IMPORTING THE DEPENDENCIES

In [2]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

DATA COLLECTION AND PROCESSING

In [2]:

```
#Loading the csv data to a Pandas Dataframe.
```

In [4]:

```
heart_data =pd.read_csv("healthcare.csv")
```

In [5]:

```
# Getting first 5 rows
heart_data.head()
```

Out[5]:

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

In [7]:
```
# Getting last 5 rows
heart_data.tail()
```

Out[7]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	ta
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3	
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3	
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3	
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	

In [9]:
```
# Checking No. of rows and columns
heart_data.shape
```

Out[9]: (303, 14)

In [10]:
```
# Getting the information about the data.
heart_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       303 non-null    int64
 1   sex       303 non-null    int64
 2   cp        303 non-null    int64
 3   trestbps  303 non-null    int64
 4   chol      303 non-null    int64
 5   fbs       303 non-null    int64
 6   restecg   303 non-null    int64
 7   thalach   303 non-null    int64
 8   exang     303 non-null    int64
 9   oldpeak   303 non-null    float64
 10  slope     303 non-null    int64
 11  ca        303 non-null    int64
 12  thal      303 non-null    int64
 13  target    303 non-null    int64
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

In [11]:
```
# Checking for missing values
heart_data.isnull().sum()
```

Out[11]:
```
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
target      0
dtype: int64
```

In [12]:
```
# STATISTICAL MEASURES .
heart_data.describe()
```

Out[12]:

chol	fbs	restecg	thalach	exang	oldpeak	slope	
303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.0000
246.264026	0.148515	0.528053	149.646865	0.326733	1.039604	1.399340	0.7293
51.830751	0.356198	0.525860	22.905161	0.469794	1.161075	0.616226	1.0226
126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	0.000000	0.0000
211.000000	0.000000	0.000000	133.500000	0.000000	0.000000	1.000000	0.0000
240.000000	0.000000	1.000000	153.000000	0.000000	0.800000	1.000000	0.0000
274.500000	0.000000	1.000000	166.000000	1.000000	1.600000	2.000000	1.0000
564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	2.000000	4.0000

In [14]:
```
heart_data.columns
```

Out[14]:
```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')
```

In [15]:
```
df2 = heart_data.copy()
```

In [10]:
```
# To check the distribution of the Target variable
heart_data['target'].value_counts()   #  gives the information of 0 and 1
```

Out[10]:
```
1    165
0    138
Name: target, dtype: int64
```

In [ ]:
```
1--- Defective heart
0---  Healthy heart
```

Splitting the Features in Target

In [16]:
```
X= heart_data.drop(columns='target', axis=1)
Y = heart_data['target']
```

In [17]:
```
print(X)
```

```
     age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  \
0     63    1   3       145   233    1        0      150      0      2.3
1     37    1   2       130   250    0        1      187      0      3.5
2     41    0   1       130   204    0        0      172      0      1.4
3     56    1   1       120   236    0        1      178      0      0.8
4     57    0   0       120   354    0        1      163      1      0.6
..   ...  ...  ..       ...   ...  ...      ...      ...    ...      ...
298   57    0   0       140   241    0        1      123      1      0.2
299   45    1   3       110   264    0        1      132      0      1.2
300   68    1   0       144   193    1        1      141      0      3.4
301   57    1   0       130   131    0        1      115      1      1.2
302   57    0   1       130   236    0        0      174      0      0.0

     slope  ca  thal
0        0   0     1
1        0   0     2
2        2   0     2
3        2   0     2
4        2   0     2
..     ...  ..   ...
298      1   0     3
299      1   0     3
300      1   2     3
301      1   1     3
302      1   1     2

[303 rows x 13 columns]
```

In [18]:
```
print(Y)
```

```
0      1
1      1
2      1
3      1
4      1
      ..
298    0
299    0
300    0
301    0
302    0
Name: target, Length: 303, dtype: int64
```

In [84]:
```
heart_data.shape
```

Out[84]:
```
(303, 14)
```

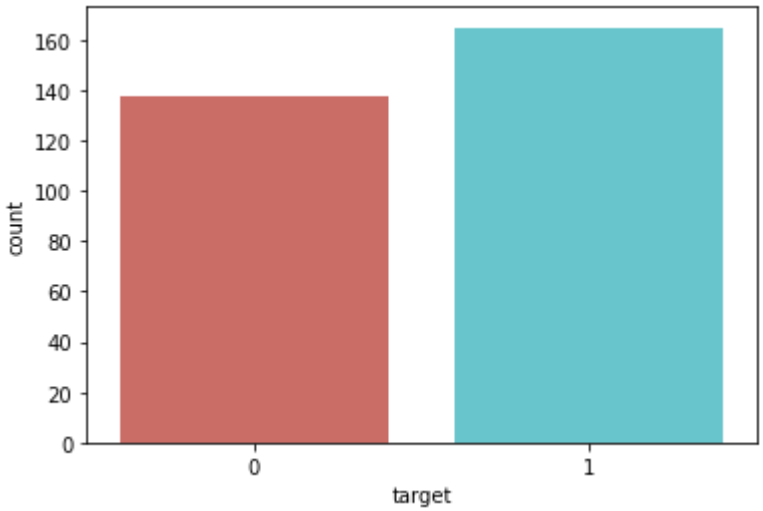
In [35]:
```
target_counts = heart_data.target.value_counts()
```

In [36]:
```
print('Class 0:', target_counts[0])
print('Class 1:', target_counts[1])
print('Proportion:', round(target_counts[0] / target_counts[1], 2), ': 1
```

```
Class 0: 138
Class 1: 165
Proportion: 0.84 : 1
```

In [37]:

```
sns.countplot(x='target', data = heart_data, palette = 'hls')
plt.show()
# sns.histplot(x='target',data=heart_data)
```



In [38]:

```
print(heart_data.apply(lambda col: col.unique()))
```

```
age         [63, 37, 41, 56, 57, 44, 52, 54, 48, 49, 64, 5...
sex                                                   [1, 0]
cp                                                [3, 2, 1, 0]
trestbps    [145, 130, 120, 140, 172, 150, 110, 135, 160, ...
chol        [233, 250, 204, 236, 354, 192, 294, 263, 199, ...
fbs                                                   [1, 0]
restecg                                              [0, 1, 2]
thalach     [150, 187, 172, 178, 163, 148, 153, 173, 162, ...
exang                                                 [0, 1]
oldpeak     [2.3, 3.5, 1.4, 0.8, 0.6, 0.4, 1.3, 0.0, 0.5, ...
slope                                                [0, 2, 1]
ca                                              [0, 2, 1, 3, 4]
thal                                               [1, 2, 3, 0]
target                                                [1, 0]
dtype: object
```

In [39]:

```
heart_data.groupby('target').mean()
```

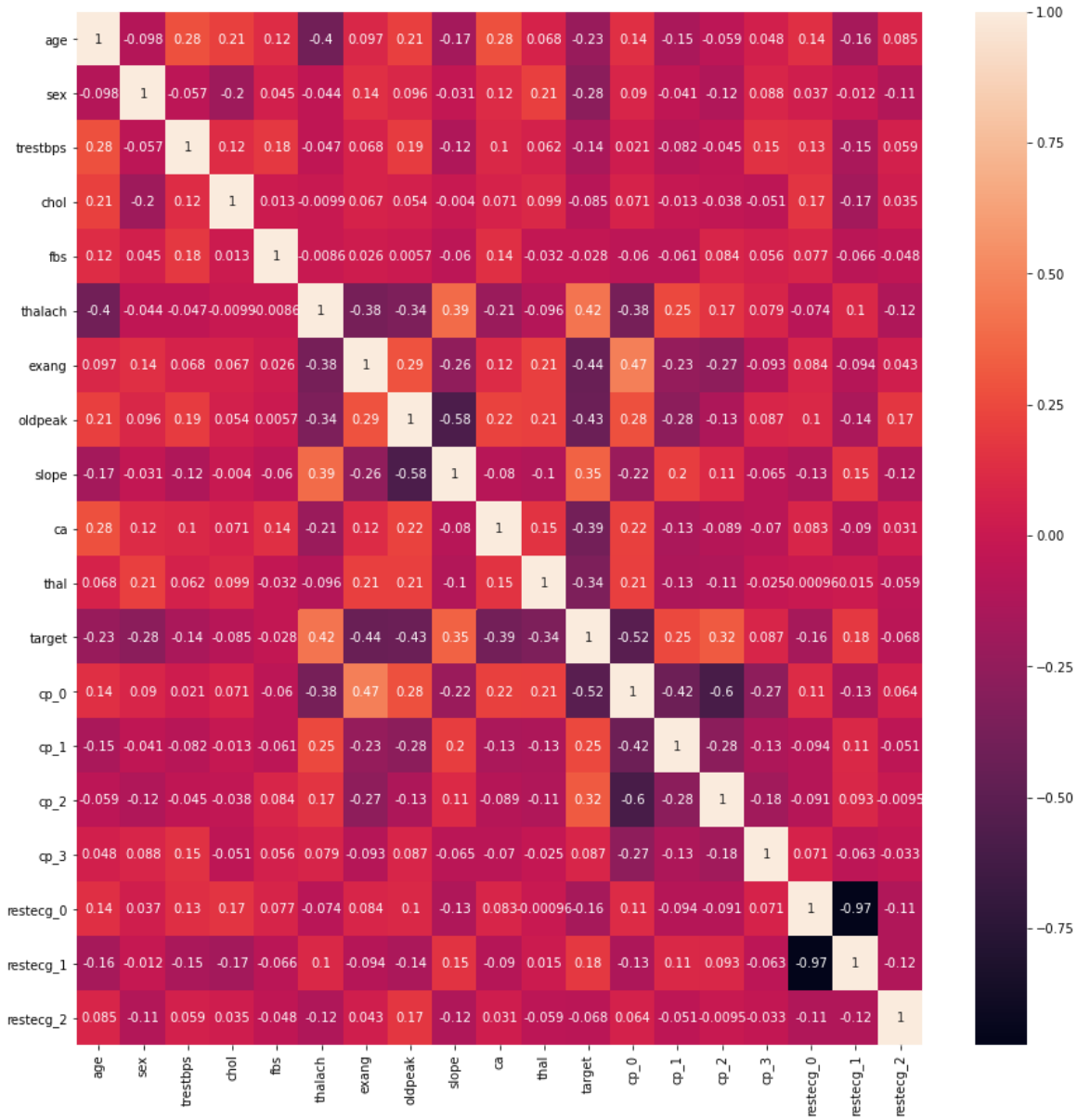
Out[39]:

target	age	sex	cp	trestbps	chol	fbs	restecg	thalach
0	56.601449	0.826087	0.478261	134.398551	251.086957	0.159420	0.449275	139.101449
1	52.496970	0.563636	1.375758	129.303030	242.230303	0.139394	0.593939	158.466667

In [54]:

```
fig, ax = plt.subplots(figsize=(15,15))
sns.heatmap(heart_data.corr(),annot=True,ax=ax)
```

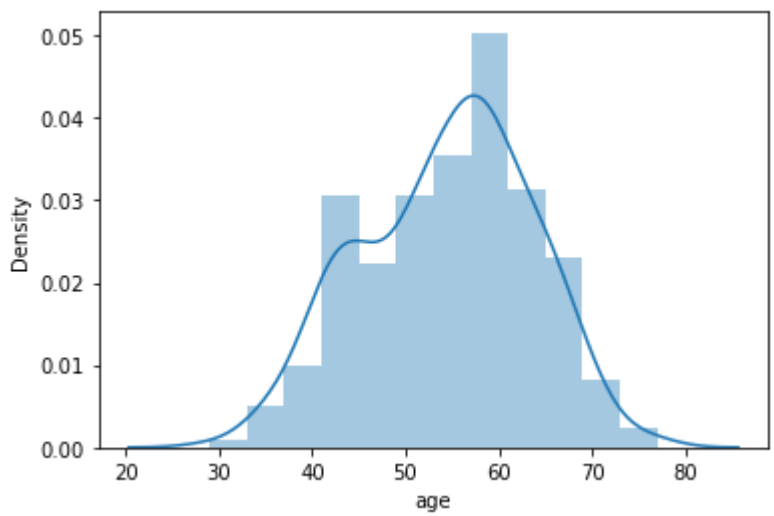
Out[54]: <AxesSubplot:>



In [40]:
```
sns.distplot(heart_data.age)
```

```
C:\Users\Kannagi\anaconda3\lib\site-packages\seaborn\distributions.py:2557:
FutureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-le
vel function with similar flexibility) or `histplot` (an axes-level functio
n for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[40]:
```
<AxesSubplot:xlabel='age', ylabel='Density'>
```



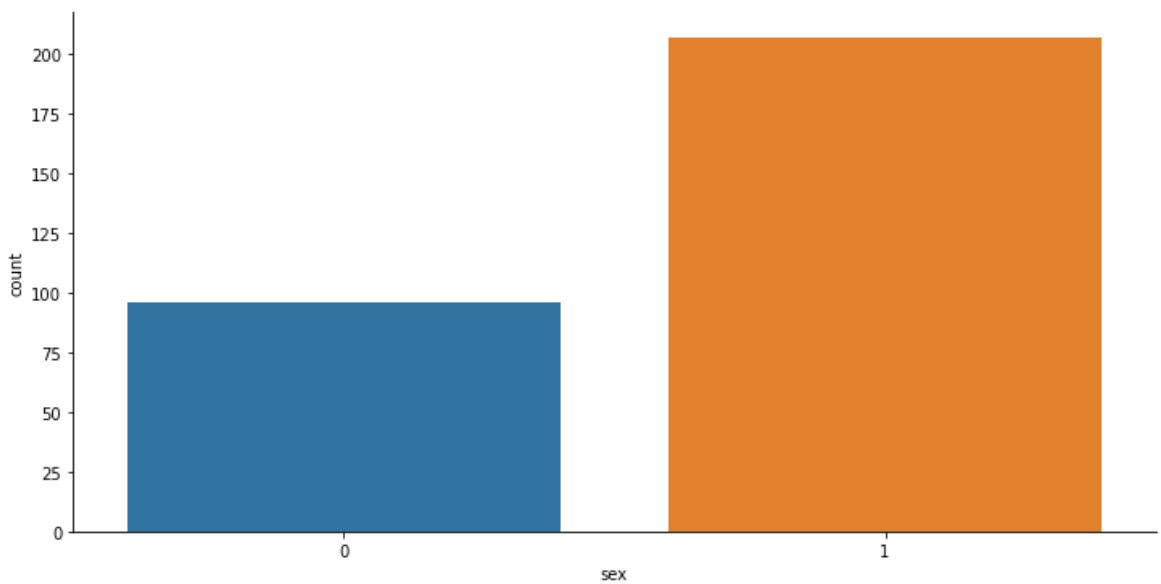
In [41]:

```
# There are significantly more men in the data than women
sns.factorplot('sex', data = heart_data, kind = 'count', aspect = 2.0)
```

```
C:\Users\Kannagi\anaconda3\lib\site-packages\seaborn\categorical.py:3714: U
serWarning: The `factorplot` function has been renamed to `catplot`. The or
iginal name will be removed in a future release. Please update your code. N
ote that the default `kind` in `factorplot` (`'point'`) has changed `'stri
p'` in `catplot`.
  warnings.warn(msg)
C:\Users\Kannagi\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Fut
ureWarning: 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 misinterpr
etation.
  warnings.warn(
```

Out[41]:

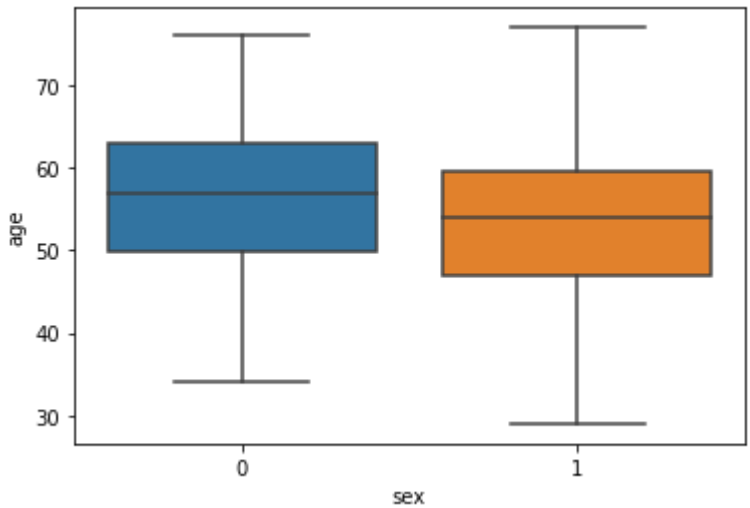
```
<seaborn.axisgrid.FacetGrid at 0x21d99fa76a0>
```



In [42]:

```
# A box plot to show the ranges of ages of the women and men in the data
sns.boxplot(data = heart_data, x = 'sex', y = 'age')
```

Out[42]: <AxesSubplot:xlabel='sex', ylabel='age'>



In [43]:

```
hist_df = heart_data.groupby('target')
```

In [44]:

```
hist_df.head()
```

Out[44]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	ta
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	
165	67	1	0	160	286	0	0	108	1	1.5	1	3	2	
166	67	1	0	120	229	0	0	129	1	2.6	1	2	3	
167	62	0	0	140	268	0	0	160	0	3.6	0	2	2	
168	63	1	0	130	254	0	0	147	0	1.4	1	1	3	
169	53	1	0	140	203	1	0	155	1	3.1	0	0	3	

In [45]:

```
heart_data.head()
```

Out[45]:

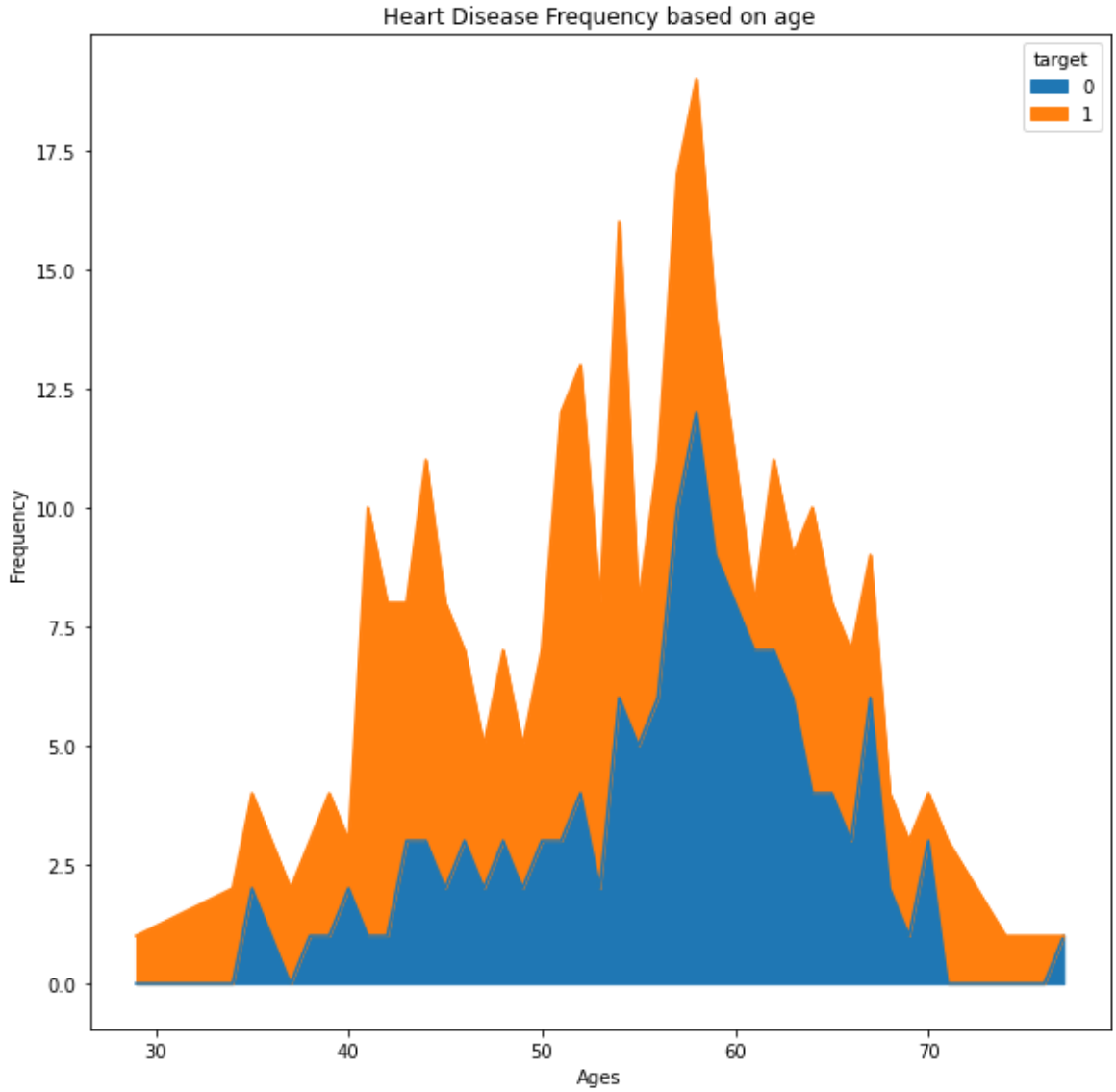
	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	targe
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	

In [ ]:

```
# Combining the Age with Target columns, we will see the frequency of hea
```

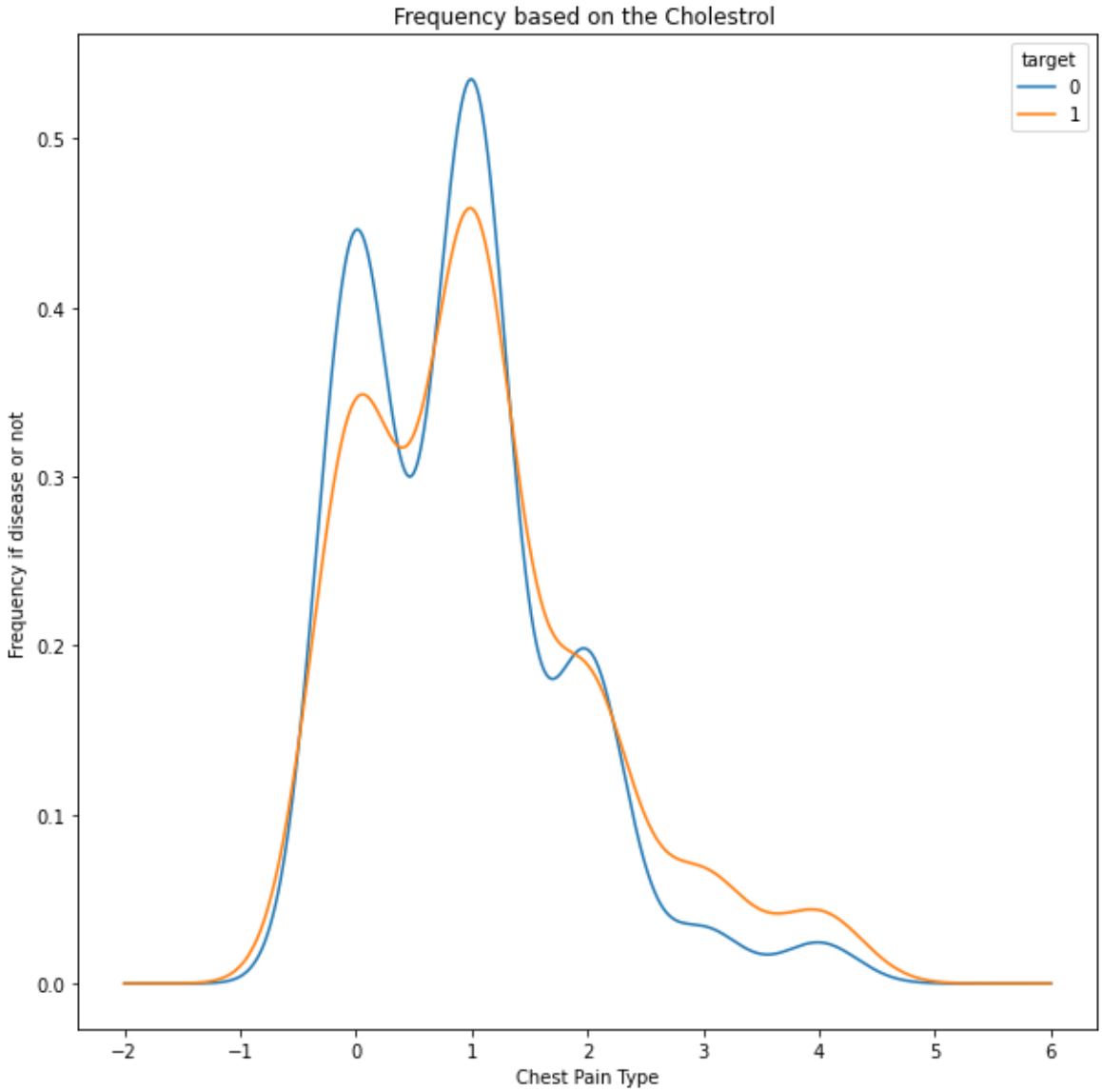
In [89]:

```
pd.crosstab(heart_data.age, heart_data.target).plot(kind= "area", figsize
plt.title('Heart Disease Frequency based on age')
plt.xlabel('Ages')
plt.ylabel('Frequency')
plt.show()
```



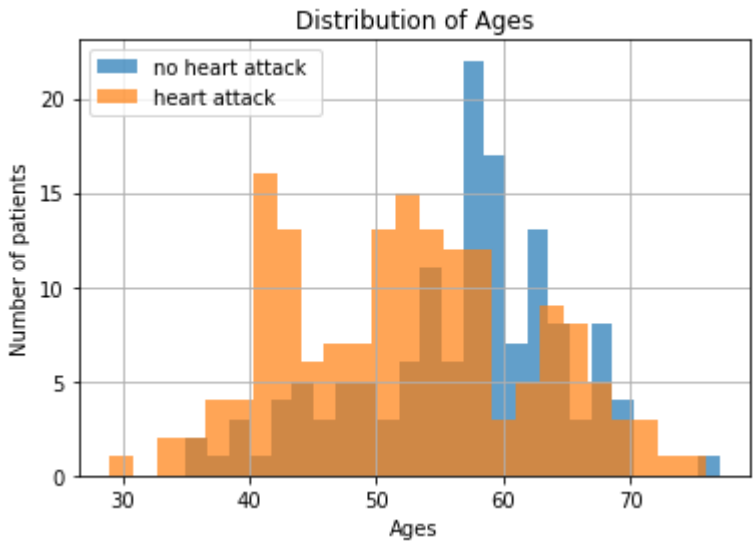
In [91]:

```
pd.crosstab(heart_data.chol, heart_data.target).plot(kind= "kde", figsize
plt.title('Frequency based on the Cholestrol ')
plt.xlabel('Chest Pain Type')
plt.xticks(rotation = 0)
plt.ylabel('Frequency if disease or not')
plt.show()
```



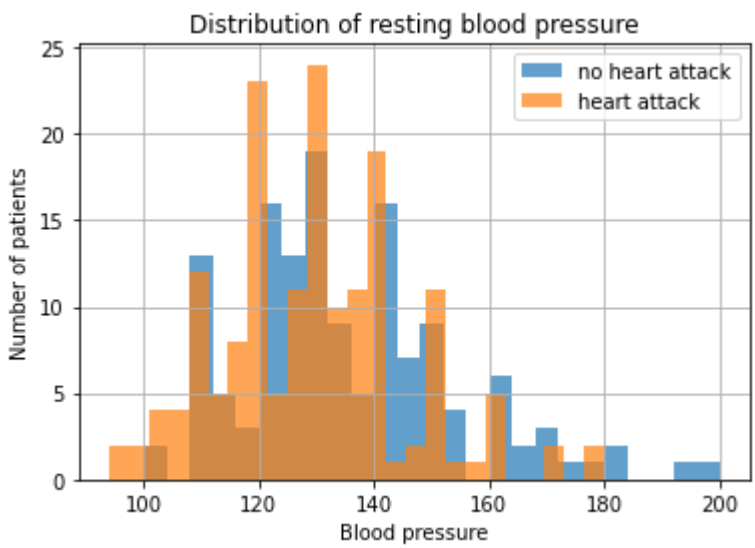
In [46]:

```
# 2. A distribution of the ages of patients that suffered a heart attack
hist_df['age'].hist(bins=25, alpha=0.7)
plt.title('Distribution of Ages')
plt.xlabel('Ages')
plt.ylabel('Number of patients')
plt.legend(('no heart attack ', 'heart attack'), loc = 'upper left')
plt.show()
```



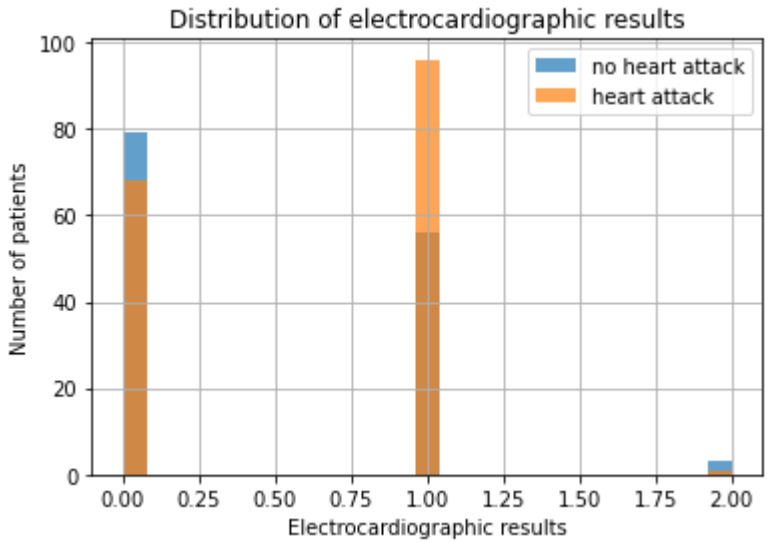
In [47]:

```
# 3
hist_df['trestbps'].hist(bins=25, alpha=0.7)
plt.title('Distribution of resting blood pressure')
plt.xlabel('Blood pressure')
plt.ylabel('Number of patients')
plt.legend(('no heart attack', 'heart attack'), loc = 'upper right')
plt.show()
```



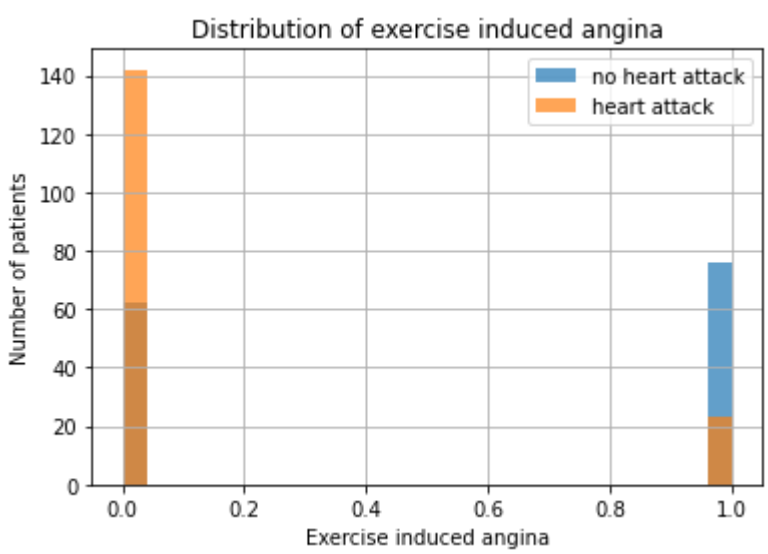
In [48]:

```
hist_df['restecg'].hist(bins = 25, alpha = 0.7)
plt.title('Distribution of electrocardiographic results')
plt.xlabel('Electrocardiographic results')
plt.ylabel('Number of patients')
plt.legend(('no heart attack', 'heart attack'), loc = 'upper right')
plt.show()
```



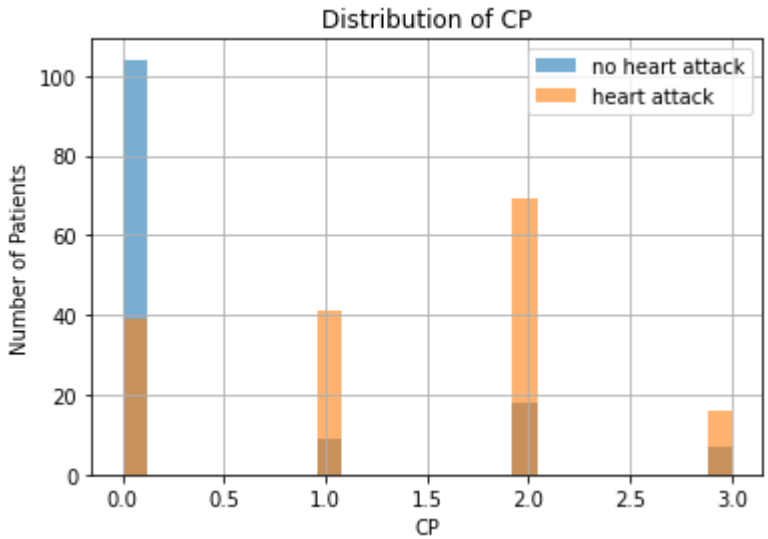
In [49]:

```
hist_df['exang'].hist(bins = 25, alpha = 0.7)
plt.title('Distribution of exercise induced angina')
plt.xlabel('Exercise induced angina')
plt.ylabel('Number of patients')
plt.legend(('no heart attack', 'heart attack'), loc = 'upper right')
plt.show()
```



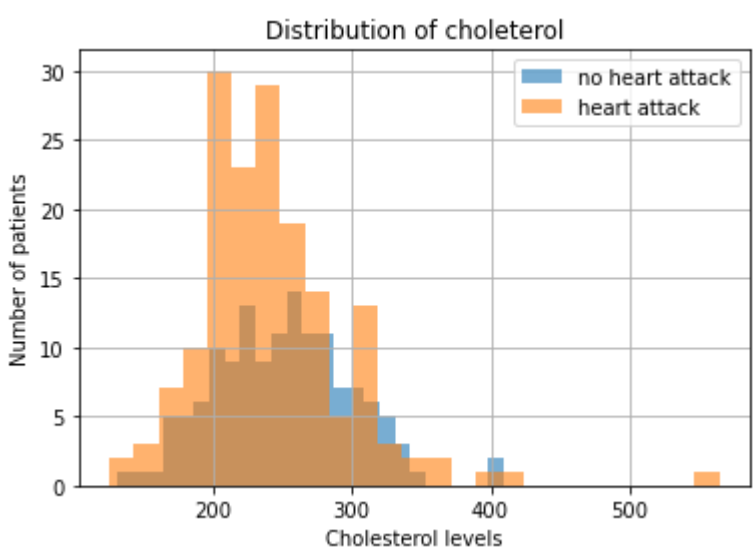
In [50]:

```
hist_df['cp'].hist(bins = 25, alpha=0.6)
plt.title('Distribution of CP')
plt.xlabel('CP')
plt.ylabel('Number of Patients')
plt.legend(('no heart attack', 'heart attack'))
plt.show()
```



In [51]:

```
hist_df['chol'].hist(bins = 25, alpha=0.6)
plt.title('Distribution of choleterol')
plt.xlabel('Cholesterol levels')
plt.ylabel('Number of patients')
plt.legend(('no heart attack', 'heart attack'), loc = 'upper right')
plt.show()
```



In [52]:

```
def get_gender(sex_value):

    gender_string = 'Male'

    if sex_value == 0:
        gender_string = 'Female'
    return gender_string

def get_cp_category(cp_value):

    if cp_value == 0:
        cp_string = 'typical'
    elif cp_value == 1:
        cp_string = 'atypical'
    elif cp_value == 2:
        cp_string = 'non_anginal'
    elif cp_value == 3:
        cp_string = 'asmptomatic'

    return(cp_string)
```

In [53]:

```
subset_df = heart_data[['sex', 'cp']]
```

In [54]:

```
subset_df.head()
```

Out[54]:

	sex	cp
0	1	3
1	1	2
2	0	1
3	1	1
4	0	0

In [55]:

```
subset_df['sex'] = subset_df.sex.map(lambda x: get_gender(x))
subset_df['cp'] = subset_df.cp.map(lambda x: get_cp_category(x))
```

```
<ipython-input-55-7af9cdc90cd8>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  subset_df['sex'] = subset_df.sex.map(lambda x: get_gender(x))
<ipython-input-55-7af9cdc90cd8>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  subset_df['cp'] = subset_df.cp.map(lambda x: get_cp_category(x))
```

In [56]:

```
subset_df.head()
```

Out[56]:

	sex	cp
0	Male	asmptomatic
1	Male	non_anginal
2	Female	atypical
3	Male	atypical
4	Female	typical

In [57]:

```
dum_df = pd.get_dummies(subset_df[['sex', 'cp']])
```

In [58]:

```
dum_df.head()
```

Out[58]:

	sex_Female	sex_Male	cp_asmptomatic	cp_atypical	cp_non_anginal	cp_typical
0	0	1	1	0	0	0
1	0	1	0	0	1	0
2	1	0	0	1	0	0
3	0	1	0	1	0	0
4	1	0	0	0	0	1

In [59]:
```
dum_df = dum_df.drop(['sex_Female'], axis = 1)
```

In [60]:
```
dum_df.rename(columns = {'sex_Male': 'Male'}, inplace = True)
```

In [62]:
```
df = pd.concat([heart_data, dum_df], axis = 1)
```

In [63]:
```
df.head()
```

Out[63]:

x	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target	Male	cp_
1	3	145	233	1	0	150	0	2.3	0	0	1	1	1	
1	2	130	250	0	1	187	0	3.5	0	0	2	1	1	
0	1	130	204	0	0	172	0	1.4	2	0	2	1	0	
1	1	120	236	0	1	178	0	0.8	2	0	2	1	1	
0	0	120	354	0	1	163	1	0.6	2	0	2	1	0	

In [64]:
```
df = df.drop(['sex', 'cp'], axis = 1)
```

In [65]:
```
df.columns
```

Out[65]:
```
Index(['age', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach', 'exang',
       'oldpeak', 'slope', 'ca', 'thal', 'target', 'Male', 'cp_asmptomati
c',
       'cp_atypical', 'cp_non_anginal', 'cp_typical'],
      dtype='object')
```

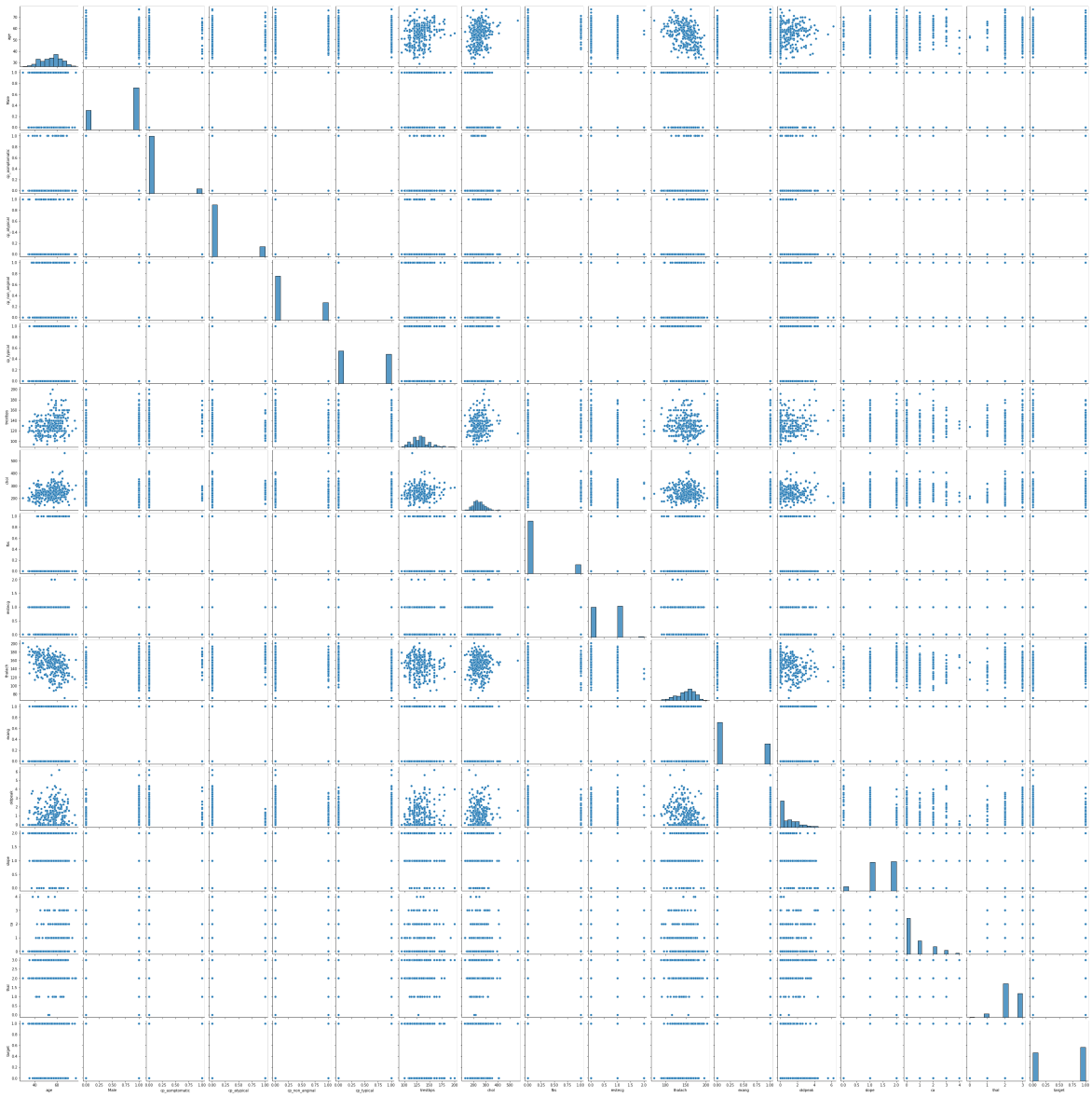
In [66]:
```
df = df[['age', 'Male', 'cp_asmptomatic','cp_atypical', 'cp_non_anginal',
        'oldpeak', 'slope', 'ca', 'thal', 'target']]
```

In [67]:
```
df.columns
```

Out[67]:
```
Index(['age', 'Male', 'cp_asmptomatic', 'cp_atypical', 'cp_non_anginal',
       'cp_typical', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach', 'exan
g',
       'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')
```

In [69]:
```
sns.pairplot(df)
```

Out[69]:
```
<seaborn.axisgrid.PairGrid at 0x21da596d460>
```



In [70]:
```
X = df.iloc[:, :-1].values
```

In [71]:

```
X
```

Out[71]:

```
array([[63.,  1.,  1., ...,  0.,  0.,  1.],
       [37.,  1.,  0., ...,  0.,  0.,  2.],
       [41.,  0.,  0., ...,  2.,  0.,  2.],
       ...,
       [68.,  1.,  0., ...,  1.,  2.,  3.],
       [57.,  1.,  0., ...,  1.,  1.,  3.],
       [57.,  0.,  0., ...,  1.,  1.,  2.]])
```

In [72]:

```
Y = df.iloc[:, 16].values
```

In [73]:

```
Y
```

Out[73]:

```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], dtype=int64)
```

Splitting the Data into training data and test Data

In [74]:

```
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.3,
```

In [81]:

```
print(X.shape, X_train.shape, X_test.shape)
```

```
(303, 16) (212, 16) (91, 16)
```

Training the LogisticRegression model with Training data

In [76]:
```
logistic_model.fit(X_train, y_train)
```

```
C:\Users\Kannagi\anaconda3\lib\site-packages\sklearn\linear_model\_logisti
c.py:763: 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 (https://sci
kit-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-regr
ession (https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression)
  n_iter_i = _check_optimize_result(
```

Out[76]:
```
LogisticRegression()
```

In [77]:
```
logistic_predictions = logistic_model.predict(X_test)
```

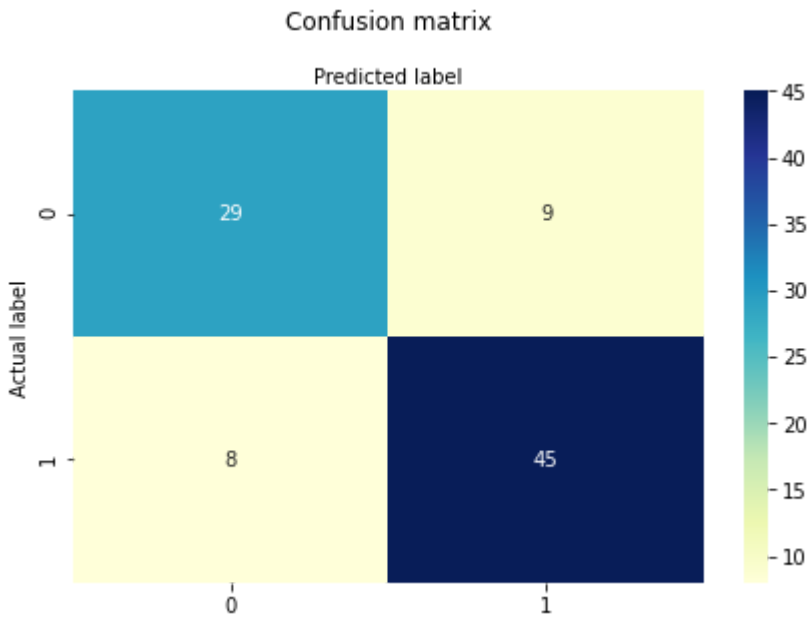
In [78]:
```
from sklearn import metrics
cnf_matrix = metrics.confusion_matrix(y_test, logistic_predictions)
cnf_matrix
```

Out[78]:
```
array([[29,  9],
       [ 8, 45]], dtype=int64)
```

In [79]:

```
# cm = confusion_matrix(y_valid, y_pred)
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
sns.heatmap(cnf_matrix, 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')
```

Out[79]: Text(0.5, 257.44, 'Predicted label')



In [80]:

```
print("Accuracy:",metrics.accuracy_score(y_test, logistic_predictions))
print("Precision:",metrics.precision_score(y_test, logistic_predictions))
print("Recall:",metrics.recall_score(y_test, logistic_predictions))
```

```
Accuracy: 0.8131868131868132
Precision: 0.8333333333333334
Recall: 0.8490566037735849
```

In [ ]:

```
# There is no overfitting,  The prediction model is good
```

Building a prediction system

In [82]:

```
input_data = (63,1,3,145,233,1,0,150,0,2.3,0,0,1)
# change the input data to a numpy array
input_data_as_numpy_array = np.asarray(input_data)
#reshape the numpy array as we are predicting for on instance
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

prediction = model.predict(input_data_reshaped)
print(prediction)
if (prediction[0]==0):
    print ('The Person does not have a Heart Disease')
else :
    print ('The Person has a Heart Disease')
```

```
[1]
The Person has a Heart Disease
```

In [83]:

```
input_data = (56,1,0,132,184,0,0,105,1,2.1,1,1,1)
# change the input data to a numpy array
input_data_as_numpy_array = np.asarray(input_data)
#reshape the numpy array as we are predicting for on instance
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

prediction = model.predict(input_data_reshaped)
print(prediction)
if (prediction[0]==0):
    print ('The Person does not have a Heart Disease')
else :
    print ('The Person has a Heart Disease')
```

```
[0]
The Person does not have a Heart Disease
```

In [ ]:

```

```