heart

July 23, 2025

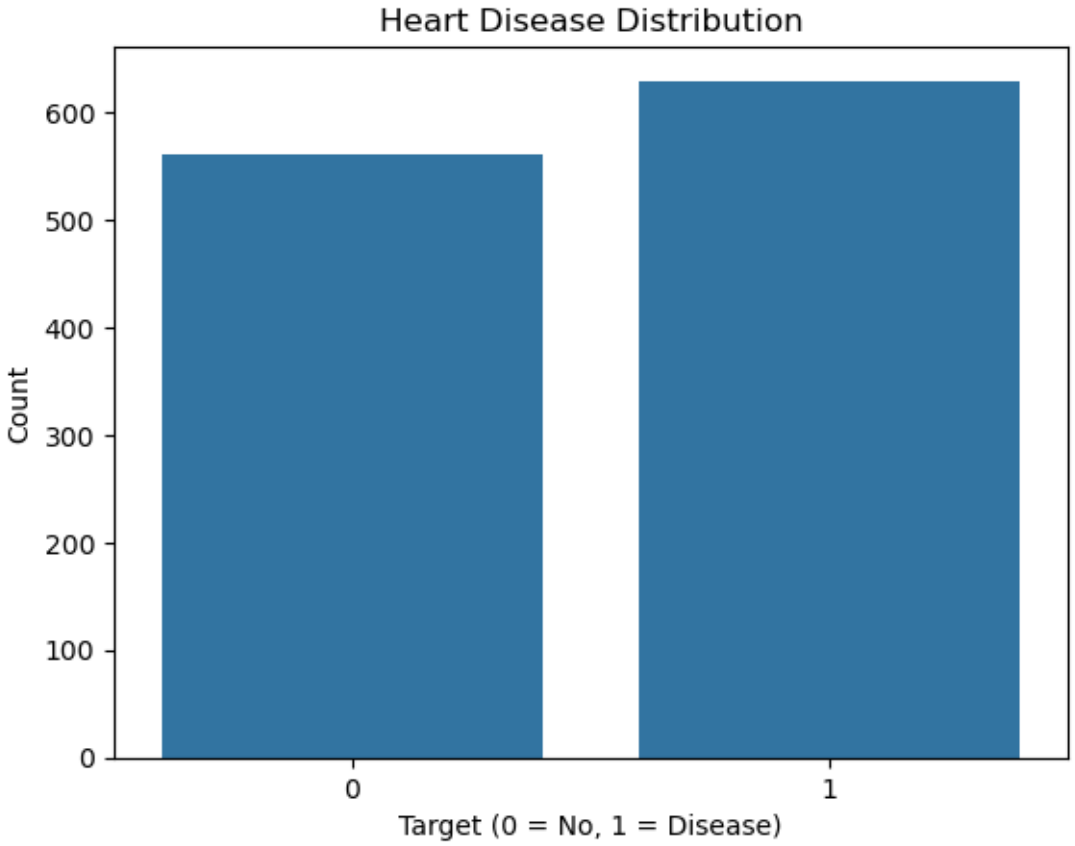
[21]:
```
# 1. Import Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, classification_report,␣
 ↪confusion_matrix, roc_auc_score, roc_curve
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from xgboost import XGBClassifier
import joblib

# 2. Load Dataset
df = pd.read_csv("/Users/hashithareddy/Desktop/HEART.csv")
print("First 5 rows:\n", df.head())

# 3. Check for Missing Values
print("\nMissing values:\n", df.isnull().sum())
print("\nData types:\n", df.dtypes)

# 4. Exploratory Data Analysis (EDA)
sns.countplot(x='target', data=df)
plt.title('Heart Disease Distribution')
plt.xlabel('Target (0 = No, 1 = Disease)')
plt.ylabel('Count')
plt.show()

plt.figure(figsize=(12, 8))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.title('Feature Correlation')
plt.show()

# 5. Data Preprocessing
```

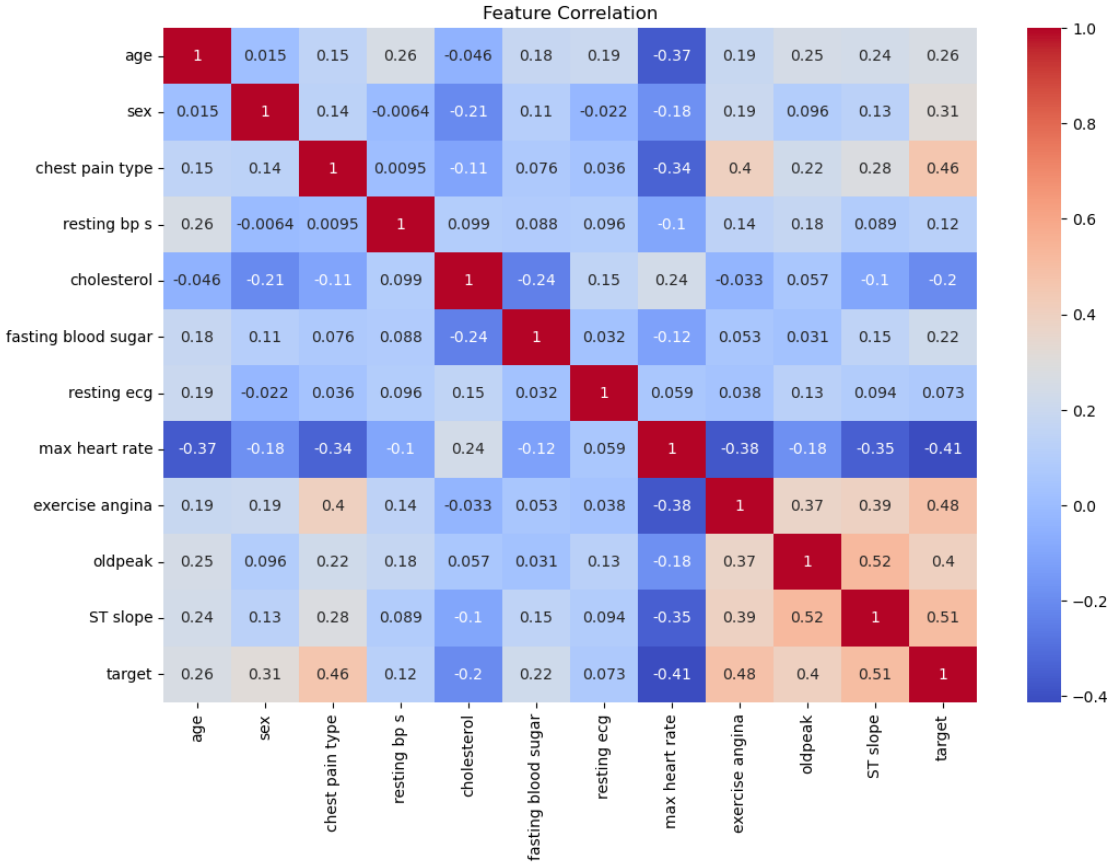
```
X = df.drop('target', axis=1)
y = df['target']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# 6. Initialize Models
models = {
    'Logistic Regression': LogisticRegression(max_iter=1000),
    'Random Forest': RandomForestClassifier(n_estimators=100, random_state=42),
    'Support Vector Machine': SVC(probability=True),
    'XGBoost': XGBClassifier(use_label_encoder=False, eval_metric='logloss')
}

# 7. Train and Evaluate Models
for name, model in models.items():
    print(f"\n\U0001f9e0 {name}")
    model.fit(X_train_scaled, y_train)
    y_pred = model.predict(X_test_scaled)
    y_prob = model.predict_proba(X_test_scaled)[:, 1]

    acc = accuracy_score(y_test, y_pred)
    roc = roc_auc_score(y_test, y_prob)
    print(f"Accuracy: {acc:.4f}")
    print(f"ROC AUC Score: {roc:.4f}")
    print("Classification Report:\n", classification_report(y_test, y_pred))
    print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))

    fpr, tpr, _ = roc_curve(y_test, y_prob)
    plt.plot(fpr, tpr, label=f'{name} (AUC = {roc:.2f})')

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

# 9. Save the Best Model
best_model = models['Random Forest']
joblib.dump(best_model, 'heart_disease_model.pkl')
```

```
print("\n\u2705 Model saved as heart_disease_model.pkl")

joblib.dump(scaler, 'scaler.pkl')
print("\u2705 Scaler saved as scaler.pkl")
```

```
First 5 rows:
    age  sex  chest pain type  resting bp s  cholesterol  fasting blood sugar  \
0   40    1                2           140          289                    0
1   49    0                3           160          180                    0
2   37    1                2           130          283                    0
3   48    0                4           138          214                    0
4   54    1                3           150          195                    0

   resting ecg  max heart rate  exercise angina  oldpeak  ST slope  target
0            0             172                0      0.0         1       0
1            0             156                0      1.0         2       1
2            1              98                0      0.0         1       0
3            0             108                1      1.5         2       1
4            0             122                0      0.0         1       0

Missing values:
 age                   0
sex                    0
chest pain type        0
resting bp s           0
cholesterol            0
fasting blood sugar    0
resting ecg            0
max heart rate         0
exercise angina        0
oldpeak                0
ST slope               0
target                 0
dtype: int64

Data types:
 age                     int64
sex                      int64
chest pain type          int64
resting bp s             int64
cholesterol              int64
fasting blood sugar      int64
resting ecg              int64
max heart rate           int64
exercise angina          int64
oldpeak                float64
ST slope                 int64
```

```
target                    int64
dtype: object
```

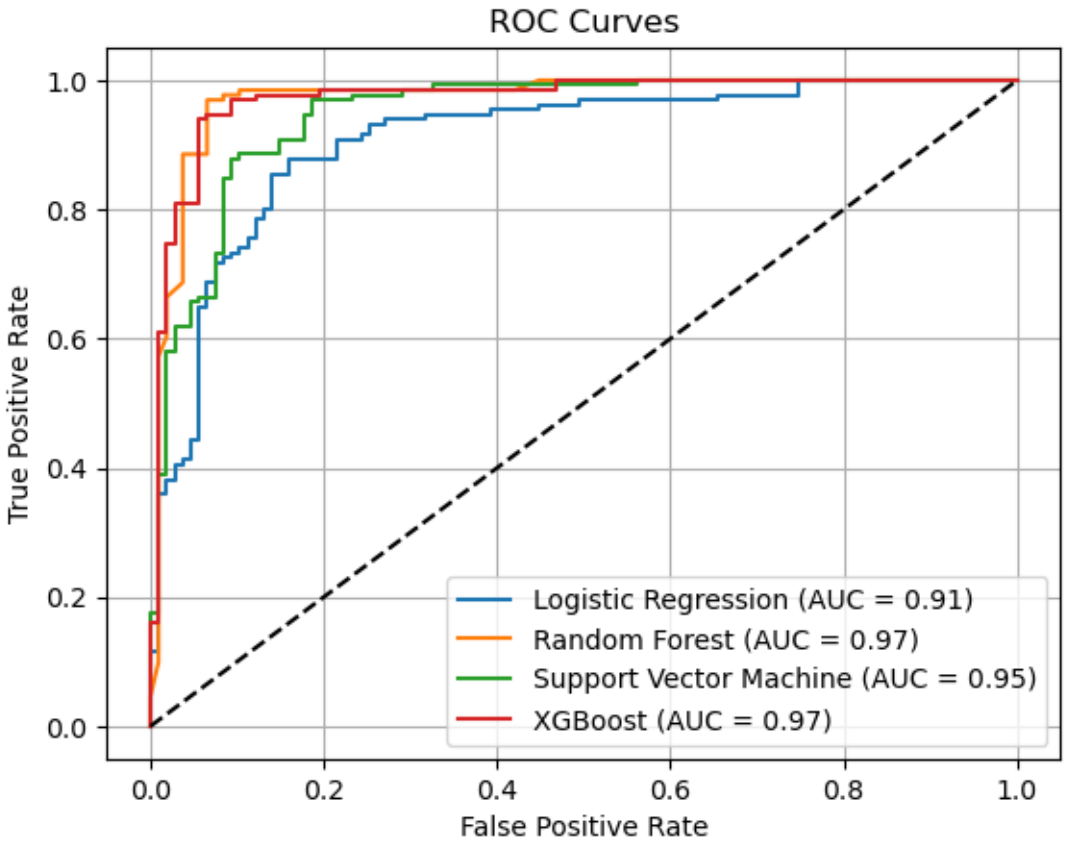





```
  Logistic Regression
Accuracy: 0.8613
ROC AUC Score: 0.9088
Classification Report:
              precision    recall  f1-score   support

           0       0.85      0.84      0.85       107
           1       0.87      0.88      0.87       131

    accuracy                           0.86       238
   macro avg       0.86      0.86      0.86       238
weighted avg       0.86      0.86      0.86       238

Confusion Matrix:
 [[ 90  17]
 [ 16 115]]

  Random Forest
Accuracy: 0.9538
```

```
ROC AUC Score: 0.9721
Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.93      0.95       107
           1       0.95      0.97      0.96       131

    accuracy                           0.95       238
   macro avg       0.95      0.95      0.95       238
weighted avg       0.95      0.95      0.95       238

Confusion Matrix:
 [[100   7]
 [  4 127]]

  Support Vector Machine
Accuracy: 0.8908
ROC AUC Score: 0.9475
Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.82      0.87       107
           1       0.87      0.95      0.91       131

    accuracy                           0.89       238
   macro avg       0.90      0.88      0.89       238
weighted avg       0.89      0.89      0.89       238

Confusion Matrix:
 [[ 88  19]
 [  7 124]]

  XGBoost
Accuracy: 0.9286
ROC AUC Score: 0.9720
Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.91      0.92       107
           1       0.93      0.95      0.94       131

    accuracy                           0.93       238
   macro avg       0.93      0.93      0.93       238
weighted avg       0.93      0.93      0.93       238

Confusion Matrix:
 [[ 97  10]
 [  7 124]]
```

```
/opt/anaconda3/lib/python3.12/site-packages/xgboost/training.py:183:
UserWarning: [20:51:56] WARNING:
/Users/runner/work/xgboost/xgboost/src/learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
```




```
Model saved as heart_disease_model.pkl
Scaler saved as scaler.pkl
```

[22]:
```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier()
model.fit(X_train, y_train)
```

[22]:
```
RandomForestClassifier()
```

[41]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report,␣
 ↪roc_auc_score, confusion_matrix, roc_curve
```

```
# Train the model
log_reg = LogisticRegression(max_iter=1000)
log_reg.fit(X_train_scaled, y_train)

# Predict
y_pred_log = log_reg.predict(X_test_scaled)
y_prob_log = log_reg.predict_proba(X_test_scaled)[:, 1]

# Evaluate
print(" Logistic Regression")
print("Accuracy:", accuracy_score(y_test, y_pred_log))
print("ROC AUC Score:", roc_auc_score(y_test, y_prob_log))
print("Classification Report:\n", classification_report(y_test, y_pred_log))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_log))
```

```
 Logistic Regression
Accuracy: 0.8613445378151261
ROC AUC Score: 0.9088249982164515
Classification Report:
              precision    recall  f1-score   support

           0       0.85      0.84      0.85       107
           1       0.87      0.88      0.87       131

    accuracy                           0.86       238
   macro avg       0.86      0.86      0.86       238
weighted avg       0.86      0.86      0.86       238

Confusion Matrix:
 [[ 90  17]
 [ 16 115]]
```

[27]:
```
from sklearn.svm import SVC

# Train the model
svm_model = SVC(probability=True)
svm_model.fit(X_train_scaled, y_train)

# Predict
y_pred_svm = svm_model.predict(X_test_scaled)
y_prob_svm = svm_model.predict_proba(X_test_scaled)[:, 1]

# Evaluate
print(" Support Vector Machine")
print("Accuracy:", accuracy_score(y_test, y_pred_svm))
print("ROC AUC Score:", roc_auc_score(y_test, y_prob_svm))
```

```
print("Classification Report:\n", classification_report(y_test, y_pred_svm))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_svm))
```

```
 Support Vector Machine
Accuracy: 0.8907563025210085
ROC AUC Score: 0.9475636726831704
Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.82      0.87       107
           1       0.87      0.95      0.91       131

    accuracy                           0.89       238
   macro avg       0.90      0.88      0.89       238
weighted avg       0.89      0.89      0.89       238

Confusion Matrix:
 [[ 88  19]
 [  7 124]]
```

[29]:
```
from xgboost import XGBClassifier

# Train the model
xgb_model = XGBClassifier(use_label_encoder=False, eval_metric='logloss')
xgb_model.fit(X_train_scaled, y_train)

# Predict
y_pred_xgb = xgb_model.predict(X_test_scaled)
y_prob_xgb = xgb_model.predict_proba(X_test_scaled)[:, 1]

# Evaluate
print("  XGBoost")
print("Accuracy:", accuracy_score(y_test, y_pred_xgb))
print("ROC AUC Score:", roc_auc_score(y_test, y_prob_xgb))
print("Classification Report:\n", classification_report(y_test, y_pred_xgb))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_xgb))
```

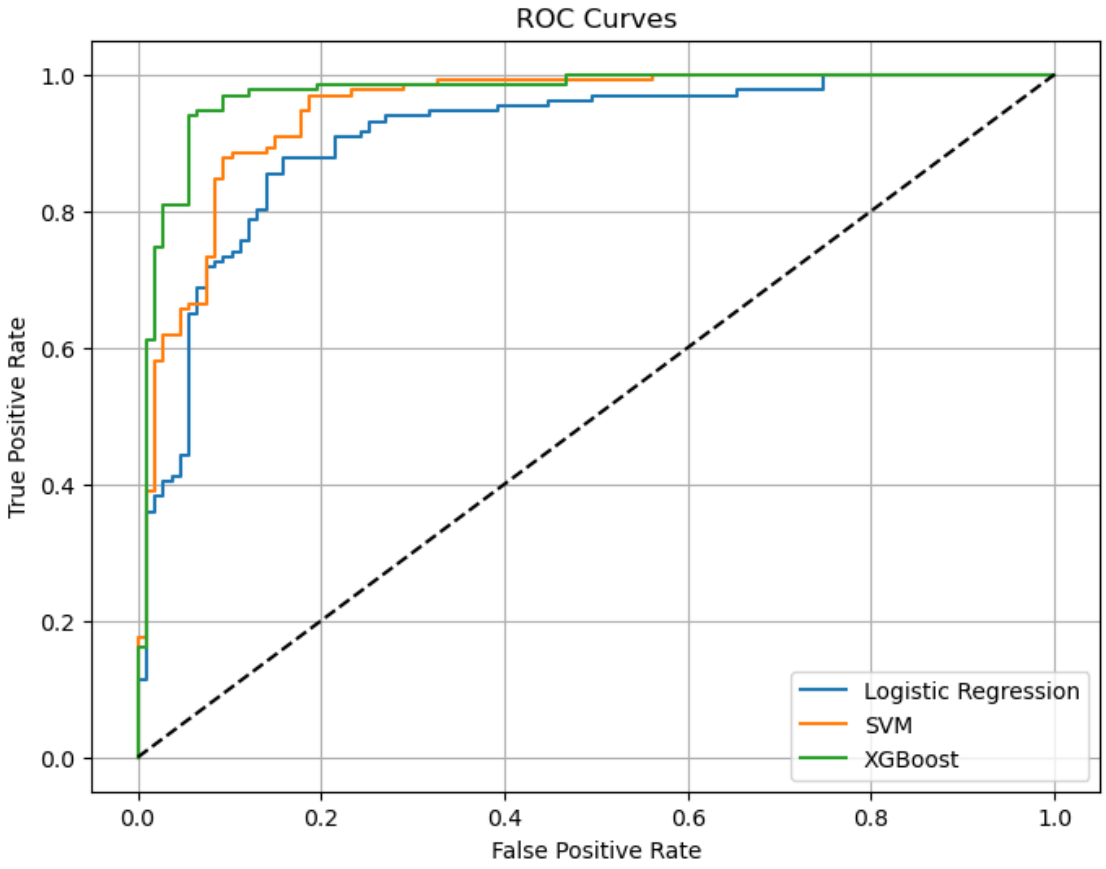
```
  XGBoost
Accuracy: 0.9285714285714286
ROC AUC Score: 0.9720339587643575
Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.91      0.92       107
           1       0.93      0.95      0.94       131

    accuracy                           0.93       238
   macro avg       0.93      0.93      0.93       238
```

```
weighted avg       0.93      0.93      0.93       238

Confusion Matrix:
 [[ 97  10]
 [  7 124]]
/opt/anaconda3/lib/python3.12/site-packages/xgboost/training.py:183:
UserWarning: [20:51:57] WARNING:
/Users/runner/work/xgboost/xgboost/src/learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
```

[31]:
```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

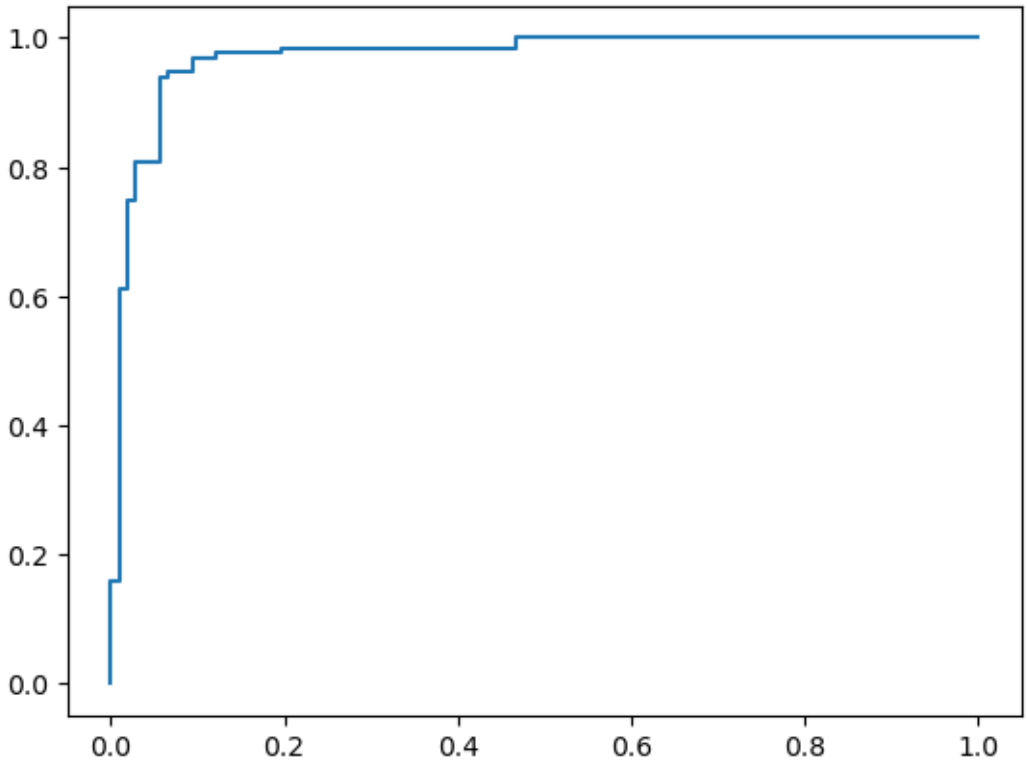
```
              precision    recall  f1-score   support

           0       0.93      0.91      0.92       107
           1       0.93      0.95      0.94       131

    accuracy                           0.93       238
   macro avg       0.93      0.93      0.93       238
weighted avg       0.93      0.93      0.93       238
```

[33]:
```
import matplotlib.pyplot as plt

fpr_log, tpr_log, _ = roc_curve(y_test, y_prob_log)
fpr_svm, tpr_svm, _ = roc_curve(y_test, y_prob_svm)
fpr_xgb, tpr_xgb, _ = roc_curve(y_test, y_prob_xgb)

plt.figure(figsize=(8,6))
plt.plot(fpr_log, tpr_log, label='Logistic Regression')
plt.plot(fpr_svm, tpr_svm, label='SVM')
plt.plot(fpr_xgb, tpr_xgb, label='XGBoost')
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()
```



[35]:
```
from sklearn.metrics import roc_curve
fpr, tpr, _ = roc_curve(y_test, y_prob)
plt.plot(fpr, tpr)
```

[35]: [<matplotlib.lines.Line2D at 0x14ce7c260>]



[37]:
```
import joblib
joblib.dump(model, "heart_disease_model.pkl")
```

[37]:
```
['heart_disease_model.pkl']
```

[5]:
```
import pandas as pd
from sklearn.preprocessing import StandardScaler
import joblib

# Load your dataset
df = pd.read_csv("/Users/hashithareddy/Desktop/HEART.csv")

# Drop the target column if it exists
X = df.drop('target', axis=1) if 'target' in df.columns else df.copy()

# Initialize and fit the scaler
scaler = StandardScaler()
scaler.fit(X)

# Save the scaler
joblib.dump(scaler, "scaler.pkl")
print(" Scaler saved as scaler.pkl")
```

```
Scaler saved as scaler.pkl
```

[ ]: