Decision Tree Model For Heart Disease

March 10, 2025

[31]:
```
## Import Required Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.tree import DecisionTreeClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, classification_report,␣
 ↪confusion_matrix
from sklearn import tree
import joblib
from sklearn.preprocessing import LabelEncoder
import warnings
warnings.filterwarnings("ignore")
```

[32]:
```
## Load the Dataset
df = pd.read_csv("C:/Users/PC/OneDrive/Desktop/Data Science/Datasets/Datasets/
 ↪heart.csv")
```

[33]:
```
## Inspect the first view observations of the dataset
df.head(10)
```

[33]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	0
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	0
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	0
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3	0
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	0
5	58	0	0	100	248	0	0	122	0	1.0	1	0	2	0
6	58	1	0	114	318	0	2	140	0	4.4	0	3	1	0
7	55	1	0	160	289	0	0	145	1	0.8	1	1	3	0
8	46	1	0	120	249	0	0	144	0	0.8	2	0	3	0
9	54	1	0	122	286	0	0	116	1	3.2	1	2	3	0

```
2   0     3       0
3   1     3       0
4   3     2       0
5   0     2       1
6   3     1       0
7   1     3       0
8   0     3       0
9   2     2       0
```

[34]:
```
## Assessing the structure of the dataset
df.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
```

[35]:
```
## Checking for missing values
df.isnull().sum()
```

[35]:
```
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
```

[36]:
```
## Check for duplicates
df.duplicated().sum()
```

[36]:
```
723
```

[37]:
```
## Remove duplicates
df = df.drop_duplicates()
```

[38]:
```
## ## Check the unique entries of the outcome variable
print(df["target"].unique())
```

```
[0 1]
```

[39]:
```
## Check target class distribution
print(df["target"].value_counts())
```

```
target
1    164
0    138
Name: count, dtype: int64
```

[40]:
```
## Summary statistics
df.describe()
```

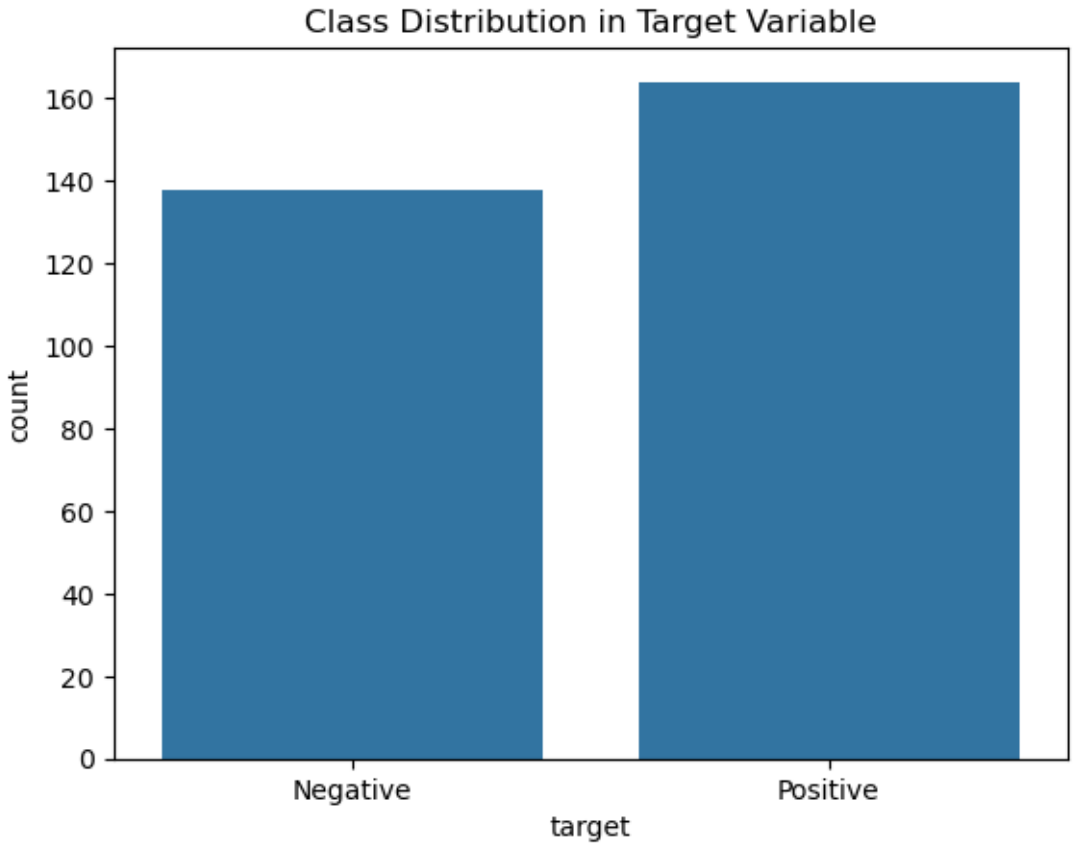
[40]:

	age	sex	cp	trestbps	chol	fbs	\
count	302.00000	302.000000	302.000000	302.000000	302.000000	302.000000	
mean	54.42053	0.682119	0.963576	131.602649	246.500000	0.149007	
std	9.04797	0.466426	1.032044	17.563394	51.753489	0.356686	
min	29.00000	0.000000	0.000000	94.000000	126.000000	0.000000	
25%	48.00000	0.000000	0.000000	120.000000	211.000000	0.000000	
50%	55.50000	1.000000	1.000000	130.000000	240.500000	0.000000	
75%	61.00000	1.000000	2.000000	140.000000	274.750000	0.000000	
max	77.00000	1.000000	3.000000	200.000000	564.000000	1.000000	

	restecg	thalach	exang	oldpeak	slope	ca	\
count	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	
mean	0.526490	149.569536	0.327815	1.043046	1.397351	0.718543	
std	0.526027	22.903527	0.470196	1.161452	0.616274	1.006748	
min	0.000000	71.000000	0.000000	0.000000	0.000000	0.000000	
25%	0.000000	133.250000	0.000000	0.000000	1.000000	0.000000	
50%	1.000000	152.500000	0.000000	0.800000	1.000000	0.000000	
75%	1.000000	166.000000	1.000000	1.600000	2.000000	1.000000	
max	2.000000	202.000000	1.000000	6.200000	2.000000	4.000000	

```
              thal      target
count   302.000000  302.000000
mean      2.314570    0.543046
std       0.613026    0.498970
min       0.000000    0.000000
25%       2.000000    0.000000
50%       2.000000    1.000000
75%       3.000000    1.000000
max       3.000000    1.000000
```

[41]:
```
# Visualize class distribution
sns.countplot(x=df['target'])
plt.title("Class Distribution in Target Variable")
plt.xticks([0, 1], labels = ["Negative", "Positive"])
plt.show()
```



[42]:
```
## Define Features and Target variable
X = df.drop(columns = ["target"])
y = df["target"]
```

[43]:
```
## Split the Data into Training and Testing Sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

# Check the shape
print("Training data shape:", X_train.shape)
print("Testing data shape:", X_test.shape)
```

```
Training data shape: (241, 13)
Testing data shape: (61, 13)
```

[44]:
```
## Feature scaling
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.fit_transform(X_test)
```

[45]:
```
## Train a Decision Tree Model
# Initialize the model
dt_model = DecisionTreeClassifier(criterion='gini', max_depth=3,␣
 ↪random_state=42)

# Train the model
dt_model.fit(X_train, y_train)
```

[45]:
```
DecisionTreeClassifier(max_depth=3, random_state=42)
```

[46]:
```
## Make Predictions on Test Data
# Predict
y_pred = dt_model.predict(X_test)

# Display predictions
print(y_pred)
```

```
[1 1 0 1 1 0 0 1 0 1 0 0 1 1 1 0 0 1 0 1 1 0 0 1 0 0 1 1 0 1 1 1 1 1 1 1 1
 1 0 0 0 0 0 1 0 0 1 0 1 1 0 0 0 1 1 1 1 1 0 0 0]
```

[47]:
```
## Evaluate the Model Performance
# Accuracy score
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
```

```
Accuracy: 0.74
```

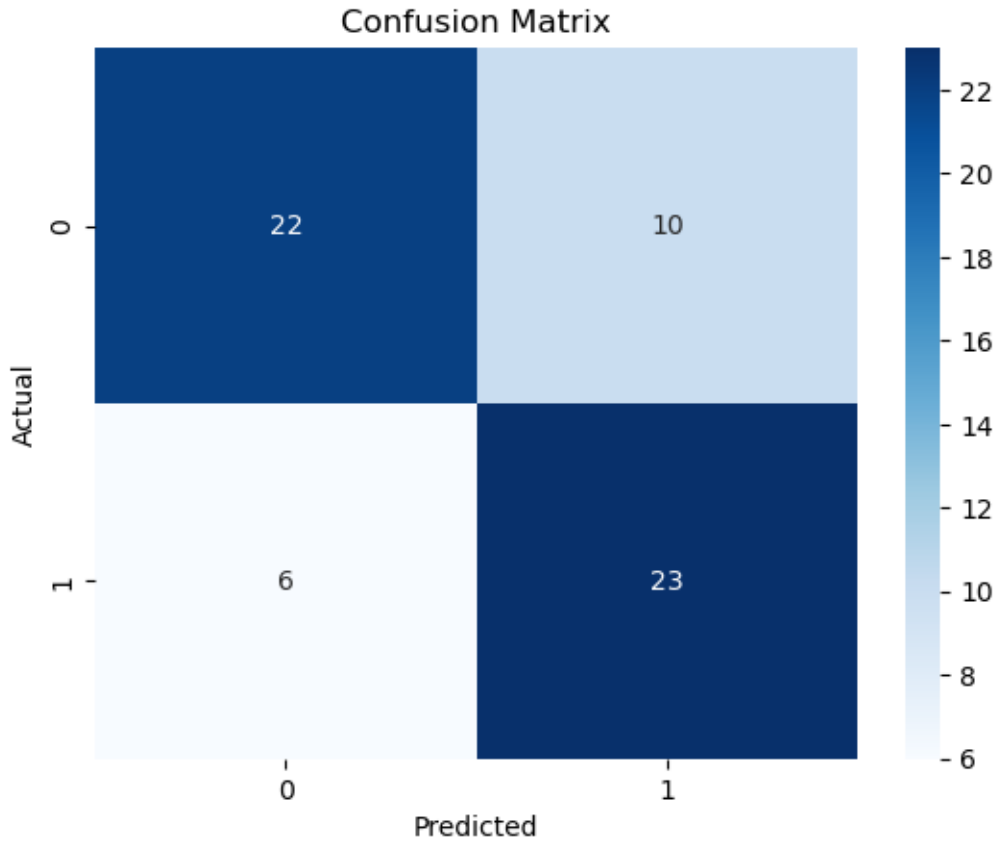
[48]:
```
# Classification report
print("Classification Report:")
print(classification_report(y_test, y_pred))
```

```
Classification Report:
              precision    recall  f1-score   support
```

```
              0       0.79      0.69      0.73        32
              1       0.70      0.79      0.74        29

    accuracy                            0.74        61
   macro avg        0.74      0.74      0.74        61
weighted avg        0.74      0.74      0.74        61
```

[49]:
```
# Confusion matrix
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, cmap="Blues", fmt='d')
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()
```

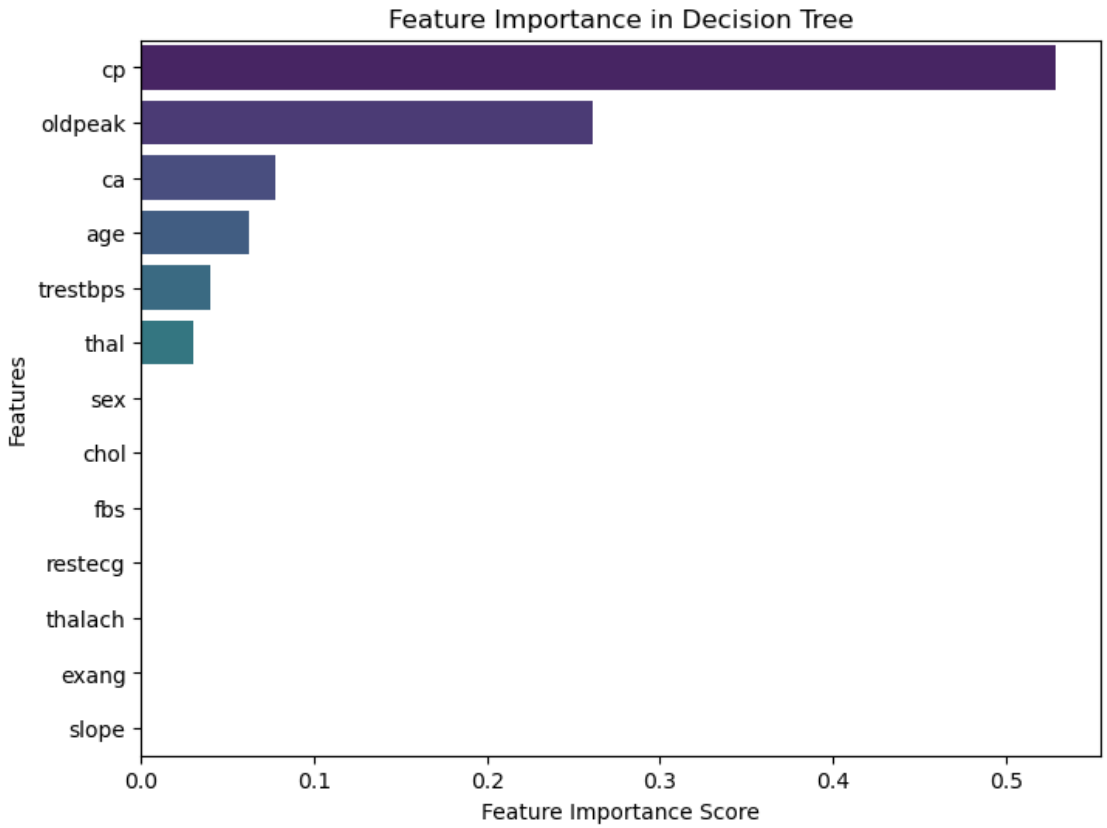


[50]:
```
## Feature Importance Plot
# Get feature importance scores
feature_importance = dt_model.feature_importances_
```

```
feature_names = X.columns  # Feature names
# Create a DataFrame for better visualization
feature_imp_df = pd.DataFrame({'Feature': feature_names, 'Importance':␣
 ↪feature_importance})

# Sort features by importance
feature_imp_df = feature_imp_df.sort_values(by='Importance', ascending=False)

# Plot
plt.figure(figsize=(8, 6))
sns.barplot(x='Importance', y='Feature', data=feature_imp_df, palette='viridis')
plt.xlabel("Feature Importance Score")
plt.ylabel("Features")
plt.title("Feature Importance in Decision Tree")
plt.show()
```



[51]:
```
# Save the model to a file
joblib.dump(dt_model, "decision_tree_model.pkl")

print("Model saved successfully!")
```

```
Model saved successfully!
```