ML_Healthcare

August 2, 2023

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

import warnings
warnings.filterwarnings('ignore')
```

[2]:
```
# load the data

data = pd.read_excel('1645792390_cep1_dataset.xlsx')
```

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

[3]:
```
   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
```

[4]:
```
data.shape
```

[4]:
```
(303, 14)
```

Preliminary analysis: Perform preliminary data inspection and report the findings on the structure of the data, missing values, duplicates, etc.
Based on these findings, remove duplicates (if any) and treat missing values using an appropriate strategy

[5]:
```
data['target'].value_counts()
```

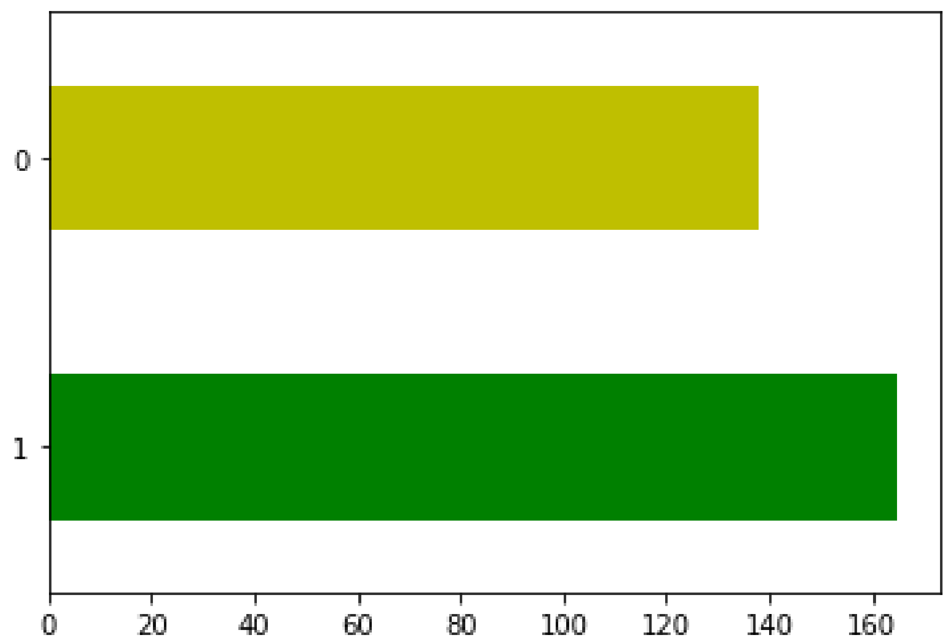
[5]:
```
1    165
0    138
Name: target, dtype: int64
```

[6]:
```
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
```

[7]:
```
data['target'].value_counts().plot(kind='barh', color=['g','y'])
```

[7]:
```
<AxesSubplot:>
```



[8]:
```
data.describe()
```

[8]:

	age	sex	cp	trestbps	chol	fbs	\
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	

	restecg	thalach	exang	oldpeak	slope	ca	\
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	
mean	0.528053	149.646865	0.326733	1.039604	1.399340	0.729373	
std	0.525860	22.905161	0.469794	1.161075	0.616226	1.022606	
min	0.000000	71.000000	0.000000	0.000000	0.000000	0.000000	
25%	0.000000	133.500000	0.000000	0.000000	1.000000	0.000000	
50%	1.000000	153.000000	0.000000	0.800000	1.000000	0.000000	
75%	1.000000	166.000000	1.000000	1.600000	2.000000	1.000000	
max	2.000000	202.000000	1.000000	6.200000	2.000000	4.000000	

	thal	target
count	303.000000	303.000000
mean	2.313531	0.544554
std	0.612277	0.498835

```
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
```

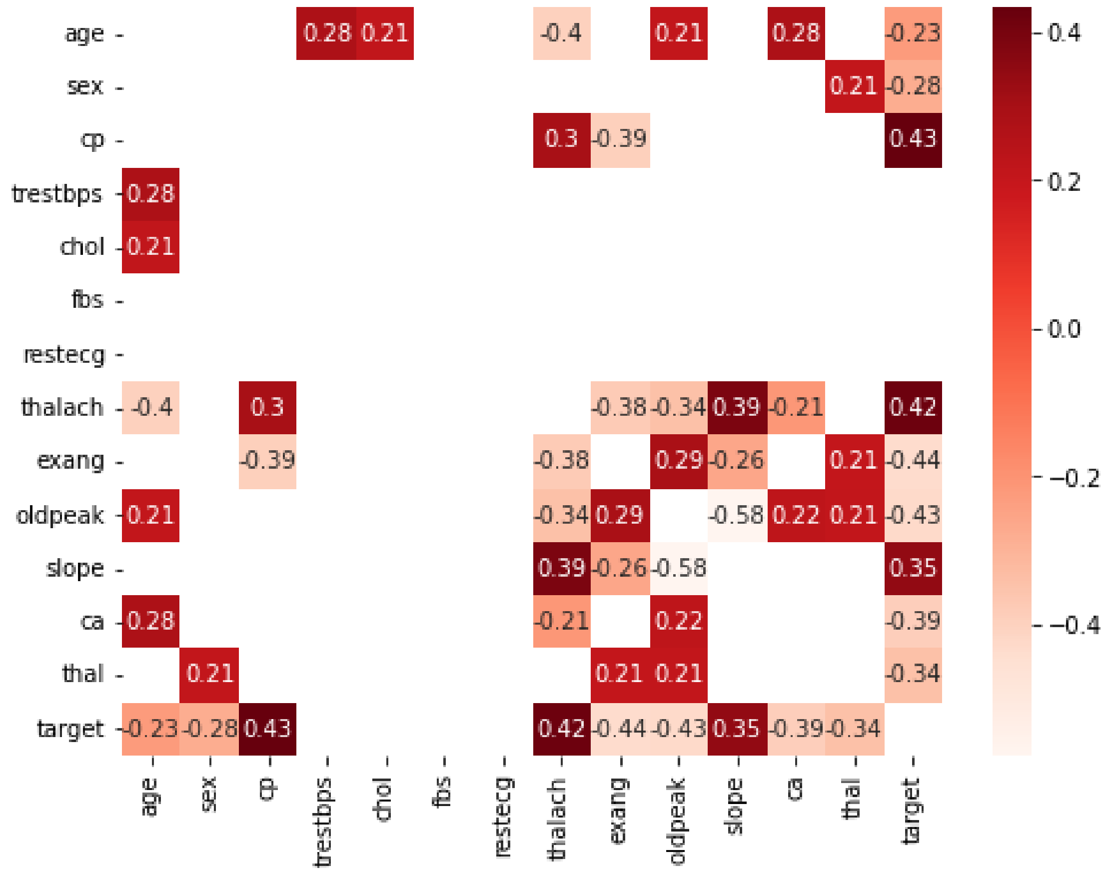
[9]:
```
# Check the correlation between the variables

print(data.corr()['target'].abs().sort_values(ascending = False))
```

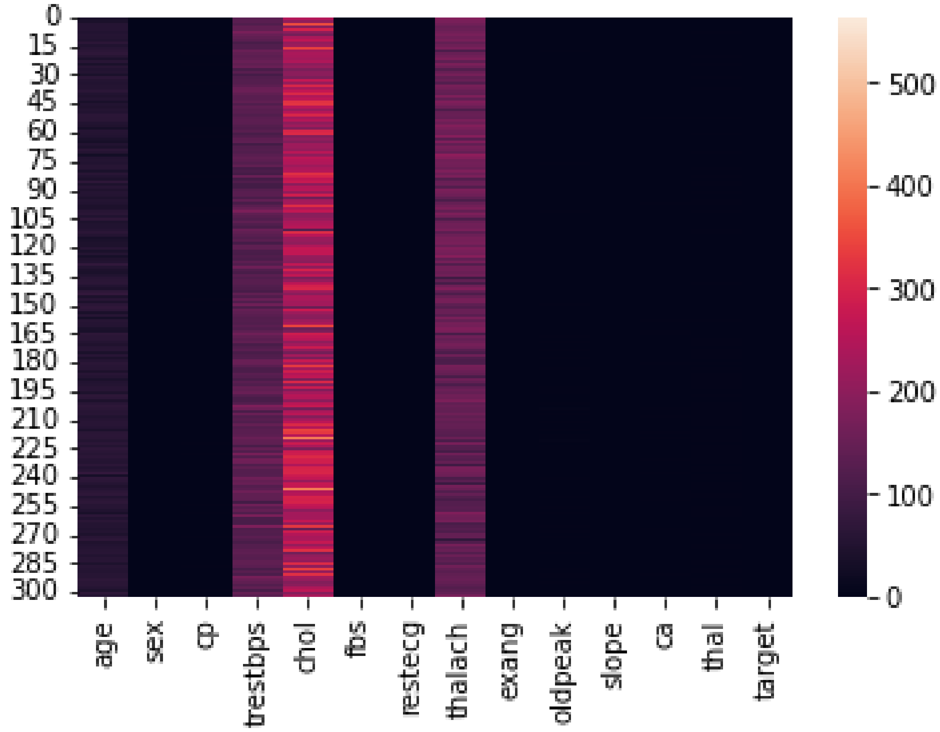
```
target      1.000000
exang       0.436757
cp          0.433798
oldpeak     0.430696
thalach     0.421741
ca          0.391724
slope       0.345877
thal        0.344029
sex         0.280937
age         0.225439
trestbps    0.144931
restecg     0.137230
chol        0.085239
fbs         0.028046
Name: target, dtype: float64
```

[10]:
```
corr=data.corr()
thresh=0.2
kot=corr[((corr>=thresh)|(corr<=-thresh))&(corr!=1)]
plt.figure(figsize=(8,6))
sns.heatmap(kot,cmap='Reds',annot=True)
```

[10]:
```
<AxesSubplot:>
```



[11]:
```
sns.heatmap(data=data)
plt.show()
```



Get a preliminary statistical summary of the data and explore the measures of central tendencies and spread of the data

Identify the data variables which are categorical and describe and explore these variables using the appropriate tools, such as count plot
Study the occurrence of CVD across the Age category
Study the composition of all patients with respect to the Sex category

Study if one can detect heart attacks based on anomalies in the resting blood pressure (trestbps) of a patient [You dont't have to do a boxplot here as it's already been said it has outliers]

[12]:
```
data.sex.value_counts()   # 1-> Male
```

[12]:
```
1    207
0     96
Name: sex, dtype: int64
```

[13]:
```
# To understand the relation betwen sex and cardiovascular disease (target)
# Creating Contingency table to compare sex with target

pd.crosstab(data.target, data.sex)
```

[13]:
```
sex      0    1
target
```

```
0          24   114
1          72    93
```

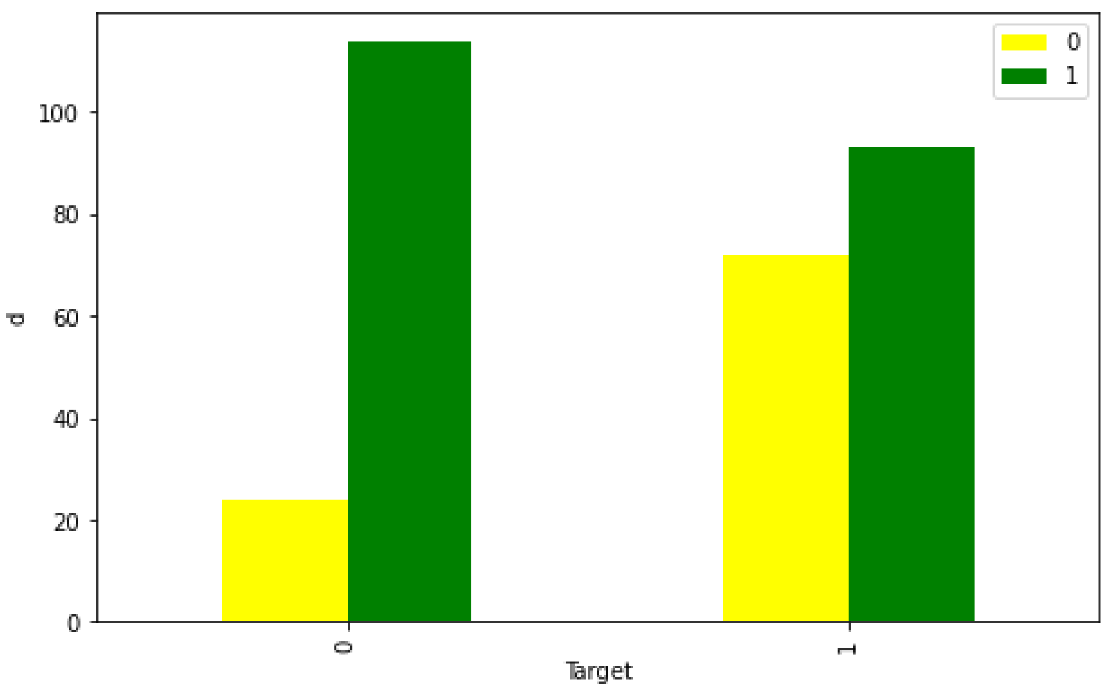
1 -> Male

93 males as compared to 72 females are detected with CVD. So males are at a higher risk of CVD.

[14]:
```
# Create plot of CVD against sex

pd.crosstab(data.target, data.sex).plot(kind = 'bar', figsize = (8,5),␣
 ↪color=['yellow','green'])
plt.xlabel('Target')
plt.ylabel('d')
plt.legend()
```

[14]:
```
<matplotlib.legend.Legend at 0x7f26804d94d0>
```



No of males suffering from cardiovascular diseases is more than no of females

[15]:
```
# Heart disease frequency vs Chest Pain

# 0 - asymptomatic
# 1 - atypical angina
# 2 - non-anginal pain
```

```
# 3 - typical angina
```

[16]:
```
# Creating a crosstab for heart disease frequency vs chest pain
pd.crosstab(data.cp, data.target)
```

[16]:

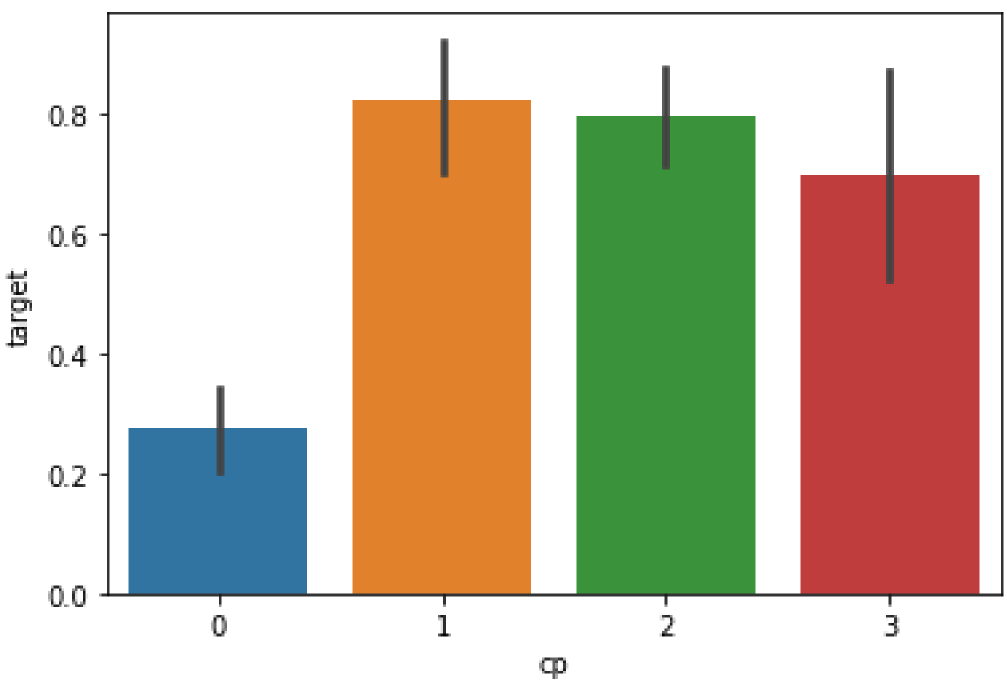
target	0	1
cp		
0	104	39
1	9	41
2	18	69
3	7	16

[17]:
```
data['cp'].unique()
```

[17]: array([3, 2, 1, 0])

[18]:
```
# FOR UPDATED VERSIONS
# sns.barplot(x='cp', y='target', data=data)

sns.barplot(data['cp'], data['target'])
plt.show()
```



[19]:
```
# 1 (atypical angina)- is impacting the most

# Asymptomatic people are least likely to suffer from heart diseases.
```

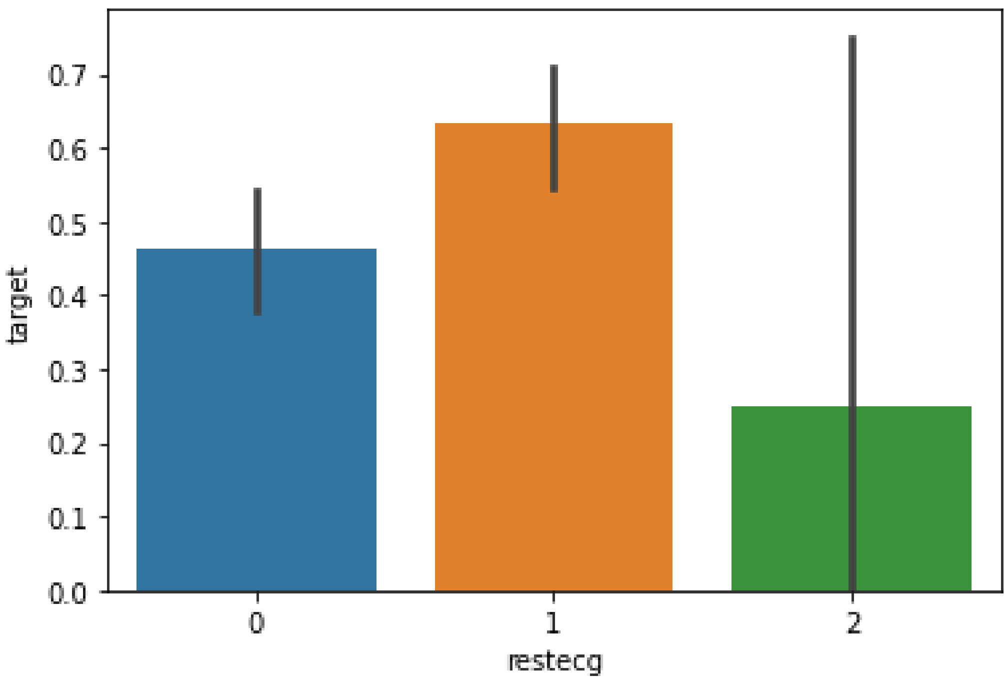
[41]:
```
# Analysing the restecg feature
# 0  = 'normal'
# 1  = 'abnormal'
# 2  = 'hyper'

data['restecg'].unique()
```

[41]:
```
array([0, 1, 2])
```

[21]:
```
# Can do countplot also

sns.barplot(data['restecg'], data['target'])
plt.show()
```



Category 1 - (abnormal) Resting electrocardiographic results show maximum occurences of a CVD ----- REPEAT FOR ALL CATEGORICAL VARIABLES -----

[22]:
```
#thalash is a categorical variable

data['thalach'].unique()
```

[22]:
```
array([150, 187, 172, 178, 163, 148, 153, 173, 162, 174, 160, 139, 171,
       144, 158, 114, 151, 161, 179, 137, 157, 123, 152, 168, 140, 188,
       125, 170, 165, 142, 180, 143, 182, 156, 115, 149, 146, 175, 186,
```

```
185, 159, 130, 190, 132, 147, 154, 202, 166, 164, 184, 122, 169,
138, 111, 145, 194, 131, 133, 155, 167, 192, 121,  96, 126, 105,
181, 116, 108, 129, 120, 112, 128, 109, 113,  99, 177, 141, 136,
 97, 127, 103, 124,  88, 195, 106,  95, 117,  71, 118, 134,  90])
```

Describe the relationship between cholesterol levels and a target variable

State what relationship exists between peak exercising and the occurrence of a heart attack

Check if thalassemia is a major cause of CVD

List how the other factors determine the occurrence of CVD

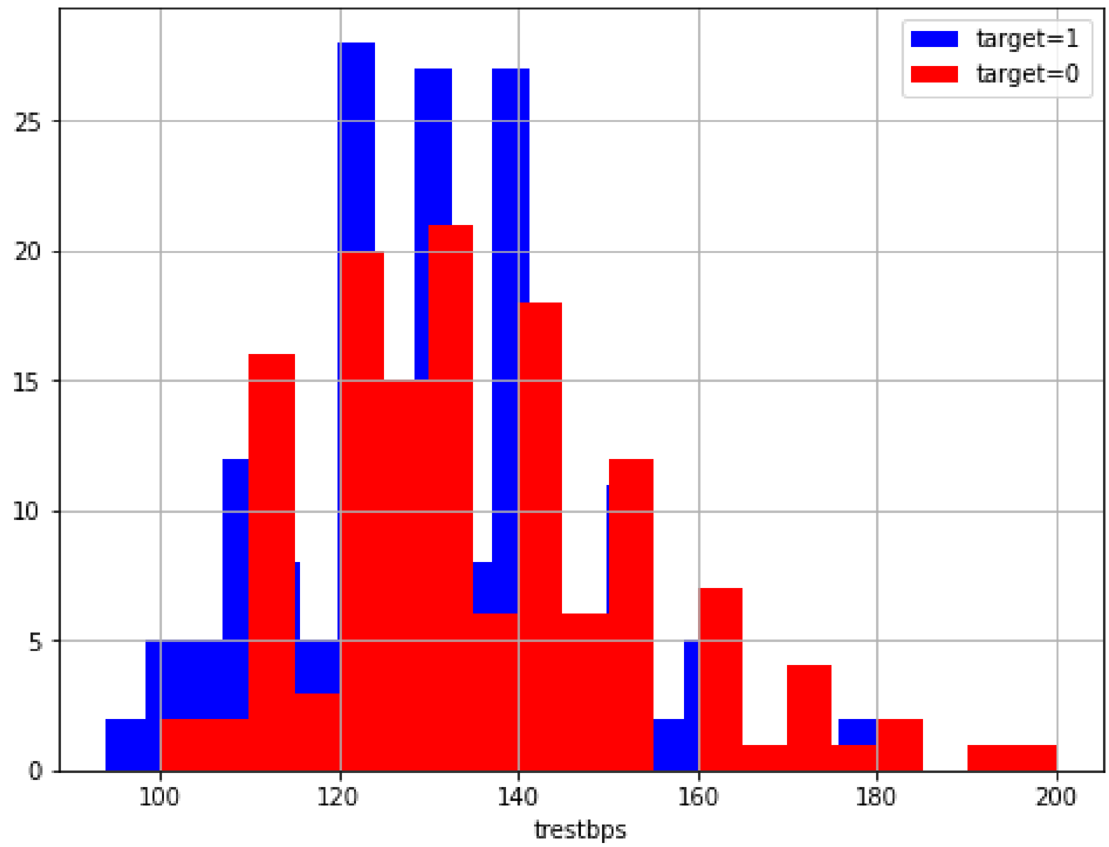
Use a pair plot to understand the relationship between all the given variables

Build a baseline model to predict the risk of a heart attack using a logistic regression and random forest and explore the results while using correlation analysis and logistic regression (leveraging standard error and p-values from statsmodels) for feature selection

[23]:
```
#e. Study if one can detect heart attacks based on anomalies in the resting
 ↪blood pressure (trestbps) of a patient
plt.figure(figsize=(8,6))
data[data['target']==1]['trestbps'].hist(color='blue',bins=20,label='target=1')
data[data['target']==0]['trestbps'].hist(color='red',bins=20,label='target=0')
plt.legend()
plt.xlabel('trestbps')
```

[23]:
```
Text(0.5, 0, 'trestbps')
```

if trestbps is between 120 to 140, higher are the chances of CVD

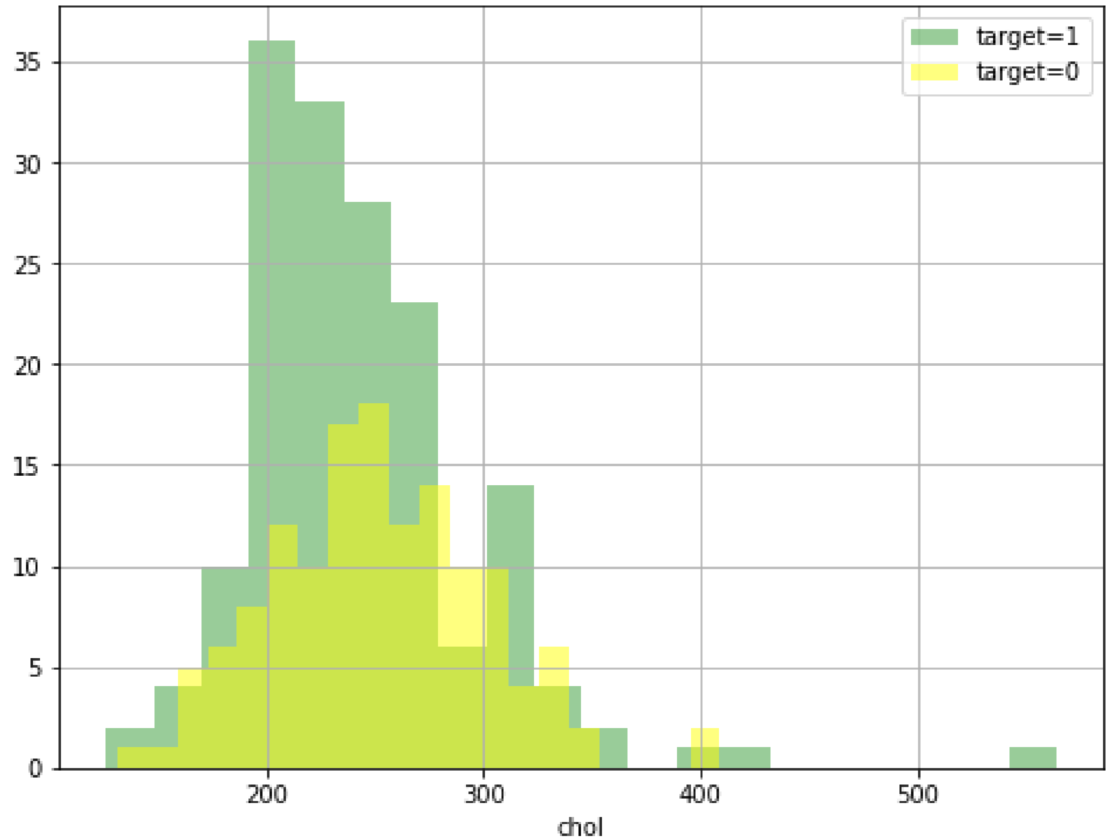


if trestbps is between 120 to 140 have higher chances of CVD

[24]:
```
#f.        Describe the relationship between cholesterol levels and a target␣
↪variable
plt.figure(figsize=(8,6))
data[data['target']==1]['chol'].hist(alpha=0.
↪4,color='green',bins=20,label='target=1')
data[data['target']==0]['chol'].hist(alpha=0.
↪5,color='yellow',bins=20,label='target=0')
plt.legend()
plt.xlabel('chol')
```

alpha = opacity

[24]:
```
Text(0.5, 0, 'chol')
```

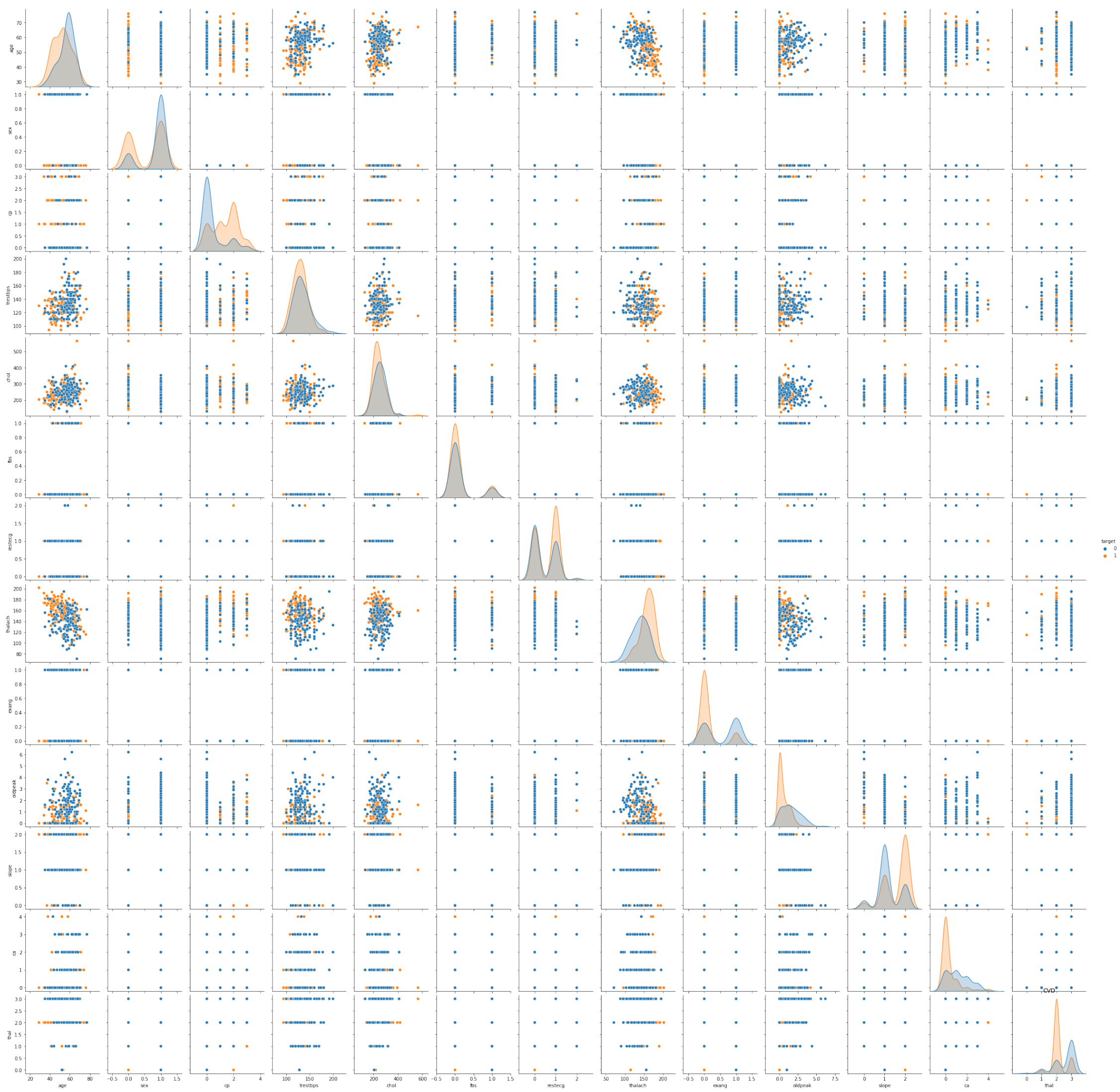


if chol is between 180 to 275 higher chances of CVD and above 300

[25]:
```
#j.         Use a pair plot to understand the relationship between all the given␣
↪variables
plt.figure(figsize=(8,6))
sns.pairplot(data,hue='target')
plt.title('CVD')
```

[25]:
```
Text(0.5, 1.0, 'CVD')
```

```
<Figure size 576x432 with 0 Axes>
```



[26]:
```
from sklearn.model_selection import train_test_split
predictors= data.drop('target',axis=1)
target = data['target']
X_train,X_test,y_train,y_test=train_test_split(predictors,target,test_size=0.
 ↪25,random_state=0)
```

[27]:
```
#building the logistic regression model
from sklearn.linear_model import LogisticRegression
lr= LogisticRegression()
lr.fit(X_train,y_train)
```

[27]: LogisticRegression()

On GitHub, the HTML representation is unable to render, please try loading this

page with nbviewer.org.

[28]:
```
y_pred = lr.predict(X_test)
```

[29]:
```
from sklearn.metrics import accuracy_score
score_lr = round(accuracy_score(y_pred,y_test)*100,2)
print("The accuracy score achieved using logistic regression is: "+␣
↪str(score_lr)+" %")
```

```
The accuracy score achieved using logistic regression is: 84.21 %
```

[30]:
```
#fitting a stats logistic regression model
import statsmodels.api as sm
log_reg=sm.Logit(y_train,X_train).fit()
```

```
Optimization terminated successfully.
         Current function value: 0.343229
         Iterations 7
```

[31]:
```
#printing the summary reports
print(log_reg.summary())
```

```
                           Logit Regression Results
==============================================================================
Dep. Variable:                 target   No. Observations:                  227
Model:                          Logit   Df Residuals:                      214
Method:                           MLE   Df Model:                           12
Date:                Wed, 02 Aug 2023   Pseudo R-squ.:                  0.5028
Time:                        07:33:51   Log-Likelihood:                -77.913
converged:                       True   LL-Null:                       -156.71
Covariance Type:            nonrobust   LLR p-value:                 1.627e-27
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
age            0.0176      0.022      0.803      0.422      -0.025       0.060
sex           -2.0263      0.531     -3.815      0.000      -3.067      -0.985
cp             0.8986      0.223      4.027      0.000       0.461       1.336
trestbps      -0.0065      0.011     -0.582      0.561      -0.028       0.015
chol          -0.0057      0.004     -1.363      0.173      -0.014       0.003
fbs           -0.6415      0.598     -1.072      0.284      -1.814       0.531
restecg        0.2590      0.402      0.645      0.519      -0.528       1.046
thalach        0.0321      0.010      3.280      0.001       0.013       0.051
exang         -0.8697      0.472     -1.843      0.065      -1.795       0.055
oldpeak       -0.5817      0.247     -2.356      0.018      -1.066      -0.098
slope          0.2910      0.428      0.680      0.497      -0.548       1.130
ca            -0.8349      0.222     -3.762      0.000      -1.270      -0.400
thal          -0.8006      0.327     -2.446      0.014      -1.442      -0.159
==============================================================================
```

[32]:
```
from sklearn.ensemble import RandomForestClassifier
clf=RandomForestClassifier(criterion='gini',
                           max_depth=7,
                           n_estimators=200,
                           #min_samples_split=10,
                           random_state=5)
```

[33]:
```
#fitting the model
clf.fit(X_train,y_train)
```

[33]:
```
RandomForestClassifier(max_depth=7, n_estimators=200, random_state=5)
```

[34]:
```
y_predt=clf.predict(X_test)
```

[35]:
```
clf.feature_importances_
```

[35]:
```
array([0.07794457, 0.05003741, 0.15391964, 0.06958547, 0.07236738,
       0.01105043, 0.0165406 , 0.11611831, 0.05549952, 0.11698825,
       0.04239796, 0.11501138, 0.1025391 ])
```

[36]:
```
data.columns
```

[36]:
```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')
```

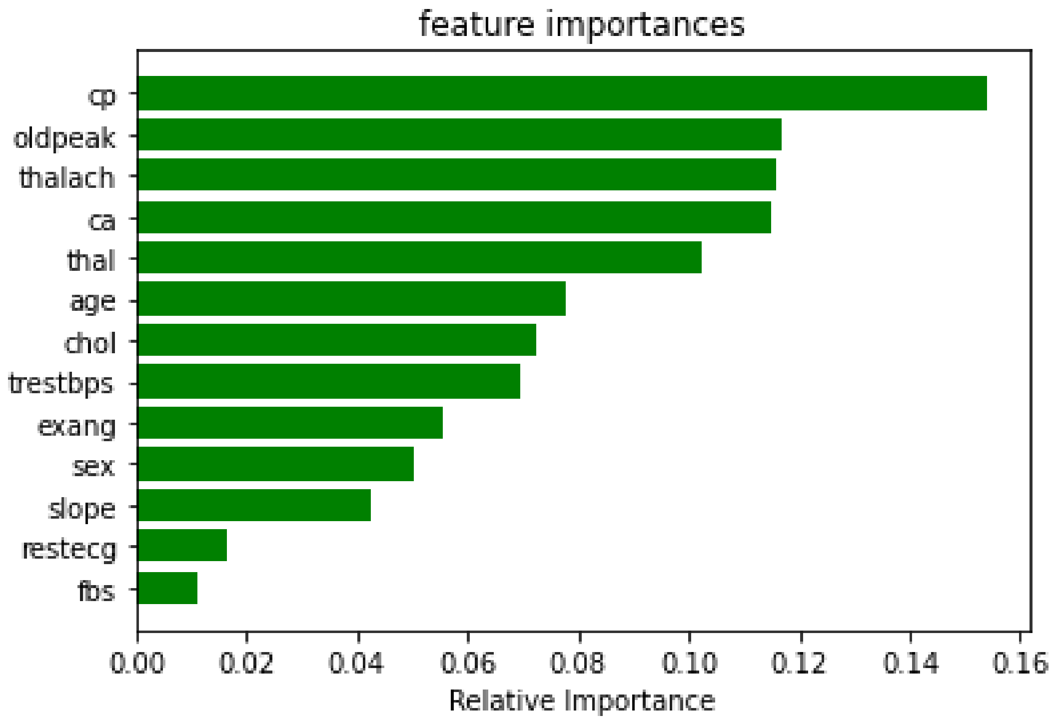
[37]:
```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test,y_predt)
```

[37]:
```
array([[25,  8],
       [ 3, 40]])
```

[38]:
```
accuracy_score(y_test,y_predt)
```

[38]:
```
0.8552631578947368
```

[39]:
```
#variable importance plot
features=data.columns
importances=clf.feature_importances_
indices=np.argsort(importances)
plt.title('feature importances')
plt.barh(range(len(indices)),importances[indices],color='g',align='center')
plt.yticks(range(len(indices)),[features[i] for i in indices])
plt.xlabel('Relative Importance')
plt.show()
```


feature importances
cp
oldpeak
thalach
ca
thal
age
chol
trestbps
exang
sex
slope
restecg
fbs
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
0.16
Relative Importance

```
[ ]:
```