Unified Mentor Data Analytics Internship

Name: Suman Das

Project: Heart Disease Diagnostic Analysis

Problem Statement

Health is real wealth in the pandemic time we all realized the brute effects of covid-19 on all irrespective of any status. Required to analyze this health and medical data for better future preparation. Extract- Transform and Load data from the heart disease diagnostic database. Perform EDA through python. The database extracts various information such as Heart disease rates, Heart disease by gender, by age. Compare attributes of the data set to extract necessary information. Find key metrics and factors and show the meaningful relationships between attributes.

In [1]:
```
# import libraries

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

# ignore any error type warnings
import warnings
warnings.filterwarnings('ignore')

# suppressing any scientific notations
np.set_printoptions(suppress= True)

# controlling over maximum rows and columns to be displayed
pd.options.display.max_rows= 100
pd.options.display.max_columns= 50
```

Load the Dataset

In [2]:
```
# read the Heart Disease (HD) csv file
HD = pd.read_csv(filepath_or_buffer= 'Heart Disease data.csv', encoding='la
tin-1', sep=',')

# total no of rows and columns
print('no of rows and columns in the dataset: ', HD.shape)

# check duplicate values in the dataset
HD = HD.drop_duplicates()
print('no of rows and columns in the dataset after removing duplicate rows:
', HD.shape)
# there are some duplicate values here. I remove those values.

print('*'*50)

# print first 50 rows of the dataset to see a quick glance
HD.head(10)
```

```
no of rows and columns in the dataset:  (1025, 14)
no of rows and columns in the dataset after removing duplicate rows:  (30
2, 14)
**************************************************
```

Out[2]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	targe
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3	
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	
5	58	0	0	100	248	0	0	122	0	1.0	1	0	2	
6	58	1	0	114	318	0	2	140	0	4.4	0	3	1	
7	55	1	0	160	289	0	0	145	1	0.8	1	1	3	
8	46	1	0	120	249	0	0	144	0	0.8	2	0	3	
9	54	1	0	122	286	0	0	116	1	3.2	1	2	2	

Data Description

In [3]:
```
# Columns Details

# age:      age of individual (continuous)
# sex:      sex of individual, 1 for male and 0 for female (categorical)
# cp:       chest pain type (4 levels of chest pain - Typical Angina, Atypi
cal Angina, Non-Anginal Pain, Asymptomatic) (categorical)
# trestbps: resting blood pressure (continuous)
# chol:     serum cholestoral (in mg/dl) (continuous)
# fbs:      fasting blood sugar (> 120 mg/dl), 0 for False, 1 for True (cat
egorical)
# restecg:  resting electrocardiographic results, values 0= normal,1= non-s
pecific disturbances,2= significant (categorical)
# thalach:  maximum heart rate achieved (continuous)
# exang:    exercise induced angina (0 for no, 1 for yes) (categorical)
# oldpeak:  ST depression induced by exercise relative to rest (continuous)
# slope:    the slope of the peak exercise ST segment (categorical)
# ca:       number of major vessels (0-4) colored by flourosopy (categorica
l)
# thal:     0 = normal; 1 = fixed defect; 2 = reversable defect (categorica
l)
# target:   0 = less chance for heart disease, 1 = more chance for heart di
sease (categorical) [TARGET VARIABLE]
```

Basic Exploratory Data Analysis

In [4]:
```
# find datatype, no of columns, no of non-null vales
HD.info()  # information about the dataframe
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 302 entries, 0 to 878
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       302 non-null    int64  
 1   sex       302 non-null    int64  
 2   cp        302 non-null    int64  
 3   trestbps  302 non-null    int64  
 4   chol      302 non-null    int64  
 5   fbs       302 non-null    int64  
 6   restecg   302 non-null    int64  
 7   thalach   302 non-null    int64  
 8   exang     302 non-null    int64  
 9   oldpeak   302 non-null    float64
 10  slope     302 non-null    int64  
 11  ca        302 non-null    int64  
 12  thal      302 non-null    int64  
 13  target    302 non-null    int64  
dtypes: float64(1), int64(13)
memory usage: 35.4 KB
```

In [5]:
```
# check total no of unique values under each variable
HD.nunique()
```

Out[5]:
```
age          41
sex           2
cp            4
trestbps     49
chol        152
fbs           2
restecg       3
thalach      91
exang         2
oldpeak      40
slope         3
ca            5
thal          4
target        2
dtype: int64
```

In [6]:
```
# find statistical values of each columns
HD.describe()
```

Out[6]:

	age	sex	cp	trestbps	chol	fbs	restecg	
count	302.00000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	30
mean	54.42053	0.682119	0.963576	131.602649	246.500000	0.149007	0.526490	14
std	9.04797	0.466426	1.032044	17.563394	51.753489	0.356686	0.526027	2
min	29.00000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	7
25%	48.00000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	13
50%	55.50000	1.000000	1.000000	130.000000	240.500000	0.000000	1.000000	15
75%	61.00000	1.000000	2.000000	140.000000	274.750000	0.000000	1.000000	16
max	77.00000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	20

In [7]:
```
# detect null values in the dataset
HD.isnull().sum()  # there is no null values in any columns
```

Out[7]:
```
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
target      0
dtype: int64
```

In [8]:
```
# print name of all columns present in dataset
HD.columns
```

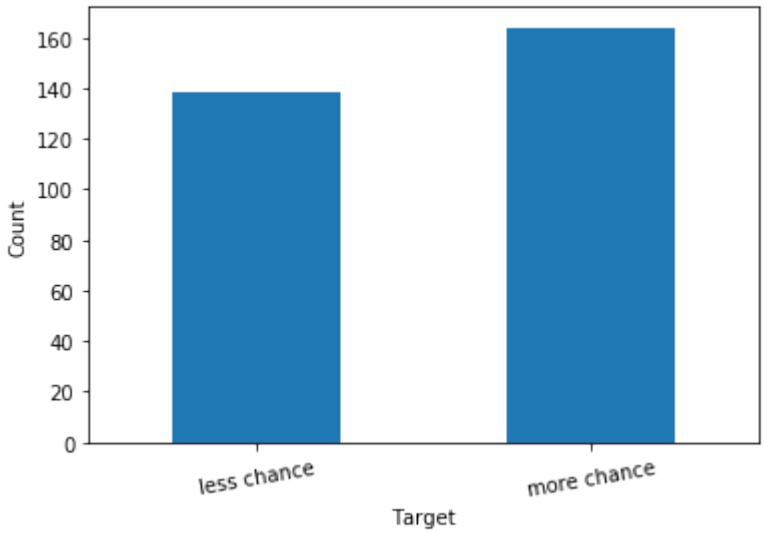
Out[8]:
```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalac
h',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')
```

Visual Exploratory Data Analysis

Now start the visualization part

In [9]:
```
# plot bar chart for target variable
# to check how many people have heart disesse or not
HD.groupby('target').size().plot(kind='bar')
plt.xticks([0,1], ['less chance', 'more chance'], rotation= 10)
plt.xlabel('Target', fontsize=10)
plt.ylabel('Count', fontsize=10)
```

Out[9]:
```
Text(0, 0.5, 'Count')
```



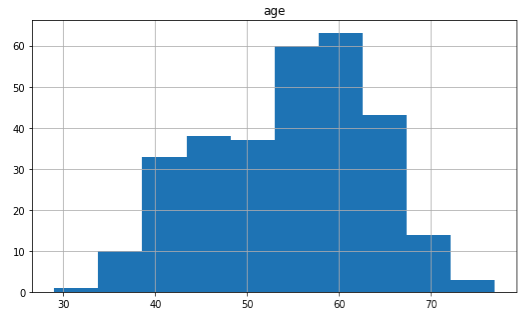
Here, both the bars under Target Variable have enough data to explain further analysis.

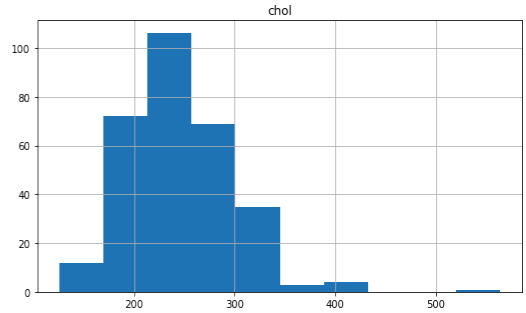
Continuous Variables - Histogram

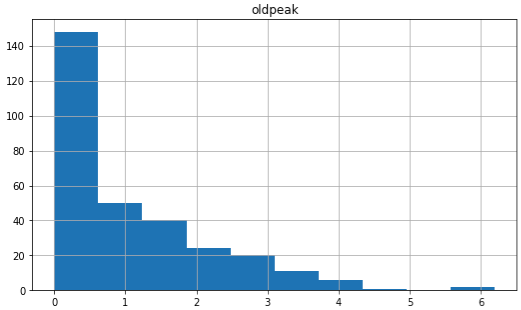
Those variables which are continuous in nature like Age, resting blood pressure, cholestoral require 'Histogram' type of chart to explain the distribution and check outliers.

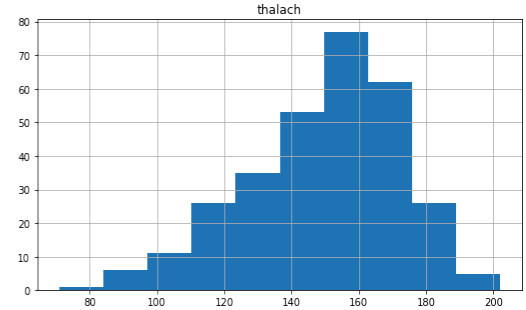
In [10]:
```
# plot the distribution of continuous variables
Continuous= ['age', 'trestbps', 'chol', 'thalach', 'oldpeak']
HD.hist(Continuous, figsize=(20,18))
```

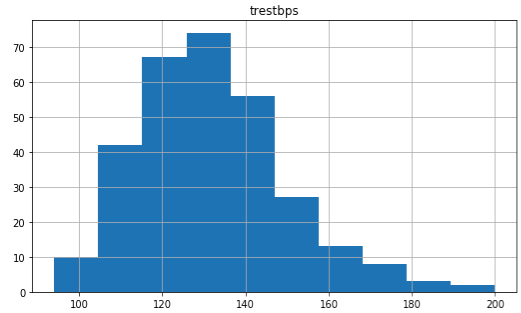
Out[10]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x000002327526228
8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x00000232752D244
8>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x000002327530B24
8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002327534438
8>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x000002327537C48
8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x00000232753B558
8>]],
      dtype=object)
```











Here, 'chol' and 'oldpeak' columns have outliers. Outliers are those data points which are far away from main dataset's all other data points and it is a kind of separated long thin tail kind of. Outliers can be treated as to replace those isolated datapoints with nearest logical data point which are close to rest of data points of dataset.

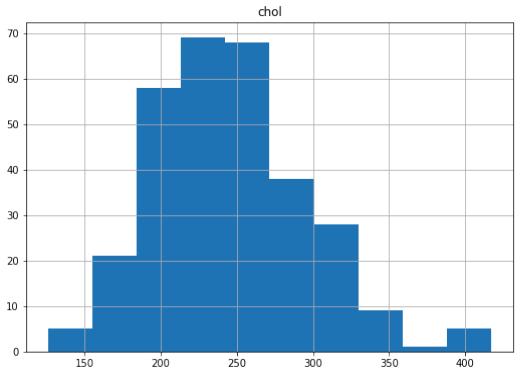
outliers treatment for continuous variable

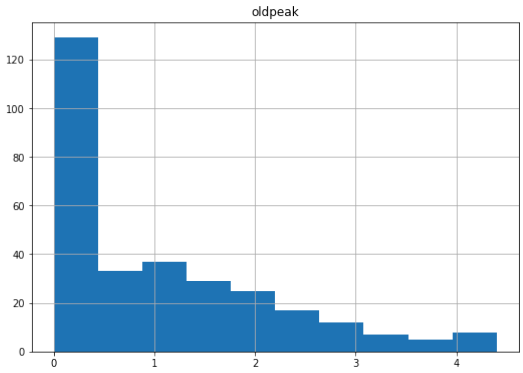
In [11]:
```
# cholestoral column has outliers, minimise the outliers
HD[HD['chol']>425].shape[0]
HD['chol'][HD['chol']<=425].sort_values(ascending= False)
HD['chol'][HD['chol']>425]= 417
```

In [12]:
```
# ST depression column has outliers, minimise the outliers
HD[HD['oldpeak']>5].shape[0]
HD['oldpeak'][HD['oldpeak']<=5].sort_values(ascending= False)
HD['oldpeak'][HD['oldpeak']>5]= 4.4
```

In [13]:
```
# plot again after treating the outliers
Continuous= ['chol', 'oldpeak']
HD.hist(Continuous, figsize=(20,6))
```

Out[13]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x000002327553778
8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000023275CFD3C
8>]],
      dtype=object)
```





Featured Engineering

Featured Engineering is a process to include a new column based on some existing columns and some conditions. Here I form a new column AgeGroup based on existing Age column to see which age group has a higher chance for occuring heart disease.

In [14]:
```
def FunctionAge(inpAge):
    if(inpAge <= 40):
        return 'young age'
    elif(inpAge <= 60):
        return 'middle age'
    else:
        return 'old age'

# Call the function
HD['agegroup'] = HD['age'].apply(FunctionAge)
HD.head()
```

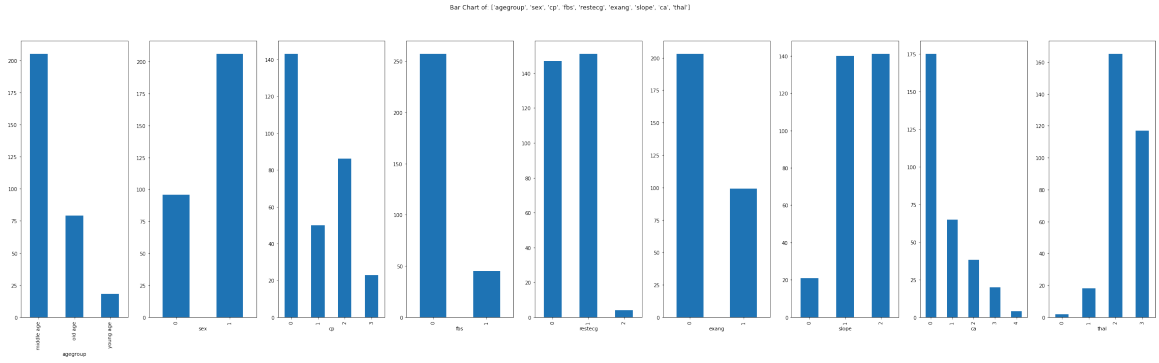
Out[14]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	targe
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3	
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	

Categorical Variables - Bar Chart

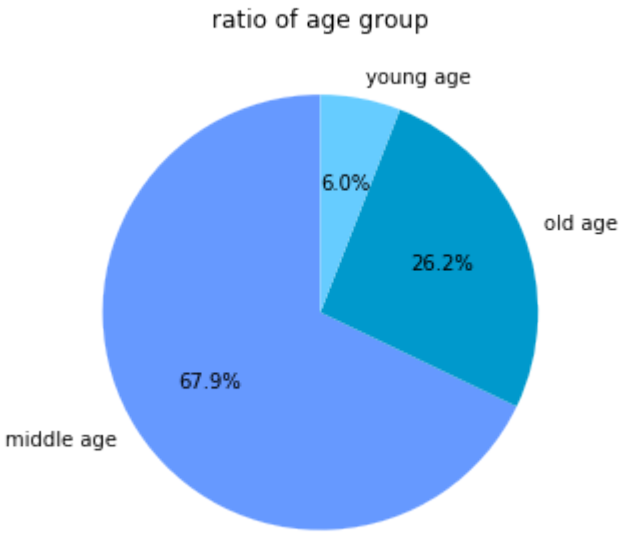
Those variables which are categorical in nature like agegroup, sex require 'Bar Chart' type of chart to explain the distribution

In [15]:
```
# plot the distribution of categorical variable
Categorical= ['agegroup','sex', 'cp', 'fbs', 'restecg', 'exang', 'slope',
'ca', 'thal']

fig, canvas= plt.subplots(nrows= 1, ncols= len(Categorical), figsize=(40,1
0))
fig.suptitle('Bar Chart of: '+ str(Categorical))

for predictor, i in zip(Categorical, range(len(Categorical))):
    HD.groupby(predictor).size().plot(kind= 'bar', ax= canvas[i])

# here all bars have sufficient information to explain, no such outliers ar
e present
```

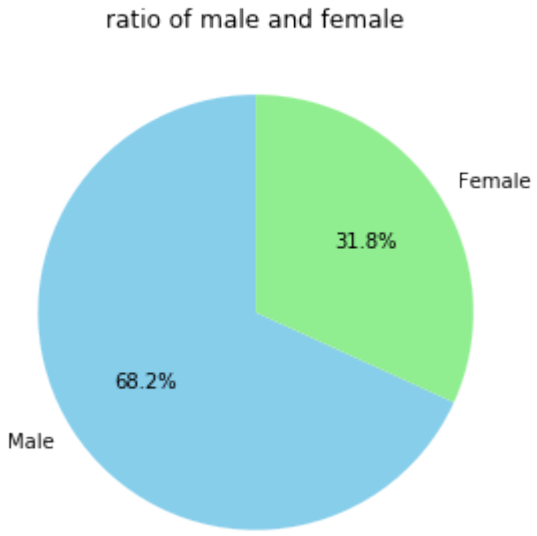


In [16]:
```
# focus on agegroup, sex, cp, fbs, restecg column
```

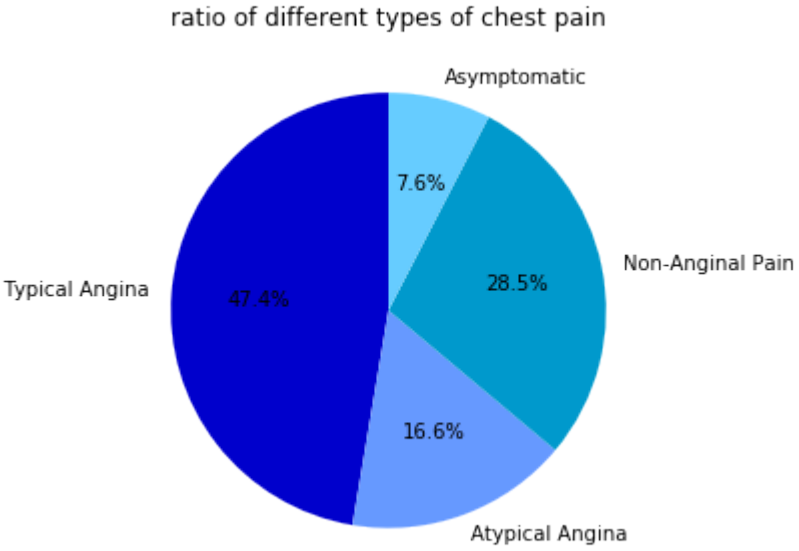
In [17]:
```
# extract data for pie chart
categories= HD['agegroup'].unique()
values= [HD['agegroup'].value_counts()['middle age'], HD['agegroup'].value_
counts()['old age'],
         HD['agegroup'].value_counts()['young age']]

# plot the pie chart
plt.figure(figsize=(5,5))
plt.pie(values, labels= categories, autopct= '%1.1f%%', startangle=90, colo
rs=['#6699FF','#0099CC','#66CCFF'])

plt.title('ratio of age group')
plt.show()
```



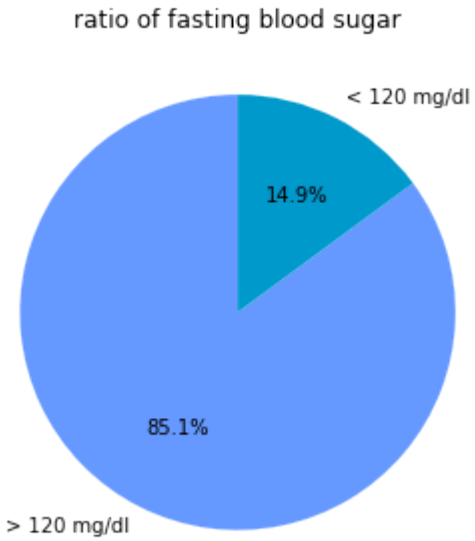
In [18]:
```
# extract data for pie chart
HD['sex']= HD['sex'].replace({1:'Male', 0:'Female'})
categories= HD['sex'].unique()
values= [HD['sex'].value_counts()['Male'], HD['sex'].value_counts()['Femal
e']]

# plot the pie chart
plt.figure(figsize=(5,5))
plt.pie(values, labels=categories, autopct='%1.1f%%', startangle=90, colors
=['skyblue','lightgreen'])

plt.title('ratio of male and female')
plt.show()
```



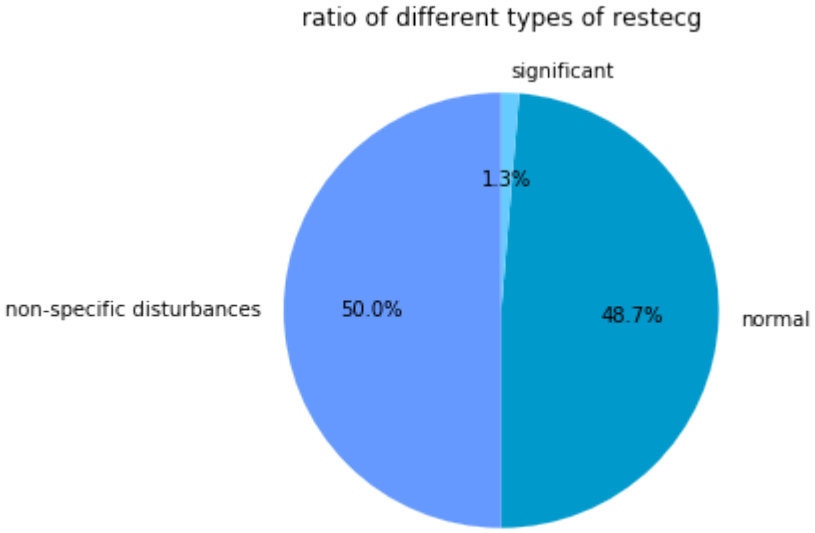
In [19]:
```
# extract data for pie chart
HD['cp']= HD['cp'].replace({0:'Typical Angina', 1:'Atypical Angina', 2:'Non
-Anginal Pain', 3:'Asymptomatic'})
categories= HD['cp'].unique()
values= [HD['cp'].value_counts()['Typical Angina'], HD['cp'].value_counts()
['Atypical Angina'],
         HD['cp'].value_counts()['Non-Anginal Pain'], HD['cp'].value_counts
()['Asymptomatic']]

# plot the pie chart
plt.figure(figsize=(5,5))
plt.pie(values, labels=categories, autopct='%1.1f%%', startangle=90, colors
=['#0000CC','#6699FF','#0099CC','#66CCFF'])

plt.title('ratio of different types of chest pain')
plt.show()
```



In [20]:
```
# extract data for pie chart
HD['fbs']= HD['fbs'].replace({0:'> 120 mg/dl', 1:'< 120 mg/dl'})
categories= HD['fbs'].unique()
values= [HD['fbs'].value_counts()['> 120 mg/dl'], HD['fbs'].value_counts()
['< 120 mg/dl']]

# plot the pie chart
plt.figure(figsize=(5,5))
plt.pie(values, labels=categories, autopct='%1.1f%%', startangle=90, colors
=['#6699FF','#0099CC'])

plt.title('ratio of fasting blood sugar')
plt.show()
```



In [21]:
```
# extract data for pie chart
HD['restecg']= HD['restecg'].replace({0:'normal', 1:'non-specific disturban
ces', 2:'significant'})
categories= HD['restecg'].unique()
values= [HD['restecg'].value_counts()['non-specific disturbances'], HD['res
tecg'].value_counts()['normal'],
         HD['restecg'].value_counts()['significant']]

# plot the pie chart
plt.figure(figsize=(5,5))
plt.pie(values, labels=categories, autopct='%1.1f%%', startangle=90, colors
=['#6699FF','#0099CC','#66CCFF'])

plt.title('ratio of different types of restecg')
plt.show()
```

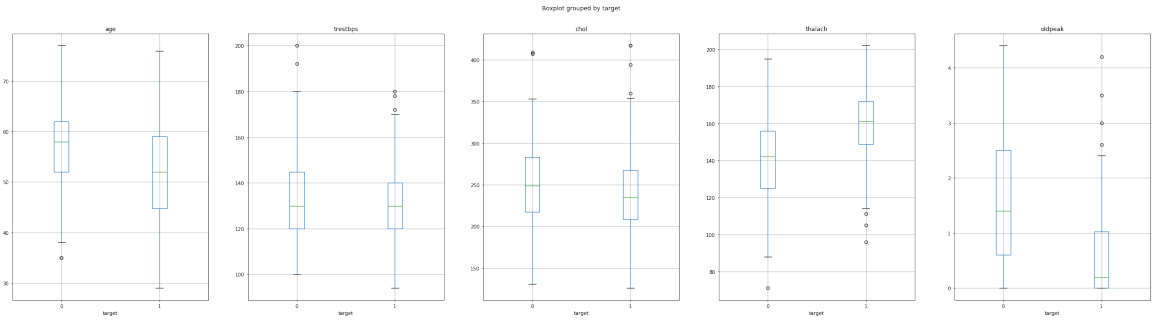


Feature Selection -- Bi-Variate Analysis

distribution of continuous variables according to categorical variable

Here, Target Variable is Categorical one. So, for Continuous variable vs Categorical Variable situation, box plot can be good option to analysis. If the boxes are mis-aligned, then that particular column has a correlation with target column, so they are good for Machine Learning analysis, otherwise not. One can detect this more accurately by using statistical Anova Test. If Anova Test Result, p-value is less than 0.05, then one can consider this variable for Machine learning analysis.

In [22]:
```
# to show the distribution of Continuous variables as per Categorical Targe
t Variable
# box plot - best way to explain

Continuous= ['age', 'trestbps', 'chol', 'thalach', 'oldpeak']

fig, canvas= plt.subplots(nrows= 1, ncols= len(Continuous), figsize= (40,1
0))

for predictor, i in zip(Continuous, range(len(Continuous))):
    HD.boxplot(column= predictor, by= 'target', vert= True, ax= canvas[i])
```



In [23]:
```
# ANOVA Test - statistical test to explained box plot diagram to find best
machine learning oriented columns

from scipy.stats import f_oneway

def FunctionAnova(inpData, TargetVariable, PredictorList):

    FinalPredictor=[]

    for predictor in PredictorList:
        AnovaTest= inpData.groupby(TargetVariable)[predictor].apply(list)
        AnovaResult= f_oneway(*AnovaTest)

        if(AnovaResult[1] < 0.05):
            print(predictor, 'is correlated with ', TargetVariable, '|| P-V
alue: ', AnovaResult[1])
            FinalPredictor.append(predictor)

        else:
            print(predictor, 'is NOT correlated with ', TargetVariable, '||
P-Value: ', AnovaResult[1])

    return(FinalPredictor)
```

In [24]:
```
# call the function

Continuous= ['age', 'trestbps', 'chol', 'thalach', 'oldpeak']
FunctionAnova(inpData = HD, TargetVariable = 'target', PredictorList = Cont
inuous)
```

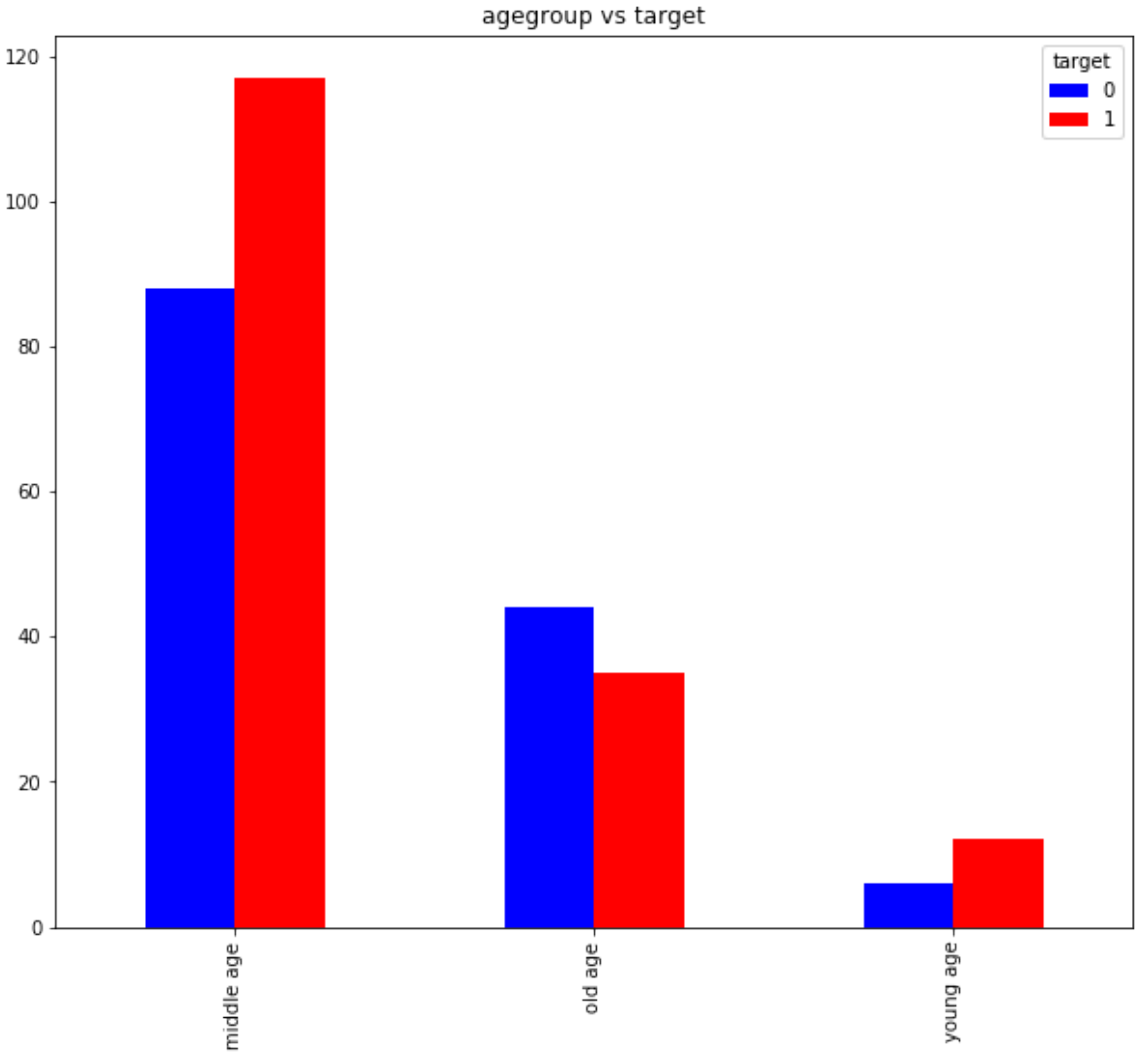
```
age is correlated with  target || P-Value:  0.00010394837285417
trestbps is correlated with  target || P-Value:  0.010926538861949038
chol is NOT correlated with  target || P-Value:  0.10162875184380436
thalach is correlated with  target || P-Value:  2.4761460479234675e-14
oldpeak is correlated with  target || P-Value:  2.98895776437653e-15
```

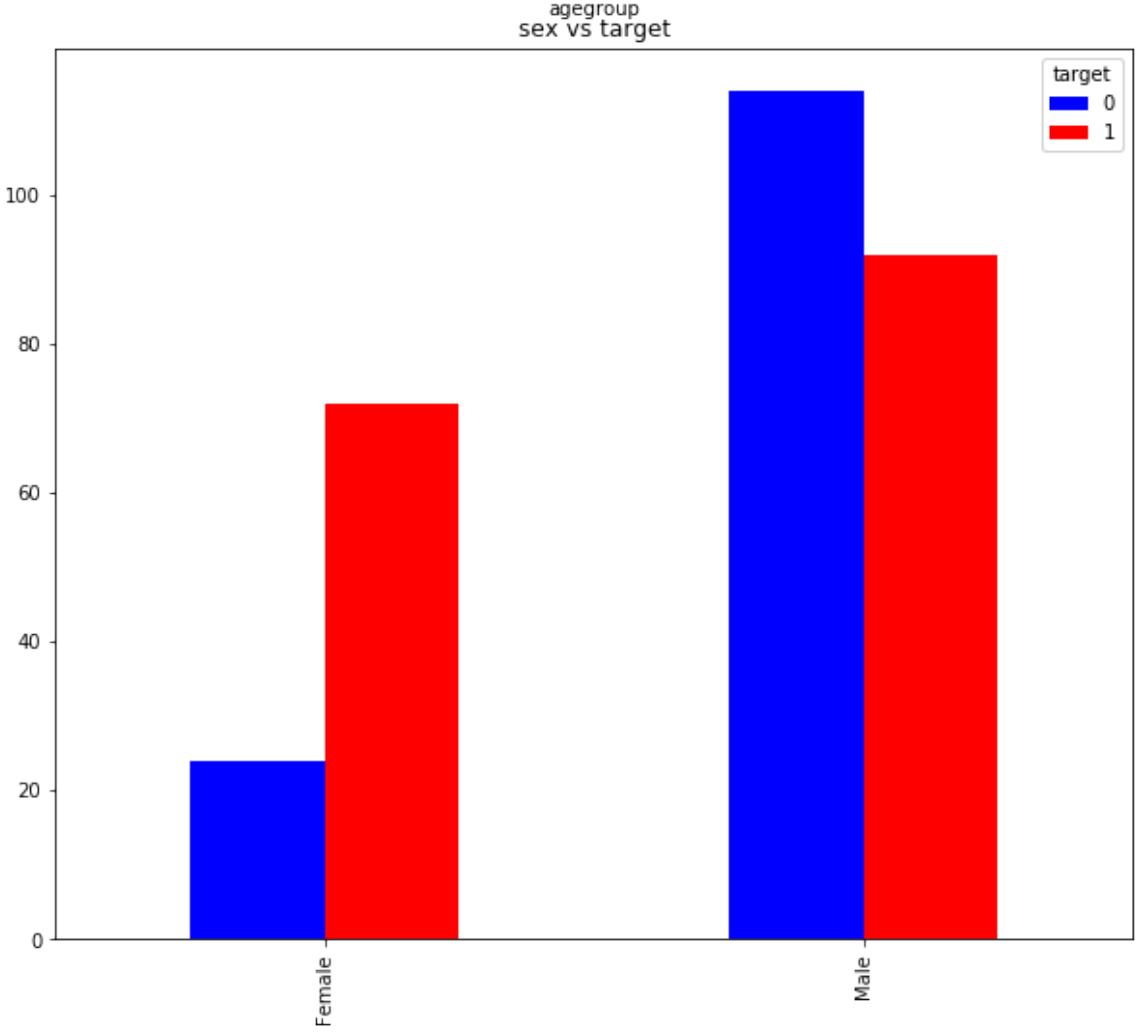
Out[24]:
```
['age', 'trestbps', 'thalach', 'oldpeak']
```

distribution of categorical variables according to categorical variable

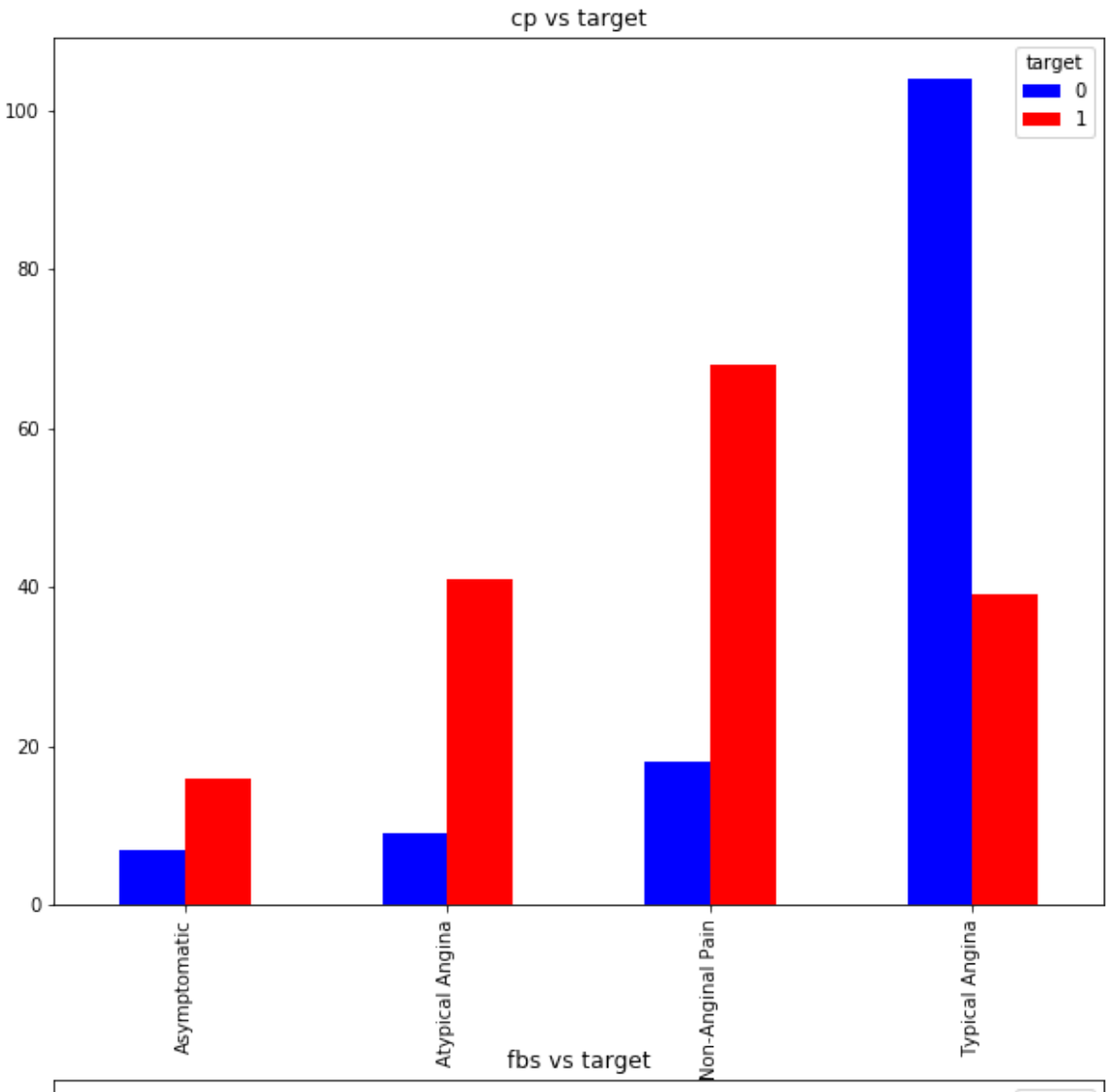
Here, Target Variable is Categorical one. So, for Categorical variable vs Categorical Variable situation, grouped bar chart can be a good option to analysis. If the ratios of cross-tab are different, then that particular column has correlation with target column, so they are good for Machine Learning analysis, otherwise not. One can detect this more accurately by using statistical Chi-Square Test. If Chi-Square Test Result, p-value is less than 0.05, then one can consider this variable for Machine learning analysis.

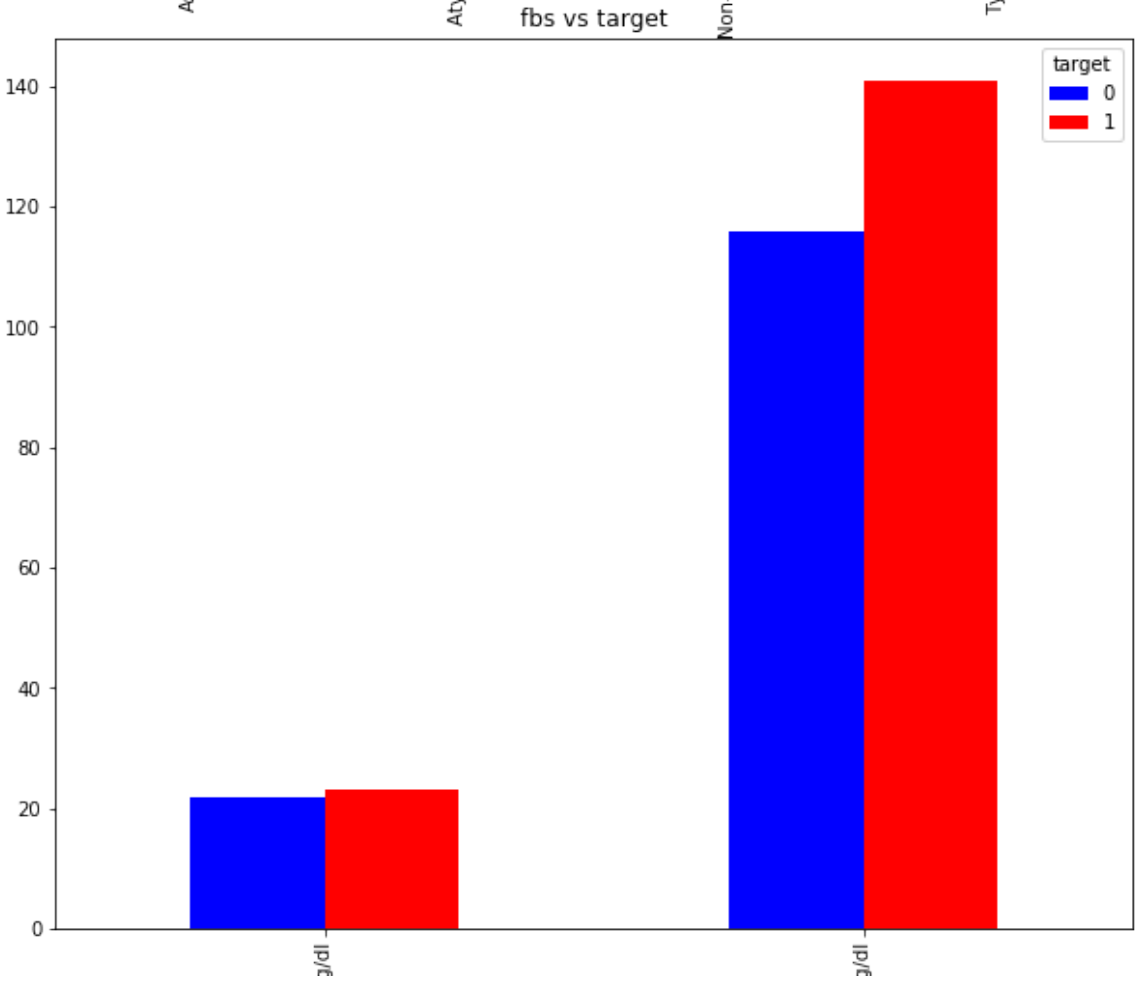
In [25]:
```
# to show the distribution of Categorical variables as per Categorical Targ
et Variable
# grouped bar chart - best way to explain

Categorical= ['agegroup','sex', 'cp', 'fbs', 'restecg', 'exang', 'slope',
'ca', 'thal']

fig, canvas = plt.subplots(ncols=1, nrows= len(Categorical), figsize=(10,9
0))

for predictor, i in zip(Categorical, range(len(Categorical))):
    CrossTabResult= pd.crosstab(index= HD[predictor], columns= HD['targe
t'])
    CrossTabResult.plot(kind='bar', color= ['blue', 'red'], ax= canvas[i],
title= predictor + ' vs '+ 'target')
```


agegroup vs target
target
0
1
120
100
80
60
40
20
0
middle age
old age
young age
agegroup


sex vs target
target
0
1
100
80
60
40
20
0
Female
Male

sex


cp vs target
target
0
1
0
20
40
60
80
100
Asymptomatic
Atypical Angina
Non-Anginal Pain
Typical Angina

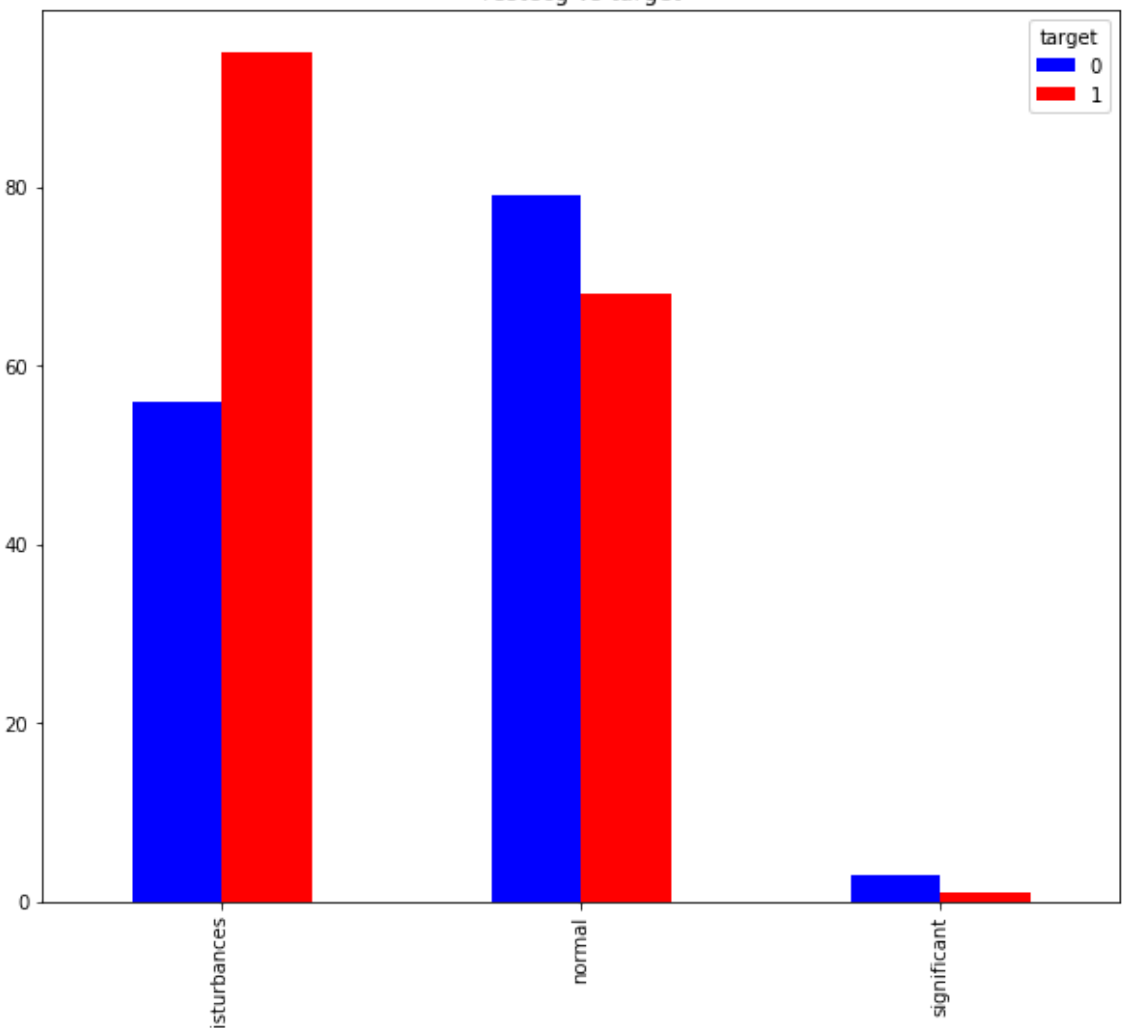

fbs vs target
target
0
1
0
20
40
60
80
100
120
140

< 120 m

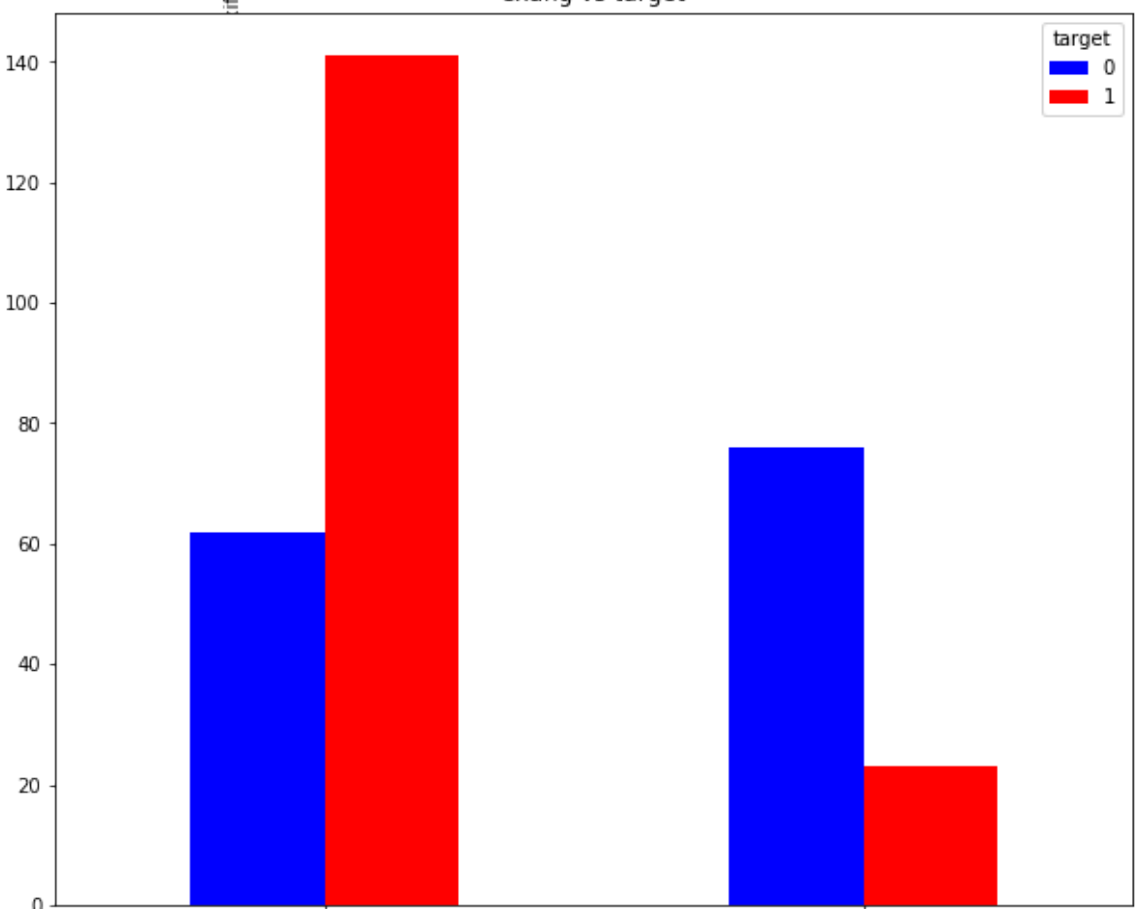
fbs

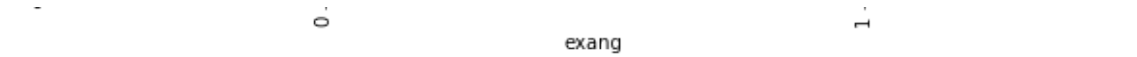
> 120 m

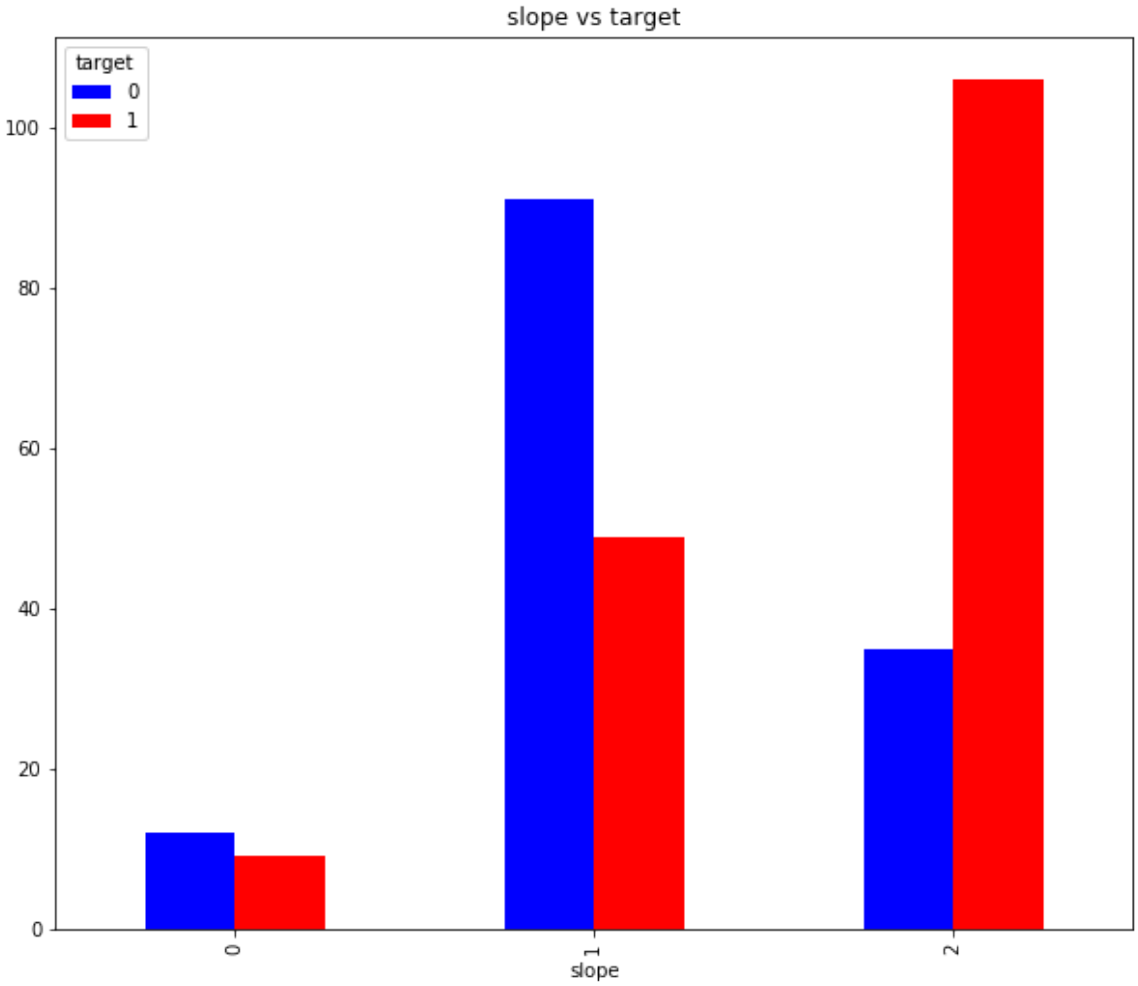
restecg vs target

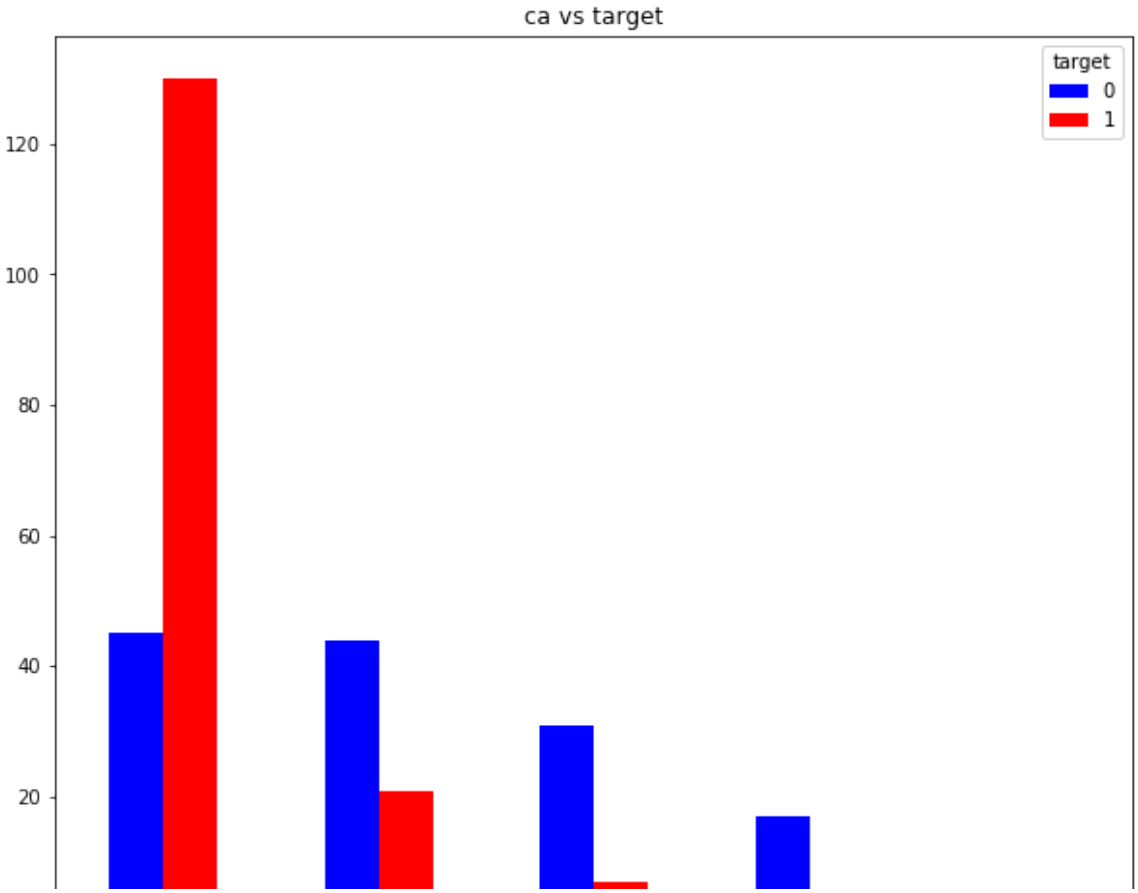


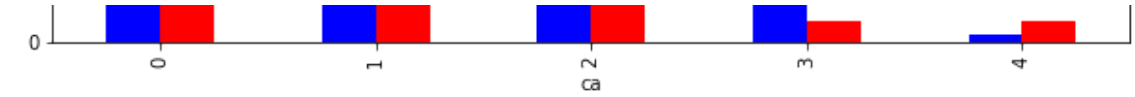
exang vs target

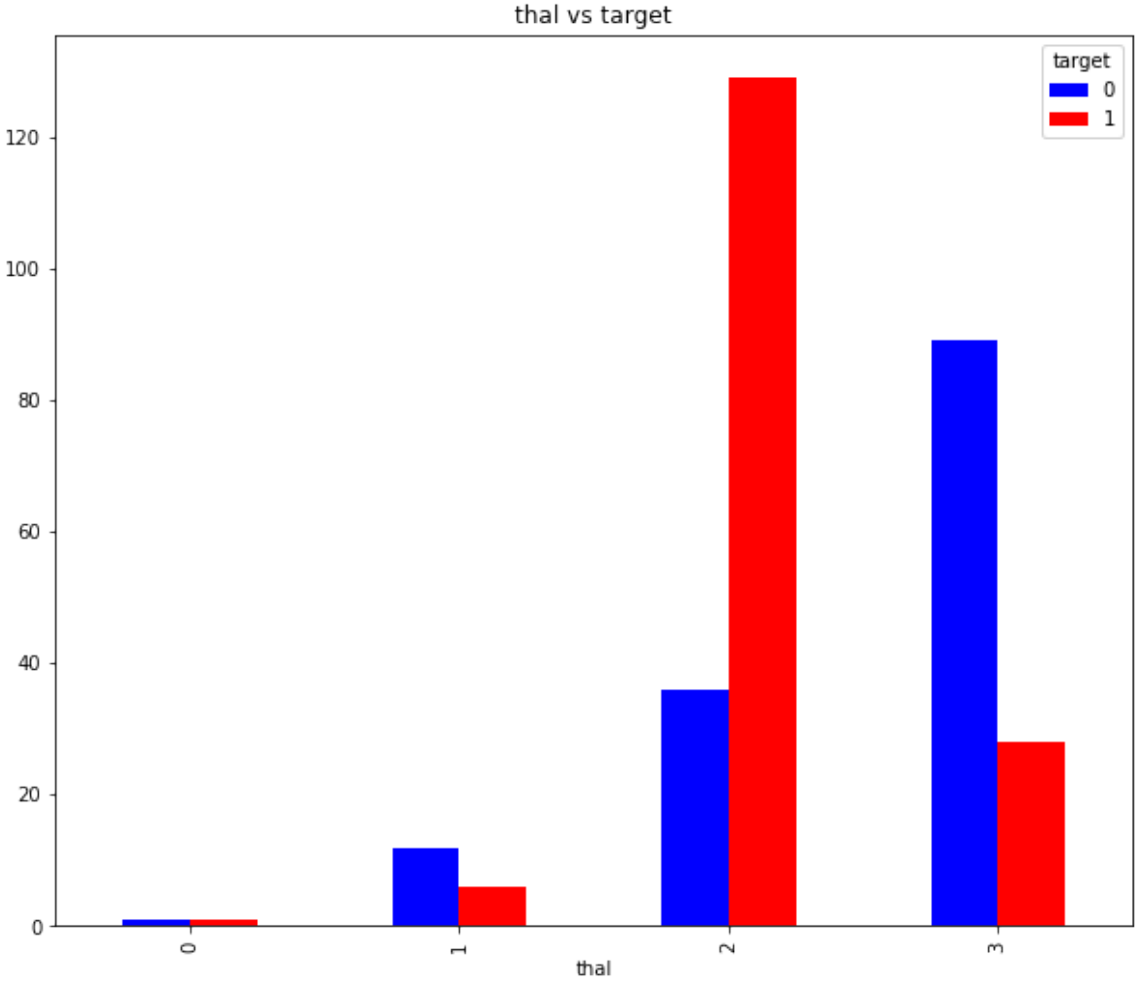



0
1
exang


slope vs target
target
0
1
100
80
60
40
20
0
0
1
2
slope


ca vs target
target
0
1
120
100
80
60
40
20

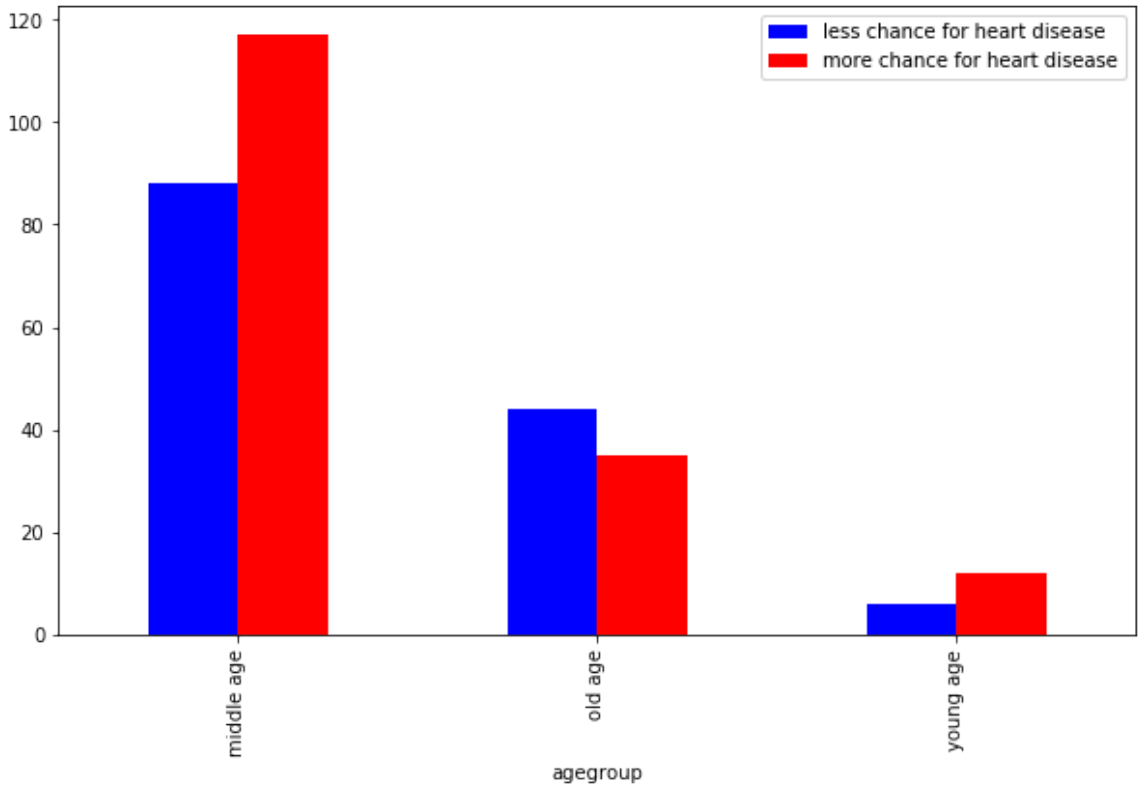




In [26]:
```
# focus on age with respect to Target variable
```

In [27]:
```
crosstab= pd.crosstab(index= HD['agegroup'], columns= HD['target'])
crosstab.plot(kind='bar', color= ['blue', 'red'], figsize=(10,6))
plt.legend(['less chance for heart disease','more chance for heart diseas
e'])
```

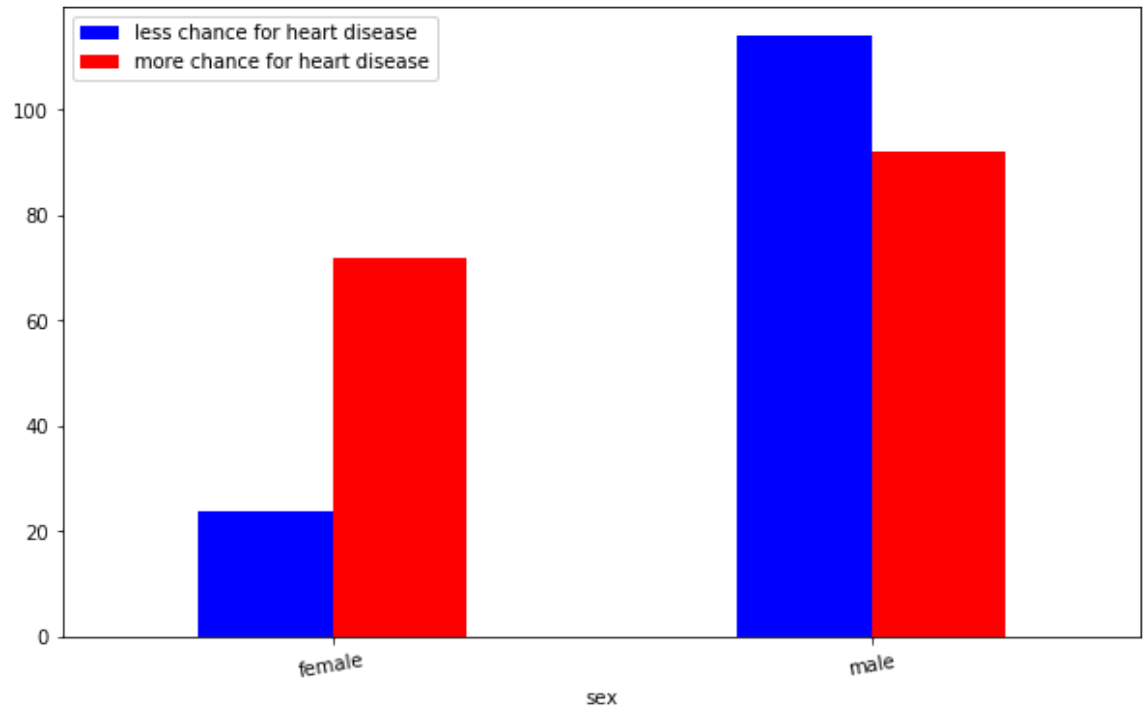
Out[27]: <matplotlib.legend.Legend at 0x232757710c8>



In [28]:
```
# focus on sex with respect to Target variable
```

In [29]:
```
crosstab= pd.crosstab(index= HD['sex'], columns= HD['target'])
crosstab.plot(kind='bar', color= ['blue', 'red'], figsize=(10,6))
plt.xticks([0,1], ['female', 'male'], rotation= 10)
plt.legend(['less chance for heart disease','more chance for heart diseas
e'])
```

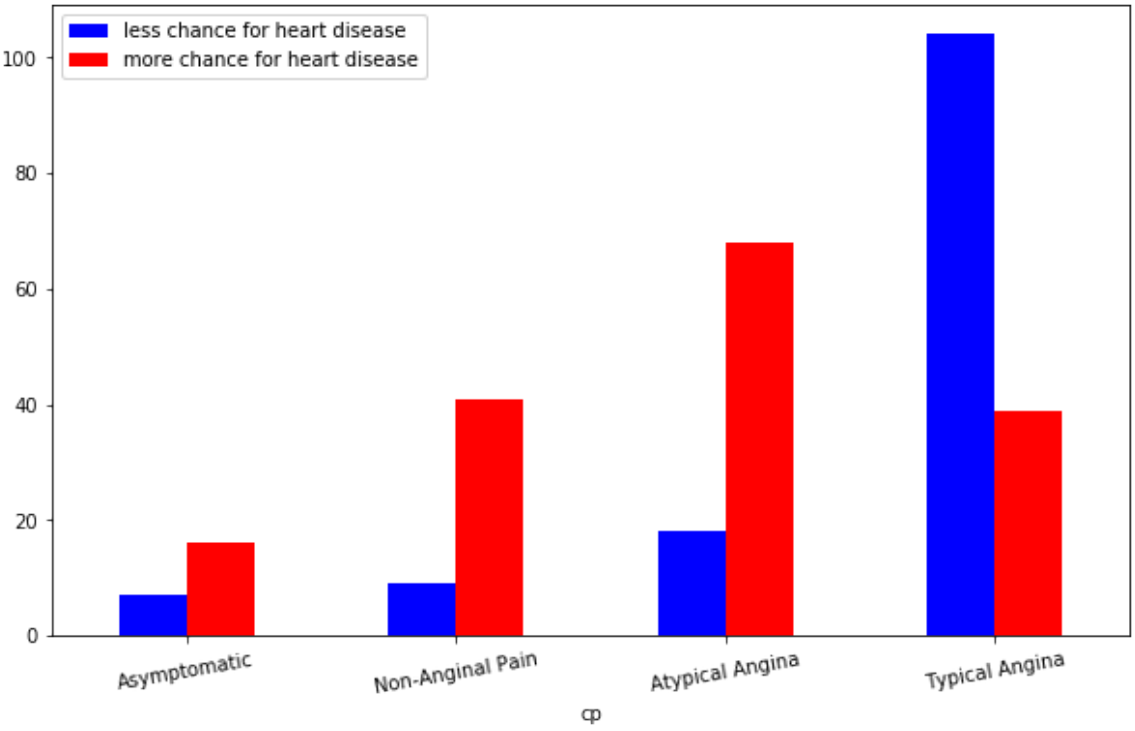
Out[29]:
```
<matplotlib.legend.Legend at 0x232756bfd08>
```



In [30]:
```
# focus on chest pain with respect to Target variable
```

In [31]:
```
crosstab= pd.crosstab(index= HD['cp'], columns= HD['target'])
crosstab.plot(kind='bar', color= ['blue', 'red'], figsize=(10,6))
plt.xticks([3,2,1,0], ['Typical Angina', 'Atypical Angina', 'Non-Anginal Pa
in', 'Asymptomatic'], rotation= 10)
plt.legend(['less chance for heart disease','more chance for heart diseas
e'])
```

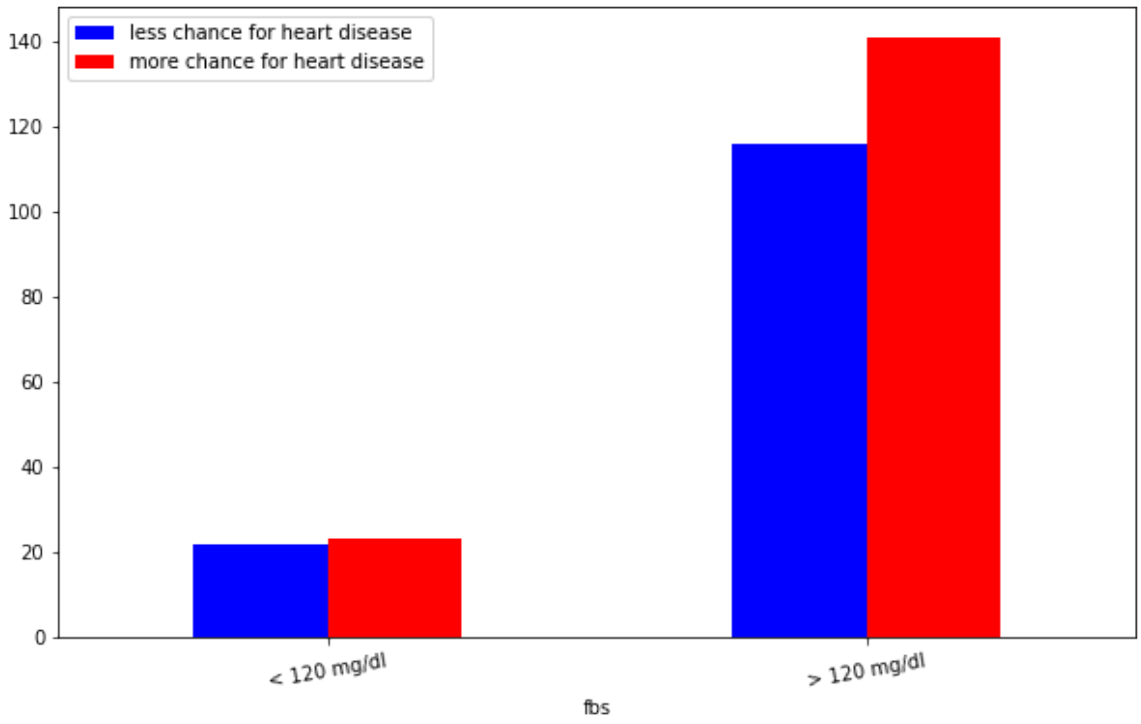
Out[31]:
```
<matplotlib.legend.Legend at 0x23275678308>
```



In [32]:
```
# focus on fasting blood sugar with respect to Target variable
```

In [34]:
```
crosstab= pd.crosstab(index= HD['fbs'], columns= HD['target'])
crosstab.plot(kind='bar', color= ['blue', 'red'], figsize=(10,6))
plt.xticks([0,1], ['< 120 mg/dl', '> 120 mg/dl'], rotation= 10)
plt.legend(['less chance for heart disease','more chance for heart diseas
e'])
```

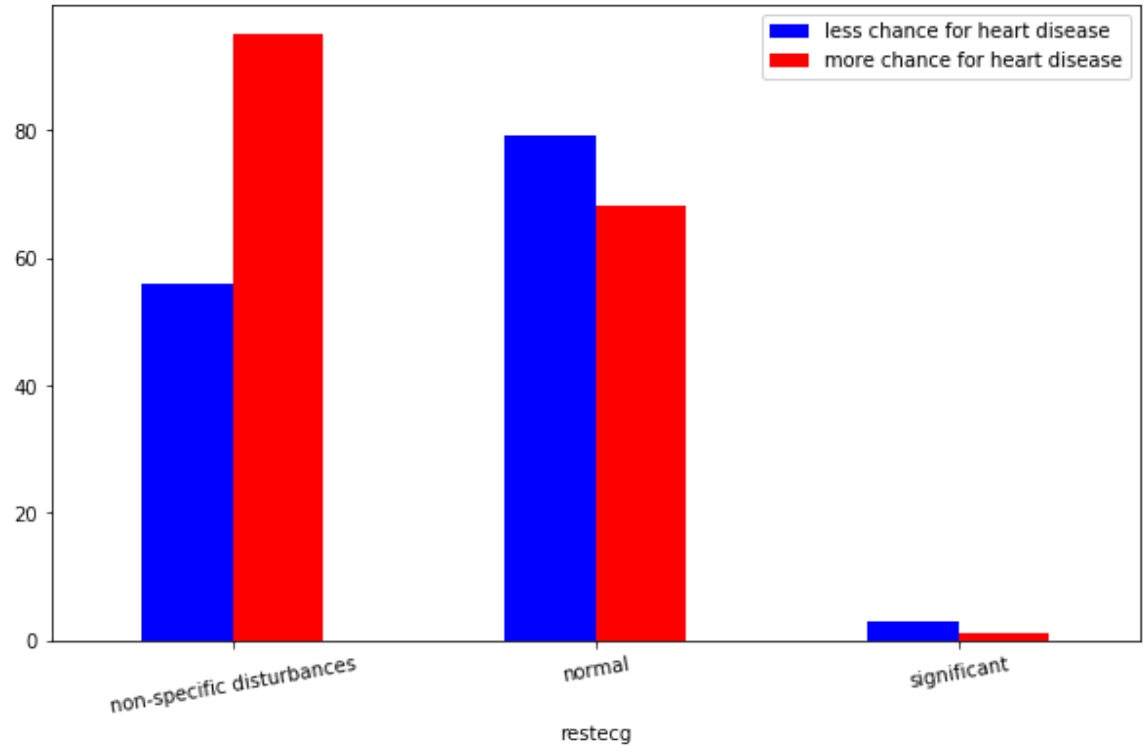
Out[34]:
```
<matplotlib.legend.Legend at 0x23276486088>
```



In [35]:
```
# focus on resting ecg with respect to Target variable
```

In [36]:
```
crosstab= pd.crosstab(index= HD['restecg'], columns= HD['target'])
crosstab.plot(kind='bar', color= ['blue', 'red'], figsize=(10,6))
plt.xticks([0,1,2], ['non-specific disturbances', 'normal','significant'],
rotation= 10)
plt.legend(['less chance for heart disease','more chance for heart diseas
e'])
```

Out[36]:
```
<matplotlib.legend.Legend at 0x232766dcd48>
```



In [ ]:
```
```

In [37]:
```
# Chi-Squre Test - statistical test to explained grouped bar chart diagram
to find best machine learning oriented columns

from scipy.stats import chi2_contingency

def FunctionChiSq(inpData, TargetVariable, PredictorList):

    FinalPredictor=[]

    for predictor in PredictorList:
        CrossTabResult= pd.crosstab(index= HD[predictor], columns= HD['targ
et'])
        ChiSqResult = chi2_contingency(CrossTabResult)

        if(ChiSqResult[1] < 0.05):
            print(predictor, 'is correlated with ', TargetVariable, '|| P-V
alue: ', ChiSqResult[1])
            FinalPredictor.append(predictor)

        else:
            print(predictor, 'is NOT correlated with ', TargetVariable, '||
P-Value: ', ChiSqResult[1])

    return(FinalPredictor)
```

In [38]:
```
# call the function

Categorical= ['agegroup','sex', 'cp', 'fbs', 'restecg', 'exang', 'slope',
'ca', 'thal']
FunctionChiSq(inpData = HD, TargetVariable = 'target', PredictorList = Cate
gorical)
```

```
agegroup is NOT correlated with  target || P-Value:  0.08518520148455873
sex is correlated with  target || P-Value:  1.5508552054949547e-06
cp is correlated with  target || P-Value:  1.8926838351935918e-17
fbs is NOT correlated with  target || P-Value:  0.7611374700928197
restecg is correlated with  target || P-Value:  0.007713053269318974
exang is correlated with  target || P-Value:  9.556466486179178e-14
slope is correlated with  target || P-Value:  6.577782760917924e-11
ca is correlated with  target || P-Value:  3.771038067427657e-15
thal is correlated with  target || P-Value:  3.146295138318122e-18
```

Out[38]:
```
['sex', 'cp', 'restecg', 'exang', 'slope', 'ca', 'thal']
```

In [ ]:
```

```