full_project_of_heart_disease

May 31, 2023

1 Full project of heart disease

2 1)1. import libraries

[325]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
```

3 2) 2. import datasets

[326]:
```
data=pd.read_csv("heartdisease.csv")
```

[327]:
```
data
```

[327]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	\
0	63	Male	3	145.0	233	1	0	150	0	2.3	
1	37	Male	2	130.0	250	0	1	187	0	3.5	
2	41	Female	1	130.0	204	0	0	172	0	1.4	
3	56	Male	1	120.0	236	0	1	178	0	0.8	
4	57	Female	0	120.0	354	0	1	163	1	0.6	
..	..	…	..	…	…	…	…	…	…	…	
298	57	Female	0	140.0	241	0	1	123	1	0.2	
299	45	Male	3	110.0	264	0	1	132	0	1.2	
300	68	Male	0	144.0	193	1	1	141	0	3.4	
301	57	Male	0	130.0	131	0	1	115	1	1.2	
302	57	Female	1	130.0	236	0	0	174	0	0.0	

	slope	ca	thal	target	Unnamed: 14
0	0	0.0	1	1.0	Male

```
1         0  0.0     2     1.0        Male
2         2  0.0     2     1.0      Female
3         2  0.0     2     1.0        Male
4         2  0.0     2     1.0      Female
..      …  …   …     …         …
298       1  0.0     3     0.0      Female
299       1  0.0     3     0.0        Male
300       1  2.0     3     0.0        Male
301       1  1.0     3     0.0        Male
302       1  1.0     2     0.0      Female

[303 rows x 15 columns]
```

[173]:
```
data.isnull().sum()
```

[173]:
```
age             0
sex             0
cp              0
trestbps        1
chol            0
fbs             0
restecg         0
thalach         0
exang           0
oldpeak         0
slope           0
ca            163
thal            0
target          2
Unnamed: 14     0
dtype: int64
```

[328]:
```
data=data.drop('ca', axis=1)
```

[329]:
```
data.head(5)
```

[329]:
```
   age     sex cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  \
0   63    Male  3     145.0   233    1        0      150      0      2.3
1   37    Male  2     130.0   250    0        1      187      0      3.5
2   41  Female  1     130.0   204    0        0      172      0      1.4
3   56    Male  1     120.0   236    0        1      178      0      0.8
4   57  Female  0     120.0   354    0        1      163      1      0.6

   slope  thal  target Unnamed: 14
0      0     1     1.0        Male
1      0     2     1.0        Male
2      2     2     1.0      Female
```

```
3      2     2     1.0        Male
4      2     2     1.0      Female
```

4 3)3. describe features

5 The following attributes describe each of the datasets features used by the model:

1)AGE-age in years

2)sex =(0:female,1:male)

3)cp =chest pain type

0: Typical angina: chest pain related decrease blood supply to the heart. 1: Atypical angina: chest pain not related to heart 2: Non-anginal pain: typically esophageal spasms (non heart related) 3: Asymptomatic: chest pain not showing signs of disease

4)trestbps= resting of blood pressure(anything above 130-140 is a typical cause of concern

5)chol =serum cholestoral in mg/dl serum = LDL + HDL + .2 * triglycerides above 200 is cause for concern

6)fbs = fasting blood sugar (1=true and 0=false), anything greater than 126 signals diabatics

7)restecg - resting electrocardiographic results

0: Nothing to note 1: ST-T Wave abnormality can range from mild symptoms to severe problems signals non-normal heart beat 2: Possible or definite left ventricular hypertrophy Enlarged heart's main pumping chamber

8)thalach - maximum heart rate achieved

9)exang - exercise induced angina (1 = yes; 0 = no)

10)oldpeak - ST depression induced by exercise relative to rest looks at stress of heart during excercise unhealthy heart will stress more

11)slope - the slope of the peak exercise ST segment

0: Upsloping: better heart rate with excercise (uncommon) 1: Flatsloping: minimal change (typical healthy heart) 2: Downslopins: signs of unhealthy heart

12)ca - number of major vessels (0-3) colored by flourosopy colored vessel means the doctor can see the blood passing through

the more blood movement the better (no clots)

13)thal - thalium stress result

1,3: normal 6: fixed defect: used to be defect but ok now

7: reversable defect: no proper blood movement when excercising

14)target - have disease or not (1=yes, 0=no)

6 4. data cleansing

6.1 deal with data type issue. deal with abnormal values(“a”,”40+”).

[330]:
```
print(data['age'].unique())
```

```
['63' '37' '41' '56' '57' '44' '52' '54' '48' '49' '64' '58' '50' '66'
 '43' '69' '59' '42' '61' '40' '71' '51' '0' '53' '65' '46' '45' '39' '47'
 '62' '34' '35' '29' '55' '60' '67' '68' '74' '76' '70' '38' '77' '40+']
```

[331]:
```
data['age']=data['age'].apply(lambda x:"40"if x=="40+" else x)
```

[332]:
```
data[data['age']=="40+"]
```

[332]:
```
Empty DataFrame
Columns: [age, sex, cp, trestbps, chol, fbs, restecg, thalach, exang, oldpeak,
slope, thal, target, Unnamed: 14]
Index: []
```

[333]:
```
data['age']=data['age'].astype(int)
data['age'].dtype
```

[333]:
```
dtype('int32')
```

[335]:
```
print(data['cp'].unique())
```

```
['3' '2' '1' '0' 'a']
```

[336]:
```
data=data[data['cp']!='a']
```

[337]:
```
data[data['cp']=="a"]
```

[337]:
```
Empty DataFrame
Columns: [age, sex, cp, trestbps, chol, fbs, restecg, thalach, exang, oldpeak,
slope, thal, target, Unnamed: 14]
Index: []
```

[338]:
```
data['cp'] = data['cp'].astype('int')  # convert data type into int
```

[339]:
```
data['cp'].dtype
```

[339]:
```
dtype('int32')
```

[340]:
```
data.dtypes
```

[340]:
```
age             int32
sex            object
cp              int32
trestbps      float64
chol            int64
```

```
fbs              int64
restecg          int64
thalach          int64
exang            int64
oldpeak        float64
slope            int64
thal             int64
target         float64
Unnamed: 14     object
dtype: object
```

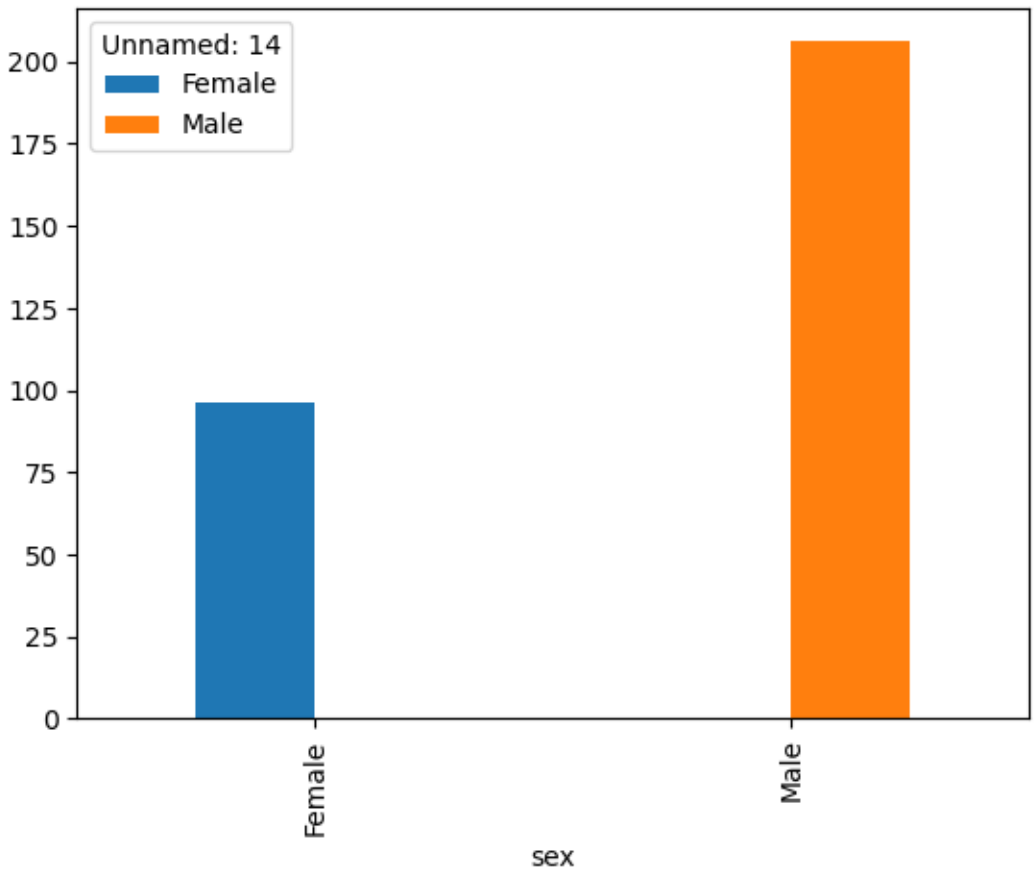
6.1.1 sex and unnamed:14 columns are same so deal with them

[341]:
```
pd.crosstab(data['sex'], data['Unnamed: 14'])
```

[341]:
```
Unnamed: 14  Female  Male
sex
Female           96     0
Male              0   206
```

[342]:
```
pd.crosstab(data['sex'], data['Unnamed: 14']).plot(kind='bar')
```

[342]:
```
<AxesSubplot:xlabel='sex'>
```



[343]:
```
data = data.drop('Unnamed: 14', axis=1)
```

[344]:
```
data
```

[344]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	\
0	63	Male	3	145.0	233	1	0	150	0	2.3	
1	37	Male	2	130.0	250	0	1	187	0	3.5	
2	41	Female	1	130.0	204	0	0	172	0	1.4	
3	56	Male	1	120.0	236	0	1	178	0	0.8	
4	57	Female	0	120.0	354	0	1	163	1	0.6	
..	…	…	..	…	…	…	…	…	…	…	
298	57	Female	0	140.0	241	0	1	123	1	0.2	
299	45	Male	3	110.0	264	0	1	132	0	1.2	
300	68	Male	0	144.0	193	1	1	141	0	3.4	
301	57	Male	0	130.0	131	0	1	115	1	1.2	
302	57	Female	1	130.0	236	0	0	174	0	0.0	

	slope	thal	target
0	0	1	1.0

```
1        0     2     1.0
2        2     2     1.0
3        2     2     1.0
4        2     2     1.0
..     ...   ...     ...
298      1     3     0.0
299      1     3     0.0
300      1     3     0.0
301      1     3     0.0
302      1     2     0.0

[302 rows x 13 columns]
```

[345]:
```
data.isnull().sum()
```

[345]:
```
age         0
sex         0
cp          0
trestbps    1
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
thal        0
target      2
dtype: int64
```

[346]:
```
data.dropna(subset=['trestbps'],inplace=True)
```

[347]:
```
data['target'].fillna(data['target'].mean(), inplace=True)
```

[348]:
```
data.isnull().sum()
```

[348]:
```
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
```

```
thal        0
target      0
dtype: int64
```

[349]:
```
data.dtypes
```

[349]:
```
age           int32
sex          object
cp            int32
trestbps    float64
chol          int64
fbs           int64
restecg       int64
thalach       int64
exang         int64
oldpeak     float64
slope         int64
thal          int64
target      float64
dtype: object
```

[350]:
```
data['trestbps'] = data['trestbps'].astype('int')  # convert data type into int
data['oldpeak'] = data['oldpeak'].astype('int')
data['target'] = data['target'].astype('int')
```

[351]:
```
data.dtypes
```

[351]:
```
age           int32
sex          object
cp            int32
trestbps      int32
chol          int64
fbs           int64
restecg       int64
thalach       int64
exang         int64
oldpeak       int32
slope         int64
thal          int64
target        int32
dtype: object
```

6.2 5. EDA

7 a) Outlers

Detecting outliers is not challenging at all. You can detect outliers by using the following:

1.Boxplot

2.Histogram

3.Mean and Standard Deviation

4.IQR (Inter Quartile Range)

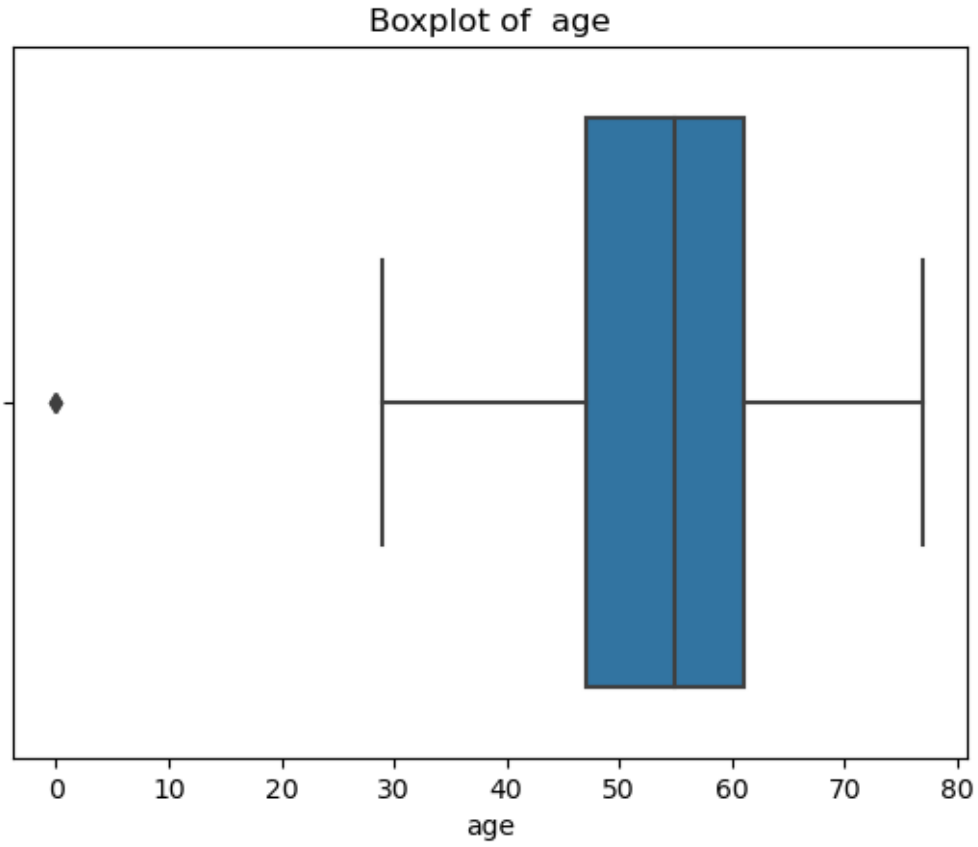
5.Z-score

6.Percentile

[26]:
```
# age
```

[352]:
```
sns.boxplot(x=data['age'])

plt.title("Boxplot of  age")
```

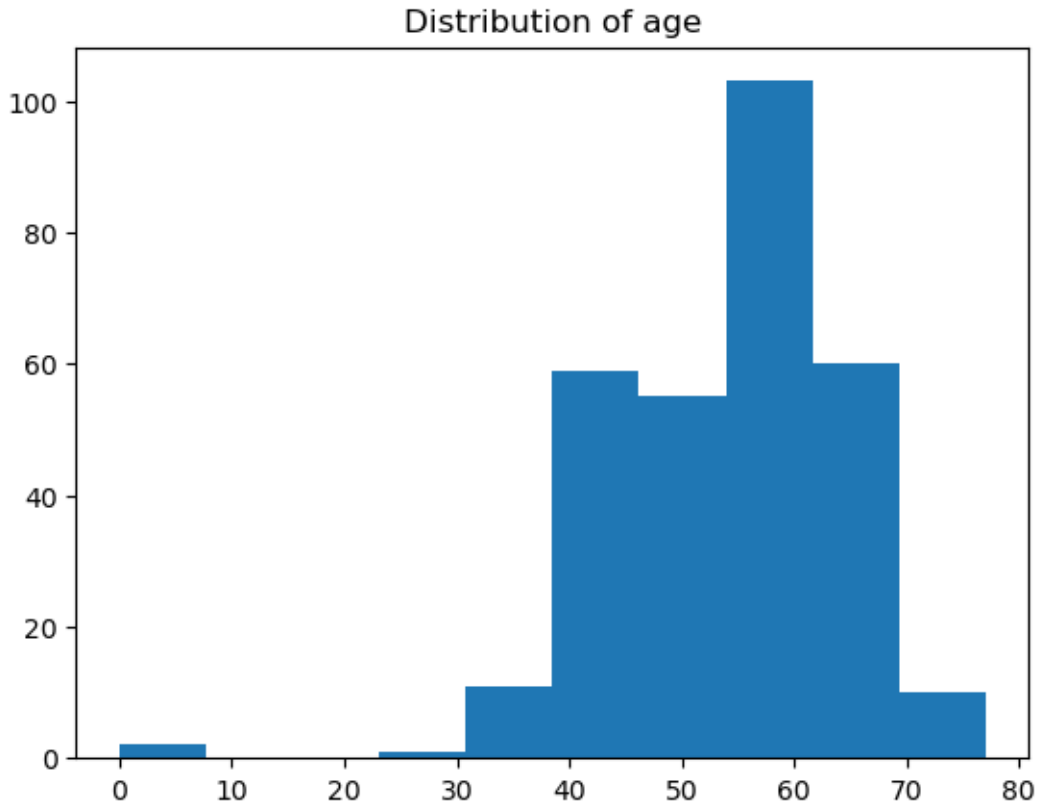
[352]:
```
Text(0.5, 1.0, 'Boxplot of  age')
```



[353]:
```
data[data["age"]<20]
```

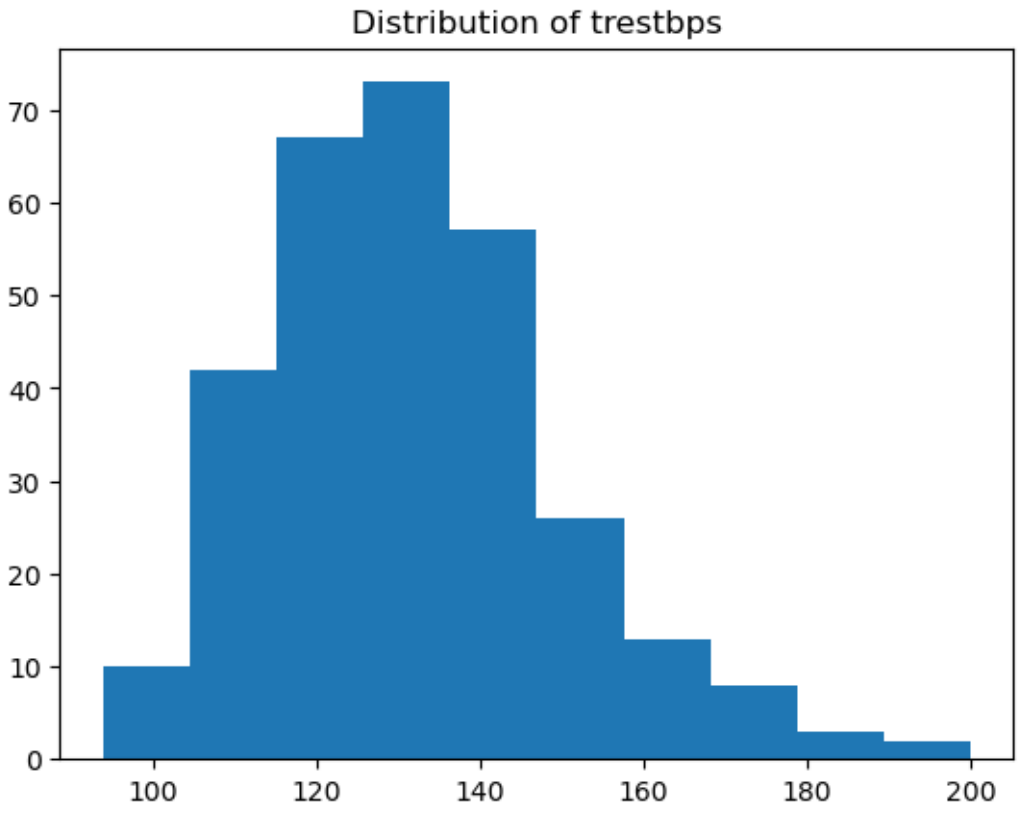
[353]:
```
      age     sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  \
28      0  Female   2       140   417    1        0      157      0        0
273     0    Male   0       100   234    0        1      156      0        0

     slope  thal  target
28       2     2       1
273      2     3       0
```

[354]:
```
plt.hist(data['age'])
plt.title("Distribution of age")
plt.show()
```



[355]:
```
# 2)trestbps ..find outlier using histogram
```

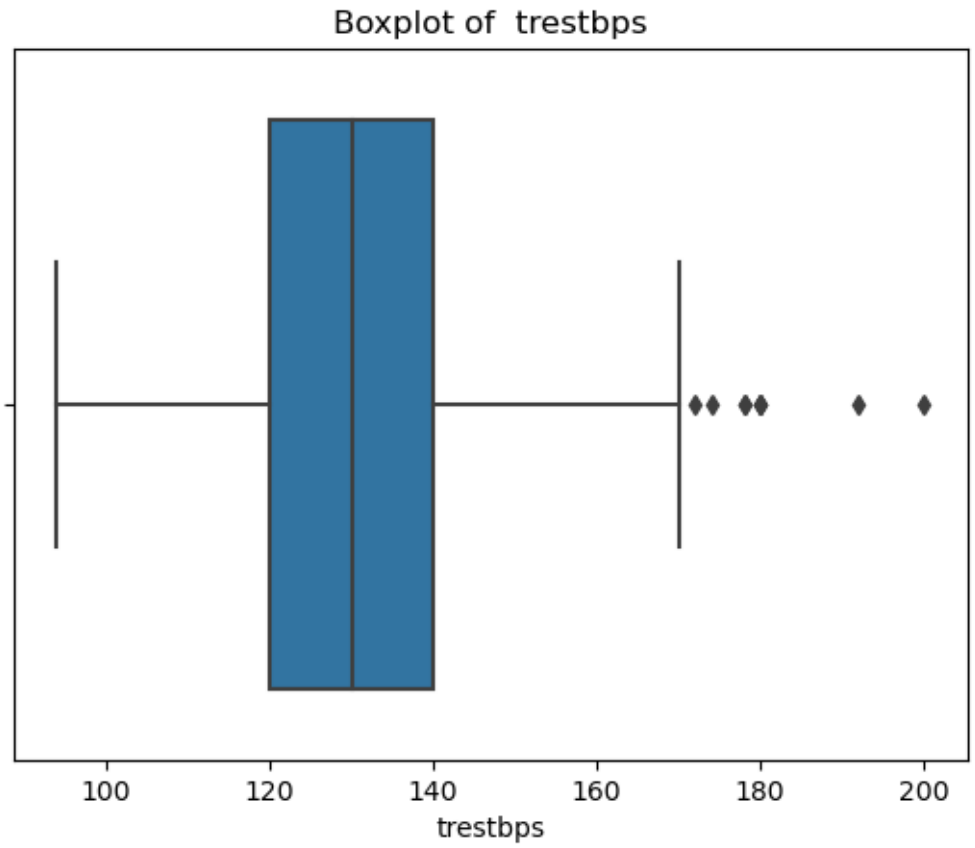
[356]:
```
plt.hist(data['trestbps'])
plt.title("Distribution of trestbps")
plt.show()
```



[357]:
```
# using boxplot
sns.boxplot(data['trestbps'])
plt.title("Boxplot of  trestbps")
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```

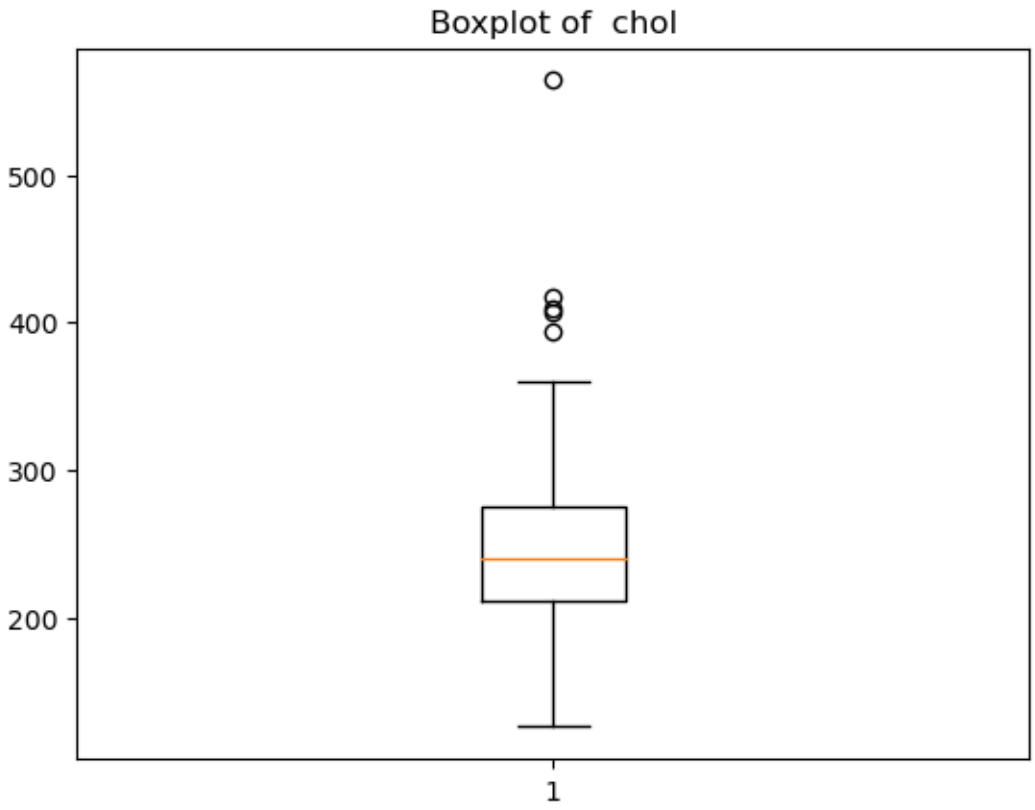
[357]:
```
Text(0.5, 1.0, 'Boxplot of  trestbps')
```



[358]:
```
# 3)chol
```

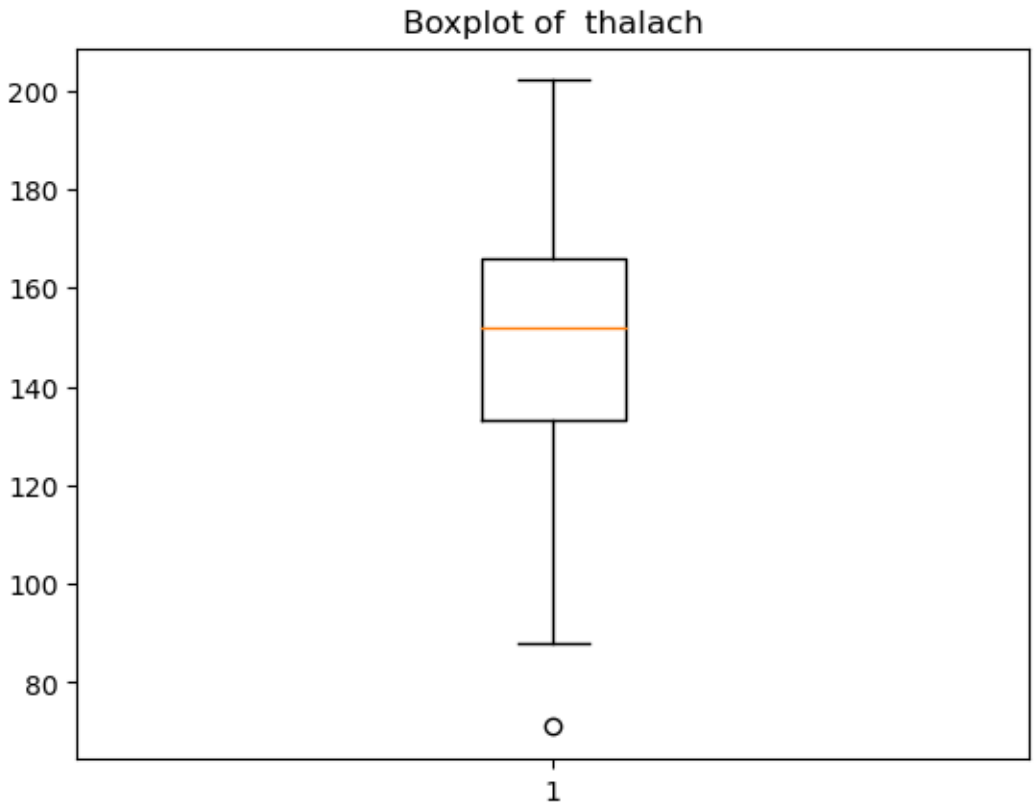
[359]:
```
plt.boxplot(data['chol'])
plt.title("Boxplot of  chol")
```

[359]:
```
Text(0.5, 1.0, 'Boxplot of  chol')
```



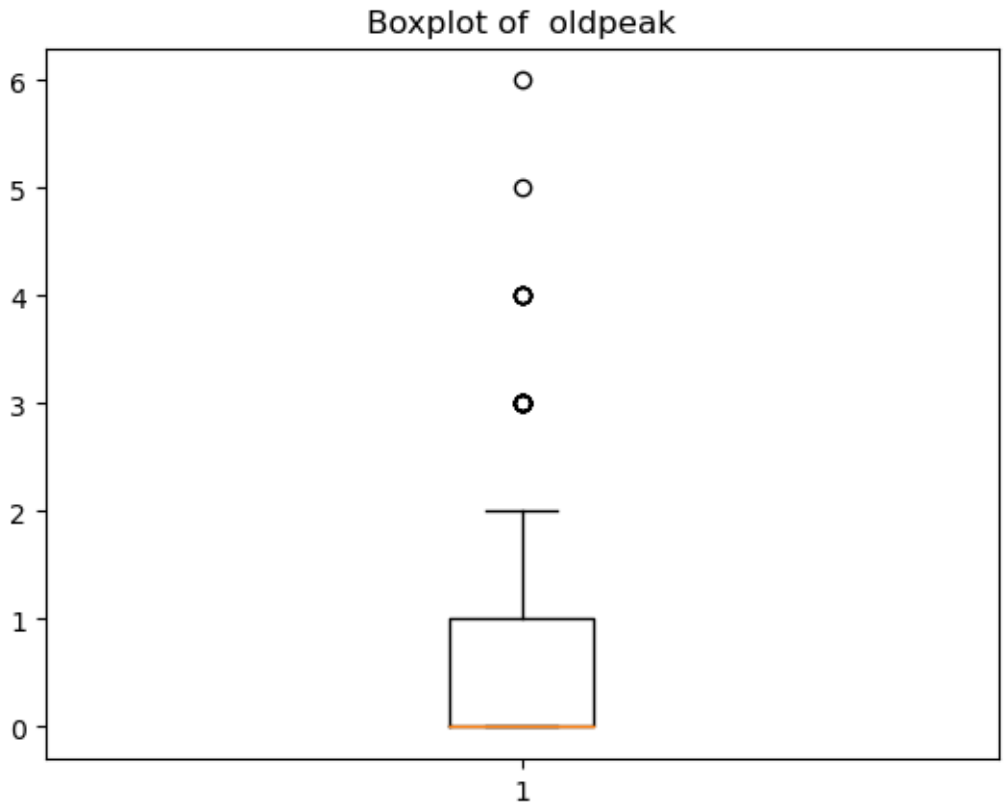
[360]:
```
plt.boxplot(data['thalach'])
plt.title("Boxplot of  thalach")
```

[360]:
```
Text(0.5, 1.0, 'Boxplot of  thalach')
```



[361]:
```
plt.boxplot(data['oldpeak'])
plt.title("Boxplot of  oldpeak")
```

[361]:
```
Text(0.5, 1.0, 'Boxplot of  oldpeak')
```



8 dealing with outliers

[ ]:
```
## age has a value 0. It is abnormal value.So i am replacing it with mean.
```

[362]:
```
age_mean = round(data["age"].mean())
```

[363]:
```
data["age"]=data["age"].apply(lambda x: age_mean if x == 0 else x)
```

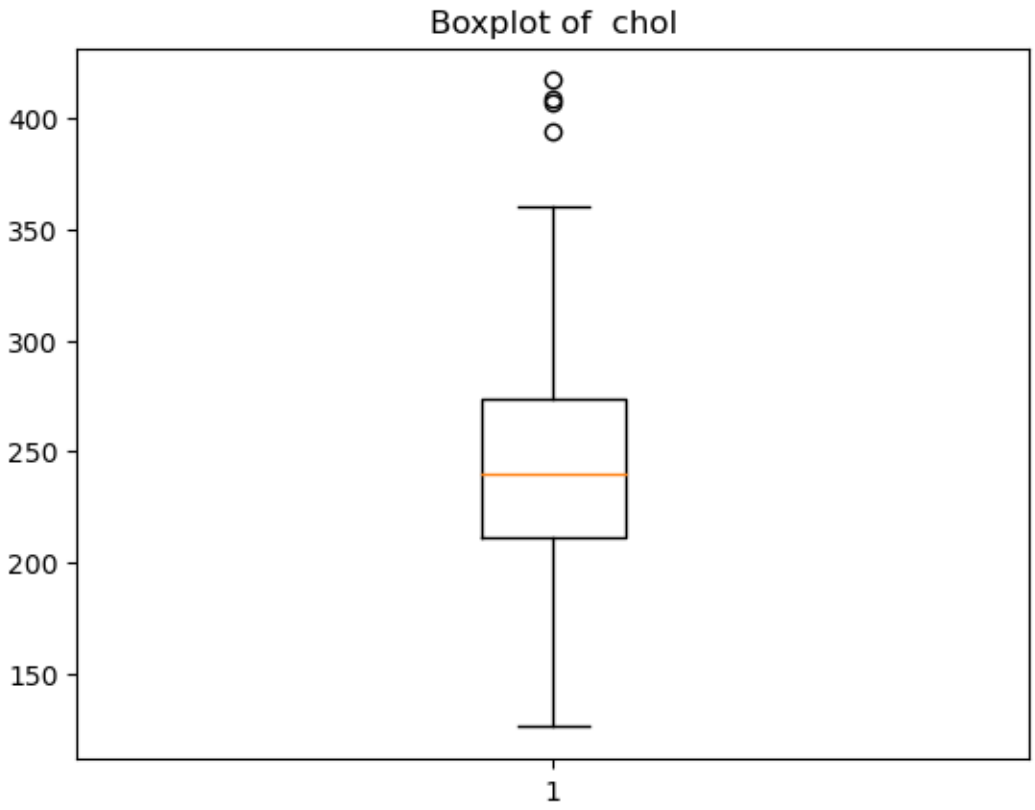
[364]:
```
# chol has four will replace last outlier that may change the distribution.
```

[365]:
```
chol_m = round(data["chol"].mean())
```

[366]:
```
data["chol"]=data["chol"].apply(lambda x: chol_m if x > 500 else x)
```

[367]:
```
plt.boxplot(data['chol'])
plt.title("Boxplot of  chol")
```

[367]:
```
Text(0.5, 1.0, 'Boxplot of  chol')
```



[368]:
```
## thalach has a value one outlier. It is abnormal value.So i am replacing it␣
↪with mean.
```

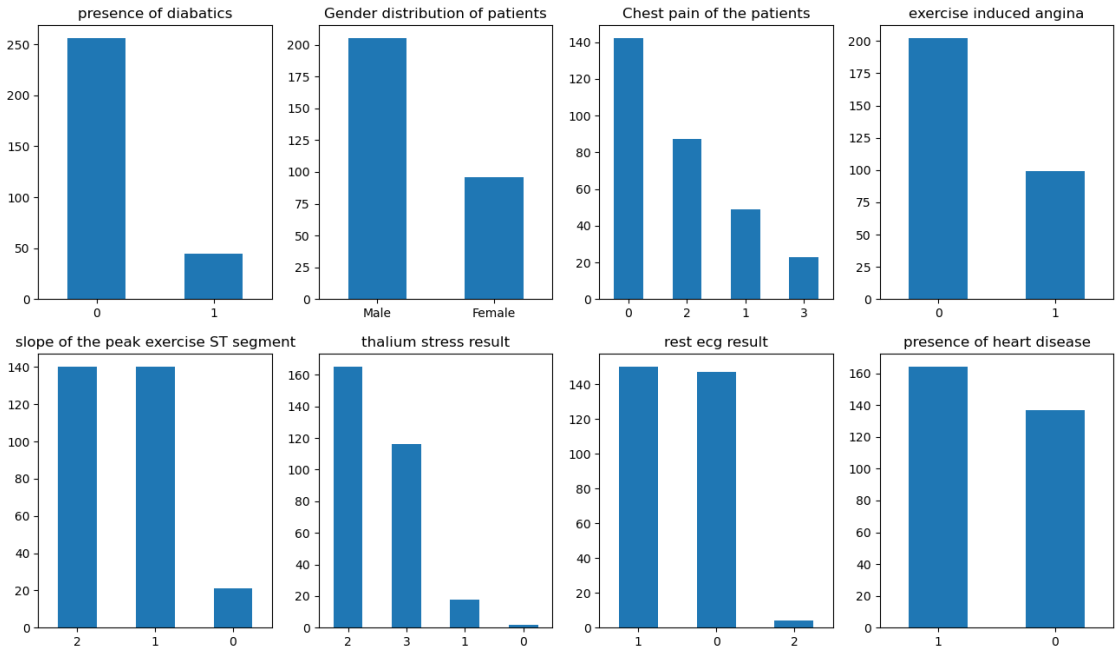
[369]:
```
thalach_m = round(data["thalach"].mean())
```

[370]:
```
data["thalach"]=data["thalach"].apply(lambda x: thalach_m if x < 70 else x)
```

9 Univariate Analysis

[371]:
```
plt.subplot(241)

data['fbs'].value_counts().plot(kind='bar', title='presence of diabatics',␣
↪figsize=(16,9))

plt.xticks(rotation=0)

plt.subplot(242)

data['sex'].value_counts().plot(kind='bar', title='Gender distribution of␣
↪patients', figsize=(16,9))
```

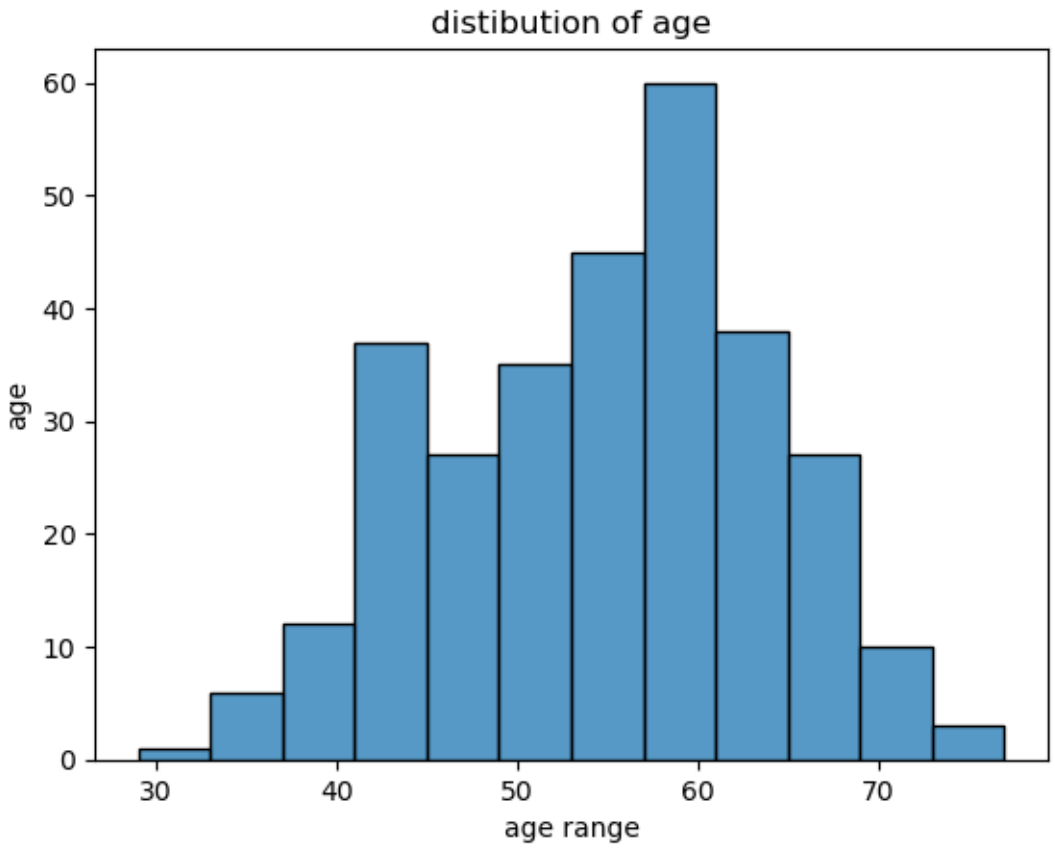
```
plt.xticks(rotation=0)

plt.subplot(243)

data['cp'].value_counts().plot(kind='bar', title='Chest pain of the patients',␣
 ↪figsize=(16,9))

plt.xticks(rotation=0)

plt.subplot(244)

data['exang'].value_counts().plot(kind='bar', title='exercise induced angina',␣
 ↪figsize=(16,9))

plt.xticks(rotation=0)

plt.subplot(245)

data['slope'].value_counts().plot(kind='bar', title='slope of the peak exercise␣
 ↪ST segment', figsize=(16,9))

plt.xticks(rotation=0)

plt.subplot(246)

data['thal'].value_counts().plot(kind='bar', title='thalium stress result',␣
 ↪figsize=(16,9))

plt.xticks(rotation=0)
plt.subplot(247)

data['restecg'].value_counts().plot(kind='bar', title='rest ecg result',␣
 ↪figsize=(16,9))

plt.xticks(rotation=0)


plt.subplot(248)

data['target'].value_counts().plot(kind='bar', title='presence of heart␣
 ↪disease', figsize=(16,9))

plt.xticks(rotation=0)

plt.show()
```



10 observation from the above plots

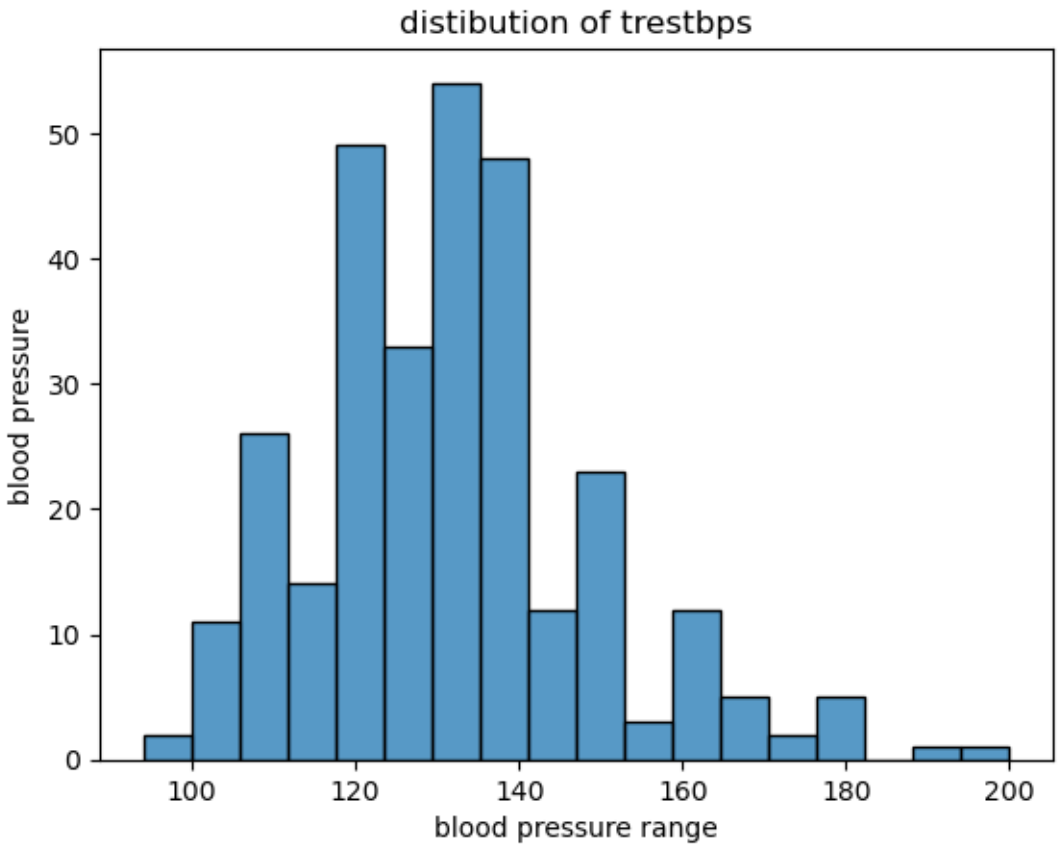
1.these are very less patients having diabatics.
2.The ratio of male and female patients is 2:1.
3.majority of the patients diagnosed chest pain as is typical.
4.Majority of the patients has 0 vessels filled.

[372]:
```
# numerical variables
# age
# histogram representation of age
sns.histplot(data,x="age")        # create histogram of the "age"
plt.title("distibution of age")
plt.xlabel("age range")
plt.ylabel("age")
plt.show()
```



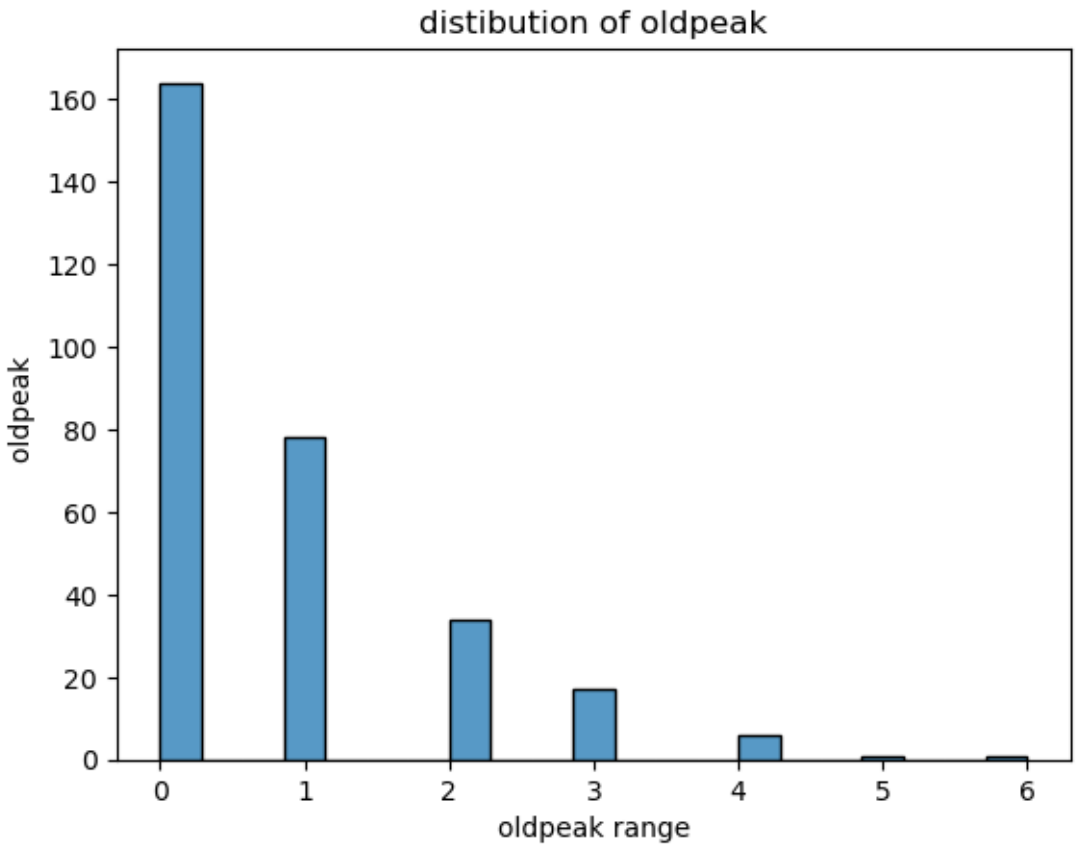
[ ]:
```
# observation :# age is normally distributed
```

[373]:
```
sns.histplot(data,x="trestbps")          # create histogram of the "age"
plt.title("distibution of trestbps")
plt.xlabel("blood pressure range")
plt.ylabel("blood pressure")
plt.show()
```



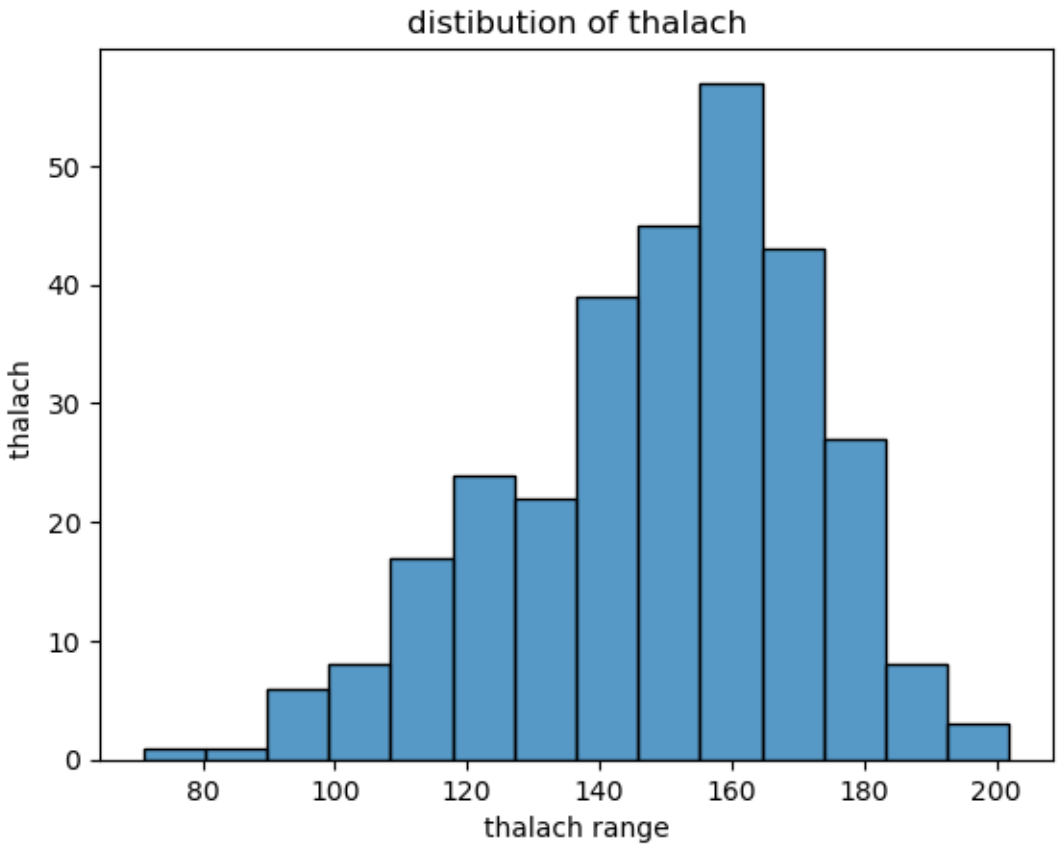
[211]:
```
#observartion:=> trestbps data positively skewed
```

[374]:
```
sns.histplot(data,x="oldpeak")          # create histogram of the "age"
plt.title("distibution of oldpeak")
plt.xlabel("oldpeak range")
plt.ylabel("oldpeak ")
plt.show()
```



[ ]:
```
# Observation : it is clear positievly skewed
```

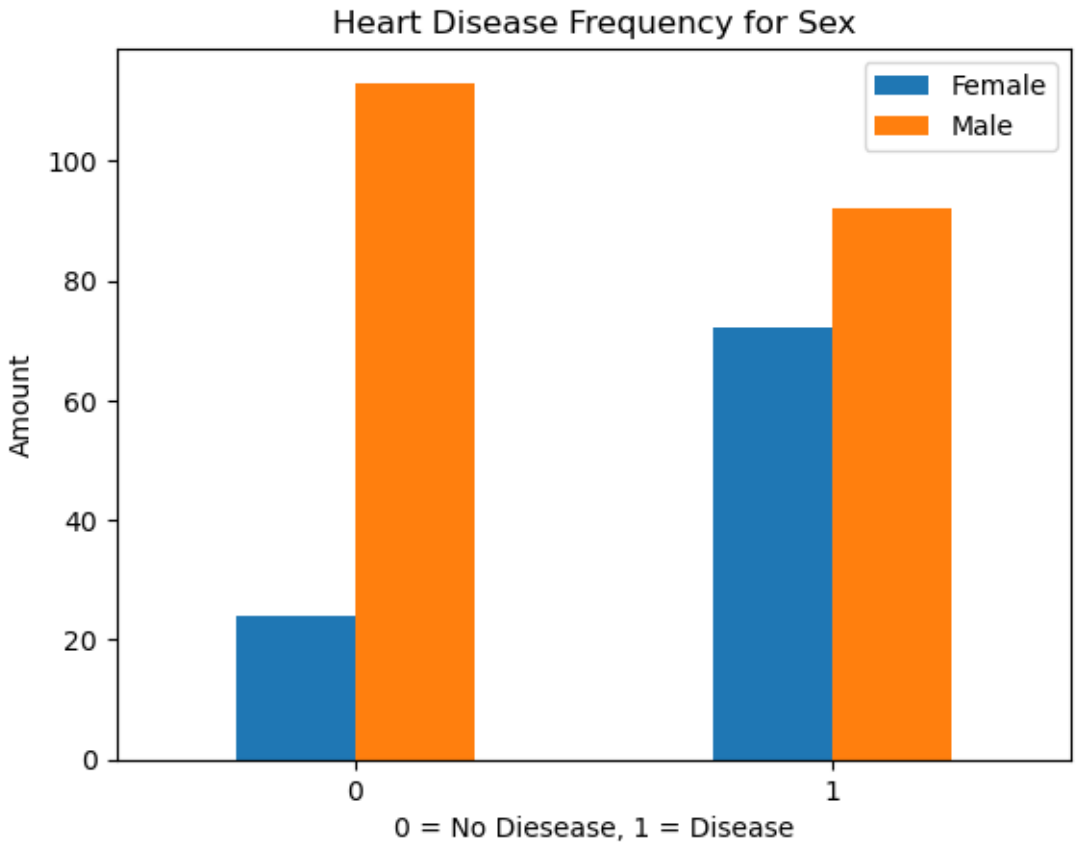
[375]:
```
sns.histplot(data,x="thalach")        # create histogram of the "age"
plt.title("distibution of thalach")
plt.xlabel("thalach range")
plt.ylabel("thalach ")
plt.show()
```



[ ]:
```
#it is negatively skewed
```

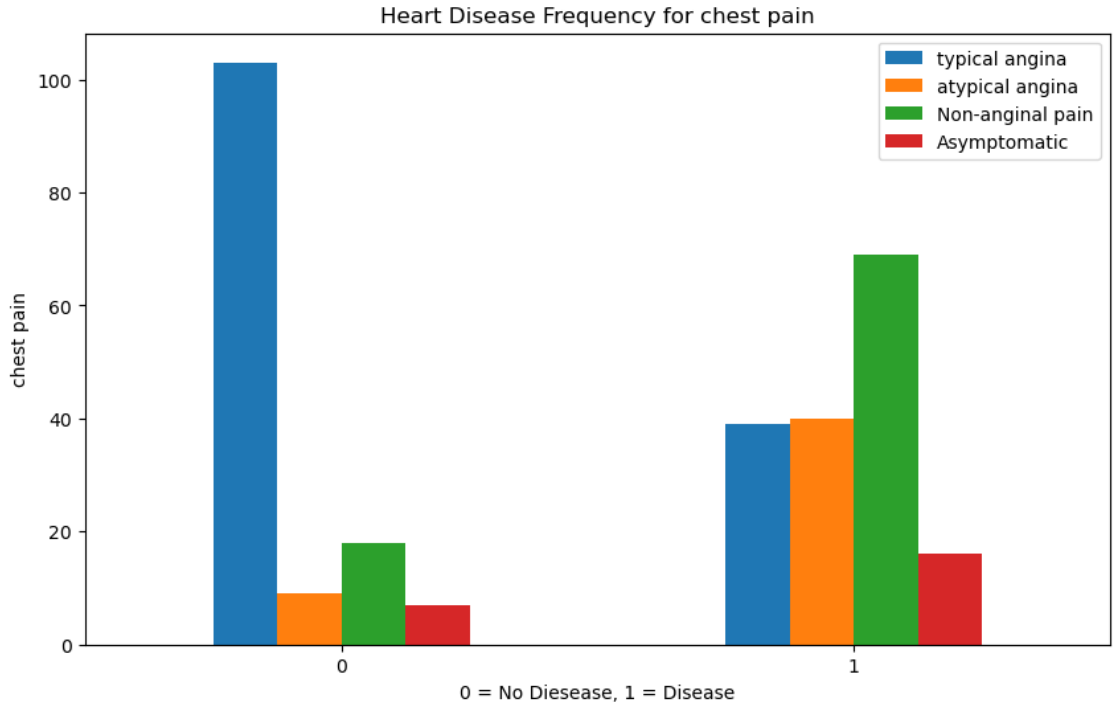
11 ## Bivariate Analysis

[376]:
```
pd.crosstab(data.target, data.sex).plot(kind='bar')
plt.title("Heart Disease Frequency for Sex")
plt.xlabel("0 = No Diesease, 1 = Disease")
plt.ylabel("Amount")
plt.legend(["Female", "Male"]);
plt.xticks(rotation=0);
```



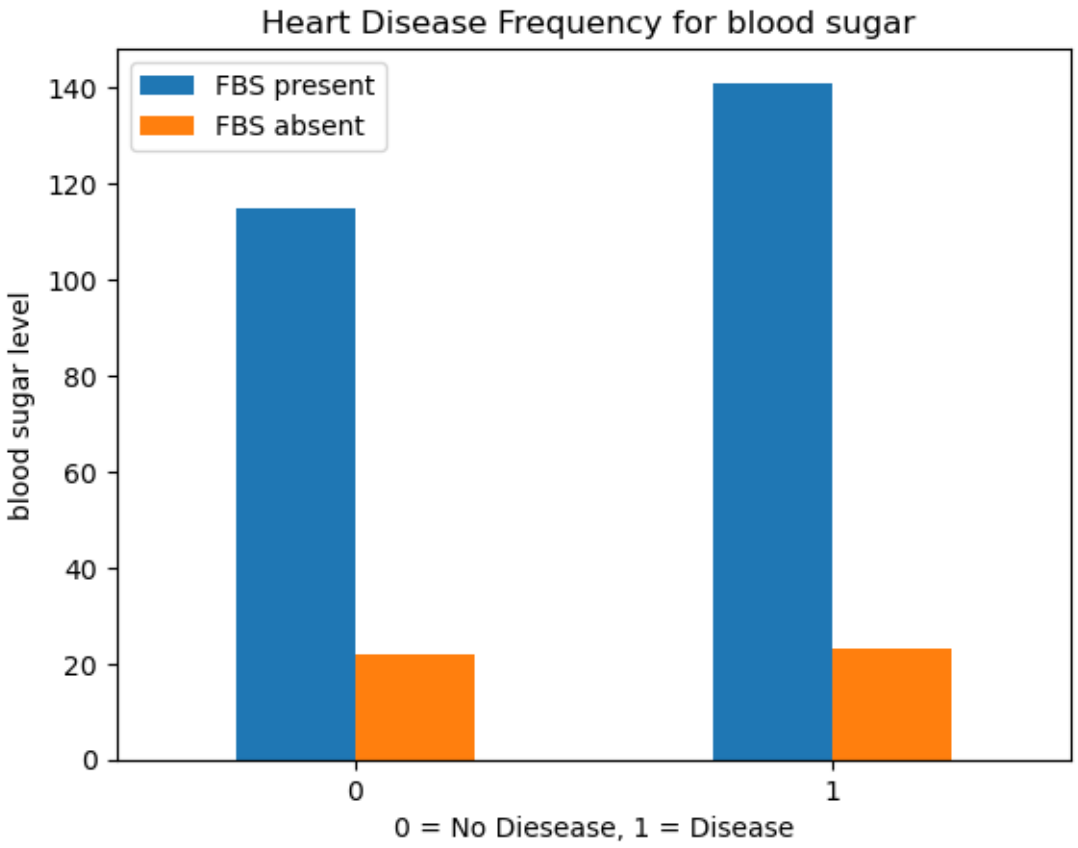
with the data and ratio of the male and female gender, we can say that female category has more heart disease compared to male.

[377]:
```
pd.crosstab(data.target, data.cp).plot(kind='bar',figsize=(10, 6))
plt.title("Heart Disease Frequency for chest pain")
plt.xlabel("0 = No Diesease, 1 = Disease")
plt.ylabel("chest pain")
plt.legend(["typical angina", "atypical angina","Non-anginal␣
 ↪pain","Asymptomatic"]);
plt.xticks(rotation=0);
```



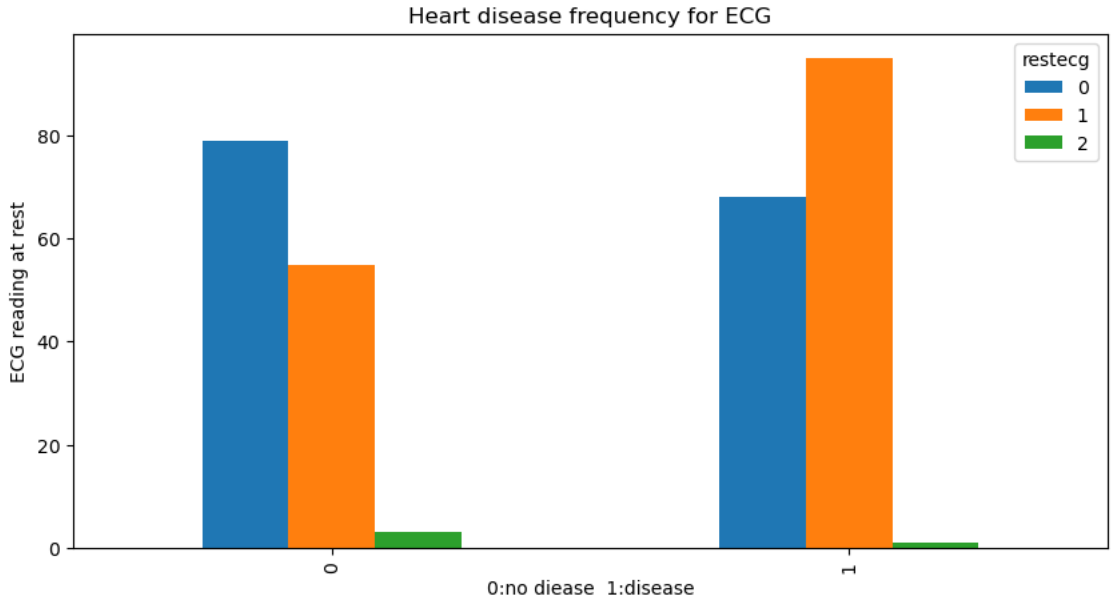
#People having atypical angina and non anginal pain has high probability of heart disease

[378]:
```
pd.crosstab( data.target,data.fbs).plot(kind='bar')
plt.title("Heart Disease Frequency for blood sugar ")
plt.xlabel("0 = No Diesease, 1 = Disease")
plt.ylabel("blood sugar level")
plt.legend(["FBS present", "FBS absent"]);
plt.xticks(rotation=0);
plt.show()
```

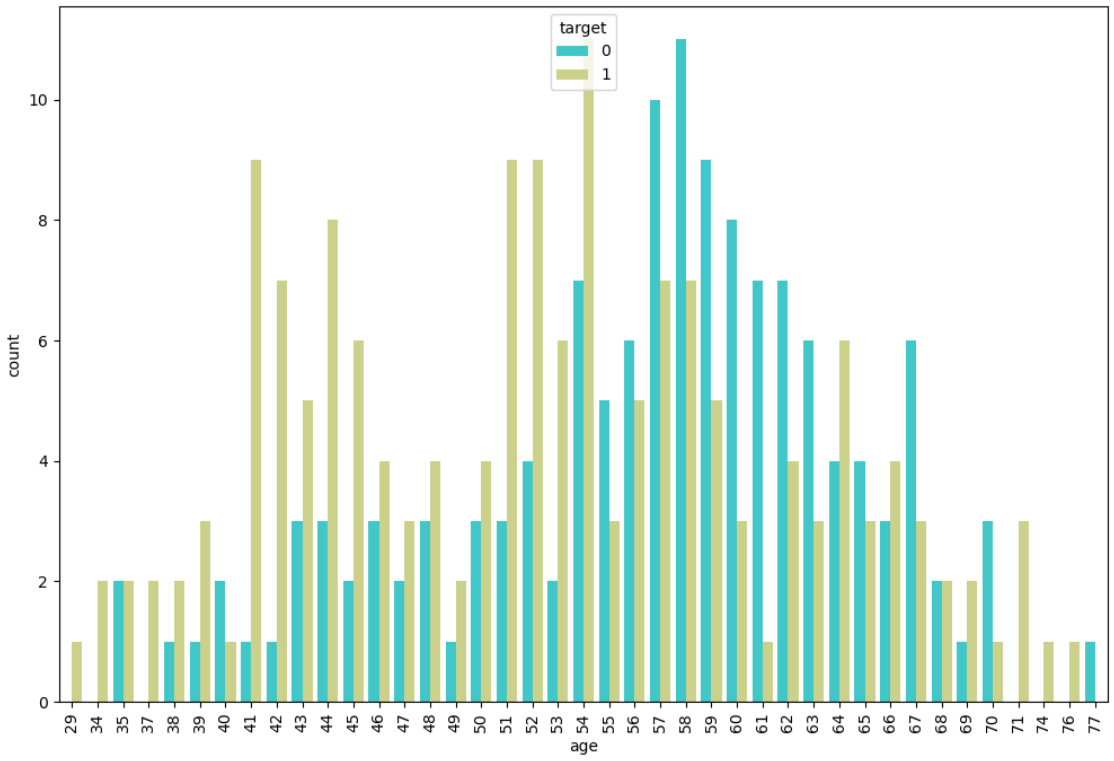


[ ]:
```
# if FBS is present that is having a little impact on heart diease
```

[379]:
```
pd.crosstab(data.target,data.restecg).plot(kind='bar',figsize=(10,5))
plt.title("Heart disease frequency for ECG")
plt.xlabel("0:no diease  1:disease")
plt.ylabel("ECG reading at rest")
plt.show()
```

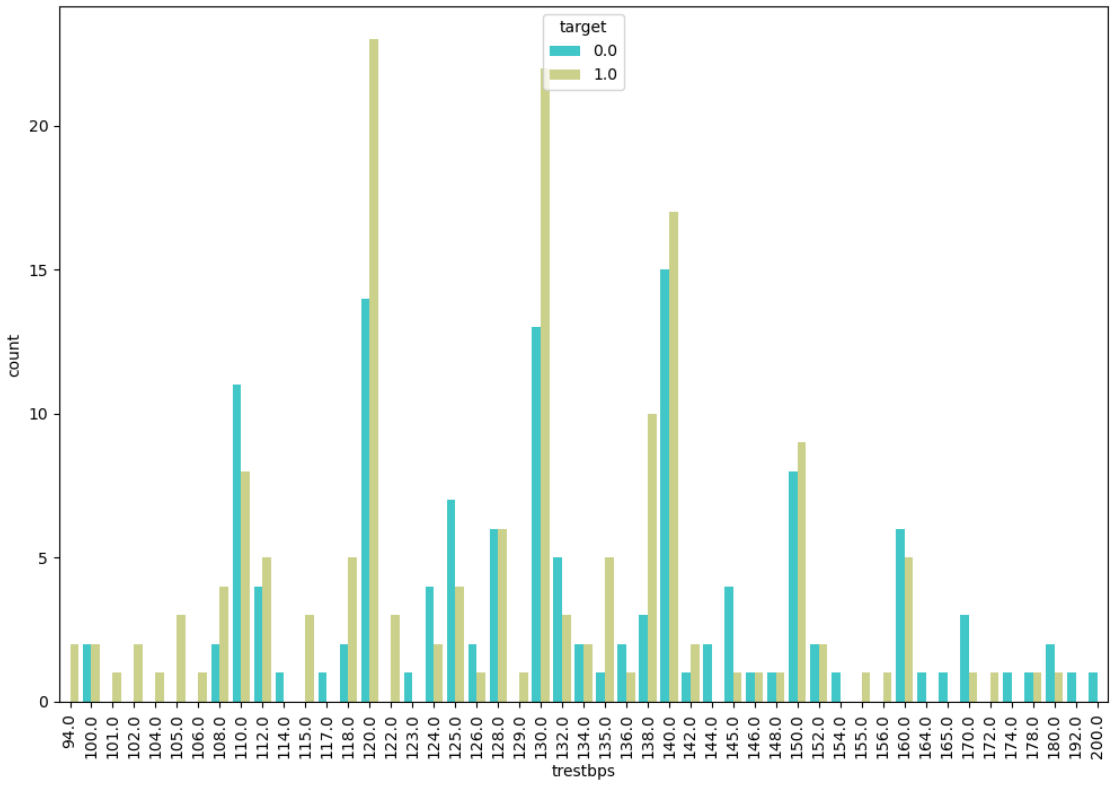


[380]:
```
fig, ax1 = plt.subplots (figsize= (12,8))
graph = sns.countplot (ax=ax1, data=data,x =␣
 ↪"age",hue="target",palette="rainbow")
graph.set_xticklabels (graph.get_xticklabels(),rotation=90)
for P in graph.patches:
     height = P.get_height()
```



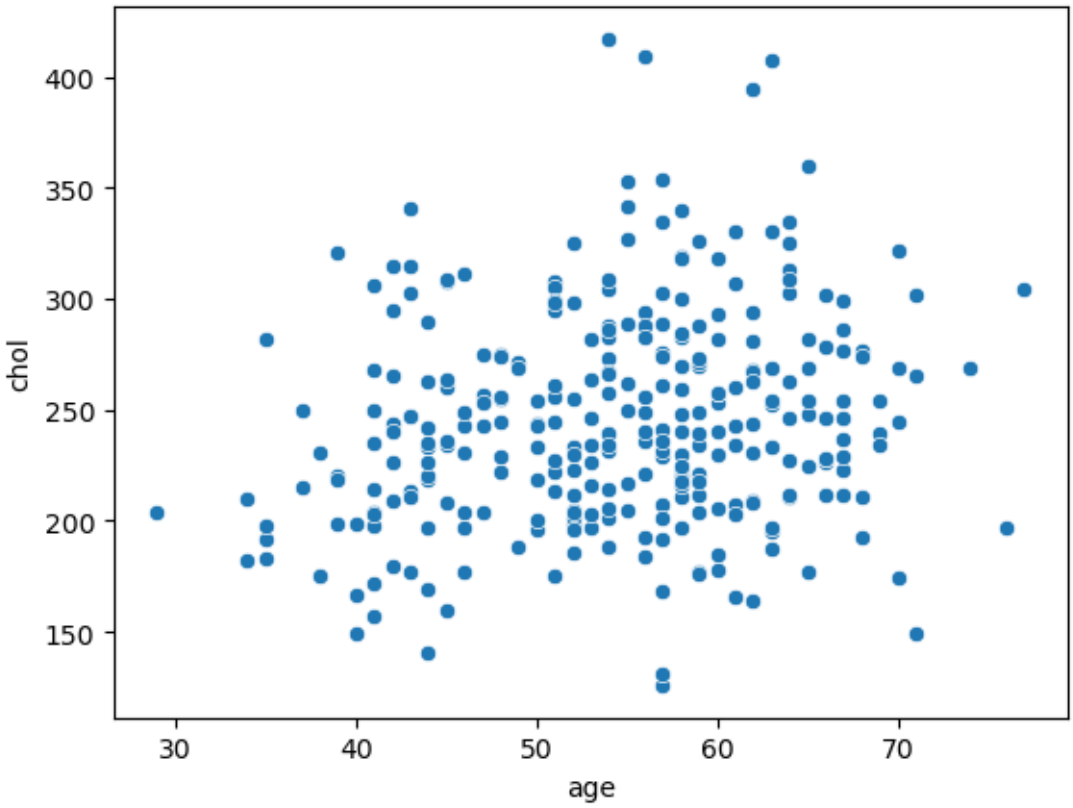
we can see that from age 41 to 60 has high heart disease patients

[48]:
```
fig, ax1 = plt.subplots (figsize= (12,8))
graph = sns.countplot (ax=ax1, data=data,x =␣
  ↪"trestbps",hue="target",palette="rainbow")
graph.set_xticklabels (graph.get_xticklabels(),rotation=90)
for P in graph.patches:
     height = P.get_height()
```



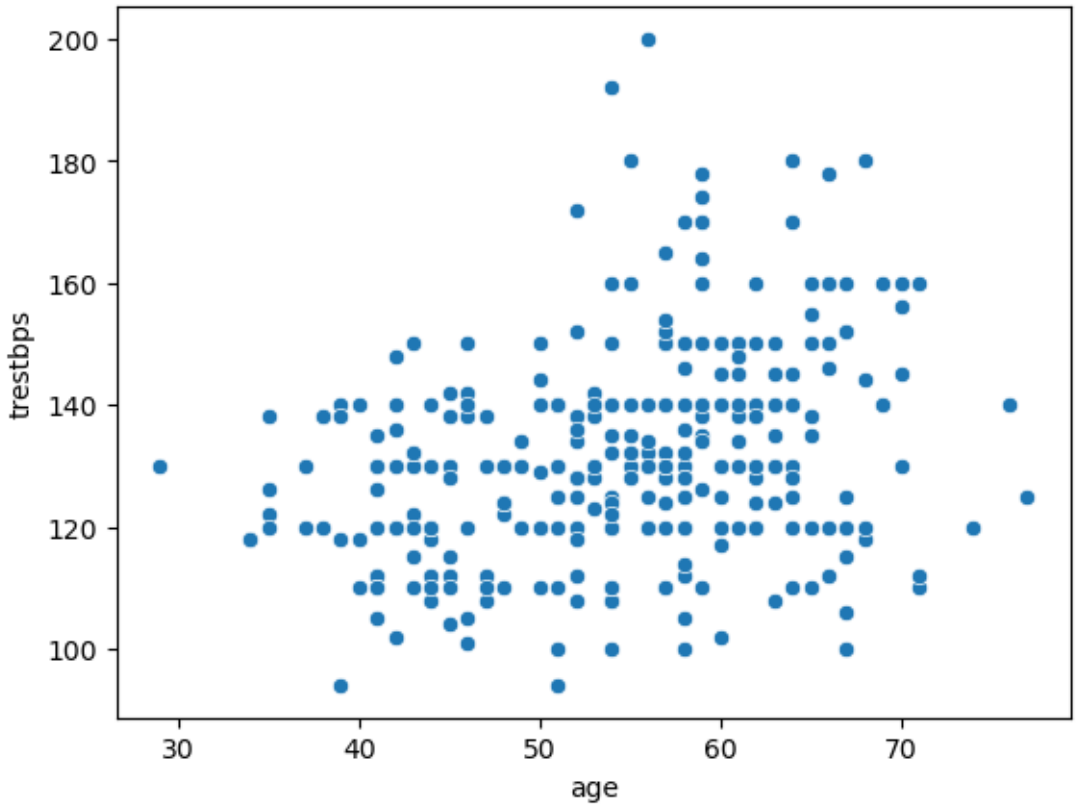
[381]:
```
sns.scatterplot (x = data["age"],y= data["chol"])
```

[381]: <AxesSubplot:xlabel='age', ylabel='chol'>



[50]:
```
sns.scatterplot(x = data["age"],y= data["trestbps"])
```

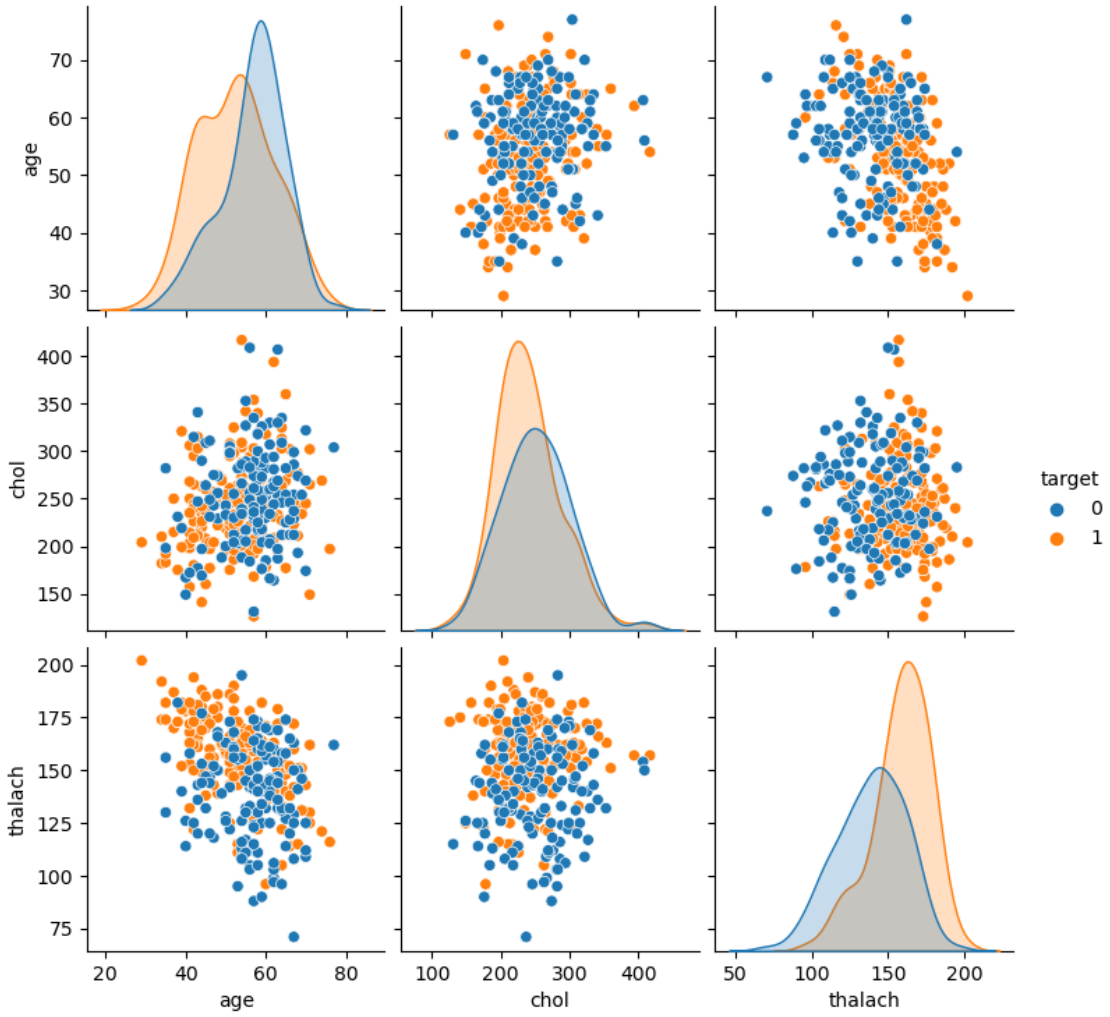
[50]: <AxesSubplot:xlabel='age', ylabel='trestbps'>



[ ]:
```
#no perticular relation
```

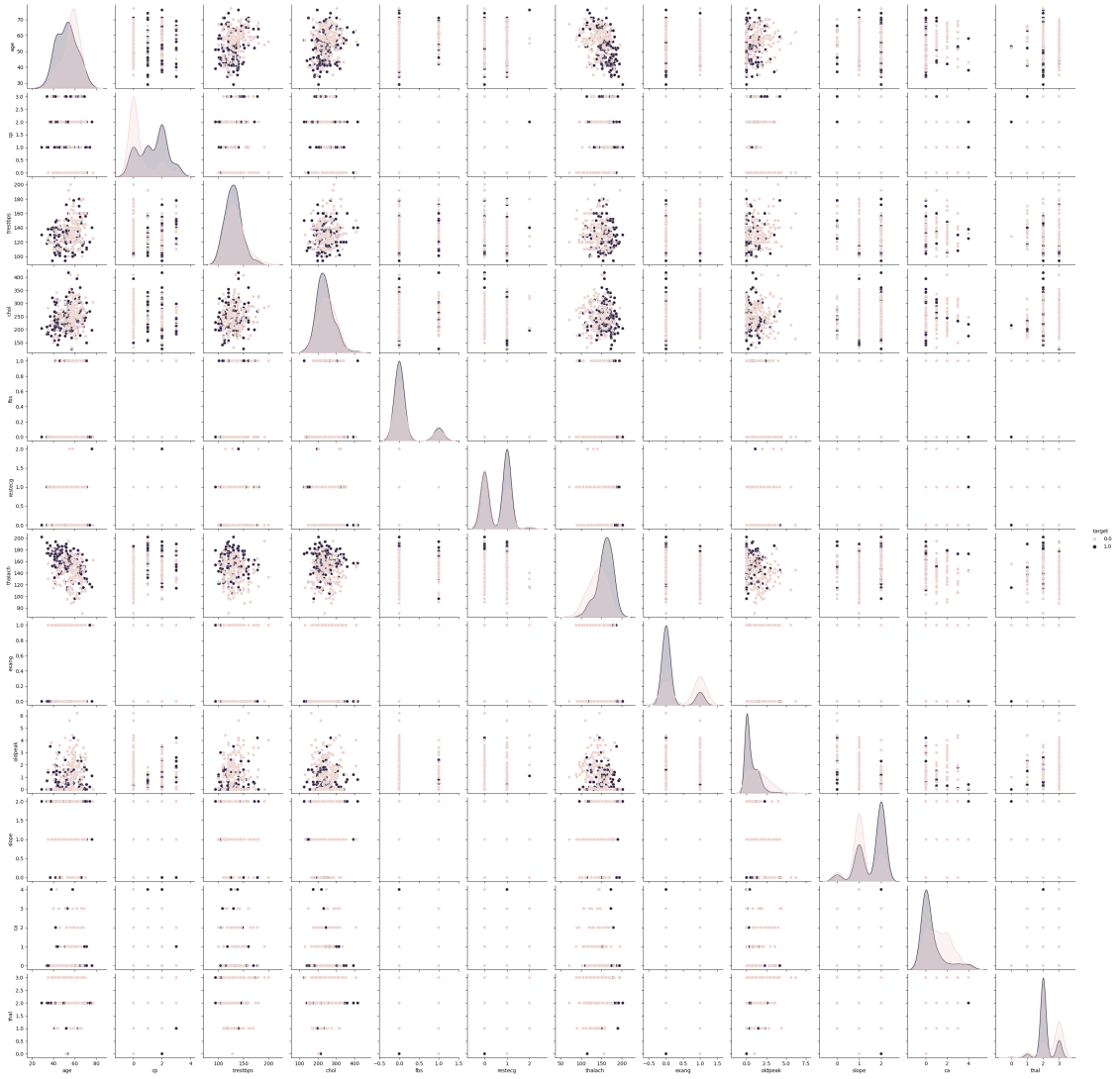
12 multivariate analysis

[382]:
```
# in multivariate analysis   use pairplot
sns.pairplot(data, vars=['age','chol','thalach'], hue='target')
#plt.title('Age, Sex vs. Heart Disease')
plt.show()
```



[54]:
```
#sns.pairplot(data,hue = "target")
```

[54]: <seaborn.axisgrid.PairGrid at 0x139d3033130>

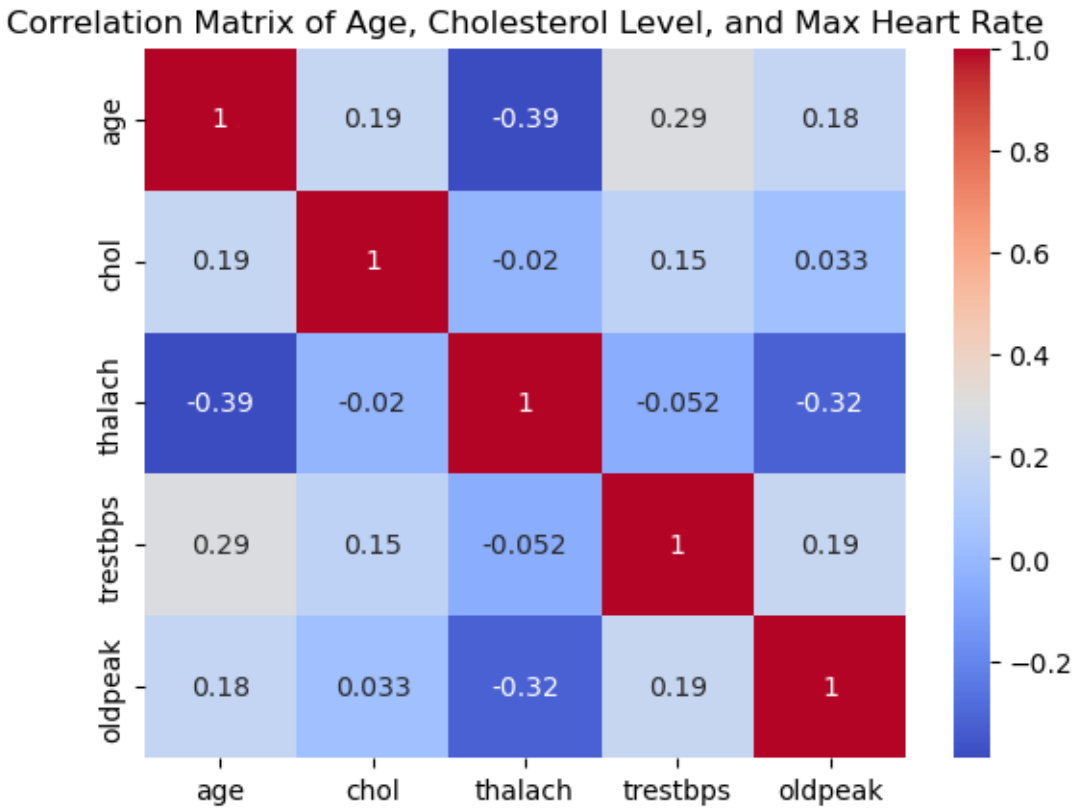


[383]:
```
corr_matrix=data[['age','chol','thalach','trestbps','oldpeak']].corr()
print("correlation matrix")
print(corr_matrix)
```

```
correlation matrix
                age      chol   thalach  trestbps   oldpeak
age        1.000000  0.190107 -0.391074  0.290238  0.180489
chol       0.190107  1.000000 -0.019791  0.154596  0.033182
thalach   -0.391074 -0.019791  1.000000 -0.052060 -0.324179
trestbps   0.290238  0.154596 -0.052060  1.000000  0.193591
oldpeak    0.180489  0.033182 -0.324179  0.193591  1.000000
```

[384]:
```
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')
plt.title("Correlation Matrix of Age, Cholesterol Level, and Max Heart Rate")
```

```
plt.show()
```



#This matrix shows the linear relationships between these variables, with values ranging from -1 (perfect negative correlation) to 1 (perfect positive correlation).

13 Task 3

[385]:
```
from sklearn.compose import ColumnTransformer
from sklearn.impute import SimpleImputer
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler, OneHotEncoder
```

[388]:
```
## Separate numeric and categorical features

numeric_features = ['age', 'trestbps', 'chol', 'thalach', 'oldpeak']
categorical_features = [ 'sex','cp', 'fbs', 'restecg', 'exang', 'slope', ␣
 ↪'thal']
```

[389]:
```
# Create transformers for numeric and categorical features

```

```
numeric_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='mean')),
    ('scaler', StandardScaler())
])
```

[390]:
```
categorical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('encoder', OneHotEncoder(handle_unknown='ignore'))
])
```

[391]:
```
# Use ColumnTransformer to apply appropriate transformers to each column
preprocessor = ColumnTransformer(transformers=[
    ('num', numeric_transformer, numeric_features),
    ('cat', categorical_transformer, categorical_features)
])
```

[392]:
```
transformed_data # it is as an array
```

[392]:
```
array([[ 0.95700217,  0.76798036, -0.25056818, …,  1.        ,
         0.        ,  0.        ],
       [-1.90452907, -0.08923381,  0.1001453 , …,  0.        ,
         1.        ,  0.        ],
       [-1.46429349, -0.08923381, -0.8488441 , …,  0.        ,
         1.        ,  0.        ],
       …,
       [ 1.50729664,  0.71083274, -1.07577634, …,  0.        ,
         0.        ,  1.        ],
       [ 0.29664881, -0.08923381, -2.354849  , …,  0.        ,
         0.        ,  1.        ],
       [ 0.29664881, -0.08923381, -0.18867756, …,  0.        ,
         1.        ,  0.        ]])
```

[292]:
```
# encoders for categorical features.
```

[437]:
```
categorical_features = [ 'cp', 'fbs', 'restecg', 'exang', 'slope', 'thal']
```

[438]:
```
categorical_features
```

[438]:
```
['cp', 'fbs', 'restecg', 'exang', 'slope', 'thal']
```

[396]:
```
# Create an instance of OneHotEncoder
encoder = OneHotEncoder(handle_unknown='ignore')
```

[439]:
```
# Fit and transform the categorical features
encoded_features = encoder.fit_transform(data[categorical_features])

# Convert the encoded features to a DataFrame
```

```
encoded_df = pd.DataFrame(encoded_features.toarray(), columns=encoder.
 ↪get_feature_names_out(categorical_features))

# Concatenate the encoded features with the original dataset
heart_disease_encoded = pd.concat([data.drop(columns=categorical_features),␣
 ↪encoded_df], axis=1)

# The heart_disease_encoded DataFrame now contains the encoded categorical␣
 ↪features
```

[440]:
```
heart_disease_encoded
```

[440]:
```
      age     sex  trestbps   chol  thalach  oldpeak  target  cp_0  cp_1  \
0    63.0    Male     145.0  233.0    150.0      2.0     1.0   0.0   0.0
1    37.0    Male     130.0  250.0    187.0      3.0     1.0   0.0   0.0
2    41.0  Female     130.0  204.0    172.0      1.0     1.0   0.0   1.0
3    56.0    Male     120.0  236.0    178.0      0.0     1.0   0.0   1.0
4    57.0  Female     120.0  354.0    163.0      0.0     1.0   1.0   0.0
..    …      …        …      …        …        …      …     …     …
300  68.0    Male     144.0  193.0    141.0      3.0     0.0   0.0   1.0
301  57.0    Male     130.0  131.0    115.0      1.0     0.0   NaN   NaN
302  57.0  Female     130.0  236.0    174.0      0.0     0.0   NaN   NaN
12    NaN     NaN       NaN    NaN      NaN      NaN     NaN   0.0   0.0
283   NaN     NaN       NaN    NaN      NaN      NaN     NaN   1.0   0.0

     cp_2  …  restecg_2  exang_0  exang_1  slope_0  slope_1  slope_2  \
0     0.0  …        0.0      1.0      0.0      1.0      0.0      0.0
1     1.0  …        0.0      1.0      0.0      1.0      0.0      0.0
2     0.0  …        0.0      1.0      0.0      0.0      0.0      1.0
3     0.0  …        0.0      1.0      0.0      0.0      0.0      1.0
4     0.0  …        0.0      0.0      1.0      0.0      0.0      1.0
..    …  …          …        …        …        …        …        …
300   0.0  …        0.0      1.0      0.0      0.0      1.0      0.0
301   NaN  …        NaN      NaN      NaN      NaN      NaN      NaN
302   NaN  …        NaN      NaN      NaN      NaN      NaN      NaN
12    0.0  …        0.0      0.0      1.0      0.0      1.0      0.0
283   0.0  …        0.0      0.0      1.0      0.0      1.0      0.0

     thal_0  thal_1  thal_2  thal_3
0       0.0     1.0     0.0     0.0
1       0.0     0.0     1.0     0.0
2       0.0     0.0     1.0     0.0
3       0.0     0.0     1.0     0.0
4       0.0     0.0     1.0     0.0
..      …       …       …       …
300     0.0     0.0     1.0     0.0
301     NaN     NaN     NaN     NaN
```

```
302     NaN     NaN     NaN     NaN
12      0.0     0.0     1.0     0.0
283     0.0     0.0     0.0     1.0

[303 rows x 25 columns]
```

[441]:
```
# Select the features to be standardized
features_to_scale = ['age', 'trestbps', 'chol', 'thalach', 'oldpeak']
```

[442]:
```
# Create an instance of StandardScaler
scaler = StandardScaler()
```

[443]:
```
# Fit and transform the selected features
scaled_features = scaler.fit_transform(data[features_to_scale])
```

[444]:
```
scaled_features
```

[444]:
```
array([[ 0.95413119,  0.76670781, -0.25175562,  0.02312975,  1.14787329],
       [-1.91449375, -0.08908595,  0.09849819,  1.64323059,  2.08168372],
       [-1.47316683, -0.08908595, -0.8492474 ,  0.98643295,  0.21406286],
       …,
       [ 1.50578984,  0.70965489, -1.07588221, -0.37094883,  2.08168372],
       [ 0.29214082, -0.08908595, -2.35327843, -1.50939807,  0.21406286],
       [ 0.29214082, -0.08908595, -0.18994612,  1.07400597, -0.71974758]])
```

[447]:
```
# Create a DataFrame with the scale features
scaled_df = pd.DataFrame(scaled_features, columns=features_to_scale)

# Replace the original features with the scaled features in the dataset
heart_disease_scaled = pd.concat([data.drop(columns=features_to_scale),␣
 ↪scaled_df], axis=1)

# The heart_disease_scaled DataFrame now contains the standardized features
```

[448]:
```
heart_disease_scaled
```

[448]:
```
        sex   cp  fbs  restecg  exang  slope  thal  target       age  \
0      Male  3.0  1.0      0.0    0.0    0.0   1.0     1.0  0.954131
1      Male  2.0  0.0      1.0    0.0    0.0   2.0     1.0 -1.914494
2    Female  1.0  0.0      0.0    0.0    2.0   2.0     1.0 -1.473167
3      Male  1.0  0.0      1.0    0.0    2.0   2.0     1.0  0.181809
4    Female  0.0  0.0      1.0    1.0    2.0   2.0     1.0  0.292141
..      …  …  …       …    …    …   …     …       …
300    Male  0.0  1.0      1.0    0.0    1.0   3.0     0.0  0.292141
301    Male  0.0  0.0      1.0    1.0    1.0   3.0     0.0       NaN
302  Female  1.0  0.0      0.0    0.0    1.0   2.0     0.0       NaN
12      NaN  NaN  NaN      NaN    NaN    NaN   NaN     NaN  1.064463
```

```
283     NaN  NaN  NaN      NaN    NaN    NaN   NaN     NaN -0.921508

     trestbps      chol   thalach   oldpeak
0    0.766708 -0.251756  0.023130  1.147873
1   -0.089086  0.098498  1.643231  2.081684
2   -0.089086 -0.849247  0.986433  0.214063
3   -0.659615 -0.189946  1.249152 -0.719748
4   -0.659615  2.241227  0.592354 -0.719748
..        …       …        …       …
300 -0.089086 -0.189946  1.074006 -0.719748
301       NaN       NaN       NaN       NaN
302       NaN       NaN       NaN       NaN
12  -1.230144 -0.705025 -0.239589  0.214063
283  0.481443  1.355291 -1.290466  0.214063

[303 rows x 13 columns]
```

[449]:
```
heart_disease_scaled.dropna(inplace=True)
```

[450]:
```
heart_disease_scaled
```

[450]:
```
        sex   cp  fbs  restecg  exang  slope  thal  target       age  \
0      Male  3.0  1.0      0.0    0.0    0.0   1.0     1.0  0.954131
1      Male  2.0  0.0      1.0    0.0    0.0   2.0     1.0 -1.914494
2    Female  1.0  0.0      0.0    0.0    2.0   2.0     1.0 -1.473167
3      Male  1.0  0.0      1.0    0.0    2.0   2.0     1.0  0.181809
4    Female  0.0  0.0      1.0    1.0    2.0   2.0     1.0  0.292141
..      …  …  …        …      …    …   …     …       …
296  Female  0.0  0.0      1.0    1.0    1.0   2.0     0.0  0.292141
297    Male  0.0  1.0      0.0    0.0    1.0   1.0     0.0 -1.031840
298  Female  0.0  0.0      1.0    1.0    1.0   3.0     0.0  1.505790
299    Male  3.0  0.0      1.0    0.0    1.0   3.0     0.0  0.292141
300    Male  0.0  1.0      1.0    0.0    1.0   3.0     0.0  0.292141

     trestbps      chol   thalach   oldpeak
0    0.766708 -0.251756  0.023130  1.147873
1   -0.089086  0.098498  1.643231  2.081684
2   -0.089086 -0.849247  0.986433  0.214063
3   -0.659615 -0.189946  1.249152 -0.719748
4   -0.659615  2.241227  0.592354 -0.719748
..        …       …        …       …
296  0.481443 -0.086930 -1.159106 -0.719748
297 -1.230144  0.386942 -0.765027  0.214063
298  0.709655 -1.075882 -0.370949  2.081684
299 -0.089086 -2.353278 -1.509398  0.214063
300 -0.089086 -0.189946  1.074006 -0.719748
```

```
[299 rows x 13 columns]
```

14 Pipeline

[451]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
```

[452]:
```
# Select the features and target variable
features = data.drop(columns=['target'])
target = data['target']
```

[453]:
```
#Split the dataset into train and test sets
X_train, X_test, y_train, y_test = train_test_split(features, target,␣
 ↪test_size=0.2, random_state=42)
```

[454]:
```
X_train.shape
```

[454]:
```
(240, 12)
```

[455]:
```
X_test.shape
```

[455]:
```
(61, 12)
```

[456]:
```
y_train.shape
```

[456]:
```
(240,)
```

[457]:
```
y_test.shape
```

[457]:
```
(61,)
```

[458]:
```
numeric_features = ['age', 'trestbps', 'chol', 'thalach', 'oldpeak']
categorical_features = ['cp', 'fbs', 'restecg', 'exang', 'slope', 'thal']
```

[459]:
```
# Create transformers for imputation, encoding, and scaling
imputer = SimpleImputer(strategy='most_frequent')
encoder = OneHotEncoder(handle_unknown='ignore')
scaler = StandardScaler()
```

[460]:
```
# Define the ColumnTransformer
preprocessor = ColumnTransformer(transformers=[
    ('imputer', imputer, categorical_features),
    ('encoder', encoder, categorical_features),
    ('scaler', scaler, numeric_features)
])
```

[461]:
```
# Create the pipeline with preprocessing steps and a classifier
pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('classifier', LogisticRegression())
])
```

[462]:
```
pipeline
```

[462]:
```
Pipeline(steps=[('preprocessor',
                 ColumnTransformer(transformers=[('imputer',
SimpleImputer(strategy='most_frequent'),
                                                  ['cp', 'fbs', 'restecg',
                                                   'exang', 'slope', 'thal']),
                                                 ('encoder',
OneHotEncoder(handle_unknown='ignore'),
                                                  ['cp', 'fbs', 'restecg',
                                                   'exang', 'slope', 'thal']),
                                                 ('scaler', StandardScaler(),
                                                  ['age', 'trestbps', 'chol',
                                                   'thalach', 'oldpeak'])])),
                ('classifier', LogisticRegression())])
```

[463]:
```
from  sklearn import set_config
set_config(display="diagram")
from sklearn.linear_model import LogisticRegression
```

[464]:
```
model = LogisticRegression()
```

[466]:
```
pipeline.fit(X_train, y_train)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\impute\_base.py:49:
FutureWarning: Unlike other reduction functions (e.g. `skew`, `kurtosis`), the
default behavior of `mode` typically preserves the axis it acts along. In SciPy
1.11.0, this behavior will change: the default value of `keepdims` will become
False, the `axis` over which the statistic is taken will be eliminated, and the
value None will no longer be accepted. Set `keepdims` to True or False to avoid
this warning.
  mode = stats.mode(array)
```

[466]:
```
Pipeline(steps=[('preprocessor',
                 ColumnTransformer(transformers=[('imputer',
SimpleImputer(strategy='most_frequent'),
                                                  ['cp', 'fbs', 'restecg',
                                                   'exang', 'slope', 'thal']),
                                                 ('encoder',
OneHotEncoder(handle_unknown='ignore'),
                                                  ['cp', 'fbs', 'restecg',
                                                   'exang', 'slope', 'thal']),
```

```
                                                  ('scaler', StandardScaler(),
                                                   ['age', 'trestbps', 'chol',
                                                    'thalach', 'oldpeak'])])),
                    ('classifier', LogisticRegression())])
```

[467]:
```
# Calculate the accuracy on the test data
accuracy = pipeline.score(X_test, y_test)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.8032786885245902
```