Great! You've uploaded a file named `heart (1).csv`. Let's start by loading the data and taking a quick look at it. After that, we can apply SVM, Random Forest, and KNN, and compute the F1 score, ROC curve, and accuracy for each model.

Let's begin.

The dataset appears to contain information about patients, including their medical attributes and a target variable indicating the presence or absence of heart disease.

Next, we'll proceed with the following steps:

1. Preprocess the data (if necessary).

2. Split the data into training and testing sets.

3. Train and evaluate models using SVM, Random Forest, and KNN.

4. Calculate the F1 score, ROC curve, and accuracy for each model.

Let's start with preprocessing and splitting the data.

The data has been split into training and testing sets, with 242 samples for training and 61 samples for testing. The features have also been standardized.

Now, let's train the models (SVM, Random Forest, and KNN) and evaluate their performance by calculating the F1 score, ROC curve, and accuracy for each model.

Here are the performance metrics for each model:

\*\*Accuracy:\*\*

- SVM: 86.89%

- Random Forest: 83.61%

- KNN: 90.16%

\*\*F1 Score:\*\*

- SVM: 0.871

- Random Forest: 0.844

- KNN: 0.903

The ROC curves for each model have also been plotted, showing the trade-off between the true positive rate and the false positive rate.

Is there anything else you would like to explore or analyze further?

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.svm import SVC

from sklearn.ensemble import RandomForestClassifier

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import accuracy\_score, f1\_score, roc\_curve, auc, RocCurveDisplay

import matplotlib.pyplot as plt

# Load the data

file\_path = '/mnt/data/heart (1).csv'

data = pd.read\_csv(file\_path)

# Separate features and target

X = data.drop('target', axis=1)

y = data['target']

# 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)

# Standardize the features

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.transform(X\_test)

# Initialize models

svm = SVC(probability=True, random\_state=42)

rf = RandomForestClassifier(random\_state=42)

knn = KNeighborsClassifier()

# Train models

svm.fit(X\_train, y\_train)

rf.fit(X\_train, y\_train)

knn.fit(X\_train, y\_train)

# Predict probabilities

svm\_probs = svm.predict\_proba(X\_test)[:, 1]

rf\_probs = rf.predict\_proba(X\_test)[:, 1]

knn\_probs = knn.predict\_proba(X\_test)[:, 1]

# Predict classes

svm\_preds = svm.predict(X\_test)

rf\_preds = rf.predict(X\_test)

knn\_preds = knn.predict(X\_test)

# Calculate accuracy

svm\_accuracy = accuracy\_score(y\_test, svm\_preds)

rf\_accuracy = accuracy\_score(y\_test, rf\_preds)

knn\_accuracy = accuracy\_score(y\_test, knn\_preds)

# Calculate F1 score

svm\_f1 = f1\_score(y\_test, svm\_preds)

rf\_f1 = f1\_score(y\_test, rf\_preds)

knn\_f1 = f1\_score(y\_test, knn\_preds)

# Calculate ROC curve

svm\_fpr, svm\_tpr, \_ = roc\_curve(y\_test, svm\_probs)

rf\_fpr, rf\_tpr, \_ = roc\_curve(y\_test, rf\_probs)

knn\_fpr, knn\_tpr, \_ = roc\_curve(y\_test, knn\_probs)

# Calculate AUC

svm\_auc = auc(svm\_fpr, svm\_tpr)

rf\_auc = auc(rf\_fpr, rf\_tpr)

knn\_auc = auc(knn\_fpr, knn\_tpr)

# Plot ROC curve

plt.figure(figsize=(10, 6))

RocCurveDisplay(fpr=svm\_fpr, tpr=svm\_tpr, roc\_auc=svm\_auc, estimator\_name='SVM').plot()

RocCurveDisplay(fpr=rf\_fpr, tpr=rf\_tpr, roc\_auc=rf\_auc, estimator\_name='Random Forest').plot()

RocCurveDisplay(fpr=knn\_fpr, tpr=knn\_tpr, roc\_auc=knn\_auc, estimator\_name='KNN').plot()

plt.title('ROC Curve')

plt.show()

# Results

print(f'SVM Accuracy: {svm\_accuracy}')

print(f'Random Forest Accuracy: {rf\_accuracy}')

print(f'KNN Accuracy: {knn\_accuracy}')

print(f'SVM F1 Score: {svm\_f1}')

print(f'Random Forest F1 Score: {rf\_f1}')

print(f'KNN F1 Score: {knn\_f1}')