FACTORS RESPONSIBLE FOR ATTACK

October 14, 2021

[ ]:
```
import pandas as pd
import numpy as np
import warnings
import matplotlib.pyplot as plt
import seaborn as sns
warnings.filterwarnings('ignore')
plt.style.use('ggplot')
```

[ ]:
```
data = pd.read_excel('data.xlsx')
```

1 1. Preliminary analysis:

```
1. Perform preliminary data inspection and report the findings as the structure of the data, mi
2. Based on the findings from the previous question remove duplicates (if any) , treat missing
```

1.1 Understanding the data

[41]:
```
data.head()
```

[41]:
```
   age  sex  cp  trestbps  chol  fbs  …  exang  oldpeak  slope  ca  thal
target
0   63    1   3       145   233    1  …      0      2.3      0   0     1
1
1   37    1   2       130   250    0  …      0      3.5      0   0     2
1
2   41    0   1       130   204    0  …      0      1.4      2   0     2
1
3   56    1   1       120   236    0  …      0      0.8      2   0     2
1
4   57    0   0       120   354    0  …      1      0.6      2   0     2
1

[5 rows x 14 columns]
```

[42]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
age         303 non-null int64
sex         303 non-null int64
cp          303 non-null int64
trestbps    303 non-null int64
chol        303 non-null int64
fbs         303 non-null int64
restecg     303 non-null int64
thalach     303 non-null int64
exang       303 non-null int64
oldpeak     303 non-null float64
slope       303 non-null int64
ca          303 non-null int64
thal        303 non-null int64
target      303 non-null int64
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

[43]:
```
data.duplicated().sum()
```

[43]:
```
1
```

1.2 we can remove the duplicate

[ ]:
```
data.drop_duplicates(inplace = True)
data.reset_index(drop = True, inplace = True)
```

```
A. Get a preliminary statistical summary of the data. Explore the measures of central tendenci
```

1.3 exactly 1 duplicate row may be removed

[45]:
```
data.describe()
```

[45]:
```
              age         sex          cp  …          ca        thal  \
target
count  302.00000  302.000000  302.000000  …  302.000000  302.000000
302.000000
mean    54.42053    0.682119    0.963576  …    0.718543    2.314570
0.543046
std      9.04797    0.466426    1.032044  …    1.006748    0.613026
0.498970
min     29.00000    0.000000    0.000000  …    0.000000    0.000000
0.000000
```

```
25%     48.00000    0.000000    0.000000  …    0.000000    2.000000
0.000000
50%     55.50000    1.000000    1.000000  …    0.000000    2.000000
1.000000
75%     61.00000    1.000000    2.000000  …    1.000000    3.000000
1.000000
max     77.00000    1.000000    3.000000  …    4.000000    3.000000
1.000000

[8 rows x 14 columns]
```

2 based on data description :

sex -> sex male = 1 female = 0 cp -> chest pain type fbs -> fasting blood sugar 0 = lower than 120 mg/dl 1 = greater than 120 mg/dl exang -> exercise induced angina 0 = No 1 = Yes slope -> 0 = upsloping 1 = flat 2 = downsloping thal -> thalessimia 1 = normal 2 = fixed defect 3 = reversible defect

2.1 Changing variable names to more representative names

[ ]:
```
data.rename({'cp' :'chest_pain_type',
             'trestbps':'resting_blood_pressure',
             'chol':'cholesterol',
             'fbs' :'fasting_blood_sugar',
             'restecg':'resting_ecg',
             'thalach' : 'max_heart_rate',
             'exang':'exercise_induced_angina',
             'oldpeak':'st_depression',
             'slope':'st_slope',
             'ca' : 'major_vessels',
             'thal' :'thalessimia' },axis = 1, inplace = True)
```

[47]:
```
data.columns
```

[47]:
```
Index(['age', 'sex', 'chest_pain_type', 'resting_blood_pressure',
       'cholesterol', 'fasting_blood_sugar', 'resting_ecg', 'max_heart_rate',
       'exercise_induced_angina', 'st_depression', 'st_slope', 'major_vessels',
       'thalessimia', 'target'],
      dtype='object')
```

```
B. Identify the data variables which might be categorical in nature. Describe and explore these
```

2.2 creating a list of categorical columns for explicit understanding

[ ]:
```
cat =␣
↪['sex','chest_pain_type','fasting_blood_sugar','exercise_induced_angina','st_slope','thales
```

3 Statistical Description

[49]:
```
data.loc[ : , ~data.columns.isin(cat)].describe()
```

[49]:
```
             age  resting_blood_pressure  …  major_vessels      target
count  302.00000              302.000000  …     302.000000  302.000000
mean    54.42053              131.602649  …       0.718543    0.543046
std      9.04797               17.563394  …       1.006748    0.498970
min     29.00000               94.000000  …       0.000000    0.000000
25%     48.00000              120.000000  …       0.000000    0.000000
50%     55.50000              130.000000  …       0.000000    1.000000
75%     61.00000              140.000000  …       1.000000    1.000000
max     77.00000              200.000000  …       4.000000    1.000000

[8 rows x 8 columns]
```

[50]:
```
desc= pd.DataFrame(index = cat)
desc['nuinque'] = data[cat].apply(lambda x : x.nunique(), axis = 0)
desc['unique'] = 0
for i in cat :
    desc.loc[i,'unique'] = str(list(data[i].value_counts().index))
desc.T
```

[50]:
```
            sex chest_pain_type  …   st_slope   thalessimia
nuinque       2               4  …          3             4
unique   [1, 0]    [0, 2, 1, 3]  …  [2, 1, 0]  [2, 3, 1, 0]

[2 rows x 6 columns]
```

[51]:
```
data.thalessimia.value_counts()
```

[51]:
```
2    165
3    117
1     18
0      2
Name: thalessimia, dtype: int64
```

[ ]:
```
data.loc[data.thalessimia==0 , 'thalessimia'] = 2
```

Note :

- thalessimia has 4 unique categories according to data however in description there are only 3.
- there are 2 records which are identified as '0' ; these can be seen as missing values and hence need to be imputed.
- for imputation we can put in the category with modal value of '2'

3.0.1 converting the numeric categories for each column to relevent descriptors

[ ]:
```
data.loc[data.sex == 0 , 'sex'] = 'female'
data.loc[data.sex == 1, 'sex'] = 'male'

data.loc[data.chest_pain_type == 0,'chest_pain_type'] = 'typical angina'
data.loc[data.chest_pain_type == 1,'chest_pain_type'] = 'atypical angina'
data.loc[data.chest_pain_type == 2,'chest_pain_type'] = 'non-anginal pain'
data.loc[data.chest_pain_type == 3,'chest_pain_type'] = 'asymptomatic'

data.loc[data.fasting_blood_sugar == 0,'fasting_blood_sugar'] = '< 120mg/ml'
data.loc[data.fasting_blood_sugar == 1,'fasting_blood_sugar'] = '> 120mg/ml'

data.loc[data.resting_ecg == 0, 'resting_ecg']  = 'normal'
data.loc[data.resting_ecg == 1 , 'resting_ecg'] = 'abnormal'
data.loc[data.resting_ecg == 2 , 'resting_ecg'] = 'hyper'

data.loc[data.exercise_induced_angina == 0, 'exercise_induced_angina'] = 'no'
data.loc[data.exercise_induced_angina == 1, 'exercise_induced_angina'] = 'yes'

data.loc[data.st_slope == 0, 'st_slope'] = 'upsloping'
data.loc[data.st_slope == 1, 'st_slope'] = 'flat'
data.loc[data.st_slope == 2, 'st_slope'] = 'downsloping'

data.loc[data.thalessimia == 1,'thalessimia'] = 'normal'
data.loc[data.thalessimia == 2,'thalessimia'] = 'fixed defect'
data.loc[data.thalessimia == 3,'thalessimia'] = 'reversable defect'

#data.loc[data.target == 0, 'target']= 'Disease -'
#data.loc[data.target == 1, 'target']= 'Disease +'
```

[ ]:
```
dsprsnt = data[data.target == 1].copy()
dsabsnt = data[data.target == 0].copy()
```

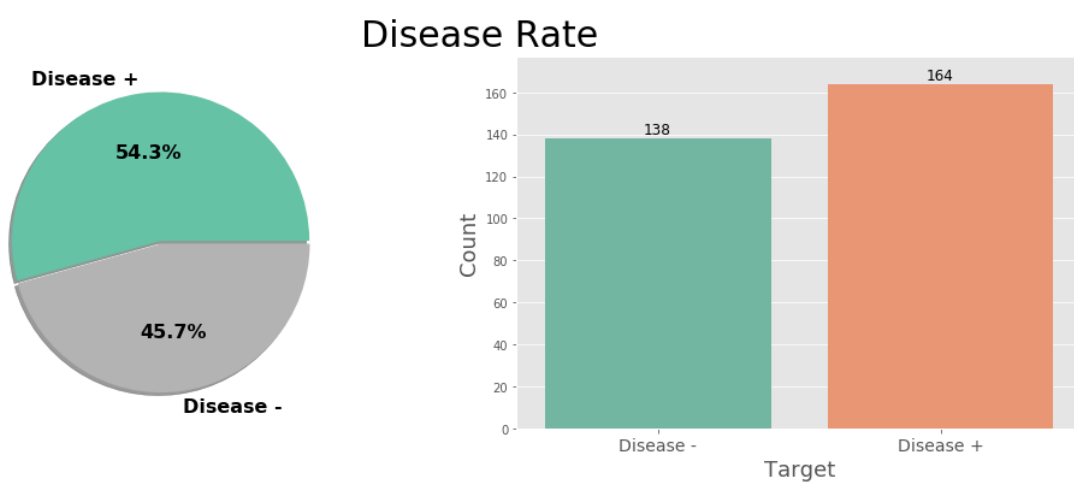
4 Target Distribution

[55]:
```
vc = data.target.value_counts()
vc
```

[55]:
```
1    164
0    138
Name: target, dtype: int64
```

[56]:
```
f,axes = plt.subplots(1,2, figsize = (15,6))
# plot no. 1
vc.plot.pie(ax = axes[0], radius = 1, cmap = 'Set2' , explode = [0.01,0.01],␣
↪shadow = True, autopct = '%1.1f%%',
            textprops = {'family': 'times','color': 'black','weight':␣
↪'bold','size': 16}, labels = ['Disease +','Disease -'])
axes[0].set_ylabel('')

# plot no. 2
sns.countplot(data.target,ax = axes[1],palette= 'Set2')
for i in range(len(vc)):
    axes[1].annotate(vc[i], (i-0.05,vc[i]+2), fontsize = 12)
axes[1].set_ylim(0,axes[1].set_ylim()[1]+5)
axes[1].set_xlabel('Target',fontsize = 18, family = 'times')
axes[1].set_ylabel('Count',fontsize = 18, family = 'times')
axes[1].set_xticklabels( ['Disease -','Disease +'], fontsize = 14, family =␣
↪'times')
f.suptitle('Disease Rate\n', fontsize = 30, family = 'times')
plt.tight_layout(pad = 4)
plt.show()
```

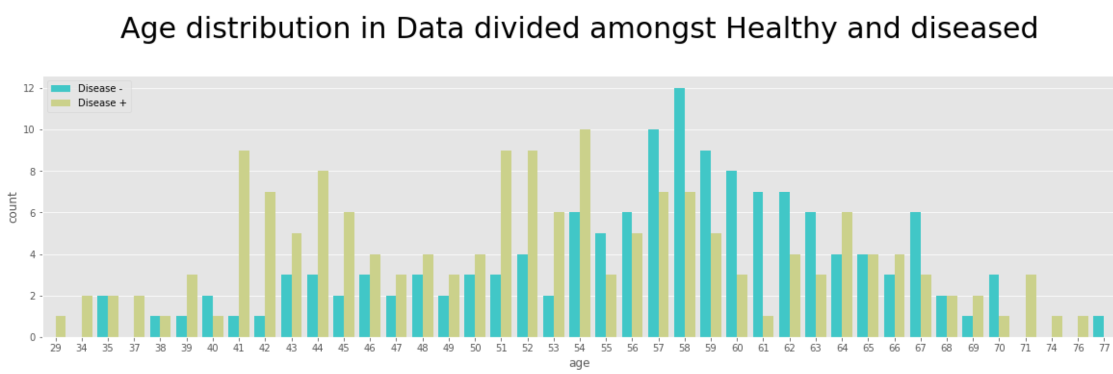


Note :

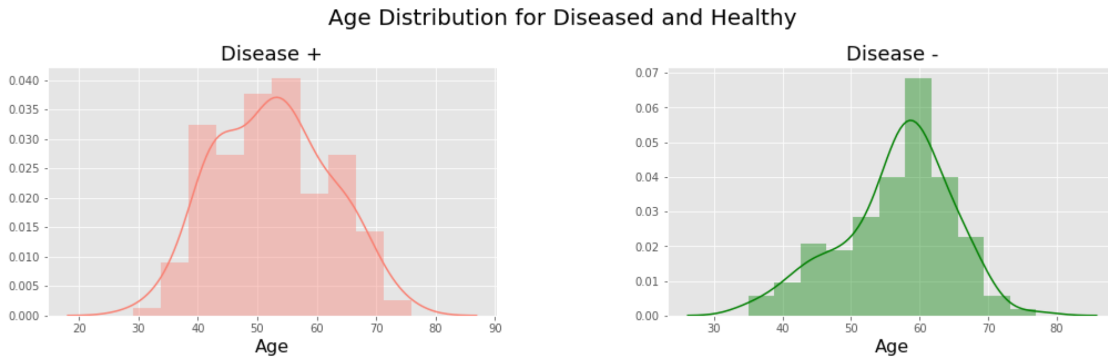
- To understand the data better, it is necessary to understand our target variable better.
- The data shows the data collected has almost equal representation of Diseased and Healthy samples

```
C. Study the occurrence of CVD across Age.
```

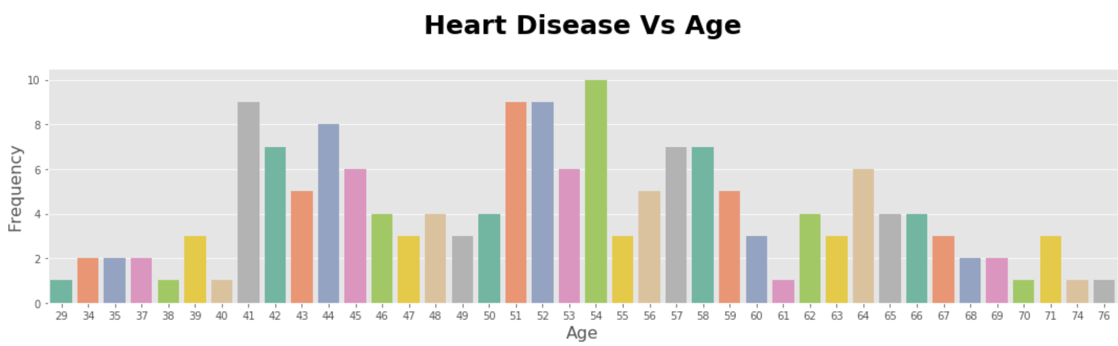
[57]:
```
plt.figure(figsize = (20,5))
sns.countplot(data.age, hue = data.target, palette='rainbow')
plt.legend(['Disease -','Disease +'], loc = 'upper left')
plt.title('\nAge distribution in Data divided amongst Healthy and diseased\n',␣
 ↪fontsize = 30)
plt.show()
```



[58]:
```
f,axes = plt.subplots(1,2, figsize = (15,5))
sns.distplot(dsprsnt.age,ax = axes[0], color = 'salmon')
sns.distplot(dsabsnt.age, ax = axes[1], color = 'green')
axes[0].set_title('Disease +',fontdict = {'family': 'times','size': 18})
axes[1].set_title('Disease -',fontdict = {'family': 'times','size': 18})
axes[0].set_xlabel('Age', fontdict = {'family': 'georgia','color':␣
 ↪'black','weight': 'normal','size': 16})
axes[1].set_xlabel('Age',fontdict = {'family': 'georgia','color':␣
 ↪'black','weight': 'normal','size': 16})
f.suptitle('Age Distribution for Diseased and Healthy\n\n ',fontsize= 20)
plt.tight_layout(w_pad= 12, pad = 4 )
plt.show()
```




```
[59]: plt.figure(figsize = (15,5))
sns.countplot(dsprsnt.age, palette='Set2')
plt.title('\nHeart Disease Vs Age\n',family='times', weight ='bold',fontsize= 25)
plt.tight_layout( )
plt.xlabel('Age',family='times',fontsize= 16)
plt.ylabel('Frequency',family='georgia',fontsize= 16)
plt.show()
```



Note :

- the chances of heart attack across age has intermittent peaks
- tendency of disease increases after 40
- the age groups 41 - 45 and 51 - 54 have the highest chances of heart attack

```
[ ]: def cat_plot(var):
    f,axes = plt.subplots(1,2, figsize = (18,5))
    vc = data[var].value_counts()
    nouniq = data[var].nunique()
    # overall pie
```

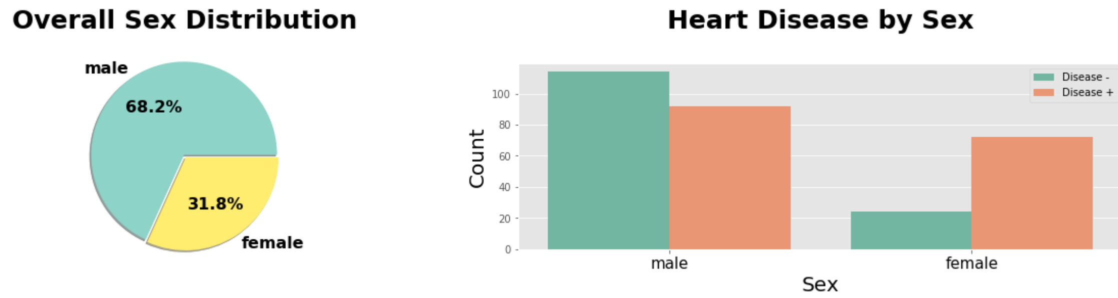
```
    vc.plot.pie(radius = 1.25,ax = axes[0], cmap = 'Set3', autopct = '%0.1f%%',
                                            textprops = {'family': 'times','color':␣
↪'black','weight': 'bold','size': 16},
                                            explode = [0.02]*nouniq,shadow = True,)
    axes[0].set_ylabel('')
    axes[0].set_title('Overall {} Distribution\n'.format(var.
↪capitalize()),family='times', weight ='bold',fontsize= 25)

    # count plot
    #pd.crosstab(data[var], data.target).plot.bar(cmap = 'Set2', ax = axes[1])
    sns.countplot(x = data[var], hue = data.target, ax = axes[1],␣
↪palette='Set2')
    plt.xticks( fontsize = 15, color = 'black' , family =  'times', rotation =␣
↪0)
    axes[1].set_xlabel(var.capitalize(),fontsize = 20, color = 'black' , family␣
↪=  'times', rotation = 0)
    axes[1].set_ylabel('Count',fontsize = 20, color = 'black' , family = ␣
↪'times')
    axes[1].legend(['Disease -','Disease +'])
    axes[1].set_title('Heart Disease by {}\n'.format( var.capitalize())␣
↪,family='times', weight ='bold',fontsize= 25)
    plt.tight_layout(pad = 4 )
    plt.show()
```

[ ]:
```
def real_distribution(var):
    f,axes = plt.subplots(1,2, figsize = (15,5))
    sns.distplot(dsprsnt[var],ax = axes[0], color = 'salmon')
    sns.distplot(dsabsnt[var], ax = axes[0], color = 'green')
    sns.boxplot(y = data[var], x = data.target, ax = axes[1], palette='Set2')
    axes[0].set_xlabel(var, fontdict = {'family': 'times','color':␣
↪'black','weight': 'normal','size': 16})
    axes[1].set_ylabel(var, fontdict = {'family': 'times','color':␣
↪'black','weight': 'normal','size': 16})
    axes[1].set_xlabel('Target', fontdict = {'family': 'times','color':␣
↪'black','weight': 'normal','size': 16})
    axes[1].set_xticklabels(['Disease -','Disease +'],{'family':␣
↪'times','color': 'black','weight': 'normal','size': 12})
    f.suptitle('{} vs Disease\n\n '.format(var.capitalize()),fontsize= 20,␣
↪family = 'times')
    plt.tight_layout(w_pad= 12, pad = 4 )
    plt.show()
```

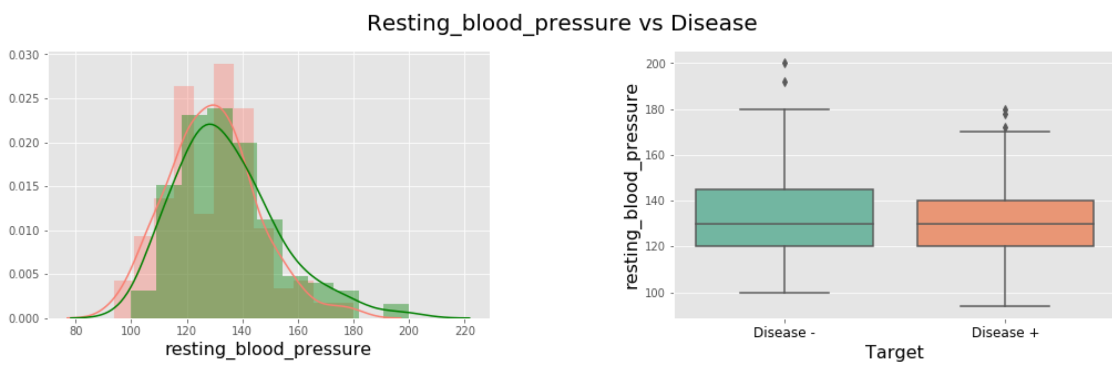
4.0.1 D. Study the composition of overall patients w.r.t . Gender

[62]:
```
cat_plot('sex')
```



5 E. Can we detect heart attack based on anomalies in Resting Blood Pressure of the patient?

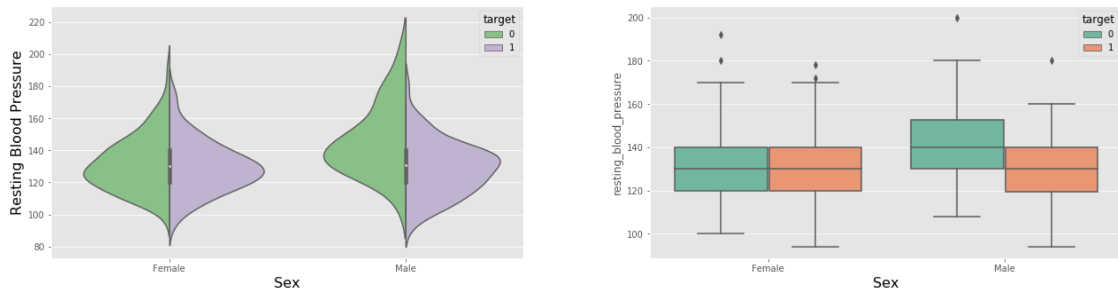
[63]:
```
real_distribution('resting_blood_pressure')
```



6 Resting blood pressure for male and female vs target

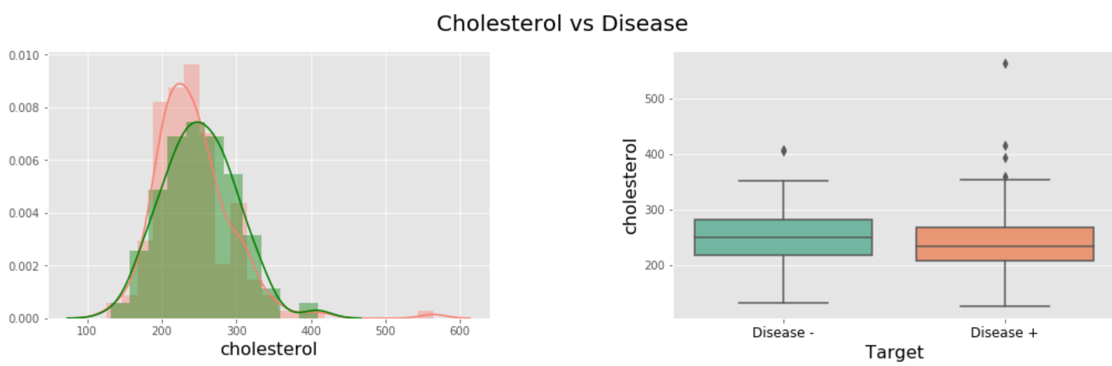
[64]:
```
f,axes = plt.subplots(1,2,figsize = (18,5))
sns.violinplot(y = 'resting_blood_pressure', x = 'sex',hue = 'target',data =␣
↪data, split = True, palette= 'Accent', ax = axes[0])
axes[0].set_xlabel('Sex', fontdict = {'family': 'georgia','color':␣
↪'black','weight': 'normal','size': 16})
axes[0].set_ylabel('Resting Blood Pressure', fontdict = {'family':␣
↪'georgia','color': 'black','weight': 'normal','size': 16})
```

```
axes[0].set_xticklabels(['Female','Male'])
sns.boxplot(x = data.sex,  y = data.resting_blood_pressure, hue = data.target,␣
↪ax = axes[1], palette='Set2')
axes[1].set_xticklabels(['Female','Male'])
axes[1].set_xlabel('Sex',fontdict = {'family': 'georgia','color':␣
↪'black','weight': 'normal','size': 16})
plt.tight_layout(w_pad = 10, pad = 2)
plt.show()
```



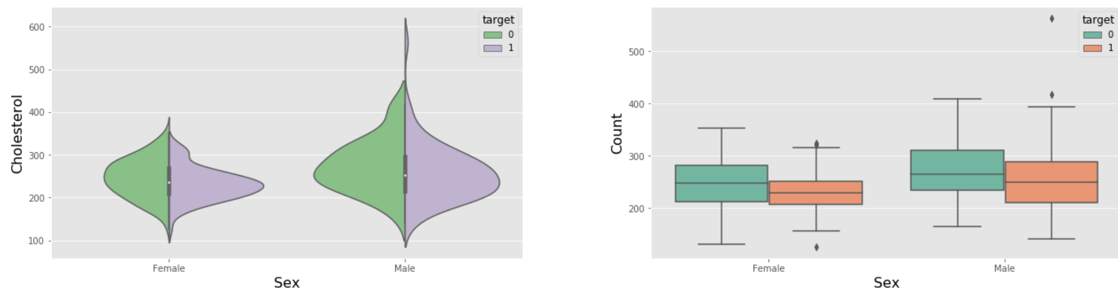
F. Describe the relationship between Cholesterol levels and our target variable.

[65]:
```
real_distribution('cholesterol')
```



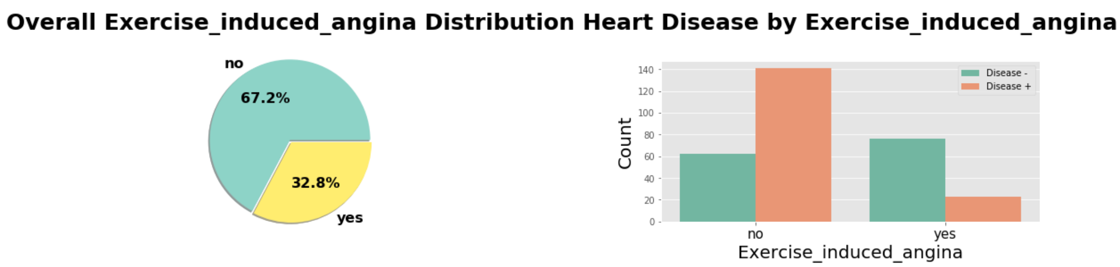
[66]:
```
f,axes = plt.subplots(1,2,figsize = (18,5))
sns.violinplot(y = 'cholesterol', x = 'sex',hue = 'target',data = data, split =␣
↪True, palette= 'Accent', ax = axes[0])
axes[0].set_xlabel('Sex', fontdict = {'family': 'georgia','color':␣
↪'black','weight': 'normal','size': 16})
axes[0].set_ylabel('Cholesterol', fontdict = {'family': 'georgia','color':␣
↪'black','weight': 'normal','size': 16})
axes[0].set_xticklabels(['Female','Male'])
```

```
sns.boxplot(x = data.sex,  y = data.cholesterol, hue = data.target, ax =␣
↪axes[1], palette='Set2')
axes[1].set_xticklabels(['Female','Male'])
axes[1].set_xlabel('Sex',fontdict = {'family': 'georgia','color':␣
↪'black','weight': 'normal','size': 16} )
axes[1].set_ylabel('Count',fontdict = {'family': 'georgia','color':␣
↪'black','weight': 'normal','size': 16} )
plt.tight_layout(w_pad = 10, pad = 2)
plt.show()
```



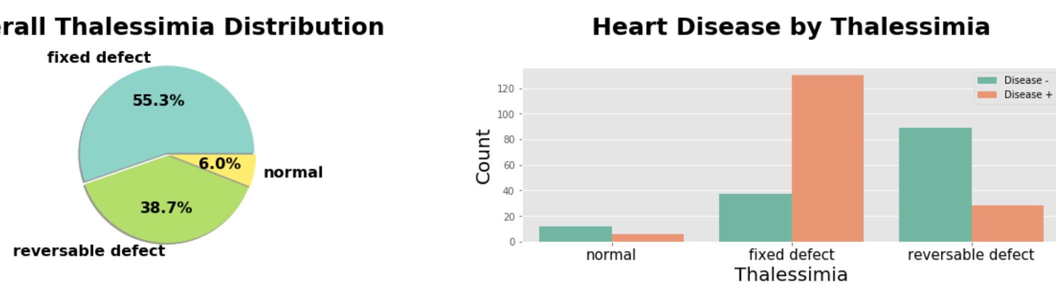
G. What can be concluded about the relationship between peak exercising and occurrence of hear

[67]:
```
cat_plot('exercise_induced_angina')
```



H. Is thalassemia a major cause of CVD?

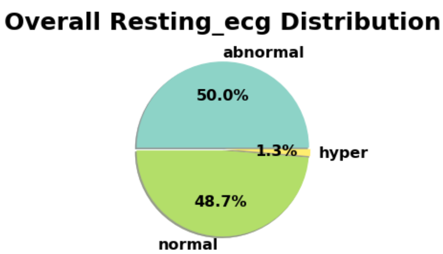
[68]:
```
cat_plot('thalessimia')
```

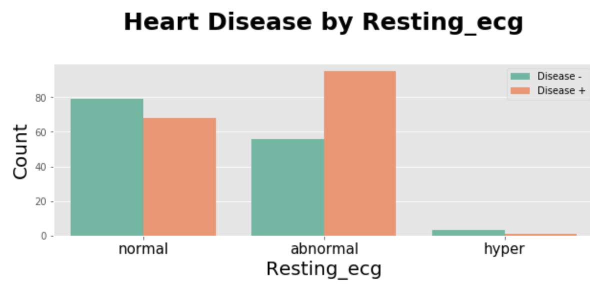


I. How are the other factors determining the occurrence of CVD?

6.1 1. Resting_ecg

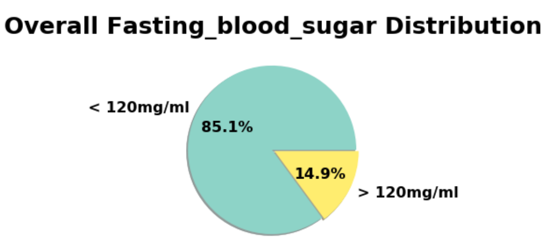
[69]:
```
cat_plot('resting_ecg')
```

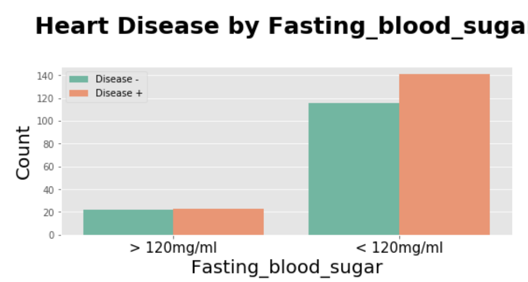




6.2 2. Fasting Blood Sugar

[70]:
```
cat_plot('fasting_blood_sugar')
```

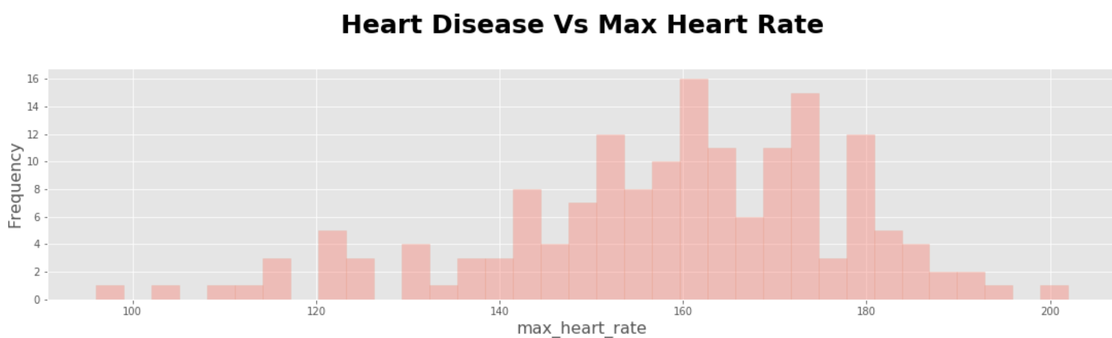




6.3 3. Max Heart Rate Achieved

[71]:
```
#real_distribution('max_heart_rate')
plt.figure(figsize = (15,5))
sns.distplot(dsprsnt.max_heart_rate, kde = False, bins = 35, hist_kws =␣
↪{'edgecolor':'darksalmon', 'color' :'salmon'})
plt.title('\nHeart Disease Vs Max Heart Rate\n',family='times', weight␣
↪='bold',fontsize= 25)
plt.tight_layout( )
plt.xlabel('max_heart_rate',family='times',fontsize= 16)
```

```
plt.ylabel('Frequency',family='georgia',fontsize= 16)
plt.show()
```

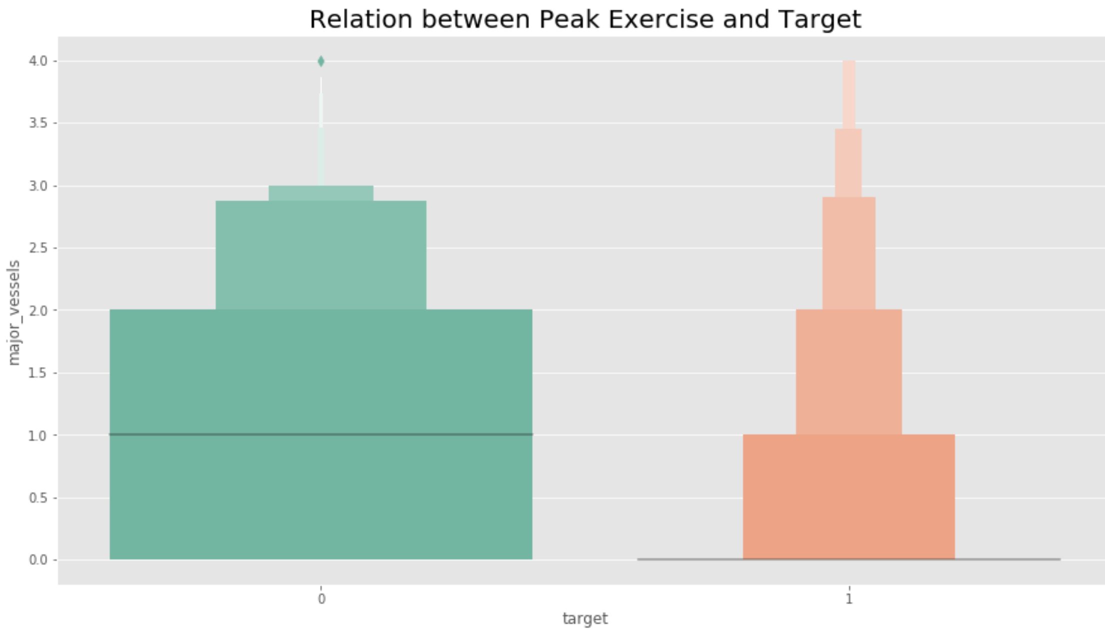


7 4. major_vessels

[72]:
```
data.columns
```

[72]:
```
Index(['age', 'sex', 'chest_pain_type', 'resting_blood_pressure',
       'cholesterol', 'fasting_blood_sugar', 'resting_ecg', 'max_heart_rate',
       'exercise_induced_angina', 'st_depression', 'st_slope', 'major_vessels',
       'thalessimia', 'target'],
      dtype='object')
```

[73]:
```
plt.figure(figsize= (15,8))
sns.boxenplot(data['target'], data['major_vessels'], palette = 'Set2')
plt.title('Relation between Peak Exercise and Target', fontsize = 20,␣
 ↪fontweight = 30)
plt.show()
```

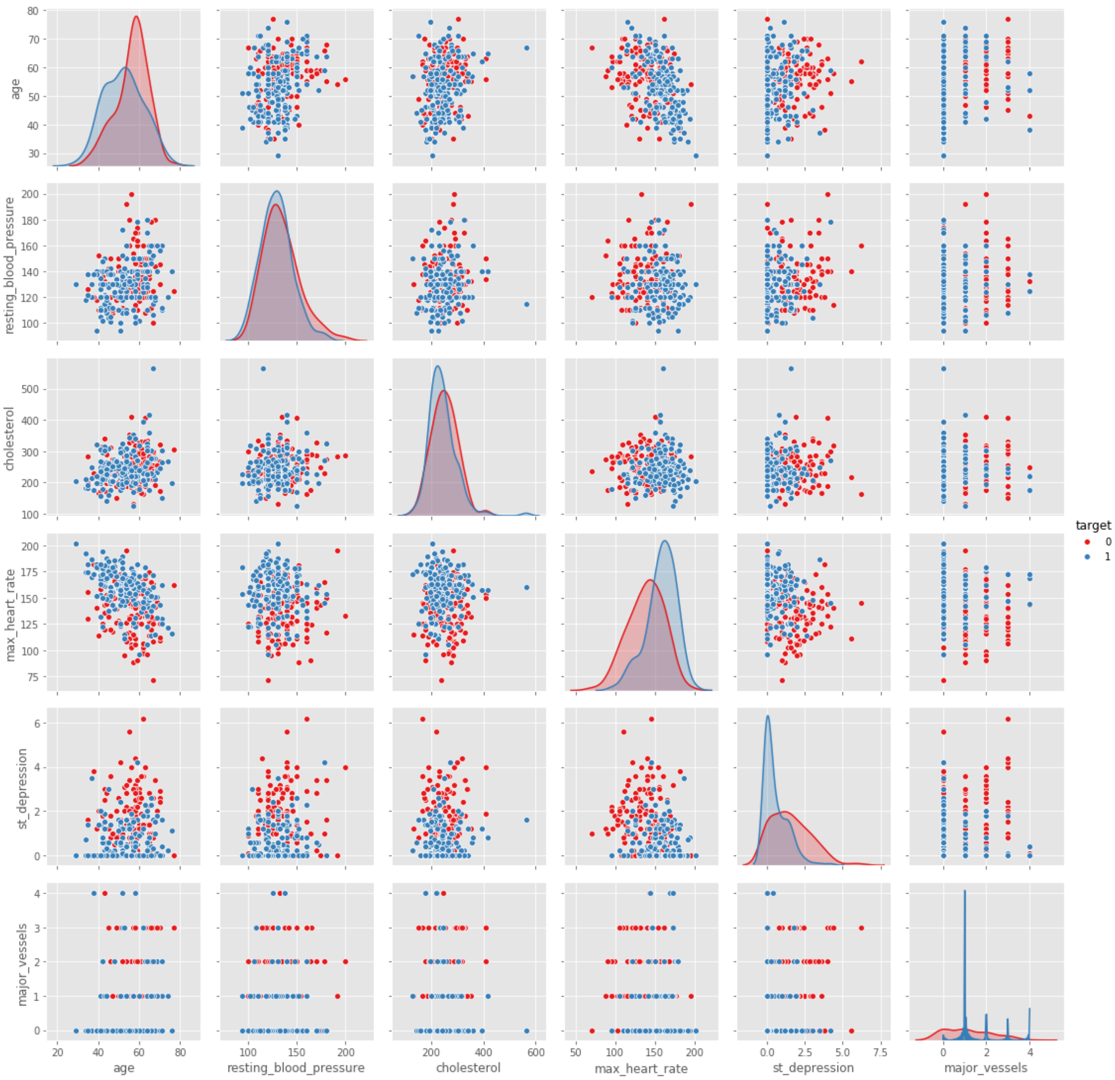


The above Bivariate plot between Target and Number of Major Vessels, shows that the patients who are more likely to suffer from Heart diseases are having high values of Major Vessels wheras the patiets who are very less likely to suffer from any kind of heart diseases have very low values of Major Vessels.

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

[74]:
```
sns.pairplot(data, hue = 'target', palette='Set1')
```

[74]: <seaborn.axisgrid.PairGrid at 0x7fe7998af080>



7.1 apply logistic Regression

[75]:
```
from sklearn.model_selection import train_test_split as split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, accuracy_score

data_dummy = pd.get_dummies(data)
data_dummy.columns = data_dummy.columns.str.replace(' ','_')

train, test = split(data_dummy, test_size = .30, random_state = 12)
train.shape

train.head(2)
```

```python
X_train = train.drop('target', axis = 1)
Y_train = train.target
X_test = test.drop('target', axis = 1)
Y_test = test.target
lr = LogisticRegression()
lr.fit(X_train,Y_train)

pred = lr.predict(X_test)

accuracy_score(y_true = Y_test,y_pred = pred)

print(classification_report(y_true=Y_test,y_pred = pred))
```

```
              precision    recall  f1-score   support

           0       0.82      0.89      0.85        45
           1       0.88      0.80      0.84        46

    accuracy                           0.85        91
   macro avg       0.85      0.85      0.85        91
weighted avg       0.85      0.85      0.85        91
```

[83]:
```python
from sklearn.metrics import confusion_matrix
print(confusion_matrix(Y_test, pred))
```

```
[[40  5]
 [ 9 37]]
```

[ ]:
```python

```