```
!pip install \
    scikit-learn==1.2.2 \
    numpy==1.25.2 \
    pandas==2.0.3 \
    scipy==1.11.2 \
    joblib==1.2.0 \
    threadpoolctl==3.1.0 \
    cython==0.29.36 \
    imbalanced-learn==0.12.0
```

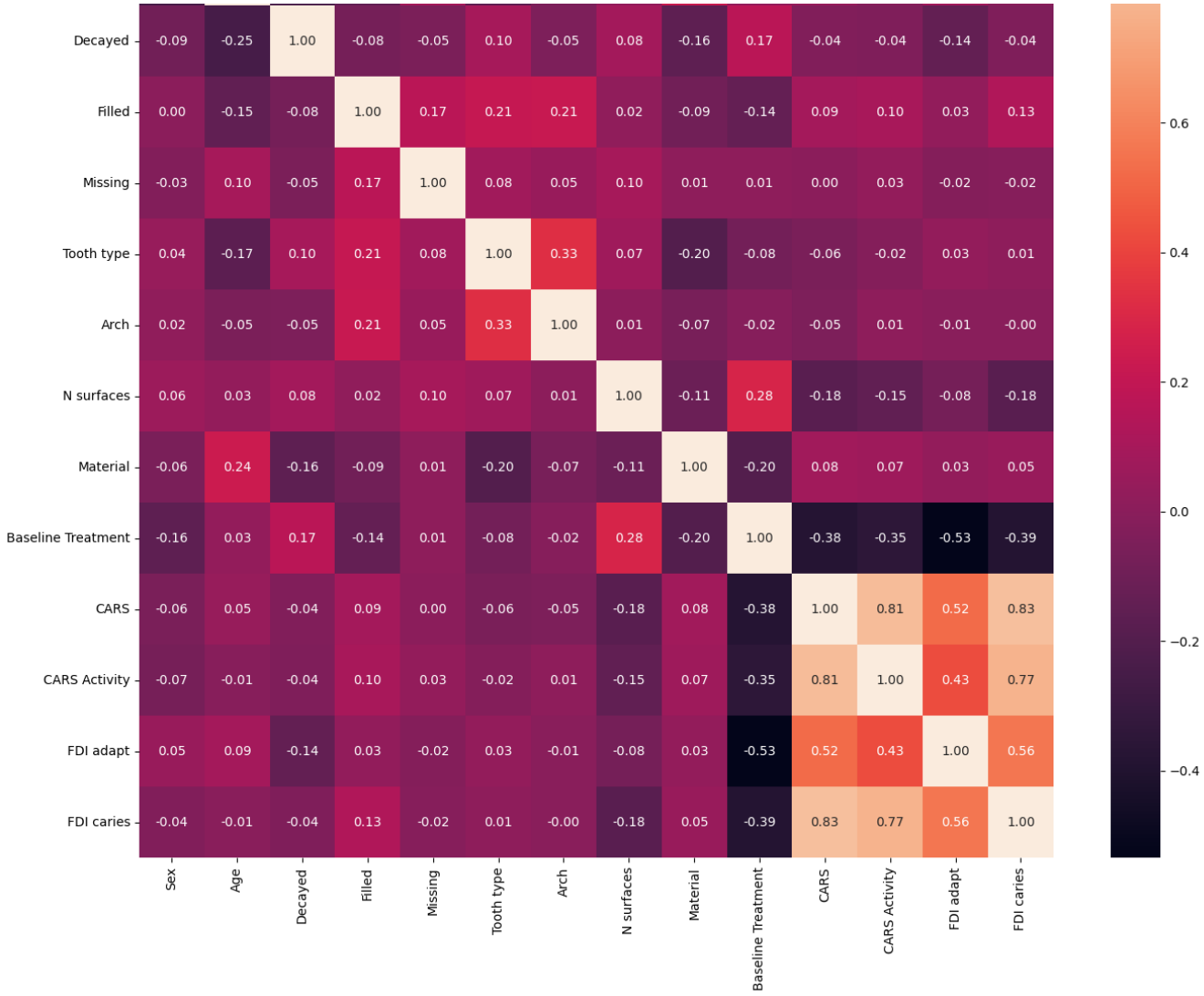
```
Requirement already satisfied: scikit-learn==1.2.2 in /usr/local/lib/python
Requirement already satisfied: numpy==1.25.2 in /usr/local/lib/python3.11/d
Requirement already satisfied: pandas==2.0.3 in /usr/local/lib/python3.11/d
Requirement already satisfied: scipy==1.11.2 in /usr/local/lib/python3.11/d
Requirement already satisfied: joblib==1.2.0 in /usr/local/lib/python3.11/d
Requirement already satisfied: threadpoolctl==3.1.0 in /usr/local/lib/pytho
Requirement already satisfied: cython==0.29.36 in /usr/local/lib/python3.11
Requirement already satisfied: imbalanced-learn==0.12.0 in /usr/local/lib/p
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/pyt
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/di
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.11/
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-p
```

```
pip freeze > new_env_requirements.txt
```

```
!python --version
```

```
Python 3.11.11
```

```
# Importing necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split

# Load the data from an Excel file
data = pd.read_excel('2024_corrected_global_CARDEC_3_ML_Vitor.xlsx')

# Split the dataset into training and testing sets based on a unique identifier
# This ensures that data related to the same 'IDpac' is not split across both t
unique_n_part = data['IDpac'].unique()
train_n_part, test_n_part = train_test_split(unique_n_part, test_size=0.2, rand

# Filter the original dataset to create training data that includes only the 'I
train_data = data[data['IDpac'].isin(train_n_part)]
# Similarly, filter the original dataset to create testing data that includes c
test_data = data[data['IDpac'].isin(test_n_part)]

# Separate features and target variable for training set
# 'drop' removes specified columns from the dataset, in this case removing targ
X_train = train_data.drop(['Failure', 'IDrest', 'IDpac'], axis=1)
y_train = train_data['Failure']  # Isolate the target variable for the training

# Separate features and target variable for testing set following the same proc
X_test = test_data.drop(['Failure', 'IDrest', 'IDpac'], axis=1)
y_test = test_data['Failure']  # Isolate the target variable for the testing se


import seaborn as sns
import matplotlib.pyplot as plt

# Calculate the correlation matrix of the training data.
# The correlation matrix quantifies the linear relationships between the variabl
corr_matrix = X_train.corr()

# Initialize a matplotlib figure with a specified size (width=16 inches, height=
# This size is chosen to make the heatmap large enough to be easily readable.
plt.figure(figsize=(16, 14))

# Draw the heatmap using seaborn to visualize the correlation matrix.
sns.heatmap(corr_matrix, annot=True, annot_kws={"size": 10}, fmt=".2f", cbar_kws

# Display the plot on the screen. This command is necessary to show the figure w
plt.show()
```




```
import pandas as pd
```

```
# Define lists for each type of variable in the dataset: numeric, binary, and or
numeric_vars = ['Age', 'Decayed', 'Filled', 'Missing']
binary_vars = ['Sex', 'Tooth type', 'Arch', 'Failure', 'CARS Activity']
categorical_vars = ['N surfaces', 'Material', 'Baseline Treatment', 'CARS', 'FDI

def descriptive_statistics(train_data, test_data):
    # Print a heading for the descriptive statistics of numeric variables.
    print("Descriptive Statistics for Numeric Variables:")
    # Display descriptive statistics (like count, mean, std, min, max, etc.) for
    print("\nTraining Set:")
    print(train_data[numeric_vars].describe())
    # Repeat the process for the test set.
    print("\nTest Set:")
    print(test_data[numeric_vars].describe())

    # Initialize an empty dictionary to store statistics for binary and ordinal
    stats = {}
    # Loop through each variable in the binary and ordinal lists to calculate th
    for var in binary_vars + categorical_vars:
        stats[var] = {
            "Training Set": {
                "Count": train_data[var].value_counts().to_dict(),  # Count occu
                "Percentage": (train_data[var].value_counts(normalize=True) * 10
            },
            "Test Set": {
                "Count": test_data[var].value_counts().to_dict(),  # Count occur
                "Percentage": (test_data[var].value_counts(normalize=True) * 100
            }
        }

    # Loop through the stats dictionary to print the statistics for each categor
    for var, data in stats.items():
        print(f"\n{var} Statistics:")  # Print the variable name.
        for dataset, values in data.items():
            print(f"\n{dataset}:")  # Print which dataset (training or test) the
            for metric, metric_values in values.items():
                print(f"{metric}: {metric_values}")  # Print the count and perce

# Call the function with the training and test datasets as arguments to display
descriptive_statistics(train_data, test_data)
```

```
N surfaces Statistics:

Training Set:
Count: {1: 207, 2: 139, 3: 72, 4: 60, 5: 29}
Percentage: {1: 40.828402366863905, 2: 27.416173570019726, 3: 14.2011834319
```

```
Test Set:
Count: {1: 56, 2: 27, 3: 20, 4: 14, 5: 13}
Percentage: {1: 43.07692307692308, 2: 20.76923076923077, 3: 15.384615384615

Material Statistics:

Training Set:
Count: {1: 304, 0: 189, 2: 14}
Percentage: {1: 59.96055226824457, 0: 37.278106508875744, 2: 2.761341222879

Test Set:
Count: {1: 74, 0: 51, 2: 5}
Percentage: {1: 56.92307692307692, 0: 39.23076923076923, 2: 3.8461538461538

Baseline Treatment Statistics:

Training Set:
Count: {0: 292, 1: 167, 2: 48}
Percentage: {0: 57.59368836291914, 1: 32.938856015779095, 2: 9.467455621301

Test Set:
Count: {0: 75, 1: 34, 2: 21}
Percentage: {0: 57.692307692307686, 1: 26.153846153846157, 2: 16.1538461538

CARS Statistics:

Training Set:
Count: {0: 399, 2: 62, 1: 46}
Percentage: {0: 78.69822485207101, 2: 12.22879684418146, 1: 9.0729783037475

Test Set:
Count: {0: 99, 2: 20, 1: 11}
Percentage: {0: 76.15384615384615, 2: 15.384615384615385, 1: 8.461538461538

FDI adapt Statistics:

Training Set:
Count: {0: 333, 1: 155, 2: 19}
Percentage: {0: 65.68047337278107, 1: 30.57199211045365, 2: 3.7475345167652

Test Set:
Count: {0: 82, 1: 43, 2: 5}
Percentage: {0: 63.07692307692307, 1: 33.07692307692307, 2: 3.8461538461538

FDI caries Statistics:

Training Set:
Count: {0: 401, 1: 98, 2: 8}
Percentage: {0: 79.09270216962526, 1: 19.32938856015779, 2: 1.5779092702169

Test Set:
Count: {0: 97, 1: 30, 2: 3}
Percentage: {0: 74.61538461538461, 1: 23.076923076923077, 2: 2.307692307692
```

```
import pandas as pd
from sklearn.preprocessing import StandardScaler

# Convert specified categorical variables in the training data to 'category' dt
X_train['Material'] = X_train['Material'].astype('category')
X_train['Baseline Treatment'] = X_train['Baseline Treatment'].astype('category'
X_train['CARS'] = X_train['CARS'].astype('category')
X_train['FDI adapt'] = X_train['FDI adapt'].astype('category')
X_train['FDI caries'] = X_train['FDI caries'].astype('category')

# Apply one-hot encoding to the specified categorical columns in the training d
# 'prefix' argument specifies the prefix to add to the columns resulting from t
one_hot_train = pd.get_dummies(X_train[['Material', 'Baseline Treatment', 'CARS
                               prefix=['Material', 'Baseline_Treatment', 'CARS'

# Concatenate the original training data (minus the now-encoded variables) with
X_train = pd.concat([X_train.drop(['Material', 'Baseline Treatment', 'CARS', 'F

# Initialize new one-hot encoded columns in the test data with zeros to match t
for col in one_hot_train.columns:
    X_test[col] = 0

# Convert specified categorical variables in the test data to 'category' dtype
X_test['Material'] = X_test['Material'].astype('category')
X_test['Baseline Treatment'] = X_test['Baseline Treatment'].astype('category')
X_test['CARS'] = X_test['CARS'].astype('category')
X_test['FDI adapt'] = X_test['FDI adapt'].astype('category')
X_test['FDI caries'] = X_test['FDI caries'].astype('category')

one_hot_test = pd.get_dummies(X_test[['Material', 'Baseline Treatment', 'CARS',
                              prefix=['Material', 'Baseline_Treatment', 'CARS',

# Update the test data with the new one-hot encoded columns.
X_test.update(one_hot_test)

# Check for any columns that are present in the training data but missing in th
# which might happen if the test data lacks certain categories.
missing_cols = set(X_train.columns) - set(X_test.columns)
for c in missing_cols:
    X_test[c] = 0  # Add these missing columns to the test data, initializing w

# Ensure the column order in the test data matches that of the training data fo
X_test = X_test[X_train.columns]

# Define a dictionary to rename the one-hot encoded columns for clarity, making
column_renaming = {'Material_0': 'Composite',
    'Material_1': 'Glass Ionomer Cement',
```

```
    'Material_2': 'Amalgam',
    'Baseline_Treatment_0': 'No initial intervention',
    'Baseline_Treatment_1': 'Repaired baseline',
    'Baseline_Treatment_2': 'Replaced baseline',
    'CARS_0': 'CARS No caries',
    'CARS_1': 'CARS Initial',
    'CARS_2': 'CARS Moderate/advanced',
    'FDI_adapt_0': 'FDI No adaptation',
    'FDI_adapt_1': 'FDI Initial adaptation',
    'FDI_adapt_2': 'FDI Moderate/advanced adaptation',
    'FDI_caries_0': 'FDI No caries',
    'FDI_caries_1': 'FDI Initial caries',
    'FDI_caries_2': 'FDI Moderate/advanced caries'}

# Rename the columns in both the training and test datasets according to the de
X_train.rename(columns=column_renaming, inplace=True)
X_test.rename(columns=column_renaming, inplace=True)

# Scale the numerical features in both training and test datasets to have mean
# This is crucial for models that are sensitive to the scale of input features.
scaler = StandardScaler()
X_train.loc[:, ['Age', 'Decayed', 'Filled', 'Missing']] = scaler.fit_transform(
X_test.loc[:, ['Age', 'Decayed', 'Filled', 'Missing']] = scaler.transform(X_tes

# Define which columns are considered categorical, excluding numerical columns
categorical_features = list(range(len(X_train.columns)))
for col in ['Age', 'Decayed', 'Filled', 'Missing']:  # Assuming these are your
    categorical_features.remove(X_train.columns.get_loc(col))
```

```
<ipython-input-7-1e795d60a953>:64: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs
  X_test.rename(columns=column_renaming, inplace=True)
```

```
import pandas as pd

# Define lists categorizing the types of variables in the dataset.
numeric_vars = ['Age', 'Decayed', 'Filled', 'Missing']  # Numeric variables
original_categorical_vars = ['Sex', 'Tooth type', 'Arch', 'N surfaces', 'Failure
# One-hot encoded variables, representing categories as separate binary columns.
one_hot_encoded_vars = ['Composite', 'Glass Ionomer Cement', 'Amalgam', 'No init
                        'Repaired baseline', 'Replaced baseline', 'CARS No carie
                        'CARS Moderate/advanced', 'FDI No adaptation', 'FDI Init
                        'FDI Moderate/advanced adaptation', 'FDI No caries', 'FD
                        'FDI Moderate/advanced caries']
```

```
def descriptive_statistics(X_train, y_train, X_test, y_test):
    # Merge the feature DataFrame (X) and target variable Series (y) for both tr
    # This facilitates combined operations for descriptive statistics.
    train_data = pd.concat([X_train, y_train], axis=1)
    test_data = pd.concat([X_test, y_test], axis=1)

    # Print a heading and then descriptive statistics (count, mean, std, min, qu
    print("Descriptive Statistics for Numeric Variables:")
    print("\nTraining Set:")
    print(train_data[numeric_vars].describe())
    print("\nTest Set:")
    print(test_data[numeric_vars].describe())

    # Initialize a dictionary to hold statistics for categorical variables.
    stats = {}
    # Calculate and store counts and percentages for original (non-encoded) cate
    for var in original_categorical_vars:
        stats[var] = {
            "Training Set": {
                "Count": train_data[var].value_counts().to_dict(),
                "Percentage": (train_data[var].value_counts(normalize=True) * 10
            },
            "Test Set": {
                "Count": test_data[var].value_counts().to_dict(),
                "Percentage": (test_data[var].value_counts(normalize=True) * 100
            }
        }

    # Handle one-hot encoded variables by identifying all columns that match the
    # Then calculate counts and percentages for these as well.
    for var in one_hot_encoded_vars:
        encoded_columns = [col for col in train_data if col.startswith(var)]
        for col in encoded_columns:
            stats[col] = {
                "Training Set": {
                    "Count": train_data[col].value_counts().to_dict(),
                    "Percentage": (train_data[col].value_counts(normalize=True)
                },
                "Test Set": {
                    "Count": test_data[col].value_counts().to_dict(),
                    "Percentage": (test_data[col].value_counts(normalize=True) *
                }
            }

    # Print the calculated statistics for each categorical variable, both origin
    for var, data in stats.items():
        print(f"\n{var} Statistics:")
        for dataset, values in data.items():
            print(f"\n{dataset}:")
```

```
            print(f"\n{dataset}:")
            for metric, metric_values in values.items():
                print(f"{metric}: {metric_values}")

# Call the function, passing the training and test datasets (features and target
descriptive_statistics(X_train, y_train, X_test, y_test)
```

```
FDI No adaptation Statistics:

Training Set:
Count: {True: 333, False: 174}
Percentage: {True: 65.68047337278107, False: 34.319526627218934}

Test Set:
Count: {1: 82, 0: 48}
Percentage: {1: 63.07692307692307, 0: 36.92307692307693}

FDI Initial adaptation Statistics:

Training Set:
Count: {False: 352, True: 155}
Percentage: {False: 69.42800788954635, True: 30.57199211045365}

Test Set:
Count: {0: 87, 1: 43}
Percentage: {0: 66.92307692307692, 1: 33.07692307692307}

FDI Moderate/advanced adaptation Statistics:

Training Set:
Count: {False: 488, True: 19}
Percentage: {False: 96.25246548323472, True: 3.7475345167652856}

Test Set:
Count: {0: 125, 1: 5}
Percentage: {0: 96.15384615384616, 1: 3.8461538461538463}

FDI No caries Statistics:

Training Set:
Count: {True: 401, False: 106}
Percentage: {True: 79.09270216962526, False: 20.907297830374755}

Test Set:
Count: {1: 97, 0: 33}
Percentage: {1: 74.61538461538461, 0: 25.384615384615383}

FDI Initial caries Statistics:

Training Set:
Count: {False: 409, True: 98}
Percentage: {False: 80.6706114398422, True: 19.32938856015779}
```

```
    Test Set:
    Count: {0: 100, 1: 30}
    Percentage: {0: 76.92307692307693, 1: 23.076923076923077}

    FDI Moderate/advanced caries Statistics:

    Training Set:
    Count: {False: 499, True: 8}
    Percentage: {False: 98.42209072978304, True: 1.5779092702169626}

    Test Set:
    Count: {0: 127, 1: 3}
    Percentage: {0: 97.6923076923077, 1: 2.307692307692308}
```

```
# Define custom metrics
def sensitivity(y_true, y_pred):
    tn, fp, fn, tp = confusion_matrix(y_true, y_pred).ravel()
    return tp / (tp + fn)

def specificity(y_true, y_pred):
    tn, fp, fn, tp = confusion_matrix(y_true, y_pred).ravel()
    return tn / (tn + fp)


import pandas as pd
import numpy as np
import shap
import sys
import tensorflow as tf
import matplotlib.pyplot as plt
import random
import seaborn as sns
from sklearn.model_selection import cross_val_score
from sklearn.calibration import CalibratedClassifierCV
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from sklearn.model_selection import cross_validate, StratifiedKFold, GridSearch
from sklearn.metrics import make_scorer, accuracy_score, roc_auc_score, f1_scor
from sklearn.preprocessing import StandardScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, BatchNormalization
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau, Learni
from tensorflow.keras.regularizers import l2
from scipy import stats


def evaluate_model(model, name, grid, X_train, y_train, X_test, y_test, cv, scor
    print(f"\nEvaluating {name} with seed {seed}...")
```

```
inner_cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=seed)
outer_cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=seed)

clf = GridSearchCV(model, grid, cv=inner_cv, scoring='roc_auc')
nested_scores = cross_validate(clf, X=X_train, y=y_train, cv=outer_cv, scori

clf.fit(X_train, y_train)
best_model = clf.best_estimator_
best_params = clf.best_params_
print(f"Best parameters for {name}: {best_params}")

calibrated_clf = CalibratedClassifierCV(estimator=best_model, method='sigmoi
calibrated_clf.fit(X_train, y_train)

y_probs = calibrated_clf.predict_proba(X_test)[:, 1]

# Calculate ROC curve and AUC
fpr, tpr, thresholds = roc_curve(y_test, y_probs)
roc_auc = auc(fpr, tpr)

print("\n--- ROC Data for Copying ---")
print("FPR =", fpr.tolist())
print("TPR =", tpr.tolist())
print("AUC =", roc_auc)
print("--- End of ROC Data ---\n")

# --- Calculate Training Metrics ---
y_train_pred = best_model.predict(X_train)
y_train_probs = best_model.predict_proba(X_train)[:, 1]
train_acc = accuracy_score(y_train, y_train_pred)
train_sens = sensitivity(y_train, y_train_pred)
train_spec = specificity(y_train, y_train_pred)
train_f1 = f1_score(y_train, y_train_pred)
train_roc_auc = roc_auc_score(y_train, y_train_probs)

print(f"Training - Accuracy: {train_acc:.3f}, Sensitivity: {train_sens:.3f},
      f"Specificity: {train_spec:.3f}, F1: {train_f1:.3f}, ROC AUC: {train_r

# --- Calculate Test Metrics for the manually set threshold ---
y_pred_manual = (y_probs >= manual_threshold).astype(int)
manual_acc = accuracy_score(y_test, y_pred_manual)
manual_sens = sensitivity(y_test, y_pred_manual)
manual_spec = specificity(y_test, y_pred_manual)
manual_f1 = f1_score(y_test, y_pred_manual)
manual_roc_auc = roc_auc_score(y_test, y_probs)

print(f"\nTest Metrics for manual threshold {manual_threshold}:")
print(f"Accuracy: {manual_acc:.3f}, Sensitivity: {manual_sens:.3f}, "
```

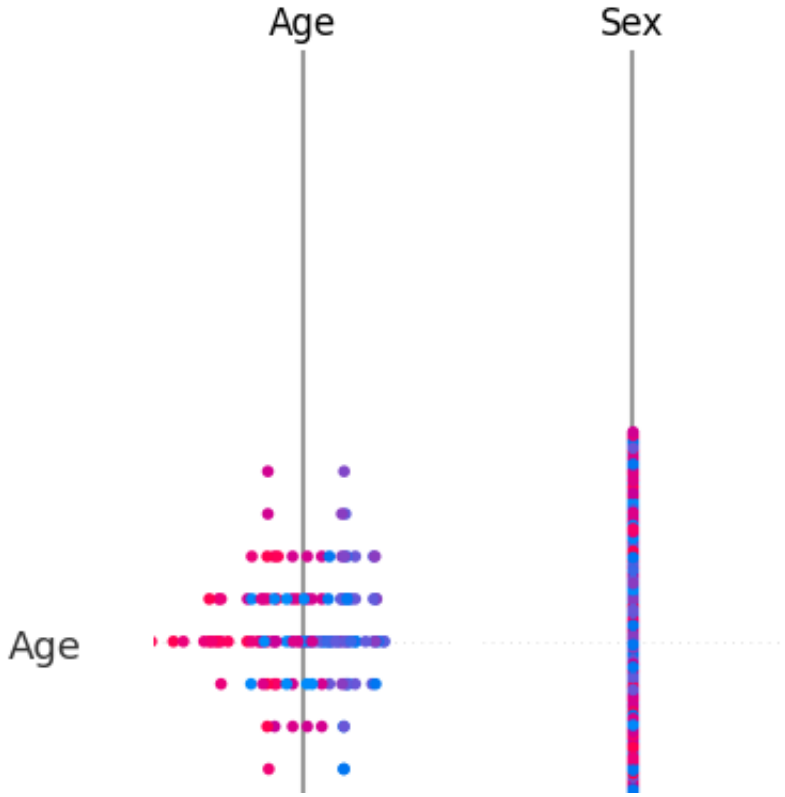
```
        f"Specificity: {manual_spec:.3f}, F1: {manual_f1:.3f}, ROC AUC: {manua

    # --- Evaluate metrics across a range of thresholds ---
    threshold_metrics = {}
    for threshold in threshold_list:
        y_pred_threshold = (y_probs >= threshold).astype(int)
        threshold_acc = accuracy_score(y_test, y_pred_threshold)
        threshold_sens = sensitivity(y_test, y_pred_threshold)
        threshold_spec = specificity(y_test, y_pred_threshold)
        threshold_f1 = f1_score(y_test, y_pred_threshold)
        threshold_metrics[threshold] = {
            'Accuracy': threshold_acc,
            'Sensitivity': threshold_sens,
            'Specificity': threshold_spec,
            'F1': threshold_f1,
            'ROC AUC': manual_roc_auc  # Same ROC AUC regardless of threshold
        }
    for threshold, metrics in threshold_metrics.items():
        print(f"Threshold: {threshold:.2f}, Metrics: {metrics}")

    calculate_and_plot_shap(best_model, X_train, X_test, name)

    # Prepare dictionary of test metrics for aggregation
    test_metrics = {
        "accuracy": manual_acc,
        "sensitivity": manual_sens,
        "specificity": manual_spec,
        "f1": manual_f1,
        "roc_auc": manual_roc_auc
    }

    return best_model, manual_threshold, best_params, nested_scores, calibrated_

def calculate_and_plot_shap(model, X_train, X_test, model_name):
    if isinstance(model, DecisionTreeClassifier):
        explainer = shap.TreeExplainer(model)
    else:
        explainer = shap.KernelExplainer(model.predict_proba, X_train.sample(100
    shap_values = explainer.shap_values(X_test)
    print(f"SHAP Summary for {model_name}")
    shap.summary_plot(shap_values, X_test, max_display=10)

def plot_confusion_matrix(y_true, y_pred):
    matrix = confusion_matrix(y_true, y_pred)
    sns.heatmap(matrix, annot=True, fmt='d', cmap='Blues',
                xticklabels=['Predicted Success', 'Predicted Failure'],
                yticklabels=['Actual Success', 'Actual Failure'])
    plt.title('Confusion Matrix')
```

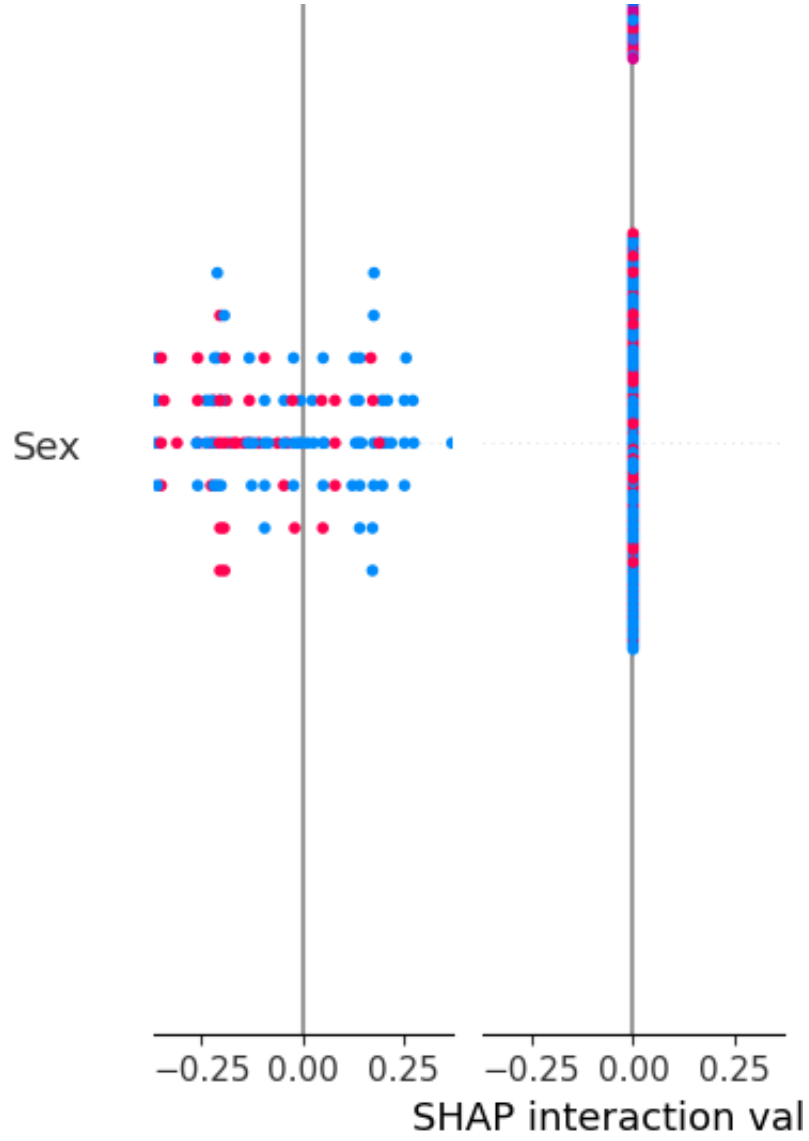
```
    plt.show()

def plot_roc_curve(y_true, y_probs):
    fpr, tpr, thresholds = roc_curve(y_true, y_probs)
    roc_auc = auc(fpr, tpr)
    plt.figure()
    plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (area = {roc_
    plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver Operating Characteristic')
    plt.legend(loc="lower right")
    plt.show()

def evaluate_decision_tree(X_train, y_train, X_test, y_test, cv, scoring, manual
    model = DecisionTreeClassifier(random_state=seed)
    grid = {
        'max_depth': [6],
        'criterion': ['gini'],
        'min_samples_split': [4],
        'min_samples_leaf': [8],
        'ccp_alpha': [0.001]
    }
    return evaluate_model(model, "Decision Tree", grid, X_train, y_train, X_test

def main(X_train, y_train, X_test, y_test):
    cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=10, random_state=42)
    scoring = {
        'accuracy': make_scorer(accuracy_score),
        'sensitivity': make_scorer(sensitivity),
        'specificity': make_scorer(specificity),
        'f1': make_scorer(f1_score),
        'roc_auc': make_scorer(roc_auc_score)
    }
    manual_threshold = 0.5
    threshold_list = np.arange(0.1, 1.05, 0.05)

    aggregated_metrics = []

    # Loop over seeds
    for seed in range(40, 50):
        print(f"\nRunning evaluation with seed {seed}")
        (best_model, manual_threshold, best_params, nested_scores,
         calibrated_clf, threshold_metrics, test_metrics) = evaluate_decision_tr
            X_train, y_train, X_test, y_test, cv, scoring, manual_threshold, thr
        )
```

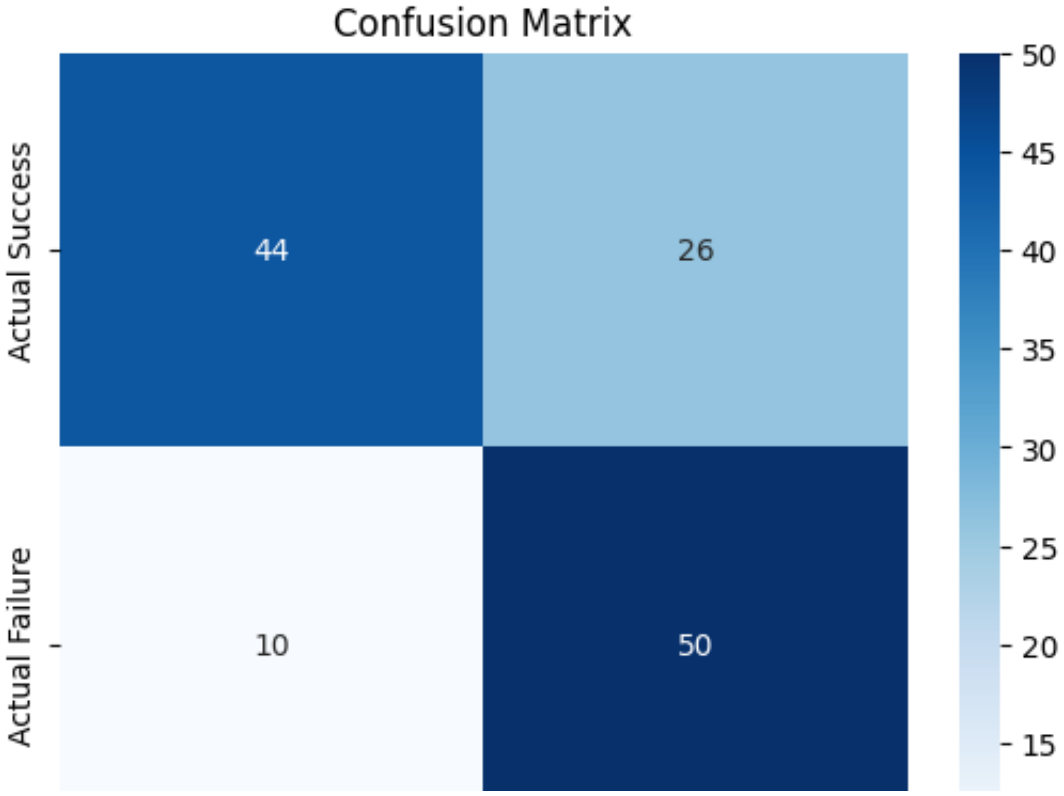
```
        # Use calibrated classifier for plotting
        y_probs = calibrated_clf.predict_proba(X_test)[:, 1]
        y_pred_manual = (y_probs >= manual_threshold).astype(int)

        plot_confusion_matrix(y_test, y_pred_manual)
        plot_roc_curve(y_test, y_probs)

        aggregated_metrics.append(test_metrics)

    # Aggregate results across seeds
    results_df = pd.DataFrame(aggregated_metrics)
    n = len(results_df)
    print("\nAggregated Test Set Metrics Across Seeds:")
    print(results_df)

    # Compute mean, standard error, and 95% confidence interval for each metric
    def summarize_metric(metric_values):
        mean_val = metric_values.mean()
        std_val = metric_values.std(ddof=1)
        se = std_val / np.sqrt(n)
        t_crit = stats.t.ppf(0.975, df=n - 1)
        ci_lower = mean_val - t_crit * se
        ci_upper = mean_val + t_crit * se
        return mean_val, se, (ci_lower, ci_upper)

    metrics_summary = {}
    for metric in results_df.columns:
        mean_val, se, ci = summarize_metric(results_df[metric])
        metrics_summary[metric] = {
            "Mean": mean_val,
            "Standard Error": se,
            "95% CI": ci
        }

    print("\nSummary of Test Set Metrics (Mean, Standard Error, 95% Confidence I
    for metric, summary in metrics_summary.items():
        print(f"{metric.capitalize()}: Mean = {summary['Mean']:.3f}, SE = {summa
              f"95% CI = [{summary['95% CI'][0]:.3f}, {summary['95% CI'][1]:.3f}

# RUN THE MAIN FUNCTION (Ensure X_train, y_train, X_test, y_test are defined)
if __name__ == '__main__':
    main(X_train, y_train, X_test, y_test)
```

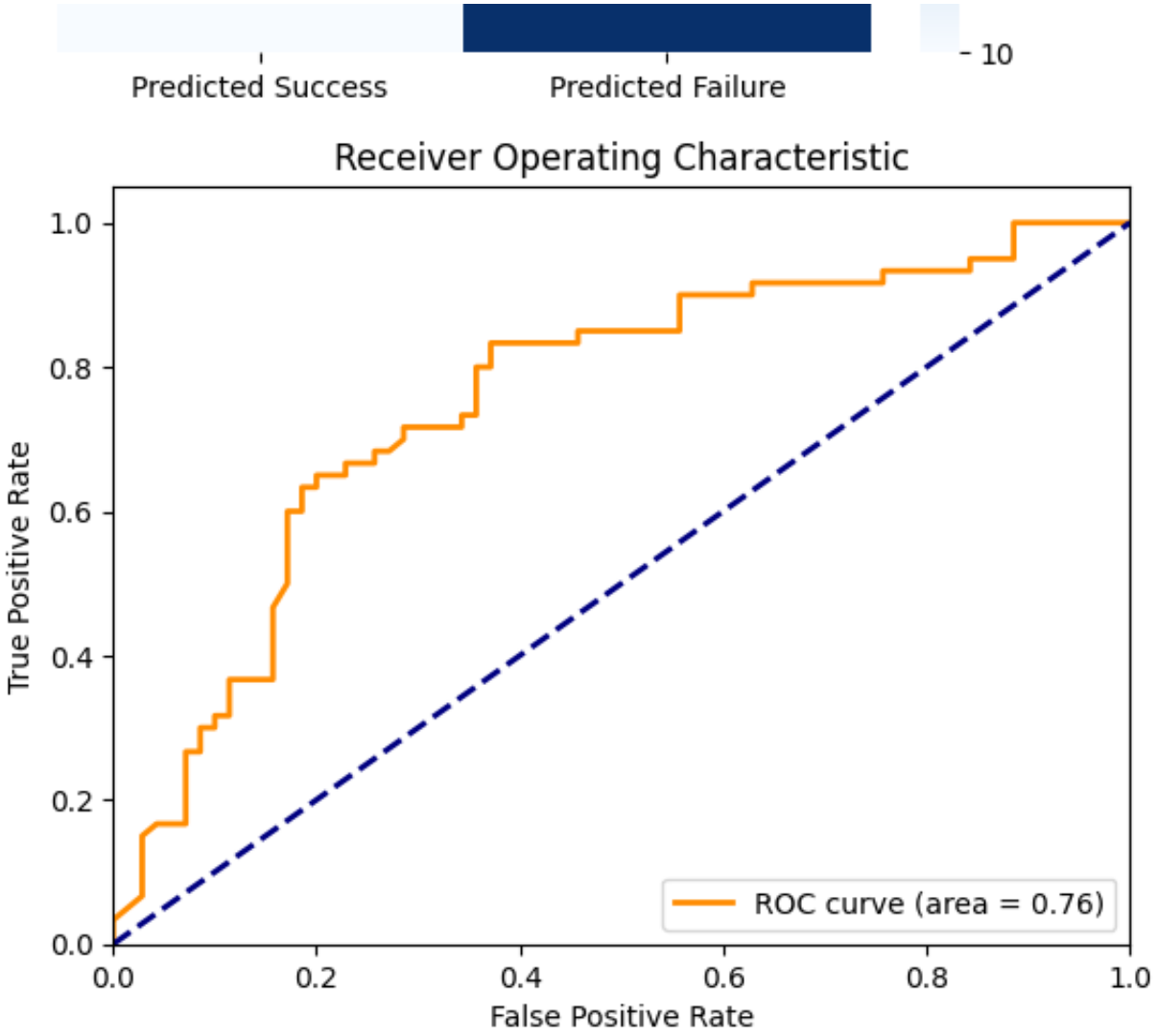
```
Running evaluation with seed 40

Evaluating Decision Tree with seed 40...
```

```
Best parameters for Decision Tree: {'ccp_alpha': 0.001, 'criterion': 'gini'

--- ROC Data for Copying ---
FPR = [0.0, 0.0, 0.0, 0.02857142857142857, 0.02857142857142857, 0.028571428
TPR = [0.0, 0.016666666666666666, 0.03333333333333333, 0.06666666666666667,
AUC = 0.7550000000000001
--- End of ROC Data ---

Training - Accuracy: 0.728, Sensitivity: 0.805, Specificity: 0.652, F1: 0.7

Test Metrics for manual threshold 0.5:
Accuracy: 0.723, Sensitivity: 0.833, Specificity: 0.629, F1: 0.735, ROC AUC
Threshold: 0.10, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.15, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.20, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.25, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.30, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.35, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.40, Metrics: {'Accuracy': 0.49230769230769234, 'Sensitivity':
Threshold: 0.45, Metrics: {'Accuracy': 0.5538461538461539, 'Sensitivity': 0
Threshold: 0.50, Metrics: {'Accuracy': 0.7230769230769231, 'Sensitivity': 0
Threshold: 0.55, Metrics: {'Accuracy': 0.7230769230769231, 'Sensitivity': 0
Threshold: 0.60, Metrics: {'Accuracy': 0.5538461538461539, 'Sensitivity': 0
Threshold: 0.65, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.70, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.75, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.80, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.85, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.90, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.95, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 1.00, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
SHAP Summary for Decision Tree
```



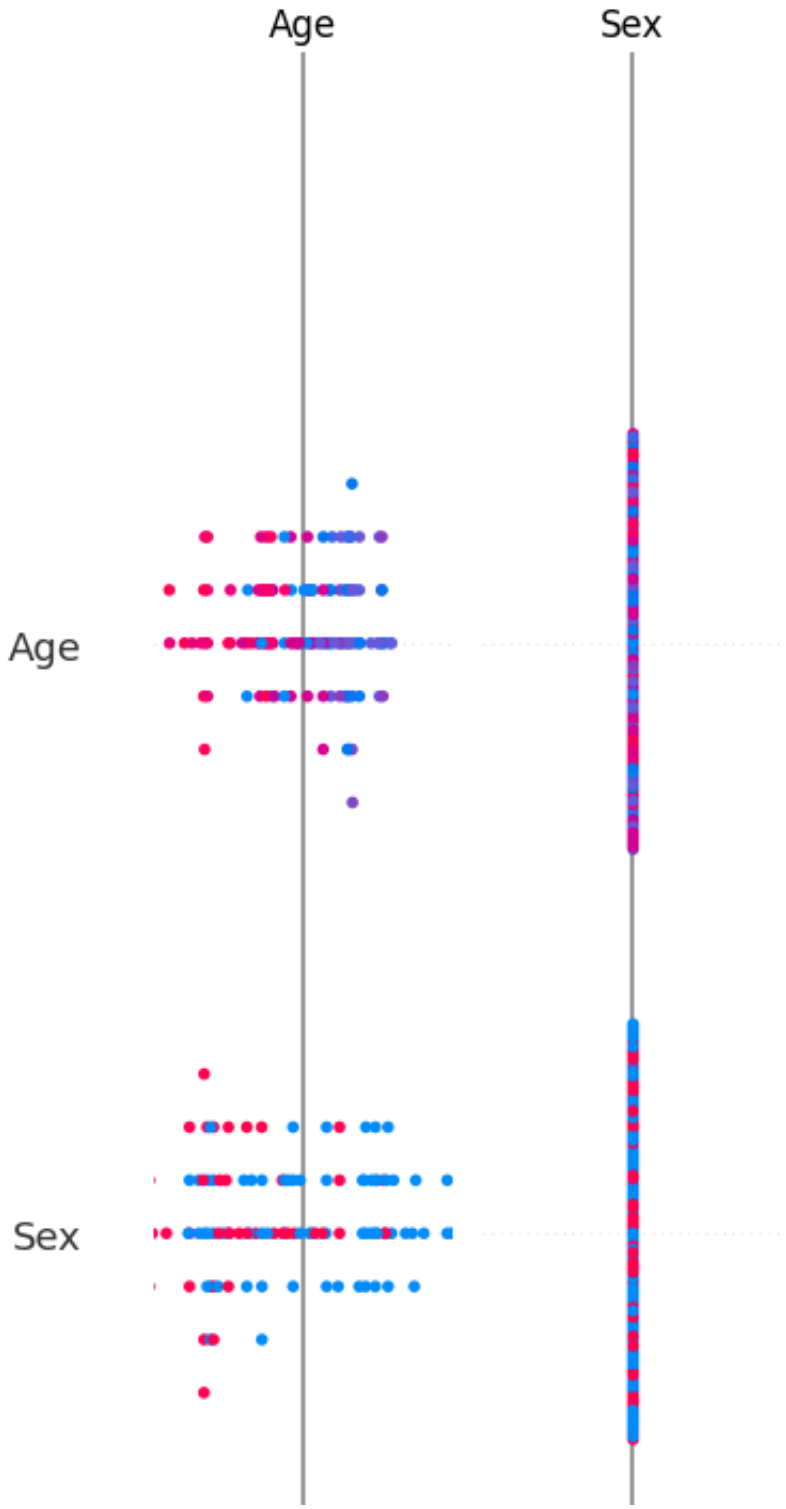

Sex
−0.25 0.00 0.25
−0.25 0.00 0.25
SHAP interaction val

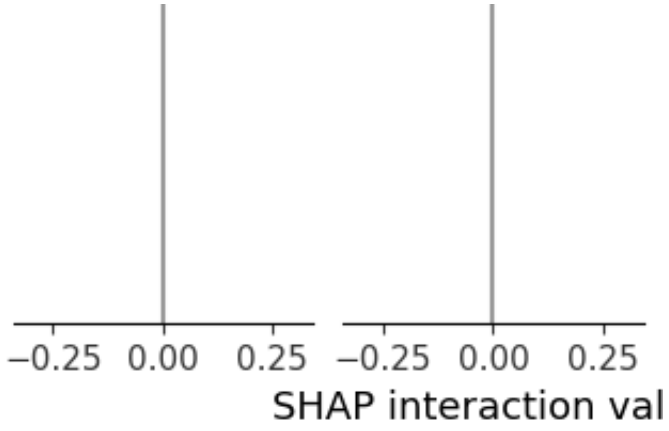

Confusion Matrix
Actual Success
Actual Failure
44
26
10
50
50
45
40
35
30
25
20
15

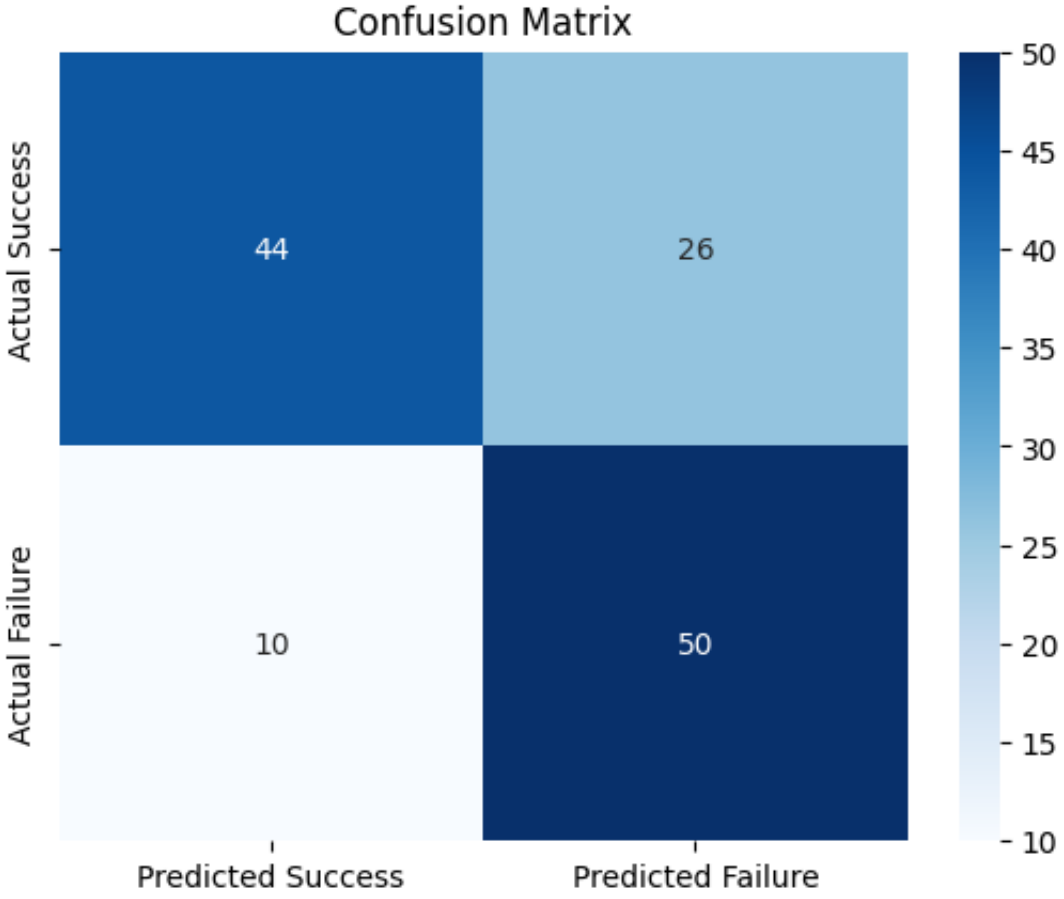


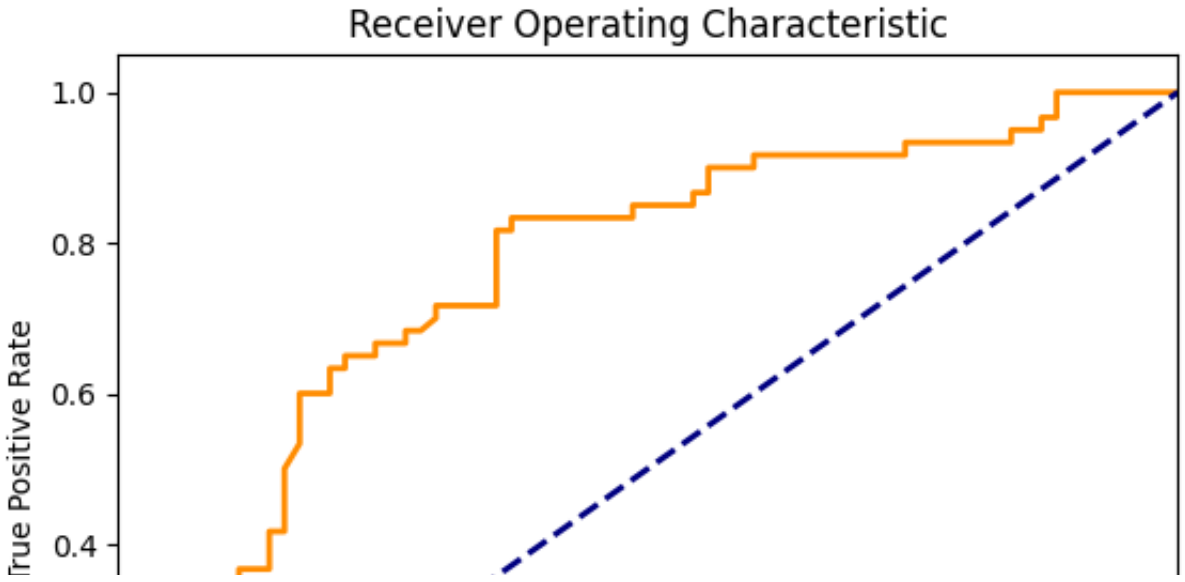

```
Running evaluation with seed 41

Evaluating Decision Tree with seed 41...
Best parameters for Decision Tree: {'ccp_alpha': 0.001, 'criterion': 'gini'

--- ROC Data for Copying ---
FPR = [0.0, 0.0, 0.0, 0.02857142857142857, 0.02857142857142857, 0.028571428
TPR = [0.0, 0.016666666666666666, 0.03333333333333333, 0.06666666666666667,
AUC = 0.7557142857142857
--- End of ROC Data ---

Training - Accuracy: 0.728, Sensitivity: 0.805, Specificity: 0.652, F1: 0.7

Test Metrics for manual threshold 0.5:
Accuracy: 0.723, Sensitivity: 0.833, Specificity: 0.629, F1: 0.735, ROC AUC
Threshold: 0.10, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.15, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.20, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.25, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.30, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.35, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.40, Metrics: {'Accuracy': 0.49230769230769234, 'Sensitivity':
Threshold: 0.45, Metrics: {'Accuracy': 0.5615384615384615, 'Sensitivity': 0
```

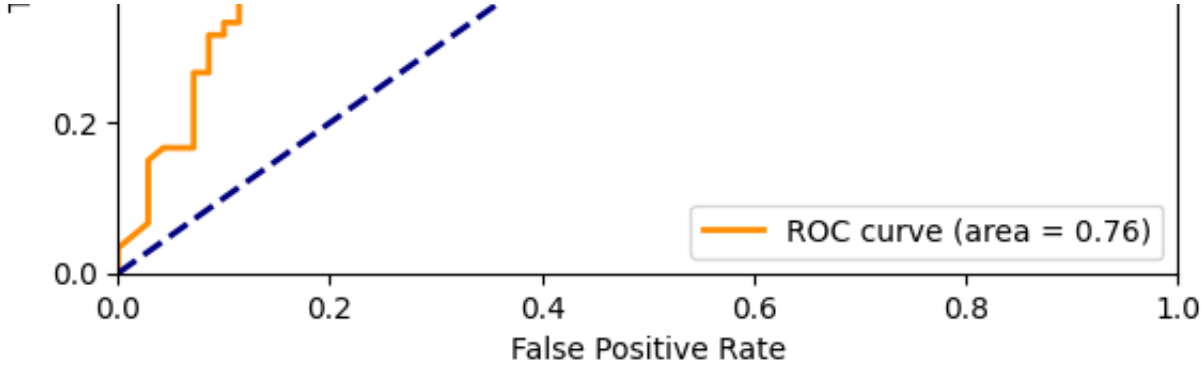
```
Threshold: 0.50, Metrics: {'Accuracy': 0.7230769230769231, 'Sensitivity': 0
Threshold: 0.55, Metrics: {'Accuracy': 0.7076923076923077, 'Sensitivity': 0
Threshold: 0.60, Metrics: {'Accuracy': 0.5538461538461539, 'Sensitivity': 0
Threshold: 0.65, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.70, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.75, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.80, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.85, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.90, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.95, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 1.00, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
SHAP Summary for Decision Tree
```



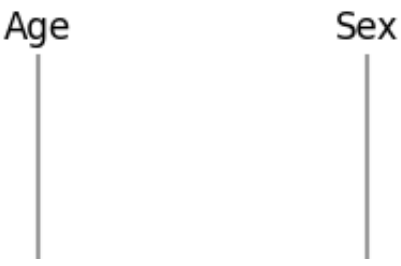

−0.25 0.00 0.25 −0.25 0.00 0.25
SHAP interaction val

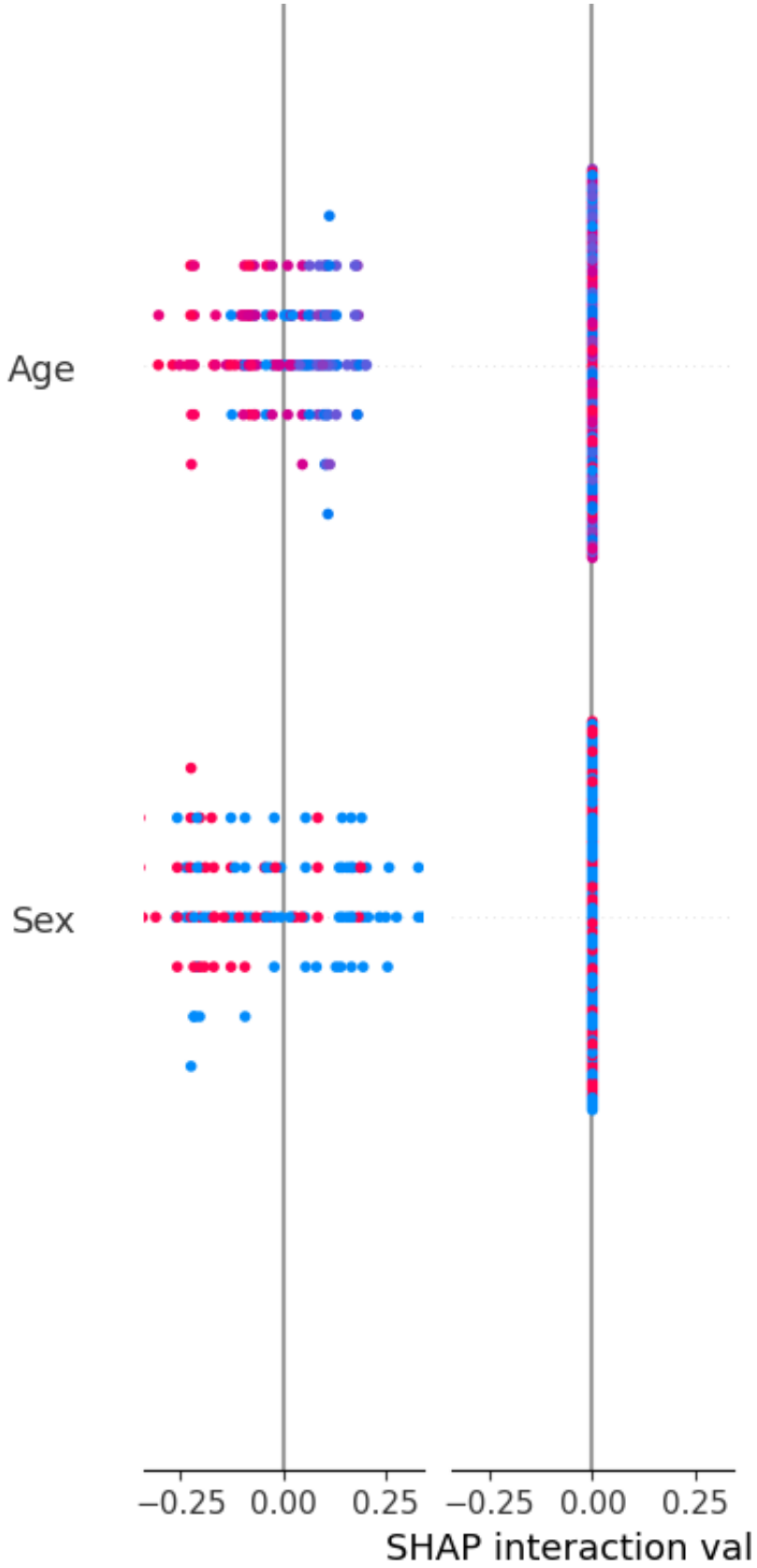

Confusion Matrix
Actual Success
Actual Failure
44
26
10
50
Predicted Success
Predicted Failure
50
45
40
35
30
25
20
15
10

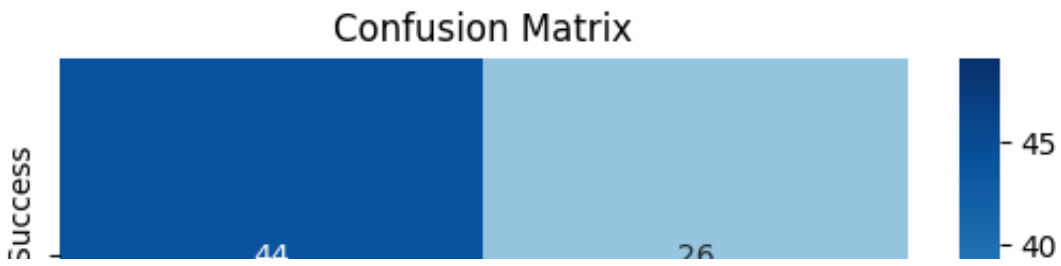

Receiver Operating Characteristic
True Positive Rate
1.0
0.8
0.6
0.4

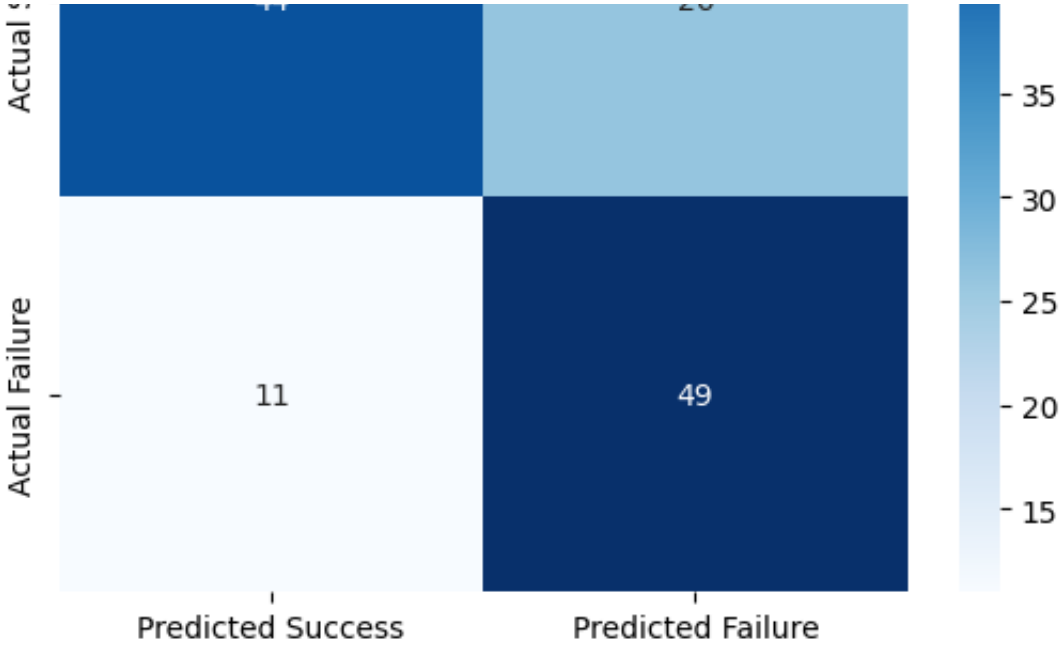


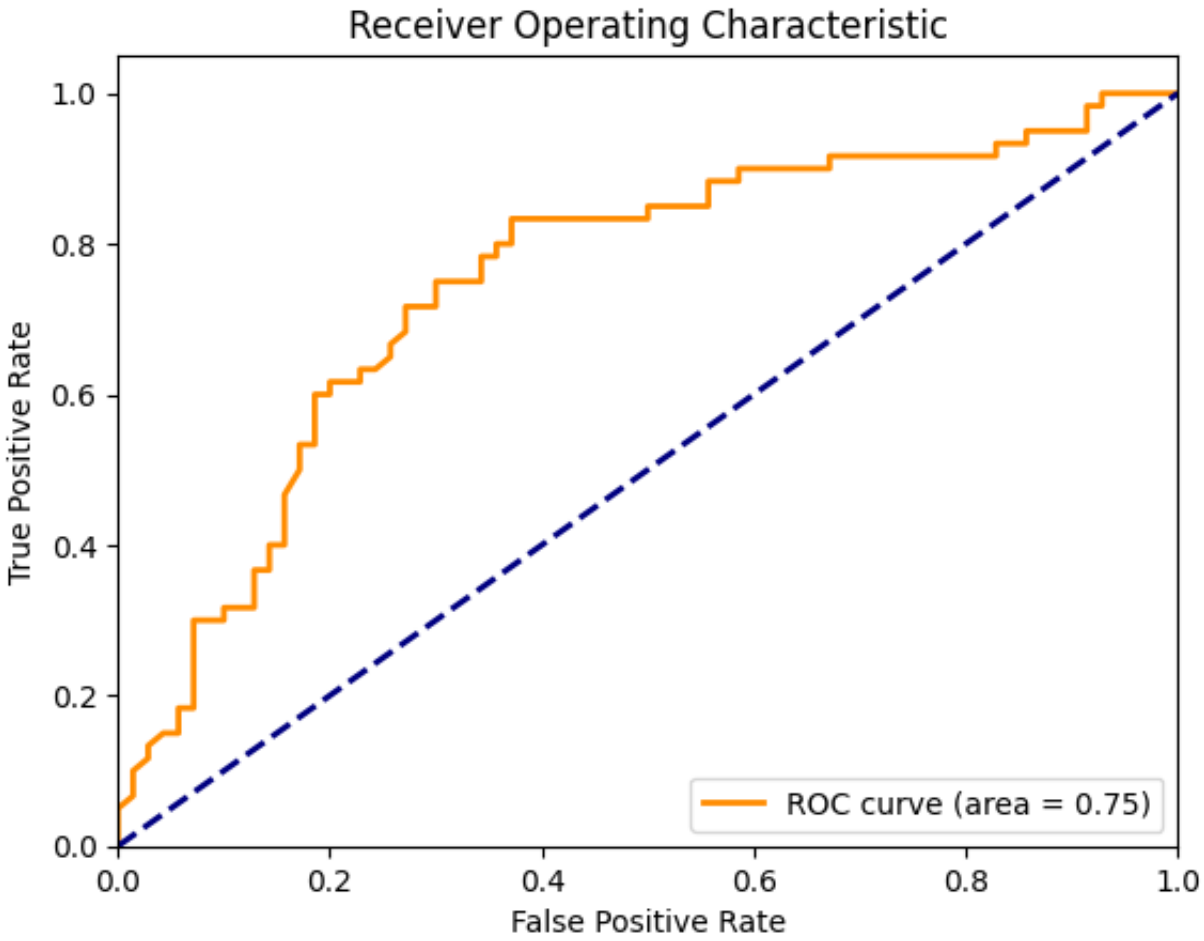

```
Running evaluation with seed 42

Evaluating Decision Tree with seed 42...
Best parameters for Decision Tree: {'ccp_alpha': 0.001, 'criterion': 'gini'

--- ROC Data for Copying ---
FPR = [0.0, 0.0, 0.0, 0.014285714285714285, 0.014285714285714285, 0.0285714
TPR = [0.0, 0.016666666666666666, 0.05, 0.06666666666666667, 0.1, 0.1166666
AUC = 0.7501190476190476
--- End of ROC Data ---

Training - Accuracy: 0.728, Sensitivity: 0.805, Specificity: 0.652, F1: 0.7

Test Metrics for manual threshold 0.5:
Accuracy: 0.715, Sensitivity: 0.817, Specificity: 0.629, F1: 0.726, ROC AUC
Threshold: 0.10, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.15, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.20, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.25, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.30, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.35, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.40, Metrics: {'Accuracy': 0.49230769230769234, 'Sensitivity':
Threshold: 0.45, Metrics: {'Accuracy': 0.5538461538461539, 'Sensitivity': 0
Threshold: 0.50, Metrics: {'Accuracy': 0.7153846153846154, 'Sensitivity': 0
Threshold: 0.55, Metrics: {'Accuracy': 0.6692307692307692, 'Sensitivity': 0
Threshold: 0.60, Metrics: {'Accuracy': 0.5461538461538461, 'Sensitivity': 0
Threshold: 0.65, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.70, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.75, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.80, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.85, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.90, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.95, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 1.00, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
SHAP Summary for Decision Tree
```




Age
Sex
−0.25
0.00
0.25
−0.25
0.00
0.25
SHAP interaction val

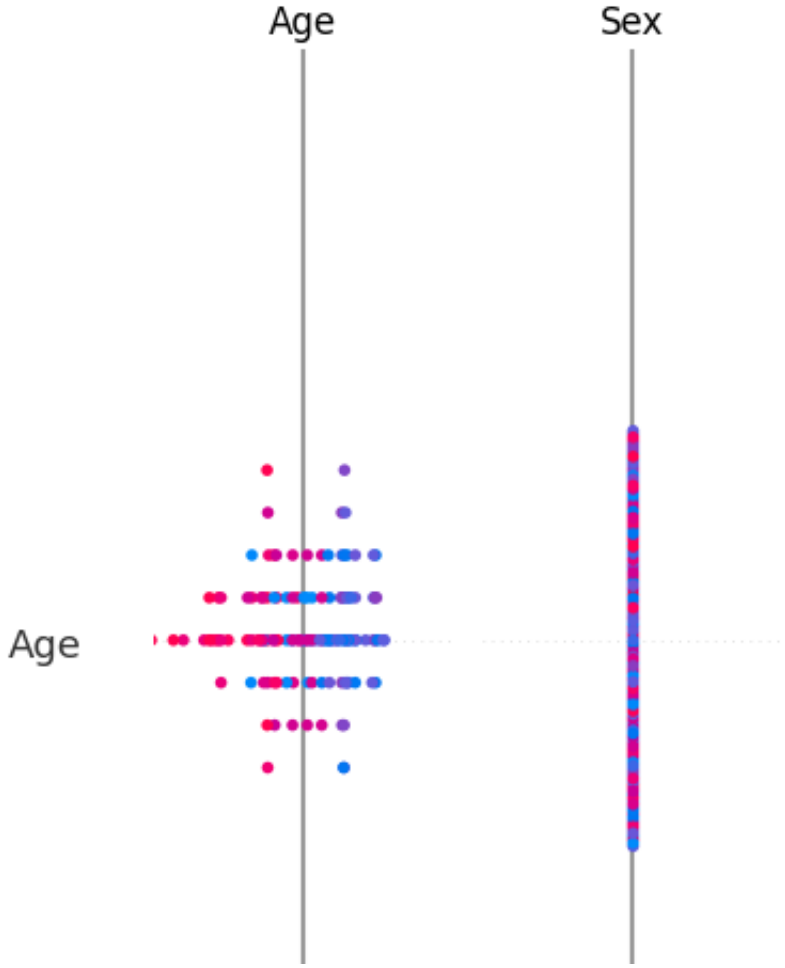

Confusion Matrix
Success
44
26
45
40

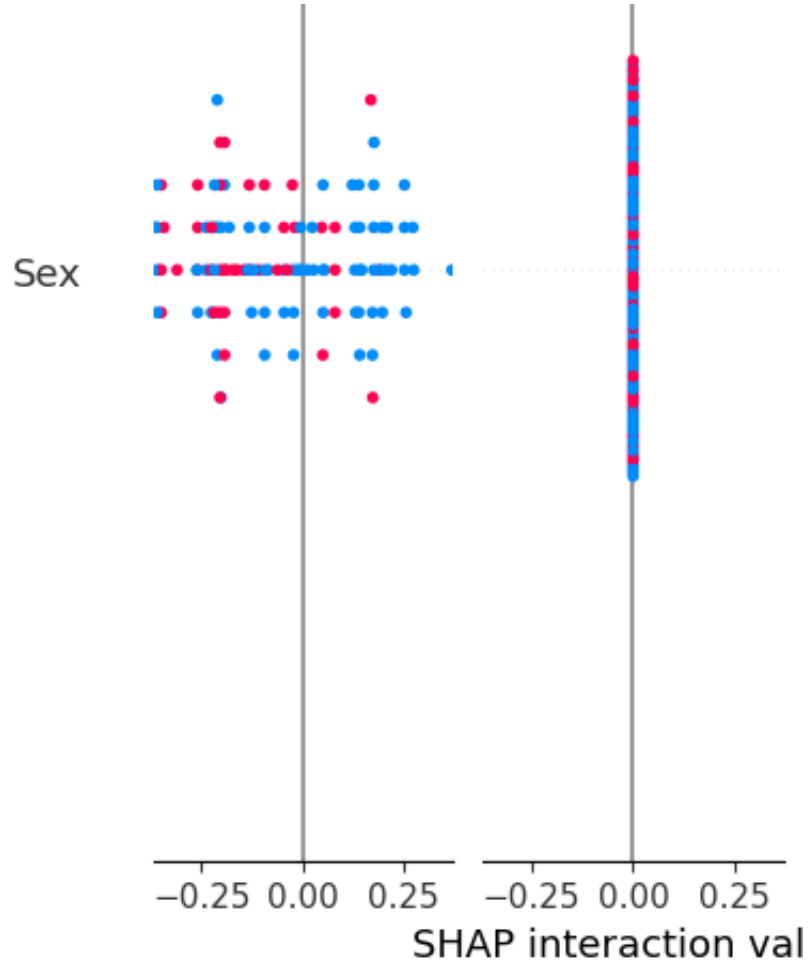


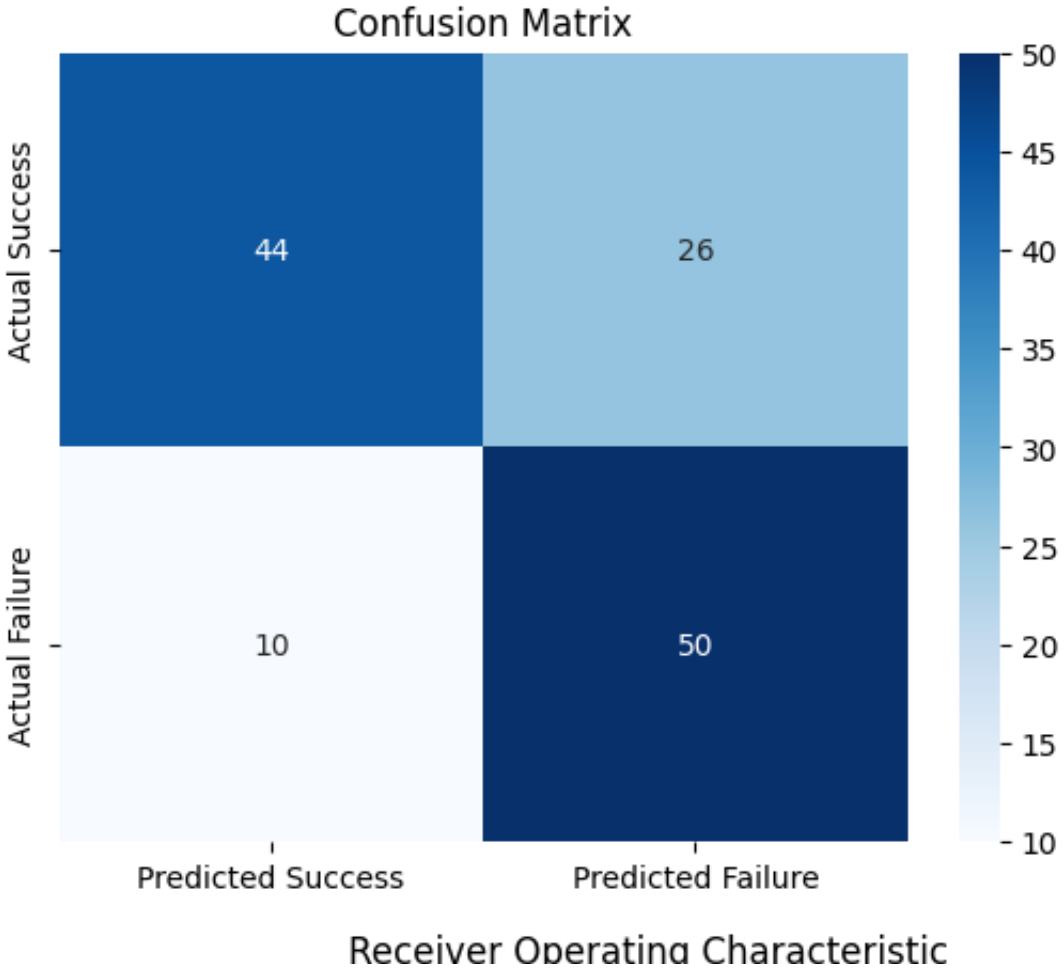



```
Running evaluation with seed 43

Evaluating Decision Tree with seed 43...
Best parameters for Decision Tree: {'ccp_alpha': 0.001, 'criterion': 'gini'

--- ROC Data for Copying ---
FPR = [0.0, 0.0, 0.0, 0.02857142857142857, 0.02857142857142857, 0.028571428
TPR = [0.0, 0.016666666666666666, 0.03333333333333333, 0.06666666666666667,
```

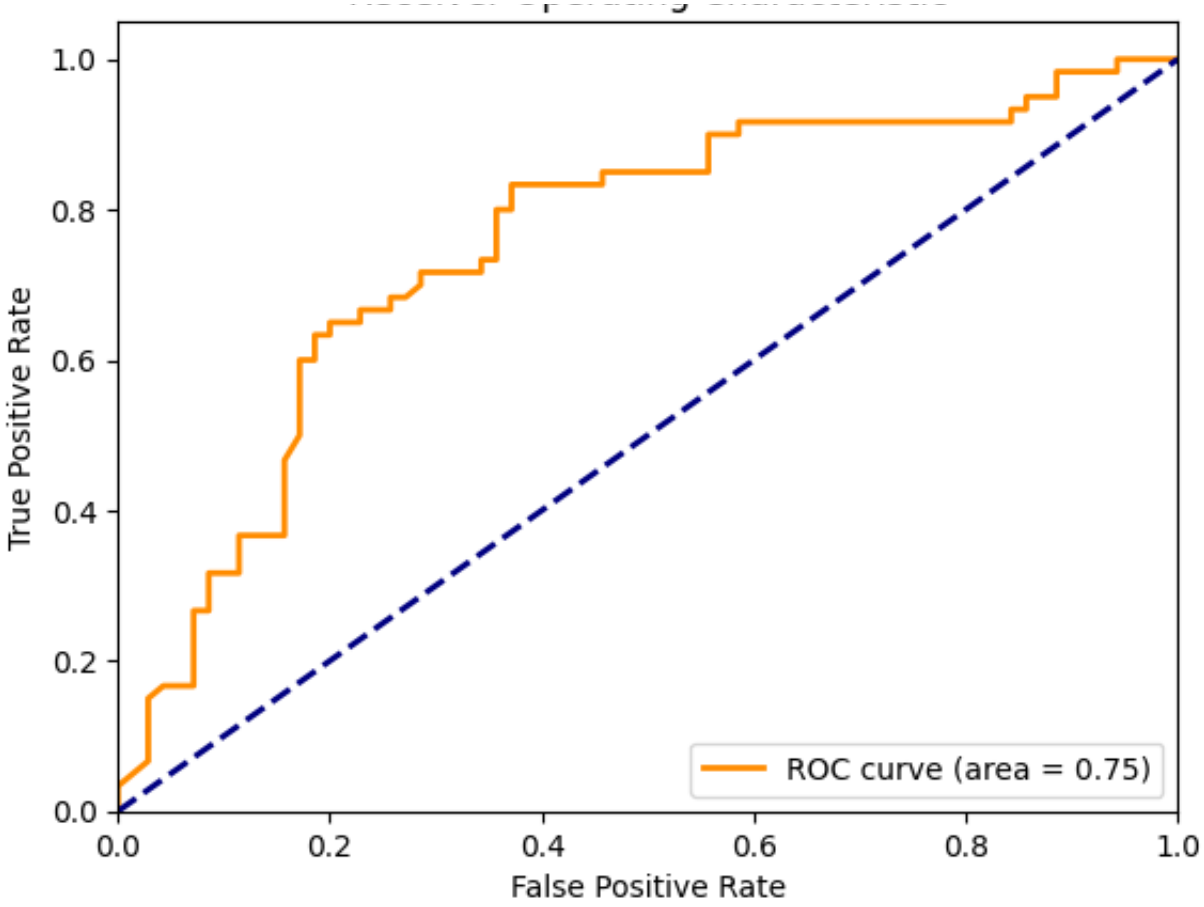
```
AUC = 0.7533333333333333
--- End of ROC Data ---

Training - Accuracy: 0.728, Sensitivity: 0.805, Specificity: 0.652, F1: 0.7

Test Metrics for manual threshold 0.5:
Accuracy: 0.723, Sensitivity: 0.833, Specificity: 0.629, F1: 0.735, ROC AUC
Threshold: 0.10, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.15, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.20, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.25, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.30, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.35, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.40, Metrics: {'Accuracy': 0.4846153846153846, 'Sensitivity': 0
Threshold: 0.45, Metrics: {'Accuracy': 0.5769230769230769, 'Sensitivity': 0
Threshold: 0.50, Metrics: {'Accuracy': 0.7230769230769231, 'Sensitivity': 0
Threshold: 0.55, Metrics: {'Accuracy': 0.7230769230769231, 'Sensitivity': 0
Threshold: 0.60, Metrics: {'Accuracy': 0.5538461538461539, 'Sensitivity': 0
Threshold: 0.65, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.70, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.75, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.80, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.85, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.90, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.95, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 1.00, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
SHAP Summary for Decision Tree
```





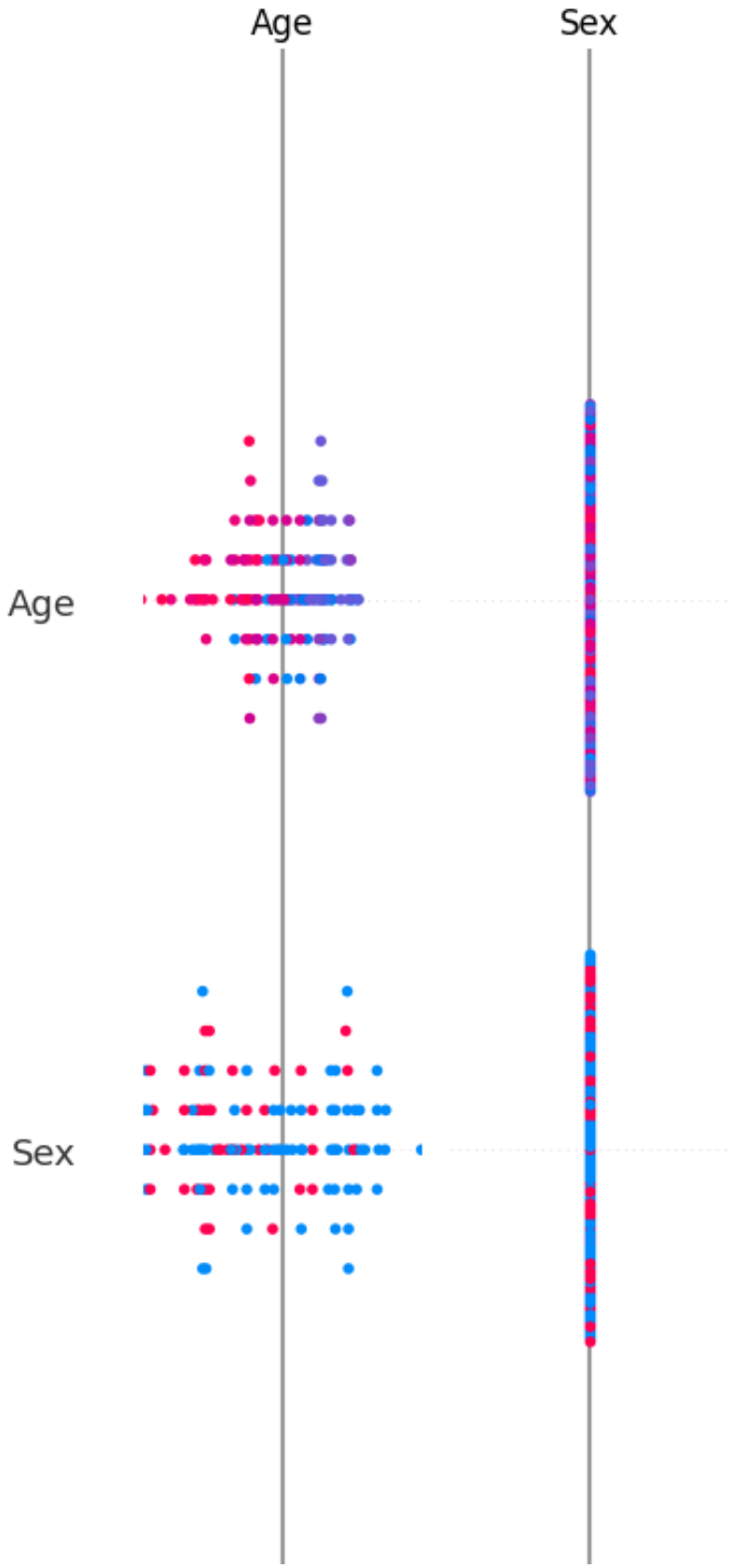


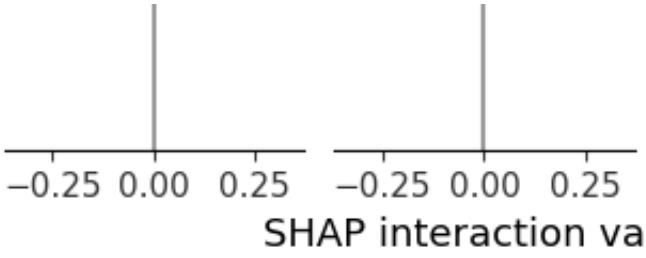
Receiver Operating Characteristic

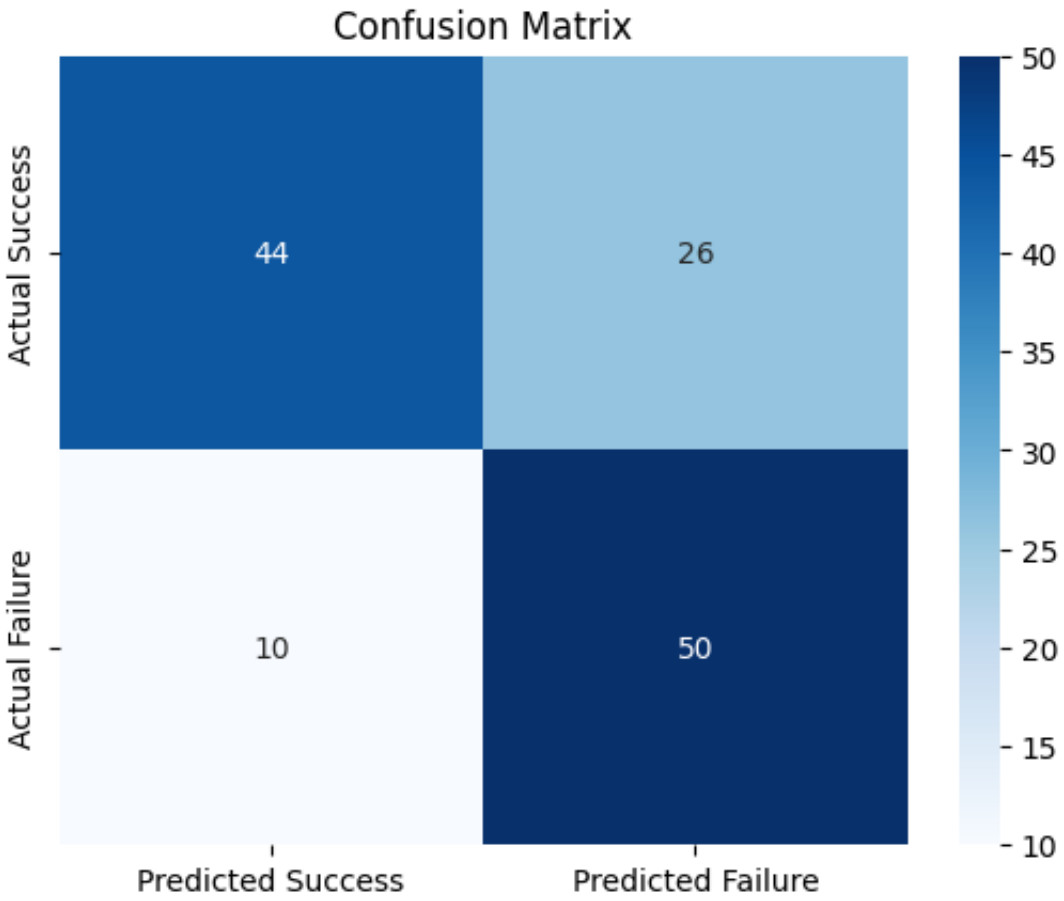


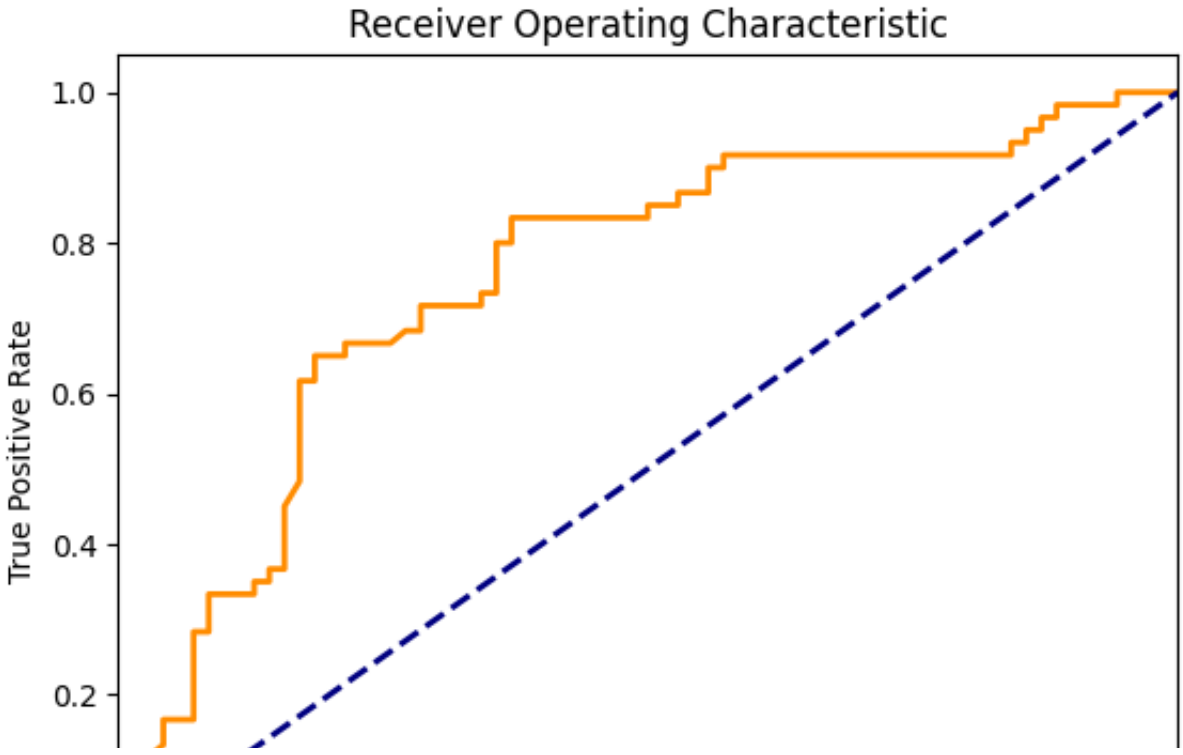

```
Running evaluation with seed 44

Evaluating Decision Tree with seed 44...
Best parameters for Decision Tree: {'ccp_alpha': 0.001, 'criterion': 'gini'

--- ROC Data for Copying ---
FPR = [0.0, 0.0, 0.0, 0.02857142857142857, 0.02857142857142857, 0.028571428
TPR = [0.0, 0.016666666666666666, 0.03333333333333333, 0.06666666666666667,
AUC = 0.7533333333333332
--- End of ROC Data ---

Training - Accuracy: 0.728, Sensitivity: 0.805, Specificity: 0.652, F1: 0.7

Test Metrics for manual threshold 0.5:
Accuracy: 0.723, Sensitivity: 0.833, Specificity: 0.629, F1: 0.735, ROC AUC
Threshold: 0.10, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.15, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.20, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.25, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.30, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.35, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.40, Metrics: {'Accuracy': 0.4923076923076923, 'Sensitivity':
Threshold: 0.45, Metrics: {'Accuracy': 0.5769230769230769, 'Sensitivity': 0
Threshold: 0.50, Metrics: {'Accuracy': 0.7230769230769231, 'Sensitivity': 0
Threshold: 0.55, Metrics: {'Accuracy': 0.7307692307692307, 'Sensitivity': 0
Threshold: 0.60, Metrics: {'Accuracy': 0.5538461538461539, 'Sensitivity': 0
Threshold: 0.65, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.70, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
```

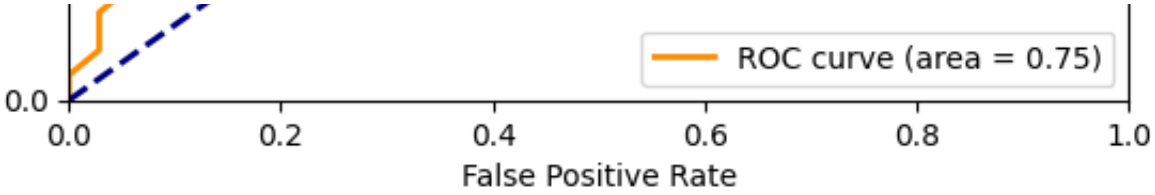
```
Threshold: 0.75, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.80, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.85, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.90, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.95, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 1.00, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
SHAP Summary for Decision Tree
```



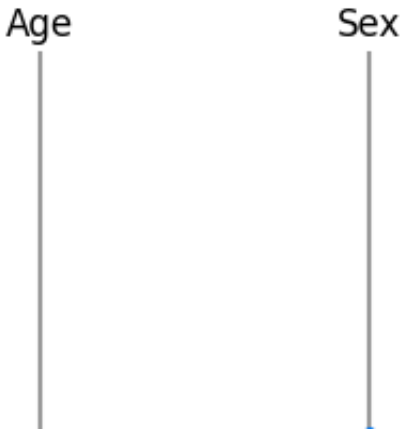

−0.25 0.00 0.25
−0.25 0.00 0.25
SHAP interaction val

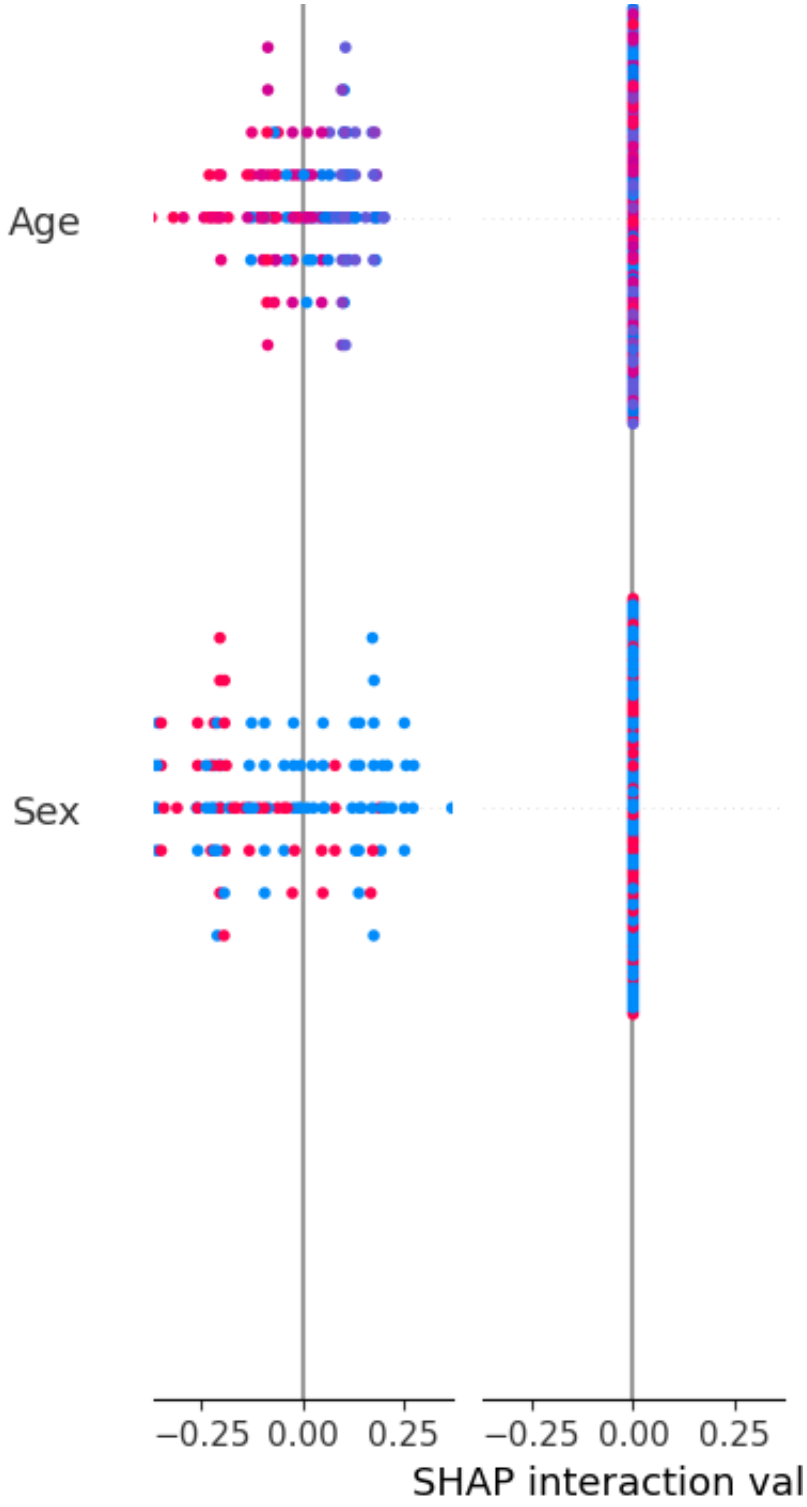

Confusion Matrix
Actual Success
Actual Failure
44
26
10
50
Predicted Success
Predicted Failure
50
45
40
35
30
25
20
15
10

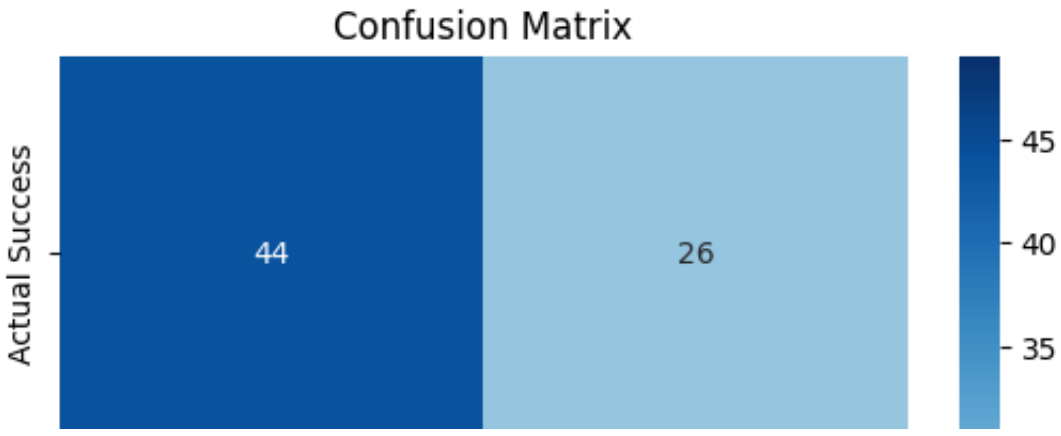

Receiver Operating Characteristic
True Positive Rate
1.0
0.8
0.6
0.4
0.2

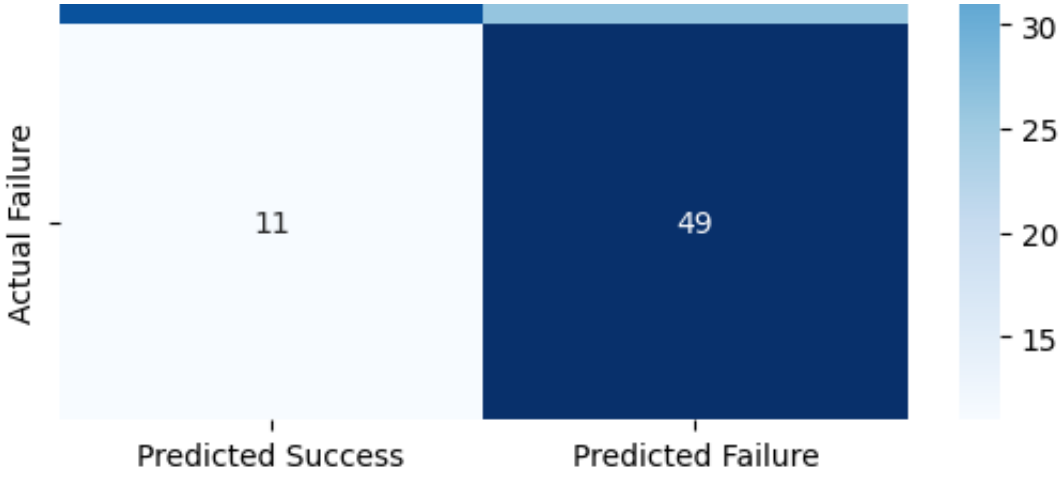


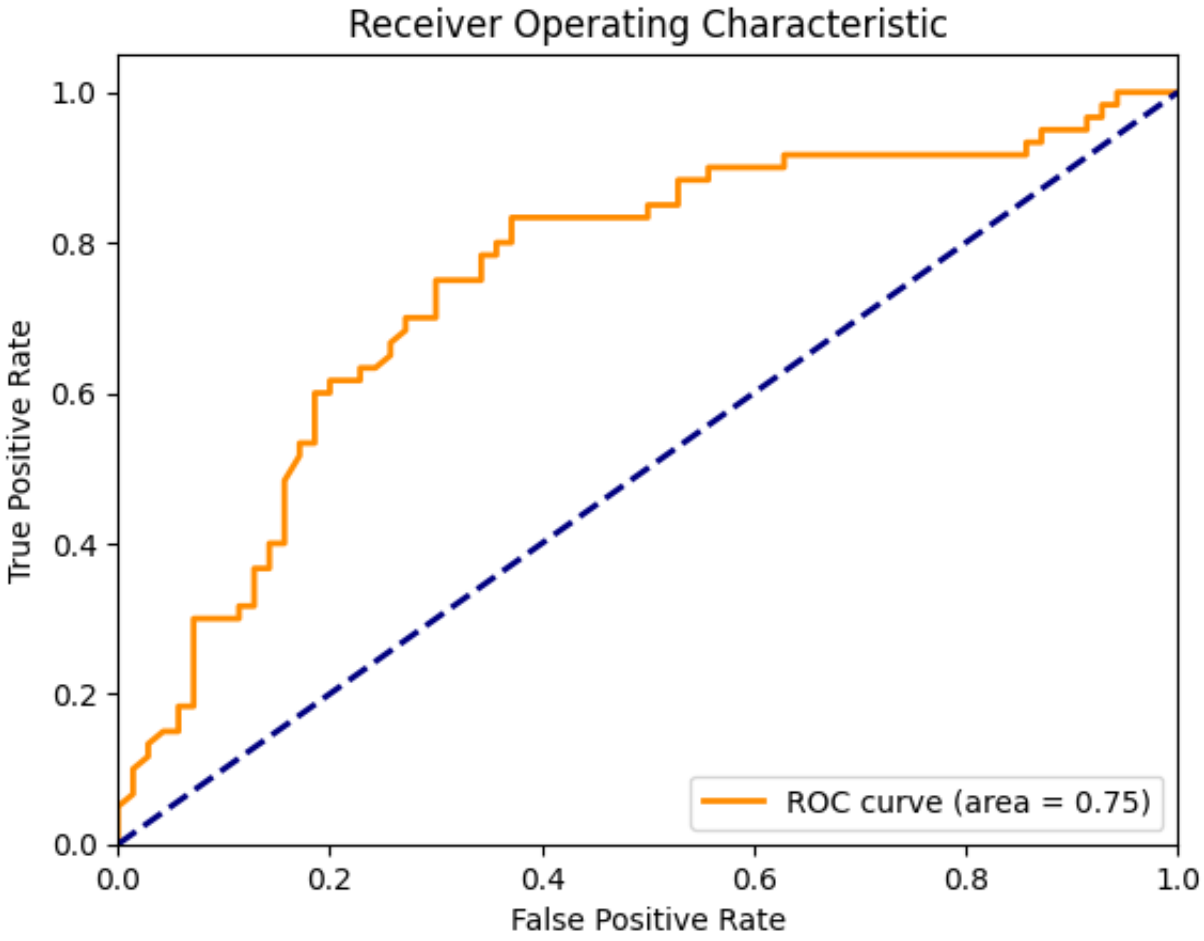

```
Running evaluation with seed 45

Evaluating Decision Tree with seed 45...
Best parameters for Decision Tree: {'ccp_alpha': 0.001, 'criterion': 'gini'

--- ROC Data for Copying ---
FPR = [0.0, 0.0, 0.0, 0.014285714285714285, 0.014285714285714285, 0.0285714
TPR = [0.0, 0.016666666666666666, 0.05, 0.06666666666666667, 0.1, 0.1166666
AUC = 0.7505952380952381
--- End of ROC Data ---

Training - Accuracy: 0.728, Sensitivity: 0.805, Specificity: 0.652, F1: 0.7

Test Metrics for manual threshold 0.5:
Accuracy: 0.715, Sensitivity: 0.817, Specificity: 0.629, F1: 0.726, ROC AUC
Threshold: 0.10, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.15, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.20, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.25, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.30, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.35, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.40, Metrics: {'Accuracy': 0.4846153846153846, 'Sensitivity': 0
Threshold: 0.45, Metrics: {'Accuracy': 0.5615384615384615, 'Sensitivity': 0
Threshold: 0.50, Metrics: {'Accuracy': 0.7153846153846154, 'Sensitivity': 0
Threshold: 0.55, Metrics: {'Accuracy': 0.6846153846153846, 'Sensitivity': 0
Threshold: 0.60, Metrics: {'Accuracy': 0.5461538461538461, 'Sensitivity': 0
Threshold: 0.65, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.70, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.75, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.80, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.85, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.90, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.95, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 1.00, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
SHAP Summary for Decision Tree
```




Age
Sex
−0.25 0.00 0.25
−0.25 0.00 0.25
SHAP interaction val

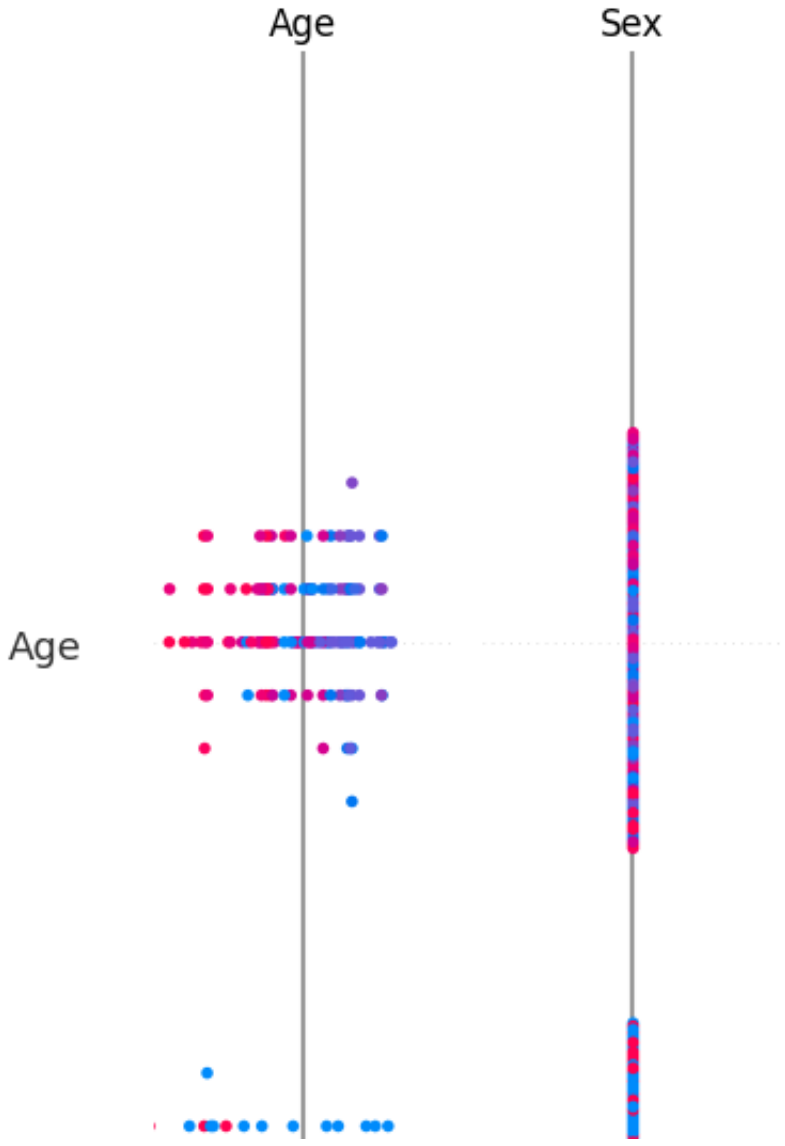

Confusion Matrix
Actual Success
44
26
45
40
35

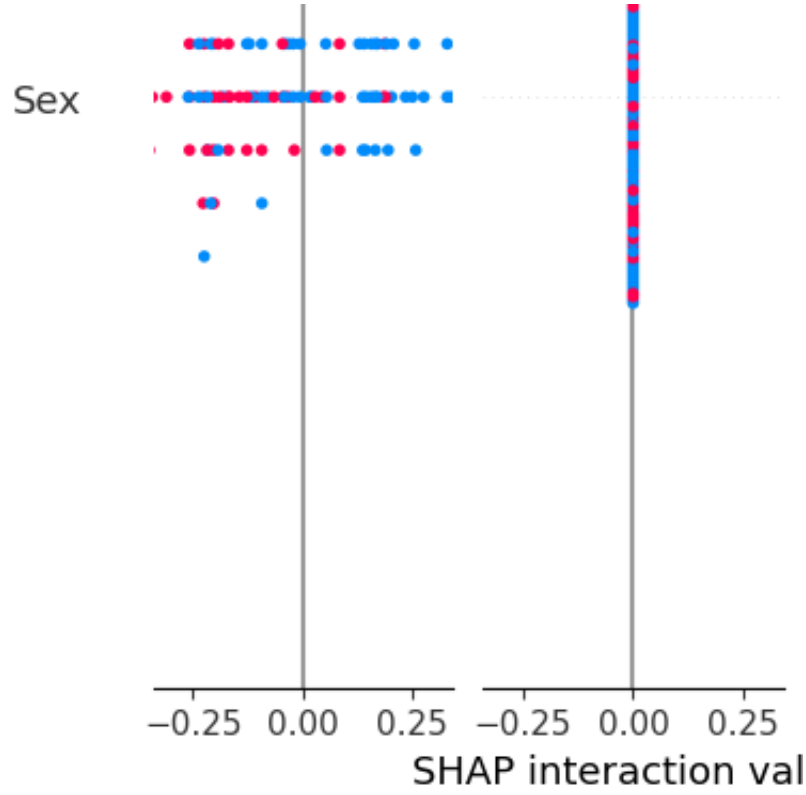


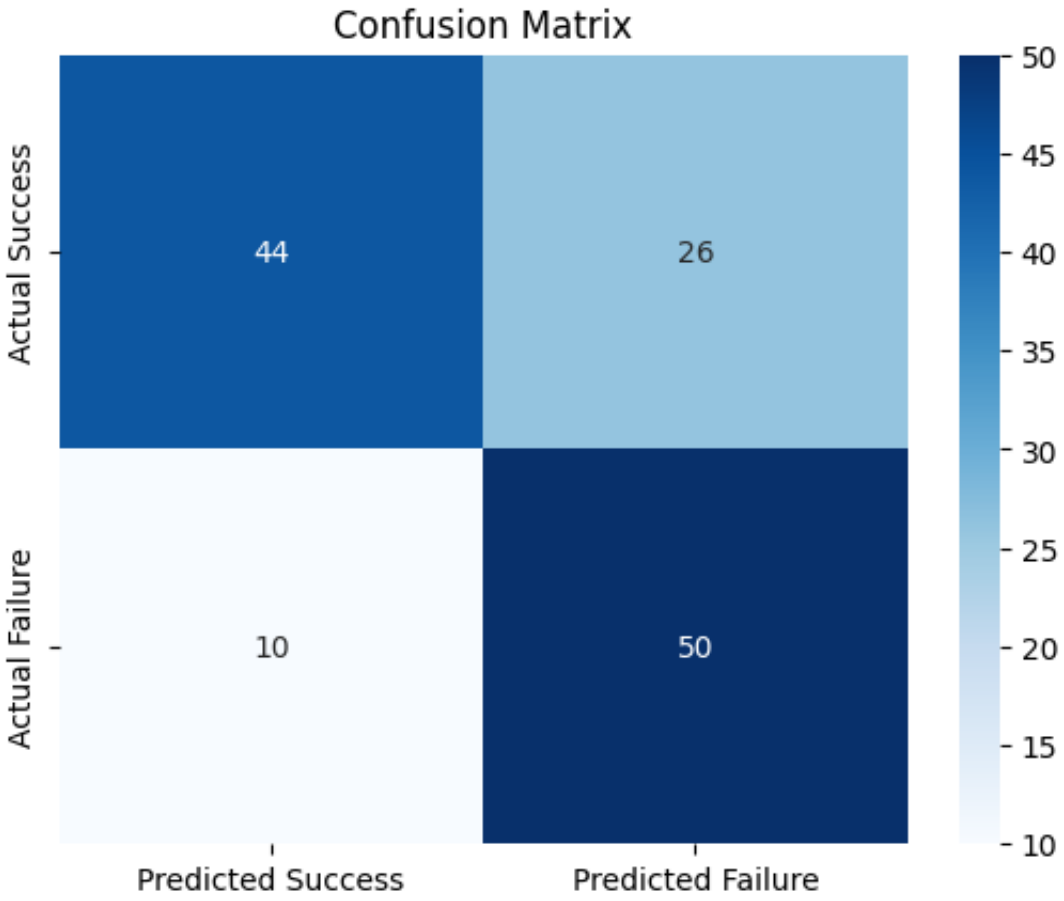



```
Running evaluation with seed 46

Evaluating Decision Tree with seed 46...
Best parameters for Decision Tree: {'ccp_alpha': 0.001, 'criterion': 'gini'

--- ROC Data for Copying ---
FPR = [0.0, 0.0, 0.0, 0.02857142857142857, 0.02857142857142857, 0.028571428
TPR = [0.0, 0.016666666666666666, 0.03333333333333333, 0.06666666666666667,
AUC = 0.7573809523809523
--- End of ROC Data ---

Training - Accuracy: 0.728, Sensitivity: 0.805, Specificity: 0.652, F1: 0.7
```

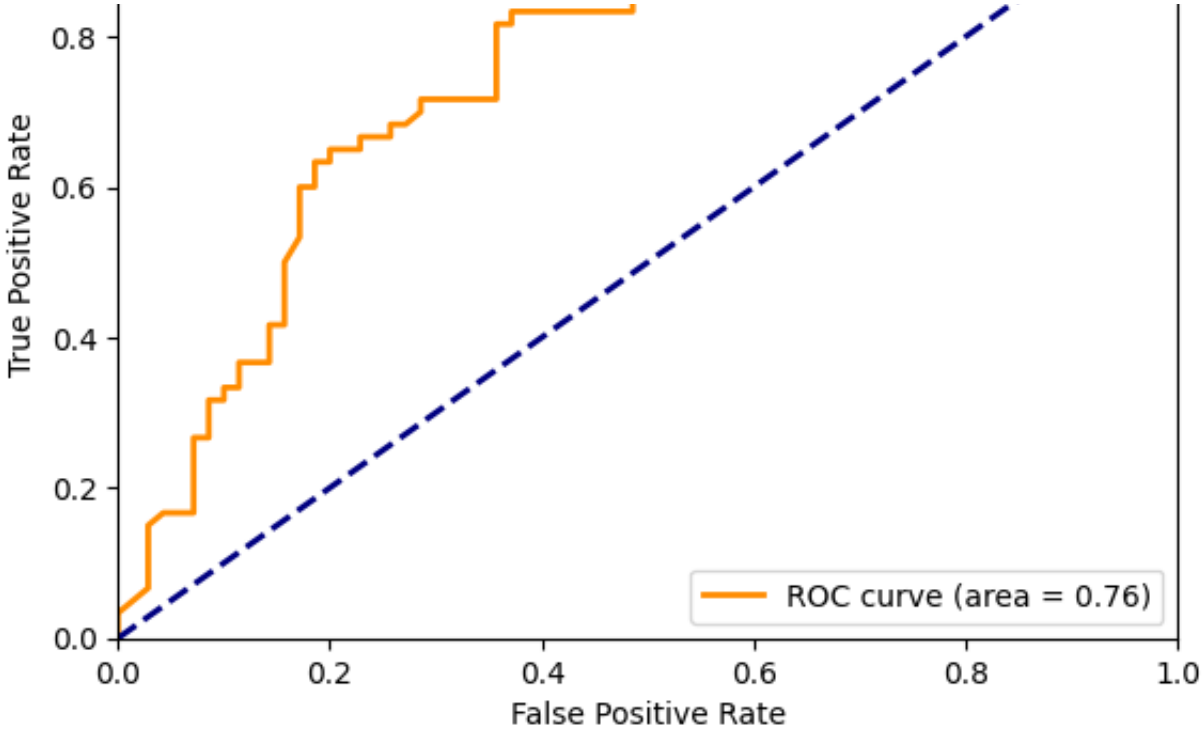
```
Test Metrics for manual threshold 0.5:
Accuracy: 0.723, Sensitivity: 0.833, Specificity: 0.629, F1: 0.735, ROC AUC
Threshold: 0.10, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.15, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.20, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.25, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.30, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.35, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.40, Metrics: {'Accuracy': 0.49230769230769234, 'Sensitivity':
Threshold: 0.45, Metrics: {'Accuracy': 0.5615384615384615, 'Sensitivity': 0
Threshold: 0.50, Metrics: {'Accuracy': 0.7230769230769231, 'Sensitivity': 0
Threshold: 0.55, Metrics: {'Accuracy': 0.7153846153846154, 'Sensitivity': 0
Threshold: 0.60, Metrics: {'Accuracy': 0.5538461538461539, 'Sensitivity': 0
Threshold: 0.65, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.70, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.75, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.80, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.85, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.90, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.95, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 1.00, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
SHAP Summary for Decision Tree
```




Sex
−0.25 0.00 0.25 −0.25 0.00 0.25
SHAP interaction val

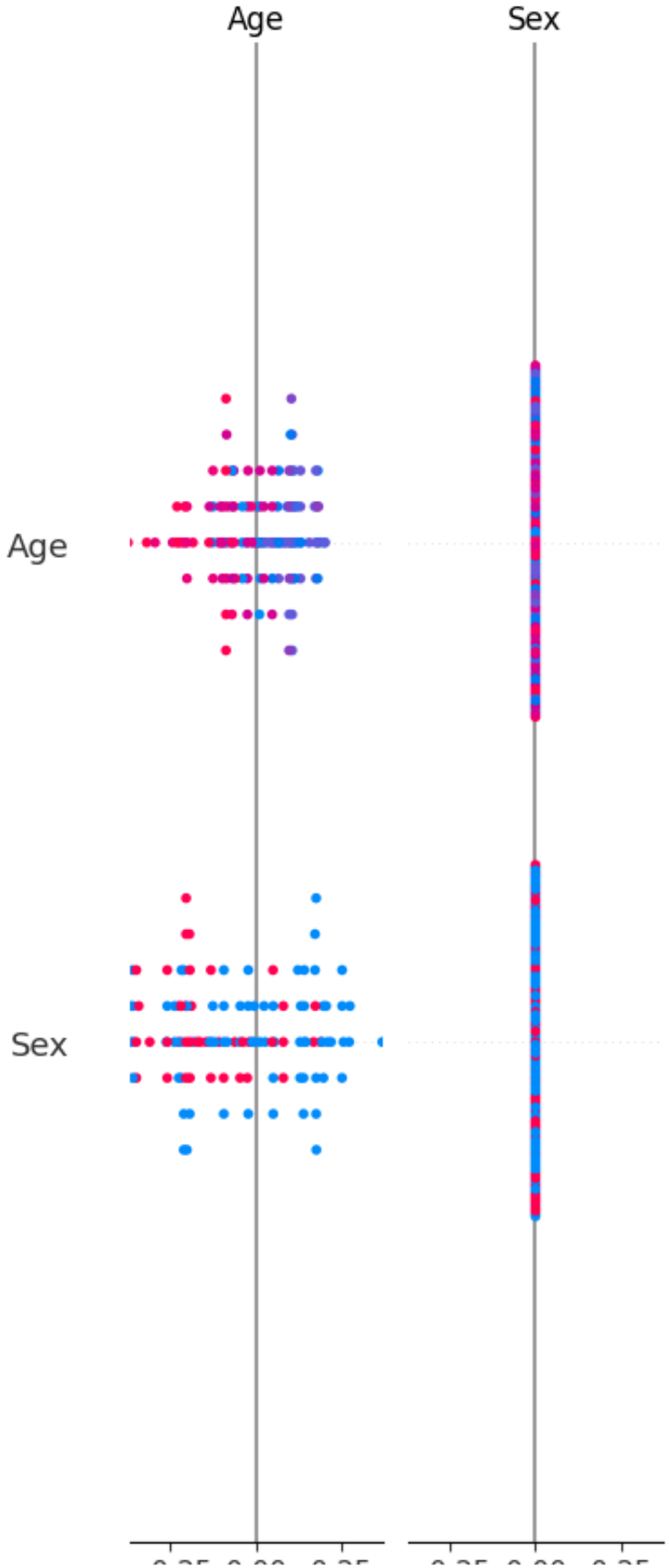

Confusion Matrix
Actual Success
Actual Failure
44
26
10
50
Predicted Success
Predicted Failure
50
45
40
35
30
25
20
15
10

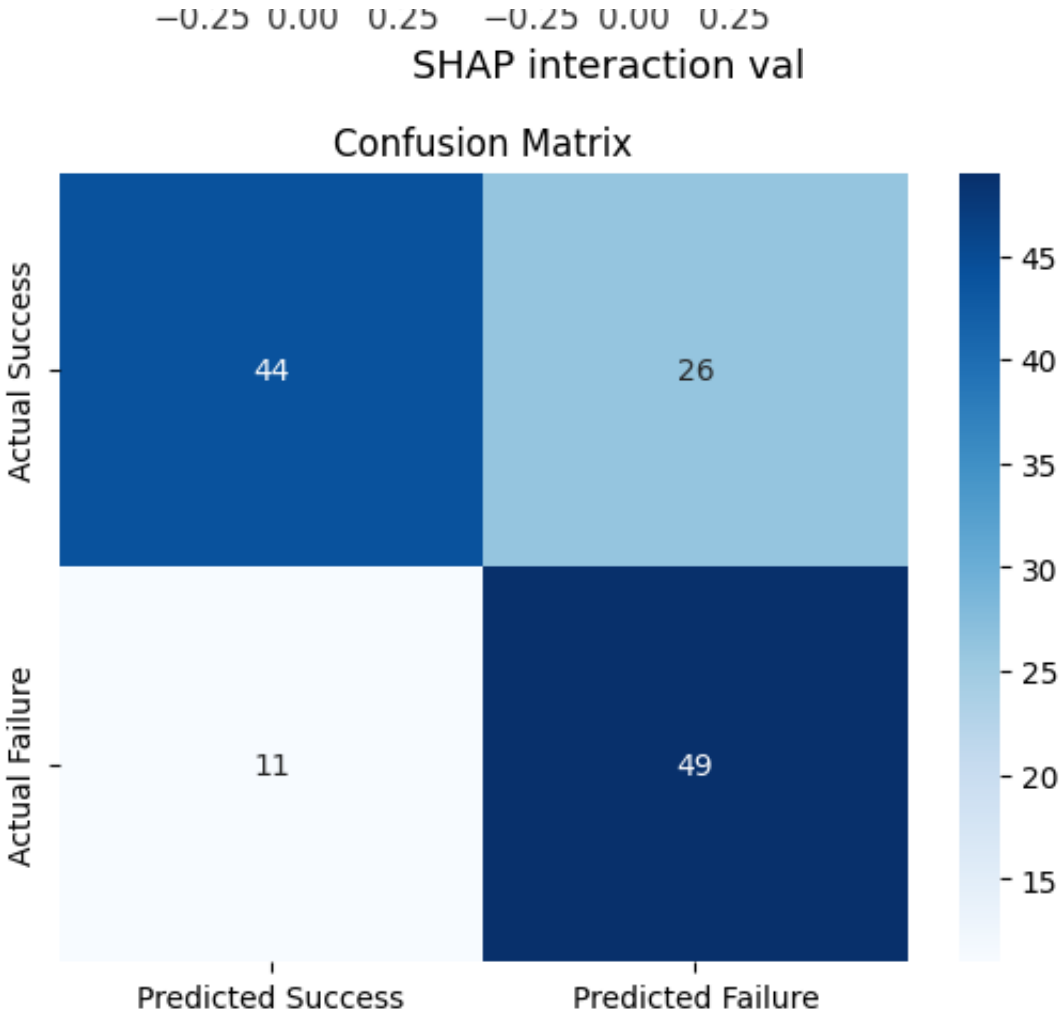

Receiver Operating Characteristic
1.0

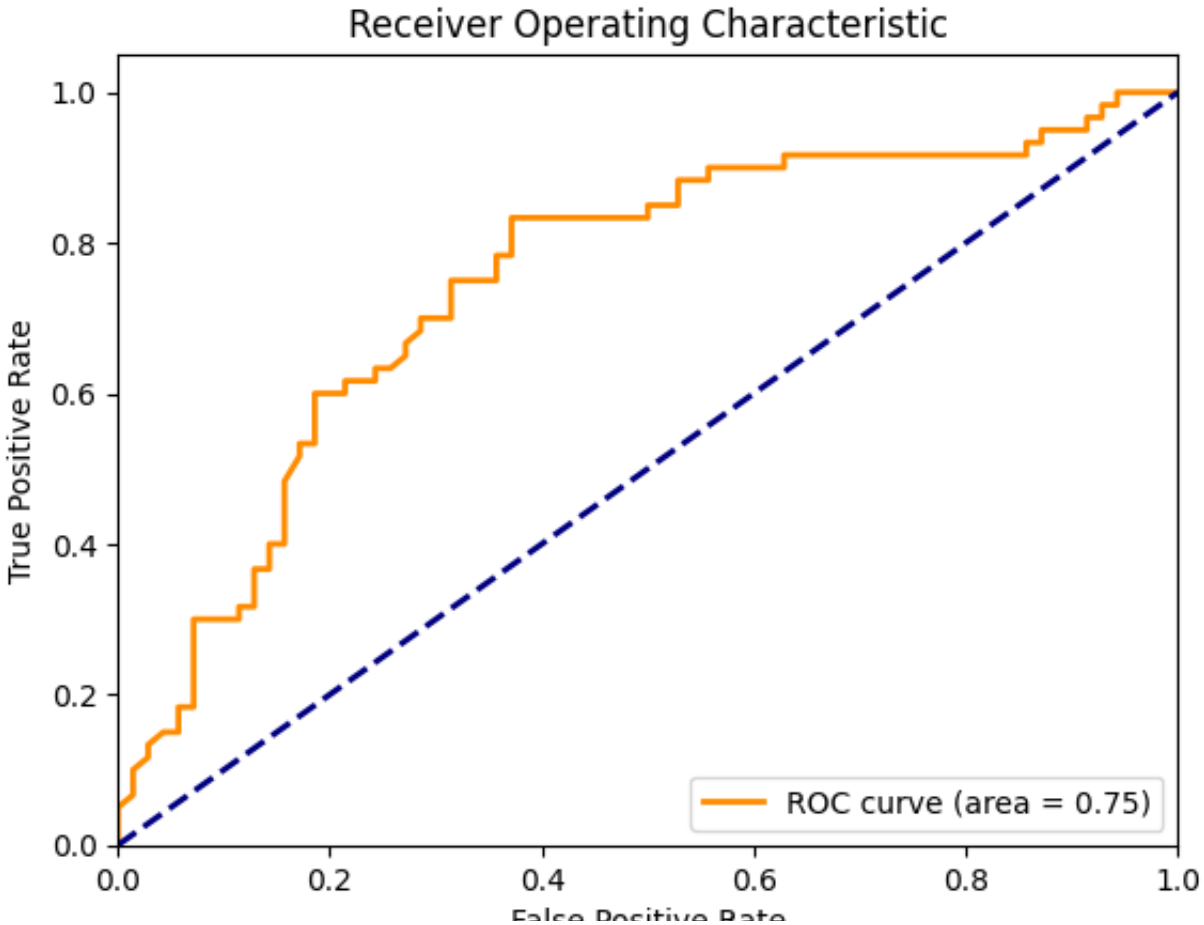



```
Running evaluation with seed 47

Evaluating Decision Tree with seed 47...
Best parameters for Decision Tree: {'ccp_alpha': 0.001, 'criterion': 'gini'

--- ROC Data for Copying ---
FPR = [0.0, 0.0, 0.0, 0.014285714285714285, 0.014285714285714285, 0.0285714
TPR = [0.0, 0.016666666666666666, 0.05, 0.06666666666666667, 0.1, 0.1166666
AUC = 0.7477380952380953
--- End of ROC Data ---

Training - Accuracy: 0.728, Sensitivity: 0.805, Specificity: 0.652, F1: 0.7

Test Metrics for manual threshold 0.5:
Accuracy: 0.715, Sensitivity: 0.817, Specificity: 0.629, F1: 0.726, ROC AUC
Threshold: 0.10, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.15, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.20, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.25, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.30, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.35, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.40, Metrics: {'Accuracy': 0.4846153846153846, 'Sensitivity': 0
Threshold: 0.45, Metrics: {'Accuracy': 0.5615384615384615, 'Sensitivity': 0
Threshold: 0.50, Metrics: {'Accuracy': 0.7153846153846154, 'Sensitivity': 0
Threshold: 0.55, Metrics: {'Accuracy': 0.6846153846153846, 'Sensitivity': 0
Threshold: 0.60, Metrics: {'Accuracy': 0.5461538461538461, 'Sensitivity': 0
Threshold: 0.65, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.70, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.75, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.80, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.85, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.90, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
```

```
Threshold: 0.95, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 1.00, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
SHAP Summary for Decision Tree
```

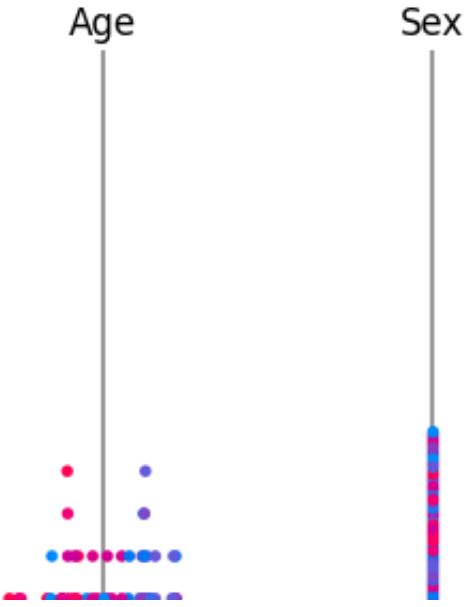


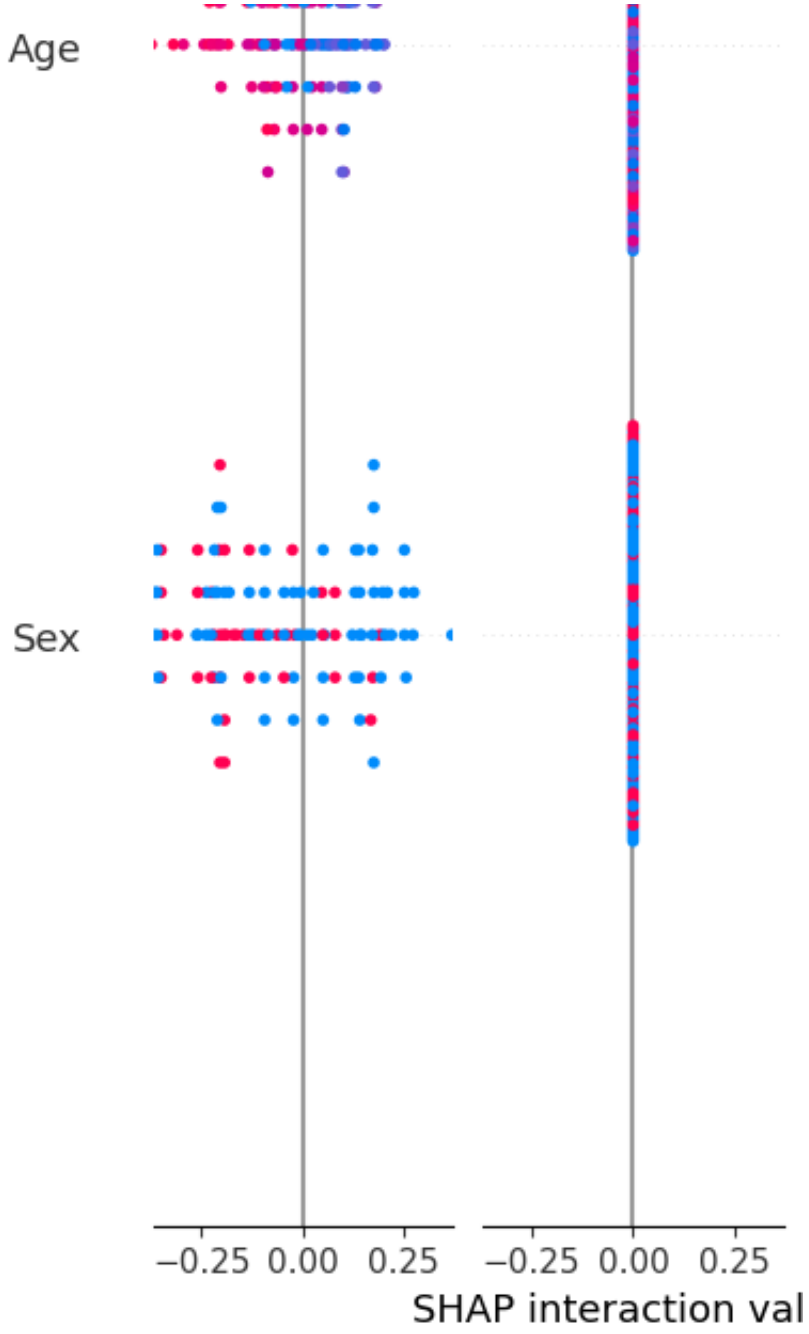

−0.25 0.00 0.25 −0.25 0.00 0.25
SHAP interaction val
Confusion Matrix
Actual Success
Actual Failure
44
26
11
49
45
40
35
30
25
20
15
Predicted Success
Predicted Failure

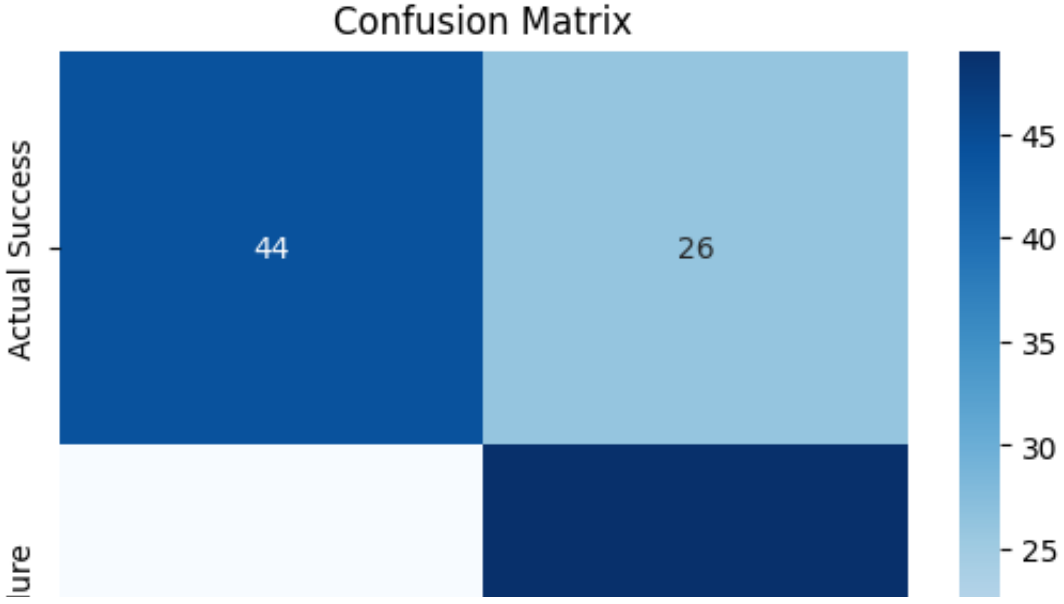

Receiver Operating Characteristic
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (area = 0.75)

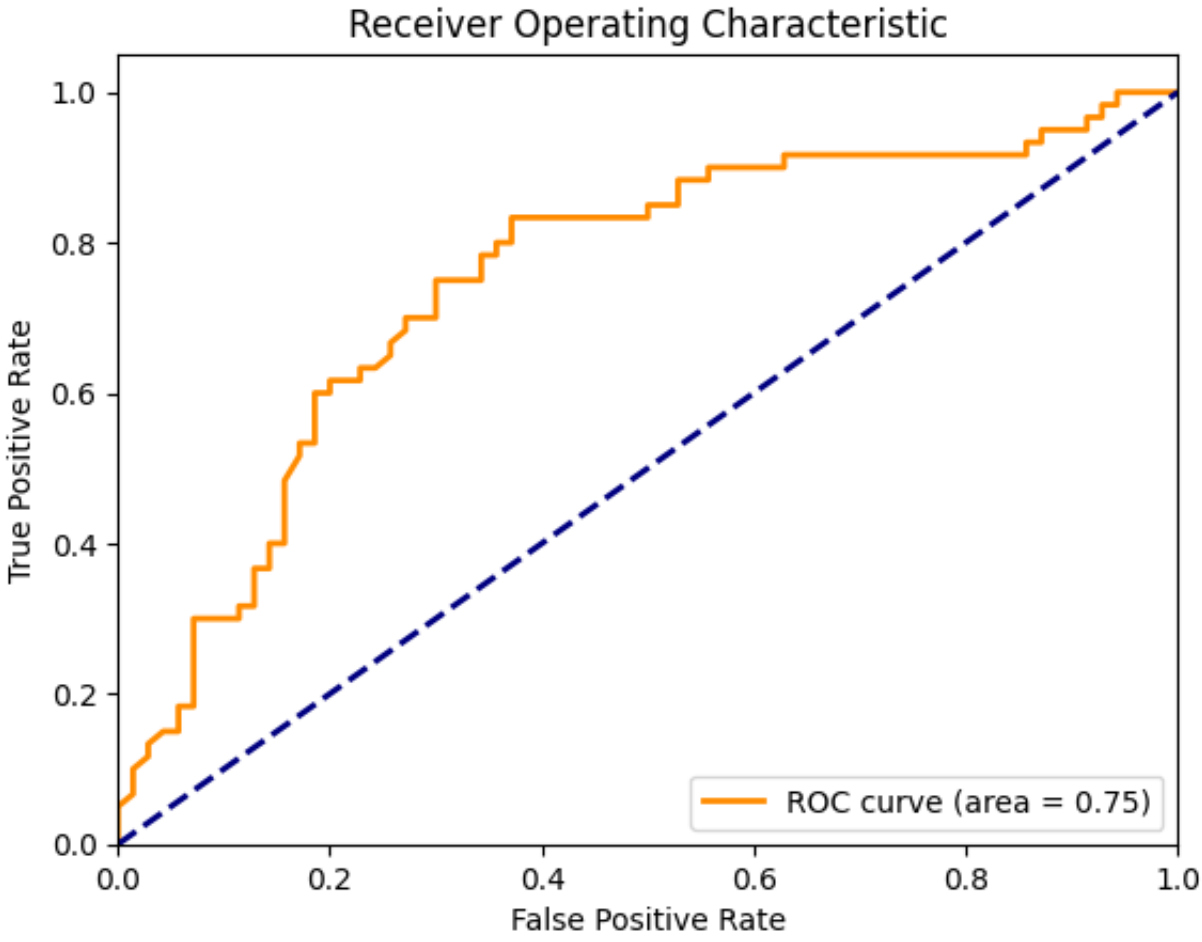
False Positive Rate

```
Running evaluation with seed 48

Evaluating Decision Tree with seed 48...
Best parameters for Decision Tree: {'ccp_alpha': 0.001, 'criterion': 'gini'

--- ROC Data for Copying ---
FPR = [0.0, 0.0, 0.0, 0.014285714285714285, 0.014285714285714285, 0.0285714
TPR = [0.0, 0.016666666666666666, 0.05, 0.06666666666666667, 0.1, 0.1166666
AUC = 0.7505952380952381
--- End of ROC Data ---

Training - Accuracy: 0.728, Sensitivity: 0.805, Specificity: 0.652, F1: 0.7

Test Metrics for manual threshold 0.5:
Accuracy: 0.715, Sensitivity: 0.817, Specificity: 0.629, F1: 0.726, ROC AUC
Threshold: 0.10, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.15, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.20, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.25, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.30, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.35, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.40, Metrics: {'Accuracy': 0.4846153846153846, 'Sensitivity': 0
Threshold: 0.45, Metrics: {'Accuracy': 0.5615384615384615, 'Sensitivity': 0
Threshold: 0.50, Metrics: {'Accuracy': 0.7153846153846154, 'Sensitivity': 0
Threshold: 0.55, Metrics: {'Accuracy': 0.6846153846153846, 'Sensitivity': 0
Threshold: 0.60, Metrics: {'Accuracy': 0.5461538461538461, 'Sensitivity': 0
Threshold: 0.65, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.70, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.75, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.80, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.85, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.90, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.95, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 1.00, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
SHAP Summary for Decision Tree
```



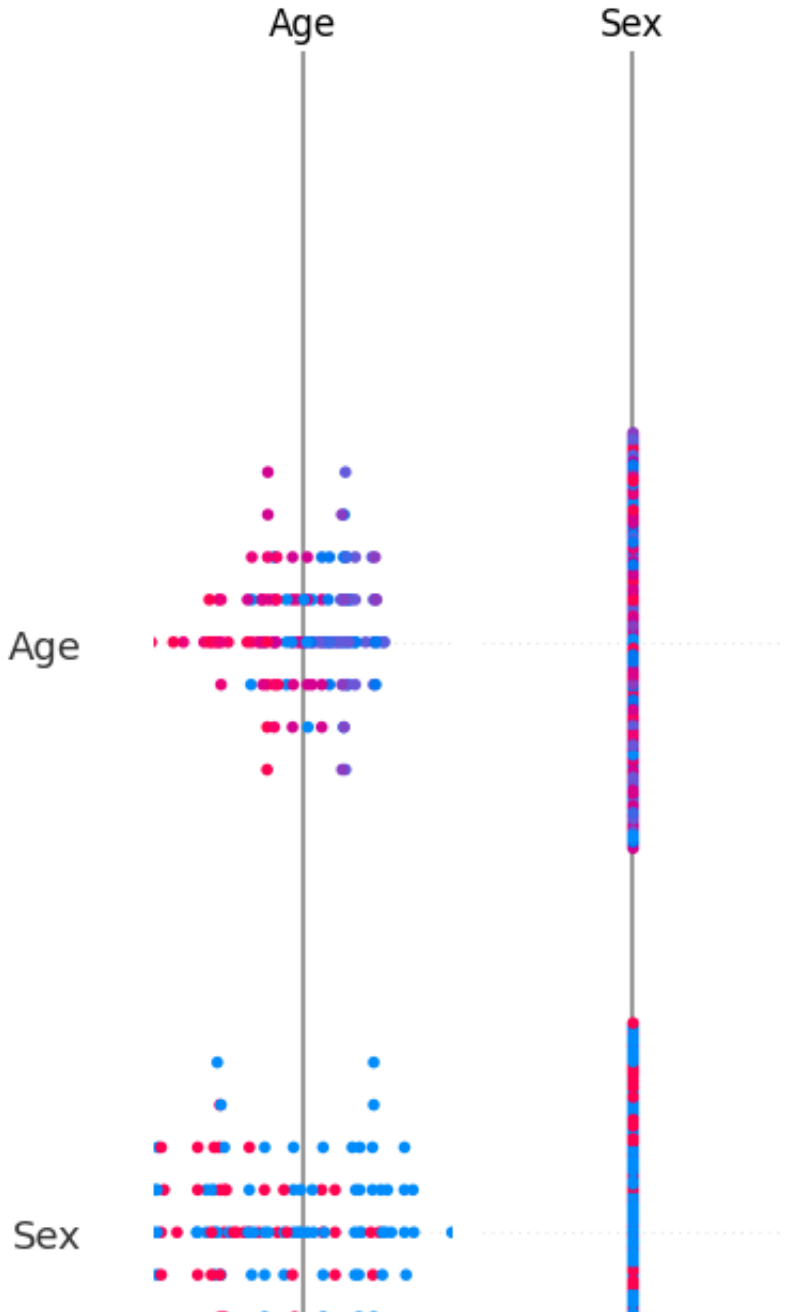

Age
Sex
−0.25 0.00 0.25
−0.25 0.00 0.25
SHAP interaction val

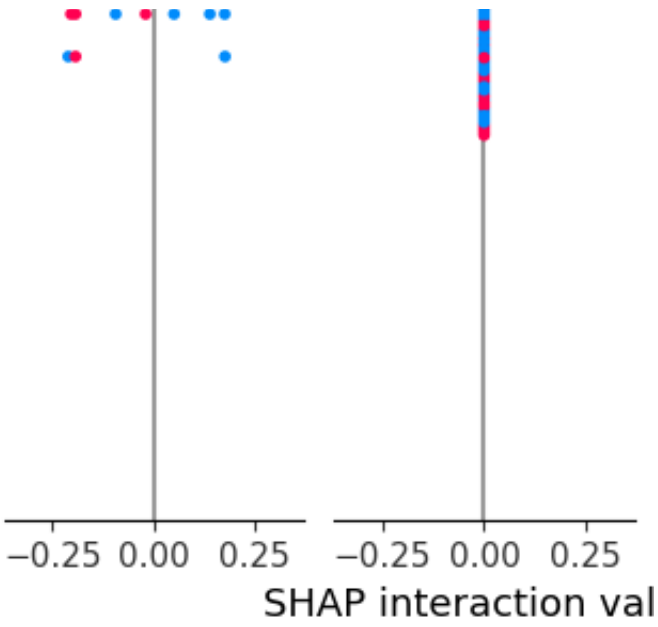

Confusion Matrix
Actual Success
ure
44
26
45
40
35
30
25

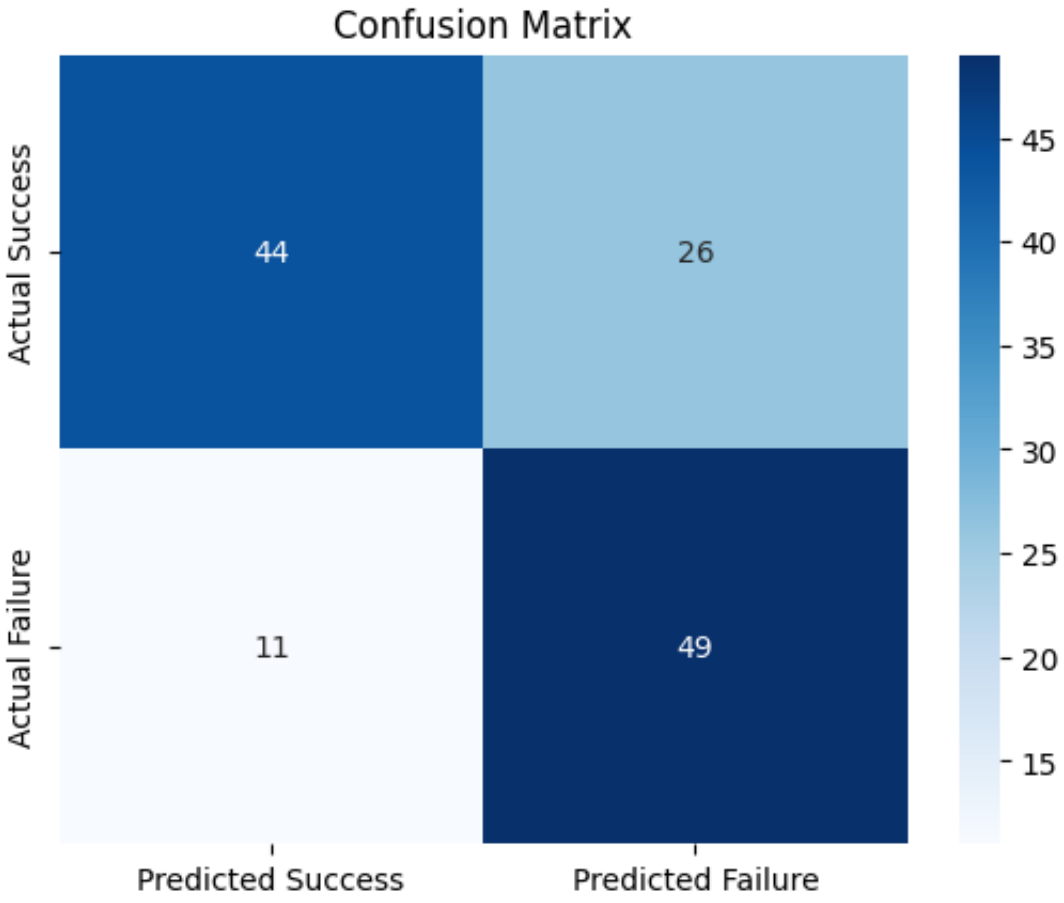


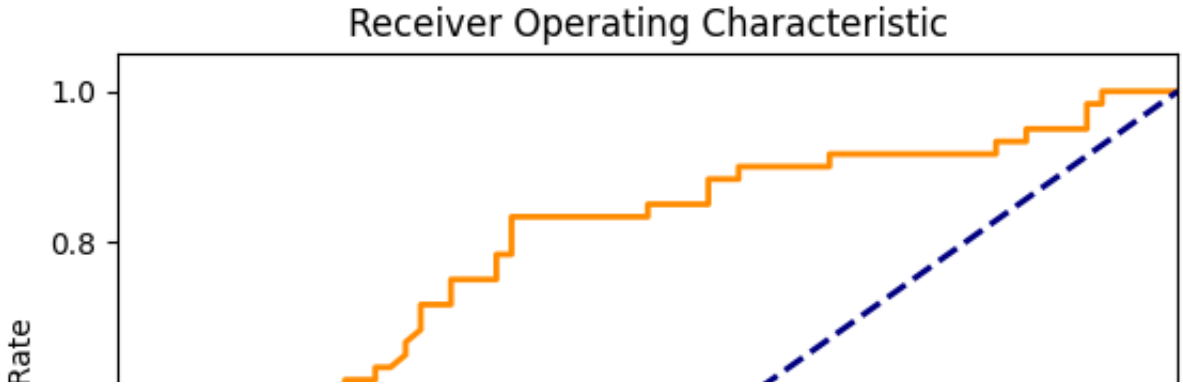

```
Running evaluation with seed 49

Evaluating Decision Tree with seed 49...
Best parameters for Decision Tree: {'ccp_alpha': 0.001, 'criterion': 'gini'

--- ROC Data for Copying ---
FPR = [0.0, 0.0, 0.0, 0.014285714285714285, 0.014285714285714285, 0.0285714
TPR = [0.0, 0.016666666666666666, 0.05, 0.06666666666666667, 0.1, 0.1166666
AUC = 0.7472619047619048
--- End of ROC Data ---

Training - Accuracy: 0.728, Sensitivity: 0.805, Specificity: 0.652, F1: 0.7

Test Metrics for manual threshold 0.5:
Accuracy: 0.715, Sensitivity: 0.817, Specificity: 0.629, F1: 0.726, ROC AUC
Threshold: 0.10, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.15, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.20, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
```

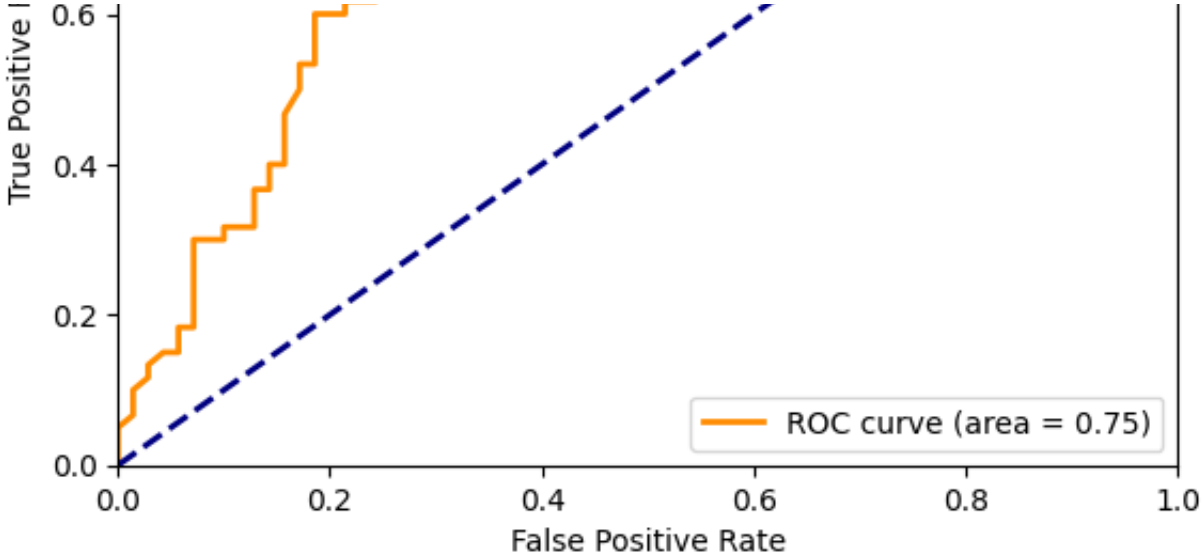
```
Threshold: 0.25, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.30, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.35, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.40, Metrics: {'Accuracy': 0.49230769230769234, 'Sensitivity':
Threshold: 0.45, Metrics: {'Accuracy': 0.5538461538461539, 'Sensitivity': 0
Threshold: 0.50, Metrics: {'Accuracy': 0.7153846153846154, 'Sensitivity': 0
Threshold: 0.55, Metrics: {'Accuracy': 0.6692307692307692, 'Sensitivity': 0
Threshold: 0.60, Metrics: {'Accuracy': 0.5461538461538461, 'Sensitivity': 0
Threshold: 0.65, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.70, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.75, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.80, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.85, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.90, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.95, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 1.00, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
SHAP Summary for Decision Tree
```




−0.25 0.00 0.25
−0.25 0.00 0.25
SHAP interaction val


Confusion Matrix
Actual Success
Actual Failure
44
26
11
49
Predicted Success
Predicted Failure
45
40
35
30
25
20
15


Receiver Operating Characteristic
1.0
0.8
Rate




```
Aggregated Test Set Metrics Across Seeds:
   accuracy  sensitivity  specificity        f1   roc_auc
0  0.723077     0.833333     0.628571  0.735294  0.755000
1  0.723077     0.833333     0.628571  0.735294  0.755714
2  0.715385     0.816667     0.628571  0.725926  0.750119
3  0.723077     0.833333     0.628571  0.735294  0.753333
4  0.723077     0.833333     0.628571  0.735294  0.753333
5  0.715385     0.816667     0.628571  0.725926  0.750595
6  0.723077     0.833333     0.628571  0.735294  0.757381
7  0.715385     0.816667     0.628571  0.725926  0.747738
8  0.715385     0.816667     0.628571  0.725926  0.750595
9  0.715385     0.816667     0.628571  0.725926  0.747262

Summary of Test Set Metrics (Mean, Standard Error, 95% Confidence Interval)
Accuracy: Mean = 0.719, SE = 0.001, 95% CI = [0.716, 0.722]
Sensitivity: Mean = 0.825, SE = 0.003, 95% CI = [0.819, 0.831]
Specificity: Mean = 0.629, SE = 0.000, 95% CI = [0.629, 0.629]
F1: Mean = 0.731, SE = 0.002, 95% CI = [0.727, 0.734]
Roc_auc: Mean = 0.752, SE = 0.001, 95% CI = [0.750, 0.755]
```

```
def evaluate_model(model, name, grid, X_train, y_train, X_test, y_test, cv, scor
    print(f"Evaluating {name} with seed {seed}...")

    inner_cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=seed)
    outer_cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=seed)

    clf = GridSearchCV(model, grid, cv=inner_cv, scoring='roc_auc')
    nested_scores = cross_validate(clf, X=X_train, y=y_train, cv=outer_cv, scori

    clf.fit(X_train, y_train)
    best_model = clf.best_estimator_
    best_params = clf.best_params_

    calibrated_clf = CalibratedClassifierCV(estimator=best_model, method='sigmoi
```

```
calibrated_clf = CalibratedClassifierCV(estimator=best_model, method='sigmoi
calibrated_clf.fit(X_train, y_train)

y_probs = calibrated_clf.predict_proba(X_test)[:, 1]

# Calcular FPR, TPR e AUC
fpr, tpr, thresholds = roc_curve(y_test, y_probs)
roc_auc = auc(fpr, tpr)

# Imprimir os valores de FPR, TPR e AUC de forma fácil de copiar
print("\n--- Dados ROC para copiar ---")
print("FPR =", fpr.tolist())
print("TPR =", tpr.tolist())
print("AUC =", roc_auc)
print("--- Fim dos Dados ROC ---\n")

# Calculate metrics for the training set
y_train_pred = best_model.predict(X_train)
y_train_probs = best_model.predict_proba(X_train)[:, 1]
train_acc = accuracy_score(y_train, y_train_pred)
train_sens = sensitivity(y_train, y_train_pred)
train_spec = specificity(y_train, y_train_pred)
train_f1 = f1_score(y_train, y_train_pred)
train_roc_auc = roc_auc_score(y_train, y_train_probs)

print(f"Training - Accuracy: {train_acc}, Sensitivity: {train_sens}, Specifi

# Metrics for the manually set threshold
y_pred_manual = (y_probs >= manual_threshold).astype(int)
manual_acc = accuracy_score(y_test, y_pred_manual)
manual_sens = sensitivity(y_test, y_pred_manual)
manual_spec = specificity(y_test, y_pred_manual)
manual_f1 = f1_score(y_test, y_pred_manual)
manual_roc_auc = roc_auc_score(y_test, y_probs)

print(f"Metrics for manual threshold {manual_threshold}:")
print(f"Accuracy: {manual_acc}, Sensitivity: {manual_sens}, Specificity: {ma

threshold_metrics = {}
for threshold in threshold_list:
    y_pred_threshold = (y_probs >= threshold).astype(int)
    threshold_acc = accuracy_score(y_test, y_pred_threshold)
    threshold_sens = sensitivity(y_test, y_pred_threshold)
    threshold_spec = specificity(y_test, y_pred_threshold)
    threshold_f1 = f1_score(y_test, y_pred_threshold)
    threshold_metrics[threshold] = {
        'Accuracy': threshold_acc,
        'Sensitivity': threshold_sens,
        'Specificity': threshold_spec,
```

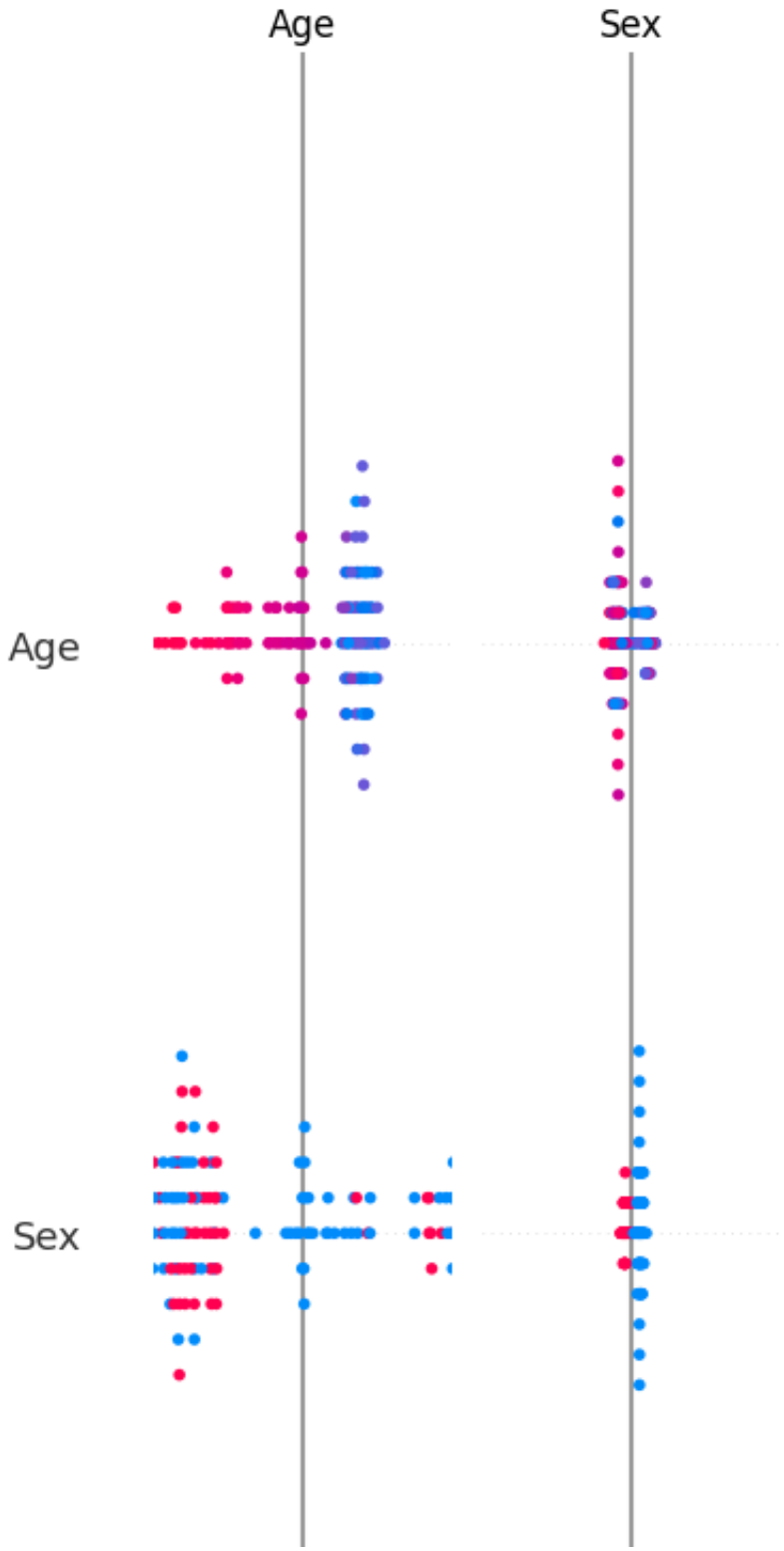
```
            'F1': threshold_f1,
            'ROC AUC': manual_roc_auc
        }

    for threshold, metrics in threshold_metrics.items():
        print(f"Threshold: {threshold:.2f}, Metrics: {metrics}")

    calculate_and_plot_shap(best_model, X_train, X_test, name)

    # Prepare dictionary of test metrics for aggregation
    test_metrics = {
        "accuracy": manual_acc,
        "sensitivity": manual_sens,
        "specificity": manual_spec,
        "f1": manual_f1,
        "roc_auc": manual_roc_auc
    }

    return best_model, manual_threshold, best_params, nested_scores, calibrated_

def calculate_and_plot_shap(model, X_train, X_test, model_name):
    if isinstance(model, (RandomForestClassifier)):
        explainer = shap.TreeExplainer(model)
    else:
        explainer = shap.KernelExplainer(model.predict_proba, X_train.sample(100
    shap_values = explainer.shap_values(X_test)
    print(f"SHAP Summary for {model_name}")
    shap.summary_plot(shap_values, X_test, max_display=10)

def plot_confusion_matrix(y_true, y_pred):
    matrix = confusion_matrix(y_true, y_pred)
    sns.heatmap(matrix, annot=True, fmt='d', cmap='Blues',
                xticklabels=['Predicted Success', 'Predicted Failure'],
                yticklabels=['Actual Success', 'Actual Failure'])
    plt.title('Confusion Matrix Random Forest')
    plt.show()

def plot_roc_curve(y_true, y_probs):
    fpr, tpr, thresholds = roc_curve(y_true, y_probs)
    roc_auc = auc(fpr, tpr)

    plt.figure()
    plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (area = {roc_
    plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
```

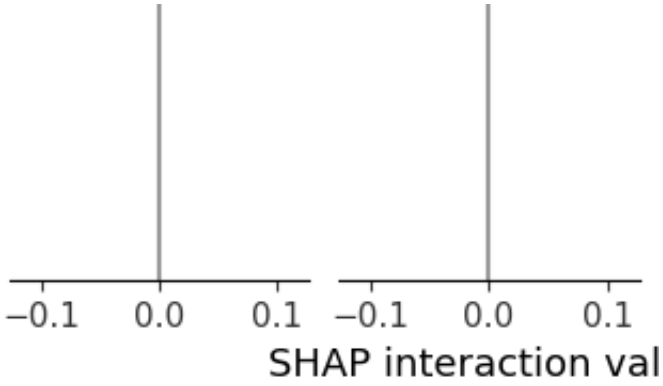
```
    plt.title('Receiver Operating Characteristic Random Forest')
    plt.legend(loc="lower right")
    plt.show()

def evaluate_random_forest(X_train, y_train, X_test, y_test, cv, scoring, manual
    model = RandomForestClassifier(n_jobs=-1, random_state=seed)
    grid = {
        'n_estimators': [500],
        'max_depth': [5],
        'min_samples_split': [2],
        'min_samples_leaf': [6],
        'max_features': ['sqrt'],
    }
    return evaluate_model(model, "Random Forest", grid, X_train, y_train, X_test

def main(X_train, y_train, X_test, y_test):
    cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=10, random_state=42)
    scoring = {
        'accuracy': make_scorer(accuracy_score),
        'sensitivity': make_scorer(sensitivity),
        'specificity': make_scorer(specificity),
        'f1': make_scorer(f1_score),
        'roc_auc': make_scorer(roc_auc_score)
    }
    manual_threshold = 0.55
    threshold_list = np.arange(0.1, 1.05, 0.05)

    aggregated_metrics = []

    for seed in range(40, 50):
        print(f"Running evaluation with seed {seed}")
        best_model, manual_threshold, best_params, nested_scores, calibrated_clf
            X_train, y_train, X_test, y_test, cv, scoring, manual_threshold, thr
        )

        # Use calibrated_clf for prediction probabilities
        y_probs = calibrated_clf.predict_proba(X_test)[:, 1]
        y_pred_manual = (y_probs >= manual_threshold).astype(int)

        plot_confusion_matrix(y_test, y_pred_manual)
        plot_roc_curve(y_test, y_probs)

        aggregated_metrics.append(test_metrics)

    # Aggregate results across seeds
    results_df = pd.DataFrame(aggregated_metrics)
    n = len(results_df)
    print("\nAggregated Test Set Metrics Across Seeds:")
    print(results_df)
```

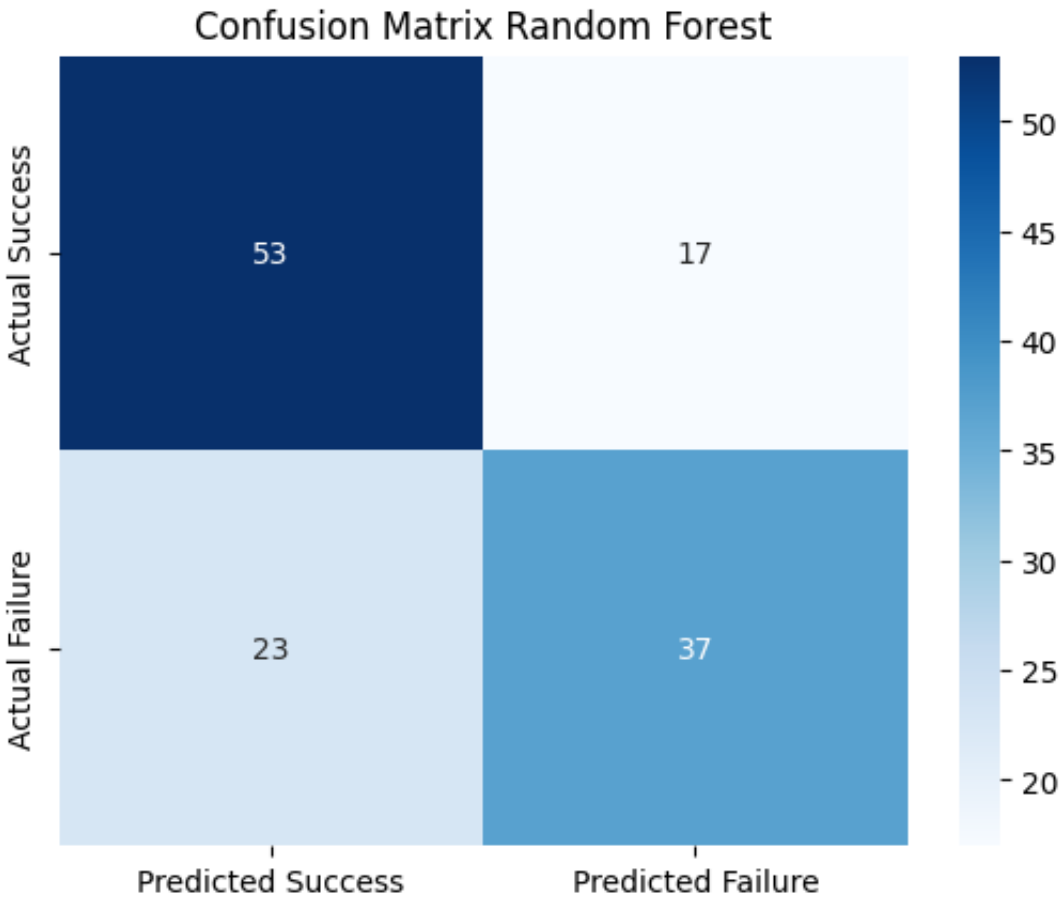
```
    print(results_df)

    # Compute mean, standard error, and 95% confidence interval for each metric
    def summarize_metric(metric_values):
        mean_val = metric_values.mean()
        std_val = metric_values.std(ddof=1)
        se = std_val / np.sqrt(n)
        t_crit = stats.t.ppf(0.975, df=n - 1)
        ci_lower = mean_val - t_crit * se
        ci_upper = mean_val + t_crit * se
        return mean_val, se, (ci_lower, ci_upper)

    metrics_summary = {}
    for metric in results_df.columns:
        mean_val, se, ci = summarize_metric(results_df[metric])
        metrics_summary[metric] = {
            "Mean": mean_val,
            "Standard Error": se,
            "95% CI": ci
        }

    print("\nSummary of Test Set Metrics (Mean, Standard Error, 95% Confidence I
    for metric, summary in metrics_summary.items():
        print(f"{metric.capitalize()}: Mean = {summary['Mean']:.3f}, SE = {summa
              f"95% CI = [{summary['95% CI'][0]:.3f}, {summary['95% CI'][1]:.3f}

if __name__ == '__main__':
    main(X_train, y_train, X_test, y_test)
```

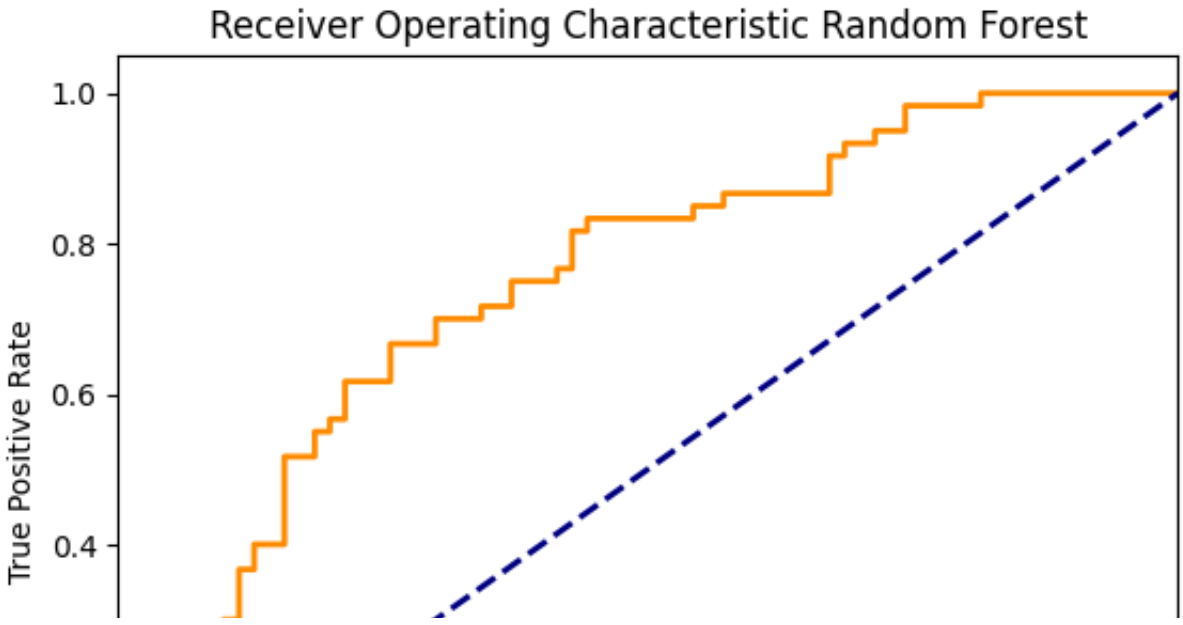
```
Running evaluation with seed 40
Evaluating Random Forest with seed 40...

--- Dados ROC para copiar ---
FPR = [0.0, 0.0, 0.0, 0.014285714285714285, 0.02857142857142857, 0.02857142
TPR = [0.0, 0.016666666666666666, 0.03333333333333333, 0.05, 0.05, 0.1, 0.1
AUC = 0.7471428571428571
--- Fim dos Dados ROC ---

Training - Accuracy: 0.7238658777120316, Sensitivity: 0.7529880478087649, S
Metrics for manual threshold 0.55:
Accuracy: 0.6923076923076923, Sensitivity: 0.6166666666666667, Specificity:
Threshold: 0.10, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.15, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.20, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.25, Metrics: {'Accuracy': 0.46923076923076923, 'Sensitivity':
Threshold: 0.30, Metrics: {'Accuracy': 0.5, 'Sensitivity': 1.0, 'Specificit
Threshold: 0.35, Metrics: {'Accuracy': 0.5538461538461539, 'Sensitivity': 0
Threshold: 0.40, Metrics: {'Accuracy': 0.5923076923076923, 'Sensitivity': 0
Threshold: 0.45, Metrics: {'Accuracy': 0.6384615384615384, 'Sensitivity': 0
Threshold: 0.50, Metrics: {'Accuracy': 0.676923076923077, 'Sensitivity': 0.
```

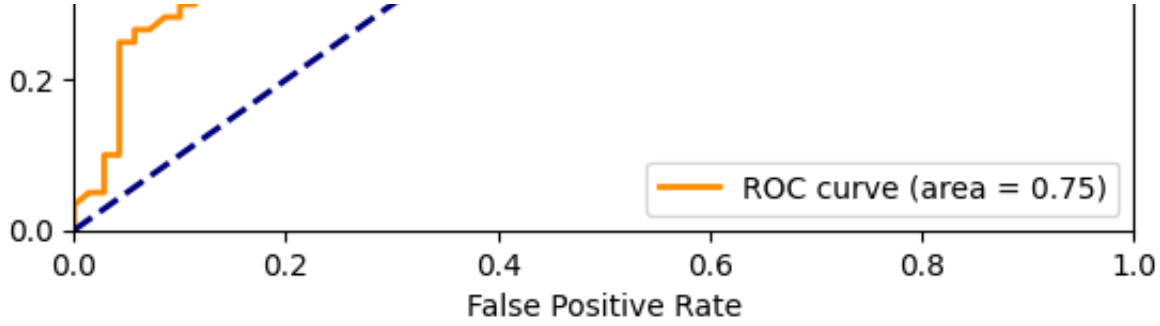
```
Threshold: 0.55, Metrics: {'Accuracy': 0.6923076923076923, 'Sensitivity': 0
Threshold: 0.60, Metrics: {'Accuracy': 0.6538461538461539, 'Sensitivity': 0
Threshold: 0.65, Metrics: {'Accuracy': 0.6307692307692307, 'Sensitivity': 0
Threshold: 0.70, Metrics: {'Accuracy': 0.5692307692307692, 'Sensitivity': 0
Threshold: 0.75, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.80, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.85, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.90, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.95, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 1.00, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
SHAP Summary for Random Forest
```



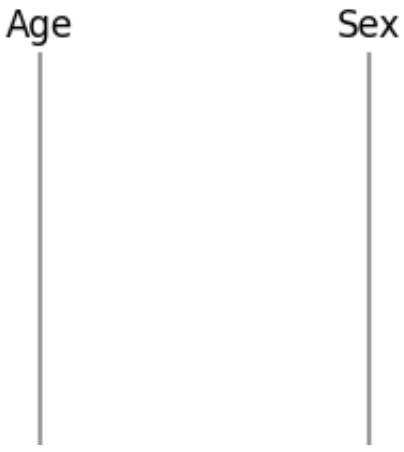

−0.1
0.0
0.1
−0.1
0.0
0.1
SHAP interaction val

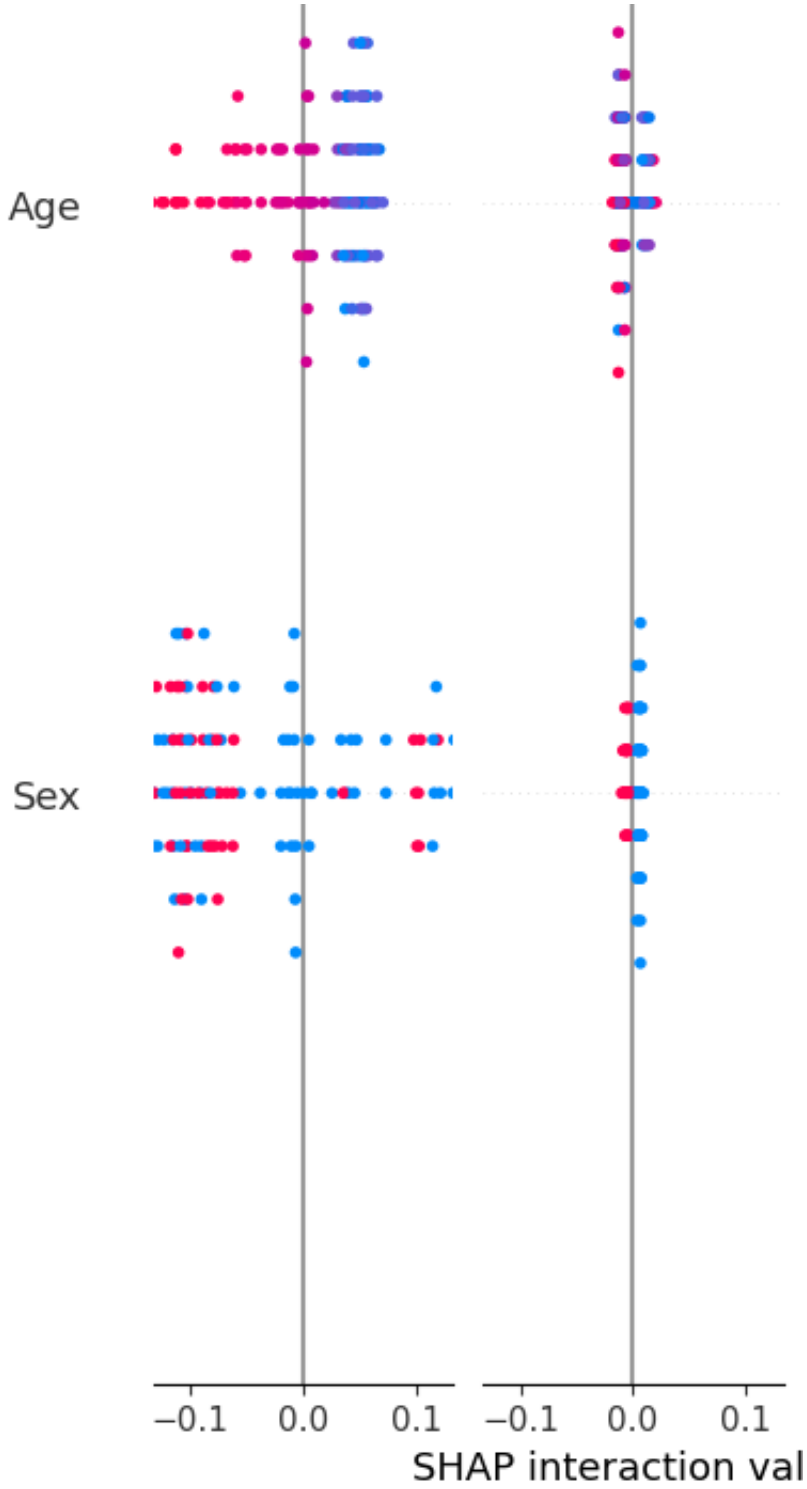

Confusion Matrix Random Forest
Actual Success
Actual Failure
53
17
23
37
Predicted Success
Predicted Failure
50
45
40
35
30
25
20

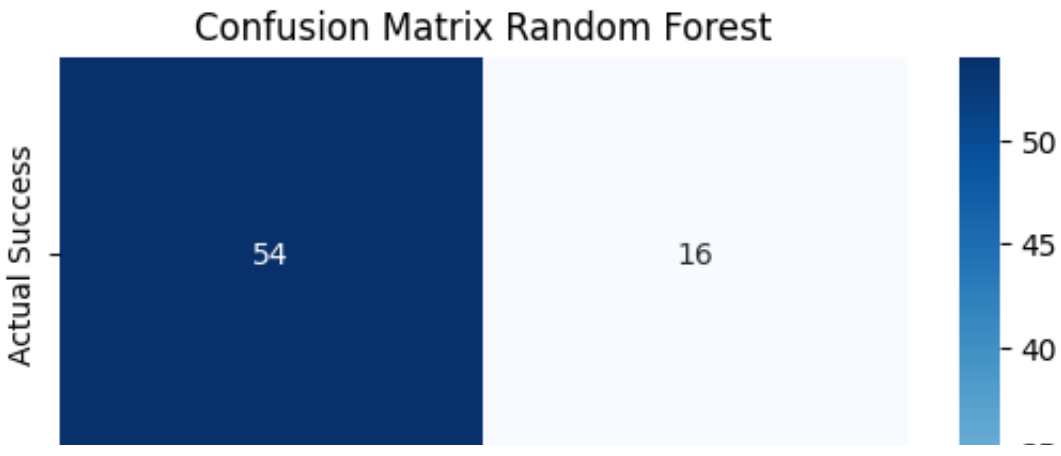

Receiver Operating Characteristic Random Forest
True Positive Rate
1.0
0.8
0.6
0.4

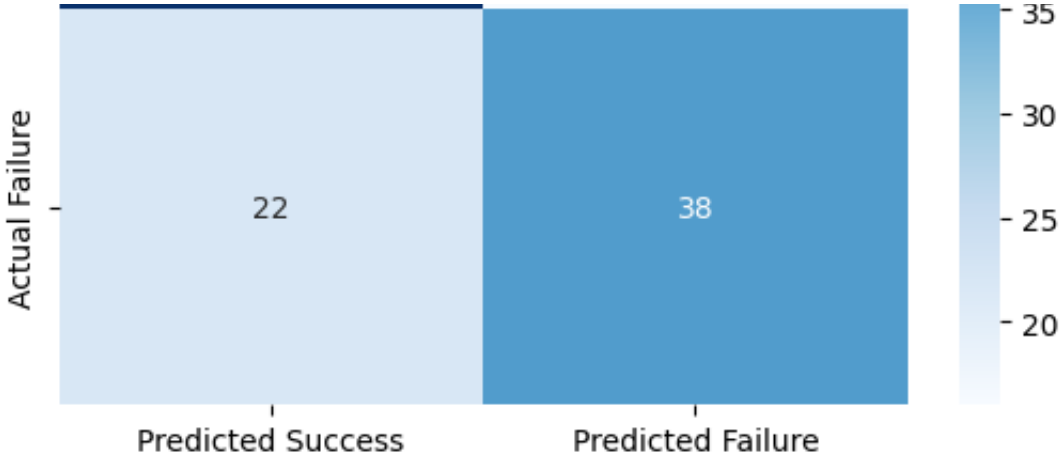


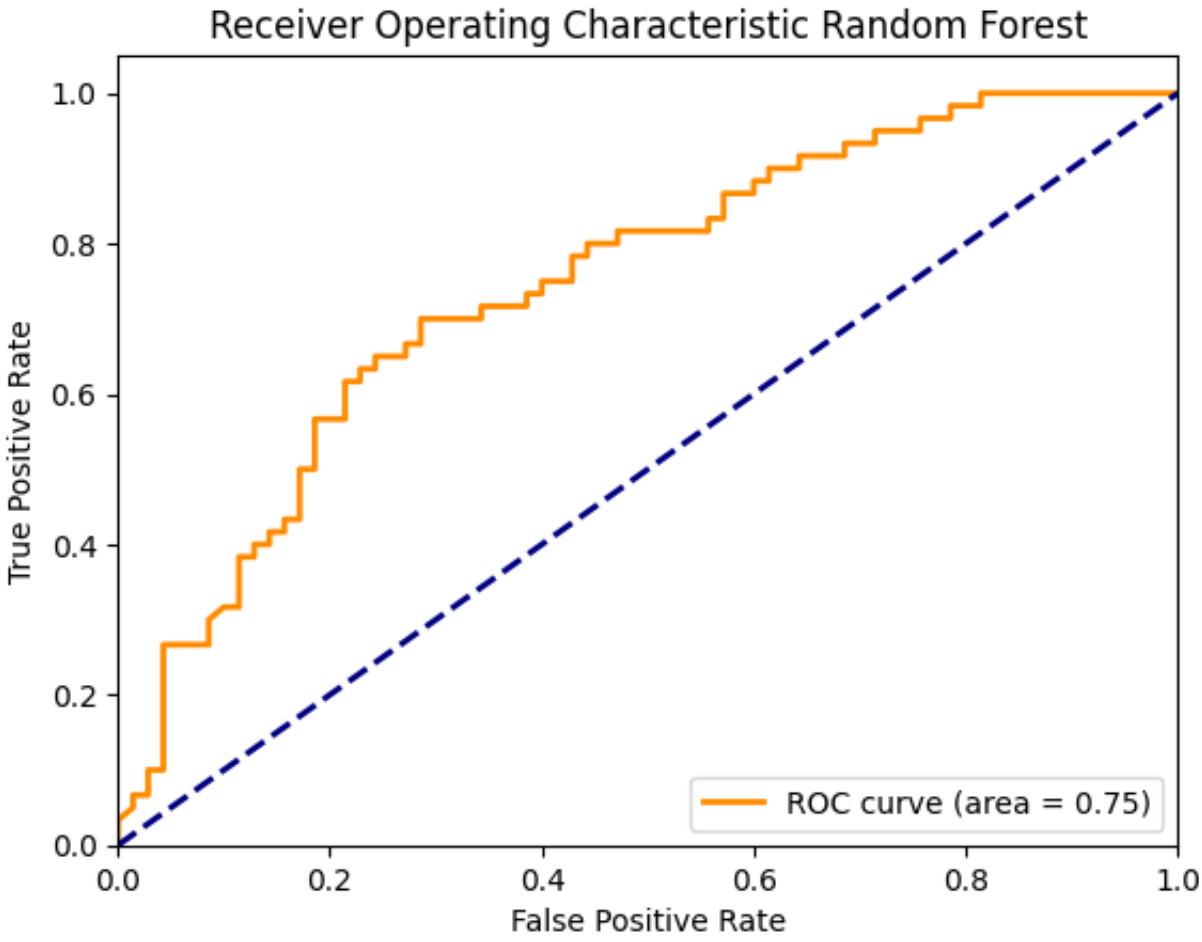

```
Running evaluation with seed 41
Evaluating Random Forest with seed 41...

--- Dados ROC para copiar ---
FPR = [0.0, 0.0, 0.0, 0.014285714285714285, 0.014285714285714285, 0.0285714
TPR = [0.0, 0.016666666666666666, 0.03333333333333333, 0.05, 0.066666666666
AUC = 0.7457142857142857
--- Fim dos Dados ROC ---

Training - Accuracy: 0.7218934911242604, Sensitivity: 0.7569721115537849, S
Metrics for manual threshold 0.55:
Accuracy: 0.7076923076923077, Sensitivity: 0.6333333333333333, Specificity:
Threshold: 0.10, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.15, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.20, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.25, Metrics: {'Accuracy': 0.46923076923076923, 'Sensitivity':
Threshold: 0.30, Metrics: {'Accuracy': 0.5153846153846153, 'Sensitivity': 1
Threshold: 0.35, Metrics: {'Accuracy': 0.5538461538461539, 'Sensitivity': 0
Threshold: 0.40, Metrics: {'Accuracy': 0.6, 'Sensitivity': 0.93333333333333
Threshold: 0.45, Metrics: {'Accuracy': 0.6153846153846154, 'Sensitivity': 0
Threshold: 0.50, Metrics: {'Accuracy': 0.6615384615384615, 'Sensitivity': 0
Threshold: 0.55, Metrics: {'Accuracy': 0.7076923076923077, 'Sensitivity': 0
Threshold: 0.60, Metrics: {'Accuracy': 0.6538461538461539, 'Sensitivity': 0
Threshold: 0.65, Metrics: {'Accuracy': 0.6230769230769231, 'Sensitivity': 0
Threshold: 0.70, Metrics: {'Accuracy': 0.5692307692307692, 'Sensitivity': 0
Threshold: 0.75, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.80, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.85, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.90, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.95, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 1.00, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
SHAP Summary for Random Forest
```




Age
Sex
−0.1
0.0
0.1
−0.1
0.0
0.1
SHAP interaction val

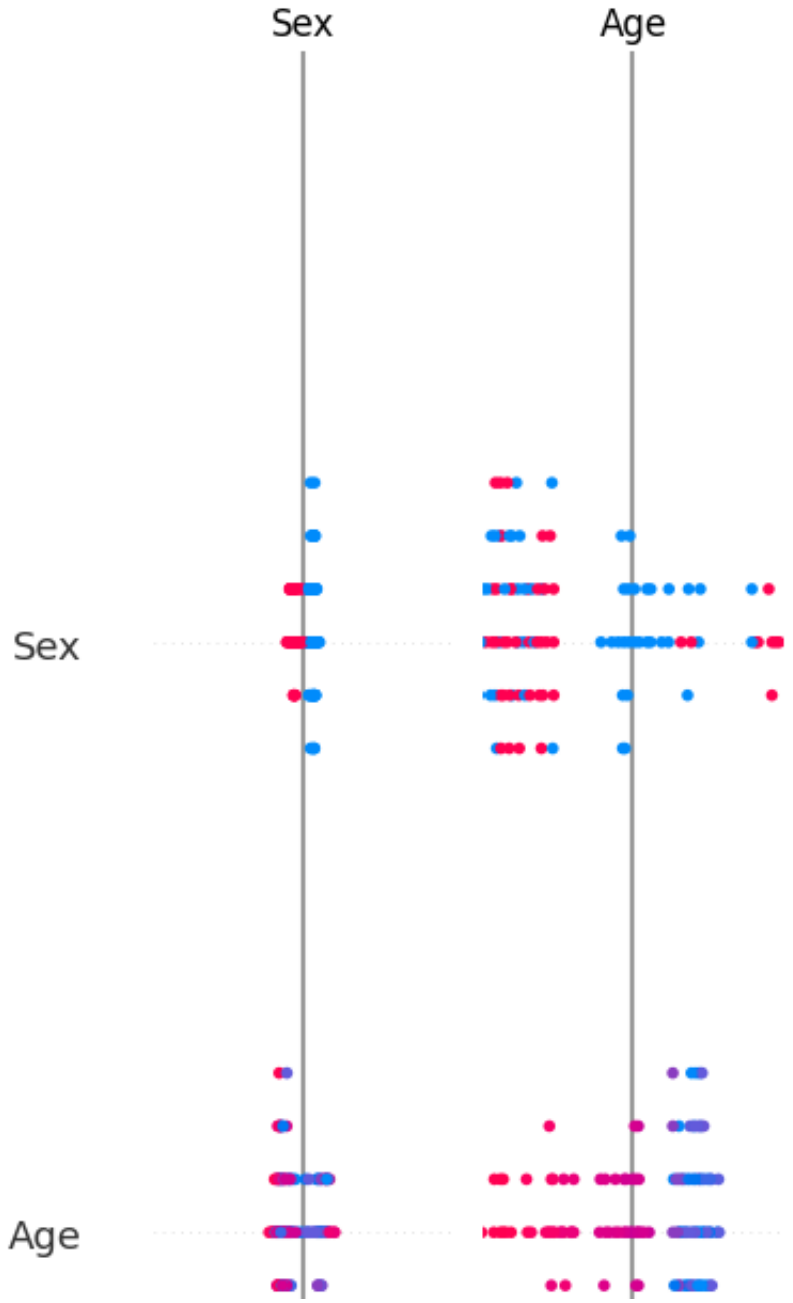

Confusion Matrix Random Forest
Actual Success
54
16
50
45
40

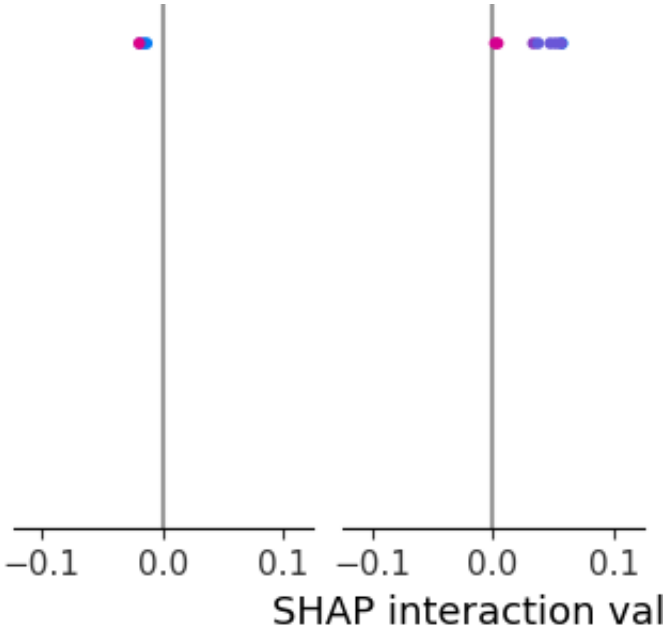


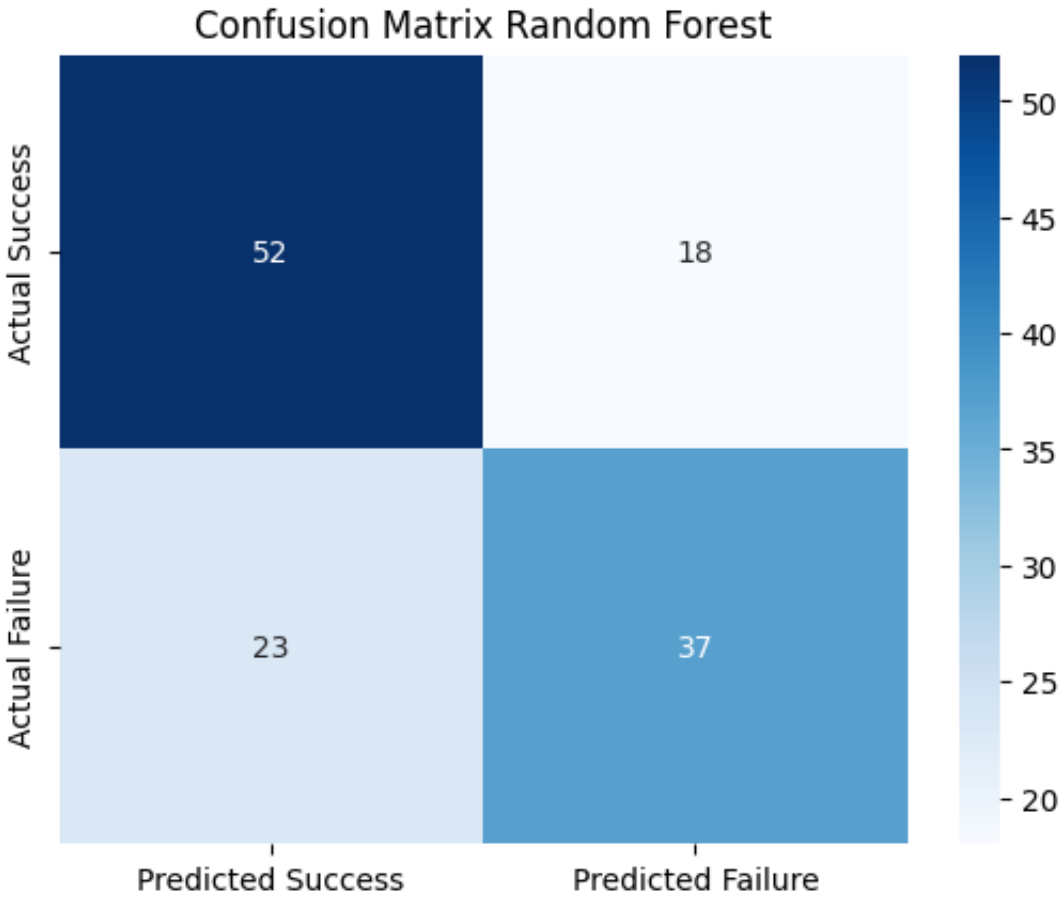


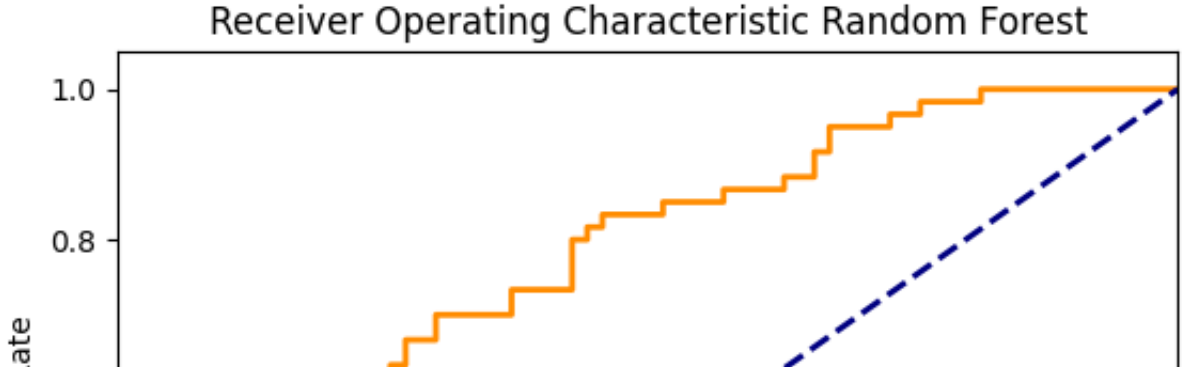

```
Running evaluation with seed 42
Evaluating Random Forest with seed 42...

--- Dados ROC para copiar ---
FPR = [0.0, 0.0, 0.0, 0.014285714285714285, 0.014285714285714285, 0.0285714
TPR = [0.0, 0.016666666666666666, 0.03333333333333333, 0.05, 0.066666666666
AUC = 0.75
--- Fim dos Dados ROC ---

Training - Accuracy: 0.7140039447731755, Sensitivity: 0.7569721115537849, S
Metrics for manual threshold 0.55:
Accuracy: 0.6846153846153846, Sensitivity: 0.6166666666666667, Specificity:
Threshold: 0.10, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.15, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
```

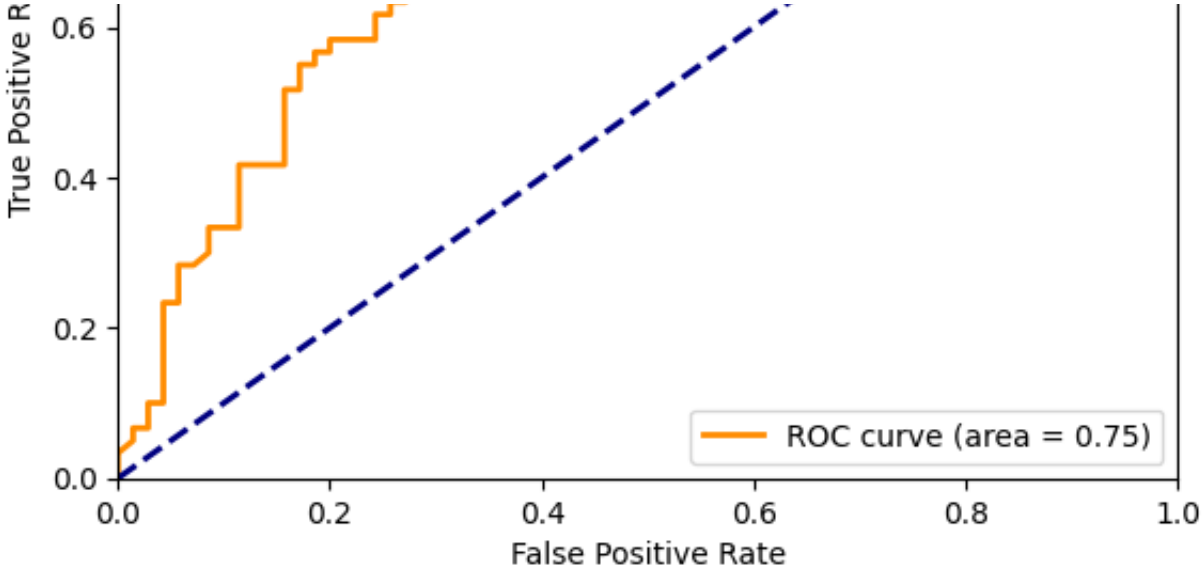
```
Threshold: 0.20, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.25, Metrics: {'Accuracy': 0.46923076923076923, 'Sensitivity':
Threshold: 0.30, Metrics: {'Accuracy': 0.5, 'Sensitivity': 1.0, 'Specificit
Threshold: 0.35, Metrics: {'Accuracy': 0.5615384615384615, 'Sensitivity': 0
Threshold: 0.40, Metrics: {'Accuracy': 0.6076923076923076, 'Sensitivity': 0
Threshold: 0.45, Metrics: {'Accuracy': 0.6461538461538462, 'Sensitivity': 0
Threshold: 0.50, Metrics: {'Accuracy': 0.6461538461538462, 'Sensitivity': 0
Threshold: 0.55, Metrics: {'Accuracy': 0.6846153846153846, 'Sensitivity': 0
Threshold: 0.60, Metrics: {'Accuracy': 0.6615384615384615, 'Sensitivity': 0
Threshold: 0.65, Metrics: {'Accuracy': 0.6153846153846154, 'Sensitivity': 0
Threshold: 0.70, Metrics: {'Accuracy': 0.5692307692307692, 'Sensitivity': 0
Threshold: 0.75, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.80, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.85, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.90, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.95, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 1.00, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
SHAP Summary for Random Forest
```



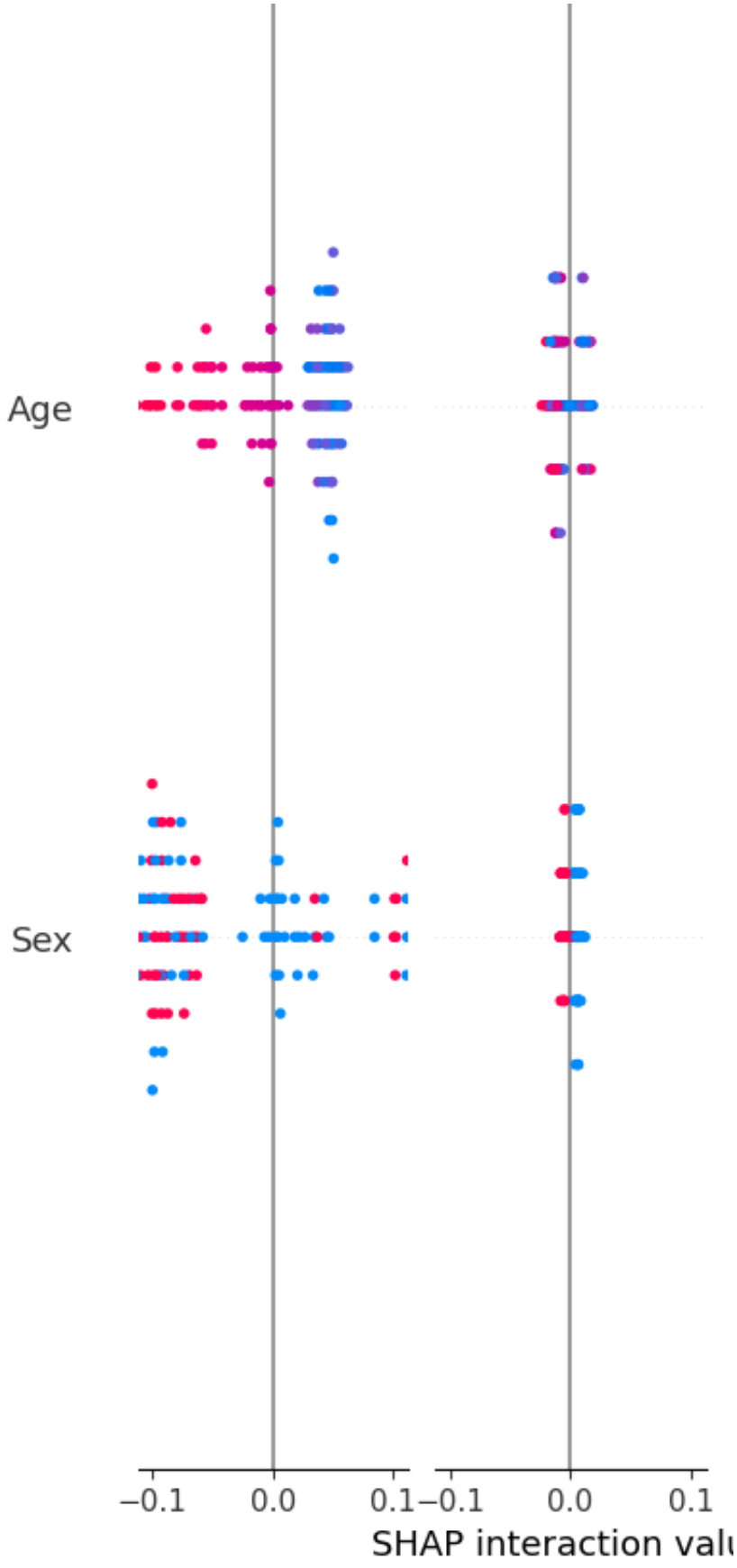

−0.1
0.0
0.1
−0.1
0.0
0.1
SHAP interaction val


Confusion Matrix Random Forest
Actual Success
Actual Failure
52
18
23
37
Predicted Success
Predicted Failure
50
45
40
35
30
25
20

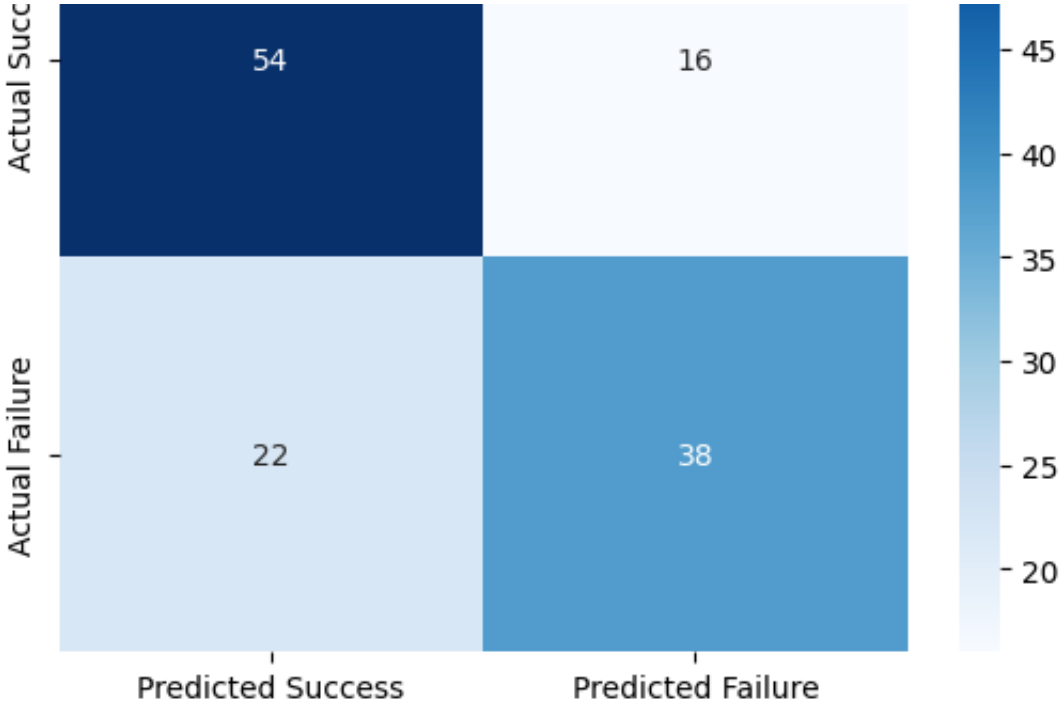

Receiver Operating Characteristic Random Forest
1.0
0.8

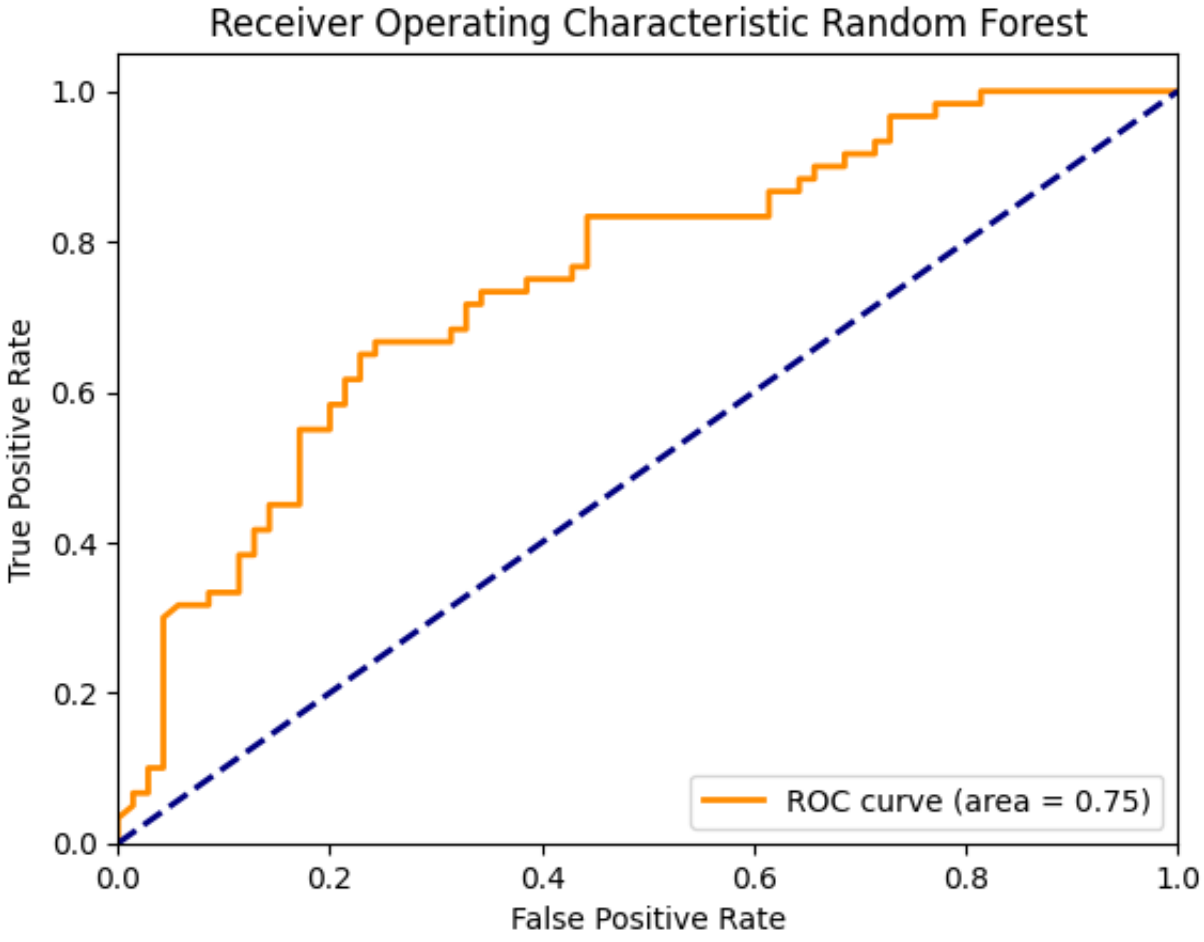



```
Running evaluation with seed 43
Evaluating Random Forest with seed 43...

--- Dados ROC para copiar ---
FPR = [0.0, 0.0, 0.0, 0.014285714285714285, 0.014285714285714285, 0.0285714
TPR = [0.0, 0.016666666666666666, 0.03333333333333333, 0.05, 0.066666666666
AUC = 0.7492857142857143
--- Fim dos Dados ROC ---

Training - Accuracy: 0.7199211045364892, Sensitivity: 0.7529880478087649, S
Metrics for manual threshold 0.55:
Accuracy: 0.7076923076923077, Sensitivity: 0.6333333333333333, Specificity:
Threshold: 0.10, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.15, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.20, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.25, Metrics: {'Accuracy': 0.46923076923076923, 'Sensitivity':
Threshold: 0.30, Metrics: {'Accuracy': 0.5153846153846153, 'Sensitivity': 1
Threshold: 0.35, Metrics: {'Accuracy': 0.5769230769230769, 'Sensitivity': 0
Threshold: 0.40, Metrics: {'Accuracy': 0.5846153846153846, 'Sensitivity': 0
Threshold: 0.45, Metrics: {'Accuracy': 0.6384615384615384, 'Sensitivity': 0
Threshold: 0.50, Metrics: {'Accuracy': 0.6692307692307692, 'Sensitivity': 0
Threshold: 0.55, Metrics: {'Accuracy': 0.7076923076923077, 'Sensitivity': 0
Threshold: 0.60, Metrics: {'Accuracy': 0.6615384615384615, 'Sensitivity': 0
Threshold: 0.65, Metrics: {'Accuracy': 0.6384615384615384, 'Sensitivity': 0
Threshold: 0.70, Metrics: {'Accuracy': 0.5692307692307692, 'Sensitivity': 0
Threshold: 0.75, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.80, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.85, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.90, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.95, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 1.00, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
SHAP Summary for Random Forest
```


Age
Sex
−0.1
0.0
0.1
−0.1
0.0
0.1
SHAP interaction valu

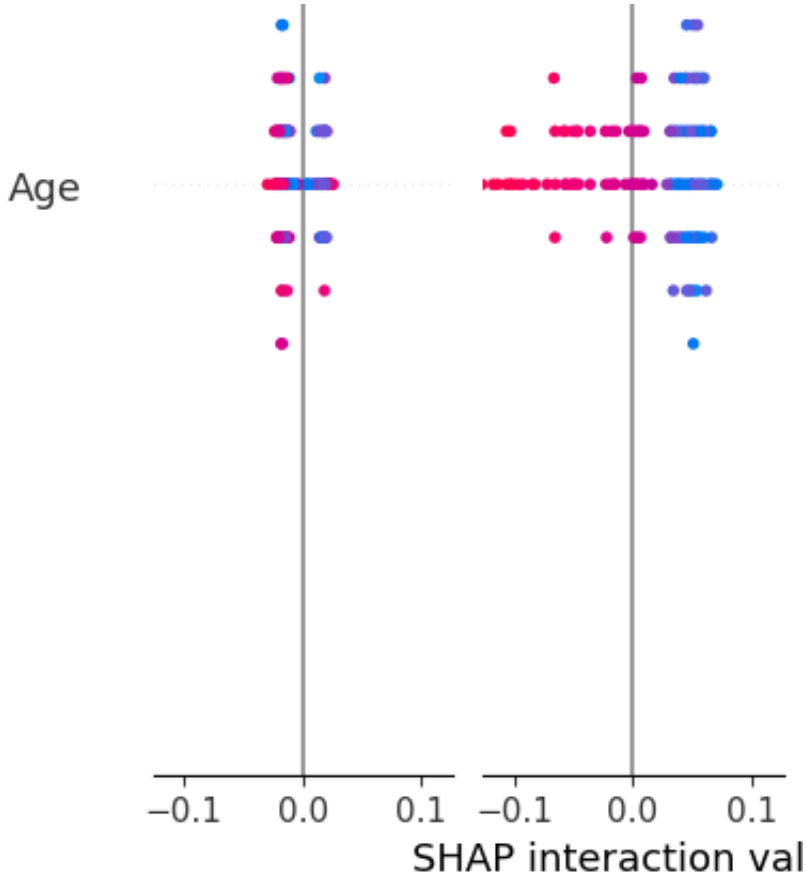

Confusion Matrix Random Forest
50

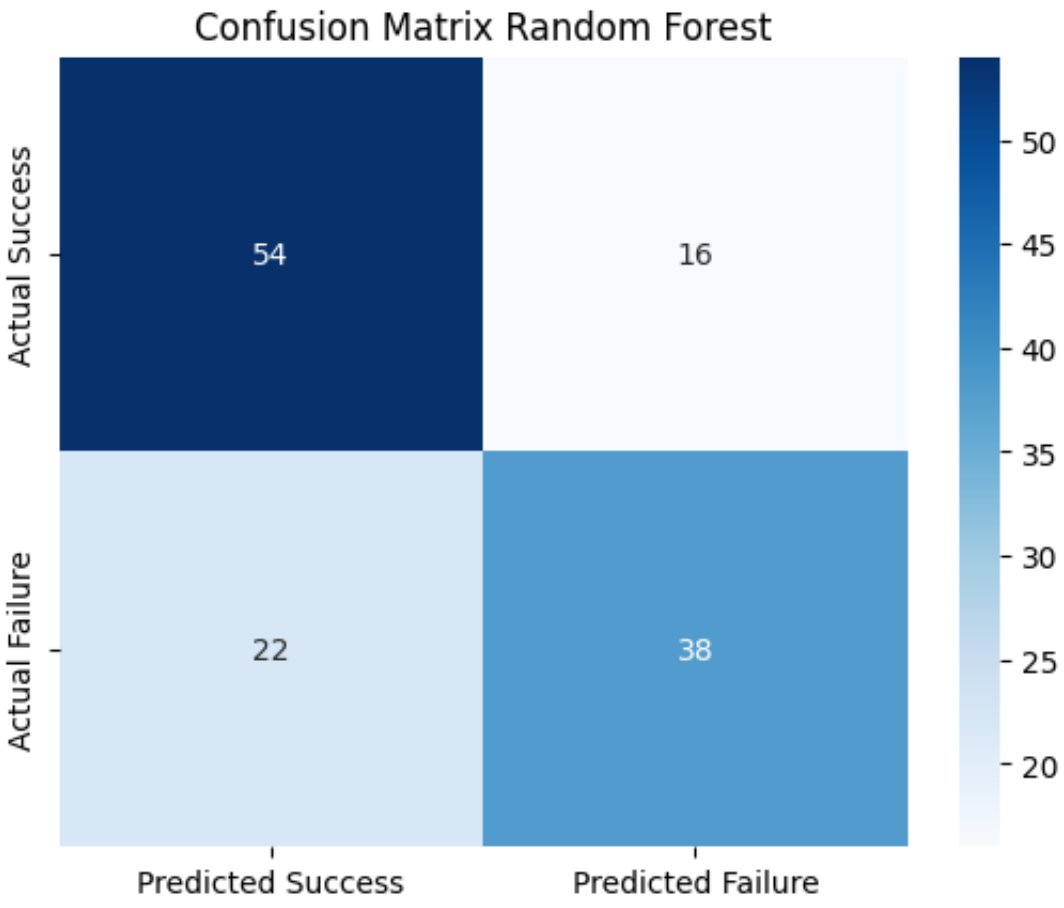


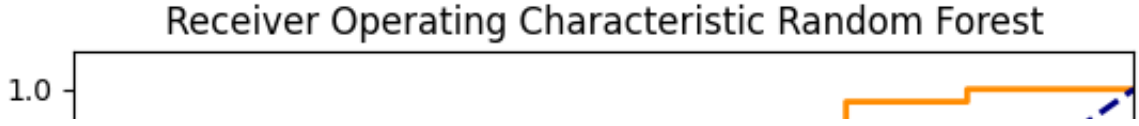


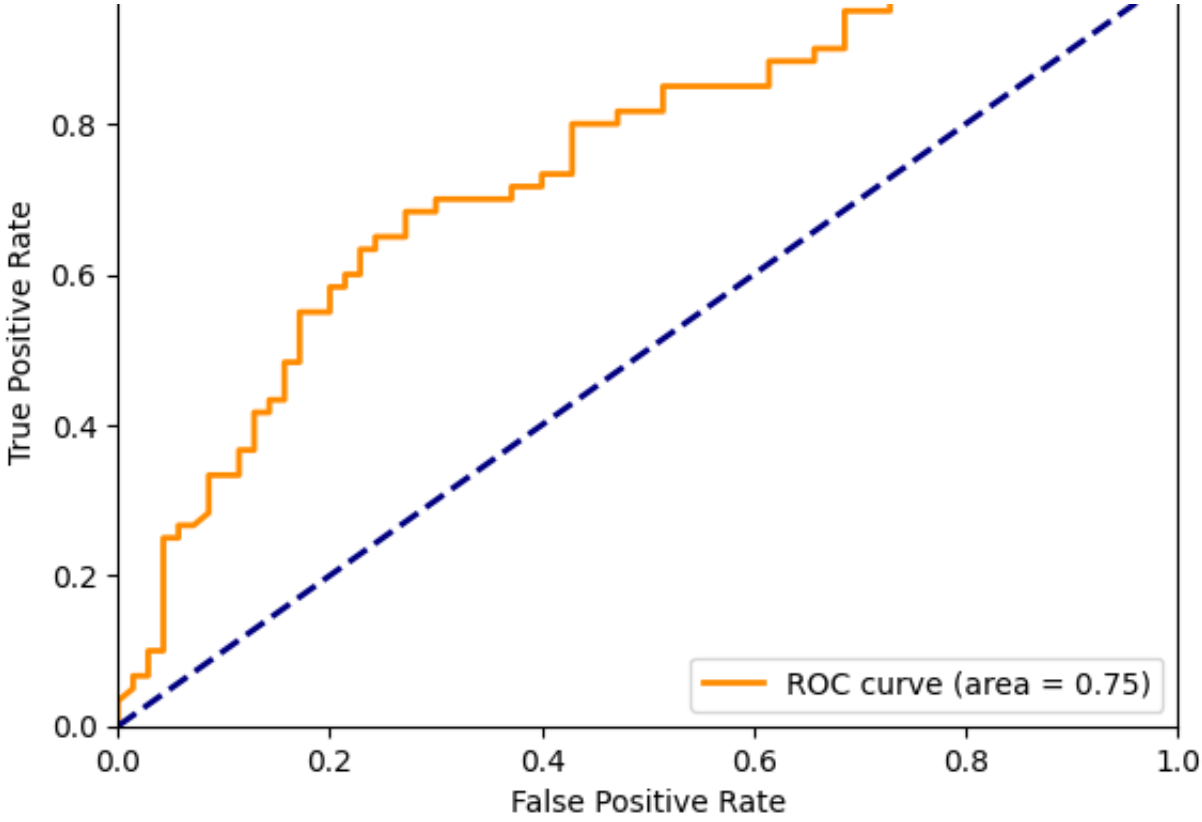

```
Running evaluation with seed 44
Evaluating Random Forest with seed 44...

--- Dados ROC para copiar ---
FPR = [0.0, 0.0, 0.0, 0.014285714285714285, 0.014285714285714285, 0.0285714
TPR = [0.0, 0.016666666666666666, 0.03333333333333333, 0.05, 0.066666666666
AUC = 0.7473809523809524
```

```
--- Fim dos Dados ROC ---

Training - Accuracy: 0.717948717948718, Sensitivity: 0.7569721115537849, Sp
Metrics for manual threshold 0.55:
Accuracy: 0.7076923076923077, Sensitivity: 0.6333333333333333, Specificity:
Threshold: 0.10, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.15, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.20, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.25, Metrics: {'Accuracy': 0.46923076923076923, 'Sensitivity':
Threshold: 0.30, Metrics: {'Accuracy': 0.5, 'Sensitivity': 1.0, 'Specificit
Threshold: 0.35, Metrics: {'Accuracy': 0.5769230769230769, 'Sensitivity': 0
Threshold: 0.40, Metrics: {'Accuracy': 0.6076923076923076, 'Sensitivity': 0
Threshold: 0.45, Metrics: {'Accuracy': 0.6461538461538462, 'Sensitivity': 0
Threshold: 0.50, Metrics: {'Accuracy': 0.6461538461538462, 'Sensitivity': 0
Threshold: 0.55, Metrics: {'Accuracy': 0.7076923076923077, 'Sensitivity': 0
Threshold: 0.60, Metrics: {'Accuracy': 0.6615384615384615, 'Sensitivity': 0
Threshold: 0.65, Metrics: {'Accuracy': 0.6230769230769231, 'Sensitivity': 0
Threshold: 0.70, Metrics: {'Accuracy': 0.5692307692307692, 'Sensitivity': 0
Threshold: 0.75, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.80, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.85, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.90, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.95, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 1.00, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
SHAP Summary for Random Forest
```

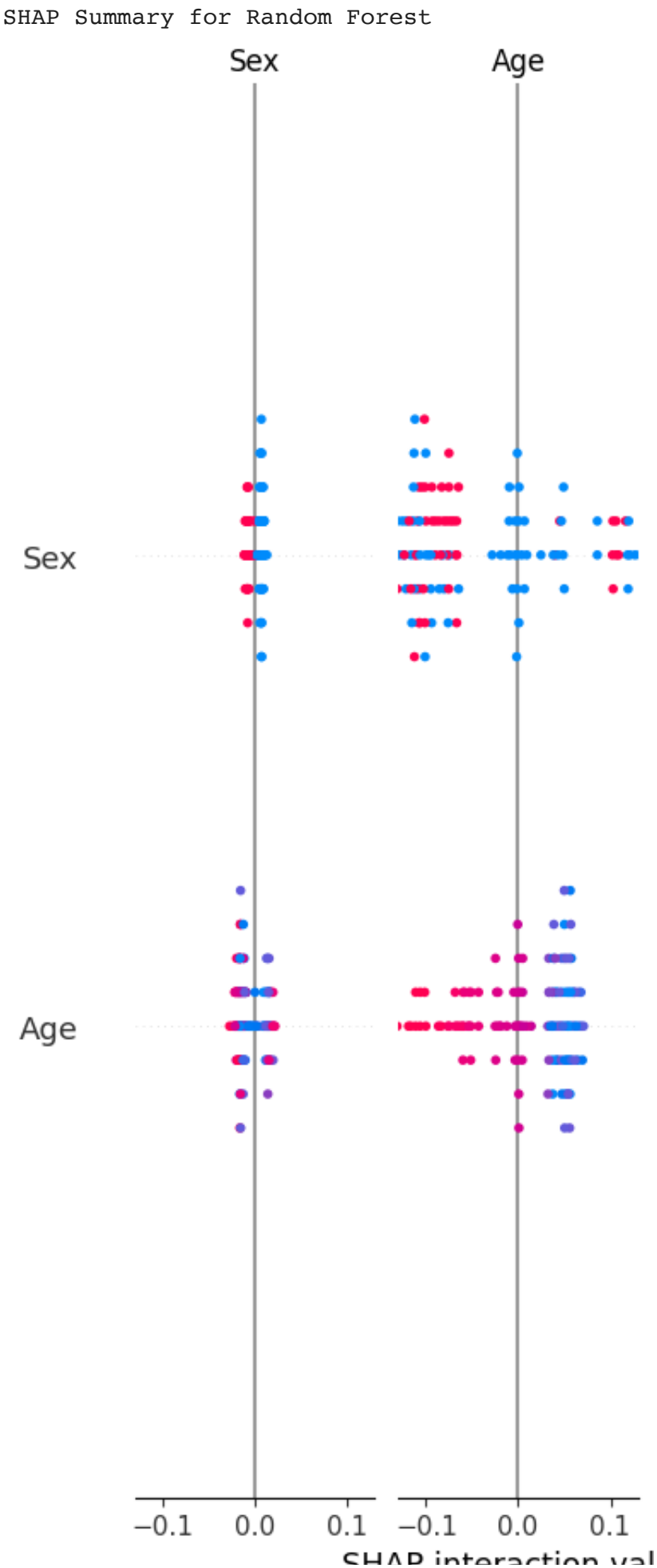

Age
−0.1
0.0
0.1
−0.1
0.0
0.1
SHAP interaction val

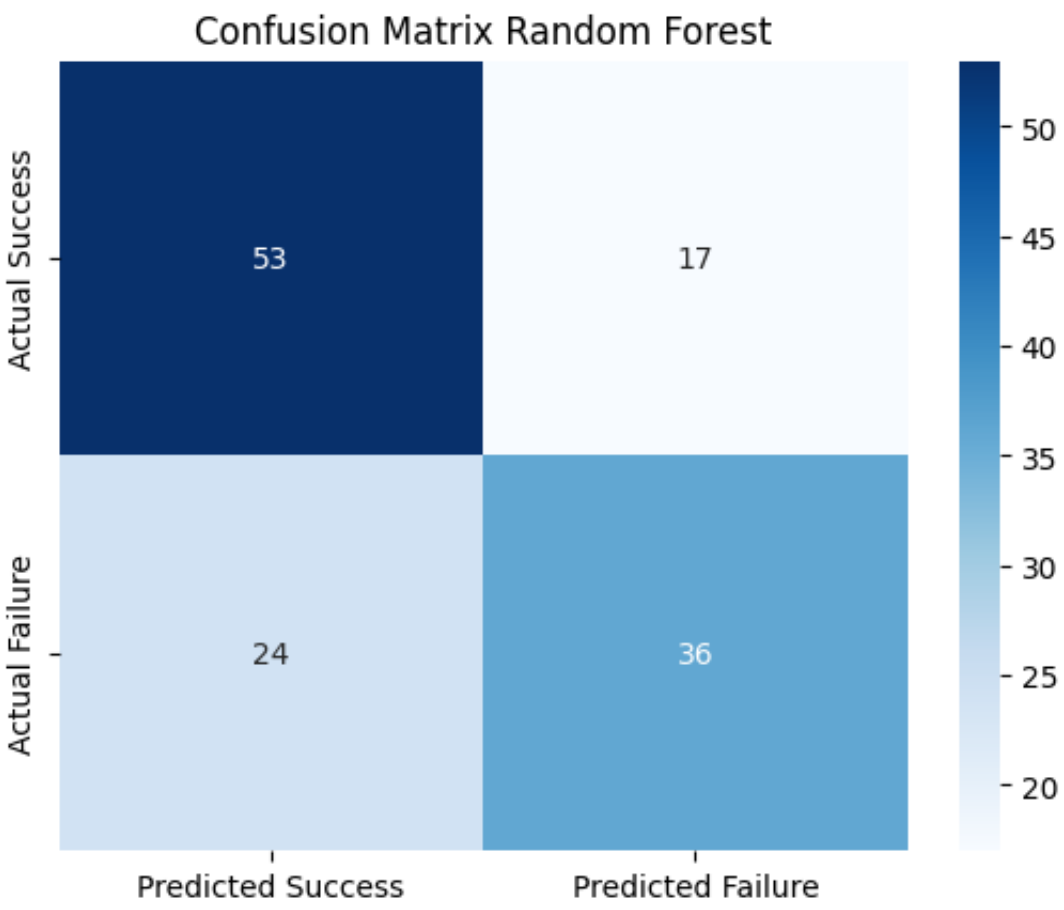

Confusion Matrix Random Forest
Actual Success
Actual Failure
54
16
22
38
Predicted Success
Predicted Failure
50
45
40
35
30
25
20

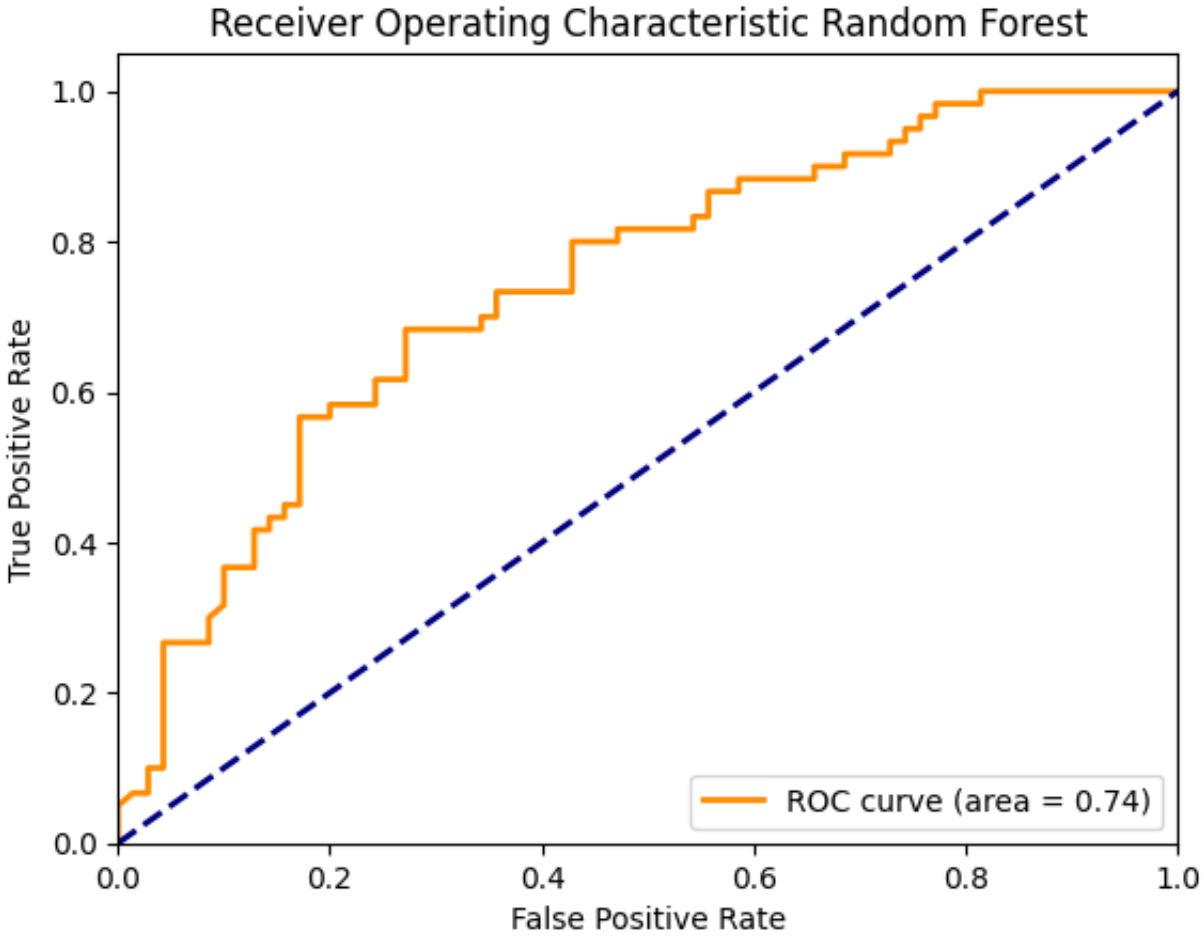

Receiver Operating Characteristic Random Forest
1.0




```
Running evaluation with seed 45
Evaluating Random Forest with seed 45...

--- Dados ROC para copiar ---
FPR = [0.0, 0.0, 0.0, 0.014285714285714285, 0.02857142857142857, 0.02857142
TPR = [0.0, 0.016666666666666666, 0.05, 0.06666666666666667, 0.066666666666
AUC = 0.7440476190476191
--- Fim dos Dados ROC ---

Training - Accuracy: 0.7238658777120316, Sensitivity: 0.7569721115537849, S
Metrics for manual threshold 0.55:
Accuracy: 0.6846153846153846, Sensitivity: 0.6, Specificity: 0.757142857142
Threshold: 0.10, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.15, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.20, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.25, Metrics: {'Accuracy': 0.4692307692307692, 'Sensitivity':
Threshold: 0.30, Metrics: {'Accuracy': 0.5076923076923077, 'Sensitivity': 1
Threshold: 0.35, Metrics: {'Accuracy': 0.5538461538461539, 'Sensitivity': 0
Threshold: 0.40, Metrics: {'Accuracy': 0.5923076923076923, 'Sensitivity': 0
Threshold: 0.45, Metrics: {'Accuracy': 0.6230769230769231, 'Sensitivity': 0
Threshold: 0.50, Metrics: {'Accuracy': 0.6538461538461539, 'Sensitivity': 0
Threshold: 0.55, Metrics: {'Accuracy': 0.6846153846153846, 'Sensitivity': 0
Threshold: 0.60, Metrics: {'Accuracy': 0.6615384615384615, 'Sensitivity': 0
Threshold: 0.65, Metrics: {'Accuracy': 0.6384615384615384, 'Sensitivity': 0
Threshold: 0.70, Metrics: {'Accuracy': 0.5692307692307692, 'Sensitivity': 0
Threshold: 0.75, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.80, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.85, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.90, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.95, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 1.00, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
```

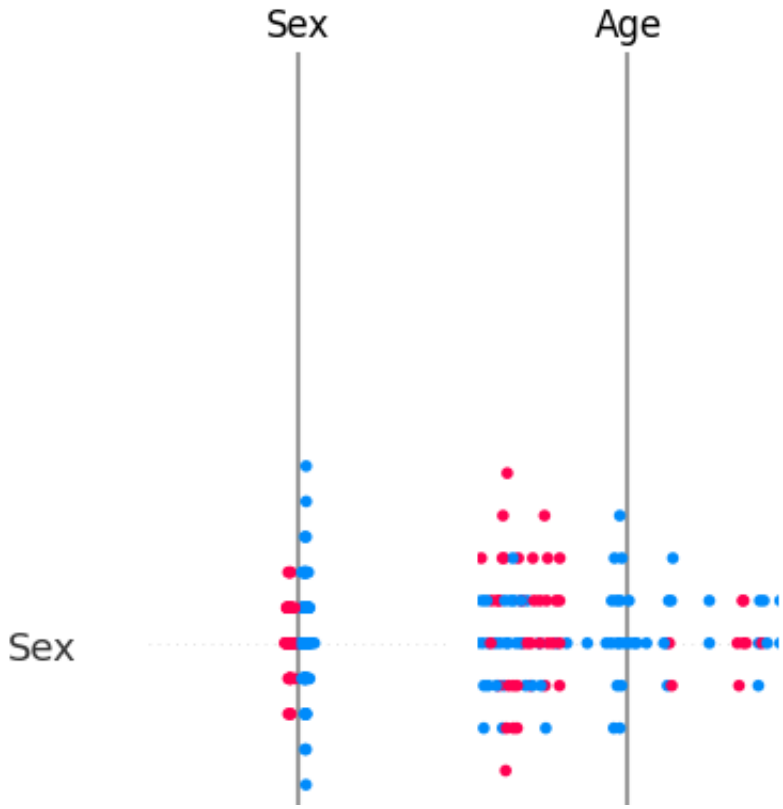

SHAP Summary for Random Forest
Sex
Age
Sex
Age
−0.1
0.0
0.1
−0.1
0.0
0.1

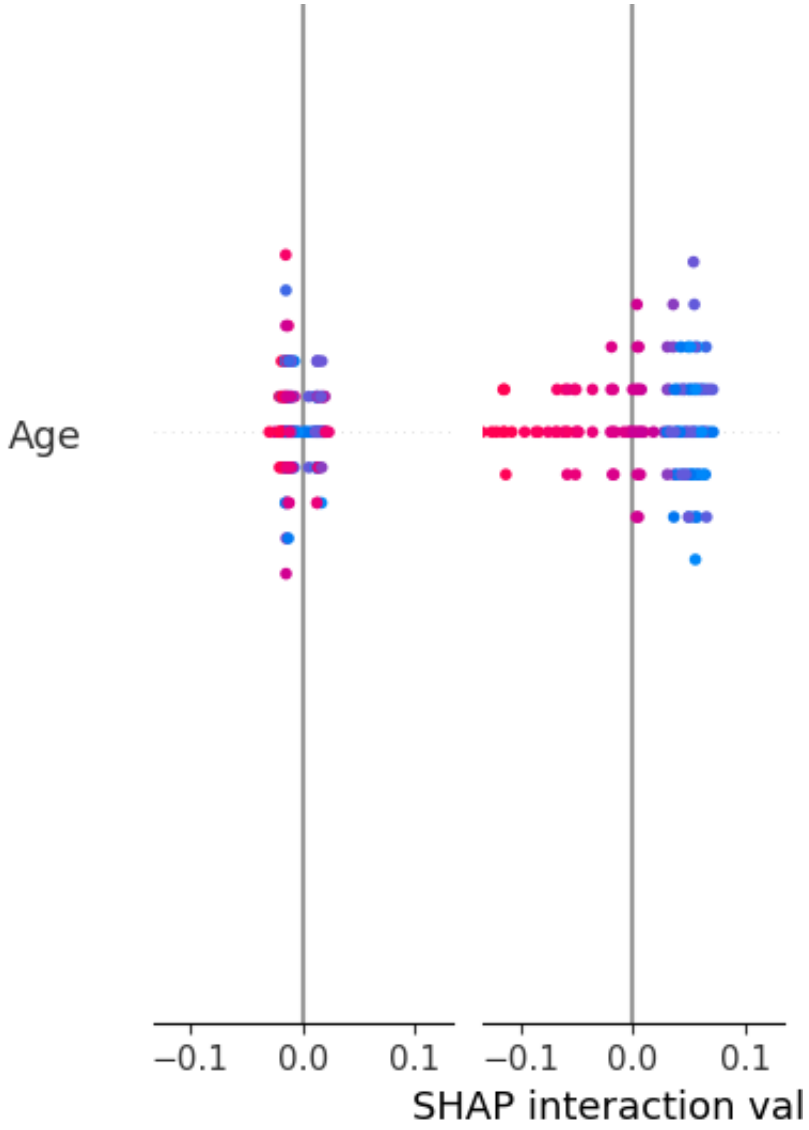


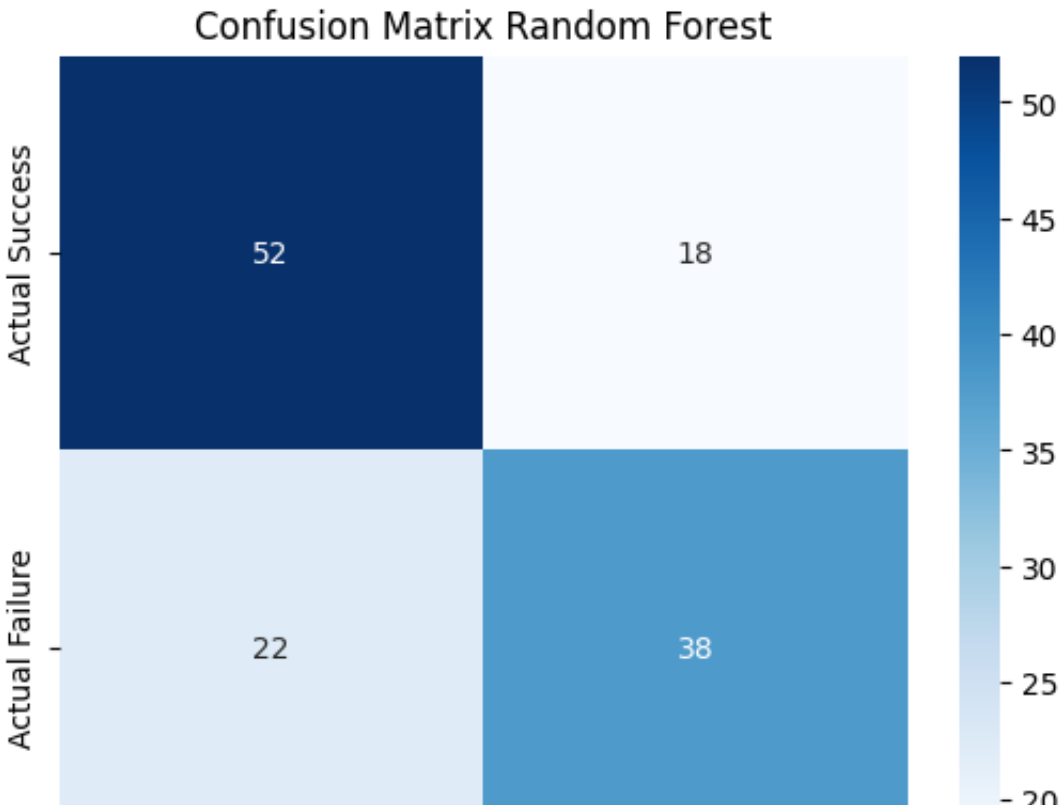


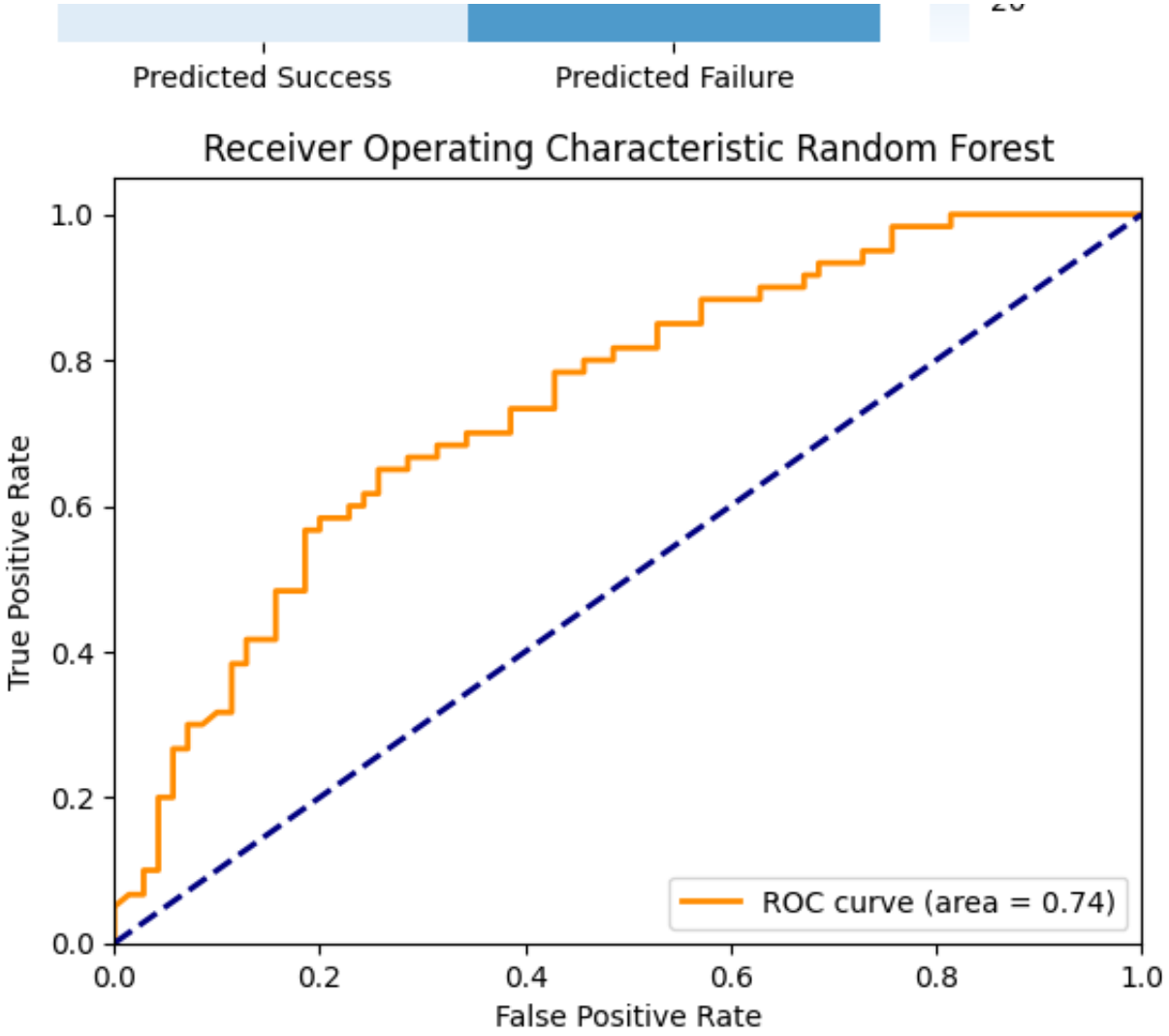
Running evaluation with seed 46

```
Running evaluation with seed 46
Evaluating Random Forest with seed 46...

--- Dados ROC para copiar ---
FPR = [0.0, 0.0, 0.0, 0.014285714285714285, 0.02857142857142857, 0.02857142
TPR = [0.0, 0.016666666666666666, 0.05, 0.06666666666666667, 0.066666666666
AUC = 0.7428571428571429
--- Fim dos Dados ROC ---

Training - Accuracy: 0.727810650887574, Sensitivity: 0.7729083665338645, Sp
Metrics for manual threshold 0.55:
Accuracy: 0.6923076923076923, Sensitivity: 0.6333333333333333, Specificity:
Threshold: 0.10, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.15, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.20, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.25, Metrics: {'Accuracy': 0.4846153846153846, 'Sensitivity': 1
Threshold: 0.30, Metrics: {'Accuracy': 0.5153846153846153, 'Sensitivity': 1
Threshold: 0.35, Metrics: {'Accuracy': 0.5692307692307692, 'Sensitivity': 0
Threshold: 0.40, Metrics: {'Accuracy': 0.6, 'Sensitivity': 0.93333333333333
Threshold: 0.45, Metrics: {'Accuracy': 0.6307692307692307, 'Sensitivity': 0
Threshold: 0.50, Metrics: {'Accuracy': 0.6615384615384615, 'Sensitivity': 0
Threshold: 0.55, Metrics: {'Accuracy': 0.6923076923076923, 'Sensitivity': 0
Threshold: 0.60, Metrics: {'Accuracy': 0.6538461538461539, 'Sensitivity': 0
Threshold: 0.65, Metrics: {'Accuracy': 0.6230769230769231, 'Sensitivity': 0
Threshold: 0.70, Metrics: {'Accuracy': 0.5692307692307692, 'Sensitivity': 0
Threshold: 0.75, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.80, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.85, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.90, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.95, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 1.00, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
SHAP Summary for Random Forest
```



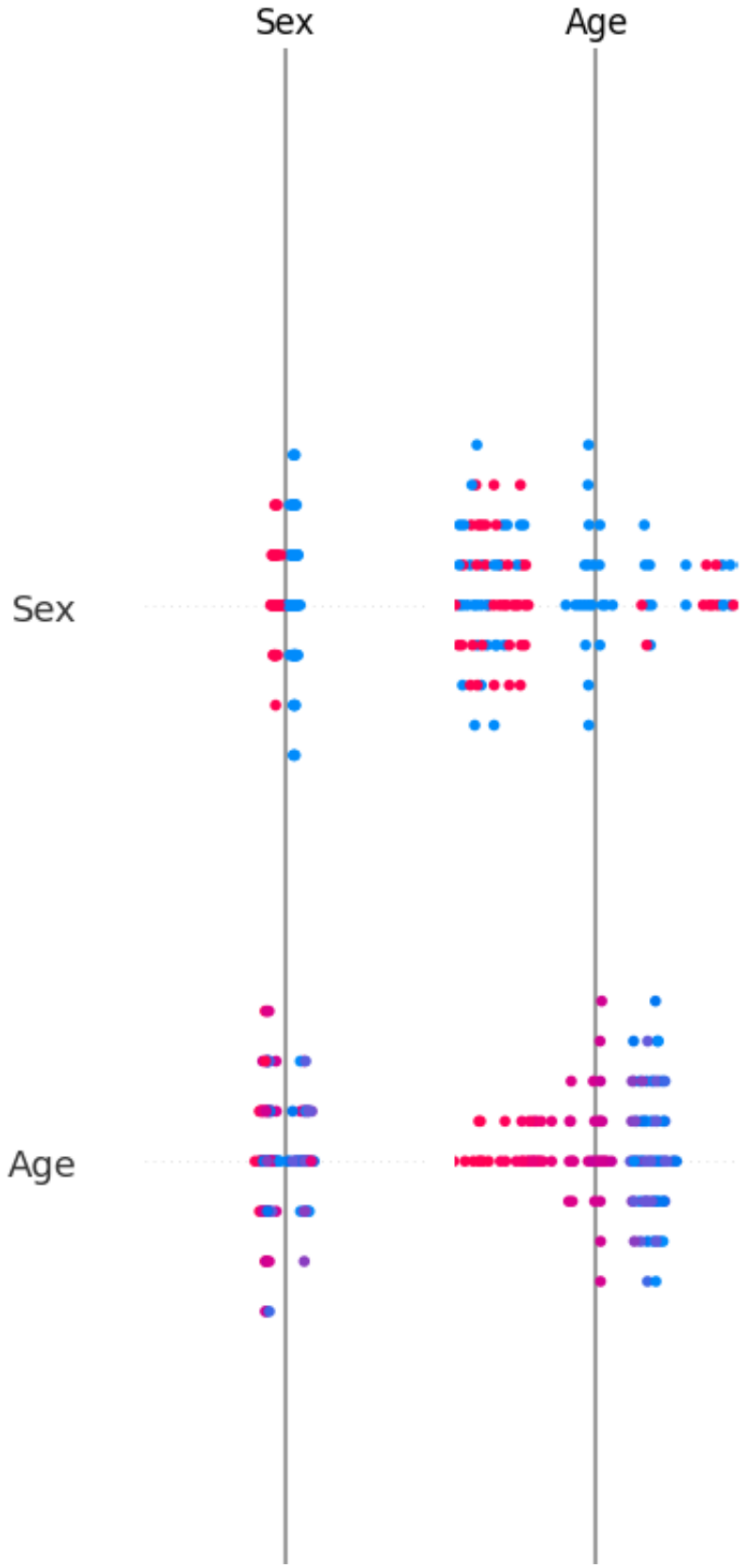

Age
−0.1
0.0
0.1
−0.1
0.0
0.1
SHAP interaction val

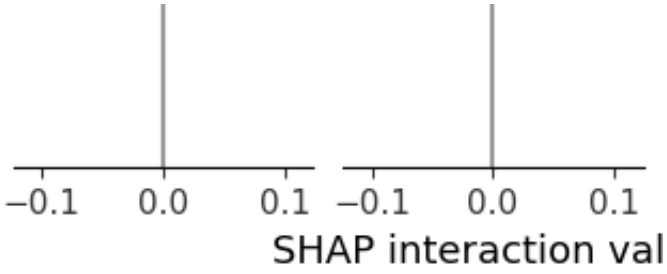

Confusion Matrix Random Forest
Actual Success
Actual Failure
52
18
22
38
50
45
40
35
30
25
20

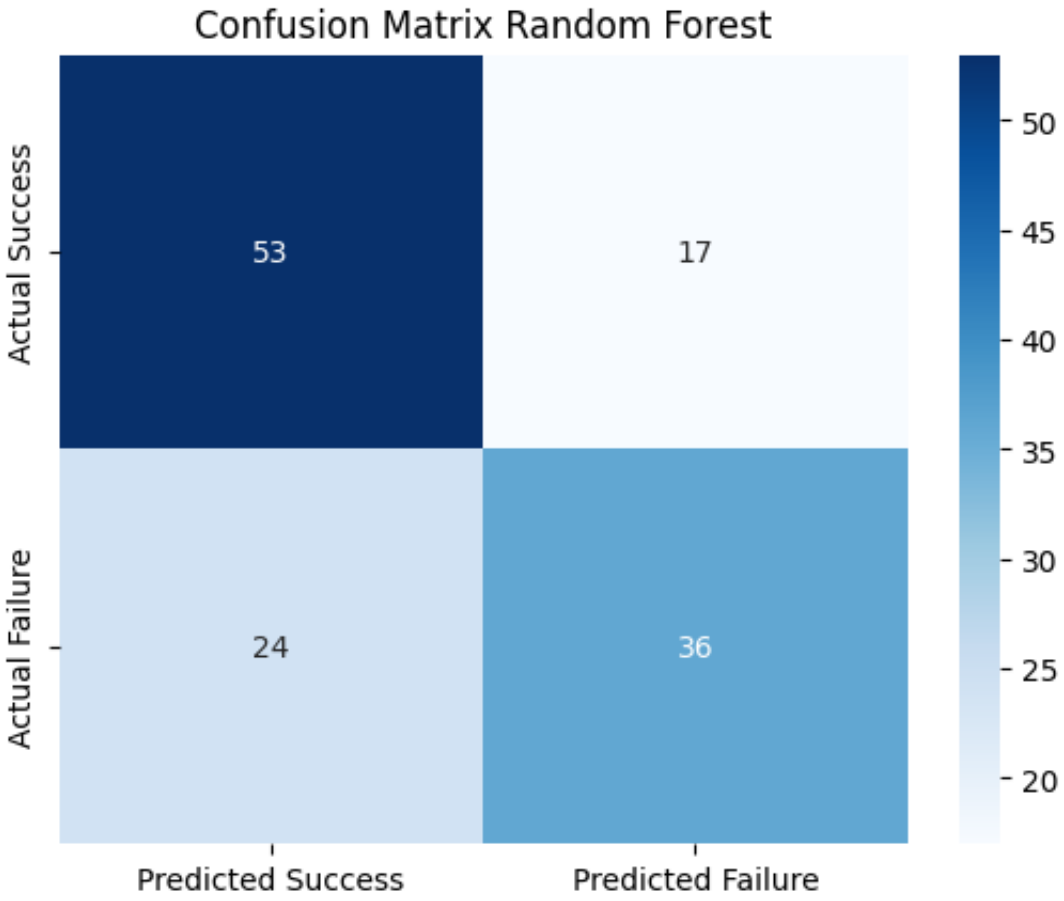


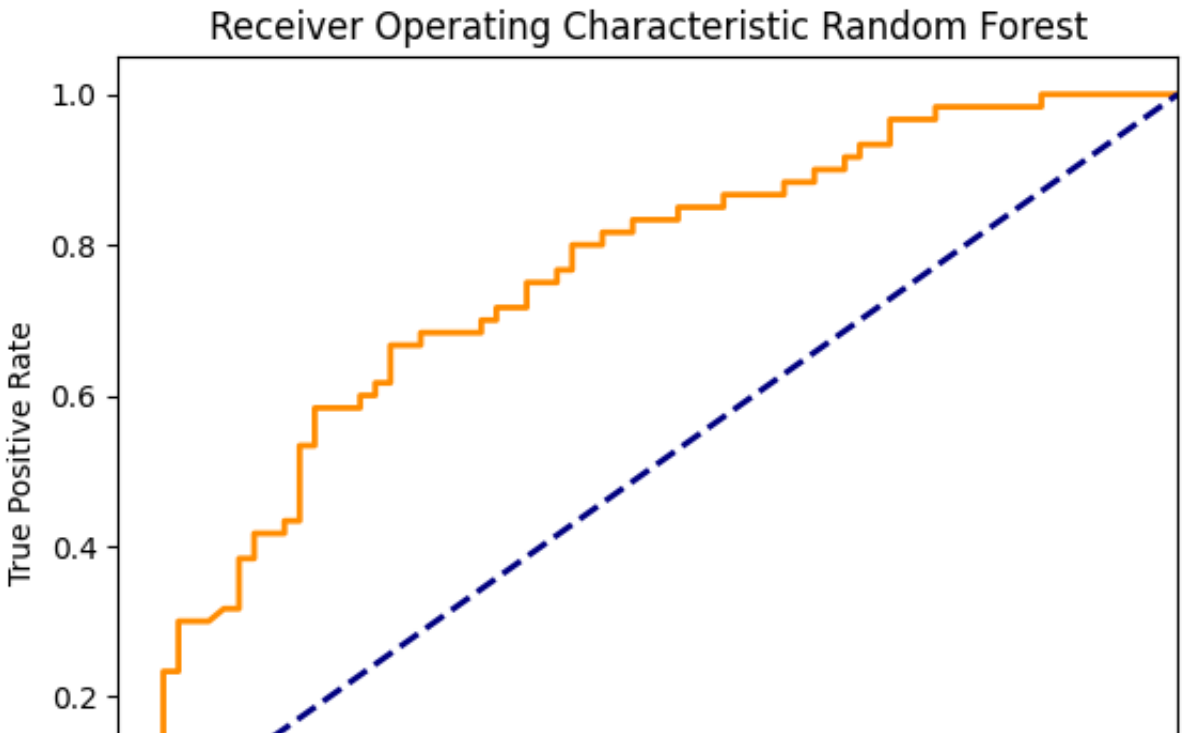

```
Running evaluation with seed 47
Evaluating Random Forest with seed 47...

--- Dados ROC para copiar ---
FPR = [0.0, 0.0, 0.0, 0.014285714285714285, 0.02857142857142857, 0.02857142
TPR = [0.0, 0.016666666666666666, 0.05, 0.06666666666666667, 0.066666666666
AUC = 0.7454761904761905
--- Fim dos Dados ROC ---

Training - Accuracy: 0.7140039447731755, Sensitivity: 0.749003984063745, Sp
Metrics for manual threshold 0.55:
Accuracy: 0.6846153846153846, Sensitivity: 0.6, Specificity: 0.757142857142
Threshold: 0.10, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.15, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.20, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.25, Metrics: {'Accuracy': 0.46923076923076923, 'Sensitivity':
Threshold: 0.30, Metrics: {'Accuracy': 0.5153846153846153, 'Sensitivity': 1
Threshold: 0.35, Metrics: {'Accuracy': 0.5692307692307692, 'Sensitivity': 0
Threshold: 0.40, Metrics: {'Accuracy': 0.5846153846153846, 'Sensitivity': 0
Threshold: 0.45, Metrics: {'Accuracy': 0.6384615384615384, 'Sensitivity': 0
Threshold: 0.50, Metrics: {'Accuracy': 0.676923076923077, 'Sensitivity': 0.
Threshold: 0.55, Metrics: {'Accuracy': 0.6846153846153846, 'Sensitivity': 0
Threshold: 0.60, Metrics: {'Accuracy': 0.6538461538461539, 'Sensitivity': 0
Threshold: 0.65, Metrics: {'Accuracy': 0.6307692307692307, 'Sensitivity': 0
```

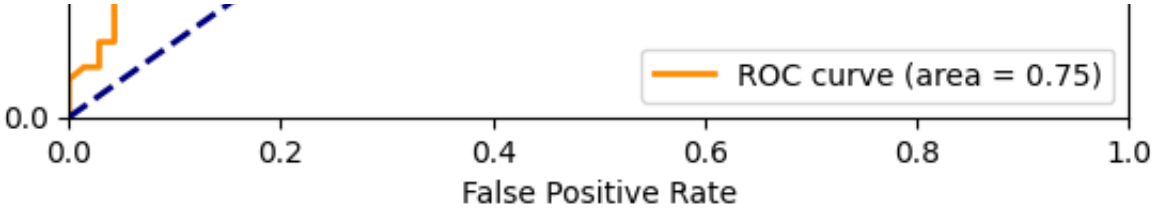
```
Threshold: 0.70, Metrics: {'Accuracy': 0.5692307692307692, 'Sensitivity': 0
Threshold: 0.75, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.80, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.85, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.90, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.95, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 1.00, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
SHAP Summary for Random Forest
```



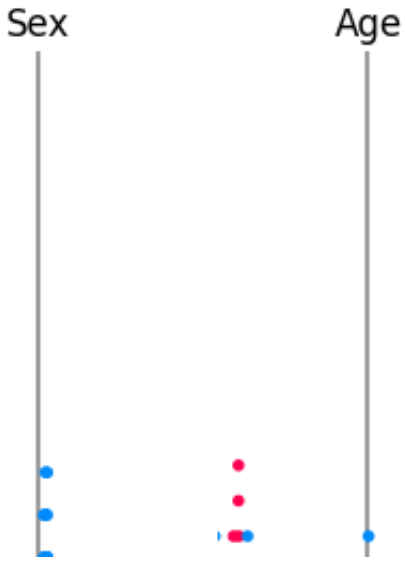

−0.1
0.0
0.1
−0.1
0.0
0.1
SHAP interaction val

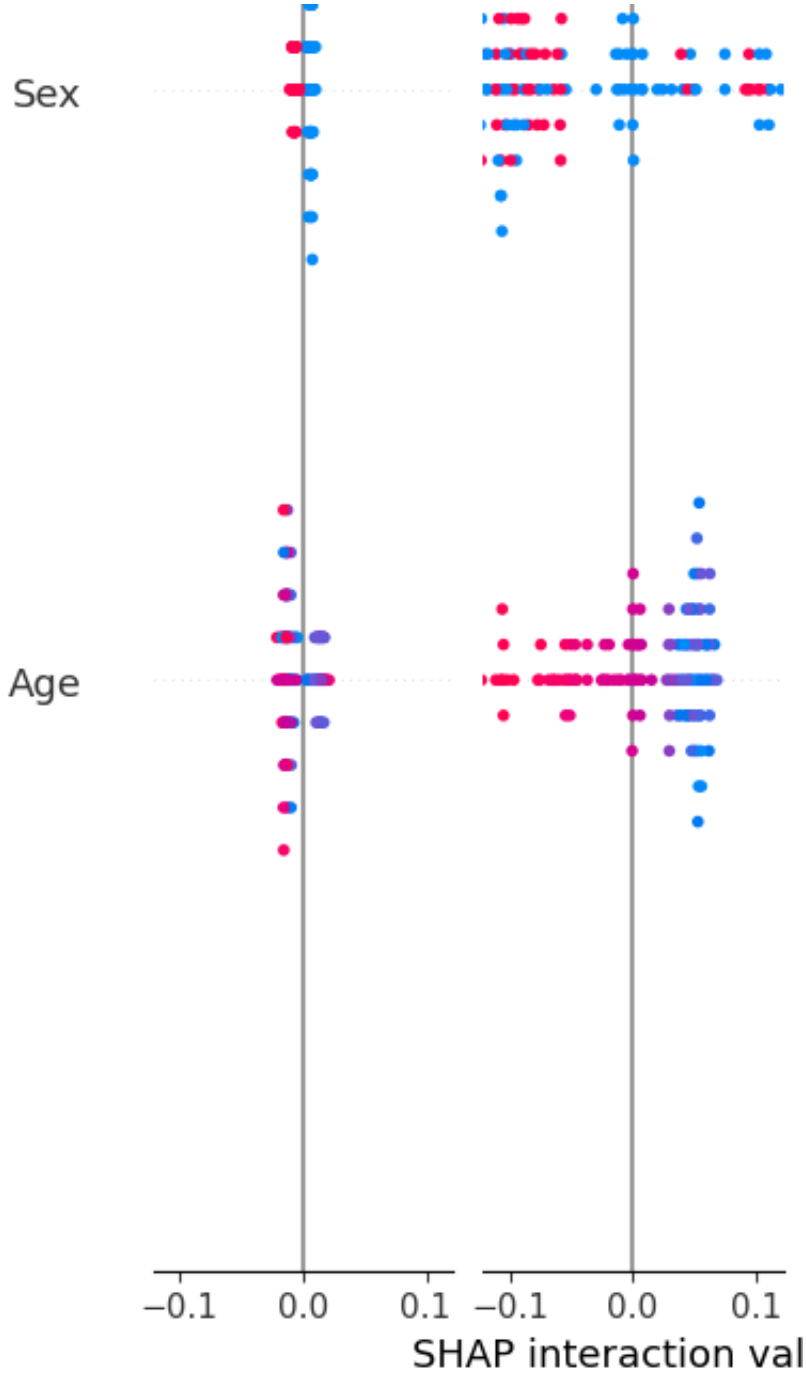

Confusion Matrix Random Forest
Actual Success
Actual Failure
53
17
24
36
Predicted Success
Predicted Failure
50
45
40
35
30
25
20

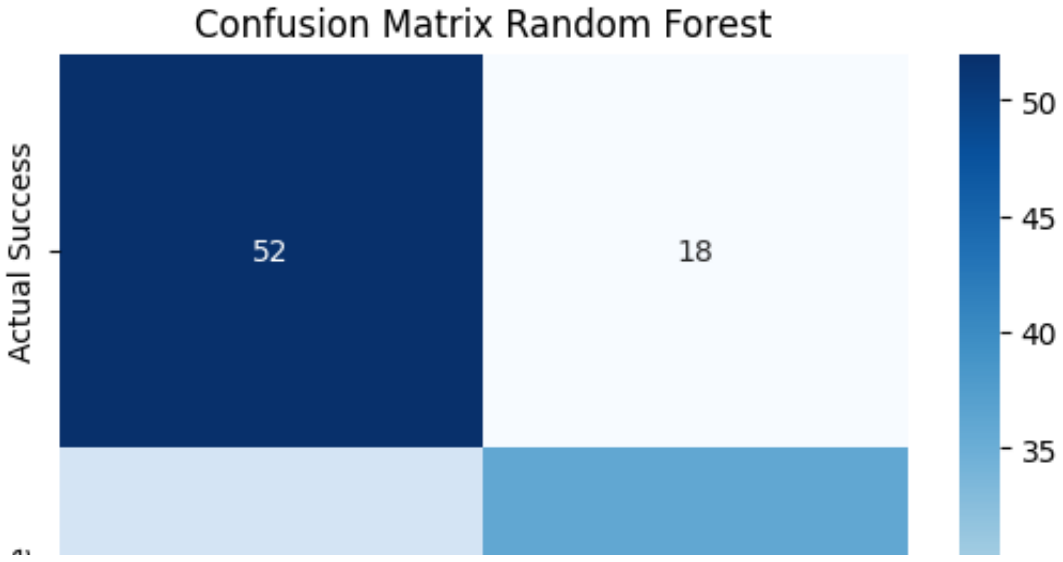

Receiver Operating Characteristic Random Forest
True Positive Rate
1.0
0.8
0.6
0.4
0.2

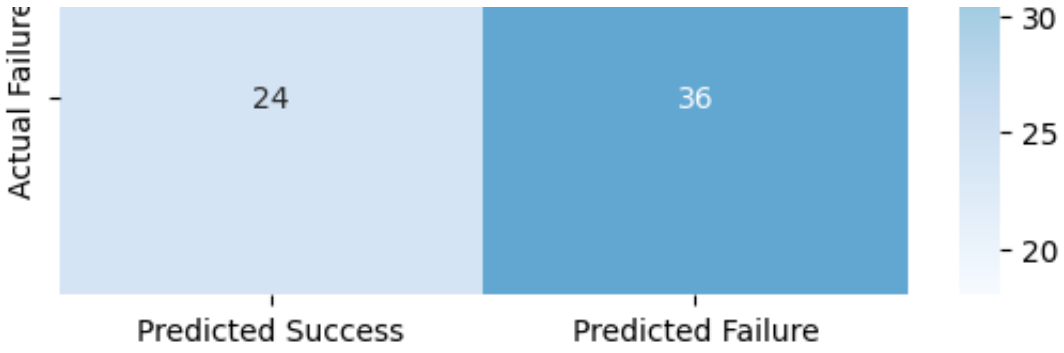


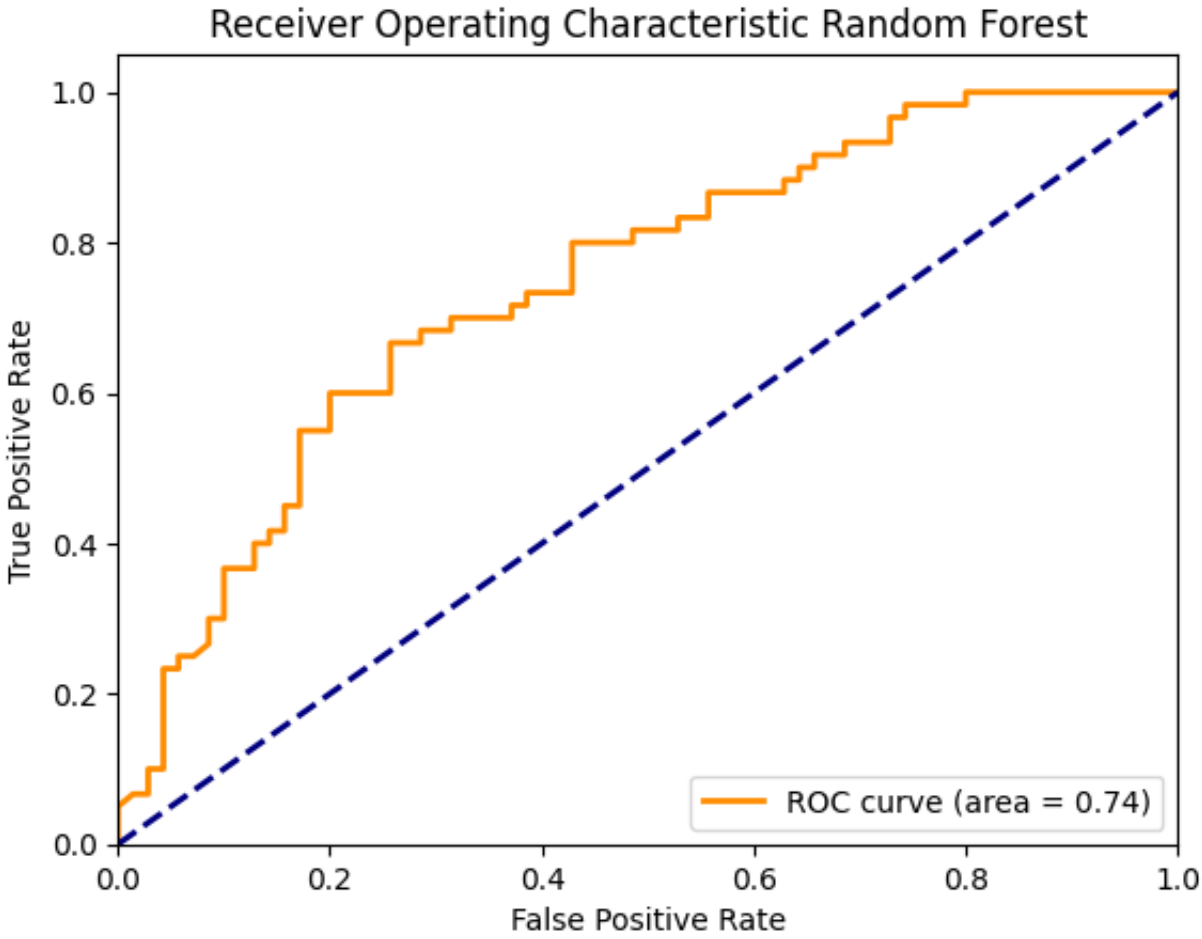

```
Running evaluation with seed 48
Evaluating Random Forest with seed 48...

--- Dados ROC para copiar ---
FPR = [0.0, 0.0, 0.0, 0.014285714285714285, 0.02857142857142857, 0.02857142
TPR = [0.0, 0.016666666666666666, 0.05, 0.06666666666666667, 0.066666666666
AUC = 0.745
--- Fim dos Dados ROC ---

Training - Accuracy: 0.7199211045364892, Sensitivity: 0.749003984063745, Sp
Metrics for manual threshold 0.55:
Accuracy: 0.676923076923077, Sensitivity: 0.6, Specificity: 0.7428571428571
Threshold: 0.10, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.15, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.20, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.25, Metrics: {'Accuracy': 0.46923076923076923, 'Sensitivity':
Threshold: 0.30, Metrics: {'Accuracy': 0.5153846153846153, 'Sensitivity': 1
Threshold: 0.35, Metrics: {'Accuracy': 0.5615384615384615, 'Sensitivity': 0
Threshold: 0.40, Metrics: {'Accuracy': 0.5923076923076923, 'Sensitivity': 0
Threshold: 0.45, Metrics: {'Accuracy': 0.6384615384615384, 'Sensitivity': 0
Threshold: 0.50, Metrics: {'Accuracy': 0.6692307692307692, 'Sensitivity': 0
Threshold: 0.55, Metrics: {'Accuracy': 0.676923076923077, 'Sensitivity': 0.
Threshold: 0.60, Metrics: {'Accuracy': 0.6461538461538462, 'Sensitivity': 0
Threshold: 0.65, Metrics: {'Accuracy': 0.6230769230769231, 'Sensitivity': 0
Threshold: 0.70, Metrics: {'Accuracy': 0.5692307692307692, 'Sensitivity': 0
Threshold: 0.75, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.80, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.85, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.90, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.95, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 1.00, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
SHAP Summary for Random Forest
```




Sex
Age
−0.1
0.0
0.1
−0.1
0.0
0.1
SHAP interaction val

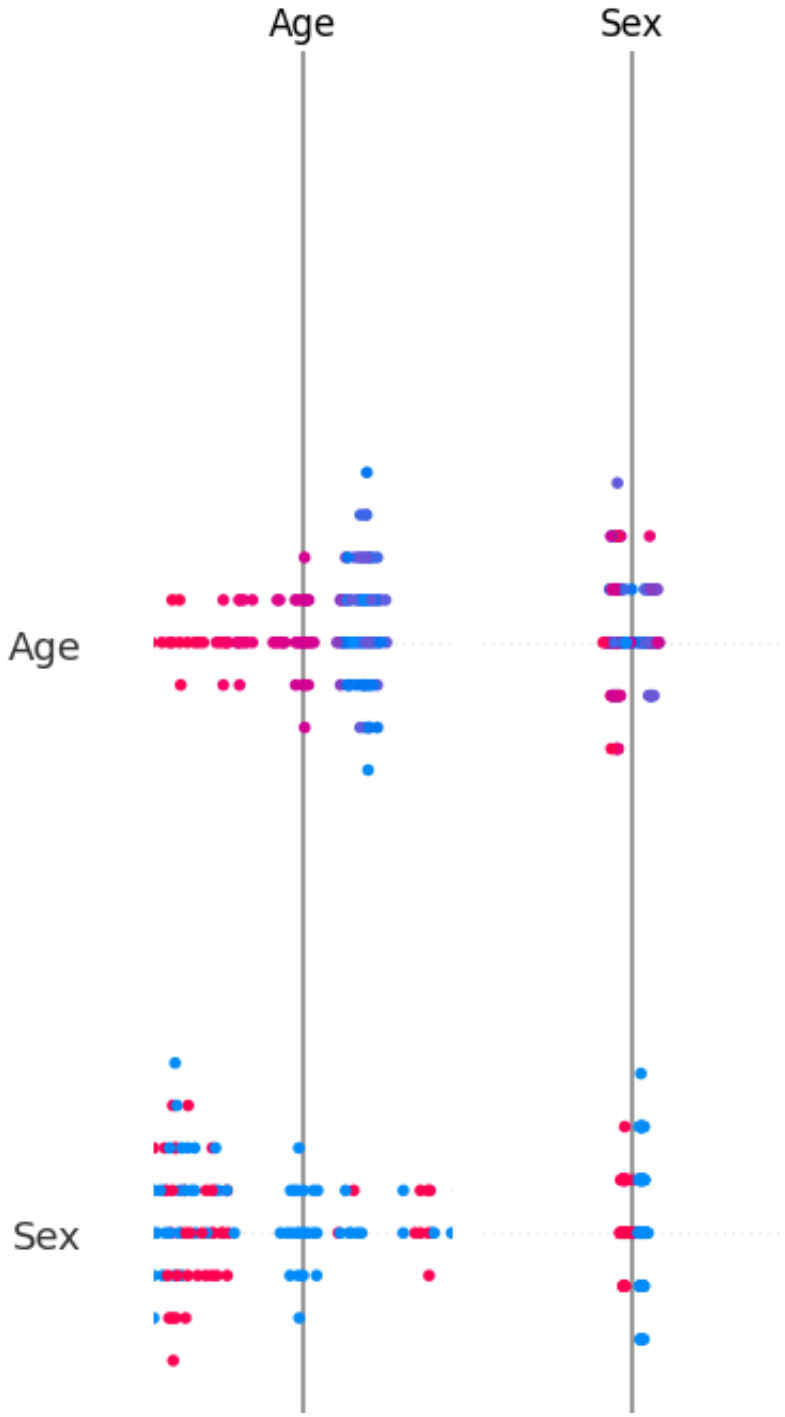

Confusion Matrix Random Forest
Actual Success
52
18
50
45
40
35

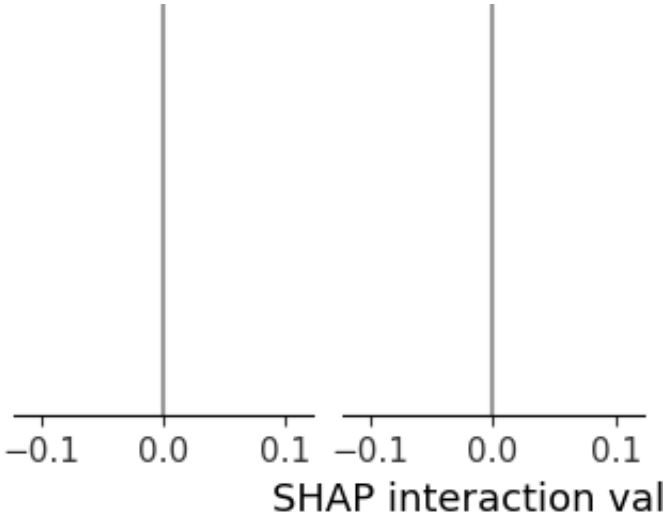


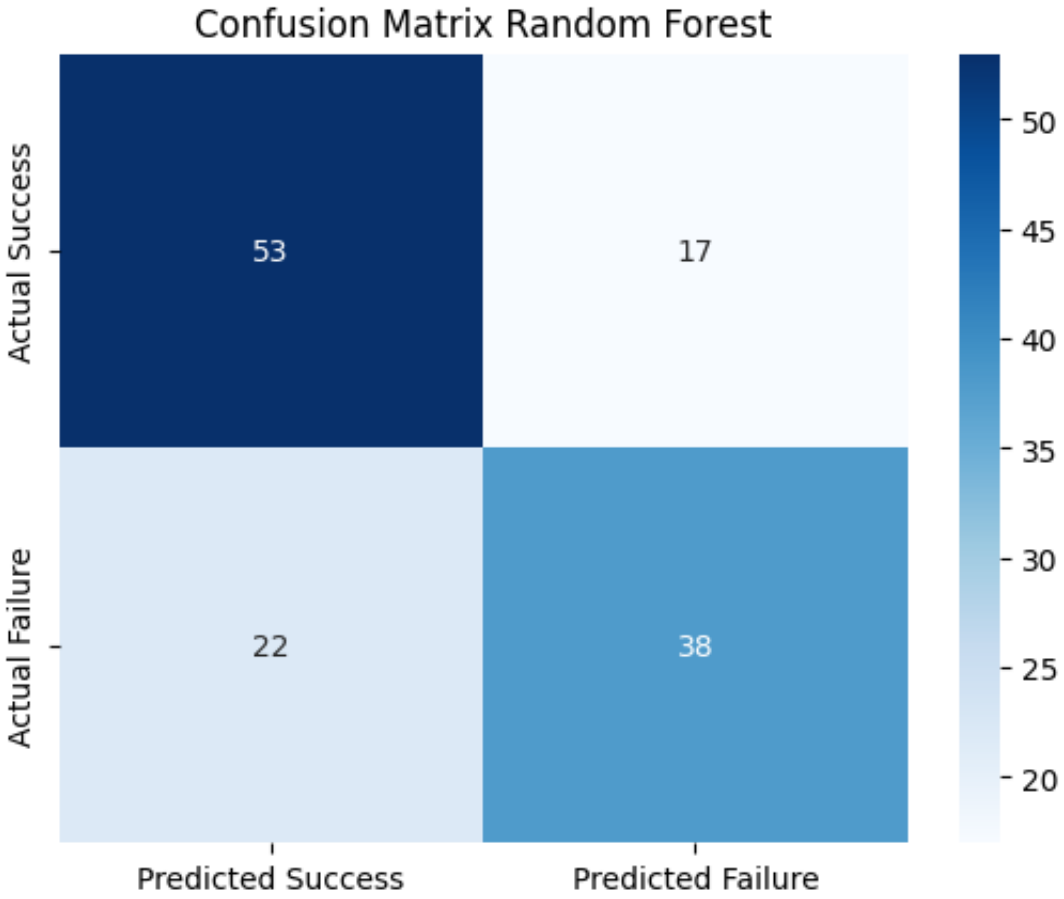


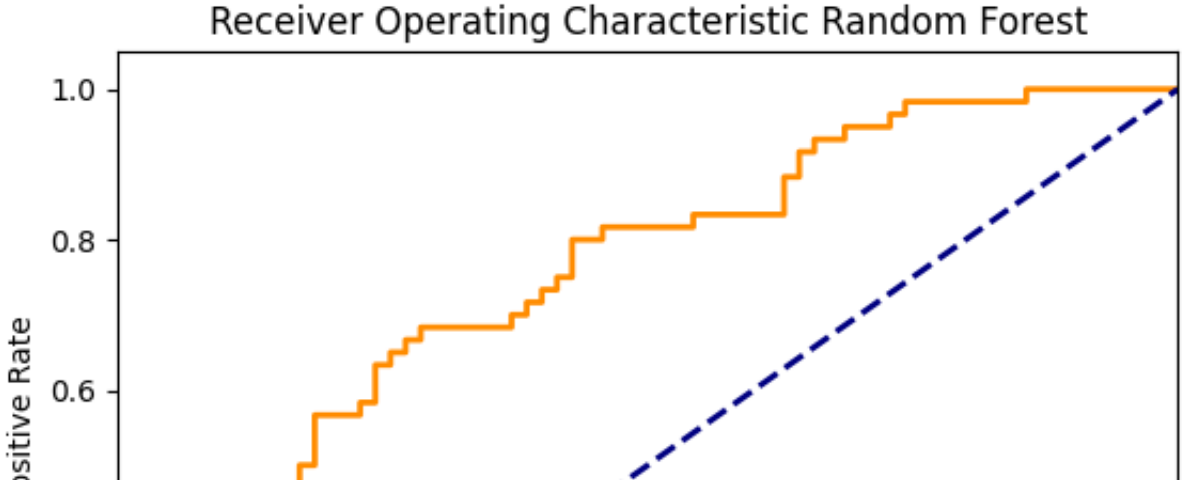

```
Running evaluation with seed 49
Evaluating Random Forest with seed 49...

--- Dados ROC para copiar ---
FPR = [0.0, 0.0, 0.0, 0.014285714285714285, 0.014285714285714285, 0.0285714
TPR = [0.0, 0.016666666666666666, 0.03333333333333333, 0.05, 0.066666666666
AUC = 0.7402380952380953
--- Fim dos Dados ROC ---

Training - Accuracy: 0.7159763313609467, Sensitivity: 0.7450199203187251, S
Metrics for manual threshold 0.55:
Accuracy: 0.7, Sensitivity: 0.6333333333333333, Specificity: 0.757142857142
Threshold: 0.10, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.15, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.20, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.25, Metrics: {'Accuracy': 0.4846153846153846, 'Sensitivity': 1
Threshold: 0.30, Metrics: {'Accuracy': 0.5153846153846153, 'Sensitivity': 1
```

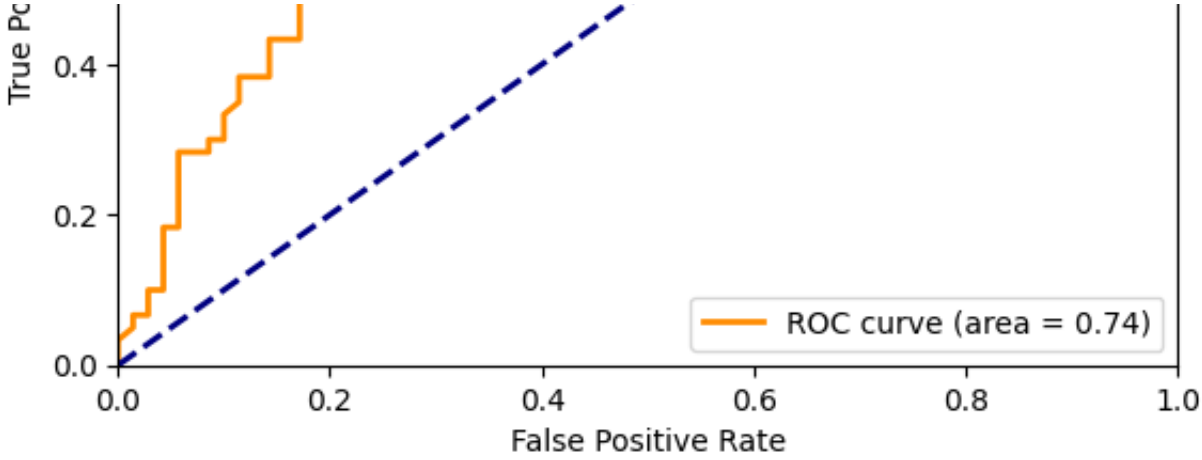
```
Threshold: 0.35, Metrics: {'Accuracy': 0.5692307692307692, 'Sensitivity': 0
Threshold: 0.40, Metrics: {'Accuracy': 0.6, 'Sensitivity': 0.93333333333333
Threshold: 0.45, Metrics: {'Accuracy': 0.6461538461538462, 'Sensitivity': 0
Threshold: 0.50, Metrics: {'Accuracy': 0.6538461538461539, 'Sensitivity': 0
Threshold: 0.55, Metrics: {'Accuracy': 0.7, 'Sensitivity': 0.63333333333333
Threshold: 0.60, Metrics: {'Accuracy': 0.6615384615384615, 'Sensitivity': 0
Threshold: 0.65, Metrics: {'Accuracy': 0.6307692307692307, 'Sensitivity': 0
Threshold: 0.70, Metrics: {'Accuracy': 0.5692307692307692, 'Sensitivity': 0
Threshold: 0.75, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.80, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.85, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.90, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.95, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 1.00, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
SHAP Summary for Random Forest
```




−0.1
0.0
0.1
−0.1
0.0
0.1
SHAP interaction val


Confusion Matrix Random Forest
Actual Success
Actual Failure
53
17
22
38
Predicted Success
Predicted Failure
50
45
40
35
30
25
20


Receiver Operating Characteristic Random Forest
1.0
0.8
0.6




```
Aggregated Test Set Metrics Across Seeds:
   accuracy  sensitivity  specificity        f1   roc_auc
0  0.692308     0.616667     0.757143  0.649123  0.747143
1  0.707692     0.633333     0.771429  0.666667  0.745714
2  0.684615     0.616667     0.742857  0.643478  0.750000
3  0.707692     0.633333     0.771429  0.666667  0.749286
4  0.707692     0.633333     0.771429  0.666667  0.747381
5  0.684615     0.600000     0.757143  0.637168  0.744048
6  0.692308     0.633333     0.742857  0.655172  0.742857
7  0.684615     0.600000     0.757143  0.637168  0.745476
8  0.676923     0.600000     0.742857  0.631579  0.745000
9  0.700000     0.633333     0.757143  0.660870  0.740238

Summary of Test Set Metrics (Mean, Standard Error, 95% Confidence Interval)
Accuracy: Mean = 0.694, SE = 0.004, 95% CI = [0.686, 0.702]
Sensitivity: Mean = 0.620, SE = 0.005, 95% CI = [0.609, 0.631]
Specificity: Mean = 0.757, SE = 0.004, 95% CI = [0.749, 0.765]
F1: Mean = 0.651, SE = 0.004, 95% CI = [0.642, 0.661]
Roc_auc: Mean = 0.746, SE = 0.001, 95% CI = [0.744, 0.748]
```

```
def evaluate_model(model, name, grid, X_train, y_train, X_test, y_test, cv, scc
    print(f"Evaluating {name}...")

    inner_cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=seed)
    outer_cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=seed)

    clf = GridSearchCV(model, grid, cv=inner_cv, scoring='roc_auc')
    nested_scores = cross_validate(clf, X=X_train, y=y_train, cv=outer_cv, scor

    clf.fit(X_train, y_train)
    best_model = clf.best_estimator_
    best_params = clf.best_params_

    print(f"Best parameters for {name}: {best_params}")  # Print the best param

    calibrated_clf = CalibratedClassifierCV(estimator=best_model, method='sigmc
```

```
calibrated_clf.fit(X_train, y_train)

y_probs = calibrated_clf.predict_proba(X_test)[:, 1]

# Calcular FPR, TPR e AUC
fpr, tpr, thresholds = roc_curve(y_test, y_probs)
roc_auc = auc(fpr, tpr)

# Imprimir os valores de FPR, TPR e AUC de forma fácil de copiar
print("\n--- Dados ROC para copiar ---")
print("FPR =", fpr.tolist())
print("TPR =", tpr.tolist())
print("AUC =", roc_auc)
print("--- Fim dos Dados ROC ---\n")

# Calculate metrics for the training set
y_train_pred = best_model.predict(X_train)
y_train_probs = best_model.predict_proba(X_train)[:, 1]
train_acc = accuracy_score(y_train, y_train_pred)
train_sens = sensitivity(y_train, y_train_pred)
train_spec = specificity(y_train, y_train_pred)
train_f1 = f1_score(y_train, y_train_pred)
train_roc_auc = roc_auc_score(y_train, y_train_probs)

# Print training set metrics
print(f"Training - Accuracy: {train_acc}, Sensitivity: {train_sens}, Specif

# Metrics for the manually set threshold
y_pred_manual = (y_probs >= manual_threshold).astype(int)
manual_acc = accuracy_score(y_test, y_pred_manual)
manual_sens = sensitivity(y_test, y_pred_manual)
manual_spec = specificity(y_test, y_pred_manual)
manual_f1 = f1_score(y_test, y_pred_manual)
manual_roc_auc = roc_auc_score(y_test, y_probs)

print(f"Metrics for manual threshold {manual_threshold}:")
print(f"Accuracy: {manual_acc}, Sensitivity: {manual_sens}, Specificity: {m

threshold_metrics = {}
for threshold in threshold_list:
    y_pred_threshold = (y_probs >= threshold).astype(int)
    threshold_acc = accuracy_score(y_test, y_pred_threshold)
    threshold_sens = sensitivity(y_test, y_pred_threshold)
    threshold_spec = specificity(y_test, y_pred_threshold)
    threshold_f1 = f1_score(y_test, y_pred_threshold)
    threshold_metrics[threshold] = {'Accuracy': threshold_acc, 'Sensitivity

for threshold, metrics in threshold_metrics.items():
```

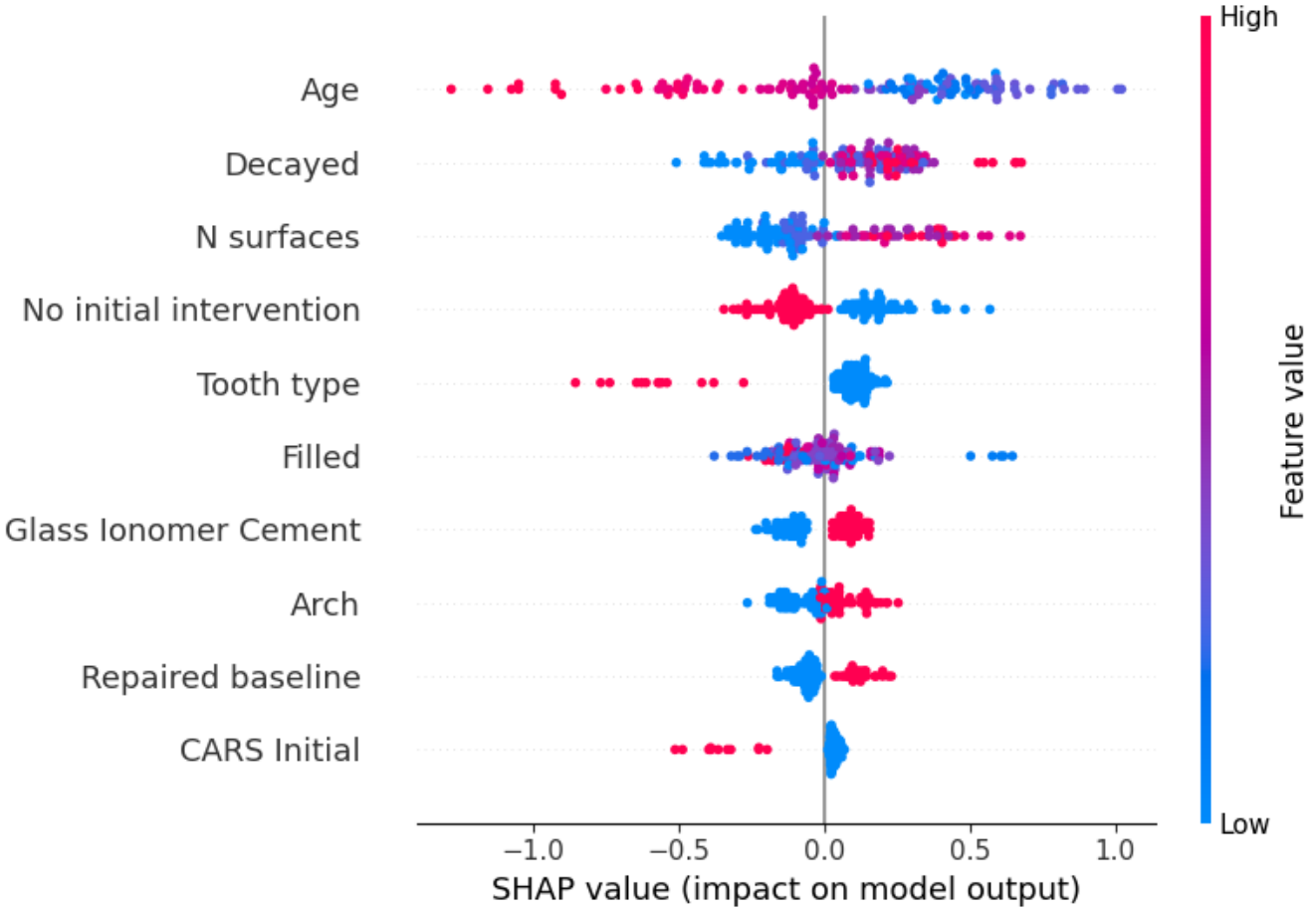
```
            print(f"Threshold: {threshold:.2f}, Metrics: {metrics}")

        calculate_and_plot_shap(best_model, X_train, X_test, name)

        # Prepare dictionary of test metrics for aggregation
        test_metrics = {
            "accuracy": manual_acc,
            "sensitivity": manual_sens,
            "specificity": manual_spec,
            "f1": manual_f1,
            "roc_auc": manual_roc_auc
        }

        return best_model, manual_threshold, best_params, nested_scores, calibrated

    def calculate_and_plot_shap(model, X_train, X_test, model_name):
        if isinstance(model, (XGBClassifier)):
            explainer = shap.TreeExplainer(model)
        else:
            explainer = shap.KernelExplainer(model.predict_proba, X_train.sample(10
        shap_values = explainer.shap_values(X_test)
        print(f"SHAP Summary for {model_name}")
        shap.summary_plot(shap_values, X_test, max_display=10)

    # Plotting functions for confusion matrix and ROC curve visualization.
    def plot_confusion_matrix(y_true, y_pred):
        matrix = confusion_matrix(y_true, y_pred)
        sns.heatmap(matrix, annot=True, fmt='d', cmap='Blues', xticklabels=['Predic
        plt.title('Confusion Matrix XGBoosting')
        plt.show()

    def plot_roc_curve(y_true, y_probs):
        fpr, tpr, thresholds = roc_curve(y_true, y_probs)
        roc_auc = auc(fpr, tpr)

        plt.figure()
        plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (area = {roc
        plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
        plt.xlim([0.0, 1.0])
        plt.ylim([0.0, 1.05])
        plt.xlabel('False Positive Rate')
        plt.ylabel('True Positive Rate')
        plt.title('Receiver Operating Characteristic XGBoosting')
        plt.legend(loc="lower right")
        plt.show()

    def evaluate_xgboost(X_train, y_train, X_test, y_test, cv, scoring, manual_thre
        print("Inside evaluate_xgboost function")
```

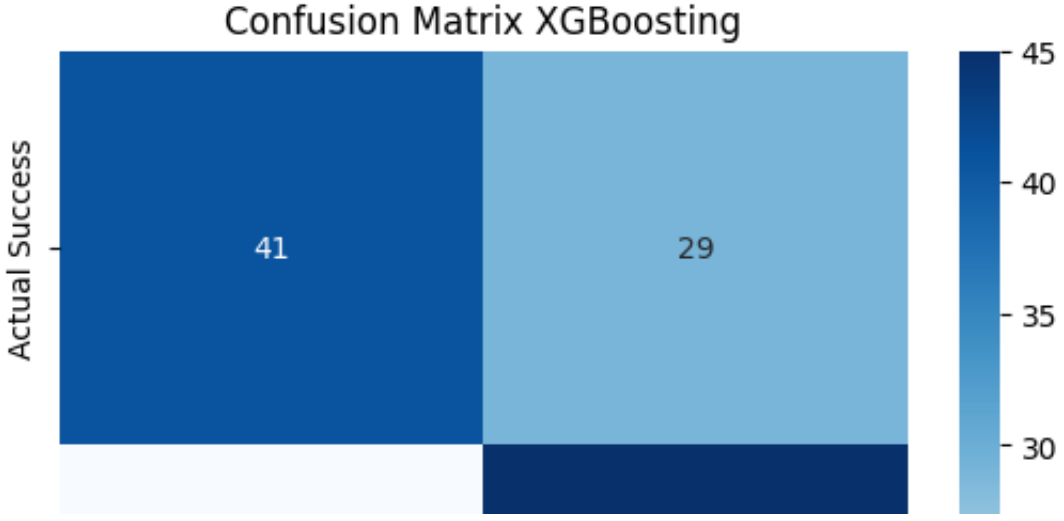
```
    model = XGBClassifier(eval_metric='logloss', random_state=seed)
    grid = {
        'max_depth': [6],
        'gamma': [0.1],
        'learning_rate': [0.01],
        'subsample': [0.8],
        'colsample_bytree': [1],
        'reg_alpha': [0],
        'reg_lambda': [1],
        'n_estimators': [300]
    }

    return evaluate_model(model, "XGBoost", grid, X_train, y_train, X_test, y_t

def main(X_train, y_train, X_test, y_test):
    cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=10, random_state=42)
    scoring = {
        'accuracy': make_scorer(accuracy_score),
        'sensitivity': make_scorer(sensitivity),
        'specificity': make_scorer(specificity),
        'f1': make_scorer(f1_score),
        'roc_auc': make_scorer(roc_auc_score)
    }
    manual_threshold = 0.5
    threshold_list = np.arange(0.1, 1.05, 0.05)

    aggregated_metrics = []

    for seed in range(40, 50):
        print(f"Running evaluation with seed {seed}")
        best_model, manual_threshold, best_params, nested_scores, calibrated_cl
            X_train, y_train, X_test, y_test, cv, scoring, manual_threshold, th
        )

        # Use calibrated_clf for prediction probabilities
        y_probs = calibrated_clf.predict_proba(X_test)[:, 1]
        y_pred_manual = (y_probs >= manual_threshold).astype(int)

        # Plotting functions assuming they are imported or defined elsewhere
        plot_confusion_matrix(y_test, y_pred_manual)
        plot_roc_curve(y_test, y_probs)

        aggregated_metrics.append(test_metrics)

    # Aggregate results across seeds
    results_df = pd.DataFrame(aggregated_metrics)
    n = len(results_df)
    print("\nAggregated Test Set Metrics Across Seeds:")
```

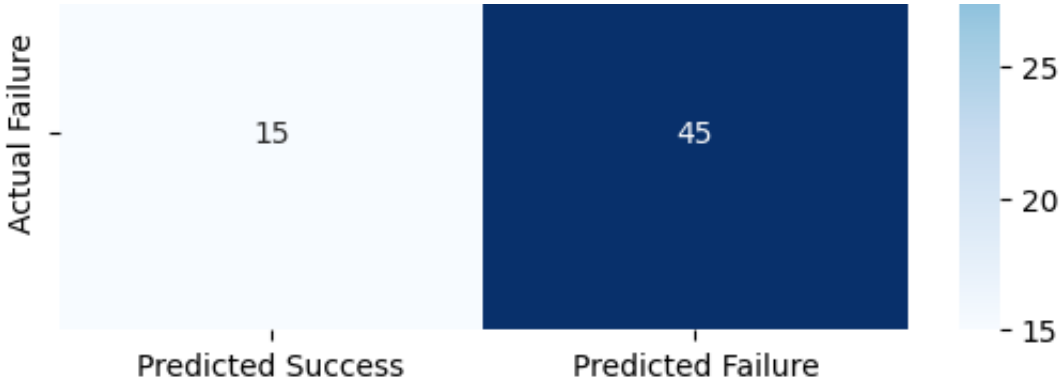
```
    print(results_df)

    # Compute mean, standard error, and 95% confidence interval for each metric
    def summarize_metric(metric_values):
        mean_val = metric_values.mean()
        std_val = metric_values.std(ddof=1)
        se = std_val / np.sqrt(n)
        t_crit = stats.t.ppf(0.975, df=n - 1)
        ci_lower = mean_val - t_crit * se
        ci_upper = mean_val + t_crit * se
        return mean_val, se, (ci_lower, ci_upper)

    metrics_summary = {}
    for metric in results_df.columns:
        mean_val, se, ci = summarize_metric(results_df[metric])
        metrics_summary[metric] = {
            "Mean": mean_val,
            "Standard Error": se,
            "95% CI": ci
        }

    print("\nSummary of Test Set Metrics (Mean, Standard Error, 95% Confidence
    for metric, summary in metrics_summary.items():
        print(f"{metric.capitalize()}: Mean = {summary['Mean']:.3f}, SE = {summ
              f"95% CI = [{summary['95% CI'][0]:.3f}, {summary['95% CI'][1]:.3f

if __name__ == '__main__':
    main(X_train, y_train, X_test, y_test)
```

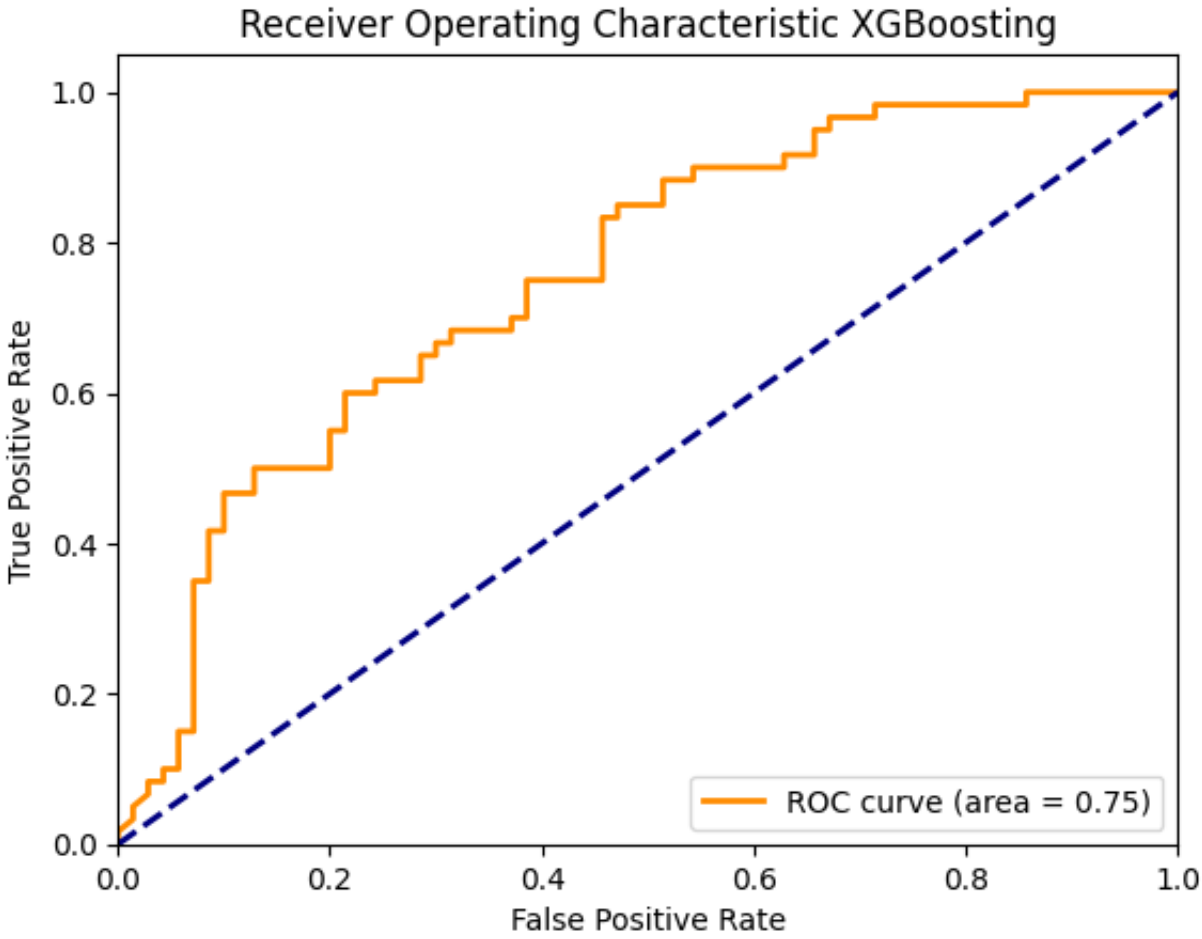
```
Running evaluation with seed 40
Inside evaluate_xgboost function
Evaluating XGBoost...
Best parameters for XGBoost: {'colsample_bytree': 1, 'gamma': 0.1, 'learnin

--- Dados ROC para copiar ---
FPR = [0.0, 0.0, 0.014285714285714285, 0.014285714285714285, 0.028571428571
TPR = [0.0, 0.016666666666666666, 0.03333333333333333, 0.05, 0.066666666666
AUC = 0.7540476190476191
--- Fim dos Dados ROC ---

Training - Accuracy: 0.8678500986193294, Sensitivity: 0.8804780876494024, S
Metrics for manual threshold 0.5:
Accuracy: 0.6615384615384615, Sensitivity: 0.75, Specificity: 0.58571428571
Threshold: 0.10, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.15, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.20, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.25, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.30, Metrics: {'Accuracy': 0.46923076923076923, 'Sensitivity':
Threshold: 0.35, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.40, Metrics: {'Accuracy': 0.5923076923076923, 'Sensitivity': 0
```

```
Threshold: 0.40, Metrics: {'Accuracy': 0.5923076923076923, 'Sensitivity': 0
Threshold: 0.45, Metrics: {'Accuracy': 0.6692307692307692, 'Sensitivity': 0
Threshold: 0.50, Metrics: {'Accuracy': 0.6615384615384615, 'Sensitivity': 0
Threshold: 0.55, Metrics: {'Accuracy': 0.6692307692307692, 'Sensitivity': 0
Threshold: 0.60, Metrics: {'Accuracy': 0.676923076923077, 'Sensitivity': 0.
Threshold: 0.65, Metrics: {'Accuracy': 0.5846153846153846, 'Sensitivity': 0
Threshold: 0.70, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.75, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.80, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.85, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.90, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.95, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 1.00, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
SHAP Summary for XGBoost
```



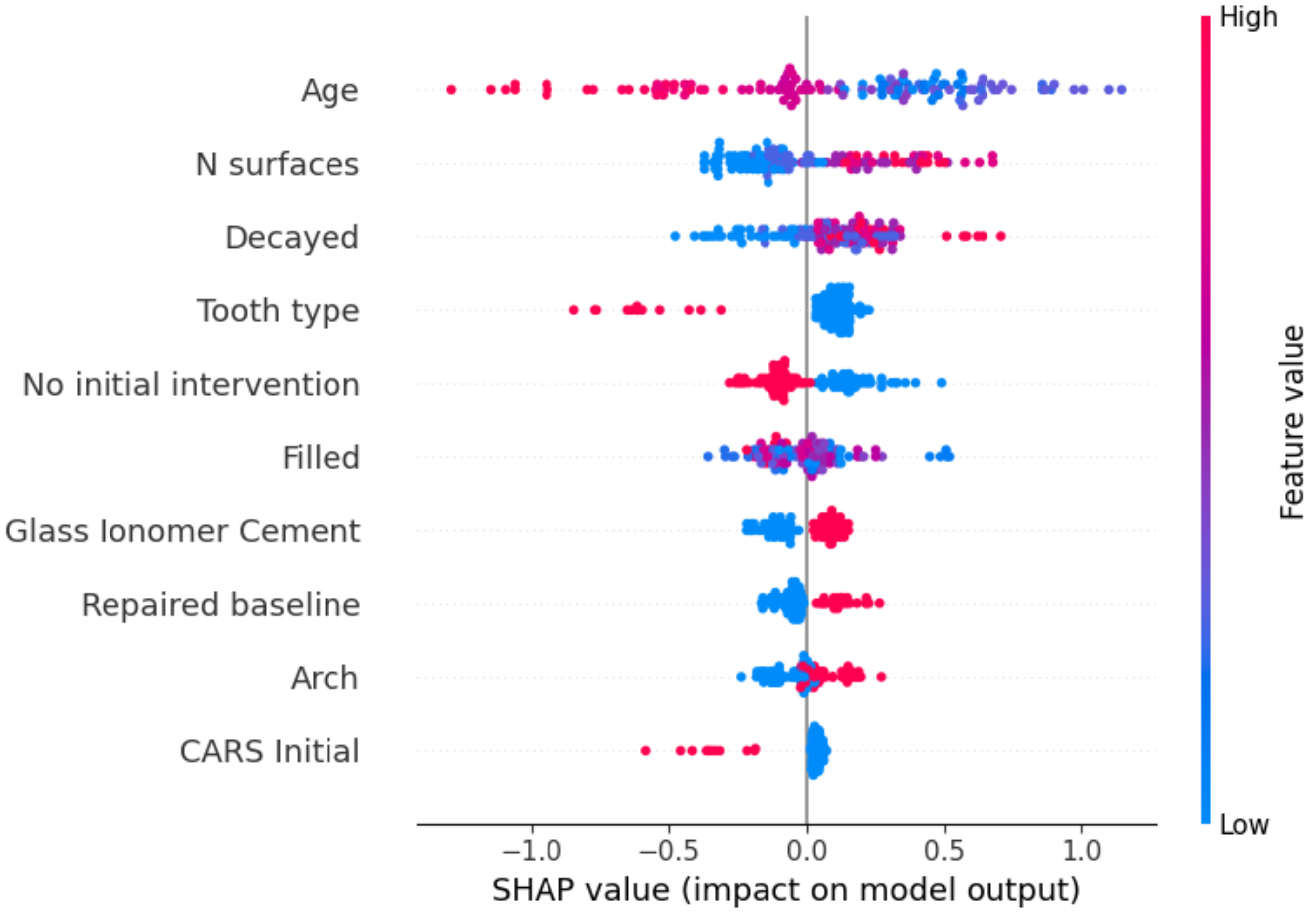


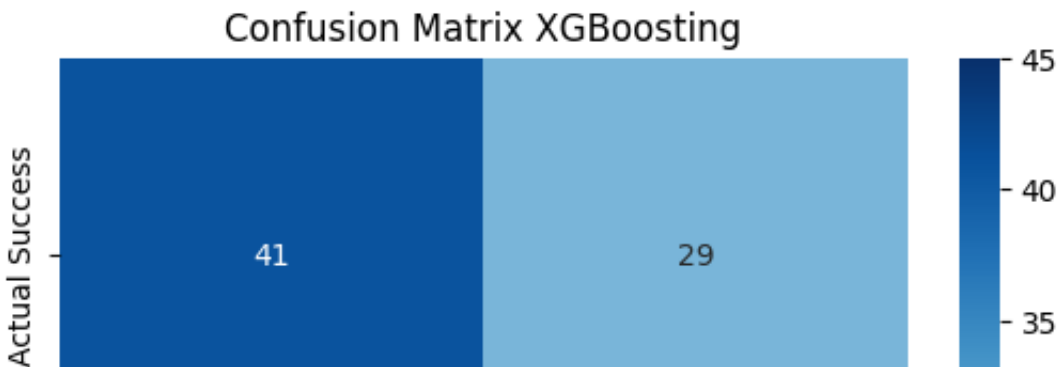


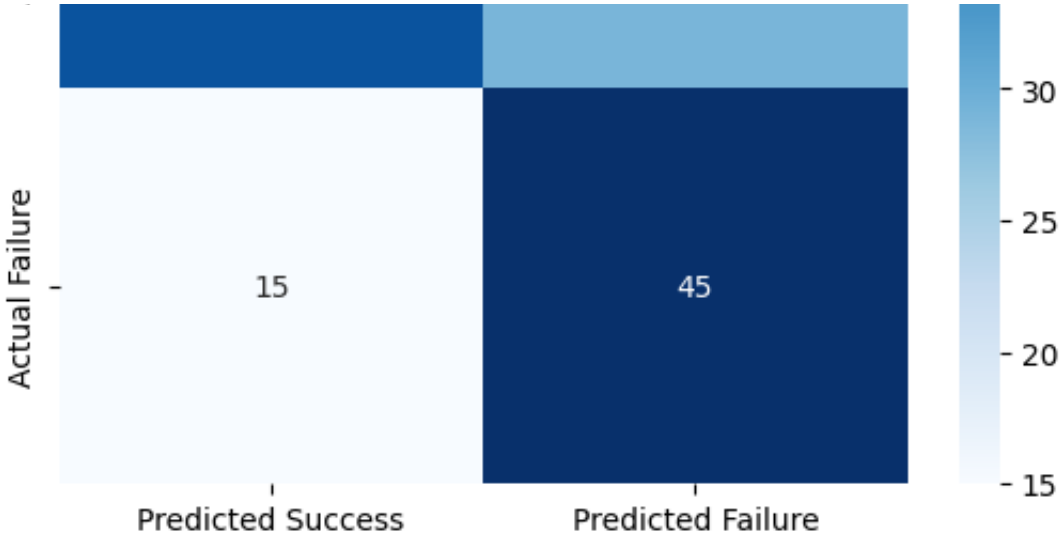


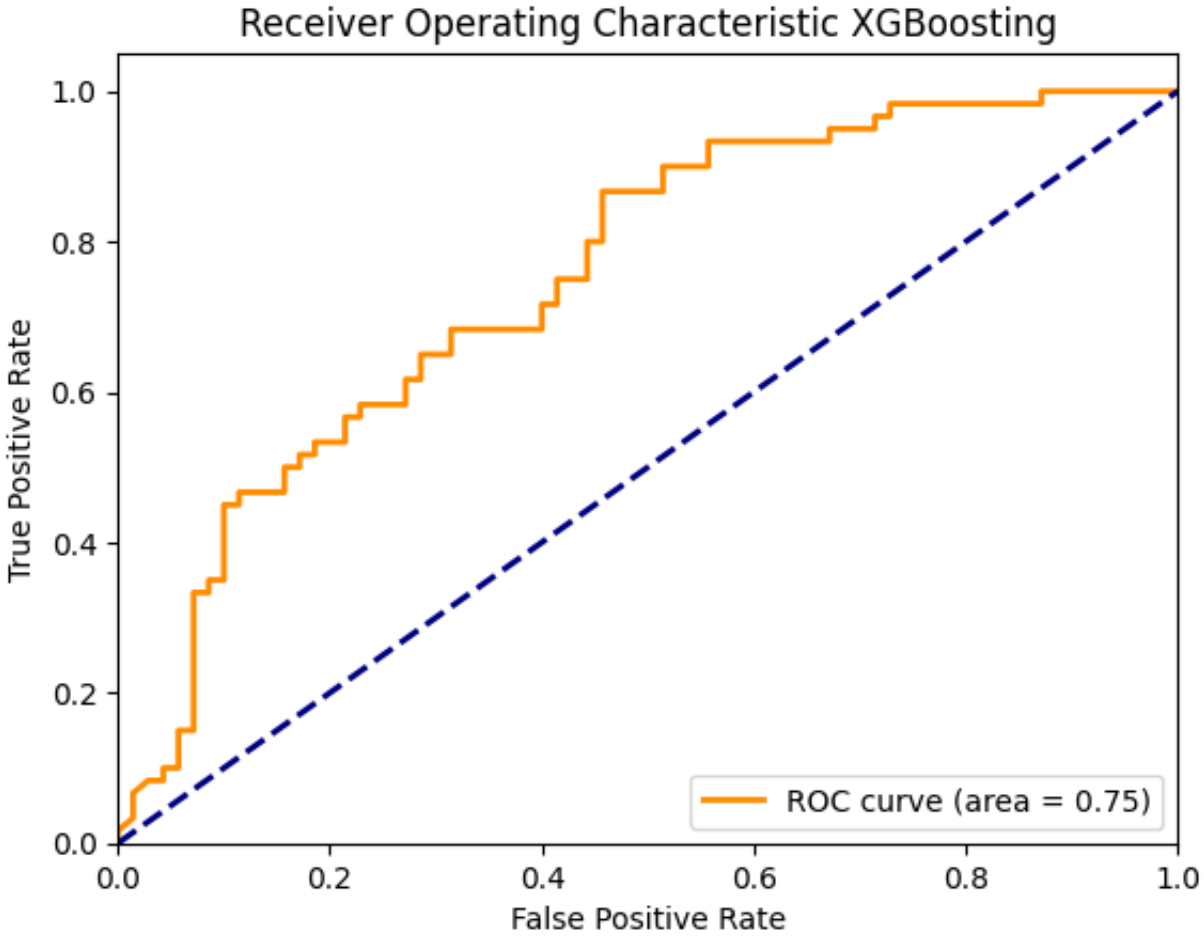

```
Running evaluation with seed 41
Inside evaluate_xgboost function
Evaluating XGBoost...
Best parameters for XGBoost: {'colsample_bytree': 1, 'gamma': 0.1, 'learnin

--- Dados ROC para copiar ---
FPR = [0.0, 0.0, 0.014285714285714285, 0.014285714285714285, 0.028571428571
TPR = [0.0, 0.016666666666666666, 0.03333333333333333, 0.06666666666666667,
AUC = 0.7526190476190475
--- Fim dos Dados ROC ---

Training - Accuracy: 0.8757396449704142, Sensitivity: 0.8804780876494024, S
Metrics for manual threshold 0.5:
Accuracy: 0.6615384615384615, Sensitivity: 0.75, Specificity: 0.58571428571
Threshold: 0.10, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.15, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
```

```
Threshold: 0.20, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.25, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.30, Metrics: {'Accuracy': 0.47692307692307695, 'Sensitivity':
Threshold: 0.35, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.40, Metrics: {'Accuracy': 0.5923076923076923, 'Sensitivity': 0
Threshold: 0.45, Metrics: {'Accuracy': 0.6615384615384615, 'Sensitivity': 0
Threshold: 0.50, Metrics: {'Accuracy': 0.6615384615384615, 'Sensitivity': 0
Threshold: 0.55, Metrics: {'Accuracy': 0.676923076923077, 'Sensitivity': 0.
Threshold: 0.60, Metrics: {'Accuracy': 0.676923076923077, 'Sensitivity': 0.
Threshold: 0.65, Metrics: {'Accuracy': 0.5769230769230769, 'Sensitivity': 0
Threshold: 0.70, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.75, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.80, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.85, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.90, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.95, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 1.00, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
SHAP Summary for XGBoost
```



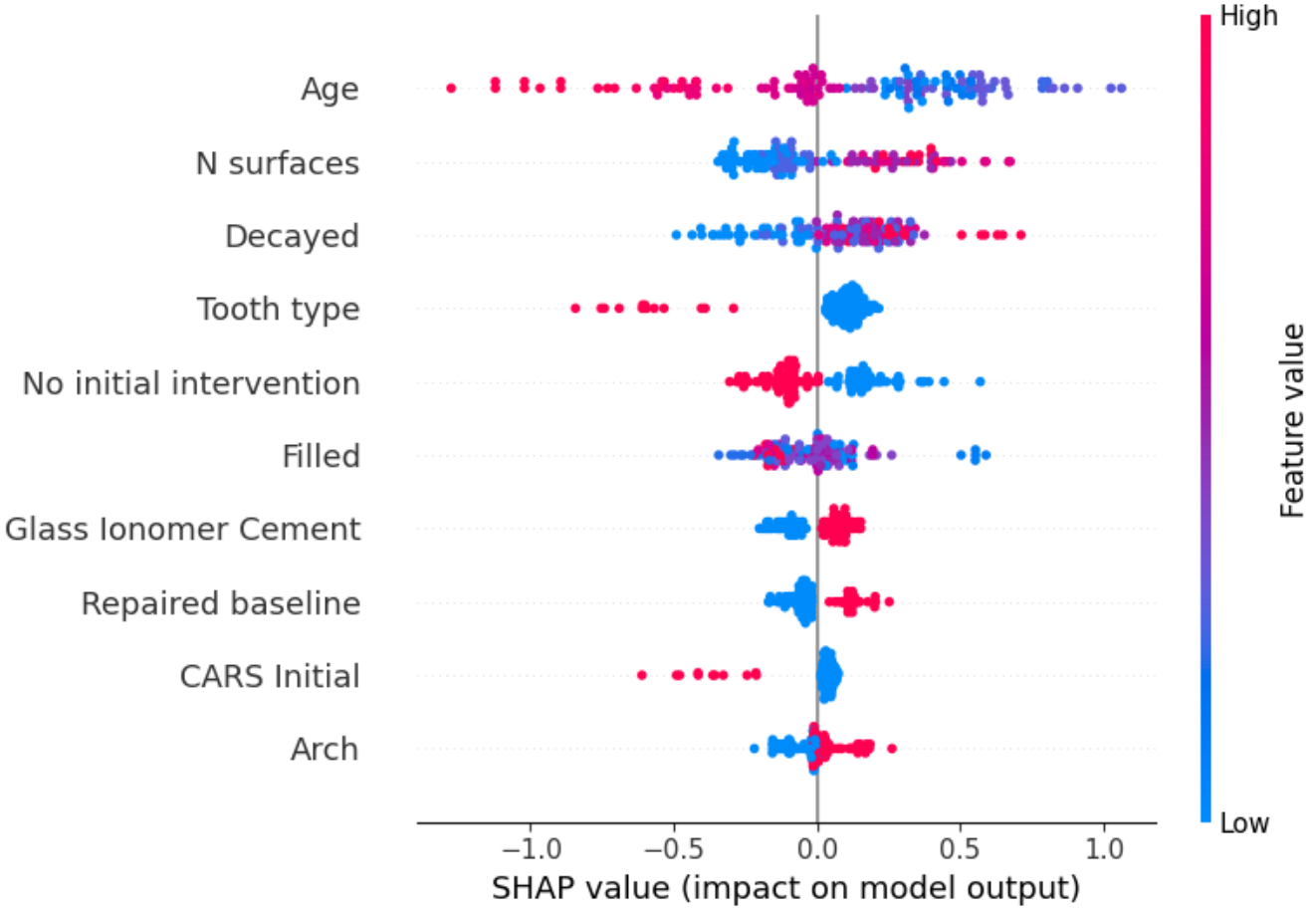


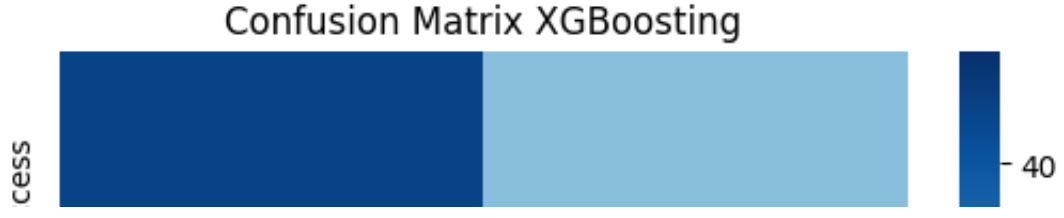


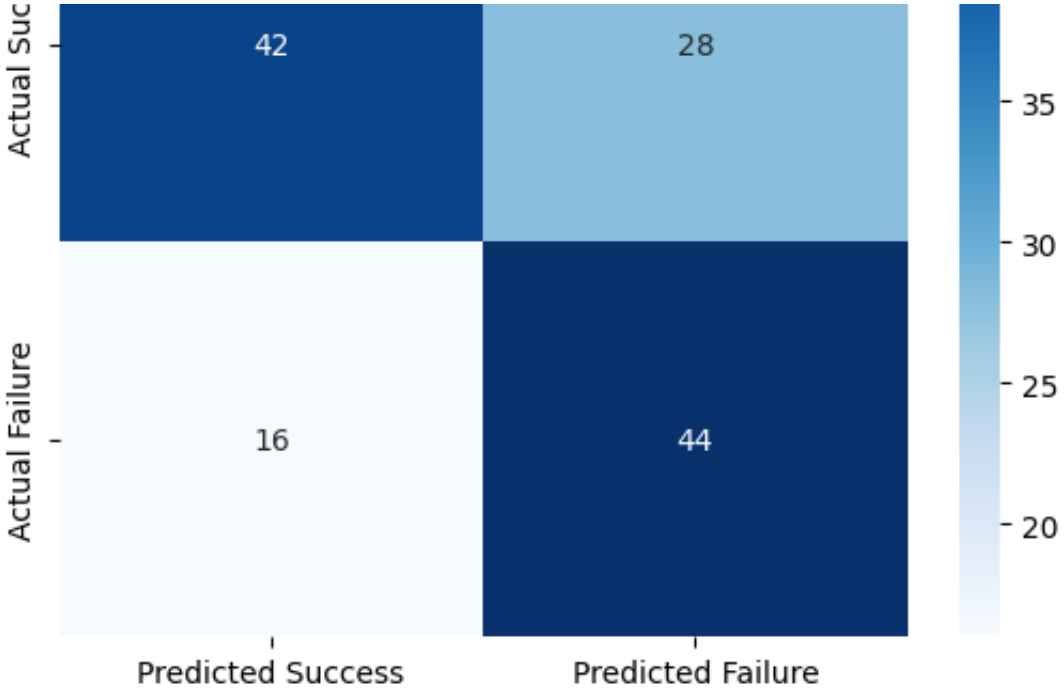


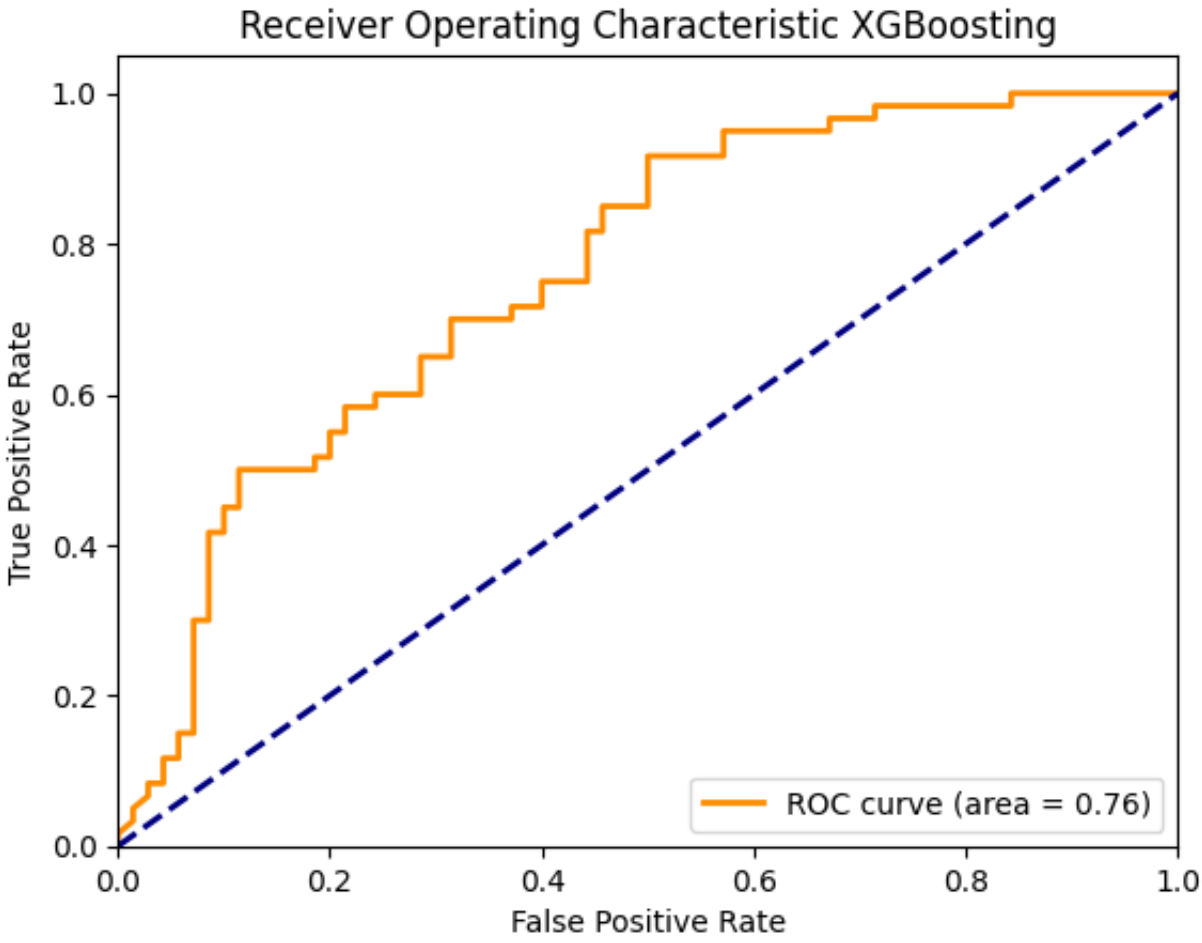

```
Running evaluation with seed 42
Inside evaluate_xgboost function
Evaluating XGBoost...
Best parameters for XGBoost: {'colsample_bytree': 1, 'gamma': 0.1, 'learnin

--- Dados ROC para copiar ---
FPR = [0.0, 0.0, 0.014285714285714285, 0.014285714285714285, 0.028571428571
TPR = [0.0, 0.016666666666666666, 0.03333333333333333, 0.05, 0.066666666666
AUC = 0.7609523809523809
--- Fim dos Dados ROC ---

Training - Accuracy: 0.8658777120315582, Sensitivity: 0.8804780876494024, S
```

```
Metrics for manual threshold 0.5:
Accuracy: 0.6615384615384615, Sensitivity: 0.7333333333333333, Specificity:
Threshold: 0.10, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.15, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.20, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.25, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.30, Metrics: {'Accuracy': 0.4692307692307692, 'Sensitivity':
Threshold: 0.35, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.40, Metrics: {'Accuracy': 0.5923076923076923, 'Sensitivity': 0
Threshold: 0.45, Metrics: {'Accuracy': 0.676923076923077, 'Sensitivity': 0.
Threshold: 0.50, Metrics: {'Accuracy': 0.6615384615384615, 'Sensitivity': 0
Threshold: 0.55, Metrics: {'Accuracy': 0.6692307692307692, 'Sensitivity': 0
Threshold: 0.60, Metrics: {'Accuracy': 0.676923076923077, 'Sensitivity': 0.
Threshold: 0.65, Metrics: {'Accuracy': 0.5769230769230769, 'Sensitivity': 0
Threshold: 0.70, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.75, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.80, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.85, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.90, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.95, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 1.00, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
SHAP Summary for XGBoost
```



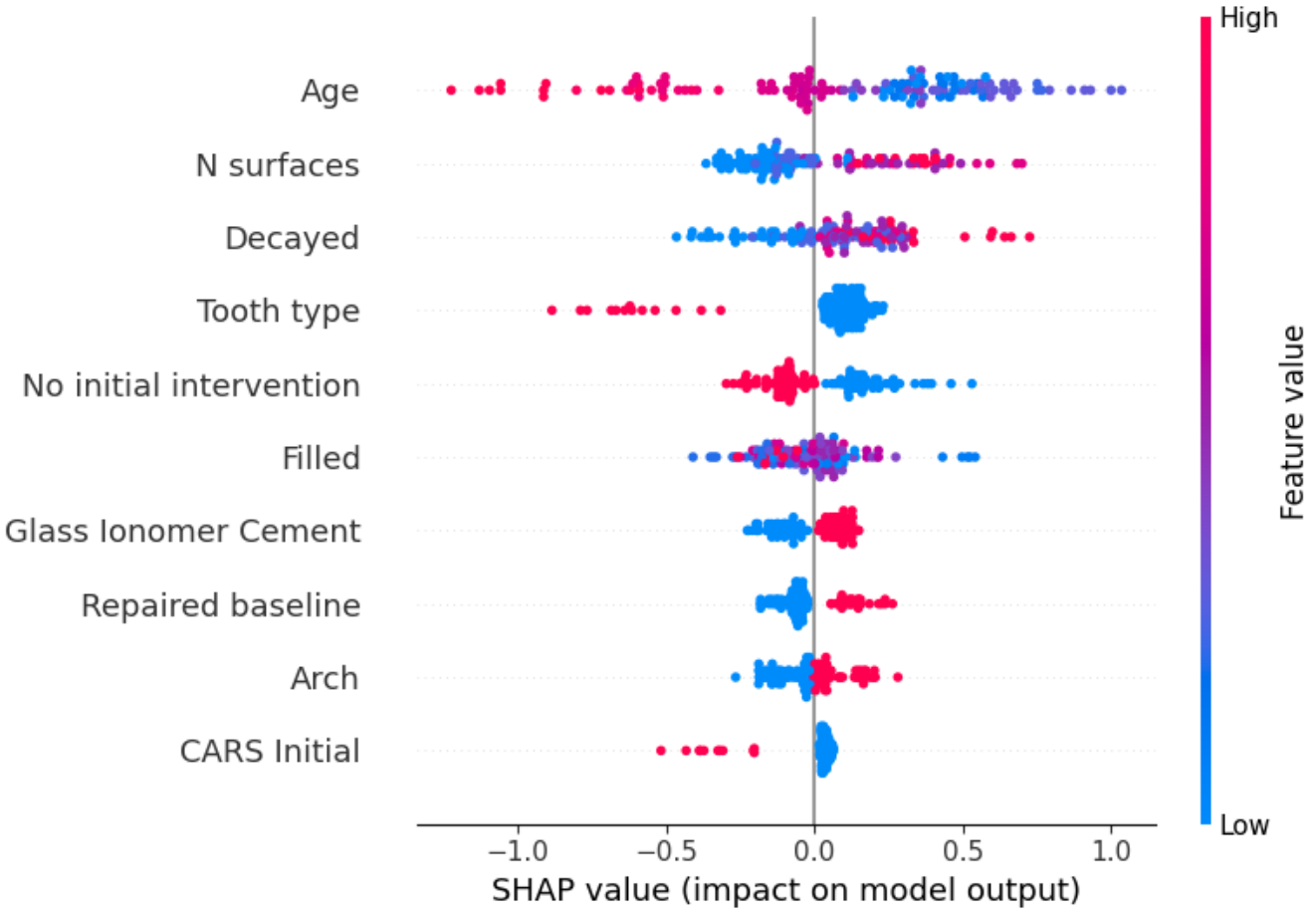




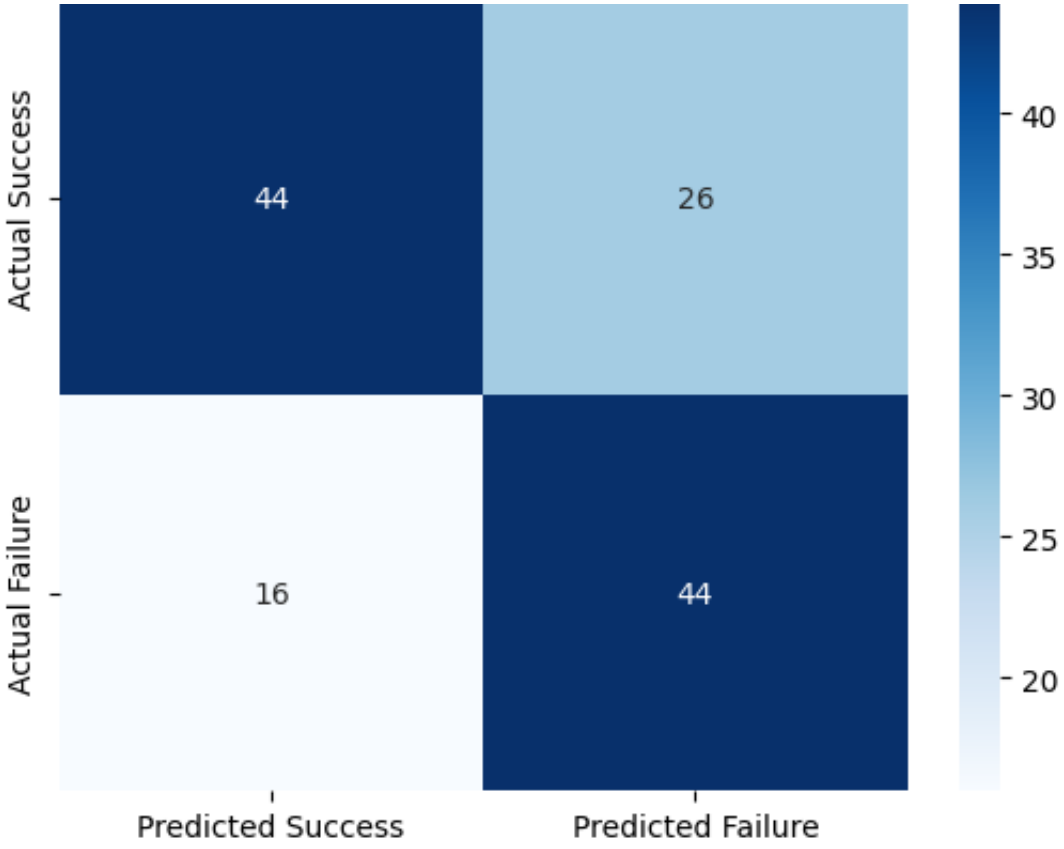


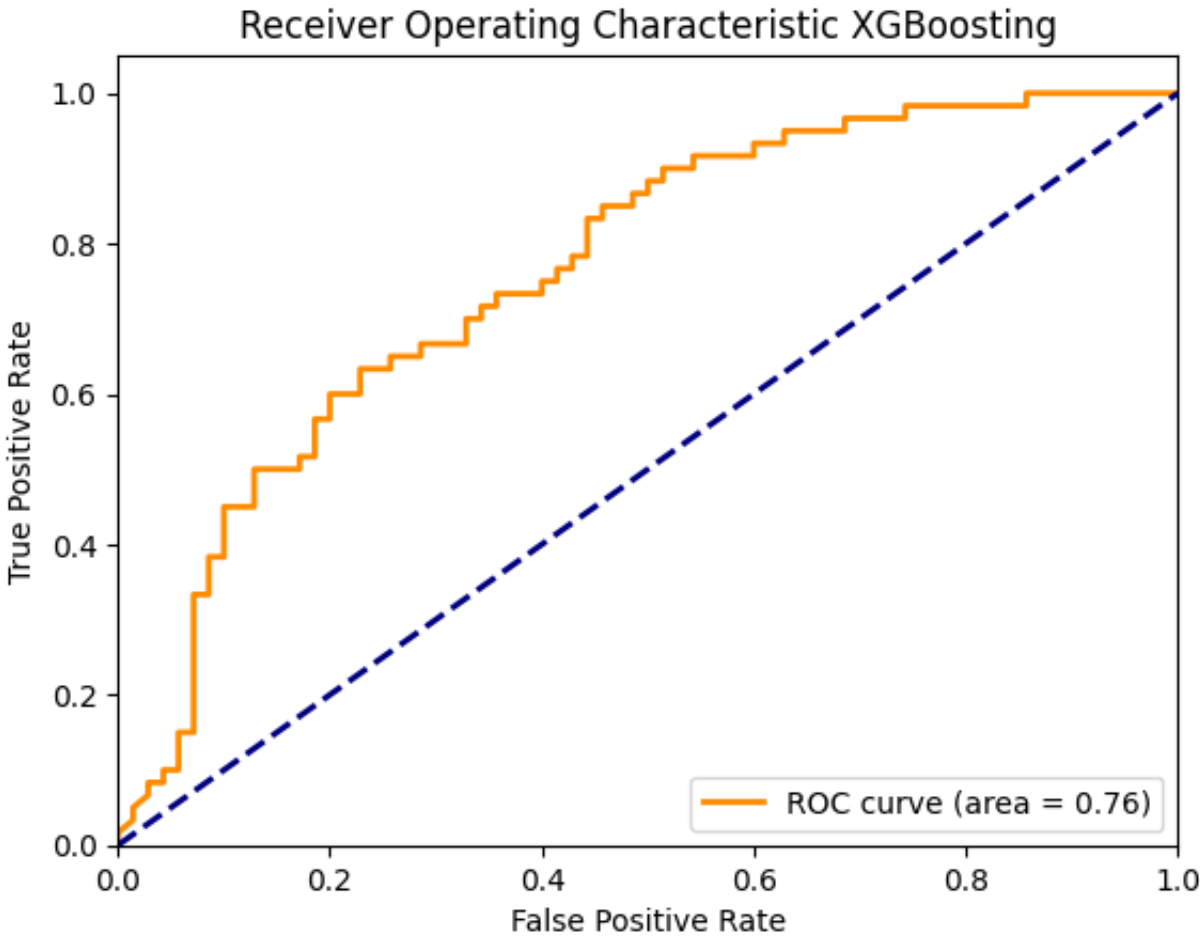

```
Running evaluation with seed 43
Inside evaluate_xgboost function
Evaluating XGBoost...
Best parameters for XGBoost: {'colsample_bytree': 1, 'gamma': 0.1, 'learnin

--- Dados ROC para copiar ---
FPR = [0.0, 0.0, 0.014285714285714285, 0.014285714285714285, 0.028571428571
TPR = [0.0, 0.016666666666666666, 0.03333333333333333, 0.05, 0.066666666666
```

```
AUC = 0.7635714285714285
--- Fim dos Dados ROC ---

Training - Accuracy: 0.8619329388560157, Sensitivity: 0.8725099601593626, S
Metrics for manual threshold 0.5:
Accuracy: 0.676923076923077, Sensitivity: 0.7333333333333333, Specificity:
Threshold: 0.10, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.15, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.20, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.25, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.30, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.35, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.40, Metrics: {'Accuracy': 0.5923076923076923, 'Sensitivity': 0
Threshold: 0.45, Metrics: {'Accuracy': 0.676923076923077, 'Sensitivity': 0.
Threshold: 0.50, Metrics: {'Accuracy': 0.676923076923077, 'Sensitivity': 0.
Threshold: 0.55, Metrics: {'Accuracy': 0.7, 'Sensitivity': 0.61666666666666
Threshold: 0.60, Metrics: {'Accuracy': 0.6923076923076923, 'Sensitivity': 0
Threshold: 0.65, Metrics: {'Accuracy': 0.5769230769230769, 'Sensitivity': 0
Threshold: 0.70, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.75, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.80, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.85, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.90, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.95, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 1.00, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
SHAP Summary for XGBoost
```



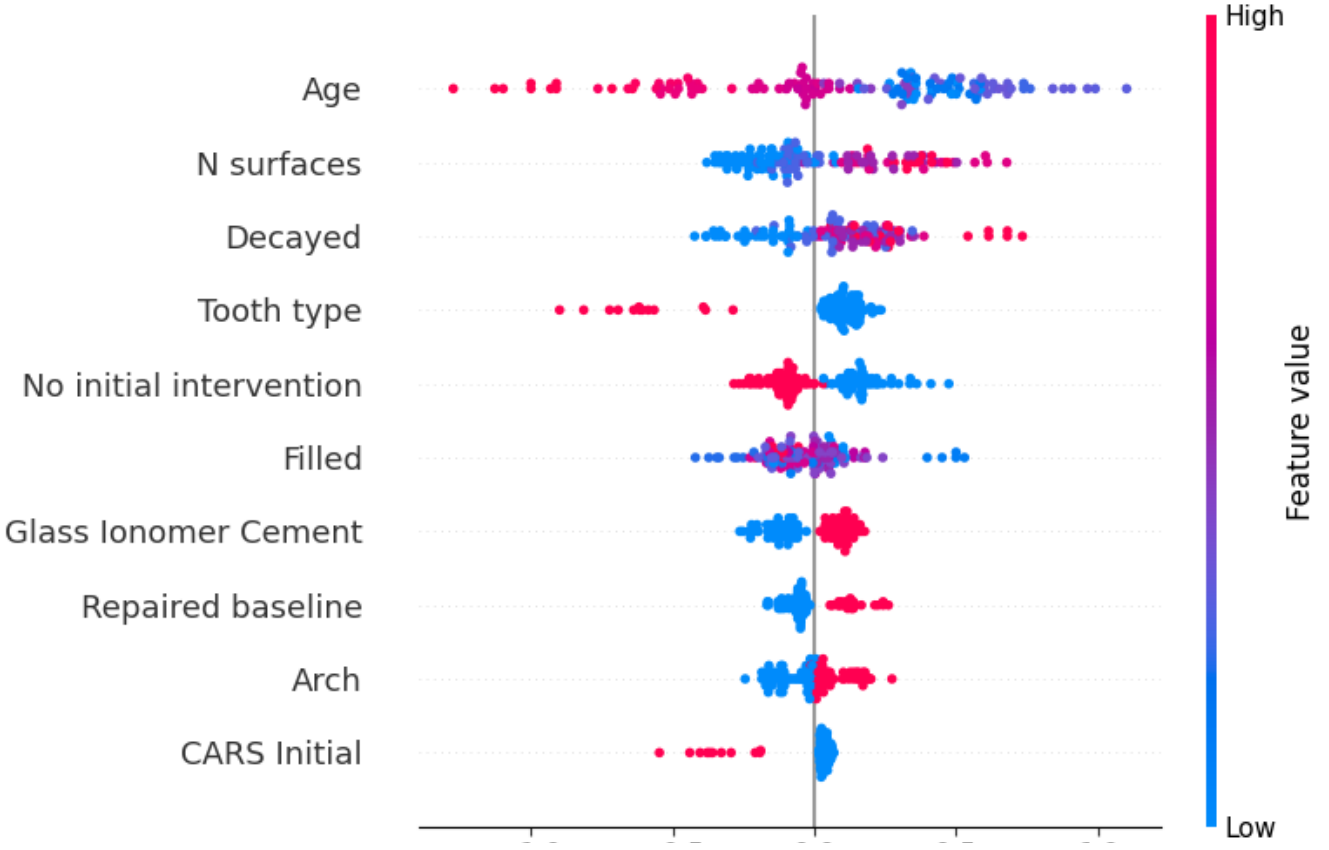
Confusion Matrix XGBoosting

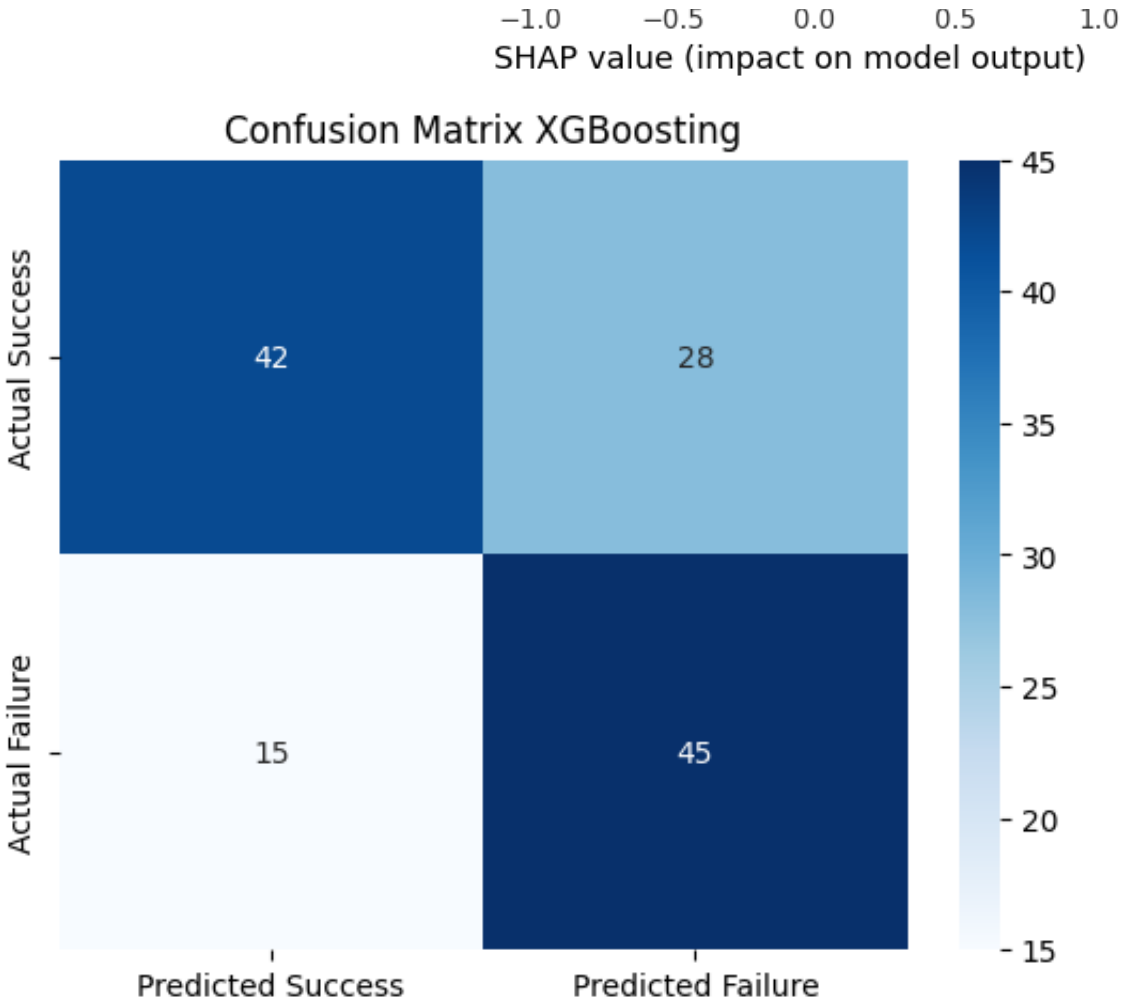


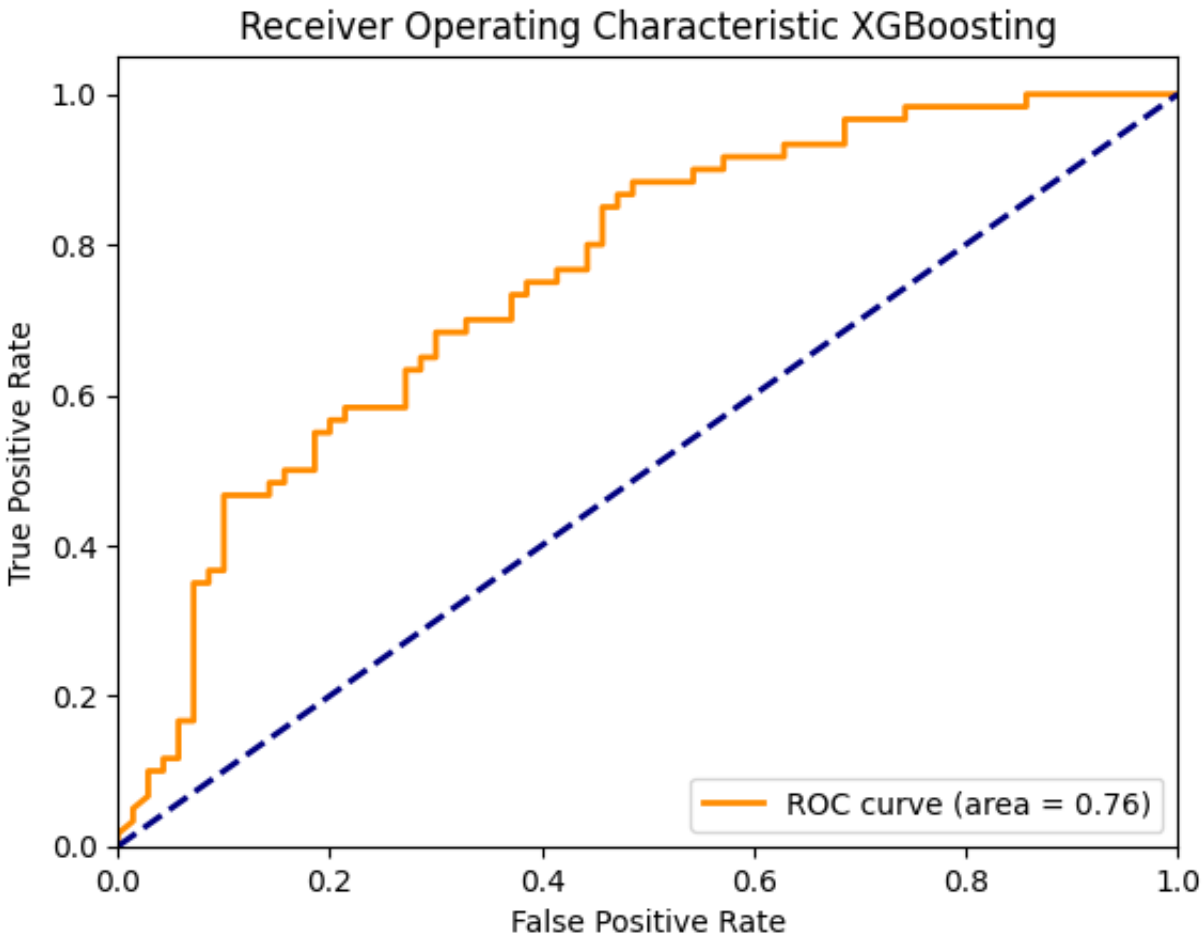



```
Running evaluation with seed 44
Inside evaluate_xgboost function
Evaluating XGBoost...
Best parameters for XGBoost: {'colsample_bytree': 1, 'gamma': 0.1, 'learnin
```

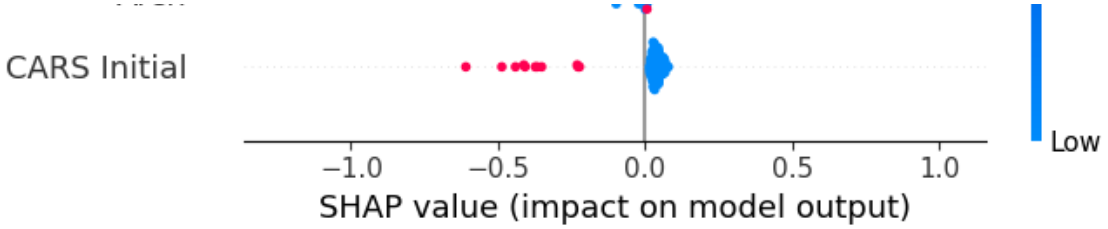
```
Best parameters for XGBoost: { colsample_bytree : 1,  gamma : 0.1,  learnin

--- Dados ROC para copiar ---
FPR = [0.0, 0.0, 0.014285714285714285, 0.014285714285714285, 0.028571428571
TPR = [0.0, 0.016666666666666666, 0.03333333333333333, 0.05, 0.066666666666
AUC = 0.7573809523809524
--- Fim dos Dados ROC ---

Training - Accuracy: 0.863905325443787, Sensitivity: 0.8725099601593626, Sp
Metrics for manual threshold 0.5:
Accuracy: 0.6692307692307692, Sensitivity: 0.75, Specificity: 0.6, F1: 0.67
Threshold: 0.10, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.15, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.20, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.25, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.30, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.35, Metrics: {'Accuracy': 0.5307692307692308, 'Sensitivity': 0
Threshold: 0.40, Metrics: {'Accuracy': 0.5846153846153846, 'Sensitivity': 0
Threshold: 0.45, Metrics: {'Accuracy': 0.6692307692307692, 'Sensitivity': 0
Threshold: 0.50, Metrics: {'Accuracy': 0.6692307692307692, 'Sensitivity': 0
Threshold: 0.55, Metrics: {'Accuracy': 0.676923076923077, 'Sensitivity': 0.
Threshold: 0.60, Metrics: {'Accuracy': 0.6846153846153846, 'Sensitivity': 0
Threshold: 0.65, Metrics: {'Accuracy': 0.5615384615384615, 'Sensitivity': 0
Threshold: 0.70, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.75, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.80, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.85, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.90, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.95, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 1.00, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
SHAP Summary for XGBoost
```

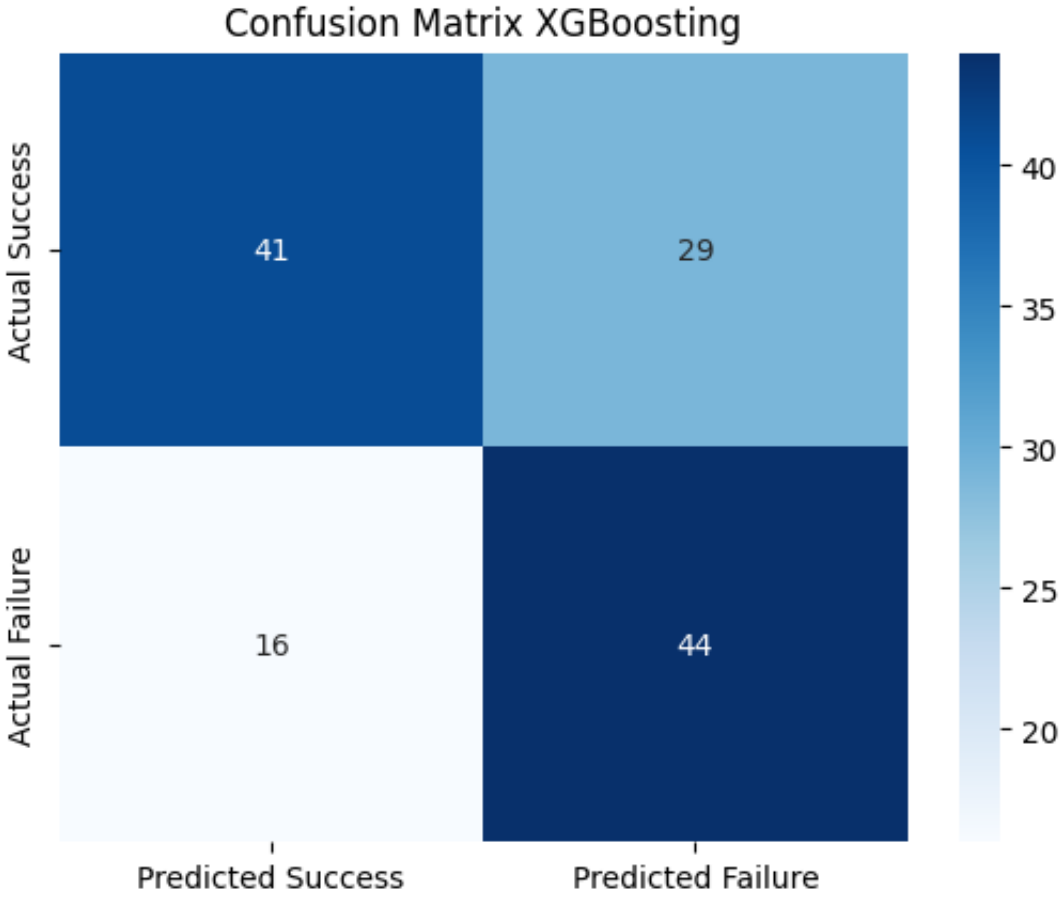


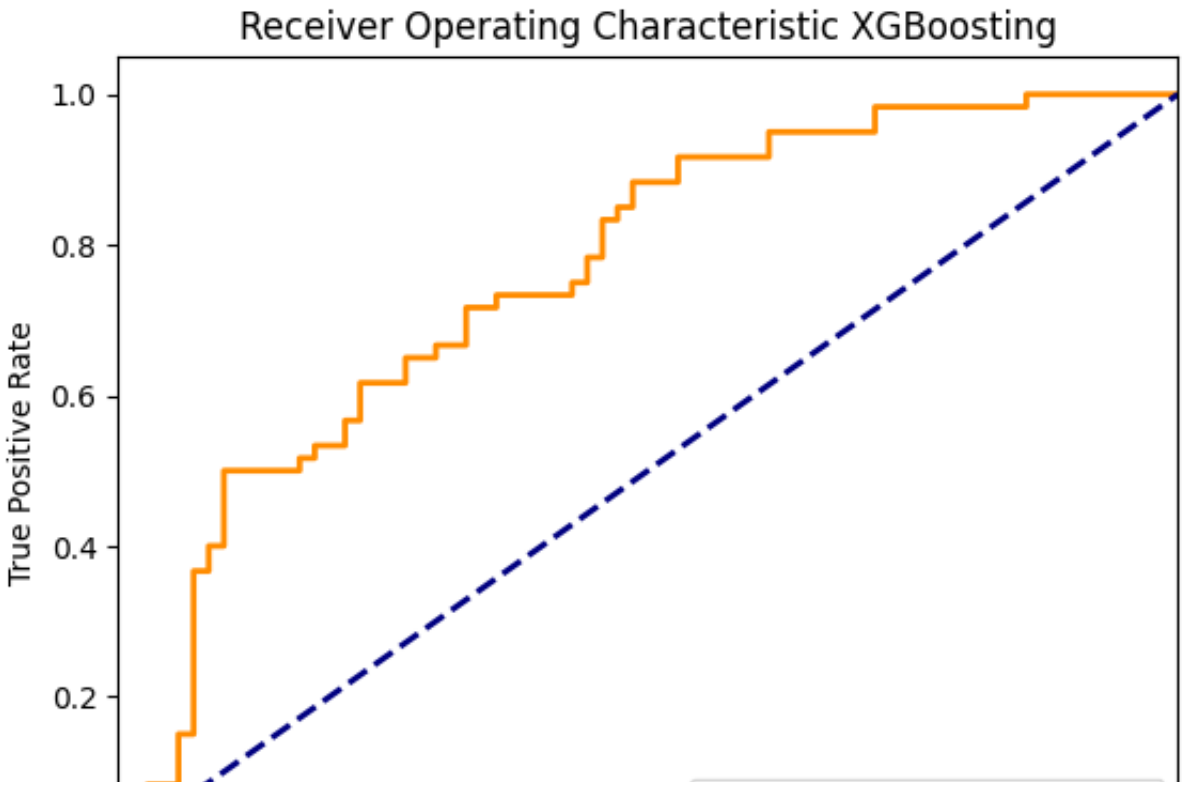

−1.0
−0.5
0.0
0.5
1.0
SHAP value (impact on model output)
Confusion Matrix XGBoosting
Actual Success
Actual Failure
42
28
15
45
Predicted Success
Predicted Failure
45
40
35
30
25
20
15

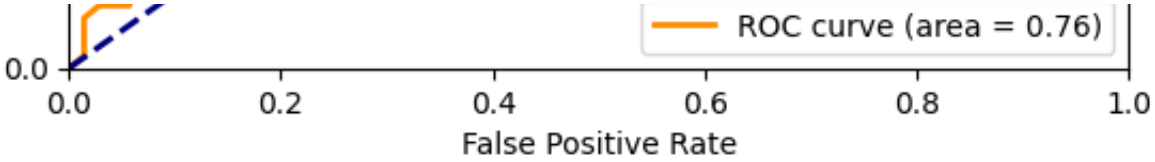

Receiver Operating Characteristic XGBoosting
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (area = 0.76)

```
Running evaluation with seed 45
Inside evaluate_xgboost function
Evaluating XGBoost...
Best parameters for XGBoost: {'colsample_bytree': 1, 'gamma': 0.1, 'learnin

--- Dados ROC para copiar ---
FPR = [0.0, 0.014285714285714285, 0.014285714285714285, 0.02857142857142857
TPR = [0.0, 0.016666666666666666, 0.06666666666666667, 0.08333333333333333,
AUC = 0.7607142857142857
--- Fim dos Dados ROC ---

Training - Accuracy: 0.8678500986193294, Sensitivity: 0.8844621513944223, S
Metrics for manual threshold 0.5:
Accuracy: 0.6538461538461539, Sensitivity: 0.7333333333333333, Specificity:
Threshold: 0.10, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.15, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.20, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.25, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.30, Metrics: {'Accuracy': 0.4692307692307692, 'Sensitivity':
Threshold: 0.35, Metrics: {'Accuracy': 0.5461538461538461, 'Sensitivity': 0
Threshold: 0.40, Metrics: {'Accuracy': 0.5923076923076923, 'Sensitivity': 0
Threshold: 0.45, Metrics: {'Accuracy': 0.6692307692307692, 'Sensitivity': 0
Threshold: 0.50, Metrics: {'Accuracy': 0.6538461538461539, 'Sensitivity': 0
Threshold: 0.55, Metrics: {'Accuracy': 0.7, 'Sensitivity': 0.61666666666666
Threshold: 0.60, Metrics: {'Accuracy': 0.6846153846153846, 'Sensitivity': 0
Threshold: 0.65, Metrics: {'Accuracy': 0.5769230769230769, 'Sensitivity': 0
Threshold: 0.70, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.75, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.80, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.85, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.90, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.95, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 1.00, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
SHAP Summary for XGBoost
```

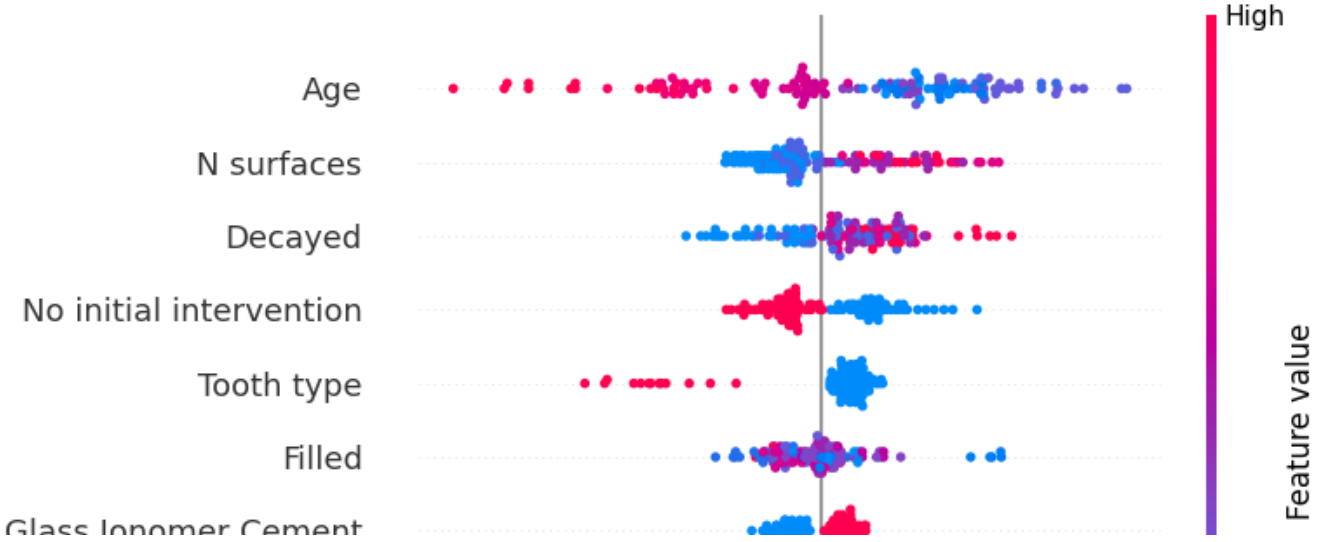

CARS Initial
−1.0
−0.5
0.0
0.5
1.0
SHAP value (impact on model output)
Low

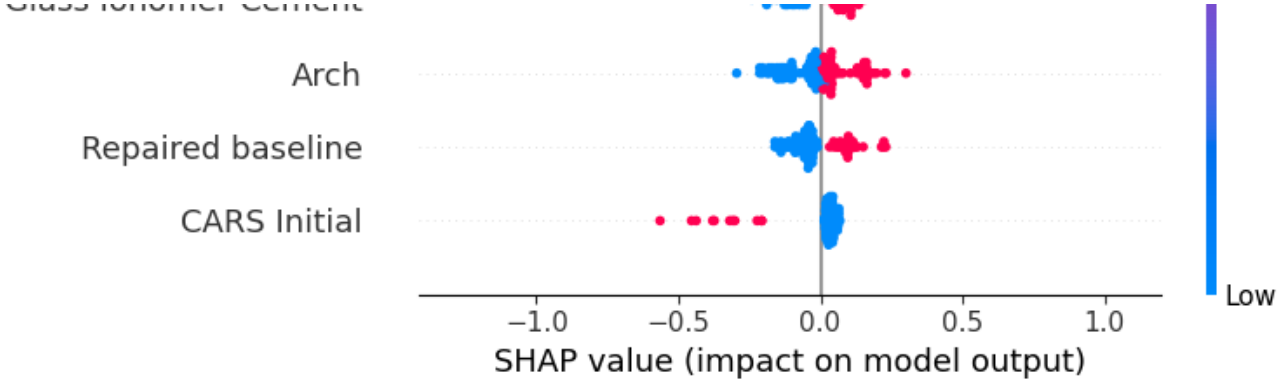

Confusion Matrix XGBoosting
Actual Success
Actual Failure
41
29
16
44
40
35
30
25
20
Predicted Success
Predicted Failure

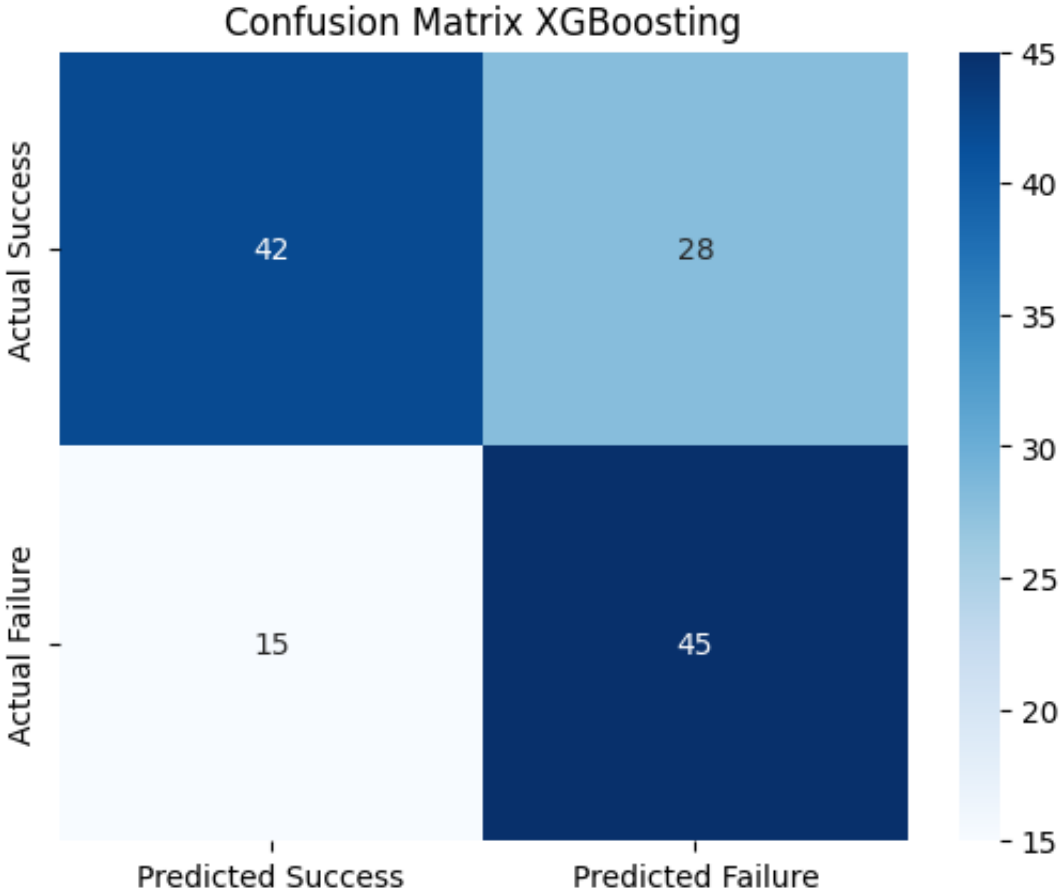

Receiver Operating Characteristic XGBoosting
True Positive Rate
1.0
0.8
0.6
0.4
0.2

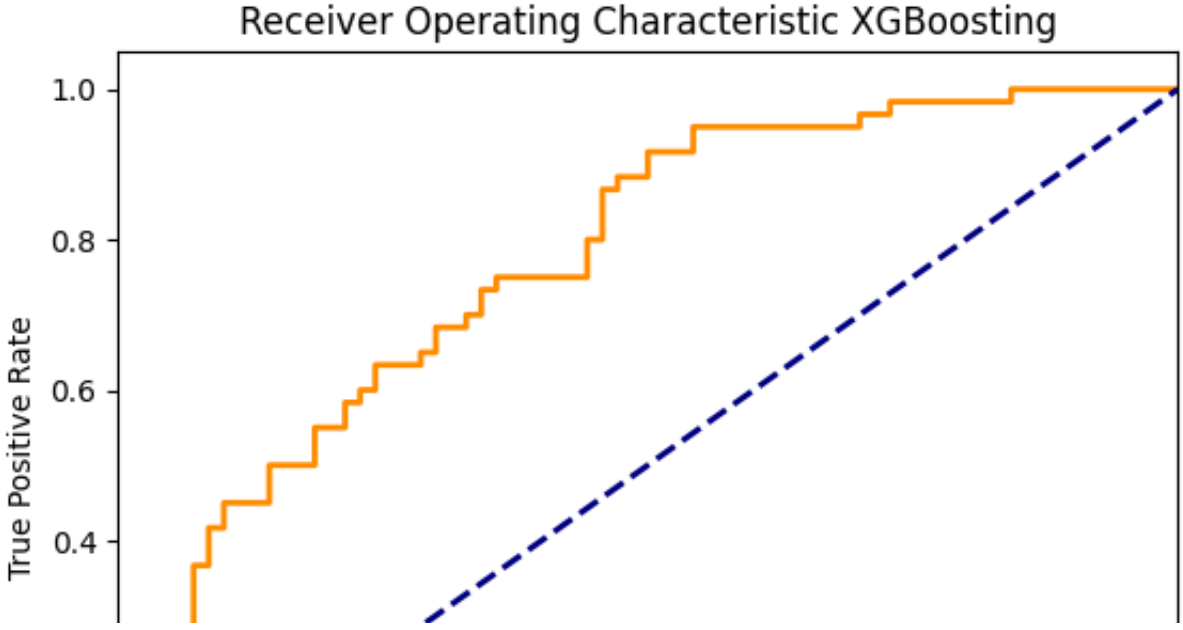


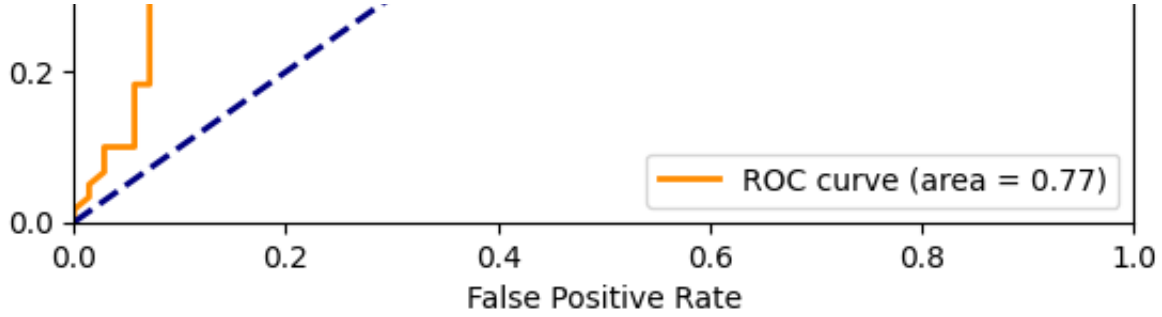

```
Running evaluation with seed 46
Inside evaluate_xgboost function
Evaluating XGBoost...
Best parameters for XGBoost: {'colsample_bytree': 1, 'gamma': 0.1, 'learnin

--- Dados ROC para copiar ---
FPR = [0.0, 0.0, 0.014285714285714285, 0.014285714285714285, 0.028571428571
TPR = [0.0, 0.016666666666666666, 0.03333333333333333, 0.05, 0.066666666666
AUC = 0.7666666666666667
--- Fim dos Dados ROC ---

Training - Accuracy: 0.8678500986193294, Sensitivity: 0.8844621513944223, S
Metrics for manual threshold 0.5:
Accuracy: 0.6692307692307692, Sensitivity: 0.75, Specificity: 0.6, F1: 0.67
Threshold: 0.10, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.15, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.20, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.25, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.30, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.35, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.40, Metrics: {'Accuracy': 0.5923076923076923, 'Sensitivity': 0
Threshold: 0.45, Metrics: {'Accuracy': 0.6923076923076923, 'Sensitivity': 0
Threshold: 0.50, Metrics: {'Accuracy': 0.6692307692307692, 'Sensitivity': 0
Threshold: 0.55, Metrics: {'Accuracy': 0.6923076923076923, 'Sensitivity': 0
Threshold: 0.60, Metrics: {'Accuracy': 0.676923076923077, 'Sensitivity': 0.
Threshold: 0.65, Metrics: {'Accuracy': 0.5769230769230769, 'Sensitivity': 0
Threshold: 0.70, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.75, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.80, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.85, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.90, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.95, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 1.00, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
SHAP Summary for XGBoost
```



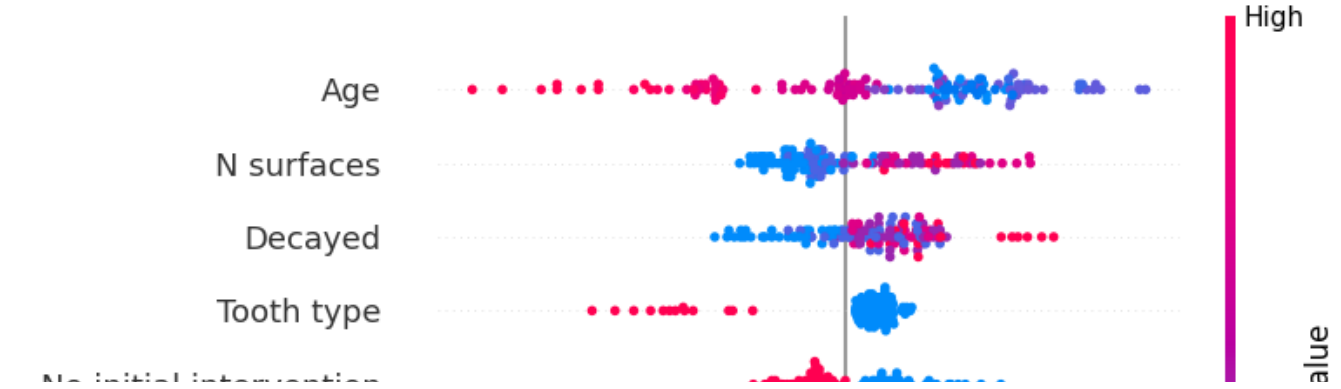

Arch
Repaired baseline
CARS Initial
−1.0
−0.5
0.0
0.5
1.0
SHAP value (impact on model output)
Low

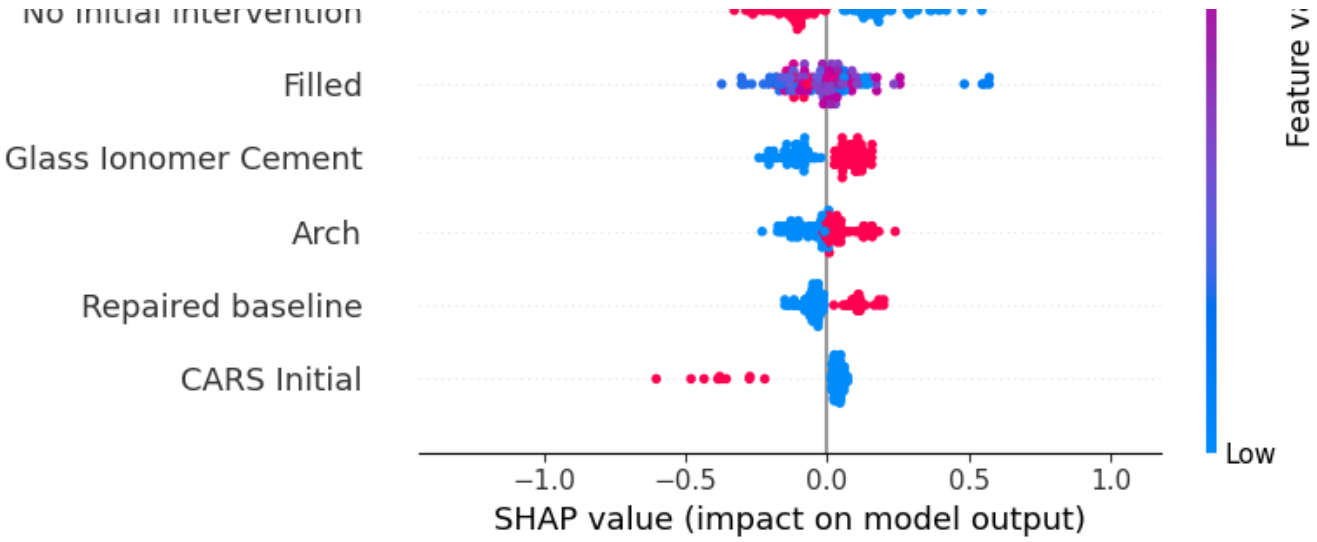

Confusion Matrix XGBoosting
Actual Success
Actual Failure
42
28
15
45
Predicted Success
Predicted Failure
45
40
35
30
25
20
15

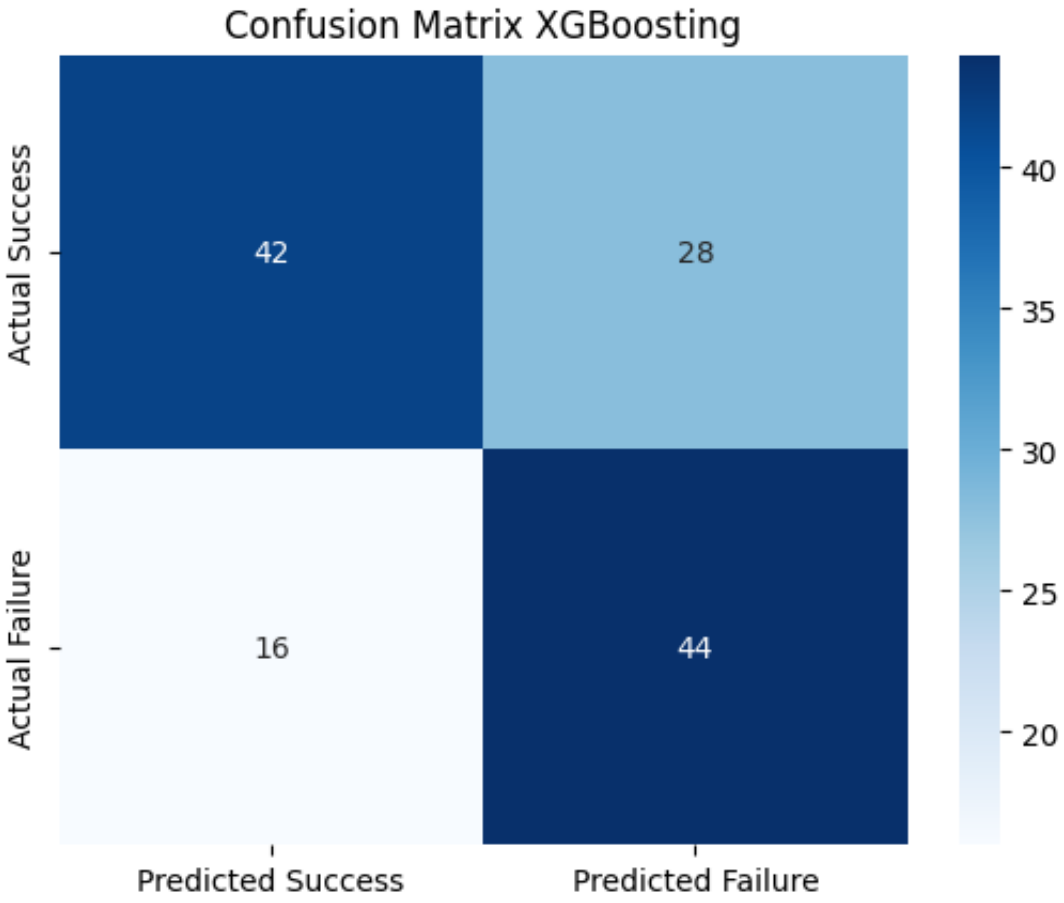

Receiver Operating Characteristic XGBoosting
True Positive Rate
1.0
0.8
0.6
0.4

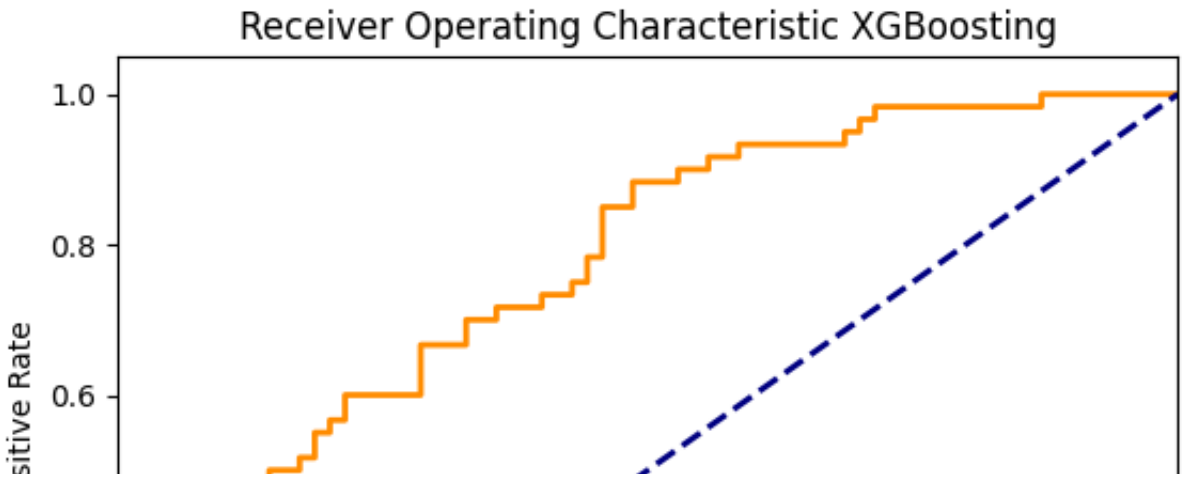


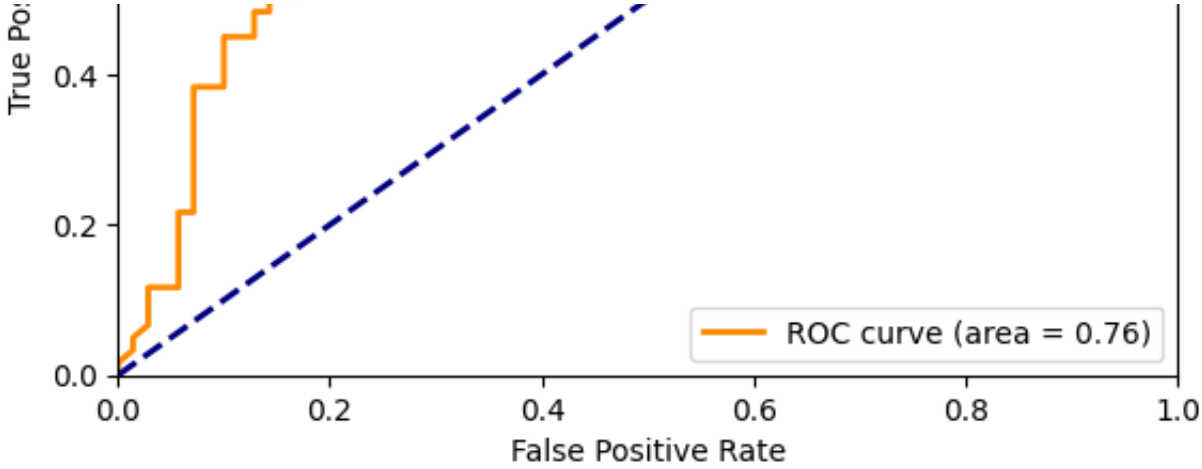

```
Running evaluation with seed 47
Inside evaluate_xgboost function
Evaluating XGBoost...
Best parameters for XGBoost: {'colsample_bytree': 1, 'gamma': 0.1, 'learnin

--- Dados ROC para copiar ---
FPR = [0.0, 0.0, 0.014285714285714285, 0.014285714285714285, 0.028571428571
TPR = [0.0, 0.016666666666666666, 0.03333333333333333, 0.05, 0.066666666666
AUC = 0.7588095238095238
--- Fim dos Dados ROC ---

Training - Accuracy: 0.8678500986193294, Sensitivity: 0.8844621513944223, S
Metrics for manual threshold 0.5:
Accuracy: 0.6615384615384615, Sensitivity: 0.7333333333333333, Specificity:
Threshold: 0.10, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.15, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.20, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.25, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.30, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.35, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.40, Metrics: {'Accuracy': 0.5923076923076923, 'Sensitivity': 0
Threshold: 0.45, Metrics: {'Accuracy': 0.6692307692307692, 'Sensitivity': 0
Threshold: 0.50, Metrics: {'Accuracy': 0.6615384615384615, 'Sensitivity': 0
Threshold: 0.55, Metrics: {'Accuracy': 0.6615384615384615, 'Sensitivity': 0
Threshold: 0.60, Metrics: {'Accuracy': 0.6923076923076923, 'Sensitivity': 0
Threshold: 0.65, Metrics: {'Accuracy': 0.5846153846153846, 'Sensitivity': 0
Threshold: 0.70, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.75, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.80, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.85, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.90, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.95, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 1.00, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
SHAP Summary for XGBoost
```



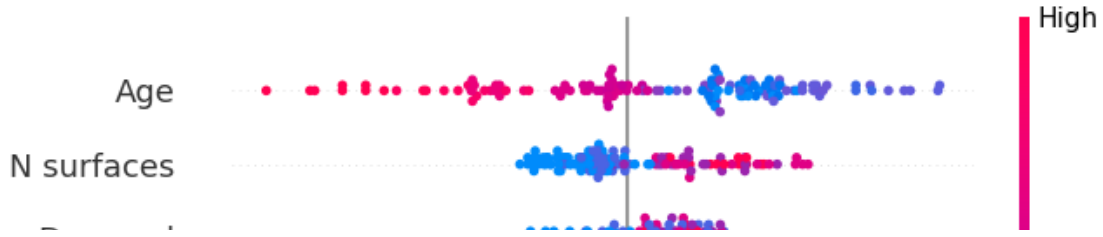

No Initial Intervention
Filled
Glass Ionomer Cement
Arch
Repaired baseline
CARS Initial
−1.0
−0.5
0.0
0.5
1.0
SHAP value (impact on model output)
Low

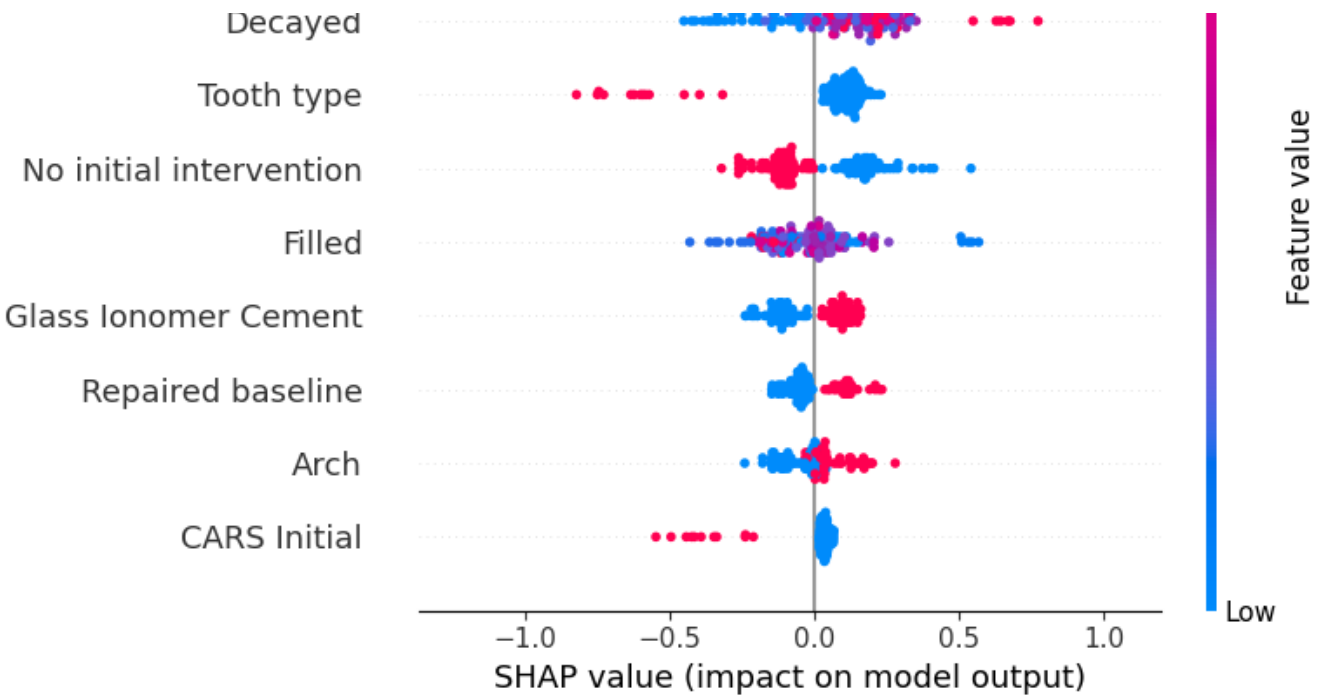

Confusion Matrix XGBoosting
Actual Success
Actual Failure
42
28
16
44
Predicted Success
Predicted Failure
40
35
30
25
20

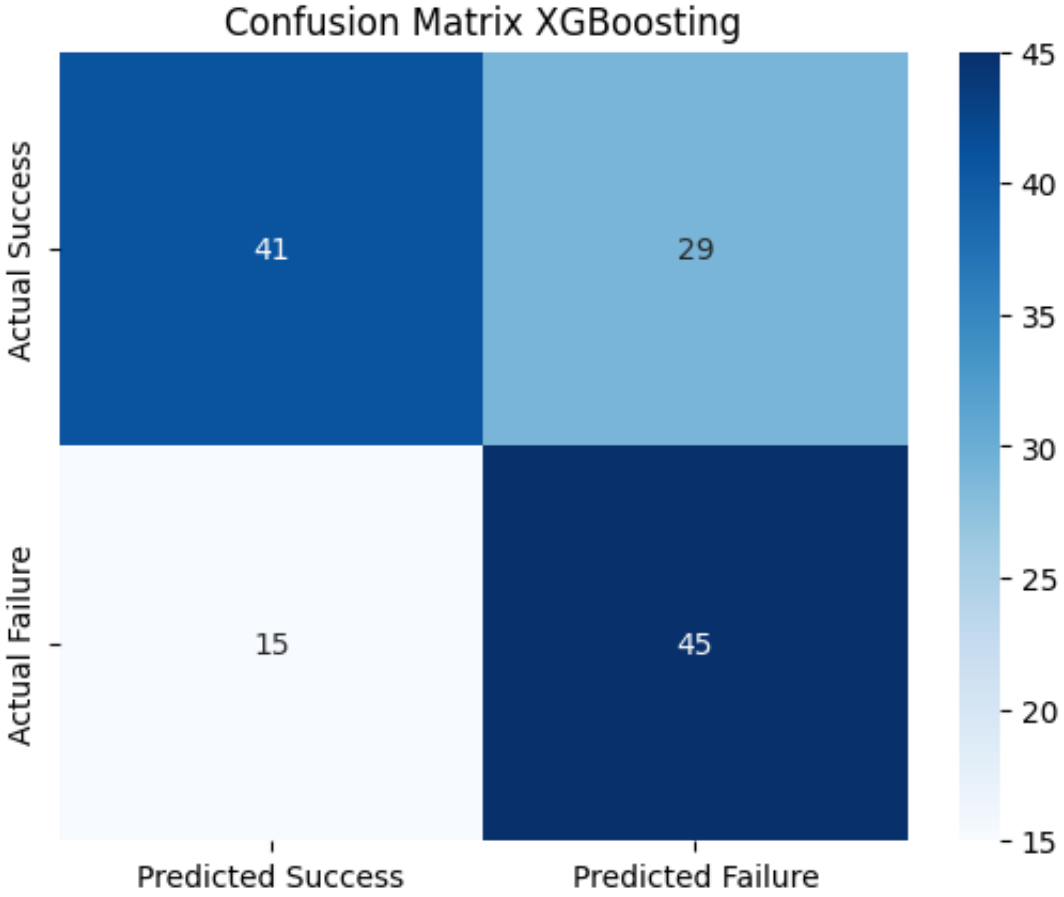

Receiver Operating Characteristic XGBoosting
1.0
0.8
0.6

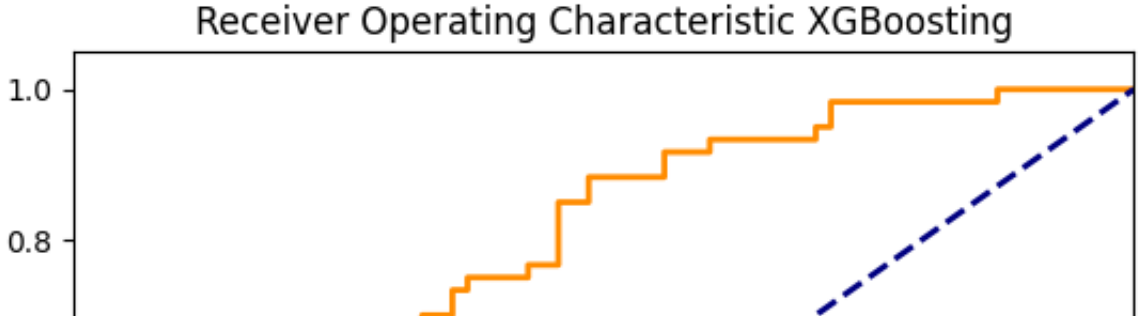


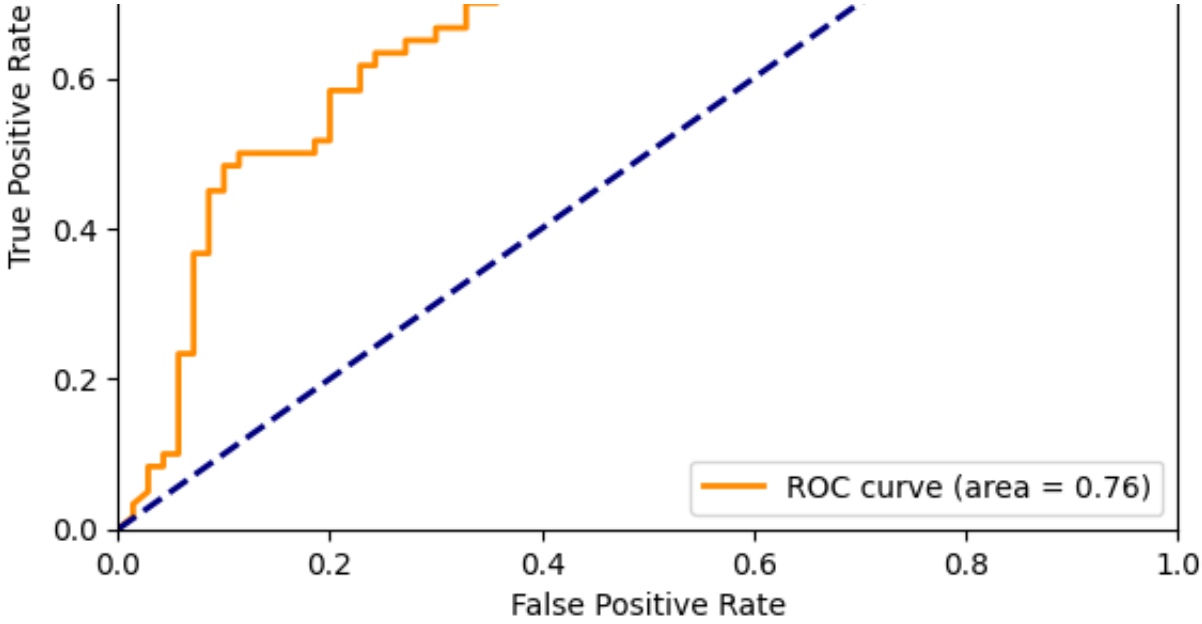

```
Running evaluation with seed 48
Inside evaluate_xgboost function
Evaluating XGBoost...
Best parameters for XGBoost: {'colsample_bytree': 1, 'gamma': 0.1, 'learnin

--- Dados ROC para copiar ---
FPR = [0.0, 0.014285714285714285, 0.014285714285714285, 0.02857142857142857
TPR = [0.0, 0.016666666666666666, 0.03333333333333333, 0.05, 0.083333333333
AUC = 0.7614285714285713
--- Fim dos Dados ROC ---

Training - Accuracy: 0.8619329388560157, Sensitivity: 0.8605577689243028, S
Metrics for manual threshold 0.5:
Accuracy: 0.6615384615384615, Sensitivity: 0.75, Specificity: 0.58571428571
Threshold: 0.10, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.15, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.20, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.25, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.30, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.35, Metrics: {'Accuracy': 0.5307692307692308, 'Sensitivity': 0
Threshold: 0.40, Metrics: {'Accuracy': 0.5923076923076923, 'Sensitivity': 0
Threshold: 0.45, Metrics: {'Accuracy': 0.6461538461538462, 'Sensitivity': 0
Threshold: 0.50, Metrics: {'Accuracy': 0.6615384615384615, 'Sensitivity': 0
Threshold: 0.55, Metrics: {'Accuracy': 0.7, 'Sensitivity': 0.63333333333333
Threshold: 0.60, Metrics: {'Accuracy': 0.7076923076923077, 'Sensitivity': 0
Threshold: 0.65, Metrics: {'Accuracy': 0.5769230769230769, 'Sensitivity': 0
Threshold: 0.70, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.75, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.80, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.85, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.90, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.95, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 1.00, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
SHAP Summary for XGBoost
```



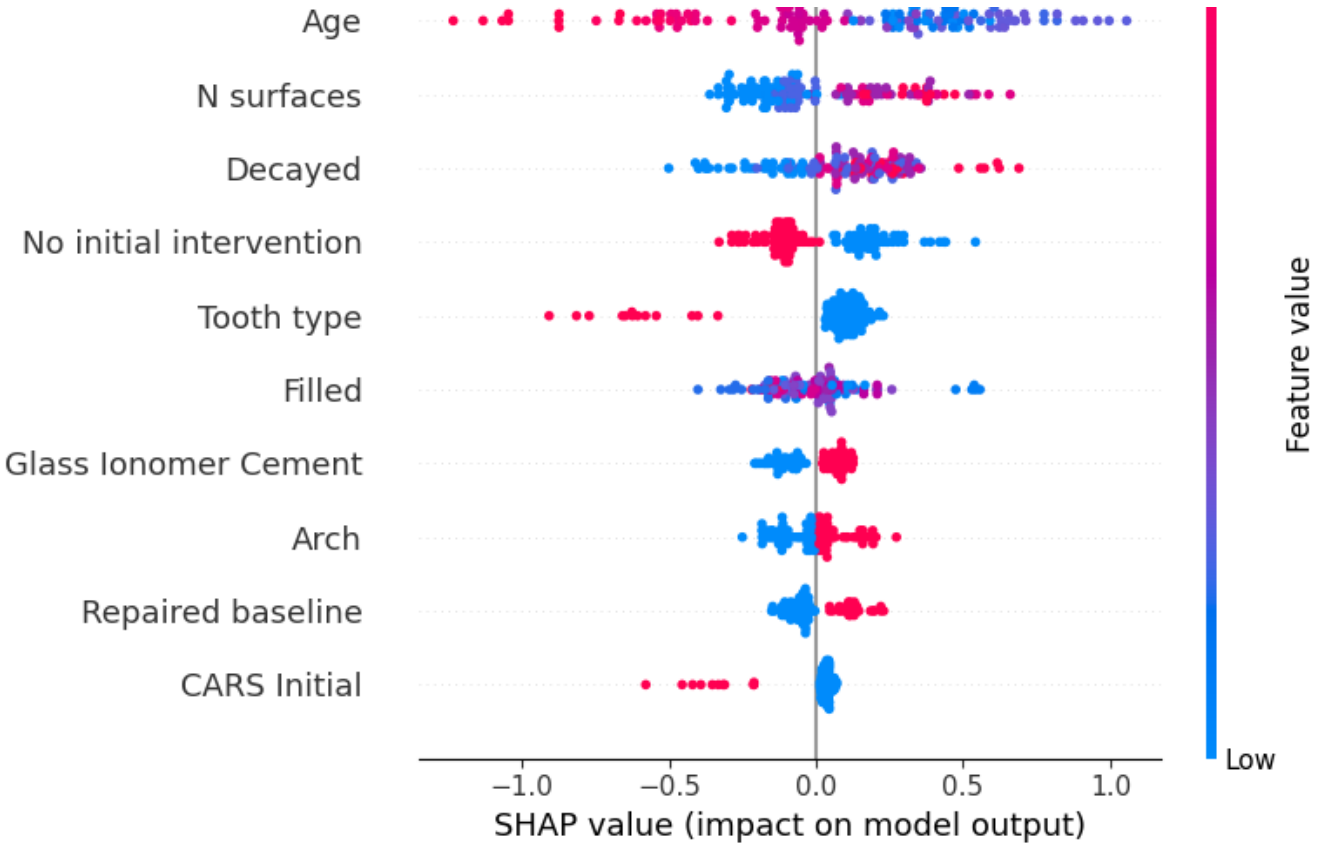

Decayed
Tooth type
No initial intervention
Filled
Glass Ionomer Cement
Repaired baseline
Arch
CARS Initial
−1.0
−0.5
0.0
0.5
1.0
SHAP value (impact on model output)
Feature value
Low

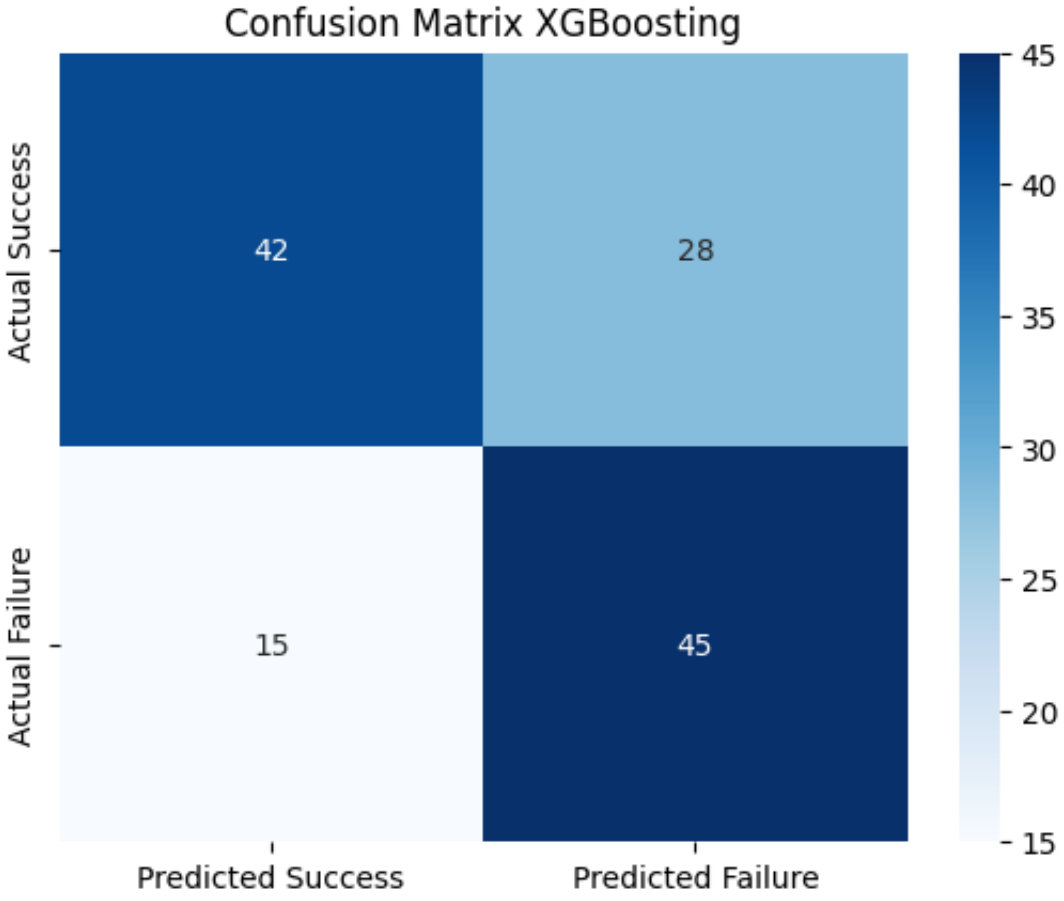

Confusion Matrix XGBoosting
Actual Success
Actual Failure
41
29
15
45
Predicted Success
Predicted Failure
45
40
35
30
25
20
15

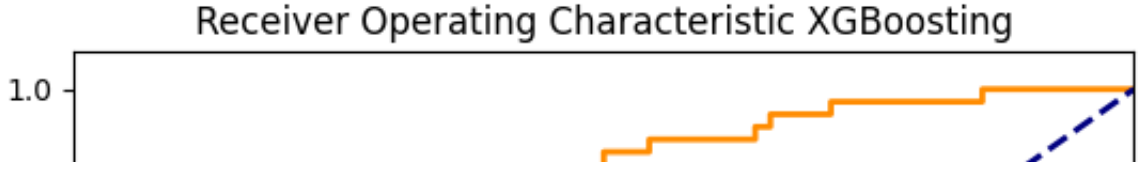

Receiver Operating Characteristic XGBoosting
1.0
0.8

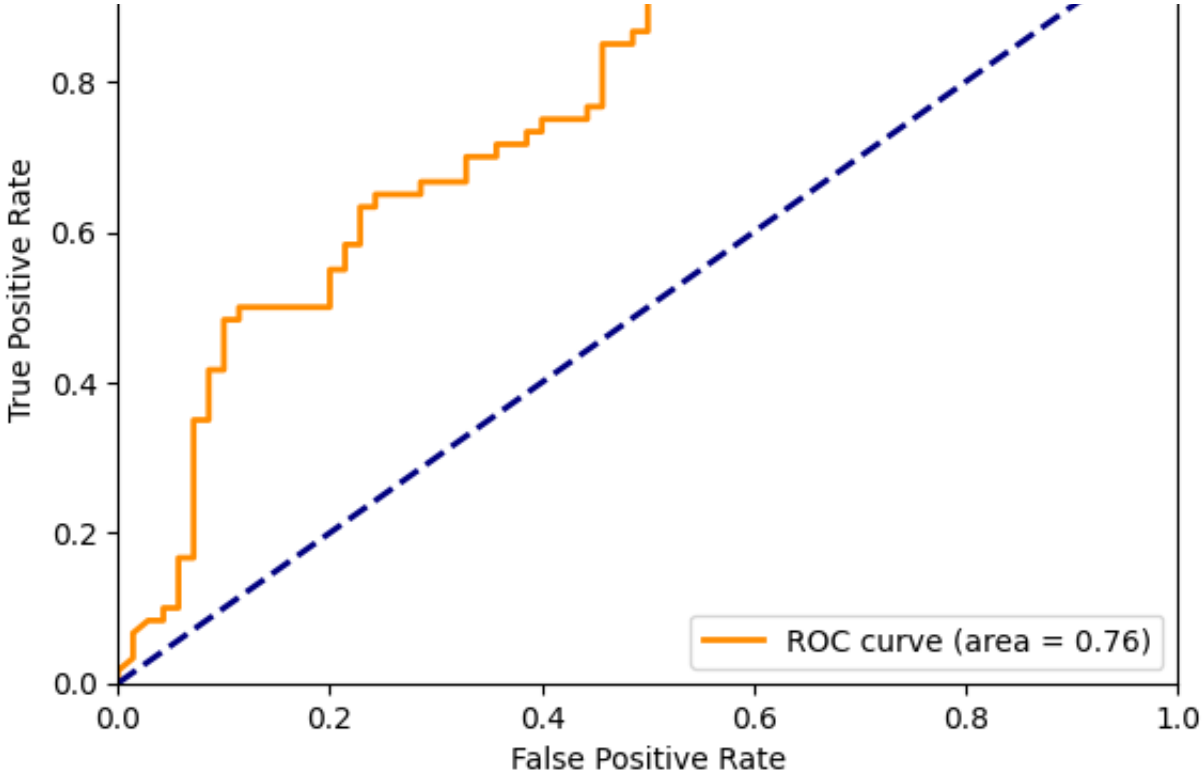



```
Running evaluation with seed 49
Inside evaluate_xgboost function
Evaluating XGBoost...
Best parameters for XGBoost: {'colsample_bytree': 1, 'gamma': 0.1, 'learnin

--- Dados ROC para copiar ---
FPR = [0.0, 0.0, 0.014285714285714285, 0.014285714285714285, 0.028571428571
TPR = [0.0, 0.016666666666666666, 0.03333333333333333, 0.06666666666666667,
AUC = 0.7642857142857143
--- Fim dos Dados ROC ---

Training - Accuracy: 0.8560157790927022, Sensitivity: 0.8645418326693227, S
Metrics for manual threshold 0.5:
Accuracy: 0.6692307692307692, Sensitivity: 0.75, Specificity: 0.6, F1: 0.67
Threshold: 0.10, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.15, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.20, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.25, Metrics: {'Accuracy': 0.46153846153846156, 'Sensitivity':
Threshold: 0.30, Metrics: {'Accuracy': 0.4692307692307692, 'Sensitivity':
Threshold: 0.35, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.40, Metrics: {'Accuracy': 0.5923076923076923, 'Sensitivity': 0
Threshold: 0.45, Metrics: {'Accuracy': 0.676923076923077, 'Sensitivity': 0.
Threshold: 0.50, Metrics: {'Accuracy': 0.6692307692307692, 'Sensitivity': 0
Threshold: 0.55, Metrics: {'Accuracy': 0.7076923076923077, 'Sensitivity': 0
Threshold: 0.60, Metrics: {'Accuracy': 0.6923076923076923, 'Sensitivity': 0
Threshold: 0.65, Metrics: {'Accuracy': 0.5769230769230769, 'Sensitivity': 0
Threshold: 0.70, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.75, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.80, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.85, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.90, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 0.95, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
Threshold: 1.00, Metrics: {'Accuracy': 0.5384615384615384, 'Sensitivity': 0
SHAP Summary for XGBoost
```


Age
N surfaces
Decayed
No initial intervention
Tooth type
Filled
Glass Ionomer Cement
Arch
Repaired baseline
CARS Initial
−1.0
−0.5
0.0
0.5
1.0
SHAP value (impact on model output)
Feature value
Low


Confusion Matrix XGBoosting
Actual Success
Actual Failure
42
28
15
45
Predicted Success
Predicted Failure
45
40
35
30
25
20
15


Receiver Operating Characteristic XGBoosting
1.0



Aggregated Test Set Metrics Across Seeds:

	accuracy	sensitivity	specificity	f1	roc_auc
0	0.661538	0.750000	0.585714	0.671642	0.754048
1	0.661538	0.750000	0.585714	0.671642	0.752619
2	0.661538	0.733333	0.600000	0.666667	0.760952
3	0.676923	0.733333	0.628571	0.676923	0.763571
4	0.669231	0.750000	0.600000	0.676692	0.757381
5	0.653846	0.733333	0.585714	0.661654	0.760714
6	0.669231	0.750000	0.600000	0.676692	0.766667
7	0.661538	0.733333	0.600000	0.666667	0.758810
8	0.661538	0.750000	0.585714	0.671642	0.761429
9	0.669231	0.750000	0.600000	0.676692	0.764286

Summary of Test Set Metrics (Mean, Standard Error, 95% Confidence Interval)
Accuracy: Mean = 0.665, SE = 0.002, 95% CI = [0.660, 0.669]
Sensitivity: Mean = 0.743, SE = 0.003, 95% CI = [0.737, 0.749]
Specificity: Mean = 0.597, SE = 0.004, 95% CI = [0.588, 0.607]
F1: Mean = 0.672, SE = 0.002, 95% CI = [0.668, 0.675]
Roc_auc: Mean = 0.760, SE = 0.001, 95% CI = [0.757, 0.763]

```
# Set seeds for reproducibility
seed_value = 42
np.random.seed(seed_value)
random.seed(seed_value)
```

```
tf.random.set_seed(seed_value)

# Define a function to build, train, and evaluate a neural network model.
def evaluate_neural_network(X_train, y_train, X_test, y_test, threshold_list):
    # Initialize the neural network model with specified layers.
    model = Sequential([
        Dense(128, activation='relu', kernel_regularizer=l2(0.01), input_shape=(
        BatchNormalization(),
        Dropout(0.3),
        Dense(64, activation='relu', kernel_regularizer=l2(0.01)),
        BatchNormalization(),
        Dropout(0.3),
        Dense(1, activation='sigmoid')
    ])

    # Compile the model specifying the optimizer, loss function, and metrics.
    model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accura

    # Define callbacks for early stopping and learning rate reduction.
    early_stopping = EarlyStopping(monitor='val_loss', patience=10, restore_best
    reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=5, mi

    # Train the model with a validation split, epochs, batch size, and callbacks
    model.fit(X_train, y_train, validation_split=0.2, epochs=30, batch_size=32,

    # Training set evaluation
    y_train_probs = model.predict(X_train).ravel()
    y_train_pred = (y_train_probs >= threshold_list[-1]).astype(int)  # Use the
    train_acc = accuracy_score(y_train, y_train_pred)
    train_sens = sensitivity(y_train, y_train_pred)
    train_spec = specificity(y_train, y_train_pred)
    train_f1 = f1_score(y_train, y_train_pred)
    train_roc_auc = roc_auc_score(y_train, y_train_probs)

    # Test set evaluation for multiple thresholds
    y_probs = model.predict(X_test).ravel()
    thresholds_metrics = []

    # Calcular FPR, TPR e AUC
    fpr, tpr, thresholds = roc_curve(y_test, y_probs)
    roc_auc = auc(fpr, tpr)

    # Imprimir os valores de FPR, TPR e AUC de forma fácil de copiar
    print("\n--- Dados ROC para copiar ---")
    print("FPR =", fpr.tolist())
    print("TPR =", tpr.tolist())
    print("AUC =", roc_auc)
    print("--- Fim dos Dados ROC ---\n")
```

```
    for threshold in threshold_list:
        y_pred = (y_probs >= threshold).astype(int)
        acc = accuracy_score(y_test, y_pred)
        sens = sensitivity(y_test, y_pred)
        spec = specificity(y_test, y_pred)
        f1 = f1_score(y_test, y_pred)
        thresholds_metrics.append({
            'threshold': threshold,
            'accuracy': acc,
            'sensitivity': sens,
            'specificity': spec,
            'f1_score': f1
        })

    # Print training set metrics
    print(f"Training - Accuracy: {train_acc:.4f}, Sensitivity: {train_sens:.4f},

    # Test set ROC AUC
    test_roc_auc = roc_auc_score(y_test, y_probs)

    # Print test set metrics for each threshold
    for metrics in thresholds_metrics:
        print(f"Threshold: {metrics['threshold']:.2f}, Accuracy: {metrics['accur

    return model, train_acc, train_sens, train_spec, train_f1, train_roc_auc, te

# Plotting functions for confusion matrix and ROC curve visualization.
def plot_confusion_matrix(y_true, y_pred):
    matrix = confusion_matrix(y_true, y_pred)
    sns.heatmap(matrix, annot=True, fmt='d', cmap='Blues',
                xticklabels=['Predicted Success', 'Predicted Failure'],
                yticklabels=['Actual Success', 'Actual Failure'])
    plt.title('Confusion Matrix Neural Network')
    plt.show()

def plot_roc_curve(y_true, y_probs):
    fpr, tpr, thresholds = roc_curve(y_true, y_probs)
    roc_auc = auc(fpr, tpr)

    plt.figure()
    plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (area = {roc_
    plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver Operating Characteristic Neural Network')
    plt.legend(loc="lower right")
```

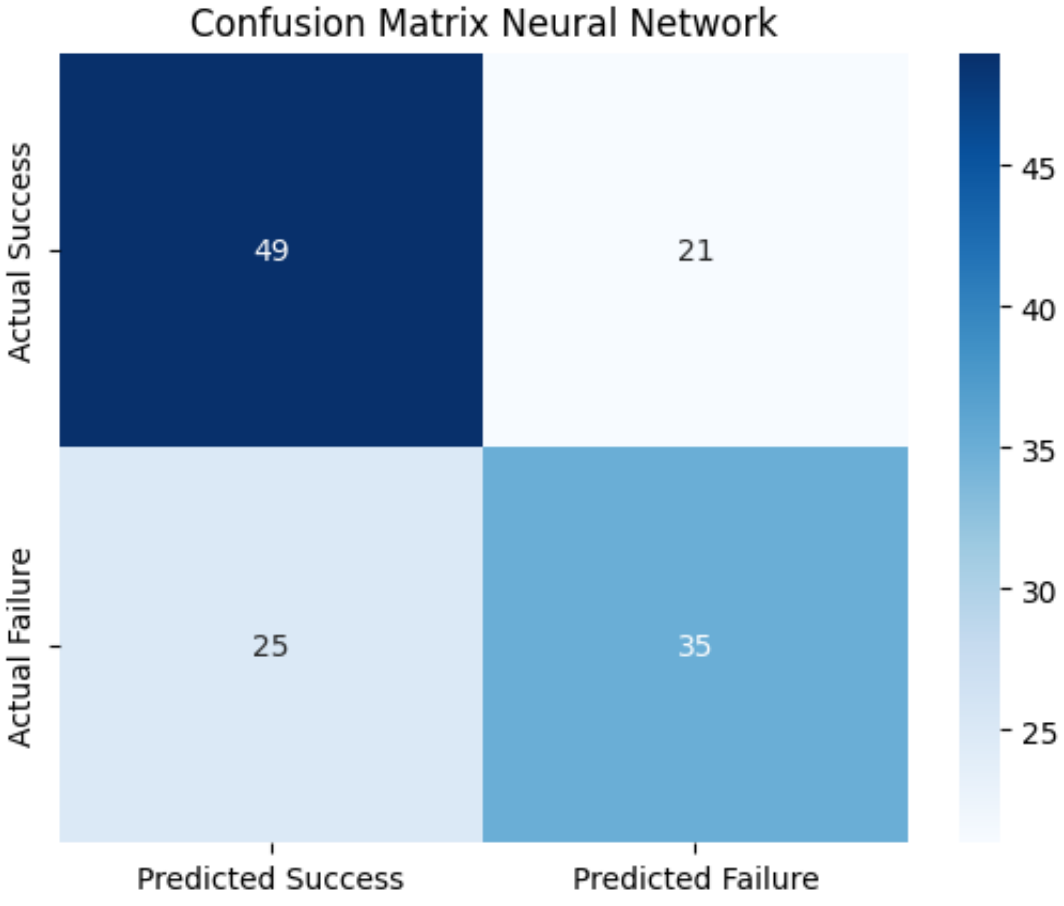
```
        plt.legend(loc='lower right')
        plt.show()

    # Main function where the evaluation process is initiated.
    def main(X_train, y_train, X_test, y_test):
        # Define a list of thresholds to evaluate
        threshold_list = np.arange(0.1, 1.05, 0.05)
        aggregated_metrics = []

        for seed_value in range(40, 50):
            np.random.seed(seed_value)
            random.seed(seed_value)
            tf.random.set_seed(seed_value)

            # Evaluate the neural network model and store its performance metrics.
            model, train_acc, train_sens, train_spec, train_f1, train_roc_auc, test_

            # Choose a threshold for detailed evaluation
            chosen_threshold = 0.45
            y_test_probs = model.predict(X_test).ravel()
            y_test_pred = (y_test_probs >= chosen_threshold).astype(int)

            # Calculate and print the metrics for the chosen threshold
            chosen_acc = accuracy_score(y_test, y_test_pred)
            chosen_sens = sensitivity(y_test, y_test_pred)
            chosen_spec = specificity(y_test, y_test_pred)
            chosen_f1 = f1_score(y_test, y_test_pred)

            print(f"\nMetrics for chosen threshold {chosen_threshold}:")
            print(f"Accuracy: {chosen_acc:.4f}, Sensitivity: {chosen_sens:.4f}, Spec

            # Append the chosen test metrics for aggregation
            test_metrics = {
                "accuracy": chosen_acc,
                "sensitivity": chosen_sens,
                "specificity": chosen_spec,
                "f1": chosen_f1,
                "roc_auc": test_roc_auc
            }
            aggregated_metrics.append(test_metrics)

            # Plotting functions assuming they are defined elsewhere
            plot_confusion_matrix(y_test, y_test_pred)
            plot_roc_curve(y_test, y_test_probs)

        # Aggregate results across seeds
        results_df = pd.DataFrame(aggregated_metrics)
        n = len(results_df)
        print("\nAggregated Test Set Metrics Across Seeds:")
```

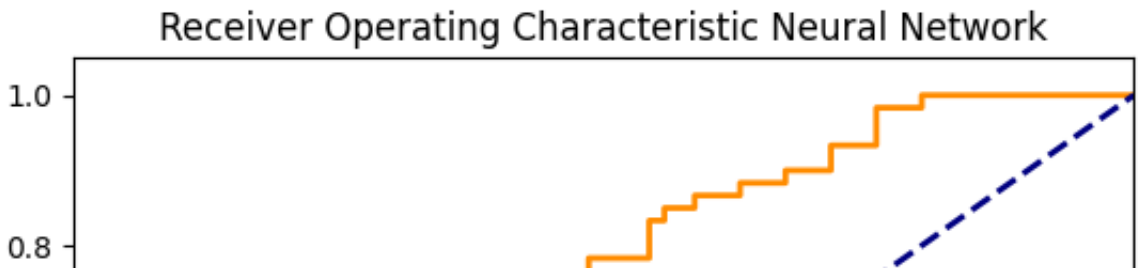
```
    print(results_df)

    # Compute mean, standard error, and 95% confidence interval for each metric
    def summarize_metric(metric_values):
        mean_val = metric_values.mean()
        std_val = metric_values.std(ddof=1)
        se = std_val / np.sqrt(n)
        t_crit = stats.t.ppf(0.975, df=n - 1)
        ci_lower = mean_val - t_crit * se
        ci_upper = mean_val + t_crit * se
        return mean_val, se, (ci_lower, ci_upper)

    metrics_summary = {}
    for metric in results_df.columns:
        mean_val, se, ci = summarize_metric(results_df[metric])
        metrics_summary[metric] = {
            "Mean": mean_val,
            "Standard Error": se,
            "95% CI": ci
        }

    print("\nSummary of Test Set Metrics (Mean, Standard Error, 95% Confidence I
    for metric, summary in metrics_summary.items():
        print(f"{metric.capitalize()}: Mean = {summary['Mean']:.4f}, SE = {summa

# Entry point of the script to run the main function.
if __name__ == '__main__':
    main(X_train, y_train, X_test, y_test)
```

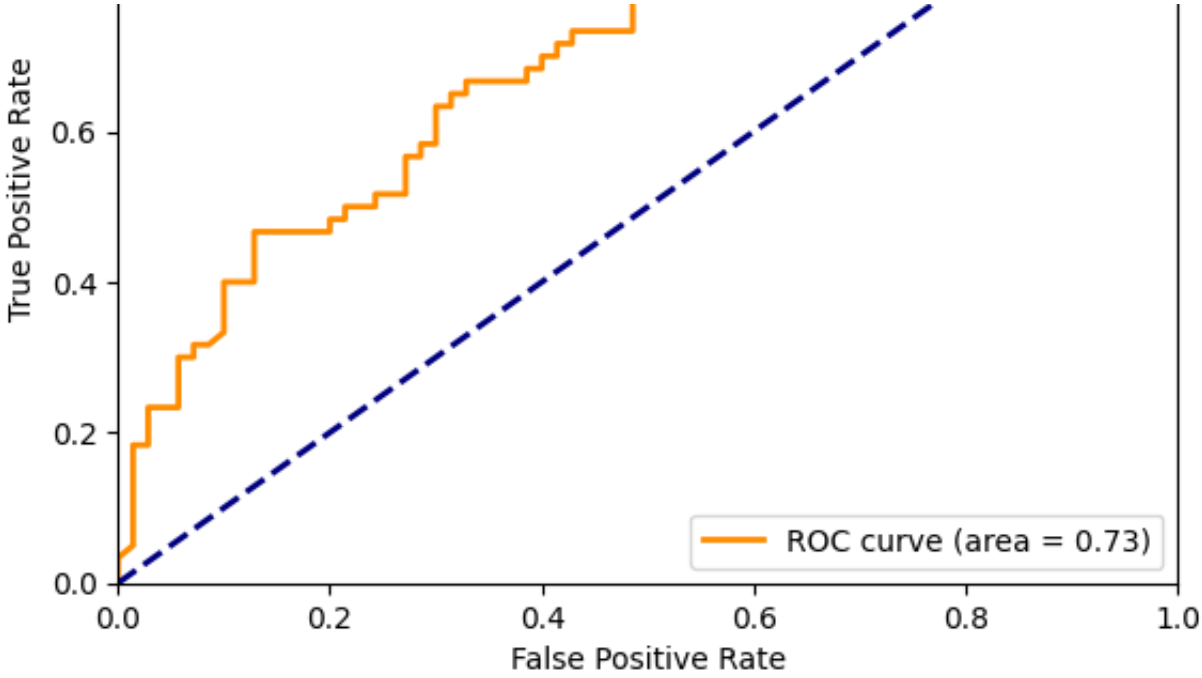
```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.py:87: 
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Epoch 1/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 4s 48ms/step - accuracy: 0.5086 - loss: 2.0989 -
Epoch 2/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - accuracy: 0.6734 - loss: 1.8383 -
Epoch 3/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - accuracy: 0.7051 - loss: 1.7589 -
Epoch 4/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - accuracy: 0.6936 - loss: 1.7038 -
Epoch 5/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - accuracy: 0.7085 - loss: 1.6292 -
Epoch 6/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step - accuracy: 0.7388 - loss: 1.5911 -
Epoch 7/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - accuracy: 0.7382 - loss: 1.4999 -
Epoch 8/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - accuracy: 0.7651 - loss: 1.4798 -
Epoch 9/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 0.7644 - loss: 1.4070 -
```

```
Epoch 10/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.7694 - loss: 1.3464 -
Epoch 11/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.7153 - loss: 1.3720 -
Epoch 12/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.7501 - loss: 1.2866 -
Epoch 13/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.7415 - loss: 1.2845 -
Epoch 14/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7749 - loss: 1.2227 -
Epoch 15/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7625 - loss: 1.2023 -
Epoch 16/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 0.7657 - loss: 1.2051 -
Epoch 17/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.7925 - loss: 1.0891 -
Epoch 18/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.7870 - loss: 1.0877 -
Epoch 19/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.7871 - loss: 1.1080 -
Epoch 20/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.7894 - loss: 1.0091 -
Epoch 21/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7959 - loss: 1.0219 -
Epoch 22/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7777 - loss: 1.0230 -
Epoch 23/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 0.8096 - loss: 0.9915 -
Epoch 24/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8145 - loss: 0.9585 -
Epoch 25/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8315 - loss: 0.9332 -
Epoch 26/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.8061 - loss: 0.9258 -
Epoch 27/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.8106 - loss: 0.9196 -
Epoch 28/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.7950 - loss: 0.8941 -
Epoch 29/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.8474 - loss: 0.8310 -
Epoch 30/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.8493 - loss: 0.8405 -
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step

--- Dados ROC para copiar ---
FPR = [0.0, 0.0, 0.0, 0.014285714285714285, 0.014285714285714285, 0.0285714
TPR = [0.0, 0.016666666666666666, 0.03333333333333333, 0.05, 0.183333333333
AUC = 0.7254761904761905
--- Fim dos Dados ROC ---

Training - Accuracy: 0.5049, Sensitivity: 0.0000, Specificity: 1.0000, F1:
Threshold: 0.10, Accuracy: 0.5154, Sensitivity: 1.0000, Specificity: 0.1000
Threshold: 0.15, Accuracy: 0.5462, Sensitivity: 1.0000, Specificity: 0.1571
```

```
Threshold: 0.20, Accuracy: 0.5846, Sensitivity: 0.9833, Specificity: 0.2429
Threshold: 0.25, Accuracy: 0.5923, Sensitivity: 0.9000, Specificity: 0.3286
Threshold: 0.30, Accuracy: 0.6231, Sensitivity: 0.8500, Specificity: 0.4286
Threshold: 0.35, Accuracy: 0.6154, Sensitivity: 0.7833, Specificity: 0.4714
Threshold: 0.40, Accuracy: 0.6385, Sensitivity: 0.7167, Specificity: 0.5714
Threshold: 0.45, Accuracy: 0.6462, Sensitivity: 0.5833, Specificity: 0.7000
Threshold: 0.50, Accuracy: 0.6538, Sensitivity: 0.4667, Specificity: 0.8143
Threshold: 0.55, Accuracy: 0.6846, Sensitivity: 0.4667, Specificity: 0.8714
Threshold: 0.60, Accuracy: 0.6692, Sensitivity: 0.4000, Specificity: 0.9000
Threshold: 0.65, Accuracy: 0.6385, Sensitivity: 0.3167, Specificity: 0.9143
Threshold: 0.70, Accuracy: 0.6385, Sensitivity: 0.2833, Specificity: 0.9429
Threshold: 0.75, Accuracy: 0.6154, Sensitivity: 0.2000, Specificity: 0.9714
Threshold: 0.80, Accuracy: 0.5769, Sensitivity: 0.1000, Specificity: 0.9857
Threshold: 0.85, Accuracy: 0.5538, Sensitivity: 0.0500, Specificity: 0.9857
Threshold: 0.90, Accuracy: 0.5385, Sensitivity: 0.0000, Specificity: 1.0000
Threshold: 0.95, Accuracy: 0.5385, Sensitivity: 0.0000, Specificity: 1.0000
Threshold: 1.00, Accuracy: 0.5385, Sensitivity: 0.0000, Specificity: 1.0000
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step

Metrics for chosen threshold 0.45:
Accuracy: 0.6462, Sensitivity: 0.5833, Specificity: 0.7000, F1: 0.6034, ROC
```



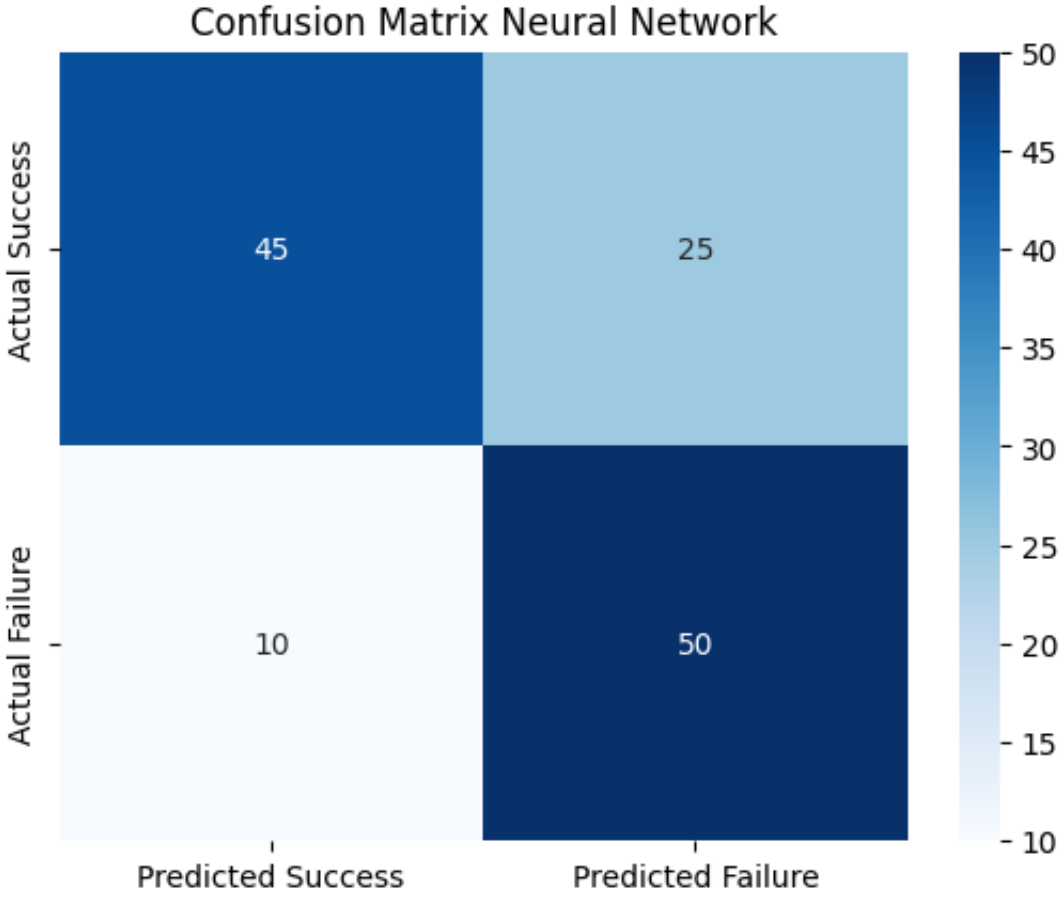


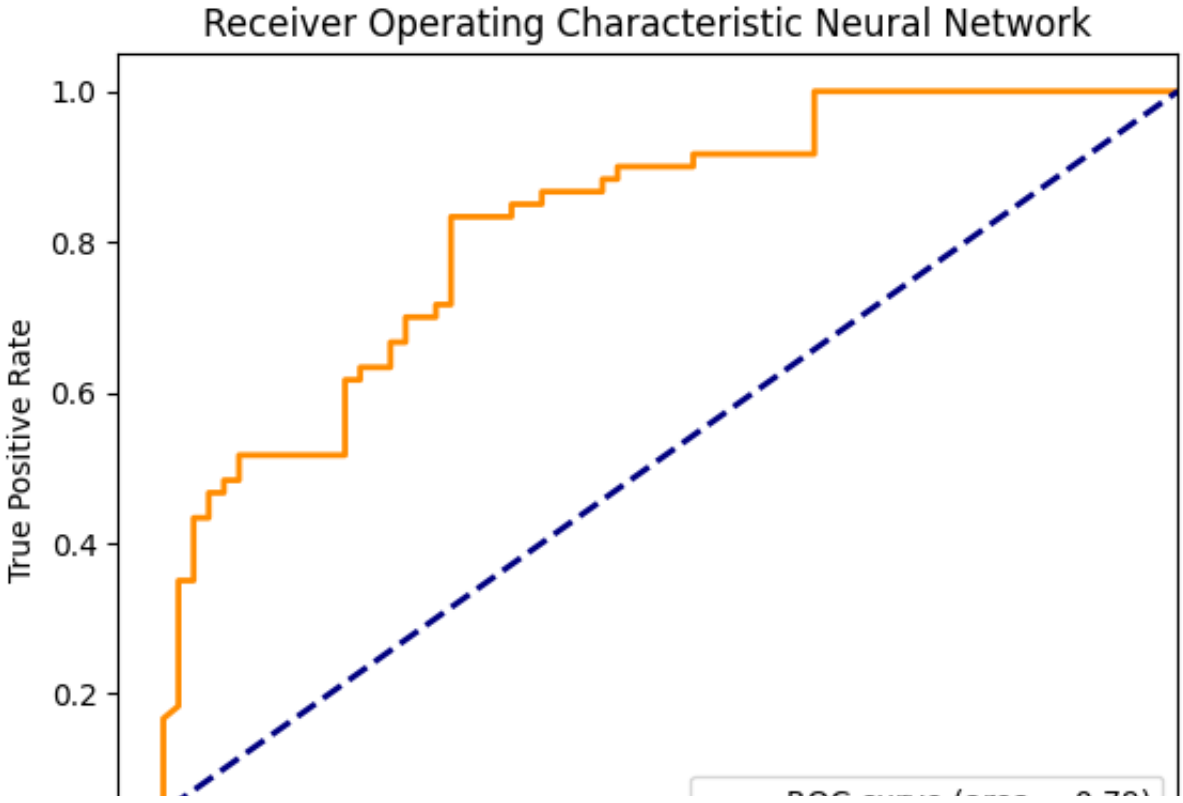


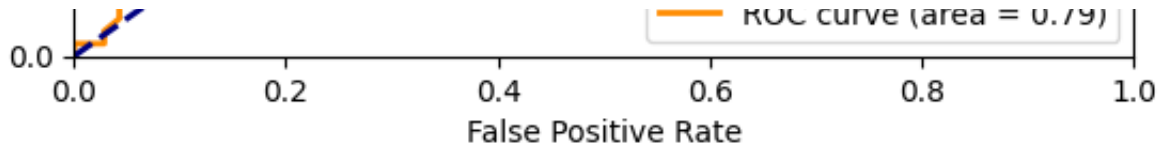

```
Epoch 1/30
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.py:87:
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
13/13 ━━━━━━━━━━━━━━━━━━━━ 3s 37ms/step - accuracy: 0.5119 - loss: 2.1441 -
Epoch 2/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.6365 - loss: 1.8846 -
Epoch 3/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.6955 - loss: 1.7827 -
Epoch 4/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 0.6312 - loss: 1.7600 -
Epoch 5/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7123 - loss: 1.6380 -
Epoch 6/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.6744 - loss: 1.6064 -
Epoch 7/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7121 - loss: 1.5266 -
Epoch 8/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.6523 - loss: 1.5357 -
Epoch 9/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - accuracy: 0.7204 - loss: 1.4699 -
Epoch 10/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - accuracy: 0.7516 - loss: 1.3927 -
Epoch 11/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step - accuracy: 0.7398 - loss: 1.3899 -
Epoch 12/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - accuracy: 0.7524 - loss: 1.3142 -
Epoch 13/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - accuracy: 0.7408 - loss: 1.3336 -
Epoch 14/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - accuracy: 0.7366 - loss: 1.2775 -
Epoch 15/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - accuracy: 0.7621 - loss: 1.2403 -
Epoch 16/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - accuracy: 0.7639 - loss: 1.2047 -
Epoch 17/30
```

```
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step - accuracy: 0.7362 - loss: 1.1868 -
Epoch 18/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - accuracy: 0.6972 - loss: 1.1922 -
Epoch 19/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.7522 - loss: 1.1494 -
Epoch 20/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.7733 - loss: 1.0939 -
Epoch 21/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7748 - loss: 1.0970 -
Epoch 22/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.7701 - loss: 1.0659 -
Epoch 23/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.8240 - loss: 0.9895 -
Epoch 24/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7854 - loss: 1.0312 -
Epoch 25/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.8032 - loss: 1.0126 -
Epoch 26/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 0.8176 - loss: 0.9315 -
Epoch 27/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.8100 - loss: 0.9266 -
Epoch 28/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.7840 - loss: 0.9528 -
Epoch 29/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.7987 - loss: 0.9416 -
Epoch 30/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.7958 - loss: 0.8942 -
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step

--- Dados ROC para copiar ---
FPR = [0.0, 0.0, 0.02857142857142857, 0.02857142857142857, 0.04285714285714
TPR = [0.0, 0.016666666666666666, 0.016666666666666666, 0.03333333333333333
AUC = 0.7926190476190477
--- Fim dos Dados ROC ---

Training - Accuracy: 0.5049, Sensitivity: 0.0000, Specificity: 1.0000, F1:
Threshold: 0.10, Accuracy: 0.4923, Sensitivity: 1.0000, Specificity: 0.0571
Threshold: 0.15, Accuracy: 0.5462, Sensitivity: 1.0000, Specificity: 0.1571
Threshold: 0.20, Accuracy: 0.5615, Sensitivity: 1.0000, Specificity: 0.1857
Threshold: 0.25, Accuracy: 0.6231, Sensitivity: 1.0000, Specificity: 0.3000
Threshold: 0.30, Accuracy: 0.6154, Sensitivity: 0.9167, Specificity: 0.3571
Threshold: 0.35, Accuracy: 0.6692, Sensitivity: 0.9000, Specificity: 0.4714
Threshold: 0.40, Accuracy: 0.6923, Sensitivity: 0.8667, Specificity: 0.5429
Threshold: 0.45, Accuracy: 0.7308, Sensitivity: 0.8333, Specificity: 0.6429
Threshold: 0.50, Accuracy: 0.7231, Sensitivity: 0.7667, Specificity: 0.6857
Threshold: 0.55, Accuracy: 0.7154, Sensitivity: 0.7000, Specificity: 0.7286
Threshold: 0.60, Accuracy: 0.7000, Sensitivity: 0.6167, Specificity: 0.7714
Threshold: 0.65, Accuracy: 0.6615, Sensitivity: 0.5167, Specificity: 0.7857
Threshold: 0.70, Accuracy: 0.7000, Sensitivity: 0.4833, Specificity: 0.8857
Threshold: 0.75, Accuracy: 0.6846, Sensitivity: 0.4000, Specificity: 0.9286
Threshold: 0.80, Accuracy: 0.6308, Sensitivity: 0.2667, Specificity: 0.9429
Threshold: 0.85, Accuracy: 0.5462, Sensitivity: 0.0667, Specificity: 0.9571
Threshold: 0.90, Accuracy: 0.5385, Sensitivity: 0.0167, Specificity: 0.9857
```

```
Threshold: 0.95, Accuracy: 0.5385, Sensitivity: 0.0000, Specificity: 1.0000
Threshold: 1.00, Accuracy: 0.5385, Sensitivity: 0.0000, Specificity: 1.0000
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step

Metrics for chosen threshold 0.45:
Accuracy: 0.7308, Sensitivity: 0.8333, Specificity: 0.6429, F1: 0.7407, ROC
```

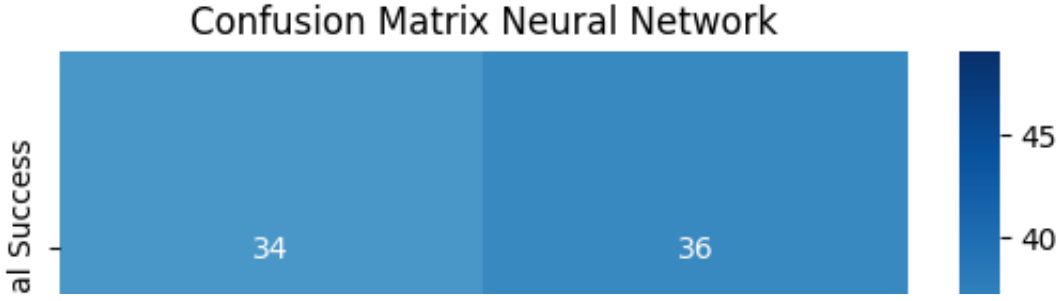


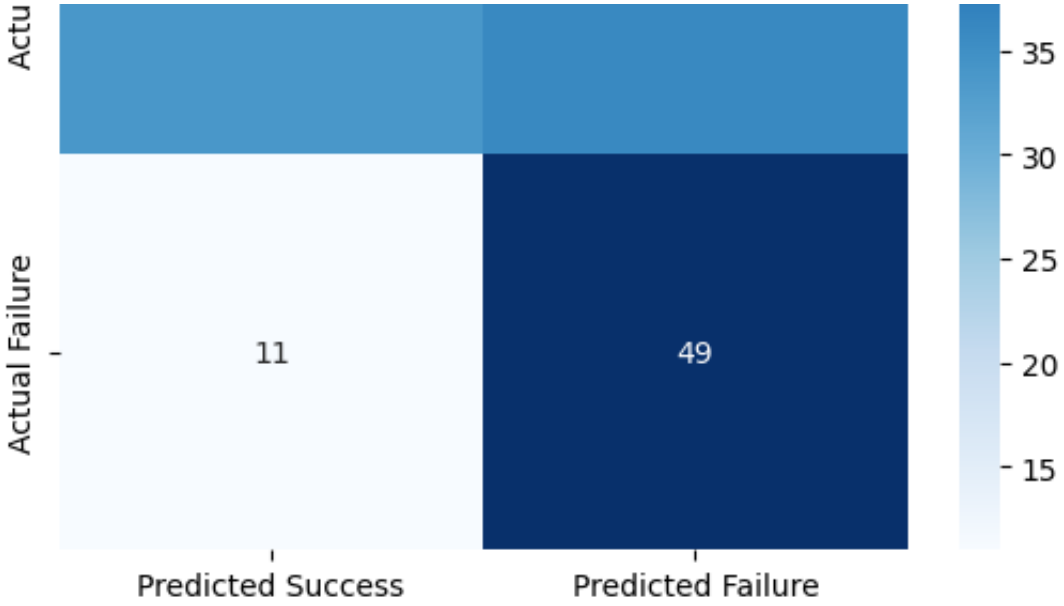


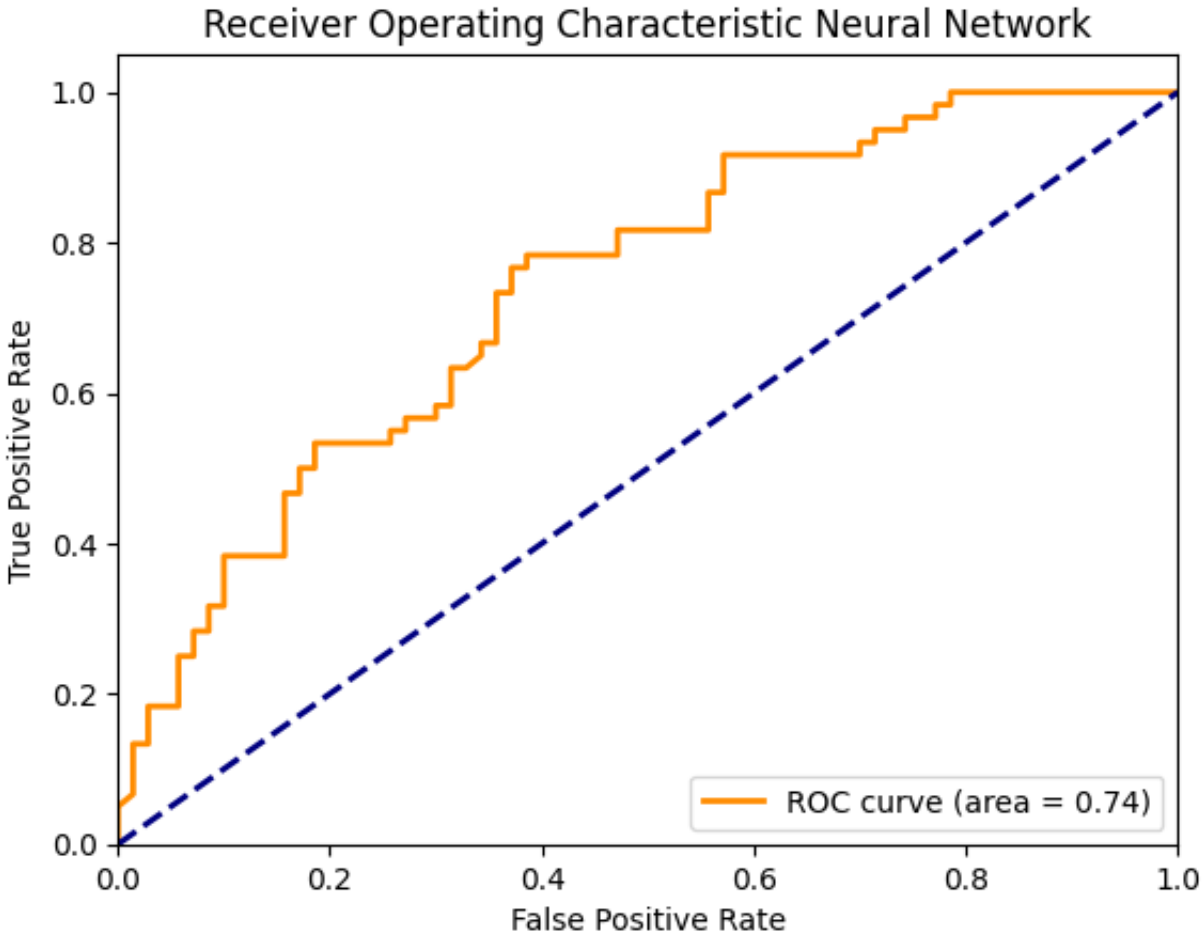



```
Epoch 1/30
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.py:87:
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
13/13 ━━━━━━━━━━━━━━━━━━━━ 3s 32ms/step - accuracy: 0.5682 - loss: 2.1212 -
Epoch 2/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.6392 - loss: 1.8197 -
Epoch 3/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 0.6403 - loss: 1.7688 -
Epoch 4/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.6855 - loss: 1.6640 -
Epoch 5/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.6925 - loss: 1.6139 -
Epoch 6/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.7221 - loss: 1.5381 -
Epoch 7/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7384 - loss: 1.5077 -
Epoch 8/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7065 - loss: 1.5191 -
Epoch 9/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7399 - loss: 1.4094 -
Epoch 10/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7210 - loss: 1.3922 -
Epoch 11/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.7357 - loss: 1.3954 -
Epoch 12/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.7794 - loss: 1.2919 -
Epoch 13/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 0.7450 - loss: 1.2893 -
Epoch 14/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.7822 - loss: 1.2232 -
Epoch 15/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8046 - loss: 1.1764 -
Epoch 16/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7271 - loss: 1.2373 -
Epoch 17/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - accuracy: 0.8297 - loss: 1.1086 -
Epoch 18/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - accuracy: 0.7667 - loss: 1.1388 -
Epoch 19/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 18ms/step - accuracy: 0.7962 - loss: 1.0652 -
Epoch 20/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - accuracy: 0.7900 - loss: 1.0504 -
Epoch 21/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 18ms/step - accuracy: 0.7993 - loss: 1.0368 -
Epoch 22/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step - accuracy: 0.7766 - loss: 1.0408 -
Epoch 23/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step - accuracy: 0.8018 - loss: 0.9764 -
Epoch 24/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step - accuracy: 0.8243 - loss: 0.9573 -
```

```
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step - accuracy: 0.8245 - loss: 0.9373 -
Epoch 25/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step - accuracy: 0.8167 - loss: 0.9637 -
Epoch 26/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 18ms/step - accuracy: 0.8128 - loss: 0.9058 -
Epoch 27/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.8197 - loss: 0.9272 -
Epoch 28/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.8244 - loss: 0.8984 -
Epoch 29/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.8519 - loss: 0.8512 -
Epoch 30/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8293 - loss: 0.8398 -
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step

--- Dados ROC para copiar ---
FPR = [0.0, 0.0, 0.0, 0.014285714285714285, 0.014285714285714285, 0.0285714
TPR = [0.0, 0.016666666666666666, 0.05, 0.06666666666666667, 0.133333333333
AUC = 0.7392857142857143
--- Fim dos Dados ROC ---

Training - Accuracy: 0.5049, Sensitivity: 0.0000, Specificity: 1.0000, F1:
Threshold: 0.10, Accuracy: 0.5000, Sensitivity: 1.0000, Specificity: 0.0714
Threshold: 0.15, Accuracy: 0.5385, Sensitivity: 1.0000, Specificity: 0.1429
Threshold: 0.20, Accuracy: 0.5615, Sensitivity: 1.0000, Specificity: 0.1857
Threshold: 0.25, Accuracy: 0.5846, Sensitivity: 0.9500, Specificity: 0.2714
Threshold: 0.30, Accuracy: 0.5923, Sensitivity: 0.9333, Specificity: 0.3000
Threshold: 0.35, Accuracy: 0.6077, Sensitivity: 0.9167, Specificity: 0.3429
Threshold: 0.40, Accuracy: 0.6154, Sensitivity: 0.8167, Specificity: 0.4429
Threshold: 0.45, Accuracy: 0.6385, Sensitivity: 0.8167, Specificity: 0.4857
Threshold: 0.50, Accuracy: 0.6769, Sensitivity: 0.7833, Specificity: 0.5857
Threshold: 0.55, Accuracy: 0.6615, Sensitivity: 0.6667, Specificity: 0.6571
Threshold: 0.60, Accuracy: 0.6385, Sensitivity: 0.5833, Specificity: 0.6857
Threshold: 0.65, Accuracy: 0.6615, Sensitivity: 0.5333, Specificity: 0.7714
Threshold: 0.70, Accuracy: 0.6615, Sensitivity: 0.4500, Specificity: 0.8429
Threshold: 0.75, Accuracy: 0.6538, Sensitivity: 0.3667, Specificity: 0.9000
Threshold: 0.80, Accuracy: 0.6154, Sensitivity: 0.2333, Specificity: 0.9429
Threshold: 0.85, Accuracy: 0.5846, Sensitivity: 0.1333, Specificity: 0.9714
Threshold: 0.90, Accuracy: 0.5692, Sensitivity: 0.0833, Specificity: 0.9857
Threshold: 0.95, Accuracy: 0.5462, Sensitivity: 0.0167, Specificity: 1.0000
Threshold: 1.00, Accuracy: 0.5385, Sensitivity: 0.0000, Specificity: 1.0000
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step

Metrics for chosen threshold 0.45:
Accuracy: 0.6385, Sensitivity: 0.8167, Specificity: 0.4857, F1: 0.6759, ROC
```



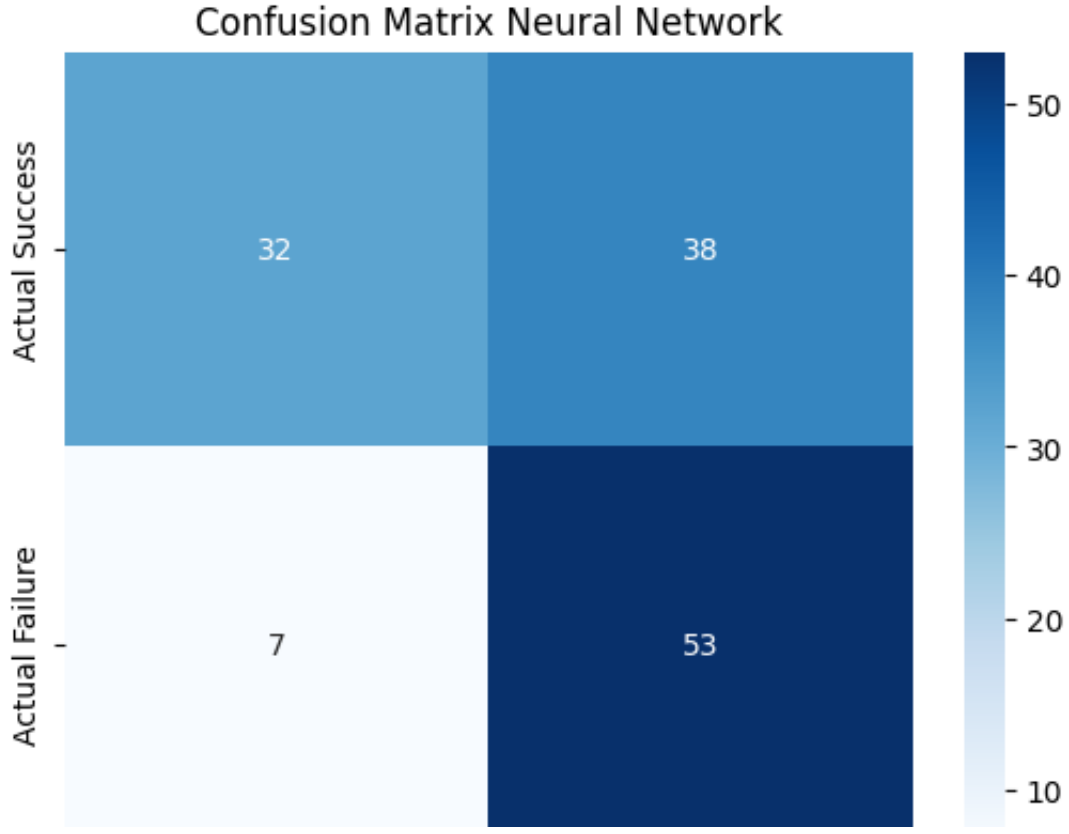


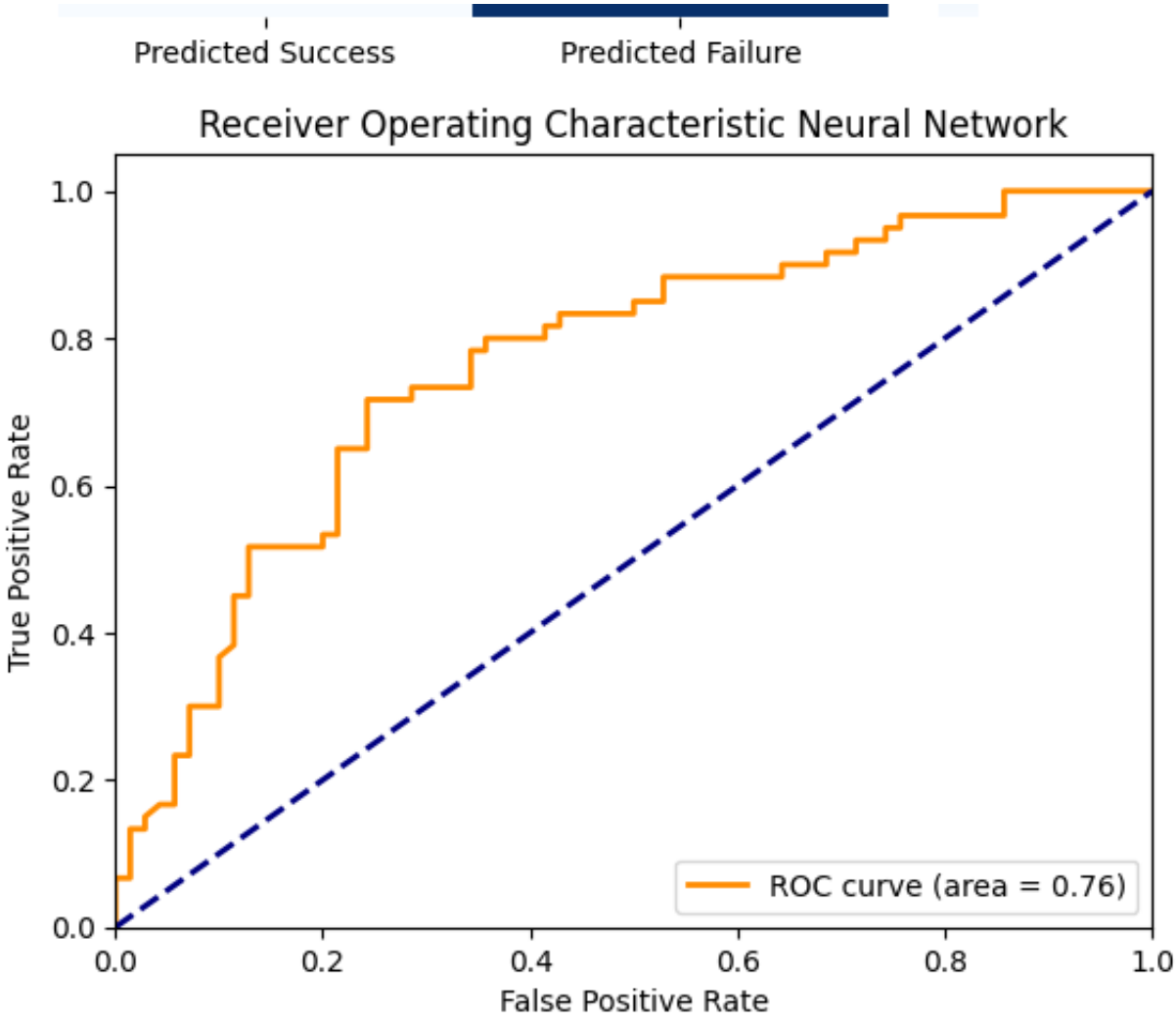



```
Epoch 1/30
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.py:87:
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
13/13 ━━━━━━━━━━━━━━━━━━━━ 3s 33ms/step - accuracy: 0.5418 - loss: 2.0681 -
Epoch 2/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.5981 - loss: 1.9176 -
Epoch 3/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.6430 - loss: 1.7329 -
Epoch 4/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.6618 - loss: 1.6970 -
```

```
Epoch 5/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.6712 - loss: 1.6605 -
Epoch 6/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7124 - loss: 1.5369 -
Epoch 7/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.6727 - loss: 1.5878 -
Epoch 8/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7574 - loss: 1.4031 -
Epoch 9/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 0.7446 - loss: 1.3933 -
Epoch 10/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.6992 - loss: 1.3609 -
Epoch 11/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.7376 - loss: 1.3128 -
Epoch 12/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - accuracy: 0.7924 - loss: 1.2587 -
Epoch 13/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7457 - loss: 1.2373 -
Epoch 14/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.7540 - loss: 1.1990 -
Epoch 15/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.7798 - loss: 1.1675 -
Epoch 16/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.7645 - loss: 1.1218 -
Epoch 17/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7941 - loss: 1.0942 -
Epoch 18/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8207 - loss: 1.0490 -
Epoch 19/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8351 - loss: 1.0368 -
Epoch 20/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8225 - loss: 0.9981 -
Epoch 21/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - accuracy: 0.7919 - loss: 1.0126 -
Epoch 22/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step - accuracy: 0.8204 - loss: 0.9708 -
Epoch 23/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step - accuracy: 0.8341 - loss: 0.9103 -
Epoch 24/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - accuracy: 0.8468 - loss: 0.8920 -
Epoch 25/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - accuracy: 0.8594 - loss: 0.8561 -
Epoch 26/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - accuracy: 0.8617 - loss: 0.8770 -
Epoch 27/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - accuracy: 0.8505 - loss: 0.8414 -
Epoch 28/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step - accuracy: 0.8377 - loss: 0.8215 -
Epoch 29/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step - accuracy: 0.8561 - loss: 0.8078 -
Epoch 30/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - accuracy: 0.8695 - loss: 0.7752 -
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step
```

```
--- Dados ROC para copiar ---
FPR = [0.0, 0.0, 0.0, 0.014285714285714285, 0.014285714285714285, 0.0285714
TPR = [0.0, 0.016666666666666666, 0.06666666666666667, 0.06666666666666667,
AUC = 0.7647619047619048
--- Fim dos Dados ROC ---

Training - Accuracy: 0.5049, Sensitivity: 0.0000, Specificity: 1.0000, F1:
Threshold: 0.10, Accuracy: 0.4846, Sensitivity: 1.0000, Specificity: 0.0429
Threshold: 0.15, Accuracy: 0.5231, Sensitivity: 1.0000, Specificity: 0.1143
Threshold: 0.20, Accuracy: 0.5231, Sensitivity: 1.0000, Specificity: 0.1143
Threshold: 0.25, Accuracy: 0.5385, Sensitivity: 0.9667, Specificity: 0.1714
Threshold: 0.30, Accuracy: 0.5769, Sensitivity: 0.9667, Specificity: 0.2429
Threshold: 0.35, Accuracy: 0.5769, Sensitivity: 0.9167, Specificity: 0.2857
Threshold: 0.40, Accuracy: 0.5923, Sensitivity: 0.9000, Specificity: 0.3286
Threshold: 0.45, Accuracy: 0.6538, Sensitivity: 0.8833, Specificity: 0.4571
Threshold: 0.50, Accuracy: 0.6615, Sensitivity: 0.8500, Specificity: 0.5000
Threshold: 0.55, Accuracy: 0.6923, Sensitivity: 0.8333, Specificity: 0.5714
Threshold: 0.60, Accuracy: 0.7000, Sensitivity: 0.7500, Specificity: 0.6571
Threshold: 0.65, Accuracy: 0.7308, Sensitivity: 0.7167, Specificity: 0.7429
Threshold: 0.70, Accuracy: 0.7154, Sensitivity: 0.6667, Specificity: 0.7571
Threshold: 0.75, Accuracy: 0.7077, Sensitivity: 0.5167, Specificity: 0.8714
Threshold: 0.80, Accuracy: 0.6385, Sensitivity: 0.3333, Specificity: 0.9000
Threshold: 0.85, Accuracy: 0.5846, Sensitivity: 0.1333, Specificity: 0.9714
Threshold: 0.90, Accuracy: 0.5692, Sensitivity: 0.0667, Specificity: 1.0000
Threshold: 0.95, Accuracy: 0.5385, Sensitivity: 0.0000, Specificity: 1.0000
Threshold: 1.00, Accuracy: 0.5385, Sensitivity: 0.0000, Specificity: 1.0000
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step

Metrics for chosen threshold 0.45:
Accuracy: 0.6538, Sensitivity: 0.8833, Specificity: 0.4571, F1: 0.7020, ROC
```

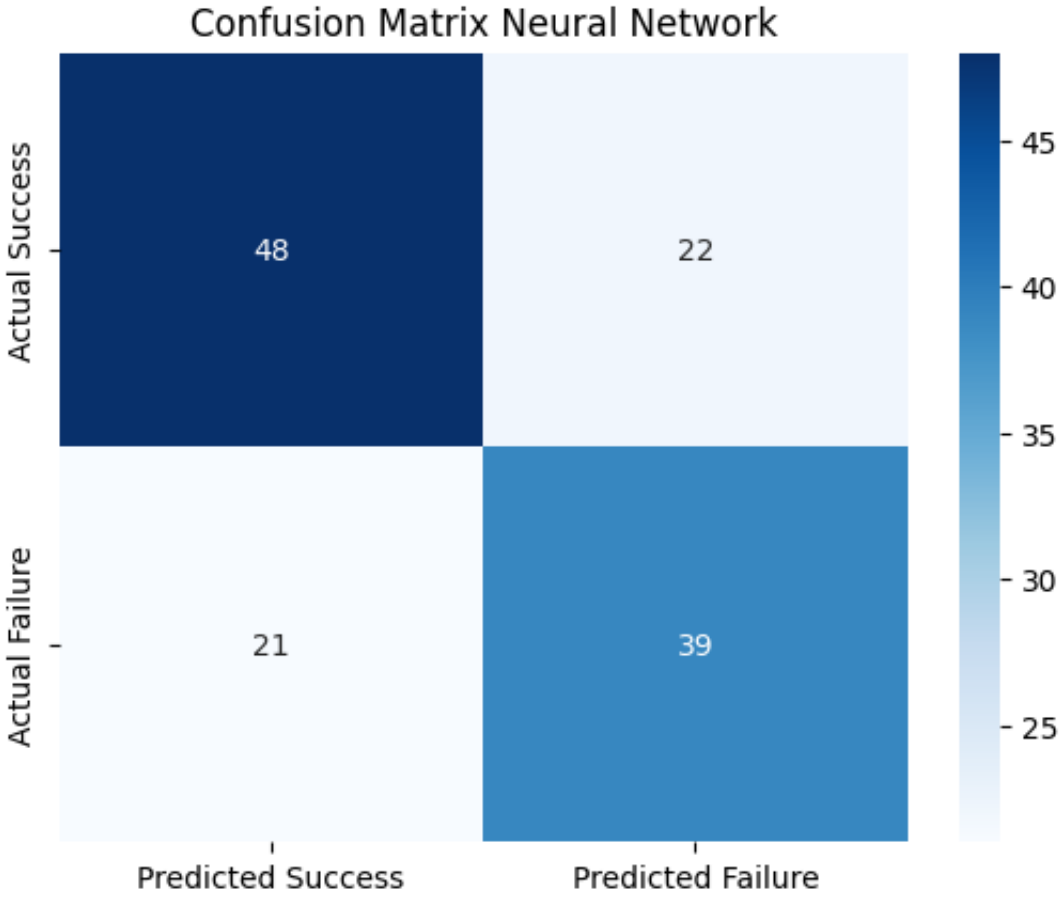


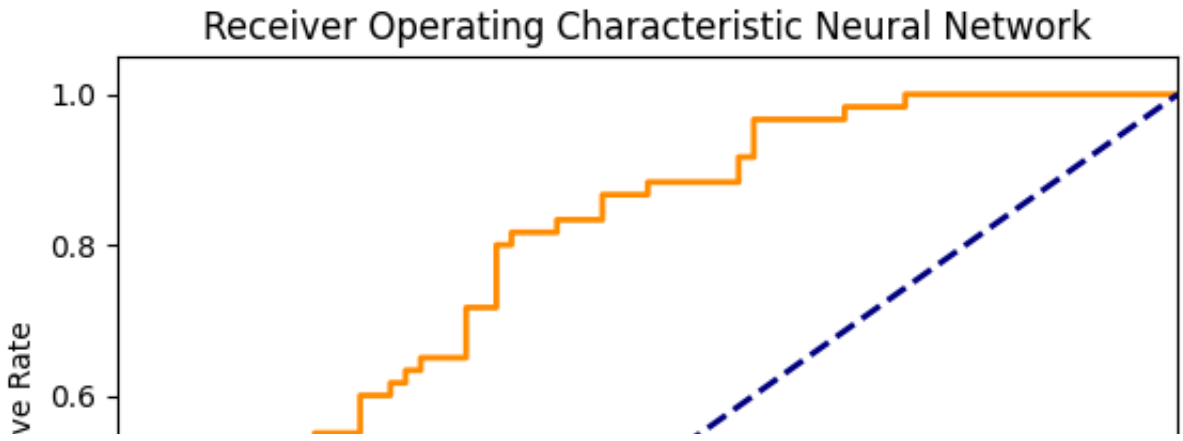


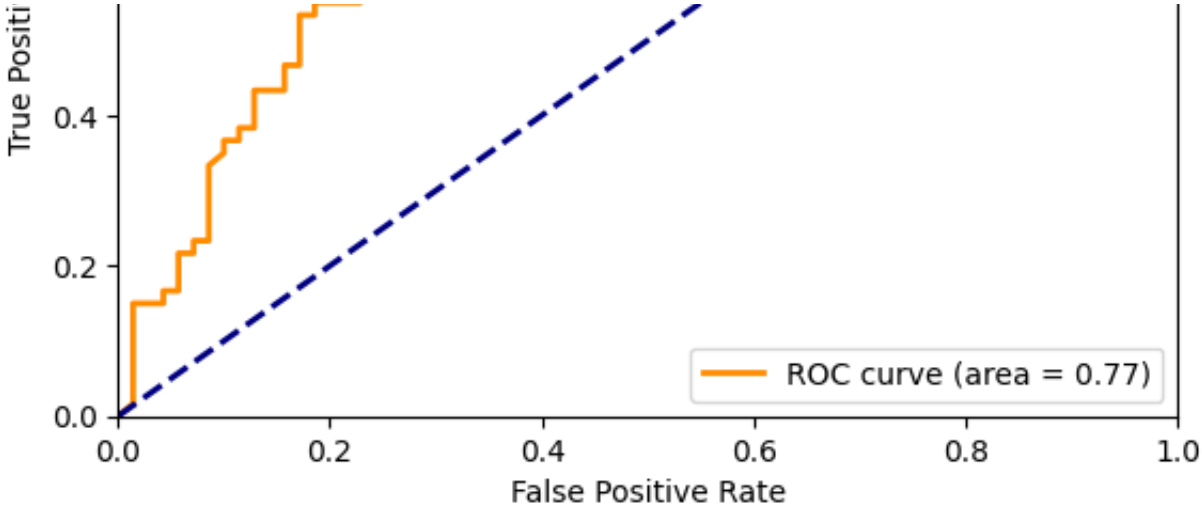

```
Epoch 1/30
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.py:87: 
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
13/13 ━━━━━━━━━━━━━━━━━━━━ 3s 33ms/step - accuracy: 0.4700 - loss: 2.2802 -
Epoch 2/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.6087 - loss: 1.9446 -
Epoch 3/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.6965 - loss: 1.7472 -
Epoch 4/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 0.6885 - loss: 1.7105 -
Epoch 5/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7276 - loss: 1.6210 -
Epoch 6/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.7140 - loss: 1.5967 -
Epoch 7/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7274 - loss: 1.5602 -
Epoch 8/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7250 - loss: 1.5112 -
Epoch 9/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7869 - loss: 1.4561 -
Epoch 10/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 0.7144 - loss: 1.4304 -
Epoch 11/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7321 - loss: 1.3675 -
Epoch 12/30
```

```
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7881 - loss: 1.2989 -
Epoch 13/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 0.7621 - loss: 1.2982 -
Epoch 14/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7592 - loss: 1.3112 -
Epoch 15/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8041 - loss: 1.2182 -
Epoch 16/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7889 - loss: 1.1982 -
Epoch 17/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8172 - loss: 1.1173 -
Epoch 18/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7839 - loss: 1.1571 -
Epoch 19/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8135 - loss: 1.1138 -
Epoch 20/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8049 - loss: 1.1112 -
Epoch 21/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8167 - loss: 1.0430 -
Epoch 22/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8196 - loss: 1.0689 -
Epoch 23/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.8408 - loss: 1.0037 -
Epoch 24/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7800 - loss: 1.0104 -
Epoch 25/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.8473 - loss: 0.9659 -
Epoch 26/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8330 - loss: 0.9497 -
Epoch 27/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.7989 - loss: 0.9561 -
Epoch 28/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8580 - loss: 0.8809 -
Epoch 29/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - accuracy: 0.8166 - loss: 0.9267 -
Epoch 30/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - accuracy: 0.8533 - loss: 0.8662 -
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step

--- Dados ROC para copiar ---
FPR = [0.0, 0.014285714285714285, 0.014285714285714285, 0.04285714285714286
TPR = [0.0, 0.016666666666666666, 0.15, 0.15, 0.16666666666666666, 0.166666
AUC = 0.7666666666666667
--- Fim dos Dados ROC ---

Training - Accuracy: 0.5049, Sensitivity: 0.0000, Specificity: 1.0000, F1:
Threshold: 0.10, Accuracy: 0.5615, Sensitivity: 1.0000, Specificity: 0.1857
Threshold: 0.15, Accuracy: 0.5846, Sensitivity: 1.0000, Specificity: 0.2286
Threshold: 0.20, Accuracy: 0.6077, Sensitivity: 0.9833, Specificity: 0.2857
Threshold: 0.25, Accuracy: 0.6538, Sensitivity: 0.9667, Specificity: 0.3857
Threshold: 0.30, Accuracy: 0.6308, Sensitivity: 0.8833, Specificity: 0.4143
Threshold: 0.35, Accuracy: 0.6846, Sensitivity: 0.8500, Specificity: 0.5429
Threshold: 0.40, Accuracy: 0.7154, Sensitivity: 0.8000, Specificity: 0.6429
```

```
Threshold: 0.45, Accuracy: 0.6692, Sensitivity: 0.6500, Specificity: 0.6857
Threshold: 0.50, Accuracy: 0.6769, Sensitivity: 0.6000, Specificity: 0.7429
Threshold: 0.55, Accuracy: 0.6923, Sensitivity: 0.5500, Specificity: 0.8143
Threshold: 0.60, Accuracy: 0.6615, Sensitivity: 0.4500, Specificity: 0.8429
Threshold: 0.65, Accuracy: 0.6462, Sensitivity: 0.3833, Specificity: 0.8714
Threshold: 0.70, Accuracy: 0.6231, Sensitivity: 0.2833, Specificity: 0.9143
Threshold: 0.75, Accuracy: 0.6000, Sensitivity: 0.2333, Specificity: 0.9143
Threshold: 0.80, Accuracy: 0.5846, Sensitivity: 0.1500, Specificity: 0.9571
Threshold: 0.85, Accuracy: 0.5692, Sensitivity: 0.0833, Specificity: 0.9857
Threshold: 0.90, Accuracy: 0.5385, Sensitivity: 0.0167, Specificity: 0.9857
Threshold: 0.95, Accuracy: 0.5385, Sensitivity: 0.0167, Specificity: 0.9857
Threshold: 1.00, Accuracy: 0.5385, Sensitivity: 0.0000, Specificity: 1.0000
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step

Metrics for chosen threshold 0.45:
Accuracy: 0.6692, Sensitivity: 0.6500, Specificity: 0.6857, F1: 0.6446, ROC
```



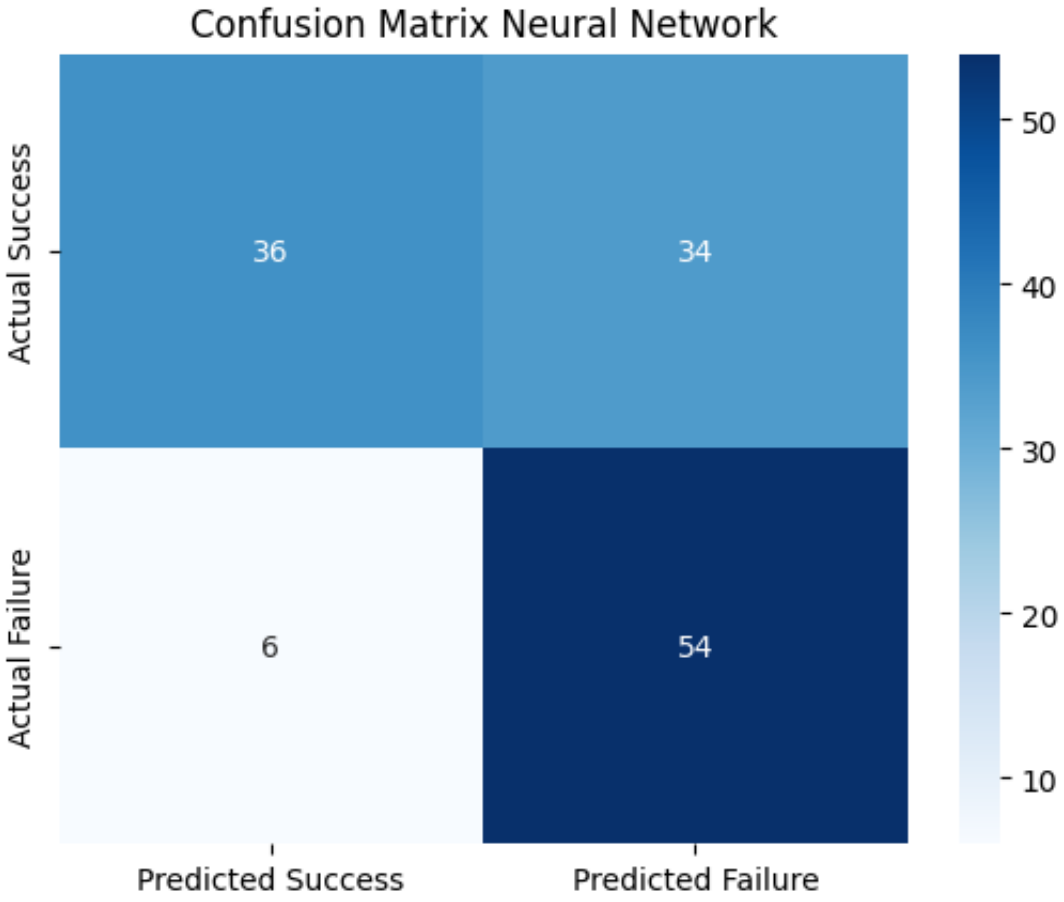


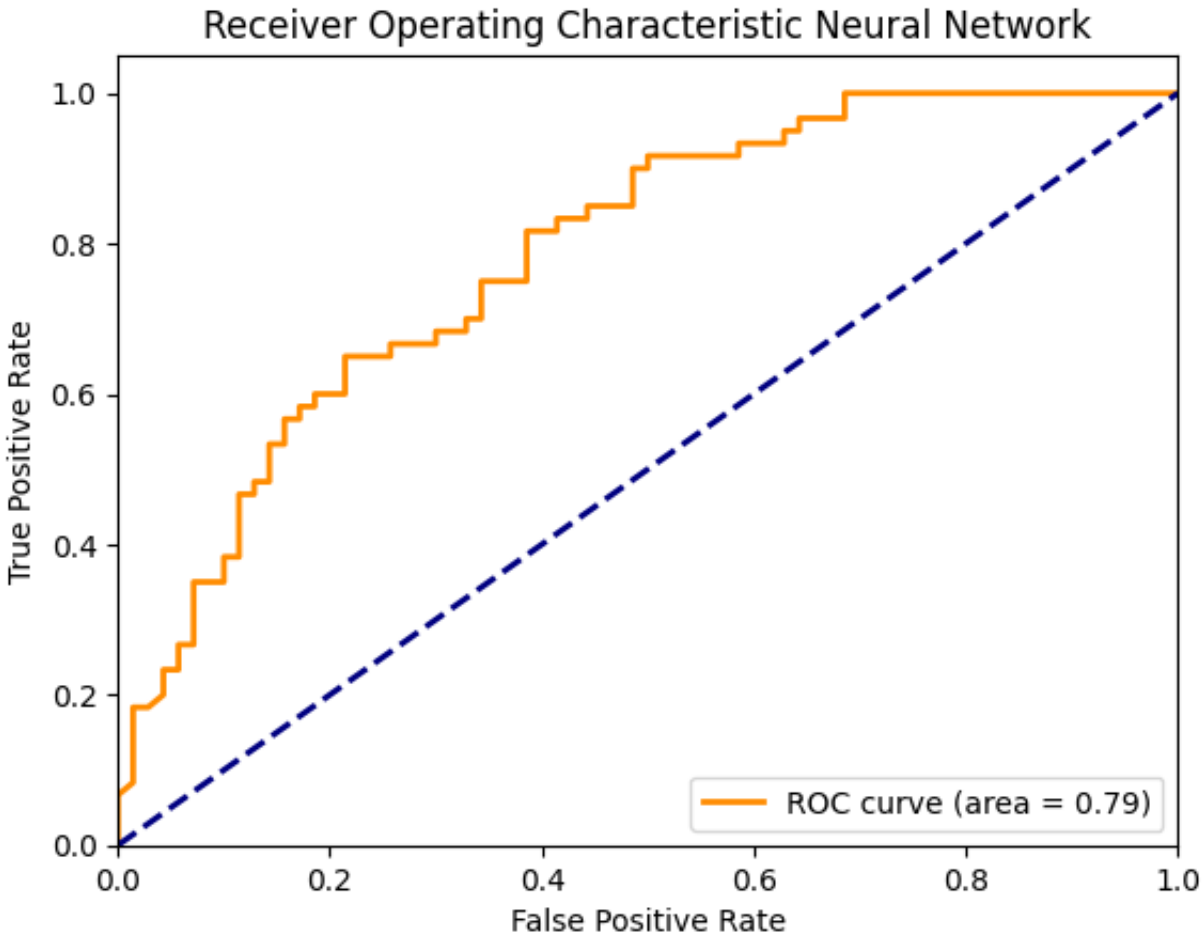



```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.py:87:
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Epoch 1/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 3s 35ms/step - accuracy: 0.5495 - loss: 2.1584 -
Epoch 2/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.6460 - loss: 1.8967 -
Epoch 3/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7103 - loss: 1.7326 -
Epoch 4/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.6574 - loss: 1.7173 -
Epoch 5/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.6823 - loss: 1.6882 -
Epoch 6/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.7146 - loss: 1.5614 -
Epoch 7/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.6954 - loss: 1.5630 -
Epoch 8/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 0.7045 - loss: 1.5072 -
Epoch 9/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.7262 - loss: 1.4283 -
Epoch 10/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 0.7474 - loss: 1.4104 -
Epoch 11/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7616 - loss: 1.3134 -
Epoch 12/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - accuracy: 0.7464 - loss: 1.3337 -
Epoch 13/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.7469 - loss: 1.3068 -
Epoch 14/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7676 - loss: 1.2278 -
Epoch 15/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.7591 - loss: 1.2235 -
Epoch 16/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7655 - loss: 1.1926 -
Epoch 17/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7902 - loss: 1.1295 -
Epoch 18/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 0.7880 - loss: 1.1369 -
Epoch 19/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.7967 - loss: 1.0834
```

```
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.7967 - loss: 1.0834 -
Epoch 20/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7792 - loss: 1.0771 -
Epoch 21/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7903 - loss: 1.0465 -
Epoch 22/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7902 - loss: 1.0368 -
Epoch 23/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.7882 - loss: 1.0198 -
Epoch 24/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.7746 - loss: 0.9969 -
Epoch 25/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8124 - loss: 0.9506 -
Epoch 26/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8005 - loss: 0.9381 -
Epoch 27/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.8151 - loss: 0.9073 -
Epoch 28/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.8180 - loss: 0.8958 -
Epoch 29/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 0.8465 - loss: 0.8803 -
Epoch 30/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.8501 - loss: 0.8481 -
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step

--- Dados ROC para copiar ---
FPR = [0.0, 0.0, 0.0, 0.014285714285714285, 0.014285714285714285, 0.0285714
TPR = [0.0, 0.016666666666666666, 0.06666666666666667, 0.08333333333333333,
AUC = 0.7861904761904762
--- Fim dos Dados ROC ---

Training - Accuracy: 0.5049, Sensitivity: 0.0000, Specificity: 1.0000, F1:
Threshold: 0.10, Accuracy: 0.4846, Sensitivity: 1.0000, Specificity: 0.0429
Threshold: 0.15, Accuracy: 0.5385, Sensitivity: 1.0000, Specificity: 0.1429
Threshold: 0.20, Accuracy: 0.5538, Sensitivity: 1.0000, Specificity: 0.1714
Threshold: 0.25, Accuracy: 0.6308, Sensitivity: 1.0000, Specificity: 0.3143
Threshold: 0.30, Accuracy: 0.6231, Sensitivity: 0.9667, Specificity: 0.3286
Threshold: 0.35, Accuracy: 0.6462, Sensitivity: 0.9167, Specificity: 0.4143
Threshold: 0.40, Accuracy: 0.6769, Sensitivity: 0.9167, Specificity: 0.4714
Threshold: 0.45, Accuracy: 0.6923, Sensitivity: 0.9000, Specificity: 0.5143
Threshold: 0.50, Accuracy: 0.6923, Sensitivity: 0.7833, Specificity: 0.6143
Threshold: 0.55, Accuracy: 0.6769, Sensitivity: 0.7000, Specificity: 0.6571
Threshold: 0.60, Accuracy: 0.7000, Sensitivity: 0.6500, Specificity: 0.7429
Threshold: 0.65, Accuracy: 0.7077, Sensitivity: 0.6000, Specificity: 0.8000
Threshold: 0.70, Accuracy: 0.7000, Sensitivity: 0.5333, Specificity: 0.8429
Threshold: 0.75, Accuracy: 0.6923, Sensitivity: 0.4667, Specificity: 0.8857
Threshold: 0.80, Accuracy: 0.6538, Sensitivity: 0.3833, Specificity: 0.8857
Threshold: 0.85, Accuracy: 0.6231, Sensitivity: 0.2667, Specificity: 0.9286
Threshold: 0.90, Accuracy: 0.6000, Sensitivity: 0.1500, Specificity: 0.9857
Threshold: 0.95, Accuracy: 0.5462, Sensitivity: 0.0167, Specificity: 1.0000
Threshold: 1.00, Accuracy: 0.5385, Sensitivity: 0.0000, Specificity: 1.0000
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step

Metrics for chosen threshold 0.45:
```

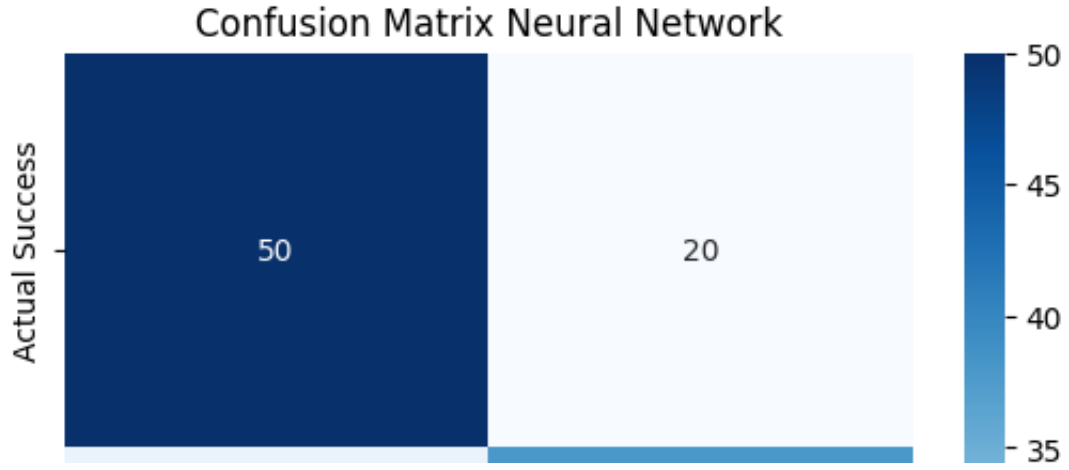
```
Metrics for chosen threshold 0.45:
Accuracy: 0.6923, Sensitivity: 0.9000, Specificity: 0.5143, F1: 0.7297, ROC
```

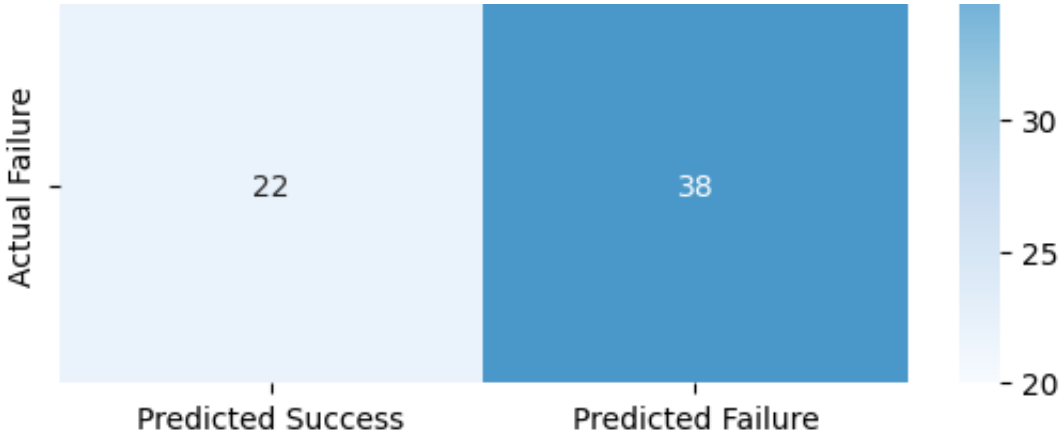


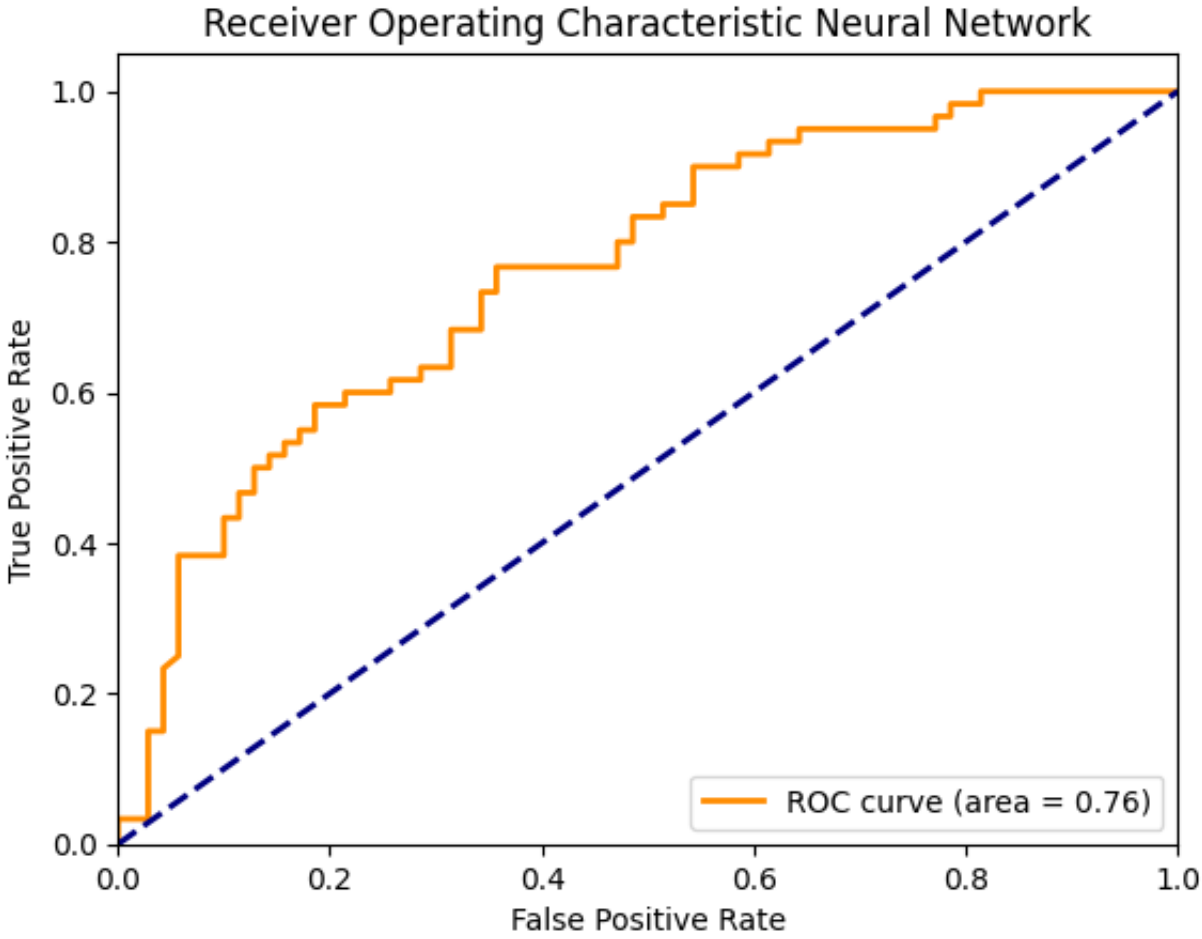


```
Epoch 1/30
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.py:87:
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
13/13 ━━━━━━━━━━━━━━━━━━━━ 3s 32ms/step - accuracy: 0.4735 - loss: 2.1764 -
Epoch 2/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.5942 - loss: 1.9289 -
Epoch 3/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.6256 - loss: 1.7993 -
Epoch 4/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.6417 - loss: 1.7814 -
Epoch 5/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.6731 - loss: 1.6785 -
Epoch 6/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.6792 - loss: 1.6658 -
Epoch 7/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.6836 - loss: 1.5624 -
Epoch 8/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.7113 - loss: 1.5461 -
Epoch 9/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.7099 - loss: 1.4486 -
Epoch 10/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.7511 - loss: 1.3663 -
Epoch 11/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.7295 - loss: 1.3722 -
Epoch 12/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.6892 - loss: 1.4068 -
Epoch 13/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 0.7310 - loss: 1.3329 -
Epoch 14/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.7116 - loss: 1.3756 -
Epoch 15/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7302 - loss: 1.2664 -
Epoch 16/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7321 - loss: 1.2188 -
Epoch 17/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.7518 - loss: 1.1935 -
Epoch 18/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7754 - loss: 1.1605 -
Epoch 19/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8205 - loss: 1.1282 -
Epoch 20/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7537 - loss: 1.1307 -
Epoch 21/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 0.8031 - loss: 1.0750 -
Epoch 22/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.7660 - loss: 1.0997 -
Epoch 23/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8121 - loss: 1.0427 -
Epoch 24/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.7953 - loss: 1.0313 -
Epoch 25/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7761 - loss: 1.0037 -
Epoch 26/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7718 - loss: 1.0332 -
```

```
Epoch 27/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 0.8267 - loss: 0.9694 -
Epoch 28/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8014 - loss: 0.9387 -
Epoch 29/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.7903 - loss: 0.9660 -
Epoch 30/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8299 - loss: 0.9280 -
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step

--- Dados ROC para copiar ---
FPR = [0.0, 0.0, 0.0, 0.02857142857142857, 0.02857142857142857, 0.042857142
TPR = [0.0, 0.016666666666666666, 0.03333333333333333, 0.03333333333333333,
AUC = 0.7622619047619047
--- Fim dos Dados ROC ---

Training - Accuracy: 0.5049, Sensitivity: 0.0000, Specificity: 1.0000, F1:
Threshold: 0.10, Accuracy: 0.5308, Sensitivity: 1.0000, Specificity: 0.1286
Threshold: 0.15, Accuracy: 0.5692, Sensitivity: 0.9667, Specificity: 0.2286
Threshold: 0.20, Accuracy: 0.5769, Sensitivity: 0.9500, Specificity: 0.2571
Threshold: 0.25, Accuracy: 0.6308, Sensitivity: 0.9500, Specificity: 0.3571
Threshold: 0.30, Accuracy: 0.6462, Sensitivity: 0.8500, Specificity: 0.4714
Threshold: 0.35, Accuracy: 0.6692, Sensitivity: 0.7667, Specificity: 0.5857
Threshold: 0.40, Accuracy: 0.6923, Sensitivity: 0.7333, Specificity: 0.6571
Threshold: 0.45, Accuracy: 0.6769, Sensitivity: 0.6333, Specificity: 0.7143
Threshold: 0.50, Accuracy: 0.6769, Sensitivity: 0.6167, Specificity: 0.7286
Threshold: 0.55, Accuracy: 0.7000, Sensitivity: 0.5833, Specificity: 0.8000
Threshold: 0.60, Accuracy: 0.7000, Sensitivity: 0.5333, Specificity: 0.8429
Threshold: 0.65, Accuracy: 0.6923, Sensitivity: 0.4833, Specificity: 0.8714
Threshold: 0.70, Accuracy: 0.6846, Sensitivity: 0.4500, Specificity: 0.8857
Threshold: 0.75, Accuracy: 0.6769, Sensitivity: 0.3667, Specificity: 0.9429
Threshold: 0.80, Accuracy: 0.6231, Sensitivity: 0.2333, Specificity: 0.9571
Threshold: 0.85, Accuracy: 0.6077, Sensitivity: 0.2000, Specificity: 0.9571
Threshold: 0.90, Accuracy: 0.5692, Sensitivity: 0.1000, Specificity: 0.9714
Threshold: 0.95, Accuracy: 0.5462, Sensitivity: 0.0167, Specificity: 1.0000
Threshold: 1.00, Accuracy: 0.5385, Sensitivity: 0.0000, Specificity: 1.0000
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step

Metrics for chosen threshold 0.45:
Accuracy: 0.6769, Sensitivity: 0.6333, Specificity: 0.7143, F1: 0.6441, ROC
```



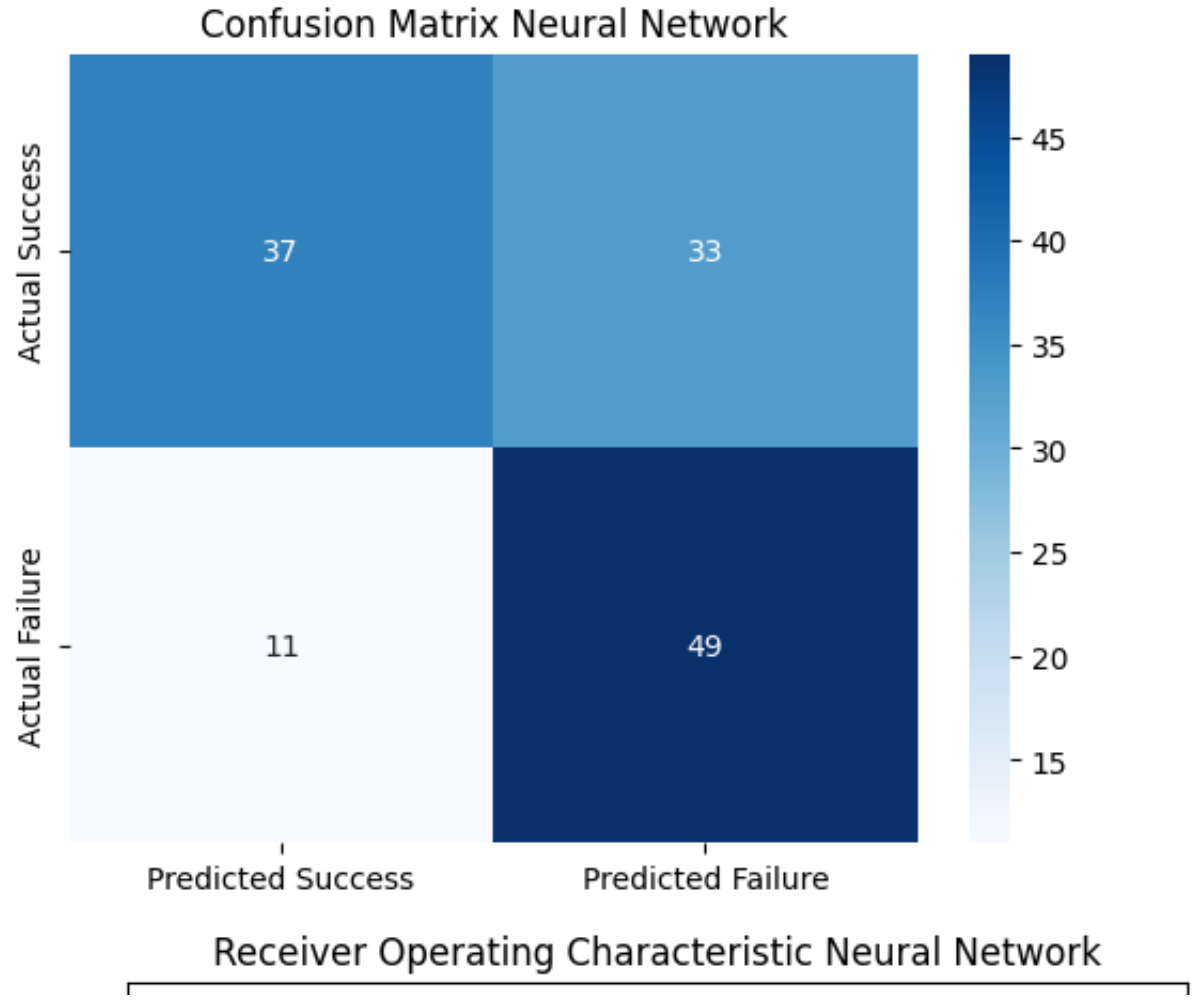


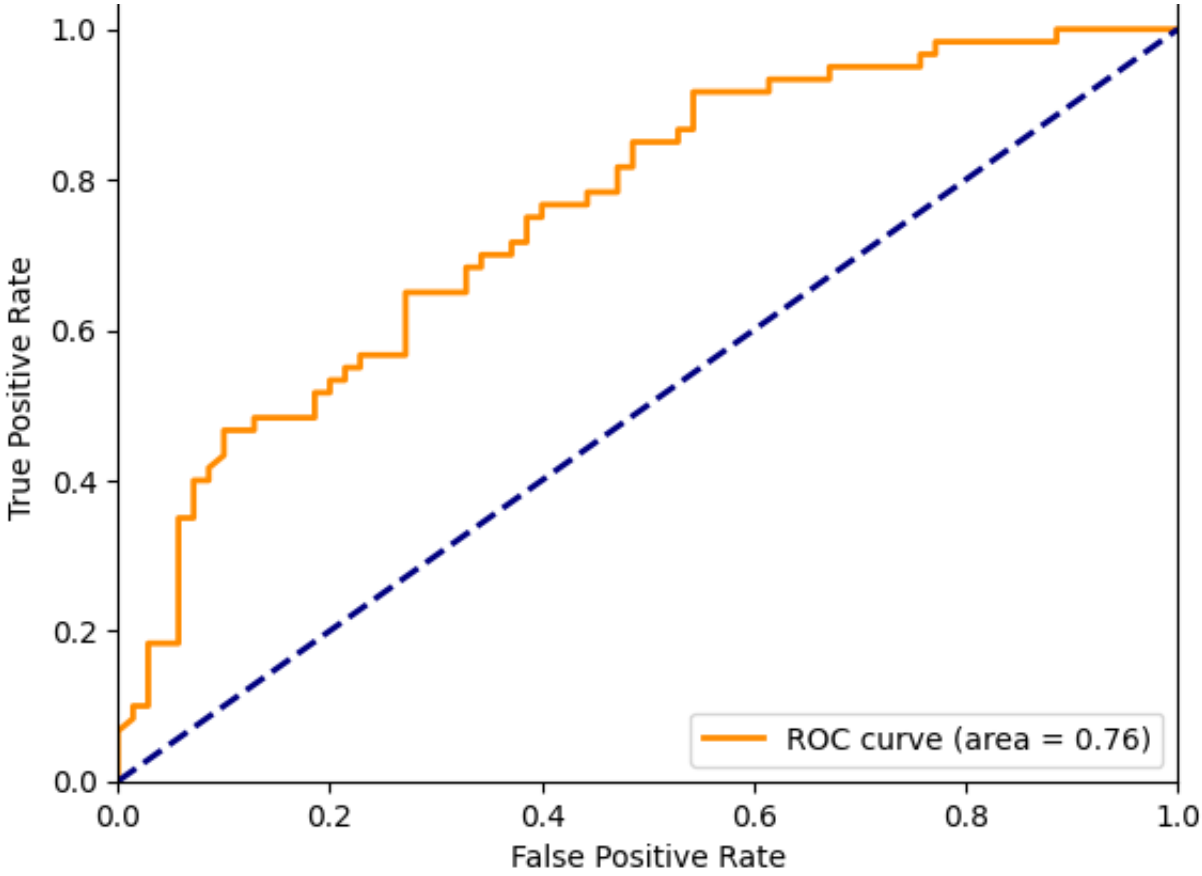



```
Epoch 1/30
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.py:87:
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
13/13 ━━━━━━━━━━━━━━━━━━━━ 4s 31ms/step - accuracy: 0.5876 - loss: 2.0551 -
Epoch 2/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.6123 - loss: 1.9082 -
Epoch 3/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.6976 - loss: 1.7679 -
Epoch 4/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.6721 - loss: 1.7501 -
Epoch 5/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7193 - loss: 1.6352 -
Epoch 6/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.6539 - loss: 1.6049 -
Epoch 7/30
```

```
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7276 - loss: 1.5181 -
Epoch 8/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7282 - loss: 1.4801 -
Epoch 9/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.7468 - loss: 1.3992 -
Epoch 10/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7202 - loss: 1.4187 -
Epoch 11/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.6842 - loss: 1.3812 -
Epoch 12/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7446 - loss: 1.2832 -
Epoch 13/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7596 - loss: 1.2542 -
Epoch 14/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7635 - loss: 1.2453 -
Epoch 15/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.7437 - loss: 1.2473 -
Epoch 16/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7580 - loss: 1.1825 -
Epoch 17/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8022 - loss: 1.1206 -
Epoch 18/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7844 - loss: 1.0930 -
Epoch 19/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.7941 - loss: 1.0979 -
Epoch 20/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8047 - loss: 1.0161 -
Epoch 21/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - accuracy: 0.8118 - loss: 1.0157 -
Epoch 22/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8148 - loss: 0.9764 -
Epoch 23/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 0.8306 - loss: 0.9503 -
Epoch 24/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8521 - loss: 0.9170 -
Epoch 25/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.8003 - loss: 0.9647 -
Epoch 26/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.8008 - loss: 0.9003 -
Epoch 27/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.8337 - loss: 0.9029 -
Epoch 28/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.8373 - loss: 0.8710 -
Epoch 29/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.8039 - loss: 0.8811 -
Epoch 30/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.8626 - loss: 0.7907 -
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step

--- Dados ROC para copiar ---
FPR = [0.0, 0.0, 0.0, 0.014285714285714285, 0.014285714285714285, 0.0285714
TPR = [0.0, 0.016666666666666666, 0.06666666666666667, 0.08333333333333333,
AUC = 0.7569047619047619
```

```
--- Fim dos Dados ROC ---

Training - Accuracy: 0.5049, Sensitivity: 0.0000, Specificity: 1.0000, F1:
Threshold: 0.10, Accuracy: 0.5077, Sensitivity: 1.0000, Specificity: 0.0857
Threshold: 0.15, Accuracy: 0.5231, Sensitivity: 1.0000, Specificity: 0.1143
Threshold: 0.20, Accuracy: 0.5154, Sensitivity: 0.9833, Specificity: 0.1143
Threshold: 0.25, Accuracy: 0.5692, Sensitivity: 0.9833, Specificity: 0.2143
Threshold: 0.30, Accuracy: 0.6077, Sensitivity: 0.9500, Specificity: 0.3143
Threshold: 0.35, Accuracy: 0.6385, Sensitivity: 0.9167, Specificity: 0.4000
Threshold: 0.40, Accuracy: 0.6538, Sensitivity: 0.8667, Specificity: 0.4714
Threshold: 0.45, Accuracy: 0.6615, Sensitivity: 0.8167, Specificity: 0.5286
Threshold: 0.50, Accuracy: 0.6692, Sensitivity: 0.7333, Specificity: 0.6143
Threshold: 0.55, Accuracy: 0.6692, Sensitivity: 0.7000, Specificity: 0.6429
Threshold: 0.60, Accuracy: 0.6615, Sensitivity: 0.5833, Specificity: 0.7286
Threshold: 0.65, Accuracy: 0.6692, Sensitivity: 0.5000, Specificity: 0.8143
Threshold: 0.70, Accuracy: 0.7000, Sensitivity: 0.4667, Specificity: 0.9000
Threshold: 0.75, Accuracy: 0.6462, Sensitivity: 0.3000, Specificity: 0.9429
Threshold: 0.80, Accuracy: 0.5923, Sensitivity: 0.1833, Specificity: 0.9429
Threshold: 0.85, Accuracy: 0.5769, Sensitivity: 0.1167, Specificity: 0.9714
Threshold: 0.90, Accuracy: 0.5462, Sensitivity: 0.0167, Specificity: 1.0000
Threshold: 0.95, Accuracy: 0.5385, Sensitivity: 0.0000, Specificity: 1.0000
Threshold: 1.00, Accuracy: 0.5385, Sensitivity: 0.0000, Specificity: 1.0000
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step

Metrics for chosen threshold 0.45:
Accuracy: 0.6615, Sensitivity: 0.8167, Specificity: 0.5286, F1: 0.6901, ROC
```

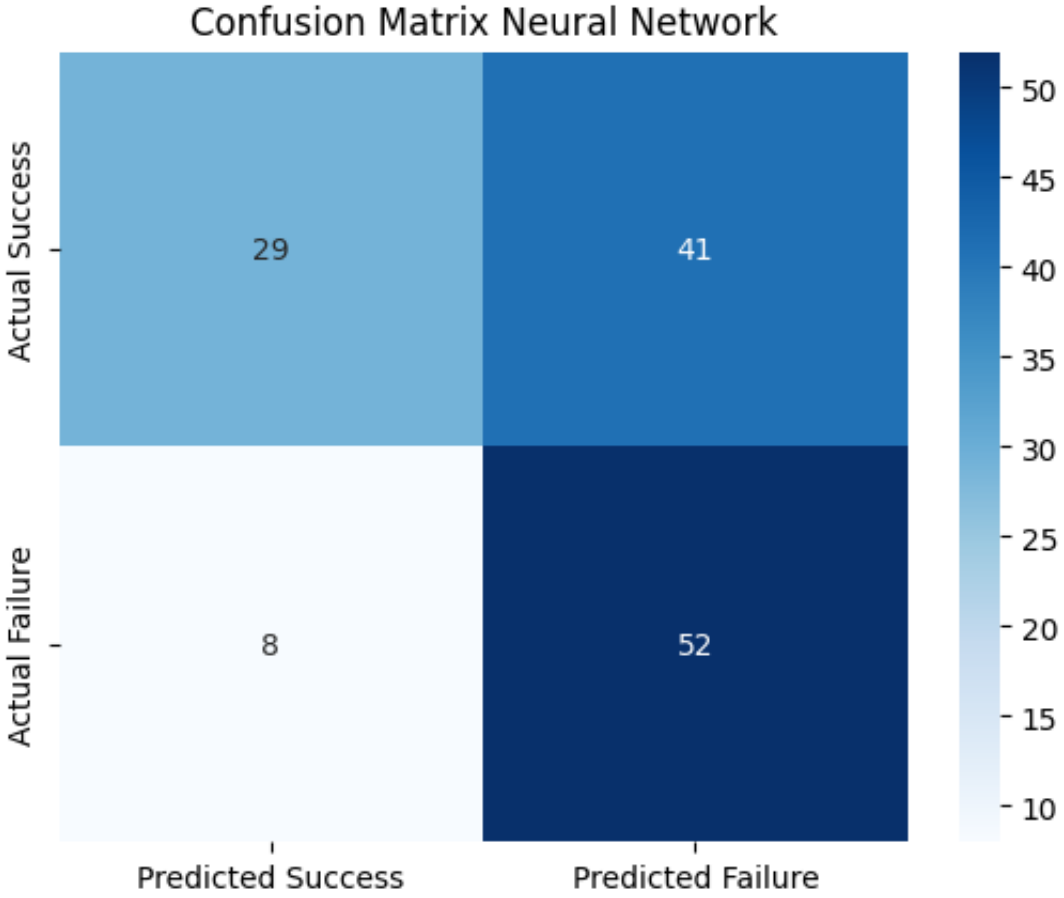


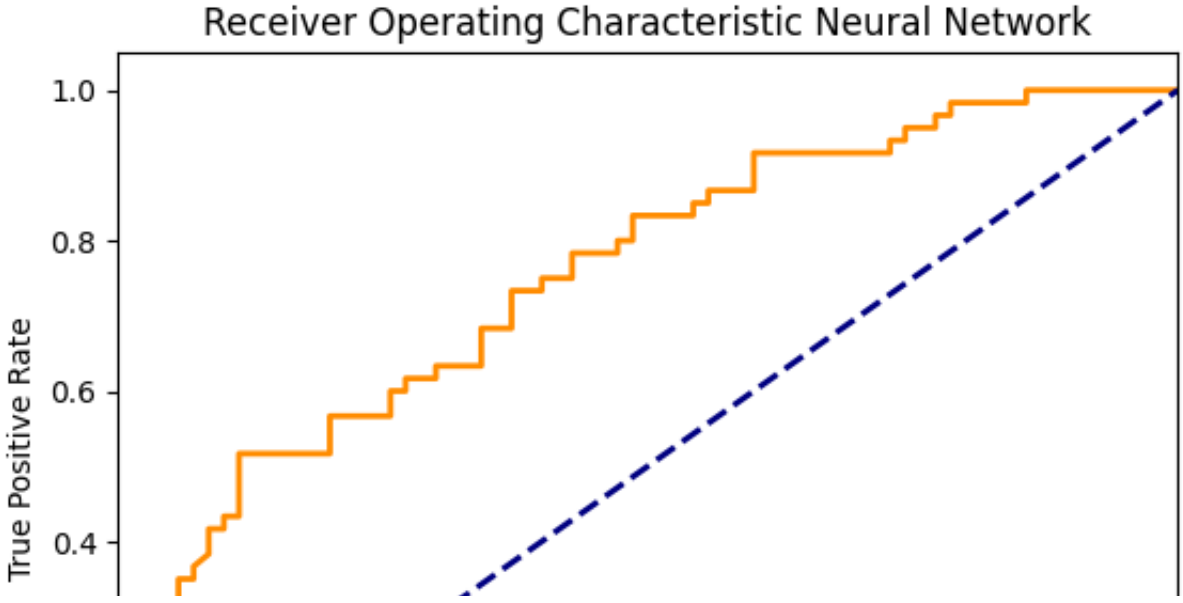


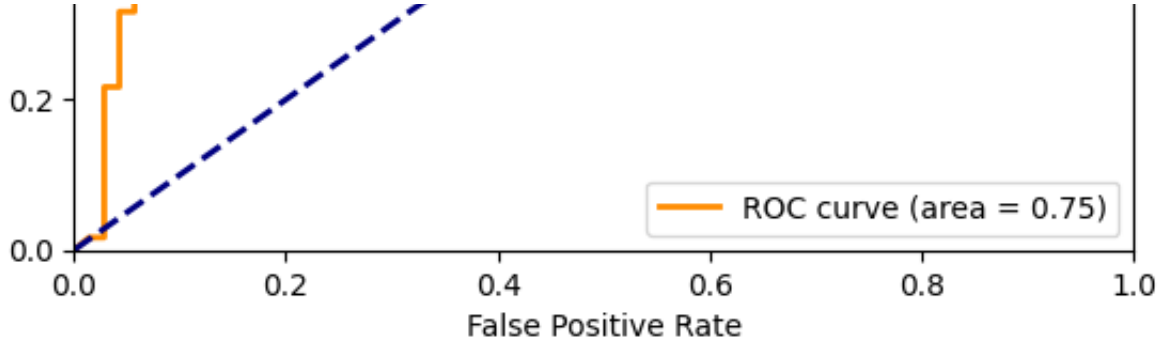

```
Epoch 1/30
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.py:87:
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
13/13 ━━━━━━━━━━━━━━━━━━━━ 4s 34ms/step - accuracy: 0.5820 - loss: 2.0933 -
Epoch 2/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.6300 - loss: 1.9992 -
Epoch 3/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.6705 - loss: 1.8165 -
Epoch 4/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.7124 - loss: 1.7261 -
Epoch 5/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.6767 - loss: 1.7591 -
Epoch 6/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.6989 - loss: 1.6073 -
Epoch 7/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.6664 - loss: 1.6151 -
Epoch 8/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.7359 - loss: 1.4982 -
Epoch 9/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 0.7117 - loss: 1.4637 -
Epoch 10/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - accuracy: 0.6850 - loss: 1.5023 -
Epoch 11/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.7269 - loss: 1.4122 -
Epoch 12/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.6957 - loss: 1.3863 -
Epoch 13/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7583 - loss: 1.2970 -
Epoch 14/30
```

```
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7089 - loss: 1.3246 -
Epoch 15/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7855 - loss: 1.2575 -
Epoch 16/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - accuracy: 0.7254 - loss: 1.2374 -
Epoch 17/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7775 - loss: 1.1986 -
Epoch 18/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7922 - loss: 1.1831 -
Epoch 19/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8048 - loss: 1.0975 -
Epoch 20/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7434 - loss: 1.1419 -
Epoch 21/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.7560 - loss: 1.1205 -
Epoch 22/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7902 - loss: 1.0650 -
Epoch 23/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7794 - loss: 1.0587 -
Epoch 24/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7965 - loss: 1.0136 -
Epoch 25/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.7883 - loss: 1.0029 -
Epoch 26/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.7947 - loss: 0.9764 -
Epoch 27/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8069 - loss: 0.9434 -
Epoch 28/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8072 - loss: 0.9467 -
Epoch 29/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.7772 - loss: 0.9336 -
Epoch 30/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8073 - loss: 0.9251 -
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step

--- Dados ROC para copiar ---
FPR = [0.0, 0.014285714285714285, 0.02857142857142857, 0.02857142857142857,
TPR = [0.0, 0.016666666666666666, 0.016666666666666666, 0.21666666666666667
AUC = 0.7495238095238095
--- Fim dos Dados ROC ---

Training - Accuracy: 0.5049, Sensitivity: 0.0000, Specificity: 1.0000, F1:
Threshold: 0.10, Accuracy: 0.4923, Sensitivity: 1.0000, Specificity: 0.0571
Threshold: 0.15, Accuracy: 0.5385, Sensitivity: 1.0000, Specificity: 0.1429
Threshold: 0.20, Accuracy: 0.5462, Sensitivity: 0.9833, Specificity: 0.1714
Threshold: 0.25, Accuracy: 0.5615, Sensitivity: 0.9833, Specificity: 0.2000
Threshold: 0.30, Accuracy: 0.5615, Sensitivity: 0.9500, Specificity: 0.2286
Threshold: 0.35, Accuracy: 0.5692, Sensitivity: 0.9167, Specificity: 0.2714
Threshold: 0.40, Accuracy: 0.6077, Sensitivity: 0.9167, Specificity: 0.3429
Threshold: 0.45, Accuracy: 0.6231, Sensitivity: 0.8667, Specificity: 0.4143
Threshold: 0.50, Accuracy: 0.6538, Sensitivity: 0.8167, Specificity: 0.5143
Threshold: 0.55, Accuracy: 0.6692, Sensitivity: 0.7333, Specificity: 0.6143
Threshold: 0.60, Accuracy: 0.6462, Sensitivity: 0.6333, Specificity: 0.6571
Threshold: 0.65, Accuracy: 0.6692, Sensitivity: 0.5667, Specificity: 0.7571
```

```
Threshold: 0.65, Accuracy: 0.6692, Sensitivity: 0.5667, Specificity: 0.7571
Threshold: 0.70, Accuracy: 0.6923, Sensitivity: 0.5167, Specificity: 0.8429
Threshold: 0.75, Accuracy: 0.7077, Sensitivity: 0.5000, Specificity: 0.8857
Threshold: 0.80, Accuracy: 0.6615, Sensitivity: 0.3333, Specificity: 0.9429
Threshold: 0.85, Accuracy: 0.5923, Sensitivity: 0.1500, Specificity: 0.9714
Threshold: 0.90, Accuracy: 0.5538, Sensitivity: 0.0667, Specificity: 0.9714
Threshold: 0.95, Accuracy: 0.5385, Sensitivity: 0.0000, Specificity: 1.0000
Threshold: 1.00, Accuracy: 0.5385, Sensitivity: 0.0000, Specificity: 1.0000
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step

Metrics for chosen threshold 0.45:
Accuracy: 0.6231, Sensitivity: 0.8667, Specificity: 0.4143, F1: 0.6797, ROC
```

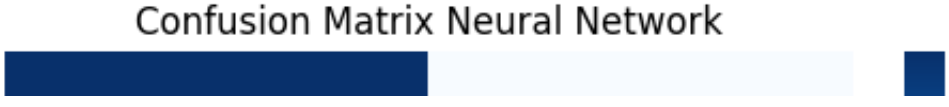


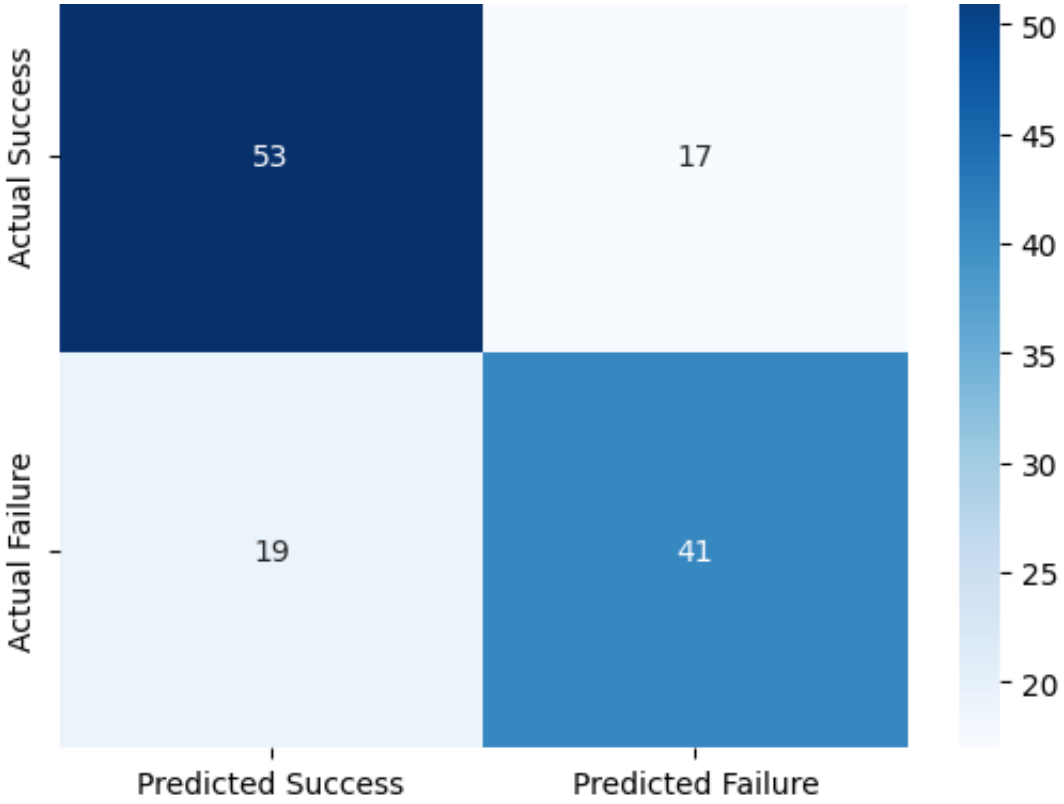


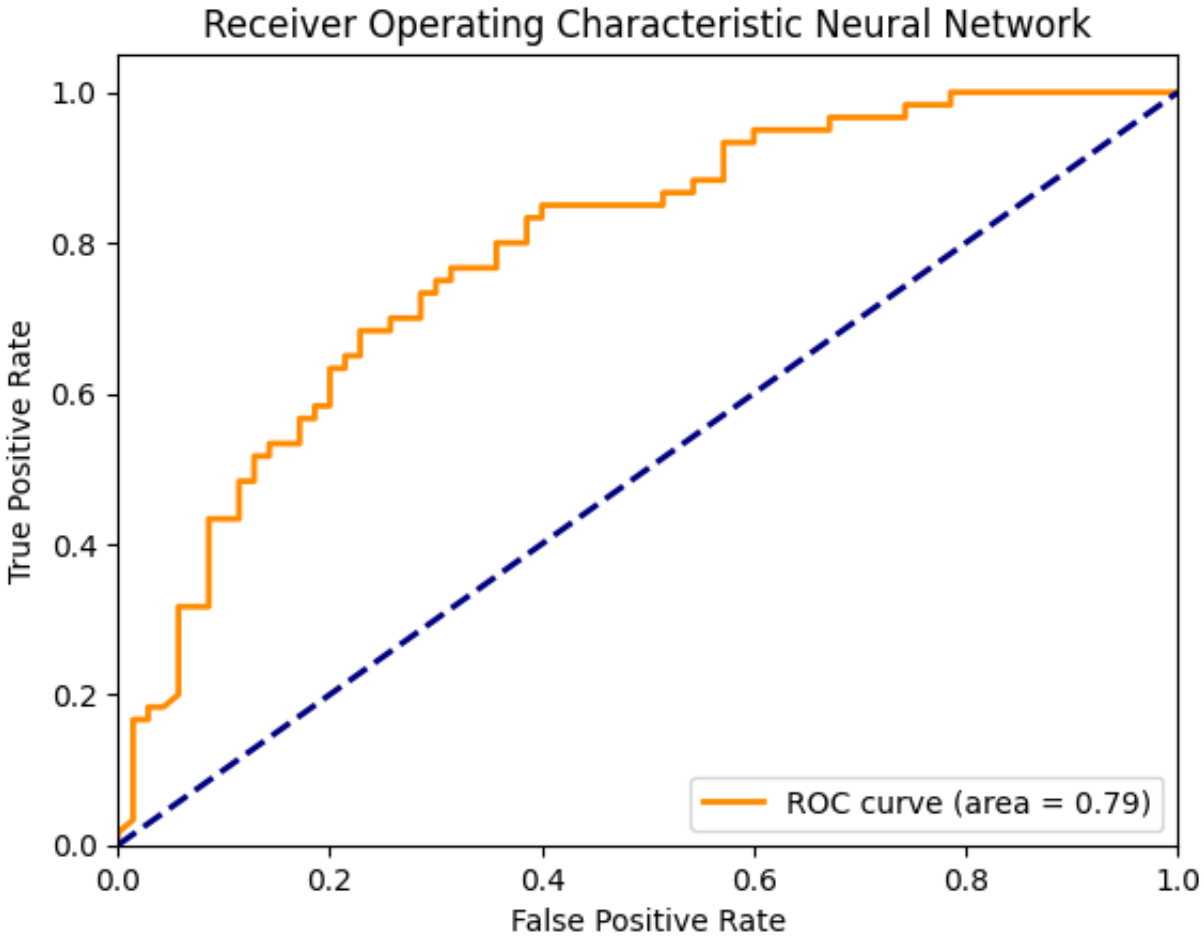



```
Epoch 1/30
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.py:87: 
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
13/13 ━━━━━━━━━━━━━━━━━━━━ 5s 45ms/step - accuracy: 0.4572 - loss: 2.3356 -
Epoch 2/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.6653 - loss: 1.8524 -
Epoch 3/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.6542 - loss: 1.8147 -
Epoch 4/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.6482 - loss: 1.6993 -
Epoch 5/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.7080 - loss: 1.6293 -
Epoch 6/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.6663 - loss: 1.6126 -
Epoch 7/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 0.7615 - loss: 1.5127 -
Epoch 8/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7101 - loss: 1.4593 -
Epoch 9/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.7223 - loss: 1.4421 -
Epoch 10/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7347 - loss: 1.4103 -
Epoch 11/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.7979 - loss: 1.2864 -
Epoch 12/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.7689 - loss: 1.2795 -
Epoch 13/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7712 - loss: 1.2544 -
Epoch 14/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7692 - loss: 1.2603 -
Epoch 15/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7675 - loss: 1.1830 -
Epoch 16/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7699 - loss: 1.2111 -
Epoch 17/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7804 - loss: 1.1415 -
Epoch 18/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.8022 - loss: 1.1195 -
Epoch 19/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8259 - loss: 1.0513 -
Epoch 20/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.8286 - loss: 1.0658 -
Epoch 21/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8131 - loss: 1.0186 -
```

```
Epoch 22/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8056 - loss: 1.0375 -
Epoch 23/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7966 - loss: 0.9970 -
Epoch 24/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8312 - loss: 0.9458 -
Epoch 25/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.8496 - loss: 0.9309 -
Epoch 26/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8151 - loss: 0.9297 -
Epoch 27/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8328 - loss: 0.9132 -
Epoch 28/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - accuracy: 0.8374 - loss: 0.8998 -
Epoch 29/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8369 - loss: 0.8788 -
Epoch 30/30
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - accuracy: 0.8361 - loss: 0.8533 -
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step

--- Dados ROC para copiar ---
FPR = [0.0, 0.0, 0.014285714285714285, 0.014285714285714285, 0.028571428571
TPR = [0.0, 0.016666666666666666, 0.03333333333333333, 0.16666666666666666,
AUC = 0.7892857142857144
--- Fim dos Dados ROC ---

Training - Accuracy: 0.5049, Sensitivity: 0.0000, Specificity: 1.0000, F1:
Threshold: 0.10, Accuracy: 0.5462, Sensitivity: 1.0000, Specificity: 0.1571
Threshold: 0.15, Accuracy: 0.5769, Sensitivity: 0.9833, Specificity: 0.2286
Threshold: 0.20, Accuracy: 0.5846, Sensitivity: 0.9667, Specificity: 0.2571
Threshold: 0.25, Accuracy: 0.6308, Sensitivity: 0.9500, Specificity: 0.3571
Threshold: 0.30, Accuracy: 0.6538, Sensitivity: 0.8833, Specificity: 0.4571
Threshold: 0.35, Accuracy: 0.6923, Sensitivity: 0.8500, Specificity: 0.5571
Threshold: 0.40, Accuracy: 0.7231, Sensitivity: 0.7667, Specificity: 0.6857
Threshold: 0.45, Accuracy: 0.7231, Sensitivity: 0.6833, Specificity: 0.7571
Threshold: 0.50, Accuracy: 0.7231, Sensitivity: 0.6333, Specificity: 0.8000
Threshold: 0.55, Accuracy: 0.6923, Sensitivity: 0.5333, Specificity: 0.8286
Threshold: 0.60, Accuracy: 0.7000, Sensitivity: 0.4833, Specificity: 0.8857
Threshold: 0.65, Accuracy: 0.6769, Sensitivity: 0.4000, Specificity: 0.9143
Threshold: 0.70, Accuracy: 0.6385, Sensitivity: 0.3167, Specificity: 0.9143
Threshold: 0.75, Accuracy: 0.6308, Sensitivity: 0.2667, Specificity: 0.9429
Threshold: 0.80, Accuracy: 0.6000, Sensitivity: 0.1833, Specificity: 0.9571
Threshold: 0.85, Accuracy: 0.6077, Sensitivity: 0.1833, Specificity: 0.9714
Threshold: 0.90, Accuracy: 0.5692, Sensitivity: 0.0833, Specificity: 0.9857
Threshold: 0.95, Accuracy: 0.5385, Sensitivity: 0.0000, Specificity: 1.0000
Threshold: 1.00, Accuracy: 0.5385, Sensitivity: 0.0000, Specificity: 1.0000
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step

Metrics for chosen threshold 0.45:
Accuracy: 0.7231, Sensitivity: 0.6833, Specificity: 0.7571, F1: 0.6949, ROC
```








```
Aggregated Test Set Metrics Across Seeds:
   accuracy  sensitivity  specificity        f1   roc_auc
0  0.646154     0.583333     0.700000  0.603448  0.725476
1  0.730769     0.833333     0.642857  0.740741  0.792619
```

2	0.638462	0.816667	0.485714	0.675862	0.739286
3	0.653846	0.883333	0.457143	0.701987	0.764762
4	0.669231	0.650000	0.685714	0.644628	0.766667
5	0.692308	0.900000	0.514286	0.729730	0.786190
6	0.676923	0.633333	0.714286	0.644068	0.762262
7	0.661538	0.816667	0.528571	0.690141	0.756905
8	0.623077	0.866667	0.414286	0.679739	0.749524
9	0.723077	0.683333	0.757143	0.694915	0.789286

Summary of Test Set Metrics (Mean, Standard Error, 95% Confidence Interval)
Accuracy: Mean = 0.6715, SE = 0.0111, 95% CI = [0.6464, 0.6967]
Sensitivity: Mean = 0.7667, SE = 0.0369, 95% CI = [0.6831, 0.8502]
Specificity: Mean = 0.5900, SE = 0.0389, 95% CI = [0.5020, 0.6780]
F1: Mean = 0.6805, SE = 0.0131, 95% CI = [0.6509, 0.7101]
Roc_auc: Mean = 0.7633, SE = 0.0069, 95% CI = [0.7476, 0.7790]