In [1]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn import tree
from sklearn.naive_bayes import GaussianNB
from sklearn import metrics
```

In [2]:
```
pip install seaborn --upgrade
```

```
Requirement already satisfied: seaborn in d:\app\anaconda\lib\site-packages (0.13.2)
Requirement already satisfied: numpy!=1.24.0,>=1.20 in d:\app\anaconda\lib\site-pack
ages (from seaborn) (1.24.3)
Requirement already satisfied: pandas>=1.2 in d:\app\anaconda\lib\site-packages (fro
m seaborn) (2.1.4)
Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in d:\app\anaconda\lib\site-p
ackages (from seaborn) (3.8.0)
Requirement already satisfied: contourpy>=1.0.1 in d:\app\anaconda\lib\site-packages
(from matplotlib!=3.6.1,>=3.4->seaborn) (1.2.0)
Requirement already satisfied: cycler>=0.10 in d:\app\anaconda\lib\site-packages (fr
om matplotlib!=3.6.1,>=3.4->seaborn) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in d:\app\anaconda\lib\site-package
s (from matplotlib!=3.6.1,>=3.4->seaborn) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in d:\app\anaconda\lib\site-package
s (from matplotlib!=3.6.1,>=3.4->seaborn) (1.4.4)
Requirement already satisfied: packaging>=20.0 in d:\app\anaconda\lib\site-packages
(from matplotlib!=3.6.1,>=3.4->seaborn) (23.1)
Requirement already satisfied: pillow>=6.2.0 in d:\app\anaconda\lib\site-packages (f
rom matplotlib!=3.6.1,>=3.4->seaborn) (10.2.0)
Requirement already satisfied: pyparsing>=2.3.1 in d:\app\anaconda\lib\site-packages
(from matplotlib!=3.6.1,>=3.4->seaborn) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in d:\app\anaconda\lib\site-pack
ages (from matplotlib!=3.6.1,>=3.4->seaborn) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in d:\app\anaconda\lib\site-packages (fr
om pandas>=1.2->seaborn) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.1 in d:\app\anaconda\lib\site-packages
(from pandas>=1.2->seaborn) (2023.3)
Requirement already satisfied: six>=1.5 in d:\app\anaconda\lib\site-packages (from p
ython-dateutil>=2.7->matplotlib!=3.6.1,>=3.4->seaborn) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

In [3]:
```
df = pd.read_csv('heart.csv')
```

In [4]:
```
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 [5]:
```
total = df.isnull().sum().sort_values(ascending=False)
percent_1=df.isnull().sum()/df.isnull().count()*100
percent_2 = (round(percent_1,1)).sort_values(ascending = False)
missing_data = pd.concat([total,percent_2],axis=1,keys=['Total','%'])
missing_data.head(5)
```

Out[5]:

| | Total | % |
|---|---|---|
| **age** | 0 | 0.0 |
| **sex** | 0 | 0.0 |
| **cp** | 0 | 0.0 |
| **trestbps** | 0 | 0.0 |
| **chol** | 0 | 0.0 |

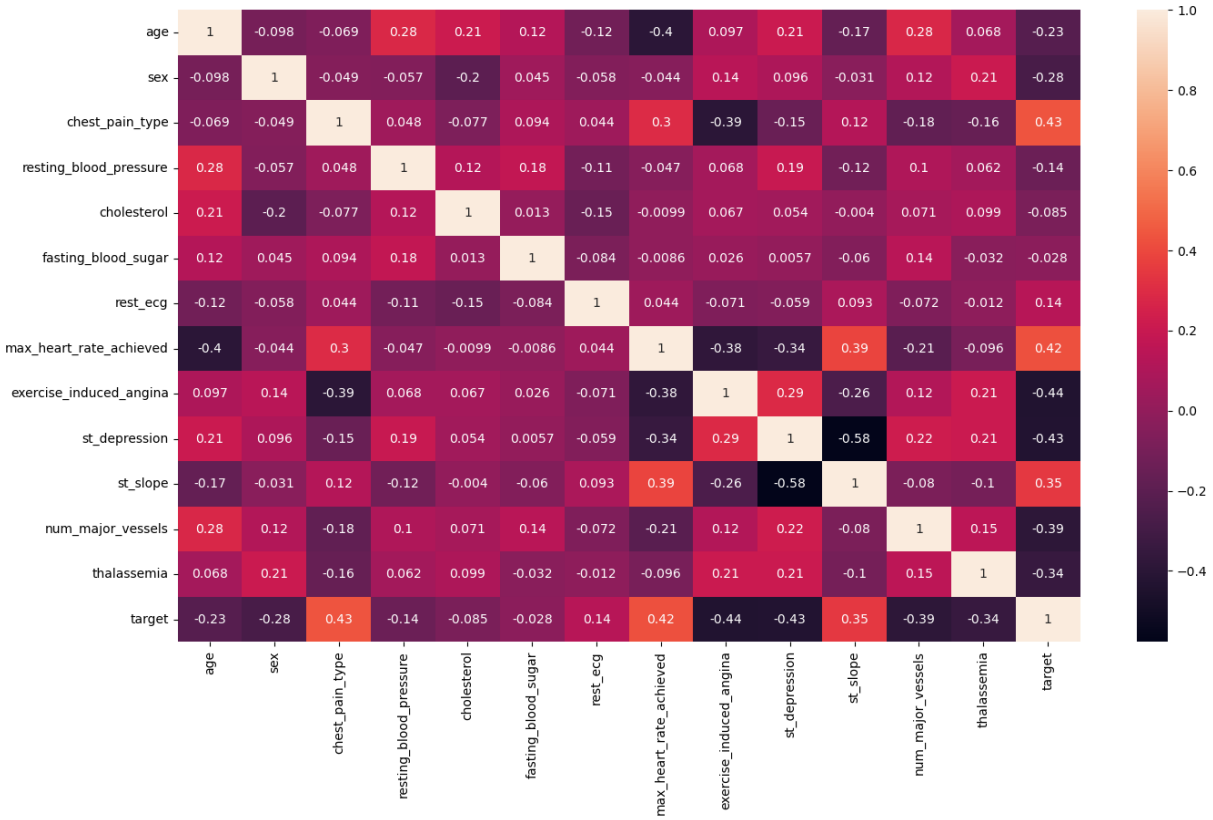
Không có dữ liệu trống. Không cần xử lý xoá rỗng

Đổi tên:

In [6]:
```
df.columns = ['age','sex','chest_pain_type','resting_blood_pressure','cholesterol',
```

In [7]:
```
plt.figure(figsize=(16,9))
sns.heatmap(df.corr(method='pearson'),annot=True)
```

Out[7]: <Axes: >




In [8]:
```
feature = df.drop('target',axis=1)
label = df['target']
```

In [9]:
```
feature.select_dtypes(exclude=['int64']).columns
```

Out[9]:
```
Index(['st_depression'], dtype='object')
```

In [10]:
```
x_train,x_test,y_train,y_test = train_test_split(feature,label,test_size=0.3,random
```

In [11]:
```
clf = tree.DecisionTreeClassifier(criterion="entropy",random_state=0)
clf.fit(x_train,y_train)
```

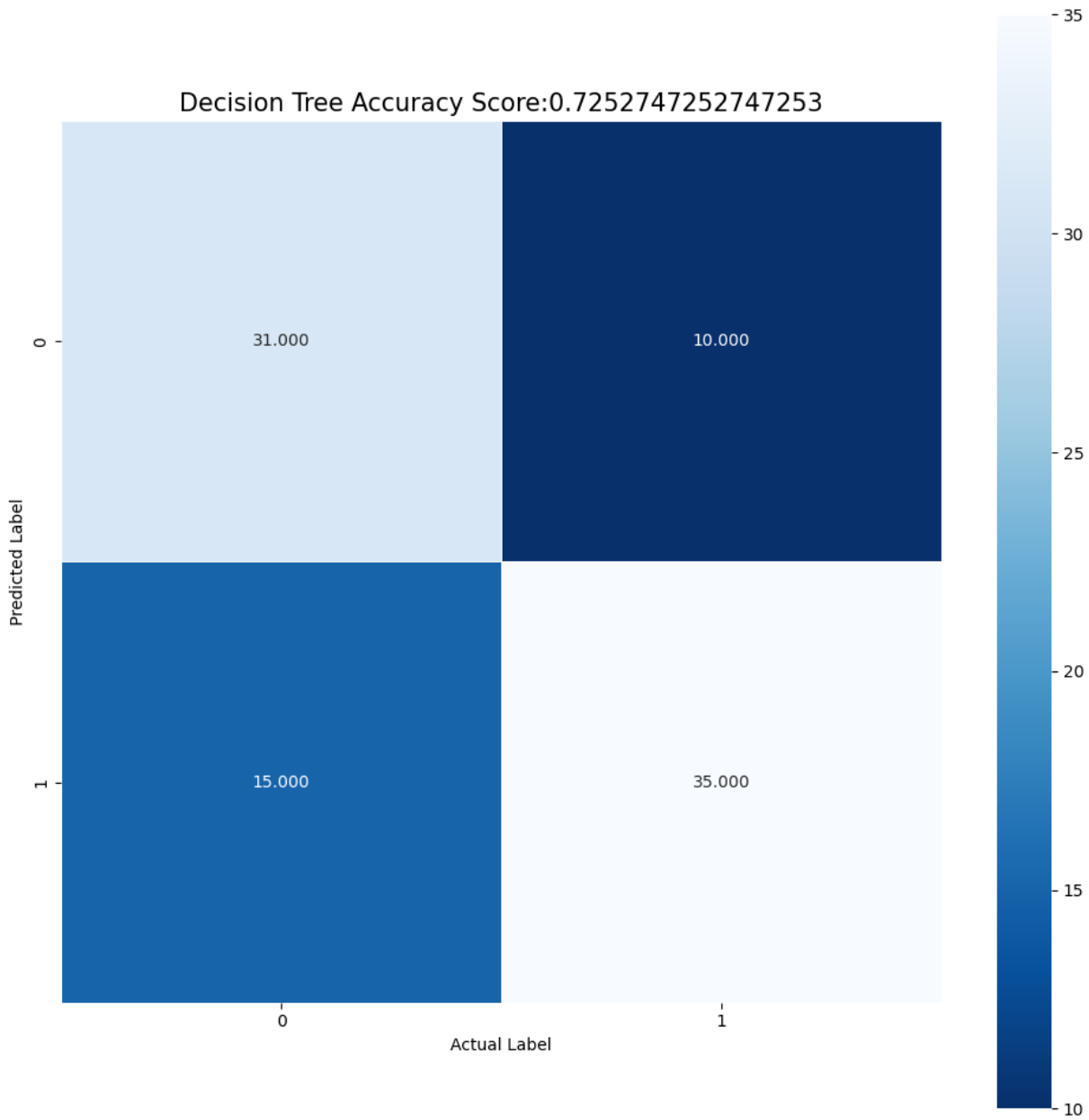
Out[11]:
```
▾                    DecisionTreeClassifier

DecisionTreeClassifier(criterion='entropy', random_state=0)
```

In [12]:
```
tree_pred = clf.predict(x_test)

tree_score = metrics.accuracy_score(y_test,tree_pred)
print("Accuracy:",tree_score)
print("Report:",metrics.classification_report(y_test,tree_pred))
```

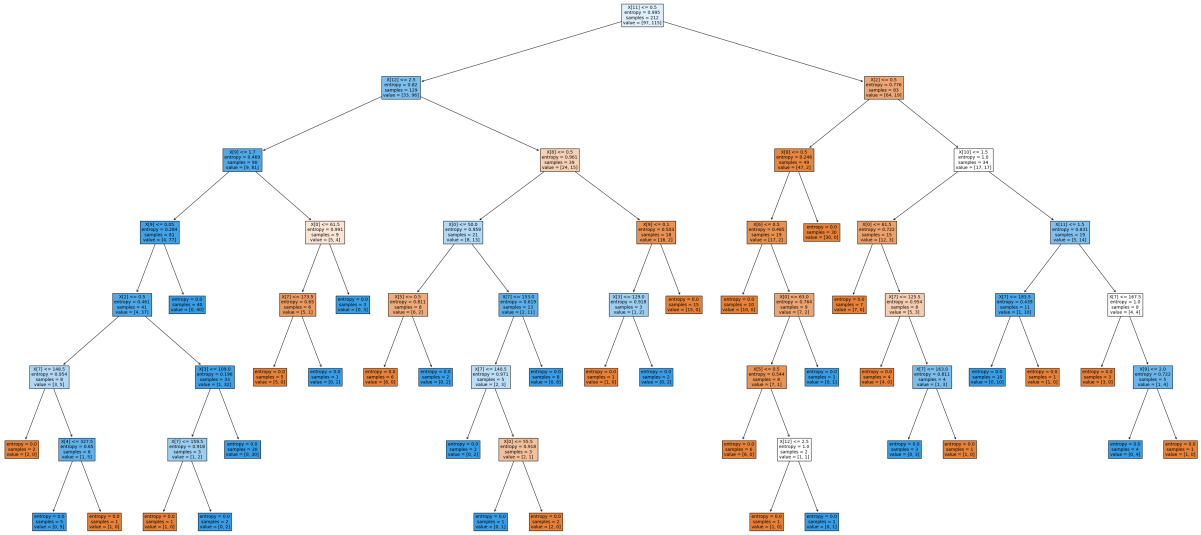
```
Accuracy: 0.7252747252747253
Report:               precision    recall  f1-score   support

           0       0.67      0.76      0.71        41
           1       0.78      0.70      0.74        50

    accuracy                           0.73        91
   macro avg       0.73      0.73      0.72        91
weighted avg       0.73      0.73      0.73        91
```

In [13]:
```
tree_cm = metrics.confusion_matrix(y_test,tree_pred)
```

In [14]:
```
plt.figure(figsize=(12,12))
sns.heatmap(tree_cm,annot=True, fmt=".3f",linewidth=.5,square=True,cmap='Blues_r');
plt.xlabel('Actual Label');
plt.ylabel('Predicted Label');
title ='Decision Tree Accuracy Score:{0}'.format(tree_score)
plt.title(title,size=15);
```




In [15]:
```
fig, ax = plt.subplots(figsize=(50,24))
tree.plot_tree(clf,filled=True,fontsize=10)
plt.savefig('decision_tree',dpi=100)
plt.show()
```



In [16]:
```
clf = tree.DecisionTreeClassifier(criterion="gini",random_state=0)
clf.fit(x_train,y_train)
```

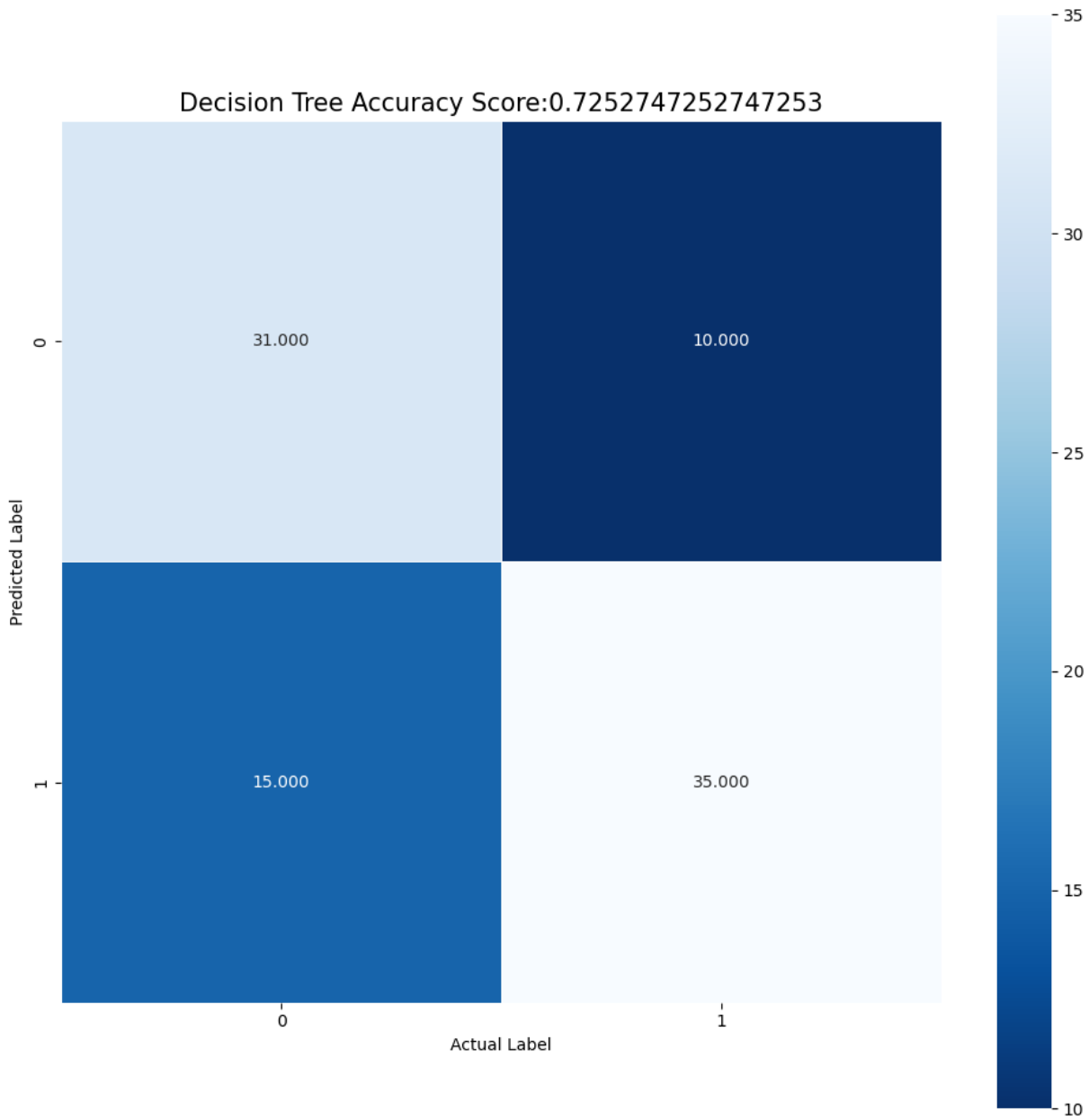
Out[16]:
```
▾        DecisionTreeClassifier

DecisionTreeClassifier(random_state=0)
```

In [17]:
```
tree_pred = clf.predict(x_test)
tree_score = metrics.accuracy_score(y_test,tree_pred)
print("Accruracy:",tree_score)
print("Report:",metrics.classification_report(y_test,tree_pred))
```

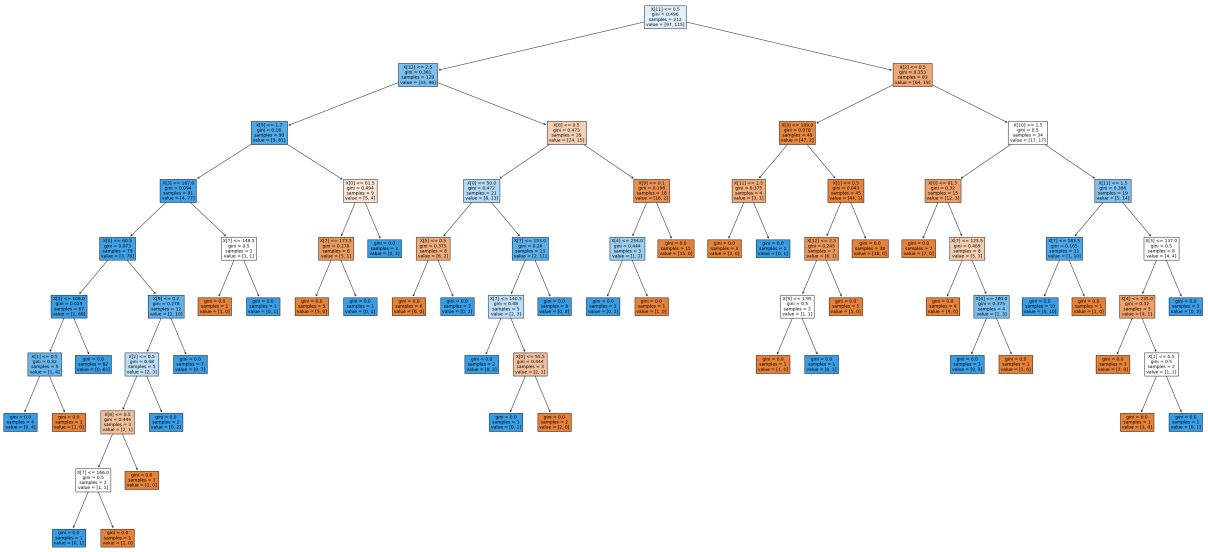
```
Accruracy: 0.7252747252747253
Report:               precision    recall  f1-score   support

           0       0.67      0.76      0.71        41
           1       0.78      0.70      0.74        50

    accuracy                           0.73        91
   macro avg       0.73      0.73      0.72        91
weighted avg       0.73      0.73      0.73        91
```

In [18]:
```
tree_cm = metrics.confusion_matrix(y_test,tree_pred)
```

In [19]:
```
plt.figure(figsize=(12,12))
sns.heatmap(tree_cm,annot=True, fmt=".3f",linewidth=.5,square=True,cmap='Blues_r');
plt.xlabel('Actual Label');
plt.ylabel('Predicted Label');
title ='Decision Tree Accuracy Score:{0}'.format(tree_score)
plt.title(title,size=15);
```



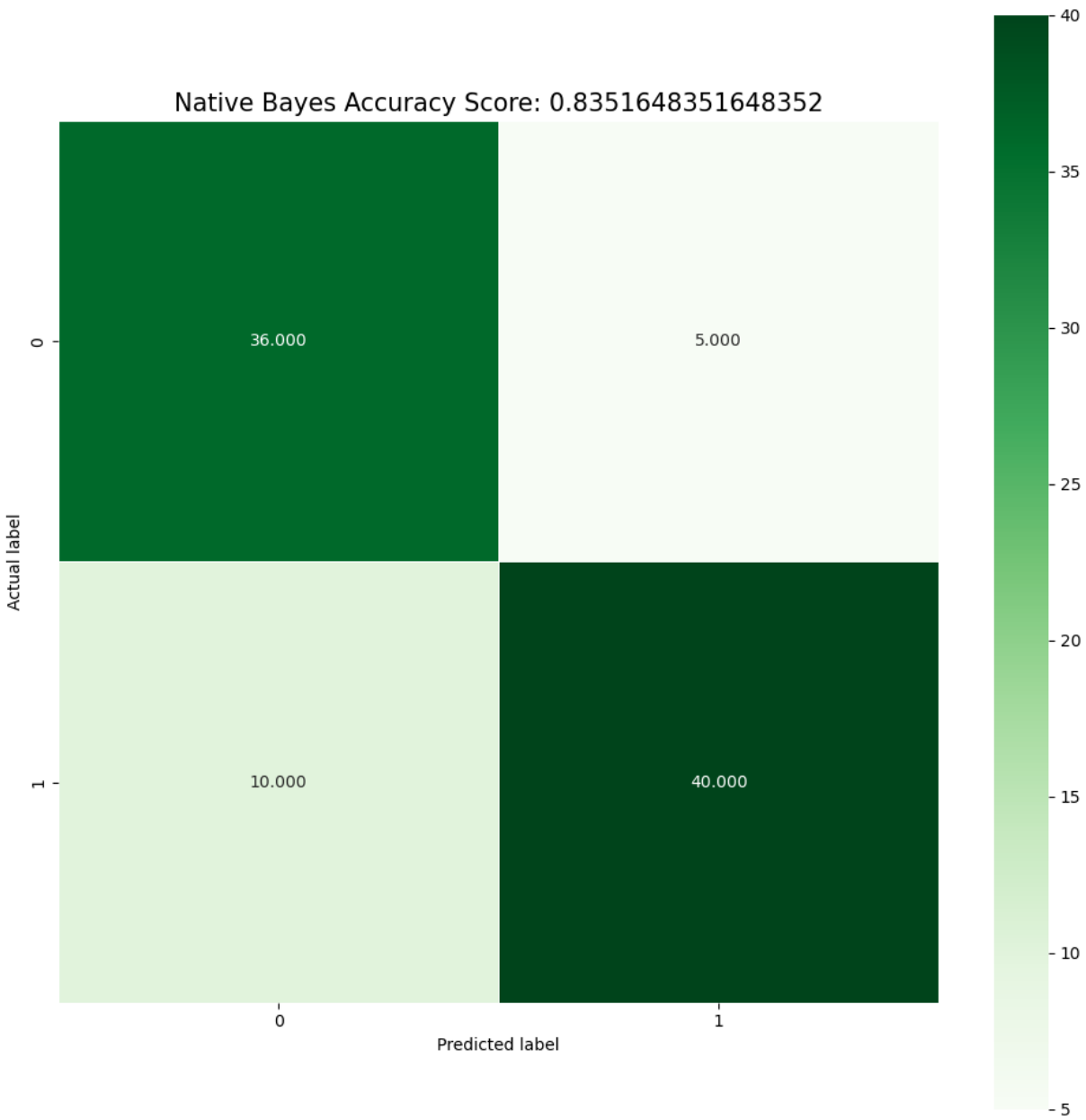

In [20]:
```
fig, ax = plt.subplots(figsize=(50,24))
tree.plot_tree(clf,filled=True,fontsize=10)
plt.savefig('decision_tree',dpi=100)
plt.show()
```



In [21]:
```
gnb = GaussianNB()
bayes_pred = gnb.fit(x_train, y_train).predict(x_test)
bayes_score = metrics.accuracy_score(y_test, bayes_pred)
print("Accuracy: ", bayes_score)
print("Report: ", metrics.classification_report(y_test, bayes_pred))
```

```
Accuracy:  0.8351648351648352
Report:                precision    recall  f1-score   support

           0       0.78      0.88      0.83        41
           1       0.89      0.80      0.84        50

    accuracy                           0.84        91
   macro avg       0.84      0.84      0.83        91
weighted avg       0.84      0.84      0.84        91
```

In [22]:
```
bayes_cm = metrics.confusion_matrix(y_test, bayes_pred)
plt.figure(figsize=(12,12))
sns.heatmap(bayes_cm,annot=True, fmt=".3f",linewidth=.5,square=True,cmap='Greens');
plt.ylabel('Actual label');
plt.xlabel('Predicted label');
title = 'Native Bayes Accuracy Score: {0}'.format(bayes_score)
plt.title(title, size=15);
```




So sánh kết quả của các mô hình trên. Dựa vào mô hình ta có độ chính xác của các thuật toán lần lượt là:

- Thuật toán cây ID3 với 72.527%
- Thuật toán Naive Bayes với 83.516%
- Thuật toán cây CART với 72.527%

Vậy đối với mô hình này sử dụng thuật toán Naive Bayes cho ra độ chính xác cao nhất

In [ ]: