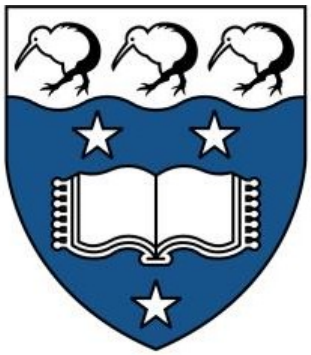
Jingyuan Zhang
INFOSYS 722
28 May 2020

Analysis of Heart Disease

Iteration 4-DBAS

GitHub: https://github.com/jzha505/BDAS



THE UNIVERSITY OF AUCKLAND

NEW ZEALAND

1.Situation understanding

We know many people died of heart disease all over the world , but do we actually know what the heart disease is? Heart disease is also known as cardiovascular diseases, which let to a group of heart disorders, blood vessel disorders, including conditions: coronary heart disease, strokes, rheumatic heart disease.
What's the reality we now face? That's see those facts from WHO:
1) Heart disease are the main cause of the death globally, taking millions of lives every year.
2) Most of heart disease deaths are happened of heart attack or stroke.
3) More than three out of four deaths happened in poor countries.

1.1 Objectives of the Situation

How to know whether this person get the heart disease or not? What can we do to reduce the people die of the heart disease? How can those high-risk people to realize that they might got heart stoke and protect themselves?
That’s the reason we jump into this study, the objectives of the situation for the study:

- Analysis whether the person have a heart disease or not
- Find out relations between features (like gender, age) and the heart disease, analysis the main factors

1.2 Assess to the situation

1.2.1 Resources

Doctors are most professional groups to diagnose and give treatments to the patients, based on the patient's symptoms and their medical knowledge. However, they are more focus on the results and feedback of the treatment plans, they are still lack the data analysis and management knowledge. To get better understanding of the situation, we need the results based on the data analysis. So, it is critical to involve the data experts and data scientist into the project to ensure the effective results.

1.2.2 Data

This dataset should be collected from open source, which is reliable, complete, and no additional collections are needed. Such as the Kaggle and KD Nuggets websites.
The features in the datasets which matches the factors used in the clinical diagnosis would be more suitable for the project.

1.2.3 Requirements

There are no healthy and legal restrictions regarding the outcomes of information and project. The datasets are anonymous, so we do not need to worry about the project outcomes of the privacy policy problems.

1.2.4 Assumptions

We assume the datasets are qualified and there will be strong relationship between risk factors and heart disease and will used to predict the possibility of getting the heart disease.

1.2.5 Constraints

The dataset used here includes 1025 patients with 14 factors, data collected from the open source website-Kaggle which does not require additional password, its free to use.

1.2.6 Risks

We want to identify if the people have a heart disease or not through the study, and relations between the main factors and the heart disease, but if the results of the study are not direct to the target or shown lower connection to the target, the dataset might not representative enough for analysis.

1.2.7 Contingencies

Collect the dataset more representative as possible, sort out the dataset and factors more related to our objectives. And the explanation should base on the truth we found.

1.3 Data-mining objectives

After known our situation and the problems, the data mining objectives are list below:

- Use our dataset to establish the models to know whether the people has heart disease or not.
- Use the models to figure out which are the main factors to the heart disease.

Success criteria of the study:

- The model accuracy reached more than 80%
- Get to know the main features.

1.4 Project plan

Phase	Days	Resources	Risks
Situation understanding	1	Analysts	Situation change, data problems
Data Understanding	3	Analysts	Data and technology problems
Data Preparation	4	Data mining and statistic source	Data and technology problems
Data Transformation	3	Data mining and statistic source	Data and technology problems
Select Data Mining method(s)	2	Data mining and statistic source	Technology knowledge, lack of knowledge for finding suitable model
Select Data Mining algorithm(s)	1	Data mining and statistic source	Technology knowledge, lack of knowledge to find suitable model
Data Mining	5	Data mining and statistic source	lack of knowledge to implementing the results
Interpretation	4	All analysts	Situation changes, the lack of knowledge to implement the results

Gantt Chart for the project plan

2.Data Understanding
2.1 Collect original dataset

The dataset named "Heart Disease Dataset" is collected from **Kaggle website**:
https://www.kaggle.com/johnsmith88/heart-disease-dataset
The dataset is uploaded by **David Lapp**
His personal Kaggle page is:
https://www.kaggle.com/johnsmith88

Searched by the keywords of "Heart Disease" in the open source websites to find and compare the different datasets. When collecting the dataset, we want to collect more than two datasets to ensure the data more enough for analysis. But after seriously compare the datasets, this dataset with 14 attribute refers to the presence of heart disease in the patient which seems more representative than others, so we decided just to use this one dataset for current project analysis.
And as we know, Kaggle is an open source website provide the multiple datasets for data mining learners to use, so the dataset is reliable and matches our project objectives.

2.2 Describe the data

2.2.1 Amount of data

- 1025 patients
- 14 columns
- The name of the file is heart.csv
- The size of the file is 38 KB
- The format of the file is .csv

2.2.2 Value Types

- age - age in years (numeric: from 29 to 77)
- sex (male = 1; female= 0)
- cp-chest pain type (4 values, numeric from 0 to 3)
- resting blood pressure – trestbps (numeric: from 94 to 200)
- serum cholestoral in mg/dl – chol (numeric: from 126 to 564)
- fasting blood sugar – fbs (fasting blood sugar >120 mg/dl) (1 equate to true; 0 equate to false)
- resting electrocardiographic results -restecg (values are 0 1 2)
- maximum heart rate achieved – thalach (numeric: from number 71 to number 202)
- exercise induced angina – exang (1 equate to yes; 0 equate to no)
- oldpeak - ST depression induced by exercise relative to rest (Decimal: from 0 to 6.2)
- slope means the slope of the peak exercise ST segment (numeric: from number 0 to number 2)
- ca - number of major vessels (0-3) colored by flourosopy (numeric: from number 0 to number 4)
- thal (normal = 0 ; fixed defect = 1; reversable defect = 2)
- target (disease =1 and no disease = 0)

2.3 Explore the data

2.3.1 Read CSV file, Create Data Frames

```
import findspark
import pyspark
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName('heart').getOrCreate()
df = spark.read.load('./heart.csv', format= "csv", header='true')
hd_data = df.toPandas()
```

2.3.2 Data Shape

```
heart_data.shape
```

```
(1025, 14)
```

2.3.3 Data Type

```
heart_data.dtypes
```

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

2.3.4 Data Head

```
heart_data.head()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	52	1	0	125	212	0	1	168	0	1	2	2	3	0
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	0
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	0
3	61	1	0	148	203	0	1	161	0	0	2	1	3	0
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	0

2.3.5 Summary of fields

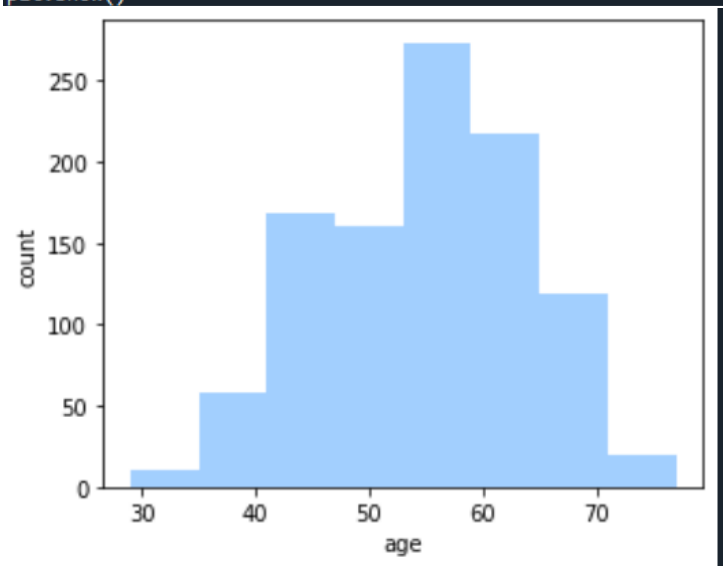
```
heart_data.describe()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
count	1025	1025	1025	1025	1025	1025	1025	1025	1025	1025	1025	1025	1025	1025
unique	41	2	4	49	152	2	3	91	2	40	3	5	4	2
top	58	1	0	120	204	0	1	162	0	0	1	0	2	1
freq	68	713	497	128	21	872	513	35	680	329	482	578	544	526

2.3.6 Main Raw Data visualizations

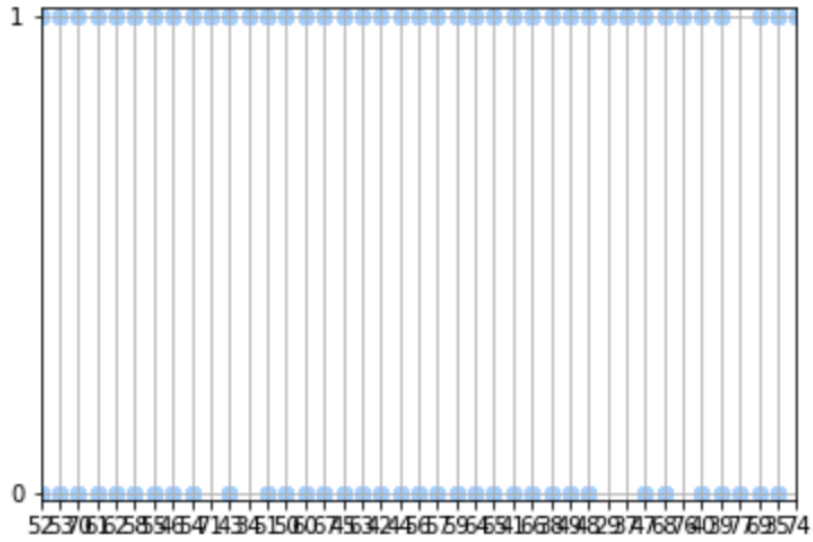
2.3.6.1 age

```
fig,ax = plt.subplots(figsize=(5,4))
sns.distplot(hd_data['age'],
             hist_kws={"alpha":1,"color":"#a2cffe"},
             kde=False, bins=8)
ax = ax.set(ylabel='count', xlabel='age')
plt.show()
```



From the age distribution above, we can see most of the patients from age 40-70.

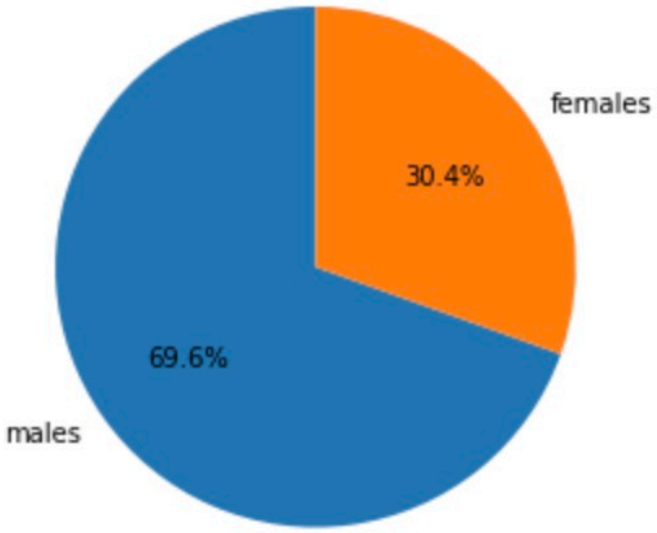
```
plt.scatter(x='age',y='target',data= hd_data,color='#a2cffe')
plt.autoscale(tight=True)
plt.grid()
plt.show()
```



And we write code to visual different age impact on the target, by compared from the above graph, there are no such big difference to the target due to the age feature.

2.3.6.2 sex

```
fig,ax = plt.subplots(figsize =(4,4))
plt.pie(hd_data['sex'].value_counts().tolist(),
        labels=['males','females'],
       autopct='%1.1f%%',startangle=90)
axis =plt.axis('equal')
```



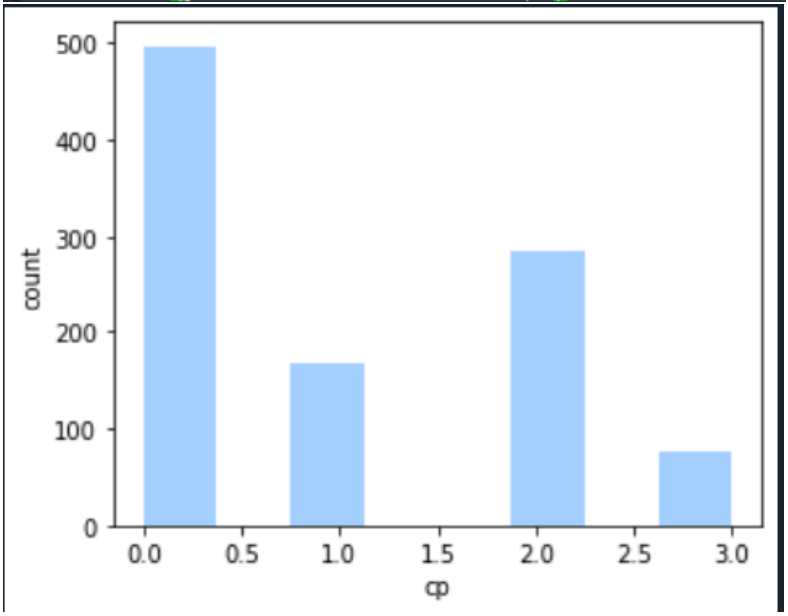
There are more males than females in the dataset, males around 70% in the data.

```
plt.scatter(x='sex',y='target',data= hd_data,color='#a2cffe')
plt.autoscale(tight=True)
plt.grid()
plt.show()
```

From the graph above, we can see here is also no much difference between males and females to the target.

2.3.6.3 cp-chest pain type

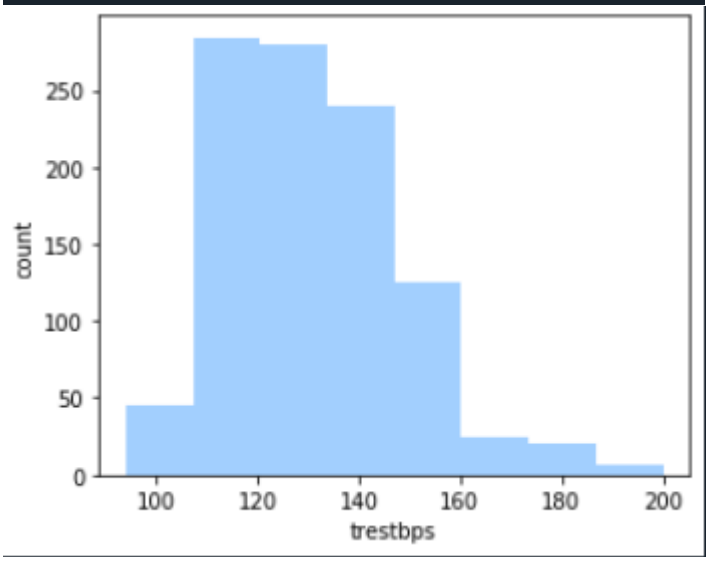
```
fig,ax = plt.subplots(figsize=(5,4))
sns.distplot(hd_data['cp'],
             hist_kws={"alpha":1,"color":"#a2cffe"},
             kde=False, bins=8)
ax = ax.set(ylabel='count', xlabel='cp')
```



There are four types of the chest pain type, and the type 0 made of the most distribution.

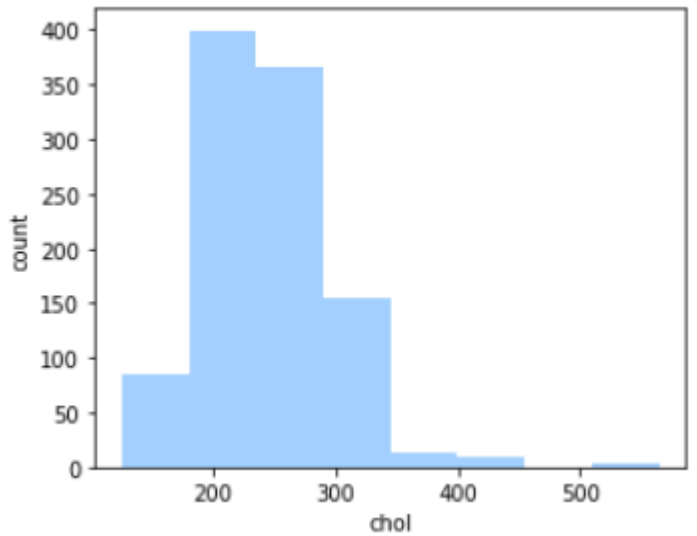
2.6.3.4 trestbps-resting blood pressure

```
fig,ax = plt.subplots(figsize=(5,4))
sns.distplot(hd_data['trestbps'],
             hist_kws={"alpha":1,"color":"#a2cffe"},
             kde=False, bins=8)
ax = ax.set(ylabel='count', xlabel='trestbps')
```



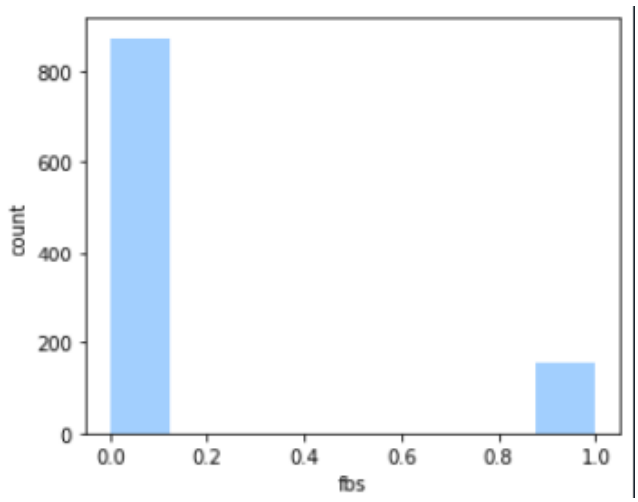
2.6.3.5 chol-serum cholestoral in mg/dl

```
fig,ax = plt.subplots(figsize=(5,4))
sns.distplot(hd_data['chol'],
             hist_kws={"alpha":1,"color":"#a2cffe"},
             kde=False, bins=8)
ax = ax.set(ylabel='count', xlabel='chol')
```



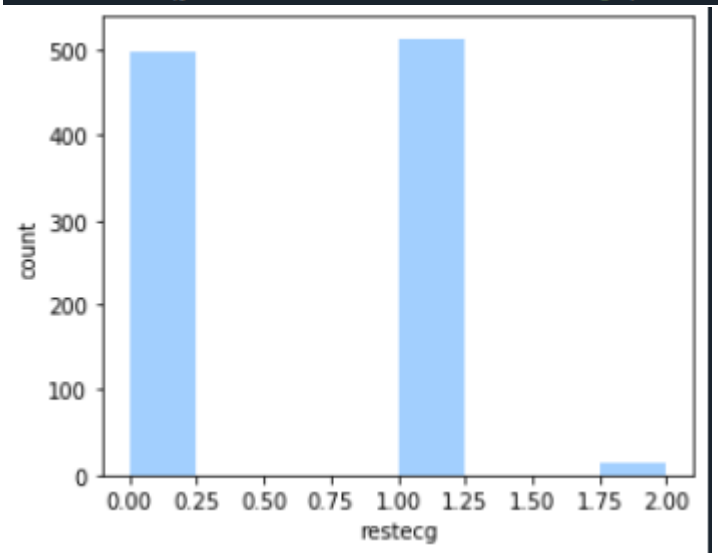
2.6.3.6 fbs-fasting blood sugar

```
fig,ax = plt.subplots(figsize=(5,4))
sns.distplot(hd_data['fbs'],
             hist_kws={"alpha":1,"color":"#a2cffe"},
             kde=False, bins=8)
ax = ax.set(ylabel='count', xlabel='fbs')
```



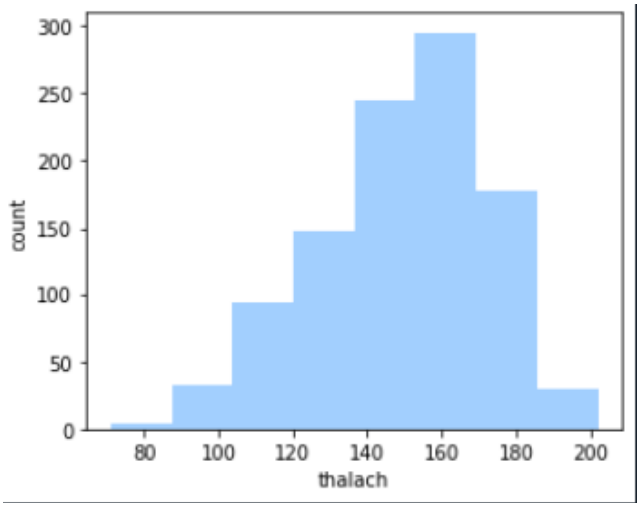
2.6.3.7 restecg-resting electrocardiographic results

```
fig,ax = plt.subplots(figsize=(5,4))
sns.distplot(hd_data['restecg'],
             hist_kws={"alpha":1,"color":"#a2cffe"},
             kde=False, bins=8)
ax = ax.set(ylabel='count', xlabel='restecg')
```



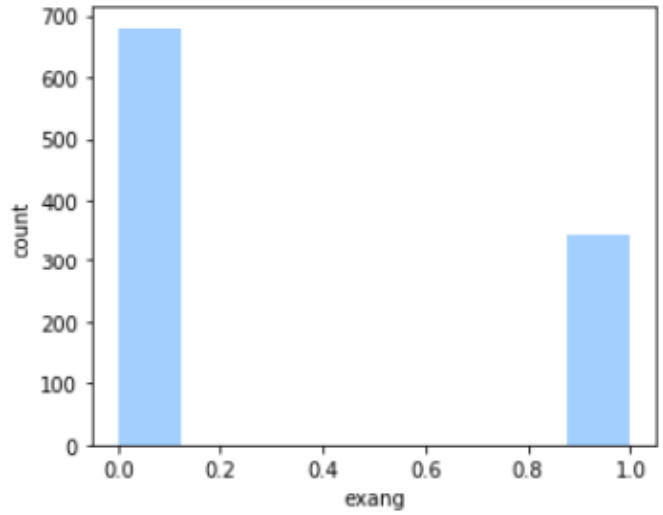
2.6.3.8 thalach-maximum heart rate achieved

```
fig,ax = plt.subplots(figsize=(5,4))
sns.distplot(hd_data['thalach'],
             hist_kws={"alpha":1,"color":"#a2cffe"},
             kde=False, bins=8)
ax = ax.set(ylabel='count', xlabel='thalach')
```



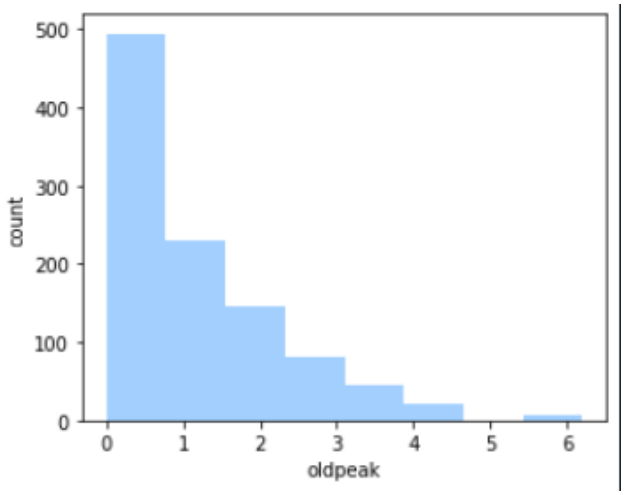
2.6.3.9 exang-exercise induced angina

```
fig,ax = plt.subplots(figsize=(5,4))
sns.distplot(hd_data['exang'],
             hist_kws={"alpha":1,"color":"#a2cffe"},
             kde=False, bins=8)
ax = ax.set(ylabel='count', xlabel='exang')
```



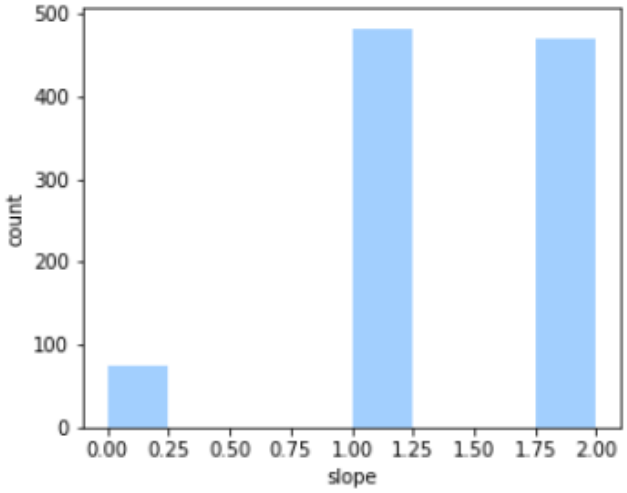
2.6.3.10 oldpeak - ST depression induced by exercise relative to rest

```
fig,ax = plt.subplots(figsize=(5,4))
sns.distplot(hd_data['oldpeak'],
             hist_kws={"alpha":1,"color":"#a2cffe"},
             kde=False, bins=8)
ax = ax.set(ylabel='count', xlabel='oldpeak')
```



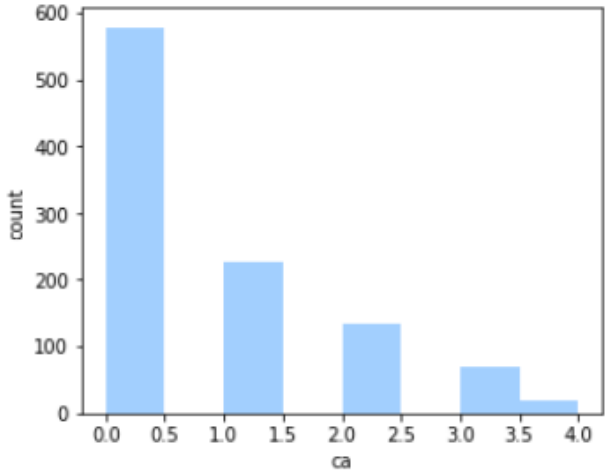
2.6.3.11 slope- the slope of the peak exercise ST segment

```
fig,ax = plt.subplots(figsize=(5,4))
sns.distplot(hd_data['slope'],
             hist_kws={"alpha":1,"color":"#a2cffe"},
             kde=False, bins=8)
ax = ax.set(ylabel='count', xlabel='slope')
```



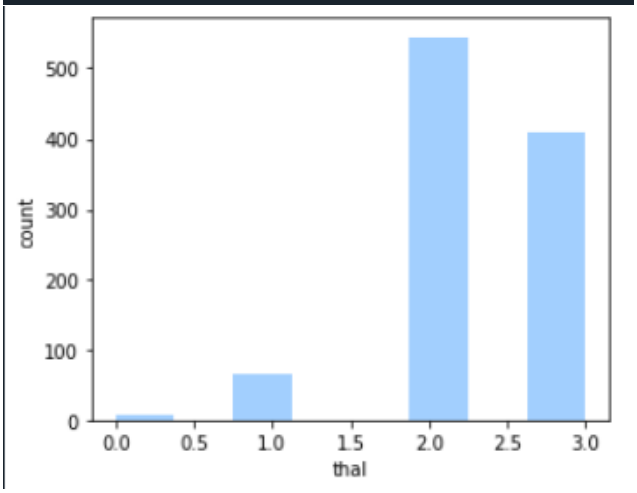
2.6.3.12 ca - number of major vessels

```
fig,ax = plt.subplots(figsize=(5,4))
sns.distplot(hd_data['ca'],
             hist_kws={"alpha":1,"color":"#a2cffe"},
             kde=False, bins=8)
ax = ax.set(ylabel='count', xlabel='ca')
```



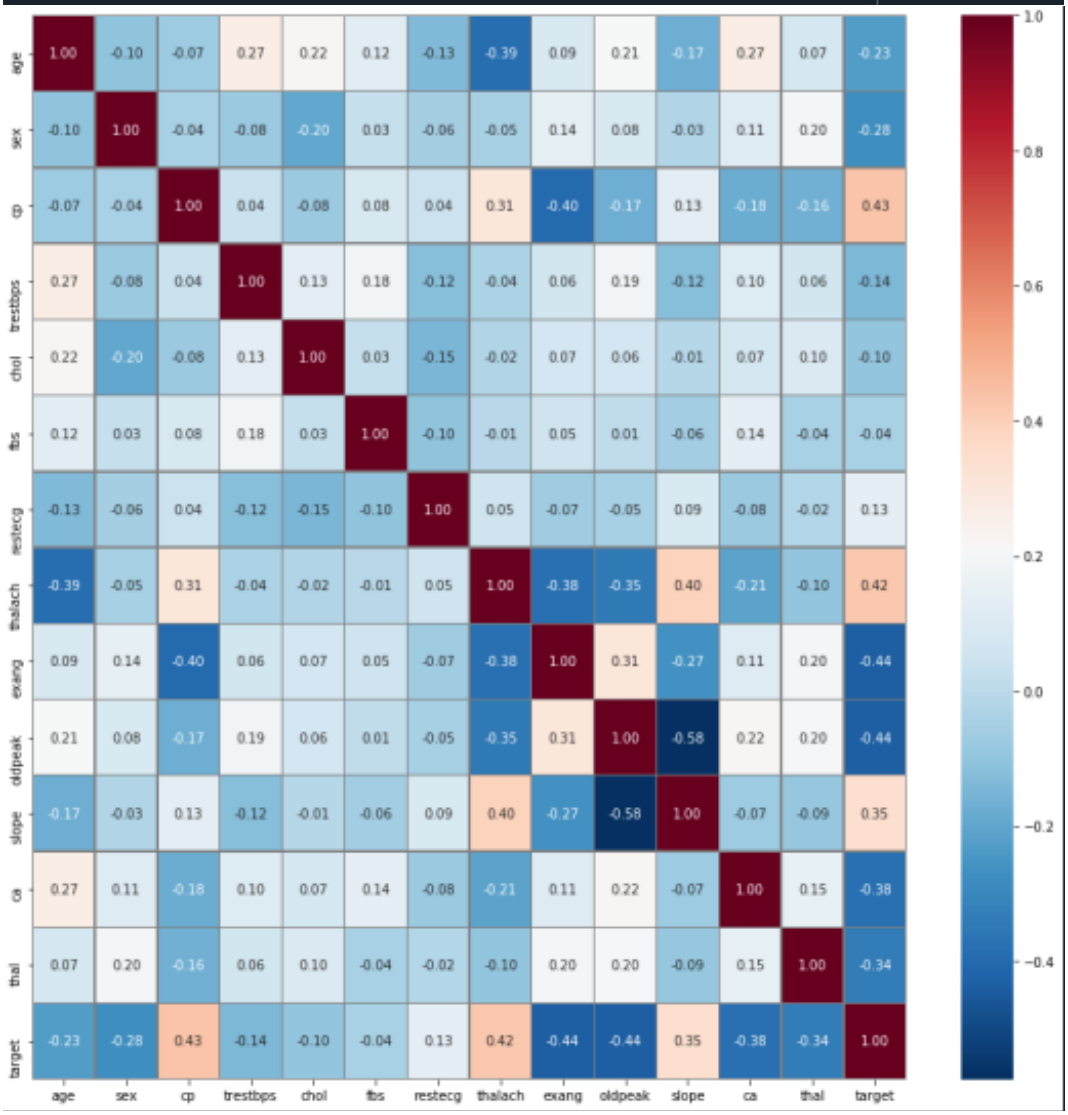
2.6.3.13 thal

```
fig,ax = plt.subplots(figsize=(5,4))
sns.distplot(hd_data['thal'],
             hist_kws={"alpha":1,"color":"#a2cffe"},
             kde=False, bins=8)
ax = ax.set(ylabel='count', xlabel='thal')
```



2.6.3.14 Correlation

```
plt.figure(figsize=(15,15))
sns.heatmap(hd_data.corr(),annot=True,fmt=".2f",linewidths=0.3,linecolor="grey",cmap="RdBu_r")
```



From the matrix above, we can see the features are most independently each from other, there are relations between **slope** and **thalach** as the number is **0.40.** Furthermore, we can see the **cp**, **thalach** and **slope** have strong relationships to the target, the numbers are **0.43**,**0.42** and **0.35** independently. Those three features should be the main features to our target.

2.4 Verify data quality

To check the quality of the data, the output below shows the result of dataset quality.

```
hd_data.isnull().any()
```

```
age         False
sex         False
cp          False
trestbps    False
chol        False
fbs         False
restecg     False
thalach     False
exang       False
oldpeak     False
slope       False
ca          False
thal        False
target      False
dtype: bool
```

Missing Data: for the dataset, all the fields and records are false, means the data are 100% complete, no any null, empty, white or blank space values. So, we do not need to handle missing values in the following steps.

3.Data Preparation

3.1 Select the data

```
import findspark
import pyspark
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName('heart').getOrCreate()
df = spark.read.load('./heart.csv', format= "csv", header='true')
hd_data = df.toPandas()
```

```
heart_data.head()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	52	1	0	125	212	0	1	168	0	1	2	2	3	0
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	0
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	0
3	61	1	0	148	203	0	1	161	0	0	2	1	3	0
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	0

```
hd_data.describe()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
count	1025	1025	1025	1025	1025	1025	1025	1025	1025	1025	1025	1025	1025	1025
unique	41	2	4	49	152	2	3	91	2	40	3	5	4	2
top	58	1	0	120	234	0	1	162	0	0	1	0	2	1
freq	68	713	497	128	21	872	513	35	680	329	482	578	544	526

```
hd_data.shape
```

```
(1025, 14)
```

This original dataset was downloaded from Kaggle, includes 1025 rows(patients) with 14 columns(features). Searched by the keywords of “Heart Disease” in the open source websites to find and compare the different datasets. After compare the datasets, this dataset with 14 attribute refers to the presence of heart disease in the patient which seems more representative than others, so we decided just to use this one dataset for current project analysis.

And as we know, Kaggle is an open source website provide the multiple datasets for data mining learners to use, so the dataset is reliable and matches our project objectives. And the data types match the features.

```
heart_data.dtypes
```

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

3.2 Clean the data

3.2.1 Missing Data:

```
hd_data.isnull().any()
```

```
age         False
sex         False
cp          False
trestbps    False
chol        False
fbs         False
restecg     False
thalach     False
exang       False
oldpeak     False
slope       False
ca          False
thal        False
target      False
dtype: bool
```

The dataset we use is 100% complete, and we can see there are no any null, empty, white or blank space values. We do not need to handle missing values here.

3.2.2 Outliers and Extremes values:

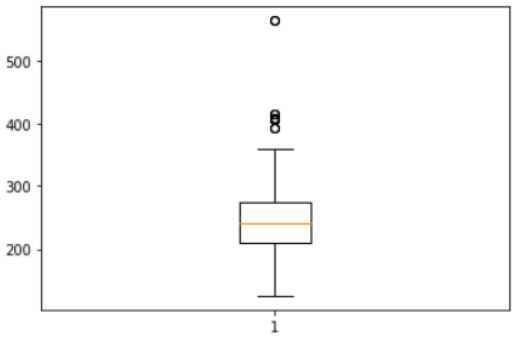
	age	sex	cp	trestbps	chol	\
count	1025.000000	1025.000000	1025.000000	1025.000000	1025.00000	
mean	54.434146	0.695610	0.942439	131.611707	246.00000	
std	9.072290	0.460373	1.029641	17.516718	51.59251	
min	29.000000	0.000000	0.000000	94.000000	126.00000	
25%	48.000000	0.000000	0.000000	120.000000	211.00000	
50%	56.000000	1.000000	1.000000	130.000000	240.00000	
75%	61.000000	1.000000	2.000000	140.000000	275.00000	
max	77.000000	1.000000	3.000000	200.000000	564.00000	

	fbs	restecg	thalach	exang	oldpeak	\
count	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	
mean	0.149268	0.529756	149.114146	0.336585	1.071512	
std	0.356527	0.527878	23.005724	0.472772	1.175053	
min	0.000000	0.000000	71.000000	0.000000	0.000000	
25%	0.000000	0.000000	132.000000	0.000000	0.000000	
50%	0.000000	1.000000	152.000000	0.000000	0.800000	
75%	0.000000	1.000000	166.000000	1.000000	1.800000	
max	1.000000	2.000000	202.000000	1.000000	6.200000	

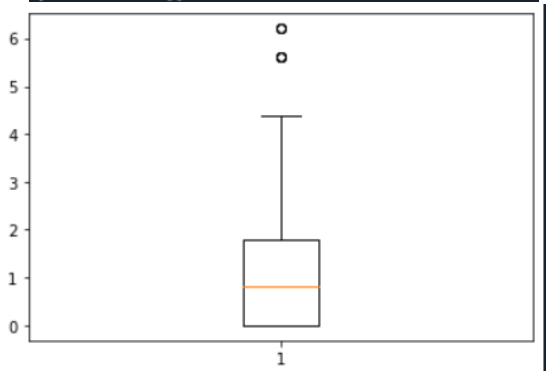
	slope	ca	thal	target
count	1025.000000	1025.000000	1025.000000	1025.000000
mean	1.385366	0.754146	2.323902	0.513171
std	0.617755	1.030798	0.620660	0.500070
min	0.000000	0.000000	0.000000	0.000000
25%	1.000000	0.000000	2.000000	0.000000
50%	1.000000	0.000000	2.000000	1.000000
75%	2.000000	1.000000	3.000000	1.000000
max	2.000000	4.000000	3.000000	1.000000

From the graphs above, we can see the extreme and outliers' values of **chol** and **oldpeak** features might have significant impact to our dataset. We are drawing graphs to see this in detail as following:

```
plt.boxplot(hd_data["chol"])
plt.show()
```




```
plt.boxplot(hd_data["oldpeak"])
plt.show()
```

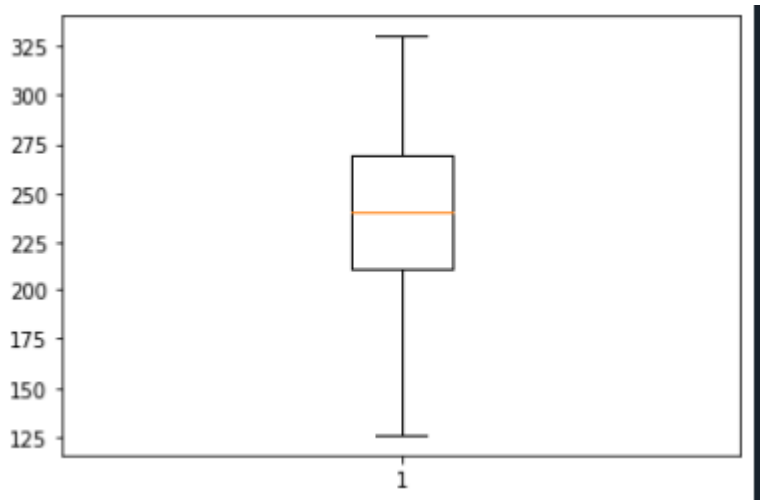


From the groups, we need to handle the extreme and outliers' values of both **chol** and **oldpeak** features. we use median values to replacing Outliers. Due to the mean values are affected by outlies, so we advised not use the mean values. The 1st line of code below prints the median value. The 2nd line prints the 95% value. The 3rd line of code below replaces all those values in the variable, which are greater than the 95%, with the median value. Finally,

the 4th line prints summary statistics after all these techniques have been employed for outlier treatment. Please see the following compare with the previous ones.

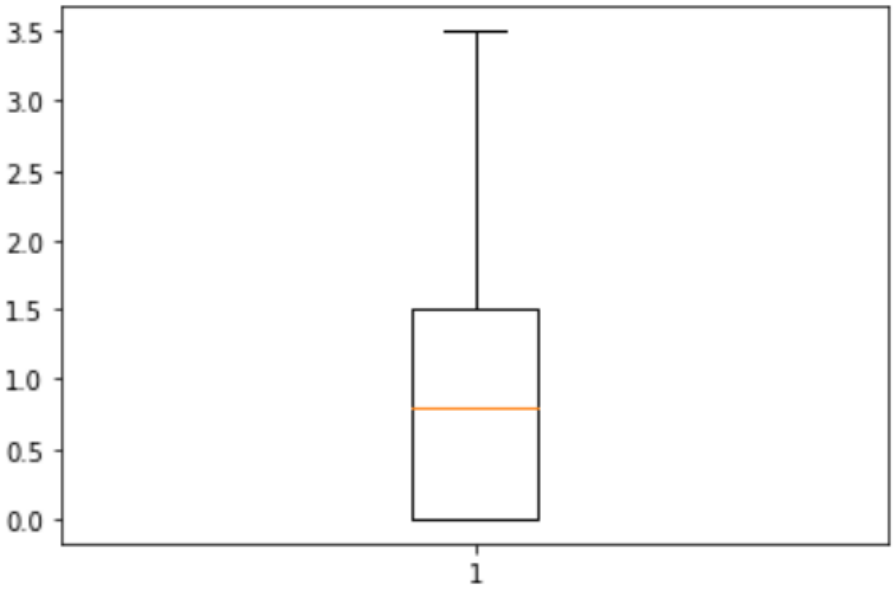
For the **chol** feature:

```
print(hd_data['chol'].quantile(0.50))
print(hd_data['chol'].quantile(0.95))
hd_data['chol'] = np.where(hd_data['chol'] > 330, 240, hd_data['chol'])
print(hd_data.describe())

plt.boxplot(hd_data["chol"])
plt.show()
```



For the **oldpeak** feature:

```
print(hd_data['oldpeak'].quantile(0.50))
print(hd_data['oldpeak'].quantile(0.95))
hd_data['oldpeak'] = np.where(hd_data['oldpeak'] > 3.5, 0.8, hd_data['oldpeak'])
print(hd_data.describe())

plt.boxplot(hd_data["oldpeak"])
plt.show()
```



After the handling of the whole outliers' values, the summary of the features:

	age	sex	cp	trestbps	chol	\
count	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	
mean	54.434146	0.695610	0.942439	131.611707	239.984390	
std	9.072290	0.460373	1.029641	17.516718	41.544683	
min	29.000000	0.000000	0.000000	94.000000	126.000000	
25%	48.000000	0.000000	0.000000	120.000000	211.000000	
50%	56.000000	1.000000	1.000000	130.000000	240.000000	
75%	61.000000	1.000000	2.000000	140.000000	269.000000	
max	77.000000	1.000000	3.000000	200.000000	330.000000	

	fbs	restecg	thalach	exang	oldpeak	\
count	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	
mean	0.149268	0.529756	149.114146	0.336585	0.912878	
std	0.356527	0.527878	23.005724	0.472772	0.937016	
min	0.000000	0.000000	71.000000	0.000000	0.000000	
25%	0.000000	0.000000	132.000000	0.000000	0.000000	
50%	0.000000	1.000000	152.000000	0.000000	0.800000	
75%	0.000000	1.000000	166.000000	1.000000	1.500000	
max	1.000000	2.000000	202.000000	1.000000	3.500000	

	slope	ca	thal	target
count	1025.000000	1025.000000	1025.000000	1025.000000
mean	1.385366	0.754146	2.323902	0.513171
std	0.617755	1.030798	0.620660	0.500070
min	0.000000	0.000000	0.000000	0.000000
25%	1.000000	0.000000	2.000000	0.000000
50%	1.000000	0.000000	2.000000	1.000000
75%	2.000000	1.000000	3.000000	1.000000
max	2.000000	4.000000	3.000000	1.000000

3.3 Construct the data

	age	sex	cp	trestbps	chol	\
count	1025.000000	1025.000000	1025.000000	1025.000000	1025.00000	
mean	54.434146	0.695610	0.942439	131.611707	246.00000	
std	9.072290	0.460373	1.029641	17.516718	51.59251	
min	29.000000	0.000000	0.000000	94.000000	126.00000	
25%	48.000000	0.000000	0.000000	120.000000	211.00000	
50%	56.000000	1.000000	1.000000	130.000000	240.00000	
75%	61.000000	1.000000	2.000000	140.000000	275.00000	
max	77.000000	1.000000	3.000000	200.000000	564.00000	

	fbs	restecg	thalach	exang	oldpeak	\
count	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	
mean	0.149268	0.529756	149.114146	0.336585	1.071512	
std	0.356527	0.527878	23.005724	0.472772	1.175053	
min	0.000000	0.000000	71.000000	0.000000	0.000000	
25%	0.000000	0.000000	132.000000	0.000000	0.000000	
50%	0.000000	1.000000	152.000000	0.000000	0.800000	
75%	0.000000	1.000000	166.000000	1.000000	1.800000	
max	1.000000	2.000000	202.000000	1.000000	6.200000	

	slope	ca	thal	target
count	1025.000000	1025.000000	1025.000000	1025.000000
mean	1.385366	0.754146	2.323902	0.513171
std	0.617755	1.030798	0.620660	0.500070
min	0.000000	0.000000	0.000000	0.000000
25%	1.000000	0.000000	2.000000	0.000000
50%	1.000000	0.000000	2.000000	1.000000
75%	2.000000	1.000000	3.000000	1.000000
max	2.000000	4.000000	3.000000	1.000000

Before the construct

	age	sex	cp	trestbps	chol	\
count	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	
mean	54.434146	0.695610	0.942439	131.611707	239.984390	
std	9.072290	0.460373	1.029641	17.516718	41.544683	
min	29.000000	0.000000	0.000000	94.000000	126.000000	
25%	48.000000	0.000000	0.000000	120.000000	211.000000	
50%	56.000000	1.000000	1.000000	130.000000	240.000000	
75%	61.000000	1.000000	2.000000	140.000000	269.000000	
max	77.000000	1.000000	3.000000	200.000000	330.000000	

	fbs	restecg	thalach	exang	oldpeak	\
count	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	
mean	0.149268	0.529756	149.114146	0.336585	0.912878	
std	0.356527	0.527878	23.005724	0.472772	0.937016	
min	0.000000	0.000000	71.000000	0.000000	0.000000	
25%	0.000000	0.000000	132.000000	0.000000	0.000000	
50%	0.000000	1.000000	152.000000	0.000000	0.800000	
75%	0.000000	1.000000	166.000000	1.000000	1.500000	
max	1.000000	2.000000	202.000000	1.000000	3.500000	

	slope	ca	thal	target
count	1025.000000	1025.000000	1025.000000	1025.000000
mean	1.385366	0.754146	2.323902	0.513171
std	0.617755	1.030798	0.620660	0.500070
min	0.000000	0.000000	0.000000	0.000000
25%	1.000000	0.000000	2.000000	0.000000
50%	1.000000	0.000000	2.000000	1.000000
75%	2.000000	1.000000	3.000000	1.000000
max	2.000000	4.000000	3.000000	1.000000

After the construct

We have constructed the **chol** and **oldpeak** data features to handling the outliers' values. And we can see the numbers of the **trestbps**, **chol** and **thalach** features are more than others, this might cause problems later in our analysis. So, we try to contrast the log data of those three features, log_trestbps, log_chol and log_thalach. Please see the python code and the results of the constructed all those three features as following:

```
hd_data['log_chol']=np.log(hd_data['chol'])
hd_data['log_trestbps']= np.log(hd_data['trestbps'])
hd_data['log_thalach']=np.log(hd_data['thalach'])
print(hd_data.head())
print(hd_data.describe())
```

	age	sex	cp	trestbps	chol	\
count	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	
mean	54.434146	0.695610	0.942439	131.611707	239.984390	
std	9.072290	0.460373	1.029641	17.516718	41.544683	
min	29.000000	0.000000	0.000000	94.000000	126.000000	
25%	48.000000	0.000000	0.000000	120.000000	211.000000	
50%	56.000000	1.000000	1.000000	130.000000	240.000000	
75%	61.000000	1.000000	2.000000	140.000000	269.000000	
max	77.000000	1.000000	3.000000	200.000000	330.000000	

	fbs	restecg	thalach	exang	oldpeak	\
count	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	
mean	0.149268	0.529756	149.114146	0.336585	0.912878	
std	0.356527	0.527878	23.005724	0.472772	0.937016	
min	0.000000	0.000000	71.000000	0.000000	0.000000	
25%	0.000000	0.000000	132.000000	0.000000	0.000000	
50%	0.000000	1.000000	152.000000	0.000000	0.800000	
75%	0.000000	1.000000	166.000000	1.000000	1.500000	
max	1.000000	2.000000	202.000000	1.000000	3.500000	

	slope	ca	thal	target
count	1025.000000	1025.000000	1025.000000	1025.000000
mean	1.385366	0.754146	2.323902	0.513171
std	0.617755	1.030798	0.620660	0.500070
min	0.000000	0.000000	0.000000	0.000000
25%	1.000000	0.000000	2.000000	0.000000
50%	1.000000	0.000000	2.000000	1.000000
75%	2.000000	1.000000	3.000000	1.000000
max	2.000000	4.000000	3.000000	1.000000

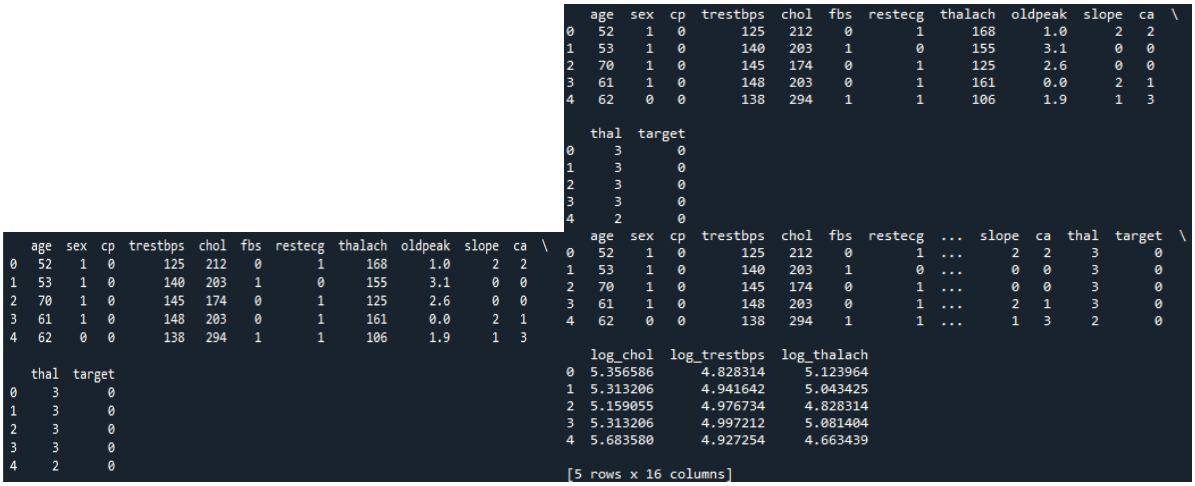
	age	sex	cp	trestbps	chol	\
count	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	
mean	54.434146	0.695610	0.942439	131.611707	239.984390	
std	9.072290	0.460373	1.029641	17.516718	41.544683	
min	29.000000	0.000000	0.000000	94.000000	126.000000	
25%	48.000000	0.000000	0.000000	120.000000	211.000000	
50%	56.000000	1.000000	1.000000	130.000000	240.000000	
75%	61.000000	1.000000	2.000000	140.000000	269.000000	
max	77.000000	1.000000	3.000000	200.000000	330.000000	

	fbs	restecg	...	slope	ca	thal	\
count	1025.000000	1025.000000	...	1025.000000	1025.000000	1025.000000	
mean	0.149268	0.529756	...	1.385366	0.754146	2.323902	
std	0.356527	0.527878	...	0.617755	1.030798	0.620660	
min	0.000000	0.000000	...	0.000000	0.000000	0.000000	
25%	0.000000	0.000000	...	1.000000	0.000000	2.000000	
50%	0.000000	1.000000	...	1.000000	0.000000	2.000000	
75%	0.000000	1.000000	...	2.000000	1.000000	3.000000	
max	1.000000	2.000000	...	2.000000	4.000000	3.000000	

	target	log_chol	log_trestbps	log_thalach
count	1025.000000	1025.000000	1025.000000	1025.000000
mean	0.513171	5.465053	4.871322	4.991667
std	0.500070	0.178507	0.129845	0.165960
min	0.000000	4.836282	4.543295	4.262680
25%	0.000000	5.351858	4.787492	4.882802
50%	1.000000	5.480639	4.867534	5.023881
75%	1.000000	5.594711	4.941642	5.111988
max	1.000000	5.799093	5.298317	5.308268

[8 rows x 16 columns]

Before vs After



Before vs After

Compared each summary data above, we can see there are 3 more features in our dataset, **log_chol**, **log_trestbps** and **log_thalach**. Now, the data quality is good for next step.

3.4 Integrate various datasets

In this study, we just use one dataset, so there is no merging data needed for the project. We will consider the integration various datasets if there are any specific datasets match our data mining goals as well.

3.5 Format the data

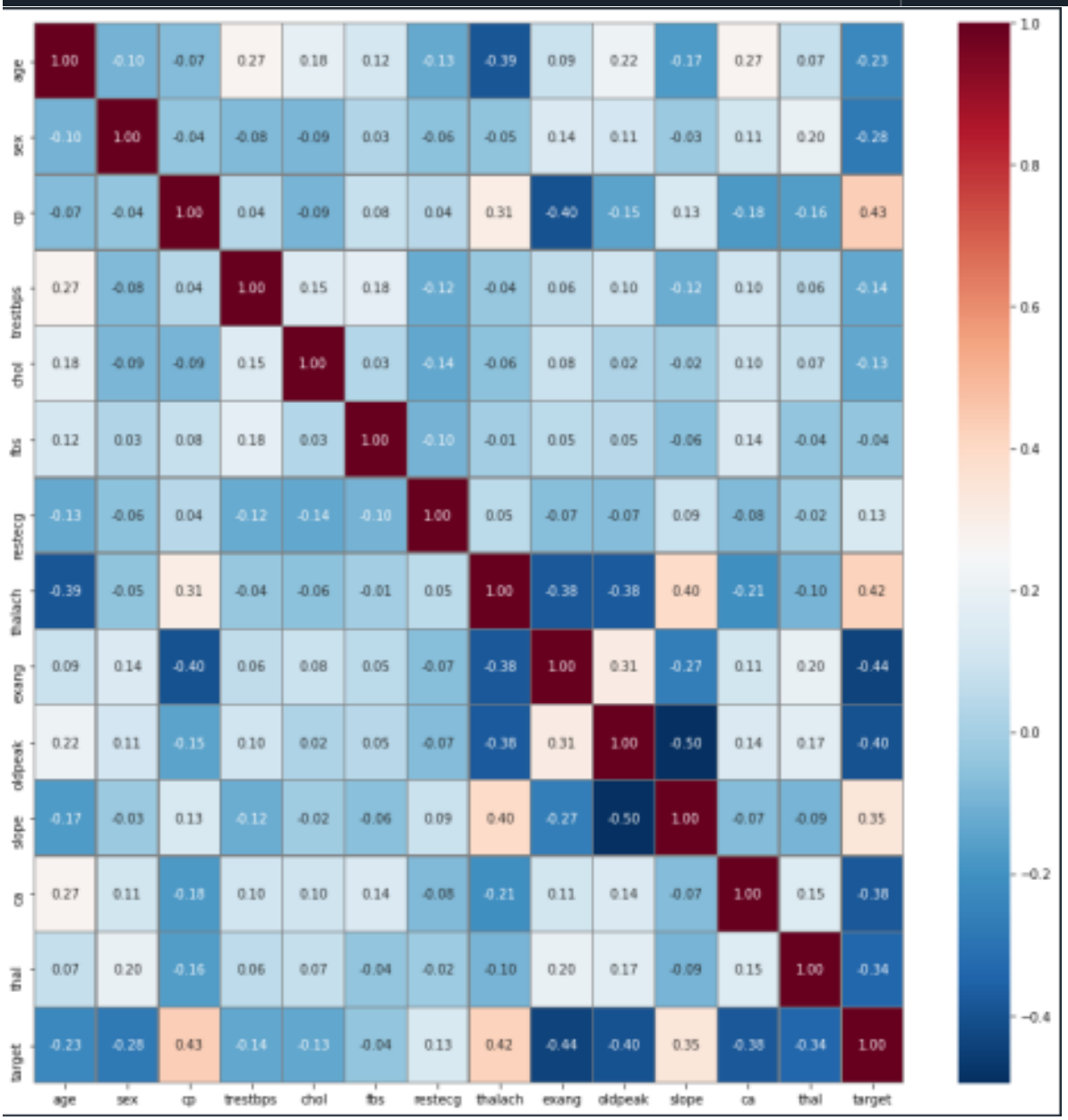
We have only one dataset, and this dataset used in here is relatively small, already sorting before modelling (refer to explore the data), so it's not necessary to reformat the data in this step.

4.Data Transformation

4.1 Reduce the data

Importance check:

```
plt.figure(figsize=(15,15))
sns.heatmap(hd_data.corr(),annot=True,fmt=".2f",linewidths=0.3,linecolor="grey",cmap="RdBu_r")
```



From the matrix, we can the **exang** is **-0.44** which is the most unimportant feature to the target. So, we reduce the **exang** feature here.

Remove unimportant features:

```
heart_data.head()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	52	1	0	125	212	0	1	168	0	1	2	2	3	0
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	0
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	0
3	61	1	0	148	203	0	1	161	0	0	2	1	3	0
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	0

We can see the whole data head before the remove of the unimportant feature above, and we use python **del** method to delete the unimportant feature **exang**. Thus, we can see the new data head as following:

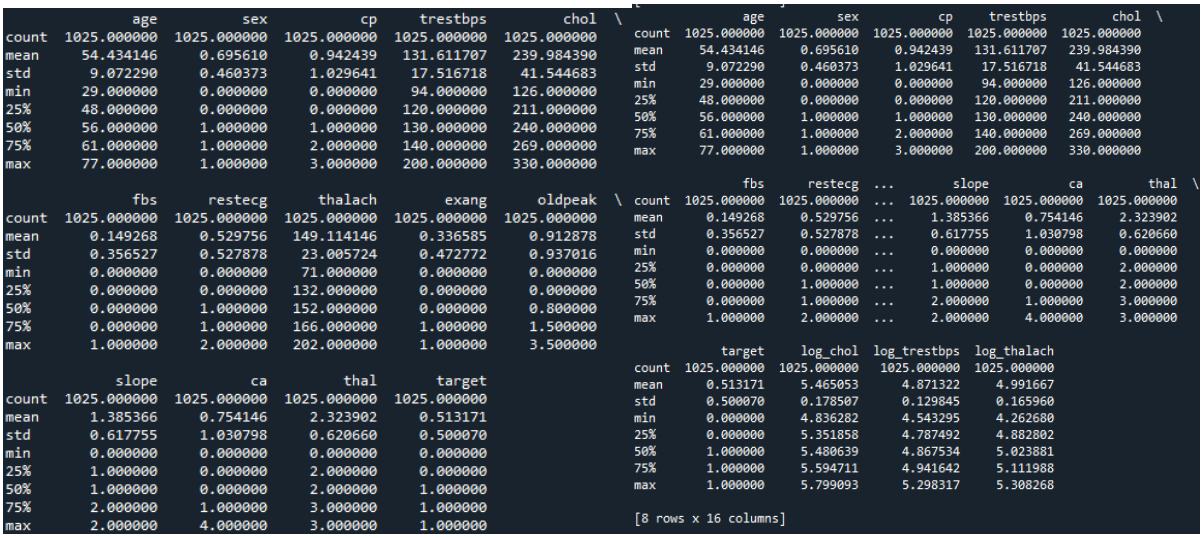
```
del hd_data['exang']
hd_data.head()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	oldpeak	slope	ca	thal	target
0	52	1	0	125	212	0	1	168	1	2	2	3	0
1	53	1	0	140	203	1	0	155	3.1	0	0	3	0
2	70	1	0	145	174	0	1	125	2.6	0	0	3	0
3	61	1	0	148	203	0	1	161	0	2	1	3	0
4	62	0	0	138	294	1	1	106	1.9	1	3	2	0

4.2 Project the data

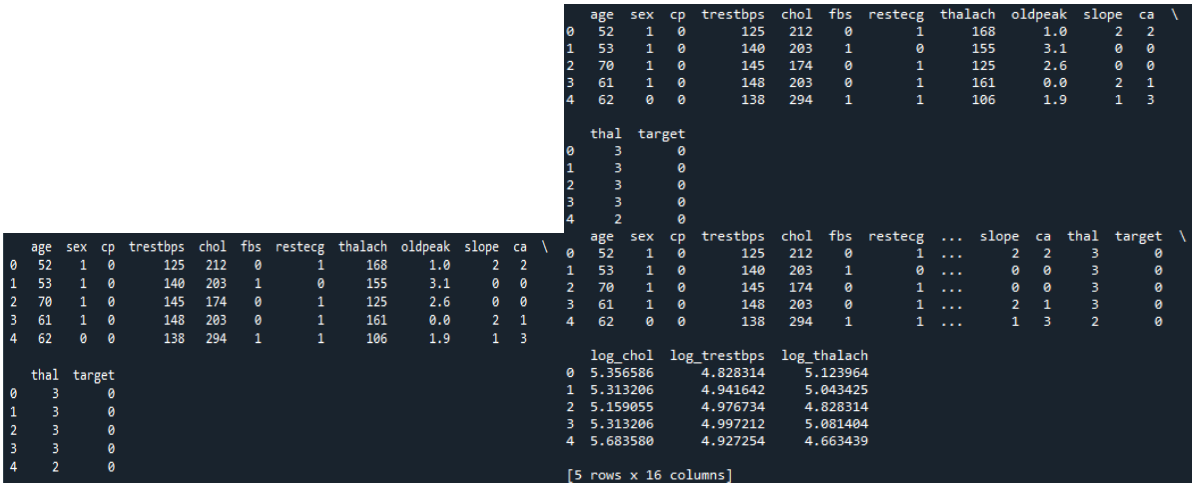
Log transform: The distributions of features are not in a similar range. We can see the numbers of the **trestbps**, **chol** and **thalach** features are more than others, this might cause problems later in our analysis. So, we try to contrast the log data of those three features, log_trestbps, log_chol and log_thalach. Please see the python code and the results of the constructed all those three features as following:

```
hd_data['log_chol']=np.log(hd_data['chol'])
hd_data['log_trestbps']= np.log(hd_data['trestbps'])
hd_data['log_thalach']=np.log(hd_data['thalach'])
print(hd_data.head())
print(hd_data.describe())
```



	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
count	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000
mean	54.434146	0.695610	0.942439	131.611707	239.984390	0.149268	0.529756	149.114146	0.336585	0.912878	1.385366	0.754146	2.323902	0.513171
std	9.072290	0.460373	1.029641	17.516718	41.544683	0.356527	0.527878	23.005724	0.472772	0.937016	0.617755	1.030798	0.620660	0.500070
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	0.000000	0.000000
25%	48.000000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	132.000000	0.000000	0.000000	1.000000	0.000000	2.000000	0.000000
50%	56.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	152.000000	0.000000	0.800000	1.000000	0.000000	2.000000	1.000000
75%	61.000000	1.000000	2.000000	140.000000	269.000000	0.000000	1.000000	166.000000	1.000000	1.500000	2.000000	1.000000	3.000000	1.000000
max	77.000000	1.000000	3.000000	200.000000	330.000000	1.000000	2.000000	202.000000	1.000000	3.500000	2.000000	4.000000	3.000000	1.000000

Before vs After



	age	sex	cp	trestbps	chol	fbs	restecg	thalach	oldpeak	slope	ca	thal	target
0	52	1	0	125	212	0	1	168	1.0	2	2	3	0
1	53	1	0	140	203	1	0	155	3.1	0	0	3	0
2	70	1	0	145	174	0	1	125	2.6	0	0	3	0
3	61	1	0	148	203	0	1	161	0.0	2	1	3	0
4	62	0	0	138	294	1	1	106	1.9	1	3	2	0

	log_chol	log_trestbps	log_thalach
0	5.356586	4.828314	5.123964
1	5.313206	4.941642	5.043425
2	5.159055	4.976734	4.828314
3	5.313206	4.997212	5.081404
4	5.683580	4.927254	4.663439

[5 rows x 16 columns]

Before vs After

Compared each summary data above, we can see there are 3 more features in our dataset, **log_chol**, **log_trestbps** and **log_thalach**. Now, the data quality is good for next step.

5.Select Data-mining Method(s)

5.1 Match and discuss data-mining methods to the data mining objectives

There are two data-mining objectives in the study:

- Use the dataset to establish models to identify whether the patient have the heart disease, or the patient not have the heart disease. The aim of the prediction model is the ability to classify a new case is have or haven't, our model has one single target
- Regarding objective 2, use the models to find out significant factors to the heart disease, the relationship between those features and the heart disease

And there are three types of data-mining methods in the study: Supervised Learning: Classification and Regression, Unsupervised Learning-Clustering, we are discussing those methods separately here.

Supervised Learning: Classification
Classification is a subcategory of supervised learning; the goal is to predict the categorical class labels of new instances based on past observations. There are two main types of the classification: Binary Classification and Multi-class Classification
For Binary Classification: There are two outcomes of this classification, for example: true or false. For Multi-class Classification: There are more than two classes in this type of classification, and each sample is assigned to one and only one label or target.
Since we know our main objective is to identify whether the patient have heart disease problems or not, our goal matches the Binary Classification, as there are only two classes: has the heart disease or not have. So, we can use classification method to build the prediction model based on our input dataset, once the model is trained and test accurately, we can use those models to identify whether the heart disease exist or not for the new patient data.

Supervised Learning: Regression
Regression is a subtype of supervised learning, requires continuous or real values, predicting a continuous quantity, such as the task to predicting the age of a person.
In our study, the main objective is a binary problem, not match the scope of the Regression problem definition.

Unsupervised Learning: Clustering
Unsupervised Learning are given no labels, and Clustering is a process to sort out the similar values into groups or to find the patterns in the data by itself.
As target is very clearly, and the problem we study it's not the clustering. So, we are not use this method for our next step.

5.2 Select the appropriate data-mining method(s)
Based on our 5.1 discussion, in our study, the target is **heart disease**, we are not going to use the unsupervised learning: clustering here, since it's not a clustering problem. And we already know it's a binary classification problem, the output will not be a continuous value, it should be having the heart disease or no heart disease. Therefore, it's a classification problem, so we are going to use the classification methods. The prediction models will use classification methods applicable for the results. And possible subtypes of classification methods are decision tree, regression and network, we will discuss the specific algorithms in later steps.

6.Data-mining Algorithm(s)Selection
6.1 Conduct exploratory analysis and discuss
There are many types of classification algorithms, scikit-learn also include many different classification algorithms, which are Linear SVC,Naive Bayes, KNeighbors Classifier, SVC, Ensemble Classifiers, SGD Classifier, kernel approximation.

Since we analysis each of the classification algorithms to match our objectives, we know some of the algorithms are a good choice for us. Like the **Logistic regression** works for the predicted binary variable and will be good at identify the main factors for heart disease. The **Bayesian Network** requires small number of training data for the results, it's fast and good use to predict the disease as well. The **Random Forest** is more accurate than the decision trees, but it's a little bit complex and slow in the prediction. Since we are predicting a binary target, the proper algorithms for us are: **Neural Net**, **SVM** include **Linear SVC** and **NuSVC**, **Decision Tree Classifier**, **Random Forest Classifier** and **Nearest Neighbours**.

Our dataset is not that big, so we include all ten classification algorithms, tthen we compare and select the specific algorithms based on our model accuracy performance and considerations, we know the algorithms also used case by case, so it would be a good choice to decide the algorithms on my real dataset to get the best results.

6.2 Select data mining algorithm(s)

In this iteration, conduct classification analysis using all classification algorithms and compare the results in the figures below. We will keep algorithms with good performance to enable the models with best prediction.

```
feature_names=['age','sex','cp','trestbps','chol','fbs','restecg',
               'thalach','oldpeak','slope','ca','thal']
X= hd_data[feature_names]
y= hd_data['target']
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

```
#Build Models
#Logistic Regression
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression()
logreg.fit(X_train, y_train)
print('Accuracy of Logistic regression classifier on training set: {:.2f}'
     .format(logreg.score(X_train, y_train)))
print('Accuracy of Logistic regression classifier on test set: {:.2f}'
     .format(logreg.score(X_test, y_test)))
```

```
Accuracy of Logistic regression classifier on training set: 0.86
Accuracy of Logistic regression classifier on test set: 0.82
```

```
#Desion Tree
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier().fit(X_train, y_train)
print('Accuracy of Decision Tree classifier on training set: {:.2f}'
     .format(clf.score(X_train, y_train)))
print('Accuracy of Decision Tree classifier on test set: {:.2f}'
     .format(clf.score(X_test, y_test)))
```

```
Accuracy of Decision Tree classifier on training set: 1.00
Accuracy of Decision Tree classifier on test set: 0.97
```

```
#K-Nearest Neighbors
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier()
knn.fit(X_train, y_train)
print('Accuracy of K-NN classifier on training set: {:.2f}'
     .format(knn.score(X_train, y_train)))
print('Accuracy of K-NN classifier on test set: {:.2f}'
     .format(knn.score(X_test, y_test)))
```

```
Accuracy of K-NN classifier on training set: 0.88
Accuracy of K-NN classifier on test set: 0.74
```

```
#Linear Discriminant Analysis
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
lda = LinearDiscriminantAnalysis()
lda.fit(X_train, y_train)
print('Accuracy of LDA classifier on training set: {:.2f}'
     .format(lda.score(X_train, y_train)))
print('Accuracy of LDA classifier on test set: {:.2f}'
     .format(lda.score(X_test, y_test)))
```

```
Accuracy of LDA classifier on training set: 0.87
Accuracy of LDA classifier on test set: 0.83
```

```
#Gaussian Naive Bayes
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb.fit(X_train, y_train)
print('Accuracy of GNB classifier on training set: {:.2f}'
     .format(gnb.score(X_train, y_train)))
print('Accuracy of GNB classifier on test set: {:.2f}'
    .format(gnb.score(X_test, y_test)))
```

```
Accuracy of GNB classifier on training set: 0.86
Accuracy of GNB classifier on test set: 0.83
```

```
#Support Vector Machine
from sklearn.svm import SVC
svm = SVC()
svm.fit(X_train, y_train)
print('Accuracy of SVM classifier on training set: {:.2f}'
     .format(svm.score(X_train, y_train)))
print('Accuracy of SVM classifier on test set: {:.2f}'
    .format(svm.score(X_test, y_test)))
```

```
Accuracy of SVM classifier on training set: 1.00
Accuracy of SVM classifier on test set: 0.97
```

From results above, we can see the **Decision Tree** and **Support Vector Machine** performance best, which are 100% on the training and 97% test sets. Then **Linear Discriminant Analysis, Gaussian Naïve Bayes** performance little bit better than others, we keep those four models for later discussion and choose.

6.3 Select/Build appropriate models and parameter(s)

6.3.1 Parameters of Models

1)Parameter(s)of Decision Tree Model

```
feature_names=['age','sex','cp','trestbps','chol','fbs','restecg',
               'thalach','oldpeak','slope','ca','thal']
X= hd_data[feature_names]
y= hd_data['target']
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

```
#Desion Tree
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier().fit(X_train, y_train)
print('Accuracy of Decision Tree classifier on training set: {:.2f}'
     .format(clf.score(X_train, y_train)))
print('Accuracy of Decision Tree classifier on test set: {:.2f}'
     .format(clf.score(X_test, y_test)))
```

```
Accuracy of Decision Tree classifier on training set: 1.00
Accuracy of Decision Tree classifier on test set: 0.97
```

The first model we used is **Decision Tree**, we set the training set for 70%, and the test set for 30% (refer to 7.1), we set our features into **feature_names**, and our target as **target**. Then we train our dataset, and print out our model accuracy of the training set and the test set.

2)Parameters of Support Vector Machine

```
feature_names=['age','sex','cp','trestbps','chol','fbs','restecg',
               'thalach','oldpeak','slope','ca','thal']
X= hd_data[feature_names]
y= hd_data['target']
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

```
#Support Vector Machine
from sklearn.svm import SVC
svm = SVC()
svm.fit(X_train, y_train)
print('Accuracy of SVM classifier on training set: {:.2f}'
     .format(svm.score(X_train, y_train)))
print('Accuracy of SVM classifier on test set: {:.2f}'
    .format(svm.score(X_test, y_test)))
```

```
Accuracy of SVM classifier on training set: 1.00
Accuracy of SVM classifier on test set: 0.97
```

The second model we used is **Support Vector Machine**, we set the training set for 70%, and the test set for 30% (refer to 7.1), we set our features into **feature_names**, and our target as **target**. Then we train our dataset, and print out our model accuracy of the training set and the test set.

3) Parameters of Linear Discriminant Analysis

```
feature_names=['age','sex','cp','trestbps','chol','fbs','restecg',
               'thalach','oldpeak','slope','ca','thal']
X= hd_data[feature_names]
y= hd_data['target']
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

```
#Linear Discriminant Analysis
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
lda = LinearDiscriminantAnalysis()
lda.fit(X_train, y_train)
print('Accuracy of LDA classifier on training set: {:.2f}'
     .format(lda.score(X_train, y_train)))
print('Accuracy of LDA classifier on test set: {:.2f}'
     .format(lda.score(X_test, y_test)))
```

```
Accuracy of LDA classifier on training set: 0.87
Accuracy of LDA classifier on test set: 0.83
```

The third model we used is **Linear Discriminant Analysis**, we set the training set for 70%, and the test set for 30% (refer to 7.1), we set our features into **feature_names**, and our target as **target**. Then we train our dataset, and print out our model accuracy of the training set and the test set.

4) Parameters of Gaussian Naïve Bayes

```
feature_names=['age','sex','cp','trestbps','chol','fbs','restecg',
               'thalach','oldpeak','slope','ca','thal']
X= hd_data[feature_names]
y= hd_data['target']
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

```
#Gaussian Naive Bayes
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb.fit(X_train, y_train)
print('Accuracy of GNB classifier on training set: {:.2f}'
     .format(gnb.score(X_train, y_train)))
print('Accuracy of GNB classifier on test set: {:.2f}'
    .format(gnb.score(X_test, y_test)))
```

```
Accuracy of GNB classifier on training set: 0.86
Accuracy of GNB classifier on test set: 0.83
```

The fourth model we used is **Gaussian Naïve Bayes**, we set the training set for 70%, and the test set for 30% (refer to 7.1), we set our features into **feature_names**, and our target as **target**. Then we train our dataset, and print out our model accuracy of the training set and the test set.

6.3.2 Choose proper models

By comparing the accuracy of each model to select three models, which are **Decision Tree**, **SVM** and. **Linear Discriminant Analysis.**

```
Accuracy of Decision Tree classifier on training set: 1.00
Accuracy of Decision Tree classifier on test set: 0.97
```

```
Accuracy of SVM classifier on training set: 1.00
Accuracy of SVM classifier on test set: 0.97
```

```
Accuracy of LDA classifier on training set: 0.87
Accuracy of LDA classifier on test set: 0.83

Accuracy of GNB classifier on training set: 0.86
Accuracy of GNB classifier on test set: 0.83
```

7.Data Mining phase

7.1 Test designs created and justified

We import **train_test_split** from **sklearn.model_selection**, then split the data into the size of the training partition and testing partition are 70%/30% which is **test_size=0.3**.When you got small datasets, like number between 100 to 10,000 samples, this 70%/30% training/testing split works well.(Ng.2018).Our dataset between 100-10,000,it's on this condition, and the sets will get more stable prediction for final models.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

7.2 Conduct data-mining

```
Accuracy of Decision Tree classifier on training set: 1.00
Accuracy of Decision Tree classifier on test set: 0.97
```

```
Accuracy of SVM classifier on training set: 1.00
Accuracy of SVM classifier on test set: 0.97
```

From above graphs, the accuracy of **Decision Tree** and **SVM** prediction are 97%, the training set reached 100%, those are perfect.

```
Accuracy of LDA classifier on training set: 0.87
Accuracy of LDA classifier on test set: 0.83
```

This 83% prediction accuracy of **Linear Discriminant Analysis** from above graph, looks good as well.

7.3 Search for patterns

After running all the models, we can see the results from the following table, **Decision Tree** and **SVM** are the best choice, and we are discussing the patterns separately.

Patterns	Training Accuracy/%	Testing Accuracy/%
Decision Tree	100	97
SVM	100	97
Linear Discriminant Analysis	87	83

7.3.1 Decision Tree

```
feature_names=['age','sex','cp','trestbps','chol','fbs','restecg',
               'thalach','oldpeak','slope','ca','thal']
X= hd_data[feature_names]
y= hd_data['target']
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

```
#Desion Tree
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier().fit(X_train, y_train)
print('Accuracy of Decision Tree classifier on training set: {:.2f}'
     .format(clf.score(X_train, y_train)))
print('Accuracy of Decision Tree classifier on test set: {:.2f}'
     .format(clf.score(X_test, y_test)))
```

```
Accuracy of Decision Tree classifier on training set: 1.00
Accuracy of Decision Tree classifier on test set: 0.97
```

The first model we used is **Decision Tree**, we set the training set for 70%, and the test set for 30% (refer to 7.1), we set our features into **feature_names**, and our target as **target**. Then we train our dataset, and print out our model accuracy of the training set and the test set.

The Decision Tree performance best, the accuracy of the training set reached 100%, and the accuracy of the test set also reached 97%.

7.3.2 Support Vector Machine Model

```
feature_names=['age','sex','cp','trestbps','chol','fbs','restecg',
               'thalach','oldpeak','slope','ca','thal']
X= hd_data[feature_names]
y= hd_data['target']
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

```
#Support Vector Machine
from sklearn.svm import SVC
svm = SVC()
svm.fit(X_train, y_train)
print('Accuracy of SVM classifier on training set: {:.2f}'
     .format(svm.score(X_train, y_train)))
print('Accuracy of SVM classifier on test set: {:.2f}'
    .format(svm.score(X_test, y_test)))
```

```
Accuracy of SVM classifier on training set: 1.00
Accuracy of SVM classifier on test set: 0.97
```

The second model we used is **SVM Model**, we set the training set for 70%, and the test set for 30% (refer to 7.1), we set our features into **feature_names**, and our target as **target**. Then we train our dataset, and print out our model accuracy of the training set and the test set.

The SVM performance also very good, the accuracy of the training set reached 100%, and the accuracy of the test set is 97%.

7.3.3 Linear Discriminant Analysis

```
feature_names=['age','sex','cp','trestbps','chol','fbs','restecg',
               'thalach','oldpeak','slope','ca','thal']
X= hd_data[feature_names]
y= hd_data['target']
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

```
#Linear Discriminant Analysis
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
lda = LinearDiscriminantAnalysis()
lda.fit(X_train, y_train)
print('Accuracy of LDA classifier on training set: {:.2f}'
     .format(lda.score(X_train, y_train)))
print('Accuracy of LDA classifier on test set: {:.2f}'
     .format(lda.score(X_test, y_test)))
```

```
Accuracy of LDA classifier on training set: 0.87
Accuracy of LDA classifier on test set: 0.83
```

The third model we used is **Linear Discriminant Analysis**, we set the training set for 70%, and the test set for 30% (refer to 7.1), we set our features into **feature_names**, and our target as **target**. Then we train our dataset, and print out our model accuracy of the training set and the test set.

The Linear Discriminant Analysis, the accuracy of the training set is 87%, and the accuracy of the test set is 83%.

Patterns	Training Accuracy/%	Testing Accuracy/%
Decision Tree	100	97
SVM	100	97
Linear Discriminant Analysis	87	83

Decision Tree **Decision** Tree**Linear Discriminant AnalysisDecision** Tree

8.Interpretion

8.1 Study and discuss the mined patterns

In this study, we have collected the dataset, after compared the similar types of the datasets to make sure the one we collected is qualified for our data-mining use and match our goals. Then we explored the dataset, formatted the dataset. Then during clean data step, we write code to deal with the outliers and extreme values problem. After done all the steps to make sure our dataset is prepared ready and qualified.

Since we know our data mining is a classification problem, so we choose to use the classification algorithms to help us mined the patterns. The whole selection process is

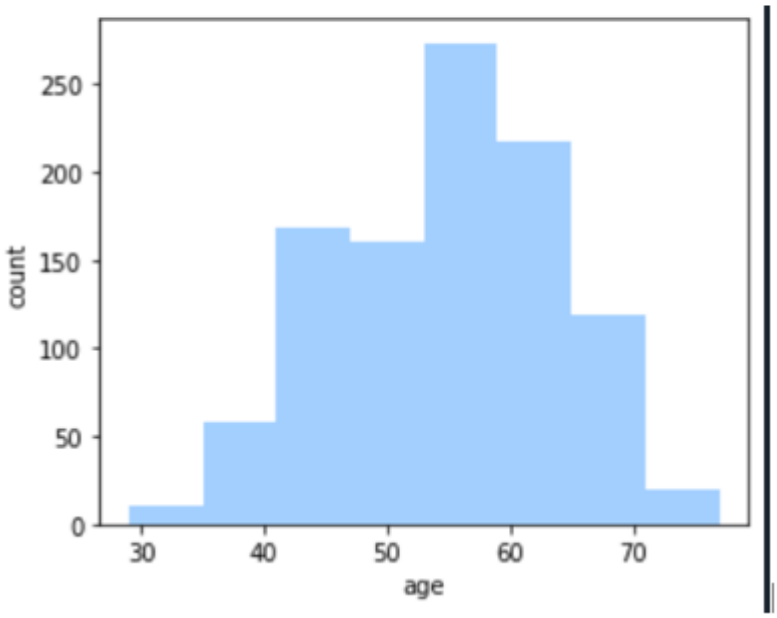
dynamic process, we need consider our objectives and data-mining goals, and our dataset is small, so how those algorithms performed, trained to a good fit to use to predict. After compared and several models' results, we use **Decision Tree**, **SVM** and. **Linear Discriminant Analysis.** With the same dataset, we can compare the patterns mined from those three algorithms. We discussed previous before; the **Decision Tree** and **SVM** have the highest accuracy. Thus, we would interpret it in 8.3. The **Linear Discriminant Analysis** also get a good accuracy, we also interpreted it in the later section 8.3.

From the results, we strongly think the mined patterns are meaningful to us, all the model accuracy is reached above 80%. We will discuss detailed in the later steps.

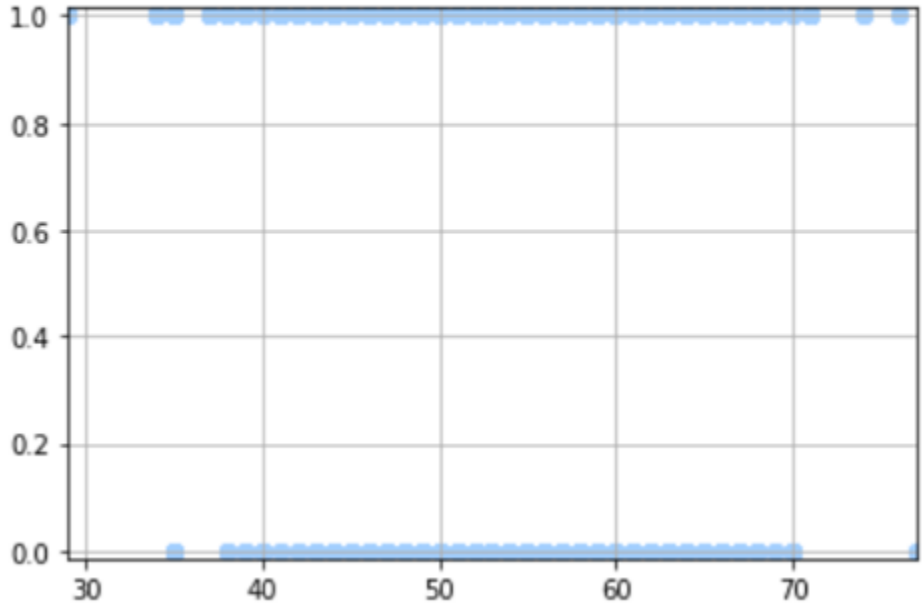
8.2 Visualise the data, results, models and patterns

The visualise graphs attached below, and we would like to discuss more on Part 8.3.

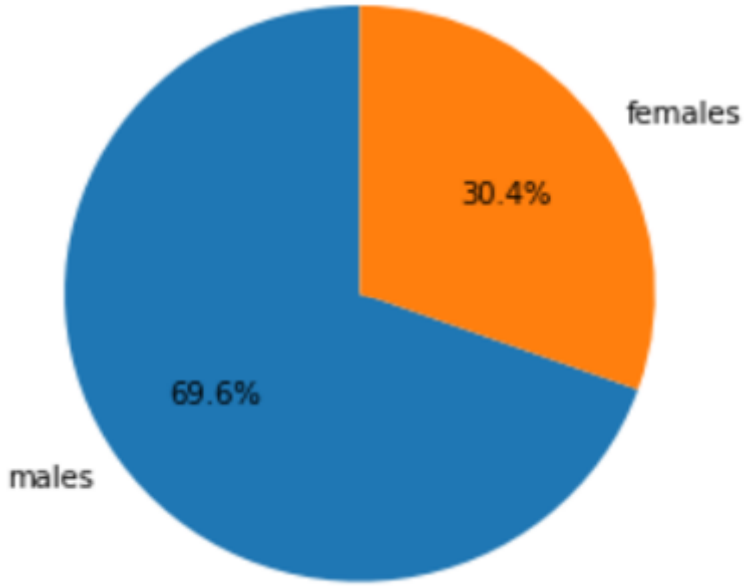
```
fig,ax = plt.subplots(figsize=(5,4))
sns.distplot(hd_data['age'],
             hist_kws={"alpha":1,"color":"#a2cffe"},
             kde=False, bins=8)
ax = ax.set(ylabel='count', xlabel='age')
plt.show()
```



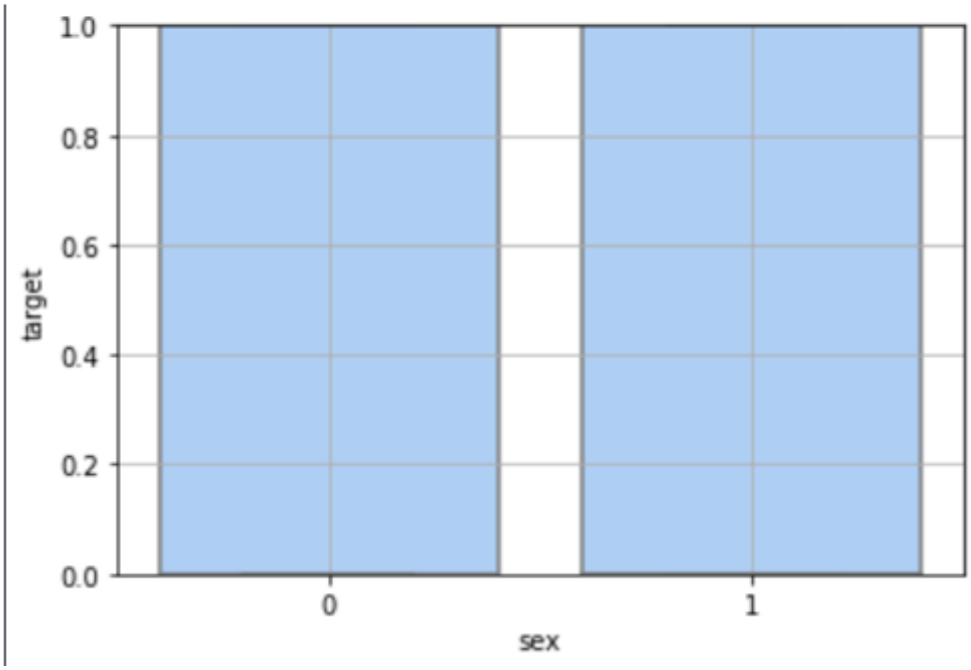
```
plt.scatter(x='age',y='target',data=hd_data,color='#a2cffe')
plt.autoscale(tight=True)
plt.grid()
plt.show()
```



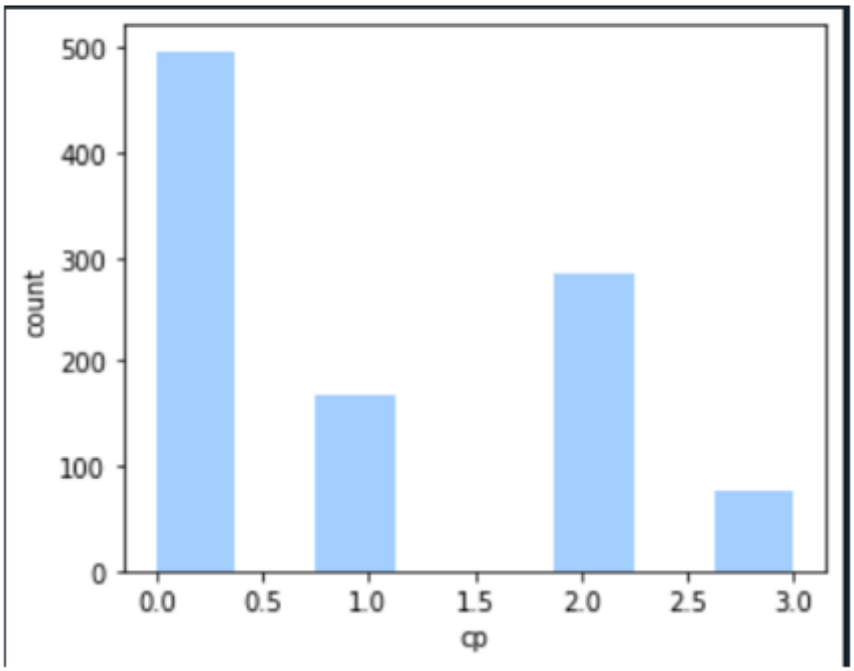

```
fig,ax = plt.subplots(figsize=(4,4))
plt.pie(hd_data['sex'].value_counts().tolist(),
        labels=['males','females'],
        autopct='%1.1f%%',startangle=90)
axis = plt.axis('equal')
```



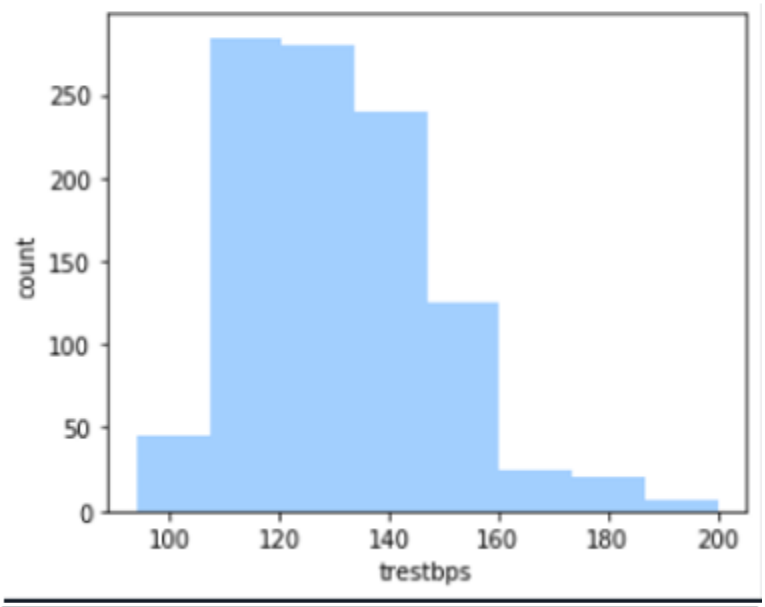
```
sns.boxplot(x='sex',y='target',data=hd_data,color='#a2cffe')
plt.autoscale(tight=True)
plt.grid()
plt.show()
```



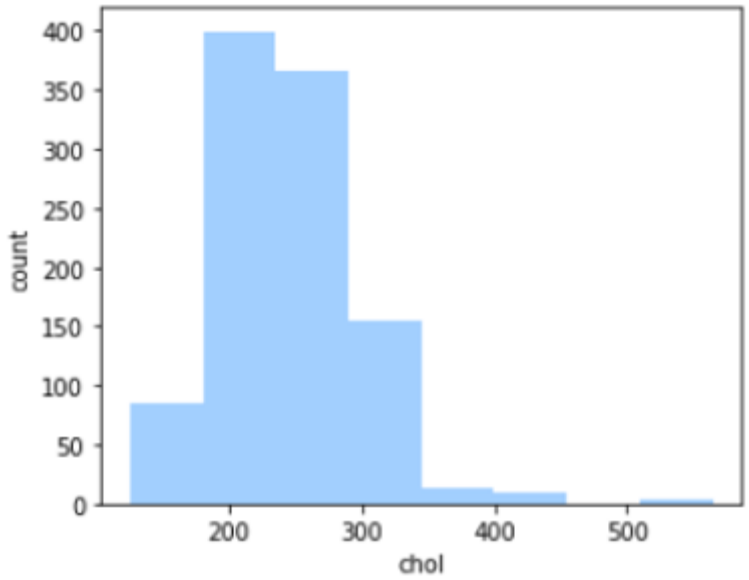

```
fig,ax = plt.subplots(figsize=(5,4))
sns.distplot(hd_data['cp'],
             hist_kws={"alpha":1,"color":"#a2cffe"},
             kde=False, bins=8)
ax = ax.set(ylabel='count', xlabel='cp')
```



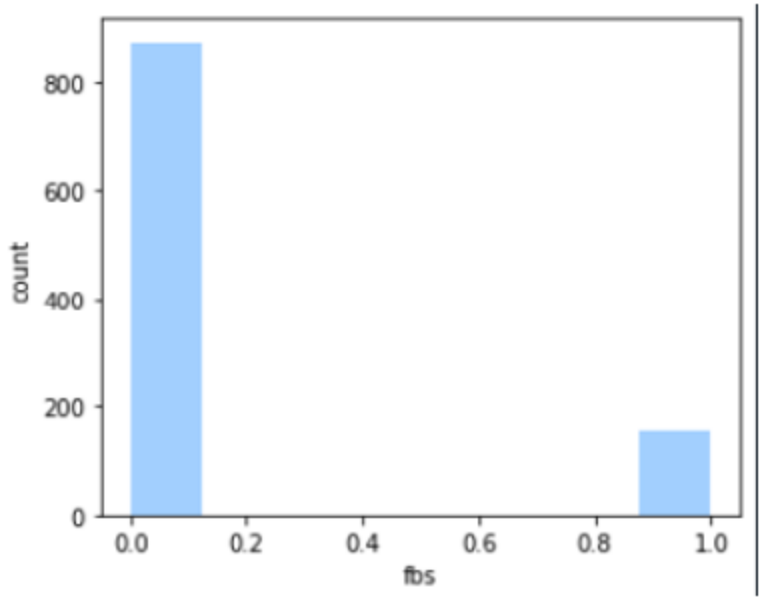
```
fig,ax = plt.subplots(figsize=(5,4))
sns.distplot(hd_data['trestbps'],
             hist_kws={"alpha":1,"color":"#a2cffe"},
             kde=False, bins=8)
ax = ax.set(ylabel='count', xlabel='trestbps')
```



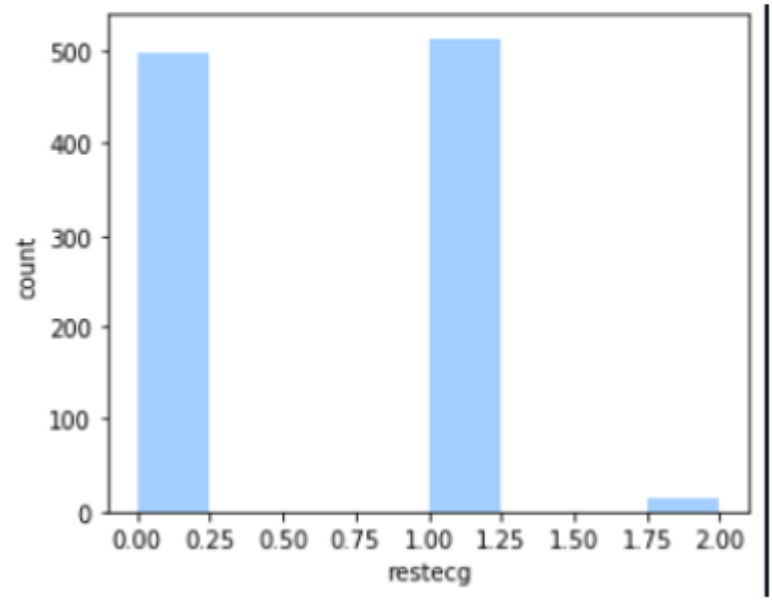

```
fig,ax = plt.subplots(figsize=(5,4))
sns.distplot(hd_data['chol'],
             hist_kws={"alpha":1,"color":"#a2cffe"},
             kde=False, bins=8)
ax = ax.set(ylabel='count', xlabel='chol')
```



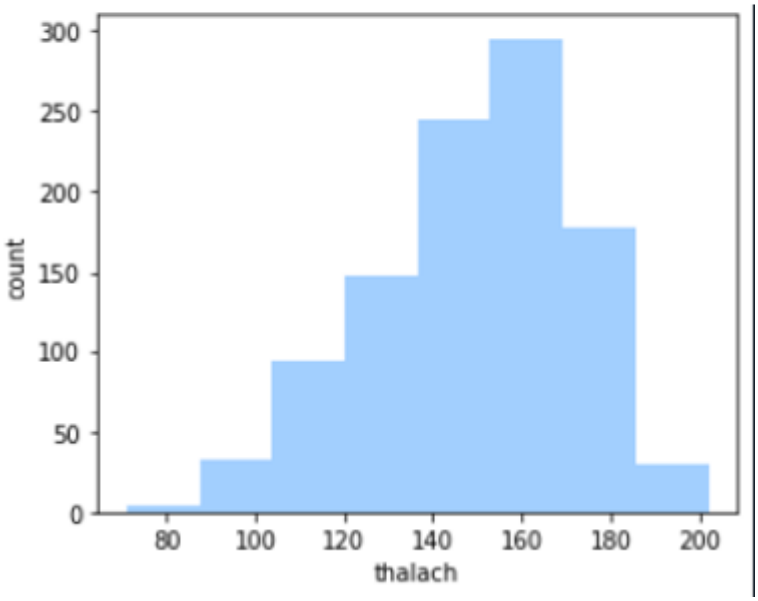
```
fig,ax = plt.subplots(figsize=(5,4))
sns.distplot(hd_data['fbs'],
             hist_kws={"alpha":1,"color":"#a2cffe"},
             kde=False, bins=8)
ax = ax.set(ylabel='count', xlabel='fbs')
```



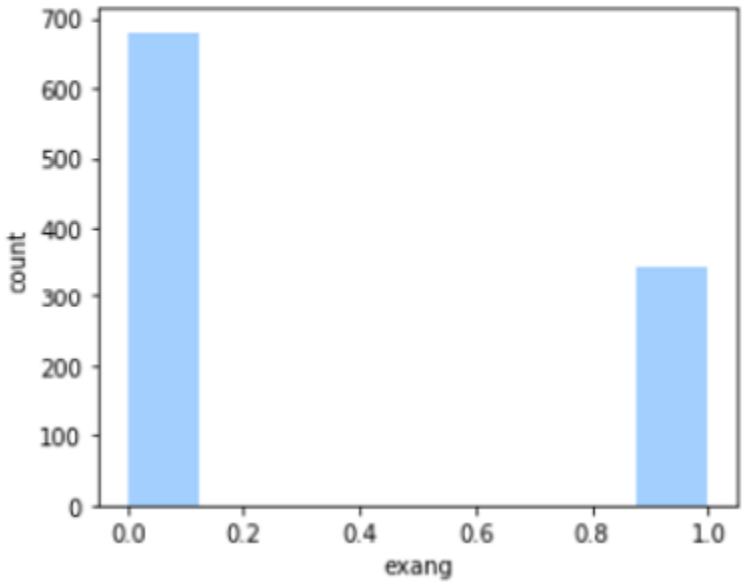

```
fig,ax = plt.subplots(figsize=(5,4))
sns.distplot(hd_data['restecg'],
             hist_kws={"alpha":1,"color":"#a2cffe"},
             kde=False, bins=8)
ax = ax.set(ylabel='count', xlabel='restecg')
```



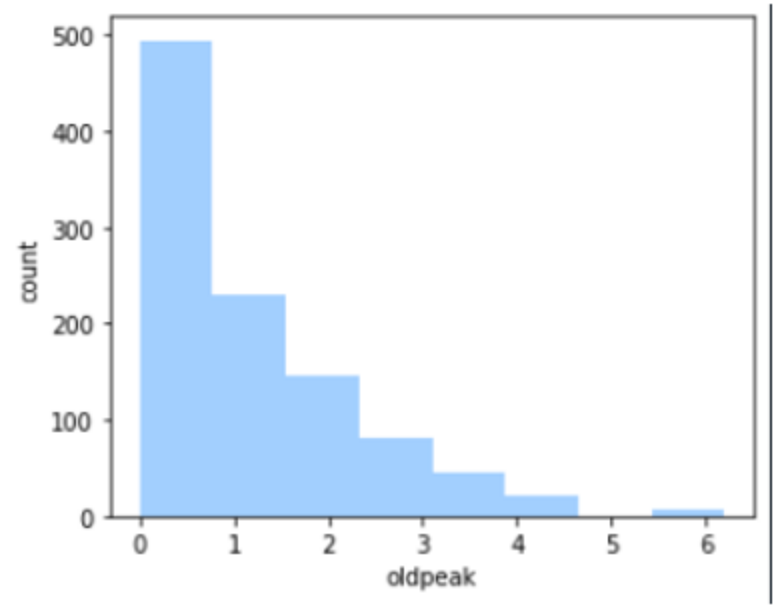
```
fig,ax = plt.subplots(figsize=(5,4))
sns.distplot(hd_data['thalach'],
             hist_kws={"alpha":1,"color":"#a2cffe"},
             kde=False, bins=8)
ax = ax.set(ylabel='count', xlabel='thalach')
```



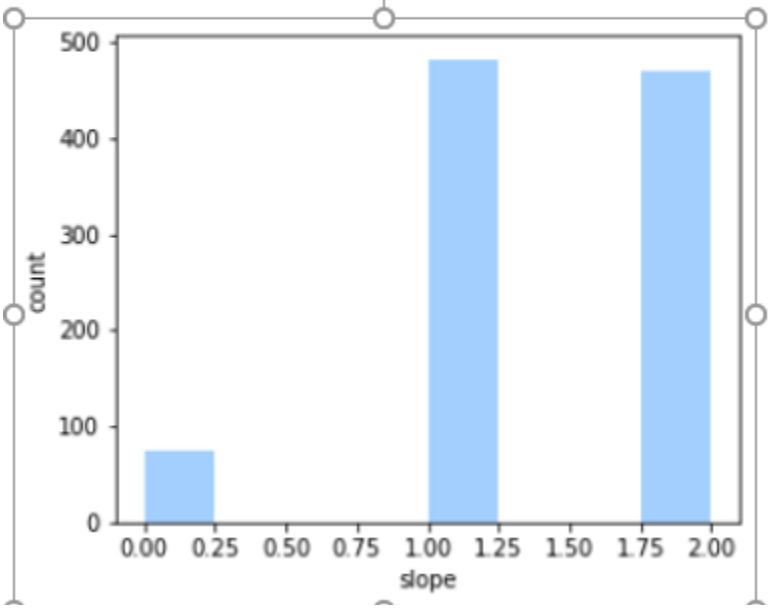

```
fig,ax = plt.subplots(figsize=(5,4))
sns.distplot(hd_data['exang'],
             hist_kws={"alpha":1,"color":"#a2cffe"},
             kde=False, bins=8)
ax = ax.set(ylabel='count', xlabel='exang')
```



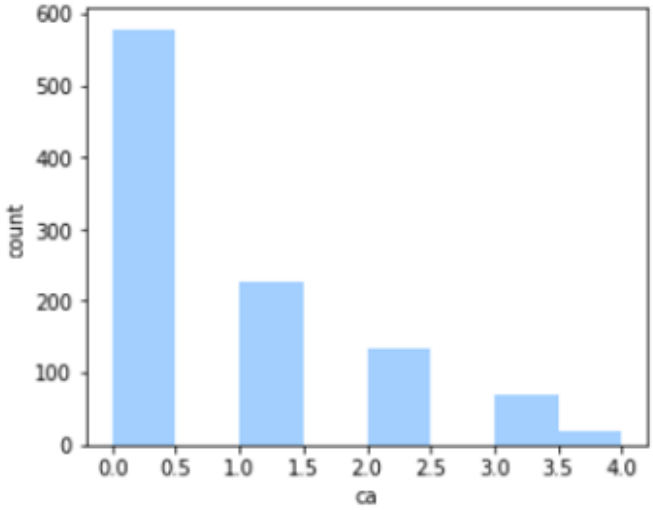
```
fig,ax = plt.subplots(figsize=(5,4))
sns.distplot(hd_data['oldpeak'],
             hist_kws={"alpha":1,"color":"#a2cffe"},
             kde=False, bins=8)
ax = ax.set(ylabel='count', xlabel='oldpeak')
```



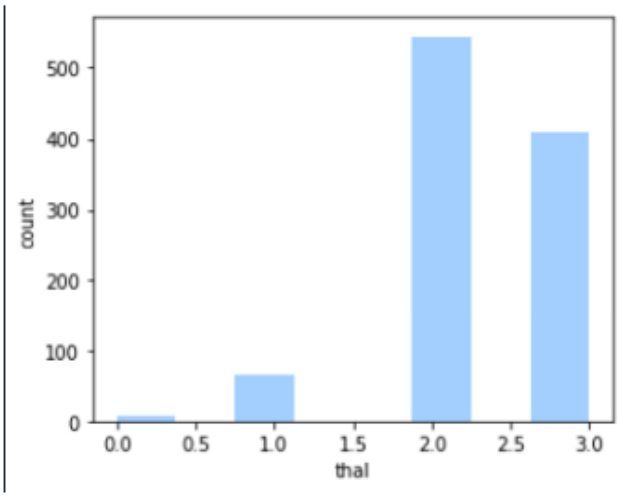

```
fig,ax = plt.subplots(figsize=(5,4))
sns.distplot(hd_data['slope'],
             hist_kws={"alpha":1,"color":"#a2cffe"},
             kde=False, bins=8)
ax = ax.set(ylabel='count', xlabel='slope')
```



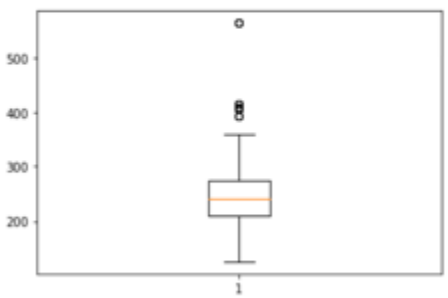
```
fig,ax = plt.subplots(figsize=(5,4))
sns.distplot(hd_data['ca'],
             hist_kws={"alpha":1,"color":"#a2cffe"},
             kde=False, bins=8)
ax = ax.set(ylabel='count', xlabel='ca')
```



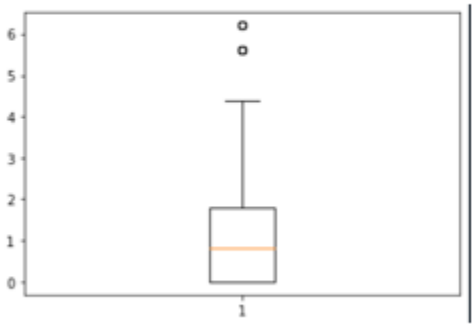

```
fig,ax = plt.subplots(figsize=(5,4))
sns.distplot(hd_data['thal'],
             hist_kws={"alpha":1,"color":"#a2cffe"},
             kde=False, bins=8)
ax = ax.set(ylabel='count', xlabel='thal')
```




```
plt.boxplot(hd_data["chol"])
plt.show()
```

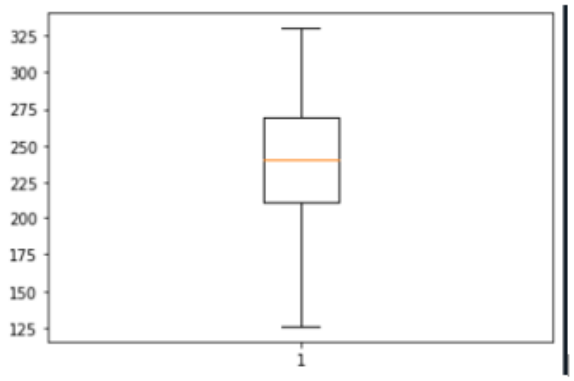



```
plt.boxplot(hd_data["oldpeak"])
plt.show()
```



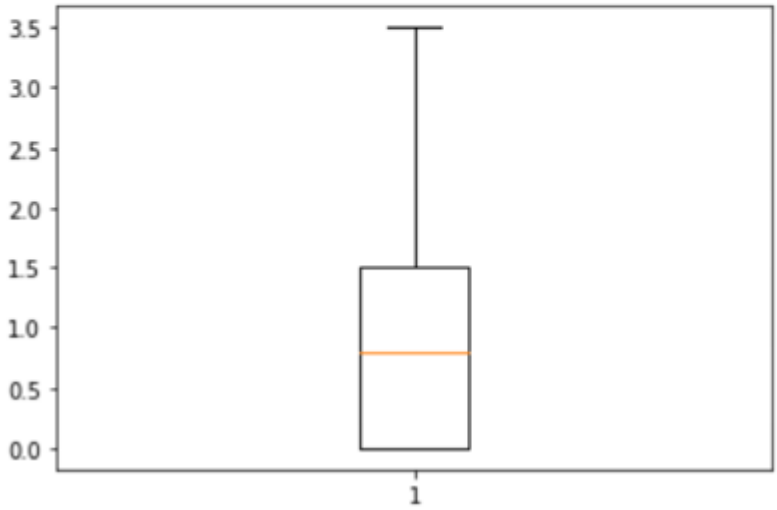
For the **chol** feature:

```
print(hd_data['chol'].quantile(0.50))
print(hd_data['chol'].quantile(0.95))
hd_data['chol'] = np.where(hd_data['chol'] > 330, 240, hd_data['chol'])
print(hd_data.describe())

plt.boxplot(hd_data["chol"])
plt.show()
```

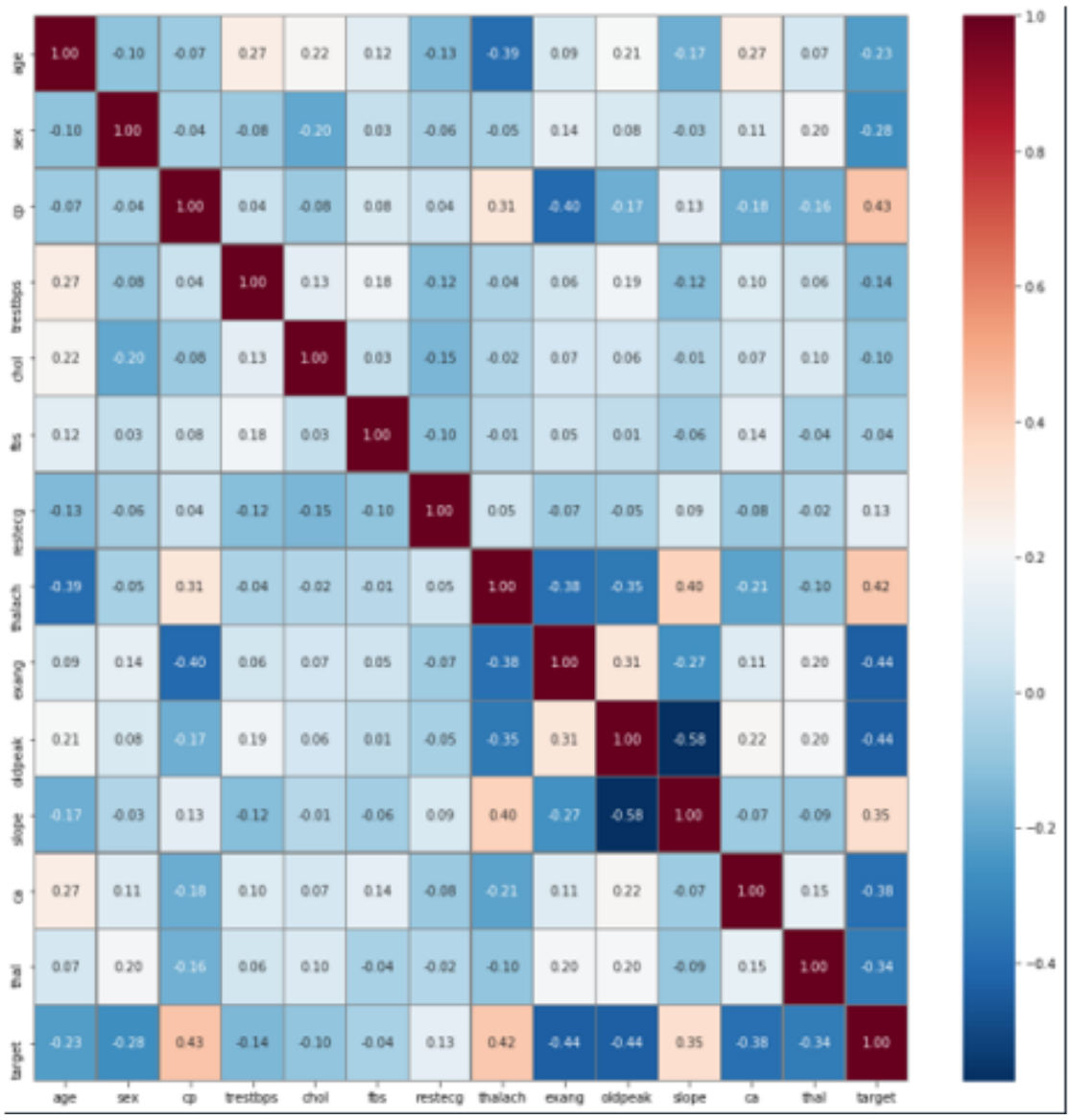


For the **oldpeak** feature:

```
print(hd_data['oldpeak'].quantile(0.50))
print(hd_data['oldpeak'].quantile(0.95))
hd_data['oldpeak'] = np.where(hd_data['oldpeak'] > 3.5, 0.8, hd_data['oldpeak'])
print(hd_data.describe())

plt.boxplot(hd_data["oldpeak"])
plt.show()
```



```
plt.figure(figsize=(15,15))
sns.heatmap(hd_data.corr(),annot=True,fmt=".2f",linewidths=0.3,linecolor="grey",cmap="RdBu_r")
```



8.3 Interpret the results, models and patterns

Follow the part of 8.2, we are explaining the results

Combined the distribution of the dataset, the feature importance check and the results from models implement, we can find:

1) The dataset age is more likely around 45-70, but there is no such difference to our target.
2) There are more males than females in our dataset, and not the main factor to our target.
3) The most unimportant features are **exang**, **oldpeak** and **ca**.
4) The key features to our target are **cp**, **thalach** and **slope**, so those three features are the key factors to heart disease. Furthermore, there is a strong connection with the thalach and slope, those two together may lead to bad situation to the heart disease.

8.4 Assess and evaluate results, models and patterns

```
#Desion Tree
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier().fit(X_train, y_train)
print('Accuracy of Decision Tree classifier on training set: {:.2f}'
     .format(clf.score(X_train, y_train)))
print('Accuracy of Decision Tree classifier on test set: {:.2f}'
     .format(clf.score(X_test, y_test)))
```

```
Accuracy of Decision Tree classifier on training set: 1.00
Accuracy of Decision Tree classifier on test set: 0.97
```

```
#Support Vector Machine
from sklearn.svm import SVC
svm = SVC()
svm.fit(X_train, y_train)
print('Accuracy of SVM classifier on training set: {:.2f}'
     .format(svm.score(X_train, y_train)))
print('Accuracy of SVM classifier on test set: {:.2f}'
    .format(svm.score(X_test, y_test)))
```

```
Accuracy of SVM classifier on training set: 1.00
Accuracy of SVM classifier on test set: 0.97
```

```
#Linear Discriminant Analysis
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
lda = LinearDiscriminantAnalysis()
lda.fit(X_train, y_train)
print('Accuracy of LDA classifier on training set: {:.2f}'
     .format(lda.score(X_train, y_train)))
print('Accuracy of LDA classifier on test set: {:.2f}'
     .format(lda.score(X_test, y_test)))
```

```
Accuracy of LDA classifier on training set: 0.87
Accuracy of LDA classifier on test set: 0.83
```

Based on our above discussion, one of our data-mining objectives is accuracy of models more than 80%, and those three models reached more than 80%, so it evaluates the study is success in this point. The Decision Tree and SVM performance best, the accuracy of the training set reached 100%, and the accuracy of the test set also reached 97%.

For the second data mining objective is to find the important features, then we can find the main features which are **cp**, **thalach** and **slope**. Therefore, according to the results of the data mining models, we have found our risk factors increased the possibility of getting the heart disease.

8.5 Multiple iterations

8.5.1 Business Understanding

The aim of this step is to find a business problem as a topic and know more related background about it. After done some study, I found the heart disease is an interesting topic. Then I set the data-mining goals and business objectives in this step based on the heart disease background. And I noticed that the important to match the data-mining goals to our business objectives. Thus, we planned time, resources, risk and other related parts to the project on our above consideration through the whole dynamic process.

8.5.2 Data understanding

We expected to find more than two qualified datasets from open source websites. After long time compare and search for the datasets, we just collected one qualified dataset from Kaggle website which matches our data-mining goals very well. So, we decided just to use this only one dataset to use.

To ensure the data quality for the data mining, we checked the dataset format for each feature, explored the data features by visualize each feature. We tried different graphs; finial choose the proper one. Through the exploring the dataset, we made the preliminary predict in our mind.

8.5.3 Data Preparation

3.1 Select the data

```
import findspark
findspark.init('/home/ubuntu/spark-2.1.1-bin-hadoop2.7')
import pyspark
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName('heart').getOrCreate()
```

```
df = spark.read.load('./heart.csv', format= "csv", header='true')
```

```
heart_data.head()
```

3.2 Clean the data

3.2.1 Missing Data:

```
hd_data.isnull().any()
```

3.2.2 Outliers and Extremes values:

```
print(hd_data.describe())
```

```
plt.boxplot(hd_data["chol"])
plt.show()
```

```
plt.boxplot(hd_data["oldpeak"])
plt.show()
```

For the chol feature:

For the oldpeak feature:

3.3 Construct the data

```
hd_data['log_chol']=np.log(hd_data['chol'])
hd_data['log_trestbps']= np.log(hd_data['trestbps'])
hd_data['log_thalach']=np.log(hd_data['thalach'])
print(hd_data.head())
print(hd_data.describe())
```

3.4 Integrate various datasets

In this study, we just use one dataset, so there is no merging data needed for the project.

3.5 Format the data

We have only one dataset, and this dataset used in here is relatively small, already sorting before modelling (refer to explore the data), so it's not necessary to reformat the data in this step.

In this step, we imported our dataset, read the dataset head and types to further check the data. Then we also checked the missing value and the outliers and extremes' values, we were handling the outliers and extremes' values in the stage and compared the dataset before and after to see the difference. We also constructed the data, and compared the difference before and after the construct, we can see we get 3 more features in our dataset.

8.5.4 Data Transformation

4.1 Reduce the data

Importance check:

```
plt.figure(figsize=(15,15))
sns.heatmap(hd_data.corr(),annot=True,fmt=".2f",linewidths=0.3,linecolor="grey",cmap="RdBu_r")
```

Remove unimportant features:

```
heart_data.head()
```

```
del hd_data['exang']
hd_data.head()
```

4.2 Project the data

```
hd_data['log_chol']=np.log(hd_data['chol'])
hd_data['log_trestbps']= np.log(hd_data['trestbps'])
hd_data['log_thalach']=np.log(hd_data['thalach'])
print(hd_data.head())
print(hd_data.describe())
```

We checked the feature importance and delete the unimportant feature.
Log transform: The distributions of features are not in a similar range. Hence, we choose 'chol', 'trestbps', 'thalach' features transform to 'log_chol', 'log_trestbps', 'log_thalach' values. and compared the difference before and after the construct, we can see we get 3 more features in our dataset.

8.5.5 Select Data Mining Methods

After compared the supervised and unsupervised methods, since we know our main objective is a binary problem which is match the definition of the classification, so we choose the classification methods to ensure meet our data-mining goals and our business goals.

8.5.6 Select Data Mining Algorithms

```
feature_names=['age','sex','cp','trestbps','chol','fbs','restecg',
               'thalach','oldpeak','slope','ca','thal']
X= hd_data[feature_names]
y= hd_data['target']
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

```
#Build Models
#Logistic Regression
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression()
logreg.fit(X_train, y_train)
print('Accuracy of Logistic regression classifier on training set: {:.2f}'
     .format(logreg.score(X_train, y_train)))
print('Accuracy of Logistic regression classifier on test set: {:.2f}'
     .format(logreg.score(X_test, y_test)))
```

```
Accuracy of Logistic regression classifier on training set: 0.86
Accuracy of Logistic regression classifier on test set: 0.82
```

```
#Desion Tree
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier().fit(X_train, y_train)
print('Accuracy of Decision Tree classifier on training set: {:.2f}'
     .format(clf.score(X_train, y_train)))
print('Accuracy of Decision Tree classifier on test set: {:.2f}'
     .format(clf.score(X_test, y_test)))
```

```
Accuracy of Decision Tree classifier on training set: 1.00
Accuracy of Decision Tree classifier on test set: 0.97
```

```
#K-Nearest Neighbors
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier()
knn.fit(X_train, y_train)
print('Accuracy of K-NN classifier on training set: {:.2f}'
     .format(knn.score(X_train, y_train)))
print('Accuracy of K-NN classifier on test set: {:.2f}'
     .format(knn.score(X_test, y_test)))
```

```
Accuracy of K-NN classifier on training set: 0.88
Accuracy of K-NN classifier on test set: 0.74
```

```
#Linear Discriminant Analysis
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
lda = LinearDiscriminantAnalysis()
lda.fit(X_train, y_train)
print('Accuracy of LDA classifier on training set: {:.2f}'
     .format(lda.score(X_train, y_train)))
print('Accuracy of LDA classifier on test set: {:.2f}'
     .format(lda.score(X_test, y_test)))
```

```
Accuracy of LDA classifier on training set: 0.87
Accuracy of LDA classifier on test set: 0.83
```

```
#Gaussian Naive Bayes
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb.fit(X_train, y_train)
print('Accuracy of GNB classifier on training set: {:.2f}'
     .format(gnb.score(X_train, y_train)))
print('Accuracy of GNB classifier on test set: {:.2f}'
    .format(gnb.score(X_test, y_test)))
```

```
Accuracy of GNB classifier on training set: 0.86
Accuracy of GNB classifier on test set: 0.83
```

```
#Support Vector Machine
from sklearn.svm import SVC
svm = SVC()
svm.fit(X_train, y_train)
print('Accuracy of SVM classifier on training set: {:.2f}'
     .format(svm.score(X_train, y_train)))
print('Accuracy of SVM classifier on test set: {:.2f}'
    .format(svm.score(X_test, y_test)))
```

```
Accuracy of SVM classifier on training set: 1.00
Accuracy of SVM classifier on test set: 0.97
```

Since we decided to use the classification methods, there are several algorithms, we need to select to match our dataset and data-mining goals. Thus, we expected to select the several good performance models, we write several models to find out more accuracy algorithms, then based on that, we check each model to find out three best algorithms. They are **Decision Tree**, **SVM** and. **Linear Discriminant Analysis.** Those models give good accuracy and predictions.

8.5.7 Data Mining

7.3.1 Decision Tree

```
feature_names=['age','sex','cp','trestbps','chol','fbs','restecg',
               'thalach','oldpeak','slope','ca','thal']
X= hd_data[feature_names]
y= hd_data['target']
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

```
#Desion Tree
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier().fit(X_train, y_train)
print('Accuracy of Decision Tree classifier on training set: {:.2f}'
     .format(clf.score(X_train, y_train)))
print('Accuracy of Decision Tree classifier on test set: {:.2f}'
     .format(clf.score(X_test, y_test)))
```

```
Accuracy of Decision Tree classifier on training set: 1.00
Accuracy of Decision Tree classifier on test set: 0.97
```

7.3.2 Support Vector Machine Model

```
feature_names=['age','sex','cp','trestbps','chol','fbs','restecg',
               'thalach','oldpeak','slope','ca','thal']
X= hd_data[feature_names]
y= hd_data['target']
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

```
#Support Vector Machine
from sklearn.svm import SVC
svm = SVC()
svm.fit(X_train, y_train)
print('Accuracy of SVM classifier on training set: {:.2f}'
     .format(svm.score(X_train, y_train)))
print('Accuracy of SVM classifier on test set: {:.2f}'
    .format(svm.score(X_test, y_test)))
```

```
Accuracy of SVM classifier on training set: 1.00
Accuracy of SVM classifier on test set: 0.97
```

7.3.3 Linear Discriminant Analysis

```
feature_names=['age','sex','cp','trestbps','chol','fbs','restecg',
               'thalach','oldpeak','slope','ca','thal']
X= hd_data[feature_names]
y= hd_data['target']
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

```
#Linear Discriminant Analysis
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
lda = LinearDiscriminantAnalysis()
lda.fit(X_train, y_train)
print('Accuracy of LDA classifier on training set: {:.2f}'
     .format(lda.score(X_train, y_train)))
print('Accuracy of LDA classifier on test set: {:.2f}'
     .format(lda.score(X_test, y_test)))
```

```
Accuracy of LDA classifier on training set: 0.87
Accuracy of LDA classifier on test set: 0.83
```

Here, we build and run the models. We split the data into training (70%) set and testing set (30%). After mining the data and compared each of the results, we can see all the data mining accuracy reached more than 80% and it's achieved our objective goals. And we have found our risk factors in step3 as well. The results of the data-mining perfect match our data-mining goals and business objectives.

REFERENCES

IBM. (n.d.). IBM SPSS Modeler CRISP-DM Guide.
Ng, A. (2018). Machine Learning Yearning (Draft Vers). Retrieved from https://d2wvfoqc9gyqzf.cloudfront.net/content/uploads/2018/09/Ng-MLY01-13.pdf

Disclaimer

"I acknowledge that the submitted work is my own original work in accordance with the University of Auckland guidelines and policies on academic integrity and copyright. (See: https://www.auckland.ac.nz/en/students/forms-policies-and-guidelines/student-policies-and-guidelines/academic-integrity-copyright.htmlLinks to an external site.).
I also acknowledge that I have appropriate permission to use the data that I have utilised in this project. (For example, if the data belongs to an organisation and the data has not been published in the public domain then the data must be approved by the rights holder.) This includes permission to upload the data file to Canvas. The University of Auckland bears no responsibility for the student's misuse of data."