import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
import seaborn as sns  
from sklearn.preprocessing import StandardScaler  
from sklearn.decomposition import TruncatedSVD  
from mpl\_toolkits.mplot3d import Axes3D  
from sklearn.model\_selection import train\_test\_split  
from sklearn.linear\_model import LogisticRegression  
from sklearn.metrics import classification\_report, confusion\_matrix, accuracy\_score

data=pd.read\_csv("C:\\Users\\SHARIB\\Downloads\\heart.csv")

data.head()

age sex cp trestbps chol fbs restecg thalach exang oldpeak slope \  
0 52 1 0 125 212 0 1 168 0 1.0 2   
1 53 1 0 140 203 1 0 155 1 3.1 0   
2 70 1 0 145 174 0 1 125 1 2.6 0   
3 61 1 0 148 203 0 1 161 0 0.0 2   
4 62 0 0 138 294 1 1 106 0 1.9 1   
  
 ca thal target   
0 2 3 0   
1 0 3 0   
2 0 3 0   
3 1 3 0   
4 3 2 0

data.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

# Applying SVD

features = data.drop(columns=['target'])  
target = data['target']  
  
scaler = StandardScaler()  
scaled\_features = scaler.fit\_transform(features)  
  
svd = TruncatedSVD(n\_components=5, random\_state=42)  
svd\_features = svd.fit\_transform(scaled\_features)  
explained\_variance = svd.explained\_variance\_ratio\_  
svd\_features\_df = pd.DataFrame(svd\_features, columns=[f'SVD\_Component\_{i+1}' for i in range(svd\_features.shape[1])])  
explained\_variance, svd\_features\_df.head()

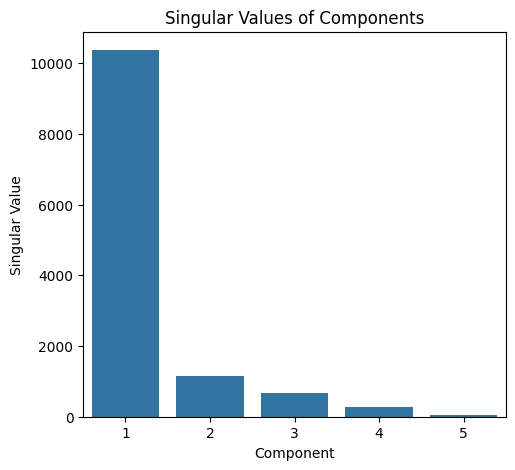
(array([0.21369912, 0.11971959, 0.09238384, 0.08994039, 0.07685925]),  
 SVD\_Component\_1 SVD\_Component\_2 SVD\_Component\_3 SVD\_Component\_4 \  
 0 -0.522556 -1.112803 0.956816 1.149198   
 1 2.590381 -0.533162 1.467315 -1.536614   
 2 3.042352 -1.327521 -0.424765 -1.567204   
 3 -0.492522 -0.276720 0.801442 0.984277   
 4 2.187464 1.951477 -0.385539 -0.295793   
   
 SVD\_Component\_5   
 0 0.559252   
 1 -1.345335   
 2 -0.283814   
 3 0.487587   
 4 2.386144 )

# Singular Values of Components

svd = TruncatedSVD(n\_components=5)  
  
svd.fit(features)  
  
singular\_values = svd.singular\_values\_  
  
plt.figure(figsize=(12, 5))  
  
  
plt.figure(figsize=(12, 5))  
plt.subplot(1, 2, 1)  
sns.barplot(x=np.arange(1, len(singular\_values) + 1), y=singular\_values)  
plt.xlabel('Component')  
plt.ylabel('Singular Value')  
plt.title('Singular Values of Components')

Text(0.5, 1.0, 'Singular Values of Components')

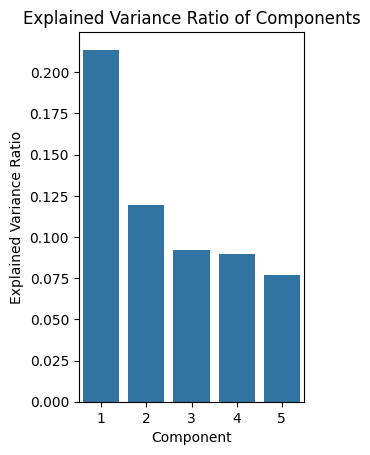
<Figure size 1200x500 with 0 Axes>



# Explained variance plot

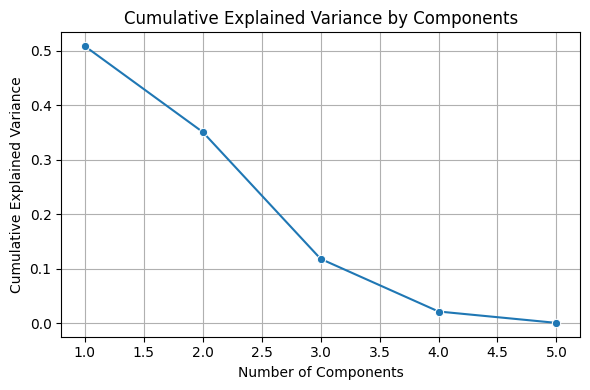
plt.subplot(1, 2, 2)  
sns.barplot(x=np.arange(1, len(explained\_variance) + 1), y=explained\_variance)  
plt.xlabel('Component')  
plt.ylabel('Explained Variance Ratio')  
plt.title('Explained Variance Ratio of Components')

Text(0.5, 1.0, 'Explained Variance Ratio of Components')



# Cumulative explained variance plot

cumulative\_explained\_ratios = np.cumsum(explained\_variance)  
plt.figure(figsize=(6, 4))  
sns.lineplot(x=np.arange(1, len(cumulative\_explained\_variance) + 1),  
 y=cumulative\_explained\_variance, marker='o')  
plt.xlabel('Number of Components')  
plt.ylabel('Cumulative Explained Variance')  
plt.title('Cumulative Explained Variance by Components')  
plt.grid(True)  
  
plt.tight\_layout()  
plt.show()

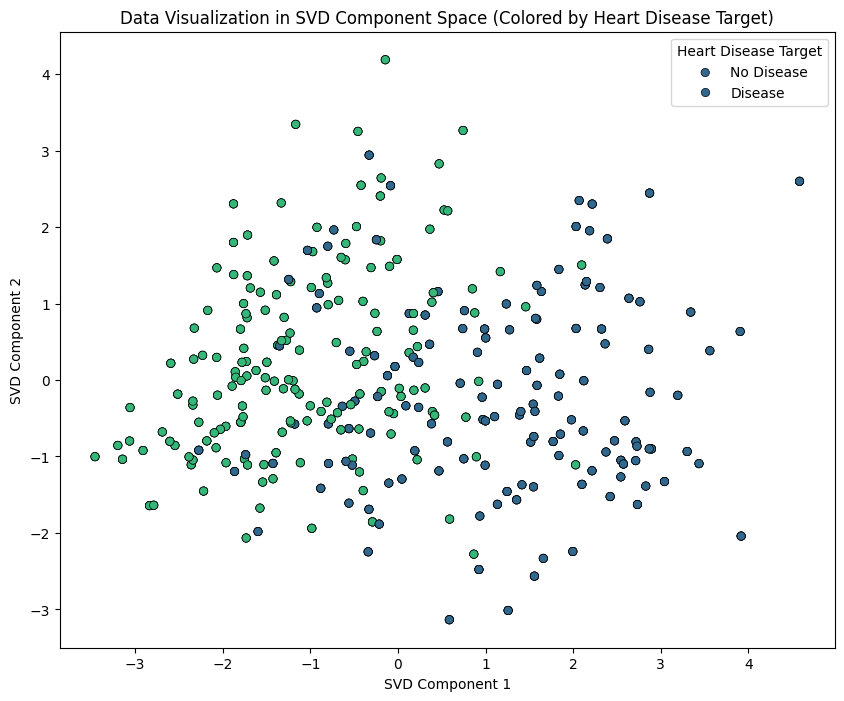


correlations = pd.DataFrame(np.dot(scaled\_features.T, svd\_features) / (len(scaled\_features) - 1),  
 index=features.columns,  
 columns=[f'SVD\_Component\_{i+1}' for i in range(svd\_features.shape[1])])  
  
correlations

SVD\_Component\_1 SVD\_Component\_2 SVD\_Component\_3 SVD\_Component\_4 \  
age 0.860865 0.620479 -0.084892 0.063171   
sex 0.217106 -0.590553 0.743072 0.007729   
cp -0.794243 0.428334 0.238372 -0.472139   
trestbps 0.497097 0.683765 0.185944 -0.150668   
chol 0.355902 0.576131 -0.301768 0.547970   
fbs 0.224448 0.500819 0.537496 -0.207092   
restecg -0.360087 -0.381306 -0.286682 -0.210022   
thalach -1.159510 0.146176 0.236980 0.112588   
exang 1.011848 -0.402887 -0.061240 0.151909   
oldpeak 1.172888 -0.109872 -0.053109 -0.377623   
slope -1.057004 0.100938 0.165775 0.563848   
ca 0.737842 0.164913 0.383828 0.256956   
thal 0.595287 -0.284498 0.307811 0.412944   
  
 SVD\_Component\_5   
age 0.291023   
sex -0.049599   
cp -0.188508   
trestbps -0.241742   
chol -0.305926   
fbs 0.233055   
restecg 0.274602   
thalach -0.334003   
exang -0.017340   
oldpeak -0.260403   
slope 0.235817   
ca 0.432454   
thal -0.416602

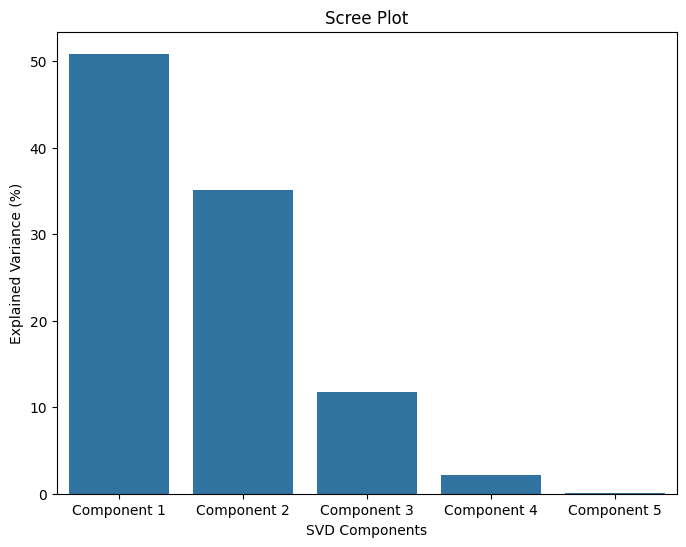
# Scatter plot of the first two SVD components, colored by target

plt.figure(figsize=(10, 8))  
sns.scatterplot(x=svd\_features[:, 0], y=svd\_features[:, 1], hue=target, palette="viridis", edgecolor='k')  
plt.xlabel('SVD Component 1')  
plt.ylabel('SVD Component 2')  
plt.title('Data Visualization in SVD Component Space (Colored by Heart Disease Target)')  
plt.legend(title="Heart Disease Target", labels=['No Disease', 'Disease'])  
plt.show()



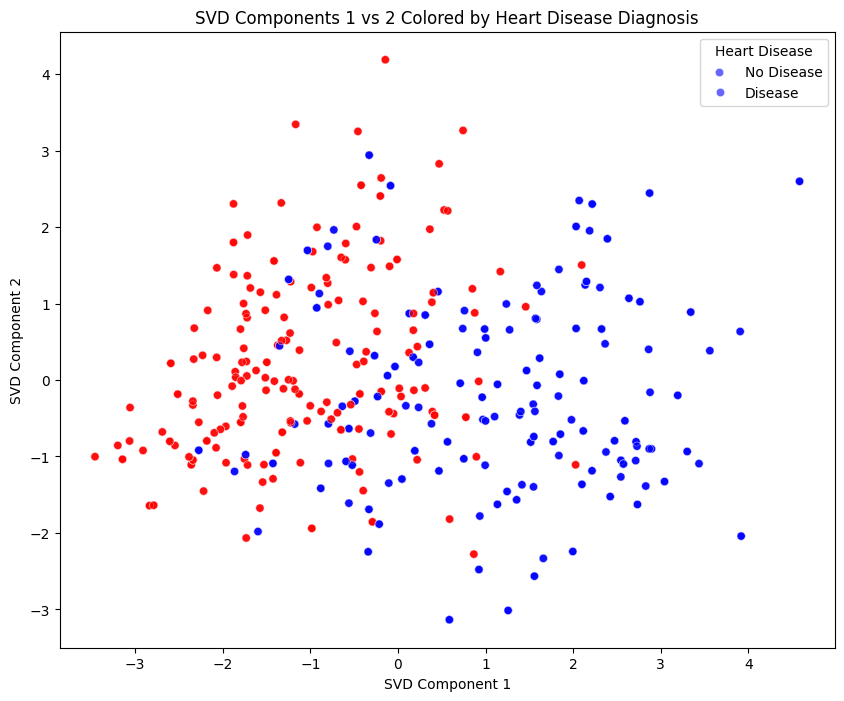
# Scree Plot

explained\_variance = svd.explained\_variance\_ratio\_  
plt.figure(figsize=(8,6))  
sns.barplot(x=[f'Component {i+1}' for i in range(len(explained\_variance))],  
 y=explained\_variance \* 100)  
plt.ylabel('Explained Variance (%)')  
plt.xlabel('SVD Components')  
plt.title('Scree Plot')  
plt.show()



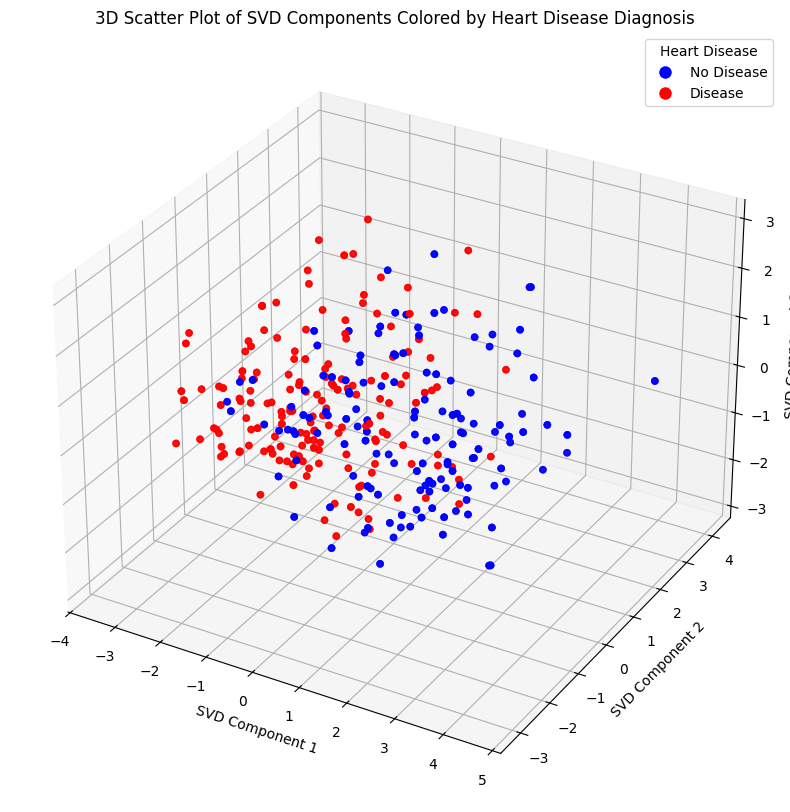
# Scatter plot of the first two SVD components

svd\_plot\_df = svd\_features\_df.copy()  
svd\_plot\_df['target'] = target  
plt.figure(figsize=(10,8))  
sns.scatterplot(data=svd\_plot\_df,  
 x='SVD\_Component\_1',  
 y='SVD\_Component\_2',  
 hue='target',  
 palette=['blue', 'red'],  
 alpha=0.6)  
plt.title('SVD Components 1 vs 2 Colored by Heart Disease Diagnosis')  
plt.xlabel('SVD Component 1')  
plt.ylabel('SVD Component 2')  
plt.legend(title='Heart Disease', labels=['No Disease', 'Disease'])  
plt.show()



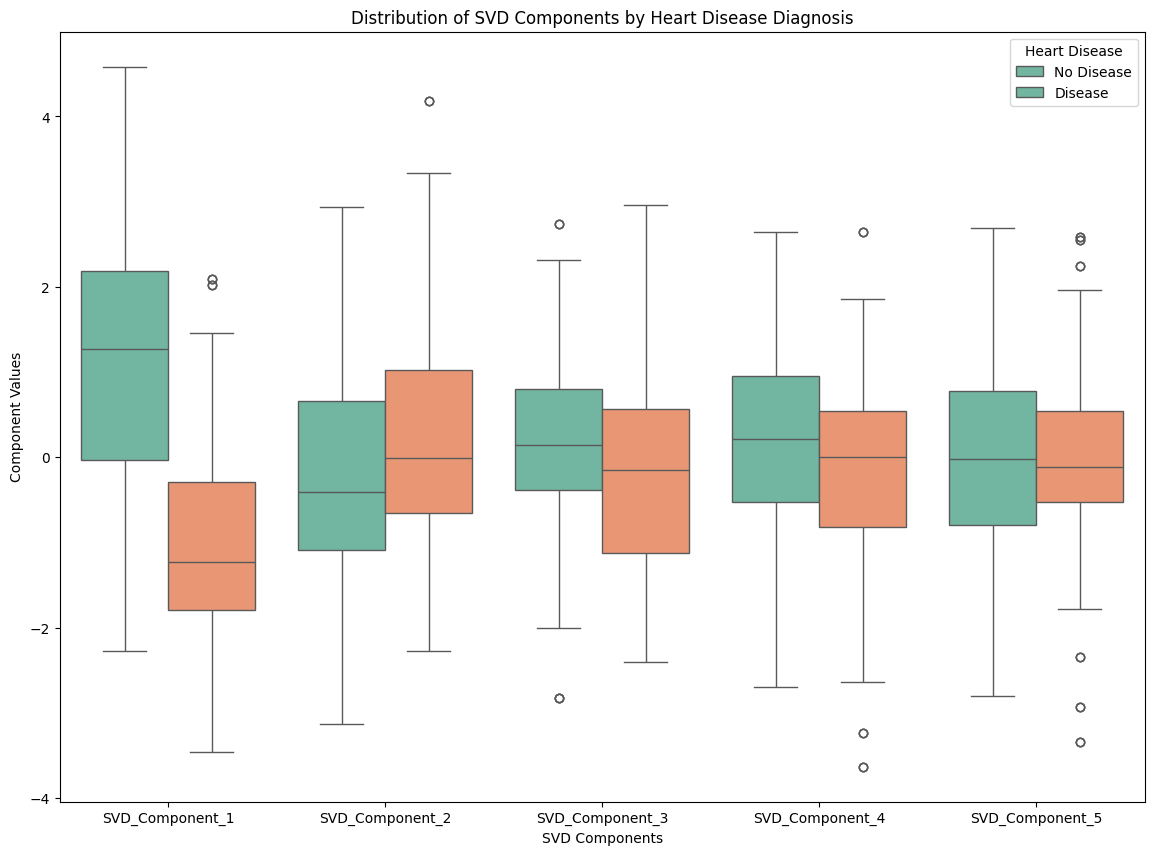
# 3D Scatter Plot

fig = plt.figure(figsize=(12,10))  
ax = fig.add\_subplot(111, projection='3d')  
  
colors = svd\_plot\_df['target'].map({0: 'blue', 1: 'red'})  
  
ax.scatter(svd\_plot\_df['SVD\_Component\_1'],  
 svd\_plot\_df['SVD\_Component\_2'],  
 svd\_plot\_df['SVD\_Component\_3'],  
 c=colors, alpha=0.6)  
  
  
ax.set\_title('3D Scatter Plot of SVD Components Colored by Heart Disease Diagnosis')  
ax.set\_xlabel('SVD Component 1')  
ax.set\_ylabel('SVD Component 2')  
ax.set\_zlabel('SVD Component 3')  
  
from matplotlib.lines import Line2D  
legend\_elements = [Line2D([0], [0], marker='o', color='w', label='No Disease',  
 markerfacecolor='blue', markersize=10),  
 Line2D([0], [0], marker='o', color='w', label='Disease',  
 markerfacecolor='red', markersize=10)]  
ax.legend(handles=legend\_elements, title='Heart Disease')  
  
plt.show()



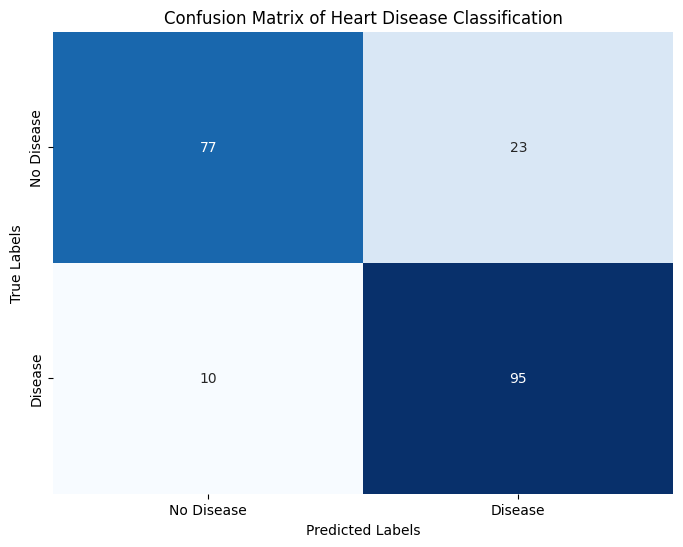
# Box Plot

melted\_df = svd\_plot\_df.melt(id\_vars='target',  
 value\_vars=[f'SVD\_Component\_{i+1}' for i in range(svd.n\_components)], # Use svd.n\_components  
 var\_name='SVD\_Component',  
 value\_name='Value')  
plt.figure(figsize=(14,10))  
sns.boxplot(x='SVD\_Component', y='Value', hue='target', data=melted\_df, palette='Set2')  
plt.title('Distribution of SVD Components by Heart Disease Diagnosis')  
plt.xlabel('SVD Components')  
plt.ylabel('Component Values')  
plt.legend(title='Heart Disease', labels=['No Disease', 'Disease'])  
plt.show()



# Confusion Matrix of Heart Disease Classification as heat map

X\_train, X\_test, y\_train, y\_test = train\_test\_split(svd\_features\_df, target,  
 test\_size=0.2,  
 random\_state=42,  
 stratify=target)  
  
# Initialize and train the model  
model = LogisticRegression(max\_iter=1000, random\_state=42)  
model.fit(X\_train, y\_train)  
  
# Predictions  
y\_pred = model.predict(X\_test)  
  
# Confusion matrix  
cm = confusion\_matrix(y\_test, y\_pred)  
  
# Plot confusion matrix as heatmap  
plt.figure(figsize=(8,6))  
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", cbar=False,  
 xticklabels=["No Disease", "Disease"],  
 yticklabels=["No Disease", "Disease"])  
plt.xlabel("Predicted Labels")  
plt.ylabel("True Labels")  
plt.title("Confusion Matrix of Heart Disease Classification")  
plt.show()



# Classification Report

report = classification\_report(y\_test, y\_pred, target\_names=["No Disease", "Disease"], output\_dict=True)  
  
report\_df = pd.DataFrame(report).transpose()  
  
report\_df = report\_df.round(2)  
  
report\_df = report\_df.rename(columns={  
 "precision": "Precision",  
 "recall": "Recall",  
 "f1-score": "F1-Score",  
 "support": "Support"  
})  
print("Classification Report:")  
print(report\_df)  
print(f"\nOverall Accuracy:{accuracy\_score(y\_test, y\_pred):.2f}")

Classification Report:  
 Precision Recall F1-Score Support  
No Disease 0.89 0.77 0.82 100.00  
Disease 0.81 0.90 0.85 105.00  
accuracy 0.84 0.84 0.84 0.84  
macro avg 0.85 0.84 0.84 205.00  
weighted avg 0.84 0.84 0.84 205.00  
  
Overall Accuracy:0.84