

```
In [432]: # 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.preprocessing import StandardScaler
%matplotlib inline
import seaborn as sns
sns.set(color_codes=True)
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
```

```
In [288]: # df = pd.read_csv('C:\\Anaconda\\Lib\\site-packages\\sklearn\\datasets\\data\\heart.csv')
df.head(None)
```

```
Out[288]:
```

	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	66	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

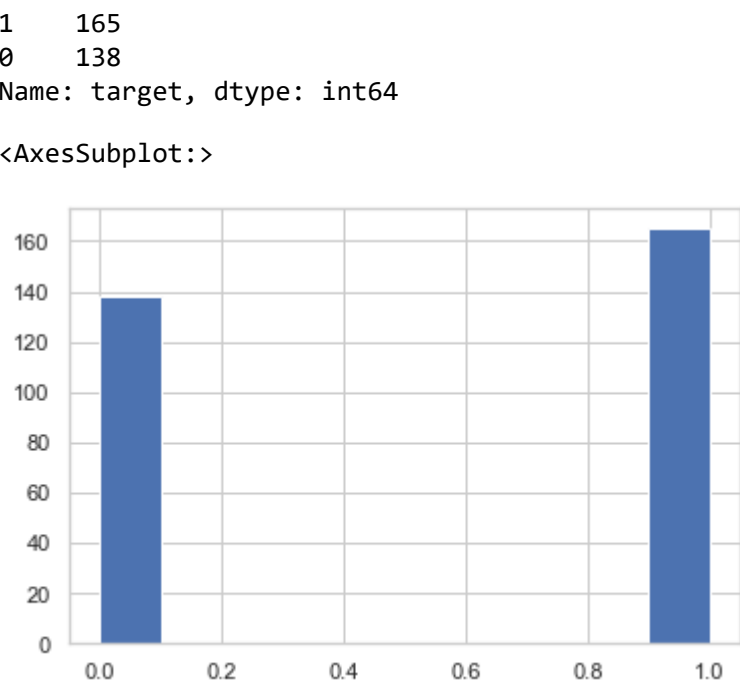
dtype: bool

```
In [90]: # df.describe()
```

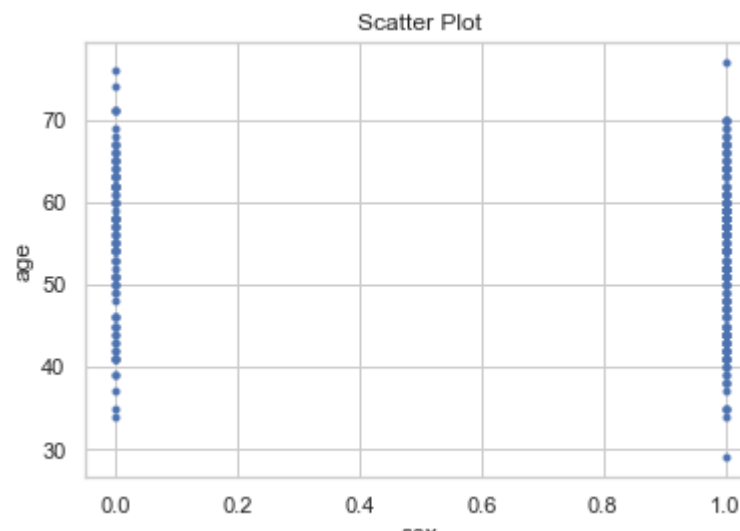
```
Out[90]:
```

	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	0.468011	1.032052	17.538143	51.830751	0.356198	0.525860	22.905161	0.489794	1.161075	0.616228	1.022806	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

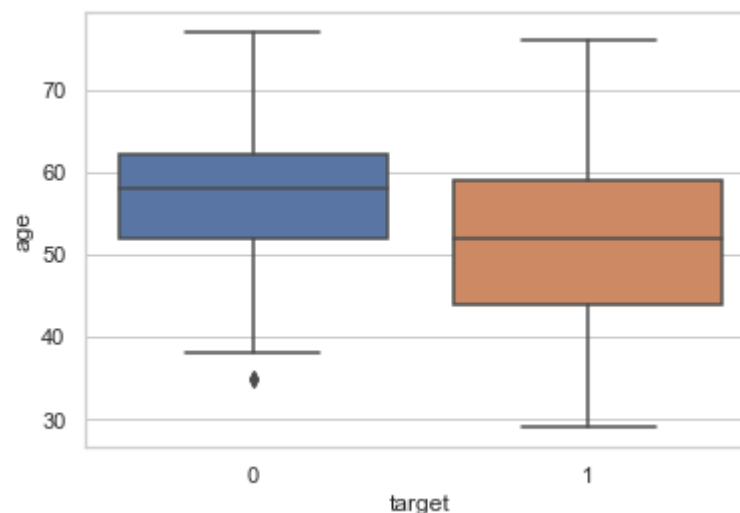
```
In [105]: # print(df['target'].value_counts())
df['target'].hist()
```



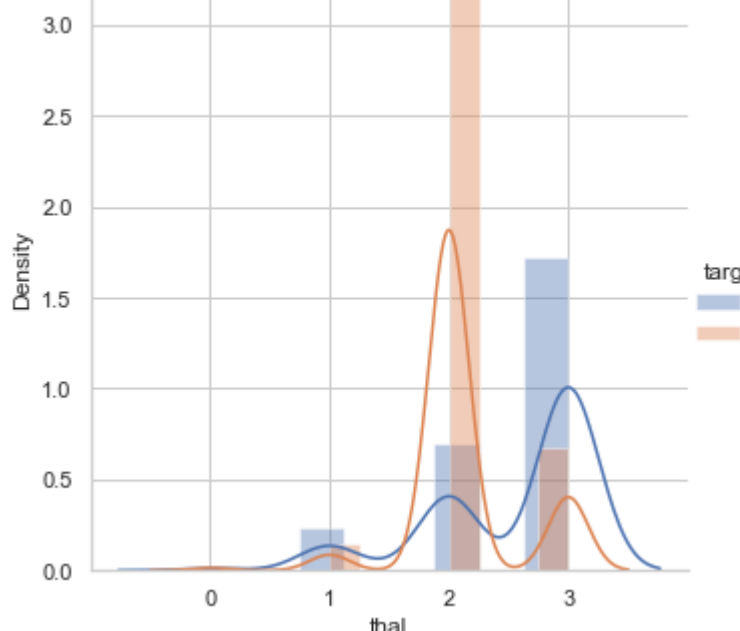
```
In [98]: # plt.scatter(slic_df[['sex']], slic_df[['age']], color = "b", marker = ".")
plt.xlabel('sex')
plt.ylabel('age')
plt.title('Scatter Plot')
plt.show()
```



```
In [110]: # sns.boxplot(x=df['target'], y=df['age'])
plt.show()
```



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



```
In [413]: # #split the data into train and test
from sklearn.model_selection import train_test_split
y = df['target'].values
X = df.drop('target', axis=1).values
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=0)
```

```
In [414]: # X_train
```

```
Out[414]: array([[43., 0., 2., ..., 1., 0., 2.],
 [66., 0., 2., ..., 1., 1., 2.],
 [58., 1., 2., ..., 2., 0., 2.],
 ...,
 [56., 1., 3., ..., 1., 0., 3.],
 [47., 1., 2., ..., 2., 0., 2.],
 [58., 1., 1., ..., 1., 0., 2.]])
```

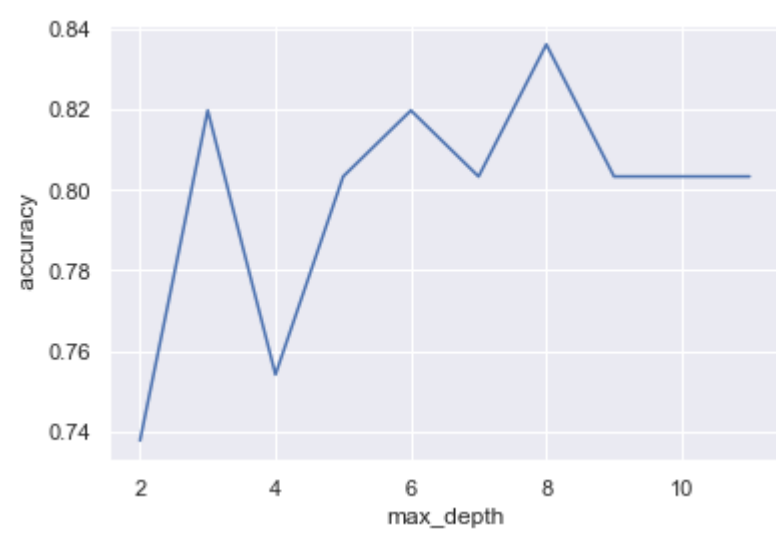
```
In [415]: # X_test
```

```
array([[43., 0., 2., ..., 1., 0., 2.],
 [66., 0., 2., ..., 1., 1., 2.],
 [58., 1., 2., ..., 2., 0., 2.],
 ...,
 [56., 1., 3., ..., 1., 0., 3.],
 [47., 1., 2., ..., 2., 0., 2.],
 [58., 1., 1., ..., 1., 0., 2.]])
```

[illegible]

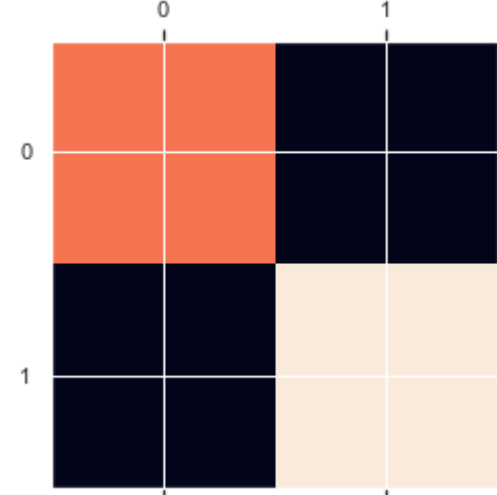
```
Out[417]: array([0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0,
                0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0,
                1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1], dtype=int64)
```

```
Out[418]: Text(0, 0.5, ' accuracy')
```



DecisionTreeClassifier accuracy score: 83.60655737704919

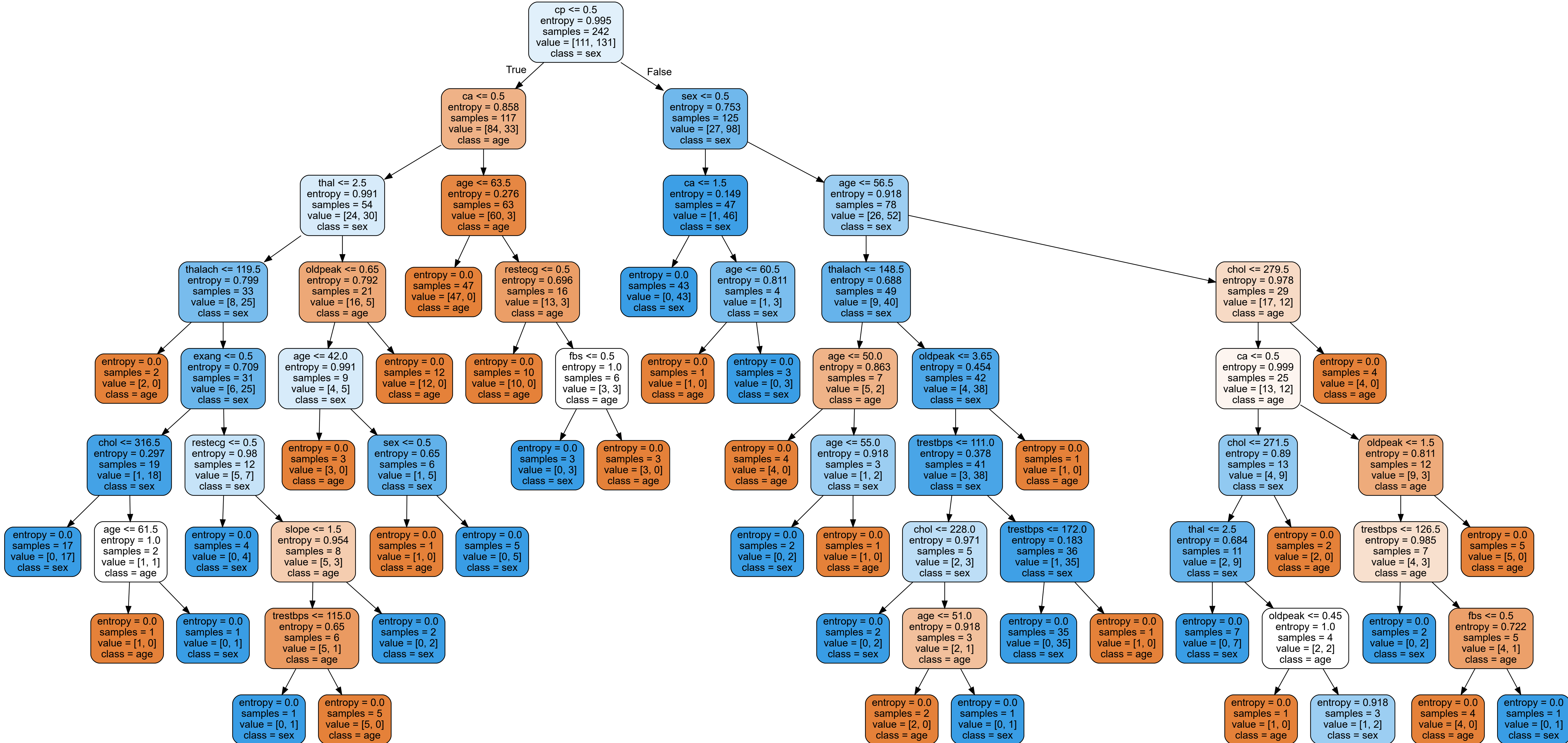
```
Confusion Matrix is
[[22  5]
 [ 5 29]]
```



	precision	recall	f1-score	support
1	0.85	0.85	0.85	34
0	0.81	0.81	0.81	27
accuracy			0.84	61
macro avg	0.83	0.83	0.83	61
weighted avg	0.84	0.84	0.84	61

87.32394366197184

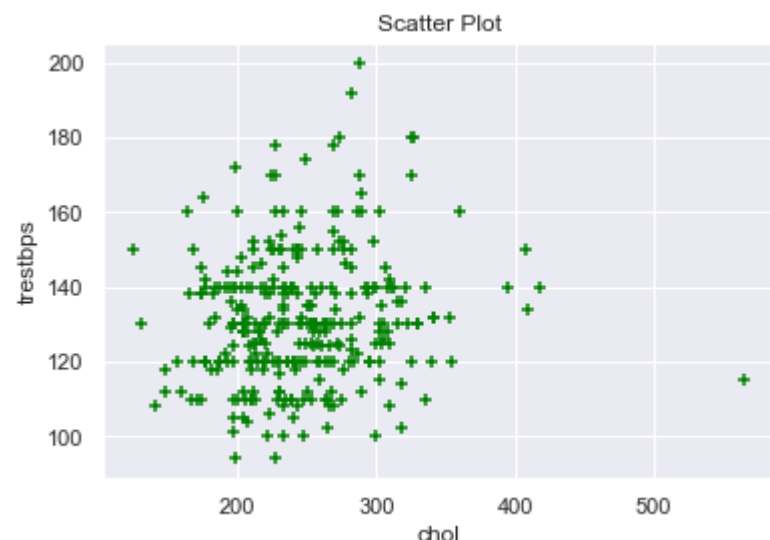
Out[423]:



	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]
```

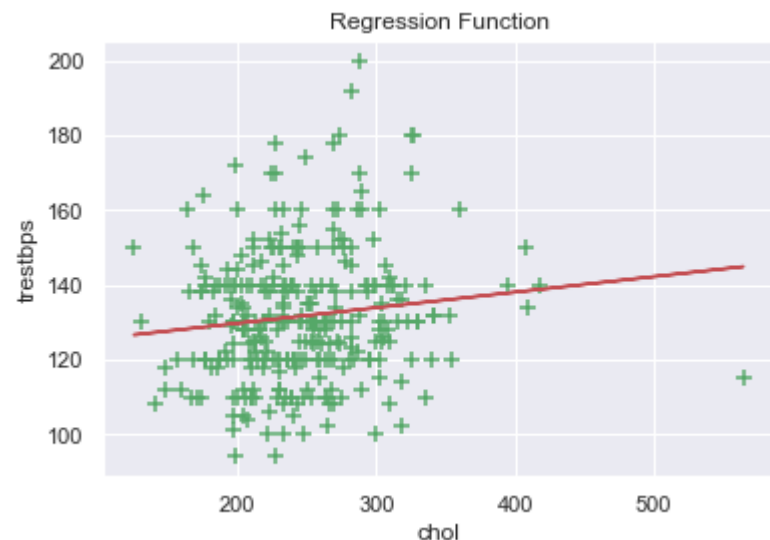
```
In [425]: plt.scatter(lin_df[['chol']], lin_df[['trestbps']], color = 'green',marker = "+")
plt.xlabel('chol')
plt.ylabel('trestbps')
plt.title('Scatter Plot')
plt.show()
```



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

```
[133.94677444]
[130.98757509]
[127.23647733]
[131.86283124]
[129.73728917]
[130.61246532]
[132.19626215]
[128.94531876]
[133.98845331]
[132.40465647]
[134.23852649]
[129.11205521]
[129.8205669 ]
[130.15399781]
[128.90698962]
[130.61246532]
[131.11261169]
[130.52918759]
[130.07064809]
[133.11940447]
```

```
In [427]: #visualize regression function
plt.scatter(lin_df[['chol']],lin_df[['trestbps']], color = "g", marker = "+", s = 50)
plt.plot(lin_df[['chol']], y_pred, color = "r")
plt.xlabel('chol')
plt.ylabel('trestbps')
plt.title('Regression Function')
plt.show()
```



In []:

#