Heart Failure Prediction with EDA, Modeling & Evaluation

```
# Step 1: Mount Google Drive and Import Libraries
from google.colab import drive
drive.mount('/content/drive')
```

```
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, GridSearchCV
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, classification_report, roc_auc_score, roc_curve
import warnings
warnings.filterwarnings('ignore')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

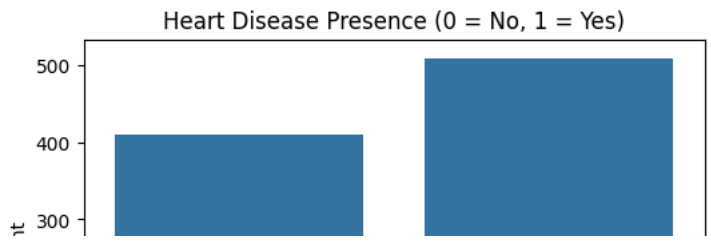
Step 2: Load Data

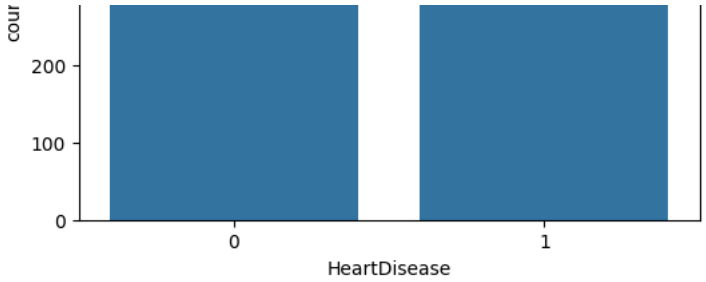
```
path = '/content/drive/MyDrive/Colab Notebooks/datasets/heart.csv'
df = pd.read_csv(path)
```

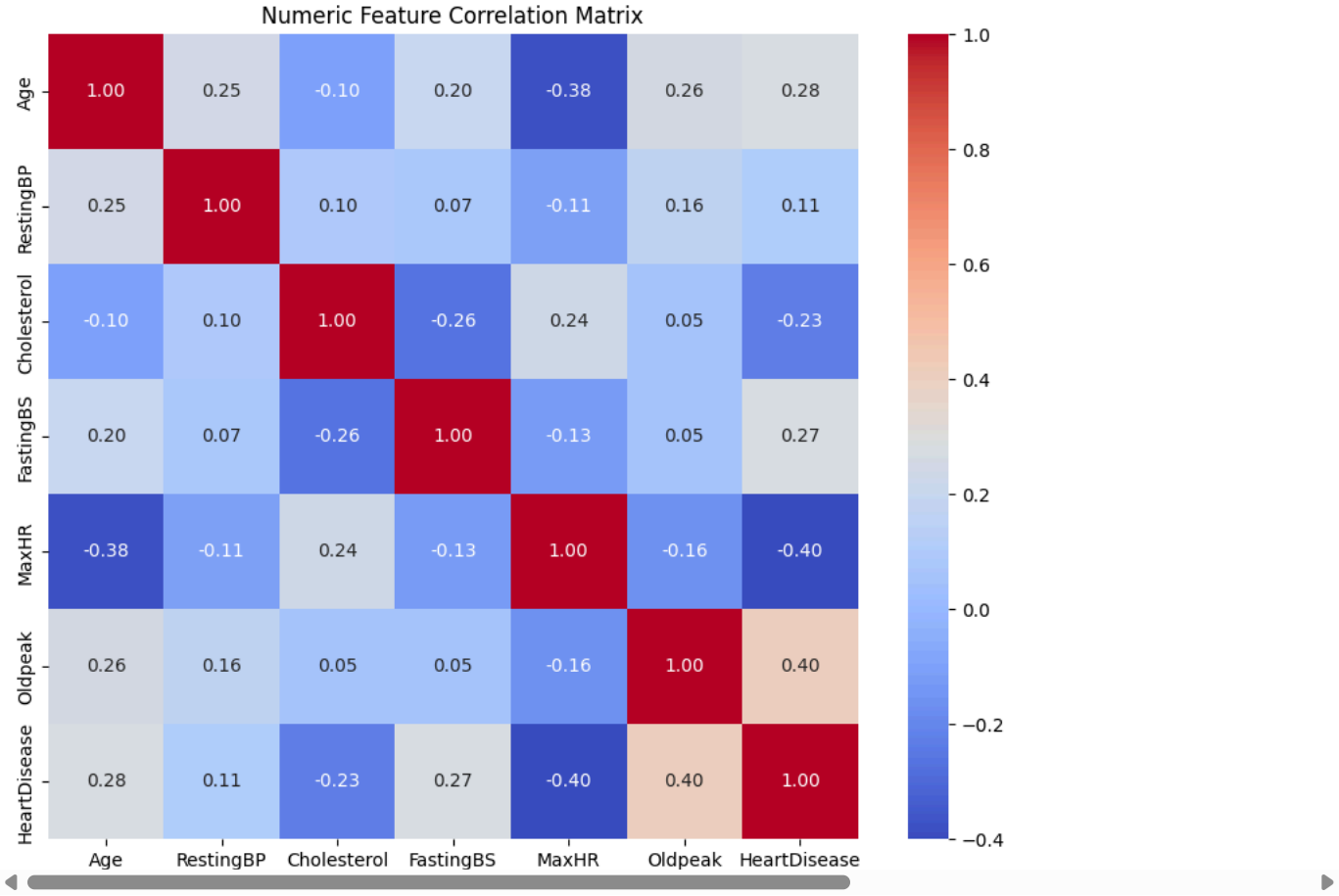
Step 3: Exploratory Data Analysis (EDA)

```
# Step 3: Exploratory Data Analysis (EDA)
print("First 5 rows:\n", df.head(), "\n")
print("Info:\n"); df.info(); print("\n")
print("Stats:\n", df.describe(), "\n")
print("Missing values:\n", df.isnull().sum(), "\n")

# 4a: Target distribution
plt.figure(figsize=(6,4))
sns.countplot(x='HeartDisease', data=df)
plt.title('Heart Disease Presence (0 = No, 1 = Yes)')
plt.show()

# 4b: Correlation heatmap on numeric features only
num_cols = df.select_dtypes(include=['number']).columns
plt.figure(figsize=(10,8))
sns.heatmap(df[num_cols].corr(), annot=True, fmt=".2f", cmap='coolwarm')
plt.title("Numeric Feature Correlation Matrix")
plt.show()
```

```
First 5 rows:
   Age Sex ChestPainType  RestingBP  Cholesterol  FastingBS RestingECG  MaxHR  \
0   40   M           ATA        140          289          0     Normal    172
1   49   F           NAP        160          180          0     Normal    156
2   37   M           ATA        130          283          0         ST     98
3   48   F           ASY        138          214          0     Normal    108
4   54   M           NAP        150          195          0     Normal    122

  ExerciseAngina  Oldpeak ST_Slope  HeartDisease
0              N      0.0       Up             0
1              N      1.0     Flat             1
2              N      0.0       Up             0
3              Y      1.5     Flat             1
4              N      0.0       Up             0

Info:

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 918 entries, 0 to 917
Data columns (total 12 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Age             918 non-null    int64
 1   Sex             918 non-null    object
 2   ChestPainType   918 non-null    object
 3   RestingBP       918 non-null    int64
 4   Cholesterol     918 non-null    int64
 5   FastingBS       918 non-null    int64
 6   RestingECG      918 non-null    object
 7   MaxHR           918 non-null    int64
 8   ExerciseAngina  918 non-null    object
 9   Oldpeak         918 non-null    float64
 10  ST_Slope        918 non-null    object
 11  HeartDisease    918 non-null    int64
dtypes: float64(1), int64(6), object(5)
memory usage: 86.2+ KB


Stats:
               Age   RestingBP  Cholesterol   FastingBS       MaxHR  \
count  918.000000  918.000000   918.000000  918.000000  918.000000
mean    53.510893  132.396514   198.799564    0.233115  136.809368
std      9.432617   18.514154   109.384145    0.423046   25.460334
min     28.000000    0.000000     0.000000    0.000000   60.000000
25%     47.000000  120.000000   173.250000    0.000000  120.000000
50%     54.000000  130.000000   223.000000    0.000000  138.000000
75%     60.000000  140.000000   267.000000    0.000000  156.000000
max     77.000000  200.000000   603.000000    1.000000  202.000000

          Oldpeak  HeartDisease
count  918.000000    918.000000
mean     0.887364      0.553377
std      1.066570      0.497414
min     -2.600000      0.000000
25%      0.000000      0.000000
50%      0.600000      1.000000
75%      1.500000      1.000000
max      6.200000      1.000000

Missing values:
 Age               0
Sex                0
ChestPainType      0
RestingBP          0
Cholesterol        0
FastingBS          0
RestingECG         0
MaxHR              0
ExerciseAngina     0
Oldpeak            0
ST_Slope           0
HeartDisease       0
dtype: int64
```




cour
200
100
0
0
1
HeartDisease


Numeric Feature Correlation Matrix
Age
RestingBP
Cholesterol
FastingBS
MaxHR
Oldpeak
HeartDisease
1.00 0.25 -0.10 0.20 -0.38 0.26 0.28
0.25 1.00 0.10 0.07 -0.11 0.16 0.11
-0.10 0.10 1.00 -0.26 0.24 0.05 -0.23
0.20 0.07 -0.26 1.00 -0.13 0.05 0.27
-0.38 -0.11 0.24 -0.13 1.00 -0.16 -0.40
0.26 0.16 0.05 0.05 -0.16 1.00 0.40
0.28 0.11 -0.23 0.27 -0.40 0.40 1.00
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4

Step 4: Preprocess (encode categoricals) & Split

```
X = pd.get_dummies(df.drop('HeartDisease', axis=1), drop_first=True)
y = df['HeartDisease']

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, stratify=y, random_state=42)
```

Step 5: Feature Scaling (only for SVM)

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

Step 6: Define Models & Grids

```
models = {
    'SVM': SVC(probability=True, random_state=42),
    'RandomForest': RandomForestClassifier(random_state=42),
    'GBM': GradientBoostingClassifier(random_state=42)
}
param_grids = {
    'SVM': {
        'kernel': ['linear','rbf'],
        'C': [0.1, 1, 10],
        'gamma': ['scale','auto']
    },
    'RandomForest': {
        'n_estimators': [50, 100, 200],
        'max_depth': [None, 5, 10],
        'min_samples_split': [2, 5]
    },
    'GBM': {
        'n_estimators': [50, 100, 200],
        'learning_rate': [0.01, 0.1, 0.2],
        'max_depth': [3, 5, 7]
    }
}
```

Step 7: Hyperparameter Tuning

```
best_estimators = {}
for name, model in models.items():
    print(f"Tuning {name}...")
    grid = GridSearchCV(model, param_grids[name], cv=5, scoring='roc_auc', n_jobs=-1)
    X_tr = X_train_scaled if name=='SVM' else X_train
    grid.fit(X_tr, y_train)
    best_estimators[name] = grid.best_estimator_
    print(f"→ Best {name} params: {grid.best_params_}\n")
```

```
Tuning SVM...
→ Best SVM params: {'C': 10, 'gamma': 'scale', 'kernel': 'linear'}

Tuning RandomForest...
→ Best RandomForest params: {'max_depth': 10, 'min_samples_split': 2, 'n_estimators': 200}

Tuning GBM...
→ Best GBM params: {'learning_rate': 0.1, 'max_depth': 3, 'n_estimators': 50}
```

Step 8: Evaluate & Summarize

```
results = []
for name, est in best_estimators.items():
    X_te = X_test_scaled if name=='SVM' else X_test
    y_pred = est.predict(X_te)
    y_proba = est.predict_proba(X_te)[:,1]
    rpt = classification_report(y_test, y_pred, output_dict=True)

    results.append({
        'Model':    name,
        'Accuracy': accuracy_score(y_test, y_pred),
        'Precision': rpt['1']['precision'],
        'Recall':    rpt['1']['recall'],
        'F1-score':  rpt['1']['f1-score'],
        'AUC-ROC':   roc_auc_score(y_test, y_proba)
    })

res_df = pd.DataFrame(results).set_index('Model')
print("\nModel Comparison:\n", res_df, "\n")
```

```
Model Comparison:
               Accuracy  Precision    Recall  F1-score   AUC-ROC
Model
SVM           0.875000   0.855856  0.931373  0.892019  0.930536
RandomForest  0.880435   0.884615  0.901961  0.893204  0.936275
GBM           0.891304   0.894231  0.911765  0.902913  0.937171
```

Step 9: Plot ROC Curves

```
plt.figure(figsize=(8,6))
for name, est in best_estimators.items():
    X_te = X_test_scaled if name=='SVM' else X_test
    y_proba = est.predict_proba(X_te)[:,1]
    fpr, tpr, _ = roc_curve(y_test, y_proba)
    plt.plot(fpr, tpr, label=f"{name} (AUC={roc_auc_score(y_test, y_proba):.2f})")

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