MAJOR PROJECT

Exploring the workflow of a Random Forest Classifier utilized in classifying heart disease.

PROBLEM STATEMENT:

Understand the Random Forest algorithm's intuition, advantages, and disadvantages. Learn about feature selection using Random Forests and the difference between Random Forests and Decision Trees. Explore the relationship between Random Forests and nearest neighbors, import necessary libraries and datasets, conduct exploratory data analysis, perform feature engineering, and build Random Forest Classifier models with default and tuned parameters. Evaluate model performance using confusion matrix and classification report, visualize important features, and draw conclusions based on the results.

SOLUTION:

Using jupyter notebook for the operation

In [2]:
```python
import pandas as pd
td=pd.read_excel("heart-disease.xlsx")
df=pd.DataFrame(td)
df
```

Out[2]:

	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	68	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 [3]:
```python
df.describe()
```

Out[3]:

	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	303.00
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	0.00
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	2.00
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	2.00
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	3.00
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	3.00

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]:
```
df.head(10)
```

Out[5]:

	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
5	57	1	0	140	192	0	1	148	0	0.4	1	0	1	1
6	56	0	1	140	294	0	0	153	0	1.3	1	0	2	1
7	44	1	1	120	263	0	1	173	0	0.0	2	0	3	1
8	52	1	2	172	199	1	1	162	0	0.5	2	0	3	1
9	57	1	2	150	168	0	1	174	0	1.6	2	0	2	1

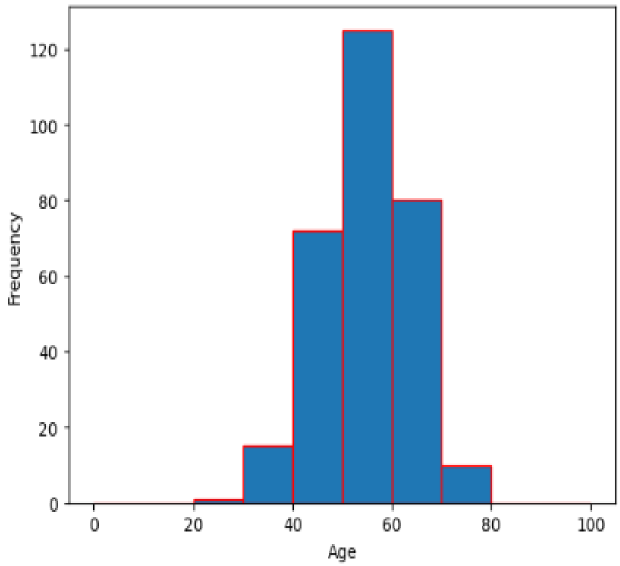
In [6]:
```
df.tail(20)
```
Out[6]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
283	40	1	0	152	223	0	1	181	0	0.0	2	0	3	0
284	61	1	0	140	207	0	0	138	1	1.9	2	1	3	0
285	46	1	0	140	311	0	1	120	1	1.8	1	2	3	0
286	59	1	3	134	204	0	1	162	0	0.8	2	2	2	0
287	57	1	1	154	232	0	0	164	0	0.0	2	1	2	0
288	57	1	0	110	335	0	1	143	1	3.0	1	1	3	0
289	55	0	0	128	205	0	2	130	1	2.0	1	1	3	0
290	61	1	0	148	203	0	1	161	0	0.0	2	1	3	0
291	58	1	0	114	318	0	2	140	0	4.4	0	3	1	0
292	58	0	0	170	225	1	0	146	1	2.8	1	2	1	0
293	67	1	2	152	212	0	0	150	0	0.8	1	0	3	0
294	44	1	0	120	169	0	1	144	1	2.8	0	0	1	0
295	63	1	0	140	187	0	0	144	1	4.0	2	2	3	0
296	63	0	0	124	197	0	1	136	1	0.0	1	0	2	0
297	59	1	0	164	176	1	0	90	0	1.0	1	2	1	0
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	68	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

In [7]:
```
df.shape
```
Out[7]:
```
(303, 14)
```

In [8]:
```
df["age"].describe()
```
Out[8]:
```
count    303.000000
mean      54.366337
std        9.082101
min       29.000000
25%       47.500000
50%       55.000000
75%       61.000000
max       77.000000
Name: age, dtype: float64
```

In [9]:
```
import matplotlib.pyplot as plt
plt.hist(df["age"],bins=[0,10,20,30,40,50,60,70,80,90,100],edgecolor="red")
plt.xlabel("Age")
plt.ylabel("Frequency")
plt.show()
```



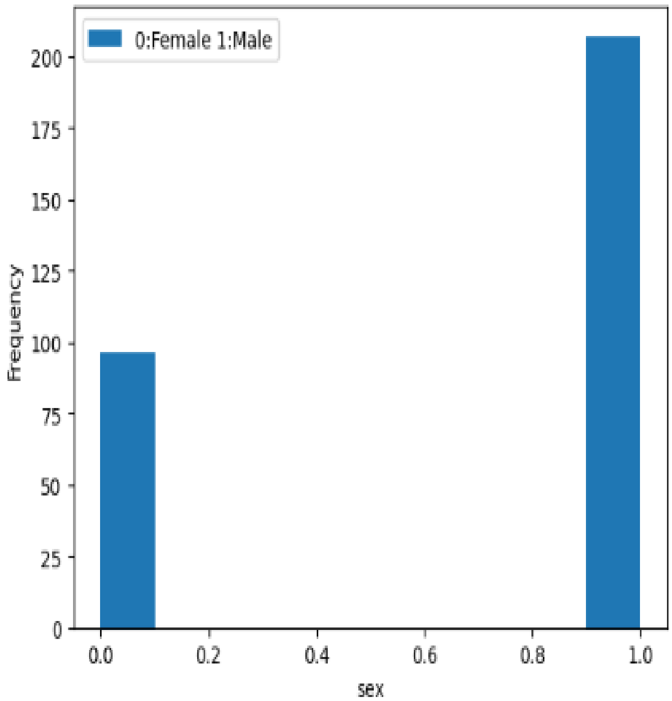
In [10]:
```
df["age"].describe()
```

Out[10]:
```
count    303.000000
mean      54.366337
std        9.082101
min       29.000000
25%       47.500000
50%       55.000000
75%       61.000000
max       77.000000
Name: age, dtype: float64
```

In [11]:
```
df.isnull().sum()
```

Out[11]:
```
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 [12]:
```
plt.hist(df["sex"],label="0:Female 1:Male")
plt.xlabel("sex")
plt.ylabel("Frequency")
plt.legend()
plt.show()
```

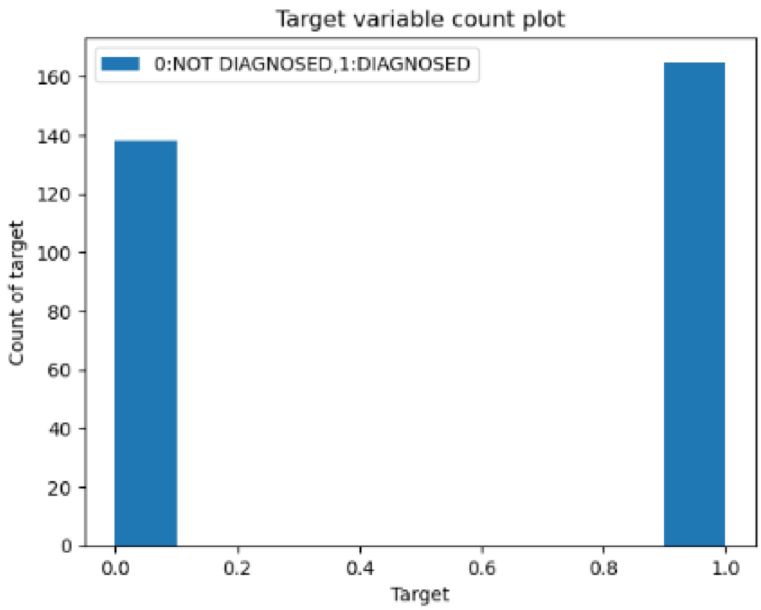


In [13]:
```
df["target"].value_counts()
```

Out[13]:
```
target
1    165
0    138
Name: count, dtype: int64
```

In [42]:

```
import seaborn as sns
plt.hist(df["target"],label="0:NOT DIAGNOSED,1:DIAGNOSED")
plt.xlabel("Target")
plt.ylabel("Count of target")
plt.title("Target variable count plot")
plt.legend()
plt.show()
```



In [26]:
```
X=df.iloc[:,:-1]
Y=df.iloc[:,-1]
```

In [27]:
```
X.shape
```

Out[27]: (303, 13)

In [28]:
```
Y.shape
```

Out[28]: (303,)

In [29]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,random_state=99)
```

In [30]:
```
from sklearn.ensemble import RandomForestClassifier
clf=RandomForestClassifier(criterion="gini",
                           max_depth=8,
                           min_samples_split=10,
                            random_state=5)
```

In [31]:
```
clf.fit(X_train,Y_train)
```

Out[31]:
```
RandomForestClassifier
RandomForestClassifier(max_depth=8, min_samples_split=10, random_state=5)
```

In [32]:
```
clf.feature_importances_
```

Out[32]:
```
array([0.07336235, 0.0394339 , 0.19516544, 0.0612697 , 0.06545926,
       0.00484371, 0.01369233, 0.10298298, 0.04925415, 0.10699116,
       0.03431487, 0.12064637, 0.13258378])
```

In [33]:
```
df.columns
```

Out[33]:
```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')
```

In [34]:
```
Y_pred=clf.predict(X_test)
Y_pred
```

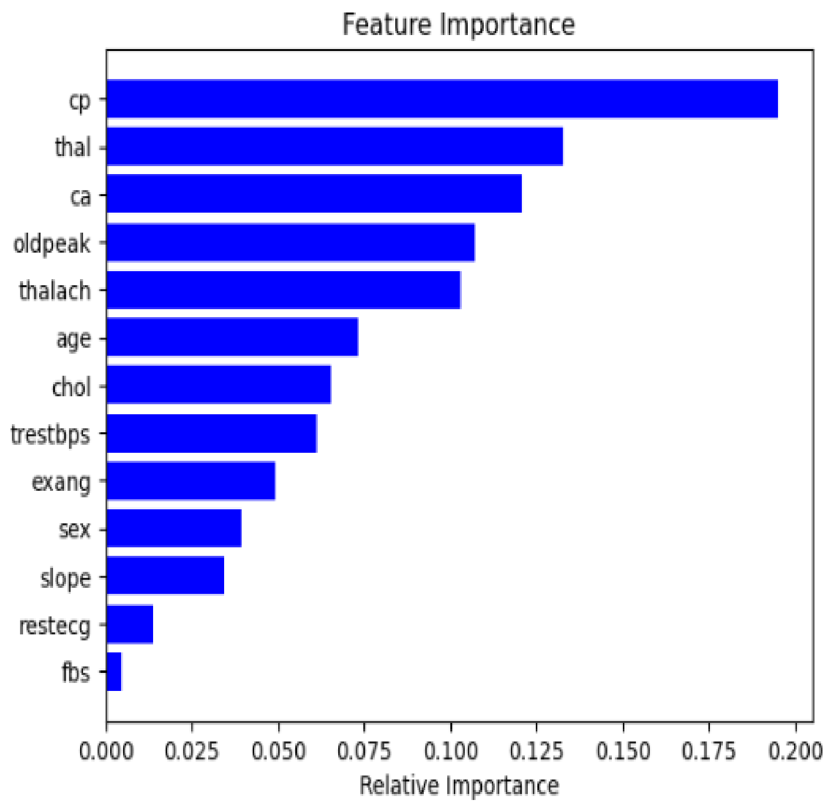
Out[34]:
```
array([1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1,
       0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0,
       1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1,
       0, 1, 1, 0, 1, 1, 1, 0, 1, 1], dtype=int64)
```

In [35]:
```
from sklearn.metrics import confusion_matrix
confusion_matrix(Y_test,Y_pred)
```
Out[35]:
```
array([[23,  6],
       [ 5, 42]], dtype=int64)
```
In [36]:
```
from sklearn.metrics import accuracy_score
accuracy_score(Y_test,Y_pred)
```
Out[36]:
```
0.8552631578947368
```
In [37]:
```
from sklearn.model_selection import cross_val_score
cross_val_score(clf,X_train,Y_train,cv=10)
```
Out[37]:
```
array([0.7826087 , 0.7826087 , 0.82608696, 0.7826087 , 0.73913043,
       0.7826087 , 0.73913043, 0.90909091, 0.90909091, 0.86363636])
```
In [38]:
```
from sklearn.metrics import classification_report
print(classification_report(Y_pred,Y_test))
```
```
              precision    recall  f1-score   support

           0       0.79      0.82      0.81        28
           1       0.89      0.88      0.88        48

    accuracy                           0.86        76
   macro avg       0.84      0.85      0.85        76
weighted avg       0.86      0.86      0.86        76
```
In [40]:
```
import numpy as np
features=df.columns
importances=clf.feature_importances_
indices=np.argsort(importances)

plt.title("Feature Importance")
plt.barh(range(len(indices)),importances[indices],color="b",align="center")
plt.yticks(range(len(indices)),[features[i] for i in indices])
plt.xlabel("Relative Importance")
plt.show()
```



Feature Importance
cp
thal
ca
oldpeak
thalach
age
chol
trestbps
exang
sex
slope
restecg
fbs
0.000
0.025
0.050
0.075
0.100
0.125
0.150
0.175
0.200
Relative Importance