EXPLORATORY DATA ANALYSIS on HEART DISEASE UCI DATASET WITH PYTHON

1.Abstract

This project's main aim is to understand how the features play a part in prevalence of heart disease by using UCI Data Set (UCI Machine Learning)[3] and determine if the importance of features change in genders. Besides, the most appropriate predictive model will be chosen for future work for predictive data analysis.

2.Introduction

Cardiovascular disease is the top global cause of death, in order of the total number of lives lost worldwide. According to World Health Organization (WHO), ischemic heart disease and stroke are responsible for approximately 16% and 11% of the world's total deaths, respectively. In 2019, around 15 million people died from these two diseases worldwide.[1] Therefore it is essential to analyze the factors contributing to the development of heart disease to reduce deaths across the world.

In this dataset, there are both categorical, ordinal and numerical independent variables such as characteristics of patients, underlying health conditions and heart diagnosing tests results which are considered to have impact on developing heart disease.
To provide the best descriptive and exploratory analysis and to choose the most appropriate predictive model , cross industry standard process (CRISP_DM) [2] will be used, which contains six phases:

1- Business Understanding: Introduction
2- Data Understanding: Exploration of data
3- Data Preparation: Pre-processing of data
4- Modeling: Apply Model
5- Evaluation: Conclusion
6- Deployment:

3.Objective and Scope

In this project:

For descriptive and exploratory data analysis, Python programming language will be used.
will be applied to categorical attributes and Kendall's rank coefficient (nonlinear) will be applied to numerical attributes.

3.1. Steps:

1. *Data will be imported using requests library.*
2. *Data wrangling (cleaning, formatting, constructing), selecting, filtering data, handling missing data, data aggregation and group operations will be done by using panda's library.*
3. *To determine the relationships between the dependent feature and independent features* Cramer coefficient correlation *will be applied to categorical attributes and Kendall's rank coefficient (nonlinear) will be applied to numerical attributes.*
4. *Bar charts, pivot tables will be used to visualize the data by using matplotlib and seaborn libraries.*
5. *Cross validation will be used to choose best classification model for predictive analysis as a future work.*
6. *The report will be provided as a pdf and python scripts will be* loaded to GitHub web address as a jupyter notebook.

3.2. Questions to be explored.

- Are there outlier values in this dataset?
- Is there any feature to be ignored because of weak association with labeled attribute?
- Which classification models can be used for this data for predictive analysis as a future work?
- Which features have the most significant impact on causing heart disease?
- Is there any difference in the importance of attributes in gender groups?

4.Data source

Heart Disease UCI Data Set is taken from UCI Machine Learning Repository to the V.A. Medical Center, Long Beach and Cleveland Clinic Foundation. It is from UCI Learning Repository to the Cleveland database consisting of 76 attributes of 303 patients. The names and social security numbers of the patients were replaced with dummy values in this dataset. [3] The principal investigators who are responsible for the data collection are: Hungarian Institute of Cardiology, Budapest: Andras Janosi, M.D., University Hospital, Zurich, Switzerland: William Steinbrunn, M.D., University Hospital, Basel, Switzerland: Matthias Pfisterer, M.D., V.A. Medical Centre, Long Beach and Cleveland Clinic Foundation: Robert Detrano, M.D., PhD[3]

There are 303 patients and 76 attributes in the Heart Disease UCI dataset. The analysis will be made by 14 features referred to as used according to previously published experiments.[3] .

4.1 Attribute names and terminology:

Age	Age in years
Sex	1 =male ,0 = female
Chest pain (Angina) Angina is chest pain that happens when blood arteries supplying blood to the heart are blocked.	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 hospital
Chol	Serum cholesterol levels in mg/dl
Fbs (Fasting blood sugar level)	Fbs>120 mg/dl (1 =true; 0 = false)
Restecg (Resting electrocardiography 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 (Max heart rate achieved)	By convention, the maximum predicted heart rate is calculated as 220 (210 for women) minus the patient's age. [4]
Exang (exercise induced chest pain)	(1 = yes; 0 = no)
Old peak	ST depression induced by exercise relative to rest
Slope (The slope of the peak exercise ST segment) ST segment depression (horizontal or down sloping) is the most reliable indicator of exercise-induced ischemia which a restriction in blood supply to tissues[4]	--Value1:upsloping --Value2:flat -- Value 3: down sloping

Ca (C-arm fluoroscopy results) C-arm is a medical imaging device	Number of blocked major vessels supplying blood (0-3) colored by fluoroscopy

Thal (Thalassemia is a blood disorder .)	(3 = normal; 6 = fixed defect; 7 = reversable defect)
Target: Diagnosis of heart disease (angiographic disease status)	Value 0: < 50% diameter narrowing: absence of illness Value 1: > 50% diameter narrowing: presence of illness

5.Data Preparation

5.1 Data Loading from Web

In [1]:
```
#Pulling dataset from 'https://archive.ics.uci.edu/ml/machine-learning-databases/heart-disease/ '
import requests
file = requests.get("https://archive.ics.uci.edu/ml/machine-learning-databases/heart-disease/processed.cleveland.data")
```

In [2]:
```
heart_dis = file.text
heart_dis
```

Out[2]:
```
'63.0,1.0,1.0,145.0,233.0,1.0,2.0,150.0,0.0,2.3,3.0,0.0,6.0,0\n67.0,1.0,4.0,160.0,286.0,0.0,2.0,108.0,1.0,1.5,2.0,
3.0,3.0,2\n67.0,1.0,4.0,120.0,229.0,0.0,2.0,129.0,1.0,2.6,2.0,2.0,7.0,1\n37.0,1.0,3.0,130.0,250.0,0.0,0.0,187.0,0.
0,3.5,3.0,0.0,3.0,0\n41.0,0.0,2.0,130.0,204.0,0.0,2.0,172.0,0.0,1.4,1.0,0.0,3.0,0\n56.0,1.0,2.0,120.0,236.0,0.0,0.
0,178.0,0.0,0.8,1.0,0.0,3.0,0\n62.0,0.0,4.0,140.0,268.0,0.0,2.0,160.0,0.0,3.6,3.0,2.0,3.0,3\n57.0,0.0,4.0,120.0,35
4.0,0.0,0.0,163.0,1.0,0.6,1.0,0.0,3.0,0\n63.0,1.0,4.0,130.0,254.0,0.0,2.0,147.0,0.0,1.4,2.0,1.0,7.0,2\n53.0,1.0,4.
```

In [3]:
```
#read file and assign it to variable
#give write permission to the opened file
#write the contents of the heart_dis variable to the file, then close the file by appyling the changes
hd = open("heart_dis.csv", "w")
hd.write(heart_dis)
hd.close()
```

In [4]:
```
import pandas as pd
```

In [5]:
```
my_data = pd.read_csv('C:\\Users\\serta\\Desktop\\my_data\\heart_dis.csv',header=None)
my_data.head(2)
```

Out[5]:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	63.0	1.0	1.0	145.0	233.0	1.0	2.0	150.0	0.0	2.3	3.0	0.0	6.0	0
1	67.0	1.0	4.0	160.0	286.0	0.0	2.0	108.0	1.0	1.5	2.0	3.0	3.0	2

5.2. Data Wrangling

Naming columns :

In [7]:
```
my_data.columns =['Age','Sex','Chestpain','RestingBloodPressure','Chol','FastingBloodSugar','RestECG','MaxHeartRate',
                  'Exang','Oldpeak','Slope','Ca','Thal','Health Condition']
my_data.head(2)
```

Out[7]:

	Age	Sex	Chestpain	RestingBloodPressure	Chol	FastingBloodSugar	RestECG	MaxHeartRate	Exang	Oldpeak	Slope	Ca	Thal	Health Condition
0	63.0	1.0	1.0	145.0	233.0	1.0	2.0	150.0	0.0	2.3	3.0	0.0	6.0	0
1	67.0	1.0	4.0	160.0	286.0	0.0	2.0	108.0	1.0	1.5	2.0	3.0	3.0	2

Adding new_column :

In [8]:
```
my_data.insert(0, 'New_ID', range(1, 1 + len(my_data)))
```

In [9]:
```
my_data.head(2)
```

Out[9]:

	New_ID	Age	Sex	Chestpain	RestingBloodPressure	Chol	FastingBloodSugar	RestECG	MaxHeartRate	Exang	Oldpeak	Slope	Ca	Thal	Health Condition
0	1	63.0	1.0	1.0	145.0	233.0	1.0	2.0	150.0	0.0	2.3	3.0	0.0	6.0	0
1	2	67.0	1.0	4.0	160.0	286.0	0.0	2.0	108.0	1.0	1.5	2.0	3.0	3.0	2

Creating Target Column from Health Condition Values :

'0' MEANS ABSENCE OF HEART DISEASE, '1,2,3,4' VALUES REPRESENT PRESENCE OF THE DISEASE

In [10]:
```
my_data['Health Condition'].unique()
```

Out[10]:
```
array([0, 2, 1, 3, 4], dtype=int64)
```

In [11]:
```
my_data['Target'] = [0 if x == 0 else 1 for x in my_data['Health Condition']]
my_data.head(2)
```

Out[11]:

	New_ID	Age	Sex	Chestpain	RestingBloodPressure	Chol	FastingBloodSugar	RestECG	MaxHeartRate	Exang	Oldpeak	Slope	Ca	Thal	Health Condition	Tar
0	1	63.0	1.0	1.0	145.0	233.0	1.0	2.0	150.0	0.0	2.3	3.0	0.0	6.0	0	
1	2	67.0	1.0	4.0	160.0	286.0	0.0	2.0	108.0	1.0	1.5	2.0	3.0	3.0	2	

In [12]:
```
#Deleting column
my_data.drop(['Health Condition'],axis=1,inplace = True)
my_data.head(1)
```

Out[12]:

	New_ID	Age	Sex	Chestpain	RestingBloodPressure	Chol	FastingBloodSugar	RestECG	MaxHeartRate	Exang	Oldpeak	Slope	Ca	Thal	Target
0	1	63.0	1.0	1.0	145.0	233.0	1.0	2.0	150.0	0.0	2.3	3.0	0.0	6.0	0

In [13]:
```
#concise summary of a DataFrame
my_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 15 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   New_ID                303 non-null    int32
 1   Age                   303 non-null    float64
 2   Sex                   303 non-null    float64
 3   Chestpain             303 non-null    float64
 4   RestingBloodPressure  303 non-null    float64
 5   Chol                  303 non-null    float64
 6   FastingBloodSugar     303 non-null    float64
 7   RestECG               303 non-null    float64
 8   MaxHeartRate          303 non-null    float64
 9   Exang                 303 non-null    float64
 10  Oldpeak               303 non-null    float64
 11  Slope                 303 non-null    float64
 12  Ca                    303 non-null    object
 13  Thal                  303 non-null    object
 14  Target                303 non-null    int64
dtypes: float64(11), int32(1), int64(1), object(2)
memory usage: 34.4+ KB
```

Checking duplicated rows

In [14]:
```
duplicate_rows = my_data[my_data.duplicated()]
duplicate_rows.shape
```

Out[14]: (0, 15)

According to info method there is no null value but to be sure unique values will be checked.

In [15]:
```
my_data['Chestpain'].unique()
```

Out[15]: array([1., 4., 3., 2.])

In [16]:
```
my_data['Ca'].unique()
```

Out[16]: array(['0.0', '3.0', '2.0', '1.0', '?'], dtype=object)

In [17]:
```
my_data['Thal'].unique()
```

Out[17]: array(['6.0', '3.0', '7.0', '?'], dtype=object)

In [18]:
```
my_data['Sex'].unique()
```

Out[18]: array([1., 0.])

Finding number of '?' values in Ca and Thal attributes:

In [27]:
```
my_data['Thal'].value_counts()
```

Out[27]:
```
3.0    166
7.0    117
6.0     18
?        2
Name: Thal, dtype: int64
```

In [28]:
```
my_data['Ca'].value_counts()
```

Out[28]:
```
0.0    176
1.0     65
2.0     38
3.0     20
?        4
Name: Ca, dtype: int64
```

5.3. Handling Missing Data

DETERMINING THE PERCENTAGE OF MISSING ROWS IN DATAFRAME :

In [29]:
```
missing_data1 = len(my_data[my_data['Ca']=='?'])
missing_data1
missing_data2 = len(my_data[my_data['Thal']=='?'])
missing_data2
sum_missing = missing_data1 + missing_data2
total_columns =303
percent_missing_data = (sum_missing/total_columns) * 100
percent_missing_data
```

Out[29]: 1.9801980198019802

Because the percentage of missing data is lower than 2%, missing data will be evaluated to discover if missing data is completely at random or not. If missing values are MCAR, values will be deleted to avoid bias. Otherwise, data imputation will be made. [5]

For determining the correlation between categorical variables ,Cramer's V measurement will be used which is based on a nominal variation of Pearson's Chi-square test.[6]

For determination of association between Ca and continuous variables Kendall's rank coefficient (nonlinear) measurement will be used because as an input Ca attribute's datatype is ordinal and output variables are numerical.[7], [8] , [9]

5.3.1..Determination of the datatypes of attributes :

```
my_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 15 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   New_ID                303 non-null    int32
 1   Age                   303 non-null    float64
 2   Sex                   303 non-null    float64
 3   Chestpain             303 non-null    float64
 4   RestingBloodPressure  303 non-null    float64
 5   Chol                  303 non-null    float64
 6   FastingBloodSugar     303 non-null    float64
 7   RestECG               303 non-null    float64
 8   MaxHeartRate          303 non-null    float64
 9   Exang                 303 non-null    float64
 10  Oldpeak               303 non-null    float64
 11  Slope                 303 non-null    float64
 12  Ca                    303 non-null    object
 13  Thal                  303 non-null    object
 14  Target                303 non-null    int64
dtypes: float64(11), int32(1), int64(1), object(2)
memory usage: 34.4+ KB
```

```
# IN ORDER TO HAVE VALID RESULTS: Datatypes of categorical and numerical attributes will be corrected.
```

```
category_features =['Chestpain','FastingBloodSugar','RestECG','Exang','Thal','Sex','Target','Ca','New_ID','Slope']
my_data[category_features] = my_data[category_features].astype('category')
num_features = ['Age','RestingBloodPressure','Chol','MaxHeartRate']
my_data[num_features]= my_data[num_features].astype('int')

my_data.info()
```

5.3.2 Crammer Coefficient Correlation between categorical attributes:

Ca and Thal attribute's association with categorical features will be evaluated.

Reference : https://researchpy.readthedocs.io/en/latest/crosstab_documentation.html

In [32]:
```
#to determine the correlations between 'Thal' and 'Target ' attribute
```

In [33]:
```
import researchpy
croostab = researchpy.crosstab(my_data['Thal'],my_data['Target'],test = 'chi-square')
croostab
```

Out[33]:
```
(       Target
 Target      0    1  All
 Thal
 3.0       129   37  166
 6.0         6   12   18
 7.0        28   89  117
 ?           1    1    2
 All       164  139  303,                Chi-square test  results
 0  Pearson Chi-square ( 3.0) =   83.2957
 1                    p-value =    0.0000
 2                 Cramer's V =    0.5243)
```

There is a strong association between Thal and Target attribute.Thus, missing values are not completely at random,so missing values will be imputated with mode value of '3' of Thal attribute.

In [34]:
```
my_data['Thal'] = my_data['Thal'].replace('6.0', 6)
my_data['Thal'] = my_data['Thal'].replace('3.0', 3)
my_data['Thal'] = my_data['Thal'].replace('7.0', 7)
my_data['Thal'] = my_data['Thal'].replace('?', 3)
```

In [43]:
```
researchpy.crosstab(my_data['Ca'],my_data['Slope'],test = 'chi-square')
```

Out[43]:
```
(      Slope
 Slope   1.0  2.0 3.0  All
 Ca
 0.0      92   69  15  176
 1.0      28   35   2   65
 2.0      15   21   2   38
 3.0       5   13   2   20
 ?         2    2   0    4
 All     142  140  21  303,                 Chi-square test  results
 0  Pearson Chi-square ( 8.0) =   11.4314
 1                    p-value =    0.1784
 2                 Cramer's V =    0.1373)
```

In [44]:
```
researchpy.crosstab(my_data['Ca'],my_data['Thal'],test = 'chi-square')
```

Out[44]:
```
(     Thal
 Thal    3   6    7  All
 Ca
 0.0   117   8   51  176
 1.0    29   4   32   65
 2.0    14   4   20   38
 3.0     6   2   12   20
 ?       2   0    2    4
 All   168  18  117  303,                 Chi-square test  results
 0  Pearson Chi-square ( 8.0) =   23.2532
 1                    p-value =    0.0031
 2                 Cramer's V =    0.1959)
```

```
    Cramer's V coefficient association between Ca and categorical attributes

Ca-Sex : 0.13  ,  Ca-Chestpain : 0.185,  Ca-FastingBloodSugar : 0.15  , Ca- RestECg : 0.12 ,
Ca-Exang :0.21 ,   Ca-Slope :0.14   ,    Ca-Thal : 0.2
```

There is no strong association between categorical values and ‘Ca’ attribute, so the relationship between numeric attributes and ‘Ca’ attribute will be evaluated with Kendall’s rank coefficient measurement.

5.3.3 Kendall's R Coefficient Correlation between 'Ca' and numerical attributes:

```
#Kendall's rank coefficient (nonlinear) measurement
from scipy import stats
x1 = my_data['Ca']
x2 = my_data['MaxHeartRate']
tau, p_value = stats.kendalltau(x1, x2)
tau
```

```
-0.24062153072076026
```

```
x1 = my_data['Ca']
x2 = my_data['Oldpeak']
tau, p_value = stats.kendalltau(x1, x2)
tau
```

```
0.20614659737112176
```

```
x1 = my_data['Ca']
x2 = my_data['Age']
tau, p_value = stats.kendalltau(x1, x2)
tau
```

```
0.33405707876236157
```

There is a strong association between 'Ca' and 'Age' attributes, so missing values will be imputed with mode value of Ca attribute.

```
In [52]: #lst = list(mask.index)
```

```
In [53]: #my_data.drop(index =lst,inplace = True)
```

```
In [54]: my_data['Ca'] = my_data['Ca'].replace('0.0', 0)
         my_data['Ca'] = my_data['Ca'].replace('3.0', 3)
         my_data['Ca'] = my_data['Ca'].replace('2.0', 2)
         my_data['Ca'] = my_data['Ca'].replace('?', 0)
         my_data['Ca'] = my_data['Ca'].replace('1.0', 1)
```

6. Exploratory Data Analysis and Visualizations

For numerical values, Group by () function is used to splitting data and describe () function is used to calculate the descriptive statistics of the data.
The Pivot table () method and count aggregation function are used to determine the number of patients having each unique values of categorical columns.
For visualization, matplotlib and seaborn libraries are used. [10]

6.1 Determination of statistical data of numerical attributes by using group by () and describe () method.

In [57]:
```
#statistical data of sick and non-sick patients;
targets = my_data.groupby('Target')
targets.describe().stack()
#The stack() function is used to reshape the dataframe from columns to index.
```

Out[57]:

Target		Age	RestingBloodPressure	Chol	MaxHeartRate	Oldpeak
0	count	164.000000	164.000000	164.000000	164.000000	164.000000
	mean	52.585366	129.250000	242.640244	158.378049	0.586585
	std	9.511957	16.204739	53.456580	19.199080	0.781734
	min	29.000000	94.000000	126.000000	96.000000	0.000000
	25%	44.750000	120.000000	208.750000	148.750000	0.000000
	50%	52.000000	130.000000	234.500000	161.000000	0.200000
	75%	59.000000	140.000000	267.250000	172.000000	1.025000
	max	76.000000	180.000000	564.000000	202.000000	4.200000
1	count	139.000000	139.000000	139.000000	139.000000	139.000000
	mean	56.625899	134.568345	251.474820	139.258993	1.574101
	std	7.938416	18.769019	49.486835	22.593233	1.302580
	min	35.000000	100.000000	131.000000	71.000000	0.000000
	25%	52.000000	120.000000	217.500000	125.000000	0.550000
	50%	58.000000	130.000000	249.000000	142.000000	1.400000
	75%	62.000000	145.000000	283.500000	156.500000	2.500000
	max	77.000000	200.000000	409.000000	195.000000	6.200000

```
##statistical data of sick and non-sick patients in two genders;
targets1 = my_data.groupby(['Target','Sex'])
targets1.describe().stack()
```

Target	Sex		Age	RestingBloodPressure	Chol	MaxHeartRate	Oldpeak
0	0.0	count	72.000000	72.000000	72.000000	72.000000	72.000000
		mean	54.555556	128.736111	256.750000	154.027778	0.554167
		std	10.265337	16.536765	66.216195	19.252929	0.647843
		min	34.000000	94.000000	141.000000	96.000000	0.000000
		25%	46.000000	119.500000	210.750000	146.750000	0.000000
		50%	54.000000	130.000000	249.000000	159.000000	0.250000
		75%	63.250000	140.000000	289.500000	167.250000	1.025000
		max	76.000000	180.000000	564.000000	192.000000	2.600000
	1.0	count	92.000000	92.000000	92.000000	92.000000	92.000000
		mean	51.043478	129.652174	231.597826	161.782609	0.611957
		std	8.623904	16.019517	37.640888	18.556611	0.875031
		min	29.000000	94.000000	126.000000	105.000000	0.000000
		25%	44.000000	120.000000	206.500000	150.000000	0.000000
		50%	52.000000	130.000000	229.500000	163.000000	0.150000
		75%	57.000000	140.000000	250.750000	175.750000	1.050000
		max	70.000000	178.000000	325.000000	202.000000	4.200000

1	0.0	count	25.000000	25.000000	25.000000	25.000000	25.000000
		mean	59.080000	146.600000	276.160000	143.160000	1.768000
		std	4.864155	21.118712	59.884389	20.175645	1.617024
		min	43.000000	108.000000	164.000000	97.000000	0.000000
		25%	57.000000	130.000000	236.000000	133.000000	0.200000
		50%	60.000000	140.000000	268.000000	146.000000	1.400000
		75%	62.000000	158.000000	307.000000	157.000000	2.800000
		max	66.000000	200.000000	409.000000	174.000000	6.200000
	1.0	count	114.000000	114.000000	114.000000	114.000000	114.000000
		mean	56.087719	131.929825	246.061404	138.403509	1.531579
		std	8.385155	17.217361	45.439113	23.083043	1.227438
		min	35.000000	100.000000	131.000000	71.000000	0.000000
		25%	51.000000	120.000000	212.000000	125.000000	0.600000
		50%	57.500000	130.000000	247.500000	141.000000	1.400000
		75%	61.000000	140.000000	282.000000	156.000000	2.400000
		max	77.000000	192.000000	353.000000	195.000000	5.600000

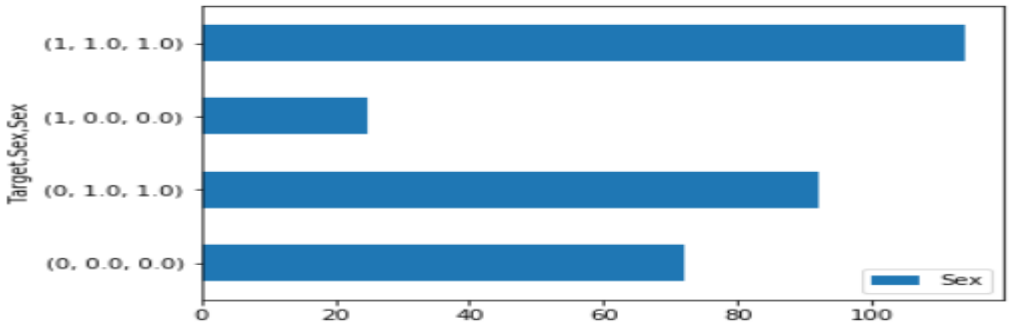
6.2 Attribute analysis: SEX

- There are 303 patients of 206 male and 97 females; although the number of male patients is almost twice as female, the number of male patients is nearly four times the number of females in target 1. (1 =male ,0 = female)

In [68]:
```
import matplotlib.pyplot as plt
#the code below is necessary to see the visualization in the actual notebook itself,inline with the text.
%matplotlib inline
import seaborn as sns
```

```
targets['Sex'].value_counts().plot(kind = 'barh',legend=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x2493cba4eb8>
```




```
targets = my_data.groupby(['Target','Sex'])
targets.size()
```

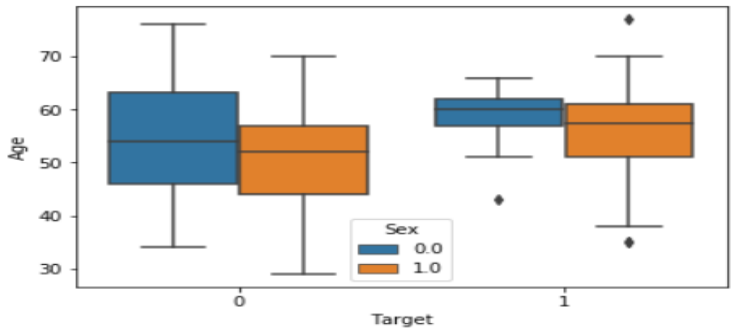
```
Target  Sex
0       0.0      72
        1.0      92
1       0.0      25
        1.0     114
dtype: int64
```

6.3 Attribute analysis: AGE

- Age ranges from 29 to 77 with an average of 54.
- Patients having heart disease (hd) are by majority between 57 to 60 years with an average of around 57.
- In target 0, the average age is nearly 53.
- Male patients are most likely to develop hd in younger age than females. The male patients with hd have an average age of 56 and female patients has an average age of 59.

```
#https://seaborn.pydata.org/generated/seaborn.boxplot.html#seaborn.boxplot
sns.boxplot(x='Target',y='Age',hue ='Sex',data= my_data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x145d27272e8>
```



As seen in boxplot diagram, there are outlier values in patient's age. The outlier value is the value that is over 1.5 times the interquartile, which is the difference between the third quartile and first quartile.

To get accurate mean values of age in target groups and genders, IQR method (Interquartile range method) will be applied to data.[11]

In [72]:
```
#determination of outlier values
def outliers(x):
    q1 = my_data[x].quantile(0.25)
    q3 = my_data[x].quantile(0.75)
    Interquar=q3-q1
    print(q1)
    print(q3)
    print(Interquar)
    Lower_outlier_value = q1-(1.5*Interquar)
    Upper_outlier_value = q3+(1.5*Interquar)
    print(Lower_outlier_value, Upper_outlier_value)
```

In [73]:
```
outliers('Age')
```

```
48.0
61.0
13.0
28.5 80.5
```

In [74]:
```
my_data_age = my_data[my_data['Age'] > 80.5]
my_data_age.shape
#there is no upper outlier value
```

Out[74]:
```
(0, 15)
```

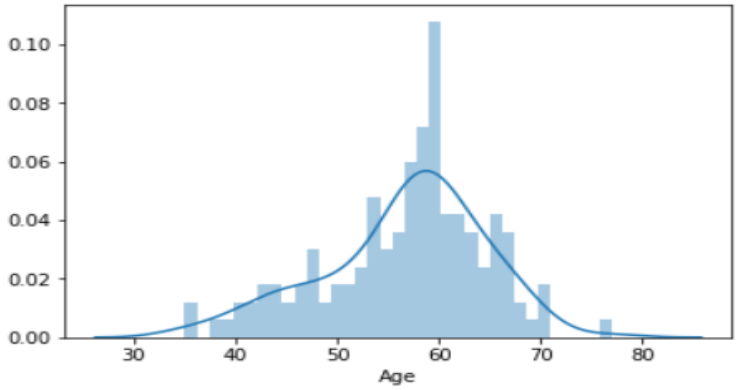
In [75]:
```
my_data_age = my_data[my_data['Age'] < 28.5]
my_data_age.shape
##there is no lower outlier value
```

Out[75]:
```
(0, 15)
```

- As a result, there is no outlier value in Age.

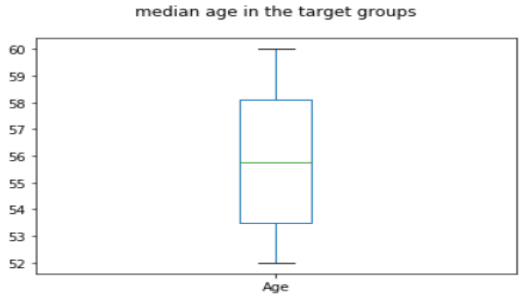
```
#https://seaborn.pydata.org/generated/seaborn.boxplot.html#seaborn.boxplot
import seaborn as sns
sns.distplot(my_data[my_data['Target']==1]['Age'],kde=True,bins=35)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x2493edaec18>
```



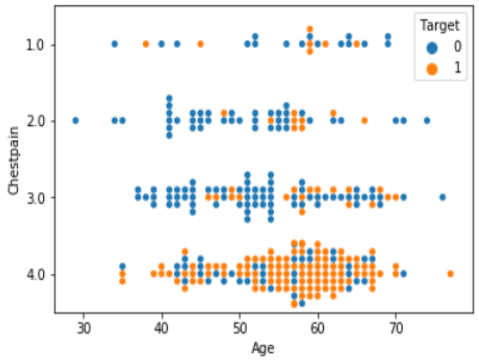

```
targets['Age'].median().plot(kind = 'box' ,legend = True,title =('median age in the target groups\n'))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x2493eebed30>
```



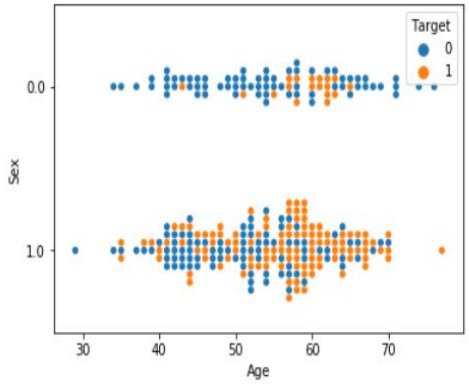

```
sns.swarmplot(x='Age', y='Chestpain', data=my_data, hue='Target')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1f6134d3fd0>
```




```
sns.swarmplot(x='Age', y='Sex', data=my_data, hue='Target')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x23385e6b898>
```



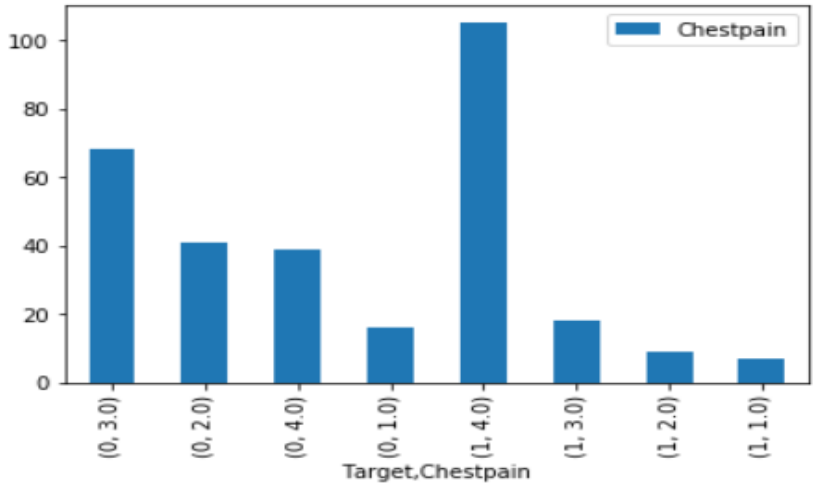
6.4. Attribute analysis: CHESTPAIN

Angina is a chest pain that happens when blood arteries supplying blood to the heart are blocked. (Value 1: typical angina, Value 2: atypical angina, Value 3: non-anginal pain, Value 4: asymptomatic)

- Around 75% (105/139) of people have asymptomatic chest pain in target1, whereas only almost 25% (39/164) of people have asymptomatic chest pain in target 0.
- The percentage of sick women with asymptomatic chest pain is almost 3.5 times higher than the percentage of non-sick women having asymptomatic chest pain, in comparison the percentage of it is nearly 3.2 times higher in men than in those who are not sick. Thus, asymptomatic chest pain has slightly higher correlation in female patients.

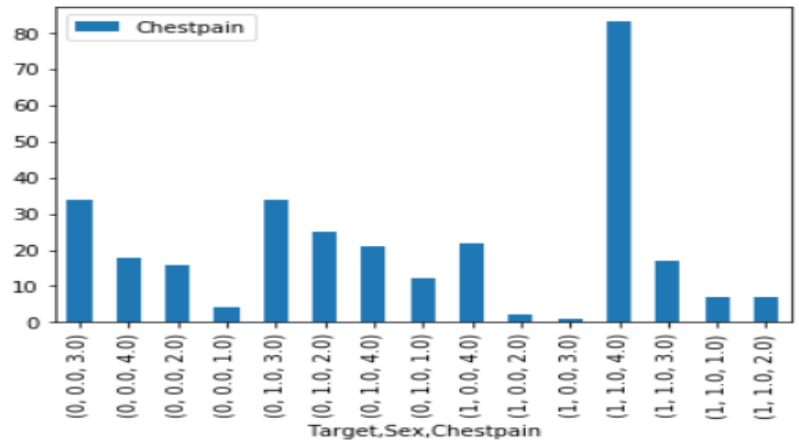
```
targets = my_data.groupby('Target')
targets['Chestpain'].value_counts().plot(kind = 'bar',legend=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x2493ef3a080>
```




```
targets1 = my_data.groupby(['Target','Sex'])
targets1['Chestpain'].value_counts().plot(kind = 'bar',legend=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x2493eea3908>
```



- To determine the number of patients in both gender and target groups; target_genders () and chestpain_targets_genders () functions are written.

```
#to determine the number of patients of both gender and target groups
def target_genders(a,b) :
    x = my_data['Target']== a
    y = my_data['Sex']== b
    return my_data[x&y].shape[0]
number_female_targetone= target_genders(1,0)
number_female_targetzero= target_genders(0,0)
number_male_targetone= target_genders(1,1)
number_male_targetzero= target_genders(0,1)
#Determining the change in the incidence rate of chestpain 4 among the target groups
def chestpain_targets_genders(a,b,c) :
    x = my_data['Target']== a
    y = my_data['Sex']== b
    z = my_data['Chestpain']== c
    return my_data[x&y&z].shape[0]
chestpain_female_target0_percentage = chestpain_targets_genders(0,0,4)/number_female_targetzero*100
chestpain_female_target0_percentage
chestpain_female_target1_percentage = chestpain_targets_genders(1,0,4)/number_female_targetone*100
chestpain_female_target1_percentage
chestpain_male_target0_percentage = chestpain_targets_genders(0,1,4)/number_male_targetzero*100
chestpain_male_target0_percentage
chestpain_male_target1_percentage = chestpain_targets_genders(1,1,4)/number_male_targetone*100
chestpain_male_target1_percentage
#percentage ratio of two target groups with same gender
x = chestpain_female_target1_percentage/chestpain_female_target0_percentage
y = chestpain_male_target1_percentage/ chestpain_male_target0_percentage
result = [x,y]
result
```

```
[3.52, 3.1896407685881374]
```

```
In [83]: targets1['Chestpain'].value_counts()
```

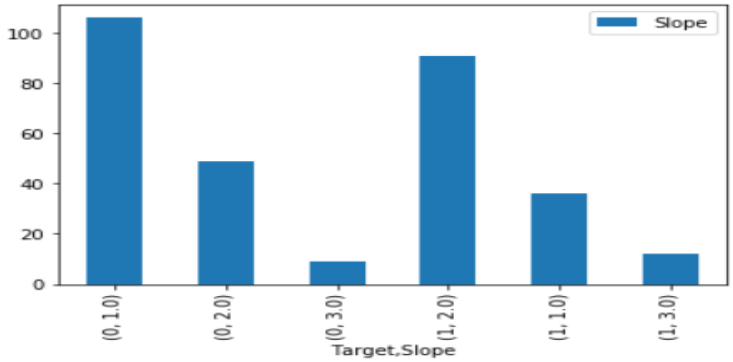
```
Out[83]: Target  Sex  Chestpain
         0       0.0  3.0          34
                      4.0          18
                      2.0          16
                      1.0           4
                 1.0  3.0          34
                      2.0          25
                      4.0          21
                      1.0          12
         1       0.0  4.0          22
                      2.0           2
                      3.0           1
                 1.0  4.0          83
                      3.0          17
                      1.0           7
                      2.0           7
         Name: Chestpain, dtype: int64
```

6.5. Attribute analysis: SLOPE

ST segment depression (horizontal or down sloping) is the most reliable indicator of exercise-induced ischemia (a restriction in blood supply to tissues. (--Value 1: upsloping –Value 2: flat -- Value 3: down sloping)

- For patients in Target 1, the exercise-induced ST segment is mostly flat with a percentage of 65% (91 out of 139 patients), whereas, in Target 0, it is upsloping. (106 out of 164)
- The percentage of slope flat values seen in women with illness is 2.1 times higher than women in target 0, while this value is 2.52 in men. Therefore, having flat slope has slightly more impact on males than females.
- The percentage of down sloping values seen in women with illness is 4.32 times higher than women in target 0, while this value is 1.04 in men. Therefore, having flat slope has slightly more impact on females than males.
- Having down sloping slope has more impact and females than males.

```
In [84]: targets['Slope'].value_counts().plot(kind = 'bar',legend=True)
Out[84]: <matplotlib.axes._subplots.AxesSubplot at 0x29c1bf76438>
```

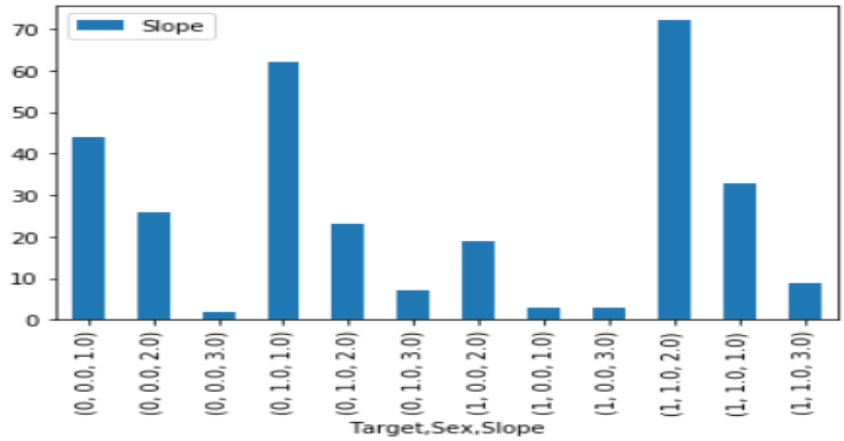



```
slope_female_target0_percentage = slope_targets_genders(0,0,3)/number_female_targetzero*100
slope_female_target0_percentage
slope_female_target1_percentage = slope_targets_genders(1,0,3)/number_female_targetone*100
slope_female_target1_percentage
slope_male_target0_percentage = slope_targets_genders(0,1,3)/number_male_targetzero*100
slope_male_target0_percentage
slope_male_target1_percentage = slope_targets_genders(1,1,3)/number_male_targetone*100
slope_male_target1_percentage
#percentage difference with same gender between two target groups
x = slope_female_target1_percentage/slope_female_target0_percentage
y = slope_male_target1_percentage/ slope_male_target0_percentage
result = [x,y]
result
```

```
[4.32, 1.0375939849624058]
```

```
targets1 = my_data.groupby(['Target','Sex'])
targets1['Slope'].value_counts().plot(kind = 'bar',legend=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x2493f0c4860>
```



6.6. Attribute analysis: THAL

Thalassemia is a blood disorder. (3 = normal; 6 = fixed defect; 7 = reversable defect)

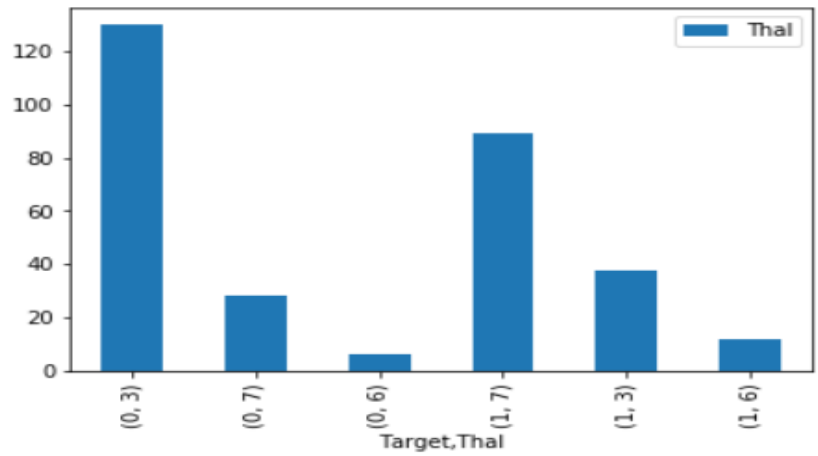
- Around 64% people with heart disease have reversible defect type of thalassemia, represented by number 7. In comparison, it is around 17% in target 0.
- While only 2 out of 72 women who are not sick had a thal 7 value, this value increased from 13 out of 25 in sick women, in comparison 26 out of 92 men who were not sick had a thal value of 7 and value was observed in 76 out of 114 patients in sick men. Thus, reversible defect type of thalassemia has a more impact on females than men having heart disease.

```
my_data.pivot_table(values ='New_ID' ,index = ['Target','Thal','Sex'],aggfunc = 'count')
```

			New_ID
Target	**Thal**	**Sex**	
0	**3**	**0.0**	70
		1.0	60
	6	**0.0**	0
		1.0	6
	7	**0.0**	2
		1.0	26
1	**3**	**0.0**	11
		1.0	27
	6	**0.0**	1
		1.0	11
	7	**0.0**	13
		1.0	76

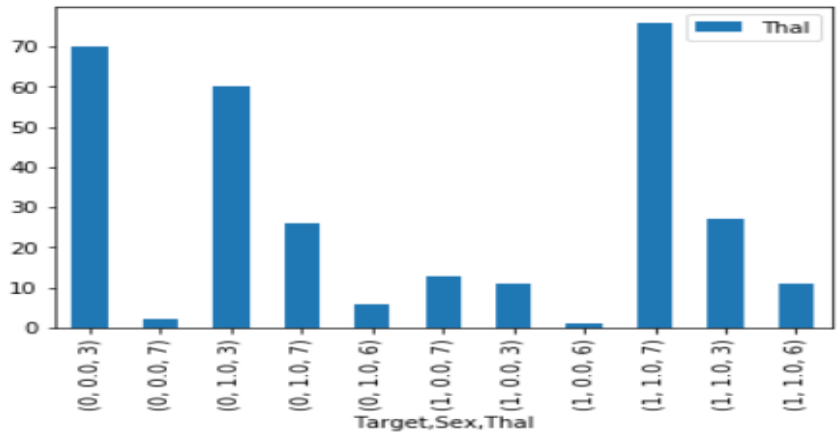
```
targets['Thal'].value_counts().plot(kind = 'bar' ,legend ='True')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x2493f15c438>
```



```
targets1['Thal'].value_counts().plot(kind = 'bar',legend=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x2493f1b6390>
```



6.7. Attribute analysis: CA

Ca (C-arm fluoroscopy results) C-arm is a medical imaging device. (number of blocked major vessels supplying blood (0-3) colored by fluoroscopy)

- When target 0 and target 1 patient numbers are compared, the number of patients with 1 blocked vessel in target 1 increased more than 2 times, the number with 2 blocked vessels increased more than 4 times, and the number with 3 blocked vessels increased almost 6 times.
- Female patients with three blocked vessels are all have heart disease.
- Having two-blocked vessels has the greatest effect on both males and females in developing hd. But the percentage of having 2-blocked vessels in men with illness is 9.28 times higher than men in target 0, while this value is 4.6 in women.

```
my_data.pivot_table(values ='New_ID' ,index = ['Target','Ca'],aggfunc = 'count')
```

		New_ID
Target	**Ca**	
0	**0**	128
	1	20
	2	7
	3	3
1	**0**	42
	1	43
	2	28
	3	13

```
In [96]: #Determining the change in the incidence rate of ca 2 among the target groups
ca_female_target0_percentage = ca_targets_genders(0,0,2)/number_female_targetzero*100
ca_female_target0_percentage
ca_female_target1_percentage = ca_targets_genders(1,0,2)/number_female_targetone*100
ca_female_target1_percentage
ca_male_target0_percentage = ca_targets_genders(0,1,2)/number_male_targetzero*100
ca_male_target0_percentage
ca_male_target1_percentage = ca_targets_genders(1,1,2)/number_male_targetone*100
ca_male_target1_percentage
#percentage ratio of two target groups with same gender
x = ca_female_target1_percentage/ca_female_target0_percentage
y = ca_male_target1_percentage/ ca_male_target0_percentage
result = [x,y]
result
```

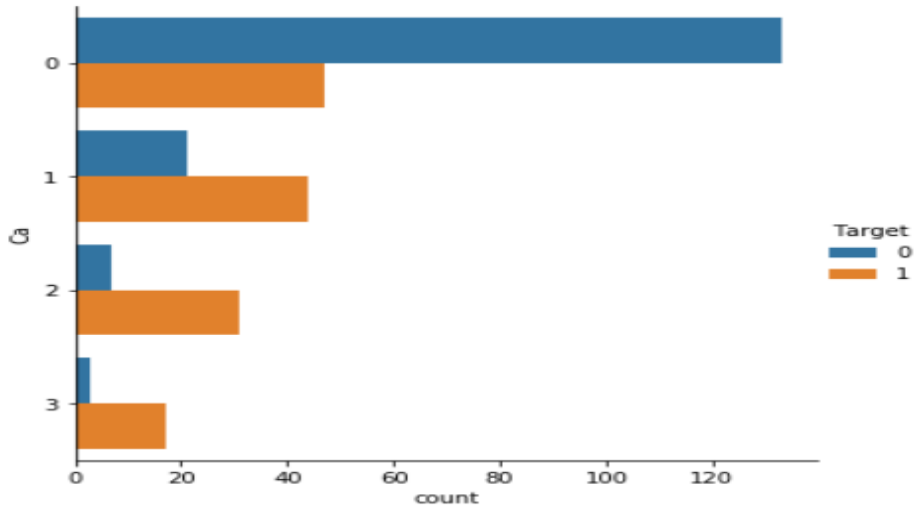
```
Out[96]: [4.608, 9.280701754385964]
```

```
my_data.pivot_table(values ='New_ID' ,index = ['Target','Ca','Sex'],aggfunc = 'count')
```

Target	Ca	Sex	New_ID
0	0	0.0	55
		1.0	78
	1	0.0	12
		1.0	9
	2	0.0	5
		1.0	2
	3	0.0	0
		1.0	3
1	0	0.0	10
		1.0	37
	1	0.0	3
		1.0	41
	2	0.0	8
		1.0	23
	3	0.0	4
		1.0	13

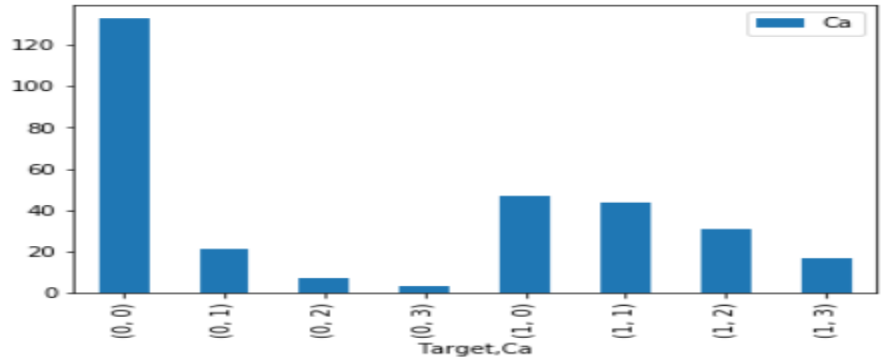
```
#https://seaborn.pydata.org/tutorial/categorical.html
sns.catplot(y="Ca", hue="Target", kind="count",data=my_data)
```

```
<seaborn.axisgrid.FacetGrid at 0x2493c6f8fd0>
```



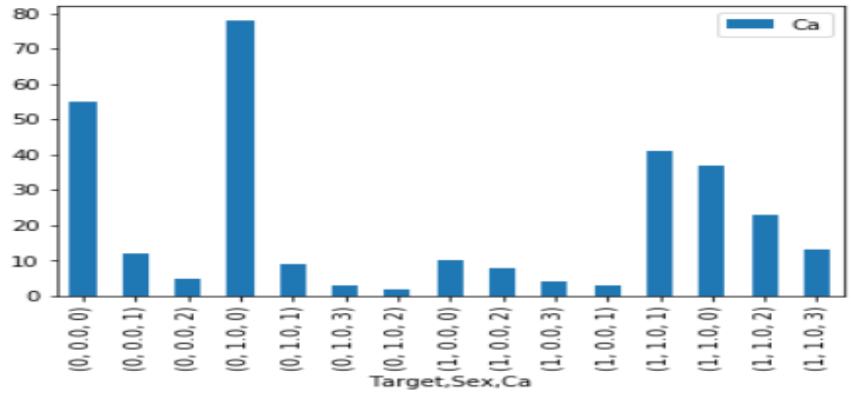
```
targets['Ca'].value_counts().plot(kind = 'bar',legend=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x2493f255358>
```




```
targets1['Ca'].value_counts().plot(kind = 'bar',legend=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x2493f2b5940>
```

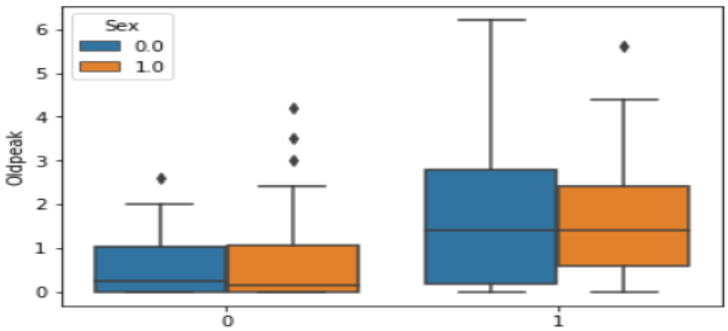


6.8. Attribute analysis: OLDPEAK (ST depression induced by exercise relative to rest)

- For patients in target 1, the avg. old peak value of 1,41 is 2,5 times the avg. old peak value in patients of target 0. (0,56)
- Males do not show disease at higher old peak values in target 0 compared to female patients. In target 1, males have higher mean value of 1.42 than females with a mean value of 1.36.

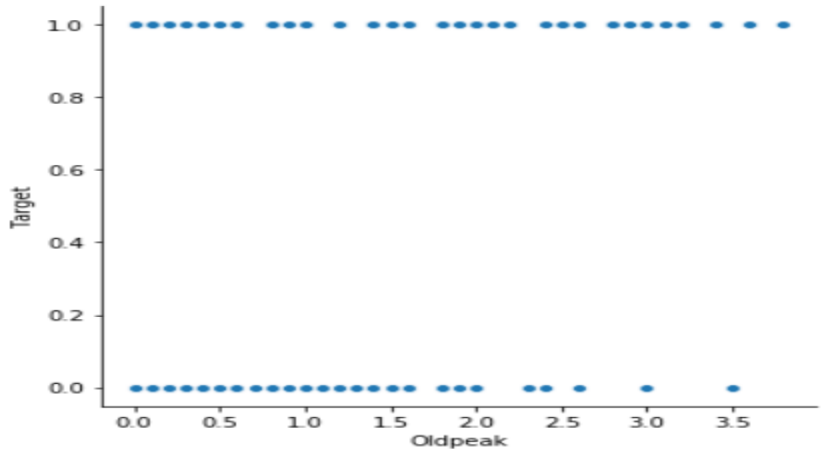
```
#https://seaborn.pydata.org/generated/seaborn.boxplot.html#seaborn.boxplot
sns.boxplot(x='Target',y='Oldpeak',hue ='Sex',data= my_data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x145cffba550>
```



```
sns.relplot(
    data=my_data_oldpeak1,
    x="Oldpeak", y="Target" )
```

```
<seaborn.axisgrid.FacetGrid at 0x14f7480de80>
```



In [105]:
```
mask = my_data_oldpeak1.groupby(['Target','Sex'])
mask.describe().stack()
```

Out[105]:

			Age	RestingBloodPressure	Chol	MaxHeartRate	Oldpeak
Target	**Sex**						
0	**0.0**	**count**	72.000000	72.000000	72.000000	72.000000	72.000000
		mean	54.555556	128.736111	256.750000	154.027778	0.554167
	1.0	**count**	91.000000	91.000000	91.000000	91.000000	91.000000
		mean	50.956044	129.120879	231.175824	161.967033	0.572527
	1.0	**count**	110.000000	110.000000	110.000000	110.000000	110.000000
		mean	56.063636	131.872727	245.736364	138.736364	1.421818
1	**0.0**	**count**	22.000000	22.000000	22.000000	22.000000	22.000000
		mean	58.909091	143.409091	274.772727	143.045455	1.363636

```
mask = my_data_oldpeak1.groupby('Target')
mask.describe().stack()
```

		Age	RestingBloodPressure	Chol	MaxHeartRate	Oldpeak
Target						
0	**count**	163.000000	163.000000	163.000000	163.000000	163.000000
	mean	52.546012	128.950920	242.472393	158.460123	0.564417
1	**count**	132.000000	132.000000	132.000000	132.000000	132.000000
	mean	56.537879	133.795455	250.575758	139.454545	1.412121

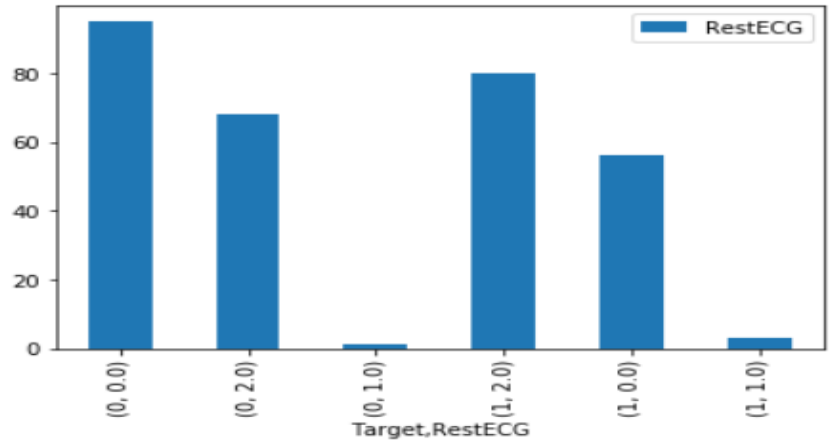
6.9. Attribute analysis: RESTECG (Resting ecg results)

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.

- 80 out of 139 (around 57%) sick patients are showing probable or definite left ventricular hypertrophy by Estes' criteria(value2). In contrast, it is 68 out of 164. (around 41%) in target 0
- The percentage of sick women with value 2 is almost 1.30 times higher than the percentage of non-sick women having this value, in comparison it is nearly 1.44 times higher in sick men than non-sick men Thus, having probable or definite left ventricular hypertrophy has more impact on males than females.

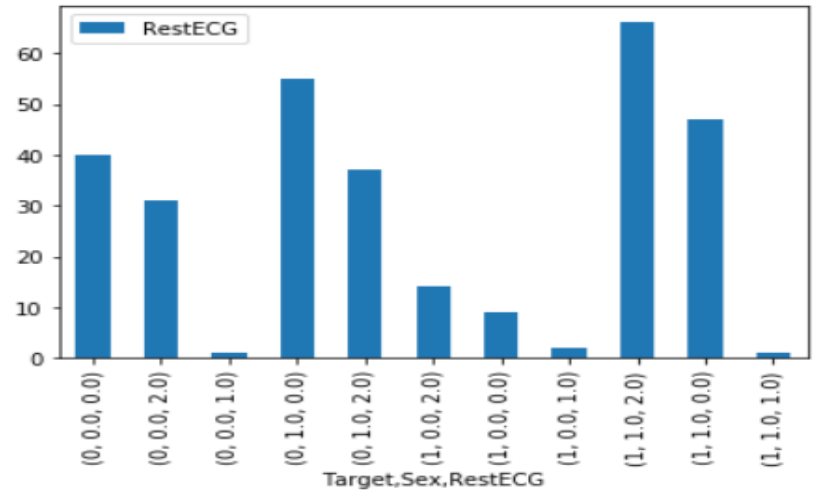
```
targets['RestECG'].value_counts().plot(kind = 'bar',legend=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x145d0050198>
```




```
targets1['RestECG'].value_counts().plot(kind = 'bar',legend=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x145d001d208>
```



```
#to determine the number of patients of different gender in target groups
def targets_genders(a,b) :
    x = my_data['Target']== a
    y = my_data['Sex']== b
    return my_data[x&y].shape[0]
number_female_targetone= targets_genders(1,0)
number_female_targetzero= targets_genders(0,0)
number_male_targetone= targets_genders(1,1)
number_male_targetzero= targets_genders(0,1)
def restecg_targets_genders(a,b,c) :
    x = my_data['Target']== a
    y = my_data['Sex']== b
    z = my_data['RestECG']== c
    return my_data[x&y&z].shape[0]
restecg2_female_target0_percentage = restecg_targets_genders(0,0,2)/number_female_targetzero*100
restecg2_female_target0_percentage
restecg2_female_target1_percentage = restecg_targets_genders(1,0,2)/number_female_targetone*100
restecg2_female_target1_percentage
restecg2_male_target0_percentage = restecg_targets_genders(0,1,2)/number_male_targetzero*100
restecg2_male_target0_percentage
restecg2_male_target1_percentage = restecg_targets_genders(1,1,2)/number_male_targetone*100
restecg2_male_target1_percentage
#percentage difference with same gender between two target groups
x = restecg2_female_target1_percentage/restecg2_female_target0_percentage
y = restecg2_male_target1_percentage/ restecg2_male_target0_percentage
result = [x,y]
result
```

```
[1.3006451612903227, 1.4395448079658606]
```

6.10. Attribute analysis: EXANG (Exercise-induced angina)

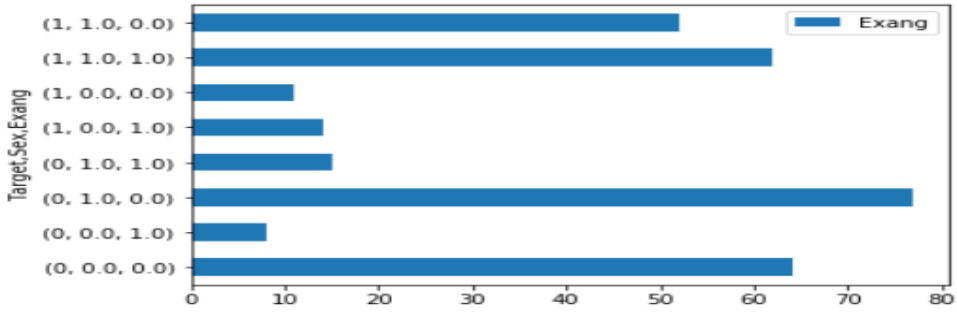
- 76 out of 139 people in target 1 have exercise-induced angina. In comparison, it is 23 out of 161 in target 0.
- The percentage of women with exercise-induced angina at target 1 is 5 times the percentage at target 0 and for men it is 3.3 times the percentage at target 0.
- Having exercise-induced angina has greater impact on females in developing heart disease than men.

```
#to determine the number of patients of different gender in target groups
def exang_targets_genders(a,b,c) :
    x = my_data['Target']== a
    y = my_data['Sex']== b
    z = my_data['Exang']== c
    return my_data[x&y&z].shape[0]
exang_female_target0_percentage = exang_targets_genders(0,0,1)/number_female_targetzero*100
exang_female_target0_percentage
exang_female_target1_percentage = exang_targets_genders(1,0,1)/number_female_targetone*100
exang_female_target1_percentage
exang_male_target0_percentage = exang_targets_genders(0,1,1)/number_male_targetzero*100
exang_male_target0_percentage
exang_male_target1_percentage = exang_targets_genders(1,1,1)/number_male_targetone*100
exang_male_target1_percentage
#percentage difference with same gender between two target groups
x = exang_female_target1_percentage/exang_female_target0_percentage
y = exang_male_target1_percentage/ exang_male_target0_percentage
result = [x,y]
result
```

```
[5.040000000000001, 3.3356725146198833]
```

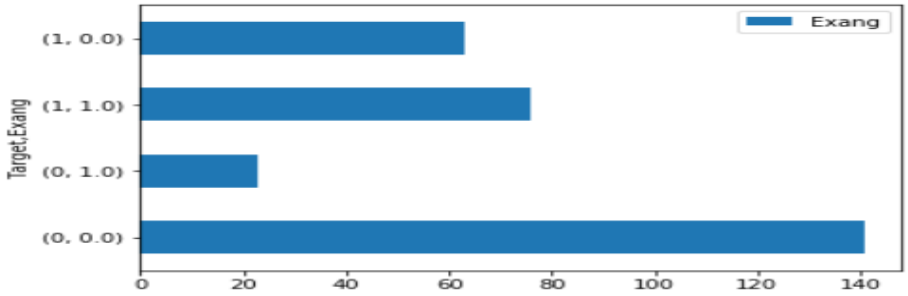
```
targets1['Exang'].value_counts().plot(kind = 'barh',legend=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x145d0128b70>
```




```
targets['Exang'].value_counts().plot(kind = 'barh',legend=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x145d01b20b8>
```

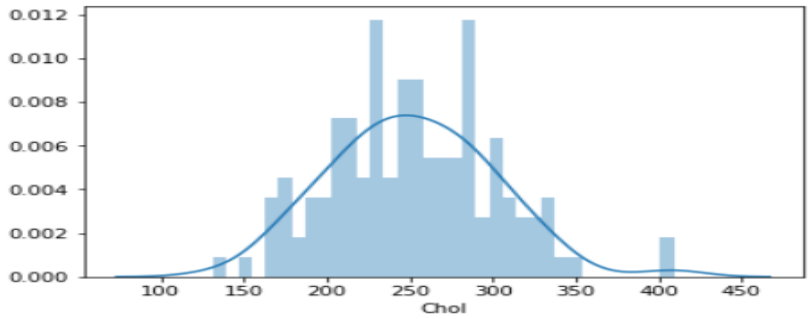


6.11. Attribute analysis: Cholesterol (Serum cholesterol in mg/dl)

- The difference in cholesterol levels between in target 0 and 1 is about 10 mmHg, and this value is about 17 mmHg in women and 15 mmHg in men.
- Male patients develop a disease with lower cholesterol levels than females..

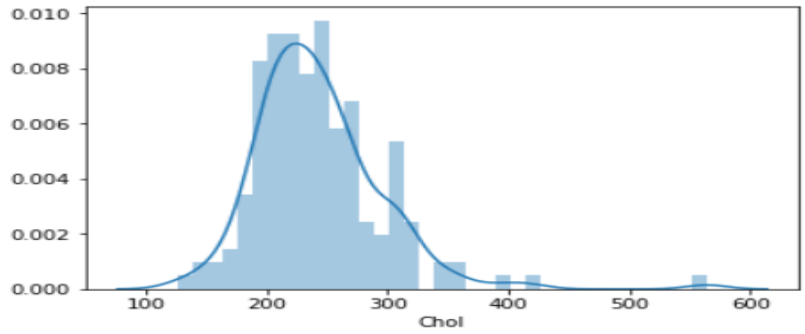
```
sns.distplot(my_data[my_data['Target']==1]['Chol'],kde=True,bins=35)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x145d020e080>
```



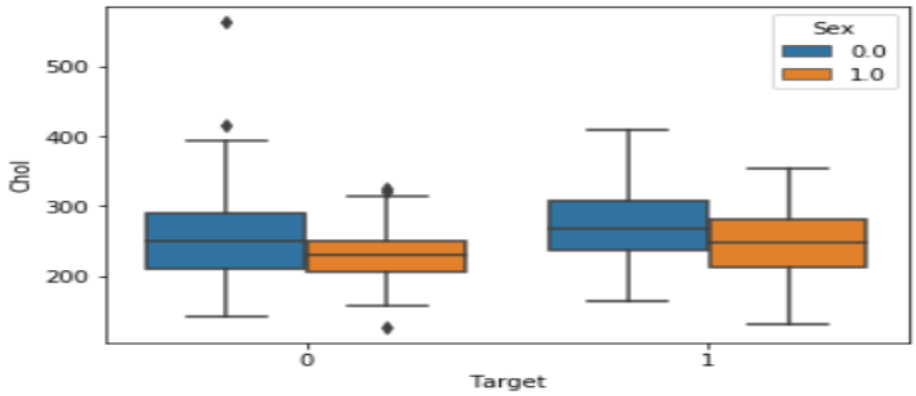
```
sns.distplot(my_data[my_data['Target']==0]['Chol'],kde=True,bins=35)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x145d02db6a0>
```




```
sns.boxplot(x='Target',y='Chol',hue ='Sex',data= my_data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x145d03843c8>
```




```
#https://medium.com/analytics-vidhya/outlier-treatment-9bbe87384d02
outliers('Chol')
```

```
211.0
275.0
64.0
115.0 371.0
```

In [120]:
```
mask = my_data['Chol'] < 371
mask1 = my_data['Chol'] > 115
my_data_chol= my_data[mask&mask1]
my_data_chol.describe()
```

Out[120]:

	Age	RestingBloodPressure	Chol	MaxHeartRate	Oldpeak
count	298.000000	298.000000	298.000000	298.000000	298.000000
mean	54.302013	131.620805	243.479866	149.506711	1.025168
std	9.038790	17.675160	45.063008	23.049244	1.156392
min	29.000000	94.000000	126.000000	71.000000	0.000000
25%	47.250000	120.000000	211.000000	133.000000	0.000000
50%	55.000000	130.000000	240.000000	152.000000	0.650000
75%	60.750000	140.000000	273.750000	166.000000	1.600000
max	77.000000	200.000000	360.000000	202.000000	6.200000

```
mask = my_data_chol.groupby(['Target'])
mask.describe().stack()
```

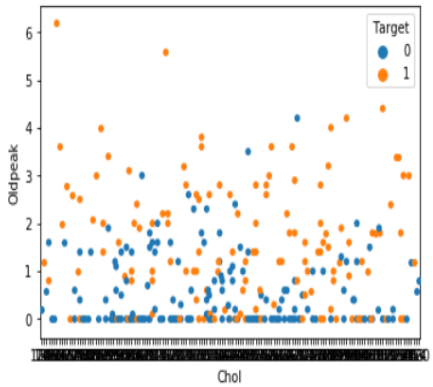
Target		Age	RestingBloodPressure	Chol	MaxHeartRate	Oldpeak
0	count	161.000000	161.000000	161.000000	161.000000	161.000000
	mean	52.360248	129.204969	238.621118	158.385093	0.575155
	std	9.450233	16.272722	43.775413	19.377197	0.783185
	min	29.000000	94.000000	126.000000	96.000000	0.000000
	25%	44.000000	120.000000	208.000000	148.000000	0.000000
	50%	52.000000	130.000000	234.000000	162.000000	0.200000
	75%	58.000000	140.000000	265.000000	172.000000	1.000000
	max	76.000000	180.000000	360.000000	202.000000	4.200000
1	count	137.000000	137.000000	137.000000	137.000000	137.000000
	mean	56.583942	134.459854	249.189781	139.072993	1.554015
	std	7.977581	18.859782	46.036835	22.704183	1.295071
	min	35.000000	100.000000	131.000000	71.000000	0.000000
	25%	52.000000	120.000000	217.000000	125.000000	0.500000
	50%	58.000000	130.000000	249.000000	142.000000	1.400000
	75%	62.000000	145.000000	283.000000	157.000000	2.500000
	max	77.000000	200.000000	353.000000	195.000000	6.200000

```
mask = my_data_chol.groupby(['Target','Sex'])
mask.describe().stack()
```

Target	Sex		Age	RestingBloodPressure	Chol	MaxHeartRate	Oldpeak
0	0.0	count	69.000000	69.000000	69.000000	69.000000	69.000000
		mean	54.115942	128.608696	247.985507	153.855072	0.526087
	1.0	count	92.000000	92.000000	92.000000	92.000000	92.000000
		mean	51.043478	129.652174	231.597826	161.782609	0.611957
1	0.0	count	23.000000	23.000000	23.000000	23.000000	23.000000
		mean	59.043478	147.000000	264.695652	142.391304	1.665217
	1.0	count	114.000000	114.000000	114.000000	114.000000	114.000000
		mean	56.087719	131.929825	246.061404	138.403509	1.531579

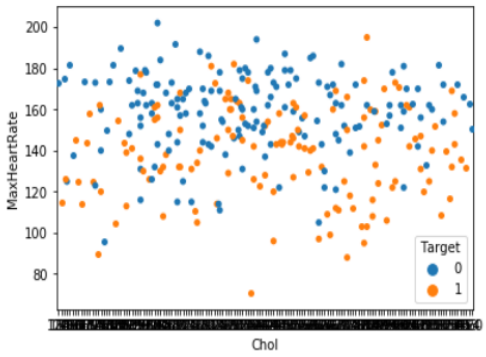
```
sns.swarmplot(x='Chol', y='Oldpeak', data=my_data_chol, hue='Target')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x14f792eea20>
```




```
sns.swarmplot(x='Chol', y='MaxHeartRate',
              data=my_data_chol, hue='Target')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x14f79787e80>
```



6.12. Attribute analysis: Maximum Heart Rate

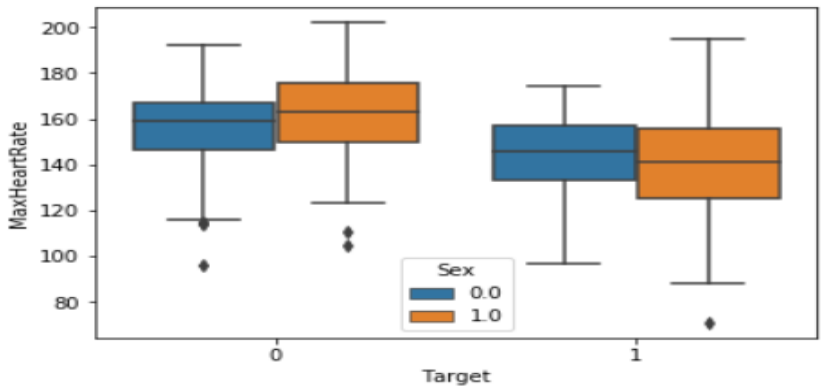
By convention, the maximum predicted heart rate is calculated as 220 (210 for women) minus the patient's age.

- Sick patients achieved less heart rate with an avg. of 139 than non-sick patients with average value of 158.
- In target 0, the avg. heart rate of men is almost 162 and it is 154 for women.
- In target 1, the avg. heart rate of men is nearly 138 and it is nearly 143 for women.
- Women are more susceptible to heart rate decline in developing hd.

.

```
sns.boxplot(x='Target',y = 'MaxHeartRate',hue='Sex',data = my_data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x267c09e0978>
```



In [131]:
```
outliers('MaxHeartRate')
```

```
133.5
166.0
32.5
84.75 214.75
```

In [132]:
```
my_data_maxheartrate = my_data[my_data['MaxHeartRate'] > Upper_outlier_value]
my_data_maxheartrate.shape
#there is no upper outlier value in maximum heart rate column
```

Out[132]: (303, 15)

In [133]:
```
my_data_maxheartrate1 = my_data[my_data['MaxHeartRate'] < 84.75]
my_data_maxheartrate1
```

Out[133]:

	New_ID	Age	Sex	Chestpain	RestingBloodPressure	Chol	FastingBloodSugar	RestECG	MaxHeartRate	Exang	Oldpeak	Slope	Ca	Thal	Target
245	246	67	1.0	4.0	120	237	0.0	0.0	71	0.0	1.0	2.0	0	3	1

In [134]:
```
#There is only one lower outlier value so it was ignored as it had no meaningful effect on the mean value.
```

```
targets = my_data.groupby(['Target'])
targets.mean()
```

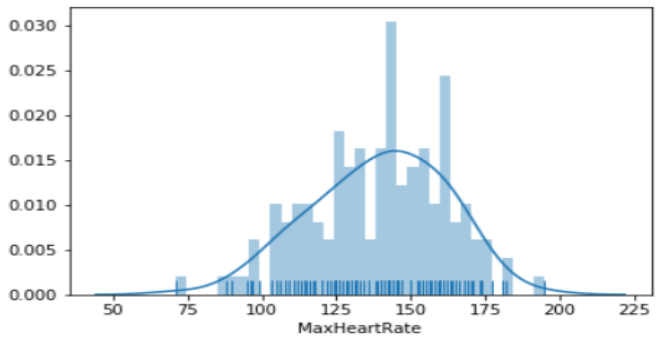
Target	Age	RestingBloodPressure	Chol	MaxHeartRate	Oldpeak
0	52.585366	129.250000	242.640244	158.378049	0.586585
1	56.625899	134.568345	251.474820	139.258993	1.574101

```
mask = my_data.groupby(['Target','Sex'])
mask['MaxHeartRate'].mean()
```

```
Target  Sex
0       0.0    154.027778
        1.0    161.782609
1       0.0    143.160000
        1.0    138.403509
Name: MaxHeartRate, dtype: float64
```

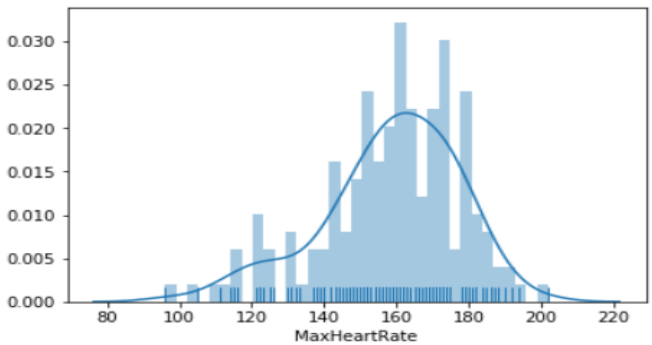
```
sns.distplot(my_data[my_data['Target']==1]['MaxHeartRate'],kde=True,bins=35,rug=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x267bea53898>
```




```
sns.distplot(my_data[my_data['Target']==0]['MaxHeartRate'],kde=True,bins=35,rug=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x267c08b4cc0>
```

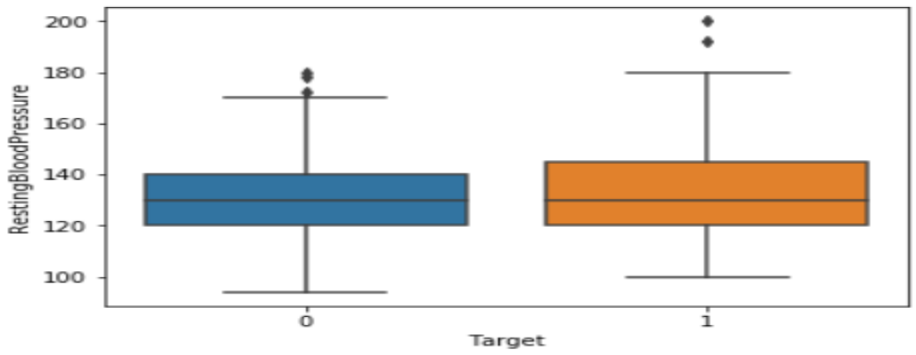


6.13. Attribute analysis: **Resting Blood Pressure (**in mmHg on admission to hospital)

- The average blood pressure is 128 mmHg in non-sick patients, and it is 131 mmHg in sick patients.
- In target 1, the avg. rbp level for women is 138.15 mmHg, it is lower in men with a value of 130.2 mmHg.
- In target 1, male patients' third quartile rbp value is 140 mmHg, whereas it is 150 mmHg in females.
- Patients having resting blood pressure above 180 mmHg are all have heart disease.
- It is observed that men develop disease in lower blood pressure than women.

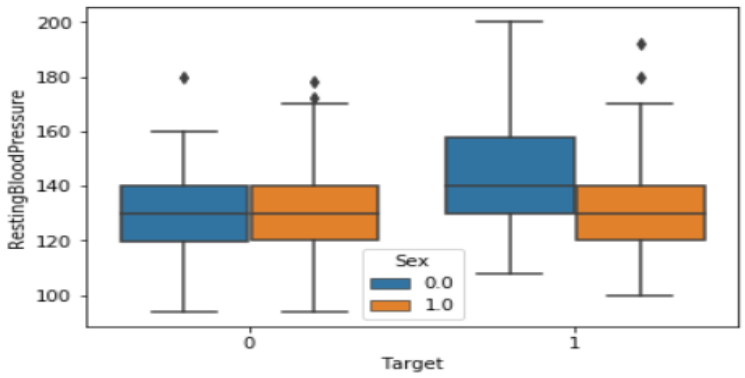
```
sns.boxplot(data=my_data,x='Target',y='RestingBloodPressure')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x267c0a96630>
```




```
sns.boxplot(x='Target',y = 'RestingBloodPressure',hue='Sex',data = my_data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x267c04697b8>
```




```
In [137]: outliers('RestingBloodPressure')
```

```
120.0
140.0
20.0
90.0 170.0
```

```
In [142]: my_data_restingbloodpressure = my_data[my_data['RestingBloodPressure'] > 170]
          my_data_restingbloodpressure.shape
```

```
Out[142]: (9, 15)
```

```
In [143]: my_data_restingbloodpressure1 = my_data[my_data['RestingBloodPressure'] < 90]
          my_data_restingbloodpressure1.shape
```

```
Out[143]: (0, 15)
```

```
mask = my_data_rbp.groupby(['Target','Sex'])
mask.describe().stack()
```

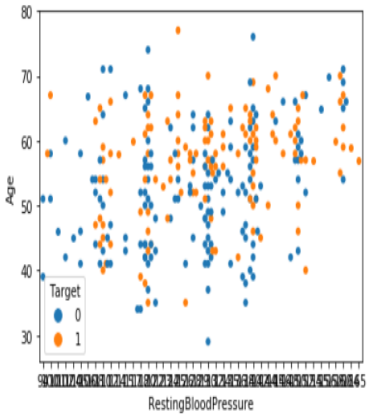
Target	Sex		Age	RestingBloodPressure	Chol	MaxHeartRate	Oldpeak
0	0.0	count	71.000000	71.000000	71.000000	71.000000	71.000000
		mean	54.422535	128.014085	255.788732	154.028169	0.561972
	1.0	count	89.000000	89.000000	89.000000	89.000000	89.000000
		mean	50.797753	128.179775	231.584270	162.044944	0.573034
1	0.0	count	20.000000	20.000000	20.000000	20.000000	20.000000
		mean	59.150000	138.150000	279.350000	143.750000	1.650000
		75%	62.000000	150.000000	310.000000	157.750000	2.150000
	1.0	count	110.000000	110.000000	110.000000	110.000000	110.000000
		mean	55.945455	130.254545	244.363636	137.581818	1.540000
		75%	61.000000	140.000000	279.750000	156.000000	2.400000

```
mask = my_data_rbp.groupby('Target')
mask.describe().stack()
```

Target		Age	RestingBloodPressure	Chol	MaxHeartRate	Oldpeak
0	count	160.000000	160.000000	160.000000	160.000000	160.000000
	mean	52.406250	128.106250	242.325000	158.487500	0.568125
1	count	130.000000	130.000000	130.000000	130.000000	130.000000
	mean	56.438462	131.469231	249.746154	138.530769	1.556923

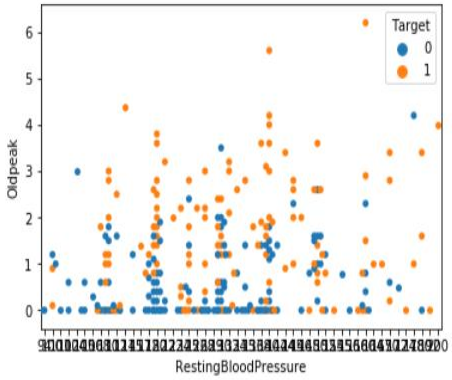
```
sns.swarmplot(x='RestingBloodPressure', y='Age', data=my_data_rbp, hue='Target')
```

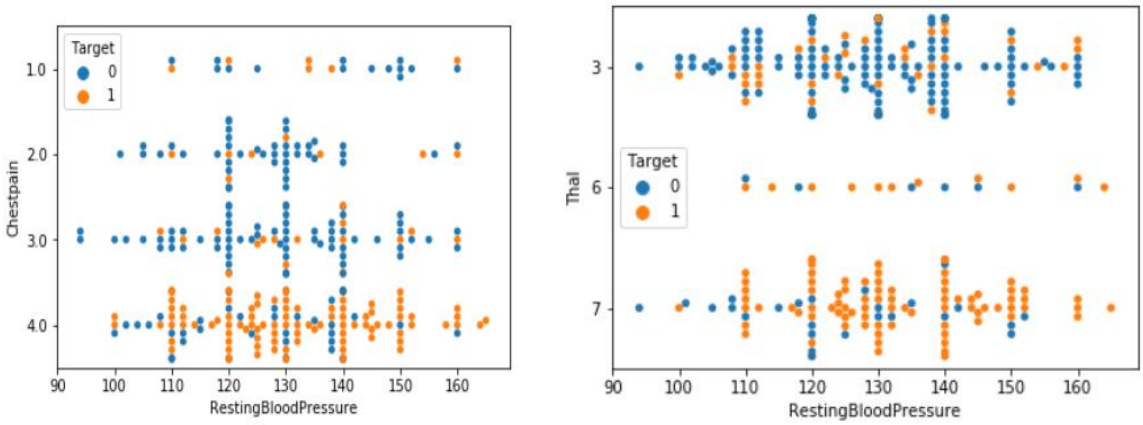
```
<matplotlib.axes._subplots.AxesSubplot at 0x14f7c21f3c8>
```




```
sns.swarmplot(x='RestingBloodPressure', y='Oldpeak',
              data=my_data_chol, hue='Target')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x14f7b685048>
```





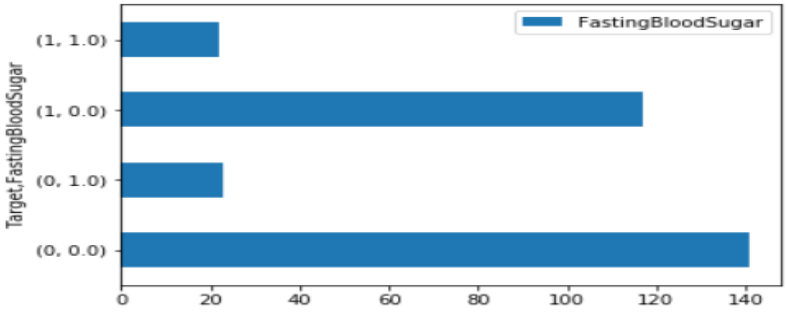
6.14. Attribute analysis: Fasting Blood Sugar (Fbs)

Fbs>120 mg/dl (1 =true; 0 = false)

- In target 1, 15.8% of patients have high fbs levels and in target 0, 14.0% of patients are having high fbs level.
- The percentage of having high fbs levels in women with illness is 2.88 times higher than women in target 0, while this value is 0.76 times lower in men.
- High fbs level has a greater effect in women than in men.

```
targets['FastingBloodSugar'].value_counts().plot(kind = 'barh',legend=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x267c0ca40f0>
```

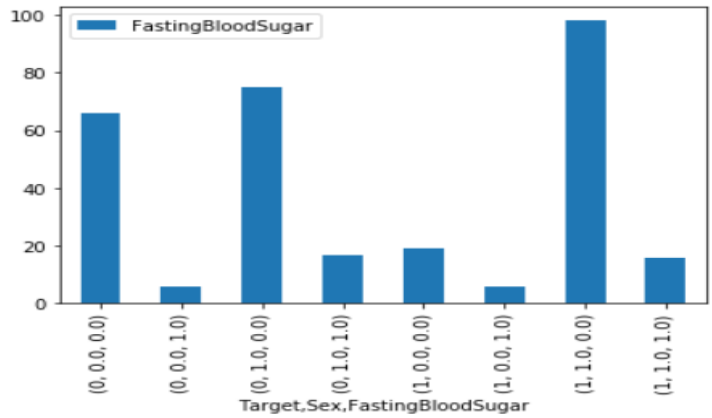



```
targets['FastingBloodSugar'].value_counts()
```

```
Target  FastingBloodSugar
0       0.0                 141
        1.0                  23
1       0.0                 117
        1.0                  22
Name: FastingBloodSugar, dtype: int64
```

```
targets1['FastingBloodSugar'].value_counts().plot(kind = 'bar',legend=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x267c0d67a90>
```




```
targets1['FastingBloodSugar'].value_counts()
```

```
Target  Sex  FastingBloodSugar
0       0.0  0.0                 66
             1.0                  6
        1.0  0.0                 75
             1.0                 17
1       0.0  0.0                 19
             1.0                  6
        1.0  0.0                 98
             1.0                 16
Name: FastingBloodSugar, dtype: int64
```

```
def target_genders(a,b) :
    x = my_data['Target']== a
    y = my_data['Sex']== b
    return my_data[x&y].shape[0]
number_female_targetone= target_genders(1,0)
number_female_targetzero= target_genders(0,0)
number_male_targetone= target_genders(1,1)
number_male_targetzero= target_genders(0,1)
def fbs_targets_genders(d,e,f) :
    k = my_data['Target']== d
    l = my_data['Sex']== e
    m = my_data['FastingBloodSugar']== f
    return my_data[k&l&m].shape[0]
fbs_female_target0_percentage = fbs_targets_genders(0,0,1)/number_female_targetzero*100
fbs_female_target0_percentage
fbs_female_target1_percentage = fbs_targets_genders(1,0,1)/number_female_targetone*100
fbs_female_target1_percentage
fbs_male_target0_percentage = fbs_targets_genders(0,1,1)/number_male_targetzero*100
fbs_male_target0_percentage
fbs_male_target1_percentage = fbs_targets_genders(1,1,1)/number_male_targetone*100
fbs_male_target1_percentage
#percentage difference with same gender between two target groups
x = fbs_female_target1_percentage/fbs_female_target0_percentage
y = fbs_male_target1_percentage/ fbs_male_target0_percentage
result = [x,y]
result
```

```
[2.8800000000000003, 0.759545923632611]
```

7.Apply Model

7.1. Feature Selection

Supervised filter methods of statistical tests will be applied on data for univariate feature selection to perform a better classification predicting model.

7.1.1. Crammer coefficient correlation between categorical attributes:

```
#https://researchpy.readthedocs.io/en/latest/crosstab_documentation.html
import researchpy
croostab = researchpy.crosstab(my_data['Target'],my_data['Sex'],test = 'chi-square')
croostab
```

```
(       Sex
 Sex    0.0  1.0  All
 Target
 0       72   92  164
 1       25  114  139
 All     97  206  303,                Chi-square test  results
 0  Pearson Chi-square ( 1.0) =   23.2181
 1                   p-value =    0.0000
 2               Cramer's phi =    0.2768)
```

```
researchpy.crosstab(my_data['Target'],my_data['RestECG'],test = 'chi-square')
```

```
(        RestECG
 RestECG     0.0 1.0  2.0  All
 Target
 0            95   1   68  164
 1            56   3   80  139
 All         151   4  148  303,                Chi-square test  results
 0  Pearson Chi-square ( 2.0) =   10.0515
 1                   p-value =    0.0066
 2                 Cramer's V =    0.1821)
```

```
researchpy.crosstab(my_data['Target'],my_data['Exang'],test = 'chi-square')
```

```
(        Exang
 Exang     0.0 1.0  All
 Target
 0          141  23  164
 1           63  76  139
 All        204  99  303,                  Chi-square test  results
 0  Pearson Chi-square ( 1.0) =   56.5193
 1                   p-value =    0.0000
 2                 Cramer's phi =    0.4319)
```

```
researchpy.crosstab(my_data['Target'],my_data['Slope'],test = 'chi-square')
```

```
(        Slope
 Slope     1.0  2.0 3.0  All
 Target
 0          106   49   9  164
 1           36   91  12  139
 All        142  140  21  303,                  Chi-square test  results
 0  Pearson Chi-square ( 2.0) =   45.7846
 1                   p-value =    0.0000
 2                 Cramer's V =    0.3887)
```

```
researchpy.crosstab(my_data['Target'],my_data['Thal'],test = 'chi-square')
```

```
(        Thal
 Thal       3    6    7  All
 Target
 0        130    6   28  164
 1         38   12   89  139
 All      168   18  117  303,                  Chi-square test  results
 0  Pearson Chi-square ( 2.0) =   82.6845
 1                   p-value =    0.0000
 2                 Cramer's V =    0.5224)
```

```
researchpy.crosstab(my_data['Target'],my_data['FastingBloodSugar'],test = 'chi-square')
```

```
(                  FastingBloodSugar
 FastingBloodSugar                0.0 1.0  All
 Target
 0                                141  23  164
 1                                117  22  139
 All                              258  45  303,
                  Chi-square test  results
 0  Pearson Chi-square ( 1.0) =    0.1934
 1                  p-value =    0.6601
 2                Cramer's phi =    0.0253)
```

```
researchpy.crosstab(my_data['Ca'], my_data['Target'], test= "chi-square")
```

```
(       Target
 Target      0    1  All
 Ca
 0         133   47  180
 1          21   44   65
 2           7   31   38
 3           3   17   20
 All       164  139  303,                  Chi-square test  results
 0  Pearson Chi-square ( 3.0) =   72.6169
 1                  p-value =    0.0000
 2                Cramer's V =    0.4896)
```

```
researchpy.crosstab(my_data['Chestpain'], my_data['Target'], test= "chi-square")
```

```
(          Target
 Target         0    1  All
 Chestpain
 1.0           16    7   23
 2.0           41    9   50
 3.0           68   18   86
 4.0           39  105  144
 All          164  139  303,                  Chi-square test  results
 0  Pearson Chi-square ( 3.0) =   81.8158
 1                  p-value =    0.0000
 2                Cramer's V =    0.5196)
```

According to Cramer's V Correlation:

- Having thalassemia disorder and angina have strong correlation with developing heart disease.(>0.5)
- C-arm fluoroscopy results, exercise induced angina and slope results have moderate association with developing heart disease. (0.3 to 0.5)
- Resting ecg results has low association (0.1 to 0.3) and fasting blood sugar level has if any association with target 1.

7.1.2 Determining the association between target attribute and numerical attributes with Logistic Regression:

```
#https://towardsdatascience.com/interpreting-coefficients-in-linear-and-logistic-regression-6ddf1295f6f1
import numpy as np
from sklearn. linear_model import LogisticRegression
X = my_data[['Age','RestingBloodPressure','Chol','MaxHeartRate','Oldpeak']]
y = my_data['Target']
logreg = LogisticRegression(solver = 'liblinear')
logreg.fit(X,y)
log_odds = logreg.coef_[0]

pd.DataFrame(log_odds,X.columns,columns = ['coef'])\
.sort_values(by='coef',ascending=False)
```

	coef
Oldpeak	0.700383
RestingBloodPressure	0.014709
Age	0.007958
Chol	0.003543
MaxHeartRate	-0.030684

Maximum heart rate and Old peak attributes have the strongest association and cholesterol levels,age and resting blood pressure levels have weaker association with developing heart disease. [12]

7.2. Model Evaluation:

In this data set, labeled attribute is binary and there are both categorical and numerical attributes. Thus, Logistic Regression, k_NN, Decision Tree, Naive Bayes and Random Forest classification models will be evaluated by k-fold cross validation. In k-cross validation, k refers to the number of groups that given data is to be split into and in this analysis, k value is fixed to 10 because it was found to have low bias. As an advantage, cross validation can be used both for small and large datasets. Also, it gives as a chance to use all our data unlike train and test splits.

Evaluation will be made without removing outlier values and weakest attributes and after with removing them.[13]

```
#MODEL SELECTION
from sklearn import model_selection
from sklearn. linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
import warnings
warnings.filterwarnings("ignore", category = DeprecationWarning)
```

```
data = my_data
values = data.values
# Preparing Data For Training
Y = values[:,14]
#target column is labeled.
X = values[:,0:14]
Y= Y.astype('int32')
```

7.2.1. Cross validation without removing outlier values and weak associated features.

```
#Cross validation (Model evaluation)
outcome = []
model_names = []
models = [('LogReg', LogisticRegression(solver = 'liblinear')),
          ('GaussianNB', GaussianNB()),
          ('RandomForest',RandomForestClassifier(n_estimators=100)),
          ('DecTree', DecisionTreeClassifier()),
          ('KNN', KNeighborsClassifier())]
```

```
#Cross validation (Model evaluation)
random_seed =12
for model_name, model in models:
    k_fold_validation = model_selection.KFold(n_splits=10, random_state=random_seed)
    results = model_selection.cross_val_score(model, X, Y, cv=k_fold_validation, scoring='accuracy')
    outcome.append(results)
    model_names.append(model_name)
    output_message = "%s| Mean=%f STD=%f" % (model_name, results.mean(), results.std())
    print(output_message)
```

```
LogReg| Mean=0.841398 STD=0.082652
GaussianNB| Mean=0.844839 STD=0.066568
RandomForest| Mean=0.824946 STD=0.074489
DecTree| Mean=0.715591 STD=0.105296
KNN| Mean=0.601075 STD=0.059963
```

Removing Outliers:

```
my_data.drop(my_data[my_data['Oldpeak'] > 4].index, inplace = True)
my_data.shape
```

```
(298, 15)
```

```
my_data.drop(my_data[my_data['Chol'] > 371].index, inplace = True)
my_data.shape
```

```
(293, 15)
```

```
my_data.drop(my_data[my_data['RestingBloodPressure'] > 170].index, inplace = True)
my_data.shape
```

```
(284, 15)
```

```
my_data.drop(my_data[my_data['MaxHeartRate'] < 84.75].index, inplace = True)
my_data.shape
```

```
(292, 15)
```

7.2.2. Cross validation after removing outlier values:

```
#MODEL COMPARISON AFTER REMOVING OUTLIERS
random_seed =12
for model_name, model in models:
    k_fold_validation = model_selection.KFold(n_splits=10, random_state=random_seed)
    results = model_selection.cross_val_score(model, X, Y, cv=k_fold_validation, scoring='accuracy')
    outcome.append(results)
    model_names.append(model_name)
    output_message = "%s| Mean=%f STD=%f" % (model_name, results.mean(), results.std())
    print(output_message)
```

```
LogReg| Mean=0.841398 STD=0.082652
GaussianNB| Mean=0.844839 STD=0.066568
RandomForest| Mean=0.824946 STD=0.085923
DecTree| Mean=0.725161 STD=0.115348
KNN| Mean=0.601075 STD=0.059963
```

Result; After removing outliers; only the performance of Decision Tree increased

7.2.3. Cross validation after removing outlier values and negligible features:

```
my_datas = my_data.drop(columns = ['Chol','FastingBloodSugar','Age','RestingBloodPressure'])
```

```
my_datas.shape
```

```
(284, 11)
```

```
data = my_datas
values = data.values

Y = values[:,10]
X = values[:,0:10]
Y= Y.astype('int32')
```

```
outcome = []
model_names = []
models = [('LogReg', LogisticRegression(solver = 'liblinear')),
          ('GaussianNB', GaussianNB()),
          ('RandomForest',RandomForestClassifier(n_estimators=100)),
          ('DecTree', DecisionTreeClassifier()),
          ('KNN', KNeighborsClassifier())]
```

```
random_seed =12
for model_name, model in models:
    k_fold_validation = model_selection.KFold(n_splits=10, random_state=random_seed)
    results = model_selection.cross_val_score(model, X, Y, cv=k_fold_validation, scoring='accuracy')
    outcome.append(results)
    model_names.append(model_name)
    output_message = "%s| Mean=%f STD=%f" % (model_name, results.mean(), results.std())
    print(output_message)
```

```
LogReg| Mean=0.848153 STD=0.067532
GaussianNB| Mean=0.844581 STD=0.078370
RandomForest| Mean=0.834236 STD=0.055583
DecTree| Mean=0.729557 STD=0.139891
KNN| Mean=0.616010 STD=0.058478
```

- Removing weak associated columns increased the performance of all models except Naïve Bayes.
- Classification models ;Logistic Regression, Naive Bayes and Random forest have better accuracies.[14],[15]

7.3. Applying Logistic Regression to determine the relationships between dependent feature and independent features:

Odds ratio is the ratio of probability of occurrence of something to probability of not occurrence of it. In logistic regression, all possible outcomes of dependent attribute are converted to log odds. By the help of coefficient size, we can measure the relevance of an independent attribute.

```
#https://towardsdatascience.com/interpreting-coefficients-in-linear-and-logistic-regression-6ddf1295f6f1
from sklearn.metrics import classification_report,confusion_matrix
import numpy as np
X = my_data[['Age','Chestpain','Sex','RestingBloodPressure','RestECG','MaxHeartRate','Exang','Oldpeak',
         'Slope','Ca','Thal','Chol','FastingBloodSugar']]
y = my_data['Target']
logreg = LogisticRegression(solver = 'liblinear')
logreg.fit(X,y)
log_odds = logreg.coef_[0]

pd.DataFrame(log_odds,X.columns,columns = ['coef'])\
.sort_values(by='coef',ascending=False)
odds = np.exp(logreg.coef_[0])
pd.DataFrame(odds,
             X.columns,
             columns=['coef'])\
            .sort_values(by='coef', ascending=False)
```

	coef
Ca	2.868377
Sex	2.308757
Exang	1.920647
Oldpeak	1.528904
Chestpain	1.471692
Thal	1.450027
RestECG	1.340953
Slope	1.241654
RestingBloodPressure	1.007509
Chol	1.005509
MaxHeartRate	0.967325
Age	0.963325
FastingBloodSugar	0.702429

Feature coefficient sizes in females:

```
#FEATURE IMPORTANCE IN FEMALES
my_datafem = my_data[my_data['Sex']== 0]
#my_datafem.head()
```

```
X = my_datafem[['Age','Chestpain','RestingBloodPressure','RestECG','MaxHeartRate','Exang','Oldpeak',
         'Slope','Ca','Thal','Chol','FastingBloodSugar']]
y = my_datafem['Target']
logreg = LogisticRegression(solver = 'liblinear')
logreg.fit(X,y)
log_odds = logreg.coef_[0]

pd.DataFrame(log_odds,X.columns,columns = ['coef'])\
.sort_values(by='coef',ascending=False)
odds = np.exp(logreg.coef_[0])
pd.DataFrame(odds,
             X.columns,
             columns=['coef'])\
            .sort_values(by='coef', ascending=False)
```

	coef
Exang	2.779505
Ca	2.690426
Thal	1.817234
Chestpain	1.544509
FastingBloodSugar	1.295185
RestECG	1.292776
Slope	1.269149
Oldpeak	1.214105
Chol	1.000372
RestingBloodPressure	0.996443
MaxHeartRate	0.974545
Age	0.968697

Feature coefficient sizes in males:

```
#FEATURE IMPORTANCE IN MALES
my_data_male = my_data[my_data['Sex']== 1]
#my_data_male.head()
```

```
X = my_data_male[['Age','Chestpain','RestingBloodPressure','RestECG','MaxHeartRate','Exang','Oldpeak',
         'Slope','Ca','Thal','Chol','FastingBloodSugar']]
y = my_data_male['Target']
logreg = LogisticRegression(solver = 'liblinear')
logreg.fit(X,y)
log_odds = logreg.coef_[0]

pd.DataFrame(log_odds,X.columns,columns = ['coef'])\
.sort_values(by='coef',ascending=False)
odds = np.exp(logreg.coef_[0])
pd.DataFrame(odds,
             X.columns,
             columns=['coef'])\
            .sort_values(by='coef', ascending=False)
```

	coef
Ca	2.824941
Oldpeak	1.670055
Chestpain	1.520203
Thal	1.379789
RestECG	1.326395
Exang	1.300336
Slope	1.198164
RestingBloodPressure	1.008970
Chol	1.008063
Age	0.970319
MaxHeartRate	0.963721
FastingBloodSugar	0.603104

8.Impediments and Challenges:

8.1. Missing data

The percentage of missing data was around 2% so it could be deleted. But in the dataset, there was only 303 examples so before deletion, it was checked whether the missing values can be imputed. The type of missing data was determined by correlation measurements. As a result, because the data was not missing completely at random, missing values were imputed with mode values.

8.2. Cholesterol and Resting Blood Pressure and Age Attributes:

In healthy adults, cholesterol levels should be under 200 mg/dL but in examples both groups' mean value of cholesterol levels are higher than 237 mg/dL and although outlier values were removed, the difference is only about 10 mmHg between sick and non-sick patients. Besides, patients' mean, and median values of resting blood pressure levels and ages are also nearly similar in both groups. Therefore, by using seaborn library, the relationships of these attributes between other attributes were visualized. It is observed that these 3 attributes have strong relationship with high correlated attributes of dataset such as asymptomatic chest pain, old peak levels, number of blocked vessels.

8.3. Data content:

According to WHO, the most important behavioral risk factors of heart disease are unhealthy diet, physical inactivity, tobacco use, and harmful alcohol use. However, in our dataset, there is no attribute related to these risk factors. So, in this analysis, I cannot get an insight into behavioral risk factors. [16]

9. Conclusion

Patients having high risk to develop heart disease are mostly.

- Male over 57 years old
- Having asymptomatic chest pain, horizontal ST segment depression, reversible type of thalassemia and two blocked vessels which are supplying blood to heart.
- With probable or definite ventricular hypertrophy by Estes' criteria
- Having angina after making exercise
- Achieving heart rate under 140 and old peak over 1.5
- With fasting blood sugar level over 120 mg/dl

Answers to the questions:

1- Are there outlier values in this dataset?

The outlier value is the value that is over 1.5 times the interquartile, which is the difference between the third quartile and first quartile. There were outliers in cholesterol levels, blood pressure levels, ST depression induced by exercise (relative to rest) values and maximum achieved heart rates. The outliers were visualized by boxplot diagrams and a function was created in python using IQR method (Interquartile range method). To have accurate descriptive statistics, outliers were removed by drop () method.

2- Is there any feature to be ignored because of weak association with labeled attribute?

Feature selection is essential to improve the performance of machine learning. Thus, to determine the importance of features, Cramer's V measurement was used for categorical values, and Logistic regression is used for numerical features. Because labelled attribute is binary and independent attributes are categorical and numerical. It is seen that removing weakly associated features increased the performance of models by only around 1%, and weak features like fasting blood sugar levels have a big correlation difference between gender groups. Thus, to have more insights in gender groups, I did not ignore the weak features.

3-Which classification models can be used for this data for predictive analysis as a future work?

Logistic Regression, k-NN, Decision Tree, Naive Bayes and Random Forest classification models were compared by using cross validation to choose the models with higher accuracies. Both Logistic Regression, Naïve Bayes and Random Forest have accuracy around 83 % which is a good score for modeling. As a future work, these three models could be processed to explore if one model works better than others for predictive analysis of the data.

4-Which features have more significant impacts on causing heart disease?

Blocked blood vessels that carry blood to the heart, feeling chest pain in both the presence or absence of exercise, being a man, presence of thalassemia disease and ST depression levels induced by exercise are the most important features in developing heart disease.

5-Is there any difference in the importance of attributes in gender groups?

For determination of the differences between genders, exploratory analysis results and logistic regression results were compared, and it was observed that the results were similar with each other. As a result:

Exercised induced angina, asymptomatic chest pain, thalassemia disorder, maximum heart rate achieved, and fasting blood sugar levels have greater impact on females than males in developing heart disease.

Blocked vessels depression values, resting electrocardiography results and resting blood pressure levels have greater effect on males than females.

10.References

[1] [https://www.who.int/news-room/fact-sheets/detail/the-top-10-causes-of-death]

[2] https://en.wikipedia.org/wiki/Cross-industry_standard_process_for_data_mining

[3] https://archive.ics.uci.edu/ml/datasets/heart+disease

[4] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1123032/

.[5] https://towardsdatascience.com/statistical-test-for-mcar-in-python-9fb617a76eac

[6] https://towardsdatascience.com/the-search-for-categorical-correlation-a1cf7f1888c9

[7] //machinelearningmastery.com/feature-selection-with-real-and-categorical-data/

[8] https://www.statstutor.ac.uk/resources/uploaded/tutorsquickguidetostatistics.pdf

[9] https://researchpy.readthedocs.io/en/latest/crosstab_documentation.html

[10] https://www.udemy.com/course/data-analysis-with-pandas/

[11] https://medium.com/analytics-vidhya/outlier-treatment-9bbe87384d02

[12] https://towardsdatascience.com/interpreting-coefficients-in-linear-and-logistic-regression-6ddf1295f6f1

[13] https://pythondata.com/comparing-machine-learning-methods/

[14] https://towardsdatascience.com/mistakes-in-applying-univariate-feature-selection-methods-34c43ce8b93d

[15] https://machinelearningmastery.com/calculate-feature-importance-with-python/

[16] https://www.who.int/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds)