In [32]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib import rcParams
from matplotlib.cm import rainbow
%matplotlib inline
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn import datasets
from sklearn import linear_model
from sklearn.metrics import mean_squared_error
from sklearn.tree import DecisionTreeClassifier
```

In [33]:

```
data= pd.read_csv(r'C:\Users\ASUS\Downloads\archive (2)\heart.csv')
print(data)
```

```
      age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  
\
0      52    1   0       125   212    0        1      168      0      1.0   
1      53    1   0       140   203    1        0      155      1      3.1   
2      70    1   0       145   174    0        1      125      1      2.6   
3      61    1   0       148   203    0        1      161      0      0.0   
4      62    0   0       138   294    1        1      106      0      1.9   
...   ...  ...  ..       ...   ...  ...      ...      ...    ...      ...   
1020   59    1   1       140   221    0        1      164      1      0.0   
1021   60    1   0       125   258    0        0      141      1      2.8   
1022   47    1   0       110   275    0        0      118      1      1.0   
1023   50    0   0       110   254    0        0      159      0      0.0   
1024   54    1   0       120   188    0        1      113      0      1.4   

      slope  ca  thal  target  
0         2   2     3       0  
1         0   0     3       0  
2         0   0     3       0  
3         2   1     3       0  
4         1   3     2       0  
...     ...  ..   ...     ...  
1020      2   0     2       1  
1021      1   1     3       0  
1022      1   1     2       0  
1023      2   0     2       1  
1024      1   1     3       0  

[1025 rows x 14 columns]
```

In [34]:

```
data.head()
```

Out[34]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	tar
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3	
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	

In [35]:

```
data.tail()
```

Out[35]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
1020	59	1	1	140	221	0	1	164	1	0.0	2	0	2
1021	60	1	0	125	258	0	0	141	1	2.8	1	1	3
1022	47	1	0	110	275	0	0	118	1	1.0	1	1	2
1023	50	0	0	110	254	0	0	159	0	0.0	2	0	2
1024	54	1	0	120	188	0	1	113	0	1.4	1	1	3

In [36]:

```
data.shape
```

Out[36]:

```
(1025, 14)
```

In [37]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1025 entries, 0 to 1024
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       1025 non-null   int64  
 1   sex       1025 non-null   int64  
 2   cp        1025 non-null   int64  
 3   trestbps  1025 non-null   int64  
 4   chol      1025 non-null   int64  
 5   fbs       1025 non-null   int64  
 6   restecg   1025 non-null   int64  
 7   thalach   1025 non-null   int64  
 8   exang     1025 non-null   int64  
 9   oldpeak   1025 non-null   float64
 10  slope     1025 non-null   int64  
 11  ca        1025 non-null   int64  
 12  thal      1025 non-null   int64  
 13  target    1025 non-null   int64  
dtypes: float64(1), int64(13)
memory usage: 112.2 KB
```

In [38]:

```
data.isnull().sum() #to check for null values
```

Out[38]:

```
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
target      0
dtype: int64
```

In [39]:

```
data.describe()
```

Out[39]:

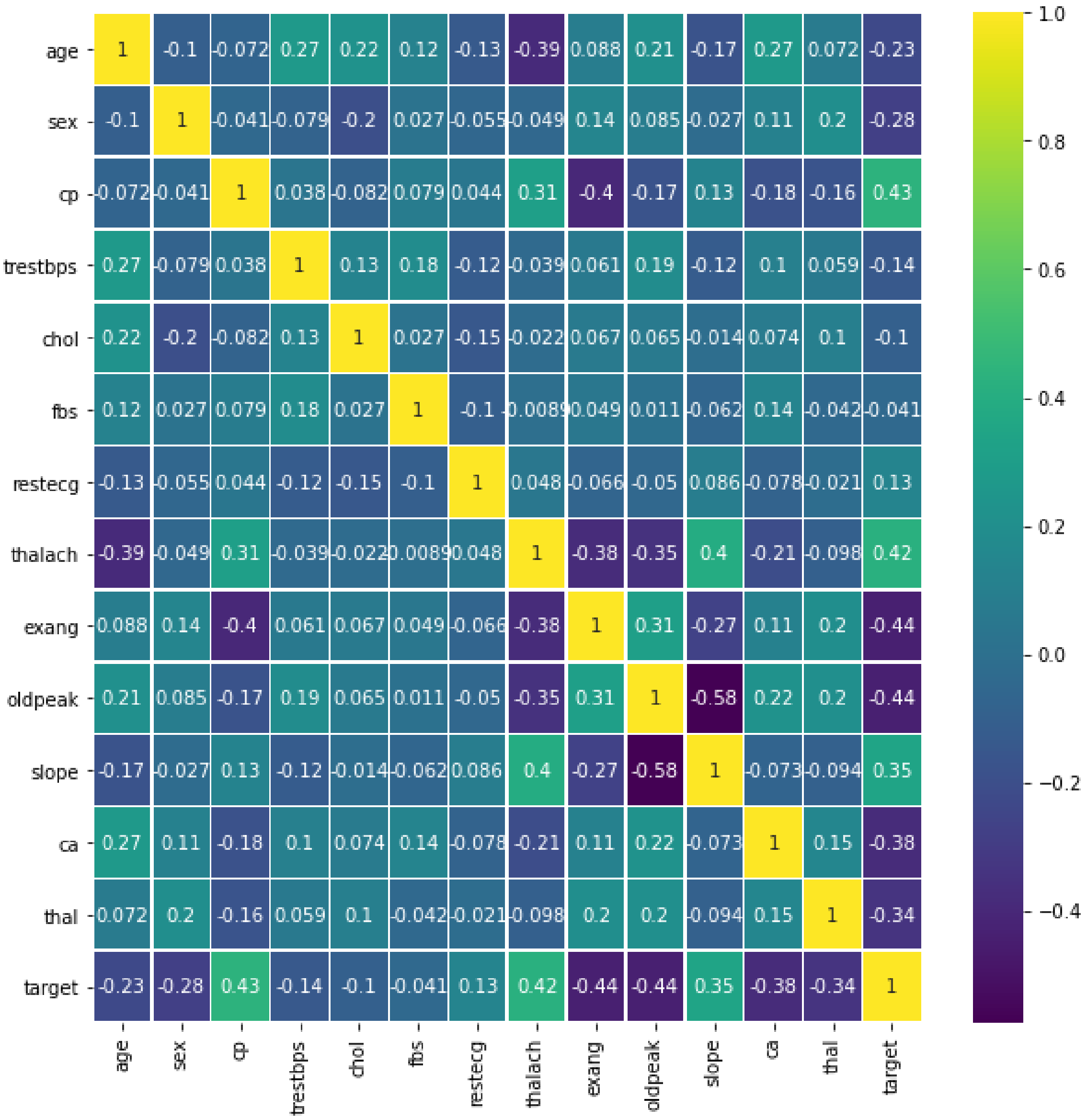
	age	sex	cp	trestbps	chol	fbs	re
count	1025.000000	1025.000000	1025.000000	1025.000000	1025.00000	1025.000000	1025.0
mean	54.434146	0.695610	0.942439	131.611707	246.00000	0.149268	0.5
std	9.072290	0.460373	1.029641	17.516718	51.59251	0.356527	0.5
min	29.000000	0.000000	0.000000	94.000000	126.00000	0.000000	0.0
25%	48.000000	0.000000	0.000000	120.000000	211.00000	0.000000	0.0
50%	56.000000	1.000000	1.000000	130.000000	240.00000	0.000000	1.0
75%	61.000000	1.000000	2.000000	140.000000	275.00000	0.000000	1.0
max	77.000000	1.000000	3.000000	200.000000	564.00000	1.000000	2.0

In [40]:

```
plt.figure(figsize = (10,10))
sns.heatmap(data.corr(),annot = True,cmap='viridis',linewidths=.5)
```

Out[40]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2caeb9d3760>
```

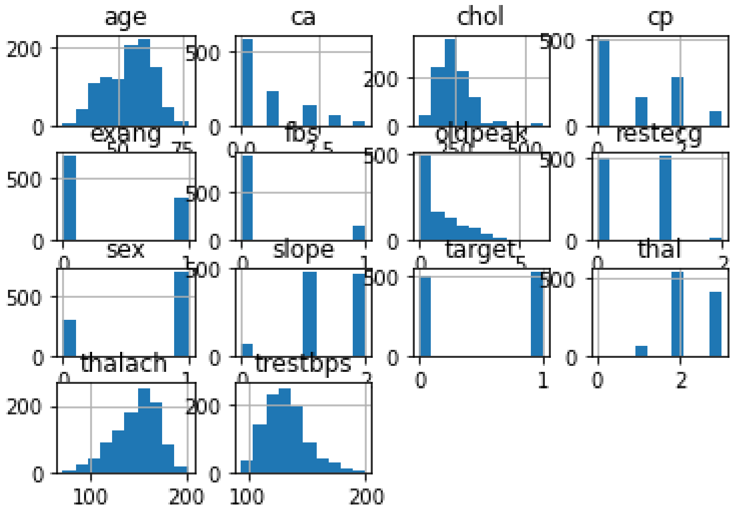


In [41]:

```
data.hist()
```

Out[41]:

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x000002CAEB11FEE
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002CAEB819A3
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002CAEB93A7C
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002CAECA0776
0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x000002CAEB9B46A
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002CAEB9EF07
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002CAEB9EF25
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002CAEB8AC3A
0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x000002CAECFB99D
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002CAED078E2
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002CAED0B22B
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002CAED0DC70
0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x000002CAED109B8
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002CAED137FD
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002CAED1703A
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002CAED1909A
0>]],
      dtype=object)
```



In [42]:

```
data['target'].value_counts() #how many ppl have heart disease
```

Out[42]:

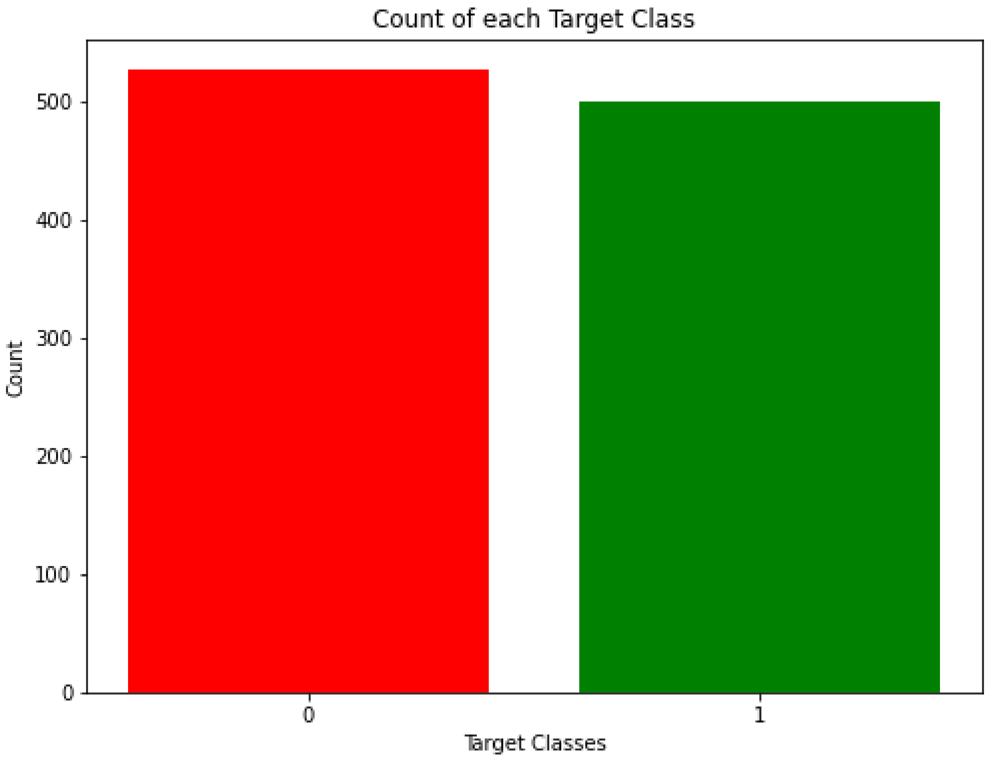
```
1    526
0    499
Name: target, dtype: int64
```

In [43]:

```
rcParams['figure.figsize'] = 8,6
plt.bar(data['target'].unique(),data['target'].value_counts(),color=['red','green'])
plt.xticks([0,1])
plt.xlabel('Target Classes')
plt.ylabel('Count')
plt.title('Count of each Target Class')
```

Out[43]:

```
Text(0.5, 1.0, 'Count of each Target Class')
```



In [44]:

```
x = data.drop(columns = 'target',axis = 1) #dropping row axis = 0
y = data['target']
```

In [45]:

```
print(x)
```

```
      age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak
\
0      52    1   0       125   212    0        1      168      0      1.0
1      53    1   0       140   203    1        0      155      1      3.1
2      70    1   0       145   174    0        1      125      1      2.6
3      61    1   0       148   203    0        1      161      0      0.0
4      62    0   0       138   294    1        1      106      0      1.9
...   ...  ...  ..       ...   ...  ...      ...      ...    ...      ...
1020   59    1   1       140   221    0        1      164      1      0.0
1021   60    1   0       125   258    0        0      141      1      2.8
1022   47    1   0       110   275    0        0      118      1      1.0
1023   50    0   0       110   254    0        0      159      0      0.0
1024   54    1   0       120   188    0        1      113      0      1.4

      slope  ca  thal
0         2   2     3
1         0   0     3
2         0   0     3
3         2   1     3
4         1   3     2
...     ...  ..   ...
1020      2   0     2
1021      1   1     3
1022      1   1     2
1023      2   0     2
1024      1   1     3

[1025 rows x 13 columns]
```

In [46]:

```
print(y)
```

```
0       0
1       0
2       0
3       0
4       0
       ..
1020    1
1021    0
1022    0
1023    1
1024    0
Name: target, Length: 1025, dtype: int64
```

In [47]:

```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,stratify=y,random_st
ate = 2)
```

In [49]:

```
print(x.shape,x_train.shape,x_test.shape)
```

```
(1025, 13) (820, 13) (205, 13)
```

In [72]:

```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x_train,y_train)
```

Out[72]:

```
LinearRegression()
```

In [73]:

```
y_predicted = model.predict(x_test)
```

In [74]:

```
print("MSE is:=",mean_squared_error(y_test,y_predicted))
```

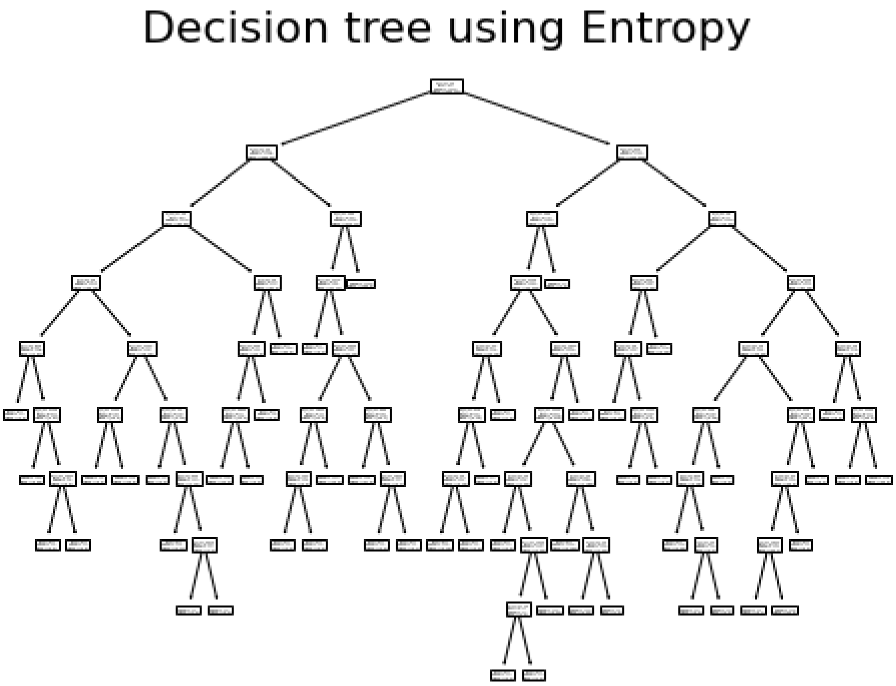
```
MSE is:= 0.13612944595506918
```

In [75]:

```
model1 = DecisionTreeClassifier()
model1.fit(x_train,y_train)
prediction = model1.predict(x_test)
import sklearn as sklearn
sklearn.tree.plot_tree(model1,max_depth=None,feature_names = None,class_names=None,label='all',filled=False,impurity=True,node_ids=False,proportion=False,rounded=False,precision=3,ax=None,fontsize=None)
plt.title('Decision tree using Entropy',fontsize=22)
```

Out[75]:

```
Text(0.5, 1.0, 'Decision tree using Entropy')
```



In [76]:

```
score = accuracy_score(y_test,prediction)
score
```

Out[76]:

```
1.0
```

In [77]:

```
model2 = LogisticRegression()
```

In [78]:

```
model2.fit(x_train,y_train) #find relationship btw feature and target
```

```
C:\anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:762: Conve
rgenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown i
n:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-reg
ression
  n_iter_i = _check_optimize_result(
```

Out[78]:

```
LogisticRegression()
```

In [70]:

```
#Accuracy on training data
x_train_prediction = model2.predict(x_train)
training_data_accuracy = accuracy_score(x_train_prediction,y_train)
print('accuracy on training data:',training_data_accuracy)
```

```
accuracy on training data: 0.848780487804878
```

In [59]:

```
#Accuracy on testing data
x_test_prediction = model2.predict(x_test)
testing_data_accuracy = accuracy_score(x_test_prediction,y_test)
print('accuracy on training data:',testing_data_accuracy)
```

```
accuracy on training data: 0.8048780487804879
```

In [60]:

```python
#predictive system

input_data = (71,0,0,112,149,0,1,125,0,1.6,1,0,2)
#change the ip data to numpy array
input_data_as_numpy_array = np.asarray(input_data)

#reshape the numpy array as we are predicting for only one instance
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

prediction = model2.predict(input_data_reshaped)
print(prediction)

if(prediction[0]==0):
    print("The person does not has heart disease.")
else:
    print("The person has heart disease.")
```

```
[1]
The person has heart disease.
```