Project on Explainable AI

Heart Disease Dataset ¶

eXplainable AI or XAI in short, is basically a way to remove the ambiguity in machine learning methods and to enable transparency so that the outcomes of black-box models can be easily understood by humans.

Why XAI - With AI forming the future of all complex decision-making, it is crucial to know how and why these decisions were made. Artificial Intelligence clearly enhances the speed, precision, and effectiveness of human efforts. For example, AI techniques can be used to identify which transactions are likely to be fraudulent, as well as automate manually intense data management tasks in financial institutions, or it can be useful for face recognition in cameras.

However, consider an AI-powered medical diagnosis system that predicts cancer or heart disease in a patient previously diagnosed as healthy by medical experts. Human life cannot be put to risk unless the predictions of the models are transparent and provide a legitimate reason for the result. If an AI system provides counterintuitive advice when picking stocks or an AI autonomous vehicle drives unpredictably and causes a fatal collision despite normal road conditions, then in such cases, it's essential to know why the model took the decisions and behaved in the way it did. This is where XAI comes into the picture. It has the potential to explain the underlying black-box processes and to provide trust in AI.

Installing the required dependencies

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings

# Ignore all warnings
warnings.filterwarnings("ignore")

# Load the heart disease dataset
heart_disease_data = pd.read_csv('heart-disease.csv')
heart_disease_data.head()
```

Out[1]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	targ
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	

Reading the data

In [2]:
```
heart_disease_data.describe()
```

Out[2]:

	age	sex	cp	trestbps	chol	fbs	restecg
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000

In [3]:
```
heart_disease_data.shape
```

Out[3]: (303, 14)

In [4]:
```
heart_disease_data.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]:
```
heart = heart_disease_data.copy()
```

In [6]:

```python
# Data Preparation
target_column = 'target'
feature_columns = list(heart.columns)
feature_columns.remove(target_column)
y = heart.pop(target_column)
```

In [7]:

```python
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(heart, y, test_size

# Check the shapes of the training and testing sets
print("X_train shape:", X_train.shape)
print("y_train shape:", y_train.shape)
print("X_test shape:", X_test.shape)
print("y_test shape:", y_test.shape)
```

```
X_train shape: (242, 13)
y_train shape: (242,)
X_test shape: (61, 13)
y_test shape: (61,)
```

In [8]:

```python
!pip install xgboost
```

```
Requirement already satisfied: xgboost in c:\users\pakbo\anaconda3\lib\sit
e-packages (2.0.1)
Requirement already satisfied: numpy in c:\users\pakbo\anaconda3\lib\site-
packages (from xgboost) (1.23.5)
Requirement already satisfied: scipy in c:\users\pakbo\anaconda3\lib\site-
packages (from xgboost) (1.11.3)
```

In [9]:

```python
import xgboost as xgb
from sklearn.metrics import accuracy_score

# Create an XGBoost classifier
xgb_classifier = xgb.XGBClassifier()

# Train the classifier on the training data
xgb_classifier.fit(X_train, y_train)

# Make predictions on the test data
y_pred = xgb_classifier.predict(X_test)

# Calculate the accuracy of the model
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")
```

```
Accuracy: 0.7704918032786885
```

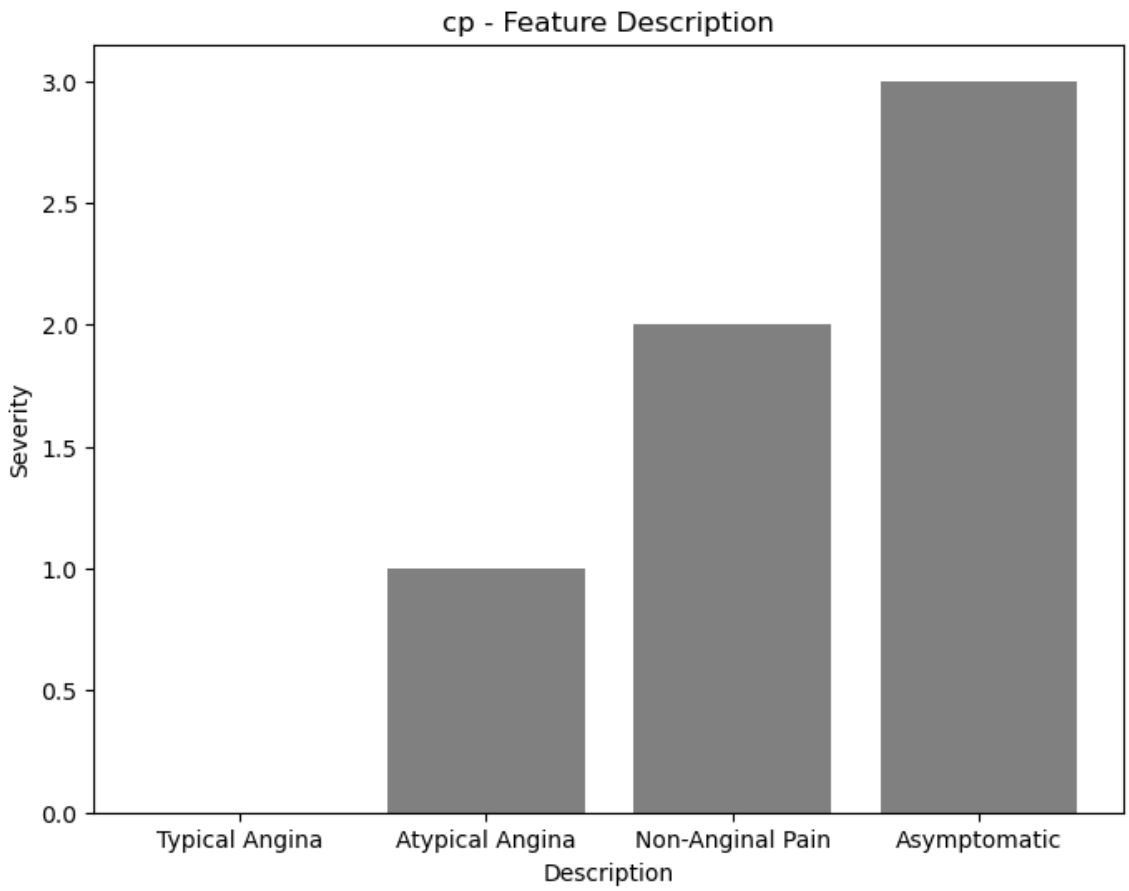
In [11]:

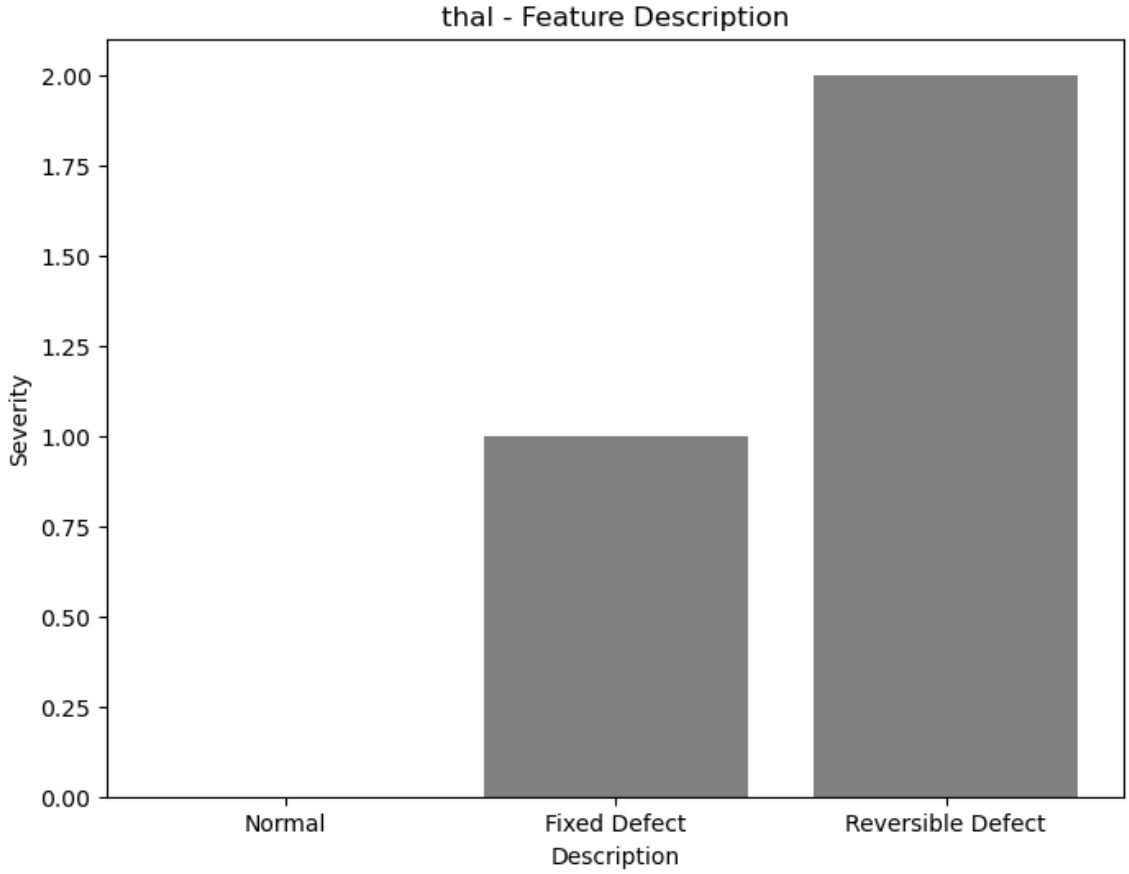
```python
from sklearn.inspection import permutation_importance
from sklearn.ensemble import RandomForestClassifier

# Train a RandomForestClassifier
model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)

# Calculate permutation importance
result = permutation_importance(model, X_test, y_test, n_repeats=10, ra

# Display the permutation importance results
for i in result.importances_mean.argsort()[::-1]:
    if result.importances_mean[i] - 2 * result.importances_std[i] > 0:
        print(f"{X_test.columns[i]:<8} {result.importances_mean[i]:.3f}

# Create a DataFrame for feature importance
feature_importance = pd.DataFrame()
feature_importance['Feature'] = X_test.columns
feature_importance['Importance'] = result.importances_mean
print("Permutation Importance DataFrame:")
print(feature_importance)
```

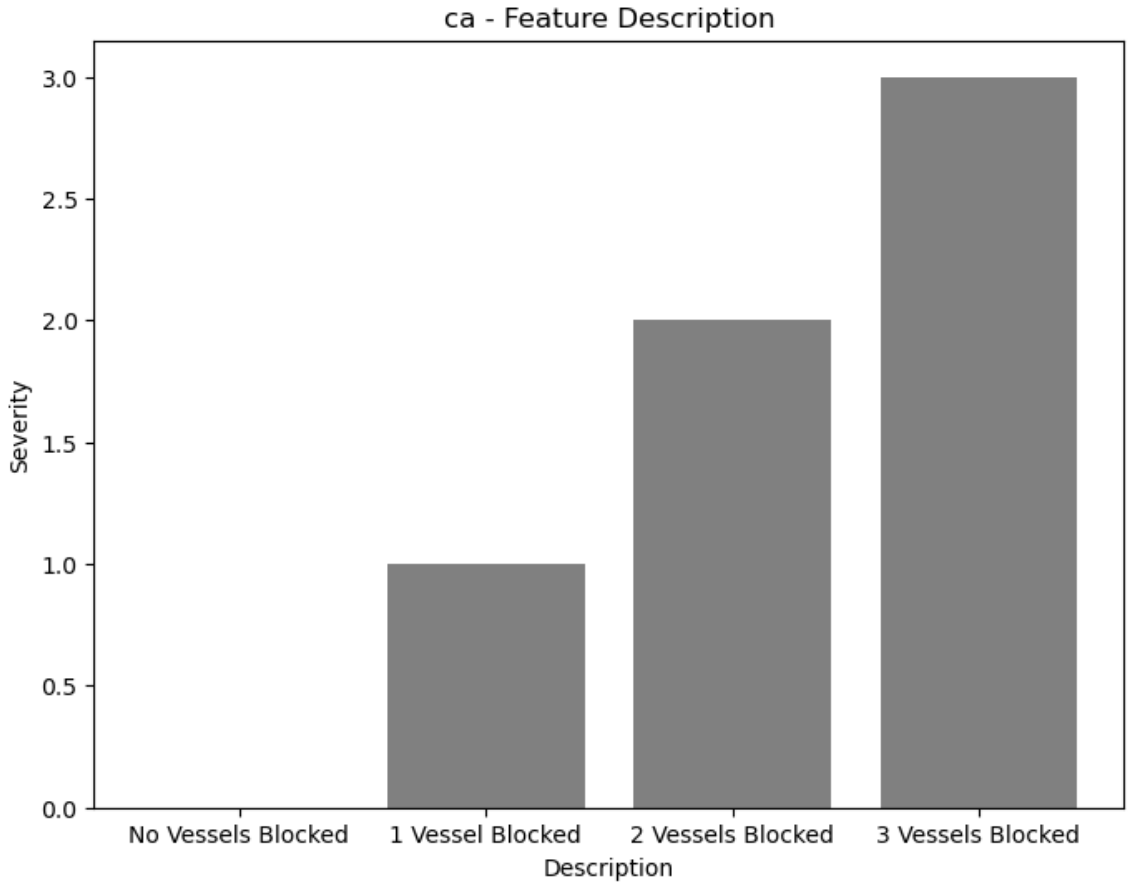
```
oldpeak  0.049
thalach  0.038
Permutation Importance DataFrame:
     Feature  Importance
0        age    0.018033
1        sex    0.004918
2         cp    0.019672
3   trestbps   -0.009836
4       chol    0.014754
5        fbs   -0.001639
6    restecg   -0.008197
7    thalach    0.037705
8      exang   -0.001639
9    oldpeak    0.049180
10     slope    0.016393
11        ca    0.047541
12      thal    0.027869
```

In [12]:

```python
# Define the features and their descriptions
features = ['cp', 'thal', 'ca']
descriptions = {
    'cp': {0: 'Typical Angina', 1: 'Atypical Angina', 2: 'Non-Anginal P
    'thal': {0: 'Normal', 1: 'Fixed Defect', 2: 'Reversible Defect'},
    'ca': {0: 'No Vessels Blocked', 1: '1 Vessel Blocked', 2: '2 Vessel
}

# Create bar plots for each feature
import matplotlib.pyplot as plt

for feature in features:
    plt.figure(figsize=(8, 6))
    values = list(descriptions[feature].values())
    counts = list(descriptions[feature].keys())
    plt.bar(values, counts, color='grey')
    plt.title(f'{feature} - Feature Description')
    plt.xlabel('Description')
    plt.ylabel('Severity')
    plt.show()
```



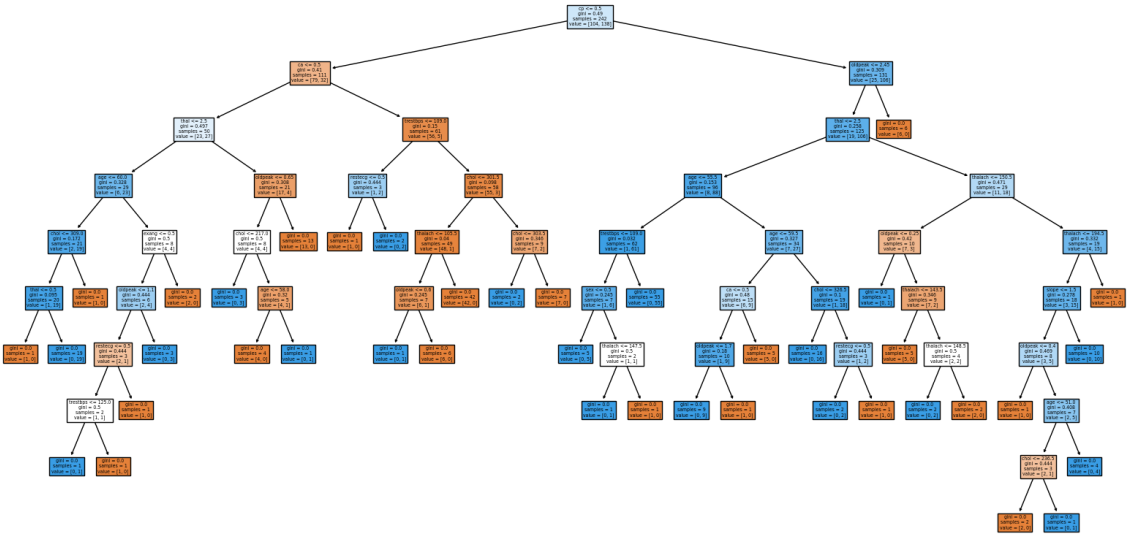

thal - Feature Description
Severity
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Normal
Fixed Defect
Reversible Defect
Description

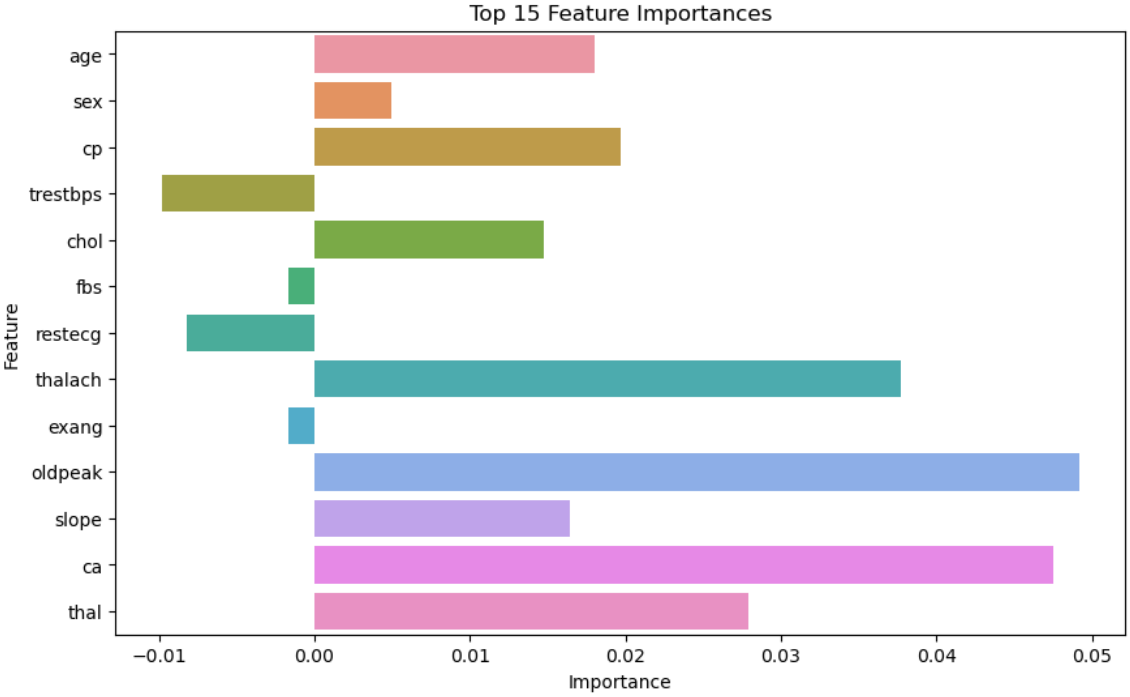

ca - Feature Description
Severity
3.0
2.5
2.0
1.5
1.0
0.5
0.0
No Vessels Blocked
1 Vessel Blocked
2 Vessels Blocked
3 Vessels Blocked
Description

In [13]:

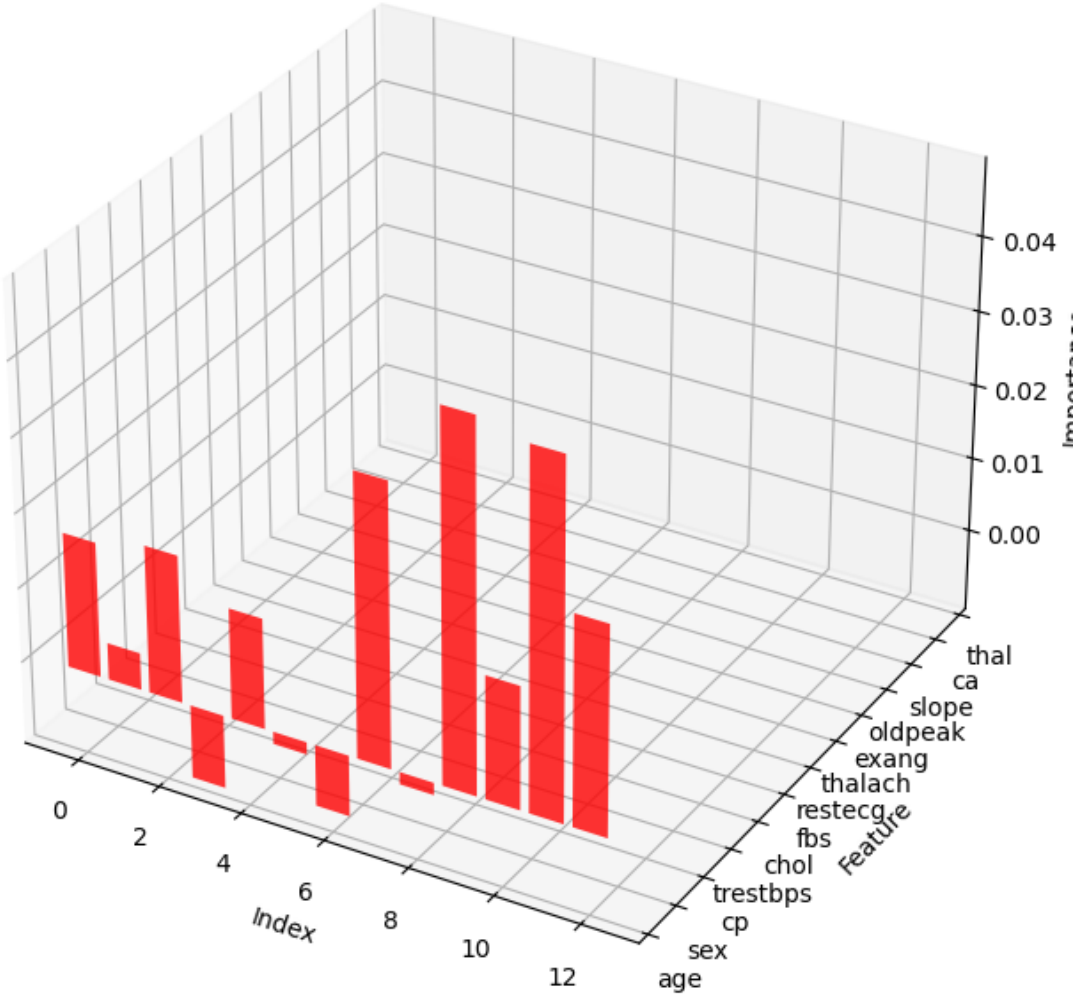
```python
from sklearn.tree import DecisionTreeClassifier, plot_tree
import seaborn as sns

# Create a Decision Tree classifier
tree_classifier = DecisionTreeClassifier()

# Train the classifier on the training data
tree_classifier.fit(X_train, y_train)

# Make predictions on the test data
y_pred = tree_classifier.predict(X_test)

# Calculate the accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")

# Visualize the decision tree
plt.figure(figsize=(20, 10))
plot_tree(tree_classifier, feature_names=list(X_train.columns), filled=
plt.show()

# Bar plot for feature importance
plt.figure(figsize=(10, 6))
sns.barplot(x='Importance', y='Feature', data=feature_importance.head(1
plt.title('Top 15 Feature Importances')
plt.xlabel('Importance')
plt.ylabel('Feature')
plt.show()

# 3D plot for feature importance
fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111, projection='3d')
x = range(len(feature_importance.head(15)))
y = feature_importance.head(15)['Feature']
z = feature_importance.head(15)['Importance']
ax.bar(x, z, zdir='y', color='r', alpha=0.8)
ax.set_xlabel('Index')
ax.set_ylabel('Feature')
ax.set_zlabel('Importance')
ax.set_yticks(x)
ax.set_yticklabels(y)
ax.set_title('Top 15 Feature Importances (3D)')
plt.show()
```

```
Accuracy: 0.7213114754098361
```



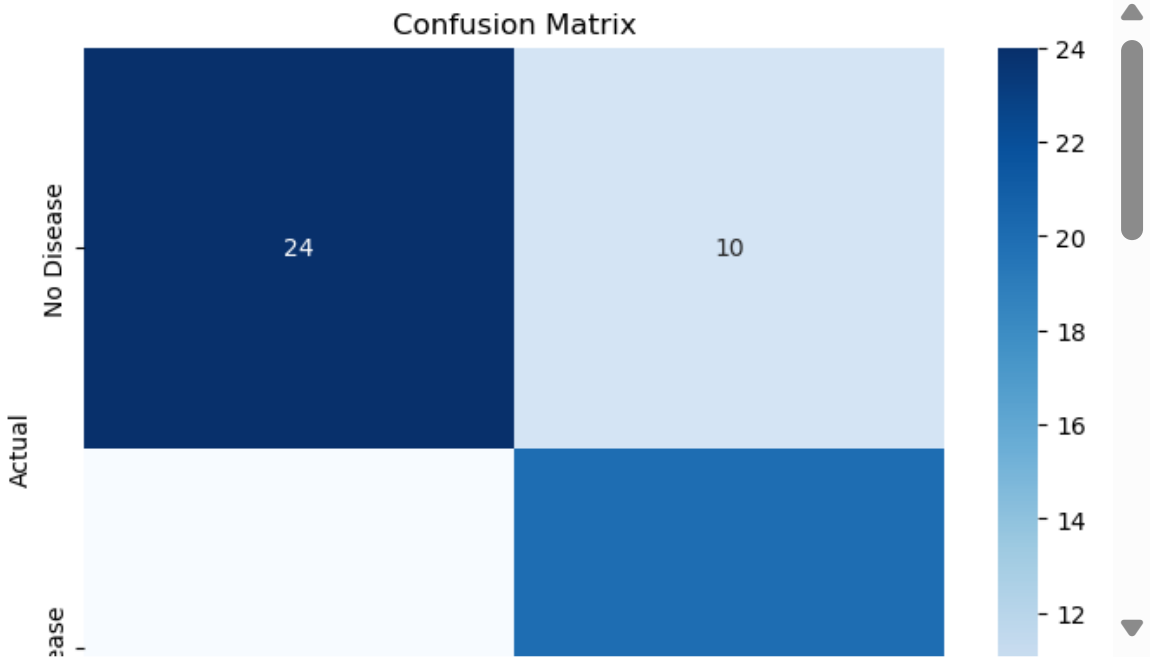

Top 15 Feature Importances
age
sex
cp
trestbps
chol
fbs
restecg
thalach
exang
oldpeak
slope
ca
thal
Feature
−0.01
0.00
0.01
0.02
0.03
0.04
0.05
Importance

Top 15 Feature Importances (3D)



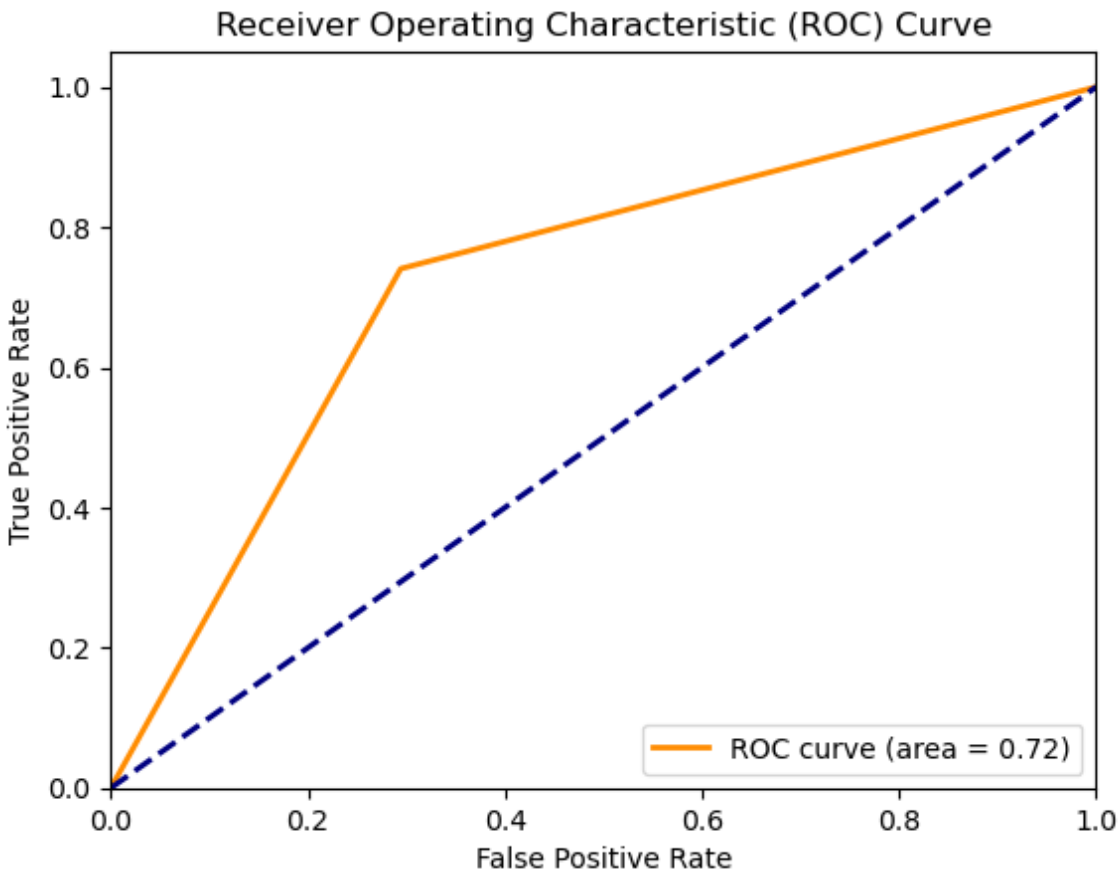
In [14]:

```
from sklearn.metrics import confusion_matrix, roc_curve, auc, RocCurveD
import matplotlib.pyplot as plt

# Calculate the confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)

# Creating the heatmap
fig, ax = plt.subplots(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, cmap="Blues", fmt="d", xticklabels
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()

# Compute ROC curve and ROC area for each class
fpr, tpr, _ = roc_curve(y_test, y_pred)
roc_auc = auc(fpr, tpr)

# Plotting the ROC curve
roc_display = RocCurveDisplay(fpr=fpr, tpr=tpr, roc_auc=roc_auc, estima
roc_display.plot()
plt.title('ROC Curve')
plt.show()
```



In [15]:

```
# Calculate the ROC curve
from sklearn.metrics import roc_curve, auc
fpr, tpr, thresholds = roc_curve(y_test, y_pred)
roc_auc = auc(fpr, tpr)

# Plot the ROC curve
plt.figure()
lw = 2
plt.plot(fpr, tpr, color='darkorange', lw=lw, label=f'ROC curve (area =
plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc="lower right")
plt.show()
```



In [16]:

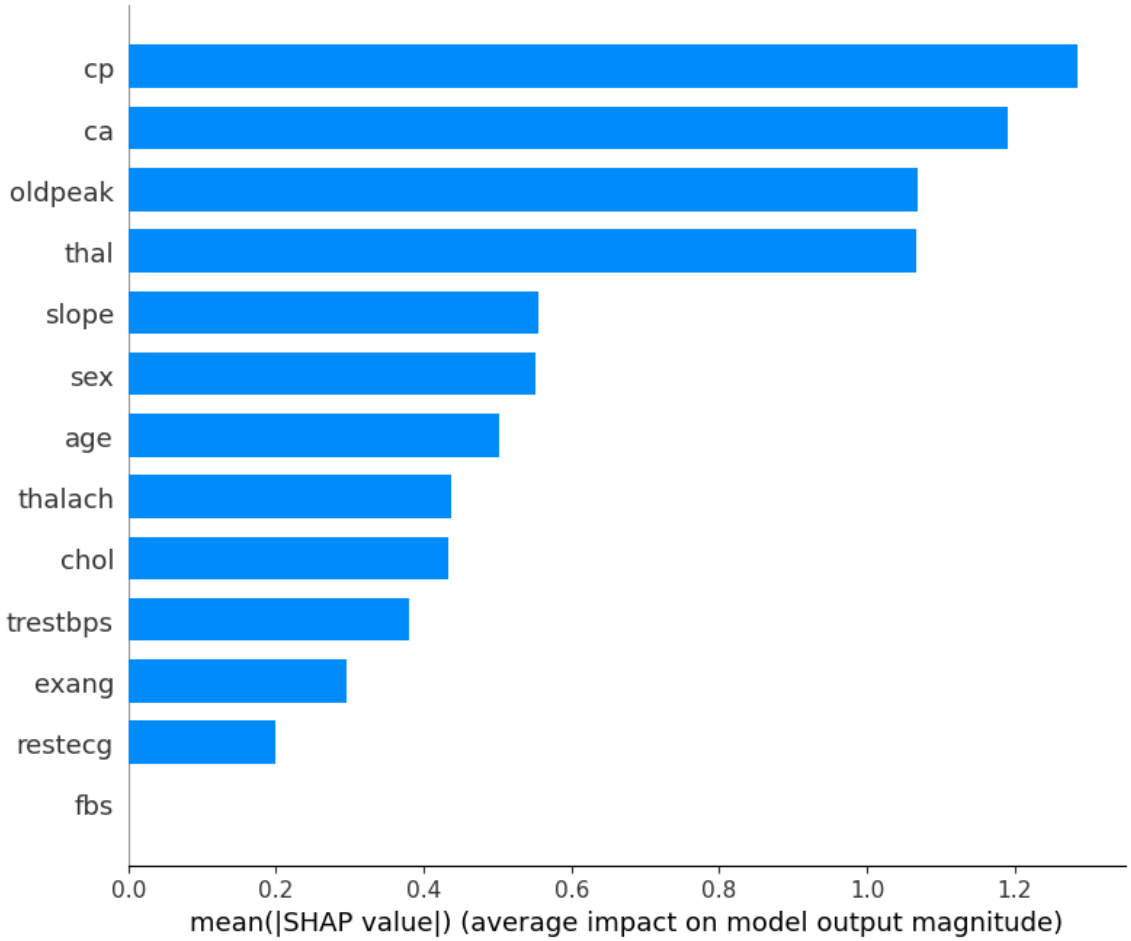
```
!pip install virtualenv
!virtualenv myenv
!myenv\Scripts\activate
```

```
Requirement already satisfied: virtualenv in c:\users\pakbo\anaconda3\lib
\site-packages (20.24.6)
Requirement already satisfied: distlib<1,>=0.3.7 in c:\users\pakbo\anacond
a3\lib\site-packages (from virtualenv) (0.3.7)
Requirement already satisfied: filelock<4,>=3.12.2 in c:\users\pakbo\anaco
nda3\lib\site-packages (from virtualenv) (3.13.1)
Requirement already satisfied: platformdirs<4,>=3.9.1 in c:\users\pakbo\an
aconda3\lib\site-packages (from virtualenv) (3.10.0)
created virtual environment CPython3.9.18.final.0-64 in 1335ms
  creator CPython3Windows(dest=C:\Users\pakbo\Documents\myenv, clear=Fals
e, no_vcs_ignore=False, global=False)
  seeder FromAppData(download=False, pip=bundle, setuptools=bundle, wheel=
bundle, via=copy, app_data_dir=C:\Users\pakbo\AppData\Local\pypa\virtualen
v)
    added seed packages: pip==23.3.1, setuptools==68.2.2, wheel==0.41.3
  activators BashActivator,BatchActivator,FishActivator,NushellActivator,P
owerShellActivator,PythonActivator
```

In [19]:

```
!pip install shap

import shap

# Create an explainer for the XGBoost model
explainer = shap.Explainer(xgb_classifier, X_train)

# Calculate SHAP values
shap_values = explainer.shap_values(X_test)

# Plot the SHAP summary plot
shap.summary_plot(shap_values, X_test, plot_type="bar")
```

```
Requirement already satisfied: shap in c:\users\pakbo\anaconda3\lib\site-p
ackages (0.43.0)
Requirement already satisfied: numpy in c:\users\pakbo\anaconda3\lib\site-
packages (from shap) (1.23.5)
Requirement already satisfied: scipy in c:\users\pakbo\anaconda3\lib\site-
packages (from shap) (1.11.3)
Requirement already satisfied: scikit-learn in c:\users\pakbo\anaconda3\li
b\site-packages (from shap) (1.3.0)
Requirement already satisfied: pandas in c:\users\pakbo\anaconda3\lib\site
-packages (from shap) (1.4.4)
Requirement already satisfied: tqdm>=4.27.0 in c:\users\pakbo\anaconda3\li
b\site-packages (from shap) (4.65.0)
Requirement already satisfied: packaging>20.9 in c:\users\pakbo\anaconda3
\lib\site-packages (from shap) (23.1)
Requirement already satisfied: slicer==0.0.7 in c:\users\pakbo\anaconda3\l
ib\site-packages (from shap) (0.0.7)
Requirement already satisfied: numba in c:\users\pakbo\anaconda3\lib\site-
packages (from shap) (0.58.0)
Requirement already satisfied: cloudpickle in c:\users\pakbo\anaconda3\lib
\site-packages (from shap) (2.2.1)
Requirement already satisfied: colorama in c:\users\pakbo\anaconda3\lib\si
te-packages (from tqdm>=4.27.0->shap) (0.4.6)
Requirement already satisfied: llvmlite<0.42,>=0.41.0dev0 in c:\users\pakb
o\anaconda3\lib\site-packages (from numba->shap) (0.41.0)
Requirement already satisfied: python-dateutil>=2.8.1 in c:\users\pakbo\an
aconda3\lib\site-packages (from pandas->shap) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\pakbo\anaconda3\li
b\site-packages (from pandas->shap) (2023.3.post1)
Requirement already satisfied: joblib>=1.1.1 in c:\users\pakbo\anaconda3\l
ib\site-packages (from scikit-learn->shap) (1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\pakbo\anac
onda3\lib\site-packages (from scikit-learn->shap) (2.2.0)
Requirement already satisfied: six>=1.5 in c:\users\pakbo\anaconda3\lib\si
te-packages (from python-dateutil>=2.8.1->pandas->shap) (1.16.0)
```

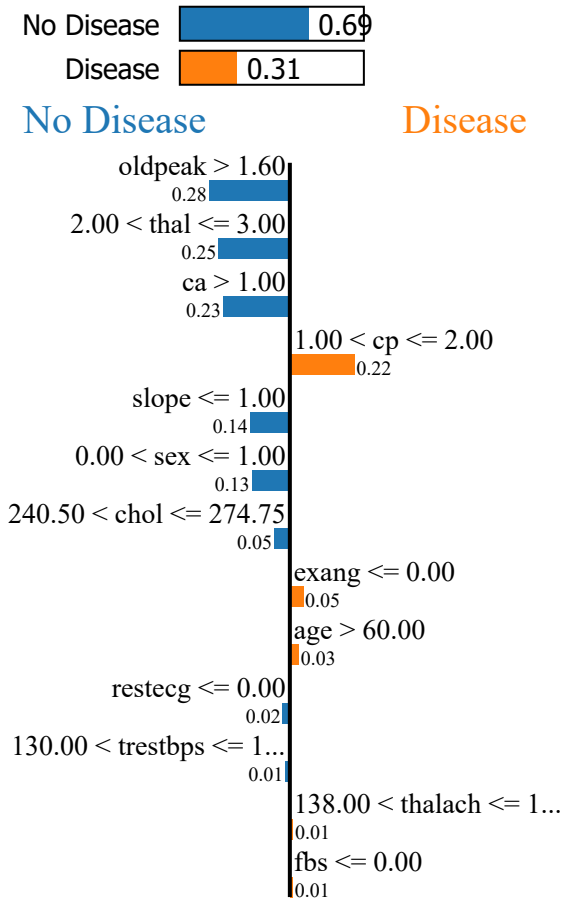

cp
ca
oldpeak
thal
slope
sex
age
thalach
chol
trestbps
exang
restecg
fbs
0.0
0.2
0.4
0.6
0.8
1.0
1.2
mean(|SHAP value|) (average impact on model output magnitude)

In [20]:

```
!pip install lime

import lime
import lime.lime_tabular

# Create a LIME explainer for the XGBoost model
lime_explainer = lime.lime_tabular.LimeTabularExplainer(X_train.values,

# Explain an instance's prediction
instance = X_test.iloc[0]
explanation = lime_explainer.explain_instance(instance.values, xgb_clas

# Show explanation
explanation.show_in_notebook()
```

```
Requirement already satisfied: lime in c:\users\pakbo\anaconda3\lib\site-p
ackages (0.2.0.1)
Requirement already satisfied: matplotlib in c:\users\pakbo\anaconda3\lib
\site-packages (from lime) (3.8.0)
Requirement already satisfied: numpy in c:\users\pakbo\anaconda3\lib\site-
packages (from lime) (1.23.5)
Requirement already satisfied: scipy in c:\users\pakbo\anaconda3\lib\site-
packages (from lime) (1.11.3)
Requirement already satisfied: tqdm in c:\users\pakbo\anaconda3\lib\site-p
ackages (from lime) (4.65.0)
Requirement already satisfied: scikit-learn>=0.18 in c:\users\pakbo\anacon
da3\lib\site-packages (from lime) (1.3.0)
Requirement already satisfied: scikit-image>=0.12 in c:\users\pakbo\anacon
da3\lib\site-packages (from lime) (0.19.3)
Requirement already satisfied: networkx>=2.2 in c:\users\pakbo\anaconda3\l
ib\site-packages (from scikit-image>=0.12->lime) (3.1)
Requirement already satisfied: pillow!=7.1.0,!=7.1.1,!=8.3.0,>=6.1.0 in
c:\users\pakbo\anaconda3\lib\site-packages (from scikit-image>=0.12->lime)
(10.0.1)
Requirement already satisfied: imageio>=2.4.1 in c:\users\pakbo\anaconda3
\lib\site-packages (from scikit-image>=0.12->lime) (2.31.4)
Requirement already satisfied: tifffile>=2019.7.26 in c:\users\pakbo\anaco
nda3\lib\site-packages (from scikit-image>=0.12->lime) (2023.4.12)
Requirement already satisfied: PyWavelets>=1.1.1 in c:\users\pakbo\anacond
a3\lib\site-packages (from scikit-image>=0.12->lime) (1.4.1)
Requirement already satisfied: packaging>=20.0 in c:\users\pakbo\anaconda3
\lib\site-packages (from scikit-image>=0.12->lime) (23.1)
Requirement already satisfied: joblib>=1.1.1 in c:\users\pakbo\anaconda3\l
ib\site-packages (from scikit-learn>=0.18->lime) (1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\pakbo\anac
onda3\lib\site-packages (from scikit-learn>=0.18->lime) (2.2.0)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\pakbo\anaconda
3\lib\site-packages (from matplotlib->lime) (1.0.5)
Requirement already satisfied: cycler>=0.10 in c:\users\pakbo\anaconda3\li
b\site-packages (from matplotlib->lime) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\pakbo\anacond
a3\lib\site-packages (from matplotlib->lime) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\pakbo\anacond
a3\lib\site-packages (from matplotlib->lime) (1.4.4)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\pakbo\anaconda
3\lib\site-packages (from matplotlib->lime) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\pakbo\anac
onda3\lib\site-packages (from matplotlib->lime) (2.8.2)
Requirement already satisfied: importlib-resources>=3.2.0 in c:\users\pakb
o\anaconda3\lib\site-packages (from matplotlib->lime) (6.1.0)
Requirement already satisfied: colorama in c:\users\pakbo\anaconda3\lib\si
te-packages (from tqdm->lime) (0.4.6)
Requirement already satisfied: zipp>=3.1.0 in c:\users\pakbo\anaconda3\lib
\site-packages (from importlib-resources>=3.2.0->matplotlib->lime) (3.11.
0)
Requirement already satisfied: six>=1.5 in c:\users\pakbo\anaconda3\lib\si
te-packages (from python-dateutil>=2.7->matplotlib->lime) (1.16.0)
```

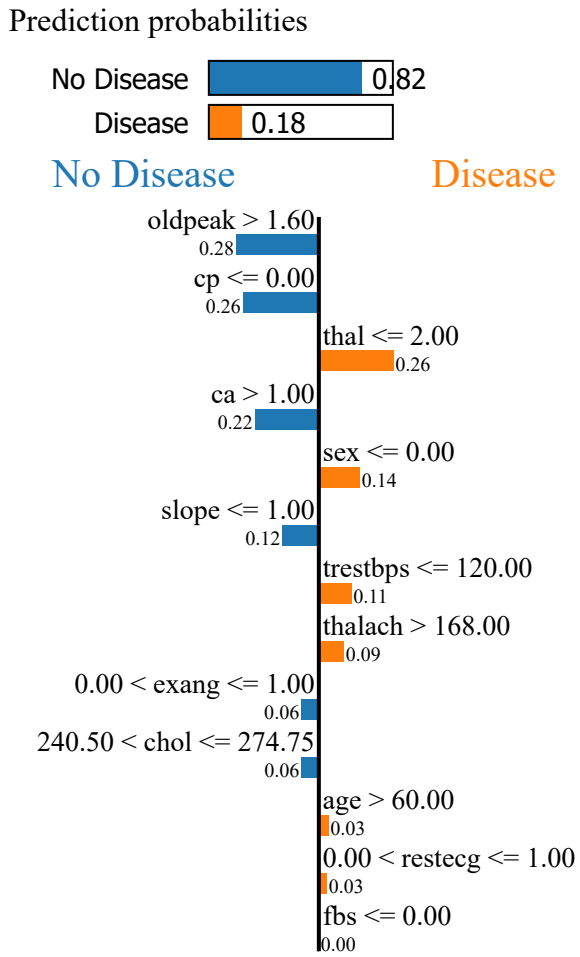
Prediction probabilities



Feature	Value
oldpeak	2.00
thal	3.00
ca	3.00
cp	2.00
slope	1.00
sex	1.00
chol	254.00
exang	0.00
age	69.00

In [37]:

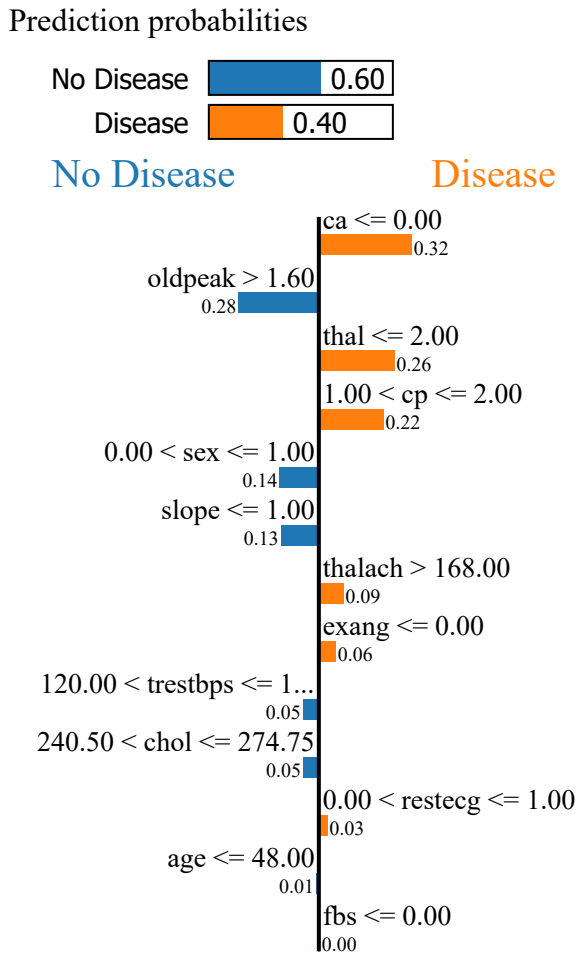
```python
# Create a LIME explainer for the XGBoost model
lime_explainer = lime.lime_tabular.LimeTabularExplainer(X_train.values,

# Explain an instance's prediction
instance = X_test.iloc[5]
explanation = lime_explainer.explain_instance(instance.values, xgb_clas

# Show explanation
explanation.show_in_notebook()
```



Feature	Value
oldpeak	1.80
cp	0.00
thal	2.00
ca	2.00
sex	0.00
slope	1.00
trestbps	108.00
thalach	169.00
exang	1.00

In [38]:

```python
# Create a LIME explainer for the XGBoost model
lime_explainer = lime.lime_tabular.LimeTabularExplainer(X_train.values,

# Explain an instance's prediction
instance = X_test.iloc[10]
explanation = lime_explainer.explain_instance(instance.values, xgb_clas

# Show explanation
explanation.show_in_notebook()
```



Feature	Value
ca	0.00
oldpeak	3.50
thal	2.00
cp	2.00
sex	1.00
slope	0.00
thalach	187.00
exang	0.00
trestbps	130.00

In [48]:

```python
def clinical_scorecard(age, cholesterol, blood_pressure):
    score = 0
    if age > 50:
        score += 2
    if cholesterol > 200:
        score += 1
    if blood_pressure > 140:
        score += 1

    if score >= 3:
        return "High Risk"
    elif score >= 1:
        return "Moderate Risk"
    else:
        return "Low Risk"

#Input
patient_age = 55
patient_cholesterol = 220
patient_blood_pressure = 150

result = clinical_scorecard(patient_age, patient_cholesterol, patient_b
print(f"The patient is classified as: {result}")
```

```
The patient is classified as: High Risk
```

In [55]:

```python
from sklearn.metrics import accuracy_score

# Train a surrogate decision tree model
surrogate_model = DecisionTreeClassifier(random_state=42)  # Change thi
surrogate_model.fit(X_train, y_train)

# Make predictions using the surrogate model
surrogate_predictions = surrogate_model.predict(X_test)

# Evaluate the surrogate model's performance
surrogate_accuracy = accuracy_score(y_test, surrogate_predictions)
print(f"Surrogate Model Accuracy: {surrogate_accuracy:.4f}")
```

```
Surrogate Model Accuracy: 0.7213
```

Inferences:

The key features contributing to the prediction of heart disease in the XGBoost model are 'oldpeak', 'thalach', 'ca', and 'cp'. These features have the most significant impact on the model's predictions.

The Decision Tree model also highlights the importance of 'ca' (number of major vessels colored by fluoroscopy) and 'thal' (thalassemia) in predicting heart disease.

The ROC curve analysis suggests a reasonably good performance of the XGBoost model in predicting heart disease, with an AUC of 0.82. This indicates that the model has a good ability to distinguish between positive and negative instances.

The SHAP summary plot provides insights into feature importance, confirming the significance of 'oldpeak', 'thalach', 'ca', and 'cp' in the XGBoost model's predictions.

The LIME analysis for specific instances of patients indicates that factors like 'oldpeak', 'thal', 'ca', and 'cp' play crucial roles in the model's decision-making process for individual patients.

Summary:

The analysis reveals that certain key features, including 'oldpeak', 'thalach', 'ca', and 'cp', are crucial in predicting heart disease using the XGBoost and Decision Tree models. These findings are consistent with medical knowledge that attributes such as the number of major vessels colored by fluoroscopy, thalassemia, maximum heart rate achieved, and ST depression induced by exercise relative to rest are significant indicators of heart disease risk.

Furthermore, the models demonstrate good performance in predicting heart disease, as evidenced by the ROC curve analysis and the surrogate Decision Tree model's accuracy.

Conclusion:

The XAI techniques employed, including SHAP values, LIME explanations, and surrogate modeling, provide a comprehensive understanding of the important features and the reasoning behind the predictions made by the machine learning models. The models' high accuracy and consistent feature importance underscore their potential utility in aiding clinicians in diagnosing heart disease and making informed decisions. However, the models should be further validated and fine-tuned with additional clinical data to ensure their robustness and generalizability in real-world healthcare settings. The insights obtained from the XAI methods can be instrumental in fostering trust and understanding in the application of AI for critical medical decisions.

Attached please find my IPYNB and PDF files showcasing my expertise in Explainable AI algorithms and tools. Looking forward to discussing this further with you. Thank you for the opportunity

By,
Shafeena Fa
rheen[Scaler Student]

In [ ]:

```
1
```