Taking 'processed.clevaland.data' data.

In [116]:

#importing library pandas, reading csv file and labeling columns

import pandas as pd

dataframe = pd.read\_csv("http://archive.ics.uci.edu/ml/machine-learning-databases/heart-disease/processed.cleveland.data", names= ['age','sex','cp','trestbps','chol','fbs','restecg','thalach','exang','oldpeak','slope','ca','thal','num'])

subset = dataframe[:100000]

In [117]:

#showing dataframe

dataframe

Out[117]:

|  | **age** | **sex** | **cp** | **trestbps** | **chol** | **fbs** | **restecg** | **thalach** | **exang** | **oldpeak** | **slope** | **ca** | **thal** | **num** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 63.0 | 1.0 | 1.0 | 145.0 | 233.0 | 1.0 | 2.0 | 150.0 | 0.0 | 2.3 | 3.0 | 0.0 | 6.0 | 0 |
| **1** | 67.0 | 1.0 | 4.0 | 160.0 | 286.0 | 0.0 | 2.0 | 108.0 | 1.0 | 1.5 | 2.0 | 3.0 | 3.0 | 2 |
| **2** | 67.0 | 1.0 | 4.0 | 120.0 | 229.0 | 0.0 | 2.0 | 129.0 | 1.0 | 2.6 | 2.0 | 2.0 | 7.0 | 1 |
| **3** | 37.0 | 1.0 | 3.0 | 130.0 | 250.0 | 0.0 | 0.0 | 187.0 | 0.0 | 3.5 | 3.0 | 0.0 | 3.0 | 0 |
| **4** | 41.0 | 0.0 | 2.0 | 130.0 | 204.0 | 0.0 | 2.0 | 172.0 | 0.0 | 1.4 | 1.0 | 0.0 | 3.0 | 0 |
| **5** | 56.0 | 1.0 | 2.0 | 120.0 | 236.0 | 0.0 | 0.0 | 178.0 | 0.0 | 0.8 | 1.0 | 0.0 | 3.0 | 0 |
| **6** | 62.0 | 0.0 | 4.0 | 140.0 | 268.0 | 0.0 | 2.0 | 160.0 | 0.0 | 3.6 | 3.0 | 2.0 | 3.0 | 3 |
| **7** | 57.0 | 0.0 | 4.0 | 120.0 | 354.0 | 0.0 | 0.0 | 163.0 | 1.0 | 0.6 | 1.0 | 0.0 | 3.0 | 0 |
| **8** | 63.0 | 1.0 | 4.0 | 130.0 | 254.0 | 0.0 | 2.0 | 147.0 | 0.0 | 1.4 | 2.0 | 1.0 | 7.0 | 2 |
| **9** | 53.0 | 1.0 | 4.0 | 140.0 | 203.0 | 1.0 | 2.0 | 155.0 | 1.0 | 3.1 | 3.0 | 0.0 | 7.0 | 1 |
| **10** | 57.0 | 1.0 | 4.0 | 140.0 | 192.0 | 0.0 | 0.0 | 148.0 | 0.0 | 0.4 | 2.0 | 0.0 | 6.0 | 0 |
| **11** | 56.0 | 0.0 | 2.0 | 140.0 | 294.0 | 0.0 | 2.0 | 153.0 | 0.0 | 1.3 | 2.0 | 0.0 | 3.0 | 0 |
| **12** | 56.0 | 1.0 | 3.0 | 130.0 | 256.0 | 1.0 | 2.0 | 142.0 | 1.0 | 0.6 | 2.0 | 1.0 | 6.0 | 2 |
| **13** | 44.0 | 1.0 | 2.0 | 120.0 | 263.0 | 0.0 | 0.0 | 173.0 | 0.0 | 0.0 | 1.0 | 0.0 | 7.0 | 0 |
| **14** | 52.0 | 1.0 | 3.0 | 172.0 | 199.0 | 1.0 | 0.0 | 162.0 | 0.0 | 0.5 | 1.0 | 0.0 | 7.0 | 0 |
| **15** | 57.0 | 1.0 | 3.0 | 150.0 | 168.0 | 0.0 | 0.0 | 174.0 | 0.0 | 1.6 | 1.0 | 0.0 | 3.0 | 0 |
| **16** | 48.0 | 1.0 | 2.0 | 110.0 | 229.0 | 0.0 | 0.0 | 168.0 | 0.0 | 1.0 | 3.0 | 0.0 | 7.0 | 1 |
| **17** | 54.0 | 1.0 | 4.0 | 140.0 | 239.0 | 0.0 | 0.0 | 160.0 | 0.0 | 1.2 | 1.0 | 0.0 | 3.0 | 0 |
| **18** | 48.0 | 0.0 | 3.0 | 130.0 | 275.0 | 0.0 | 0.0 | 139.0 | 0.0 | 0.2 | 1.0 | 0.0 | 3.0 | 0 |
| **19** | 49.0 | 1.0 | 2.0 | 130.0 | 266.0 | 0.0 | 0.0 | 171.0 | 0.0 | 0.6 | 1.0 | 0.0 | 3.0 | 0 |
| **20** | 64.0 | 1.0 | 1.0 | 110.0 | 211.0 | 0.0 | 2.0 | 144.0 | 1.0 | 1.8 | 2.0 | 0.0 | 3.0 | 0 |
| **21** | 58.0 | 0.0 | 1.0 | 150.0 | 283.0 | 1.0 | 2.0 | 162.0 | 0.0 | 1.0 | 1.0 | 0.0 | 3.0 | 0 |
| **22** | 58.0 | 1.0 | 2.0 | 120.0 | 284.0 | 0.0 | 2.0 | 160.0 | 0.0 | 1.8 | 2.0 | 0.0 | 3.0 | 1 |
| **23** | 58.0 | 1.0 | 3.0 | 132.0 | 224.0 | 0.0 | 2.0 | 173.0 | 0.0 | 3.2 | 1.0 | 2.0 | 7.0 | 3 |
| **24** | 60.0 | 1.0 | 4.0 | 130.0 | 206.0 | 0.0 | 2.0 | 132.0 | 1.0 | 2.4 | 2.0 | 2.0 | 7.0 | 4 |
| **25** | 50.0 | 0.0 | 3.0 | 120.0 | 219.0 | 0.0 | 0.0 | 158.0 | 0.0 | 1.6 | 2.0 | 0.0 | 3.0 | 0 |
| **26** | 58.0 | 0.0 | 3.0 | 120.0 | 340.0 | 0.0 | 0.0 | 172.0 | 0.0 | 0.0 | 1.0 | 0.0 | 3.0 | 0 |
| **27** | 66.0 | 0.0 | 1.0 | 150.0 | 226.0 | 0.0 | 0.0 | 114.0 | 0.0 | 2.6 | 3.0 | 0.0 | 3.0 | 0 |
| **28** | 43.0 | 1.0 | 4.0 | 150.0 | 247.0 | 0.0 | 0.0 | 171.0 | 0.0 | 1.5 | 1.0 | 0.0 | 3.0 | 0 |
| **29** | 40.0 | 1.0 | 4.0 | 110.0 | 167.0 | 0.0 | 2.0 | 114.0 | 1.0 | 2.0 | 2.0 | 0.0 | 7.0 | 3 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **273** | 71.0 | 0.0 | 4.0 | 112.0 | 149.0 | 0.0 | 0.0 | 125.0 | 0.0 | 1.6 | 2.0 | 0.0 | 3.0 | 0 |
| **274** | 59.0 | 1.0 | 1.0 | 134.0 | 204.0 | 0.0 | 0.0 | 162.0 | 0.0 | 0.8 | 1.0 | 2.0 | 3.0 | 1 |
| **275** | 64.0 | 1.0 | 1.0 | 170.0 | 227.0 | 0.0 | 2.0 | 155.0 | 0.0 | 0.6 | 2.0 | 0.0 | 7.0 | 0 |
| **276** | 66.0 | 0.0 | 3.0 | 146.0 | 278.0 | 0.0 | 2.0 | 152.0 | 0.0 | 0.0 | 2.0 | 1.0 | 3.0 | 0 |
| **277** | 39.0 | 0.0 | 3.0 | 138.0 | 220.0 | 0.0 | 0.0 | 152.0 | 0.0 | 0.0 | 2.0 | 0.0 | 3.0 | 0 |
| **278** | 57.0 | 1.0 | 2.0 | 154.0 | 232.0 | 0.0 | 2.0 | 164.0 | 0.0 | 0.0 | 1.0 | 1.0 | 3.0 | 1 |
| **279** | 58.0 | 0.0 | 4.0 | 130.0 | 197.0 | 0.0 | 0.0 | 131.0 | 0.0 | 0.6 | 2.0 | 0.0 | 3.0 | 0 |
| **280** | 57.0 | 1.0 | 4.0 | 110.0 | 335.0 | 0.0 | 0.0 | 143.0 | 1.0 | 3.0 | 2.0 | 1.0 | 7.0 | 2 |
| **281** | 47.0 | 1.0 | 3.0 | 130.0 | 253.0 | 0.0 | 0.0 | 179.0 | 0.0 | 0.0 | 1.0 | 0.0 | 3.0 | 0 |
| **282** | 55.0 | 0.0 | 4.0 | 128.0 | 205.0 | 0.0 | 1.0 | 130.0 | 1.0 | 2.0 | 2.0 | 1.0 | 7.0 | 3 |
| **283** | 35.0 | 1.0 | 2.0 | 122.0 | 192.0 | 0.0 | 0.0 | 174.0 | 0.0 | 0.0 | 1.0 | 0.0 | 3.0 | 0 |
| **284** | 61.0 | 1.0 | 4.0 | 148.0 | 203.0 | 0.0 | 0.0 | 161.0 | 0.0 | 0.0 | 1.0 | 1.0 | 7.0 | 2 |
| **285** | 58.0 | 1.0 | 4.0 | 114.0 | 318.0 | 0.0 | 1.0 | 140.0 | 0.0 | 4.4 | 3.0 | 3.0 | 6.0 | 4 |
| **286** | 58.0 | 0.0 | 4.0 | 170.0 | 225.0 | 1.0 | 2.0 | 146.0 | 1.0 | 2.8 | 2.0 | 2.0 | 6.0 | 2 |
| **287** | 58.0 | 1.0 | 2.0 | 125.0 | 220.0 | 0.0 | 0.0 | 144.0 | 0.0 | 0.4 | 2.0 | ? | 7.0 | 0 |
| **288** | 56.0 | 1.0 | 2.0 | 130.0 | 221.0 | 0.0 | 2.0 | 163.0 | 0.0 | 0.0 | 1.0 | 0.0 | 7.0 | 0 |
| **289** | 56.0 | 1.0 | 2.0 | 120.0 | 240.0 | 0.0 | 0.0 | 169.0 | 0.0 | 0.0 | 3.0 | 0.0 | 3.0 | 0 |
| **290** | 67.0 | 1.0 | 3.0 | 152.0 | 212.0 | 0.0 | 2.0 | 150.0 | 0.0 | 0.8 | 2.0 | 0.0 | 7.0 | 1 |
| **291** | 55.0 | 0.0 | 2.0 | 132.0 | 342.0 | 0.0 | 0.0 | 166.0 | 0.0 | 1.2 | 1.0 | 0.0 | 3.0 | 0 |
| **292** | 44.0 | 1.0 | 4.0 | 120.0 | 169.0 | 0.0 | 0.0 | 144.0 | 1.0 | 2.8 | 3.0 | 0.0 | 6.0 | 2 |
| **293** | 63.0 | 1.0 | 4.0 | 140.0 | 187.0 | 0.0 | 2.0 | 144.0 | 1.0 | 4.0 | 1.0 | 2.0 | 7.0 | 2 |
| **294** | 63.0 | 0.0 | 4.0 | 124.0 | 197.0 | 0.0 | 0.0 | 136.0 | 1.0 | 0.0 | 2.0 | 0.0 | 3.0 | 1 |
| **295** | 41.0 | 1.0 | 2.0 | 120.0 | 157.0 | 0.0 | 0.0 | 182.0 | 0.0 | 0.0 | 1.0 | 0.0 | 3.0 | 0 |
| **296** | 59.0 | 1.0 | 4.0 | 164.0 | 176.0 | 1.0 | 2.0 | 90.0 | 0.0 | 1.0 | 2.0 | 2.0 | 6.0 | 3 |
| **297** | 57.0 | 0.0 | 4.0 | 140.0 | 241.0 | 0.0 | 0.0 | 123.0 | 1.0 | 0.2 | 2.0 | 0.0 | 7.0 | 1 |
| **298** | 45.0 | 1.0 | 1.0 | 110.0 | 264.0 | 0.0 | 0.0 | 132.0 | 0.0 | 1.2 | 2.0 | 0.0 | 7.0 | 1 |
| **299** | 68.0 | 1.0 | 4.0 | 144.0 | 193.0 | 1.0 | 0.0 | 141.0 | 0.0 | 3.4 | 2.0 | 2.0 | 7.0 | 2 |
| **300** | 57.0 | 1.0 | 4.0 | 130.0 | 131.0 | 0.0 | 0.0 | 115.0 | 1.0 | 1.2 | 2.0 | 1.0 | 7.0 | 3 |
| **301** | 57.0 | 0.0 | 2.0 | 130.0 | 236.0 | 0.0 | 2.0 | 174.0 | 0.0 | 0.0 | 2.0 | 1.0 | 3.0 | 1 |
| **302** | 38.0 | 1.0 | 3.0 | 138.0 | 175.0 | 0.0 | 0.0 | 173.0 | 0.0 | 0.0 | 1.0 | ? | 3.0 | 0 |

303 rows × 14 columns

In [118]:

#replacing ? with NAn in columns

dataframe= dataframe.replace('?','NAn')

#converting column 'ca' and 'thal' datatype to float64

dataframe['ca']=dataframe['ca'].astype(float)

dataframe['thal']=dataframe['thal'].astype(float)

#replacing Nan with mean

dataframe = dataframe.fillna(dataframe.mean())

#drop duplicate rows

dataframe = dataframe.drop\_duplicates(keep = 'first')

In [119]:

#dataframe after replacing nan with mean value

dataframe

Out[119]:

|  | **age** | **sex** | **cp** | **trestbps** | **chol** | **fbs** | **restecg** | **thalach** | **exang** | **oldpeak** | **slope** | **ca** | **thal** | **num** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 63.0 | 1.0 | 1.0 | 145.0 | 233.0 | 1.0 | 2.0 | 150.0 | 0.0 | 2.3 | 3.0 | 0.000000 | 6.0 | 0 |
| **1** | 67.0 | 1.0 | 4.0 | 160.0 | 286.0 | 0.0 | 2.0 | 108.0 | 1.0 | 1.5 | 2.0 | 3.000000 | 3.0 | 2 |
| **2** | 67.0 | 1.0 | 4.0 | 120.0 | 229.0 | 0.0 | 2.0 | 129.0 | 1.0 | 2.6 | 2.0 | 2.000000 | 7.0 | 1 |
| **3** | 37.0 | 1.0 | 3.0 | 130.0 | 250.0 | 0.0 | 0.0 | 187.0 | 0.0 | 3.5 | 3.0 | 0.000000 | 3.0 | 0 |
| **4** | 41.0 | 0.0 | 2.0 | 130.0 | 204.0 | 0.0 | 2.0 | 172.0 | 0.0 | 1.4 | 1.0 | 0.000000 | 3.0 | 0 |
| **5** | 56.0 | 1.0 | 2.0 | 120.0 | 236.0 | 0.0 | 0.0 | 178.0 | 0.0 | 0.8 | 1.0 | 0.000000 | 3.0 | 0 |
| **6** | 62.0 | 0.0 | 4.0 | 140.0 | 268.0 | 0.0 | 2.0 | 160.0 | 0.0 | 3.6 | 3.0 | 2.000000 | 3.0 | 3 |
| **7** | 57.0 | 0.0 | 4.0 | 120.0 | 354.0 | 0.0 | 0.0 | 163.0 | 1.0 | 0.6 | 1.0 | 0.000000 | 3.0 | 0 |
| **8** | 63.0 | 1.0 | 4.0 | 130.0 | 254.0 | 0.0 | 2.0 | 147.0 | 0.0 | 1.4 | 2.0 | 1.000000 | 7.0 | 2 |
| **9** | 53.0 | 1.0 | 4.0 | 140.0 | 203.0 | 1.0 | 2.0 | 155.0 | 1.0 | 3.1 | 3.0 | 0.000000 | 7.0 | 1 |
| **10** | 57.0 | 1.0 | 4.0 | 140.0 | 192.0 | 0.0 | 0.0 | 148.0 | 0.0 | 0.4 | 2.0 | 0.000000 | 6.0 | 0 |
| **11** | 56.0 | 0.0 | 2.0 | 140.0 | 294.0 | 0.0 | 2.0 | 153.0 | 0.0 | 1.3 | 2.0 | 0.000000 | 3.0 | 0 |
| **12** | 56.0 | 1.0 | 3.0 | 130.0 | 256.0 | 1.0 | 2.0 | 142.0 | 1.0 | 0.6 | 2.0 | 1.000000 | 6.0 | 2 |
| **13** | 44.0 | 1.0 | 2.0 | 120.0 | 263.0 | 0.0 | 0.0 | 173.0 | 0.0 | 0.0 | 1.0 | 0.000000 | 7.0 | 0 |
| **14** | 52.0 | 1.0 | 3.0 | 172.0 | 199.0 | 1.0 | 0.0 | 162.0 | 0.0 | 0.5 | 1.0 | 0.000000 | 7.0 | 0 |
| **15** | 57.0 | 1.0 | 3.0 | 150.0 | 168.0 | 0.0 | 0.0 | 174.0 | 0.0 | 1.6 | 1.0 | 0.000000 | 3.0 | 0 |
| **16** | 48.0 | 1.0 | 2.0 | 110.0 | 229.0 | 0.0 | 0.0 | 168.0 | 0.0 | 1.0 | 3.0 | 0.000000 | 7.0 | 1 |
| **17** | 54.0 | 1.0 | 4.0 | 140.0 | 239.0 | 0.0 | 0.0 | 160.0 | 0.0 | 1.2 | 1.0 | 0.000000 | 3.0 | 0 |
| **18** | 48.0 | 0.0 | 3.0 | 130.0 | 275.0 | 0.0 | 0.0 | 139.0 | 0.0 | 0.2 | 1.0 | 0.000000 | 3.0 | 0 |
| **19** | 49.0 | 1.0 | 2.0 | 130.0 | 266.0 | 0.0 | 0.0 | 171.0 | 0.0 | 0.6 | 1.0 | 0.000000 | 3.0 | 0 |
| **20** | 64.0 | 1.0 | 1.0 | 110.0 | 211.0 | 0.0 | 2.0 | 144.0 | 1.0 | 1.8 | 2.0 | 0.000000 | 3.0 | 0 |
| **21** | 58.0 | 0.0 | 1.0 | 150.0 | 283.0 | 1.0 | 2.0 | 162.0 | 0.0 | 1.0 | 1.0 | 0.000000 | 3.0 | 0 |
| **22** | 58.0 | 1.0 | 2.0 | 120.0 | 284.0 | 0.0 | 2.0 | 160.0 | 0.0 | 1.8 | 2.0 | 0.000000 | 3.0 | 1 |
| **23** | 58.0 | 1.0 | 3.0 | 132.0 | 224.0 | 0.0 | 2.0 | 173.0 | 0.0 | 3.2 | 1.0 | 2.000000 | 7.0 | 3 |
| **24** | 60.0 | 1.0 | 4.0 | 130.0 | 206.0 | 0.0 | 2.0 | 132.0 | 1.0 | 2.4 | 2.0 | 2.000000 | 7.0 | 4 |
| **25** | 50.0 | 0.0 | 3.0 | 120.0 | 219.0 | 0.0 | 0.0 | 158.0 | 0.0 | 1.6 | 2.0 | 0.000000 | 3.0 | 0 |
| **26** | 58.0 | 0.0 | 3.0 | 120.0 | 340.0 | 0.0 | 0.0 | 172.0 | 0.0 | 0.0 | 1.0 | 0.000000 | 3.0 | 0 |
| **27** | 66.0 | 0.0 | 1.0 | 150.0 | 226.0 | 0.0 | 0.0 | 114.0 | 0.0 | 2.6 | 3.0 | 0.000000 | 3.0 | 0 |
| **28** | 43.0 | 1.0 | 4.0 | 150.0 | 247.0 | 0.0 | 0.0 | 171.0 | 0.0 | 1.5 | 1.0 | 0.000000 | 3.0 | 0 |
| **29** | 40.0 | 1.0 | 4.0 | 110.0 | 167.0 | 0.0 | 2.0 | 114.0 | 1.0 | 2.0 | 2.0 | 0.000000 | 7.0 | 3 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **273** | 71.0 | 0.0 | 4.0 | 112.0 | 149.0 | 0.0 | 0.0 | 125.0 | 0.0 | 1.6 | 2.0 | 0.000000 | 3.0 | 0 |
| **274** | 59.0 | 1.0 | 1.0 | 134.0 | 204.0 | 0.0 | 0.0 | 162.0 | 0.0 | 0.8 | 1.0 | 2.000000 | 3.0 | 1 |
| **275** | 64.0 | 1.0 | 1.0 | 170.0 | 227.0 | 0.0 | 2.0 | 155.0 | 0.0 | 0.6 | 2.0 | 0.000000 | 7.0 | 0 |
| **276** | 66.0 | 0.0 | 3.0 | 146.0 | 278.0 | 0.0 | 2.0 | 152.0 | 0.0 | 0.0 | 2.0 | 1.000000 | 3.0 | 0 |
| **277** | 39.0 | 0.0 | 3.0 | 138.0 | 220.0 | 0.0 | 0.0 | 152.0 | 0.0 | 0.0 | 2.0 | 0.000000 | 3.0 | 0 |
| **278** | 57.0 | 1.0 | 2.0 | 154.0 | 232.0 | 0.0 | 2.0 | 164.0 | 0.0 | 0.0 | 1.0 | 1.000000 | 3.0 | 1 |
| **279** | 58.0 | 0.0 | 4.0 | 130.0 | 197.0 | 0.0 | 0.0 | 131.0 | 0.0 | 0.6 | 2.0 | 0.000000 | 3.0 | 0 |
| **280** | 57.0 | 1.0 | 4.0 | 110.0 | 335.0 | 0.0 | 0.0 | 143.0 | 1.0 | 3.0 | 2.0 | 1.000000 | 7.0 | 2 |
| **281** | 47.0 | 1.0 | 3.0 | 130.0 | 253.0 | 0.0 | 0.0 | 179.0 | 0.0 | 0.0 | 1.0 | 0.000000 | 3.0 | 0 |
| **282** | 55.0 | 0.0 | 4.0 | 128.0 | 205.0 | 0.0 | 1.0 | 130.0 | 1.0 | 2.0 | 2.0 | 1.000000 | 7.0 | 3 |
| **283** | 35.0 | 1.0 | 2.0 | 122.0 | 192.0 | 0.0 | 0.0 | 174.0 | 0.0 | 0.0 | 1.0 | 0.000000 | 3.0 | 0 |
| **284** | 61.0 | 1.0 | 4.0 | 148.0 | 203.0 | 0.0 | 0.0 | 161.0 | 0.0 | 0.0 | 1.0 | 1.000000 | 7.0 | 2 |
| **285** | 58.0 | 1.0 | 4.0 | 114.0 | 318.0 | 0.0 | 1.0 | 140.0 | 0.0 | 4.4 | 3.0 | 3.000000 | 6.0 | 4 |
| **286** | 58.0 | 0.0 | 4.0 | 170.0 | 225.0 | 1.0 | 2.0 | 146.0 | 1.0 | 2.8 | 2.0 | 2.000000 | 6.0 | 2 |
| **287** | 58.0 | 1.0 | 2.0 | 125.0 | 220.0 | 0.0 | 0.0 | 144.0 | 0.0 | 0.4 | 2.0 | 0.672241 | 7.0 | 0 |
| **288** | 56.0 | 1.0 | 2.0 | 130.0 | 221.0 | 0.0 | 2.0 | 163.0 | 0.0 | 0.0 | 1.0 | 0.000000 | 7.0 | 0 |
| **289** | 56.0 | 1.0 | 2.0 | 120.0 | 240.0 | 0.0 | 0.0 | 169.0 | 0.0 | 0.0 | 3.0 | 0.000000 | 3.0 | 0 |
| **290** | 67.0 | 1.0 | 3.0 | 152.0 | 212.0 | 0.0 | 2.0 | 150.0 | 0.0 | 0.8 | 2.0 | 0.000000 | 7.0 | 1 |
| **291** | 55.0 | 0.0 | 2.0 | 132.0 | 342.0 | 0.0 | 0.0 | 166.0 | 0.0 | 1.2 | 1.0 | 0.000000 | 3.0 | 0 |
| **292** | 44.0 | 1.0 | 4.0 | 120.0 | 169.0 | 0.0 | 0.0 | 144.0 | 1.0 | 2.8 | 3.0 | 0.000000 | 6.0 | 2 |
| **293** | 63.0 | 1.0 | 4.0 | 140.0 | 187.0 | 0.0 | 2.0 | 144.0 | 1.0 | 4.0 | 1.0 | 2.000000 | 7.0 | 2 |
| **294** | 63.0 | 0.0 | 4.0 | 124.0 | 197.0 | 0.0 | 0.0 | 136.0 | 1.0 | 0.0 | 2.0 | 0.000000 | 3.0 | 1 |
| **295** | 41.0 | 1.0 | 2.0 | 120.0 | 157.0 | 0.0 | 0.0 | 182.0 | 0.0 | 0.0 | 1.0 | 0.000000 | 3.0 | 0 |
| **296** | 59.0 | 1.0 | 4.0 | 164.0 | 176.0 | 1.0 | 2.0 | 90.0 | 0.0 | 1.0 | 2.0 | 2.000000 | 6.0 | 3 |
| **297** | 57.0 | 0.0 | 4.0 | 140.0 | 241.0 | 0.0 | 0.0 | 123.0 | 1.0 | 0.2 | 2.0 | 0.000000 | 7.0 | 1 |
| **298** | 45.0 | 1.0 | 1.0 | 110.0 | 264.0 | 0.0 | 0.0 | 132.0 | 0.0 | 1.2 | 2.0 | 0.000000 | 7.0 | 1 |
| **299** | 68.0 | 1.0 | 4.0 | 144.0 | 193.0 | 1.0 | 0.0 | 141.0 | 0.0 | 3.4 | 2.0 | 2.000000 | 7.0 | 2 |
| **300** | 57.0 | 1.0 | 4.0 | 130.0 | 131.0 | 0.0 | 0.0 | 115.0 | 1.0 | 1.2 | 2.0 | 1.000000 | 7.0 | 3 |
| **301** | 57.0 | 0.0 | 2.0 | 130.0 | 236.0 | 0.0 | 2.0 | 174.0 | 0.0 | 0.0 | 2.0 | 1.000000 | 3.0 | 1 |
| **302** | 38.0 | 1.0 | 3.0 | 138.0 | 175.0 | 0.0 | 0.0 | 173.0 | 0.0 | 0.0 | 1.0 | 0.672241 | 3.0 | 0 |

303 rows × 14 columns

In [120]:

dataframe.describe()

Out[120]:

|  | **age** | **sex** | **cp** | **trestbps** | **chol** | **fbs** | **restecg** | **thalach** | **exang** | **oldpeak** | **slope** | **ca** | **thal** | **num** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **count** | 303.000000 | 303.000000 | 303.000000 | 303.000000 | 303.000000 | 303.000000 | 303.000000 | 303.000000 | 303.000000 | 303.000000 | 303.000000 | 303.000000 | 303.000000 | 303.000000 |
| **mean** | 54.438944 | 0.679868 | 3.158416 | 131.689769 | 246.693069 | 0.148515 | 0.990099 | 149.607261 | 0.326733 | 1.039604 | 1.600660 | 0.672241 | 4.734219 | 0.937294 |
| **std** | 9.038662 | 0.467299 | 0.960126 | 17.599748 | 51.776918 | 0.356198 | 0.994971 | 22.875003 | 0.469794 | 1.161075 | 0.616226 | 0.931209 | 1.933272 | 1.228536 |
| **min** | 29.000000 | 0.000000 | 1.000000 | 94.000000 | 126.000000 | 0.000000 | 0.000000 | 71.000000 | 0.000000 | 0.000000 | 1.000000 | 0.000000 | 3.000000 | 0.000000 |
| **25%** | 48.000000 | 0.000000 | 3.000000 | 120.000000 | 211.000000 | 0.000000 | 0.000000 | 133.500000 | 0.000000 | 0.000000 | 1.000000 | 0.000000 | 3.000000 | 0.000000 |
| **50%** | 56.000000 | 1.000000 | 3.000000 | 130.000000 | 241.000000 | 0.000000 | 1.000000 | 153.000000 | 0.000000 | 0.800000 | 2.000000 | 0.000000 | 3.000000 | 0.000000 |
| **75%** | 61.000000 | 1.000000 | 4.000000 | 140.000000 | 275.000000 | 0.000000 | 2.000000 | 166.000000 | 1.000000 | 1.600000 | 2.000000 | 1.000000 | 7.000000 | 2.000000 |
| **max** | 77.000000 | 1.000000 | 4.000000 | 200.000000 | 564.000000 | 1.000000 | 2.000000 | 202.000000 | 1.000000 | 6.200000 | 3.000000 | 3.000000 | 7.000000 | 4.000000 |

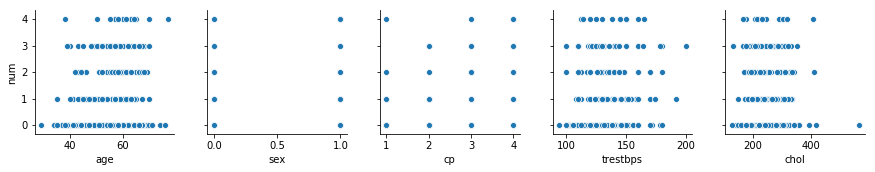
In [121]:

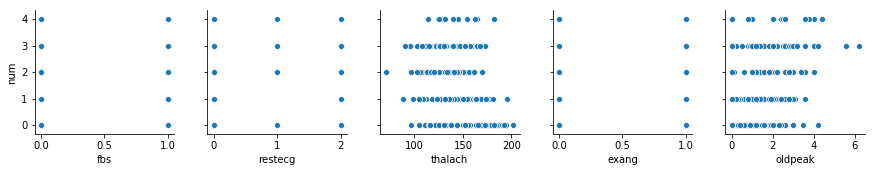
#individual relationship of num with rest columns of dataset

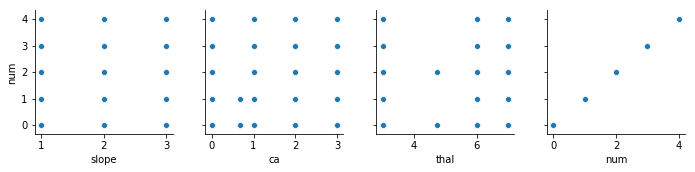
import seaborn as sns

for i in range(0, len(dataframe.columns),5):

sns.pairplot(dataframe, y\_vars =['num'],x\_vars= dataframe.columns[i:i+5])







In [122]:

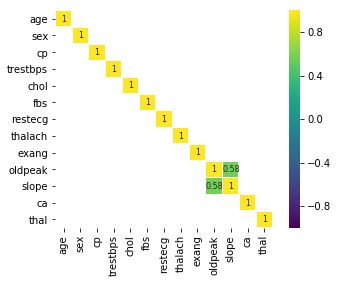
#plotting significant corr in one plot (heatmap)

corr = dataframe.drop('num', axis = 1).corr()

sns.heatmap(corr[(corr>=0.5) | (corr<=-0.4)],

cmap='viridis', vmax =1.0, vmin=-1.0, linewidths = 0.5,

annot =True, annot\_kws = {"size":8}, square =True);



In [123]:

#count of number of values num attribute hold

dataframe['num'].value\_counts()

Out[123]:

0 164

1 55

2 36

3 35

4 13

Name: num, dtype: int64

In [124]:

#replacing value 2 to 4 as 1 in num attribute

dataframe['num'][dataframe['num'] > 1] = 1

dataframe['num'].value\_counts()

G:\Software Installed\Anaconda\lib\site-packages\ipykernel\_launcher.py:2: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy

Out[124]:

0 164

1 139

Name: num, dtype: int64

In [125]:

#importing matplotlib library

import matplotlib

%matplotlib inline

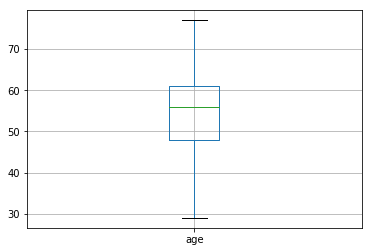
In [126]:

#using box plot for analysing outliers on column 'age'

dataframe.boxplot(column='age')

Out[126]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x1d85c0de128>



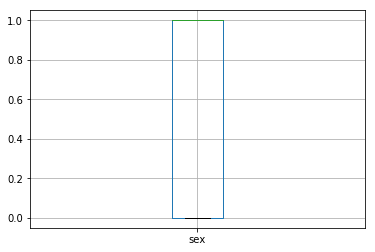
In [127]:

#using box plot for analysing outliers on column 'sex'

dataframe.boxplot(column='sex')

Out[127]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x1d85bff47b8>



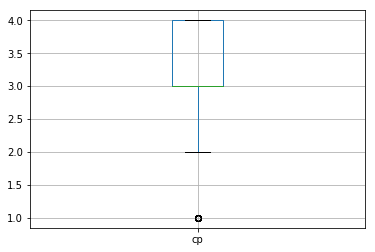
In [128]:

#using box plot for analysing outliers on column 'cp'

dataframe.boxplot(column='cp')

Out[128]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x1d85c35b2b0>



In [129]:

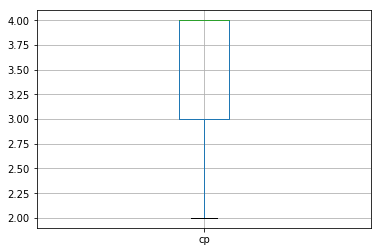
#removing outliers in column 'cp'

dataframe.drop(dataframe['cp'][dataframe['cp'] < 2].index,inplace=True)

dataframe.boxplot(column='cp')

Out[129]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x1d85c1cc4a8>



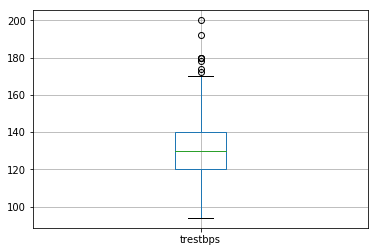
In [130]:

#using box plot for analysing outliers on column 'trestbps'

dataframe.boxplot(column='trestbps')

Out[130]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x1d85c0d0550>



In [131]:

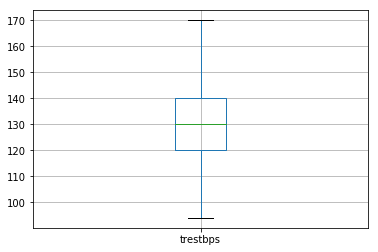
#removing outliers in column 'trestbps'

dataframe.drop(dataframe['trestbps'][dataframe['trestbps'] >170 ].index,inplace=True)

dataframe.boxplot(column='trestbps')

Out[131]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x1d85c2fc320>



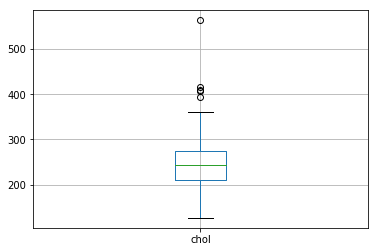
In [132]:

#using box plot for analysing outliers on column 'chol'

dataframe.boxplot(column='chol')

Out[132]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x1d85c35ce48>



In [133]:

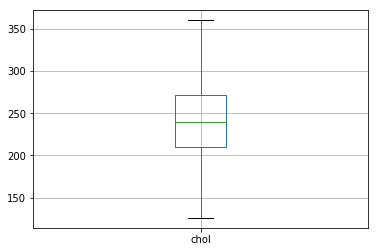
#removing outliers in column 'chol'

dataframe.drop(dataframe['chol'][dataframe['chol'] > 375 ].index,inplace=True)

dataframe.boxplot(column='chol')

Out[133]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x1d85c067320>



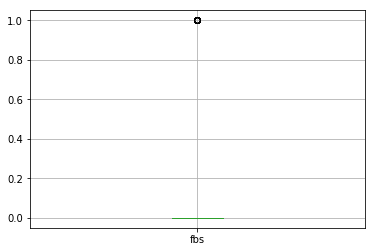
In [134]:

#using box plot for analysing outliers on column 'fbs'

dataframe.boxplot(column='fbs')

Out[134]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x1d85bdda358>



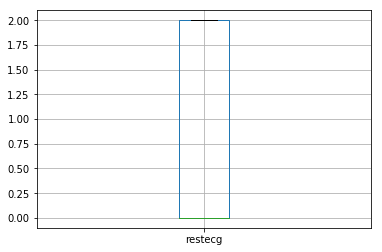
In [135]:

#using box plot for analysing outliers on column 'restecg'

dataframe.boxplot('restecg')

Out[135]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x1d85c560c50>



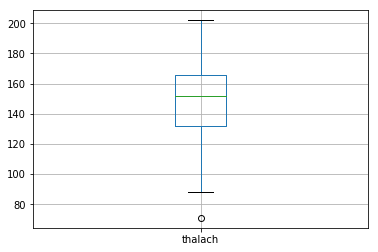
In [136]:

#using box plot for analysing outliers on column 'thalach'

dataframe.boxplot('thalach')

Out[136]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x1d85c5b97b8>



In [137]:

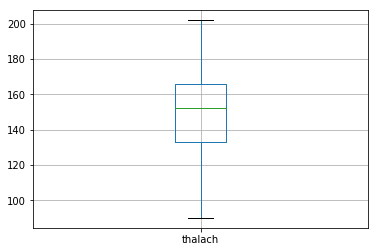
#removing outliers in column 'thalach'

dataframe.drop(dataframe['thalach'][dataframe['thalach'] <90 ].index,inplace=True)

dataframe.boxplot(column='thalach')

Out[137]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x1d85be30748>



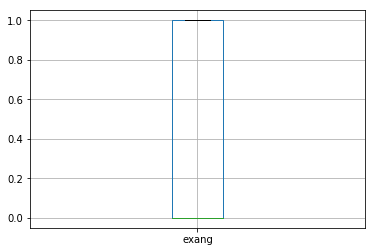
In [138]:

#using box plot for analysing outliers on column 'exang'

dataframe.boxplot('exang')

Out[138]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x1d85c46d6d8>



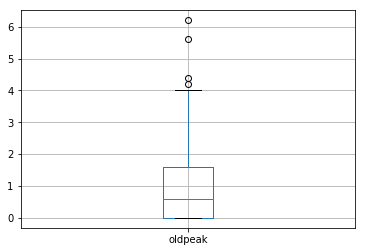
In [139]:

#using box plot for analysing outliers on column 'oldpeak'

dataframe.boxplot('oldpeak')

Out[139]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x1d85c5e1048>



In [140]:

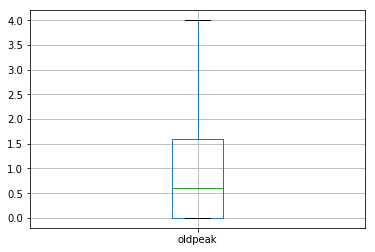
#removing outliers in column 'oldpeak'

dataframe.drop(dataframe['oldpeak'][dataframe['oldpeak'] > 4 ].index,inplace=True)

dataframe.boxplot(column='oldpeak')

Out[140]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x1d85bd35ac8>



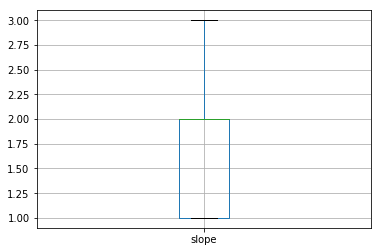
In [141]:

#using box plot for analysing outliers on column 'slope'

dataframe.boxplot('slope')

Out[141]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x1d85c661390>



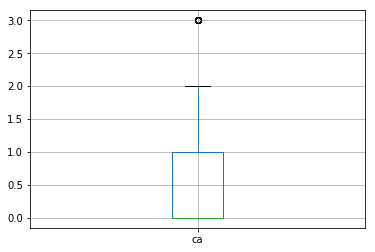
In [142]:

#using box plot for analysing outliers on column 'ca'

dataframe.boxplot('ca')

Out[142]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x1d85bfaa8d0>



In [143]:

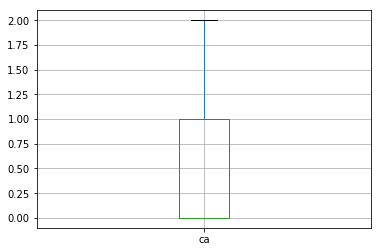
#removing outliers in column 'ca'

dataframe.drop(dataframe['ca'][dataframe['ca'] > 2 ].index,inplace=True)

dataframe.boxplot(column='ca')

Out[143]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x1d85bfd1a90>



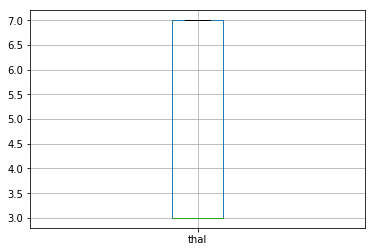
In [144]:

#using box plot for analysing outliers on column 'thal'

dataframe.boxplot('thal')

Out[144]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x1d85bf40940>



In [145]:

#correlation between columns in tabular form after removing outliers

dataframe.corr()

Out[145]:

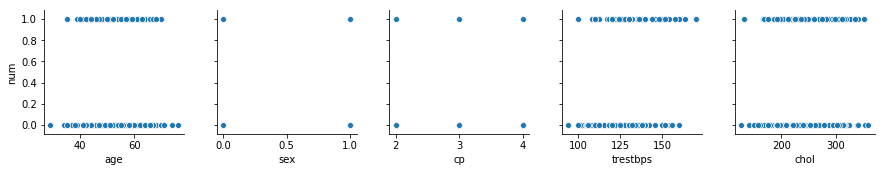
|  | **age** | **sex** | **cp** | **trestbps** | **chol** | **fbs** | **restecg** | **thalach** | **exang** | **oldpeak** | **slope** | **ca** | **thal** | **num** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **age** | 1.000000 | -0.068023 | 0.163936 | 0.244677 | 0.139946 | 0.082340 | 0.126770 | -0.403112 | 0.121760 | 0.213522 | 0.116500 | 0.396940 | 0.176240 | 0.225208 |
| **sex** | -0.068023 | 1.000000 | 0.140649 | -0.006177 | -0.144353 | 0.074238 | 0.009650 | -0.060405 | 0.182413 | 0.182307 | 0.059072 | 0.103310 | 0.423702 | 0.321612 |
| **cp** | 0.163936 | 0.140649 | 1.000000 | 0.059298 | -0.000694 | -0.029213 | 0.155978 | -0.364522 | 0.448742 | 0.340553 | 0.239560 | 0.265961 | 0.383182 | 0.491593 |
| **trestbps** | 0.244677 | -0.006177 | 0.059298 | 1.000000 | 0.074723 | 0.094153 | 0.142851 | -0.019425 | 0.020807 | 0.172846 | 0.035875 | 0.054343 | 0.096844 | 0.115971 |
| **chol** | 0.139946 | -0.144353 | -0.000694 | 0.074723 | 1.000000 | -0.064411 | 0.132662 | 0.020683 | 0.061549 | -0.015257 | -0.062850 | 0.091556 | -0.023407 | 0.069465 |
| **fbs** | 0.082340 | 0.074238 | -0.029213 | 0.094153 | -0.064411 | 1.000000 | 0.048060 | -0.030457 | 0.061977 | 0.019612 | 0.071831 | 0.132611 | 0.059046 | 0.053916 |
| **restecg** | 0.126770 | 0.009650 | 0.155978 | 0.142851 | 0.132662 | 0.048060 | 1.000000 | -0.133334 | 0.130339 | 0.143702 | 0.145192 | 0.105911 | 0.022200 | 0.202694 |
| **thalach** | -0.403112 | -0.060405 | -0.364522 | -0.019425 | 0.020683 | -0.030457 | -0.133334 | 1.000000 | -0.428763 | -0.365771 | -0.386295 | -0.255506 | -0.369852 | -0.409495 |
| **exang** | 0.121760 | 0.182413 | 0.448742 | 0.020807 | 0.061549 | 0.061977 | 0.130339 | -0.428763 | 1.000000 | 0.355118 | 0.311737 | 0.207449 | 0.377211 | 0.461310 |
| **oldpeak** | 0.213522 | 0.182307 | 0.340553 | 0.172846 | -0.015257 | 0.019612 | 0.143702 | -0.365771 | 0.355118 | 1.000000 | 0.534485 | 0.297772 | 0.339180 | 0.467442 |
| **slope** | 0.116500 | 0.059072 | 0.239560 | 0.035875 | -0.062850 | 0.071831 | 0.145192 | -0.386295 | 0.311737 | 0.534485 | 1.000000 | 0.088250 | 0.247547 | 0.344820 |
| **ca** | 0.396940 | 0.103310 | 0.265961 | 0.054343 | 0.091556 | 0.132611 | 0.105911 | -0.255506 | 0.207449 | 0.297772 | 0.088250 | 1.000000 | 0.292220 | 0.474565 |
| **thal** | 0.176240 | 0.423702 | 0.383182 | 0.096844 | -0.023407 | 0.059046 | 0.022200 | -0.369852 | 0.377211 | 0.339180 | 0.247547 | 0.292220 | 1.000000 | 0.592820 |
| **num** | 0.225208 | 0.321612 | 0.491593 | 0.115971 | 0.069465 | 0.053916 | 0.202694 | -0.409495 | 0.461310 | 0.467442 | 0.344820 | 0.474565 | 0.592820 | 1.000000 |

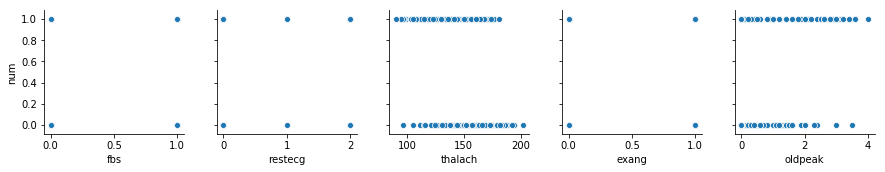
In [146]:

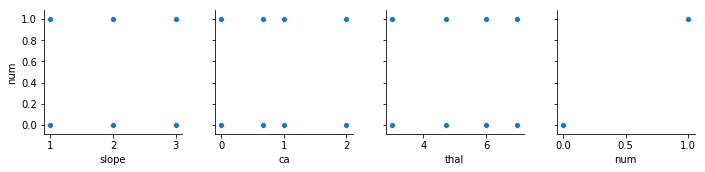
#correlation between columns plot after removing outliers

for i in range(0, len(dataframe.columns),5):

sns.pairplot(dataframe, y\_vars =['num'],x\_vars= dataframe.columns[i:i+5])







In [147]:

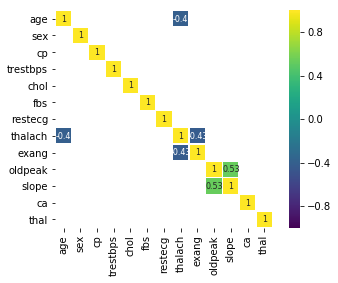
#plotting significant corr in one plot (heatmap)

corr = dataframe.drop('num', axis = 1).corr()

sns.heatmap(corr[(corr>=0.5) | (corr<=-0.4)],

cmap='viridis', vmax =1.0, vmin=-1.0, linewidths = 0.5,

annot =True, annot\_kws = {"size":8}, square =True);



In [148]:

#number of rows and columns in datarame

dataframe.shape

Out[148]:

(245, 14)

In [173]:

import numpy as np

from sklearn.model\_selection import train\_test\_split as tts

import statsmodels.api as sm

X\_train, X\_test, Y\_train, Y\_test = tts(dataframe.drop('num', axis = 1), dataframe['num'], test\_size = 0.2, random\_state = 243)

In [174]:

X\_train = sm.add\_constant(X\_train)

model = sm.GLM(Y\_train, X\_train, family = sm.families.Binomial())

result = model.fit()

In [175]:

print(result.summary2())

Results: Generalized linear model

==============================================================

Model: GLM AIC: 148.3148

Link Function: logit BIC: -840.3021

Dependent Variable: num Log-Likelihood: -60.157

Date: 2019-03-28 00:42 LL-Null: -134.62

No. Observations: 196 Deviance: 120.31

Df Model: 13 Pearson chi2: 146.

Df Residuals: 182 Scale: 1.0000

Method: IRLS

--------------------------------------------------------------

Coef. Std.Err. z P>|z| [0.025 0.975]

--------------------------------------------------------------

const -13.2826 4.6478 -2.8578 0.0043 -22.3921 -4.1731

age -0.0192 0.0325 -0.5896 0.5555 -0.0829 0.0446

sex 1.5057 0.6514 2.3115 0.0208 0.2290 2.7825

cp 0.7358 0.3497 2.1037 0.0354 0.0503 1.4213

trestbps 0.0324 0.0176 1.8399 0.0658 -0.0021 0.0669

chol 0.0120 0.0061 1.9770 0.0480 0.0001 0.0239

fbs -0.0366 0.8399 -0.0436 0.9652 -1.6828 1.6095

restecg 0.3769 0.2531 1.4894 0.1364 -0.1191 0.8730

thalach -0.0081 0.0157 -0.5135 0.6076 -0.0389 0.0228

exang 0.8509 0.5931 1.4347 0.1514 -0.3115 2.0134

oldpeak 0.3400 0.3045 1.1165 0.2642 -0.2568 0.9368

slope 0.5133 0.4934 1.0402 0.2982 -0.4538 1.4804

ca 1.9511 0.4737 4.1184 0.0000 1.0225 2.8796

thal 0.4555 0.1468 3.1033 0.0019 0.1678 0.7432

==============================================================

In [176]:

X\_train.drop(['fbs'], inplace = True, axis = 1)

model = sm.GLM(Y\_train, X\_train, family = sm.families.Binomial())

result = model.fit()

In [177]:

print(result.summary2())

Results: Generalized linear model

==============================================================

Model: GLM AIC: 146.3167

Link Function: logit BIC: -845.5783

Dependent Variable: num Log-Likelihood: -60.158

Date: 2019-03-28 00:42 LL-Null: -134.62

No. Observations: 196 Deviance: 120.32

Df Model: 12 Pearson chi2: 146.

Df Residuals: 183 Scale: 1.0000

Method: IRLS

--------------------------------------------------------------

Coef. Std.Err. z P>|z| [0.025 0.975]

--------------------------------------------------------------

const -13.2731 4.6403 -2.8604 0.0042 -22.3680 -4.1783

age -0.0192 0.0325 -0.5892 0.5557 -0.0829 0.0446

sex 1.5055 0.6516 2.3105 0.0209 0.2284 2.7825

cp 0.7381 0.3456 2.1361 0.0327 0.0609 1.4154

trestbps 0.0323 0.0175 1.8480 0.0646 -0.0020 0.0665

chol 0.0120 0.0061 1.9833 0.0473 0.0001 0.0239

restecg 0.3762 0.2524 1.4903 0.1362 -0.1186 0.8709

thalach -0.0081 0.0157 -0.5181 0.6044 -0.0389 0.0226

exang 0.8478 0.5883 1.4411 0.1496 -0.3053 2.0008

oldpeak 0.3395 0.3043 1.1159 0.2644 -0.2568 0.9359

slope 0.5118 0.4920 1.0402 0.2982 -0.4525 1.4761

ca 1.9469 0.4636 4.1990 0.0000 1.0381 2.8556

thal 0.4555 0.1468 3.1035 0.0019 0.1678 0.7431

==============================================================

In [178]:

X\_train.drop(['trestbps'], inplace = True, axis = 1)

model = sm.GLM(Y\_train, X\_train, family = sm.families.Binomial())

result = model.fit()

In [179]:

print(result.summary2())

Results: Generalized linear model

==============================================================

Model: GLM AIC: 147.8757

Link Function: logit BIC: -847.2974

Dependent Variable: num Log-Likelihood: -61.938

Date: 2019-03-28 00:42 LL-Null: -134.62

No. Observations: 196 Deviance: 123.88

Df Model: 11 Pearson chi2: 137.

Df Residuals: 184 Scale: 1.0000

Method: IRLS

---------------------------------------------------------------

Coef. Std.Err. z P>|z| [0.025 0.975]

---------------------------------------------------------------

const -10.5346 4.2713 -2.4663 0.0137 -18.9062 -2.1629

age -0.0012 0.0310 -0.0396 0.9684 -0.0619 0.0594

sex 1.2467 0.6215 2.0059 0.0449 0.0285 2.4649

cp 0.7328 0.3403 2.1536 0.0313 0.0659 1.3997

chol 0.0110 0.0059 1.8656 0.0621 -0.0006 0.0226

restecg 0.4194 0.2489 1.6847 0.0920 -0.0685 0.9073

thalach -0.0016 0.0150 -0.1064 0.9153 -0.0309 0.0277

exang 0.7849 0.5828 1.3466 0.1781 -0.3575 1.9272

oldpeak 0.4610 0.2986 1.5438 0.1226 -0.1243 1.0462

slope 0.3558 0.4708 0.7558 0.4498 -0.5669 1.2785

ca 1.7472 0.4364 4.0041 0.0001 0.8920 2.6024

thal 0.4782 0.1459 3.2772 0.0010 0.1922 0.7642

==============================================================

Looking at the P values not all values are significant we can eliminate each of the non significant columns by backward step wise elimination, we would start with the value 'age'

In [180]:

X\_train.drop(['age'], inplace = True, axis = 1)

model = sm.GLM(Y\_train, X\_train, family = sm.families.Binomial())

result = model.fit()

In [181]:

print(result.summary2())

Results: Generalized linear model

==============================================================

Model: GLM AIC: 145.8773

Link Function: logit BIC: -852.5739

Dependent Variable: num Log-Likelihood: -61.939

Date: 2019-03-28 00:42 LL-Null: -134.62

No. Observations: 196 Deviance: 123.88

Df Model: 10 Pearson chi2: 137.

Df Residuals: 185 Scale: 1.0000

Method: IRLS

---------------------------------------------------------------

Coef. Std.Err. z P>|z| [0.025 0.975]

---------------------------------------------------------------

const -10.6220 3.6578 -2.9039 0.0037 -17.7911 -3.4528

sex 1.2482 0.6203 2.0124 0.0442 0.0325 2.4639

cp 0.7334 0.3398 2.1583 0.0309 0.0674 1.3994

chol 0.0110 0.0059 1.8730 0.0611 -0.0005 0.0225

restecg 0.4200 0.2484 1.6907 0.0909 -0.0669 0.9069

thalach -0.0014 0.0143 -0.0992 0.9210 -0.0294 0.0266

exang 0.7874 0.5792 1.3596 0.1740 -0.3478 1.9227

oldpeak 0.4611 0.2986 1.5440 0.1226 -0.1242 1.0464

slope 0.3559 0.4709 0.7557 0.4498 -0.5671 1.2788

ca 1.7430 0.4230 4.1207 0.0000 0.9139 2.5720

thal 0.4778 0.1456 3.2815 0.0010 0.1924 0.7632

==============================================================

In [182]:

X\_train.drop(['thalach'], inplace = True, axis = 1)

model = sm.GLM(Y\_train, X\_train, family = sm.families.Binomial())

result = model.fit()

print(result.summary2())

Results: Generalized linear model

==============================================================

Model: GLM AIC: 143.8871

Link Function: logit BIC: -857.8422

Dependent Variable: num Log-Likelihood: -61.944

Date: 2019-03-28 00:42 LL-Null: -134.62

No. Observations: 196 Deviance: 123.89

Df Model: 9 Pearson chi2: 137.

Df Residuals: 186 Scale: 1.0000

Method: IRLS

---------------------------------------------------------------

Coef. Std.Err. z P>|z| [0.025 0.975]

---------------------------------------------------------------

const -10.8981 2.3872 -4.5652 0.0000 -15.5768 -6.2193

sex 1.2329 0.6001 2.0543 0.0399 0.0566 2.4091

cp 0.7387 0.3357 2.2004 0.0278 0.0807 1.3966

chol 0.0110 0.0059 1.8751 0.0608 -0.0005 0.0225

restecg 0.4232 0.2465 1.7169 0.0860 -0.0599 0.9063

exang 0.7994 0.5666 1.4108 0.1583 -0.3111 1.9099

oldpeak 0.4661 0.2945 1.5826 0.1135 -0.1111 1.0433

slope 0.3640 0.4633 0.7857 0.4321 -0.5441 1.2721

ca 1.7518 0.4141 4.2300 0.0000 0.9401 2.5636

thal 0.4835 0.1342 3.6014 0.0003 0.2204 0.7466

==============================================================

In [183]:

X\_train.drop(['sex'], inplace = True, axis = 1)

model = sm.GLM(Y\_train, X\_train, family = sm.families.Binomial())

result = model.fit()

print(result.summary2())

Results: Generalized linear model

==============================================================

Model: GLM AIC: 146.3694

Link Function: logit BIC: -858.6381

Dependent Variable: num Log-Likelihood: -64.185

Date: 2019-03-28 00:42 LL-Null: -134.62

No. Observations: 196 Deviance: 128.37

Df Model: 8 Pearson chi2: 161.

Df Residuals: 187 Scale: 1.0000

Method: IRLS

---------------------------------------------------------------

Coef. Std.Err. z P>|z| [0.025 0.975]

---------------------------------------------------------------

const -9.6394 2.2093 -4.3631 0.0000 -13.9695 -5.3093

cp 0.7337 0.3321 2.2091 0.0272 0.0828 1.3846

chol 0.0077 0.0054 1.4196 0.1557 -0.0029 0.0184

restecg 0.4886 0.2440 2.0022 0.0453 0.0103 0.9669

exang 0.7771 0.5479 1.4184 0.1561 -0.2967 1.8509

oldpeak 0.5594 0.2934 1.9068 0.0565 -0.0156 1.1344

slope 0.2628 0.4483 0.5862 0.5577 -0.6158 1.1414

ca 1.6536 0.4000 4.1343 0.0000 0.8697 2.4375

thal 0.5870 0.1269 4.6273 0.0000 0.3384 0.8356

==============================================================

In [184]:

X\_train.drop(['exang'], inplace = True, axis = 1)

model = sm.GLM(Y\_train, X\_train, family = sm.families.Binomial())

result = model.fit()

print(result.summary2())

Results: Generalized linear model

==============================================================

Model: GLM AIC: 146.3765

Link Function: logit BIC: -861.9091

Dependent Variable: num Log-Likelihood: -65.188

Date: 2019-03-28 00:42 LL-Null: -134.62

No. Observations: 196 Deviance: 130.38

Df Model: 7 Pearson chi2: 168.

Df Residuals: 188 Scale: 1.0000

Method: IRLS

---------------------------------------------------------------

Coef. Std.Err. z P>|z| [0.025 0.975]

---------------------------------------------------------------

const -10.3538 2.1536 -4.8076 0.0000 -14.5748 -6.1327

cp 0.9022 0.3139 2.8744 0.0040 0.2870 1.5173

chol 0.0076 0.0053 1.4347 0.1514 -0.0028 0.0180

restecg 0.5197 0.2421 2.1464 0.0318 0.0451 0.9942

oldpeak 0.5795 0.2884 2.0091 0.0445 0.0142 1.1448

slope 0.3728 0.4374 0.8522 0.3941 -0.4845 1.2301

ca 1.7235 0.4025 4.2818 0.0000 0.9346 2.5124

thal 0.6206 0.1249 4.9693 0.0000 0.3758 0.8654

==============================================================

In [185]:

X\_train.drop(['restecg'], inplace = True, axis = 1)

model = sm.GLM(Y\_train, X\_train, family = sm.families.Binomial())

result = model.fit()

print(result.summary2())

Results: Generalized linear model

==============================================================

Model: GLM AIC: 149.2158

Link Function: logit BIC: -862.3479

Dependent Variable: num Log-Likelihood: -67.608

Date: 2019-03-28 00:42 LL-Null: -134.62

No. Observations: 196 Deviance: 135.22

Df Model: 6 Pearson chi2: 177.

Df Residuals: 189 Scale: 1.0000

Method: IRLS

---------------------------------------------------------------

Coef. Std.Err. z P>|z| [0.025 0.975]

---------------------------------------------------------------

const -10.6644 2.1367 -4.9909 0.0000 -14.8523 -6.4764

cp 1.0186 0.3124 3.2610 0.0011 0.4064 1.6309

chol 0.0098 0.0052 1.8956 0.0580 -0.0003 0.0199

oldpeak 0.5571 0.2754 2.0231 0.0431 0.0174 1.0969

slope 0.5101 0.4199 1.2147 0.2245 -0.3129 1.3331

ca 1.6983 0.3978 4.2697 0.0000 0.9187 2.4779

thal 0.5685 0.1169 4.8651 0.0000 0.3395 0.7975

==============================================================

In [186]:

X\_train.drop(['oldpeak'], inplace = True, axis = 1)

model = sm.GLM(Y\_train, X\_train, family = sm.families.Binomial())

result = model.fit()

print(result.summary2())

Results: Generalized linear model

==============================================================

Model: GLM AIC: 151.4481

Link Function: logit BIC: -863.3937

Dependent Variable: num Log-Likelihood: -69.724

Date: 2019-03-28 00:42 LL-Null: -134.62

No. Observations: 196 Deviance: 139.45

Df Model: 5 Pearson chi2: 175.

Df Residuals: 190 Scale: 1.0000

Method: IRLS

---------------------------------------------------------------

Coef. Std.Err. z P>|z| [0.025 0.975]

---------------------------------------------------------------

const -11.0506 2.0875 -5.2937 0.0000 -15.1420 -6.9592

cp 1.0967 0.3056 3.5884 0.0003 0.4977 1.6957

chol 0.0093 0.0050 1.8438 0.0652 -0.0006 0.0191

slope 0.9288 0.3605 2.5768 0.0100 0.2223 1.6353

ca 1.7060 0.3800 4.4892 0.0000 0.9612 2.4509

thal 0.5747 0.1140 5.0414 0.0000 0.3513 0.7981

==============================================================

In [187]:

from sklearn.metrics import accuracy\_score

from sklearn.metrics import log\_loss

from sklearn.metrics import confusion\_matrix

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

from sklearn.metrics import f1\_score

from sklearn.metrics import roc\_curve, roc\_auc\_score

In [188]:

X\_test = sm.add\_constant(X\_test)

X\_test.drop(['age','sex','trestbps','fbs','restecg','thalach','exang','oldpeak'], inplace = True, axis = 1)

probabilities = result.predict(X\_test)

predictions = probabilities.map(lambda x: 1 if x > 0.5 else 0)

predictions.head()

Out[188]:

160 0

177 1

102 1

42 1

56 1

dtype: int64

In [189]:

accuracy\_score(Y\_test, predictions)

Out[189]:

0.8367346938775511

In [190]:

log\_loss(Y\_test,predictions)

Out[190]:

5.639098129414222

Our Accuracy Score and log loss values have been very good, accuracy of 84% without considering the ROC is itself a pretty good model, let us look at confusion matrix and f1 score to get a understanding of our model performance

In [191]:

confusion\_mat = confusion\_matrix(Y\_test, predictions)

confusion\_df = pd.DataFrame(confusion\_mat, index=['Actual neg(0)', 'Actual pos(1)'],

columns = ['Predicted neg(0)', 'Predicted pos(1)'])

In [192]:

confusion\_df

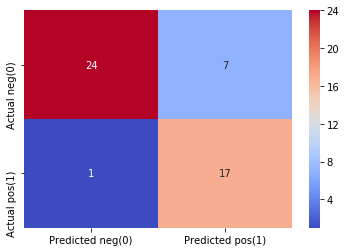
Out[192]:

|  | **Predicted neg(0)** | **Predicted pos(1)** |
| --- | --- | --- |
| **Actual neg(0)** | 24 | 7 |
| **Actual pos(1)** | 1 | 17 |

CONFUSION MATRIX

In [193]:

\_=sns.heatmap(confusion\_df, cmap = 'coolwarm', annot = True)



In [194]:

f1\_score(Y\_test, predictions)

Out[194]:

0.8095238095238096

By looking at the heat map of confusion matrix, we can see that the True positives and True negatives are higher in number which tell us its good model. f1\_score is 0.809 is also a indicator of good model

now lets plot the ROC curve and update our predictions with the optimal threshold to see if the any thing improves

In [195]:

auc = roc\_auc\_score(Y\_test, predictions)

In [196]:

auc

Out[196]:

0.8593189964157706

By looking at the classification report we can say the model is un-biased

In [197]:

#classification report

from sklearn import metrics #Import scikit-learn metrics module for accuracy calculation

cr = metrics.classification\_report(Y\_test, predictions)

print(cr)

precision recall f1-score support

0 0.96 0.77 0.86 31

1 0.71 0.94 0.81 18

micro avg 0.84 0.84 0.84 49

macro avg 0.83 0.86 0.83 49

weighted avg 0.87 0.84 0.84 49

By looking at the area under the ROC curve which is 0.859 , which is high reconfirms the goodness pf the fit of the model, if area under the curve is less than 1 then it is a perfect model

In [198]:

fpr, tpr, threshold = roc\_curve(Y\_test, probabilities)

plt.title('ROC CURVE')

plt.plot(fpr, tpr, 'b', label = 'AUC = %0.3f' % auc)

plt.legend(loc = 'lower right')

plt.plot([0,1],[0,1], 'r--')

plt.xlim([0,1])

plt.ylim([0,1])

plt.ylabel('True Positive Rate')

plt.ylabel('False Positive Rate')

plt.show()

