BIS 634 Final Report

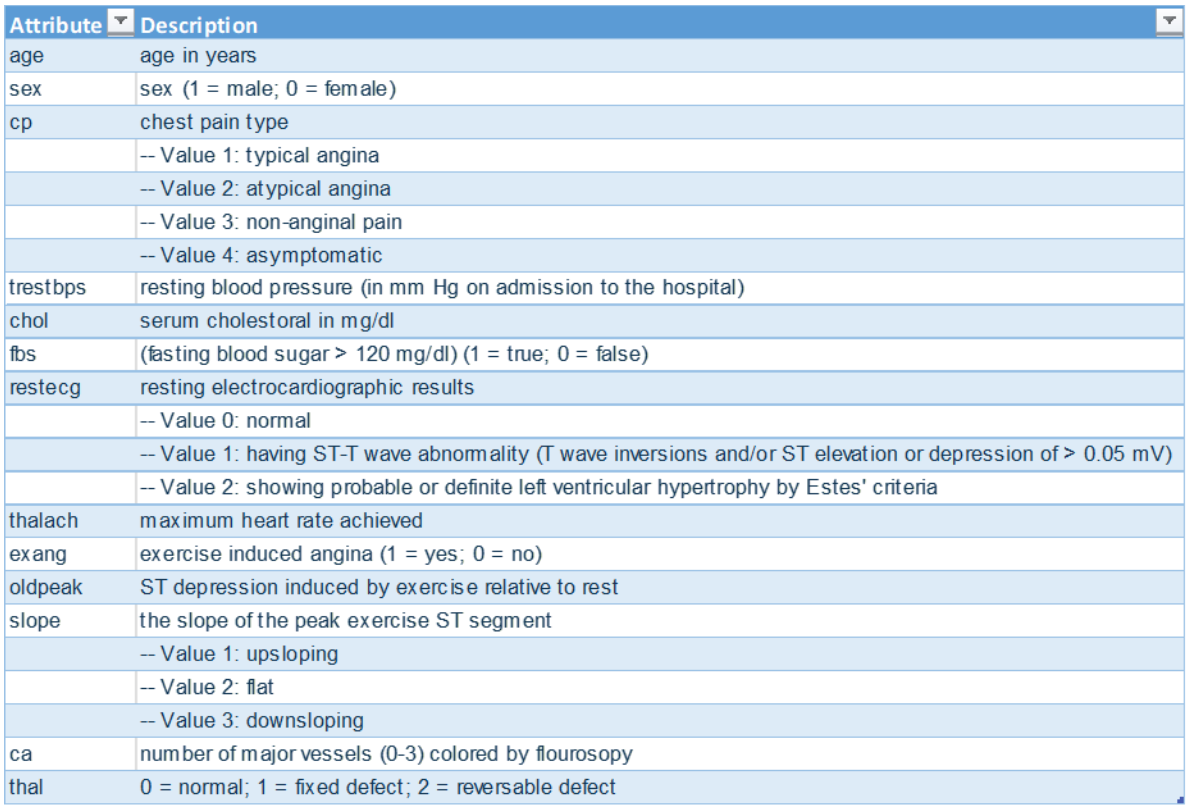
Wei Pang

GETTING STARTED

Describe dataset and why it is interesting

https://www.kaggle.com/datasets/nareshbhat/health-care-data-set-on-heart-attack-possibility

I found this dataset on Kaggle, which contains 303 samples with 13 independent variables and 1 dependent variable. The dependent variable in this dataset is the outcome of whether an individual has had a heart attack, which is recorded as a binary variable with a value of "1" indicating that the individual has had a heart attack and a value of "0" indicating that they have not. Heart attack is a major public health problem and understanding the risk factors associated with it is crucial for developing strategies to prevent and treat it. This dataset includes a wide range of potential risk factors and could be used to develop predictive models to estimate an individual's risk of heart attack based on their specific risk profile. Overall, this dataset would be useful for researchers and healthcare professionals interested in understanding and reducing the risk of heart attack.



Attribute	Description
age	age in years
sex	sex (1 = male; 0 = female)
cp	chest pain type
	-- Value 1: typical angina
	-- Value 2: atypical angina
	-- Value 3: non-anginal pain
	-- Value 4: asymptomatic
trestbps	resting blood pressure (in mm Hg on admission to the hospital)
chol	serum cholestoral in mg/dl
fbs	(fasting blood sugar > 120 mg/dl) (1 = true; 0 = false)
restecg	resting electrocardiographic results
	-- Value 0: normal
	-- Value 1: having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV)
	-- Value 2: showing probable or definite left ventricular hypertrophy by Estes' criteria
thalach	maximum heart rate achieved
exang	exercise induced angina (1 = yes; 0 = no)
oldpeak	ST depression induced by exercise relative to rest
slope	the slope of the peak exercise ST segment
	-- Value 1: upsloping
	-- Value 2: flat
	-- Value 3: downsloping
ca	number of major vessels (0-3) colored by flourosopy
thal	0 = normal; 1 = fixed defect; 2 = reversable defect

Explain how acquired it

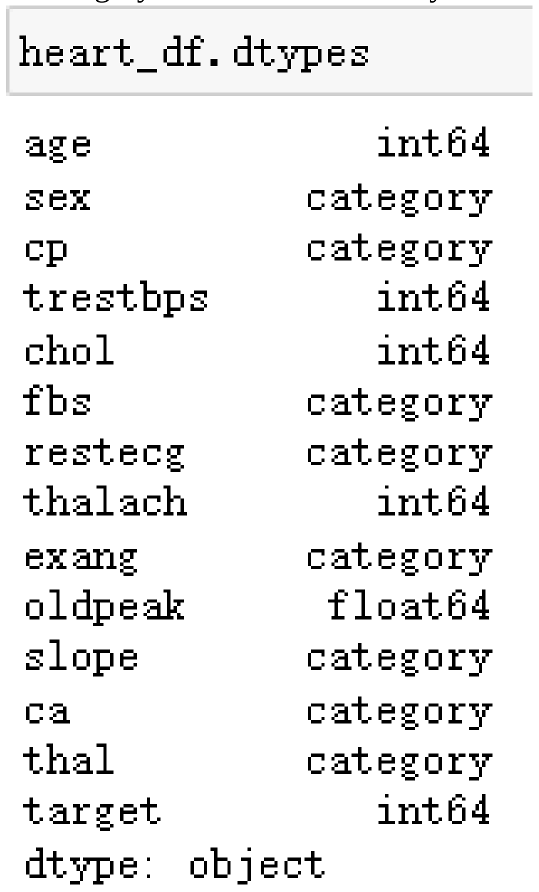
Since this dataset was found from Kaggle, it requires the use of Kaggle's library to access it via API. In addition, Kaggle account authentication is required for this to work. So even though this dataset is open source, access to the dataset requires platform authentication. So for convenience, I downloaded the dataset to the host.

Discuss FAIRness

- Findability: This dataset is open source and easy to find. And medical terminology standards are used to describe the variables.
- Accessibility: This dataset is easy to access and only requires a Kaggle account to download it. a Kaggle account is completely free. It is also possible to analyze the dataset directly on Kaggle.
- Interoperability: As a CSV document, it can easily be processed by various programming languages.
- Reusability: The usefulness of this dataset is relatively homogeneous and is limited to the prediction of the cause of heart attack and the description of patient status.

Describe data cleaning or preprocessing

The dataset is very clean with uniform data types. There are no null or residual values. No additional normalization operations are required. For ease of handling, I have marked the category variables in it for easy differentiation from continuous variables.



```
heart_df.dtypes
```

```
age            int64
sex         category
cp          category
trestbps       int64
chol           int64
fbs         category
restecg     category
thalach        int64
exang       category
oldpeak      float64
slope       category
ca          category
thal        category
target         int64
dtype: object
```

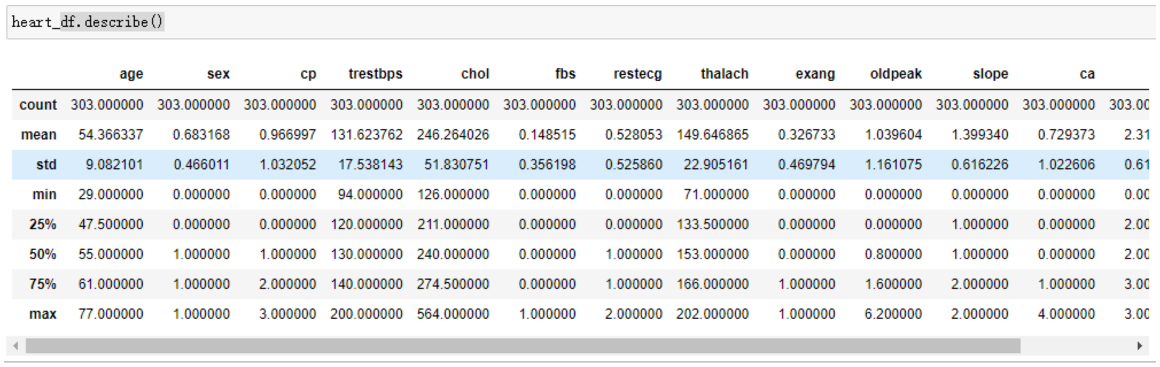
Put data in standard format if necessary

Yes, as mentioned above, I changed the data type of the category variable from int64 to category for easier processing. Data is stored in a dataframe.

ANALYSIS

Any issues with summary statistics

No issues were found from the summary statistics



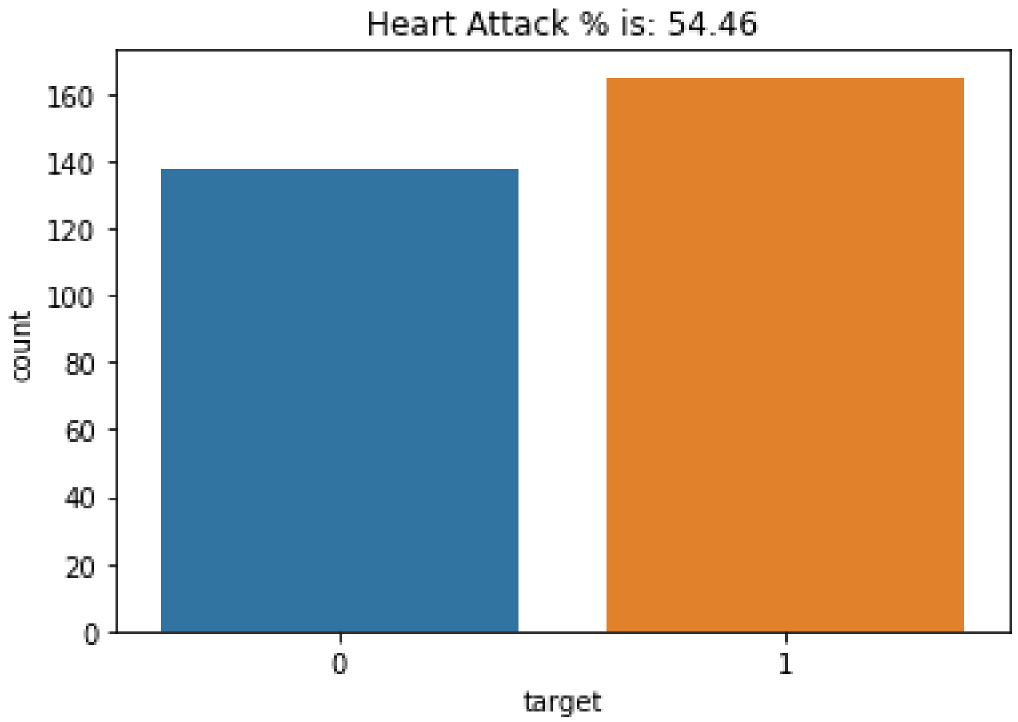
```
heart_df.describe()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.00
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.646865	0.326733	1.039604	1.399340	0.729373	2.31
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.905161	0.469794	1.161075	0.616226	1.022606	0.61
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	0.000000	0.000000	0.00
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.500000	0.000000	0.000000	1.000000	0.000000	2.00
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.000000	0.000000	0.800000	1.000000	0.000000	2.00
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.000000	1.000000	1.600000	2.000000	1.000000	3.00
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	2.000000	4.000000	3.00

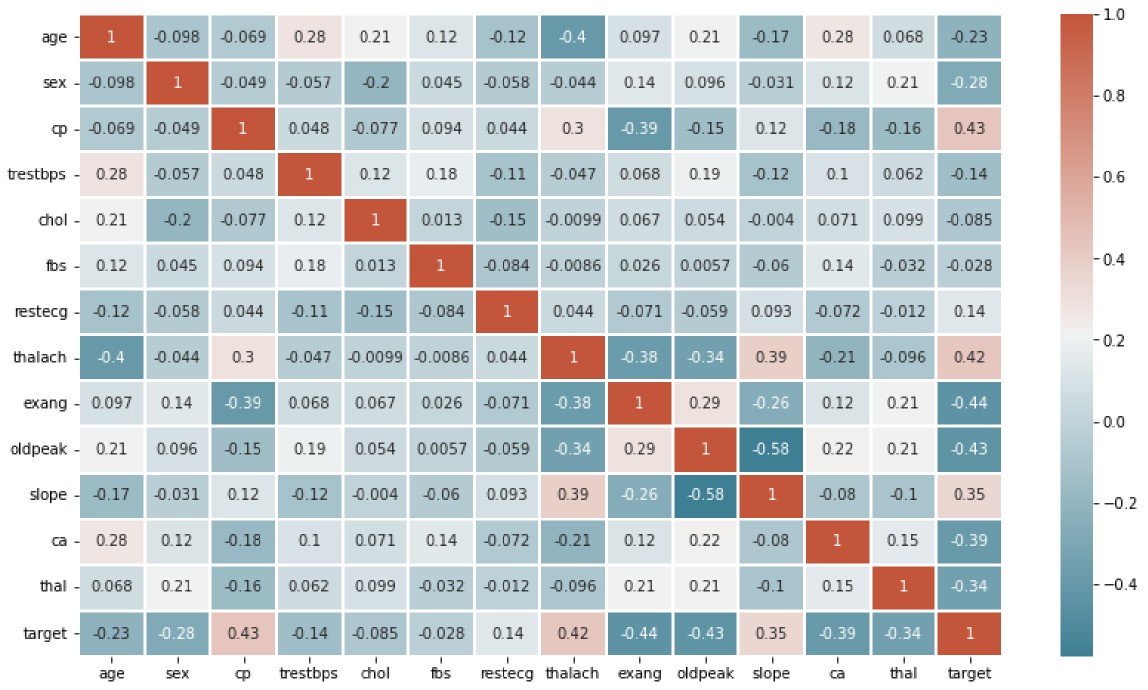
There were 303 patients, ranging in age from 29 to 77 years. The "count" showed no null values

Discuss the analyses ("interesting") you chose to run

Bar graphs were plotted for Target, and 54.46% of the sample had heart attacks.

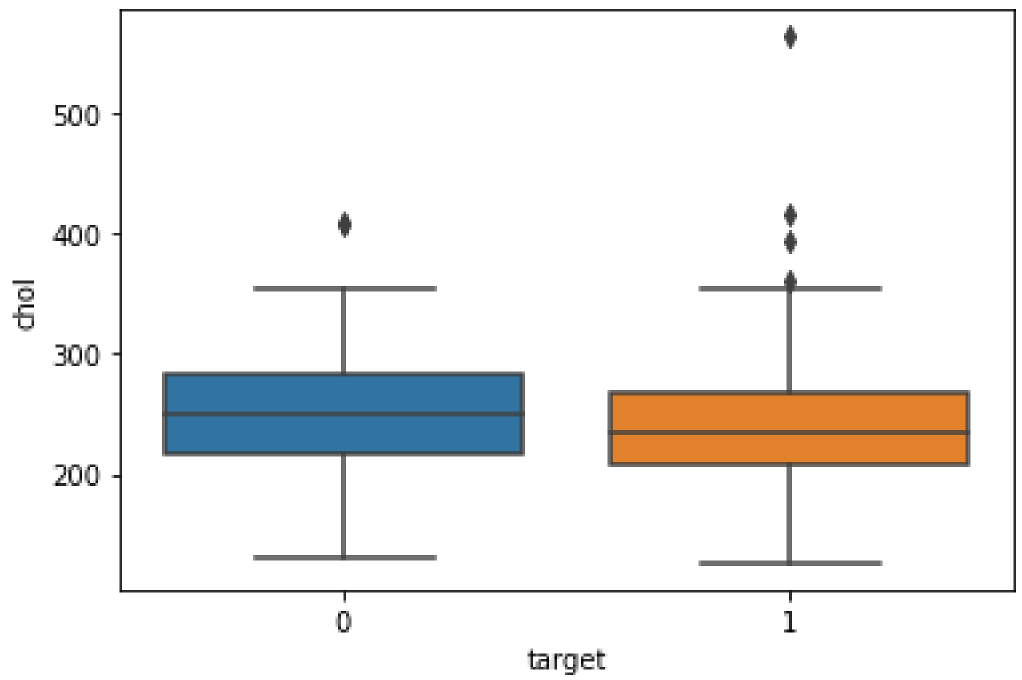


Pearson's correlation coefficient is a way to understand the correlation between variables. I use a heat map to present the results. It can be found that the variables cp, thalach and the result Target are significantly positively correlated. And exang, oldpeak and ca are significantly negatively correlated with the result Target.

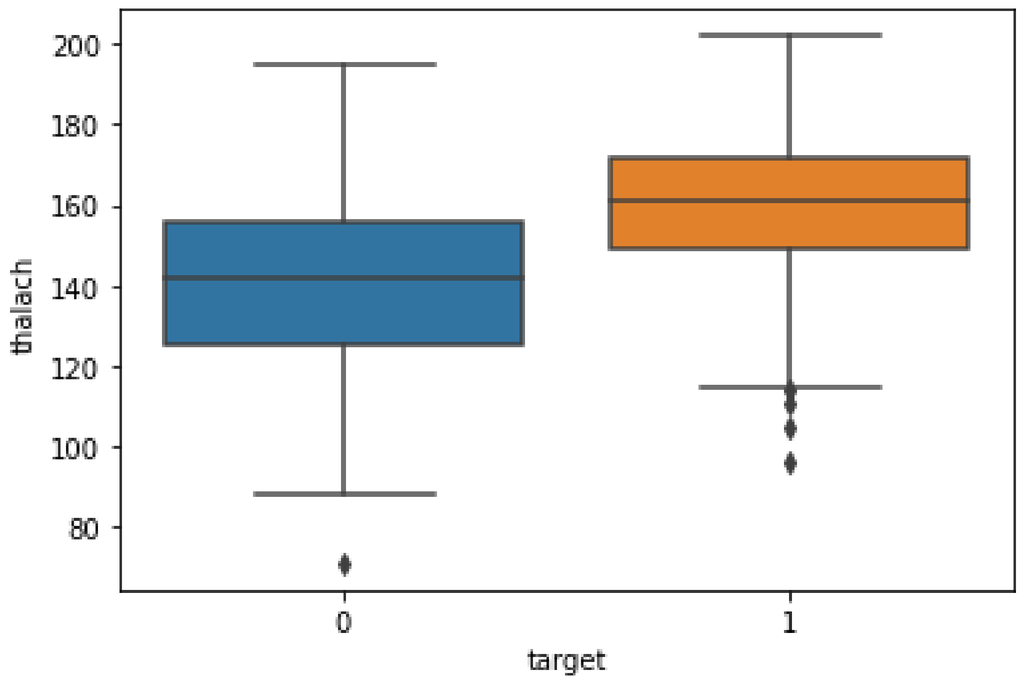


	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
age	1	-0.098	-0.069	0.28	0.21	0.12	-0.12	-0.4	0.097	0.21	-0.17	0.28	0.068	-0.23
sex	-0.098	1	-0.049	-0.057	-0.2	0.045	-0.058	-0.044	0.14	0.096	-0.031	0.12	0.21	-0.28
cp	-0.069	-0.049	1	0.048	-0.077	0.094	0.044	0.3	-0.39	-0.15	0.12	-0.18	-0.16	0.43
trestbps	0.28	-0.057	0.048	1	0.12	0.18	-0.11	-0.047	0.068	0.19	-0.12	0.1	0.062	-0.14
chol	0.21	-0.2	-0.077	0.12	1	0.013	-0.15	-0.0099	0.067	0.054	-0.004	0.071	0.099	-0.085
fbs	0.12	0.045	0.094	0.18	0.013	1	-0.084	-0.0086	0.026	0.0057	-0.06	0.14	-0.032	-0.028
restecg	-0.12	-0.058	0.044	-0.11	-0.15	-0.084	1	0.044	-0.071	-0.059	0.093	-0.072	-0.012	0.14
thalach	-0.4	-0.044	0.3	-0.047	-0.0099	-0.0086	0.044	1	-0.38	-0.34	0.39	-0.21	-0.096	0.42
exang	0.097	0.14	-0.39	0.068	0.067	0.026	-0.071	-0.38	1	0.29	-0.26	0.12	0.21	-0.44
oldpeak	0.21	0.096	-0.15	0.19	0.054	0.0057	-0.059	-0.34	0.29	1	-0.58	0.22	0.21	-0.43
slope	-0.17	-0.031	0.12	-0.12	-0.004	-0.06	0.093	0.39	-0.26	-0.58	1	-0.08	-0.1	0.35
ca	0.28	0.12	-0.18	0.1	0.071	0.14	-0.072	-0.21	0.12	0.22	-0.08	1	0.15	-0.39
thal	0.068	0.21	-0.16	0.062	0.099	-0.032	-0.012	-0.096	0.21	0.21	-0.1	0.15	1	-0.34
target	-0.23	-0.28	0.43	-0.14	-0.085	-0.028	0.14	0.42	-0.44	-0.43	0.35	-0.39	-0.34	1

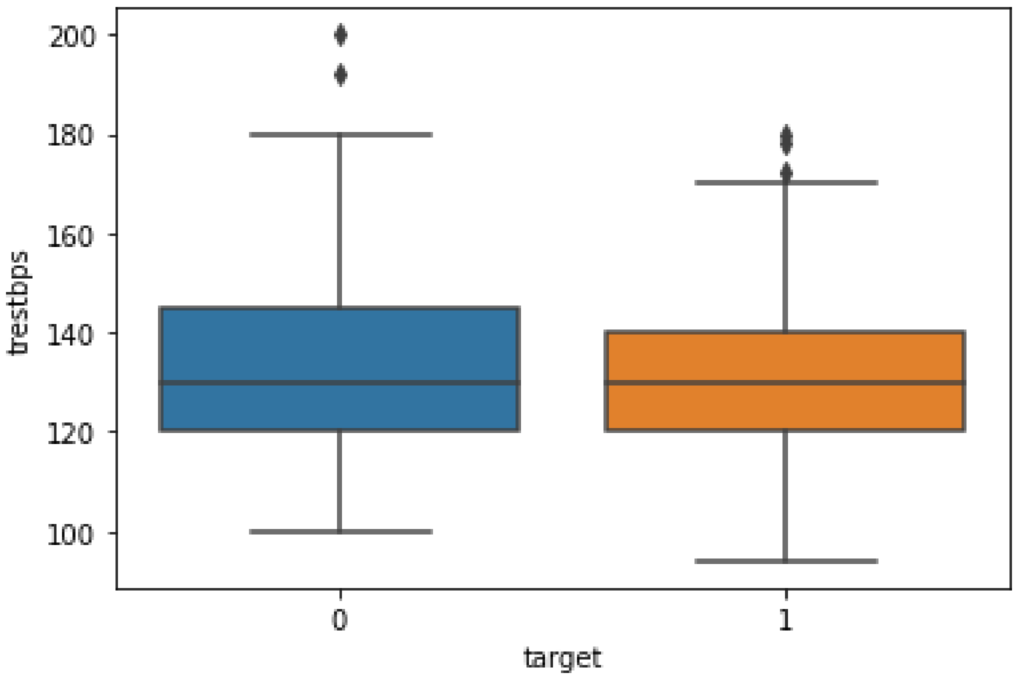
I performed a box plot analysis for each of the three continuous variables thalach, chol, and trestbps. I found no significant differences in cholesterol levels and blood pressure between patients with heart attacks and healthy individuals, but their maximum heart rate was significantly higher than that of healthy individuals.



500
400
300
200
chol
0
1
target

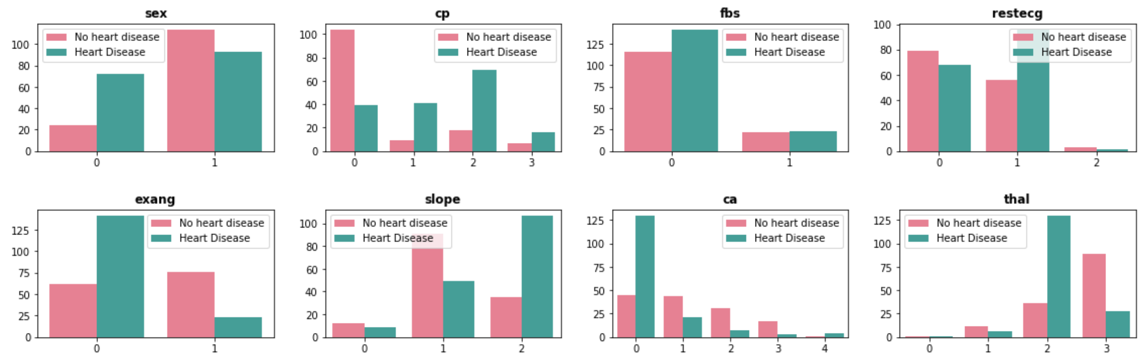


200
180
160
140
120
100
80
thalach
0
1
target



A

Visual representation of the bar chart was performed for each of the eight category variables. Men were found to be more likely to suffer from heart attacks. Heart attack and the history of diagnosis of myocardial ischemia was correlated. Patients with diseases that cause irreversible myocardial damage (e.g., thalassemia) were more likely to have heart attack. Exercise and heart attack were negatively correlated. One of the valid diagnostic markers of heart attack is the electrocardiographic index.



- Why these questions?

What causes heart attacks and what can be done to avoid them? What is the pattern of heart attack?

- What were the results?

Heart attack is a common condition. It is not clear why men in the sample were more likely to develop it. Heart attacks are associated with myocardial ischemia and a history of disease that damages the heart muscle. A healthy lifestyle and timely diagnosis and treatment can help avoid heart attacks. Studying the relationship between variables can identify ways to avoid heart attacks, which is relevant for the prevention of a disease with high morbidity.

- Any surprises?

Many people experience angina after exercise, even young people like me. At one time I would have thought that over-exercise had caused damage to my heart. But the data show a significant negative correlation between post-exercise angina and heart attack. After further research I found that post-exercise angina is normal and increases the oxygen supply to the heart, thus preventing heart attacks.

Men are more likely to have heart attacks. Probably because men are also more likely to engage in behaviors that increase the risk of heart disease, such as smoking, drinking alcohol in excess, and not exercising regularly. These behaviors can damage the heart and blood vessels, increasing the risk of a heart attack.

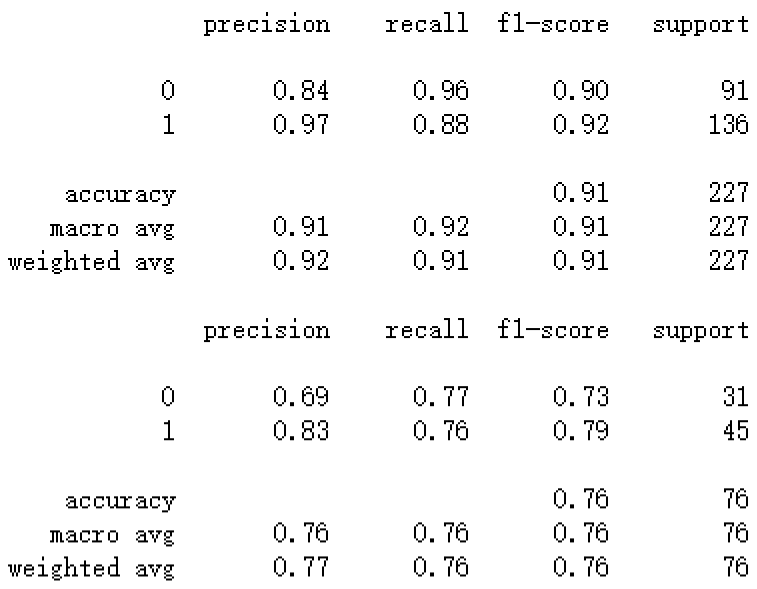
- validation of analyses?

I applied the insights derived from these analyses to a subsequent machine learning model and used it to analyze the probability of heart attack. The model achieved a 91% correct rate. Although my conclusions were not necessarily correct, there was a greater likelihood of successfully predicting the outcome.

I also reviewed several peer reviewed papers related to heart attack and found that the above statistical analysis results are indeed the main cause of heart attack.

- do more than just summary statistics

I performed machine learning on the dataset using the Free Forest classifier and GridSearchCV. First I used StratifiedShuffleSplit to split the dataset into two significantly different clusters. Then the model was trained. The accuracy for the two clusters were 0.91 and 0.76.



```
              precision    recall  f1-score   support

           0       0.84      0.96      0.90        91
           1       0.97      0.88      0.92       136

    accuracy                           0.91       227
   macro avg       0.91      0.92      0.91       227
weighted avg       0.92      0.91      0.91       227
              precision    recall  f1-score   support

           0       0.69      0.77      0.73        31
           1       0.83      0.76      0.79        45

    accuracy                           0.76        76
   macro avg       0.76      0.76      0.76        76
weighted avg       0.77      0.76      0.76        76
```

- two analyses that generate graphs

I generated a variety of graphical results such as heat maps, box plots, bar graphs, etc.

- at least one analysis that takes a parameter

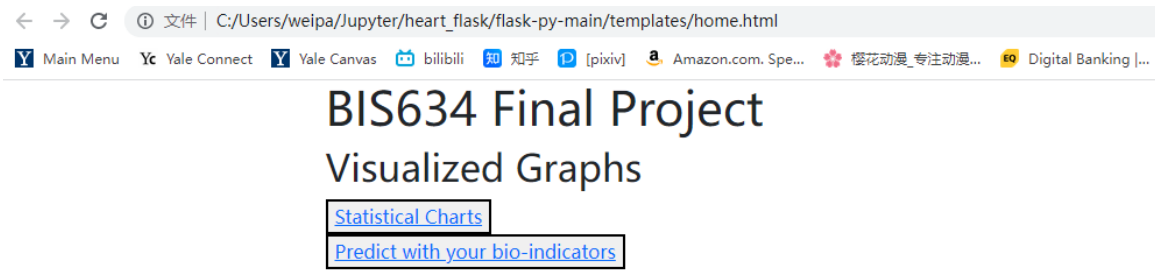
Machine learning trained predictive models use a variety of parameters. I used GridSearchCV to perform parameter tuning.

WEB BACKEND AND FRONTEND (?/25 POINTS TOTAL)

Describe your server API and the web front-end

My web page consists of a homepage, a page presenting statistical charts, and an interactive page.

Has a web interface



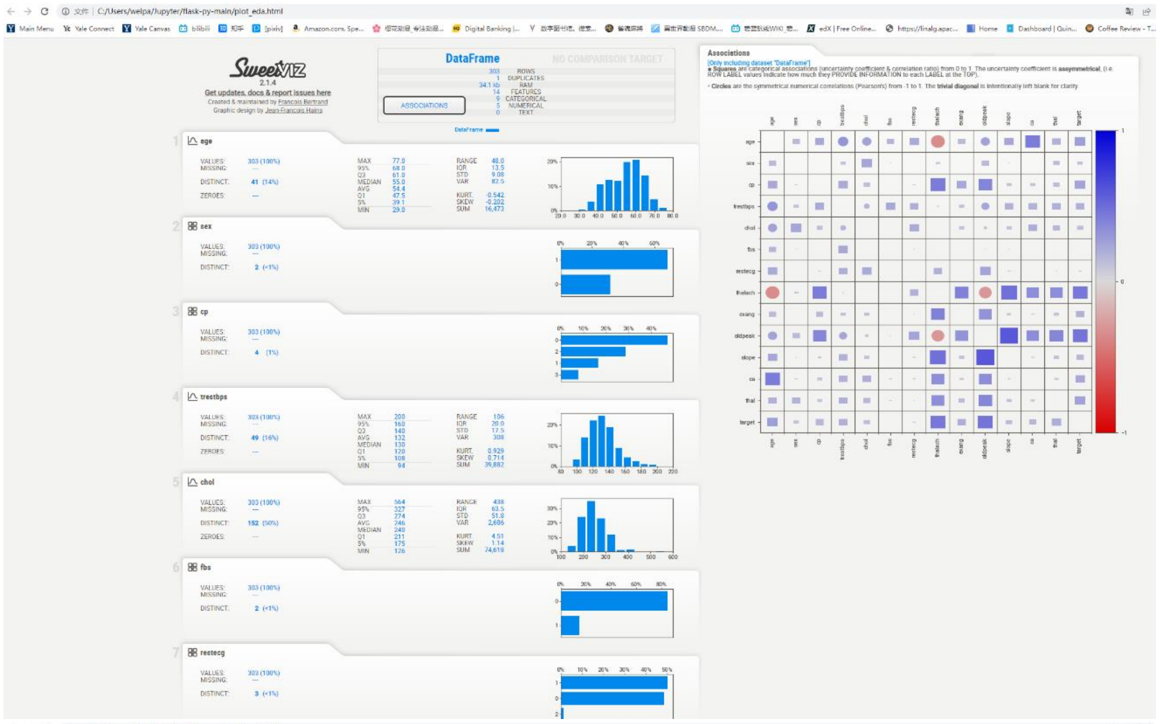
C:/Users/weipa/Jupyter/heart_flask/flask-py-main/templates/home.html

BIS634 Final Project

Visualized Graphs

[Statistical Charts]

[Predict with your bio-indicators]



C:/Users/weipa/Jupyter/flask-py-main/plot_eda.html

Sweetviz 2.1.4
Get updates, docs & report issues here
Created & maintained by Francois Bertrand
Graphic design by Jean-Francois Hains

ASSOCIATIONS

```
DataFrame                NO COMPARISON TARGET
      303   ROWS
        1   DUPLICATES
  34.1 kb   RAM
       14   FEATURES
        9   CATEGORICAL
        5   NUMERICAL
        0   TEXT
```

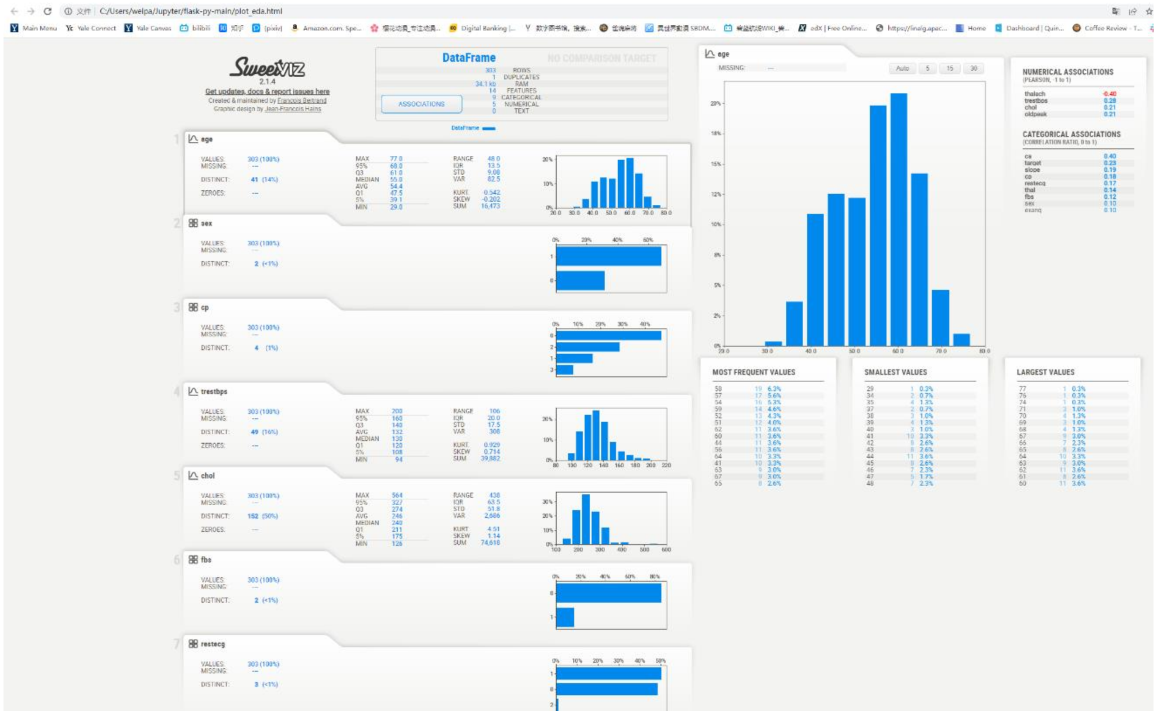
```
1 age
VALUES:    303 (100%)     MAX      77.0     RANGE   48.0
MISSING:   ---            95%      68.0     IQR     13.5
                          Q3       61.0     STD     9.08
DISTINCT:  41 (14%)       MEDIAN   55.0     VAR     82.5
                          AVG      54.4
ZEROES:    ---            Q1       47.5     KURT.   -0.542
                          5%       39.1     SKEW    -0.202
                          MIN      29.0     SUM     16,473

2 sex
VALUES:    303 (100%)
MISSING:   ---
DISTINCT:  2 (<1%)

3 cp
VALUES:    303 (100%)
MISSING:   ---
DISTINCT:  4 (1%)

4 trestbps
VALUES:    303 (100%)     MAX      200      RANGE   106
MISSING:   ---            95%      160      IQR     20.0
                          Q3       140      STD     17.5
DISTINCT:  49 (16%)       AVG      132      VAR     308
                          MEDIAN   130
ZEROES:    ---            Q1       120      KURT.   0.929
                          5%       108      SKEW    0.714
                          MIN      94       SUM     39,882

5 chol
VALUES:    303 (100%)     MAX      564      RANGE   438
MISSING:   ---            95%      327      IQR     63.5
                          Q3       274      STD     51.8
DISTINCT:  152 (50%)      AVG      246      VAR     2,686
                          MEDIAN   240
ZEROES:    ---            Q1       211      KURT.   4.51
                          5%       175      SKEW    1.14
                          MIN      126      SUM     74,618

6 fbs
VALUES:    303 (100%)
MISSING:   ---
DISTINCT:  2 (<1%)

7 restecg
VALUES:    303 (100%)
MISSING:   ---
DISTINCT:  3 (<1%)
```

Associations

[Only including dataset "DataFrame"]

■ **Squares** are categorical associations (uncertainty coefficient & correlation ratio) from 0 to 1. The uncertainty coefficient is **assymmetrical**, (i.e. ROW LABEL values indicate how much they PROVIDE INFORMATION to each LABEL at the TOP).

• **Circles** are the symmetrical numerical correlations (Pearson's) from -1 to 1. The **trivial diagonal** is intentionally left blank for clarity.



C:/Users/weipa/Jupyter/flask-py-main/plot_eda.html

Sweetviz 2.1.4
Get updates, docs & report issues here
Created & maintained by Francois Bertrand
Graphic design by Jean-Francois Hains

ASSOCIATIONS

DataFrame NO COMPARISON TARGET
303 ROWS, 1 DUPLICATES, 34.1 kb RAM, 14 FEATURES, 9 CATEGORICAL, 5 NUMERICAL, 0 TEXT

1 age, 2 sex, 3 cp, 4 trestbps, 5 chol, 6 fbs, 7 restecg

age
MISSING: ---
Auto 5 15 30

NUMERICAL ASSOCIATIONS
(PEARSON, -1 to 1)

thalach	-0.40
trestbps	0.28
chol	0.21
oldpeak	0.21

CATEGORICAL ASSOCIATIONS
(CORRELATION RATIO, 0 to 1)

ca	0.40
target	0.23
slope	0.19
cp	0.18
restecg	0.17
thal	0.14
fbs	0.12
sex	0.10
exang	0.10

MOST FREQUENT VALUES

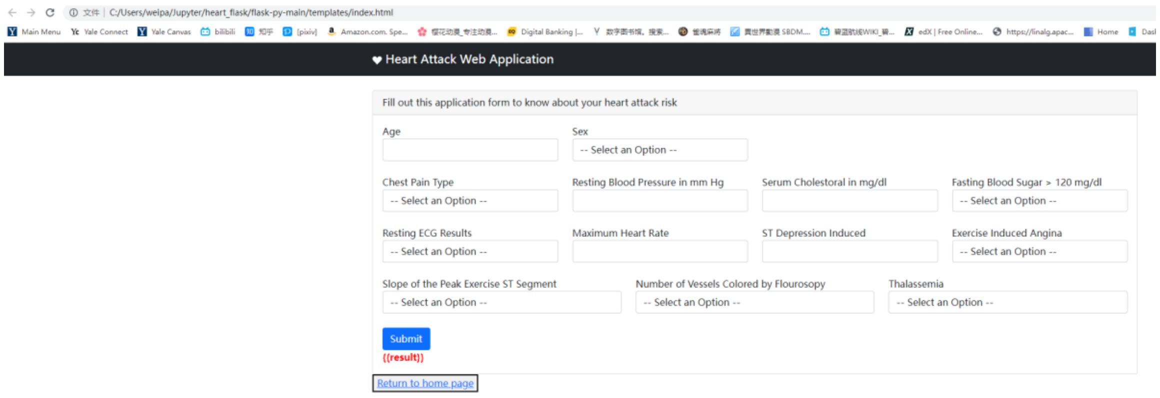
58	19	6.3%
57	17	5.6%
54	16	5.3%
59	14	4.6%
52	13	4.3%
51	12	4.0%
62	11	3.6%
60	11	3.6%
44	11	3.6%
56	11	3.6%
64	10	3.3%
41	10	3.3%
63	9	3.0%
67	9	3.0%
65	8	2.6%

SMALLEST VALUES

29	1	0.3%
34	2	0.7%
35	4	1.3%
37	2	0.7%
38	3	1.0%
39	4	1.3%
40	3	1.0%
41	10	3.3%
42	8	2.6%
43	8	2.6%
44	11	3.6%
45	8	2.6%
46	7	2.3%
47	5	1.7%
48	7	2.3%

LARGEST VALUES

77	1	0.3%
76	1	0.3%
74	1	0.3%
71	3	1.0%
70	4	1.3%
69	3	1.0%
68	4	1.3%
67	9	3.0%
66	7	2.3%
65	8	2.6%
64	10	3.3%
63	9	3.0%
62	11	3.6%
61	8	2.6%
60	11	3.6%


C:/Users/weipa/Jupyter/heart_flask/flask-py-main/templates/index.html

Heart Attack Web Application

Fill out this application form to know about your heart attack risk

Age	Sex: -- Select an Option --		
Chest Pain Type: -- Select an Option --	Resting Blood Pressure in mm Hg	Serum Cholestoral in mg/dl	Fasting Blood Sugar > 120 mg/dl: -- Select an Option --
Resting ECG Results: -- Select an Option --	Maximum Heart Rate	ST Depression Induced	Exercise Induced Angina: -- Select an Option --
Slope of the Peak Exercise ST Segment: -- Select an Option --	Number of Vessels Colored by Flourosopy: -- Select an Option --	Thalassemia: -- Select an Option --	

Submit

{{result}}

Return to home page

interface

Users can choose to access the graphical interface or access the interactive interface for parameter input and result query. The user must enter 13 parameters in the interactive interface. For different parameter inputs, the trained model will give different answers to assess the risk of heart attack.

Routing is established via flask to fetch the graphs and trained prediction models from local. The results are presented as a graph

Exploratory Data Analysis

In [7]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Get the dataset into a pandas dataframe for easier processing

In [8]:
```
heart_df = pd.read_csv('heart.csv')
heart_df.head(10) #gives us the first 10 rows of the dataset
```

Out[8]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1
5	57	1	0	140	192	0	1	148	0	0.4	1	0	1	1
6	56	0	1	140	294	0	0	153	0	1.3	1	0	2	1
7	44	1	1	120	263	0	1	173	0	0.0	2	0	3	1
8	52	1	2	172	199	1	1	162	0	0.5	2	0	3	1
9	57	1	2	150	168	0	1	174	0	1.6	2	0	2	1

Most feature names(eg: cp, trestbps etc.) are abbreviated. So we'll very briefly try to understand what they mean (only the ones are not obvious):

['sex', 'cp', 'restecg','exang', 'slope', 'ca', 'thal']

Target: 0 = less likely to have a heart attack 1 = more likely to have a heart attack

Descriptive analysis

In [9]:
```
heart_df.describe()
```

Out[9]:

	age	sex	cp	trestbps	chol	fbs	restecg	tha
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.00
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.64
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.90
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.00
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.50
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.00
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.00
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.00

There were 303 patients, ranging in age from 29 to 77 years. The "count" showed no null values

In [10]:
```
heart_df.isnull().sum()
```

Out[10]:
```
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
target      0
dtype: int64
```

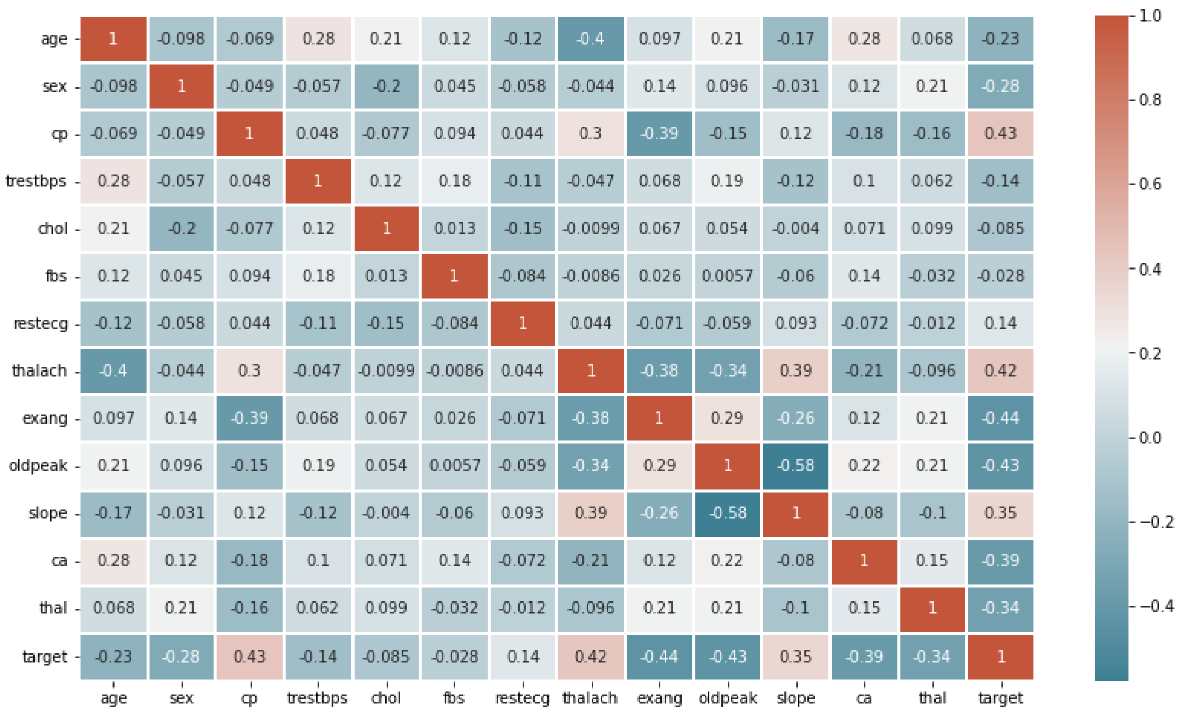
There are no null values.

In [11]:
```
heart_df.info()
```

```
<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   trestbps  303 non-null    int64  
 4   chol      303 non-null    int64  
 5   fbs       303 non-null    int64  
 6   restecg   303 non-null    int64  
 7   thalach   303 non-null    int64  
 8   exang     303 non-null    int64  
 9   oldpeak   303 non-null    float64
 10  slope     303 non-null    int64  
 11  ca        303 non-null    int64  
 12  thal      303 non-null    int64  
 13  target    303 non-null    int64  
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

`info()` method gives us the data types of each feature. int means integer, and float means floating points. But, as we explored, some of these are better represented as categorical variables, because the categories have a meaning, and an 'order'. For example, cp or chest pain type has 4 values: '1' says typical angina. Angina refers to the chest pain that occurs due to a blockage in heart vessels. So '1' is most indicative of heart disease, and '4' is least indicative.

In [12]:
```
pearson = heart_df.corr()

cmap = sns.diverging_palette(220, 20, as_cmap=True)
_, ax = plt.subplots(figsize=(14,8))
ax = sns.heatmap (pearson, annot=True, linewidth=2, cmap=cmap)
```



In [13]:
```
cat_features = ['sex', 'cp','fbs', 'restecg','exang', 'slope', 'ca', 'thal' ]
num_features = ['age', 'trestbps', 'chol', 'thalach', 'oldpeak']
label = 'target'
```

In [14]:
```
heart_df[cat_features] = heart_df[cat_features].astype('category')
```

In [15]:
```
heart_df.dtypes
```

Out[15]:
```
age            int64
sex         category
cp          category
trestbps       int64
chol           int64
fbs         category
restecg     category
thalach        int64
exang       category
oldpeak      float64
slope       category
ca          category
thal        category
target         int64
dtype: object
```

In [16]:
```
heart_df.cp.value_counts()
```

Out[16]:
```
0    143
2     87
1     50
3     23
Name: cp, dtype: int64
```

In [17]:
```
heart_df.exang.unique()
```

Out[17]:
```
[0, 1]
Categories (2, int64): [0, 1]
```

In [18]:
```
datatype = heart_df.exang.dtype

print(datatype)
```

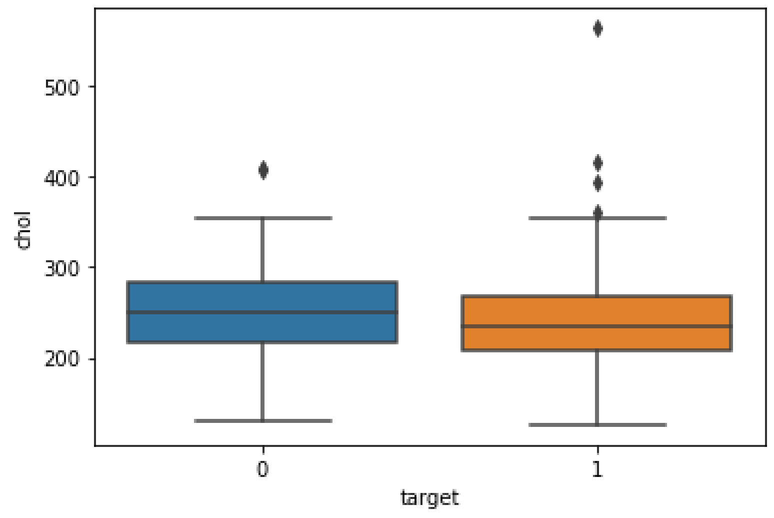
```
category
```

In [19]:
```
heart_df.groupby(heart_df.target).chol.mean()
```

Out[19]:
```
target
0    251.086957
1    242.230303
Name: chol, dtype: float64
```

In [47]:
```
sns.boxplot(x='target', y='chol', data=heart_df)
```

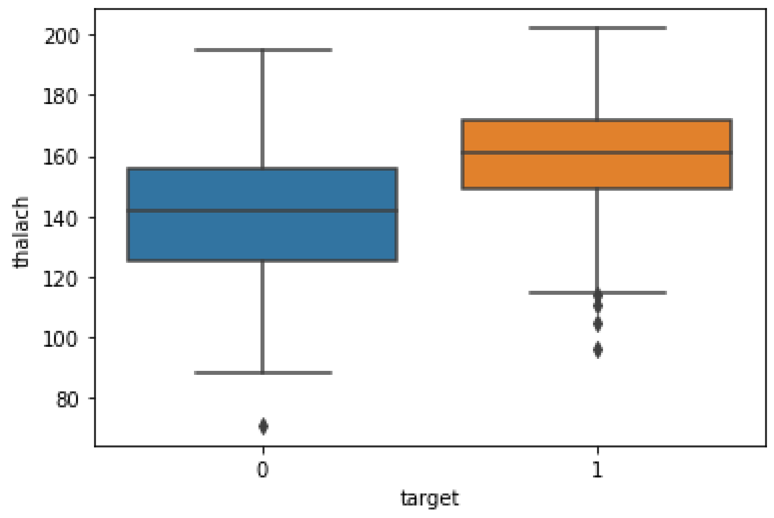
Out[47]:
```
<AxesSubplot:xlabel='target', ylabel='chol'>
```



Despite having a person with a cholesterol level of above 500, heart disease group shows no significant difference in the mean cholesteroal levels compared to no heart disase group. Outliers like this 500+ value can sway the mean towards a higher value.

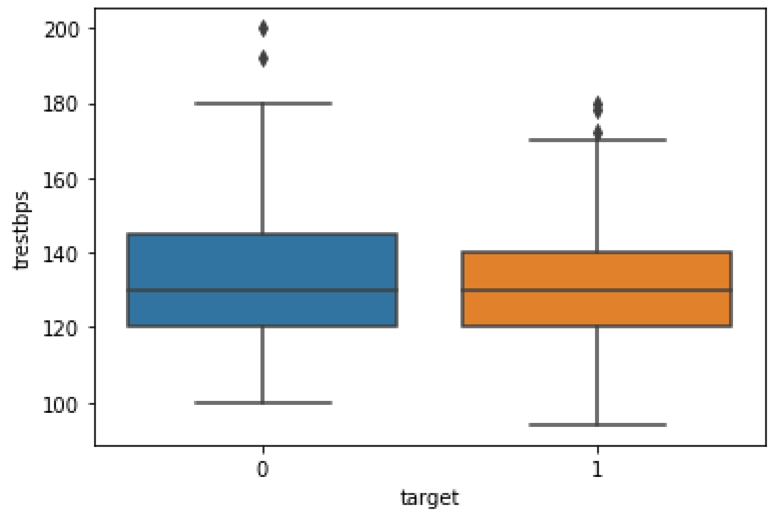
```
In [48]: sns.boxplot(x='target', y='thalach', data=heart_df)
```

```
Out[48]: <AxesSubplot:xlabel='target', ylabel='thalach'>
```



```
In [49]: sns.boxplot(x='target', y='trestbps', data=heart_df)
```

```
Out[49]: <AxesSubplot:xlabel='target', ylabel='trestbps'>
```

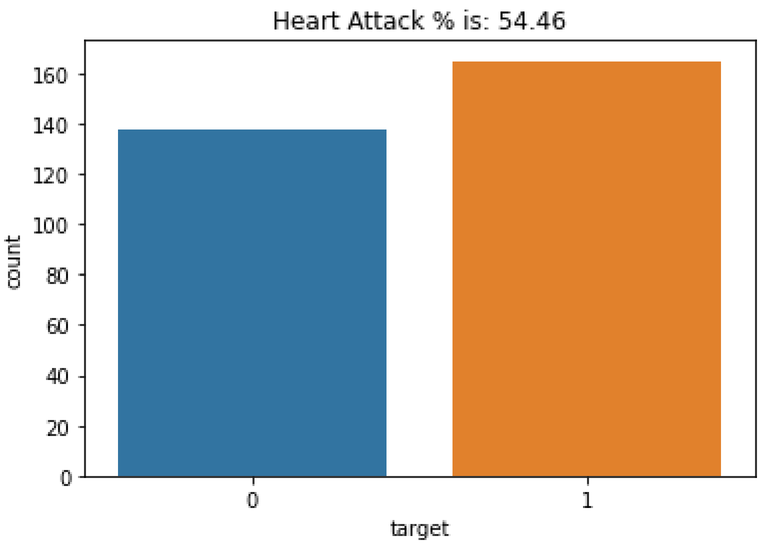


In [21]:
```
heart_pt_percentage = len(heart_df.loc[heart_df['target']==1])/len(heart_df)*100
heart_pt_percentage
```

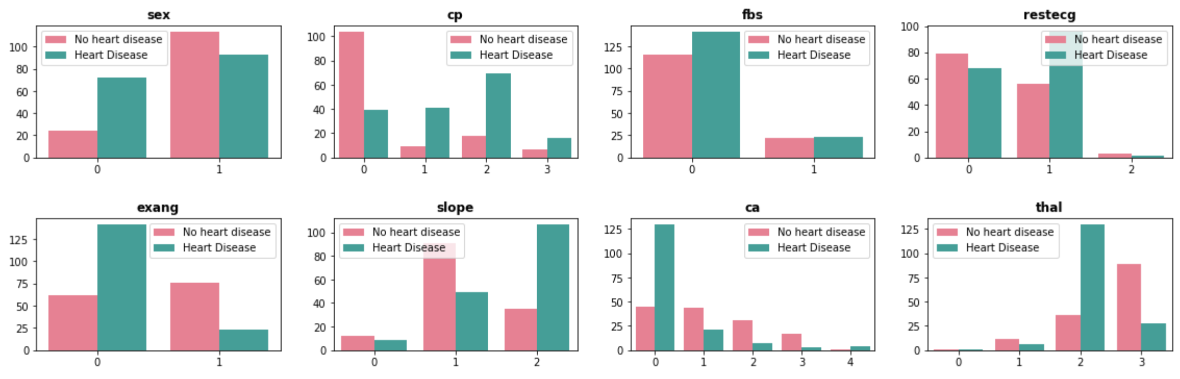
Out[21]:
```
54.45544554455446
```

In [50]:
```
#plotting target
ax = sns.countplot(x='target', data=heart_df)
ax.set_title(f'Heart Attack % is:{heart_pt_percentage: .2f}')
```

Out[50]:
```
Text(0.5, 1.0, 'Heart Attack % is: 54.46')
```



In [24]:
```
plt.figure(figsize = (16,10))
for i,col in enumerate(cat_features):
    plt.subplot(4,4,i+1)
    sns.countplot(data = heart_df , x = col, palette = "husl", hue = 'target' )
    plt.title(col,weight = 'bold', color = 'black')
    plt.legend(['No heart disease','Heart Disease'])
    plt.ylabel(" ")
    plt.xlabel(" ")
    plt.tight_layout()
```



Based on the above exploration, the variables thal and restecg were rearranged in "order" because in thal, fixed defects have the greatest weight in predicting heart disease, while ST changes in restecg have the greatest weight in a similar manner.

In [25]:
```
#value counts before reordering
heart_df.restecg.value_counts()
```

```
Out[25]: 1    152
         0    147
         2      4
         Name: restecg, dtype: int64
```

```
In [26]: heart_df['restecg'] = heart_df['restecg'].replace({0:0, 1:2, 2:1})
```

```
In [27]: #check after reordering:
         heart_df.restecg.value_counts()
```

```
Out[27]: 1    156
         0    147
         Name: restecg, dtype: int64
```

```
In [28]: heart_df.thal.value_counts()
```

```
Out[28]: 2    166
         3    117
         1     18
         0      2
         Name: thal, dtype: int64
```

```
In [30]: heart_df['thal'] = heart_df['thal'].replace({1:1, 2:3, 3:2})
```

```
In [31]: heart_df.thal
```

```
Out[31]: 0      1
         1      2
         2      2
         3      2
         4      2
               ..
         298    2
         299    2
         300    2
         301    2
         302    2
         Name: thal, Length: 303, dtype: category
         Categories (3, int64): [0, 1, 2]
```

```
In [32]: from sklearn.preprocessing import OrdinalEncoder
```

```
In [33]: from sklearn.model_selection import StratifiedShuffleSplit
         split = StratifiedShuffleSplit(n_splits=1, test_size=0.25, random_state=42)
         for train_index, test_index in split.split(heart_df, heart_df["target"]):
             strat_train = heart_df.loc[train_index]
             strat_test = heart_df.loc[test_index]
```

```
In [34]: X_train = strat_train.drop('target', axis=1)
         y_train = strat_train['target']
         x_test = strat_test.drop('target', axis=1)
         y_test = strat_test['target']
```

```
In [35]: X_train
```

Out[35]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
66	51	1	2	100	222	0	1	143	1	1.2	1	0	2
260	66	0	0	178	228	1	1	165	1	1.0	1	2	2
289	55	0	0	128	205	0	1	130	1	2.0	1	1	2
237	60	1	0	140	293	0	0	170	0	1.2	1	2	2
144	76	0	2	140	197	0	1	116	0	1.1	1	0	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...
170	56	1	2	130	256	1	0	142	1	0.6	1	1	1
60	71	0	2	110	265	1	0	130	0	0.0	2	1	2
128	52	0	2	136	196	0	0	169	0	0.1	1	0	2
53	44	0	2	108	141	0	1	175	0	0.6	1	0	2
23	61	1	2	150	243	1	1	137	1	1.0	1	0	2

227 rows × 13 columns

In [36]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [37]:
```
def rf(xs, y, n_estimators=40,
max_features=0.5, min_samples_leaf=5, **kwargs):
    return RandomForestClassifier(n_jobs=-1, n_estimators=n_estimators,
    max_features=max_features,
    min_samples_leaf=min_samples_leaf, oob_score=True).fit(xs, y)
```

In [38]:
```
X_train.shape, y_train.shape
```

Out[38]:
```
((227, 13), (227,))
```

In [39]:
```
rf_model = rf(X_train, y_train)
```

In [40]:
```
#predicting on validation set and evaluating
from sklearn.metrics import classification_report
train_preds = rf_model.predict(X_train)
print(classification_report(train_preds, y_train))
y_preds = rf_model.predict(x_test)
print(classification_report(y_preds,y_test))
```

```
              precision    recall  f1-score   support

           0       0.87      0.94      0.90        96
           1       0.95      0.90      0.93       131

    accuracy                           0.92       227
   macro avg       0.91      0.92      0.92       227
weighted avg       0.92      0.92      0.92       227

              precision    recall  f1-score   support

           0       0.71      0.83      0.77        30
           1       0.88      0.78      0.83        46

    accuracy                           0.80        76
   macro avg       0.80      0.81      0.80        76
weighted avg       0.81      0.80      0.80        76
```

In [41]:
```
from sklearn.model_selection import GridSearchCV
```

In [42]:
```
param_grid = [
{'n_estimators': [100, 150, 200, 250, 300, 350, 400, 450, 500, 600], 'max_features':
 'n_jobs':[-1]},
{'n_estimators': [100, 150, 200, 250, 300, 350, 400, 450, 500], 'max_features': [0.5
 'n_jobs':[-1]},
   ]

rf = RandomForestClassifier()
grid_search = GridSearchCV(rf, param_grid, cv=5,
scoring='accuracy',
return_train_score=True)
grid_search.fit(X_train, y_train)
```

Out[42]:
```
GridSearchCV(cv=5, estimator=RandomForestClassifier(),
             param_grid=[{'max_features': [0.5, 'sqrt', 'log2'],
                          'min_samples_leaf': [4],
                          'n_estimators': [100, 150, 200, 250, 300, 350, 400,
                                           450, 500, 600],
                          'n_jobs': [-1]},
                         {'max_features': [0.5, 'sqrt', 'log2'],
                          'n_estimators': [100, 150, 200, 250, 300, 350, 400,
                                           450, 500],
                          'n_jobs': [-1]}],
             return_train_score=True, scoring='accuracy')
```

In [43]:
```
grid_search.best_params_
```

Out[43]:
```
{'max_features': 'log2', 'n_estimators': 100, 'n_jobs': -1}
```

In [44]:
```
rf_model2 = RandomForestClassifier( oob_score=True, n_estimators=200, max_features=
rf_model2.fit(X_train, y_train)
```

Out[44]:
```
RandomForestClassifier(max_depth=5, max_features='sqrt', min_samples_leaf=4,
                       n_estimators=200, n_jobs=-1, oob_score=True)
```

In [45]:
```
from sklearn.metrics import classification_report
train_preds = rf_model2.predict(X_train)
print(classification_report(train_preds, y_train))
y_preds = rf_model2.predict(x_test)
print(classification_report(y_preds,y_test))
```

```
              precision    recall  f1-score   support

           0       0.84      0.96      0.90        91
           1       0.97      0.88      0.92       136

    accuracy                           0.91       227
   macro avg       0.91      0.92      0.91       227
weighted avg       0.92      0.91      0.91       227

              precision    recall  f1-score   support

           0       0.69      0.77      0.73        31
           1       0.83      0.76      0.79        45

    accuracy                           0.76        76
   macro avg       0.76      0.76      0.76        76
weighted avg       0.77      0.76      0.76        76
```

```python
# -*- coding: utf-8 -*-
import numpy as np
import pandas as pd
import pickle
from flask import Flask, request, render_template
import sweetviz as sv
# Load ML model
model = pickle.load(open('model_heart.pkl', 'rb'))
# load heart data
heart_df = pd.read_csv('heart.csv')
# Create application
app = Flask(__name__)

# Bind home function to URL
@app.route('/')
def home():
      return render_template('index.html')

my_report = sv.analyze(heart_df)

my_report.show_html(open_browser=True,
                        filepath='plot_eda.html')
# Bind predict function to URL
@app.route('/predict', methods =['POST'])
def predict():

      # Put all form entries values in a list
      features = [float(i) for i in request.form.values()]
      # Convert features to array
      array_features = [np.array(features)]
      # Predict features
      prediction = model.predict(array_features)

      output = prediction

      # Check the output values and retrive the result with html tag based on the value
      if output == 1:
            return render_template('index.html',
                                              result = 'Heart disease - Unlikely. No need to worry!')
      else:
            return render_template('index.html',
                                              result = 'Heart disease - Likely.Please go and see a doctor!')

if __name__ == '__main__':
#Run the application
      app.run()

```