Project : Predicting Heart Disease

Problem Statement:

You are the data scientist at a medical research facility. The facility wants you to build a machine learning model to classify if the given data of a patient should tell if the patient is at the risk of a heart attack.

Heart Disease Dataset:

UCI Heart Disease Dataset
(https://archive.ics.uci.edu/ml/datasets/Heart+Disease?spm=5176.100239.blogcont54260.8.TRNGoO)

Lab Environment:

Jupyter Notebooks

Domain:

Healthcare

Tasks To Be Performed:

1. Data Analysis:
   a. Import the dataset
   b. Get information about the dataset (mean, max, min, quartiles etc.)
   c. Find the correlation between all fields

2. Data Visualization:
   a. Visualize the number of patients having a heart disease and not having a heart disease
   b. Visualize the age and whether a patient has disease or not
   c. Visualize correlation between all features using a heat map

3. Logistic Regression:
   a. Build a simple logistic regression model:
      i. Divide the dataset in 70:30 ratio
      ii. Build the model on train set and predict the values on test set
      iii. Build the confusion matrix and get the accuracy score

4. Decision Tree:
   a. Build a decision tree model:
      i. Divide the dataset in 70:30 ratio
      ii. Build the model on train set and predict the values on test set
      iii. Build the confusion matrix and calculate the accuracy
      iv. Visualize the decision tree using the Graphviz package

5. Random Forest:
   a. Build a Random Forest model:
      i. Divide the dataset in 70:30 ratio
      ii. Build the model on train set and predict the values on test set
      iii. Build the confusion matrix and calculate the accuracy
      iv. Visualize the model using the Graphviz package

6. Select the best model
   a. Print the confusion matrix of all classifiers
   b. Print the classification report of all classifiers
   c. Calculate Recall Precision and F1 score of all the models
   d. Visualize confusion matrix using heatmaps
   e. Select the best model based on the best accuracies

1. Data Analysis: ¶

In [1]:

```
import pandas as pd   #a. Import the dataset
data = pd.read_csv('dataset.csv')
data.head()
```

Out[1]:

	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

In [2]:

```
data.describe()     #b. Get information about the dataset (mean, max, min, quartiles etc.)
```

Out[2]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	
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	303.0
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	2.31
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.905161	0.469794	1.161075	0.616226	1.022606	0.61
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	0.0
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	2.0
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	2.0
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	3.0
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	3.0

In [3]:

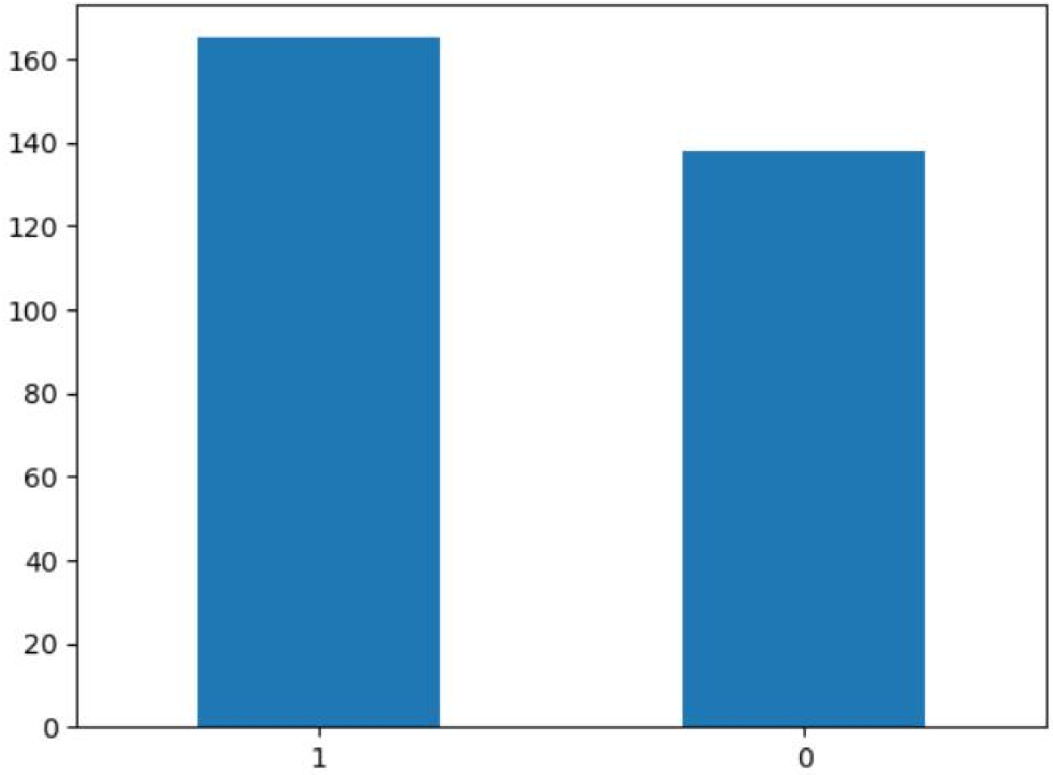
```
data.corr()  #c. Find the correlation between all fields
```

Out[3]:

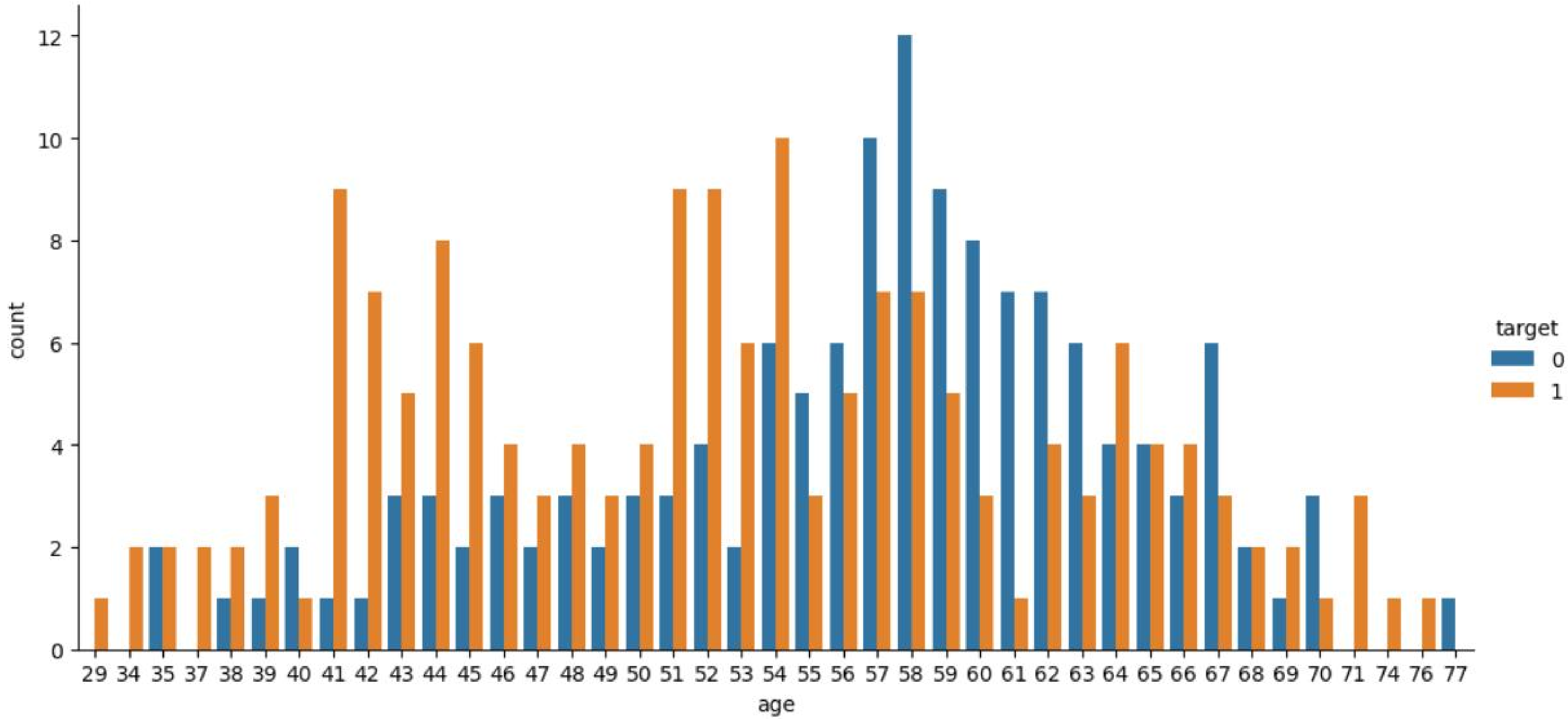
	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
age	1.000000	-0.098447	-0.068653	0.279351	0.213678	0.121308	-0.116211	-0.398522	0.096801	0.210013	-0.168814	0.276326	0.068001	-0.225439
sex	-0.098447	1.000000	-0.049353	-0.056769	-0.197912	0.045032	-0.058196	-0.044020	0.141664	0.096093	-0.030711	0.118261	0.210041	-0.280937
cp	-0.068653	-0.049353	1.000000	0.047608	-0.076904	0.094444	0.044421	0.295762	-0.394280	-0.149230	0.119717	-0.181053	-0.161736	0.433798
trestbps	0.279351	-0.056769	0.047608	1.000000	0.123174	0.177531	-0.114103	-0.046698	0.067616	0.193216	-0.121475	0.101389	0.062210	-0.144931
chol	0.213678	-0.197912	-0.076904	0.123174	1.000000	0.013294	-0.151040	-0.009940	0.067023	0.053952	-0.004038	0.070511	0.098803	-0.085239
fbs	0.121308	0.045032	0.094444	0.177531	0.013294	1.000000	-0.084189	-0.008567	0.025665	0.005747	-0.059894	0.137979	-0.032019	-0.028046
restecg	-0.116211	-0.058196	0.044421	-0.114103	-0.151040	-0.084189	1.000000	0.044123	-0.070733	-0.058770	0.093045	-0.072042	-0.011981	0.137230
thalach	-0.398522	-0.044020	0.295762	-0.046698	-0.009940	-0.008567	0.044123	1.000000	-0.378812	-0.344187	0.386784	-0.213177	-0.096439	0.421741
exang	0.096801	0.141664	-0.394280	0.067616	0.067023	0.025665	-0.070733	-0.378812	1.000000	0.288223	-0.257748	0.115739	0.206754	-0.436757
oldpeak	0.210013	0.096093	-0.149230	0.193216	0.053952	0.005747	-0.058770	-0.344187	0.288223	1.000000	-0.577537	0.222682	0.210244	-0.430696
slope	-0.168814	-0.030711	0.119717	-0.121475	-0.004038	-0.059894	0.093045	0.386784	-0.257748	-0.577537	1.000000	-0.080155	-0.104764	0.345877
ca	0.276326	0.118261	-0.181053	0.101389	0.070511	0.137979	-0.072042	-0.213177	0.115739	0.222682	-0.080155	1.000000	0.151832	-0.391724
thal	0.068001	0.210041	-0.161736	0.062210	0.098803	-0.032019	-0.011981	-0.096439	0.206754	0.210244	-0.104764	0.151832	1.000000	-0.344029
target	-0.225439	-0.280937	0.433798	-0.144931	-0.085239	-0.028046	0.137230	0.421741	-0.436757	-0.430696	0.345877	-0.391724	-0.344029	1.000000

2. Data Visualization: ✓

In [4]:
```
import matplotlib.pyplot as plt  #a. Visualize the number of patients having a heart disease and not having a heart disease.
data['target'].value_counts().plot(kind='bar')
plt.xticks(rotation=0)
plt.show()
```

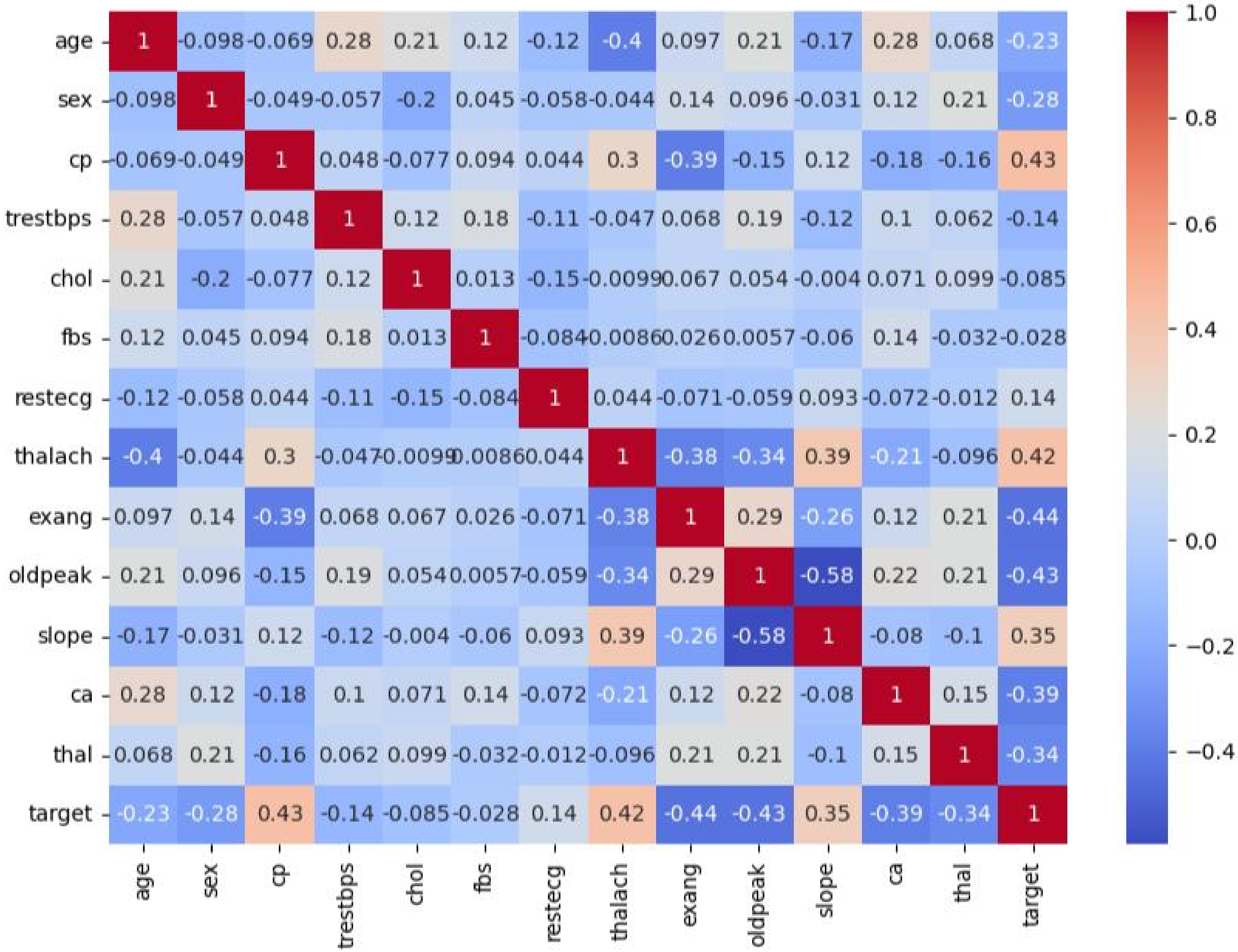


```
In [5]: import seaborn as sns  #b. Visualize the age and whether a patient has disease or not

        sns.catplot(x='age', hue='target', kind='count', data=data, aspect=2)

Out[5]: <seaborn.axisgrid.FacetGrid at 0x1fd731fb7c0>
```



```
In [6]: plt.figure(figsize=(10,7))    #c. Visualize correlation between all features using a heat map
        sns.heatmap(data.corr(), annot=True, cmap='coolwarm')
```

```
Out[6]: <Axes: >
```



3. Logistic Regression:

In [7]:
```
#a. Build a simple logistic regression model:
#i. Divide the dataset in 70:30 ratio
#ii. Build the model on train set and predict the values on test set
#iii. Build the confusion matrix and get the accuracy score
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, accuracy_score

# Split the dataset into training and testing sets
X_train_lg, X_test_lg, y_train_lg, y_test_lg = train_test_split(data.drop('target', axis=1), data['target'], test_size=0.3)

# Build the logistic regression model on the training set
model = LogisticRegression()
model.fit(X_train_lg, y_train_lg)

# Predict the values on the test set
y_pred_lg = model.predict(X_test_lg)

# Build the confusion matrix and get the accuracy score
cm_lg = confusion_matrix(y_test_lg, y_pred_lg)
accuracy_lg = accuracy_score(y_test_lg, y_pred_lg)

print('Confusion Matrix:\n', cm_lg)
print('Accuracy Score:', accuracy_lg)
```

```
Confusion Matrix:
 [[29 12]
 [ 8 42]]
Accuracy Score: 0.7802197802197802
```

4. Decision Tree:

In [8]:
```python
#a. Build a decision tree model:
#i. Divide the dataset in 70:30 ratio
#ii. Build the model on train set and predict the values on test set
#iii. Build the confusion matrix and calculate the accuracy

from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix, accuracy_score

# Split the dataset into training and testing sets
X_train_dt, X_test_dt, y_train_dt, y_test_dt = train_test_split(data.drop('target', axis=1), data['target'], test_size=0.3)

# Build the decision tree model on the training set
model = DecisionTreeClassifier()
model.fit(X_train_dt, y_train_dt)

# Predict the values on the test set
y_pred_dt = model.predict(X_test_dt)

# Build the confusion matrix and calculate the accuracy
cm_dt = confusion_matrix(y_test_dt, y_pred_dt)
accuracy_dt = accuracy_score(y_test_dt, y_pred_dt)

print('Confusion Matrix:\n', cm_dt)
print('Accuracy Score:', accuracy_dt)
```

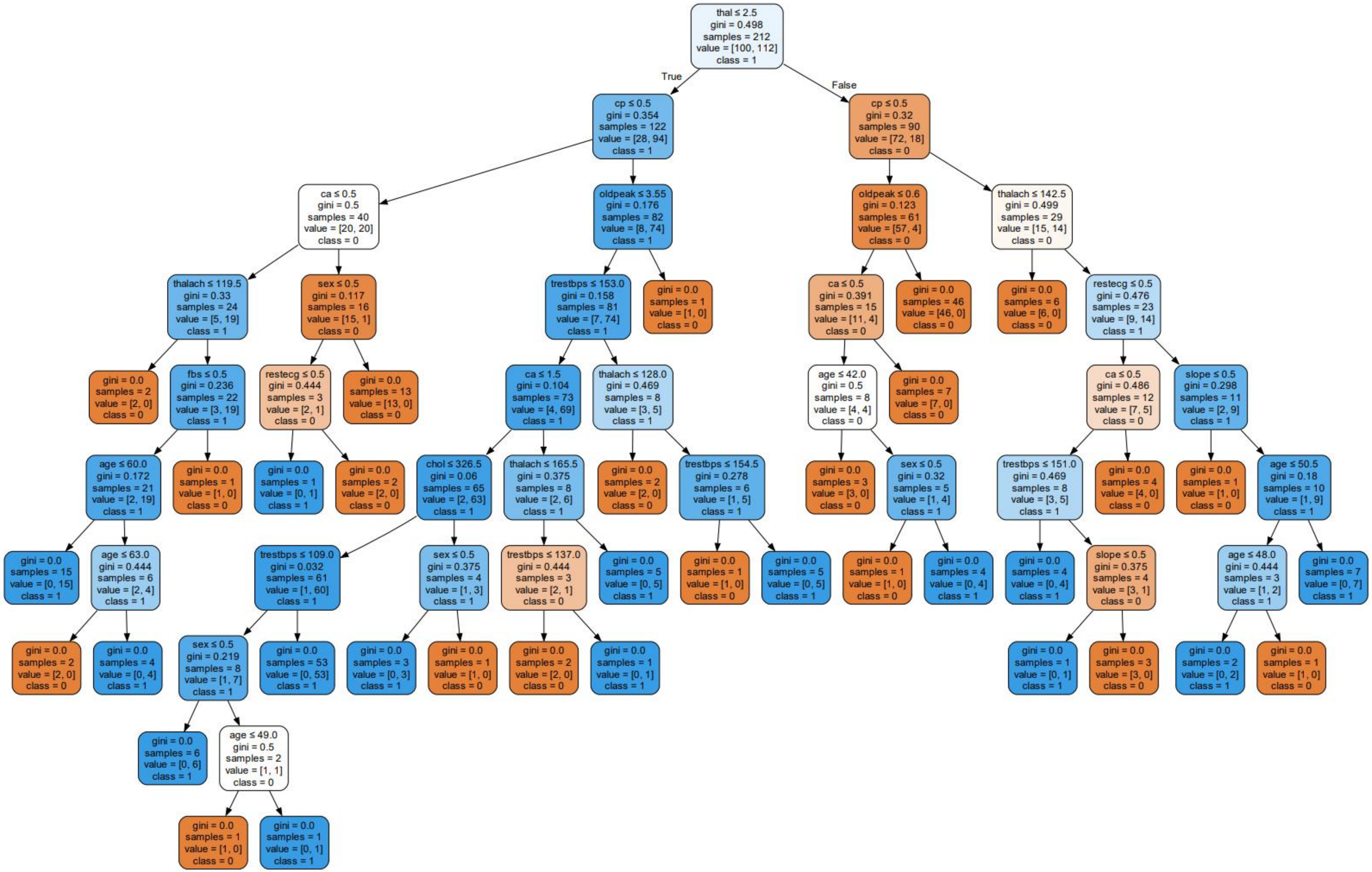
```
Confusion Matrix:
 [[25 13]
 [ 9 44]]
Accuracy Score: 0.7582417582417582
```

In [9]:
```
!pip install graphviz
```

```
Requirement already satisfied: graphviz in c:\users\user\anaconda3\lib\site-packages (0.20.1)
```

In [10]:
```
#iv. Visualize the decision tree using the Graphviz package
from sklearn.tree import export_graphviz
import graphviz

# Visualize the decision tree using Graphviz package
dot_data = export_graphviz(model, out_file=None,
                           feature_names=data.drop('target', axis=1).columns,
                           class_names=['0', '1'],
                           filled=True, rounded=True,
                           special_characters=True)
graph = graphviz.Source(dot_data)
graph.render('decision_tree')
```

Out[10]:
```
'decision_tree.pdf'
```


thal ≤ 2.5 gini = 0.498 samples = 212 value = [100, 112] class = 1
True
False
cp ≤ 0.5 gini = 0.354 samples = 122 value = [28, 94] class = 1
cp ≤ 0.5 gini = 0.32 samples = 90 value = [72, 18] class = 0
ca ≤ 0.5 gini = 0.5 samples = 40 value = [20, 20] class = 0
oldpeak ≤ 3.55 gini = 0.176 samples = 82 value = [8, 74] class = 1
oldpeak ≤ 0.6 gini = 0.123 samples = 61 value = [57, 4] class = 0
thalach ≤ 142.5 gini = 0.499 samples = 29 value = [15, 14] class = 0
thalach ≤ 119.5 gini = 0.33 samples = 24 value = [5, 19] class = 1
sex ≤ 0.5 gini = 0.117 samples = 16 value = [15, 1] class = 0
trestbps ≤ 153.0 gini = 0.158 samples = 81 value = [7, 74] class = 1
gini = 0.0 samples = 1 value = [1, 0] class = 0
ca ≤ 0.5 gini = 0.391 samples = 15 value = [11, 4] class = 0
gini = 0.0 samples = 46 value = [46, 0] class = 0
gini = 0.0 samples = 6 value = [6, 0] class = 0
restecg ≤ 0.5 gini = 0.476 samples = 23 value = [9, 14] class = 1
gini = 0.0 samples = 2 value = [2, 0] class = 0
fbs ≤ 0.5 gini = 0.236 samples = 22 value = [3, 19] class = 1
restecg ≤ 0.5 gini = 0.444 samples = 3 value = [2, 1] class = 0
gini = 0.0 samples = 13 value = [13, 0] class = 0
ca ≤ 1.5 gini = 0.104 samples = 73 value = [4, 69] class = 1
thalach ≤ 128.0 gini = 0.469 samples = 8 value = [3, 5] class = 1
age ≤ 42.0 gini = 0.5 samples = 8 value = [4, 4] class = 0
gini = 0.0 samples = 7 value = [7, 0] class = 0
ca ≤ 0.5 gini = 0.486 samples = 12 value = [7, 5] class = 0
slope ≤ 0.5 gini = 0.298 samples = 11 value = [2, 9] class = 1
age ≤ 60.0 gini = 0.172 samples = 21 value = [2, 19] class = 1
gini = 0.0 samples = 1 value = [1, 0] class = 0
gini = 0.0 samples = 1 value = [0, 1] class = 1
gini = 0.0 samples = 2 value = [2, 0] class = 0
chol ≤ 326.5 gini = 0.06 samples = 65 value = [2, 63] class = 1
thalach ≤ 165.5 gini = 0.375 samples = 8 value = [2, 6] class = 1
gini = 0.0 samples = 2 value = [2, 0] class = 0
trestbps ≤ 154.5 gini = 0.278 samples = 6 value = [1, 5] class = 1
gini = 0.0 samples = 3 value = [3, 0] class = 0
sex ≤ 0.5 gini = 0.32 samples = 5 value = [1, 4] class = 1
trestbps ≤ 151.0 gini = 0.469 samples = 8 value = [3, 5] class = 1
gini = 0.0 samples = 4 value = [4, 0] class = 0
gini = 0.0 samples = 1 value = [1, 0] class = 0
age ≤ 50.5 gini = 0.18 samples = 10 value = [1, 9] class = 1
gini = 0.0 samples = 15 value = [0, 15] class = 1
age ≤ 63.0 gini = 0.444 samples = 6 value = [2, 4] class = 1
trestbps ≤ 109.0 gini = 0.032 samples = 61 value = [1, 60] class = 1
sex ≤ 0.5 gini = 0.375 samples = 4 value = [1, 3] class = 1
trestbps ≤ 137.0 gini = 0.444 samples = 3 value = [2, 1] class = 0
gini = 0.0 samples = 5 value = [0, 5] class = 1
gini = 0.0 samples = 1 value = [1, 0] class = 0
gini = 0.0 samples = 5 value = [0, 5] class = 1
gini = 0.0 samples = 1 value = [1, 0] class = 0
gini = 0.0 samples = 4 value = [0, 4] class = 1
gini = 0.0 samples = 4 value = [0, 4] class = 1
slope ≤ 0.5 gini = 0.375 samples = 4 value = [3, 1] class = 0
age ≤ 48.0 gini = 0.444 samples = 3 value = [1, 2] class = 1
gini = 0.0 samples = 7 value = [0, 7] class = 1
gini = 0.0 samples = 2 value = [2, 0] class = 0
gini = 0.0 samples = 4 value = [0, 4] class = 1
sex ≤ 0.5 gini = 0.219 samples = 8 value = [1, 7] class = 1
gini = 0.0 samples = 53 value = [0, 53] class = 1
gini = 0.0 samples = 3 value = [0, 3] class = 1
gini = 0.0 samples = 1 value = [1, 0] class = 0
gini = 0.0 samples = 2 value = [2, 0] class = 0
gini = 0.0 samples = 1 value = [0, 1] class = 1
gini = 0.0 samples = 1 value = [0, 1] class = 1
gini = 0.0 samples = 3 value = [3, 0] class = 0
gini = 0.0 samples = 2 value = [0, 2] class = 1
gini = 0.0 samples = 1 value = [1, 0] class = 0
gini = 0.0 samples = 6 value = [0, 6] class = 1
age ≤ 49.0 gini = 0.5 samples = 2 value = [1, 1] class = 0
gini = 0.0 samples = 1 value = [1, 0] class = 0
gini = 0.0 samples = 1 value = [0, 1] class = 1

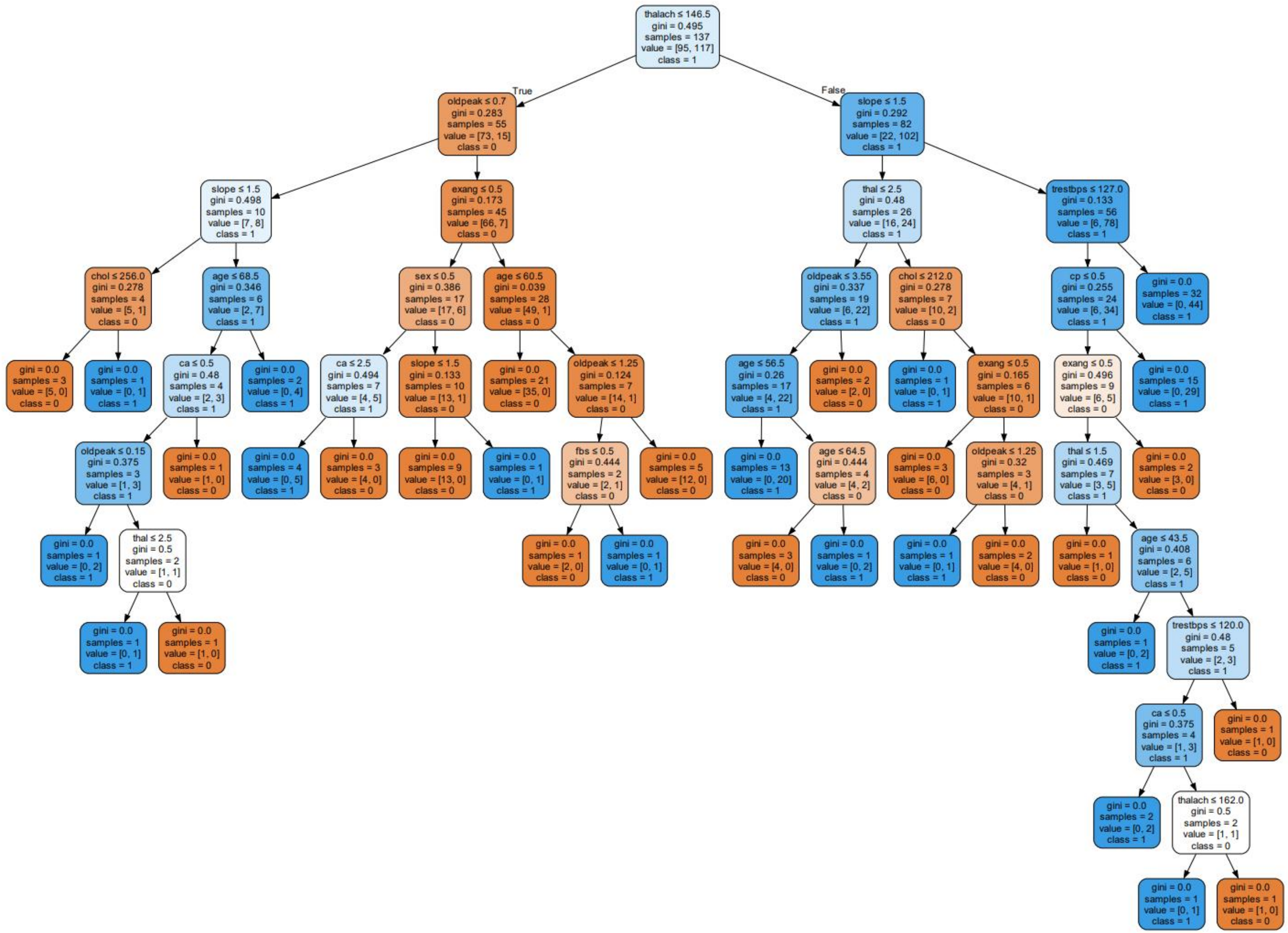
5. Random Forest:

In [11]:
```
#a. Build a Random Forest model:
##i. Divide the dataset in 70:30 ratio
#ii. Build the model on train set and predict the values on test set
#iii. Build the confusion matrix and calculate the accuracy

from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, accuracy_score

# Split the dataset into training and testing sets
X_train_rf, X_test_rf, y_train_rf, y_test_rf = train_test_split(data.drop('target', axis=1), data['target'], test_size=0.3)

# Build the Random Forest model on the training set
model = RandomForestClassifier()
model.fit(X_train_rf, y_train_rf)

# Predict the values on the test set
y_pred_rf = model.predict(X_test_rf)

# Build the confusion matrix and calculate the accuracy
cm_rf = confusion_matrix(y_test_rf, y_pred_rf)
accuracy_rf = accuracy_score(y_test_rf, y_pred_rf)

print('Confusion Matrix:\n', cm_rf)
print('Accuracy Score:', accuracy_rf)
```

```
Confusion Matrix:
 [[32 15]
 [ 2 42]]
Accuracy Score: 0.8131868131868132
```

In [12]:
```
#iv. Visualize the model using the Graphviz package
# Visualize the model using Graphviz package
dot_data = export_graphviz(model.estimators_[0], out_file=None,
                           feature_names=data.drop('target', axis=1).columns,
                           class_names=['0', '1'],
                           filled=True, rounded=True,
                           special_characters=True)
graph = graphviz.Source(dot_data)
graph.render('random_forest')
```

Out[12]:
```
'random_forest.pdf'
```


thalach ≤ 146.5 gini = 0.495 samples = 137 value = [95, 117] class = 1
True
False
oldpeak ≤ 0.7 gini = 0.283 samples = 55 value = [73, 15] class = 0
slope ≤ 1.5 gini = 0.292 samples = 82 value = [22, 102] class = 1
slope ≤ 1.5 gini = 0.498 samples = 10 value = [7, 8] class = 1
exang ≤ 0.5 gini = 0.173 samples = 45 value = [66, 7] class = 0
thal ≤ 2.5 gini = 0.48 samples = 26 value = [16, 24] class = 1
trestbps ≤ 127.0 gini = 0.133 samples = 56 value = [6, 78] class = 1
chol ≤ 256.0 gini = 0.278 samples = 4 value = [5, 1] class = 0
age ≤ 68.5 gini = 0.346 samples = 6 value = [2, 7] class = 1
sex ≤ 0.5 gini = 0.386 samples = 17 value = [17, 6] class = 0
age ≤ 60.5 gini = 0.039 samples = 28 value = [49, 1] class = 0
oldpeak ≤ 3.55 gini = 0.337 samples = 19 value = [6, 22] class = 1
chol ≤ 212.0 gini = 0.278 samples = 7 value = [10, 2] class = 0
cp ≤ 0.5 gini = 0.255 samples = 24 value = [6, 34] class = 1
gini = 0.0 samples = 32 value = [0, 44] class = 1
gini = 0.0 samples = 3 value = [5, 0] class = 0
gini = 0.0 samples = 1 value = [0, 1] class = 1
ca ≤ 0.5 gini = 0.48 samples = 4 value = [2, 3] class = 1
gini = 0.0 samples = 2 value = [0, 4] class = 1
ca ≤ 2.5 gini = 0.494 samples = 7 value = [4, 5] class = 1
slope ≤ 1.5 gini = 0.133 samples = 10 value = [13, 1] class = 0
gini = 0.0 samples = 21 value = [35, 0] class = 0
oldpeak ≤ 1.25 gini = 0.124 samples = 7 value = [14, 1] class = 0
age ≤ 56.5 gini = 0.26 samples = 17 value = [4, 22] class = 1
gini = 0.0 samples = 2 value = [2, 0] class = 0
gini = 0.0 samples = 1 value = [0, 1] class = 1
exang ≤ 0.5 gini = 0.165 samples = 6 value = [10, 1] class = 0
exang ≤ 0.5 gini = 0.496 samples = 9 value = [6, 5] class = 0
gini = 0.0 samples = 15 value = [0, 29] class = 1
oldpeak ≤ 0.15 gini = 0.375 samples = 3 value = [1, 3] class = 1
gini = 0.0 samples = 1 value = [1, 0] class = 0
gini = 0.0 samples = 4 value = [0, 5] class = 1
gini = 0.0 samples = 3 value = [4, 0] class = 0
gini = 0.0 samples = 9 value = [13, 0] class = 0
gini = 0.0 samples = 1 value = [0, 1] class = 1
fbs ≤ 0.5 gini = 0.444 samples = 2 value = [2, 1] class = 0
gini = 0.0 samples = 5 value = [12, 0] class = 0
gini = 0.0 samples = 13 value = [0, 20] class = 1
age ≤ 64.5 gini = 0.444 samples = 4 value = [4, 2] class = 0
gini = 0.0 samples = 3 value = [6, 0] class = 0
oldpeak ≤ 1.25 gini = 0.32 samples = 3 value = [4, 1] class = 0
thal ≤ 1.5 gini = 0.469 samples = 7 value = [3, 5] class = 1
gini = 0.0 samples = 2 value = [3, 0] class = 0
gini = 0.0 samples = 1 value = [0, 2] class = 1
thal ≤ 2.5 gini = 0.5 samples = 2 value = [1, 1] class = 0
gini = 0.0 samples = 1 value = [2, 0] class = 0
gini = 0.0 samples = 1 value = [0, 1] class = 1
gini = 0.0 samples = 3 value = [4, 0] class = 0
gini = 0.0 samples = 1 value = [0, 2] class = 1
gini = 0.0 samples = 1 value = [0, 1] class = 1
gini = 0.0 samples = 2 value = [4, 0] class = 0
gini = 0.0 samples = 1 value = [1, 0] class = 0
age ≤ 43.5 gini = 0.408 samples = 6 value = [2, 5] class = 1
gini = 0.0 samples = 1 value = [0, 1] class = 1
gini = 0.0 samples = 1 value = [1, 0] class = 0
gini = 0.0 samples = 1 value = [0, 2] class = 1
trestbps ≤ 120.0 gini = 0.48 samples = 5 value = [2, 3] class = 1
ca ≤ 0.5 gini = 0.375 samples = 4 value = [1, 3] class = 1
gini = 0.0 samples = 1 value = [1, 0] class = 0
gini = 0.0 samples = 2 value = [0, 2] class = 1
thalach ≤ 162.0 gini = 0.5 samples = 2 value = [1, 1] class = 0
gini = 0.0 samples = 1 value = [0, 1] class = 1
gini = 0.0 samples = 1 value = [1, 0] class = 0

6. Select the best model:

In [13]:
```
#a. Print the confusion matrix of all classifiers
print("confusion matrix of Logistic Regression:\n",cm_lg)

print("confusion matrix of  Decision Tree:\n",cm_dt)

print("confusion matrix of Random Forest:\n",cm_rf)
```

```
confusion matrix of Logistic Regression:
 [[29 12]
 [ 8 42]]
confusion matrix of  Decision Tree:
 [[25 13]
 [ 9 44]]
confusion matrix of Random Forest:
 [[32 15]
 [ 2 42]]
```

In [14]:

```python
#b. Print the classification report of all classifiers
from sklearn.metrics import classification_report

print("Logistic Regression:\n",classification_report(y_test_lg, y_pred_lg))
print("Decision Tree:\n",classification_report(y_test_dt, y_pred_dt))
print(" Random Forest:\n",classification_report(y_test_rf, y_pred_rf))
```

```
Logistic Regression:
               precision    recall  f1-score   support

           0       0.78      0.71      0.74        41
           1       0.78      0.84      0.81        50

    accuracy                           0.78        91
   macro avg       0.78      0.77      0.78        91
weighted avg       0.78      0.78      0.78        91

Decision Tree:
               precision    recall  f1-score   support

           0       0.74      0.66      0.69        38
           1       0.77      0.83      0.80        53

    accuracy                           0.76        91
   macro avg       0.75      0.74      0.75        91
weighted avg       0.76      0.76      0.76        91

 Random Forest:
               precision    recall  f1-score   support

           0       0.94      0.68      0.79        47
           1       0.74      0.95      0.83        44

    accuracy                           0.81        91
   macro avg       0.84      0.82      0.81        91
weighted avg       0.84      0.81      0.81        91
```

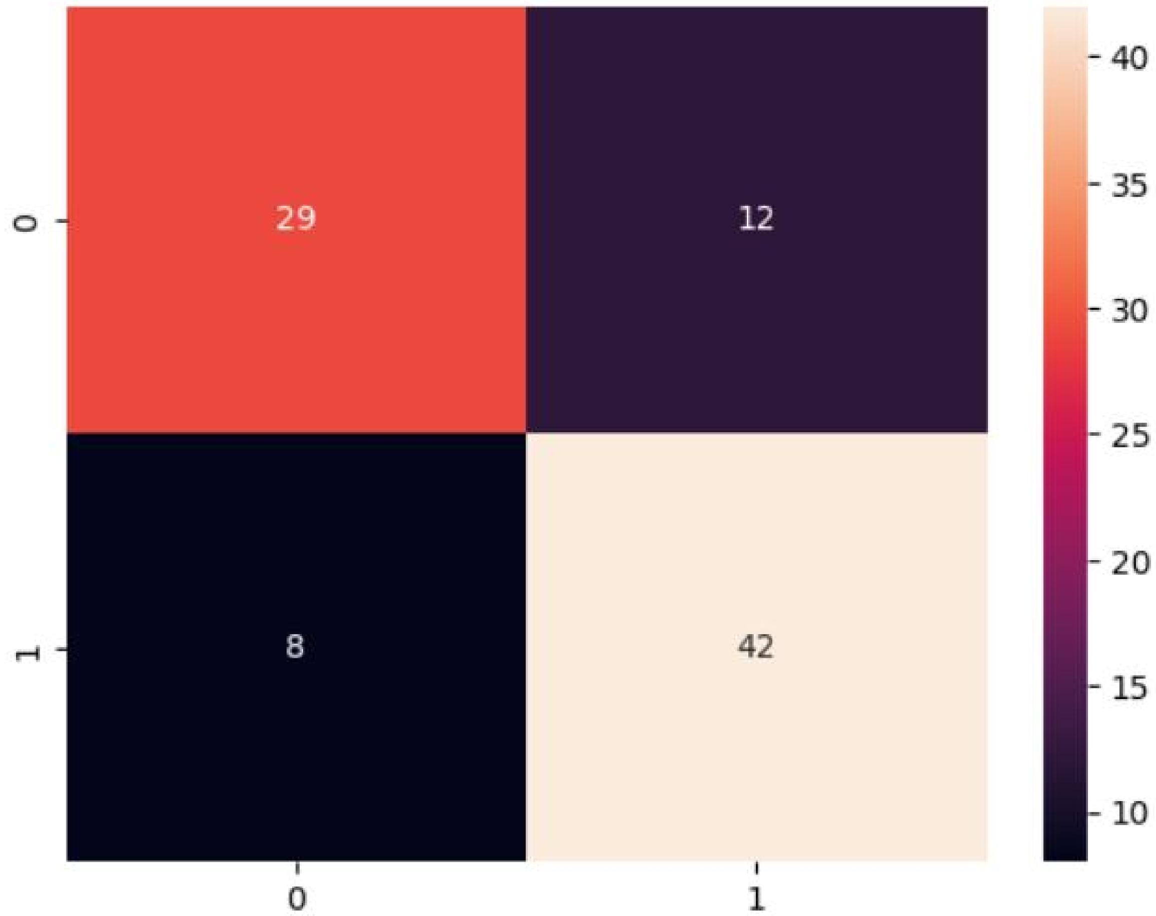
In [15]:
```
#c. Calculate Recall Precision and F1 score of all the models
from sklearn.metrics import recall_score, precision_score, f1_score
print("Logistic Regression:")
print("Recall:", recall_score(y_test_lg, y_pred_lg))
print("Precision:", precision_score(y_test_lg, y_pred_lg))
print("F1 Score:", f1_score(y_test_lg, y_pred_lg))

print("Decision Tree:")
print("Recall:", recall_score(y_test_dt, y_pred_dt))
print("Precision:", precision_score(y_test_dt, y_pred_dt))
print("F1 Score:", f1_score(y_test_dt, y_pred_dt))

print("Random Forest:")
print("Recall:", recall_score(y_test_rf, y_pred_rf))
print("Precision:", precision_score(y_test_rf, y_pred_rf))
print("F1 Score:", f1_score(y_test_rf, y_pred_rf))
```

```
Logistic Regression:
Recall: 0.84
Precision: 0.7777777777777778
F1 Score: 0.8076923076923077
Decision Tree:
Recall: 0.8301886792452831
Precision: 0.7719298245614035
F1 Score: 0.8
Random Forest:
Recall: 0.9545454545454546
Precision: 0.7368421052631579
F1 Score: 0.8316831683168316
```

In [16]:
```python
#d. Visualize confusion matrix using heatmaps
import seaborn as sns

print("Logistic Regression:")
sns.heatmap(confusion_matrix(y_test_lg, y_pred_lg), annot=True)
```

```
Logistic Regression:
```

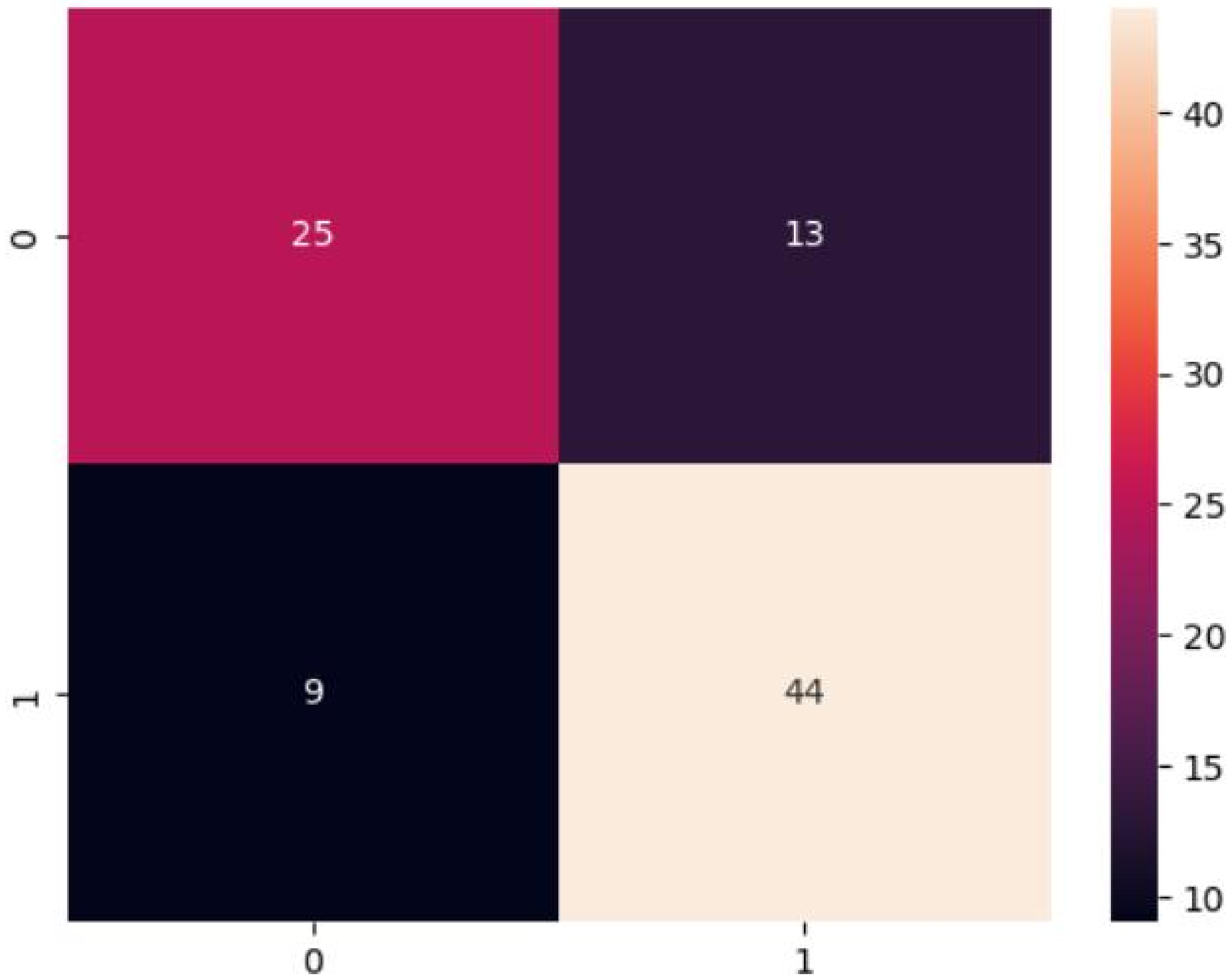
Out[16]: <Axes: >



In [17]:
```
print("Decision Tree:")
sns.heatmap(confusion_matrix(y_test_dt, y_pred_dt), annot=True)
```

```
Decision Tree:
```

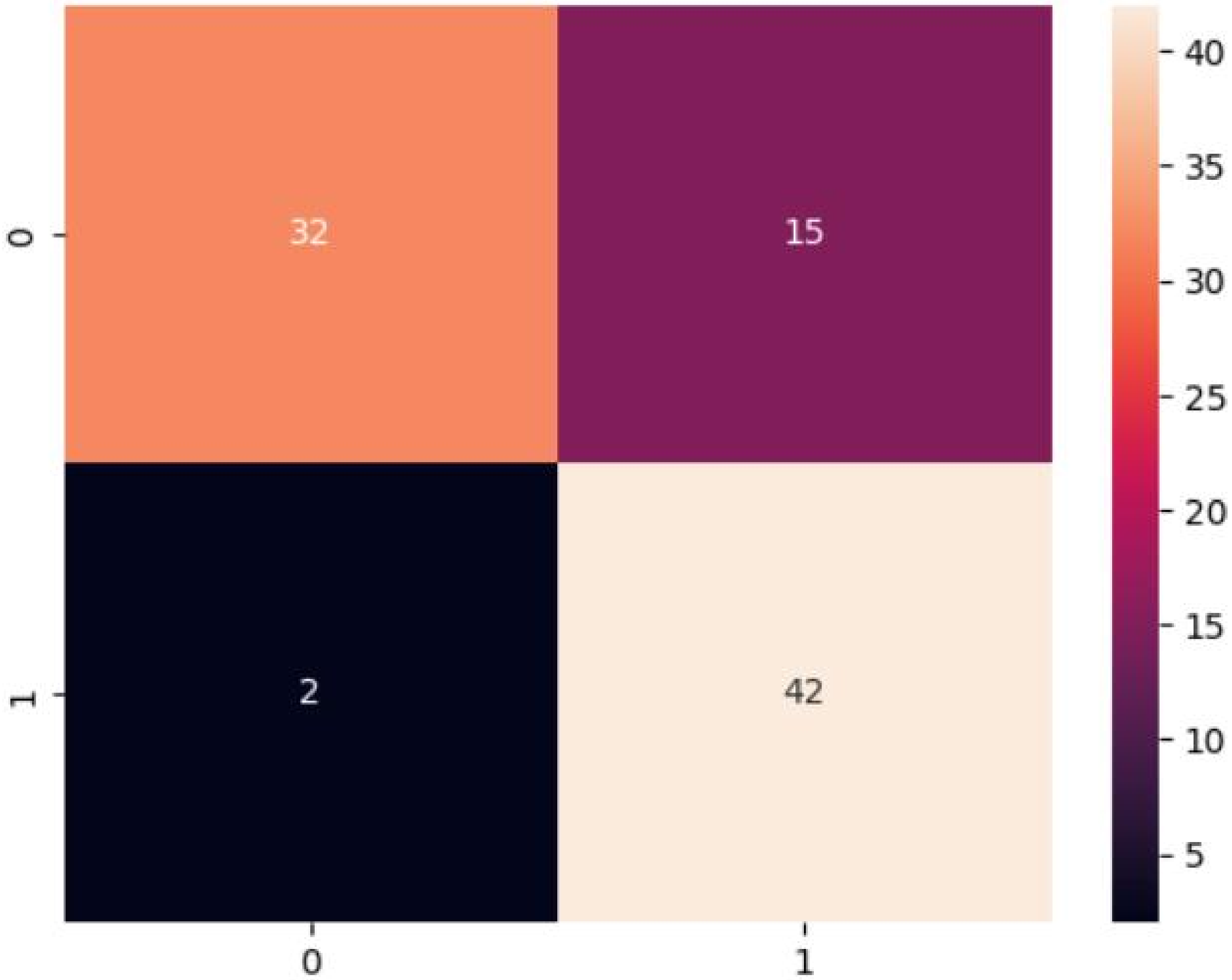
Out[17]: <Axes: >



```
In [18]: print("Random Forest:")
         sns.heatmap(confusion_matrix(y_test_rf, y_pred_rf), annot=True)
```

```
Random Forest:
```

```
Out[18]: <Axes: >
```



In [19]:
```
#e. Select the best model based on the best accuracies
best_accuracy = max(accuracy_lg, accuracy_dt, accuracy_rf)
if best_accuracy == accuracy_lg:
    print("Logistic Regression is the best model")
elif best_accuracy == accuracy_dt:
    print("Decision Tree is the best model")
else:
    print("Random Forest is the best model")
```

```
Random Forest is the best model
```