectecg (результат электрокардиограммы в покое):

sns.countplot(x='restecg', data=df, palette='pastel')  
pass

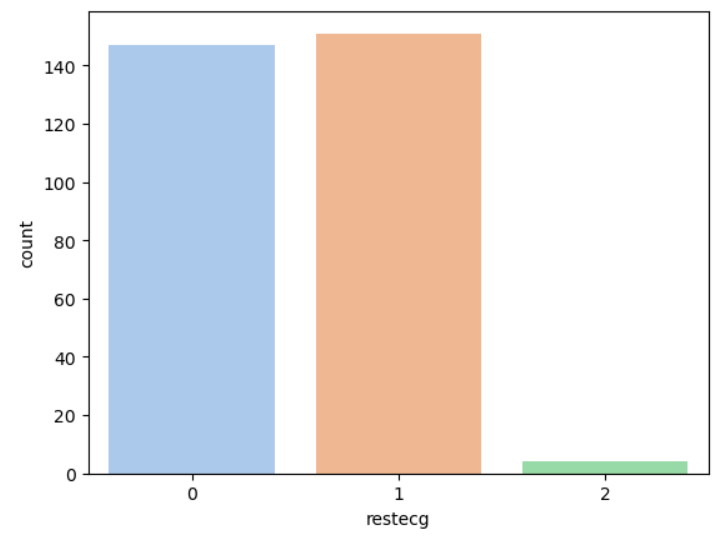


Figure 4

Наблюдение:

* Количество подмножеств с значениями 0 и 1 почти одинаково.
* Количество людей с гипертрофией левого желудочка (тип 2) незначительно.

Далее построим скрипичную диаграмму для параметров caa, age и output. Скрипичная диаграмма представляет собой график, на котором по горизонтальной оси откладываются различные категории, а по вертикальной оси - количество или доля наблюдений в каждой категории. Сама диаграмма напоминает форму скрипки, откуда и происходит ее название. Скрипичные диаграммы обычно используются для быстрого определения основных тенденций, сравнения различных сегментов данных и выявления аномалий.

plt.figure(figsize = (10,10))  
sns.violinplot(x='caa',y='age',data=df)  
sns.swarmplot(x=df['caa'],y=df['age'],hue=df['output'], palette='pastel')  
pass

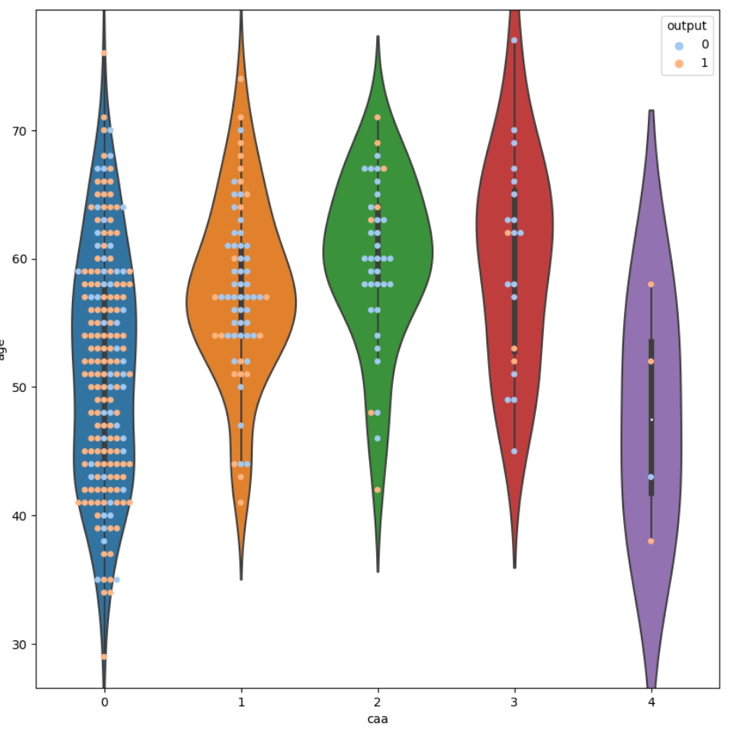


Figure 5

Наблюдение:

* По данному графику видно, что люди, принадлежащие к категории caa 0, независимо от их возраста, очень склонны к сердечному приступу.
* Количество людей категории caa 4 незначительно, но по графику можно сформировать гипотезу, что около 75% из них получают сердечные приступы.
* Люди, принадлежащие к категориям «1», «2» и «3», подвергаются более или менее аналогичному риску.

**Матрица корреляции**

Для построения модели важно рассматривать корреляцию признаков, чтобы не включать дубликаты. Так как в нашем случае признаков немного, все из них мы включим в модель. Но будет важным для дальнейшего изучения рассмотреть их зависимость между друг другом. Корреляция — это статистическая мера, которая показывает степень связи между двумя переменными. Если две переменные имеют высокую корреляцию, то это означает, что изменение одной переменной приводит к изменению другой переменной. Если корреляция отрицательная, то увеличение одной переменной приводит к уменьшению другой переменной и наоборот. Дополнительно построим таблицу:

df.corr().style.background\_gradient(cmap='RdYlGn')



Figure 6

Здесь мы видим, что болезнь сердца имеет положительную корреляцию с типом боли в грудной клетке (cp) и максимальной частотой сердечных сокращений (thalach). Также заметна отрицательная корреляция с количеством магистральных сосудов (ca / caa) и вариабельностью сердечного ритма (thal).

Рассмотрим распределение участников по параметру caa (количество магистральных сосудов), а также зависимость холестерина от возраста и пола:

fig = plt.figure(figsize = (8,4))  
  
plt.subplot(121)  
plt.title('Количество магистральных сосудов')  
sns.countplot(data = df, x = 'caa', hue = 'sex')  
  
plt.subplot(122)  
plt.title('Возраст и холестирин')  
sns.scatterplot(data = df, x = 'age', y = 'chol', hue = 'sex')  
  
plt.tight\_layout()  
plt.show()

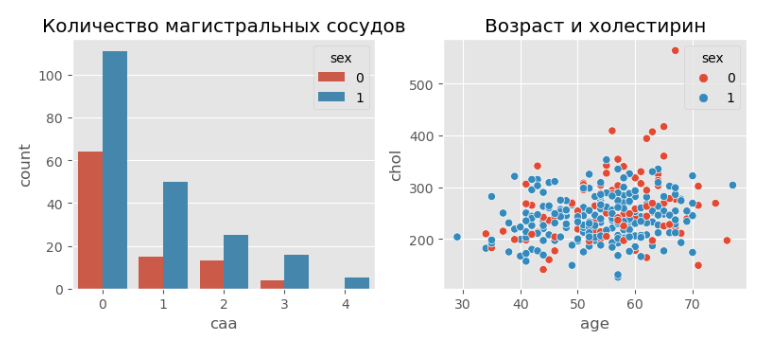


Figure 7

Видим, что у женщин в возрасте после 50 холестерин в среднем выше, чем у мужчин.

# Глубокий анализ признаков и формирование выборок

Начнем формировать данные для обучения моделей и их тестирования. Для построения части моделей используется идея обучения этой модели на части данных и проверки на оставшейся части. Напишем для этого код:

integer\_features = ['age','chol','trtbps','cp','thall','exng']  
unique\_values\_train = pd.DataFrame(df[integer\_features].nunique())  
unique\_values\_train = unique\_values\_train.reset\_index(drop=False)  
unique\_values\_train.columns = ['Features', 'Count']  
  
unique\_values\_percent\_train = pd.DataFrame(df[integer\_features].nunique()/df.shape[0])  
unique\_values\_percent\_train = unique\_values\_percent\_train.reset\_index(drop=False)  
unique\_values\_percent\_train.columns = ['Features', 'Count']

Напишем код для анализа статистических показателей по этим выборкам:

plt.rcParams['figure.dpi'] = 400  
fig = plt.figure(figsize=(6, 4), facecolor='#f6f5f5')  
gs = fig.add\_gridspec(2, 2)  
gs.update(wspace=0.4, hspace=0.5)  
  
background\_color = "#f6f5f5"  
sns.set\_palette(['#ffd514'] \* 6)  
  
ax0 = fig.add\_subplot(gs[0, 0])  
for s in ["right", "top"]:  
 ax0.spines[s].set\_visible(False)  
ax0.set\_facecolor(background\_color)  
ax0\_sns = sns.barplot(ax=ax0, y=unique\_values\_train['Features'], x=unique\_values\_train['Count'],  
 zorder=2, linewidth=0, orient='h', saturation=1, alpha=1)  
ax0\_sns.set\_xlabel("Уникальные значения", fontsize=4, weight='bold')  
ax0\_sns.set\_ylabel("Фичи", fontsize=4, weight='bold')  
ax0\_sns.tick\_params(labelsize=4, width=0.5, length=1.5)  
ax0\_sns.grid(which='major', axis='x', zorder=0, color='#EEEEEE', linewidth=0.4)  
ax0\_sns.grid(which='major', axis='y', zorder=0, color='#EEEEEE', linewidth=0.4)  
ax0.text(0, -1.5, 'Уникальные значения - Train', fontsize=6, ha='left', va='top', weight='bold')  
ax0.text(0, -1, 'можно рассматривать как классификационные признаки', fontsize=4, ha='left', va='top')  
ax0.get\_xaxis().set\_major\_formatter(matplotlib.ticker.FuncFormatter(lambda x, p: format(int(x), ',')))  
# data label  
for p in ax0.patches:  
 value = f'{p.get\_width():,.0f}'  
 x = p.get\_x() + p.get\_width() + 200  
 y = p.get\_y() + p.get\_height() / 2  
 ax0.text(x, y, value, ha='left', va='center', fontsize=4,  
 bbox=dict(facecolor='none', edgecolor='black', boxstyle='round', linewidth=0.3))  
  
ax1 = fig.add\_subplot(gs[0, 1])  
for s in ["right", "top"]:  
 ax1.spines[s].set\_visible(False)  
ax1.set\_facecolor(background\_color)  
ax1\_sns = sns.barplot(ax=ax1, y=unique\_values\_percent\_train['Features'], x=unique\_values\_percent\_train['Count'],  
 zorder=2, linewidth=0, orient='h', saturation=1, alpha=1)  
ax1\_sns.set\_xlabel("Процент уникальных значений", fontsize=4, weight='bold')  
ax1\_sns.set\_ylabel("Фичи", fontsize=4, weight='bold')  
ax1\_sns.tick\_params(labelsize=4, width=0.5, length=1.5)  
ax1\_sns.grid(which='major', axis='x', zorder=0, color='#EEEEEE', linewidth=0.4)  
ax1\_sns.grid(which='major', axis='y', zorder=0, color='#EEEEEE', linewidth=0.4)  
ax1.text(0, -1.5, 'Процент уникальных значений - Train', fontsize=6, ha='left', va='top', weight='bold')  
ax1.text(0, -1, 'можно рассматривать как классификационные признаки', fontsize=4, ha='left', va='top')  
# data label  
for p in ax1.patches:  
 value = f'{p.get\_width():.2f}'  
 x = p.get\_x() + p.get\_width() + 0.03  
 y = p.get\_y() + p.get\_height() / 2  
 ax1.text(x, y, value, ha='left', va='center', fontsize=4,  
 bbox=dict(facecolor='none', edgecolor='black', boxstyle='round', linewidth=0.3))  
  
background\_color = "#f6f5f5"  
sns.set\_palette(['#ff355d'] \* 6)

Вывод:

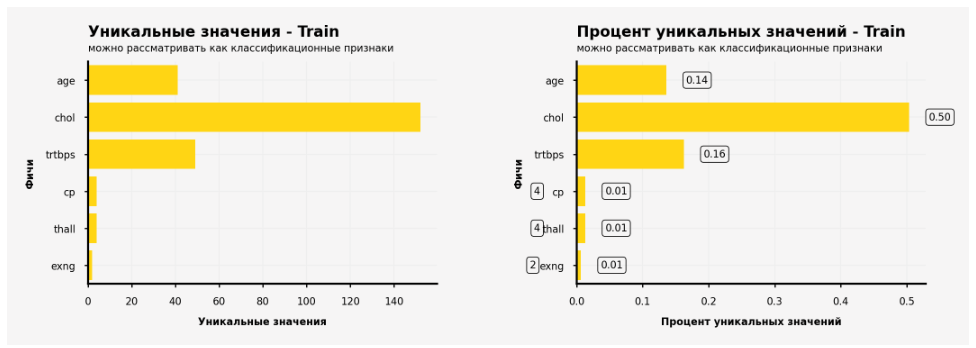


Figure 8

Наблюдения:

По этим графикам можно оценить уникальные значения и значения, которые имеют незначительный разброс.

**Анализ распределений величин**

Рассмотрим распределение величин, которые мы будем использовать в модели:

sns.color\_palette("pastel")  
sns.distplot(df.trtbps, label='trtbps', kde=True, bins=10, color='green')  
plt.legend()  
pass

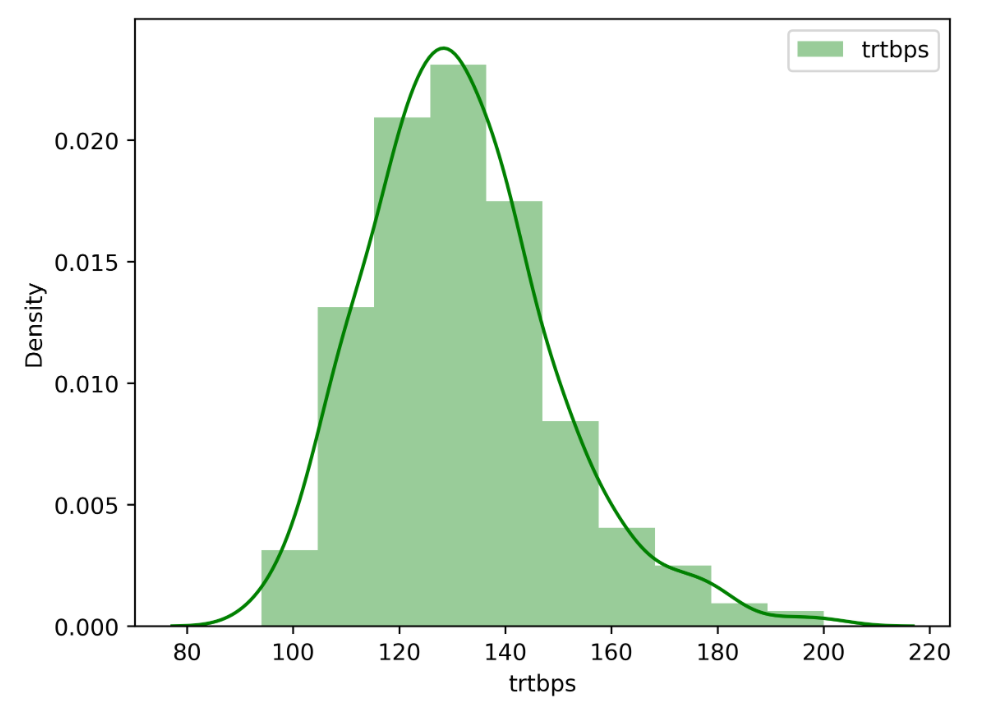
****

Figure 9

sns.distplot(df.chol, label='chol', kde=True, color='red')  
plt.legend()  
pass

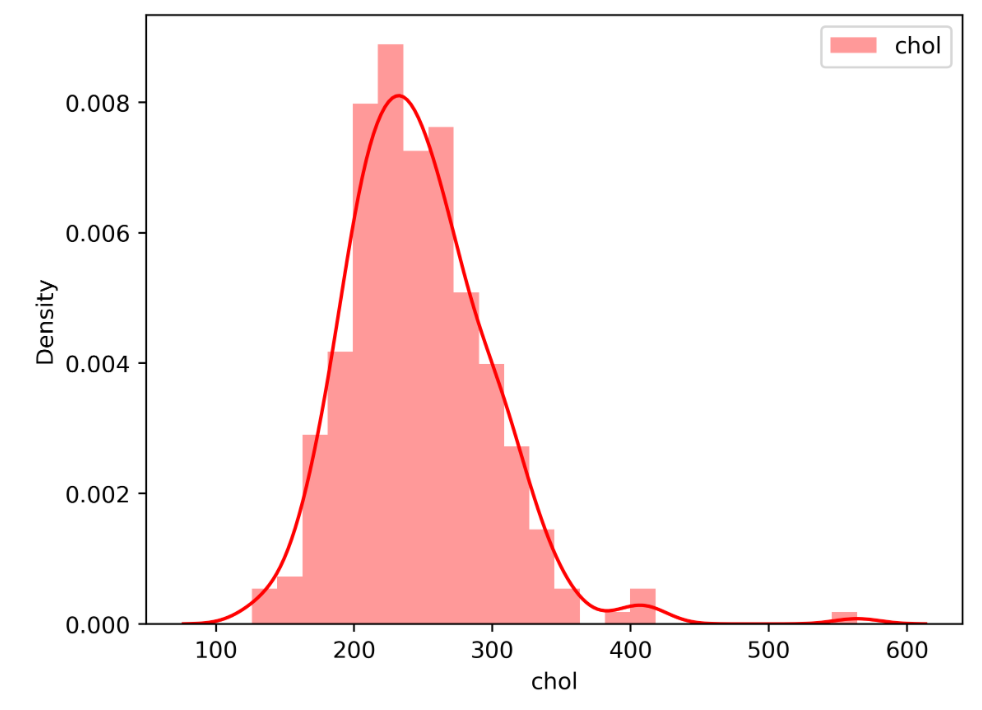
****

Figure 10

sns.distplot(df['thalachh'],label='thalachh', kde=True )  
plt.legend()  
pass

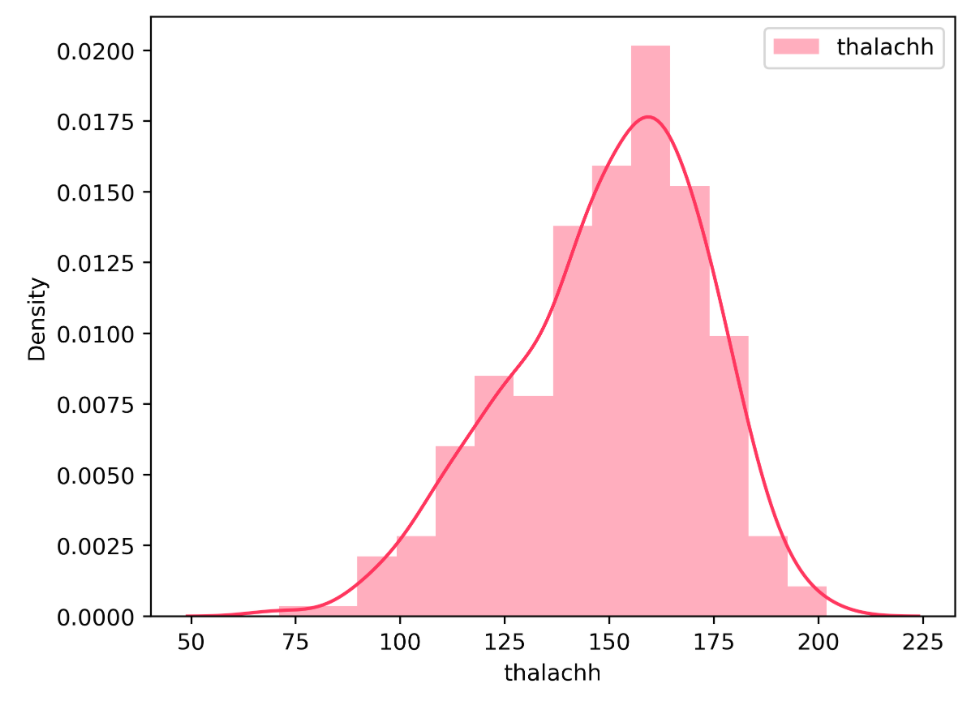
****

Figure 11

Наблюдения:

* Trtbps и chol визуально имеют нормальное распределение с незначительным сдвигом вправо.
* Для параметра thalachh данные сильно сдвинуты вправо. Выдвинуть гипотезу о распрделении данной величины не получается.

# Нормализация данных

Выделяем параметры и целевую переменную:

X =df.drop(["output"],axis=1)  
y =df["output"]

Использование minmax Scaler для масштабирования данных (нормализации). Нормализация данных производится при построении моделей машинного обучения с целью стандартизации переменных и уменьшения дисперсии. Это позволяет модели обучаться более эффективно и снижает вероятность переобучения. Кроме того, нормализация может улучшить интерпретацию результатов модели, так как она делает все переменные равнозначными и позволяет избежать ситуаций, когда одна переменная доминирует над другими.

from sklearn.preprocessing import MinMaxScaler  
scalerX = MinMaxScaler(feature\_range=(0, 1))  
X[X.columns] = scalerX.fit\_transform(X[X.columns])

# Используемые модели

Для первичного анализа результатов мы используем сразу несколько моделей, а дальнейшее изучение вопроса и выбора модели для детального изучения будет непосредственно связан с результатами. Будут использованы следующие модели:

1. Логистическая регрессия
2. Метод K-ближайших соседей
3. Метод опорных векторов (SVC)
4. Дерево принятия решения (Decisiontreeclassifier)
5. Случайный лес
6. Градиентный бустинг
7. XGB классификатора

Каждая из этих моделей имеет либо чисто-математический аппарат для построения прогноза (например, логистическая регрессия), либо итеративный (например, градиентный бустинг).

Импортируем библиотеки для моделей:

from sklearn.metrics import accuracy\_score  
from sklearn.metrics import confusion\_matrix  
from sklearn.linear\_model import LogisticRegression  
from sklearn.neighbors import KNeighborsClassifier  
from sklearn.svm import SVC  
from sklearn.tree import DecisionTreeClassifier  
from sklearn.ensemble import RandomForestClassifier  
from sklearn.ensemble import GradientBoostingClassifier  
import xgboost as xgb

Разбиваем данные:

from sklearn.model\_selection import train\_test\_split  
  
X\_train,X\_test,y\_train,y\_test = train\_test\_split(X, y, test\_size = 0.2, random\_state = 30)

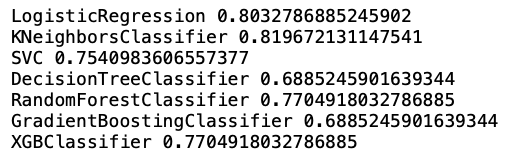
Обучаем модель:

from sklearn.ensemble import AdaBoostClassifier  
ada=AdaBoostClassifier()  
ada.fit(X\_train,y\_train)  
ada\_pre=ada.predict(X\_test)  
acc\_ada = accuracy\_score(y\_test,ada\_pre)  
acc\_ada

Предсказываем результат:

key = ['LogisticRegression','KNeighborsClassifier','SVC','DecisionTreeClassifier','RandomForestClassifier','GradientBoostingClassifier','XGBClassifier']  
value = [LogisticRegression(random\_state=9), KNeighborsClassifier(), SVC(), DecisionTreeClassifier(), RandomForestClassifier(), GradientBoostingClassifier(), xgb.XGBClassifier()]  
models = dict(zip(key,value))  
predicted =[]  
for name,algo in models.items():  
 model=algo  
 model.fit(X\_train,y\_train)  
 predict = model.predict(X\_test)  
 acc = accuracy\_score(y\_test, predict)  
 predicted.append(acc)  
 print(name,acc)

Численные результаты:



Видим, что метод K-ближайших соседей показал лучший результат. Строим матрицу ошибок для метода K-ближайших соседей. Видим, что ошибки первого и второго рода совершаются кратно реже, чем формирование корректного ответа.

cnn=KNeighborsClassifier()  
cnn.fit(X\_train,y\_train)  
cnn\_predict = cnn.predict(X\_test)  
cf\_matrix=confusion\_matrix(y\_test,cnn\_predict)  
plt.figure(figsize=(7,6))  
sns.heatmap(cf\_matrix,annot=True,fmt='d')

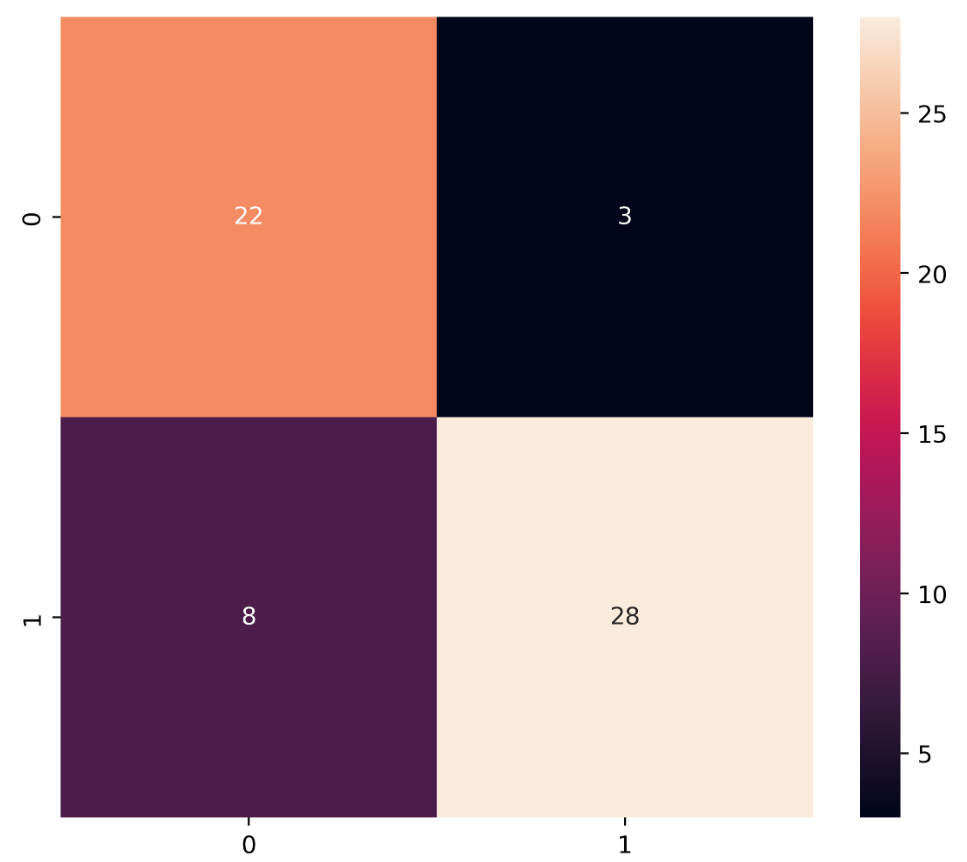
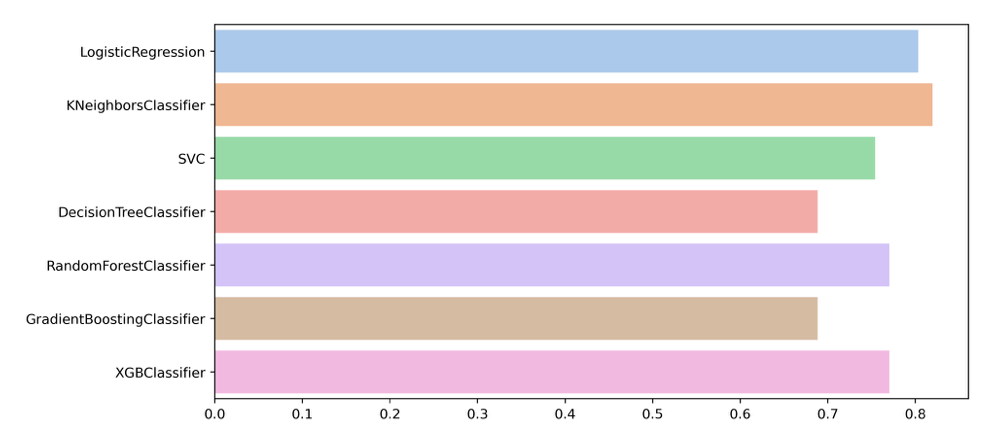


Figure 12

Отобразим результаты в виде гистограммы:

plt.figure(figsize = (10,5))  
sns.barplot(x = predicted, y = key, palette='pastel')



Наблюдение:

На рисунке выше мы видим, что модель KNeighborsClassifier (K-ближайших соседей) дает точность более 80%.

Основные результаты:

1. Высокое кровяное давление, высокий уровень холестерина и высокий сердечный ритм приводят к высокой вероятности сердечного приступа.
2. Люди в возрасте от 40 до 60 лет имеет высокий шанс сердечного приступа.
3. У мужского пола больше шансов на сердечный приступ по сравнению с женским.
4. Удалось построить модель, предсказывающую вероятность сердечного приступа с частотностью более 80%.

# Выводы

Анализ и прогнозирование возможности сердечного приступа у пациентов имеют важное значение для современной медицины. Раннее выявление риска сердечного приступа может способствовать своевременному назначению профилактических мер и лечению, что в свою очередь может помочь предотвратить серьезные осложнения, включая фатальные исходы. Точные анализы биометрических показателей пациентов, особенно при использовании компьютерных методов и методов машинного обучения, позволяют предсказывать возможность сердечного приступа с высокой степенью точности. Это открывает новые возможности для персонализированного подхода к пациентам и предоставления им необходимой медицинской помощи и советов для предупреждения сердечных заболеваний.

В процессе выполнения текущей работы нам удалось проанализировать данные с биометрическими показателями пациентов: выявить основные зависимости и закономерности. Мы составили несколько моделей для предсказания возможности сердечного приступа и оценили их эффективность. В процессе практической работы улучшили знания синтаксиса python и библиотек для анализа данных: pandas, sklearn и других.

Дальнейшее изучение текущей области может быть связано с лучшим составлением модели для предсказания: добавлением новых данных и параметров. Текущий алгоритм может быть актуален для медицинских организаций, внедряя его они смог лучше оказывать медицинские услуги и увеличить эффективность лечения для пациентов.

# Список используемой литературы

1. Логистическая регрессия: https://education.yandex.ru/handbook/data-analysis/article/logisticheskaya-regressiya
2. Метод K-ближайших соседей: <https://neerc.ifmo.ru/wiki/index.php?title=Метрический_классификатор_и_метод_ближайших_соседей>
3. Градиентный бустинг: https://education.yandex.ru/handbook/ml/article/gradientnyj-busting
4. Pyplot: <https://matplotlib.org/stable/tutorials/pyplot.html>
5. Pandas documentation: <https://pandas.pydata.org/docs/>
6. Kaggle: <https://www.kaggle.com>