PREDICTING HEART DISEASE

PROJECT-6

Group-4

2000080044

Jhansi.B

OVERVIEW

Importing Data

Identifying and Dealing with Missing Data

Format the data for Decision Trees

Build a Preliminary Classification Tree

Use Cost Complexity Pruning to improve the Tree

Build, Evaluate, Draw and Interpret a final Tree

IMPORTING LIBRARIES

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from scipy import stats
import pydotplus
import graphviz
```

What is Pydot plus package?

- Pydot plus allows **one to easily create both directed and non-directed graphs from Python**. ... Output can be in-lined .

What is a graph viz package?

- It is an opensource python module used for drawing graphs specified in DOT language which can be completed using different nodes and edges.

IMPORTING DATA

```
data=pd.read_csv("heart.csv")
data
```

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope	HeartDisease
0	40	M	ATA	140	289	0	Normal	172	N	0.0	Up	0
1	49	F	NAP	160	180	0	Normal	156	N	1.0	Flat	1
2	37	M	ATA	130	283	0	ST	98	N	0.0	Up	0
3	48	F	ASY	138	214	0	Normal	108	Y	1.5	Flat	1
4	54	M	NAP	150	195	0	Normal	122	N	0.0	Up	0
...	...	...	...	...	...	...	...	...	...	...	...	...
913	45	M	TA	110	264	0	Normal	132	N	1.2	Flat	1
914	68	M	ASY	144	193	1	Normal	141	N	3.4	Flat	1
915	57	M	ASY	130	131	0	Normal	115	Y	1.2	Flat	1
916	57	F	ATA	130	236	0	LVH	174	N	0.0	Flat	1
917	38	M	NAP	138	175	0	Normal	173	N	0.0	Up	0

918 rows × 12 columns

```
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
```

```
df.isna().sum()
```

```
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
```

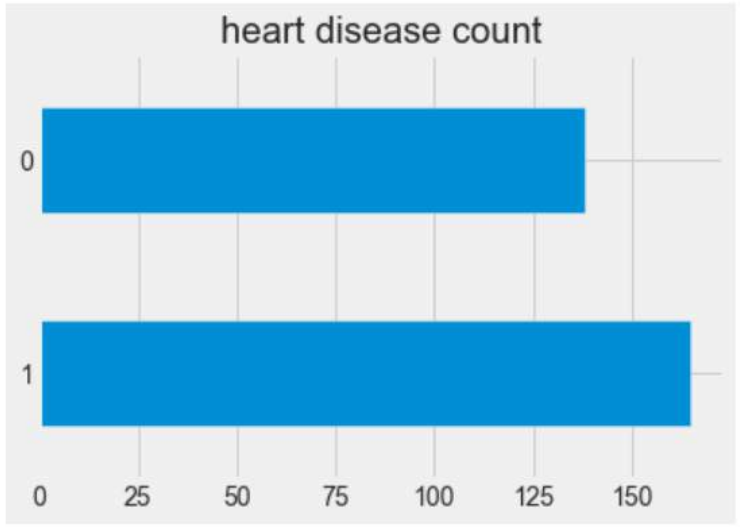
IDENTIFYING AND DEALING WITH MISSING DATA

Format the data for Decision Trees

- Heart disease dataset consists of all int and float values .
- Since there are no categorical values in the given dataset we did not perform any encoding method.

```
df.target.value_counts().plot(kind="barh",title="heart disease count")
```

```
0]: <AxesSubplot:title={'center':'heart disease count'}>
```



HEART DISEASE COUNT---PLOTTING

WE have 165 persons with heart disease and 138 persons without heart disease.

So, our problem is balanced

- Here, least repeated values are taken into categorical
- And more repeated values are taken into continuous.

```
In [11]:
categorical_val = []
continous_val = []
for column in df.columns:
    if len(df[column].unique()) <= 10:
        categorical_val.append(column)
    else:
        continous_val.append(column)
```

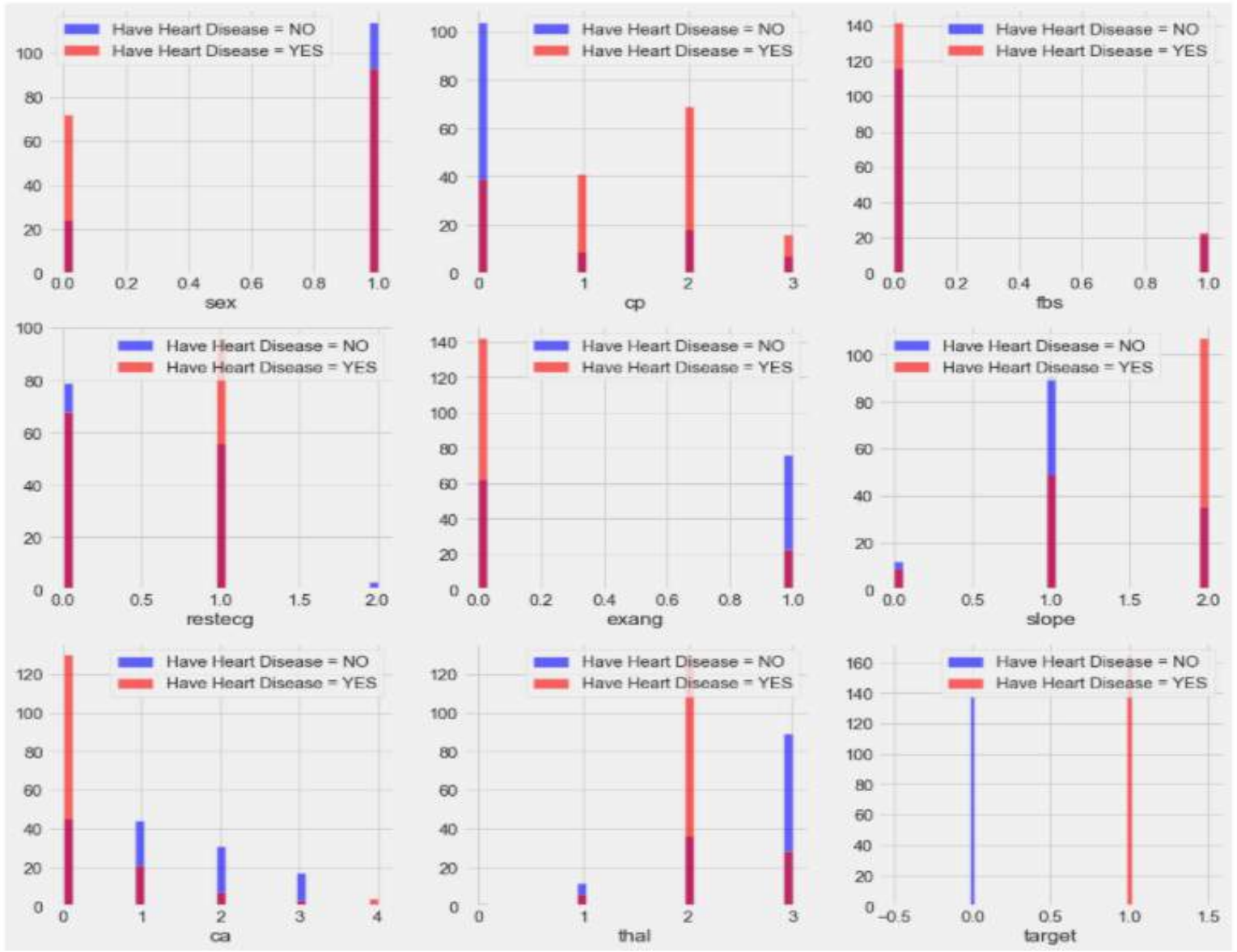
```
In [12]:
#not having more unique values
categorical_val
```

```
Out[12]: ['sex', 'cp', 'fbs', 'restecg', 'exang', 'slope', 'ca', 'thal', 'target']
```

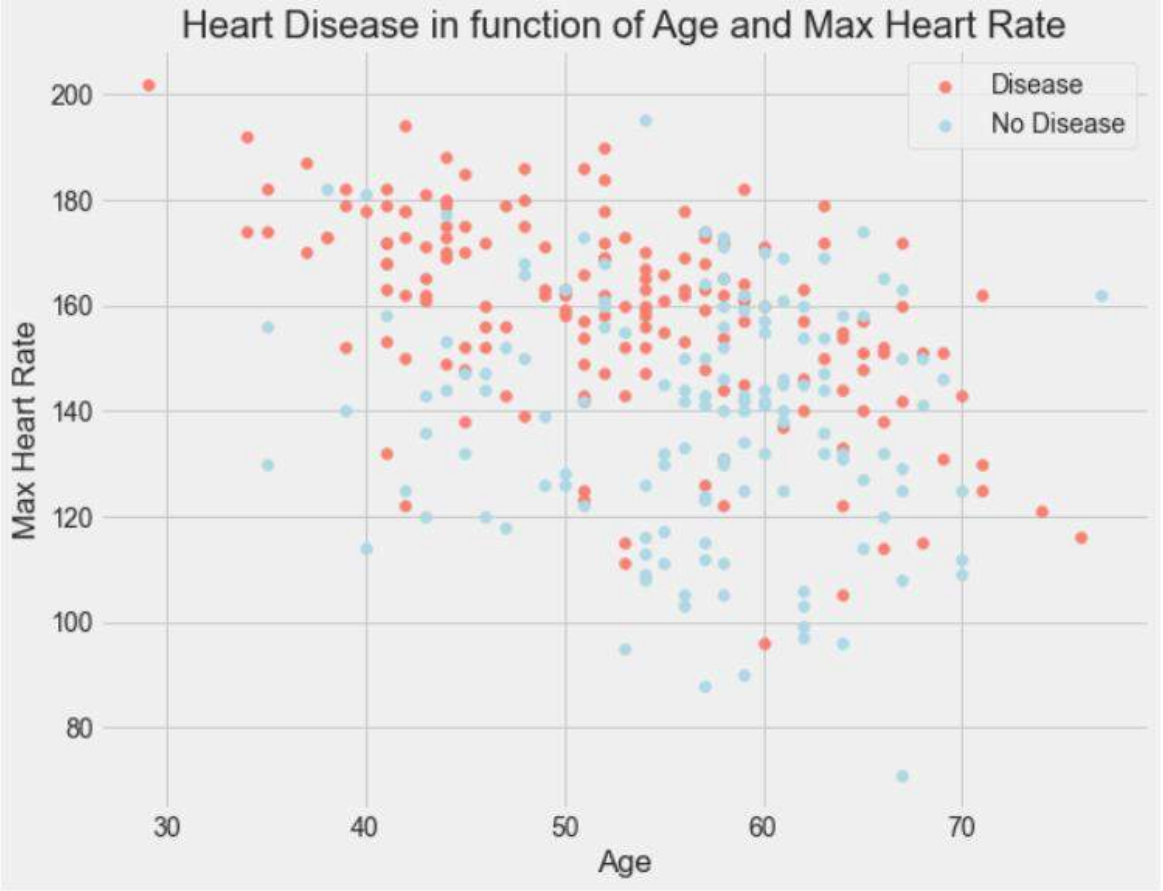
```
In [14]:
#having more unique values
continous_val
```

```
Out[14]: ['age', 'trestbps', 'chol', 'thalach', 'oldpeak']
```

Graphs

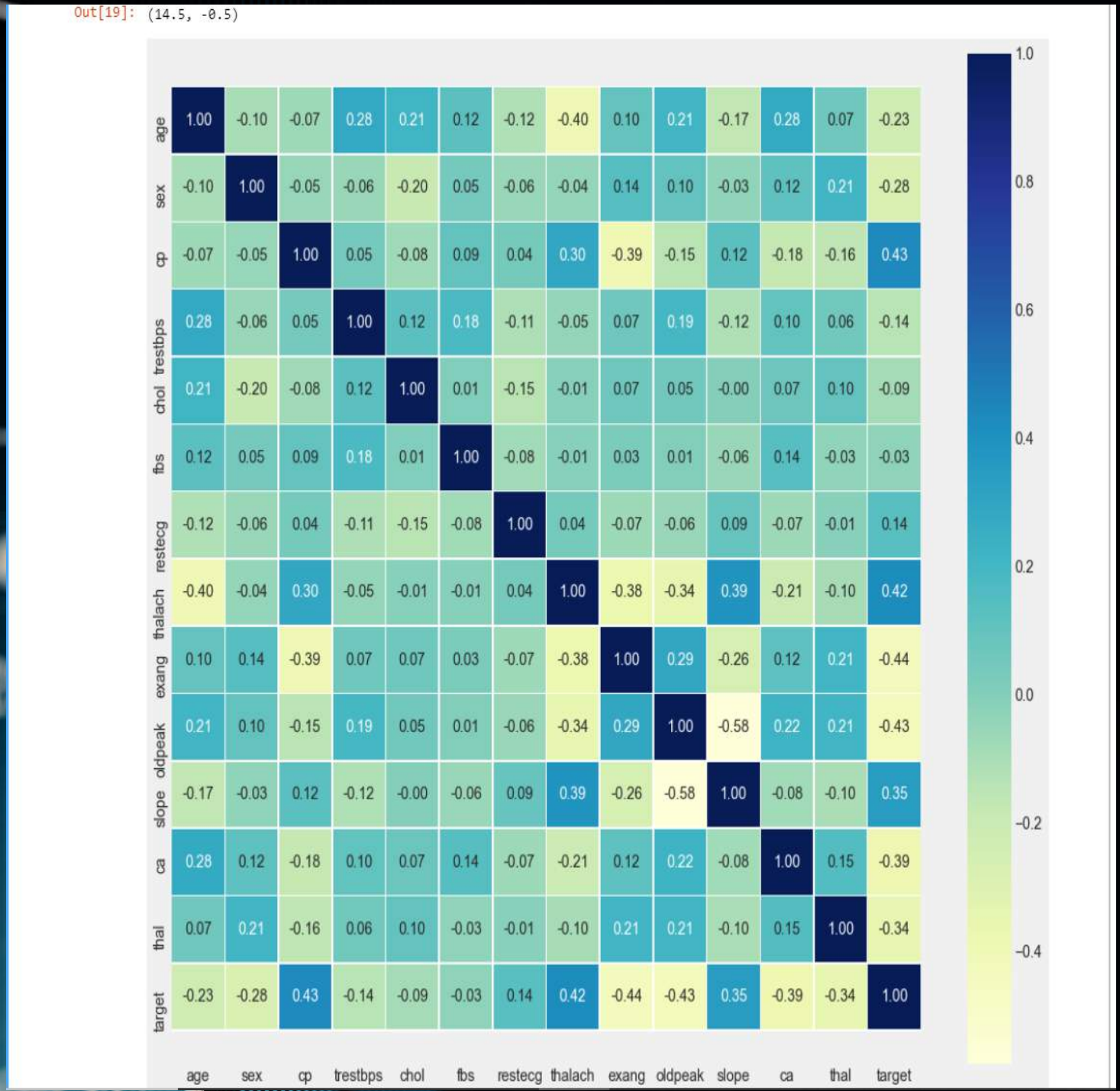

Have Heart Disease = NO
Have Heart Disease = YES
sex
cp
fbs
restecg
exang
slope
ca
thal
target

age vs maximum heart rate for heart disease



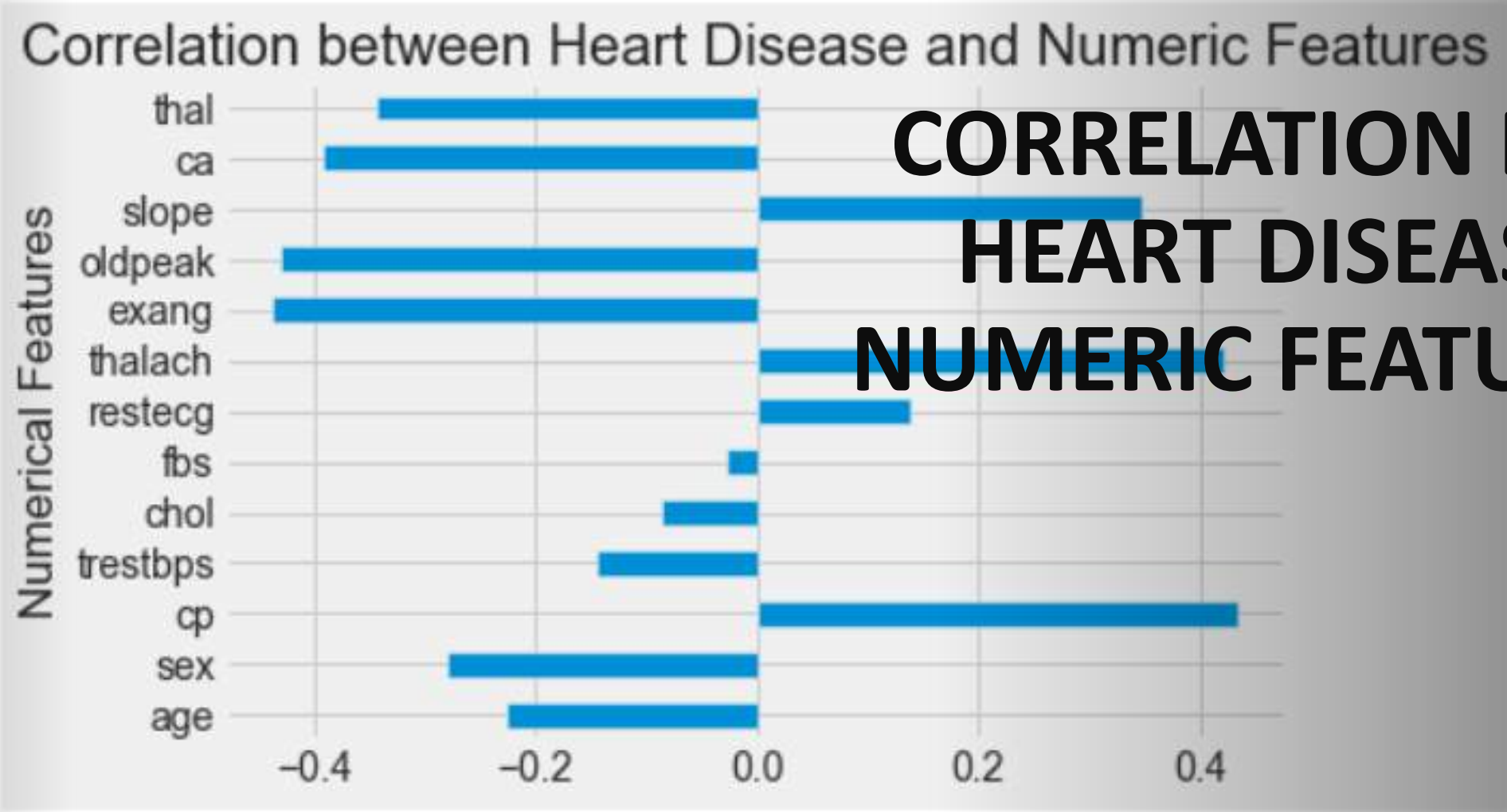
CORRELATION MATRIX

For better visualization we constructed a correlation matrix.



In [20]:
```
df.drop('target', axis=1).corrwith(df.target).plot(kind="barh",
    title="Correlation between Heart Disease and Numeric Features",
    ylabel='Correlation', xlabel='Numerical Features',
)
```

Out[20]:
```
<AxesSubplot:title={'center':'Correlation between Heart Disease and Numeric Fea
```



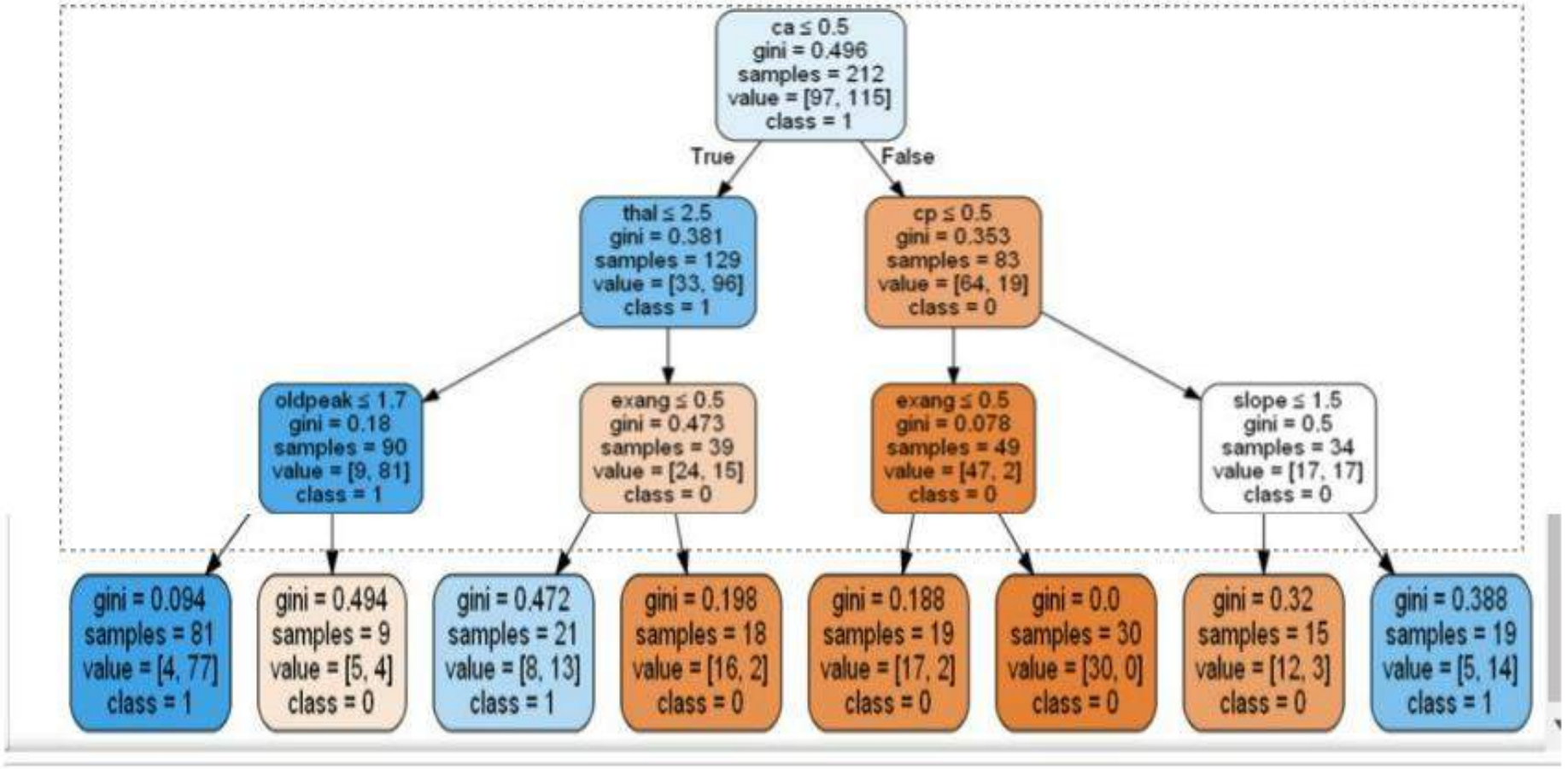
CORRELATION B/W HEART DISEASE & NUMERIC FEATURES

```
preds_pruned = clf_pruned.predict(X_test)
preds_pruned_train = clf_pruned.predict(X_train)
print(accuracy_score(y_train,preds_pruned_train))
print(accuracy_score(y_test,preds_pruned))
```

```
0.8679245283018868
0.7692307692307693
```

ACCURACY

Build a Preliminary Classification Tree



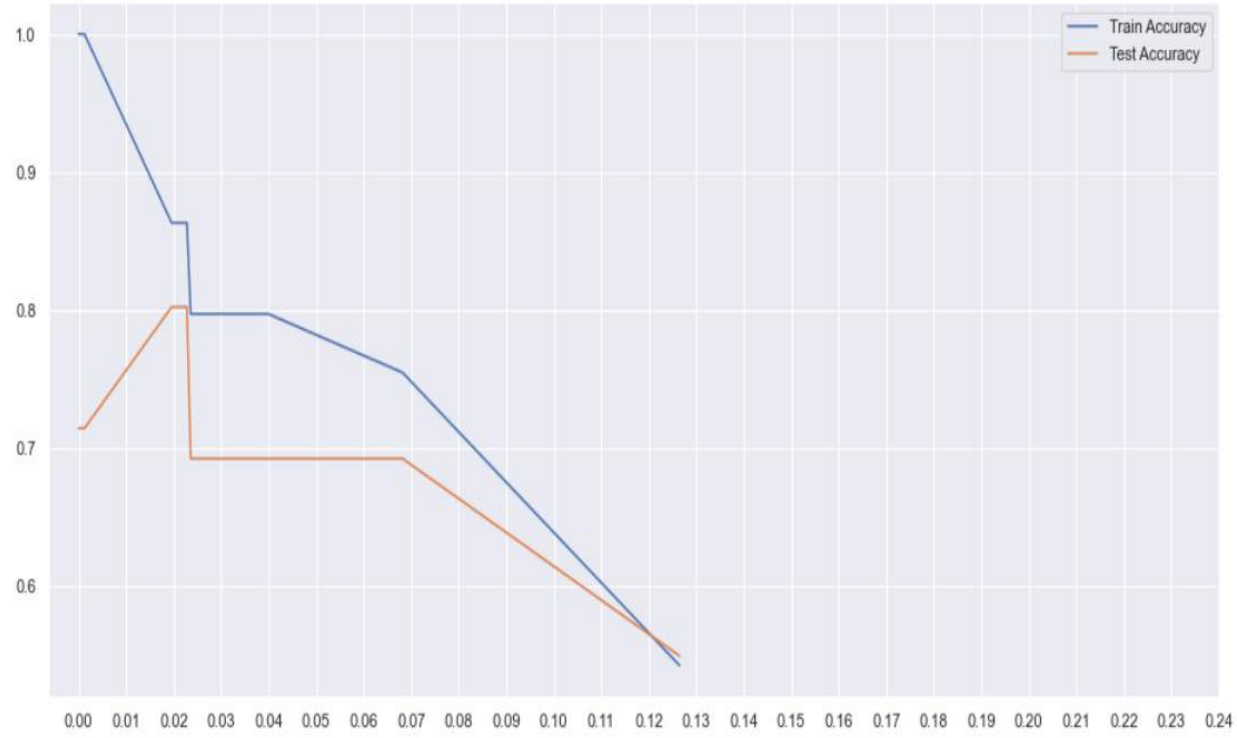
Use Cost Complexity Pruning to improve the Tree

```
In [35]: path=clf_pruned.cost_complexity_pruning_path(X_train,y_train)
         alphas=path['ccp_alphas']
         alphas

Out[35]: array([0.        , 0.00121598, 0.01957838, 0.02279047, 0.02359067,
                0.03992675, 0.06818274, 0.12650172])
```

cost complexity pruning provides another option to control size of a tree.

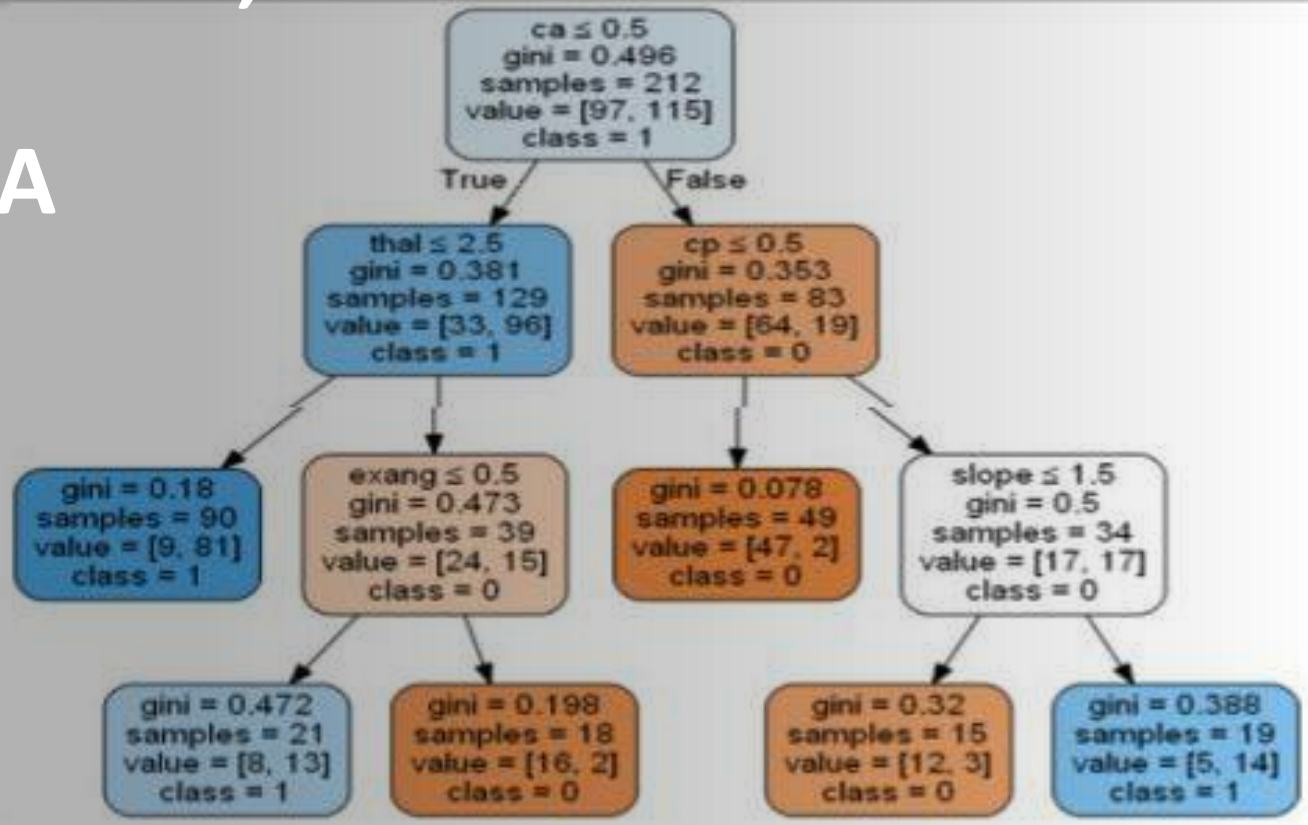
- Next , we train the decision tree using effective alphas.



BUILD, EVALUATE, DRAW AND INTERPRET A FINAL TREE

```
In [38]:
from sklearn.tree import export_graphviz
import six
import sys
sys.modules['sklearn.externals.six'] = six
from sklearn.externals.six import StringIO
from IPython.display import Image
import pydotplus
import graphviz
xvar = df.drop('target', axis=1)
feature_cols = xvar.columns
dot_data = StringIO()
export_graphviz(tree_1, out_file=dot_data,
                filled=True, rounded=True,
                special_characters=True,feature_names = feature_cols,class_names=['0','1','2'])
from pydot import graph_from_dot_data
(graph,) = graph_from_dot_data(dot_data.getvalue())
Image(graph.create_png())
```

Out[38]:



THANK YOU