

```
In [1]: import pandas as pd
```

```
In [2]: import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

```
In [3]: df=pd.read_excel("C:/Users/david/Desktop/personal/AI/course_3_PG AIML - Machine Learning/proj/healthcare/1645792390_cep1_dataset
```

```
In [4]: #1. a. Perform preliminary data inspection and report the findings on the structure of the data, missing values, duplicates, etc
```

```
In [5]: df
```

```
Out[5]:
```

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

303 rows × 14 columns

```
In [6]: df.head()
```

```
Out[6]:
```

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

```
In [7]: df.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 [8]: df.shape
```

```
Out[8]: (303, 14)
```

```
In [140... # Missing values
df.isnull().sum()
```

```
Out[140]: age      0
sex      0
cp       0
trestbps 0
chol     71
fbs      0
restecg  0
thalach  0
exang    0
oldpeak  0
slope    0
ca       0
thal     0
dtype: int64
```

```
In [10]: # no missing values
```

```
In [209... df.duplicated().sum()
```

```
Out[209]: 1
```

```
In [14]:
```

```
In [13]: # 1. b.Based on these findings, remove duplicates (if any) and treat missing values using an appropriate strategy
```

```
In [210... df2=df[df.duplicated()]
df2
```

```
Out[210]:
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
164	38	1	2	138	175	0	1	173	0	0.0	2	4	2	1

```
In [211... df=df.drop_duplicates()
df.duplicated().sum()
```

```
Out[211]: 0
```

```
In [12]: # removed duplicate value
```

In []:

In [15]: *# 2. Prepare a report about the data explaining the distribution of the disease and the related factors using the steps listed below*

In [16]: *# 2.a. Get a preliminary statistical summary of the data and explore the measures of central tendencies and spread of the data*

In [17]: `df.describe().T`

Out[17]:

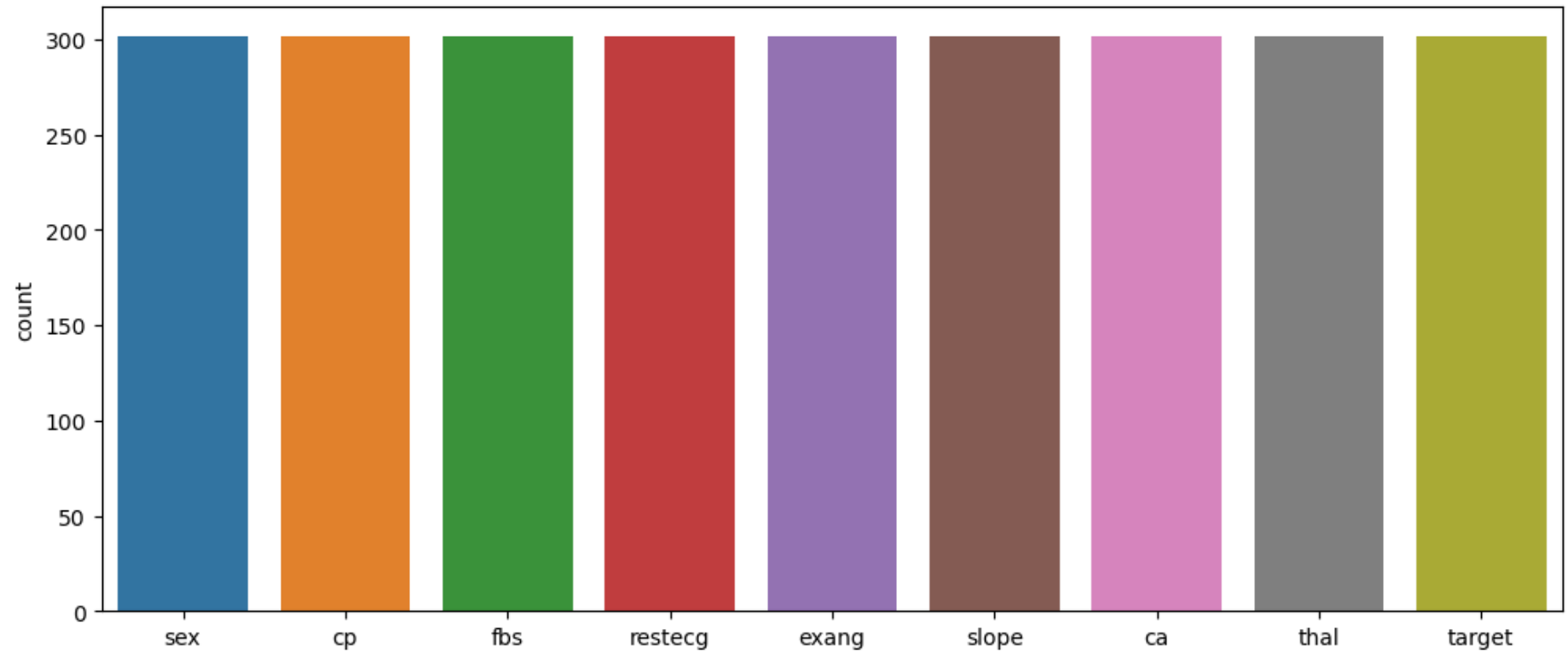
	count	mean	std	min	25%	50%	75%	max
age	303.0	54.366337	9.082101	29.0	47.5	55.0	61.0	77.0
sex	303.0	0.683168	0.466011	0.0	0.0	1.0	1.0	1.0
cp	303.0	0.966997	1.032052	0.0	0.0	1.0	2.0	3.0
trestbps	303.0	131.623762	17.538143	94.0	120.0	130.0	140.0	200.0
chol	303.0	246.264026	51.830751	126.0	211.0	240.0	274.5	564.0
fbs	303.0	0.148515	0.356198	0.0	0.0	0.0	0.0	1.0
restecg	303.0	0.528053	0.525860	0.0	0.0	1.0	1.0	2.0
thalach	303.0	149.646865	22.905161	71.0	133.5	153.0	166.0	202.0
exang	303.0	0.326733	0.469794	0.0	0.0	0.0	1.0	1.0
oldpeak	303.0	1.039604	1.161075	0.0	0.0	0.8	1.6	6.2
slope	303.0	1.399340	0.616226	0.0	1.0	1.0	2.0	2.0
ca	303.0	0.729373	1.022606	0.0	0.0	0.0	1.0	4.0
thal	303.0	2.313531	0.612277	0.0	2.0	2.0	3.0	3.0
target	303.0	0.544554	0.498835	0.0	0.0	1.0	1.0	1.0

In [290...]: *#so here categorical variables are sex , cp , fbs ,restecg , exang ,slope , ca,thal ,target*

In [19]: *# 2. b. Identify the data variables which are categorical and describe and explore these variables using the appropriate tools, such as*

In [291...]: `cat_df=df[['sex', 'cp', 'fbs', 'restecg','exang', 'slope', 'ca', 'thal', 'target']]`
`plt.figure(figsize=(12,5))`

```
sns.countplot(data=cat_df)  
plt.show()
```



In []:

In [22]: *# 2. c. Study the occurrence of CVD across the Age category*

```
In [23]: age_df=df[['age', 'target']]  
age_df.groupby(['target']).mean()
```

Out[23]:

	age
--	-----

target

0	56.601449
---	-----------

1	52.496970
---	-----------

```
In [24]: #age has more impact target 0
```

```
In [25]: # 2. d. Study the composition of all patients with respect to the Sex category
```

```
In [26]: df.groupby(['sex']).mean()
```

```
Out[26]:
```

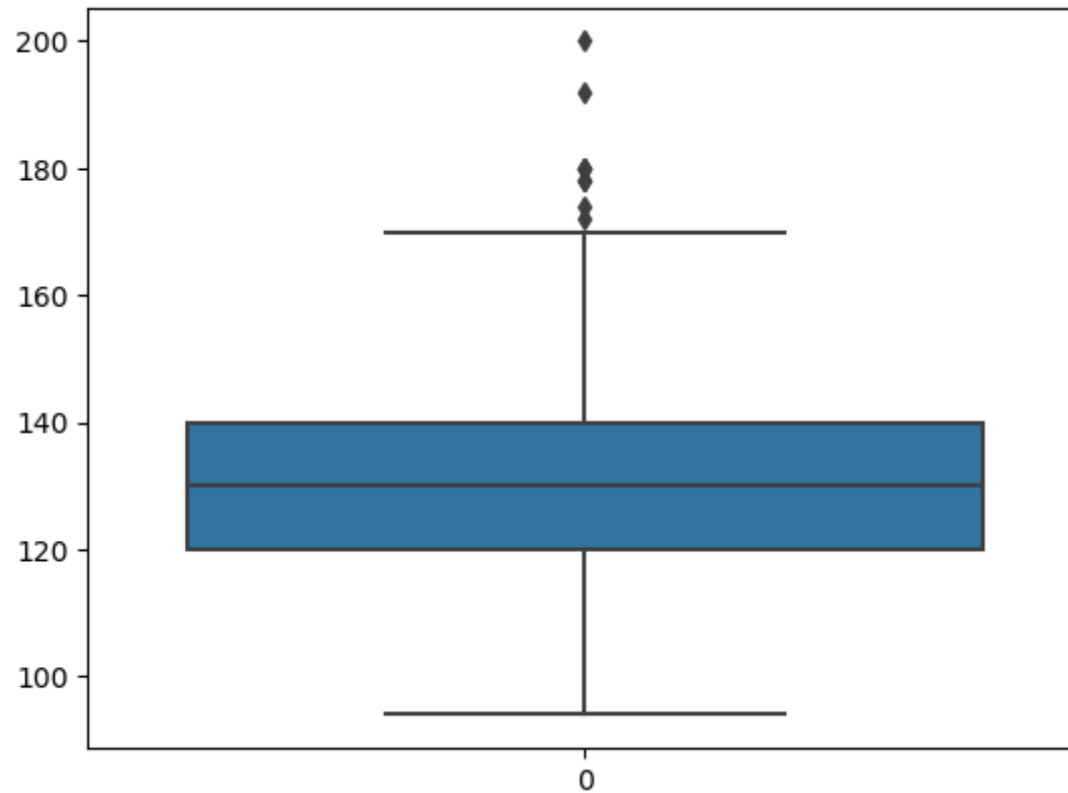
	age	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
sex													
0	55.677083	1.041667	133.083333	261.302083	0.12500	0.572917	151.125000	0.229167	0.876042	1.427083	0.552083	2.125000	0.750000
1	53.758454	0.932367	130.946860	239.289855	0.15942	0.507246	148.961353	0.371981	1.115459	1.386473	0.811594	2.400966	0.449275

```
In [27]: #sex has more impact on target 0
```

```
In [28]: #2. e. Study if one can detect heart attacks based on anomalies in the resting blood pressure (trestbps) of a patient
```

```
In [29]: sns.boxplot(df['trestbps'])
```

```
Out[29]: <Axes: >
```



```
In [30]: #here occurence of outliers are at 170
```

```
In [31]: df[df['trestbps']>170]['target'].value_counts()
```

```
Out[31]: 0    6
         1    3
         Name: target, dtype: int64
```

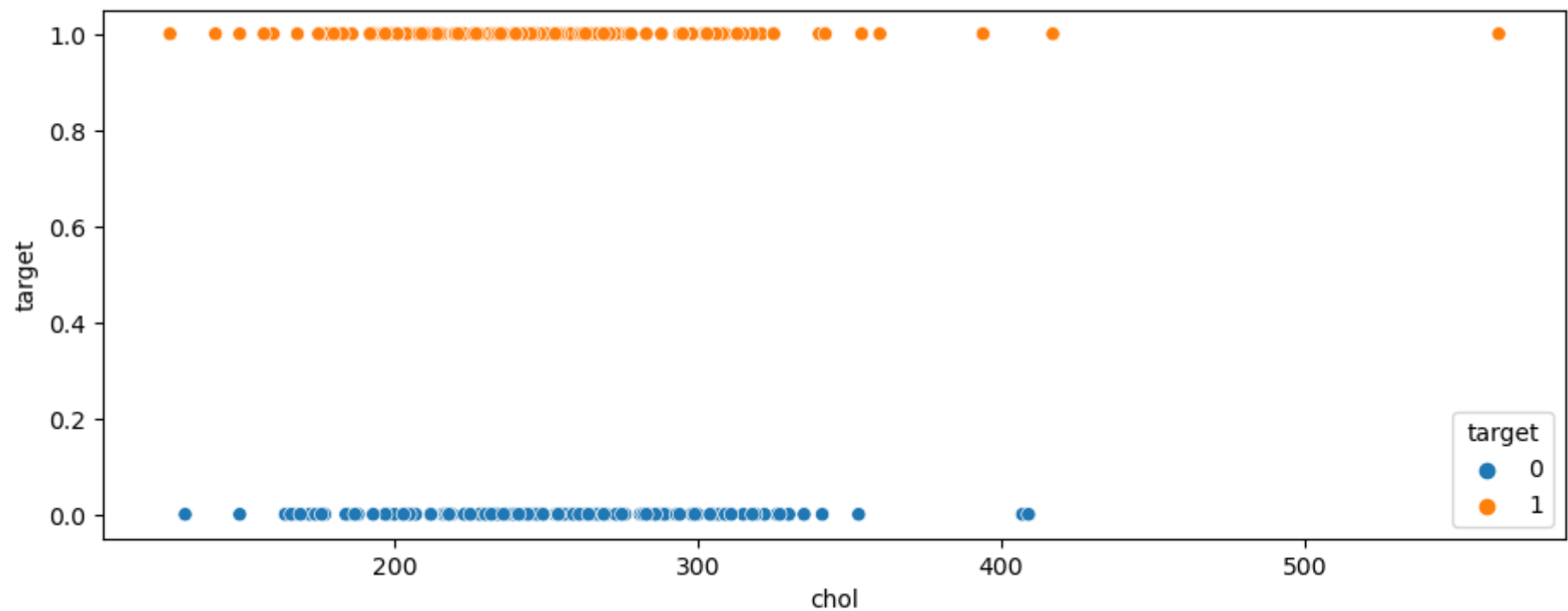
```
In [32]: df[df['trestbps']>180]['target'].value_counts()
```

```
Out[32]: 0    2
         Name: target, dtype: int64
```

```
In [33]: #it has high impact on target 0
```

```
In [34]: # 2. f.Describe the relationship between cholesterol levels and a target variable
```

```
In [35]: plt.figure(figsize=(11,4))
sns.scatterplot(data=df,x='chol',y='target',hue='target')
plt.show()
```



```
In [ ]:
```

```
In [37]: chol_df=df[['chol', 'target']]
chol_df.groupby(['target']).mean()
```

```
Out[37]:
```

	chol
--	------

target	chol
--------	------

0	251.086957
---	------------

1	242.230303
---	------------

```
In [38]: #chol has more impact on target 0
```



```
In [39]: # 2. g. State what relationship exists between peak exercising and the occurrence of a heart attack
```

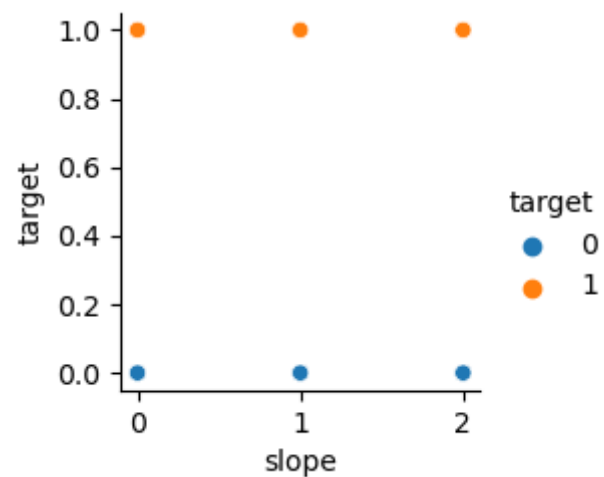
```
In [40]: slope_df=df[['slope', 'target']]
slope_df.groupby(['target']).mean()
```

```
Out[40]:
```

	slope
target	
0	1.166667
1	1.593939

```
In [41]: plt.figure(figsize=(11,4))
sns.pairplot(data=df,x_vars='slope',y_vars='target',hue='target')
plt.show()
```

<Figure size 1100x400 with 0 Axes>

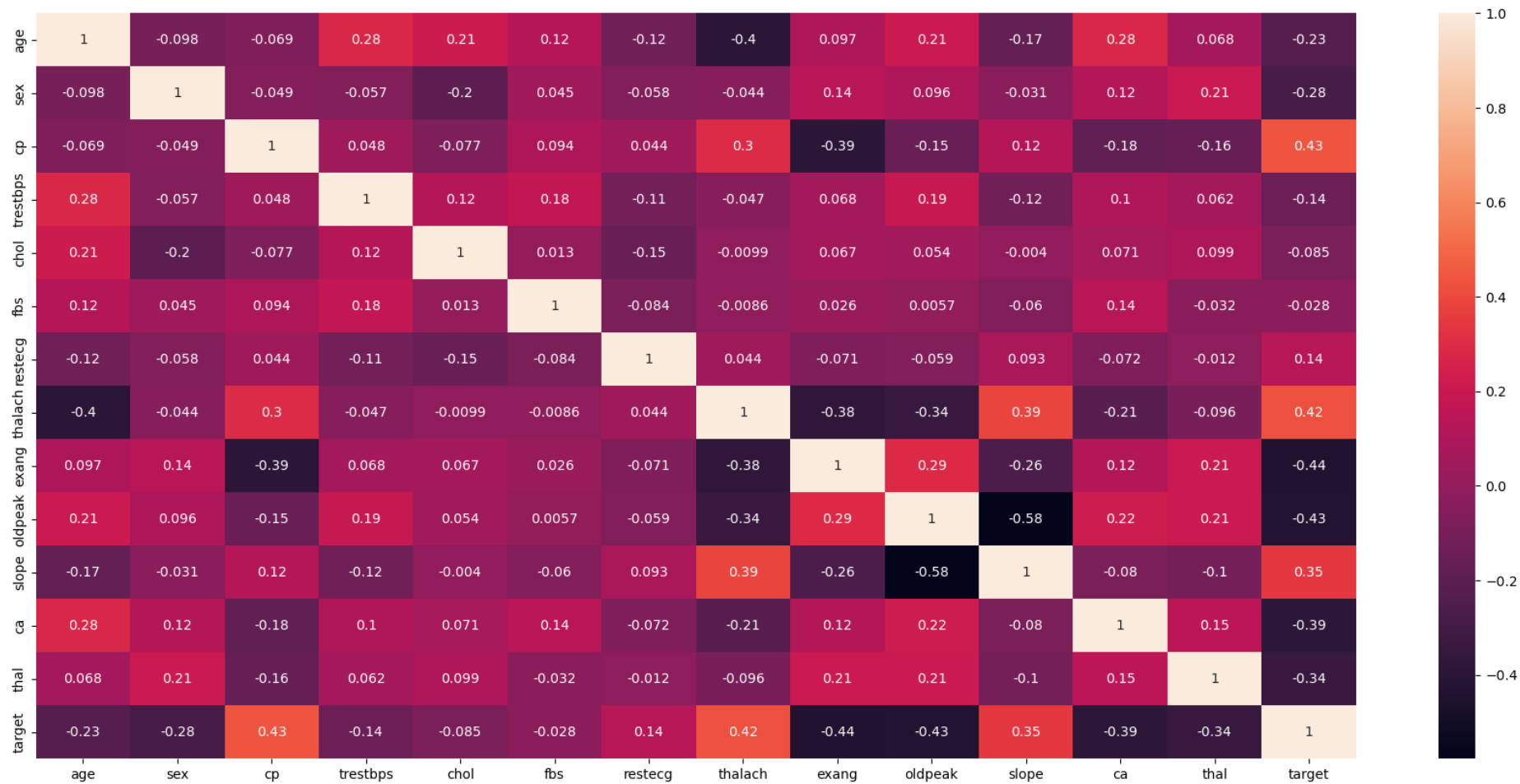


```
In [42]: # slope that us peak exercising has very less impact on both target 0 and 1
```

```
In [43]: # 2. h. Check if thalassemia is a major cause of CVD
```

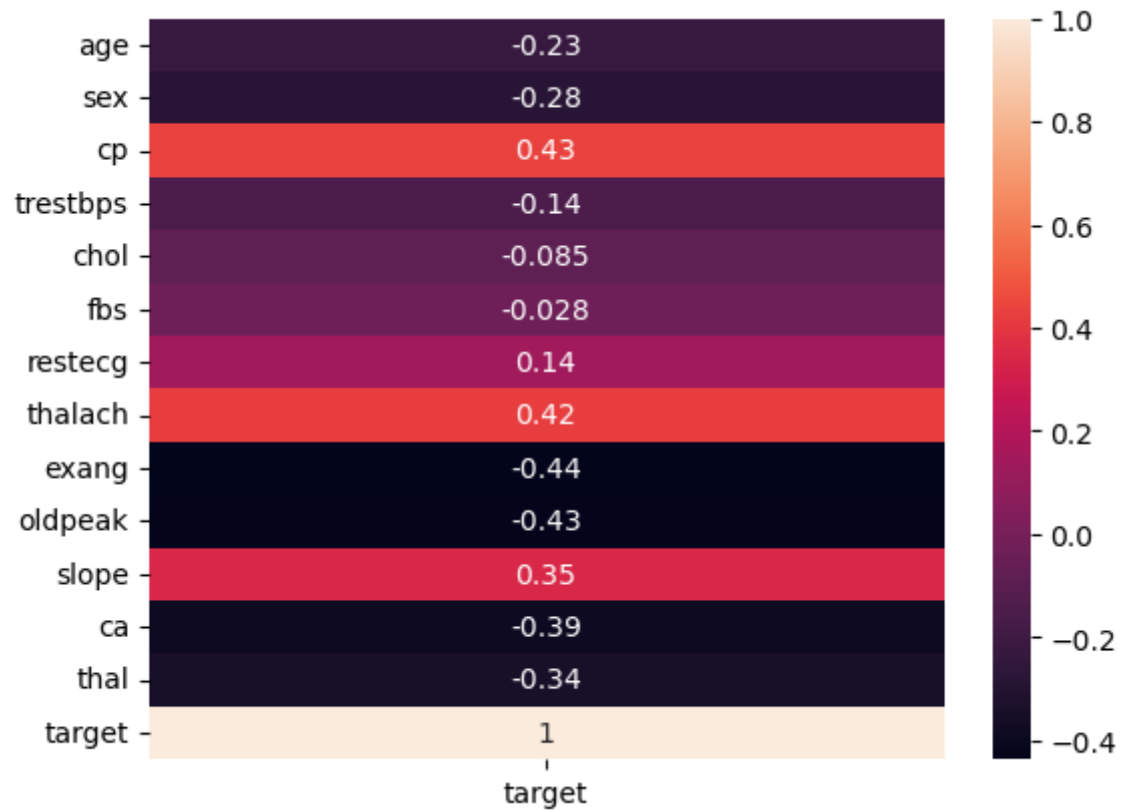
```
In [44]: plt.figure(figsize = (22,10))
sns.heatmap(df.corr(),annot=True)
```

Out[44]: <Axes: >



In [45]: sns.heatmap(df.corr()[['target']], annot = True)

Out[45]: <Axes: >

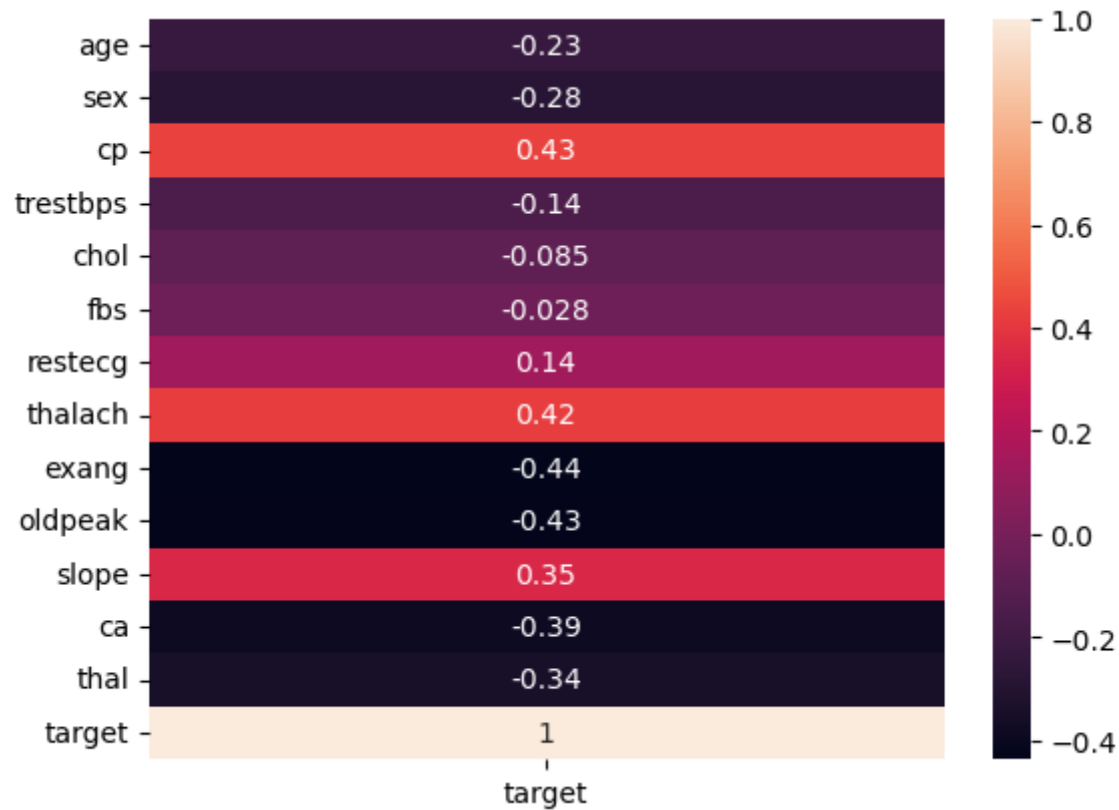


```
In [46]: # 'thal' has less impact , thalassemia is not a major cause of CVD
```

```
In [47]: # 2. i. List how the other factors determine the occurrence of CVD
```

```
In [48]: sns.heatmap(df.corr()[['target']], annot = True)
```

```
Out[48]: <Axes: >
```



```
In [49]: # features cp and thalach as major impact on target 1 and features exang and oldpeak have major impact on target 0
df.groupby(['target']).mean()
```

```
Out[49]:
```

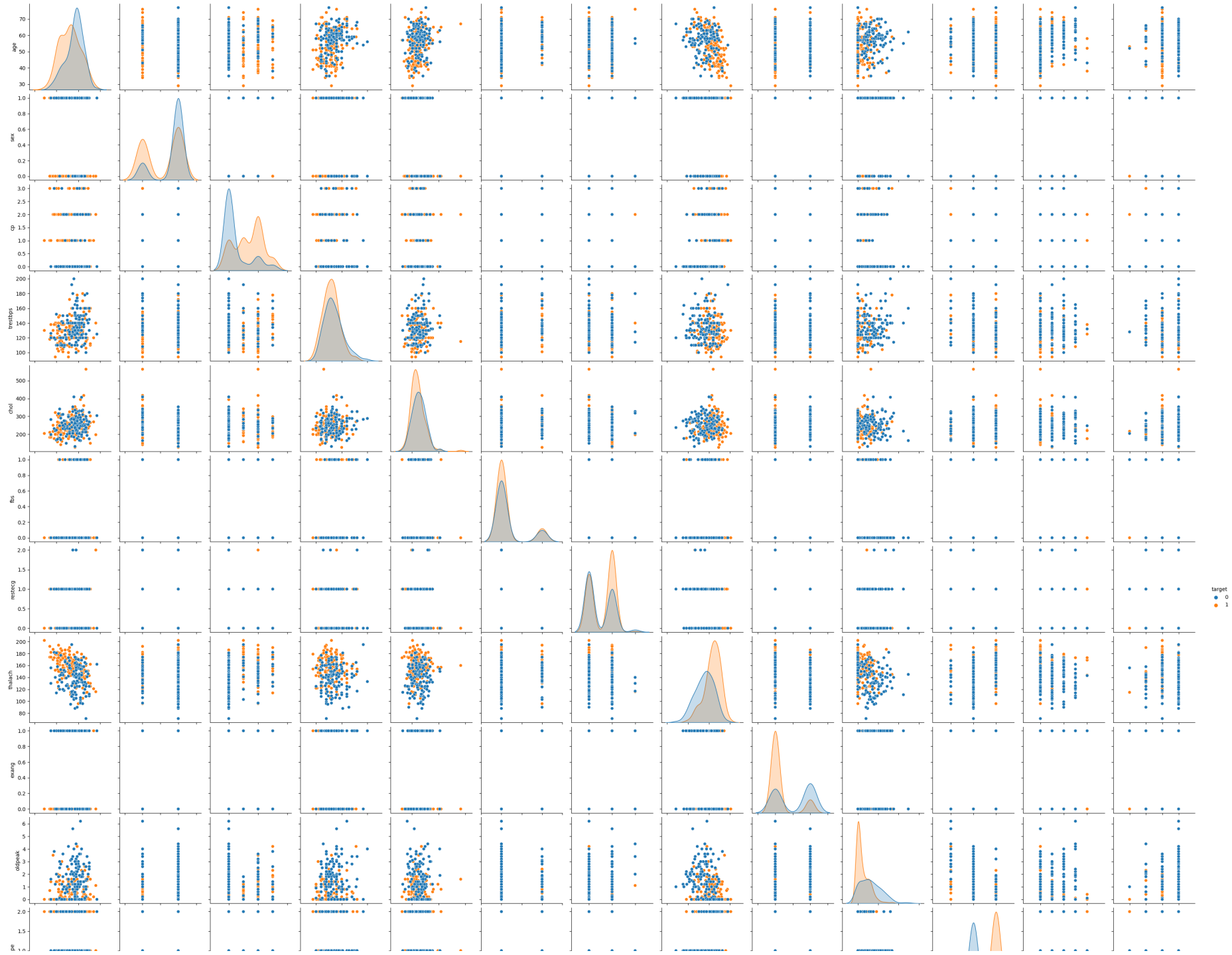
	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
target													
0	56.601449	0.826087	0.478261	134.398551	251.086957	0.159420	0.449275	139.101449	0.550725	1.585507	1.166667	1.166667	2.543478
1	52.496970	0.563636	1.375758	129.303030	242.230303	0.139394	0.593939	158.466667	0.139394	0.583030	1.593939	0.363636	2.121212

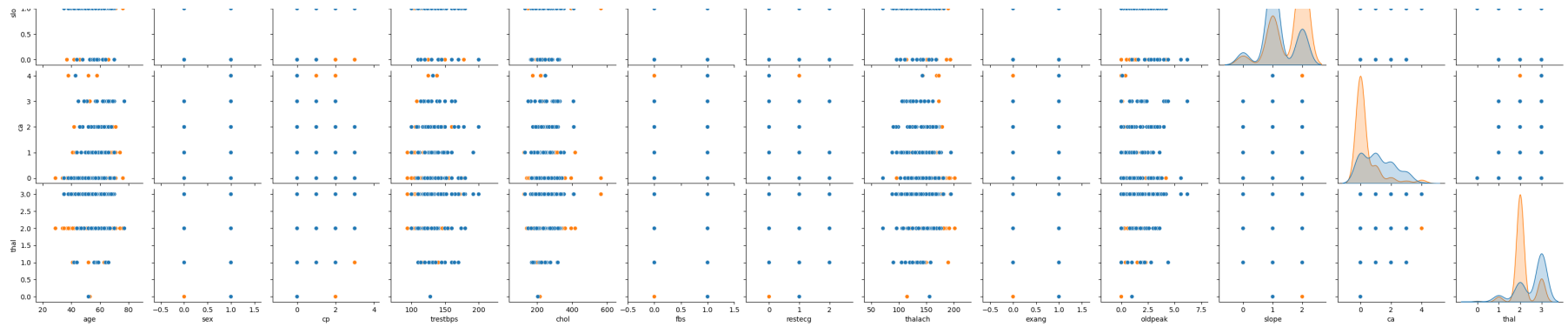
```
In [50]: # 2.j.Use a pair plot to understand the relationship between all the given variables
```

```
In [51]: plt.figure(figsize=(11,4))
sns.pairplot(data=df,hue='target')
```

```
plt.show()
```

<Figure size 1100x400 with 0 Axes>





In [285...] *# 3.Build a baseline model to predict the risk of a heart attack using a logistic regression and random forest and explore the re*

In [286...] *#Identify x and y*

```
In [287...] features = ['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
                        'exang', 'oldpeak', 'slope', 'ca', 'thal']
target = ['target']
```

```
In [288...] x = df[features]
            y = df[target]
```

```
In [267...] #splitting
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.33, random_state=42)
```

```
In [268...] import statsmodels.api as sm
```

```
In [289...] c = sm.add_constant(X_train)
model = sm.Logit(y_train, X_train).fit()
model.summary()
```

Optimization terminated successfully.
Current function value: 0.325772
Iterations 7

Out[289]:

Logit Regression Results

Dep. Variable:	target	No. Observations:	202
Model:	Logit	Df Residuals:	195
Method:	MLE	Df Model:	6
Date:	Mon, 12 Jun 2023	Pseudo R-squ.:	0.5288
Time:	21:11:03	Log-Likelihood:	-65.806
converged:	True	LL-Null:	-139.66
Covariance Type:	nonrobust	LLR p-value:	2.363e-29

	coef	std err	z	P> z	[0.025	0.975]
sex	-1.3318	0.514	-2.592	0.010	-2.339	-0.325
cp	0.8178	0.229	3.566	0.000	0.368	1.267
thalach	0.0200	0.006	3.161	0.002	0.008	0.032
exang	-1.2170	0.482	-2.525	0.012	-2.162	-0.272
slope	1.2692	0.380	3.340	0.001	0.524	2.014
ca	-1.5515	0.329	-4.716	0.000	-2.196	-0.907
thal	-1.4290	0.365	-3.919	0.000	-2.144	-0.714

```
In [270... #selecting features with p value <0.05

x=['sex','cp','thalach','exang','slope','ca','thal']
X_train=X_train[x]
X_test=X_test[x]
```

```
In [271... # model building
```

```
In [272... from sklearn.linear_model import LogisticRegression
```

```
In [273... logreg=LogisticRegression()
```



```
In [274... logreg.fit(X_train,y_train)
```

```
Out[274]: ▾ LogisticRegression  
LogisticRegression()
```

```
In [275... pred=logreg.predict(X_test)
```

```
In [276... logreg.score(X_train,y_train)
```

```
Out[276]: 0.8613861386138614
```

```
In [277... logreg.score(X_test,y_test)
```

```
Out[277]: 0.82
```

```
In [278... #testing  
from sklearn.metrics import confusion_matrix, classification_report  
confusion_matrix(y_test, pred)
```

```
Out[278]: array([[35,  8],  
               [10, 47]], dtype=int64)
```

```
In [197... print(classification_report(y_test, pred))  
  
print(classification_report())
```

	precision	recall	f1-score	support
0	0.77	0.79	0.78	42
1	0.84	0.83	0.83	58
accuracy			0.81	100
macro avg	0.80	0.81	0.81	100
weighted avg	0.81	0.81	0.81	100

```
In [260... #random forest model
```

```
In [296... from sklearn.ensemble import RandomForestClassifier
```

```
In [297... clf_rf = RandomForestClassifier()
```

```
In [298... clf_rf.fit(X_train, y_train)
```

```
Out[298]: ▼ RandomForestClassifier  
RandomForestClassifier()
```

```
In [299... clf_rf.score(X_test, y_test)
```

```
Out[299]: 0.81
```

```
In [295... clf_rf.score(X_train, y_train)
```

```
Out[295]: 0.9900990099009901
```

```
In [284... from sklearn.metrics import confusion_matrix, classification_report
```

```
In [206... predictions = clf_rf.predict(X_test)
```

```
In [207... confusion_matrix(y_test, predictions)
```

```
Out[207]: array([[32, 10],  
          [10, 48]], dtype=int64)
```

```
In [228... print(classification_report(y_test, predictions))
```

	precision	recall	f1-score	support
0	0.76	0.76	0.76	42
1	0.83	0.83	0.83	58
accuracy			0.80	100
macro avg	0.79	0.79	0.79	100
weighted avg	0.80	0.80	0.80	100

```
In [ ]:
```

