

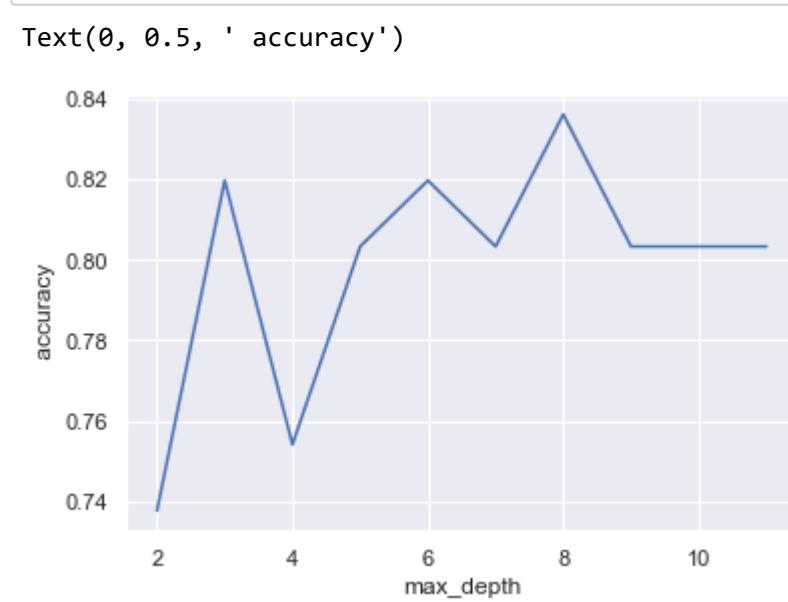


022/3/29

In [417]: ► y test

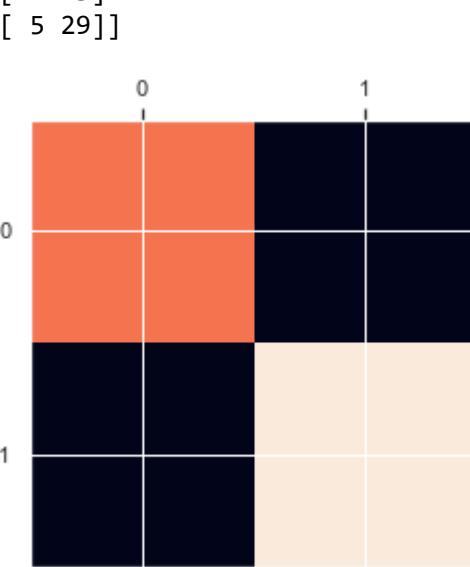
```
[418]: max_depth_range = list(range(2,12))
accuracy = []
for depth in max_depth_range:
    clf = DecisionTreeClassifier(criterion='entropy', splitter='best', min_samples_split=2, min_samples_leaf=1,
                                  max_depth=depth,
                                  random_state=seed)

    clf.fit(X_train, y_train)
    score = clf.score(X_test,y_test)
    accuracy.append(score)
plt.plot(max_depth_range,accuracy)
plt.xlabel('max_depth')
plt.ylabel ('accuracy')
```



```
[435]: tree = DecisionTreeClassifier(criterion='entropy',
                                     max_depth=8,
                                     random_state=0)
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*100))
```

```
[437]: ┆ 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()
```



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

	precision	recall	f1-score	support
1	0.85	0.85	0.85	34
0	0.81	0.81	0.81	27
cy			0.84	61
vg	0.83	0.83	0.83	61
vg	0.84	0.84	0.84	61

```
[422]: ➤ from sklearn.naive_bayes import GaussianNB  
from sklearn.metrics import f1_score  
classifier=GaussianNB()  
classifier.fit(X_train,y_train)  
y_pred2=classifier.predict(X_test)  
print(f1_score(y_test,y_pred2)*100)
```

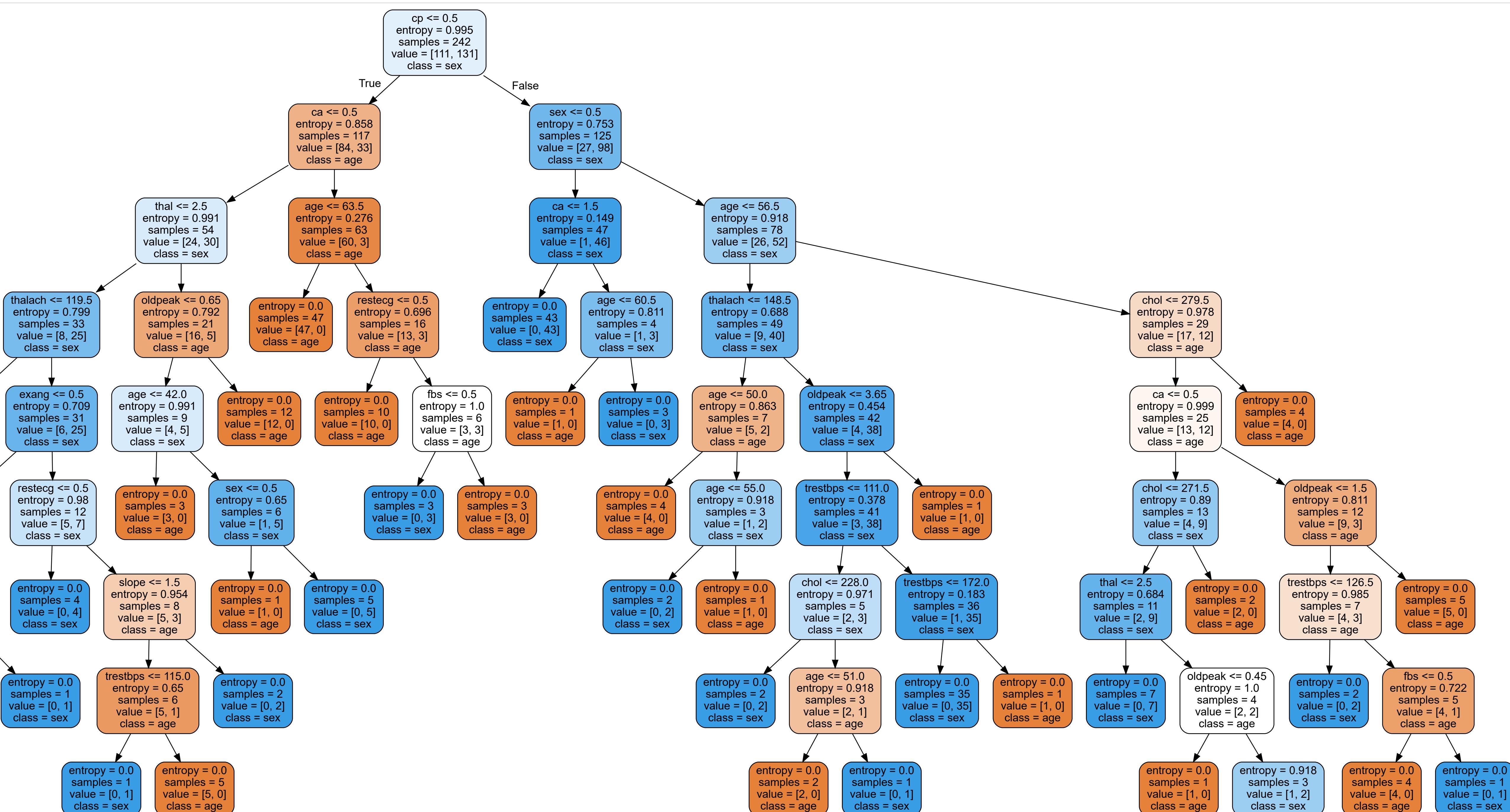
87.32394366197184

```
[423]: ➜ import os  
os.environ["PATH"] +=
```

```
def plot_tree(tree,dataframe,label_col ,label_encoder , plot_title):
    label_name = [ 'age','sex']
    graph_data = export_graphviz(tree,
                                feature_names=df.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')
```

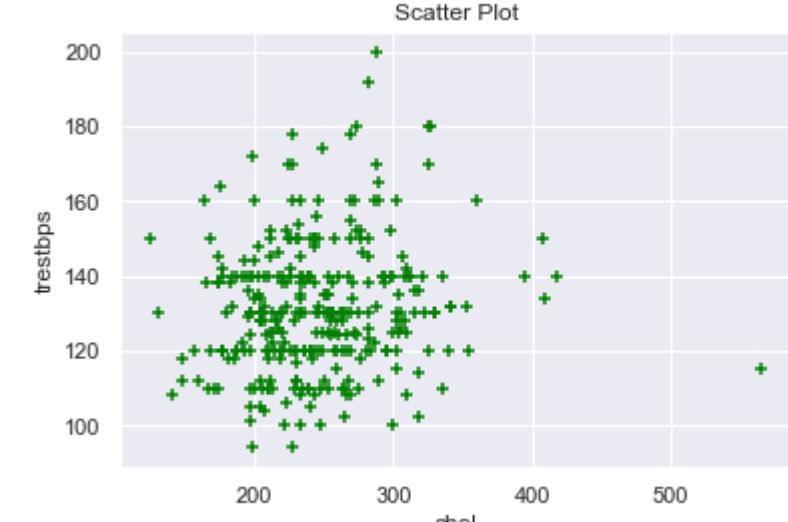


```
In [424]: X = df['chol']
Y = df['trestbps']
lin_df =pd.DataFrame({'chol': X, 'trestbps': Y})
```

```
print(linut)
```

	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	226	120

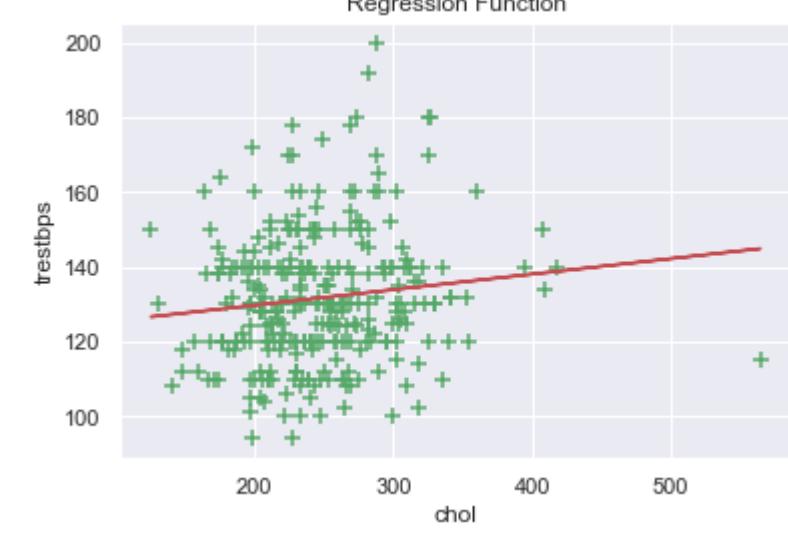
```
In [425]: # plt.scatter(lin_df[['chol']], lin_df[['trestbps']], color = 'green', marker = "+")
# plt.xlabel('chol')
# plt.ylabel('trestbps')
# plt.title('Scatter Plot')
# plt.show()
```



```
In [426]: 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 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+b)print('Coefficients:
print('intercept: \n', classifier.intercept_)
```

```
[133.94677444]
[130.8975799]
[127.23647733]
[131.86283124]
[129.37328917]
[130.86283132]
[130.89632115]
[128.94531976]
[133.98845331]
[132.40465647]
[134.21282609]
[129.12862131]
[129.8265669 ]
[130.15399781]
[128.86689862]
[130.86283132]
[130.12862169]
[130.52918759]
[130.9764809]
[133.11944421]
```

```
In [427]: #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 [ ]:

|