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

In [2]:
```
import warnings
warnings.filterwarnings("ignore")
```

In [3]:
```
cvd_1 = pd.read_excel('healthcare.xlsx')
```

In [4]:
```
cvd_1.head()
```

Out[4]:

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

In [5]:
```
cvd_1.shape
```

Out[5]:
```
(303, 14)
```

In [6]:
```
numeric_cvd_1 = cvd_1.select_dtypes(include=[np.number])
category_cvd_1 = cvd_1.select_dtypes(exclude=[np.number])
```

In [7]:
```
print (numeric_cvd_1.columns)
print (category_cvd_1.columns)
```

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

In [8]:
```
cvd_1.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 [9]:
```
cvd_1.isnull().sum()/cvd_1.shape[0]*100
```

Out[9]:
```
age         0.0
sex         0.0
cp          0.0
trestbps    0.0
chol        0.0
fbs         0.0
restecg     0.0
thalach     0.0
exang       0.0
oldpeak     0.0
slope       0.0
ca          0.0
thal        0.0
target      0.0
dtype: float64
```

In [10]:
```
cvd_1.duplicated()
```

Out[10]:
```
0      False
1      False
2      False
3      False
4      False
       ...  
298    False
299    False
300    False
301    False
302    False
Length: 303, dtype: bool
```

In [11]:
```
cvd_1.duplicated().sum()
```

Out[11]: 1

In [12]:
```
cvd_1.drop_duplicates(inplace = True, ignore_index = True)
cvd_1
```

Out[12]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	tar
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
297	57	0	0	140	241	0	1	123	1	0.2	1	0	3	
298	45	1	3	110	264	0	1	132	0	1.2	1	0	3	
299	68	1	0	144	193	1	1	141	0	3.4	1	2	3	
300	57	1	0	130	131	0	1	115	1	1.2	1	1	3	
301	57	0	1	130	236	0	0	174	0	0.0	1	1	2	

302 rows × 14 columns

```
Initial Cleaning
Null Values = 0
Duplicate =1
#--Deleted duplicate value
```

In [13]:
```
cvd_1.dtypes
```

Out[13]:
```
age           int64
sex           int64
cp            int64
trestbps      int64
chol          int64
fbs           int64
restecg       int64
thalach       int64
exang         int64
oldpeak     float64
slope         int64
ca            int64
thal          int64
target        int64
dtype: object
```

In [14]:
```
for col in cvd_1:
    print(col)
    print(cvd_1[col].unique())
    print('\n')
```

```
age
[63 37 41 56 57 44 52 54 48 49 64 58 50 66 43 69 59 42 61 40 71 51 65 53
 46 45 39 47 62 34 35 29 55 60 67 68 74 76 70 38 77]


sex
[1 0]


cp
[3 2 1 0]


trestbps
[145 130 120 140 172 150 110 135 160 105 125 142 155 104 138 128 108 134
 122 115 118 100 124  94 112 102 152 101 132 148 178 129 180 136 126 106
 156 170 146 117 200 165 174 192 144 123 154 114 164]


chol
[233 250 204 236 354 192 294 263 199 168 239 275 266 211 283 219 340 226
 247 234 243 302 212 175 417 197 198 177 273 213 304 232 269 360 308 245
 208 264 321 325 235 257 216 256 231 141 252 201 222 260 182 303 265 309
 186 203 183 220 209 258 227 261 221 205 240 318 298 564 277 214 248 255
 207 223 288 160 394 315 246 244 270 195 196 254 126 313 262 215 193 271
 268 267 210 295 306 178 242 180 228 149 278 253 342 157 286 229 284 224
 206 167 230 335 276 353 225 330 290 172 305 188 282 185 326 274 164 307
 249 341 407 217 174 281 289 322 299 300 293 184 409 259 200 327 237 218
 319 166 311 169 187 176 241 131]


fbs
[1 0]


restecg
[0 1 2]


thalach
[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]


exang
[0 1]


oldpeak
[2.3 3.5 1.4 0.8 0.6 0.4 1.3 0.  0.5 1.6 1.2 0.2 1.8 1.  2.6 1.5 3.  2.4
 0.1 1.9 4.2 1.1 2.  0.7 0.3 0.9 3.6 3.1 3.2 2.5 2.2 2.8 3.4 6.2 4.  5.6
 2.9 2.1 3.8 4.4]


slope
[0 2 1]
```

```
ca
[0 2 1 3 4]


thal
[1 2 3 0]


target
[1 0]
```

In [15]:
```
1 import seaborn as sns
```

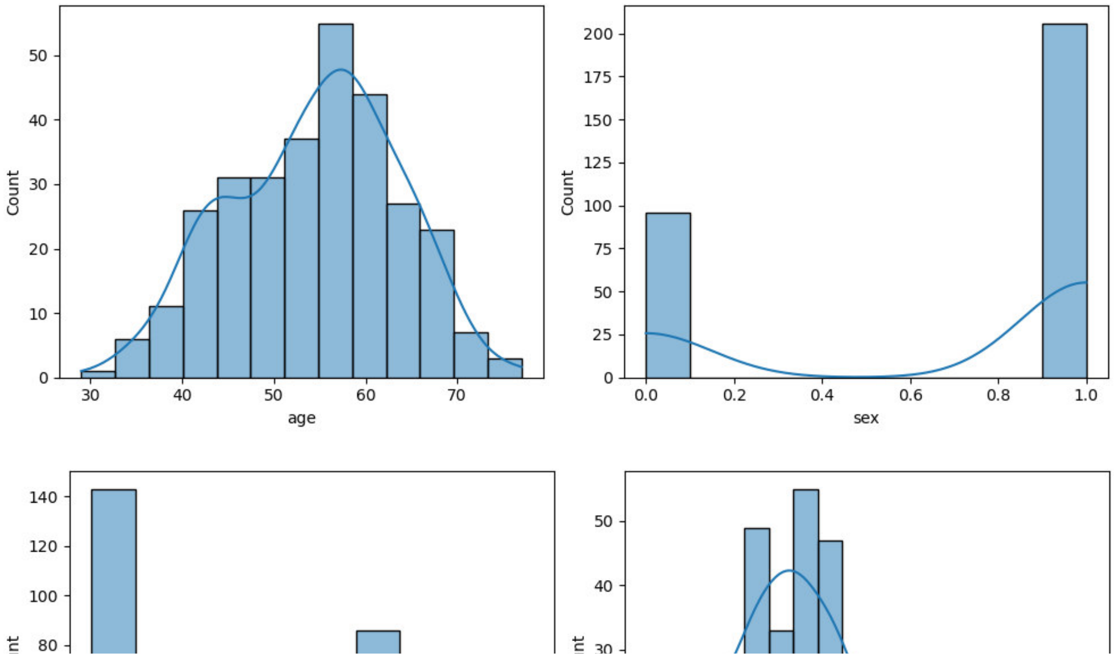
In [16]:
```
1 cvd_1.describe().T
```

Out[16]:

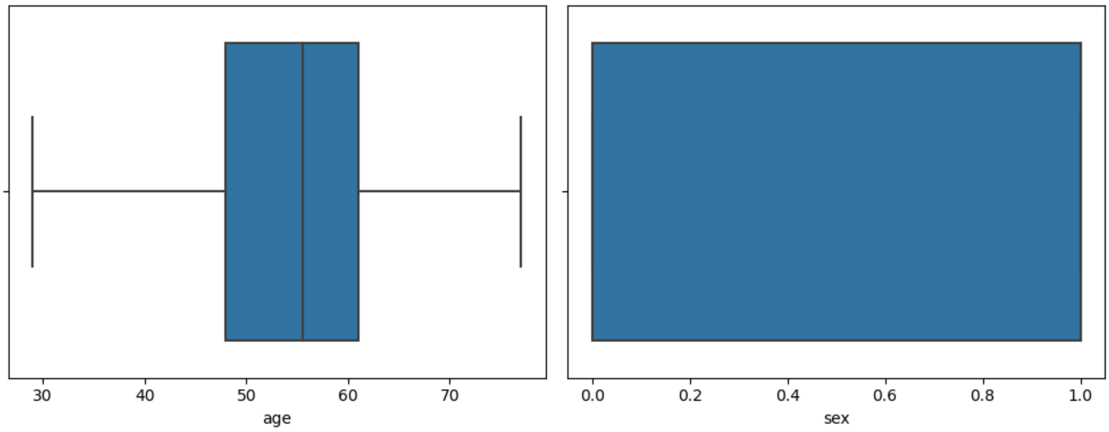
	count	mean	std	min	25%	50%	75%	max
age	302.0	54.420530	9.047970	29.0	48.00	55.5	61.00	77.0
sex	302.0	0.682119	0.466426	0.0	0.00	1.0	1.00	1.0
cp	302.0	0.963576	1.032044	0.0	0.00	1.0	2.00	3.0
trestbps	302.0	131.602649	17.563394	94.0	120.00	130.0	140.00	200.0
chol	302.0	246.500000	51.753489	126.0	211.00	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

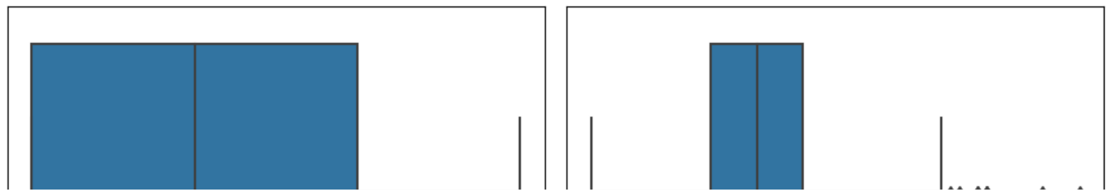
In [17]:
```
1 import matplotlib.pyplot as plt
```

In [18]:
```
num_cols = ['age' , 'sex' , 'cp' , 'trestbps' , 'chol' , 'fbs' , 'restec
len(num_cols)

for i in range(0, len(num_cols),2):
    plt.figure(figsize=(10,4))
    plt.subplot(121)
    sns.histplot(cvd_1[num_cols[i]],kde=True)
    plt.subplot(122)
    sns.histplot(cvd_1[num_cols[i+1]],kde=True)
    plt.tight_layout()
    plt.show()
```



In [19]:
```
num_cols = ['age' , 'sex' , 'cp' , 'trestbps' , 'chol' , 'fbs' , 'restec
len(num_cols)

for i in range(0, len(num_cols),2):
    plt.figure(figsize=(10,4))
    plt.subplot(121)
    sns.boxplot(cvd_1[num_cols[i]])
    plt.subplot(122)
    sns.boxplot(cvd_1[num_cols[i+1]])
    plt.tight_layout()
    plt.show()
```





In [20]:
```
# Variables with Outliers
    #-- trestbps ( upper)
    #-- chol( upper)
    #-- fbs(upper)
    #-- thalalch(lower)
    #-- oldpeak(upper)
    #-- ca(upper)
    #-- thal(lower)
```

In [21]:
```
import statsmodels.formula.api as smf
```

In [22]:
```
cvd_3 = pd.read_excel('healthcare.xlsx')
```

In [23]:
```
X1 = cvd_3.drop("target", axis=1)
y1 = cvd_3["target"]
```

In [24]:
```
model = smf.logit("target ~  age + sex + cp + trestbps + chol + fbs + re
```

```
Optimization terminated successfully.
         Current function value: 0.348904
         Iterations 7
```

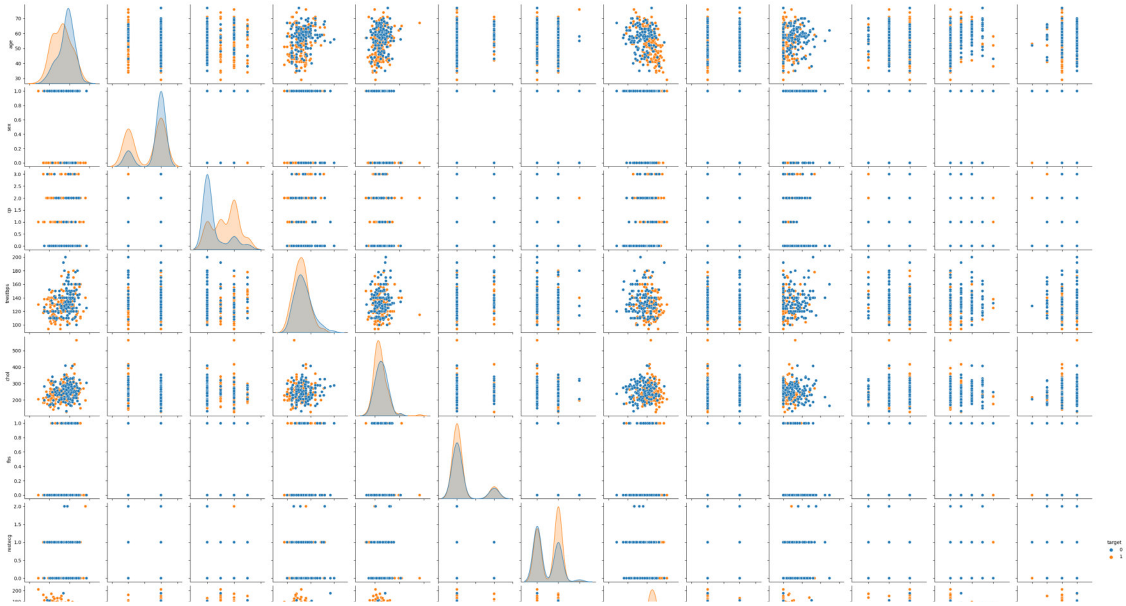
In [25]:
```
print(model.summary())
```

```
                           Logit Regression Results                           
==============================================================================
===
Dep. Variable:                 target   No. Observations:                     
303
Model:                          Logit   Df Residuals:                         
289
Method:                           MLE   Df Model:                             
13
Date:                Mon, 01 May 2023   Pseudo R-squ.:                     0.4
937
Time:                        14:03:27   Log-Likelihood:                   -10
5.72
converged:                       True   LL-Null:                          -20
8.82
Covariance Type:            nonrobust   LLR p-value:                    7.262e
-37
==============================================================================
===
                 coef    std err          z      P>|z|      [0.025      0.9
75]
------------------------------------------------------------------------------
---
Intercept      3.4505      2.571      1.342      0.180      -1.590       8.
490
age           -0.0049      0.023     -0.212      0.832      -0.050       0.
041
sex           -1.7582      0.469     -3.751      0.000      -2.677      -0.
839
cp             0.8599      0.185      4.638      0.000       0.496       1.
223
trestbps      -0.0195      0.010     -1.884      0.060      -0.040       0.
001
chol          -0.0046      0.004     -1.224      0.221      -0.012       0.
003
fbs            0.0349      0.529      0.066      0.947      -1.003       1.
073
restecg        0.4663      0.348      1.339      0.181      -0.216       1.
149
thalach        0.0232      0.010      2.219      0.026       0.003       0.
044
exang         -0.9800      0.410     -2.391      0.017      -1.783      -0.
177
oldpeak       -0.5403      0.214     -2.526      0.012      -0.959      -0.
121
slope          0.5793      0.350      1.656      0.098      -0.106       1.
265
ca            -0.7733      0.191     -4.051      0.000      -1.147      -0.
399
thal          -0.9004      0.290     -3.104      0.002      -1.469      -0.
332
==============================================================================
===
```

In [26]:
```
# Setting p value alpha as 0.05, the following are the insignificant var
# 'age', 'trestbps', 'chol', 'fbs', 'restecg', 'slope'
```

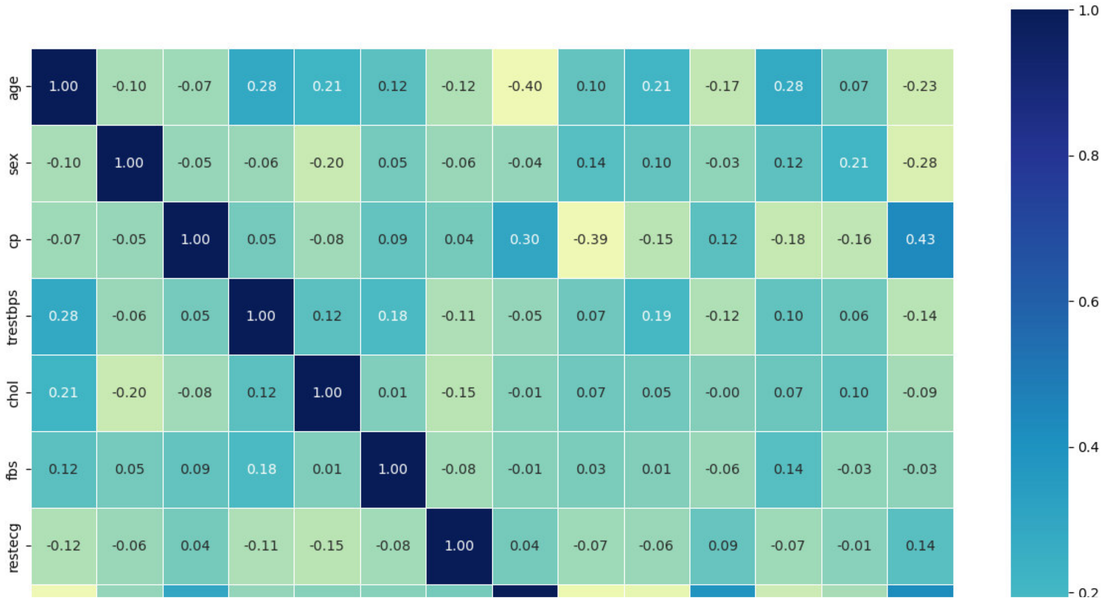
In [27]:
```
sns.pairplot(cvd_3, hue="target")
```

Out[27]: <seaborn.axisgrid.PairGrid at 0x17d1afd5ca0>



In [29]:
```
corr_matrix = cvd_3.corr()
fig, ax = plt.subplots(figsize=(15, 15))
ax = sns.heatmap(corr_matrix,
                 annot=True,
                 linewidths=0.5,
                 fmt=".2f",
                 cmap="YlGnBu");
bottom, top = ax.get_ylim()
ax.set_ylim(bottom + 0.5, top - 0.5)
```

Out[29]: (14.5, -0.5)



In [30]:
```
# Variables in relation to the other variables have no significant corre
# Correlation values are all <0.5
```

c.Study the occurrence of CVD across the Age category

In [31]:
```
min_value = cvd_1['age'].min()
max_value = cvd_1['age'].max()
print(min_value)
print(max_value)
```

```
29
77
```

In [32]:
```
max_value - min_value
```

Out[32]: 48

In [33]:
```
# Age Category Binnings
    #29-37-- Early_Late Thirties
    #38-50-- Early_ Late Forties
    #51-61-- Fifties_Early Sixties
    #62-77-- Early Sixties_Late Seventies
```

In [34]:
```
cvd_1['age_bins'] = pd.cut(cvd_1.age, [28, 37, 50, 61, 77],
                           labels = ['Early_Late Thirties', 'Early_ Late
cvd_1[['age', 'age_bins']][:49]
```

Out[34]:

	age	age_bins
0	63	Early Sixties_Late Seventies
1	37	Early_Late Thirties
2	41	Early_ Late Forties
3	56	Fifties_Early Sixties
4	57	Fifties_Early Sixties
5	57	Fifties_Early Sixties
6	56	Fifties_Early Sixties
7	44	Early_ Late Forties
8	52	Fifties_Early Sixties
9	57	Fifties_Early Sixties
10	54	Fifties_Early Sixties
11	48	Early_ Late Forties

In [35]:
```
cvd_1.head(2)
```

Out[35]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	targe
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	

In [36]:
```
# dropped Column = age
cvd_1.drop('age', axis = 1, inplace = False)
```

Out[36]:

	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	0	0	120	354	0	1	163	1	0.6	2	0	2	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...
297	0	0	140	241	0	1	123	1	0.2	1	0	3	0
298	1	3	110	264	0	1	132	0	1.2	1	0	3	0
299	1	0	144	193	1	1	141	0	3.4	1	2	3	0
300	1	0	130	131	0	1	115	1	1.2	1	1	3	0
301	0	1	130	236	0	0	174	0	0.0	1	1	2	0

302 rows × 14 columns

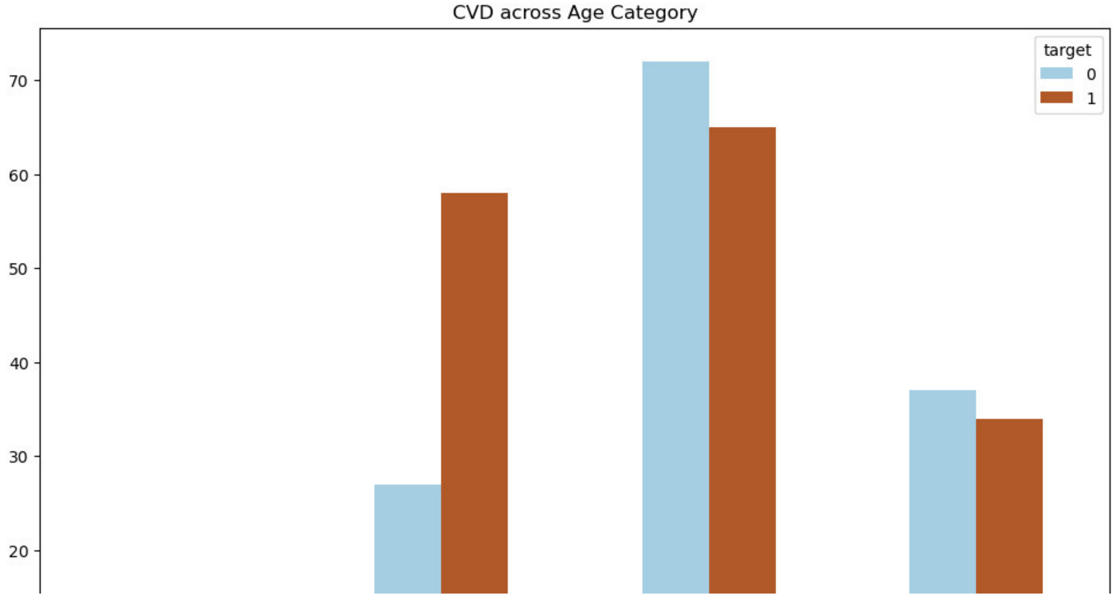
In [37]:
```
tbl = pd.crosstab(columns = cvd_1.target, index = cvd_1.age_bins)
tbl
```

Out[37]:

target	0	1
age_bins		
Early_Late Thirties	2	7
Early_ Late Forties	27	58
Fifties_Early Sixties	72	65
Early Sixties_Late Seventies	37	34

In [38]:
```
tbl.plot(kind='bar', figsize=(12,8), stacked=False, colormap="Paired")
plt.xticks(rotation=90)
plt.title('CVD across Age Category')
```

Out[38]: Text(0.5, 1.0, 'CVD across Age Category')




```
Interpretation:
1, Highest number of samples were taken from age group 51-61--Fifties_Early Sixties, followed by 38-50-- Early_ Late Forties
2. The highest number of cadiovascular diseases were found from the age group 51-61--Fifties_Early Sixties:
a but from the total number of samples of this age group, the patients with no cvd is slightly greater than the number of patients with cvd
3. The least number of cvd were found from age group 29-37--Early_Late Thirties
4. From the total number of samples from age group 38-50-- Early_ Late Forties, there is a significant gap between patients with no CVD and with CVD, wherein patients with CVD is arounf 50% higher than no CVD
```

d.Study the composition of all patients with respect to the Sex category

In [39]:
```
tbl2 = pd.crosstab(columns = cvd_1.target, index = cvd_1.sex)
tbl2
```

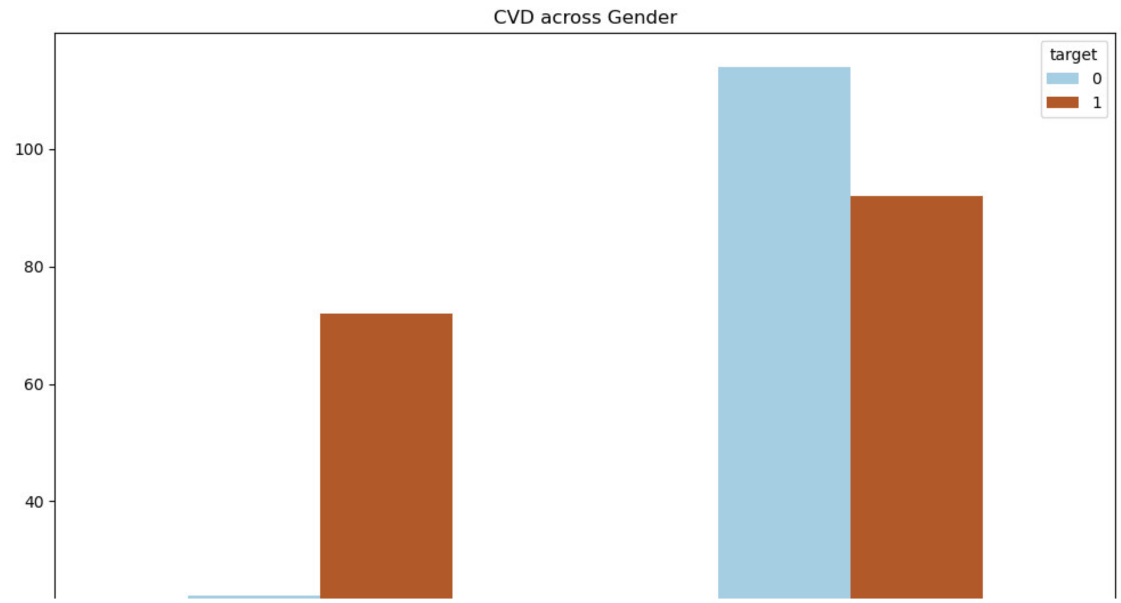
Out[39]:

target	0	1
sex		
0	24	72
1	114	92

```
# Possible class imbalance
Female samples: 96
Male samples: 206
```

In [40]:
```
tbl2.plot(kind='bar', figsize=(12,8), stacked=False, colormap="Paired")
plt.xticks([0, 1], ['Female', 'Male'])
plt.xticks(rotation=45)
plt.title('CVD across Gender')
```

Out[40]: Text(0.5, 1.0, 'CVD across Gender')




```
Interpretation:
1, There were more male patients studied than female
2. From the total number of female samples, there is a significant gap between patients with no CVD and with CVD, around 75% have CVD
```

In [41]:
```
cvd_temp= pd.read_excel('healthcare.xlsx')
```

In [42]:
```
cvd_temp['age_bins'] = pd.cut(cvd_temp.age, [28, 37, 50, 61, 77],
                              labels = ['Early_Late Thirties', 'Early_ Late
cvd_temp[['age', 'age_bins']][:49]
```

Out[42]:

	age	age_bins
0	63	Early Sixties_Late Seventies
1	37	Early_Late Thirties
2	41	Early_ Late Forties
3	56	Fifties_Early Sixties
4	57	Fifties_Early Sixties
5	57	Fifties_Early Sixties
6	56	Fifties_Early Sixties
7	44	Early_ Late Forties
8	52	Fifties_Early Sixties
9	57	Fifties_Early Sixties
10	54	Fifties_Early Sixties
11	48	Early_ Late Forties

In [43]:
```
tbl_temp = pd.crosstab(index = [cvd_temp.sex, cvd_temp.age_bins], columı
tbl_temp
```

Out[43]:

	target	0	1
sex	**age_bins**		
0	**Early_Late Thirties**	0	3
	Early_ Late Forties	1	23
	Fifties_Early Sixties	13	24
	Early Sixties_Late Seventies	10	22
1	**Early_Late Thirties**	2	4
	Early_ Late Forties	26	36
	Fifties_Early Sixties	59	41
	Early Sixties_Late Seventies	27	12

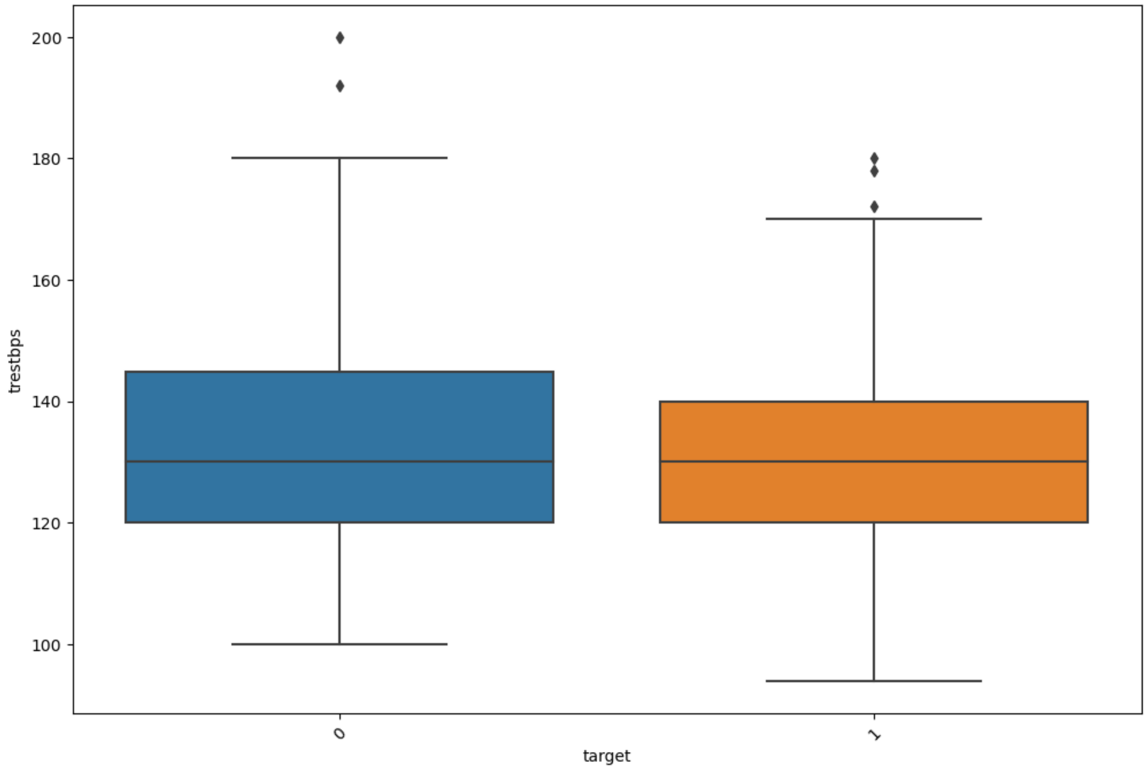
In [44]:
```
import seaborn as sns
tbl_temp.plot.bar(stacked=True, rot=0, figsize=(10, 6))
plt.xticks(rotation=45)
```

Out[44]:
```
(array([0, 1, 2, 3, 4, 5, 6, 7]),
 [Text(0, 0, '(0, Early_Late Thirties)'),
  Text(1, 0, '(0, Early_ Late Forties)'),
  Text(2, 0, '(0, Fifties_Early Sixties)'),
  Text(3, 0, '(0, Early Sixties_Late Seventies)'),
  Text(4, 0, '(1, Early_Late Thirties)'),
  Text(5, 0, '(1, Early_ Late Forties)'),
  Text(6, 0, '(1, Fifties_Early Sixties)'),
  Text(7, 0, '(1, Early Sixties_Late Seventies)')])
```


```
Interpretation:
Normal distrubiotn of positive and negative heart attacks across genders
```

e.Study if one can detect heart attacks based on anomalies in the resting trestbps blood pressure () of a patient

In [45]:
```
plt.figure(figsize=(12,8))
plt.xticks(rotation=45) # the x-axis was not readable so the text is rot
sns.boxplot(x='target',y='trestbps' , data=cvd_1);
```



In [46]:
```
min_value = cvd_1['trestbps'].min()
Q1 = cvd_1['trestbps'].quantile(0.25)
median_value = cvd_1['trestbps'].median()
Q3 = cvd_1['trestbps'].quantile(0.75)
max_value = cvd_1['trestbps'].max()
IQR = Q3 - Q1
lower_limit = Q1 - 1.5 * IQR
upper_limit = Q3 + 1.5 * IQR


print ("min_value    :", min_value)
print ("Q1           :", Q1)
print ("median_value :", median_value)
print ("Q3           :", Q3)
print ("max_value    :", max_value)
print (IQR)
print (lower_limit)
print (upper_limit)

```

```
min_value    : 94
Q1           : 120.0
median_value : 130.0
Q3           : 140.0
max_value    : 200
20.0
90.0
170.0
```

In [47]:
```
cvd_1.loc[cvd_1.trestbps <= lower_limit]
```

Out[47]:

age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target

In [48]:
```
cvd_1.loc[cvd_1.trestbps >= upper_limit]
```

Out[48]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	tar
8	52	1	2	172	199	1	1	162	0	0.5	2	0	3	
101	59	1	3	178	270	0	0	145	0	4.2	0	0	3	
110	64	0	0	180	325	0	1	154	1	0.0	2	0	2	
152	64	1	3	170	227	0	0	155	0	0.6	1	0	3	
194	59	1	0	170	326	0	0	140	1	3.4	0	0	3	
202	68	1	2	180	274	1	0	150	1	1.6	1	0	3	
222	56	0	0	200	288	1	0	133	1	4.0	0	2	3	
227	59	1	3	170	288	0	0	159	0	0.2	1	0	3	
240	59	0	0	174	249	0	1	143	1	0.0	1	0	2	
247	54	1	1	192	283	0	0	195	0	0.0	2	1	3	
259	66	0	0	178	228	1	1	165	1	1.0	1	2	3	
265	55	0	0	180	327	0	2	117	1	3.4	1	0	2	
291	58	0	0	170	225	1	0	146	1	2.8	1	2	1	

In [49]:
```
len(cvd_1.loc[cvd_1.trestbps >= upper_limit])
```

Out[49]: 13

In [50]:
```
trestbps_out = cvd_1.loc[cvd_1.trestbps >= upper_limit]
trestbps_out
```

Out[50]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	tar
8	52	1	2	172	199	1	1	162	0	0.5	2	0	3	
101	59	1	3	178	270	0	0	145	0	4.2	0	0	3	
110	64	0	0	180	325	0	1	154	1	0.0	2	0	2	
152	64	1	3	170	227	0	0	155	0	0.6	1	0	3	
194	59	1	0	170	326	0	0	140	1	3.4	0	0	3	
202	68	1	2	180	274	1	0	150	1	1.6	1	0	3	
222	56	0	0	200	288	1	0	133	1	4.0	0	2	3	
227	59	1	3	170	288	0	0	159	0	0.2	1	0	3	
240	59	0	0	174	249	0	1	143	1	0.0	1	0	2	
247	54	1	1	192	283	0	0	195	0	0.0	2	1	3	
259	66	0	0	178	228	1	1	165	1	1.0	1	2	3	
265	55	0	0	180	327	0	2	117	1	3.4	1	0	2	
291	58	0	0	170	225	1	0	146	1	2.8	1	2	1	

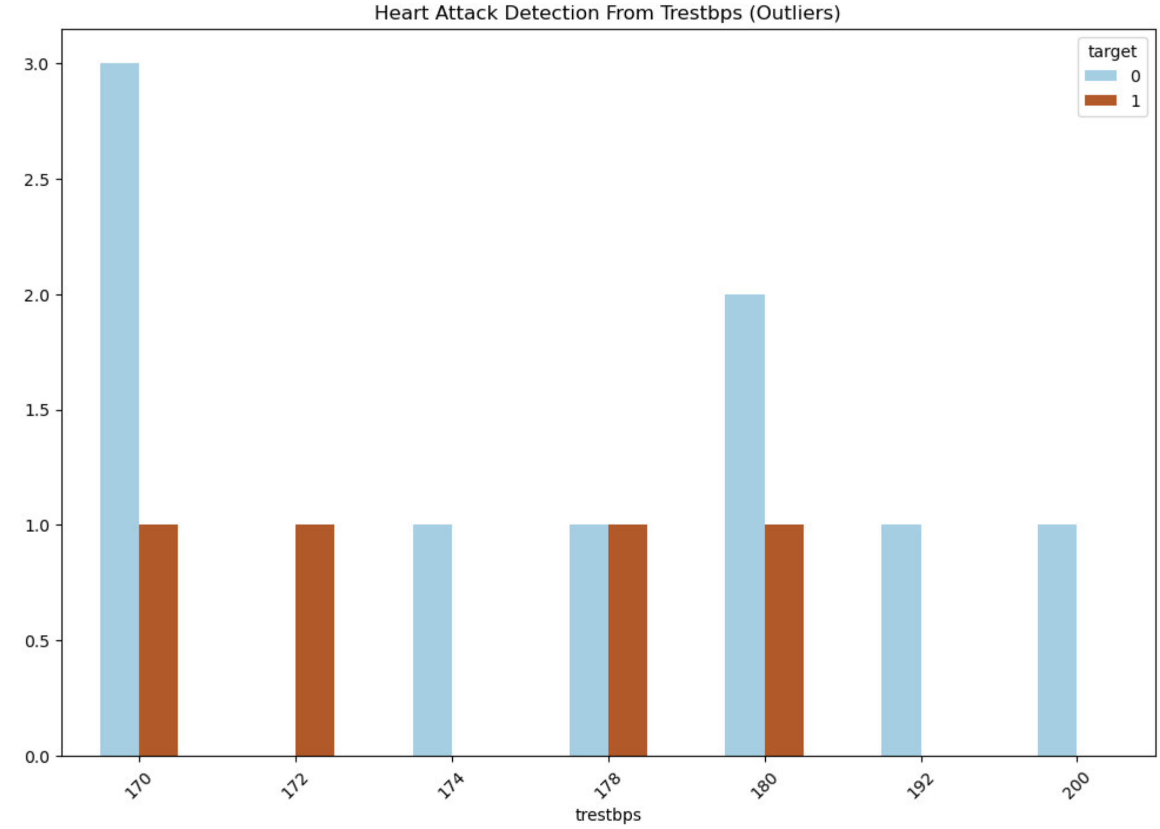
In [51]:
```
tbl_trestbps_out = pd.crosstab(columns = trestbps_out.target, index = tr
tbl_trestbps_out
```

Out[51]:

target	0	1
trestbps		
170	3	1
172	0	1
174	1	0
178	1	1
180	2	1
192	1	0
200	1	0

In [52]:
```
tbl_trestbps_out.plot(kind='bar', figsize=(12,8), stacked=False, colorma
plt.xticks(rotation=45)
plt.title('Heart Attack Detection From Trestbps (Outliers)')
```

Out[52]: Text(0.5, 1.0, 'Heart Attack Detection From Trestbps (Outliers)')




```
Interpretation:
1. Normal levels of trestbps is from 130-140 for adults.
1. There are 13 sample out of the total(patients) who have extreme trestbps ranging from 170-200 mm/Hg (outlier minimal)
2. Only 3 out 13 samples with extreme values for trestbps were diagnosed with cvd or heat attack, whilst 9 out of 13 patients with extreme values were diagnosed with no cvd.
4. There is a high probability that the levels were erroneously measured
```

In [64]:

```
# cp
cvd_1['cp'][cvd_1['cp'] == 0] = 'typical angina'
cvd_1['cp'][cvd_1['cp'] == 1] = 'atypical angina'
cvd_1['cp'][cvd_1['cp'] == 2] = 'non-anginal pain'
cvd_1['cp'][cvd_1['cp'] == 3] = 'asymptomatic'

# fbs
cvd_1['fbs'][cvd_1['fbs'] == 0] = 'lower than 120mg/ml'
cvd_1['fbs'][cvd_1['fbs'] == 1] = 'higher than 120mg/ml'

# restecg
cvd_1['restecg'][cvd_1['restecg'] == 0] = 'normal'
cvd_1['restecg'][cvd_1['restecg'] == 1] = 'ST-T wave abnormality'
cvd_1['restecg'][cvd_1['restecg'] == 2] = 'left ventricular hypertrophy

#thal
cvd_1['thal'][cvd_1['thal'] == 0 & 1] = 'normal'
cvd_1['thal'][cvd_1['thal'] == 2] = 'fixed defect'
cvd_1['thal'][cvd_1['thal'] == 3] = 'reversible defect'

# Continous Variables
# thalach (The person's maximum heart rate achieved)- Continous
# chol- Contioous ((The person's cholesterol measurement in mg/dL))
# oldpeak (ST depression induced by exercise relative to rest)
# ca (number of major vessels (0-3) colored by flourosopy)
# Age
# trestbps (The person's resting blood pressure )

#exang
cvd_1['exang'][cvd_1['exang'] == 0] = 'no'
cvd_1['exang'][cvd_1['exang'] == 1] = 'yes'

#sex
cvd_1['sex'][cvd_1['sex'] == 0] = 'female'
cvd_1['sex'][cvd_1['sex'] == 1] = 'male'

#slope
cvd_1['slope'][cvd_1['slope'] == 0] = 'Upsloping'
cvd_1['slope'][cvd_1['slope'] == 1] = 'Flatslopping'
cvd_1['slope'][cvd_1['slope'] == 2] = 'Downslopping'


```

In [65]:
```
cvd_1.head()
```

Out[65]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak
0	63	male	asymptomatic	145	233	higher than 120mg/ml	normal	150	no	2.3
1	37	male	non-anginal pain	130	250	lower than 120mg/ml	ST-T wave abnormality	187	no	3.5
2	41	female	atypical angina	130	204	lower than 120mg/ml	normal	172	no	1.4
3	56	male	atypical angina	120	236	lower than 120mg/ml	ST-T wave abnormality	178	no	0.8
4	57	female	typical angina	120	354	lower than 120mg/ml	ST-T wave abnormality	163	yes	0.6

In [66]:
```
cvd_1.shape
```

Out[66]: (302, 15)

cvd_1.drop('chol_bins', axis=1, inplace=True)

In [67]:
```
cvd_2= pd.get_dummies(cvd_1, drop_first=True)
```

In [68]:
```
cvd_2.head(2)
```

Out[68]:

	age	trestbps	chol	thalach	oldpeak	ca	target	sex_male	cp_atypical angina	cp_non-anginal pain	...	reste
0	63	145	233	150	2.3	0	1	1	0	0	...	
1	37	130	250	187	3.5	0	1	1	0	1	...	

2 rows × 23 columns

In [70]:
```
numeric_cvd_2 = cvd_2.select_dtypes(include=[np.number])
category_cvd_2 = cvd_2.select_dtypes(exclude=[np.number])
```

In [71]:
```
print (numeric_cvd_2.columns)
```

```
Index(['age', 'trestbps', 'chol', 'thalach', 'oldpeak', 'ca', 'target',
       'sex_male', 'cp_atypical angina', 'cp_non-anginal pain',
       'cp_typical angina', 'fbs_lower than 120mg/ml',
       'restecg_left ventricular hypertrophy', 'restecg_normal', 'exang_ye
s',
       'slope_Flatslopping', 'slope_Upsloping', 'thal_fixed defect',
       'thal_normal', 'thal_reversible defect', 'age_bins_Early_ Late Forti
es',
       'age_bins_Fifties_Early Sixties',
       'age_bins_Early Sixties_Late Seventies'],
      dtype='object')
```

In [72]:

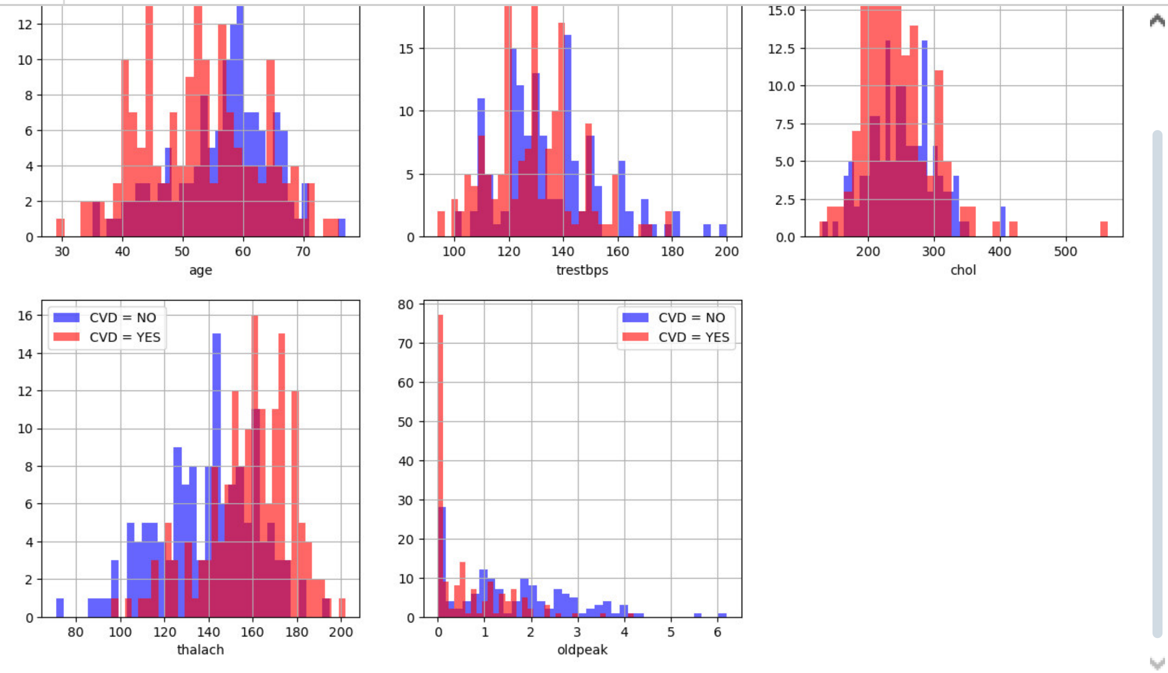
```
category_cvd_1 = []
numeric_cvd_1 = []
for column in cvd_1.columns:
    if len(cvd_1[column].unique()) <= 10:
        category_cvd_1.append(column)
    else:

        numeric_cvd_1.append(column)
```

```
category_cvd_1
```

Relationship between Numerical Variables and Target

In [74]:

```
plt.figure(figsize=(15, 15))

for i, column in enumerate(numeric_cvd_1, 1):
    plt.subplot(3, 3, i)
    cvd_1[cvd_1["target"] == 0][column].hist(bins=35, color='blue', labe
    cvd_1[cvd_1["target"] == 1][column].hist(bins=35, color='red', label
    plt.legend()
    plt.xlabel(column)
```




```
cvd_1['target_bins'] = pd.cut(cvd_1.target, [-1,0,1],
                              labels = ['CVD-Maybe', 'CVD-No', 'CVD-Yes'])
cvd_1[['target', 'target_bins']][:4]
```

```
plt.figure(figsize=(15, 15))

for i, column in enumerate(category_cvd_1, 1):
    plt.subplot(3, 3, i)
    cvd_2[cvd_2["target"] == 0][column].bar(bins=35, color='blue',
label='CVD = NO', alpha=0.6)
    cvd_2[cvd_2["target"] == 1][column].bar(bins=35, color='red',
label='CVD = YES', alpha=0.6)
    plt.legend()
```

```
    plt.xlabel(column)
```

In [75]:
```
cvd_1.corr()
```

Out[75]:

	age	trestbps	chol	thalach	oldpeak	ca	target
age	1.000000	0.283121	0.207216	-0.395235	0.206040	0.302261	-0.221476
trestbps	0.283121	1.000000	0.125256	-0.048023	0.194600	0.099248	-0.146269
chol	0.207216	0.125256	1.000000	-0.005308	0.050086	0.086878	-0.081437
thalach	-0.395235	-0.048023	-0.005308	1.000000	-0.342201	-0.228311	0.419955
oldpeak	0.206040	0.194600	0.050086	-0.342201	1.000000	0.236560	-0.429146
ca	0.302261	0.099248	0.086878	-0.228311	0.236560	1.000000	-0.408992
target	-0.221476	-0.146269	-0.081437	0.419955	-0.429146	-0.408992	1.000000

In [76]:
```
cvd_1.columns
```

Out[76]:
```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target', 'age_bins'],
      dtype='object')
```

In [77]:
```
cvd_1.drop(columns = ['age_bins'], inplace = True)
cvd_1.shape
```

Out[77]:
```
(302, 14)
```

In [78]:
```
numeric_cvd_1 = cvd_1.select_dtypes(include=[np.number])
category_cvd_1 = cvd_1.select_dtypes(exclude=[np.number])
print (numeric_cvd_1.columns)
print (category_cvd_1.columns)
```

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

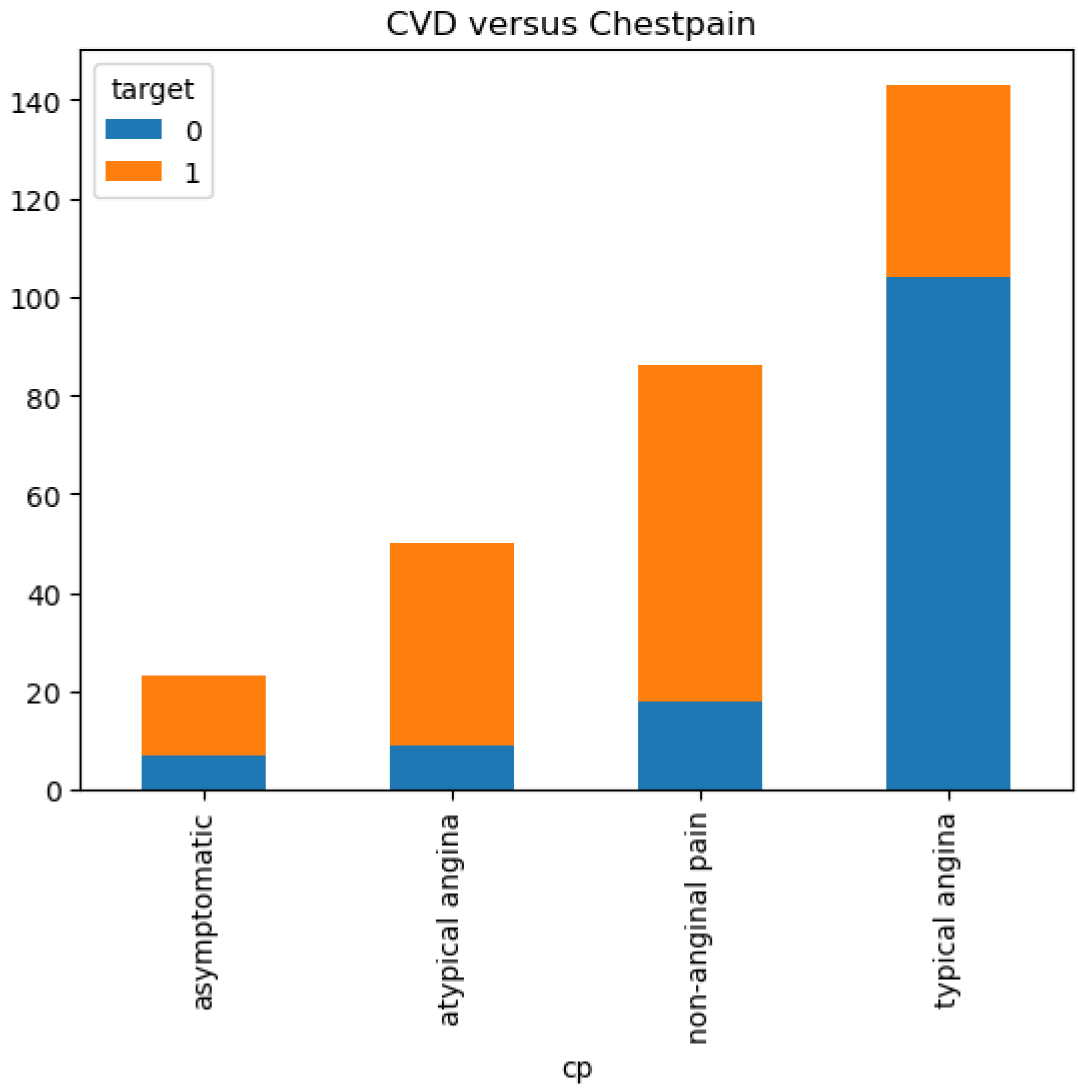
Relationship of Other Categorical Variables with CVD

```
# cp
cvd_1['cp'][cvd_1['cp'] == 0] = 'typical angina'
cvd_1['cp'][cvd_1['cp'] == 1] = 'atypical angina'
cvd_1['cp'][cvd_1['cp'] == 2] = 'non-anginal pain'
cvd_1['cp'][cvd_1['cp'] == 3] = 'asymptomatic'

# fbs
cvd_1['fbs'][cvd_1['fbs'] == 0] = 'lower than 120mg/ml'
cvd_1['fbs'][cvd_1['fbs'] == 1] = 'higher than 120mg/ml'

# restecg
cvd_1['restecg'][cvd_1['restecg'] == 0] = 'normal'
cvd_1['restecg'][cvd_1['restecg'] == 1] = 'ST-T wave abnormality'
cvd_1['restecg'][cvd_1['restecg'] == 2] = 'left ventricular hypertrophy'

```

```
#thal
cvd_1['thal'][cvd_1['thal'] == 0 & 1] = 'normal'
cvd_1['thal'][cvd_1['thal'] == 2] = 'fixed defect'
cvd_1['thal'][cvd_1['thal'] == 3] = 'reversible defect'

# Continous Variables
# thalach (The person's maximum heart rate achieved)- Continous
# chol- Contioous ((The person's cholesterol measurement in mg/dl))
# oldpeak (ST depression induced by exercise relative to rest)
# ca (number of major vessels (0-3) colored by flourosopy)
# Age
# trestbps (The person's resting blood pressure )

#exang
cvd_1['exang'][cvd_1['exang'] == 0] = 'no'
cvd_1['exang'][cvd_1['exang'] == 1] = 'yes'

#sex
cvd_1['sex'][cvd_1['sex'] == 0] = 'female'
cvd_1['sex'][cvd_1['sex'] == 1] = 'male'

#slope
cvd_1['slope'][cvd_1['slope'] == 0] = 'Upsloping'
cvd_1['slope'][cvd_1['slope'] == 1] = 'Flatslopping'
cvd_1['slope'][cvd_1['slope'] == 2] = 'Downslopping'
0: Upsloping: better heart rate with excercise (uncommon)
1: Flatsloping: minimal change (typical healthy heart)
2: Downslopins: signs of unhealthy heart
```

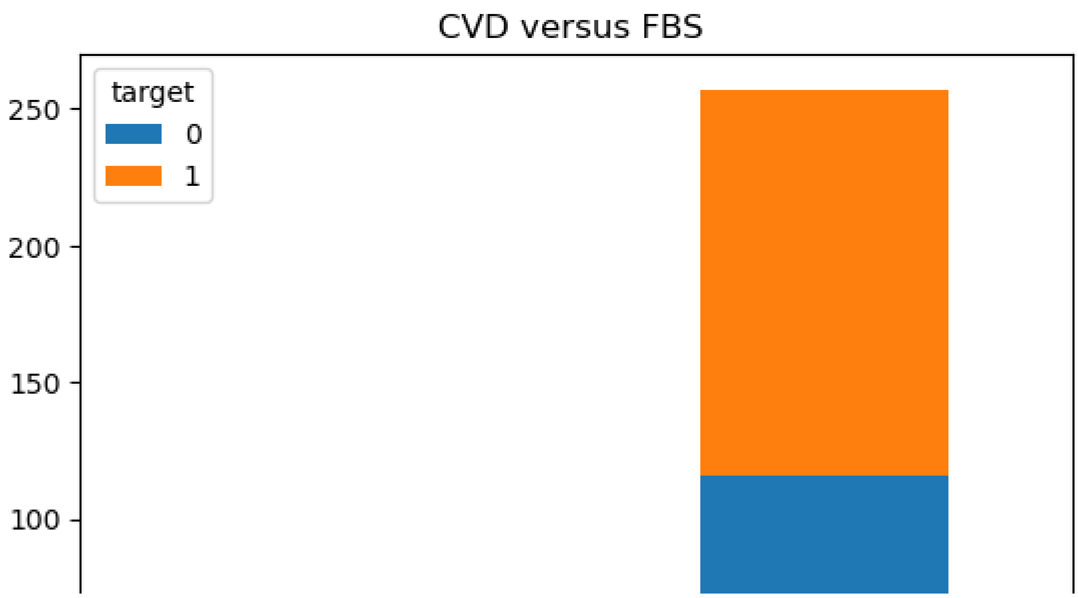
In [79]:
```
pd.crosstab(cvd_1['cp'],cvd_1['target']).plot.bar(stacked=True)
plt.title('CVD versus Chestpain ')
```

Out[79]:
```
Text(0.5, 1.0, 'CVD versus Chestpain ')
```



In [80]:
```
pd.crosstab(cvd_1['fbs'],cvd_1['target']).plot.bar(stacked=True)
plt.title('CVD versus FBS ')
```

Out[80]: Text(0.5, 1.0, 'CVD versus FBS ')

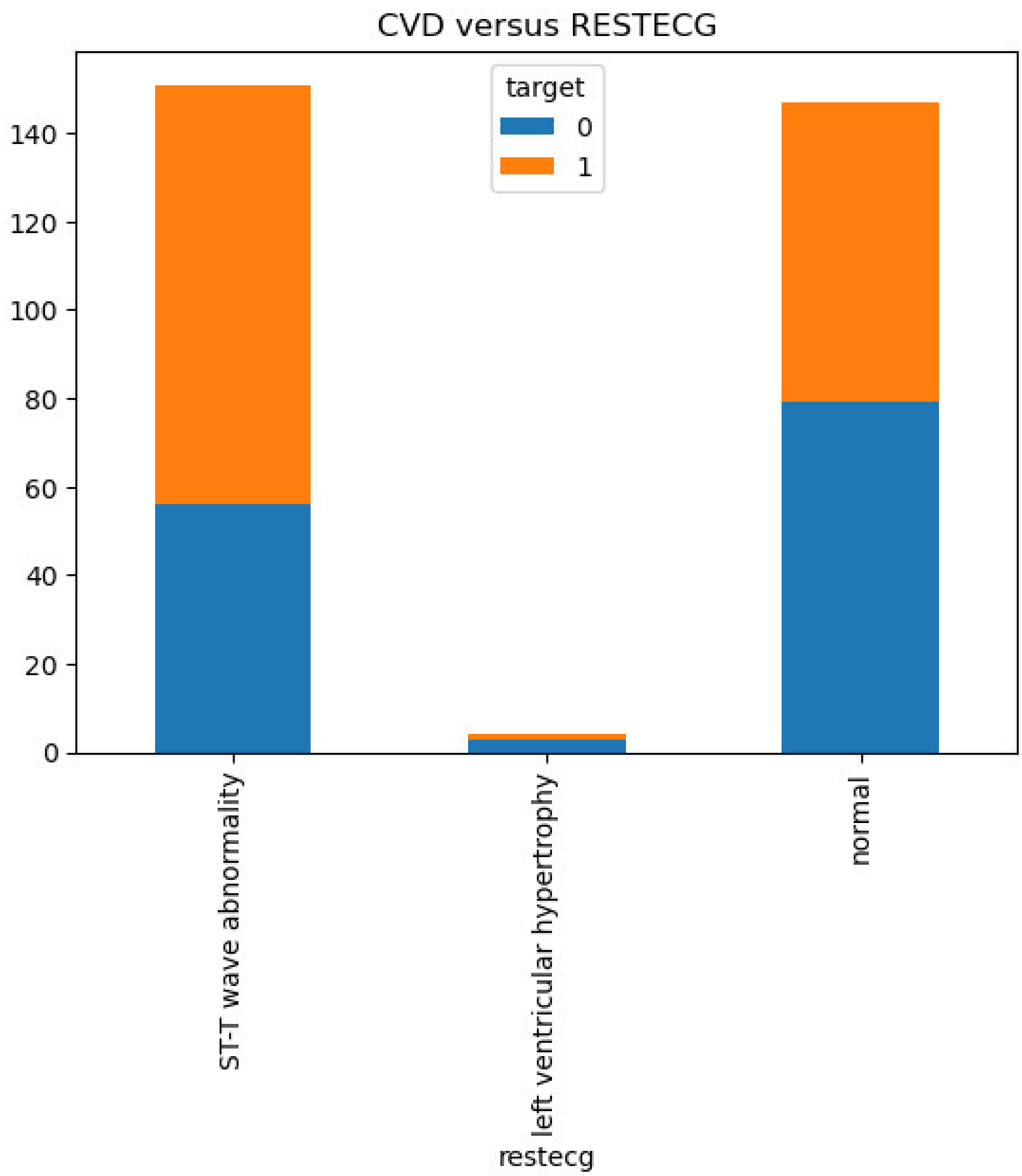



```
# fbs
cvd_1['fbs'][cvd_1['fbs'] == 0] = 'lower than 120mg/ml'
cvd_1['fbs'][cvd_1['fbs'] == 1] = 'higher than 120mg/ml'
```

```
'sex',
 'cp',
 'fbs',
 'restecg',
 'exang',
 'slope',
 'ca',
 'thal',
 'target',
 'age_bins']
```

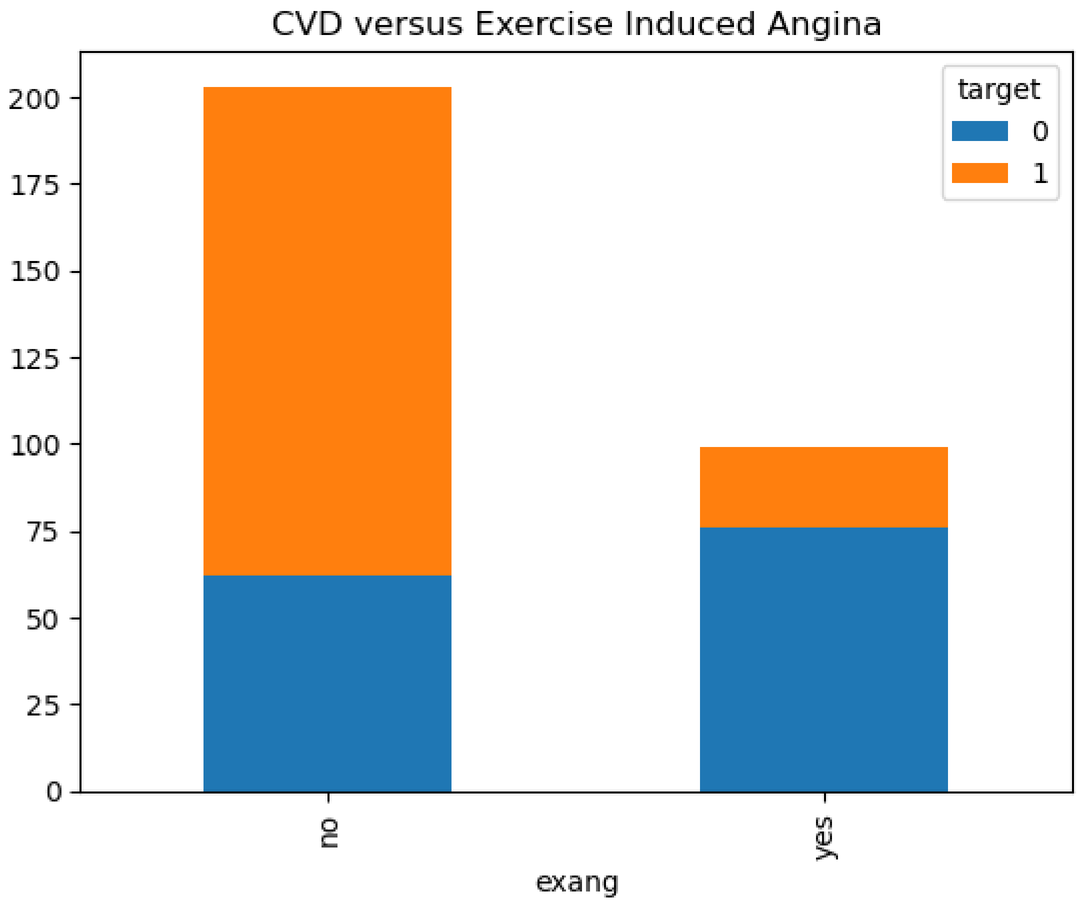
In [81]:
```
pd.crosstab(cvd_1['restecg'],cvd_1['target']).plot.bar(stacked=True)
plt.title('CVD versus RESTECG')
```

Out[81]: Text(0.5, 1.0, 'CVD versus RESTECG')



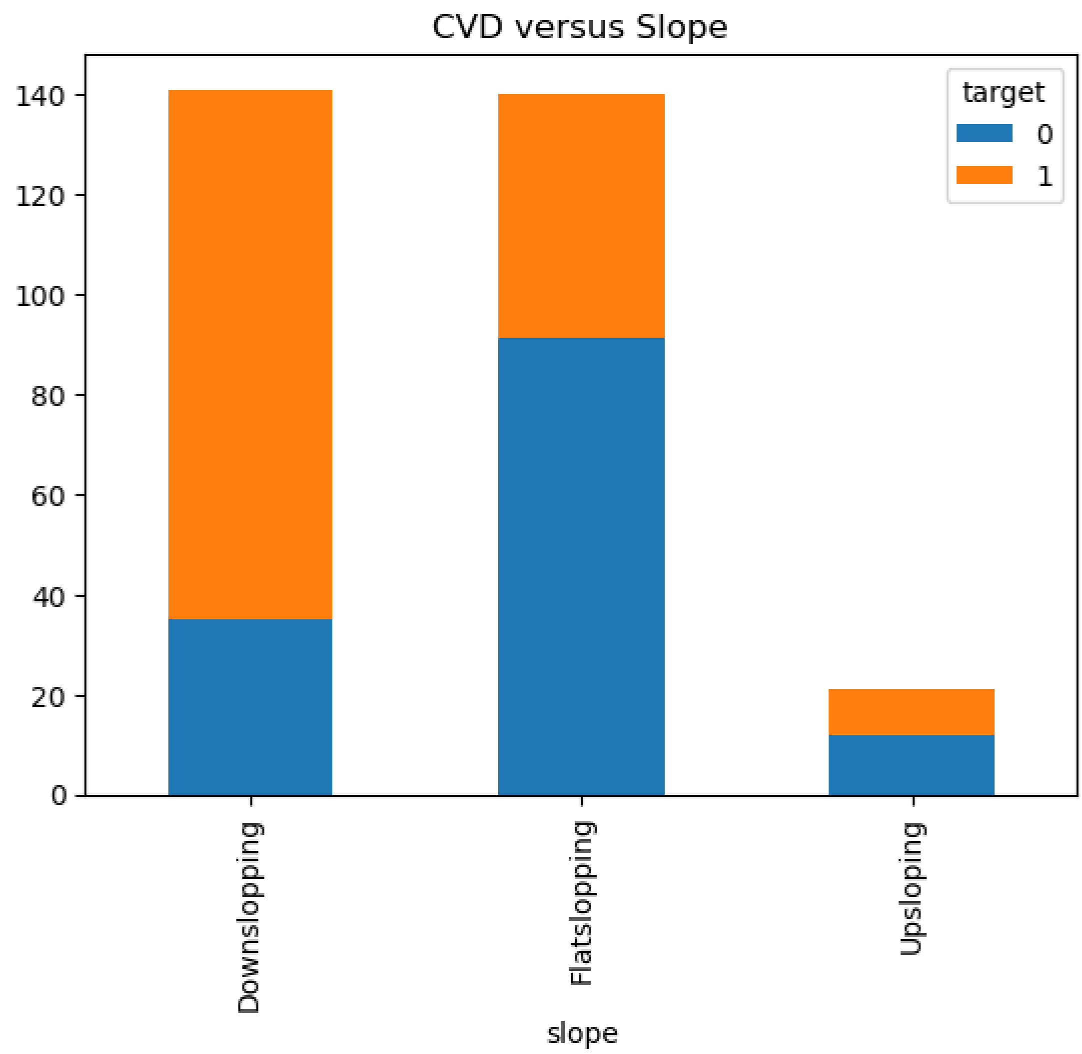
In [82]:
```
pd.crosstab(cvd_1['exang'],cvd_1['target']).plot.bar(stacked=True)
plt.title('CVD versus Exercise Induced Angina')
```

Out[82]: Text(0.5, 1.0, 'CVD versus Exercise Induced Angina')



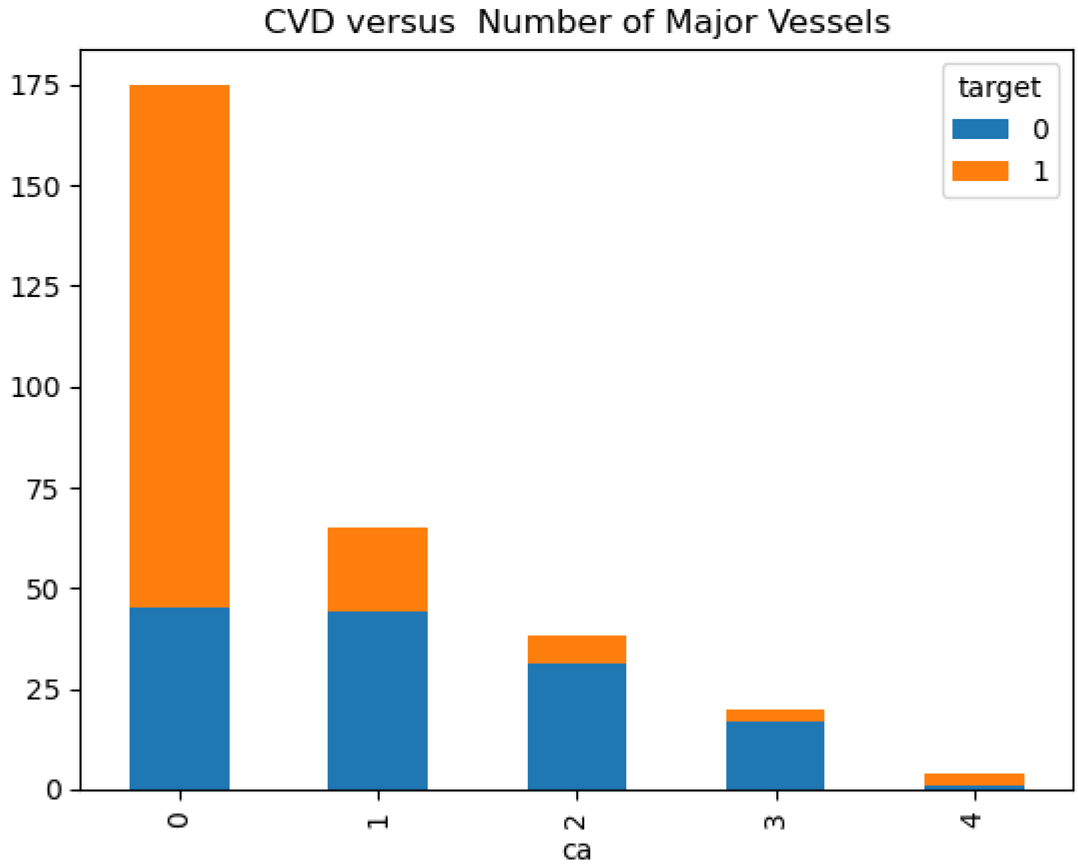
In [83]:
```
pd.crosstab(cvd_1['slope'],cvd_1['target']).plot.bar(stacked=True)
plt.title('CVD versus Slope')
```

Out[83]: Text(0.5, 1.0, 'CVD versus Slope')



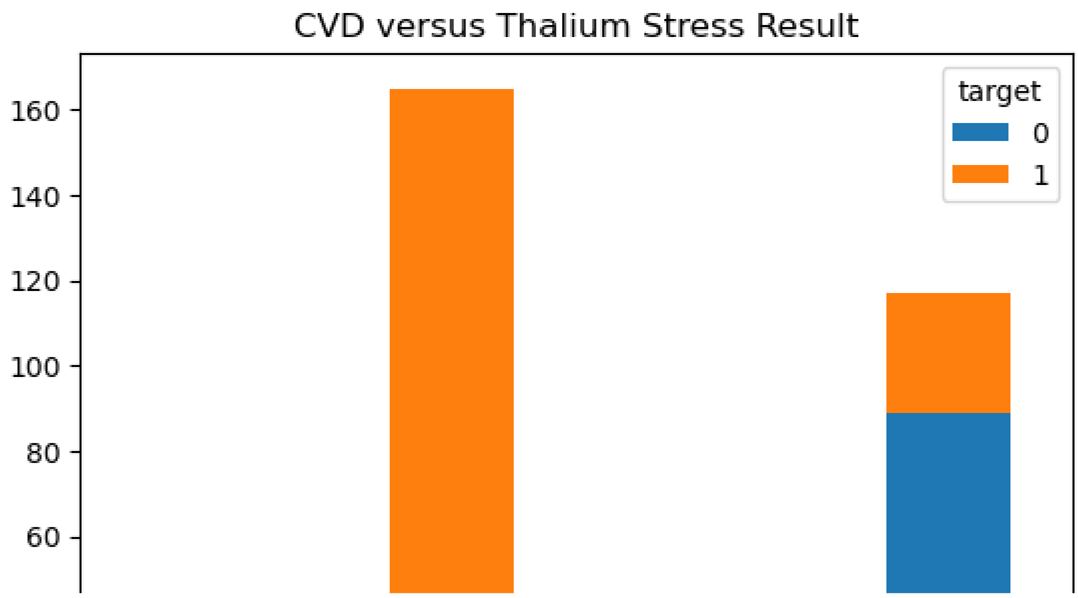
In [84]:
```
pd.crosstab(cvd_1['ca'],cvd_1['target']).plot.bar(stacked=True)
plt.title('CVD versus  Number of Major Vessels')
```

Out[84]: Text(0.5, 1.0, 'CVD versus Number of Major Vessels')



In [85]:
```
pd.crosstab(cvd_1['thal'],cvd_1['target']).plot.bar(stacked=True)
plt.title('CVD versus Thalium Stress Result')
```

Out[85]: Text(0.5, 1.0, 'CVD versus Thalium Stress Result')



Logistic Regression Model (Dummified data set = cvd_2)

In [86]:
```
cvd_2.columns
```

Out[86]:
```
Index(['age', 'trestbps', 'chol', 'thalach', 'oldpeak', 'ca', 'target',
       'sex_male', 'cp_atypical angina', 'cp_non-anginal pain',
       'cp_typical angina', 'fbs_lower than 120mg/ml',
       'restecg_left ventricular hypertrophy', 'restecg_normal', 'exang_ye
s',
       'slope_Flatslopping', 'slope_Upsloping', 'thal_fixed defect',
       'thal_normal', 'thal_reversible defect', 'age_bins_Early_ Late Forti
es',
       'age_bins_Fifties_Early Sixties',
       'age_bins_Early Sixties_Late Seventies'],
      dtype='object')
```

In [87]:
```
from sklearn.preprocessing import MinMaxScaler

mn = MinMaxScaler()
mn_df = mn.fit_transform(cvd_2)
```

In [88]:
```
mn_df_mn = pd.DataFrame(mn_df, columns=cvd_2.columns, index = cvd_2.inde
```

In [89]:
```
mn_df_mn.head()
```

Out[89]:

	age	trestbps	chol	thalach	oldpeak	ca	target	sex_male	cp_atypical angina	cp_nc angi pa
0	0.708333	0.481132	0.244292	0.603053	0.370968	0.0	1.0	1.0	0.0	(
1	0.166667	0.339623	0.283105	0.885496	0.564516	0.0	1.0	1.0	0.0	
2	0.250000	0.339623	0.178082	0.770992	0.225806	0.0	1.0	0.0	1.0	(
3	0.562500	0.245283	0.251142	0.816794	0.129032	0.0	1.0	1.0	1.0	(
4	0.583333	0.245283	0.520548	0.702290	0.096774	0.0	1.0	0.0	0.0	(

5 rows × 23 columns

In [90]:
```
mn_df_mn.corr().T
```

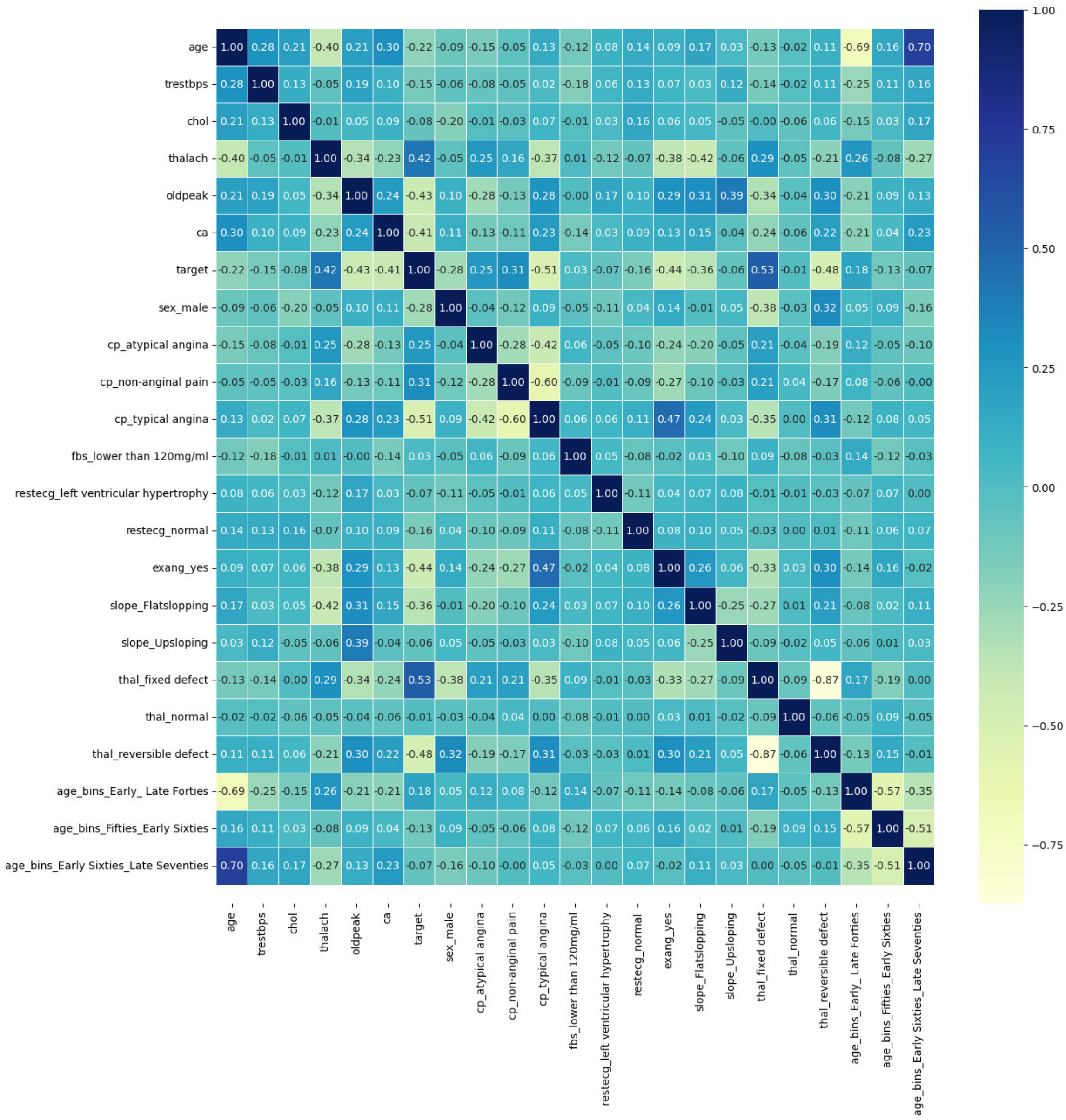
Out[90]:

	age	trestbps	chol	thalach	oldpeak	ca	t
age	1.000000	0.283121	0.207216	-0.395235	0.206040	0.302261	-0.22
trestbps	0.283121	1.000000	0.125256	-0.048023	0.194600	0.099248	-0.14
chol	0.207216	0.125256	1.000000	-0.005308	0.050086	0.086878	-0.08
thalach	-0.395235	-0.048023	-0.005308	1.000000	-0.342201	-0.228311	0.41
oldpeak	0.206040	0.194600	0.050086	-0.342201	1.000000	0.236560	-0.42
ca	0.302261	0.099248	0.086878	-0.228311	0.236560	1.000000	-0.40
target	-0.221476	-0.146269	-0.081437	0.419955	-0.429146	-0.408992	1.00
sex_male	-0.094962	-0.057647	-0.195571	-0.046439	0.098322	0.113060	-0.28
cp_atypical angina	-0.150921	-0.081359	-0.014828	0.250335	-0.279297	-0.132310	0.24
cp_non-anginal pain	-0.050494	-0.047212	-0.030957	0.161088	-0.128464	-0.108001	0.31

In [91]:

```
corr_matrix = mn_df_mn.corr()
fig, ax = plt.subplots(figsize=(15, 15))
ax = sns.heatmap(corr_matrix,
                 annot=True,
                 linewidths=0.5,
                 fmt=".2f",
                 cmap="YlGnBu");
bottom, top = ax.get_ylim()
ax.set_ylim(bottom + 0.5, top - 0.5)
```

Out[91]: (23.5, -0.5)



P-values using original variables in the original data set == cvd_3

In [92]:

```
import statsmodels.formula.api as smf
```

In [93]:

```
cvd_3 = pd.read_excel('healthcare.xlsx')
```

In [94]:
```
cvd_3.head()
```

Out[94]:

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

In [95]:
```
X = cvd_3.drop("target", axis=1)
y = cvd_3["target"]
```

In [96]:
```
model = smf.logit("target ~  age + sex + cp + trestbps + chol + fbs + re
```

```
Optimization terminated successfully.
         Current function value: 0.348904
         Iterations 7
```

In [97]:
```
print(model.summary())
```

```
                           Logit Regression Results
==========================================================================
=====
Dep. Variable:                 target   No. Observations:
303
Model:                          Logit   Df Residuals:
289
Method:                           MLE   Df Model:
13
Date:                Mon, 01 May 2023   Pseudo R-squ.:
0.4937
Time:                        14:08:27   Log-Likelihood:                 -1
05.72
converged:                       True   LL-Null:                        -2
08.82
Covariance Type:            nonrobust   LLR p-value:                  7.26
2e-37
==========================================================================
=====
```

```
# Significant variables
sex
cp
thalach
exang
oldpeak
ca
thal
```

```
# For p-value computation ONLY
X = cvd_2.drop("target", axis=1)
```

```
y = cvd_2["target"]
```

Log. Regression Model using df == cvd_2

In [162]:

```
from sklearn.preprocessing import MinMaxScaler

mn = MinMaxScaler()
mn_df = mn.fit_transform(cvd_2)
```

In [163]:

```
mn_df_mn = pd.DataFrame(mn_df, columns=cvd_2.columns, index = cvd_2.inde
```

In [164]:

```
mn_df_mn.head()
```

Out[164]:

	age	trestbps	chol	thalach	oldpeak	ca	target	sex_male	cp_atypical angina	cp_nc angi pa
0	0.708333	0.481132	0.244292	0.603053	0.370968	0.0	1.0	1.0	0.0	(
1	0.166667	0.339623	0.283105	0.885496	0.564516	0.0	1.0	1.0	0.0	
2	0.250000	0.339623	0.178082	0.770992	0.225806	0.0	1.0	0.0	1.0	(
3	0.562500	0.245283	0.251142	0.816794	0.129032	0.0	1.0	1.0	1.0	(
4	0.583333	0.245283	0.520548	0.702290	0.096774	0.0	1.0	0.0	0.0	(

5 rows × 23 columns

In [165]:

```
X = mn_df_mn
y = cvd_2["target"]
```

In [166]:

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1(
```

In [167]:

```
from sklearn.linear_model import LogisticRegression
```

In [168]:

```
lr =  LogisticRegression()
lr.fit(X_train, y_train)
```

Out[168]:

```
LogisticRegression()
```

In [169]:

```
pred = lr.predict(X_test)
```

In [170]:

```
from sklearn.metrics import accuracy_score, confusion_matrix, classifica
```

In [171]:

```
confusion_matrix(y_test, pred)
```

Out[171]:

```
array([[18,  0],
       [ 0, 13]], dtype=int64)
```

In [172]:
```
print(classification_report(y_test, pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        18
           1       1.00      1.00      1.00        13

    accuracy                           1.00        31
   macro avg       1.00      1.00      1.00        31
weighted avg       1.00      1.00      1.00        31
```

In [173]:
```
predss = lr.predict_proba(X_test)
```

In [174]:
```
predss
```

Out[174]:
```
array([[0.96746558, 0.03253442],
       [0.01572056, 0.98427944],
       [0.95308467, 0.04691533],
       [0.03357107, 0.96642893],
       [0.97537265, 0.02462735],
       [0.01924937, 0.98075063],
       [0.03540847, 0.96459153],
       [0.88711865, 0.11288135],
       [0.01676346, 0.98323654],
       [0.89756889, 0.10243111],
       [0.88175564, 0.11824436],
       [0.97767002, 0.02232998],
       [0.92085624, 0.07914376],
       [0.02942103, 0.97057897],
       [0.97852442, 0.02147558],
       [0.03173915, 0.96826085],
       [0.96764685, 0.03235315],
       [0.98524252, 0.01475748],
       [0.06301854, 0.93698146],
       [0.95567985, 0.04432015],
       [0.01591966, 0.98408034],
       [0.01930447, 0.98069553],
       [0.8478041 , 0.1521959 ],
       [0.0205194 , 0.9794806 ],
       [0.9884496 , 0.0115504 ],
       [0.94197903, 0.05802097],
       [0.9773061 , 0.0226939 ],
       [0.95991176, 0.04008824],
       [0.01347614, 0.98652386],
       [0.98963819, 0.01036181],
       [0.01950527, 0.98049473]])
```

Random Forest Model using df == cvd_2

In [175]:
```
from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier()
rf.fit(X_train, y_train)
```

Out[175]:
```
RandomForestClassifier()
```

In [176]:
```
pred = rf.predict(X_test)
```

In [177]:
```
accuracy_score(y_test, pred)
```

Out[177]: 1.0

In [178]:
```
accuracy_score(y_train, rf.predict(X_train))
```

Out[178]: 1.0

In [179]:
```
confusion_matrix(y_test, pred)
```

Out[179]:
```
array([[18,  0],
       [ 0, 13]], dtype=int64)
```

In [180]:
```
print(classification_report(y_test, pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        18
           1       1.00      1.00      1.00        13

    accuracy                           1.00        31
   macro avg       1.00      1.00      1.00        31
weighted avg       1.00      1.00      1.00        31
```

LR Model using orginal df == cvd_3

In [181]:
```
cvd_3.head()
```

Out[181]:

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

In [182]:
```
from sklearn.preprocessing import MinMaxScaler

mn = MinMaxScaler()
mn_df_2 = mn.fit_transform(cvd_3)
```

In [183]:
```
mn_df_mn_2 = pd.DataFrame(mn_df_2, columns=cvd_3.columns, index = cvd_3
```

In [184]:
```
mn_df_mn_2.head()
```

Out[184]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slop
0	0.708333	1.0	1.000000	0.481132	0.244292	1.0	0.0	0.603053	0.0	0.370968	0.
1	0.166667	1.0	0.666667	0.339623	0.283105	0.0	0.5	0.885496	0.0	0.564516	0.
2	0.250000	0.0	0.333333	0.339623	0.178082	0.0	0.0	0.770992	0.0	0.225806	1.
3	0.562500	1.0	0.333333	0.245283	0.251142	0.0	0.5	0.816794	0.0	0.129032	1.
4	0.583333	0.0	0.000000	0.245283	0.520548	0.0	0.5	0.702290	1.0	0.096774	1.

In [185]:
```
x2 = mn_df_mn_2
y2 = cvd_3["target"]
```

In [186]:
```
from sklearn.model_selection import train_test_split

x2_train, x2_test, y2_train, y2_test = train_test_split(x2, y2, test_si
```

In [187]:
```
lr_2 =  LogisticRegression()
lr_2.fit(x2_train, y2_train)
```

Out[187]:
```
LogisticRegression()
```

In [188]:
```
pred_2 = lr_2.predict(x2_test)
```

In [189]:
```
confusion_matrix(y2_test, pred_2)
```

Out[189]:
```
array([[18,  0],
       [ 0, 13]], dtype=int64)
```

In [146]:
```
print(classification_report(y2_test, pred_2))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        18
           1       1.00      1.00      1.00        13

    accuracy                           1.00        31
   macro avg       1.00      1.00      1.00        31
weighted avg       1.00      1.00      1.00        31
```

Random Forest using orignal df = cvd_3

In [190]:
```
from sklearn.ensemble import RandomForestClassifier

rf_2 = RandomForestClassifier()
rf_2.fit(x2_train, y2_train)
```

Out[190]:
```
RandomForestClassifier()
```

In [191]:
```
pred_2= rf_2.predict(x2_test)
```

In [192]:
```
accuracy_score(y2_test, pred)
```

Out[192]: 0.9354838709677419

In [193]:
```
accuracy_score(y2_train, rf_2.predict(x2_train))
```

Out[193]: 1.0

In [194]:
```
confusion_matrix(y2_test, pred_2)
```

Out[194]:
```
array([[18,  0],
       [ 0, 13]], dtype=int64)
```

In [195]:
```
print(classification_report(y2_test, pred_2))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        18
           1       1.00      1.00      1.00        13

    accuracy                           1.00        31
   macro avg       1.00      1.00      1.00        31
weighted avg       1.00      1.00      1.00        31
```

LR & RF Models with dropped insignifcant variables df= cardio

#--'age', 'trestbps', 'chol', 'fbs', 'restecg', 'slope'

In [197]:
```
cardio = pd.read_excel('healthcare.xlsx')
```

In [198]:
```
cardio.drop(columns = ['age', 'trestbps', 'chol', 'fbs', 'restecg', 'slo
cardio.shape
```

Out[198]: (303, 8)

In [199]:
```
cardio.head()
```

Out[199]:

	sex	cp	thalach	exang	oldpeak	ca	thal	target
0	1	3	150	0	2.3	0	1	1
1	1	2	187	0	3.5	0	2	1
2	0	1	172	0	1.4	0	2	1
3	1	1	178	0	0.8	0	2	1
4	0	0	163	1	0.6	0	2	1

In [214]:
```
from sklearn.preprocessing import MinMaxScaler

mn = MinMaxScaler()
mn_df_3 = mn.fit_transform(cardio)
```

In [215]:
```
mn_df_mn_3 = pd.DataFrame(mn_df_3, columns=cardio.columns, index = card:
```

In [216]:
```
x3 = mn_df_mn_3
y3 = cardio["target"]
```

In [217]:
```
from sklearn.model_selection import train_test_split

x3_train, x3_test, y3_train, y3_test = train_test_split(x3, y3, test_si:
```

In [218]:
```
from sklearn.linear_model import LogisticRegression
```

In [219]:
```
lr_3 =  LogisticRegression()
lr_3.fit(x3_train, y3_train)
```

Out[219]:
```
LogisticRegression()
```

In [220]:
```
pred_3 = lr_3.predict(x3_test)
```

In [221]:
```
from sklearn.metrics import accuracy_score, confusion_matrix, classifica
```

In [222]:
```
confusion_matrix(y3_test, pred_3)
```

Out[222]:
```
array([[18,  0],
       [ 0, 13]], dtype=int64)
```

In [223]:
```
from sklearn.ensemble import RandomForestClassifier

rf_3 = RandomForestClassifier()
rf_3.fit(x3_train, y3_train)
```

Out[223]:
```
RandomForestClassifier()
```

In [224]:
```
accuracy_score(y3_test, pred)
```

Out[224]:
```
1.0
```

In [225]:
```
accuracy_score(y3_train, rf_3.predict(x3_train))
```

Out[225]:
```
1.0
```

In [226]:
```
confusion_matrix(y3_test, pred_3)
```

Out[226]:
```
array([[18,  0],
       [ 0, 13]], dtype=int64)
```

In [227]:
```
print(classification_report(y3_test, pred_3))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        18
           1       1.00      1.00      1.00        13

    accuracy                           1.00        31
   macro avg       1.00      1.00      1.00        31
weighted avg       1.00      1.00      1.00        31
```