```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import warnings
warnings.filterwarnings('ignore')
```

```
df=pd.read_csv('/content/heart_disease_uci.csv')
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 920 entries, 0 to 919
Data columns (total 16 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   id        920 non-null    int64  
 1   age       920 non-null    int64  
 2   sex       920 non-null    object 
 3   dataset   920 non-null    object 
 4   cp        920 non-null    object 
 5   trestbps  861 non-null    float64
 6   chol      890 non-null    float64
 7   fbs       830 non-null    object 
 8   restecg   918 non-null    object 
 9   thalch    865 non-null    float64
 10  exang     865 non-null    object 
 11  oldpeak   858 non-null    float64
 12  slope     611 non-null    object 
 13  ca        309 non-null    float64
 14  thal      434 non-null    object 
 15  num       920 non-null    int64  
dtypes: float64(5), int64(3), object(8)
memory usage: 115.1+ KB
```

```
df=df.drop(['id','dataset','ca','thal'], axis=1)
```

```
print(df.isnull().sum())
```

```
age           0
sex           0
cp            0
trestbps     59
chol         30
fbs          90
restecg       2
thalch       55
exang        55
oldpeak      62
slope       309
num           0
dtype: int64
```

```
for column in df.columns:
    if df[column].dtype in ['float64', 'int64']:  # Numerical columns
        mean_value = df[column].mean()
        df[column].fillna(mean_value, inplace=True)
    else:  # Categorical columns
        mode_value = df[column].mode()[0]
        df[column].fillna(mode_value, inplace=True)
```

```
df.info()
```

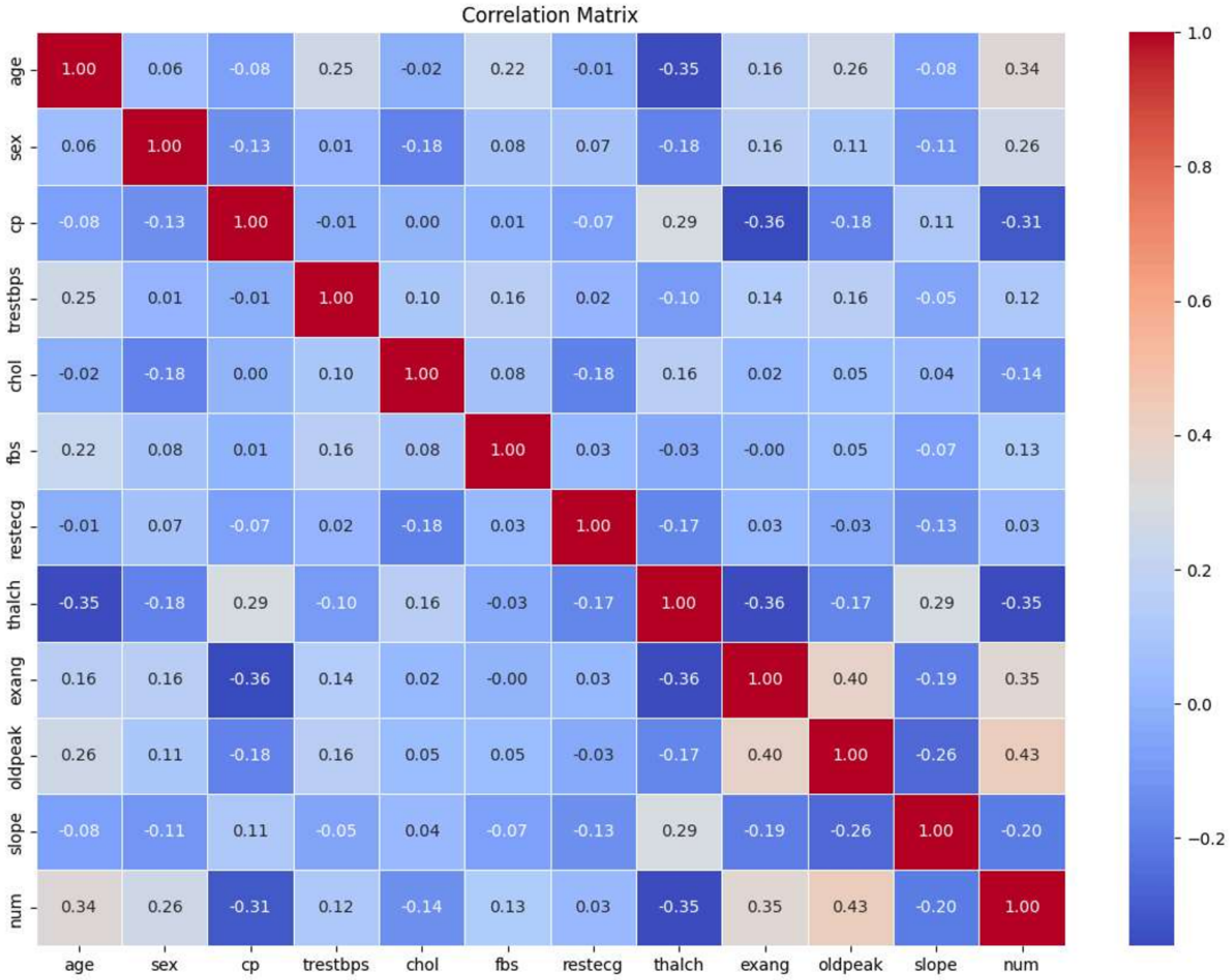
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 920 entries, 0 to 919
Data columns (total 12 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       920 non-null    int64  
 1   sex       920 non-null    object 
 2   cp        920 non-null    object 
 3   trestbps  920 non-null    float64
 4   chol      920 non-null    float64
 5   fbs       920 non-null    bool
 6   restecg   920 non-null    object
 7   thalch    920 non-null    float64
 8   exang     920 non-null    bool
 9   oldpeak   920 non-null    float64
 10  slope     920 non-null    object
 11  num       920 non-null    int64
dtypes: bool(2), float64(4), int64(2), object(4)
memory usage: 73.8+ KB
```

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
for col in df.columns:
    df[col] = le.fit_transform(df[col])
df
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalch	exang	oldpeak	slope	num
0	35	1	3	41	87	1	0	77	0	34	0	0
1	39	1	0	50	140	0	0	34	1	26	1	2
2	39	1	0	22	83	0	0	55	1	37	1	1
3	9	1	2	31	104	0	1	113	0	44	0	0
4	13	0	1	31	58	0	0	99	0	25	2	0
...	...	...	...	...	...	...	...	...	...	...	...	...
915	26	0	0	28	180	1	2	81	0	10	1	1
916	34	1	3	33	8	0	2	64	0	19	1	0
917	27	1	0	23	77	1	2	27	0	10	1	2
918	30	1	0	33	200	1	0	64	0	19	1	0
919	34	1	1	22	108	0	0	20	1	10	1	1

920 rows × 12 columns

```
import seaborn as sns
correlation_matrix = df.corr()
# Print the correlation matrix
plt.figure(figsize=(14, 10))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5)
plt.title('Correlation Matrix')
plt.show()
```



```
# Separate features (X) and target variable (y)
X = df.drop('num', axis=1)
y = df['num'].astype(int)  # Ensure target is of integer type
```

```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

Model Building

```
# Initialize and train the Decision Tree Classifier
clf = DecisionTreeClassifier(criterion='gini', random_state=42)
clf.fit(X_train, y_train)
```

```
▾        DecisionTreeClassifier
DecisionTreeClassifier(random_state=42)
```

```
y_pred = clf.predict(X_test)
```

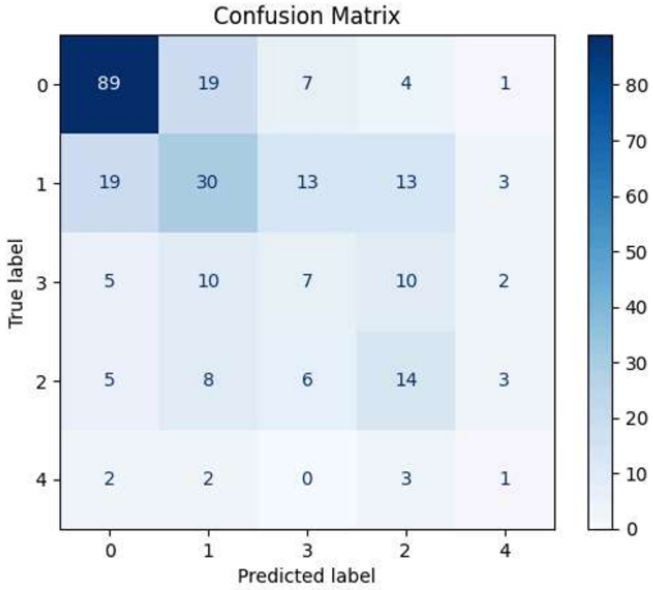
```
y_pred= clf.predict(X_test)
results=X_test.copy()
results['Actual']=y_test
results['Predicted']=y_pred
value=X.columns
results
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalch	exang	oldpeak	slope	Actual	Predicted
319	8	1	1	22	21	0	1	107	0	10	1	0	0
377	17	1	1	38	78	1	1	48	0	10	1	0	1
538	20	1	0	50	177	0	1	19	1	26	1	1	1
296	31	1	0	51	31	1	0	17	0	21	1	3	4
531	12	0	0	44	202	0	1	56	0	31	1	1	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...
447	26	0	1	38	161	0	2	67	0	10	1	0	0
420	23	0	2	14	44	0	1	46	0	10	1	0	0
133	23	1	0	38	115	0	0	112	1	10	2	0	3
490	34	1	1	38	125	0	1	79	0	21	2	0	1
558	18	1	3	38	126	1	1	102	0	31	1	1	0

276 rows × 13 columns

```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt
# Import necessary libraries
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt

# Compute the confusion matrix
cm = confusion_matrix(y_test, y_pred)

# Print the confusion matrix
print("Confusion Matrix:")
print(cm)

# Check the unique classes in y_test
unique_classes = y_test.unique()
print("Unique classes in the target variable:", unique_classes)

# Adjust the display_labels based on the unique classes
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=[str(cls) for cls in unique_classes])

# Plot the confusion matrix
disp.plot(cmap='Blues')
plt.title('Confusion Matrix')
plt.show()
```

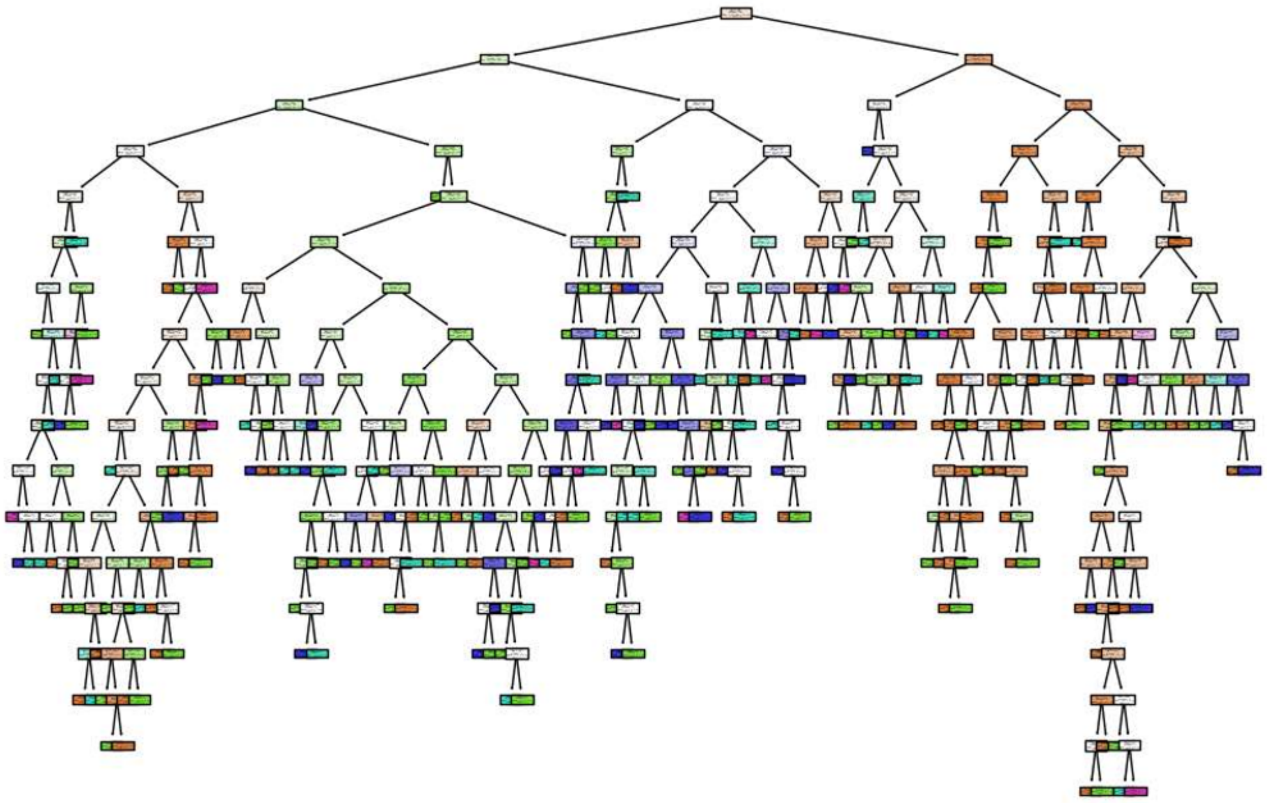
```
Confusion Matrix:
[[89 19  7  4  1]
 [19 30 13 13  3]
 [ 5 10  7 10  2]
 [ 5  8  6 14  3]
 [ 2  2  0  3  1]]
Unique classes in the target variable: [0 1 3 2 4]
```



```
from sklearn.metrics import accuracy_score
from sklearn import metrics
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.5108695652173914
```

```
import matplotlib.pyplot as plt
from sklearn import tree
plt.figure(figsize=(12,8))
tree.plot_tree(clf, feature_names=X.columns, class_names=['0', '1', '2', '3', '4'], filled=True)
plt.show()
```

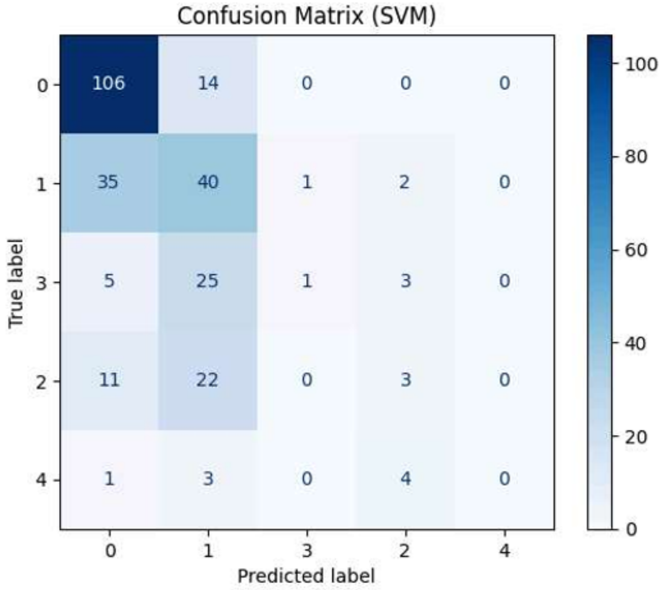


```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.74      0.74      0.74       120
           1       0.43      0.38      0.41        78
           2       0.21      0.21      0.21        34
           3       0.32      0.39      0.35        36
           4       0.10      0.12      0.11         8

    accuracy                           0.51       276
   macro avg       0.36      0.37      0.36       276
weighted avg       0.52      0.51      0.51       276
```

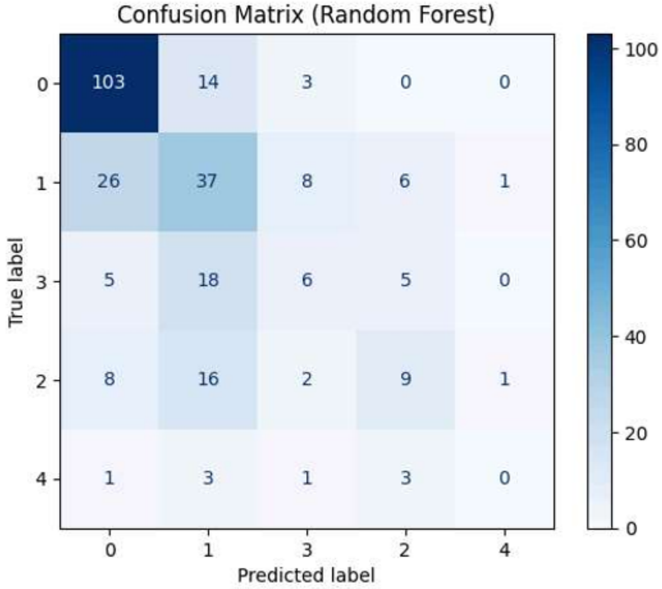
```
# prompt: build model using SVM

import matplotlib.pyplot as plt
from sklearn.svm import SVC

# Initialize the SVM classifier
svm_clf = SVC(kernel='linear', random_state=42)

# Train the model
svm_clf.fit(X_train, y_train)

# Make predictions
y_pred_svm = svm_clf.predict(X_test)

# Evaluate the model
print("Accuracy:", accuracy_score(y_test, y_pred_svm))
print(classification_report(y_test, y_pred_svm))

# Confusion matrix
cm_svm = confusion_matrix(y_test, y_pred_svm)
print("Confusion Matrix:")
print(cm_svm)

# Display the confusion matrix
disp_svm = ConfusionMatrixDisplay(confusion_matrix=cm_svm, display_labels=[str(cls) for cls in unique_classes])
disp_svm.plot(cmap='Blues')
plt.title('Confusion Matrix (SVM)')
plt.show()
```

```
Accuracy: 0.5434782608695652
              precision    recall  f1-score   support

           0       0.67      0.88      0.76       120
           1       0.38      0.51      0.44        78
           2       0.50      0.03      0.06        34
           3       0.25      0.08      0.12        36
           4       0.00      0.00      0.00         8

    accuracy                           0.54       276
   macro avg       0.36      0.30      0.28       276
weighted avg       0.49      0.54      0.48       276

Confusion Matrix:
[[106  14   0   0   0]
 [ 35  40   1   2   0]
 [  5  25   1   3   0]
 [ 11  22   0   3   0]
 [  1   3   0   4   0]]
```



```
# prompt: Build model using random forest

import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestClassifier

# Initialize the Random Forest classifier
rf_clf = RandomForestClassifier(n_estimators=100, random_state=42)

# Train the model
rf_clf.fit(X_train, y_train)

# Make predictions
y_pred_rf = rf_clf.predict(X_test)

# Evaluate the model
print("Accuracy:", accuracy_score(y_test, y_pred_rf))
print(classification_report(y_test, y_pred_rf))

# Confusion matrix
cm_rf = confusion_matrix(y_test, y_pred_rf)
print("Confusion Matrix:")
print(cm_rf)

# Display the confusion matrix
disp_rf = ConfusionMatrixDisplay(confusion_matrix=cm_rf, display_labels=[str(cls) for cls in unique_classes])
disp_rf.plot(cmap='Blues')
plt.title('Confusion Matrix (Random Forest)')
plt.show()
```

```
Accuracy: 0.5615942028985508
              precision    recall  f1-score   support

           0       0.72      0.86      0.78       120
           1       0.42      0.47      0.45        78
           2       0.30      0.18      0.22        34
           3       0.39      0.25      0.31        36
           4       0.00      0.00      0.00         8

    accuracy                           0.56       276
   macro avg       0.37      0.35      0.35       276
weighted avg       0.52      0.56      0.53       276

Confusion Matrix:
[[103  14   3   0   0]
 [ 26  37   8   6   1]
 [  5  18   6   5   0]
 [  8  16   2   9   1]
 [  1   3   1   3   0]]
```


Confusion Matrix (Random Forest)
True label
Predicted label