```python
# Import necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.svm import SVC
from sklearn.ensemble import GradientBoostingClassifier,
RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score, roc_auc_score

# Load the dataset
file_path ='heart.csv'  # Update this path if your file is in a
different location
heart_data = pd.read_csv(file_path)

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

# Split the dataset into training and test sets (80% training, 20%
test)
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Initialize the models
svm_model = SVC(probability=True, random_state=42)
gbm_model = GradientBoostingClassifier(random_state=42)
rf_model = RandomForestClassifier(random_state=42)

# Train the models
svm_model.fit(X_train, y_train)
gbm_model.fit(X_train, y_train)
rf_model.fit(X_train, y_train)

# Predict on the test set
svm_pred = svm_model.predict(X_test)
gbm_pred = gbm_model.predict(X_test)
rf_pred = rf_model.predict(X_test)

# Calculate probabilities for ROC-AUC
svm_prob = svm_model.predict_proba(X_test)[:, 1]
gbm_prob = gbm_model.predict_proba(X_test)[:, 1]
rf_prob = rf_model.predict_proba(X_test)[:, 1]

# Evaluate each model using various metrics
svm_metrics = {
    'Accuracy': accuracy_score(y_test, svm_pred),
    'Precision': precision_score(y_test, svm_pred),
    'Recall': recall_score(y_test, svm_pred),
    'F1 Score': f1_score(y_test, svm_pred),
    'AUC-ROC': roc_auc_score(y_test, svm_prob)
}

gbm_metrics = {
    'Accuracy': accuracy_score(y_test, gbm_pred),
    'Precision': precision_score(y_test, gbm_pred),
    'Recall': recall_score(y_test, gbm_pred),
    'F1 Score': f1_score(y_test, gbm_pred),
    'AUC-ROC': roc_auc_score(y_test, gbm_prob)
}

rf_metrics = {
    'Accuracy': accuracy_score(y_test, rf_pred),
    'Precision': precision_score(y_test, rf_pred),
    'Recall': recall_score(y_test, rf_pred),
    'F1 Score': f1_score(y_test, rf_pred),
    'AUC-ROC': roc_auc_score(y_test, rf_prob)
}

# Combine results into a DataFrame for comparison
results_df = pd.DataFrame([svm_metrics, gbm_metrics, rf_metrics],
                          index=['SVM', 'GBM', 'Random Forest'])

print("Initial Model Evaluation Results:")
print(results_df)

# Hyperparameter tuning using GridSearchCV for each model

# Define parameter grids for each model
param_grid_svm = {
    'C': [0.1, 1, 10, 100],
    'gamma': [1, 0.1, 0.01, 0.001],
    'kernel': ['rbf', 'linear']
}

param_grid_gbm = {
    'n_estimators': [50, 100, 150],
    'learning_rate': [0.01, 0.1, 0.2],
    'max_depth': [3, 4, 5]
}

param_grid_rf = {
    'n_estimators': [100, 200, 300],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

# Set up GridSearchCV for each model
grid_svm = GridSearchCV(SVC(probability=True, random_state=42),
param_grid_svm, cv=5, scoring='accuracy')
```

```
grid_gbm = GridSearchCV(GradientBoostingClassifier(random_state=42),
param_grid_gbm, cv=5, scoring='accuracy')
grid_rf = GridSearchCV(RandomForestClassifier(random_state=42),
param_grid_rf, cv=5, scoring='accuracy')

# Fit the grid searches to the data
grid_svm.fit(X_train, y_train)
grid_gbm.fit(X_train, y_train)
grid_rf.fit(X_train, y_train)

# Get the best models and parameters
best_svm = grid_svm.best_estimator_
best_gbm = grid_gbm.best_estimator_
best_rf = grid_rf.best_estimator_

print("\nBest Hyperparameters for each model:")
print("SVM:", grid_svm.best_params_)
print("GBM:", grid_gbm.best_params_)
print("Random Forest:", grid_rf.best_params_)

# Re-evaluate each model using the best hyperparameters

# Predict on the test set using the best models
best_svm_pred = best_svm.predict(X_test)
best_gbm_pred = best_gbm.predict(X_test)
best_rf_pred = best_rf.predict(X_test)

# Calculate probabilities for ROC-AUC using the best models
best_svm_prob = best_svm.predict_proba(X_test)[:, 1]
best_gbm_prob = best_gbm.predict_proba(X_test)[:, 1]
best_rf_prob = best_rf.predict_proba(X_test)[:, 1]

# Evaluate each model using the best parameters
best_svm_metrics = {
    'Accuracy': accuracy_score(y_test, best_svm_pred),
    'Precision': precision_score(y_test, best_svm_pred),
    'Recall': recall_score(y_test, best_svm_pred),
    'F1 Score': f1_score(y_test, best_svm_pred),
    'AUC-ROC': roc_auc_score(y_test, best_svm_prob)
}

best_gbm_metrics = {
    'Accuracy': accuracy_score(y_test, best_gbm_pred),
    'Precision': precision_score(y_test, best_gbm_pred),
    'Recall': recall_score(y_test, best_gbm_pred),
    'F1 Score': f1_score(y_test, best_gbm_pred),
    'AUC-ROC': roc_auc_score(y_test, best_gbm_prob)
}

best_rf_metrics = {
    'Accuracy': accuracy_score(y_test, best_rf_pred),
    'Precision': precision_score(y_test, best_rf_pred),
    'Recall': recall_score(y_test, best_rf_pred),
    'F1 Score': f1_score(y_test, best_rf_pred),
    'AUC-ROC': roc_auc_score(y_test, best_rf_prob)
}

# Combine results into a DataFrame for comparison after tuning
tuned_results_df = pd.DataFrame([best_svm_metrics, best_gbm_metrics,
best_rf_metrics],
                                index=['Tuned SVM', 'Tuned GBM',
'Tuned Random Forest'])

print("\nModel Evaluation Results After Hyperparameter Tuning:")
print(tuned_results_df)
```

```
Initial Model Evaluation Results:
               Accuracy  Precision   Recall  F1 Score   AUC-ROC
SVM            0.704918   0.666667  0.87500  0.756757  0.839440
GBM            0.770492   0.800000  0.75000  0.774194  0.903017
Random Forest  0.836066   0.843750  0.84375  0.843750  0.920259

Best Hyperparameters for each model:
SVM: {'C': 100, 'gamma': 1, 'kernel': 'linear'}
GBM: {'learning_rate': 0.2, 'max_depth': 5, 'n_estimators': 50}
Random Forest: {'max_depth': None, 'min_samples_leaf': 4,
'min_samples_split': 2, 'n_estimators': 200}

Model Evaluation Results After Hyperparameter Tuning:
                     Accuracy  Precision  Recall  F1 Score   AUC-ROC
Tuned SVM            0.836066   0.866667  0.8125  0.838710  0.921336
Tuned GBM            0.803279   0.857143  0.7500  0.800000  0.907328
Tuned Random Forest  0.852459   0.848485  0.8750  0.861538  0.938578
```