In [121]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import graphviz
from sklearn.preprocessing import LabelEncoder
from sklearn.tree import DecisionTreeClassifier, export_graphviz
from sklearn.model_selection import train_test_split
from sklearn .metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.cluster import KMeans
%matplotlib inline
import seaborn as sns
sns.set(color_codes=True)
seed = 10
```

In [86]:

```
df = pd.read_csv('C:\\Anaconda\\Lib\\site-packages\\sklearn\\datasets\\data\\heart.csv')
df.head(None)
```

Out[86]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	0
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3	0
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3	0
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3	0
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	0

303 rows × 14 columns

In [88]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       303 non-null    int64  
 1   sex       303 non-null    int64  
 2   cp        303 non-null    int64  
 3   trestbps  303 non-null    int64  
 4   chol      303 non-null    int64  
 5   fbs       303 non-null    int64  
 6   restecg   303 non-null    int64  
 7   thalach   303 non-null    int64  
 8   exang     303 non-null    int64  
 9   oldpeak   303 non-null    float64
 10  slope     303 non-null    int64  
 11  ca        303 non-null    int64  
 12  thal      303 non-null    int64  
 13  target    303 non-null    int64  
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

In [87]:

```
df.isnull().any()
```

Out[87]:

```
age         False
sex         False
cp          False
trestbps    False
chol        False
fbs         False
restecg     False
thalach     False
exang       False
oldpeak     False
slope       False
ca          False
thal        False
target      False
dtype: bool
```

In [90]:

```
df.describe()
```

Out[90]:

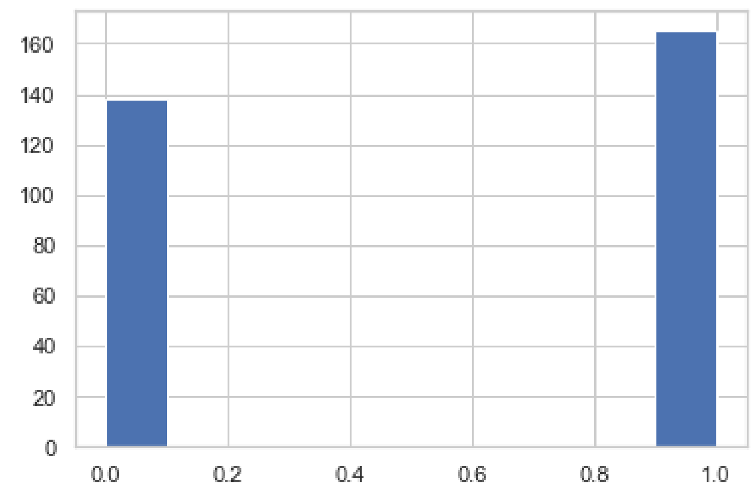
	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.646865	0.326733	1.039604	1.399340	0.729373	2.313531	0.544554
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.905161	0.469794	1.161075	0.616226	1.022606	0.612277	0.498835
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.500000	0.000000	0.000000	1.000000	0.000000	2.000000	0.000000
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.000000	0.000000	0.800000	1.000000	0.000000	2.000000	1.000000
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.000000	1.000000	1.600000	2.000000	1.000000	3.000000	1.000000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	2.000000	4.000000	3.000000	1.000000

In [105]:

```
print(df['target'].value_counts())
df['target'].hist()
```

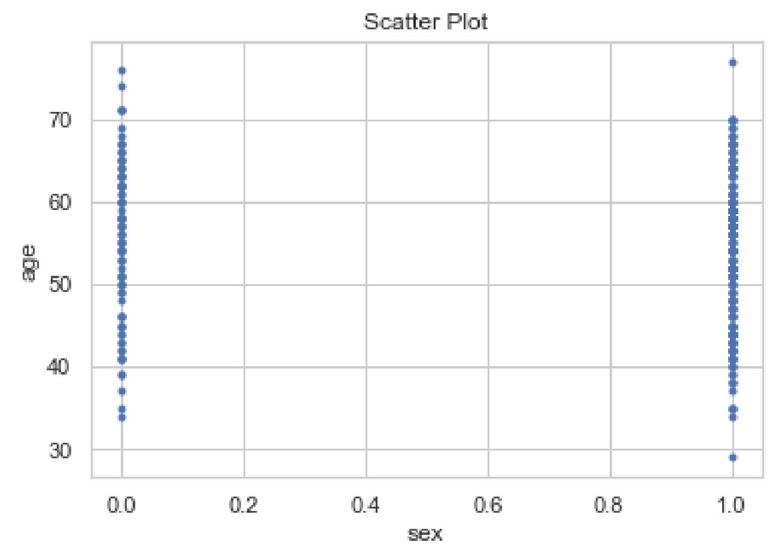
```
1    165
0    138
Name: target, dtype: int64
```

Out[105]: <AxesSubplot:>



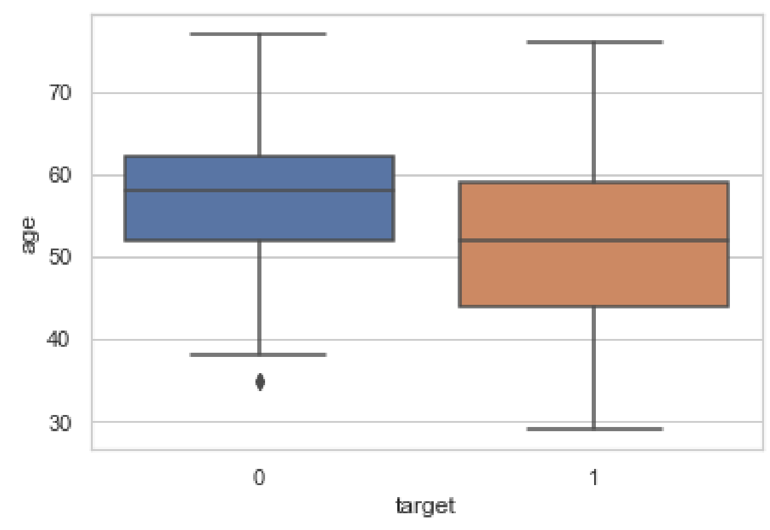
In [98]:

```
plt.scatter(slic_df[['sex' ]], slic_df[['age']], color = "b", marker = ".")
plt.xlabel('sex')
plt.ylabel('age')
plt.title('Scatter Plot')
plt.show()
```

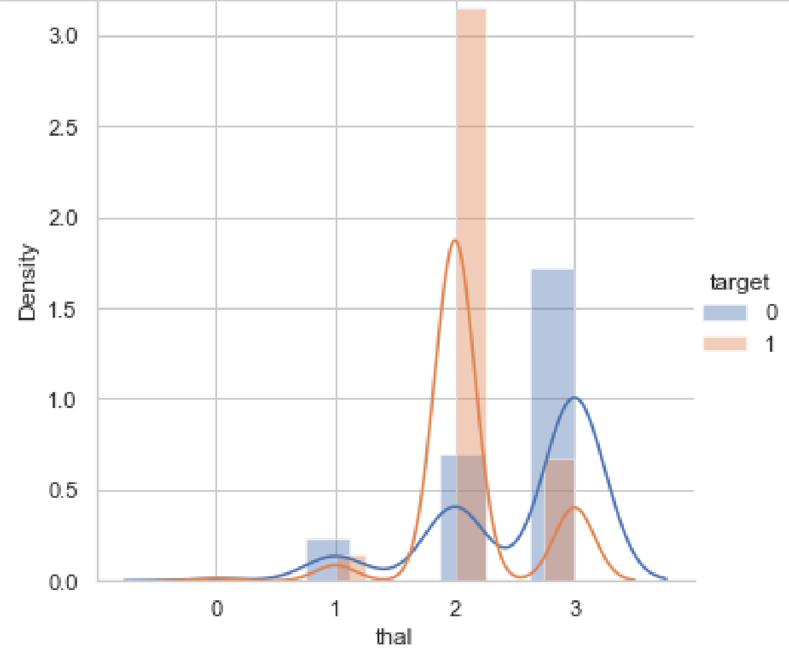


In [110]:

```
sns.boxplot(x=df['target'],y=df['age'])
plt.show()
```



In [115]:
```
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
for ojha, feature in enumerate(list(df.columns)[:-1]):
    fg = sns.FacetGrid(df, hue ='target', height=5)
    fg.map(sns.distplot, feature).add_legend()
plt.show()
```



In [122]:
```
le = LabelEncoder()
le.fit(df['target'].values)
y = le.transform(df['target'].values)
X = df.drop('target',axis=1).values
X_train, X_test , y_train , y_test = train_test_split(X,y,test_size=0.34,stratify=y,random_state=seed)
```

In [123]:
```
X_train
```

Out[123]:
```
array([[59.,  1.,  0., ...,  1.,  0.,  3.],
       [66.,  0.,  2., ...,  1.,  1.,  2.],
       [54.,  0.,  1., ...,  2.,  1.,  2.],
       ...,
       [39.,  1.,  0., ...,  1.,  0.,  3.],
       [59.,  1.,  2., ...,  2.,  0.,  2.],
       [59.,  0.,  0., ...,  1.,  0.,  2.]])
```

In [124]:
```
X_test
```

Out[124]:
```
array([[50.,  1.,  0., ...,  1.,  0.,  3.],
       [51.,  0.,  0., ...,  1.,  0.,  3.],
       [61.,  1.,  3., ...,  1.,  2.,  2.],
       ...,
       [43.,  0.,  2., ...,  1.,  0.,  2.],
       [52.,  1.,  3., ...,  1.,  0.,  1.],
       [29.,  1.,  1., ...,  2.,  0.,  2.]])
```

In [125]:
```
y_train
```

Out[125]:
```
array([1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1,
       0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1,
       1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1,
       1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0,
       0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1,
       1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1,
       0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1,
       0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0,
       1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1,
       0], dtype=int64)
```

In [126]:
```
y_test
```

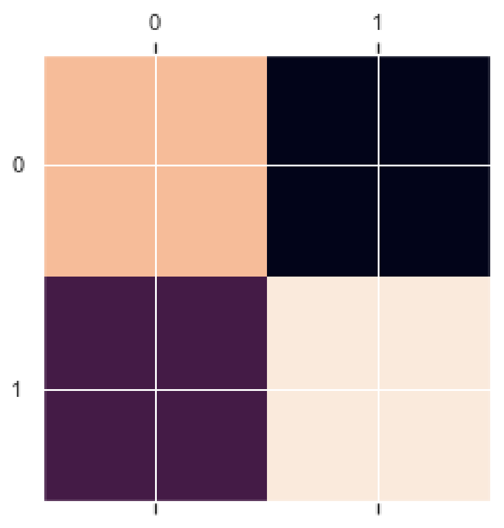
Out[126]:
```
array([0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1,
       1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1,
       0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0,
       0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0,
       1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1], dtype=int64)
```

In [161]:
```
tree = DecisionTreeClassifier(criterion='gini',
min_samples_leaf=5,
min_samples_split=5,
max_depth=None,
random_state=seed)
tree.fit(X_train, y_train)
y_pred=tree.predict(X_test)
accuracy=accuracy_score(y_test,y_pred)
print('DecisionTreeClassifier accuracy score: {}'.format(accuracy))
```

```
DecisionTreeClassifier accuracy score: 0.7692307692307693
```

In [162]:
```
from sklearn .metrics import confusion_matrix
import matplotlib.pyplot as  plt
print('Confusion Matrix is')
print(confusion_matrix(y_test,y_pred))
cm=confusion_matrix(y_test,y_pred)
plt.matshow(cm)
plt.show()
```

```
Confusion Matrix is
[[38  9]
 [15 42]]
```

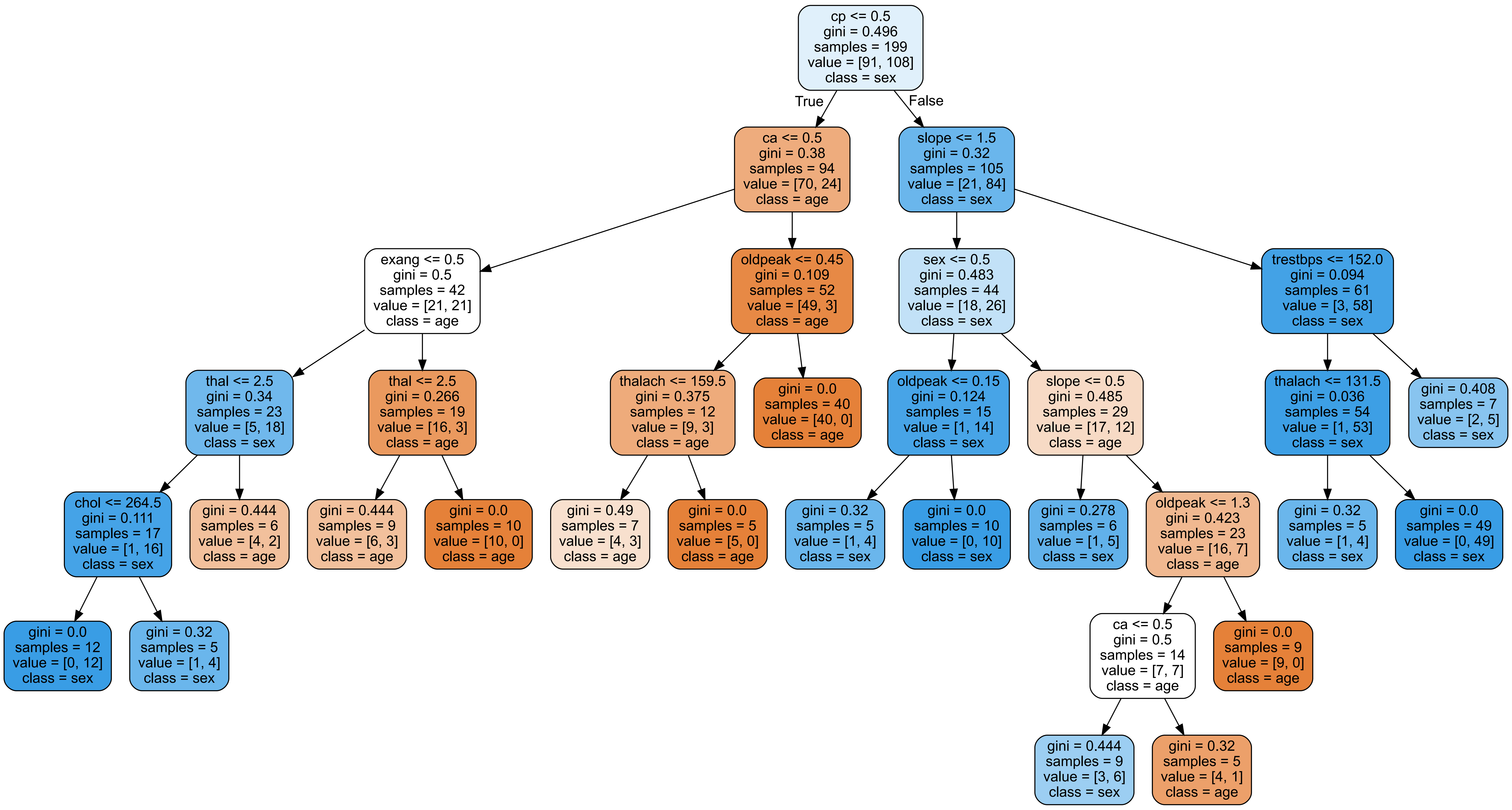


In [163]:
```
from  sklearn .metrics import  classification_report
print(classification_report(y_test,y_pred,labels=df['target'].unique()))
```

```
              precision    recall  f1-score   support

           1       0.82      0.74      0.78        57
           0       0.72      0.81      0.76        47

    accuracy                           0.77       104
   macro avg       0.77      0.77      0.77       104
weighted avg       0.78      0.77      0.77       104
```

In [164]:
```
import os
os.environ["PATH"] += os.pathsep +'C:/Program Files/Graphviz/bin'

def plot_tree(tree,dataframe,label_col ,label_encoder , plot_title):
    label_name = ['age','sex','restecg']
    graph_data = export_graphviz(tree,
feature_names=dataframe.drop(label_col,axis=1).columns,
class_names=label_name,filled=True, rounded=True, out_file=None)
    graph = graphviz.Source(graph_data)
    graph.render(plot_title)
    return graph
tree_graph = plot_tree(tree, df, 'target',le,'heart')
tree_graph
```

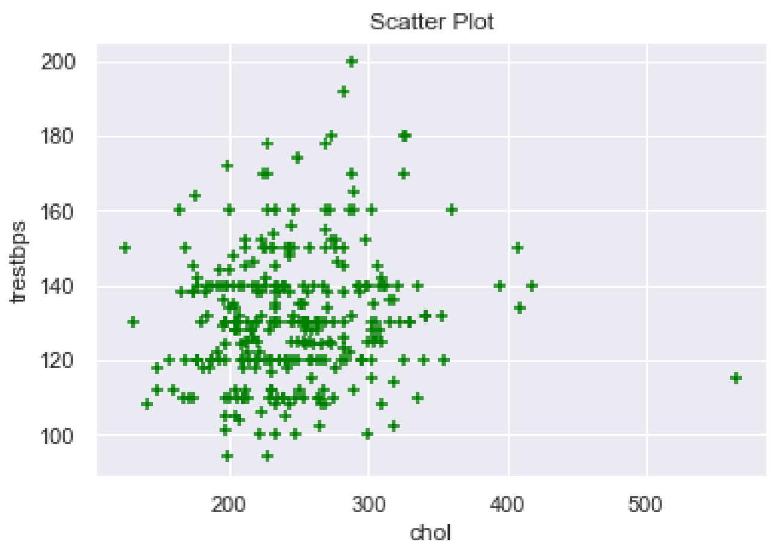
Out[164]:



In [178]:
```
X = df['chol']
Y = df['trestbps']
lin_df =pd.DataFrame({'chol': X, 'trestbps': Y})
print(lin_df)
```

```
     chol  trestbps
0     233       145
1     250       130
2     204       130
3     236       120
4     354       120
..    ...       ...
298   241       140
299   264       110
300   193       144
301   131       130
302   236       130

[303 rows x 2 columns]
```

In [179]:
```
plt.scatter(lin_df[['chol']], lin_df[['trestbps']], color ='green',marker = "+")
plt.xlabel('chol')
plt.ylabel('trestbps')
plt.title('Scatter Plot')
plt.show()
```

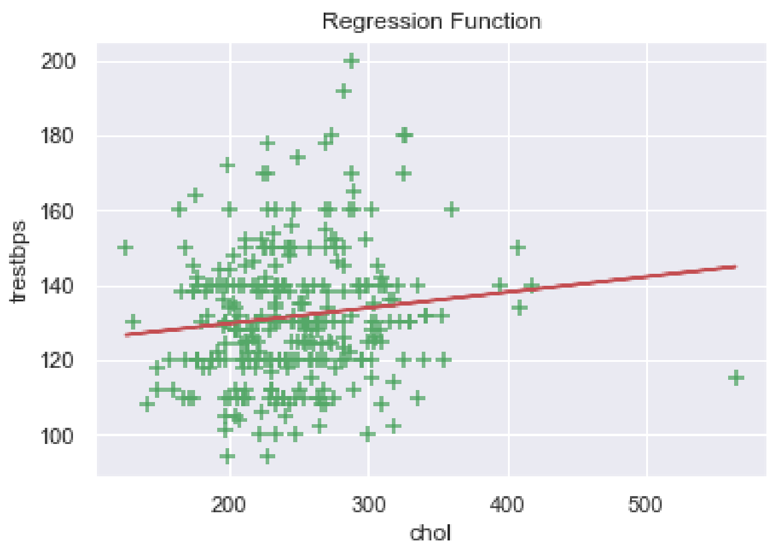


In [180]:
```
from sklearn.linear_model import LinearRegression
#define the classifier
classifier = LinearRegression()
#train the classifier
model = classifier.fit(lin_df[['chol']], lin_df[['trestbps']])
#use the trained classifier to make prediction
y_pred = classifier.predict(lin_df[['chol']])
print(y_pred)
#print coefficient (a in y=ax+b)and intercept (the constant, b in y=ax+print('Coefficients:
print('intercept: \n', classifier.intercept_)
```

```
 [129.86224576]
 [131.02925396]
 [135.32217696]
 [129.90392463]
 [129.8205669 ]
 [134.61363627]
 [130.73750191]
 [130.19567668]
 [128.40348552]
 [129.15370508]
 [129.57049372]
 [128.69523757]
 [131.40436374]
 [132.36297761]
 [129.40377826]
 [126.81968869]
 [131.19596941]]
intercept:
 [121.35975749]
```

In [182]:

```
#visualize regression function
plt.scatter(df[['chol' ]],df[['trestbps']], color = "g", marker = "+", s = 50)
plt.plot(df['chol'], y_pred, color ="r")
plt.xlabel('chol')
plt.ylabel('trestbps')
plt.title('Regression Function')
plt.show()
```



In [ ]: