health-care-project

IMPORTING THE LIBRARIES

[1]:
```
# IMPORTING ALL THE REQUIRED LIBRARIES
import  math
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

IMPORTING THE DATA SETS

[2]:
```
# Importing the datasets using pandas module
df=pd.read_excel('1645792390_cep1_dataset.xlsx')
```

[3]:
```
# Checking the type of the dataframe i.e., df
type(df)
```

[3] :
```
pandas.core.frame.DataFrame
```

[4] :
```
# Viewing the dataframe
df
```

[4]:
```
     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  target
0        0   0     1       1
1        0   0     2       1
2        2   0     2       1
```

```
3          2  0    2     1
4          2  0    2     1
..       ... ..  ...   ...
298        1  0    3     0
299        1  0    3     0
300        1  2    3     0
301        1  1    3     0
302        1  1    2     0

[303 rows x 14 columns]
```

[5] :
```python
# Knowing the info about the dataframe i.e., no.of rows and no.of columns.
df.info
```

[5] :
```
<bound method DataFrame.info of      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  target
0        0   0     1       1
1        0   0     2       1
2        2   0     2       1
3        2   0     2       1
4        2   0     2       1
..     ...  ..   ...     ...
298      1   0     3       0
299      1   0     3       0
300      1   2     3       0
301      1   1     3       0
302      1   1     2       0

[303 rows x 14 columns]>
```

CHECKING FOR NULL VALUES

[6] :
```python
# checking for null values in the dataframe
df.isnull().sum()
```

[6]:
```
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
target      0
dtype: int64
```

There are no null values in the given data set.

[7]:
```
# To view first five rows of the dataframe
df.head()
```

[7]:

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

	ca	thal	target
0	0	1	1
1	0	2	1
2	0	2	1
3	0	2	1
4	0	2	1

[8]:
```
#To view last five rows of the dataframe
df.tail()
```

[8]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	\
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	target
298	1	0	3	0

```
299        1   0    3      0
300        1   2    3      0
301        1   1    3      0
302        1   1    2      0
```

Check for Outliers

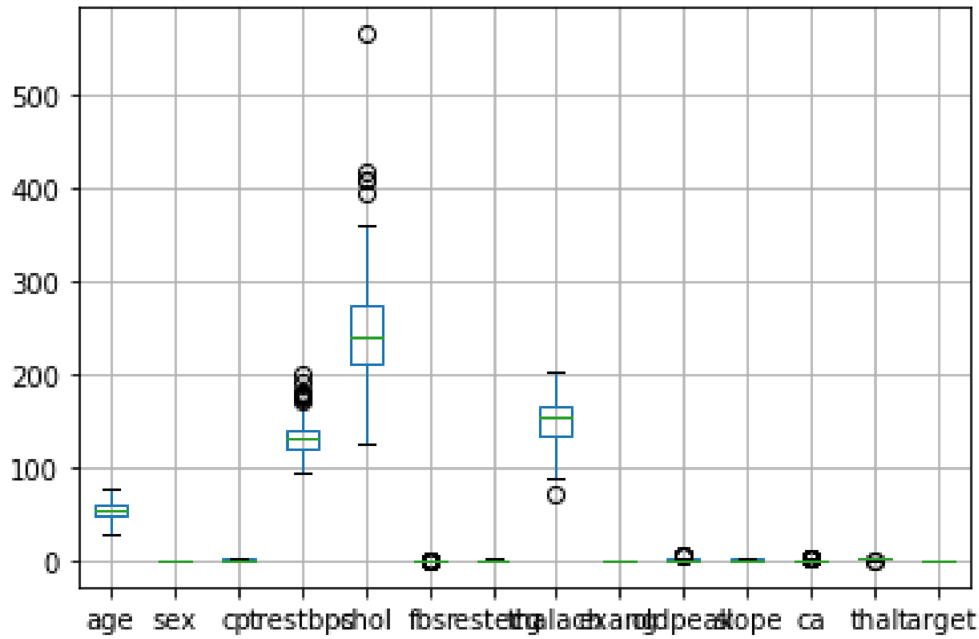
[9] :
```python
# Plotting box plot to visualize the presence of outliers in data set
Dataset=df
```

[10] :
```python
Dataset
```

[10]:
```
     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  target
0        0   0     1       1
1        0   0     2       1
2        2   0     2       1
3        2   0     2       1
4        2   0     2       1
..     ...  ..   ...     ...
298      1   0     3       0
299      1   0     3       0
300      1   2     3       0
301      1   1     3       0
302      1   1     2       0

[303 rows x 14 columns]
```

[11] :
```python
# plotting the boxplot to view the outliers present in the dataframe
Dataset.boxplot()
```

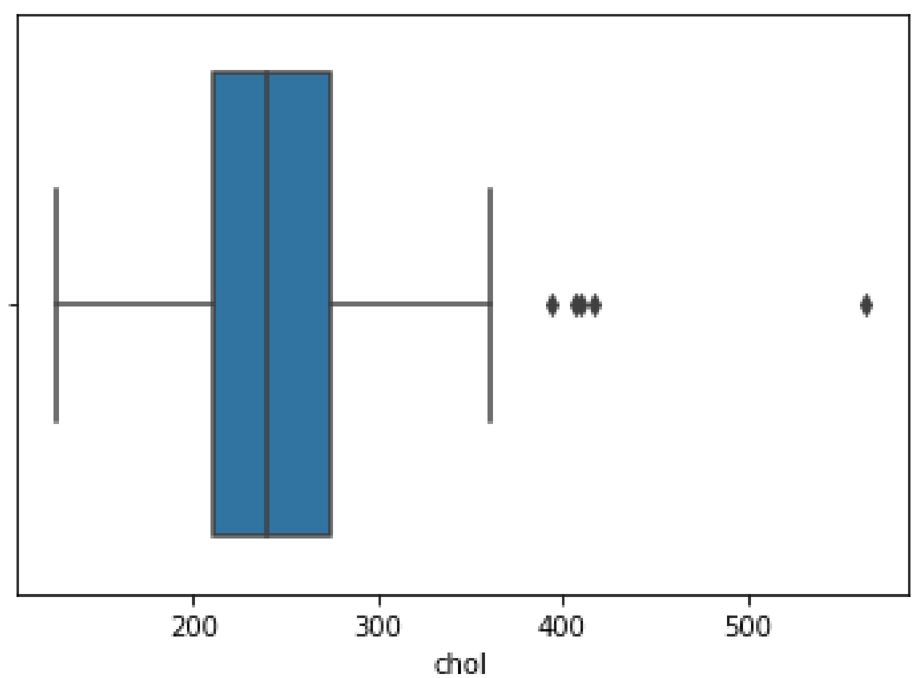
[11] :
```
<AxesSubplot:>
```



CHECKING THE OUTLIERS FOR CHOLESTRAL COLUMN OF DATAFRAME AND REMOVED OUTLIERS PRESENT IN IT.

[12] :
```
sns.boxplot(x=Dataset['chol'])
```

[12] : <AxesSubplot:xlabel='chol'>



IQR(Interquartile-range) technique for outlier treatment

[13]:
```
def outlier_treatment(col):
 sorted(col)
 Q1,Q3 = np.percentile(col , [25,75])
 IQR = Q3 - Q1
 lower_range = Q1 - (1.5 * IQR)
 upper_range = Q3 + (1.5 * IQR)
 return  lower_range,upper_range
```

[14]:
```
lower_range,upper_range = outlier_treatment(Dataset['chol'])
print("Lower Range:",lower_range)
print("Upper Range:",upper_range)
```

```
Lower Range: 115.75
Upper Range: 369.75
```

[15]:
```
lower_Dataset_chol_df = Dataset[Dataset['chol'].values < lower_range]
lower_Dataset_chol_df
```

[15]:
```
Empty DataFrame
Columns: [age, sex, cp, trestbps, chol, fbs, restecg, thalach, exang, oldpeak,
slope, ca, thal, target]
Index: []
```

[16]:
```
upper_Dataset_chol_df = Dataset[Dataset['chol'].values > upper_range]
upper_Dataset_chol_df
```

[16]:
```
     age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  \
28    65    0   2       140   417    1        0      157      0      0.8
85    67    0   2       115   564    0        0      160      0      1.6
96    62    0   0       140   394    0        0      157      0      1.2
220   63    0   0       150   407    0        0      154      0      4.0
246   56    0   0       134   409    0        0      150      1      1.9

     slope  ca  thal  target
28       2   1     2       1
85       1   0     3       1
96       1   0     2       1
220      1   3     3       0
246      1   2     3       0
```

[17]:
```
lower_outliers  =  lower_Dataset_chol_df.value_counts().sum(axis=0)
upper_outliers  =  upper_Dataset_chol_df.value_counts().sum(axis=0)
total_outliers  =  lower_outliers  +  upper_outliers
```

```
print("Total Number of Outliers:",total_outliers)
```

Total Number of Outliers: 5

[18] :
```
lower_index = list(Dataset[ Dataset['chol'] < lower_range ].index)
upper_index = list(Dataset[ Dataset['chol'] > upper_range ].index)
total_index = list(lower_index + upper_index)
print(total_index)
```

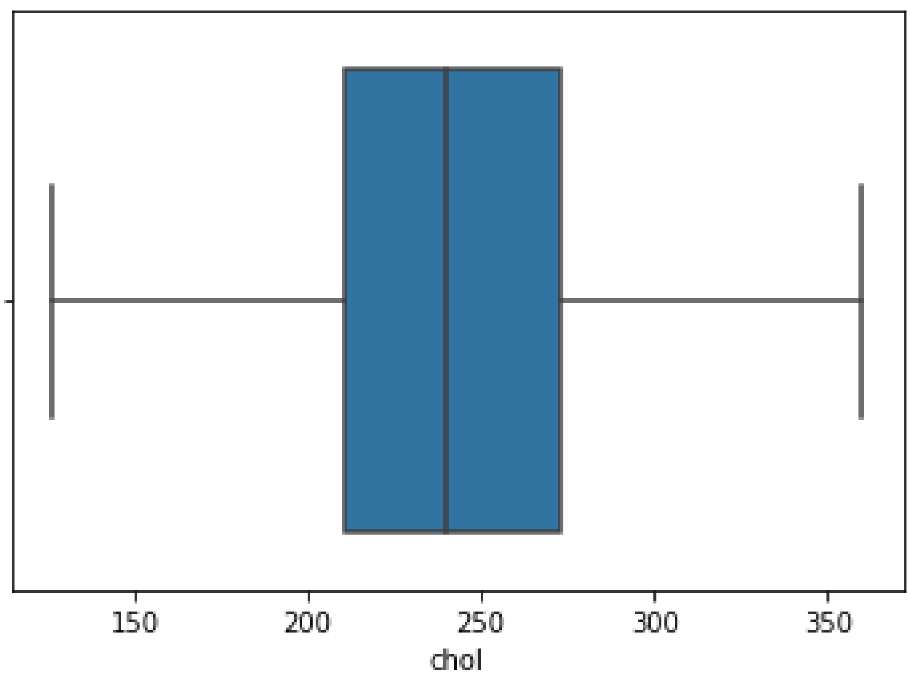
[28, 85, 96, 220, 246]

[19] :
```
print("Shape Before Dropping Outlier Rows:", Dataset.shape)
Dataset.drop(total_index, inplace = True)
print("Shape After Dropping Outlier Rows:", Dataset.shape)
```

Shape Before Dropping Outlier Rows: (303, 14)
Shape After Dropping Outlier Rows: (298, 14)

[20] :
```
sns.boxplot(x=Dataset['chol'])
```

[20] : <AxesSubplot:xlabel='chol'>



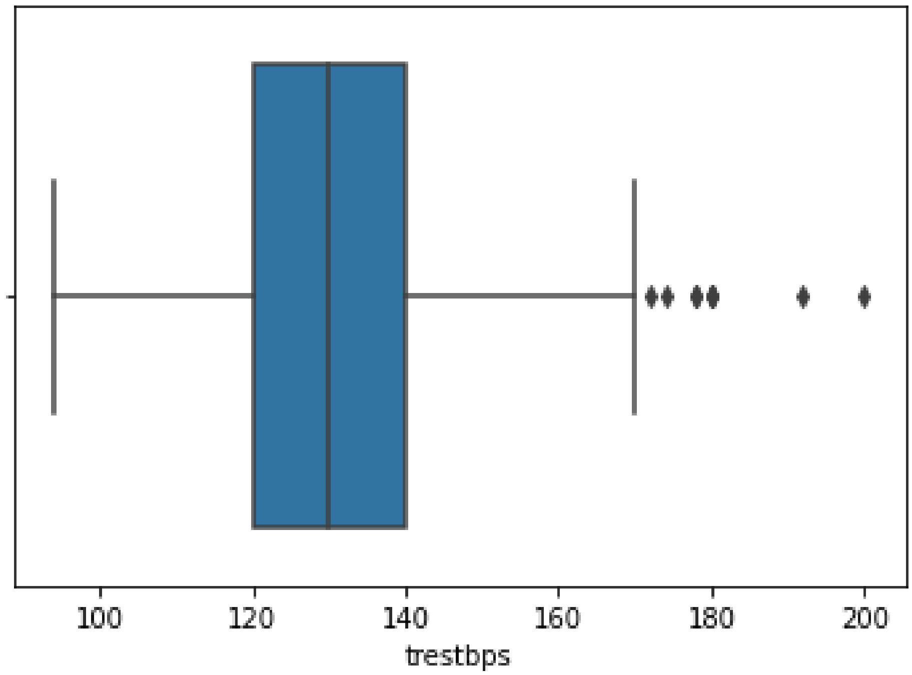
We can see that outliers of choloestral column of Dataset is dropped. It can be viewed by boxplot.

Checking outliers

CHECKING THE OUTLIERS FOR TRESTBPS COLUMN (i.e.,RESTING BP (in mm Hg on admission to the hospital)) OF DATAFRAME AND REMOVED OUTLIERS PRESENT IN IT.

[21] :
```
sns.boxplot(x=Dataset['trestbps'])
```

[21] : <AxesSubplot:xlabel='trestbps'>



IQR(Interquartile-range) technique for outlier treatment

[22] :
```
def outlier_treatment(col):
 sorted(col)
 Q1,Q3 = np.percentile(col , [25,75])
 IQR = Q3 - Q1
 lower_range = Q1 - (1.5 * IQR)
 upper_range = Q3 + (1.5 * IQR)
 return  lower_range,upper_range
```

[23] :
```
lower_range,upper_range = outlier_treatment(Dataset['trestbps'])
print("Lower Range:",lower_range)
print("Upper Range:",upper_range)
```

Lower Range: 90.0
Upper Range: 170.0

[24] :

```
lower_Dataset_trestbps_df = Dataset[Dataset['trestbps'].values < lower_range]
lower_Dataset_trestbps_df
```

[24] : Empty DataFrame

```
Columns: [age, sex, cp, trestbps, chol, fbs, restecg, thalach, exang, oldpeak,
slope, ca, thal, target]
Index: []
```

[25] :

```
upper_Dataset_trestbps_df = Dataset[Dataset['trestbps'].values > upper_range]
upper_Dataset_trestbps_df
```

[25]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	\
8	52	1	2	172	199	1	1	162	0	0.5	
101	59	1	3	178	270	0	0	145	0	4.2	
110	64	0	0	180	325	0	1	154	1	0.0	
203	68	1	2	180	274	1	0	150	1	1.6	
223	56	0	0	200	288	1	0	133	1	4.0	
241	59	0	0	174	249	0	1	143	1	0.0	
248	54	1	1	192	283	0	0	195	0	0.0	
260	66	0	0	178	228	1	1	165	1	1.0	
266	55	0	0	180	327	0	2	117	1	3.4	

	slope	ca	thal	target
8	2	0	3	1
101	0	0	3	1
110	2	0	2	1
203	1	0	3	0
223	0	2	3	0
241	1	0	2	0
248	2	1	3	0
260	1	2	3	0
266	1	0	2	0

[26] :

```
lower_outliers = lower_Dataset_trestbps_df.value_counts().sum(axis=0)
upper_outliers = upper_Dataset_trestbps_df.value_counts().sum(axis=0)
total_outliers = lower_outliers + upper_outliers
print("Total Number of Outliers:",total_outliers)
```

```
Total Number of Outliers: 9
```

[27] :

```
lower_index = list(Dataset[ Dataset['trestbps'] < lower_range ].index)
upper_index = list(Dataset[ Dataset['trestbps'] > upper_range ].index)
total_index = list(lower_index + upper_index)
print(total_index)
```

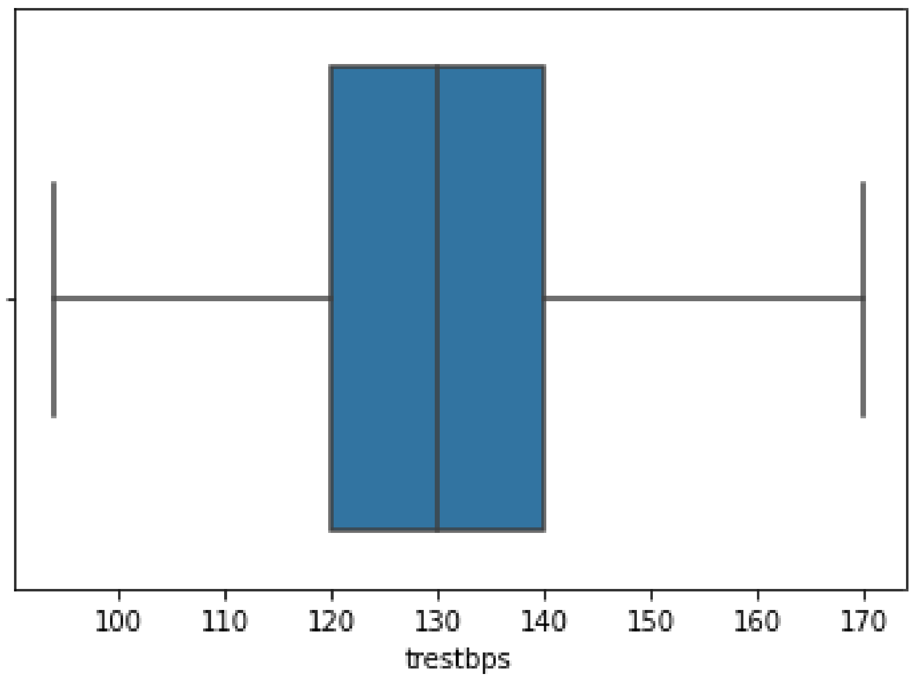
```
[8, 101, 110, 203, 223, 241, 248, 260, 266]
```

[28] :
```
print("Shape Before Dropping Outlier Rows:", Dataset.shape)
Dataset.drop(total_index, inplace = True)
print("Shape After Dropping Outlier Rows:", Dataset.shape)
```

```
Shape Before Dropping Outlier Rows: (298, 14)
Shape After Dropping Outlier Rows: (289, 14)
```

[29] :
```
sns.boxplot(x=Dataset['trestbps'])
```

[29] : <AxesSubplot:xlabel='trestbps'>



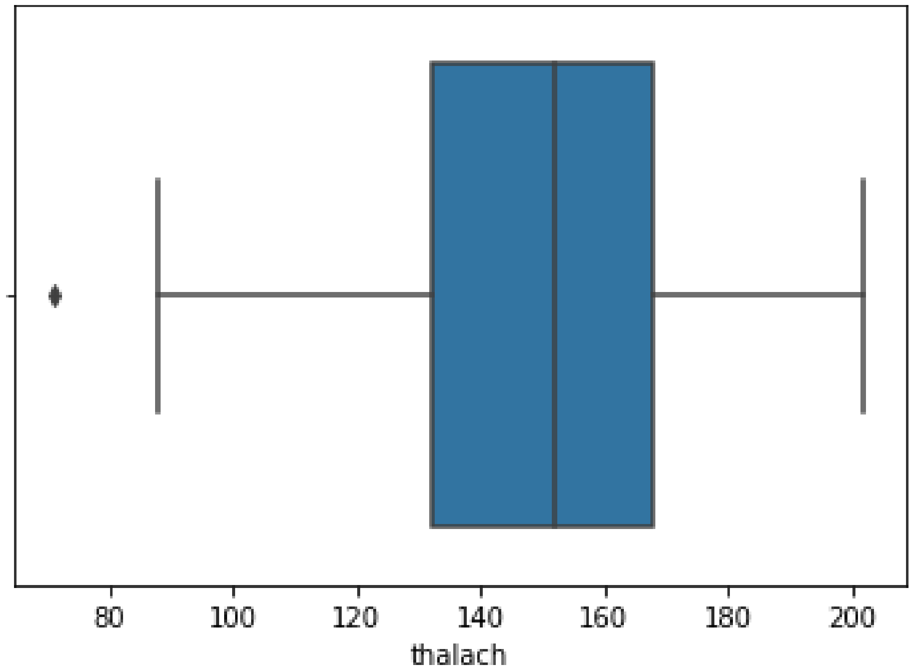
We can see that outliers of Resting blood pressure(Hg) column of Dataset is dropped. It can be viewed by boxplot.

Check for Outliers

CHECKING THE OUTLIERS FOR THALACH COLUMN (i.e.,MAXIMUM HEART RATE ACHIEVED) OF DATAFRAME AND REMOVED OUTLIERS PRESENT IN IT.

[30] :
```
sns.boxplot(x=Dataset['thalach'])
```

[30] : <AxesSubplot:xlabel='thalach'>



IQR(Interquartile-range) technique for outlier treatment

[31] :
```
def outlier_treatment(col):
 sorted(col)
 Q1,Q3 = np.percentile(col , [25,75])
 IQR = Q3 - Q1
 lower_range = Q1 - (1.5 * IQR)
 upper_range = Q3 + (1.5 * IQR)
 return  lower_range,upper_range
```

[32] :
```
lower_range,upper_range = outlier_treatment(Dataset['thalach'])
print("Lower  Range:",lower_range)
print("Upper  Range:",upper_range)
```

```
Lower Range: 78.0
Upper Range: 222.0
```

[33] :
```
lower_Dataset_thalach_df  = Dataset[Dataset['thalach'].values  <  lower_range]
lower_Dataset_thalach_df
```

[33] :

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	\
272	67	1	0	120	237	0	1	71	0	1.0	

	slope	ca	thal	target
272	1	0	2	0

[34] :
```
upper_Dataset_thalach_df = Dataset[Dataset['trestbps'].values > upper_range]
upper_Dataset_thalach_df
```

[34] : Empty DataFrame

```
Columns: [age, sex, cp, trestbps, chol, fbs, restecg, thalach, exang, oldpeak,
slope, ca, thal, target]
Index: []
```

[35] :
```
lower_outliers = lower_Dataset_thalach_df.value_counts().sum(axis=0)
upper_outliers = upper_Dataset_thalach_df.value_counts().sum(axis=0)
total_outliers = lower_outliers + upper_outliers
print("Total Number of Outliers:",total_outliers)
```

```
Total Number of Outliers: 1
```

[36] :
```
lower_index = list(Dataset[ Dataset['thalach'] < lower_range ].index)
upper_index = list(Dataset[ Dataset['thalach'] > upper_range ].index)
total_index = list(lower_index + upper_index)
print(total_index)
```

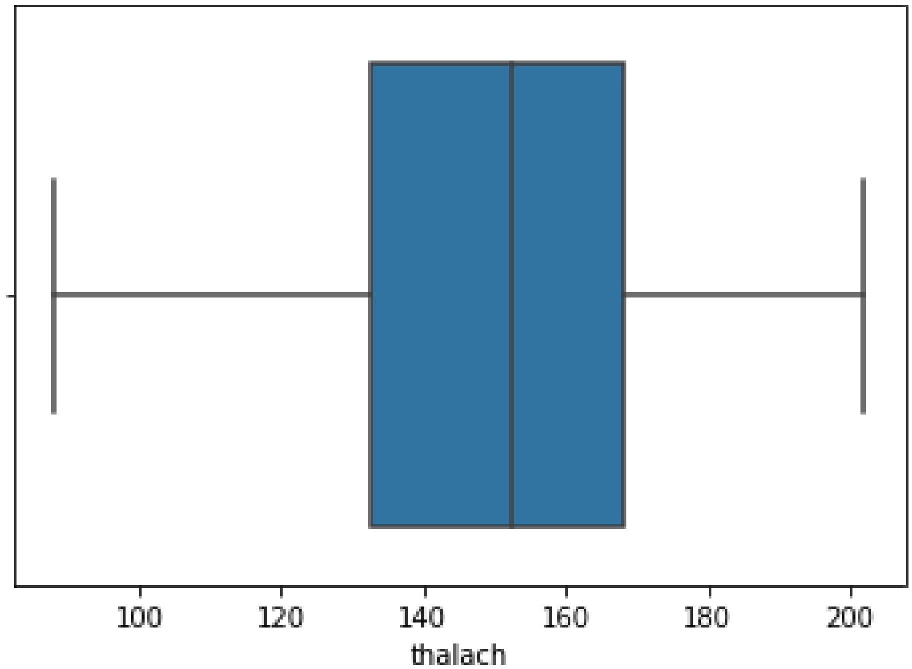
```
[272]
```

[37] :
```
print("Shape Before Dropping Outlier Rows:", Dataset.shape)
Dataset.drop(total_index, inplace = True)
print("Shape After Dropping Outlier Rows:", Dataset.shape)
```

```
Shape Before Dropping Outlier Rows: (289, 14)
Shape After Dropping Outlier Rows: (288, 14)
```

[38] :
```
sns.boxplot(x=Dataset['thalach'])
```

[38] :
```
<AxesSubplot:xlabel='thalach'>
```



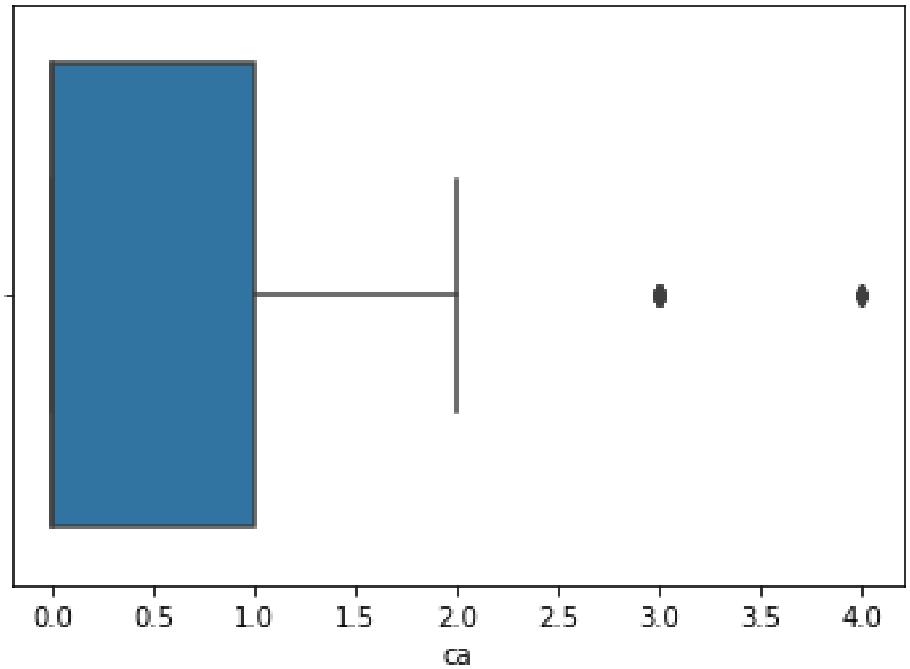
We can see that outliers of Maximum Heart Rate achieved (i.e., thalach) of Dataset is dropped. It can be viewed by boxplot.

Check for Outliers

CHECKING THE OUTLIERS FOR CA COLUMN (i.e.,NUMBER OF MAJOR VESSELS (0-3) colored by fluoroscopy) OF DATAFRAME AND REMOVED OUTLIERS PRESENT IN IT.

[39] :
```
sns.boxplot(x=Dataset['ca'])
```

[39] : <AxesSubplot:xlabel='ca'>



[40] :
```
def outlier_treatment(col):
 sorted(col)
 Q1,Q3 = np.percentile(col , [25,75])
 IQR = Q3 - Q1
 lower_range = Q1 - (1.5 * IQR)
 upper_range = Q3 + (1.5 * IQR)
 return  lower_range,upper_range
```

[41] :
```
lower_range,upper_range = outlier_treatment(Dataset['ca'])
print("Lower Range:",lower_range)
print("Upper Range:",upper_range)
```

Lower Range: -1.5
Upper Range: 2.5

[42] :
```
lower_Dataset_ca_df = Dataset[Dataset['ca'].values < lower_range]
lower_Dataset_ca_df
```

[42] : Empty DataFrame
Columns: [age, sex, cp, trestbps, chol, fbs, restecg, thalach, exang, oldpeak, slope, ca, thal, target]
Index: []

[43] :
```
upper_Dataset_ca_df = Dataset[Dataset['ca'].values > upper_range]
upper_Dataset_ca_df
```

[43]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	\
52	62	1	2	130	231	0	1	146	0	1.8	
92	52	1	2	138	223	0	1	169	0	0.0	
97	52	1	0	108	233	1	1	147	0	0.1	
99	53	1	2	130	246	1	0	173	0	0.0	
158	58	1	1	125	220	0	1	144	0	0.4	
163	38	1	2	138	175	0	1	173	0	0.0	
164	38	1	2	138	175	0	1	173	0	0.0	
165	67	1	0	160	286	0	0	108	1	1.5	
181	65	0	0	150	225	0	0	114	0	1.0	
191	58	1	0	128	216	0	0	131	1	2.2	
204	62	0	0	160	164	0	0	145	0	6.2	
208	49	1	2	120	188	0	1	139	0	2.0	
217	63	1	0	130	330	1	0	132	1	1.8	
231	57	1	0	165	289	1	0	124	0	1.0	
234	70	1	0	130	322	0	0	109	0	2.4	
238	77	1	0	125	304	0	0	162	1	0.0	
247	66	1	1	160	246	0	1	120	1	0.0	
249	69	1	2	140	254	0	0	146	0	2.0	
250	51	1	0	140	298	0	1	122	1	4.2	
251	43	1	0	132	247	1	0	143	1	0.1	
252	62	0	0	138	294	1	1	106	0	1.9	
255	45	1	0	142	309	0	0	147	1	0.0	
267	49	1	2	118	149	0	0	126	0	0.8	
291	58	1	0	114	318	0	2	140	0	4.4	

	slope	ca	thal	target
52	1	3	3	1
92	2	4	2	1
97	2	3	3	1
99	2	3	2	1
158	1	4	3	1
163	2	4	2	1
164	2	4	2	1
165	1	3	2	0
181	1	3	3	0
191	1	3	3	0
204	0	3	3	0
208	1	3	3	0
217	2	3	3	0
231	1	3	3	0
234	1	3	2	0
238	2	3	2	0
247	1	3	1	0
249	1	3	3	0
250	1	3	3	0
251	1	4	3	0

```
252        1    3      2         0
255        1    3      3         0
267        2    3      2         0
291        0    3      1         0
```

[44] :
```
lower_outliers  =  lower_Dataset_ca_df.value_counts().sum(axis=0)
upper_outliers  =  upper_Dataset_ca_df.value_counts().sum(axis=0)
total_outliers  =  lower_outliers  +  upper_outliers
print("Total Number of Outliers:",total_outliers)
```

```
Total Number of Outliers: 24
```

[45] :
```
lower_index = list(Dataset[ Dataset['ca'] < lower_range ].index)
upper_index = list(Dataset[ Dataset['ca'] > upper_range ].index)
total_index = list(lower_index + upper_index)
print(total_index)
```

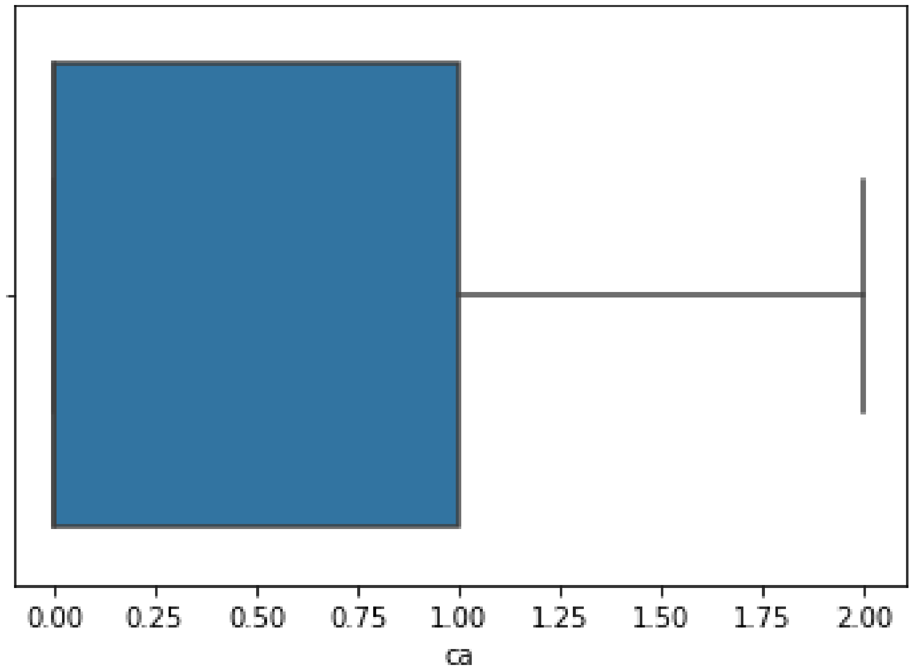
```
[52, 92, 97, 99, 158, 163, 164, 165, 181, 191, 204, 208, 217, 231, 234, 238,
247, 249, 250, 251, 252, 255, 267, 291]
```

[46] :
```
print("Shape Before Dropping Outlier Rows:", Dataset.shape)
Dataset.drop(total_index, inplace = True)
print("Shape After Dropping Outlier Rows:", Dataset.shape)
```

```
Shape Before Dropping Outlier Rows: (288, 14)
Shape After Dropping Outlier Rows: (264, 14)
```

[47] :
```
sns.boxplot(x=Dataset['ca'])
```

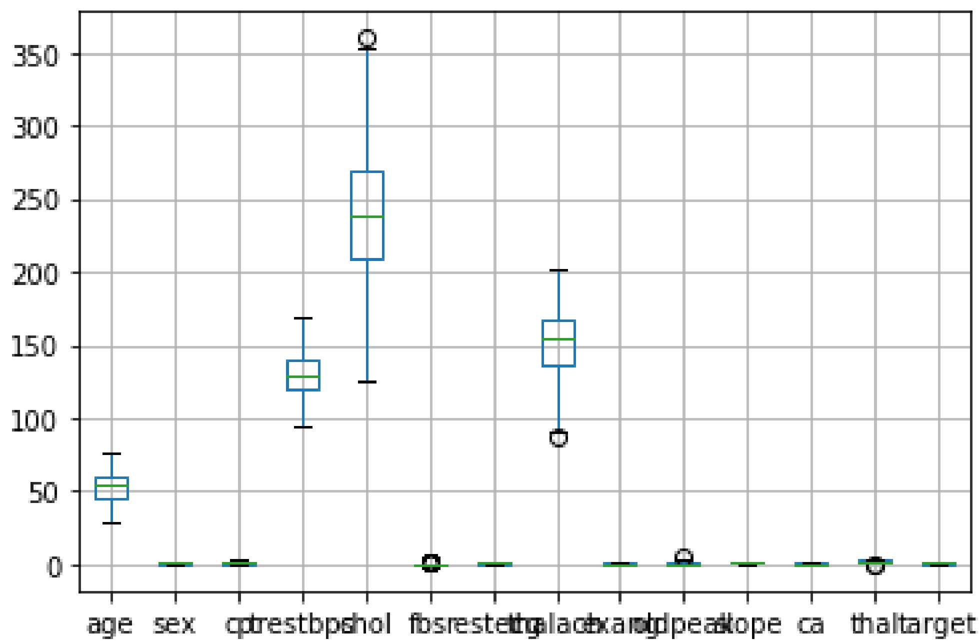
[47] : <AxesSubplot:xlabel='ca'>



We can see that outliers of Number of major vessels (0-3) colored by fluoroscopy (i.e., ‘ca’) of Dataset is dropped. It can be viewed by boxplot.

[48] :
```
Dataset.boxplot()
```

[48] : <AxesSubplot:>



Now Outiler treatment is done (removed all the outliers).

Checking the shape of the Dataset:

[49] :
```
Dataset.shape
```

[49]:
```
(264, 14)
```

Checking for na values:

[50] :
```
Dataset.isna().sum()
```

[50] :
```
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
target      0
dtype: int64
```

There are no na values in the Dataset

[51] :
```
Dataset.describe()
```

[51]:

	age	sex	cp	trestbps	chol	fbs
count	264.000000	264.000000	264.000000	264.000000	264.000000	264.000000
mean	53.772727	0.685606	0.996212	129.507576	241.685606	0.128788
std	8.993949	0.465156	1.037319	15.374413	44.265914	0.335601
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000
25%	46.000000	0.000000	0.000000	120.000000	209.000000	0.000000
50%	54.500000	1.000000	1.000000	130.000000	239.000000	0.000000
75%	60.000000	1.000000	2.000000	140.000000	269.000000	0.000000
max	76.000000	1.000000	3.000000	170.000000	360.000000	1.000000

	restecg	thalach	exang	oldpeak	slope	ca
count	264.000000	264.000000	264.000000	264.000000	264.000000	264.000000
mean	0.534091	150.723485	0.318182	0.969697	1.428030	0.503788
std	0.514775	22.677673	0.466655	1.073174	0.612396	0.719094

```
min    0.000000   88.000000  0.000000  0.000000  0.000000  0.000000
25%    0.000000  136.750000  0.000000  0.000000  1.000000  0.000000
50%    1.000000  155.000000  0.000000  0.600000  1.000000  0.000000
75%    1.000000  168.000000  1.000000  1.600000  2.000000  1.000000
max    2.000000  202.000000  1.000000  5.600000  2.000000  2.000000

             thal      target
count  264.000000  264.000000
mean     2.284091    0.575758
std      0.609473    0.495166
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
```

Exploratory Data Analysis:

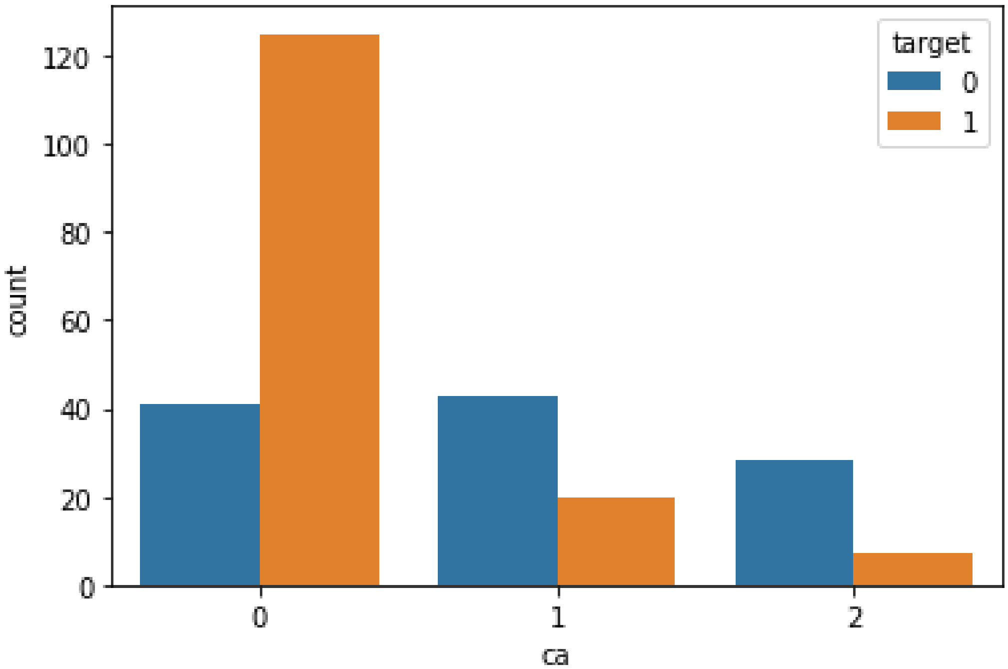
[52] :
```
# Viewing the dataframe
Dataset
```

[52]:
```
     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  target
0        0   0     1       1
1        0   0     2       1
2        2   0     2       1
3        2   0     2       1
4        2   0     2       1
..     ...  ..   ...     ...
298      1   0     3       0
299      1   0     3       0
300      1   2     3       0
301      1   1     3       0
302      1   1     2       0
```

[264 rows x 14 columns]

[53] :
```
# plotting the countplot to see the count of observations
sns.countplot(x=Dataset['ca'],data=Dataset,hue='target')
```

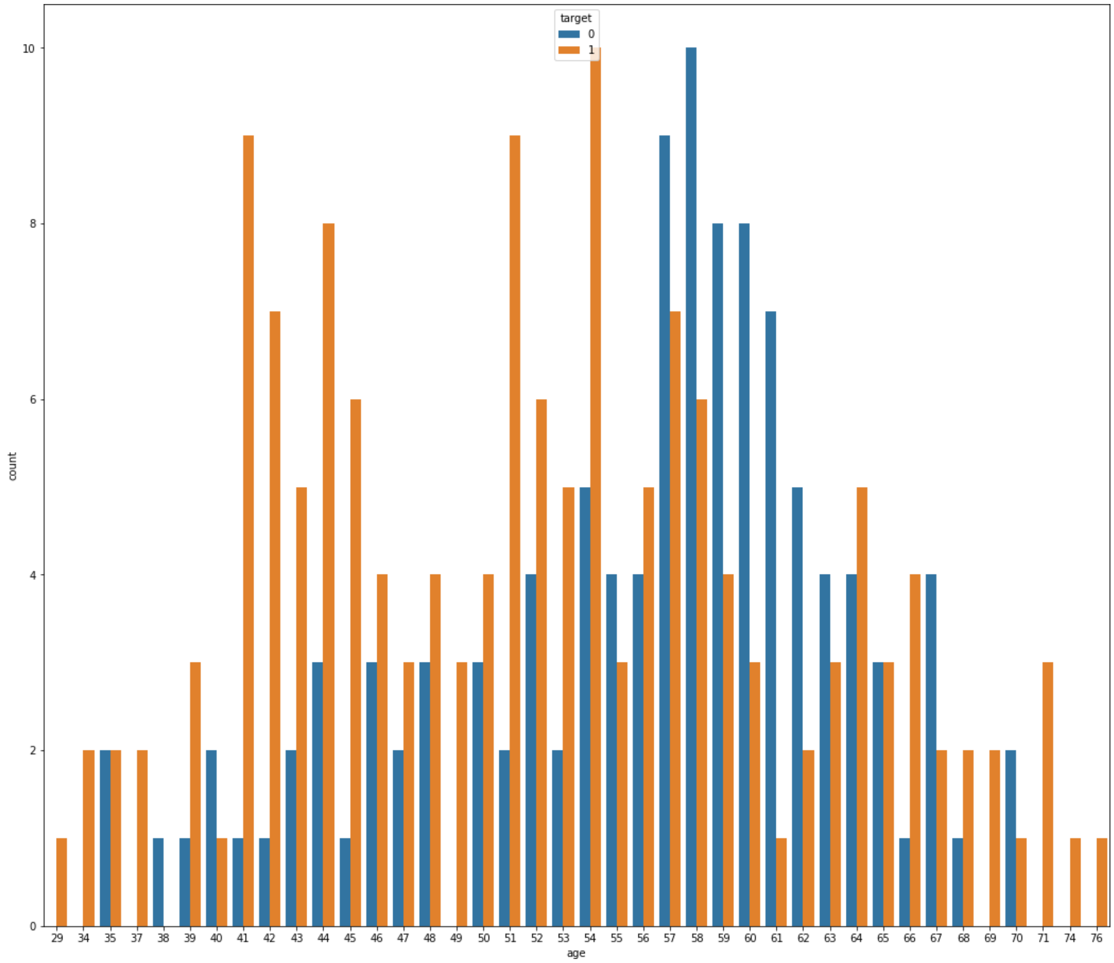
[53] : <AxesSubplot:xlabel='ca', ylabel='count'>



In the above plot we can infer: (i)0- represents no risk of heart-attack(CVD i.e., Cardiovascular Diseases) (ii)1- represents risk of heart-attack(CVD i.e., Cardiovascular Diseases) Blue color-0;Yellow color-1 Here ca means -number of major vessels (0-3) colored by fluoroscopy. We can see the risk of heart-attack is decreased if the number of vessels were colored by fluroscopy increases.

[54] :
```
# plotting the countplot to see the count of observations
plt.figure(figsize=(18,16))
sns.countplot(x=Dataset['age'],data=Dataset,hue='target',orient='v')
```

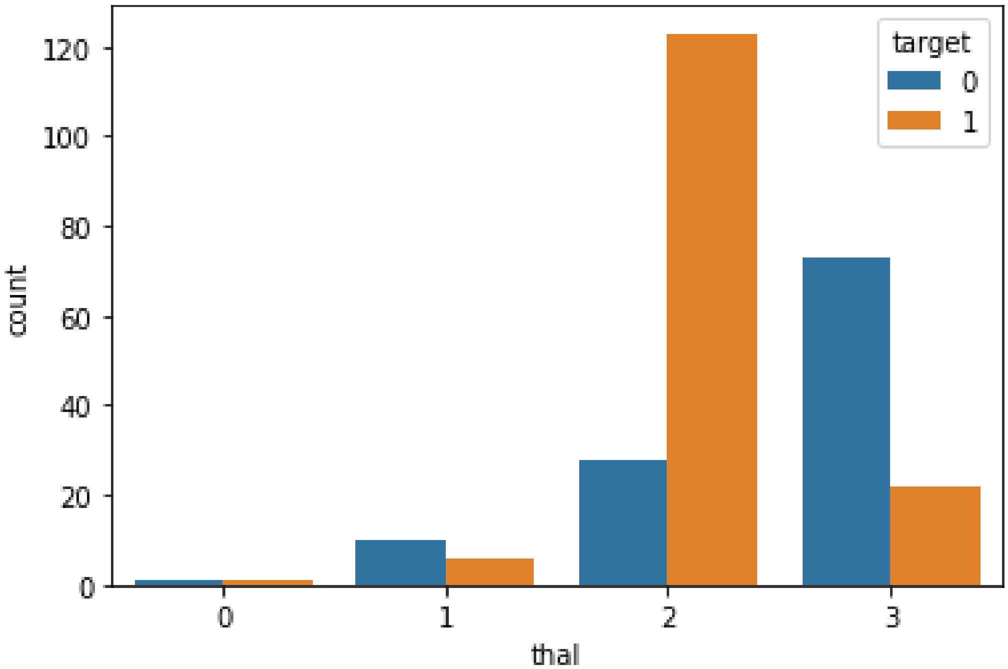
[54] : <AxesSubplot:xlabel='age', ylabel='count'>



In the above plot we can infer: (i)0- represents no risk of heart-attack(CVD i.e., Cardiovascular Diseases) (ii)1- represents risk of heart-attack(CVD i.e., Cardiovascular Diseases) Blue color-0;Yellow color-1. We can see the risk of heart-attack is more in the age group between 41 to 45 and also prominent in between 51 to 58.

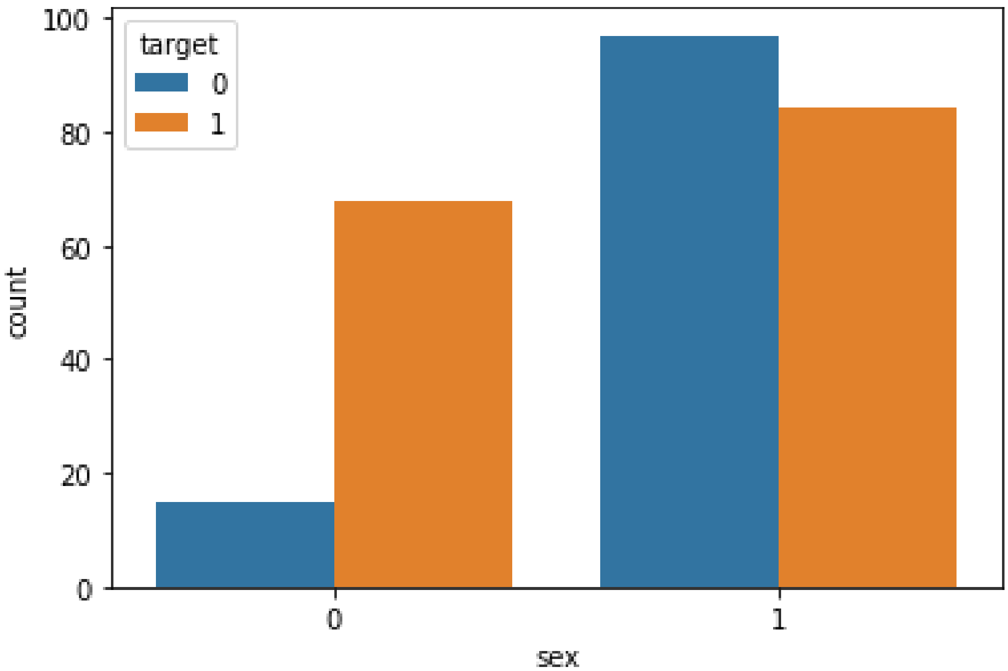
[55] :
```
sns.countplot(x=Dataset['thal'],data=Dataset,hue='target')
```

[55] : <AxesSubplot:xlabel='thal', ylabel='count'>



In the above plot we can infer: (i)0- represents no risk of heart-attack(CVD i.e., Cardiovascular Diseases) (ii)1- represents risk of heart-attack(CVD i.e., Cardiovascular Diseases) Blue color-0;Yellow color-1. (iii) As we know that Thalassemia is an inherited blood disorder in which the body makes an abnormal form of hemoglobin. (iv) From the plot we can see that the risk of heart-attack is more in 2 i.e,(more in fixed defect type) and less in normal defect type.

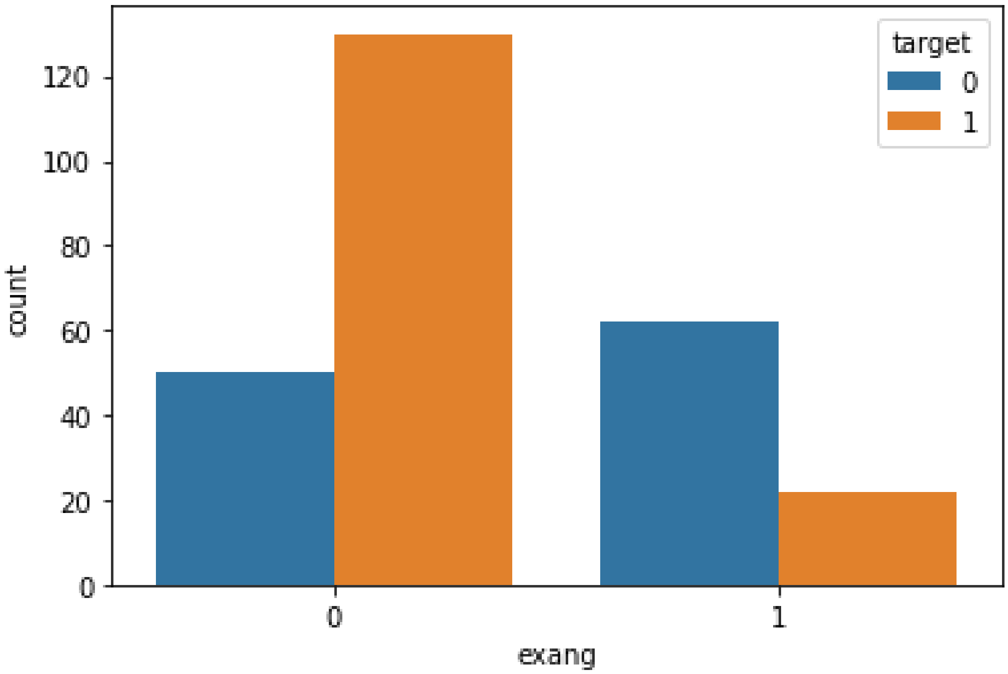
[56] :
```
ax=sns.countplot(x='sex',data=Dataset,hue='target')
```



In the above plot we can infer: (i)0- represents no risk of heart-attack(CVD i.e., Cardiovascular Diseases) (ii)1- represents risk of heart-attack(CVD i.e., Cardiovascular Diseases) Blue color-0;Yellow color-1. (iii) Here first set of count on x-axis belongs to Female and second set of count on x-axis belongs to Male. (iv) The risk of heart attack is more in Male comparitively to Female.

[66]:
```
sns.countplot(x='exang',data=Dataset,hue='target')
```

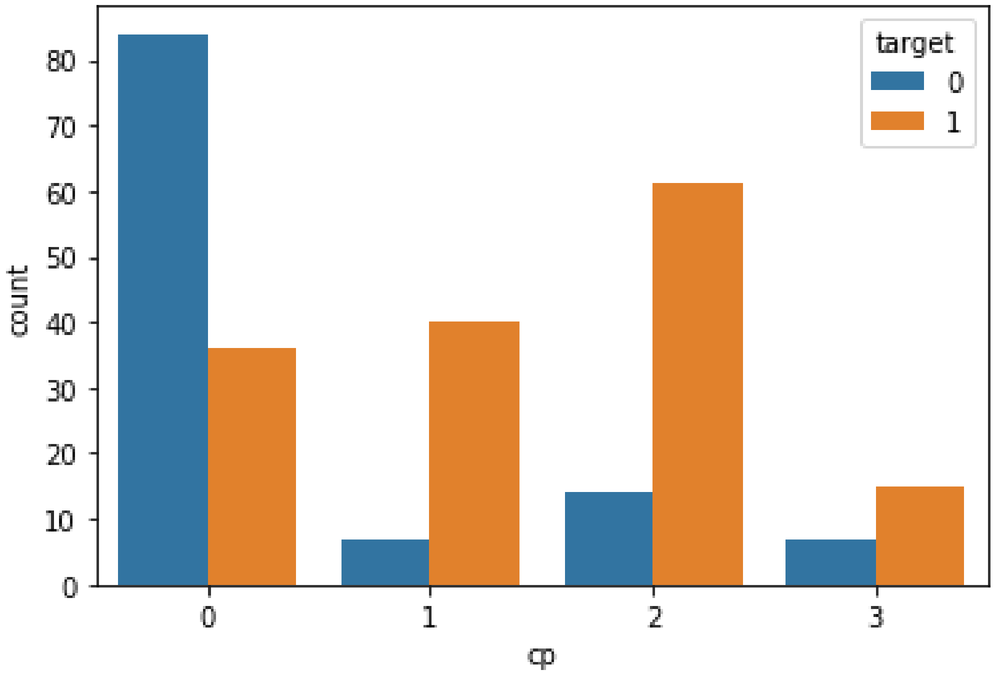
[66] : <AxesSubplot:xlabel='exang', ylabel='count'>



In the above plot we can infer: (i)0- represents no risk of heart-attack(CVD i.e., Cardiovascular Diseases) (ii)1- represents risk of heart-attack(CVD i.e., Cardiovascular Diseases) Blue color-0;Yellow color-1. (iii) Here exang means exercise induced angina(i.e, pressure on chest) & risk of heart attack increases when pressure on chest increases.

[67] :
```
sns.countplot(x='cp',data=Dataset,hue='target')
```

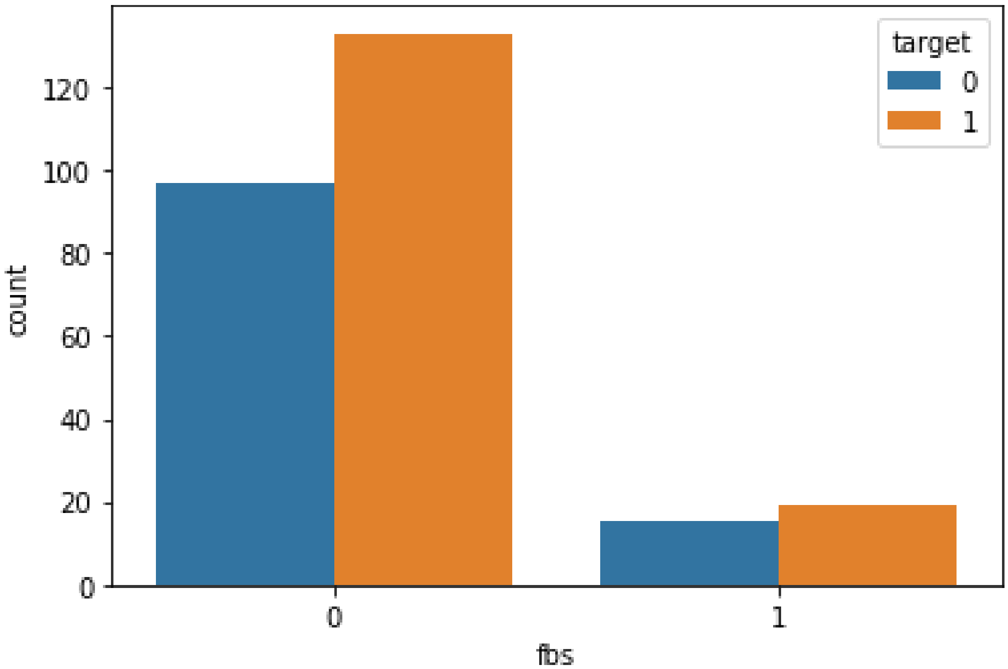
[67] : <AxesSubplot:xlabel='cp', ylabel='count'>



In the above plot we can infer: (i)0- represents no risk of heart-attack(CVD i.e., Cardiovascular Diseases) (ii)1- represents risk of heart-attack(CVD i.e., Cardiovascular Diseases) Blue color-0;Yellow color-1. (iii) Here we can see the risk of heart attack is quite related to cp(i.e.,chest pain)

[68] :
```
sns.countplot(x='fbs',data=Dataset,hue='target')
```

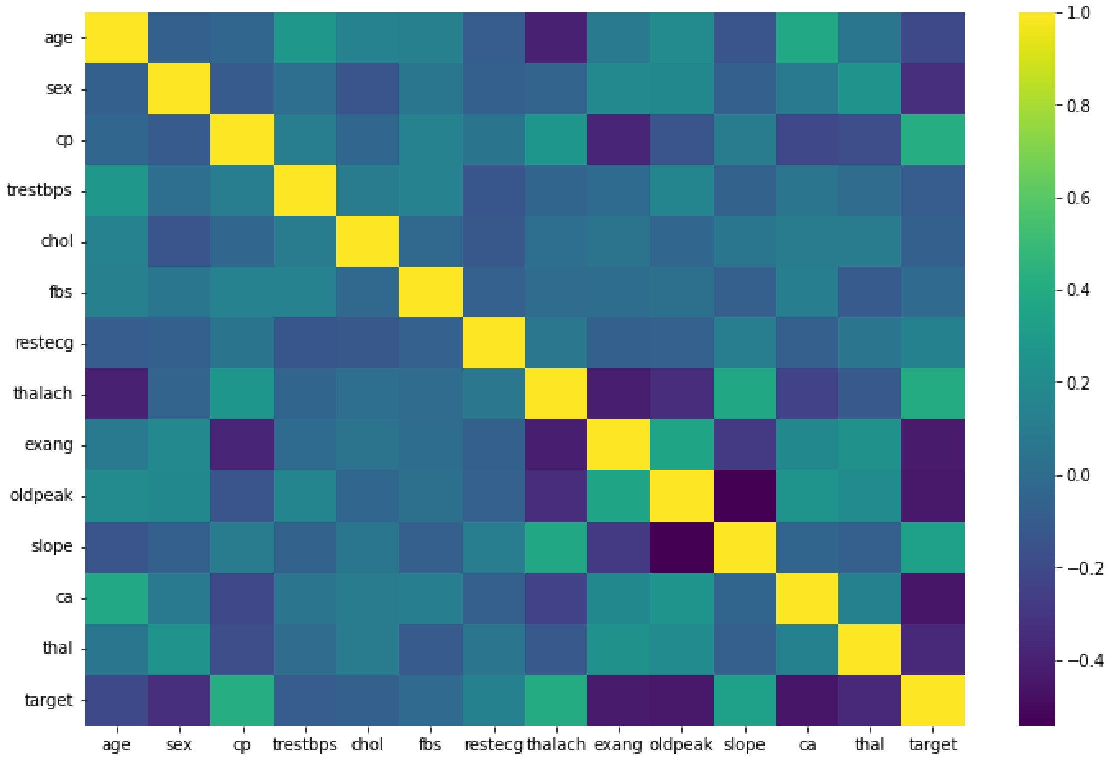
[68] : <AxesSubplot:xlabel='fbs', ylabel='count'>



In the above plot we can infer: (i)0- represents no risk of heart-attack(CVD i.e., Cardiovascular Diseases) (ii)1- represents risk of heart-attack(CVD i.e., Cardiovascular Diseases) Blue color-0;Yellow color-1. (iii) Here fbs means fasting blood sugar is quite related to risk of heart-attack.

[69] :
```
# Plotting the heat-map to see hoe features and target variables are correlated
↪to each-other.
plt.figure(figsize=(12,8))
sns.heatmap(Dataset.corr(), cmap='viridis')
```

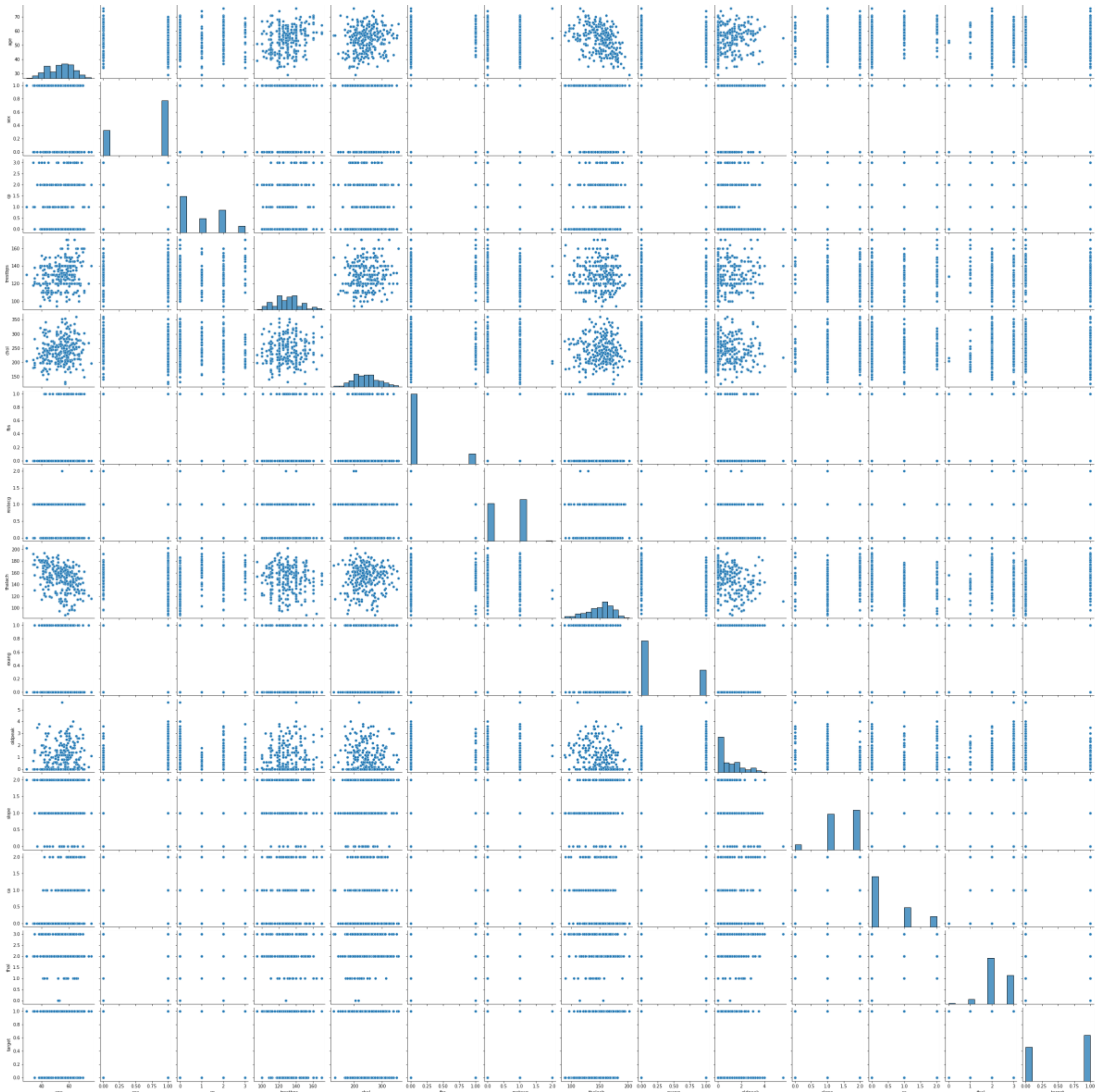
[69] : <AxesSubplot:>



[70] :
```
# PAIR PLOT
plt.figure(figsize=(12,10))
sns.pairplot(Dataset)
```

[70] : <seaborn.axisgrid.PairGrid at 0x7f14e949c410>

<Figure size 864x720 with 0 Axes>



[71] :
```python
# Correlation between the features in dataset
Dataset.corr()
```

[71]:

	age	sex	cp	trestbps	chol	fbs	\
age	1.000000	-0.072585	-0.035142	0.279086	0.133985	0.125627	
sex	-0.072585	1.000000	-0.104919	0.006449	-0.135559	0.065505	
cp	-0.035142	-0.104919	1.000000	0.104070	-0.033066	0.132472	
trestbps	0.279086	0.006449	0.104070	1.000000	0.094398	0.133930	
chol	0.133985	-0.135559	-0.033066	0.094398	1.000000	-0.027210	
fbs	0.125627	0.065505	0.132472	0.133930	-0.027210	1.000000	
restecg	-0.092764	-0.074163	0.053647	-0.127586	-0.121754	-0.069529	
thalach	-0.398934	-0.051527	0.270854	-0.042760	0.010746	-0.001798	
exang	0.087052	0.182332	-0.382386	-0.011997	0.043331	0.004414	

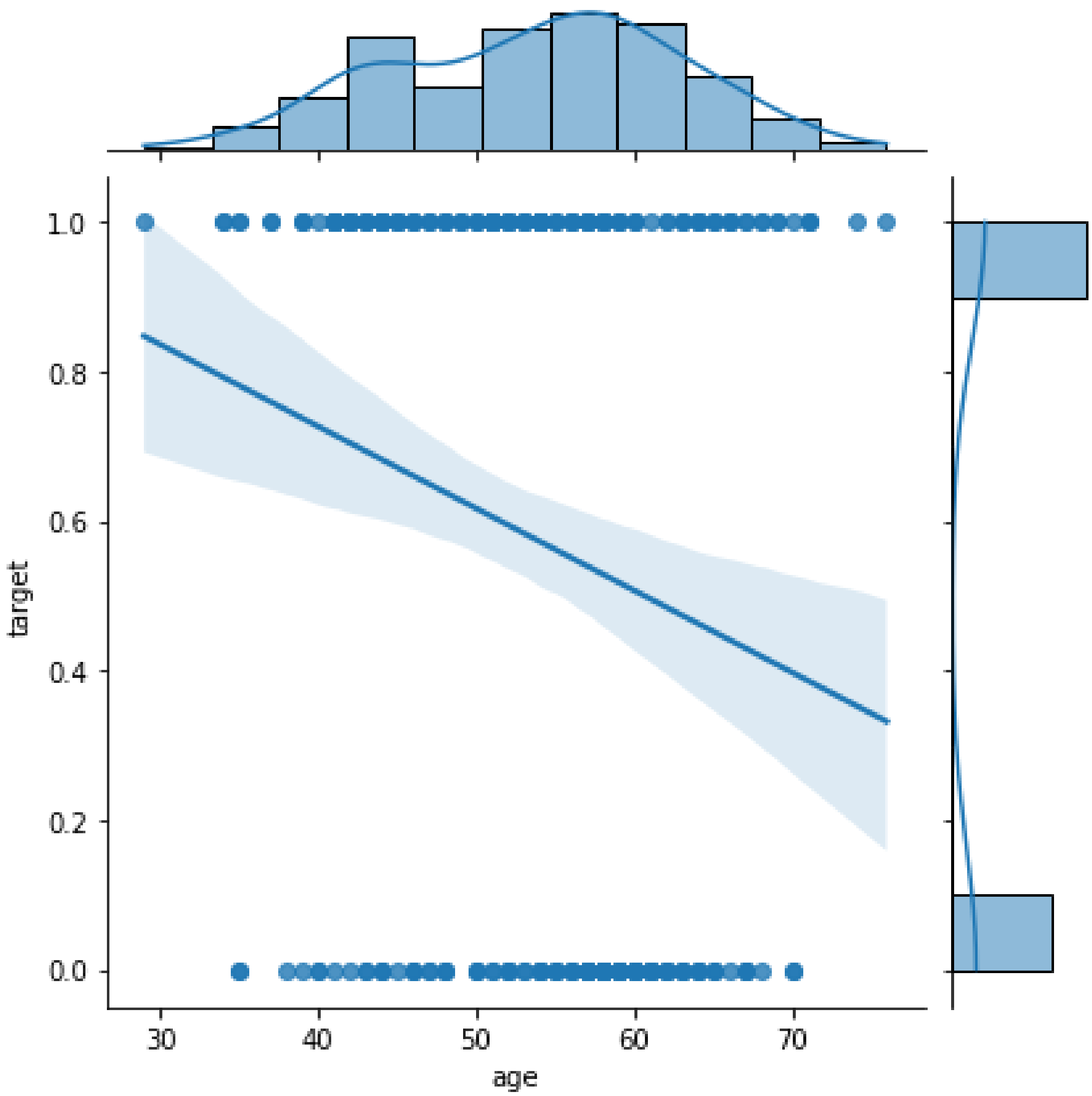
```
oldpeak    0.191918  0.178119 -0.135360  0.160568 -0.031905  0.018267
slope     -0.141739 -0.073060  0.092344 -0.065163  0.060247 -0.084234
ca         0.384037  0.088829 -0.206424  0.054853  0.092194  0.108260
thal       0.065928  0.249188 -0.178717 -0.000839  0.092113 -0.105199
target    -0.199317 -0.333662  0.411402 -0.090976 -0.076364 -0.013174

           restecg   thalach     exang   oldpeak     slope        ca  \
age      -0.092764 -0.398934  0.087052  0.191918 -0.141739  0.384037
sex      -0.074163 -0.051527  0.182332  0.178119 -0.073060  0.088829
cp        0.053647  0.270854 -0.382386 -0.135360  0.092344 -0.206424
trestbps -0.127586 -0.042760 -0.011997  0.160568 -0.065163  0.054853
chol     -0.121754  0.010746  0.043331 -0.031905  0.060247  0.092194
fbs      -0.069529 -0.001798  0.004414  0.018267 -0.084234  0.108260
restecg   1.000000  0.066441 -0.076983 -0.066950  0.104303 -0.082524
thalach   0.066441  1.000000 -0.414543 -0.345904  0.373513 -0.235780
exang    -0.076983 -0.414543  1.000000  0.358706 -0.278801  0.177688
oldpeak  -0.066950 -0.345904  0.358706  1.000000 -0.542541  0.256850
slope     0.104303  0.373513 -0.278801 -0.542541  1.000000 -0.042550
ca       -0.082524 -0.235780  0.177688  0.256850 -0.042550  1.000000
thal      0.059907 -0.118365  0.229093  0.197492 -0.082540  0.132009
target    0.131539  0.398550 -0.433813 -0.442150  0.325253 -0.454643

              thal    target
age       0.065928 -0.199317
sex       0.249188 -0.333662
cp       -0.178717  0.411402
trestbps -0.000839 -0.090976
chol      0.092113 -0.076364
fbs      -0.105199 -0.013174
restecg   0.059907  0.131539
thalach  -0.118365  0.398550
exang     0.229093 -0.433813
oldpeak   0.197492 -0.442150
slope    -0.082540  0.325253
ca        0.132009 -0.454643
thal      1.000000 -0.367664
target   -0.367664  1.000000
```

[72] :

```python
# Joint plot between age and target( Multivariate Analysis)
sns.jointplot('age','target',data=Dataset,kind='reg')
```

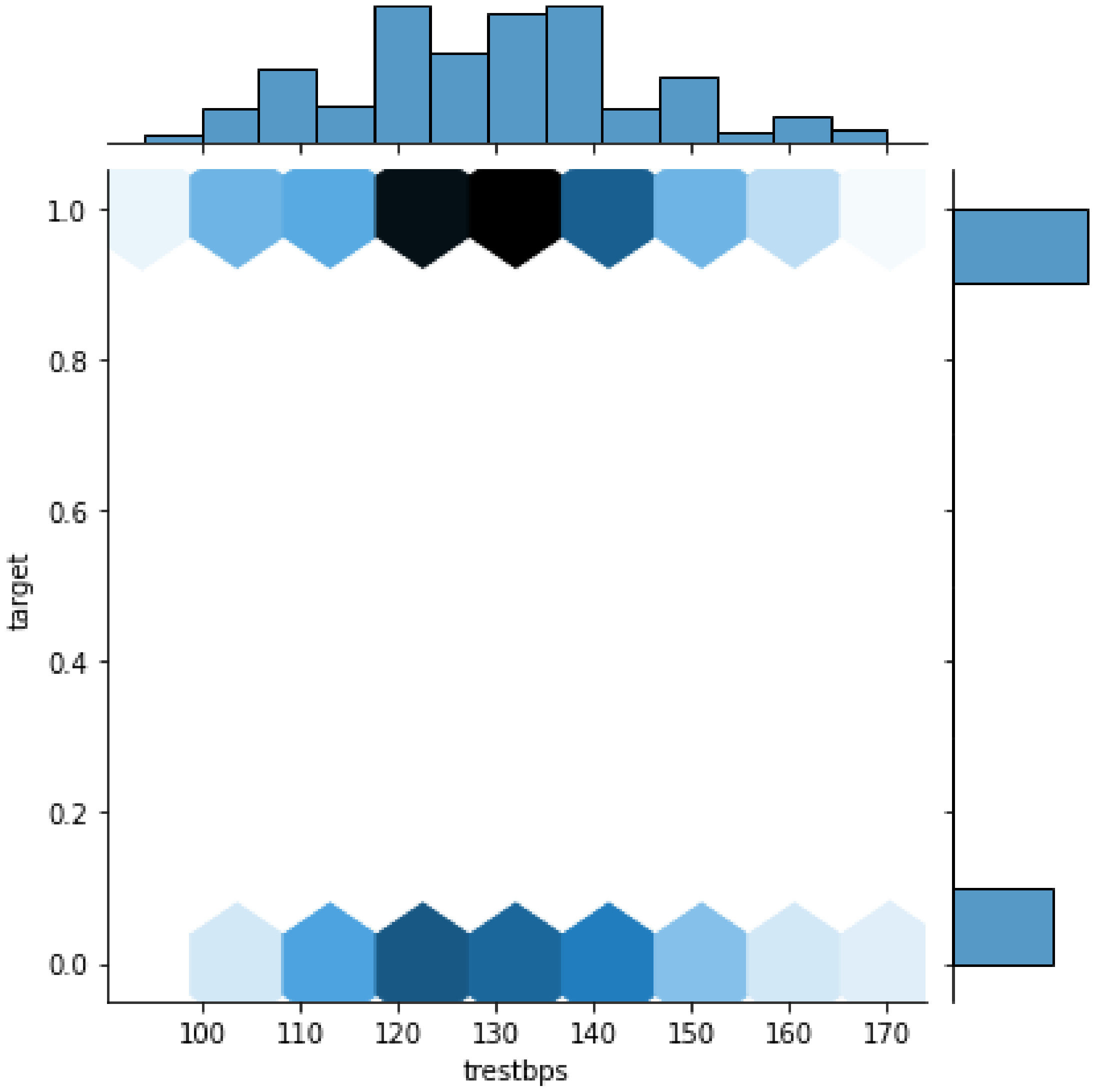
```
/usr/local/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning:
Pass the following variables as keyword args: x, y. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.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

[72]: <seaborn.axisgrid.JointGrid at 0x7f14e20aa9d0>



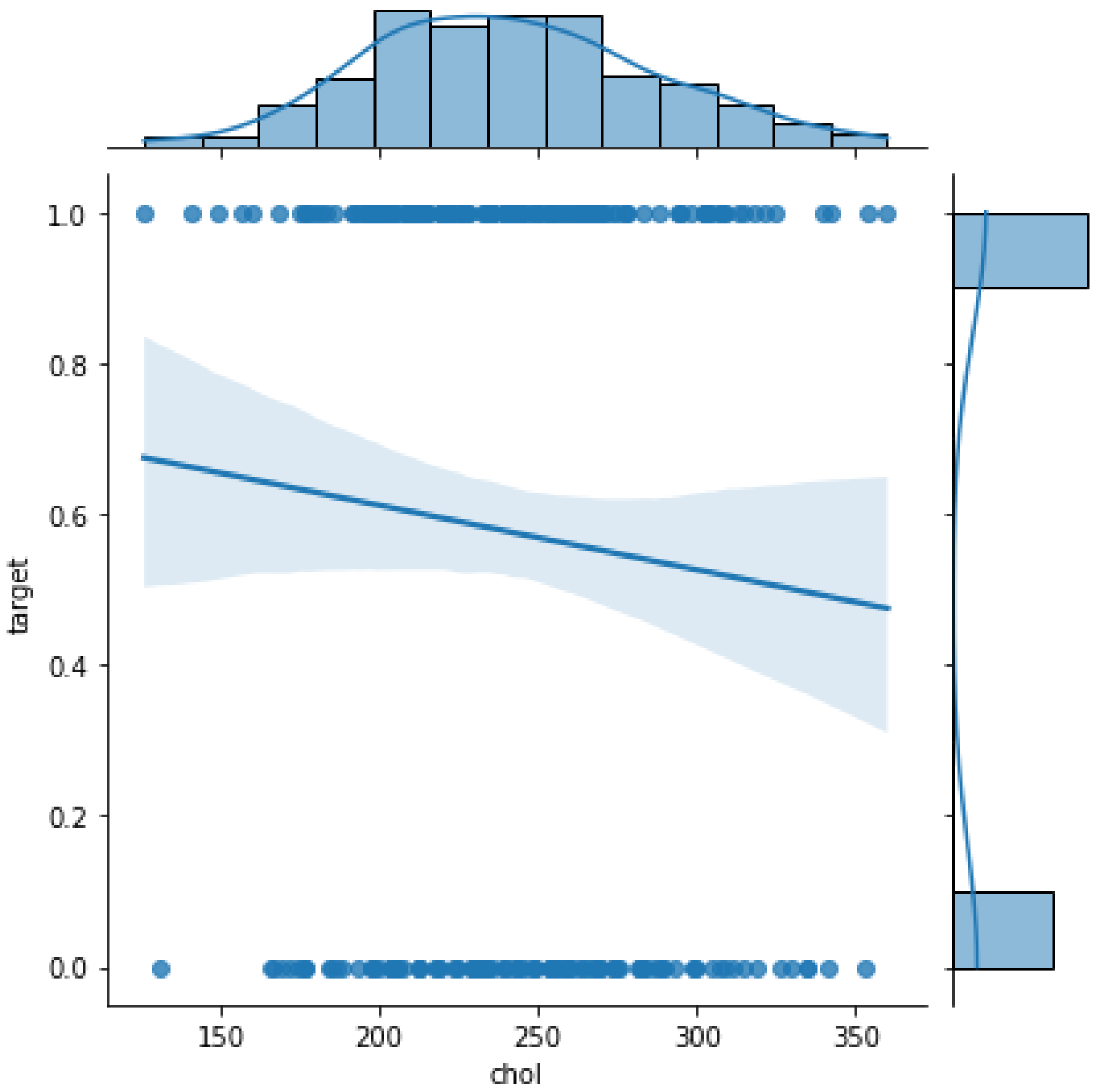
[73]:
```
# Joint plot between trestbps and target( Multivariate Analysis)
sns.jointplot(x='trestbps',y='target',data=Dataset,kind='hex')
```

[73]: <seaborn.axisgrid.JointGrid at 0x7f14dcf664d0>



[75]:
```
# Joint plot between chol and target( Multivariate Analysis)
sns.jointplot(x='chol',y='target',data=Dataset,kind='reg')
```

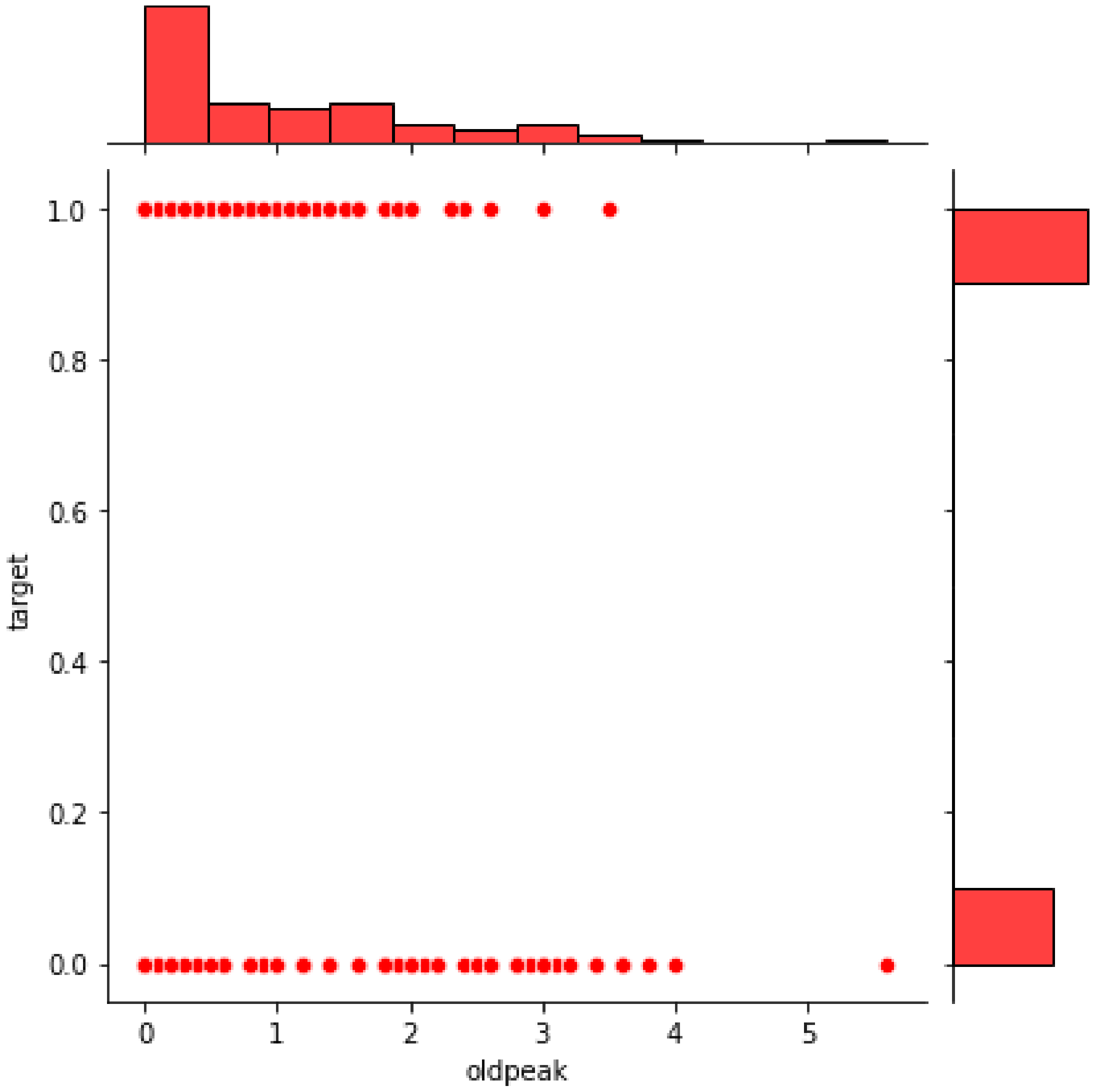
[75]: <seaborn.axisgrid.JointGrid at 0x7f14dcde6f10>



[76]:
```
# Joint plot between oldpeak and target( Multivariate Analysis)
sns.jointplot(Dataset['oldpeak'],Dataset['target'],color='r')
```

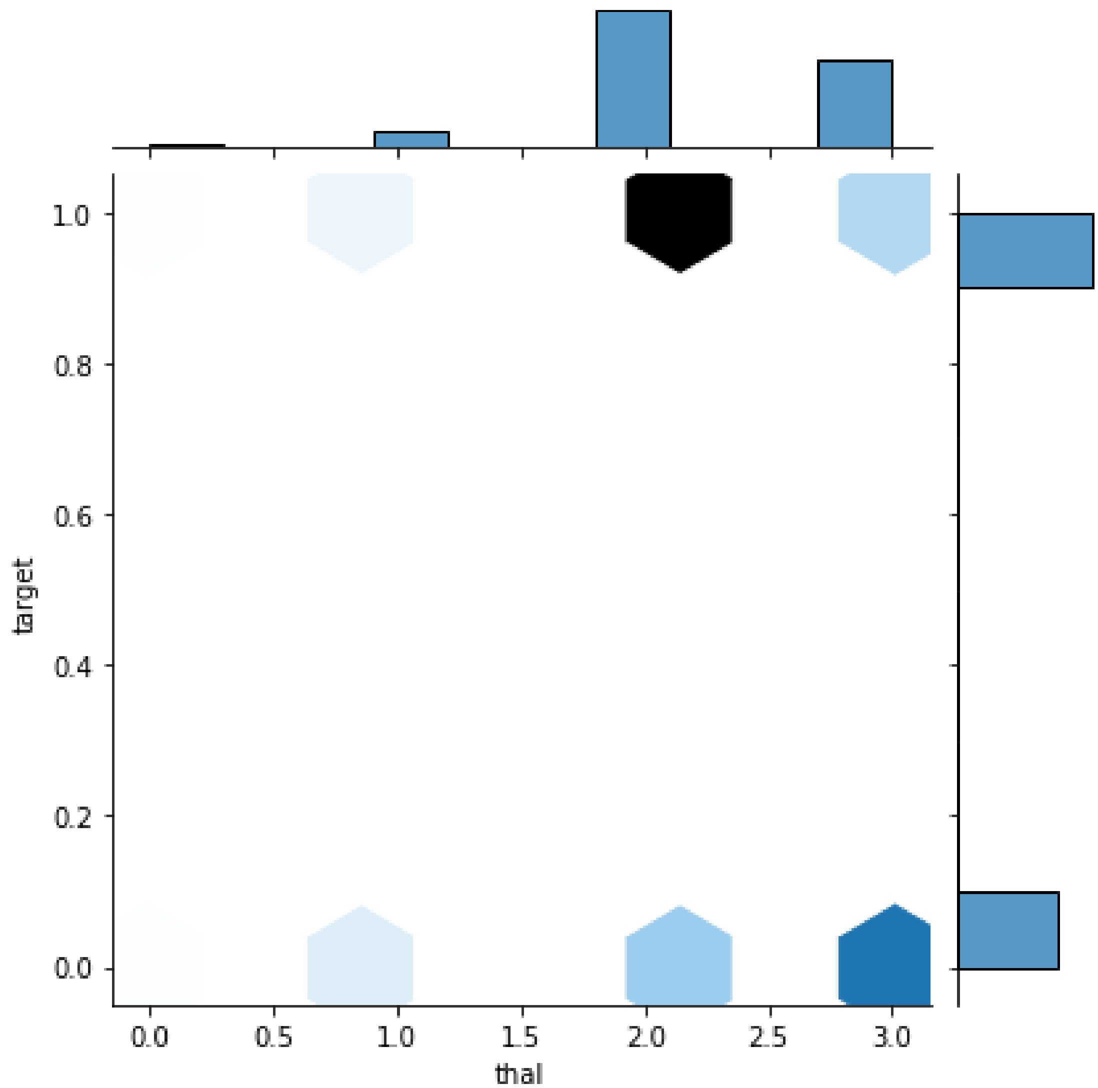
```
/usr/local/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning:
Pass the following variables as keyword args: x, y. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.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

[76]: <seaborn.axisgrid.JointGrid at 0x7f14dcc6b8d0>



[77]:
```
# Joint plot between thal and target( Multivariate Analysis)
sns.jointplot(x='thal', y='target', data=Dataset, kind='hex')
```

[77]: <seaborn.axisgrid.JointGrid at 0x7f14dcb5ced0>



Importing sklearn libraries to spilt the data sets in to number of samples and perform machine learning algorithms such as logistic regression and random forest to predict the results.

[79]:
```
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
```

[80]:
```
# Creating separate variables X and y holding features and target variables
X=Dataset.drop('target',axis=1)
y=df['target']
```

[81]:
```
# Performing Machine learning prediction
classification_models = []
classification_models.append(('Logistic Regression',
 LogisticRegression(solver="liblinear")))
classification_models.append(('Random Forest',
 RandomForestClassifier(n_estimators=100, criterion="entropy")))
```

[82]:
```
for name, model in classification_models:
  kfold = KFold(n_splits=10, random_state=(7), shuffle=(True))
  result = cross_val_score(model, X, y, cv=kfold, scoring='accuracy')
  print("%s: Mean Accuracy = %.2f%% - SD Accuracy = %.2f%%" % (name, result.
 mean()*100, result.std()*100))
```

```
Logistic Regression: Mean Accuracy = 85.54% - SD Accuracy = 7.89%
Random Forest: Mean Accuracy = 82.12% - SD Accuracy = 8.13%
```

Achieved mean accuracy of 86% by logistic regression classifiaction model and 82% mean accuracy by random forest classification model.

Before we have used kfold cross-validation process for preparing samples for prediction and tesing. Now using train & test splitting is done for model prediction. Second approach to check the accuracy with Kfold-crossvalidation technique

[83]:
```
#Splitting the dataset
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.
 25,random_state=42)
```

[84]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.fit_transform(X_test)
```

[85]:
```
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state=0,solver="liblinear")
classifier.fit(X_train,y_train)
```

[85]:
```
LogisticRegression(random_state=0, solver='liblinear')
```

[87]:
```
y_pred = classifier.predict(X_test)
```

from sklearn import metrics cm = metrics.confusion_matrix(y_test, y_pred) print(cm) accuracy = metrics.accuracy_score(y_test, y_pred) print("Accuracy score:",accuracy) precision = metrics.precision_score(y_test, y_pred) print("Precision score:",precision) recall = metrics.recall_score(y_test, y_pred) print("Recall score:",recall)

oe achieved almost same accuracy for logistic regression classification model using both k-fold cross-validation technique and train-test splits. With a precision of 0.875 and recall score of 0.8974

USING STATS MODEL:

[90]:
```python
import statsmodels.api as sm
log_reg = sm.Logit(y_train, X_train).fit()
```

```
Optimization terminated successfully.
         Current function value: 0.341254
         Iterations 7
```

[99]:
```python
y_pred = log_reg.predict(X_test)
```

[93]:
```python
prediction = list(map(round, y_pred))

# comparing original and predicted values of y
print('Actual values', list(y_test.values))
print('Predictions :', prediction)
```

```
Actual values [1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1,
1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1,
0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1]
Predictions : [1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1,
0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1,
1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1]
```

[95]:
```python
from sklearn.metrics import (confusion_matrix,
                             accuracy_score)

# confusion matrix
cm = confusion_matrix(y_test, prediction)
print ("Confusion Matrix : \n", cm)

# accuracy score of the model
print('Test accuracy = ', accuracy_score(y_test, prediction))
```

```
Confusion Matrix :
 [[22  5]
 [ 5 34]]
Test accuracy =  0.8484848484848485
```

Achieved accuracy of 85% using stats model.

[ ]:
```python

```