Project Healthcare

November 13, 2022

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

[2]:
```
health = pd.read_excel('C:\\Sachin new\\Simplilearn\\Course 3 - Machine␣
↪Learning\\Project - Healthcare\\1645792390_cep1_dataset.xlsx')
```

0.0.1 1. Preliminary Analysis

0.0.2 Step 1(a) Structure of data and finding missing values

[3]:
```
health.shape
```

[3]:
```
(303, 14)
```

[4]:
```
health.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
```

[5]:
```
health.head()
```

[5]:
```
   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
```

[6]:
```
health.target.value_counts()
```

[6]:
```
1    165
0    138
Name: target, dtype: int64
```

[7]:
```
health.isna().sum()
```

[7]:
```
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
```

0.0.3 1(b) Remove duplicates

[8]:
```python
health[health.duplicated(keep = False)]
```

[8]:
```
     age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  \
163   38    1   2       138   175    0        1      173      0      0.0
164   38    1   2       138   175    0        1      173      0      0.0

     slope  ca  thal  target
163      2   4     2       1
164      2   4     2       1
```

[9]:
```python
health.drop_duplicates(inplace=True)
```

[ ]:
```python
# There was one duplicate item which was removed from the database.
```

[10]:
```python
health.shape
```

[10]:
```
(302, 14)
```

0.0.4 Step 2(a) Statistical summary and spread of data

[11]:
```python
health.describe().T
```

[11]:
```
          count        mean        std    min     25%    50%     75%    max
age       302.0   54.420530   9.047970   29.0   48.00   55.5   61.00   77.0
sex       302.0    0.682119   0.466426    0.0    0.00    1.0    1.00    1.0
cp        302.0    0.963576   1.032044    0.0    0.00    1.0    2.00    3.0
trestbps  302.0  131.602649  17.563394   94.0  120.00  130.0  140.00  200.0
chol      302.0  246.500000  51.753489  126.0  211.00  240.5  274.75  564.0
fbs       302.0    0.149007   0.356686    0.0    0.00    0.0    0.00    1.0
restecg   302.0    0.526490   0.526027    0.0    0.00    1.0    1.00    2.0
thalach   302.0  149.569536  22.903527   71.0  133.25  152.5  166.00  202.0
exang     302.0    0.327815   0.470196    0.0    0.00    0.0    1.00    1.0
oldpeak   302.0    1.043046   1.161452    0.0    0.00    0.8    1.60    6.2
slope     302.0    1.397351   0.616274    0.0    1.00    1.0    2.00    2.0
ca        302.0    0.718543   1.006748    0.0    0.00    0.0    1.00    4.0
thal      302.0    2.314570   0.613026    0.0    2.00    2.0    3.00    3.0
target    302.0    0.543046   0.498970    0.0    0.00    1.0    1.00    1.0
```

[12]:
```python
health.mode()
```

[12]:
```
    age  sex   cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  \
0  58.0  1.0  0.0     120.0   197  0.0      1.0    162.0    0.0      0.0
1   NaN  NaN  NaN       NaN   204  NaN      NaN      NaN    NaN      NaN
2   NaN  NaN  NaN       NaN   234  NaN      NaN      NaN    NaN      NaN

   slope   ca  thal  target
```

```
0    2.0  0.0   2.0     1.0
1    NaN  NaN   NaN     NaN
2    NaN  NaN   NaN     NaN
```

[13]:
```
health.var()
```

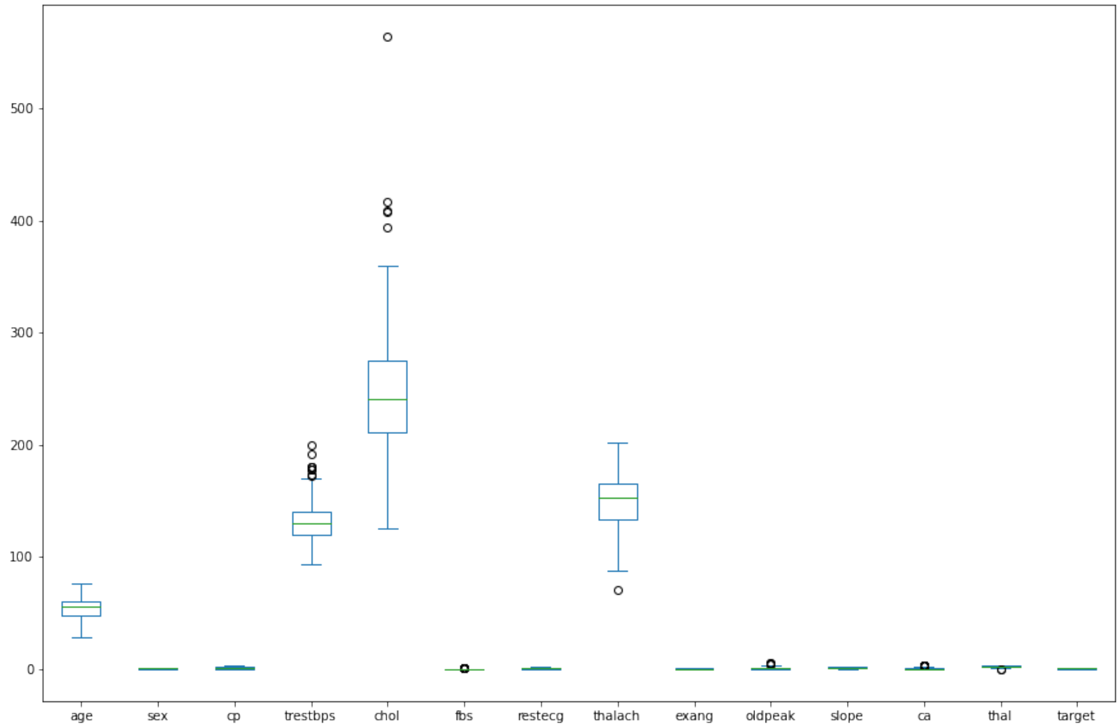
[13]:
```
age           81.865757
sex            0.217553
cp             1.065114
trestbps     308.472817
chol        2678.423588
fbs            0.127225
restecg        0.276705
thalach      524.571561
exang          0.221084
oldpeak        1.348971
slope          0.379794
ca             1.013542
thal           0.375800
target         0.248971
dtype: float64
```

[14]:
```
health.kurtosis()
```

[14]:
```
age        -0.527512
sex        -1.391273
cp         -1.183729
trestbps    0.922996
chol        4.542591
fbs         1.937947
restecg    -1.359464
thalach    -0.062186
exang      -1.466170
oldpeak     1.567876
slope      -0.629935
ca          0.781003
thal        0.295855
target     -1.983008
dtype: float64
```

[15]:
```
health.plot(kind = 'box', figsize=(15,10))
```

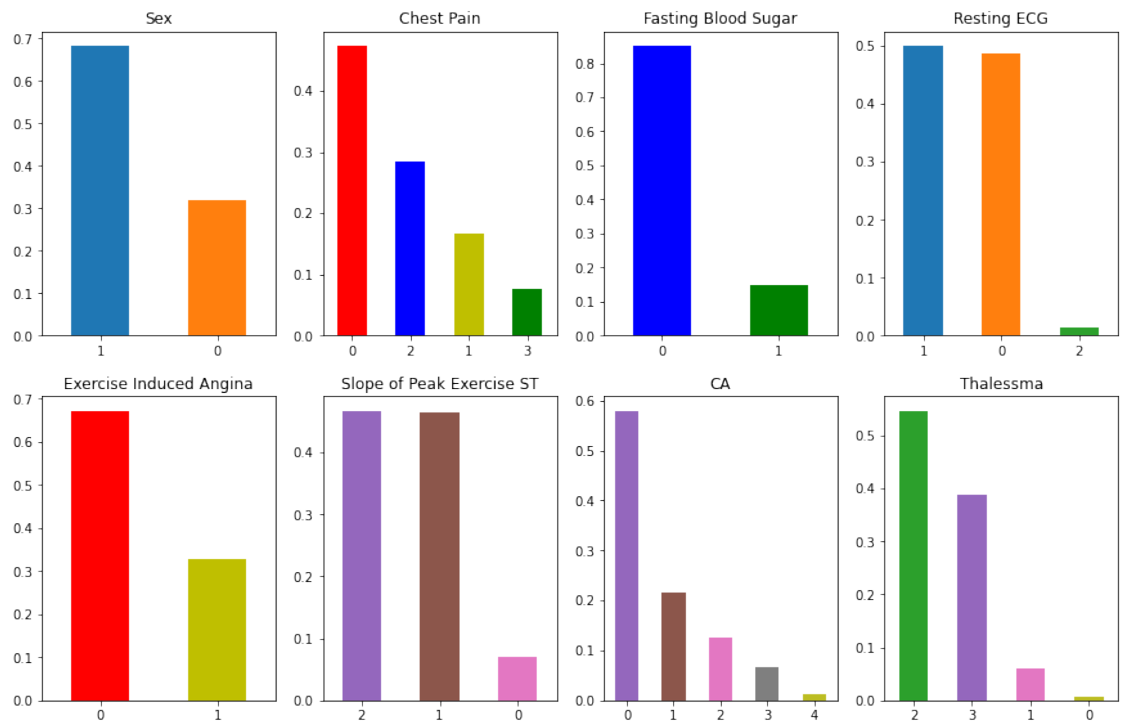
[15]:
```
<AxesSubplot:>
```



0.0.5 2(b) Describe categorical variables

[16]:
```
fig = plt.figure(figsize = (15,20))
ax1 = fig.add_subplot(441); ax1.title.set_text('Sex');
health.sex.value_counts(normalize=True).plot(kind='bar', color = ['C0','C1']);
    ↪plt.xticks(rotation=0);
ax2 = fig.add_subplot(442); ax2.title.set_text('Chest Pain');
health.cp.value_counts(normalize=True).plot(kind='bar', color =
    ↪['r','b','y','g']); plt.xticks(rotation=0)
ax3 = fig.add_subplot(443); ax3.title.set_text('Fasting Blood Sugar');
health.fbs.value_counts(normalize=True).plot(kind='bar', color = ['b','g']);
    ↪plt.xticks(rotation=0)
ax4 = fig.add_subplot(444); ax4.title.set_text('Resting ECG');
health.restecg.value_counts(normalize=True).plot(kind='bar', color =
    ↪['C0','C1','C2']); plt.xticks(rotation=0)
ax5 = fig.add_subplot(445); ax5.title.set_text('Exercise Induced Angina');
health.exang.value_counts(normalize=True).plot(kind='bar', color = ['r','y']);
    ↪plt.xticks(rotation=0)
ax6 = fig.add_subplot(446); ax6.title.set_text('Slope of Peak Exercise ST');
health.slope.value_counts(normalize=True).plot(kind='bar', color =
    ↪['C4','C5','C6']); plt.xticks(rotation=0)
ax7 = fig.add_subplot(447); ax7.title.set_text('CA');
```

```
health.ca.value_counts(normalize=True).plot(kind='bar', color =
 ↪['C4','C5','C6','C7','C8']); plt.xticks(rotation=0)
ax8 = fig.add_subplot(448); ax8.title.set_text('Thalessma');
health.thal.value_counts(normalize=True).plot(kind='bar', color =
 ↪['C2','C4','C6','C8']); plt.xticks(rotation=0)

plt.show()
```



0.0.6 2(c) Occurrence of CVD across the Age category

[17]:
```
bins = [0,40,60,80]
labels = ['0-40','40-60','60-80']
health['AgeCategory']=pd.cut(health['age'],bins = bins, labels = labels, right
 ↪= False)
```

[18]:
```
health.head()
```

[18]:

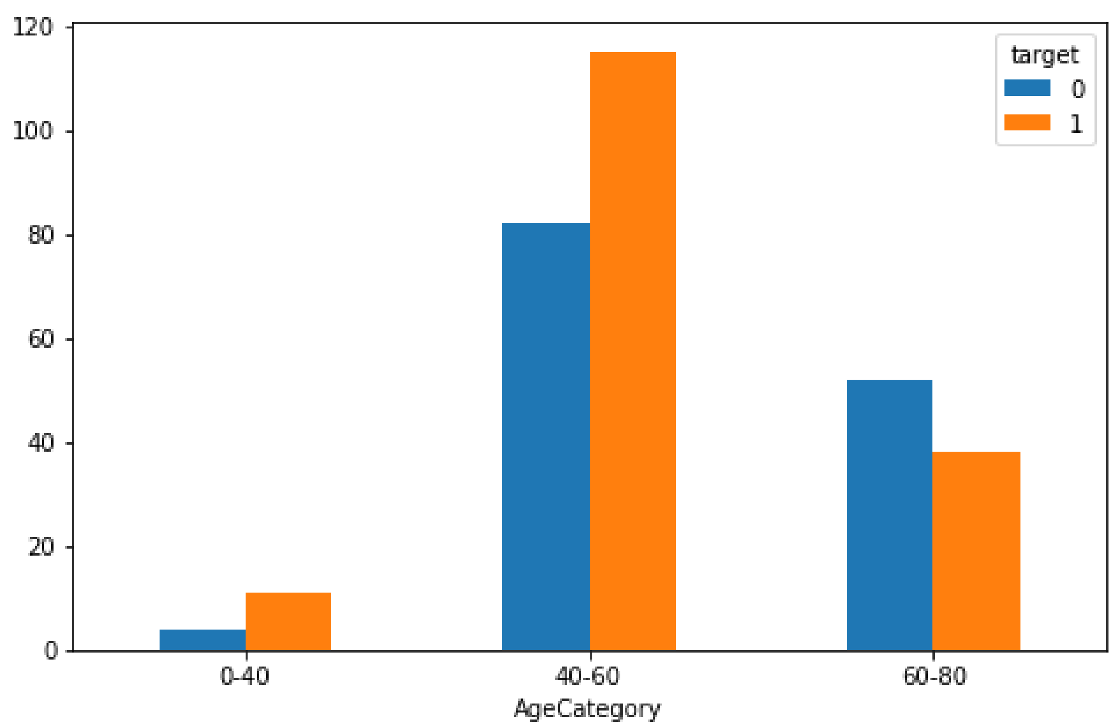
	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	\
0	63	1	3	145	233	1	0	150	0	2.3	0	
1	37	1	2	130	250	0	1	187	0	3.5	0	
2	41	0	1	130	204	0	0	172	0	1.4	2	
3	56	1	1	120	236	0	1	178	0	0.8	2	
4	57	0	0	120	354	0	1	163	1	0.6	2	

```
   ca  thal  target AgeCategory
0   0     1       1       60-80
1   0     2       1        0-40
2   0     2       1       40-60
3   0     2       1       40-60
4   0     2       1       40-60
```

[19]:
```
health.groupby(['AgeCategory','target']).count()['age'].to_frame()
```

[19]:
```
                    age
AgeCategory target
0-40        0         4
            1        11
40-60       0        82
            1       115
60-80       0        52
            1        38
```

[20]:
```
label_agecat = np.array([0,1,2])
label_agecat2 = ['0-40', '40-60','60-80']

ax = pd.crosstab(health.AgeCategory, health.target).plot(kind='bar',␣
↪figsize=(8, 5));
plt.xticks(label_agecat, label_agecat2, rotation=0);
```



```
[ ]: # The age category 40-60 was more prone to CVD as compared to other age groups.
```

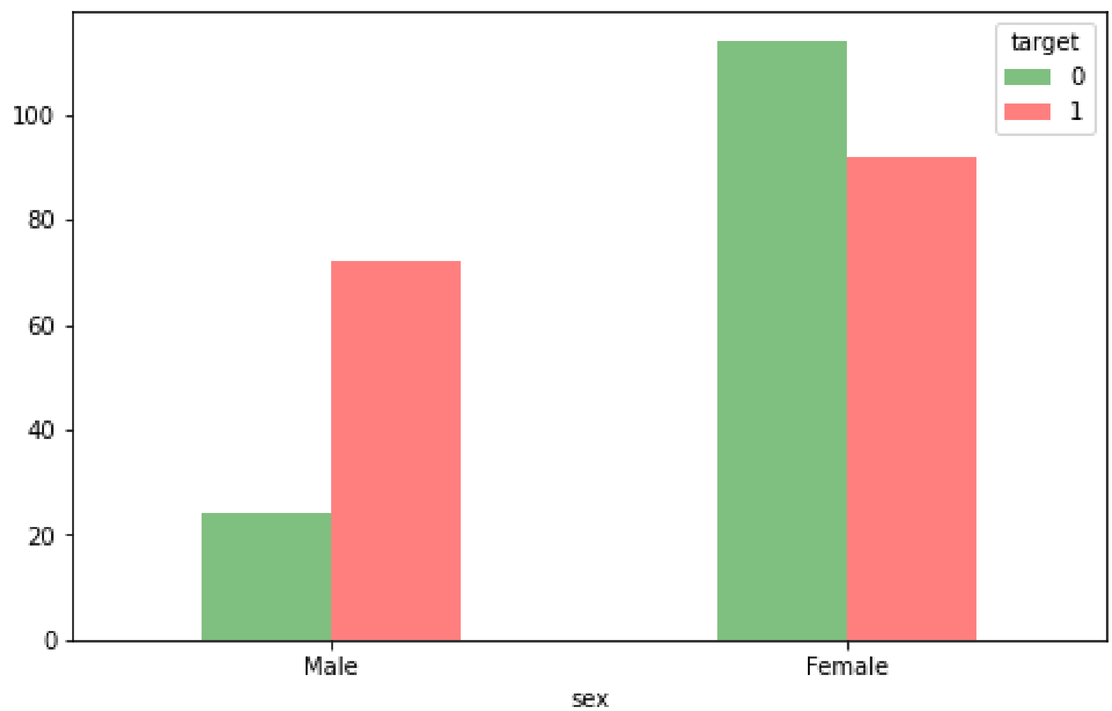
0.0.7 2(d) Composition of all patients with respect to Sex category

```
[21]: health.groupby(['sex','target']).count()['age'].to_frame()
```

[21]:

		age
sex	target	
0	0	24
	1	72
1	0	114
	1	92

```
[22]: label_sexcat = np.array([0,1])
label_sexcat2 = ['Male', 'Female']

ax = pd.crosstab(health.sex, health.target).plot(kind='bar', figsize=(8, 5),␣
 ↪color = ('g','r'), alpha = 0.5);
plt.xticks(label_sexcat, label_sexcat2, rotation=0);
```




```
[ ]: # Females were more prone to CVD as compared to Males.
```

0.0.8 2(e) Detect heart attacks based on anomalies in the resting blood pressure of a patient

[23]:
```
Q1, Q3 = health['trestbps'].quantile([0.25,0.75])
IQR = Q3-Q1
lower_range = Q1 - (1.5 * IQR)
upper_range = Q3 + (1.5 * IQR)

trestbps_out = health[(health['trestbps'] < lower_range) | (health['trestbps']>␣
↪upper_range)]
```

[24]:
```
trestbps_out[['trestbps','target']]
```

[24]:

	trestbps	target
8	172	1
101	178	1
110	180	1
203	180	0
223	200	0
241	174	0
248	192	0
260	178	0
266	180	0

[25]:
```
# since amongst the outliers i.e. anomalies in resting blood pressure, at␣
↪similar trestbps,
# there are both cases of people having CVD and people not having CVD.
# Hence, it is inconclusive to detect heart attacks only based on anomalies in␣
↪resting blood pressure.
```

0.0.9 2(f) Relationship between cholesterol levels and a target variable

[26]:
```
bins = [125,200,240,300,375,600]
labels = ['125-200','200-240','240-300','300-375','375-600']
health['CholCategory']=pd.cut(health['chol'],bins = bins, labels = labels,␣
↪right = False)
```

[27]:
```
health.head()
```

[27]:

	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	

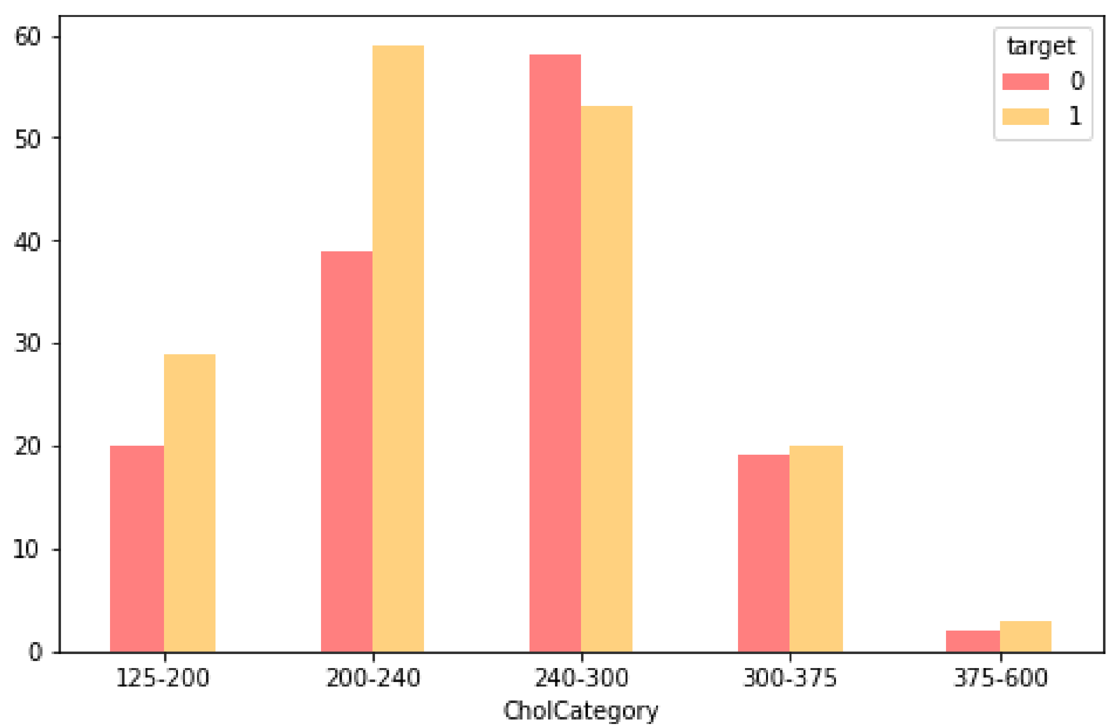
	ca	thal	target	AgeCategory	CholCategory
0	0	1	1	60-80	200-240

```
1    0    2    1      0-40    240-300
2    0    2    1     40-60    200-240
3    0    2    1     40-60    200-240
4    0    2    1     40-60    300-375
```

[28]:
```python
health.groupby(['CholCategory','target']).count()['age'].to_frame()
```

[28]:
```
                     age
CholCategory target
125-200      0        20
             1        29
200-240      0        39
             1        59
240-300      0        58
             1        53
300-375      0        19
             1        20
375-600      0         2
             1         3
```

[29]:
```python
label_cholcat = np.array([0,1,2,3,4])
label_cholcat2 = ['125-200', '200-240','240-300','300-375', '375-600']

ax = pd.crosstab(health.CholCategory, health.target).plot(kind='bar',␣
 ↪figsize=(8, 5), color = ('red','orange'), alpha = 0.5);
plt.xticks(label_cholcat, label_cholcat2, rotation=0);
```



[30]:
```
Q1, Q3 = health['chol'].quantile([0.25,0.75])
IQR = Q3-Q1
lower_range = Q1 - (1.5 * IQR)
upper_range = Q3 + (1.5 * IQR)

print(IQR)
print(lower_range)
print(upper_range)
chol_out = health[(health['chol'] < lower_range) | (health['chol']>␣
↪upper_range)]
```

```
63.75
115.375
370.375
```

[31]:
```
chol_out[['chol','target']]
```

[31]:

	chol	target
28	417	1
85	564	1
96	394	1
220	407	0
246	409	0

[ ]:
```
# People having cholestrol levels between 200 to 300 were more prone to having␣
↪CVD.
# since amongst the outliers i.e. people having very high cholestrol levels,␣
↪there are both cases of people having CVD and people not having CVD.
# Hence, it is inconclusive to detect heart attacks only based on very high␣
↪cholestrol levels.
```

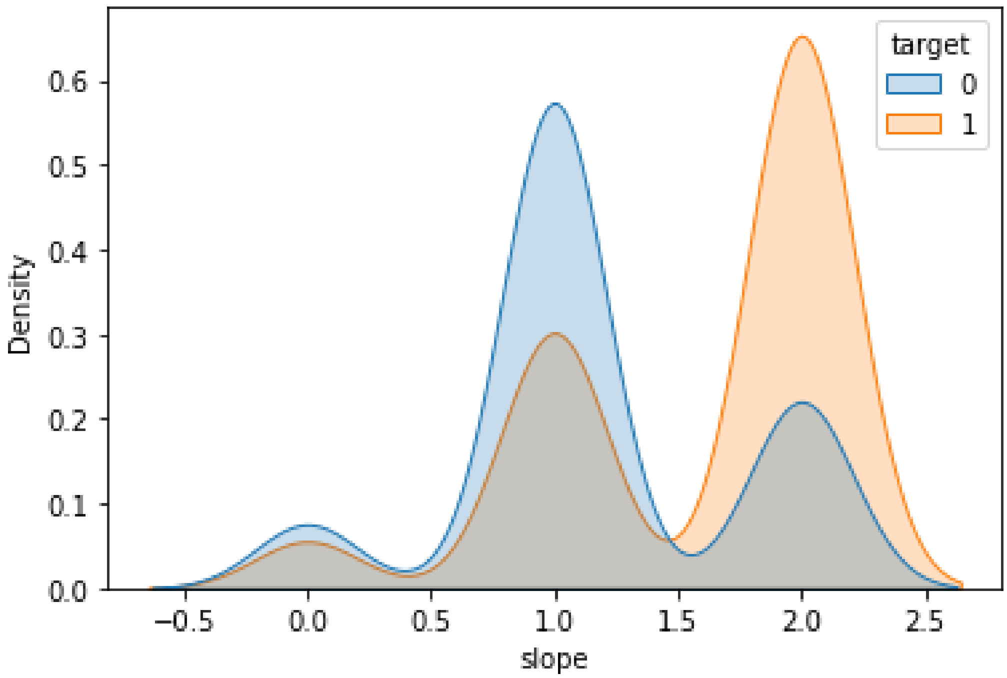
0.0.10 2(g) Relationship between peak exercising and the occurrence of a heart attack

[32]:
```
health.target.corr(health.slope).round(2)
```

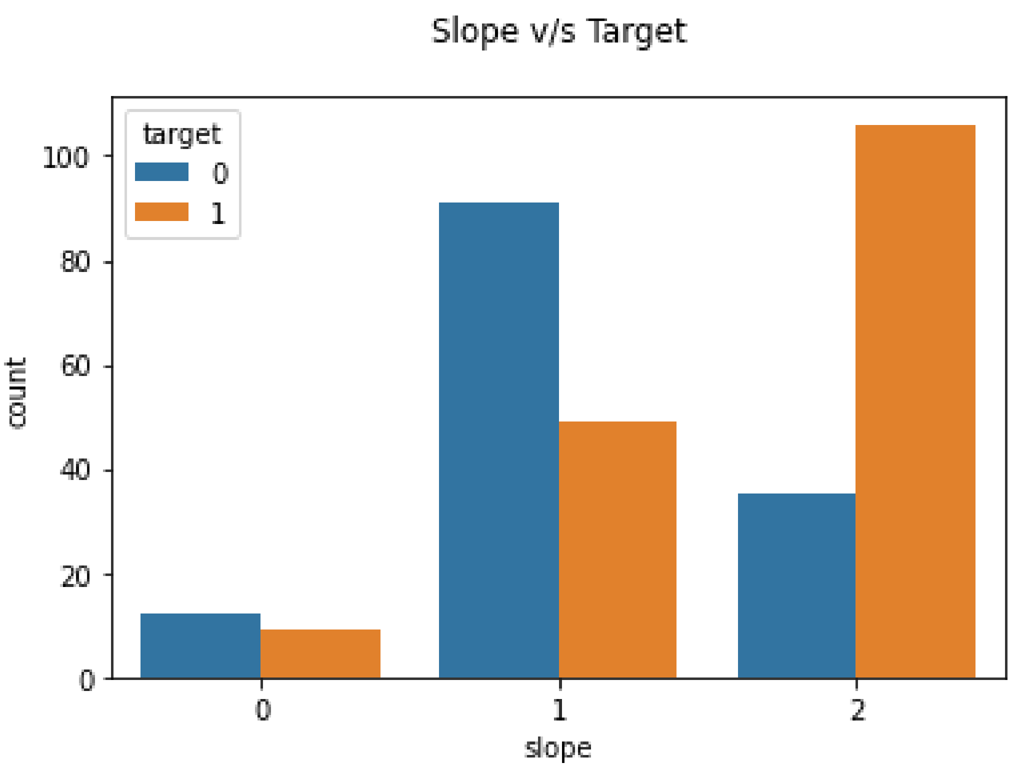
[32]: 0.34

[33]:
```
sns.kdeplot(health['slope'],hue=health['target'],shade = True)
```

[33]: <AxesSubplot:xlabel='slope', ylabel='Density'>



[34]:
```
sns.countplot(data= health, x='slope',hue='target')
plt.title('Slope v/s Target\n');
```



[ ]:
```
# People with slope = 2 i.e. peak exercise ST segment were more prone to having CVD.
```

0.0.11 2(h) Whether Thalassemia is a major cause of CVD

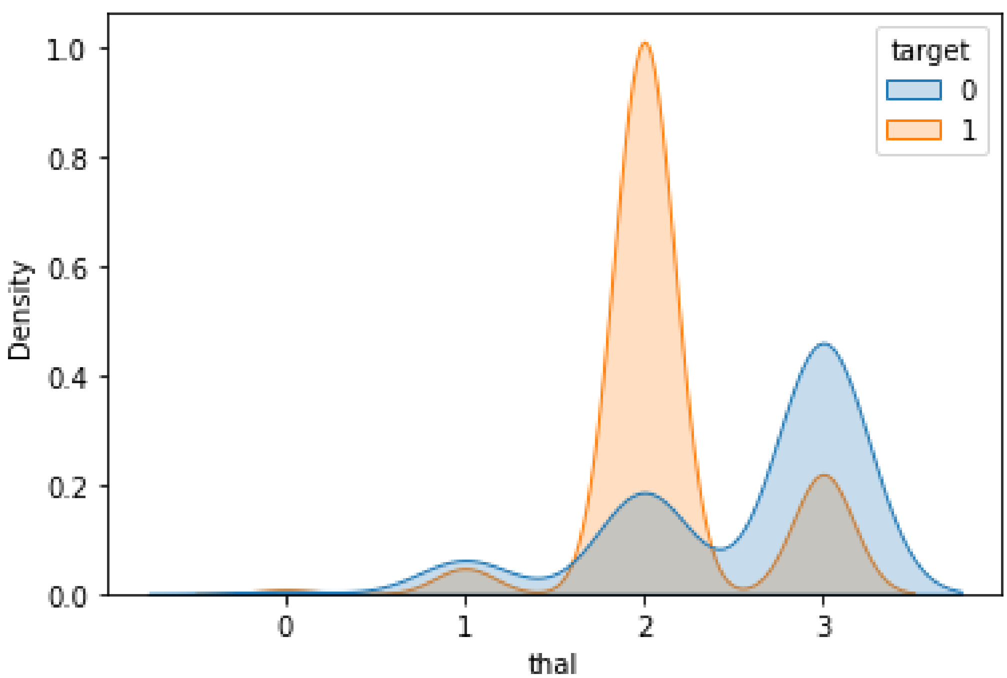
[35]:
```
pd.crosstab(health.thal,health.target)
```

[35]:

target	0	1
thal		
0	1	1
1	12	6
2	36	129
3	89	28

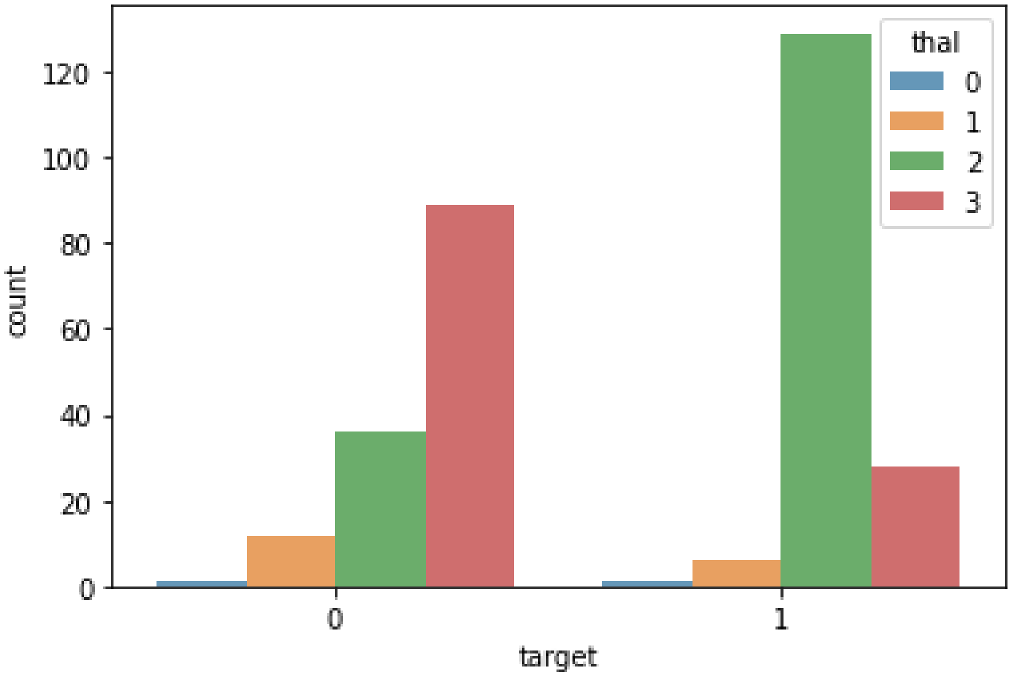
[36]:
```
sns.kdeplot(health['thal'],hue=health['target'],shade = True)
```

[36]: <AxesSubplot:xlabel='thal', ylabel='Density'>



[37]:
```
sns.countplot(data= health, x='target',hue='thal', alpha = 0.75)
```

[37]: <AxesSubplot:xlabel='target', ylabel='count'>



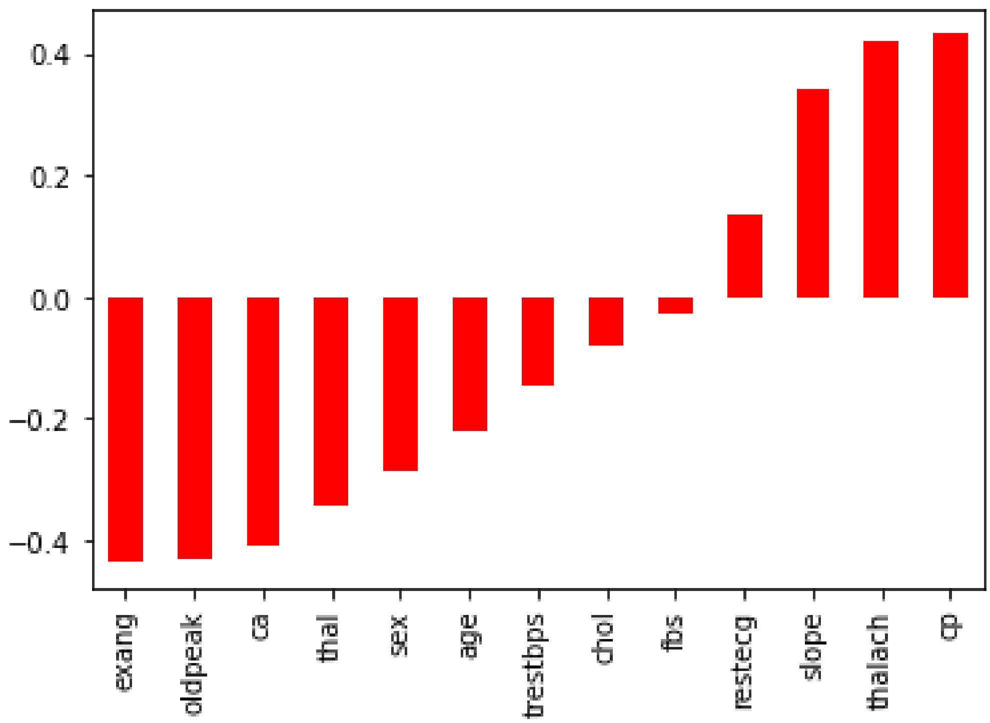

```
[38]: health.target.corr(health.thal).round(2)
```

```
[38]: -0.34
```

```
[ ]: # W.r.t. Thalesima, people with thal = 2 were more prone to having CVD.
```

0.0.12 2(i) List how the other factors determine the occurrence of CVD

```
[39]: health.corr()['target'].sort_values().drop('target').plot(kind = 'bar',␣
      ↪color="red");
```



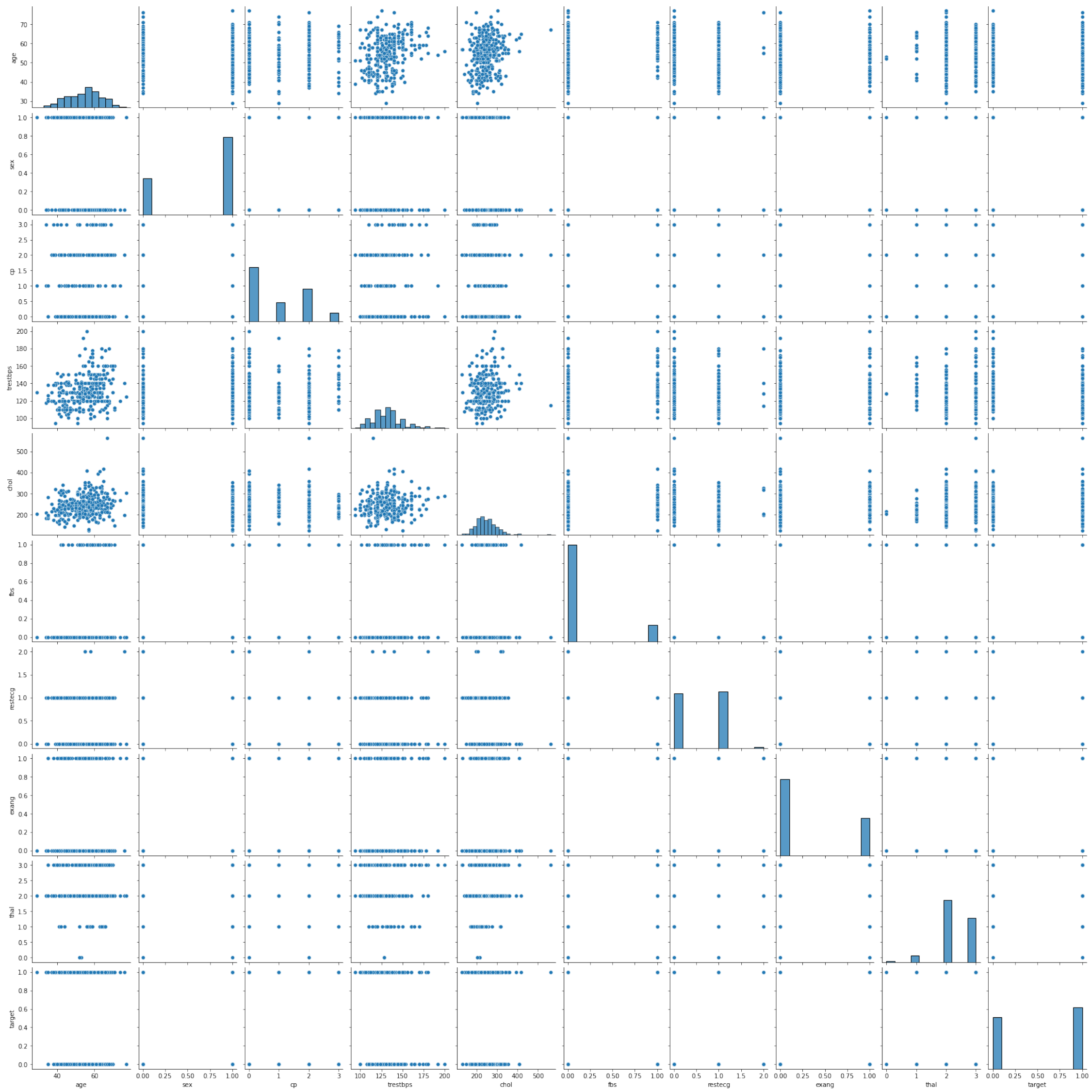
0.0.13 2 (j) Pair plot

[40]:
```
health.columns
```

[40]:
```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target', 'AgeCategory',
       'CholCategory'],
      dtype='object')
```

[41]:
```
# considering too many variables, have selected important columns for the␣
↪pairplot.

sns.pairplot(health,vars = ['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs',␣
↪'restecg', 'exang','thal', 'target'])
```

[41]:
```
<seaborn.axisgrid.PairGrid at 0x2ca408d5df0>
```



0.0.14 3. Baseline model using Logistic Regression

[42]:
```
x = health.drop(['target','AgeCategory','CholCategory'], axis = 1)
```

[43]:
```
y = health['target']
```

[44]:
```
# Splitting the data into training set and test set

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 = 15)
```

[45]:
```
x_train
```

[45]:
```
     age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  \
226   62    1   1       120   281    0        0      103      0      1.4
129   74    0   1       120   269    0        0      121      1      0.2
224   54    1   0       110   239    0        1      126      1      2.8
191   58    1   0       128   216    0        0      131      1      2.2
20    59    1   0       135   234    0        1      161      0      0.5
..   …  …  ..       …   …  …      …    …      …
200   44    1   0       110   197    0        0      177      0      0.0
155   58    0   0       130   197    0        1      131      0      0.6
156   47    1   2       130   253    0        1      179      0      0.0
133   41    1   1       110   235    0        1      153      0      0.0
246   56    0   0       134   409    0        0      150      1      1.9

     slope  ca  thal
226      1   1     3
129      2   1     2
224      1   1     3
191      1   3     3
20       1   0     3
..     …  ..   …
200      2   1     2
155      1   0     2
156      2   0     2
133      2   0     2
246      1   2     3

[241 rows x 13 columns]
```

[46]:
```
x_test
```

[46]:
```
     age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  \
75    55    0   1       135   250    0        0      161      0      1.4
288   57    1   0       110   335    0        1      143      1      3.0
64    58    1   2       140   211    1        0      165      0      0.0
94    45    0   1       112   160    0        1      138      0      0.0
144   76    0   2       140   197    0        2      116      0      1.1
..   …  …  ..       …   …  …      …    …      …
50    51    0   2       130   256    0        0      149      0      0.5
97    52    1   0       108   233    1        1      147      0      0.1
168   63    1   0       130   254    0        0      147      0      1.4
297   59    1   0       164   176    1        0       90      0      1.0
295   63    1   0       140   187    0        0      144      1      4.0

     slope  ca  thal
75       1   0     2
```

```
288       1   1     3
64        2   0     2
94        1   0     2
144       1   0     2
..      …   ..   …
50        2   0     2
97        2   3     3
168       1   1     3
297       1   2     1
295       2   2     3

[61 rows x 13 columns]
```

[47]:
```python
y_train
```

[47]:
```
226    0
129    1
224    0
191    0
20     1
      ..
200    0
155    1
156    1
133    1
246    0
Name: target, Length: 241, dtype: int64
```

[48]:
```python
y_test
```

[48]:
```
75     1
288    0
64     1
94     1
144    1
      ..
50     1
97     1
168    0
297    0
295    0
Name: target, Length: 61, dtype: int64
```

[49]:
```python
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix , classification_report
```

[50]:
```python
LR = LogisticRegression(random_state = 10)
```

[51]:
```
LR.fit(x_train,y_train);
```

```
C:\Users\14sac\anaconda3\lib\site-
packages\sklearn\linear_model\_logistic.py:814: 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
  n_iter_i = _check_optimize_result(
```

[52]:
```
y_pred = LR.predict(x_test)
```

[53]:
```
y_pred
```

[53]:
```
array([1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0,
       0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0,
       1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0], dtype=int64)
```

[54]:
```
print(confusion_matrix(y_test,y_pred))
```

```
[[15 11]
 [ 6 29]]
```

[55]:
```
print(classification_report(y_test , y_pred))
```

```
              precision    recall  f1-score   support

           0       0.71      0.58      0.64        26
           1       0.72      0.83      0.77        35

    accuracy                           0.72        61
   macro avg       0.72      0.70      0.71        61
weighted avg       0.72      0.72      0.72        61
```

[56]:
```
pd.DataFrame(data = {'Columns' : x_train.columns , 'Betas' : LR.coef_.
    ↪flatten()}).sort_values('Betas')
```

[56]:

	Columns	Betas
12	thal	-1.327547
11	ca	-1.052569
1	sex	-1.048218
8	exang	-0.822397
9	oldpeak	-0.475267
5	fbs	-0.198236

```
3   trestbps -0.009206
4       chol  0.001174
0        age  0.013713
7    thalach  0.024626
10     slope  0.465925
6    restecg  0.615359
2         cp  0.890187
```

0.0.15 3(b) Baseline model using Random Forest

[57]:
```
from sklearn.ensemble import RandomForestClassifier
```

[58]:
```
rf = RandomForestClassifier(n_estimators=50, random_state = 50)
```

[59]:
```
rf.fit(x_train, y_train)
```

[59]:
```
RandomForestClassifier(n_estimators=50, random_state=50)
```

[60]:
```
y_pred1 = rf.predict(x_test)
```

[61]:
```
y_pred1
```

[61]:
```
array([1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0,
       0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1,
       1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0], dtype=int64)
```

[62]:
```
print(confusion_matrix(y_test,y_pred1))
```

```
[[20  6]
 [ 5 30]]
```

[63]:
```
print(classification_report(y_test , y_pred1))
```

```
              precision    recall  f1-score   support

           0       0.80      0.77      0.78        26
           1       0.83      0.86      0.85        35

    accuracy                           0.82        61
   macro avg       0.82      0.81      0.81        61
weighted avg       0.82      0.82      0.82        61
```

[ ]:
```
# Accuracy score of 82% using the Random Forest model was providing the best␣
↪prediction as compared to Logistic Regression.
```

[64]:
```
pd.DataFrame(rf.feature_importances_ , index = x_train.columns).sort_values(0 ,␣
↪ascending = False)
```

[64]:
```
                0
ca        0.150653
thal      0.133014
cp        0.123803
thalach   0.114281
oldpeak   0.103517
chol      0.077661
age       0.076907
trestbps  0.063742
slope     0.055882
exang     0.044801
sex       0.030014
restecg   0.018991
fbs       0.006734
```

0.0.16 Using GridSearchCV to fine tune the parameters

[65]:
```python
from sklearn.model_selection import GridSearchCV
```

[66]:
```python
param_grid = {'n_estimators': [20,30,50,100,150,200,250],
              'criterion': ['gini' , 'entropy'],
              'max_depth' : [3, 5, 10,20],
              'min_samples_split' : [5 , 10, 20,30,50]
             }
```

[67]:
```python
grid = GridSearchCV( rf, param_grid , refit = True , verbose = 1, n_jobs = -1 ,␣
 ↪cv = 2)

grid.fit(x_train,y_train)
```

```
Fitting 2 folds for each of 280 candidates, totalling 560 fits
```

[67]:
```
GridSearchCV(cv=2,
             estimator=RandomForestClassifier(n_estimators=50, random_state=50),
             n_jobs=-1,
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [3, 5, 10, 20],
                         'min_samples_split': [5, 10, 20, 30, 50],
                         'n_estimators': [20, 30, 50, 100, 150, 200, 250]},
             verbose=1)
```

[68]:
```python
grid_predictions = grid.predict(x_test)
```

[69]:
```python
print(confusion_matrix(y_test,grid_predictions))
```

```
[[18  8]
 [ 7 28]]
```

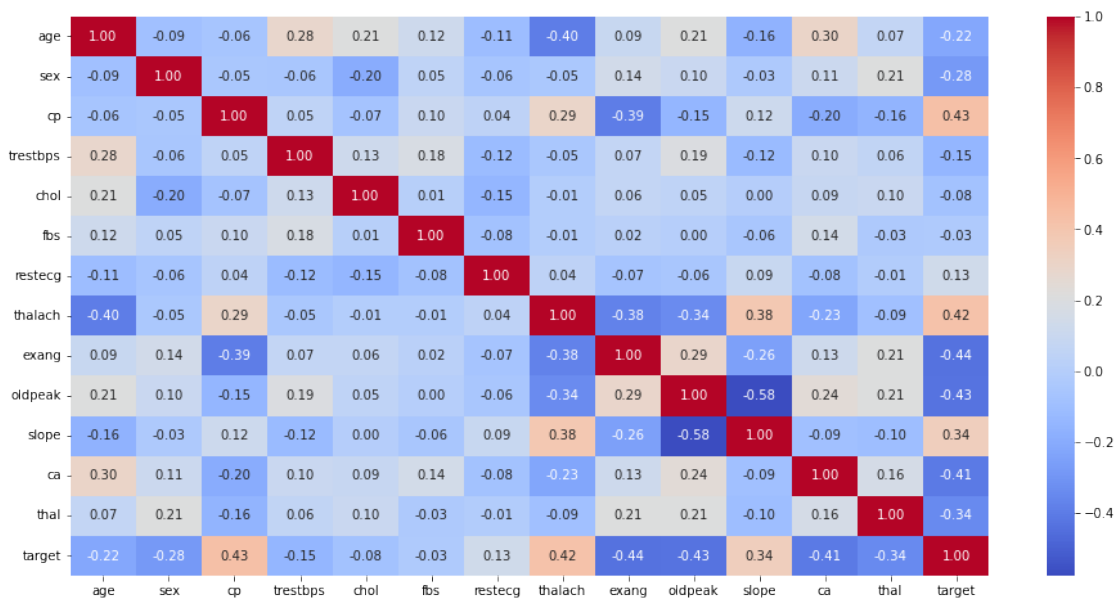
```
[70]: print(classification_report(y_test,grid_predictions))
```

```
              precision    recall  f1-score   support

           0       0.72      0.69      0.71        26
           1       0.78      0.80      0.79        35

    accuracy                           0.75        61
   macro avg       0.75      0.75      0.75        61
weighted avg       0.75      0.75      0.75        61
```

```
[71]: grid.best_params_
```

```
[71]: {'criterion': 'gini',
       'max_depth': 5,
       'min_samples_split': 30,
       'n_estimators': 100}
```

```
[72]: plt.figure(figsize= (16, 8))
      sns.heatmap(health.corr(), annot = True, cmap= 'coolwarm', fmt= '.2f');
```



0.0.17 Logistic regression values using Stats Model

```
[73]: lr_sm = sm.Logit(y_train, x_train).fit()
```

```
Optimization terminated successfully.
         Current function value: 0.317146
```

```
              Iterations 7
```

[74]:
```
print(lr_sm.summary())
```

```
                           Logit Regression Results
==============================================================================
Dep. Variable:                 target   No. Observations:                  241
Model:                          Logit   Df Residuals:                      228
Method:                           MLE   Df Model:                           12
Date:                Sun, 13 Nov 2022   Pseudo R-squ.:                  0.5408
Time:                        17:34:30   Log-Likelihood:                -76.432
converged:                       True   LL-Null:                       -166.45
Covariance Type:            nonrobust   LLR p-value:                 4.201e-32
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
age            0.0193      0.023      0.844      0.399      -0.025       0.064
sex           -1.2902      0.505     -2.557      0.011      -2.279      -0.301
cp             0.9667      0.236      4.105      0.000       0.505       1.428
trestbps      -0.0087      0.011     -0.759      0.448      -0.031       0.014
chol           0.0007      0.005      0.144      0.886      -0.009       0.010
fbs           -0.3421      0.648     -0.528      0.598      -1.613       0.928
restecg        0.7024      0.406      1.731      0.084      -0.093       1.498
thalach        0.0251      0.009      2.749      0.006       0.007       0.043
exang         -1.0469      0.481     -2.177      0.029      -1.989      -0.104
oldpeak       -0.4570      0.251     -1.819      0.069      -0.949       0.035
slope          0.5837      0.397      1.470      0.142      -0.195       1.362
ca            -1.1812      0.278     -4.251      0.000      -1.726      -0.637
thal          -1.4187      0.378     -3.755      0.000      -2.159      -0.678
==============================================================================
```

[75]:
```
y_pred2=lr_sm.predict(x_test)
```

[76]:
```
prediction = list(map(round, y_pred2))
```

[77]:
```
print('Actual values: ', list(y_test.values))
print('Predictions :', prediction)
```

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

[78]:
```
print(confusion_matrix(y_test,prediction))
```

```
[[15 11]
 [ 7 28]]
```

[79]:
```
print(classification_report(y_test,prediction))
```

```
              precision    recall  f1-score   support

           0       0.68      0.58      0.62        26
           1       0.72      0.80      0.76        35

    accuracy                           0.70        61
   macro avg       0.70      0.69      0.69        61
weighted avg       0.70      0.70      0.70        61
```

0.0.18 Using feature selection by dropping features having p-value <0.05

[80]:
```
x1 = health.
 ↪drop(['age','trestbps','chol','fbs','restecg','slope','target','AgeCategory','CholCategory']
```

[81]:
```
x1_train, x1_test, y_train, y_test = train_test_split(x1, y, test_size = 0.2,␣
 ↪random_state = 20)
```

[82]:
```
lr_sm_reduced = sm.Logit(y_train, x1_train).fit()
```

```
Optimization terminated successfully.
         Current function value: 0.376149
         Iterations 7
```

[83]:
```
print(lr_sm_reduced.summary())
```

```
                           Logit Regression Results
==============================================================================
Dep. Variable:                 target   No. Observations:                  241
Model:                          Logit   Df Residuals:                      234
Method:                           MLE   Df Model:                            6
Date:                Sun, 13 Nov 2022   Pseudo R-squ.:                  0.4565
Time:                        17:34:53   Log-Likelihood:                -90.652
converged:                       True   LL-Null:                       -166.80
Covariance Type:            nonrobust   LLR p-value:                 2.536e-30
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
sex           -1.3464      0.437     -3.082      0.002      -2.203      -0.490
cp             0.6853      0.194      3.540      0.000       0.306       1.065
thalach        0.0260      0.005      5.272      0.000       0.016       0.036
exang         -1.2595      0.456     -2.764      0.006      -2.153      -0.366
oldpeak       -0.5071      0.195     -2.599      0.009      -0.890      -0.125
ca            -0.7805      0.217     -3.589      0.000      -1.207      -0.354
thal          -0.9526      0.304     -3.132      0.002      -1.549      -0.357
==============================================================================
```

[84]:
```
y_pred3=lr_sm_reduced.predict(x1_test)
```

[85]:
```
prediction1 = list(map(round, y_pred3))
```

[86]:
```
print('Actual values: ', list(y_test.values))
print('Predictions :', prediction1)
```

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

[87]:
```
print(confusion_matrix(y_test,prediction1))
```

```
[[19  4]
 [ 7 31]]
```

[88]:
```
print(classification_report(y_test,prediction1))
```

```
              precision    recall  f1-score   support

           0       0.73      0.83      0.78        23
           1       0.89      0.82      0.85        38

    accuracy                           0.82        61
   macro avg       0.81      0.82      0.81        61
weighted avg       0.83      0.82      0.82        61
```

[ ]:
```
# Prediction of CVD after using feature engineering in Logistic Regression,␣
 ↪using the StatsModel, had an accuracy score of 82%.
```

[ ]:
```

```