Untitled33

April 20, 2025

[21]:

```
# Step 1: Load and Preview the Heart Disease dataset
import pandas as pd

try:
    df = pd.read_csv("simulated_heart_disease.csv")
    print(" Dataset loaded successfully!\n")
    print(df.head())

    print("\n Missing value counts:")
    print(df.isnull().sum())

except FileNotFoundError:
    print(" File not found. Make sure 'simulated_heart_disease.csv' is in the␣
 ↪same folder as this notebook.")
```

```
 Dataset loaded successfully!

   age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  slope  \
0   67    1   2       126   458    1        0      144      0      2.3      2
1   57    0   0       158   384    0        1      133      0      6.2      0
2   43    0   3       111   286    0        0      130      0      2.8      1
3   71    1   2       189   515    1        1      149      0      2.1      1
4   36    0   0       142   303    0        0      107      1      3.6      1

   ca  thal  target
0   2     2       1
1   4     0       0
2   0     2       0
3   2     1       1
4   0     0       1

 Missing value counts:
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
```

[22]:

```
from sklearn.model_selection import train_test_split

# Split into features and target
X = df.drop("target", axis=1)
y = df["target"]

# Train/test split
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.8,␣
 ↪random_state=42)
print("Train shape:", X_train.shape)
print("Test shape:", X_test.shape)
```

```
Train shape: (242, 13)
Test shape: (61, 13)
```

[23]:

```
from sklearn.svm import SVC
from sklearn.ensemble import GradientBoostingClassifier, RandomForestClassifier

# Models + hyperparameter grids
grid_params = {
    'SVM': {
        'model': SVC(probability=True, random_state=42),
        'params': {'kernel': ['linear', 'rbf'], 'C': [0.1, 1, 10]}
    },
    'GradientBoosting': {
        'model': GradientBoostingClassifier(random_state=42),
        'params': {'n_estimators': [100, 150], 'learning_rate': [0.1, 0.05],␣
 ↪'max_depth': [3, 5]}
    },
    'RandomForest': {
        'model': RandomForestClassifier(random_state=42),
        'params': {'n_estimators': [100, 150], 'max_depth': [None, 10],␣
 ↪'min_samples_split': [2, 5]}
    }
}
```

[24]:

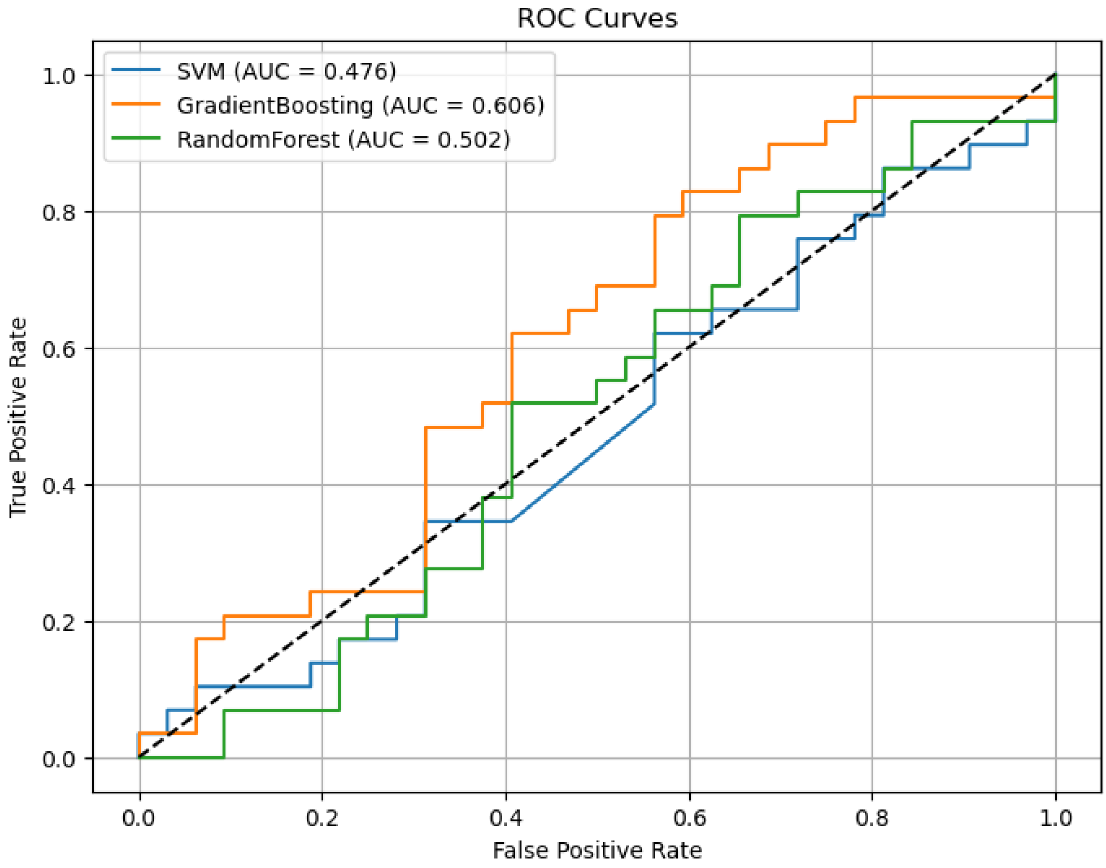
```
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score, precision_score, recall_score,␣
 ↪f1_score, roc_auc_score

results = []
best_models = {}

for name, gp in grid_params.items():
    print(f"Training {name}...")
    clf = GridSearchCV(gp['model'], gp['params'], cv=5, scoring='f1')
    clf.fit(X_train, y_train)
    best = clf.best_estimator_
    best_models[name] = best

    y_pred = best.predict(X_test)
    y_proba = best.predict_proba(X_test)[:, 1]

    # Store metrics
    results.append({
        'Model': name,
        'Accuracy': round(accuracy_score(y_test, y_pred), 3),
        'Precision': round(precision_score(y_test, y_pred), 3),
        'Recall': round(recall_score(y_test, y_pred), 3),
        'F1-score': round(f1_score(y_test, y_pred), 3),
        'AUC-ROC': round(roc_auc_score(y_test, y_proba), 3)
    })

print(" All models trained.")
```

```
Training SVM…
Training GradientBoosting…
Training RandomForest…
 All models trained.
```

[25]:

```
import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve, roc_auc_score

# Summary table
df_results = pd.DataFrame(results).set_index('Model')
print(" Model Performance Summary:")
display(df_results)  # Use display() for better notebook output

# Plot ROC Curves
plt.figure(figsize=(8, 6))

for name, model in best_models.items():
    try:
        y_proba = model.predict_proba(X_test)[:, 1]
        fpr, tpr, _ = roc_curve(y_test, y_proba)
        auc = roc_auc_score(y_test, y_proba)
        plt.plot(fpr, tpr, label=f"{name} (AUC = {round(auc, 3)})")
    except Exception as e:
        print(f"  Skipped {name} due to error: {e}")

plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curves")
plt.legend()
plt.grid(True)
plt.show()
```

```
 Model Performance Summary:

                  Accuracy  Precision  Recall  F1-score  AUC-ROC
Model
SVM                  0.475      0.452   0.483     0.467    0.476
GradientBoosting     0.590      0.567   0.586     0.576    0.606
RandomForest         0.525      0.500   0.448     0.473    0.502
```



5

[ ]:

```

```