creasing-the-rate-of-heart-attacks

March 28, 2023

0.1 Import the Packages

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

1 Importing and Inspecting Data :

1.0.1 Importing Data

[2]:
```
data= pd.read_excel("data.xlsx")
data1= pd.read_excel("variable description.xlsx")
```

[3]:
```
data.head()
```

[3]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

[4]:
```
data.columns
```

[4]:
```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')
```

[5]:
```
data.rename(columns={'cp':'chest_pain_type','trestbps':
 ↪'resting_blood_pressure','chol':'cholestoral',
                     'fbs':'fasting_blood_sugar','restecg':
 ↪'resting_electrocardiographic_results',
                     'thalach':'maximum_heart_rate_achieved','exang':
 ↪'exercise_induced_angina',
                     'oldpeak':'ST.depression(exercise/rest)','ca':
 ↪'no_of_major_vessels',
                     'thal':'thalassemia' },inplace=True)
```

[6]:
```
data.head()
```

[6]:

	age	sex	chest_pain_type	resting_blood_pressure	cholestoral	\
0	63	1	3	145	233	
1	37	1	2	130	250	
2	41	0	1	130	204	
3	56	1	1	120	236	
4	57	0	0	120	354	

	fasting_blood_sugar	resting_electrocardiographic_results	\
0	1	0	
1	0	1	
2	0	0	
3	0	1	
4	0	1	

	maximum_heart_rate_achieved	exercise_induced_angina	\
0	150	0	
1	187	0	
2	172	0	
3	178	0	
4	163	1	

	ST.depression(exercise/rest)	slope	no_of_major_vessels	thalassemia	\
0	2.3	0	0	1	
1	3.5	0	0	2	
2	1.4	2	0	2	
3	0.8	2	0	2	
4	0.6	2	0	2	

	target
0	1
1	1
2	1
3	1
4	1

[7]:
```
data.shape
```

[7]:
```
(303, 14)
```

Checking for null values

[8]:
```
data.isna().sum()
```

[8]:
```
age                                   0
sex                                   0
chest_pain_type                       0
resting_blood_pressure                0
cholestoral                           0
fasting_blood_sugar                   0
resting_electrocardiographic_results  0
maximum_heart_rate_achieved           0
exercise_induced_angina               0
ST.depression(exercise/rest)          0
slope                                 0
no_of_major_vessels                   0
thalassemia                           0
target                                0
dtype: int64
```

Checking for duplicate values

[13]:
```
data.duplicated().sum()
```

[13]:
```
0
```

Removing duplicate values

[10]:
```
data= data.drop_duplicates()
```

Treating null values

[11]:
```
from pandas.core.base import value_counts
data.isna().any().value_counts()
```

[11]:
```
False    14
dtype: int64
```

Now ,There is no Missing Value in the data

Statistical summary of the data

[14]:
```
data.describe()
```

[14]:

	age	sex	chest_pain_type	resting_blood_pressure
count	302.00000	302.000000	302.000000	302.000000
mean	54.42053	0.682119	0.963576	131.602649
std	9.04797	0.466426	1.032044	17.563394
min	29.00000	0.000000	0.000000	94.000000
25%	48.00000	0.000000	0.000000	120.000000
50%	55.50000	1.000000	1.000000	130.000000
75%	61.00000	1.000000	2.000000	140.000000
max	77.00000	1.000000	3.000000	200.000000

	cholestoral	fasting_blood_sugar	resting_electrocardiographic_results
count	302.000000	302.000000	302.000000
mean	246.500000	0.149007	0.526490
std	51.753489	0.356686	0.526027
min	126.000000	0.000000	0.000000
25%	211.000000	0.000000	0.000000
50%	240.500000	0.000000	1.000000
75%	274.750000	0.000000	1.000000
max	564.000000	1.000000	2.000000

	maximum_heart_rate_achieved	exercise_induced_angina
count	302.000000	302.000000
mean	149.569536	0.327815
std	22.903527	0.470196
min	71.000000	0.000000
25%	133.250000	0.000000
50%	152.500000	0.000000
75%	166.000000	1.000000
max	202.000000	1.000000

	ST.depression(exercise/rest)	slope	no_of_major_vessels
count	302.000000	302.000000	302.000000
mean	1.043046	1.397351	0.718543
std	1.161452	0.616274	1.006748
min	0.000000	0.000000	0.000000
25%	0.000000	1.000000	0.000000
50%	0.800000	1.000000	0.000000
75%	1.600000	2.000000	1.000000
max	6.200000	2.000000	4.000000

	thalassemia	target
count	302.000000	302.000000
mean	2.314570	0.543046
std	0.613026	0.498970
min	0.000000	0.000000
25%	2.000000	0.000000
50%	2.000000	1.000000

```
75%       3.000000    1.000000
max       3.000000    1.000000
```

[15]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 302 entries, 0 to 302
Data columns (total 14 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
 0   age                                   302 non-null    int64
 1   sex                                   302 non-null    int64
 2   chest_pain_type                       302 non-null    int64
 3   resting_blood_pressure                302 non-null    int64
 4   cholestoral                           302 non-null    int64
 5   fasting_blood_sugar                   302 non-null    int64
 6   resting_electrocardiographic_results  302 non-null    int64
 7   maximum_heart_rate_achieved           302 non-null    int64
 8   exercise_induced_angina               302 non-null    int64
 9   ST.depression(exercise/rest)          302 non-null    float64
 10  slope                                 302 non-null    int64
 11  no_of_major_vessels                   302 non-null    int64
 12  thalassemia                           302 non-null    int64
 13  target                                302 non-null    int64
dtypes: float64(1), int64(13)
memory usage: 35.4 KB
```

[ ]:
```
#Separating numeric and categorical values for calculations
```

[17]:
```
list(enumerate(data))
```

[17]:
```
[(0, 'age'),
 (1, 'sex'),
 (2, 'chest_pain_type'),
 (3, 'resting_blood_pressure'),
 (4, 'cholestoral'),
 (5, 'fasting_blood_sugar'),
 (6, 'resting_electrocardiographic_results'),
 (7, 'maximum_heart_rate_achieved'),
 (8, 'exercise_induced_angina'),
 (9, 'ST.depression(exercise/rest)'),
 (10, 'slope'),
 (11, 'no_of_major_vessels'),
 (12, 'thalassemia'),
 (13, 'target')]
```

[18]:
```
numeric_data= data.iloc[:,[0,3,4,7,9]]
numeric_data.head()
```

[18]:

	age	resting_blood_pressure	cholestoral	maximum_heart_rate_achieved	ST.depression(exercise/rest)
0	63	145	233	150	2.3
1	37	130	250	187	3.5
2	41	130	204	172	1.4
3	56	120	236	178	0.8
4	57	120	354	163	0.6

[19]:
```
categorical_data= data.iloc[:,[1,2,5,6,8,10,11,12,13]]
categorical_data.head()
```

[19]:

	sex	chest_pain_type	fasting_blood_sugar	resting_electrocardiographic_results	exercise_induced_angina	slope	no_of_major_vessels	thalassemia	target
0	1	3	1	0	0	0	0	1	1
1	1	2	0	1	0	0	0	2	1
2	0	1	0	0	0	2	0	2	1
3	1	1	0	1	0	2	0	2	1
4	0	0	0	1	1	2	0	2	1

Measures of central tendencies

[20]:
```
numeric_data.mean()
```

[20]:
```
age                          54.420530
resting_blood_pressure      131.602649
cholestoral                 246.500000
```

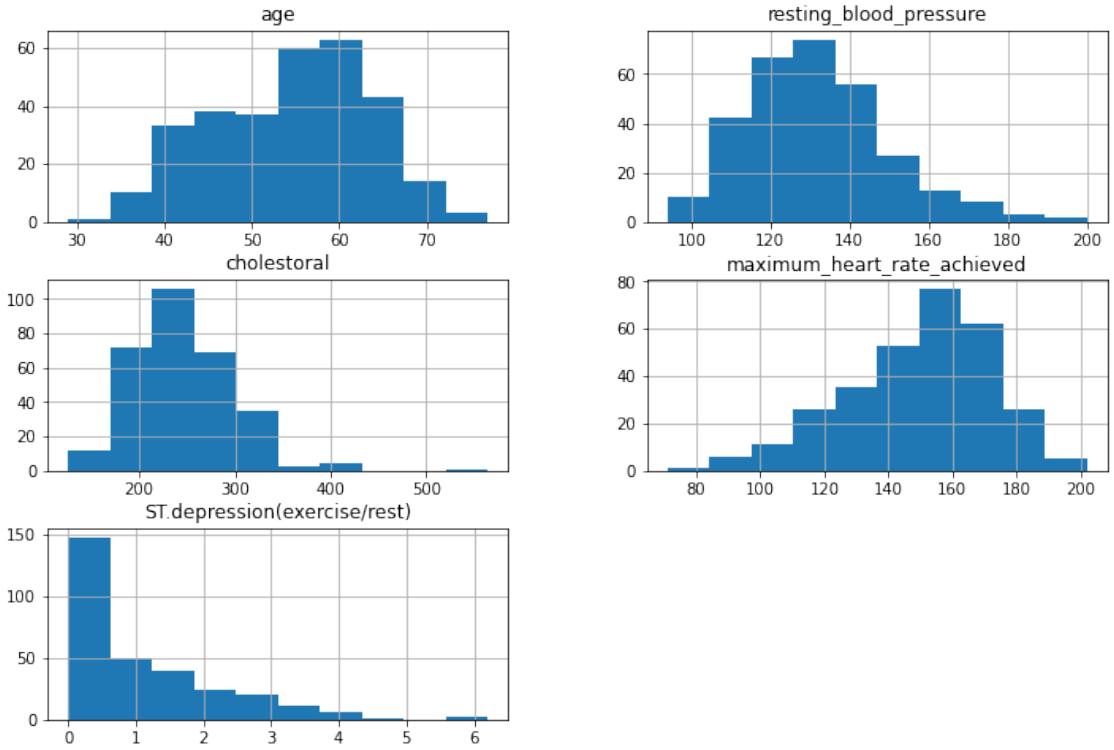
```
maximum_heart_rate_achieved      149.569536
ST.depression(exercise/rest)       1.043046
dtype: float64
```

[21]:
```
numeric_data.median()
```

[21]:
```
age                              55.5
resting_blood_pressure          130.0
cholestoral                     240.5
maximum_heart_rate_achieved     152.5
ST.depression(exercise/rest)      0.8
dtype: float64
```

Spread of the data

[22]:
```
numeric_data.hist(figsize=(12,8))
plt.show()
```



From above graphs, we observe that:

1.ST.depression(exercise/rest) is right skewed. 2.Maximum hear rate achieved is left skewed. 3.Age,Cholestrol,Resting Blood Pressure is normally distributed.

1.1 Performing EDA and Modellling

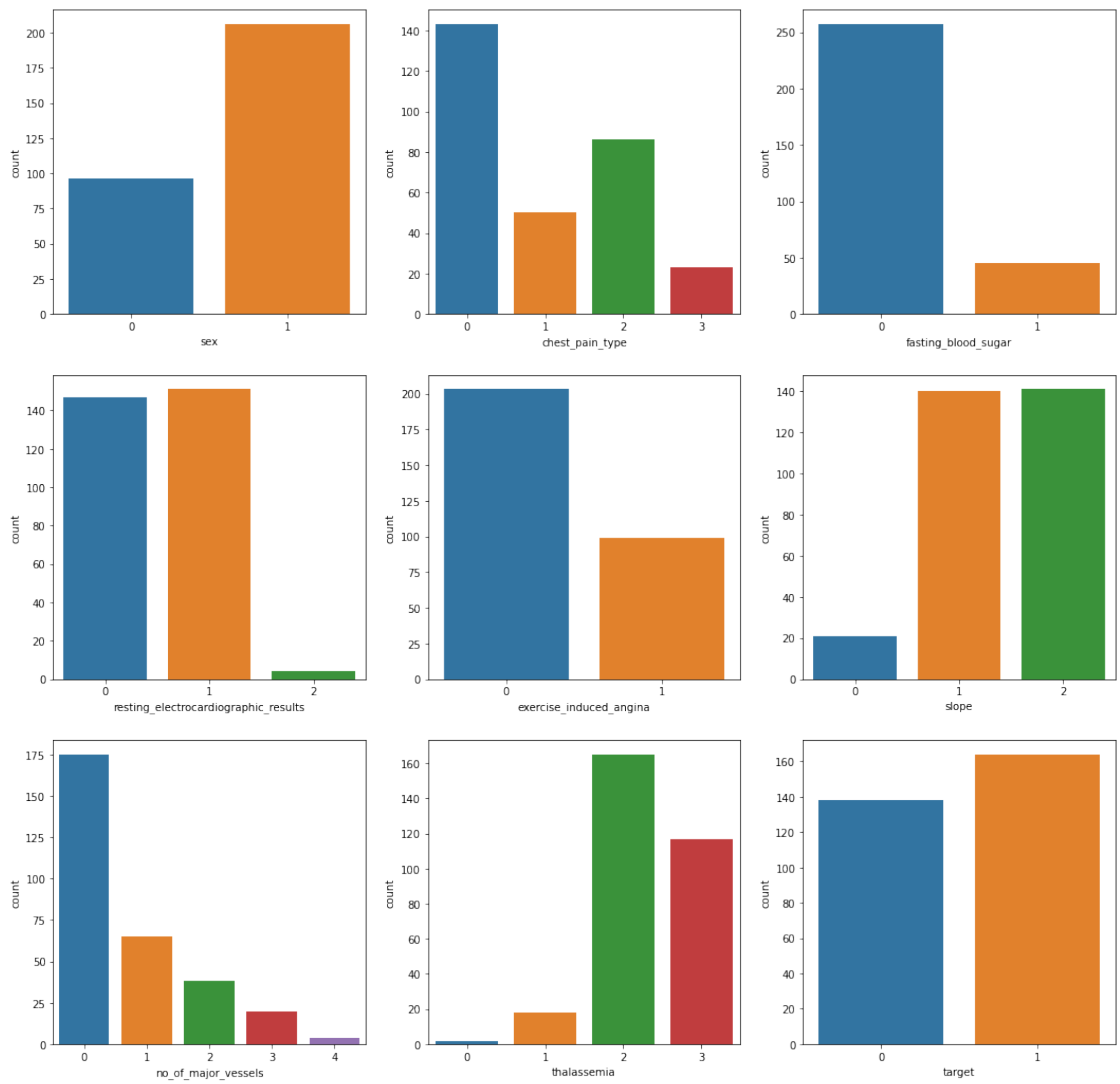
[23]:
```
list(enumerate(categorical_data))
```

[23]:
```
[(0, 'sex'),
 (1, 'chest_pain_type'),
 (2, 'fasting_blood_sugar'),
 (3, 'resting_electrocardiographic_results'),
 (4, 'exercise_induced_angina'),
 (5, 'slope'),
 (6, 'no_of_major_vessels'),
 (7, 'thalassemia'),
 (8, 'target')]
```

[24]:
```
plt.figure(figsize=(18,18))
for i in enumerate(categorical_data):
    plt.subplot(3,3,i[0]+1)
    sns.countplot(i[1],data= categorical_data)
```

```
/usr/local/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning:
Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  FutureWarning
/usr/local/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning:
Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  FutureWarning
/usr/local/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning:
Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  FutureWarning
/usr/local/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning:
Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  FutureWarning
/usr/local/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning:
Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  FutureWarning
/usr/local/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning:
Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
```

```
  FutureWarning
/usr/local/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning:
Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  FutureWarning
/usr/local/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning:
Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  FutureWarning
/usr/local/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning:
Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  FutureWarning
```



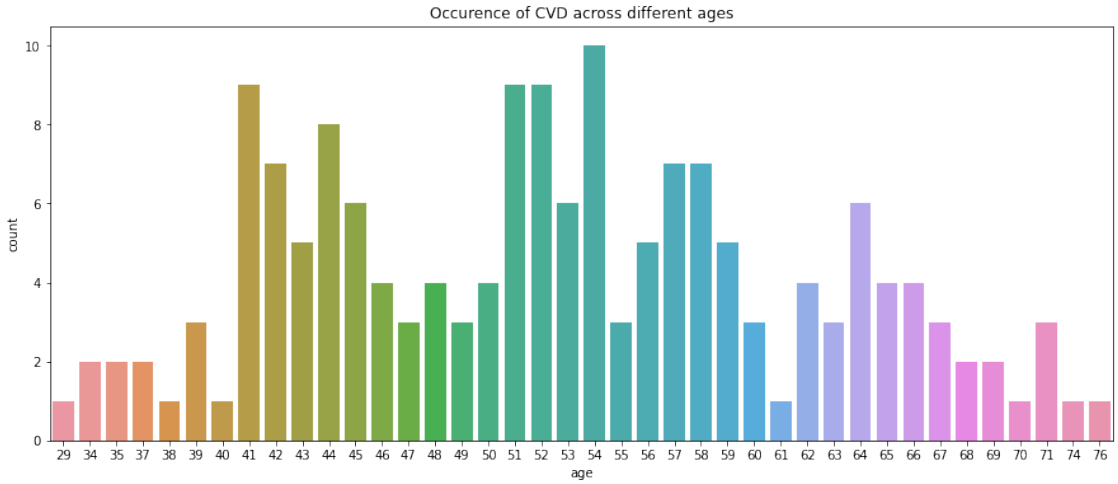
Study the occurrence of CVD across different ages.

[25]:
```
df=data[data.target==1]
```

[26]:
```
df.target.value_counts()
```

[26]:
```
1    164
Name: target, dtype: int64
```

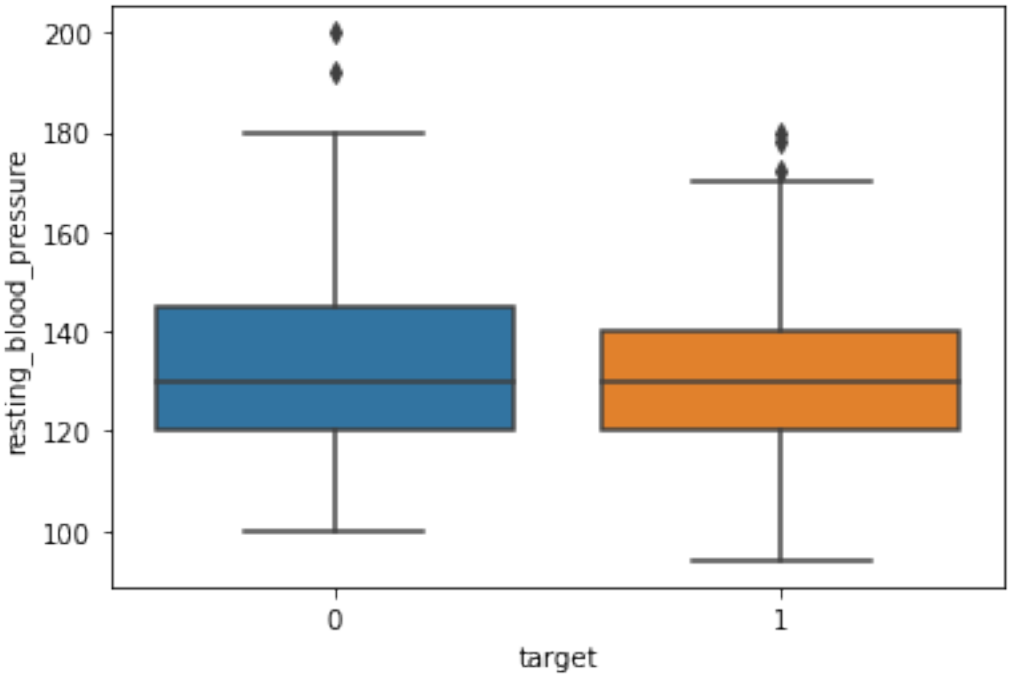
[27]:
```
plt.figure(figsize=(15,6))
sns.countplot(x ="age", data= df)
plt.title("Occurence of CVD across different ages")
plt.show()
```



We can observe that occurence of disease is more in the age group between 40 to 60, though people of age 50-60 are at more risk.

Detection of heart attack based on anomalies in resting blood pressure of the patient

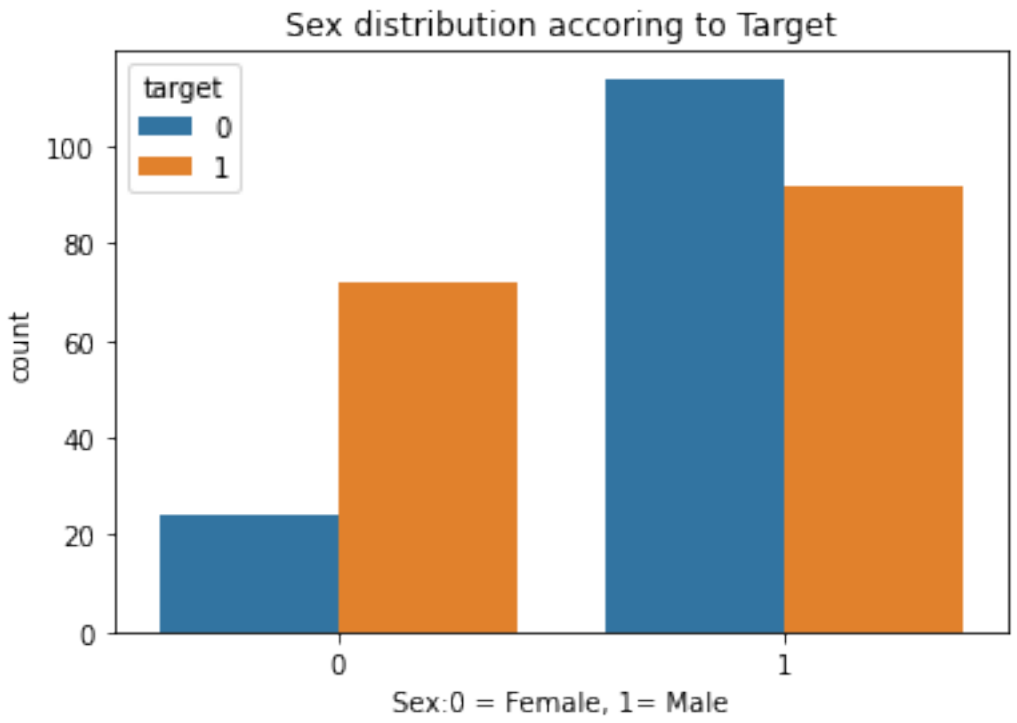
[28]:
```
sns.boxplot(y="resting_blood_pressure", x= "target", data= data)
plt.show()
```



From the above observation, there are people who does not got heart attack also have high blood pressure. Therefore, we can not detect heart attack based on resting blood pressure.

Study the composition of overall patients (Genderwise)

[29]:
```
sns.countplot(x="sex",hue="target", data=data)
plt.title("Sex distribution accoring to Target")
plt.xlabel("Sex:0 = Female, 1= Male")
plt.show()
```



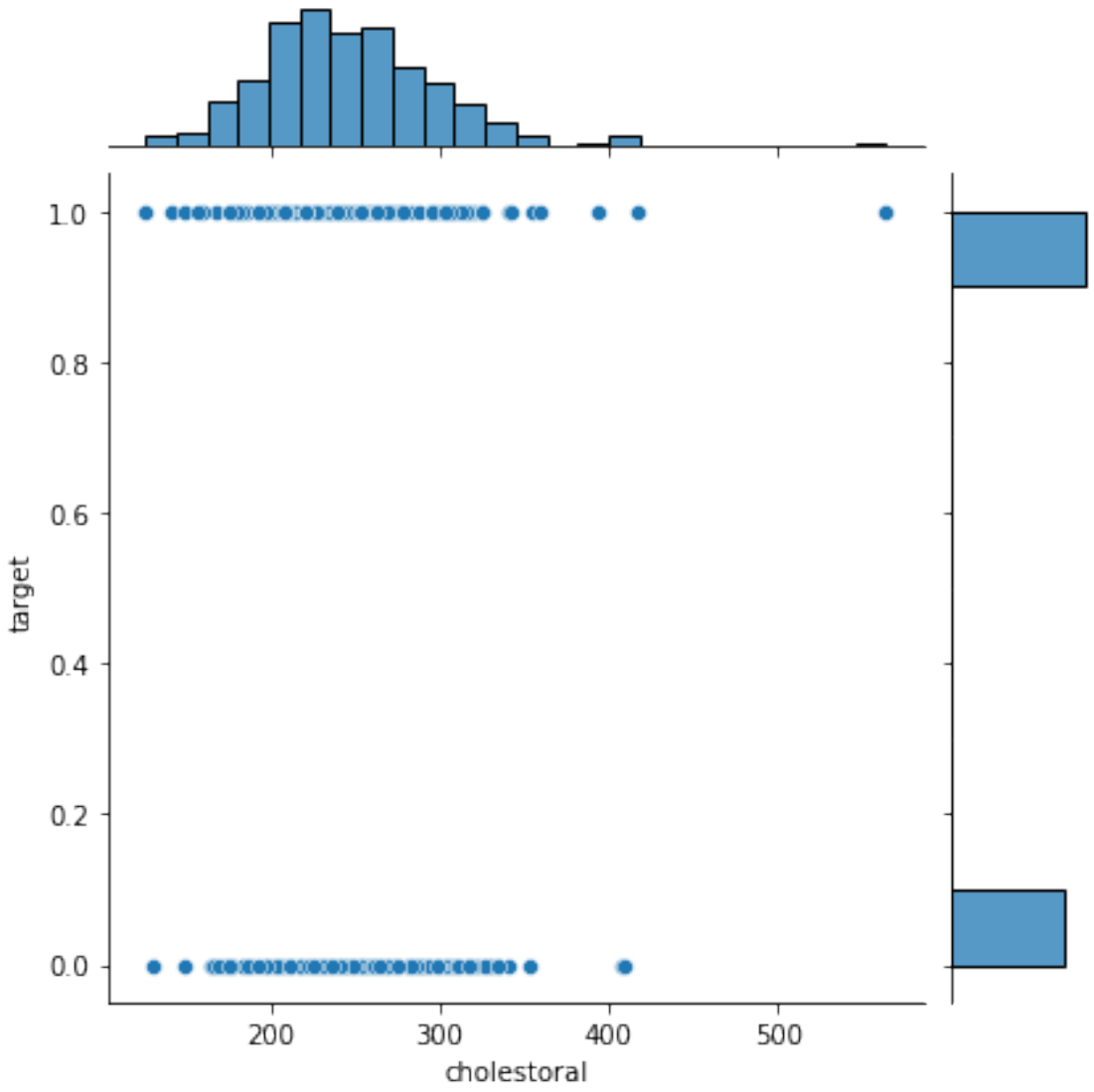
From the above graph it can be concluded that male patients are more prone to the Cardiovascular disease. Target = 0 represent Don't have disease, 1 represent have Disease

The relationship between cholesterol levels and our target variable.

[30]:
```
data.cholestoral.corr(data.target)
```

[30]:
```
-0.08143720051844144
```

[31]:
```
sns.jointplot(data=data,x="cholestoral",y="target")
plt.show()
```



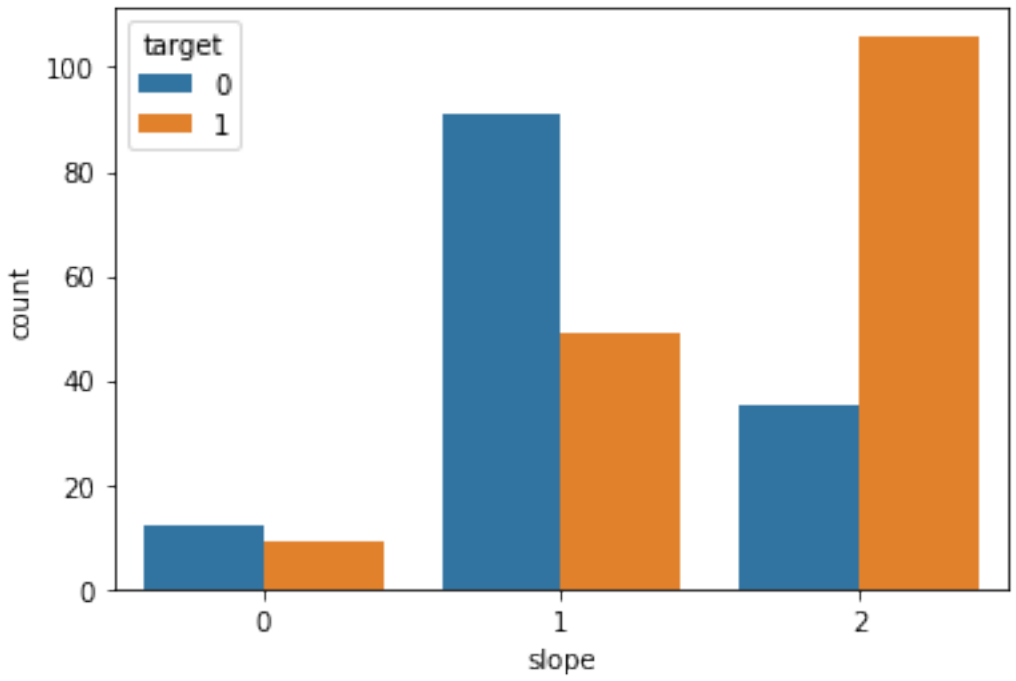
Cholestoral and target variables have weak correlation.

The relationship between peak exercising and occurrence of heart attack

[32]:
```
data.slope.corr(data.target)
```

[32]:
```
0.3439395324893888
```

[33]:
```
sns.countplot(x="slope",hue="target", data=data)
plt.show()
```



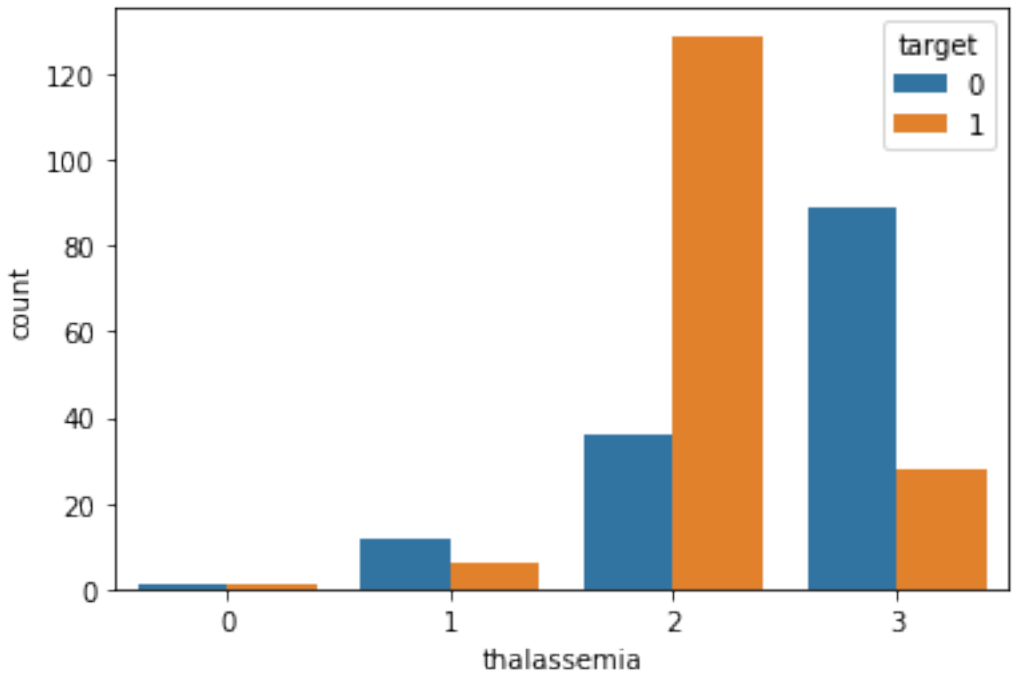
People with Downsloping(2) have more people prone to heart attack. Peak exercising is poitively correlated to the target variable

Determination of thalassemia as a major cause of CVD(if any)

[34]:
```
data.thalassemia.corr(data.target)
```

[34]:
```
-0.3431007123895653
```

[35]:
```
sns.countplot(x="thalassemia", hue="target", data= data)
plt.show()
```

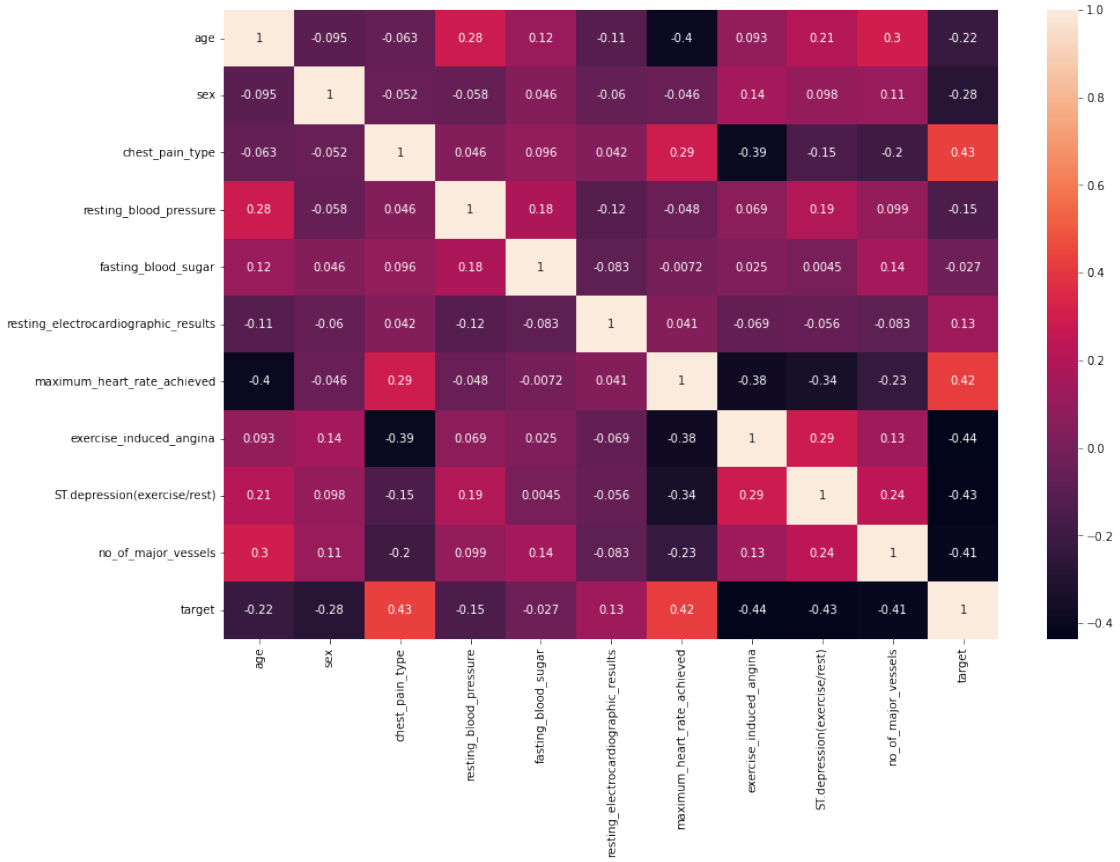


Thalassemia–0=Null, 1= Normal, 2= Fixed Defect, 3=Reversable defect. People with fixed defect are at higher risk of CVD

Roll of the other factors determining the occurrence of CVD

[36]:
```
new_data= data.drop(columns=["thalassemia","cholestoral","slope"])
```

[37]:
```
plt.figure(figsize=(15,10))
sns.heatmap(new_data.corr(),annot=True)
plt.show()
```

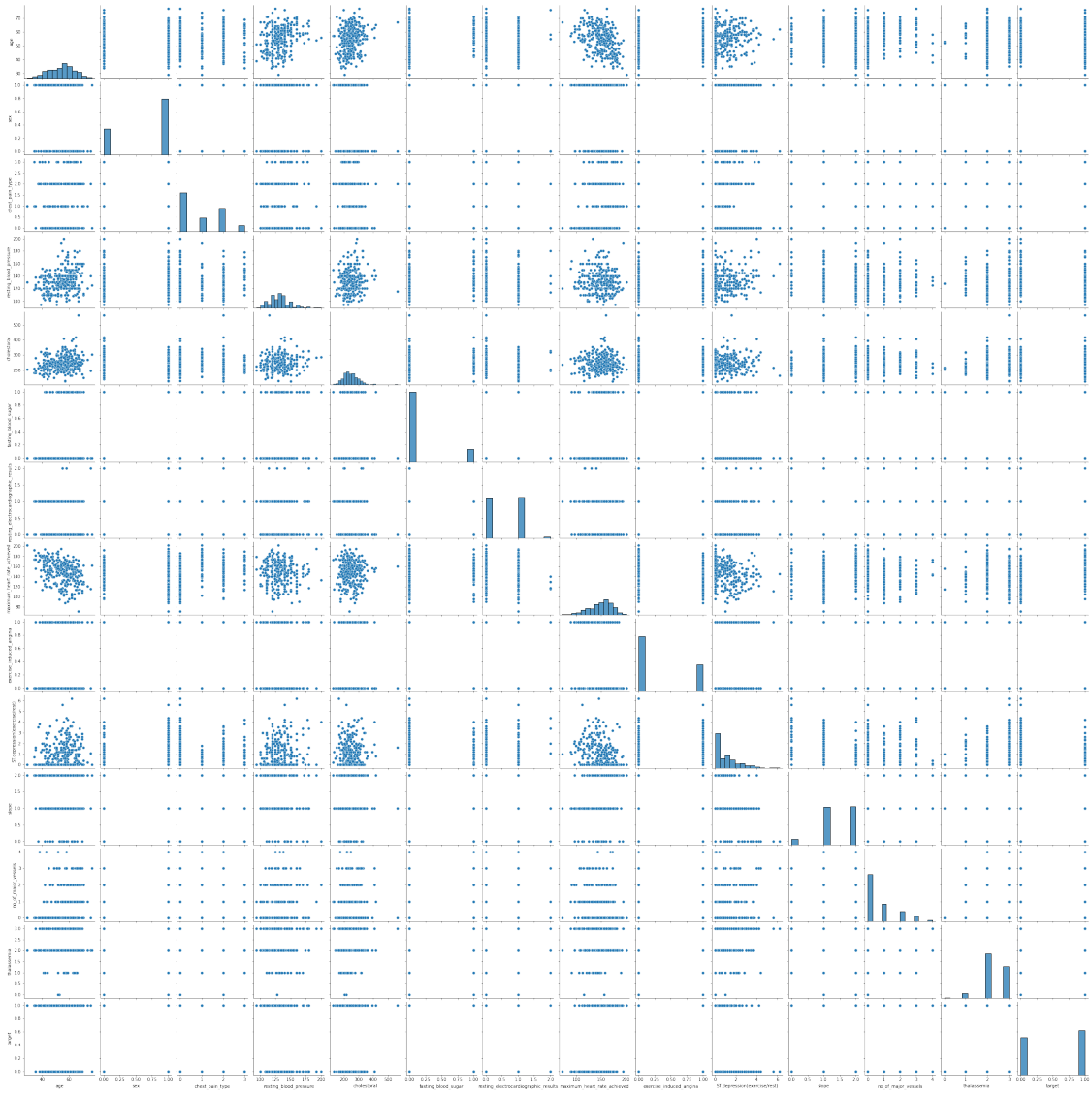


Chest pain type and maximum heart rate achieved are positively correlated to target, and they are the causes of heart attack, there are no major causes as such.

Use a pair plot to understand the relationship between all the given variables.

[38]:
```
plt.figure(figsize=(10,8))
sns.pairplot(data)
plt.show()
```

```
<Figure size 720x576 with 0 Axes>
```



Perform logistic regression, predict the outcome for test data, and validate the results by using the confusion matrix.

[39]:
```
df=data.copy()
```

[40]:
```
df.head()
```

[40]:

	age	sex	chest_pain_type	resting_blood_pressure	cholestoral	\
0	63	1	3	145	233	
1	37	1	2	130	250	
2	41	0	1	130	204	
3	56	1	1	120	236	
4	57	0	0	120	354	

```
   fasting_blood_sugar  resting_electrocardiographic_results  \
0                    1                                    0
1                    0                                    1
2                    0                                    0
3                    0                                    1
4                    0                                    1

   maximum_heart_rate_achieved  exercise_induced_angina  \
0                          150                        0
1                          187                        0
2                          172                        0
3                          178                        0
4                          163                        1

   ST.depression(exercise/rest)  slope  no_of_major_vessels  thalassemia  \
0                           2.3      0                    0            1
1                           3.5      0                    0            2
2                           1.4      2                    0            2
3                           0.8      2                    0            2
4                           0.6      2                    0            2

   target
0       1
1       1
2       1
3       1
4       1
```

[42]:
```
x= df.drop(["target"], axis=1)
y=df.target
```

[43]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,␣
 ↪random_state=1234)
```

[44]:
```
x_train.shape, x_test.shape, y_train.shape,y_test.shape
```

[44]:
```
((241, 13), (61, 13), (241,), (61,))
```

For training we have 241 data points and for testing we have 61

[45]:
```
from sklearn.linear_model import LogisticRegression
log= LogisticRegression()
```

[46]:
```
log.fit(x_train,y_train)
```

```
/usr/local/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:818:
```

```
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
```

[46]:
```
LogisticRegression()
```

[47]:
```
y_pred=log.predict(x_test)
y_pred
```

[47]:
```
array([0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1,
       1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1,
       1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0])
```

[ ]:
```

```

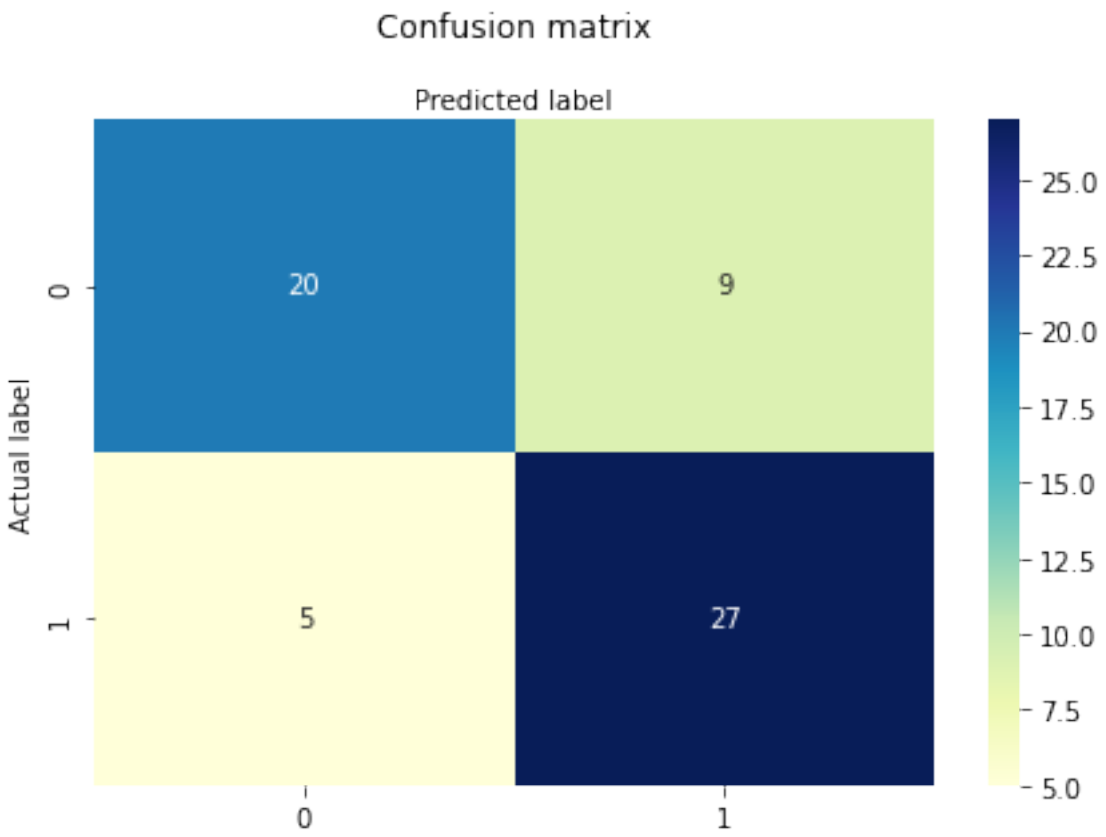
[48]:
```
from sklearn.metrics import confusion_matrix
conf_mat = confusion_matrix(y_test,y_pred)
```

[49]:
```
conf_mat
```

[49]:
```
array([[20,  9],
       [ 5, 27]])
```

[51]:
```
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
sns.heatmap(conf_mat, annot = True,cmap="YlGnBu" ,fmt='g')
ax.xaxis.set_label_position("top")
plt.tight_layout()
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
```

[51]:
```
Text(0.5, 257.44, 'Predicted label')
```



[52]:
```
from sklearn.metrics import accuracy_score,precision_score,recall_score,f1_score
print('Accuracy score: ',accuracy_score(y_test,y_pred))
print('Precision score: ',precision_score(y_test,y_pred))
print('Precision score: ',precision_score(y_test,y_pred))
print('Recall score: ',recall_score(y_test,y_pred))
print('F1 score: ',f1_score(y_test,y_pred))
```

```
Accuracy score:  0.7704918032786885
Precision score:  0.75
Precision score:  0.75
Recall score:  0.84375
F1 score:  0.7941176470588235
```

Accuracy for the logistic regression model is 77%.

#Dashboarding#

Visualize the variables using Tableau to create an understanding for attributes of a Diseased vs. a Healthy person

Attributes of a Diseased vs. a Healthy person

Demonstrate the variables associated with each other and factors to build a dashboard **Correlation**

among the Variables