```
!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 \
    keras==3.5.0 \
    tensorflow==2.17.1
```

```
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: keras==3.5.0 in /usr/local/lib/python3.11/di
Requirement already satisfied: tensorflow==2.17.1 in /usr/local/lib/python3
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: absl-py in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: rich in /usr/local/lib/python3.11/dist-packa
Requirement already satisfied: namex in /usr/local/lib/python3.11/dist-pack
Requirement already satisfied: h5py in /usr/local/lib/python3.11/dist-packa
Requirement already satisfied: optree in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: ml-dtypes in /usr/local/lib/python3.11/dist-
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.
Requirement already satisfied: flatbuffers>=24.3.25 in /usr/local/lib/pytho
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in /usr/
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.1
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python
Requirement already satisfied: setuptools in /usr/local/lib/python3.11/dist
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.11/dis
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.1
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/p
Requirement already satisfied: wrapt>=1.11.0 in /usr/local/lib/python3.11/d
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python
Requirement already satisfied: tensorboard<2.18,>=2.17 in /usr/local/lib/py
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /usr
Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/p
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/di
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.11
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in /us
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.11
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/pyth
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/py
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.11/dist
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.
```

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

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

# Split the dataset into training and testing sets based on a unique identifier
# This ensures that data related to the same 'N PART' is not split across both
unique_n_part = data['N PART'].unique()
train_n_part, test_n_part = train_test_split(unique_n_part, test_size=0.3, ranc

# Filter the original dataset to create training data that includes only the 'N
train_data = data[data['N PART'].isin(train_n_part)]
# Similarly, filter the original dataset to create testing data that includes c
test_data = data[data['N PART'].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(['ANY FAILURE', 'N TEETH', 'N PART'], axis=1)
y_train = train_data['ANY FAILURE'] # Isolate the target variable for the trair

# Separate features and target variable for testing set following the same proc
X_test = test_data.drop(['ANY FAILURE', 'N TEETH', 'N PART'], axis=1)
y_test = test_data['ANY FAILURE'] # Isolate the target variable for the test se


# Impute missing values in 'DMFT' using median
imputer = SimpleImputer(strategy='median')
X_train['DMFT'] = imputer.fit_transform(X_train[['DMFT']])
X_test['DMFT'] = imputer.transform(X_test[['DMFT']])
```

```
y_train.sum()
```

```
46
```

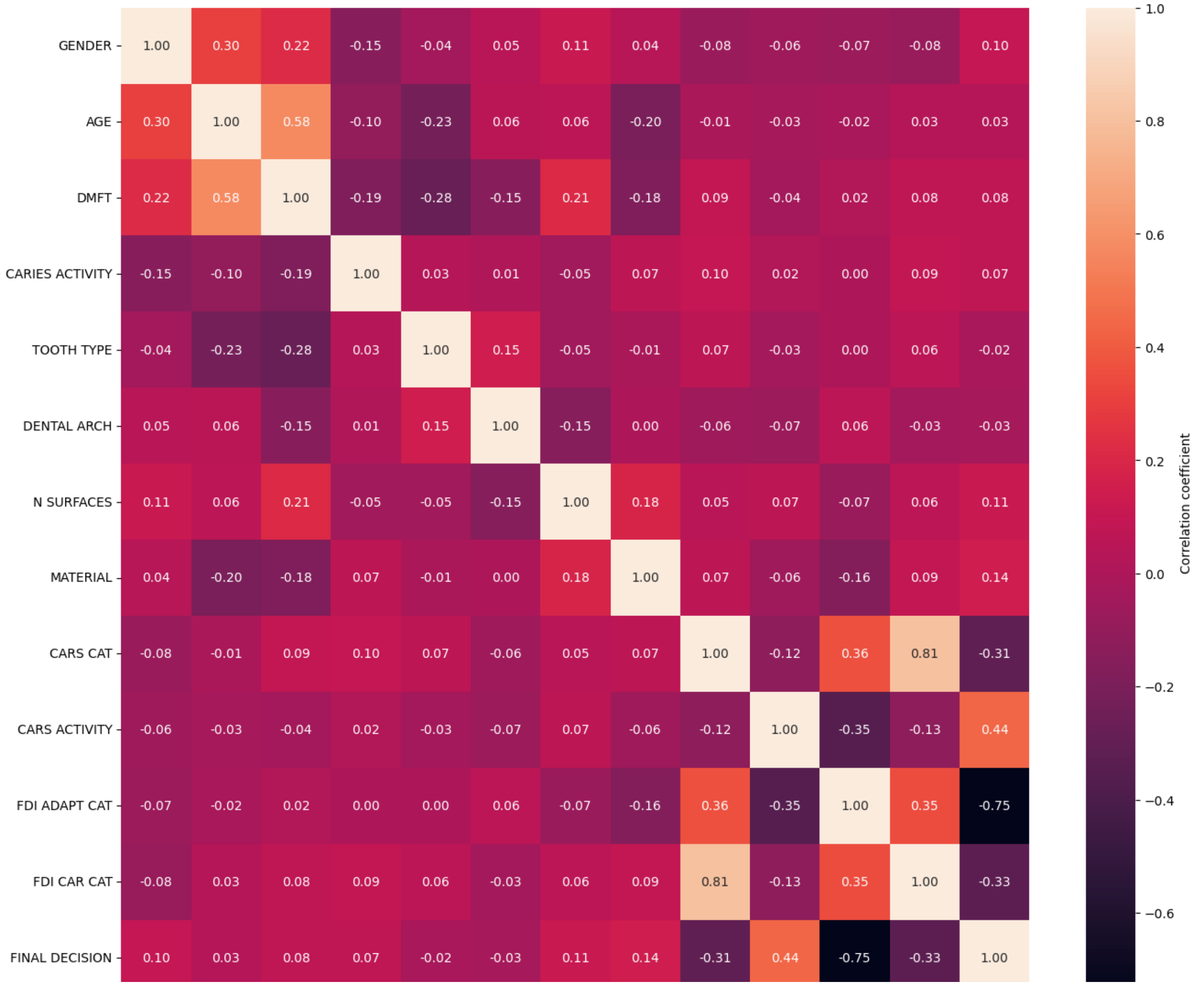
```
y_test.sum()
```

```
16
```

```
len(y_train)+len(y_test)
```

```
501
```

```
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 variab
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.
# `annot=True` displays the correlation coefficients in the heatmap cells.
# `annot_kws={"size": 10}` sets the font size of the annotations to 10 for bett
# `fmt=".2f"` formats the annotation text to show only two decimal places.
# `cbar_kws={'label': 'Correlation coefficient'}` adds a label to the color bar
sns.heatmap(corr_matrix, annot=True, annot_kws={"size": 10}, fmt=".2f", cbar_kw

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



```
import pandas as pd

# Define the lists for each variable type
numeric_vars = ['AGE', 'DMFT']
categorical_vars = ['GENDER', 'CARIES ACTIVITY', 'TOOTH TYPE', 'DENTAL ARCH', '
                    'CARS CAT', 'FDI ADAPT CAT', 'FDI CAR CAT', 'ANY FAILURE', 

def descriptive_statistics(X_train, y_train, X_test, y_test):
    # Merge features and target variable for descriptive statistics on the trai
    train_data_resampled = pd.concat([X_train, y_train], axis=1)

    # Merge features and target variable for descriptive statistics on the test
    test_data = pd.concat([X_test, y_test], axis=1)

    print("Descriptive Statistics for Numeric Variables:")
    print("\nTraining Set:")
    print(train_data_resampled[numeric_vars].describe())
    print("\nTest Set:")
    print(test_data[numeric_vars].describe())

    stats = {}
    for var in categorical_vars:
        stats[var] = {
            "Training Set": {
                "Count": train_data[var].value_counts().to_dict(),
                "Percentage": (train_data[var].value_counts(normalize=True) * 1
            },
            "Test Set": {
                "Count": test_data[var].value_counts().to_dict(),
                "Percentage": (test_data[var].value_counts(normalize=True) * 10
            }
        }

    # Print Categorical Statistics
    for var, data in stats.items():
        print(f"\n{var} Statistics:")
        for dataset, values in data.items():
            print(f"\n{dataset}:")
            for metric, metric_values in values.items():
                print(f"{metric}: {metric_values}")

# Call the function to display descriptive statistics for the train and test se
descriptive_statistics(X_train, y_train, X_test, y_test)
```

```
Descriptive Statistics for Numeric Variables:

Training Set:
              AGE        DMFT
count  353.000000  353.000000
mean    44.413598   12.626062
std     14.723552    5.977382
min     13.000000    1.000000
25%     34.000000    8.000000
50%     46.000000   12.000000
75%     54.000000   18.000000
max     83.000000   25.000000

Test Set:
              AGE        DMFT
count  148.000000  148.000000
mean    44.479730   15.513514
std     15.275461    7.626918
min     14.000000    1.000000
25%     32.250000   10.000000
50%     46.000000   14.000000
75%     58.000000   23.000000
max     81.000000   29.000000

GENDER Statistics:

Training Set:
Count: {1: 248, 0: 105}
Percentage: {1: 70.25495750708215, 0: 29.745042492917843}

Test Set:
```

```
Count: {1: 97, 0: 51}
Percentage: {1: 65.54054054054053, 0: 34.45945945945946}

CARIES ACTIVITY Statistics:

Training Set:
Count: {0: 301, 1: 52}
Percentage: {0: 85.26912181303116, 1: 14.730878186968837}

Test Set:
Count: {0: 114, 1: 34}
Percentage: {0: 77.02702702702703, 1: 22.972972972972975}

TOOTH TYPE Statistics:

Training Set:
Count: {1: 252, 0: 101}
Percentage: {1: 71.38810198300283, 0: 28.611898016997166}

Test Set:
Count: {1: 100, 0: 48}
Percentage: {1: 67.56756756756756, 0: 32.432432432432435}

DENTAL ARCH Statistics:

Training Set:
Count: {1: 177, 0: 176}
Percentage: {1: 50.14164305949008, 0: 49.858356940509914}
```

```
import pandas as pd
import numpy as np
import random
from sklearn.preprocessing import StandardScaler, OrdinalEncoder
from imblearn.over_sampling import SMOTE, SMOTENC

# Set random seeds for reproducibility
np.random.seed(1)
random.seed(1)

# Initialize OrdinalEncoder
ordinal_encoder = OrdinalEncoder()

# Apply Ordinal Encoding to 'N SURFACES' in training data
X_train[['N SURFACES']] = ordinal_encoder.fit_transform(X_train[['N SURFACES']]

# Apply the same ordinal encoding to the test data
X_test[['N SURFACES']] = ordinal_encoder.transform(X_test[['N SURFACES']])

# Convert specified categorical variables in the training data to 'category' dt
categorical_cols = ['FINAL DECISION', 'CARS CAT', 'FDI ADAPT CAT', 'FDI CAR CAT
X_train[categorical_cols] = X_train[categorical_cols].astype('category')
```

```
# Apply one-hot encoding to the specified categorical columns in the training c
one_hot_train = pd.get_dummies(X_train[categorical_cols],
                               prefix=['FINALDECISION', 'CARSCAT', 'FDI ADAPT C

# Concatenate the original training data with the new one-hot encoded columns
X_train = pd.concat([X_train.drop(categorical_cols, axis=1), one_hot_train], ax

# Repeat the process for the test data
X_test[categorical_cols] = X_test[categorical_cols].astype('category')
one_hot_test = pd.get_dummies(X_test[categorical_cols],
                              prefix=['FINALDECISION', 'CARSCAT', 'FDI ADAPT CA

# Ensure the test data has the same one-hot encoded columns as the training dat
for col in one_hot_train.columns:
    if col not in one_hot_test.columns:
        print('!!!')
        one_hot_test[col] = 0
X_test = pd.concat([X_test.drop(categorical_cols, axis=1), one_hot_test], axis=

# Align the columns of the test set to match the training set
X_test = X_test[X_train.columns]

# Define a dictionary to rename the one-hot encoded columns for clarity
column_renaming = {
    'FDI CAR CAT_1': 'FDI No Caries',
    'FDI CAR CAT_2': 'FDI Initial Caries',
    'FDI CAR CAT_3': 'FDI Advanced Caries',
    'FDI ADAPT CAT_1': 'FDI No Adaptation',
    'FDI ADAPT CAT_2': 'FDI Initial Adaptation',
    'FDI ADAPT CAT_3': 'FDI Advanced Adaptation',
    'CARSCAT_0': 'CARS No Caries',
    'CARSCAT_1': 'CARS Initial Caries',
    'CARSCAT_2': 'CARS Advanced Caries',
    'FINALDECISION_0': 'No Initial Intervention',
    'FINALDECISION_1': 'Repaired',
    'FINALDECISION_2': 'Replaced',
    'N SURFACES_0': 'N SURFACES_0',
    'N SURFACES_1': 'N SURFACES_1',
    'N SURFACES_2': 'N SURFACES_2',
    'N SURFACES_3': 'N SURFACES_3',
    'N SURFACES_4': 'N SURFACES_4',
}

# Rename the columns in both datasets
X_train.rename(columns=column_renaming, inplace=True)
X_test.rename(columns=column_renaming, inplace=True)
```

```
# Scale the numerical features
scaler = StandardScaler()
X_train[['AGE', 'DMFT']] = scaler.fit_transform(X_train[['AGE', 'DMFT']])
X_test[['AGE', 'DMFT']] = scaler.transform(X_test[['AGE', 'DMFT']])

# Convert any boolean columns to integers
bool_cols = X_train.select_dtypes(include=['bool']).columns
X_train[bool_cols] = X_train[bool_cols].astype(int)
X_test[bool_cols] = X_test[bool_cols].astype(int)

# Ensure that all data types are consistent
X_train = X_train.astype(float)
X_test = X_test.astype(float)

# Define which columns are categorical (excluding 'AGE' and 'DMFT')
categorical_features = [X_train.columns.get_loc(col) for col in X_train.columns

# Use SMOTENC to balance the train set
smote_nc = SMOTENC(categorical_features=categorical_features, sampling_strategy
                   random_state=42, k_neighbors=5)
X_train_resampled, y_train_resampled = smote_nc.fit_resample(X_train, y_train)

# Adjust 'N SURFACES' back to original range by adding 1
# X_train_resampled['N SURFACES'] = X_train_resampled['N SURFACES'] + 1
# X_test['N SURFACES'] = X_test['N SURFACES'] + 1

# # Adjust 'N SURFACES' back to original range by adding 1
# #X_train_resampled['N SURFACES'] = X_train_resampled['N SURFACES'] + 1
# #X_test['N SURFACES'] = X_test['N SURFACES'] + 1


import pandas as pd

# Define the lists for each variable type
numeric_vars = ['AGE', 'DMFT']
original_categorical_vars = ['GENDER', 'CARIES ACTIVITY', 'TOOTH TYPE', 'DENTAL
                             'ANY FAILURE']
one_hot_encoded_vars = ['N SURFACES_0','N SURFACES_1','N SURFACES_2','N SURFACE
                        'FDI No Caries', 'FDI Initial Caries', 'FDI Advanced Ca
                        'FDI No Adaptation','FDI Initial Adaptation', 'FDI Adva
                        'CARS No Caries', 'CARS Initial Caries', 'CARS Advanced
                        'No Initial Intervention', 'Repaired', 'Replaced']

def descriptive_statistics(X_train_resampled, y_train_resampled, X_test, y_test
    # Merge features and target variable for descriptive statistics on the trai
    train_data_resampled = pd.concat([X_train_resampled, y_train_resampled], ax

    # Merge features and target variable for descriptive statistics on the test
    test_data = pd.concat([X_test, y_test], axis=1)

    print("Descriptive Statistics for Numeric Variables:")
    print("\nResampled Training Set:")
    print(train_data_resampled[numeric_vars].describe())
    print("\nTest Set:")
    print(test_data[numeric_vars].describe())

    stats = {}
    for var in original_categorical_vars:
        stats[var] = {
            "Resampled Training Set": {
                "Count": train_data_resampled[var].value_counts().to_dict(),
                "Percentage": (train_data_resampled[var].value_counts(normalize
            },
            "Test Set": {
                "Count": test_data[var].value_counts().to_dict(),
                "Percentage": (test_data[var].value_counts(normalize=True) * 10
            }
        }

    # Handle one-hot encoded variables
    for var in one_hot_encoded_vars:
        encoded_columns = [col for col in train_data_resampled if col.startswit
        for col in encoded_columns:
            stats[col] = {
                "Resampled Training Set": {
                    "Count": train_data_resampled[col].value_counts().to_dict()
                    "Percentage": (train_data_resampled[col].value_counts(norma
                },
                "Test Set": {
                    "Count": test_data[col].value_counts().to_dict(),
                    "Percentage": (test_data[col].value_counts(normalize=True)
                }
            }

    # Print Categorical Statistics
    for var, data in stats.items():
        print(f"\n{var} Statistics:")
        for dataset, values in data.items():
            print(f"\n{dataset}:")
            for metric, metric_values in values.items():
                print(f"{metric}: {metric_values}")

# Call the function to display descriptive statistics for the resampled train a
descriptive_statistics(X_train_resampled, y_train_resampled, X_test, y_test)
```

```
Descriptive Statistics for Numeric Variables:
```

```
Descriptive Statistics for Numeric Variables:

Resampled Training Set:
              AGE        DMFT
count  614.000000  614.000000
mean    -0.106725    0.015178
std      0.963592    0.947491
min     -2.136590   -1.947770
25%     -0.844308   -0.723452
50%      0.039884    0.031801
75%      0.515988    0.775079
max      2.624446    2.073065

Test Set:
              AGE        DMFT
count  148.000000  148.000000
mean     0.004498    0.483749
std      1.038957    1.277774
min     -2.068575   -1.947770
25%     -0.827305   -0.439957
50%      0.107899    0.230182
75%      0.924076    1.737996
max      2.488417    2.743204

GENDER Statistics:

Resampled Training Set:
Count: {1.0: 468, 0.0: 146}
Percentage: {1.0: 76.2214983713355, 0.0: 23.778501628664493}

Test Set:
Count: {1.0: 97, 0.0: 51}
Percentage: {1.0: 65.54054054054053, 0.0: 34.45945945945946}

CARIES ACTIVITY Statistics:

Resampled Training Set:
Count: {0.0: 562, 1.0: 52}
Percentage: {0.0: 91.53094462540716, 1.0: 8.469055374592834}

Test Set:
Count: {0.0: 114, 1.0: 34}
Percentage: {0.0: 77.02702702702703, 1.0: 22.972972972972975}

TOOTH TYPE Statistics:

Resampled Training Set:
Count: {1.0: 487, 0.0: 127}
Percentage: {1.0: 79.31596091205212, 0.0: 20.684039087947884}

Test Set:
Count: {1.0: 100, 0.0: 48}
Percentage: {1.0: 67.56756756756756, 0.0: 32.432432432432435}
```

```
DENTAL ARCH Statistics:

Resampled Training Set:
Count: {0.0: 321, 1.0: 293}
Percentage: {0.0: 52.28013029315961, 1.0: 47.71986970684039}
```

```
# 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 imblearn.pipeline import Pipeline as IMBPipeline
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, LayerNo
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau, Learni
from tensorflow.keras.regularizers import l2
from scipy import stats


from keras.layers import Layer, Add
import keras
class Residual(Layer):
    def __init__(self):
        super(Residual, self).__init__()
        self.bn_0 = LayerNormalization()
        self.dense_1 = Dense(128, activation='relu', kernel_regularizer=l2(0.01
        self.bn_1 = LayerNormalization()
        self.dropout_1 = Dropout(0.2)
        self.dense_2 = Dense(128, activation='relu', kernel_regularizer=l2(0.01
        self.bn_2 = LayerNormalization()
        self.dropout_2 = Dropout(0.2)
        self.dense_3 = Dense(128, activation='relu', kernel_regularizer=l2(0.01
        self.bn_3 = LayerNormalization()
        self.dropout_3 = Dropout(0.2)
        self.dense_4 = Dense(128, activation='relu', kernel_regularizer=l2(0.01
        self.bn_4 = LayerNormalization()
        self.dropout_4 = Dropout(0.2)
        self.dense_5 = Dense(128, activation='relu', kernel_regularizer=l2(0.01
        self.bn_5 = LayerNormalization()
        self.dropout_5 = Dropout(0.2)
        self.dense_6 = Dense(1, activation='sigmoid')

        # self.dense_1 = Dense(128, activation='relu', kernel_regularizer=l2(0.
        # self.bn_1 = BatchNormalization()
        # self.dense_2 = Dense(64, activation='relu', kernel_regularizer=l2(0.0
        # self.bn_2 = BatchNormalization()
        # self.dense_3 = Dense(1, activation='sigmoid')

    def call(self, x):
        # the residual block using Keras functional API
        x =  self.bn_0(x)
        first_layer = self.dense_1(x)
        x =  self.bn_1(x)
        x =  self.dropout_1(x)
        x =  self.dense_2(x)
        x =  self.bn_2(x)
        x =  self.dropout_2(x)
        x =  self.dense_3(x)
        x =  self.bn_3(x)
        x =  self.dropout_3(x)
        x =  self.dense_4(x)
        x =  self.bn_4(x)
        x =  self.dropout_4(x)
        residual = Add()([x, first_layer])
        x =  self.dense_5(residual)
        x =  self.bn_5(x)
        x =  self.dropout_5(x)
        x = self.dense_6(x)

        # x = self.dense_1(x)
        # x =  self.bn_1(x)
        # x =  self.dense_2(x)
        # x =  self.bn_2(x)
        # x = self.dense_3(x)
        return x
    def compute_output_shape(self, input_shape):
        return input_shape
```

```
pip install focal-loss
```

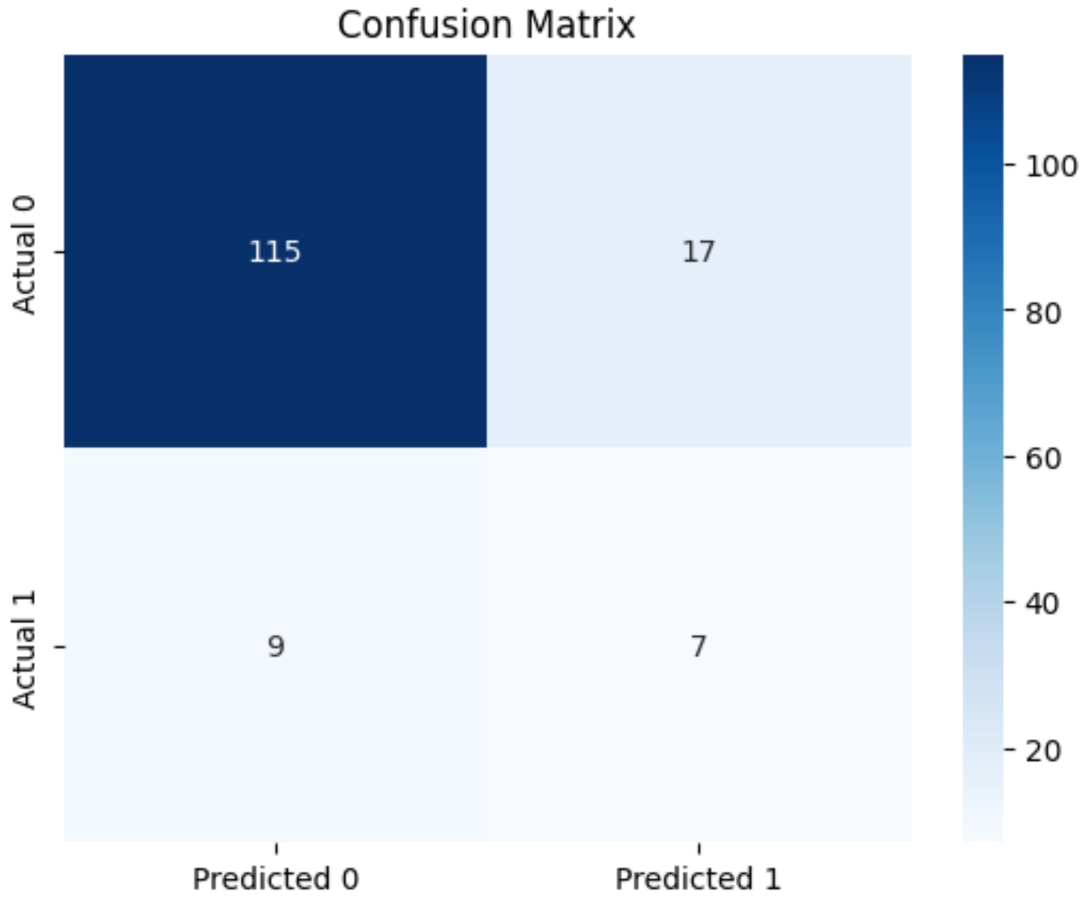
```
Collecting focal-loss
  Downloading focal_loss-0.0.7-py3-none-any.whl.metadata (5.1 kB)
Requirement already satisfied: tensorflow>=2.2 in /usr/local/lib/python3.11
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.11/
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.
Requirement already satisfied: flatbuffers>=24.3.25 in /usr/local/lib/pytho
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in /usr/
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python
Requirement already satisfied: h5py>=3.10.0 in /usr/local/lib/python3.11/di
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.1
Requirement already satisfied: ml-dtypes<0.5.0,>=0.3.1 in /usr/local/lib/py
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python
Requirement already satisfied: setuptools in /usr/local/lib/python3.11/dist
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.11/dis
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.1
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/p
Requirement already satisfied: wrapt>=1.11.0 in /usr/local/lib/python3.11/d
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python
Requirement already satisfied: tensorboard<2.18,>=2.17 in /usr/local/lib/py
Requirement already satisfied: keras>=3.2.0 in /usr/local/lib/python3.11/di
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /usr
Requirement already satisfied: numpy<2.0.0,>=1.23.5 in /usr/local/lib/pytho
Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3
Requirement already satisfied: rich in /usr/local/lib/python3.11/dist-packa
Requirement already satisfied: namex in /usr/local/lib/python3.11/dist-pack
Requirement already satisfied: optree in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/p
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/di
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.11
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in /us
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.11
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/pyth
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/py
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.11/dist
Downloading focal_loss-0.0.7-py3-none-any.whl (19 kB)
Installing collected packages: focal-loss
Successfully installed focal-loss-0.0.7
```

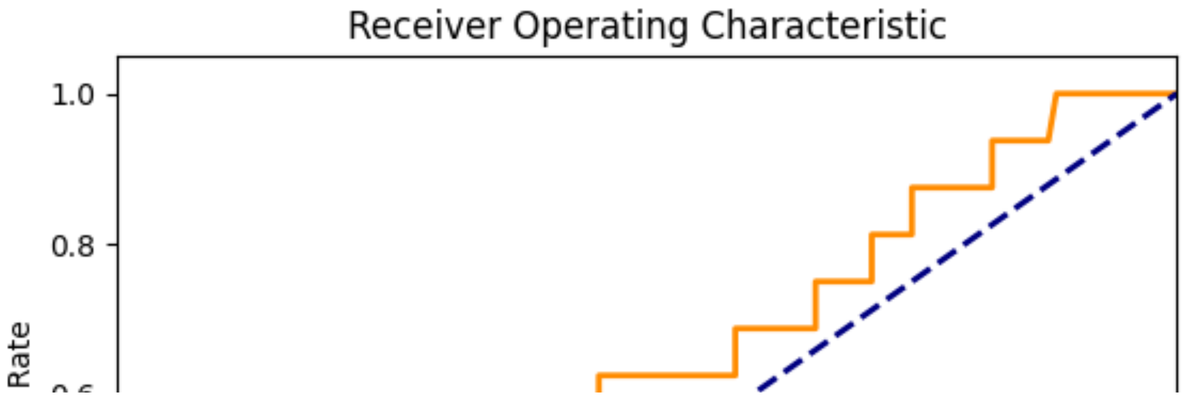
```
from keras import backend as K
from focal_loss import binary_focal_loss
def f1_weighted(true, pred): #shapes (batch, 4)

    #for metrics include these two lines, for loss, don't include them
    #these are meant to round 'pred' to exactly zeros and ones
    #predLabels = K.argmax(pred, axis=-1)
    #pred = K.one_hot(predLabels, 4)

    ground_positives = K.sum(true, axis=0) + K.epsilon()       # = TP + FN
    pred_positives = K.sum(pred, axis=0) + K.epsilon()         # = TP + FP
    true_positives = K.sum(true * pred, axis=0) + K.epsilon()  # = TP
        #all with shape (4,)

    precision = true_positives / pred_positives
    recall = true_positives / ground_positives
        #both = 1 if ground_positives == 0 or pred_positives == 0
        #shape (4,)

    f1 = 2 * (precision * recall) / (precision + recall + K.epsilon())
        #still with shape (4,)

    weighted_f1 = f1 * ground_positives / K.sum(ground_positives)
    weighted_f1 = K.sum(weighted_f1)


    return 1 - weighted_f1 #for metrics, return only 'weighted_f1'

def focal_loss_train(true, pred, pos_weight=.25, gamma=2):
  loss = binary_focal_loss(true, pred, pos_weight=0.5, gamma=2)
  return loss


def evaluate_neural_network(X_train_resampled, y_train_resampled, X_test, y_test
    # Build the model
    residual = Residual()  # Replace with your actual residual block if needed
    model = Sequential()
    model.add(residual)
    # (You can uncomment and modify additional layers if desired)
    # model.add(Dense(128, activation='relu', kernel_regularizer=keras.regulariz
    # model.add(BatchNormalization())
    # model.add(Dropout(0.3))
    # model.add(Dense(64, activation='relu', kernel_regularizer=keras.regularize
    # model.add(BatchNormalization())
    # model.add(Dropout(0.3))
    # model.add(Dense(1, activation='sigmoid'))

    # Compile the model
    opt = keras.optimizers.AdamW(learning_rate=0.005, weight_decay=0.01)
    model.compile(optimizer=opt, loss=focal_loss_train, metrics=['accuracy'])

    # Callbacks
    early_stopping = EarlyStopping(monitor='val_loss', patience=50, restore_best_
    reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.9, patience=5, mi

    # Train the model
    model.fit(X_train_resampled, y_train_resampled,
              validation_split=0.2,
              epochs=500,
              batch_size=64,
              callbacks=[reduce_lr, early_stopping],
              verbose=0)  # Set verbose=1 to see training progress

    # Evaluate on the training set (using the highest threshold in the list for
    y_train_probs = model.predict(X_train_resampled).ravel()
    y_train_pred = (y_train_probs >= threshold_list[-1]).astype(int)
    train_acc   = accuracy_score(y_train_resampled, y_train_pred)
    train_sens  = sensitivity(y_train_resampled, y_train_pred)
    train_spec  = specificity(y_train_resampled, y_train_pred)
    train_f1    = f1_score(y_train_resampled, y_train_pred)
    train_roc_auc = roc_auc_score(y_train_resampled, y_train_probs)

    print(f"Training - Accuracy: {train_acc:.4f}, Sensitivity: {train_sens:.4f},
          f"Specificity: {train_spec:.4f}, F1: {train_f1:.4f}, ROC AUC: {train_r

    # Evaluate on the test set for a range of thresholds
    y_probs = model.predict(X_test).ravel()
    thresholds_metrics = []
    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
        })

    test_roc_auc = roc_auc_score(y_test, y_probs)
    for metrics in thresholds_metrics:
        print(f"Threshold: {metrics['threshold']:.2f}, Accuracy: {metrics['accur
              f"Sensitivity: {metrics['sensitivity']:.4f}, Specificity: {metrics
              f"F1: {metrics['f1_score']:.4f}, ROC AUC: {test_roc_auc:.4f}")
              f' F1: {metrics['f1_score']:.4f}, ROC AUC: {test_roc_auc:.4f}')

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

# ------------------------------------------------------------------------------
# Plotting functions
# ------------------------------------------------------------------------------
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 0', 'Predicted 1'],
                yticklabels=['Actual 0', 'Actual 1'])
    plt.title('Confusion Matrix')
    plt.show()

def plot_roc_curve(y_true, y_probs):
    # Calculate ROC curve metrics
    fpr, tpr, thresholds = roc_curve(y_true, y_probs)
    roc_auc = auc(fpr, tpr)

    # Plot the ROC curve
    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()

    # Output ROC curve metrics
    print("ROC Curve Metrics:")
    print("FPR:", fpr)
    print("TPR:", tpr)
    print("ROC AUC: {:.3f}".format(roc_auc))

    return fpr, tpr, roc_auc

# ------------------------------------------------------------------------------
# Main function that aggregates metrics across seeds
# ------------------------------------------------------------------------------
def main(X_train_resampled, y_train_resampled, X_test, y_test):
    # Define the list of thresholds and chosen threshold for detailed evaluation
    threshold_list = np.arange(0.1, 1.05, 0.05)
    chosen_threshold = 0.45  # You can adjust this as needed

    # List to collect test metrics from each seed iteration
    # List to collect test metrics from each seed iteration
    aggregated_metrics = []
    seeds = range(20, 30)

    for seed_value in seeds:
        print(f"\nRunning evaluation with seed {seed_value}...")
        # Set seeds for reproducibility
        np.random.seed(seed_value)
        random.seed(seed_value)
        tf.random.set_seed(seed_value)

        # Evaluate the model
        model, train_acc, train_sens, train_spec, train_f1, train_roc_auc, thres
            X_train_resampled, y_train_resampled, X_test, y_test, threshold_list
        )

        # Get test set predictions using the chosen threshold
        y_test_probs = model.predict(X_test).ravel()
        y_test_pred = (y_test_probs >= chosen_threshold).astype(int)

        # Calculate test 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)
        test_roc_auc = roc_auc_score(y_test, y_test_probs)

        print(f"\nMetrics for chosen threshold {chosen_threshold}:")
        print(f"Accuracy: {chosen_acc:.4f}, Sensitivity: {chosen_sens:.4f}, "
              f"Specificity: {chosen_spec:.4f}, F1: {chosen_f1:.4f}, ROC AUC: {t

        # Optionally, plot the confusion matrix and ROC curve
        plot_confusion_matrix(y_test, y_test_pred)
        plot_roc_curve(y_test, y_test_probs)

        # Save the test metrics for later 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)

    # ----------------------------------------------------------------------------
    # AGGREGATE RESULTS ACROSS SEEDS
    # ----------------------------------------------------------------------------
    results_df = pd.DataFrame(aggregated_metrics)
```

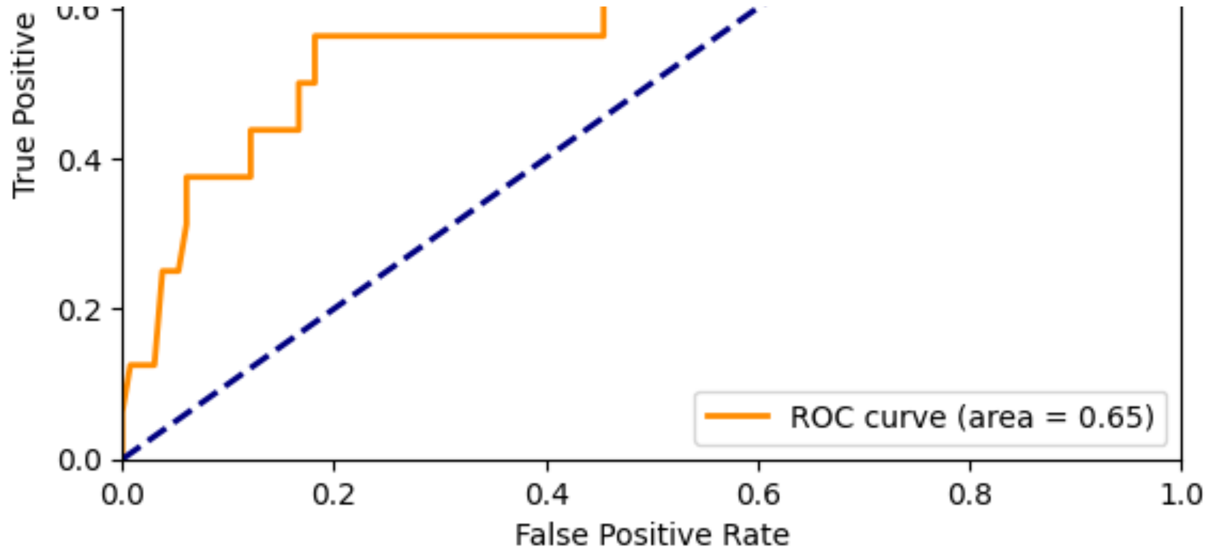
```
    n = len(results_df)
    print("\nAggregated Test Set Metrics Across Seeds:")
    print(results_df)

    # Function to compute mean, standard error, and 95% confidence interval usin
    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)  # 95% CI, two-tailed
        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}, "
              f"SE = {summary['Standard Error']:.3f}, "
              f"95% CI = [{summary['95% CI'][0]:.3f}, {summary['95% CI'][1]:.3f}

# --------------------------------------------------------------------------------
# Run the main function (make sure X_train_resampled, y_train_resampled, X_test,
# --------------------------------------------------------------------------------
if __name__ == '__main__':
    # Replace these with your actual data variables
    main(X_train_resampled, y_train_resampled.astype(float), X_test, y_test.asty
```

```
Running evaluation with seed 20...
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.py:87:
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step
Training - Accuracy: 0.5000, Sensitivity: 0.0000, Specificity: 1.0000, F1:
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
Threshold: 0.10, Accuracy: 0.1081, Sensitivity: 1.0000, Specificity: 0.0000
Threshold: 0.15, Accuracy: 0.3851, Sensitivity: 0.7500, Specificity: 0.3409
Threshold: 0.20, Accuracy: 0.4932, Sensitivity: 0.6250, Specificity: 0.4773
Threshold: 0.25, Accuracy: 0.5743, Sensitivity: 0.5625, Specificity: 0.5758
Threshold: 0.30, Accuracy: 0.6486, Sensitivity: 0.5625, Specificity: 0.6591
Threshold: 0.35, Accuracy: 0.7095, Sensitivity: 0.5625, Specificity: 0.7273
```

```
Threshold: 0.40, Accuracy: 0.7838, Sensitivity: 0.5000, Specificity: 0.8182
Threshold: 0.45, Accuracy: 0.8243, Sensitivity: 0.4375, Specificity: 0.8712
Threshold: 0.50, Accuracy: 0.8514, Sensitivity: 0.3750, Specificity: 0.9091
Threshold: 0.55, Accuracy: 0.8649, Sensitivity: 0.3750, Specificity: 0.9242
Threshold: 0.60, Accuracy: 0.8716, Sensitivity: 0.2500, Specificity: 0.9470
Threshold: 0.65, Accuracy: 0.8851, Sensitivity: 0.1250, Specificity: 0.9773
Threshold: 0.70, Accuracy: 0.8919, Sensitivity: 0.1250, Specificity: 0.9848
Threshold: 0.75, Accuracy: 0.8919, Sensitivity: 0.0000, Specificity: 1.0000
Threshold: 0.80, Accuracy: 0.8919, Sensitivity: 0.0000, Specificity: 1.0000
Threshold: 0.85, Accuracy: 0.8919, Sensitivity: 0.0000, Specificity: 1.0000
Threshold: 0.90, Accuracy: 0.8919, Sensitivity: 0.0000, Specificity: 1.0000
Threshold: 0.95, Accuracy: 0.8919, Sensitivity: 0.0000, Specificity: 1.0000
Threshold: 1.00, Accuracy: 0.8919, Sensitivity: 0.0000, Specificity: 1.0000
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step

Metrics for chosen threshold 0.45:
Accuracy: 0.8243, Sensitivity: 0.4375, Specificity: 0.8712, F1: 0.3500, ROC
```

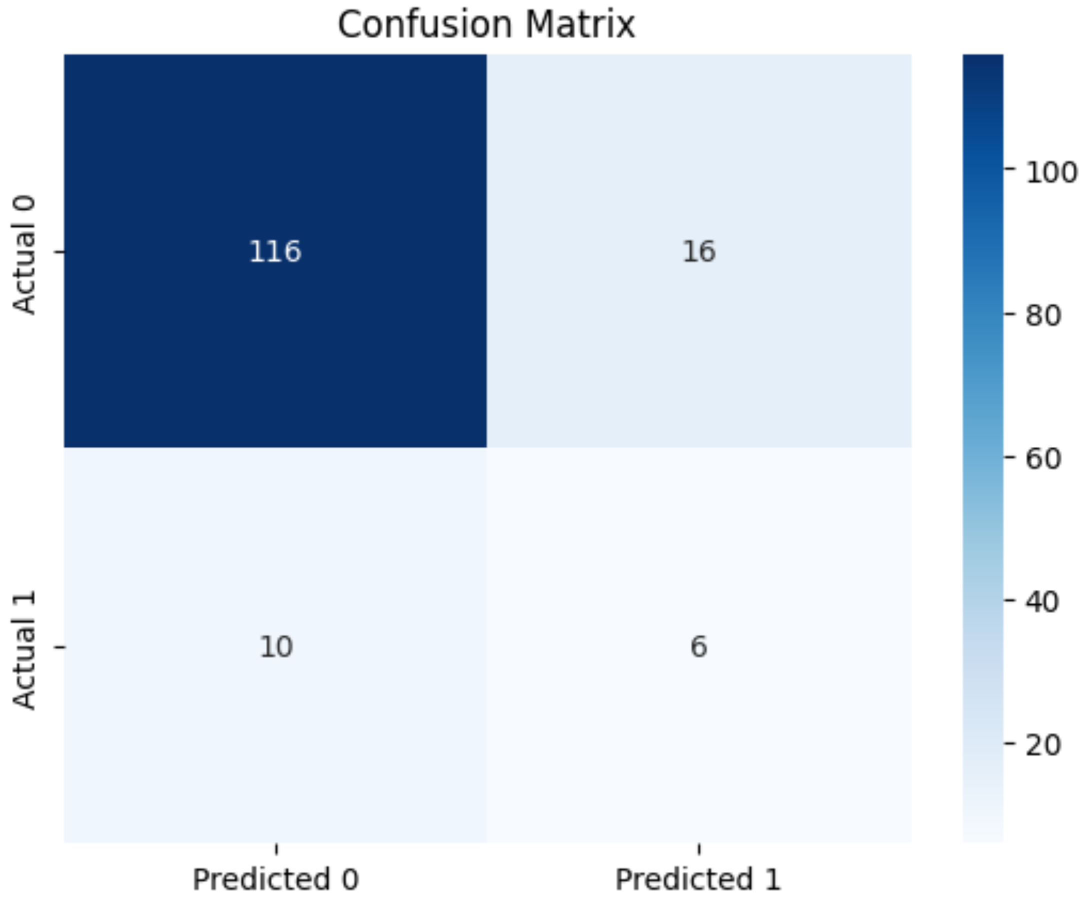


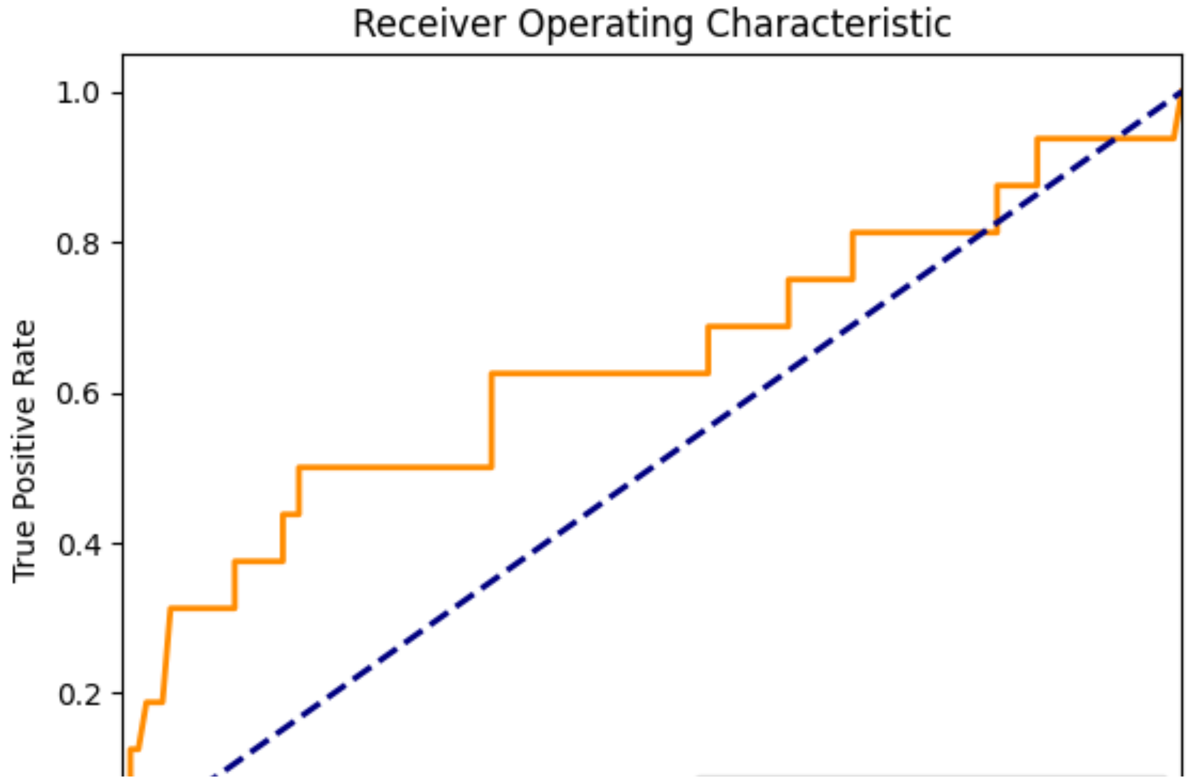


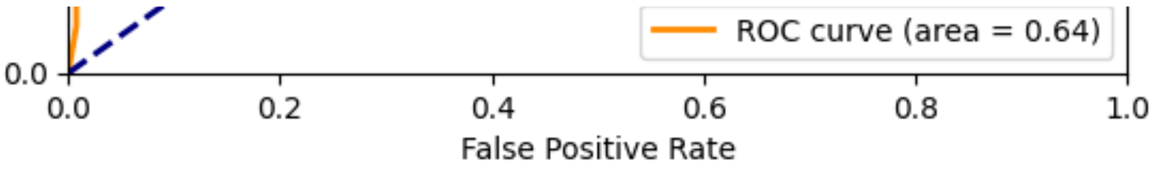


```
ROC Curve Metrics:
FPR: [0.         0.         0.00757576 0.03030303 0.03787879 0.0530303
 0.06060606 0.06060606 0.09848485 0.11363636 0.12121212 0.12121212
 0.16666667 0.16666667 0.18181818 0.18181818 0.18939394 0.20454545
 0.22727273 0.24242424 0.29545455 0.31818182 0.36363636 0.38636364
 0.45454545 0.45454545 0.46212121 0.47727273 0.48484848 0.5
 0.58333333 0.58333333 0.65909091 0.65909091 0.71212121 0.71212121
 0.75       0.75       0.76515152 0.82575758 0.82575758 0.87878788
 0.88636364 0.96212121 0.97727273 1.        ]
TPR: [0.     0.0625 0.125  0.125  0.25   0.25   0.3125 0.375  0.375  0.375
 0.375  0.4375 0.4375 0.5    0.5    0.5625 0.5625 0.5625 0.5625 0.5625
 0.5625 0.5625 0.5625 0.5625 0.5625 0.625  0.625  0.625  0.625  0.625
 0.625  0.6875 0.6875 0.75   0.75   0.8125 0.8125 0.875  0.875  0.875
 0.9375 0.9375 1.     1.     1.     1.    ]
ROC AUC: 0.655

Running evaluation with seed 21...
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.py:87:
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step
Training - Accuracy: 0.5000, Sensitivity: 0.0000, Specificity: 1.0000, F1:
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
Threshold: 0.10, Accuracy: 0.1081, Sensitivity: 0.9375, Specificity: 0.0076
Threshold: 0.15, Accuracy: 0.3649, Sensitivity: 0.8125, Specificity: 0.3106
Threshold: 0.20, Accuracy: 0.5000, Sensitivity: 0.6250, Specificity: 0.4848
Threshold: 0.25, Accuracy: 0.6216, Sensitivity: 0.6250, Specificity: 0.6212
Threshold: 0.30, Accuracy: 0.6824, Sensitivity: 0.5000, Specificity: 0.7045
Threshold: 0.35, Accuracy: 0.7703, Sensitivity: 0.5000, Specificity: 0.8030
Threshold: 0.40, Accuracy: 0.8041, Sensitivity: 0.4375, Specificity: 0.8485
Threshold: 0.45, Accuracy: 0.8243, Sensitivity: 0.3750, Specificity: 0.8788
Threshold: 0.50, Accuracy: 0.8649, Sensitivity: 0.3125, Specificity: 0.9318
Threshold: 0.55, Accuracy: 0.8851, Sensitivity: 0.3125, Specificity: 0.9545
Threshold: 0.60, Accuracy: 0.8851, Sensitivity: 0.1875, Specificity: 0.9697
Threshold: 0.65, Accuracy: 0.8851, Sensitivity: 0.1875, Specificity: 0.9697
Threshold: 0.70, Accuracy: 0.8919, Sensitivity: 0.1250, Specificity: 0.9848
Threshold: 0.75, Accuracy: 0.8986, Sensitivity: 0.1250, Specificity: 0.9924
Threshold: 0.80, Accuracy: 0.8919, Sensitivity: 0.0000, Specificity: 1.0000
Threshold: 0.85, Accuracy: 0.8919, Sensitivity: 0.0000, Specificity: 1.0000
Threshold: 0.90, Accuracy: 0.8919, Sensitivity: 0.0000, Specificity: 1.0000
```

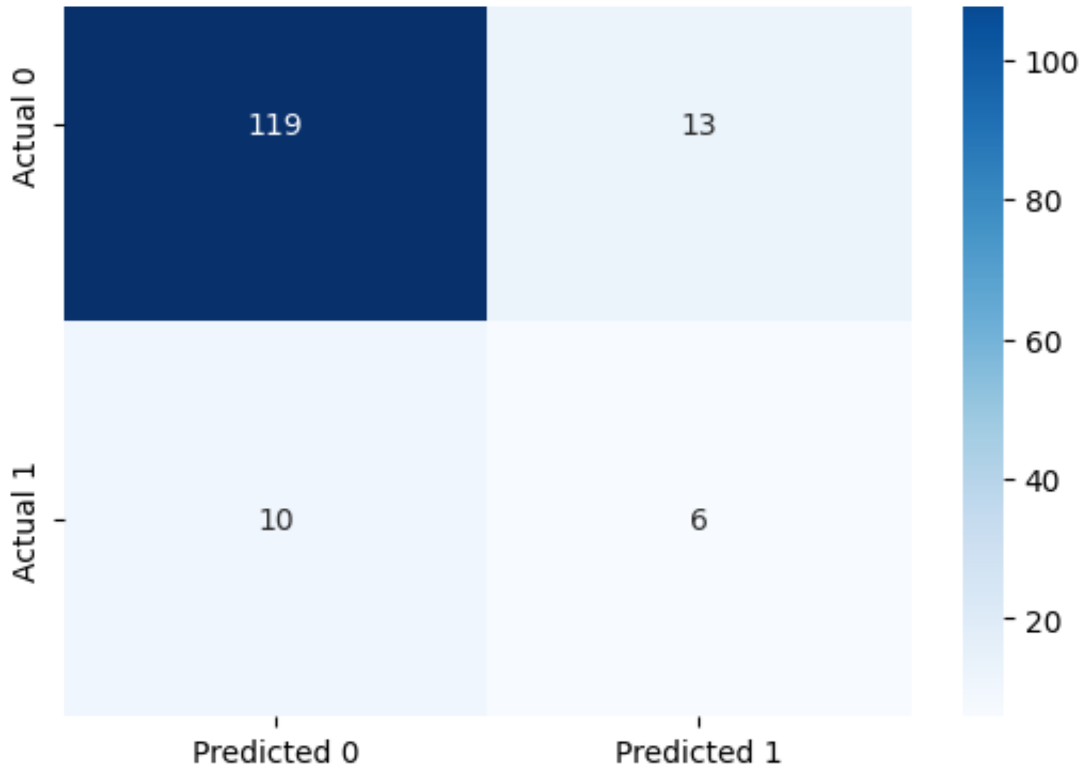
```
Threshold: 0.90, Accuracy: 0.8919, Sensitivity: 0.0000, Specificity: 1.0000
Threshold: 0.95, Accuracy: 0.8919, Sensitivity: 0.0000, Specificity: 1.0000
Threshold: 1.00, Accuracy: 0.8919, Sensitivity: 0.0000, Specificity: 1.0000
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step

Metrics for chosen threshold 0.45:
Accuracy: 0.8243, Sensitivity: 0.3750, Specificity: 0.8788, F1: 0.3158, ROC
```

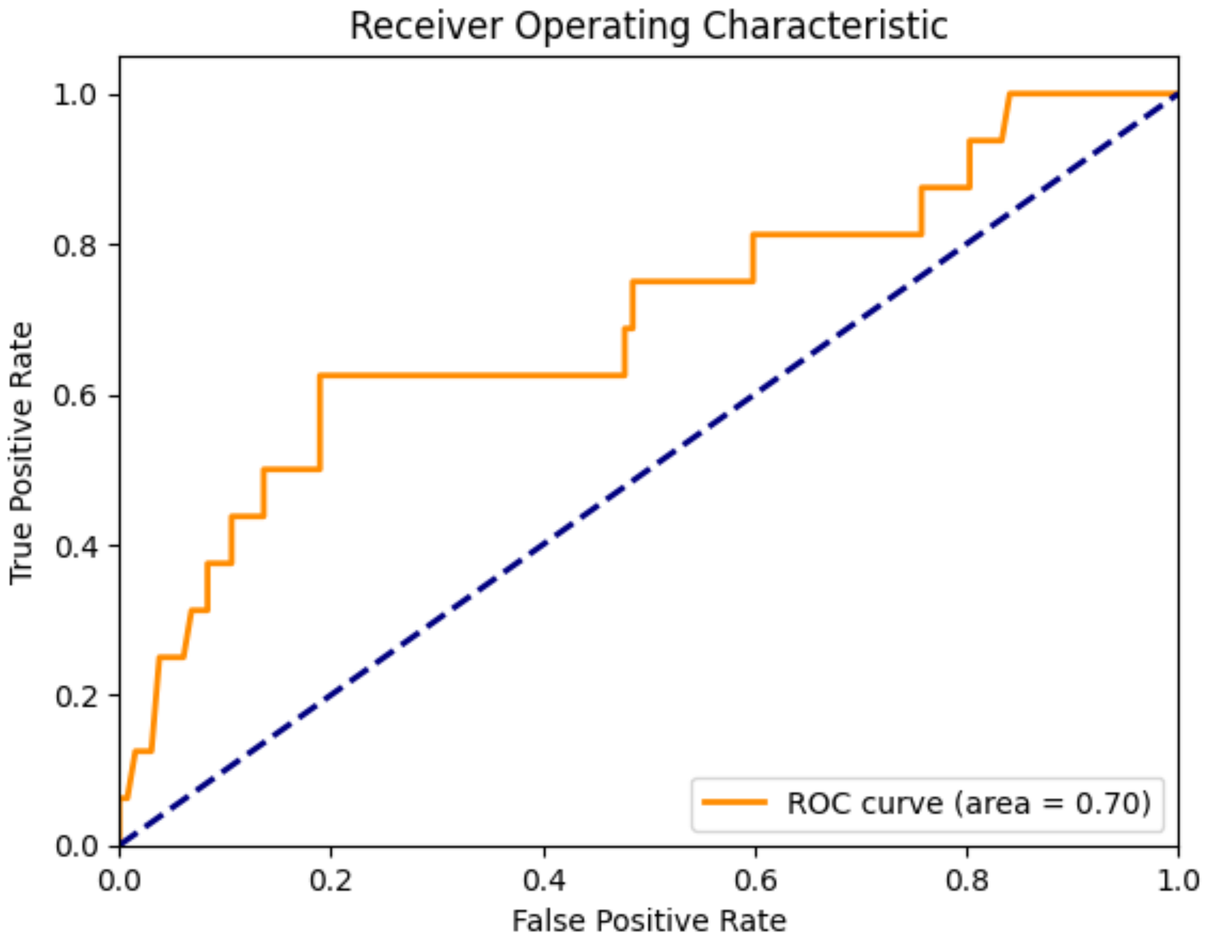




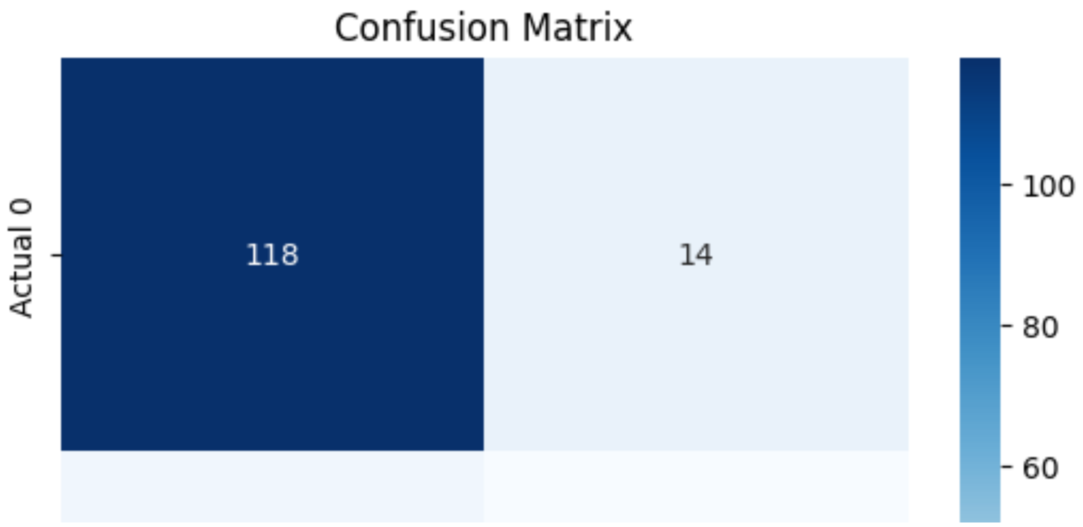


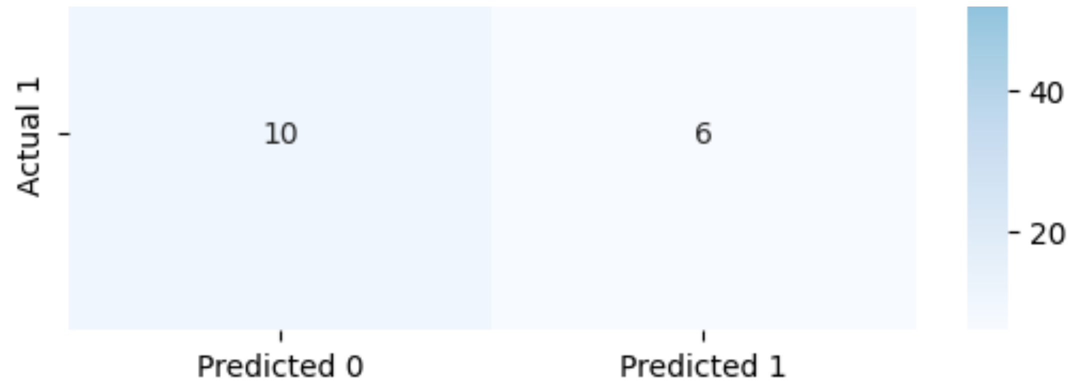
```
ROC Curve Metrics:
FPR: [0.         0.00757576 0.00757576 0.01515152 0.02272727 0.03787879
 0.04545455 0.09090909 0.10606061 0.10606061 0.12121212 0.12878788
 0.14393939 0.15151515 0.15151515 0.16666667 0.16666667 0.1969697
 0.21212121 0.34848485 0.34848485 0.37121212 0.52272727 0.53787879
 0.5530303  0.5530303  0.56060606 0.58333333 0.59090909 0.60606061
 0.62878788 0.62878788 0.67424242 0.68939394 0.68939394 0.82575758
 0.82575758 0.86363636 0.86363636 0.96969697 0.98484848 0.99242424
 1.        ]
TPR: [0.     0.0625 0.125  0.125  0.1875 0.1875 0.3125 0.3125 0.3125 0.375
 0.375  0.375  0.375  0.375  0.4375 0.4375 0.5    0.5    0.5    0.5
 0.625  0.625  0.625  0.625  0.625  0.6875 0.6875 0.6875 0.6875 0.6875
 0.6875 0.75   0.75   0.75   0.8125 0.8125 0.875  0.875  0.9375 0.9375
 0.9375 0.9375 1.     ]
ROC AUC: 0.638

Running evaluation with seed 22...
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.py:87:
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step
Training - Accuracy: 0.5000, Sensitivity: 0.0000, Specificity: 1.0000, F1:
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
Threshold: 0.10, Accuracy: 0.1081, Sensitivity: 1.0000, Specificity: 0.0000
Threshold: 0.15, Accuracy: 0.4122, Sensitivity: 0.8125, Specificity: 0.3636
Threshold: 0.20, Accuracy: 0.5743, Sensitivity: 0.6250, Specificity: 0.5682
Threshold: 0.25, Accuracy: 0.6554, Sensitivity: 0.6250, Specificity: 0.6591
Threshold: 0.30, Accuracy: 0.6757, Sensitivity: 0.6250, Specificity: 0.6818
Threshold: 0.35, Accuracy: 0.7635, Sensitivity: 0.6250, Specificity: 0.7803
Threshold: 0.40, Accuracy: 0.8243, Sensitivity: 0.5000, Specificity: 0.8636
Threshold: 0.45, Accuracy: 0.8446, Sensitivity: 0.3750, Specificity: 0.9015
Threshold: 0.50, Accuracy: 0.8581, Sensitivity: 0.3125, Specificity: 0.9242
Threshold: 0.55, Accuracy: 0.8716, Sensitivity: 0.2500, Specificity: 0.9470
Threshold: 0.60, Accuracy: 0.8851, Sensitivity: 0.2500, Specificity: 0.9621
Threshold: 0.65, Accuracy: 0.8851, Sensitivity: 0.1250, Specificity: 0.9773
Threshold: 0.70, Accuracy: 0.8851, Sensitivity: 0.1250, Specificity: 0.9773
Threshold: 0.75, Accuracy: 0.8986, Sensitivity: 0.0625, Specificity: 1.0000
Threshold: 0.80, Accuracy: 0.8919, Sensitivity: 0.0000, Specificity: 1.0000
Threshold: 0.85, Accuracy: 0.8919, Sensitivity: 0.0000, Specificity: 1.0000
Threshold: 0.90, Accuracy: 0.8919, Sensitivity: 0.0000, Specificity: 1.0000
Threshold: 0.95, Accuracy: 0.8919, Sensitivity: 0.0000, Specificity: 1.0000
Threshold: 1.00, Accuracy: 0.8919, Sensitivity: 0.0000, Specificity: 1.0000
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step

Metrics for chosen threshold 0.45:
Accuracy: 0.8446, Sensitivity: 0.3750, Specificity: 0.9015, F1: 0.3429, ROC
```

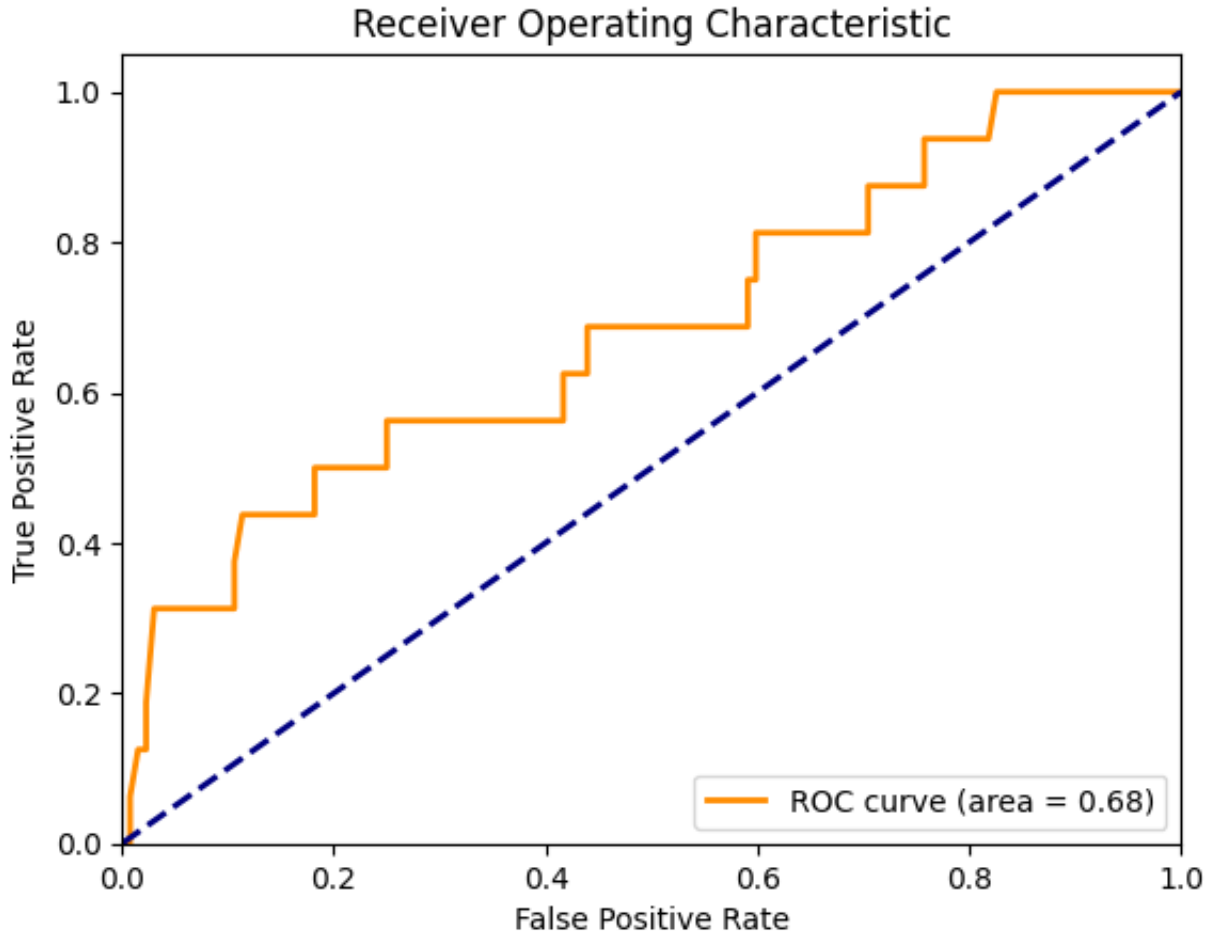




```
ROC Curve Metrics:
FPR: [0.         0.         0.00757576 0.01515152 0.03030303 0.03787879
 0.06060606 0.06818182 0.08333333 0.08333333 0.09848485 0.10606061
 0.10606061 0.13636364 0.13636364 0.18939394 0.18939394 0.20454545
 0.23484848 0.25       0.36363636 0.38636364 0.40151515 0.41666667
 0.43939394 0.45454545 0.47727273 0.47727273 0.48484848 0.48484848
 0.59848485 0.59848485 0.62121212 0.63636364 0.65909091 0.75757576
 0.75757576 0.78030303 0.79545455 0.8030303  0.8030303  0.83333333
 0.84090909 0.89393939 0.90909091 1.        ]
TPR: [0.     0.0625 0.0625 0.125  0.125  0.25   0.25   0.3125 0.3125 0.375
 0.375  0.375  0.4375 0.4375 0.5    0.5    0.625  0.625  0.625  0.625
 0.625  0.625  0.625  0.625  0.625  0.625  0.625  0.6875 0.6875 0.75
 0.75   0.8125 0.8125 0.8125 0.8125 0.8125 0.875  0.875  0.875  0.875
 0.9375 0.9375 1.     1.     1.     1.    ]
ROC AUC: 0.700

Running evaluation with seed 23...
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.py:87: 
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step
Training - Accuracy: 0.5000, Sensitivity: 0.0000, Specificity: 1.0000, F1:
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
Threshold: 0.10, Accuracy: 0.1081, Sensitivity: 1.0000, Specificity: 0.0000
Threshold: 0.15, Accuracy: 0.1081, Sensitivity: 1.0000, Specificity: 0.0000
Threshold: 0.20, Accuracy: 0.4189, Sensitivity: 0.8125, Specificity: 0.3712
Threshold: 0.25, Accuracy: 0.6622, Sensitivity: 0.5625, Specificity: 0.6742
Threshold: 0.30, Accuracy: 0.7432, Sensitivity: 0.5000, Specificity: 0.7727
Threshold: 0.35, Accuracy: 0.7770, Sensitivity: 0.5000, Specificity: 0.8106
Threshold: 0.40, Accuracy: 0.8176, Sensitivity: 0.4375, Specificity: 0.8636
Threshold: 0.45, Accuracy: 0.8378, Sensitivity: 0.3750, Specificity: 0.8939
Threshold: 0.50, Accuracy: 0.8581, Sensitivity: 0.3125, Specificity: 0.9242
Threshold: 0.55, Accuracy: 0.8716, Sensitivity: 0.3125, Specificity: 0.9394
Threshold: 0.60, Accuracy: 0.8919, Sensitivity: 0.3125, Specificity: 0.9621
Threshold: 0.65, Accuracy: 0.8851, Sensitivity: 0.1250, Specificity: 0.9773
Threshold: 0.70, Accuracy: 0.8919, Sensitivity: 0.0000, Specificity: 1.0000
Threshold: 0.75, Accuracy: 0.8919, Sensitivity: 0.0000, Specificity: 1.0000
Threshold: 0.80, Accuracy: 0.8919, Sensitivity: 0.0000, Specificity: 1.0000
Threshold: 0.85, Accuracy: 0.8919, Sensitivity: 0.0000, Specificity: 1.0000
Threshold: 0.90, Accuracy: 0.8919, Sensitivity: 0.0000, Specificity: 1.0000
Threshold: 0.95, Accuracy: 0.8919, Sensitivity: 0.0000, Specificity: 1.0000
Threshold: 1.00, Accuracy: 0.8919, Sensitivity: 0.0000, Specificity: 1.0000
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step

Metrics for chosen threshold 0.45:
Accuracy: 0.8378, Sensitivity: 0.3750, Specificity: 0.8939, F1: 0.3333, ROC
```

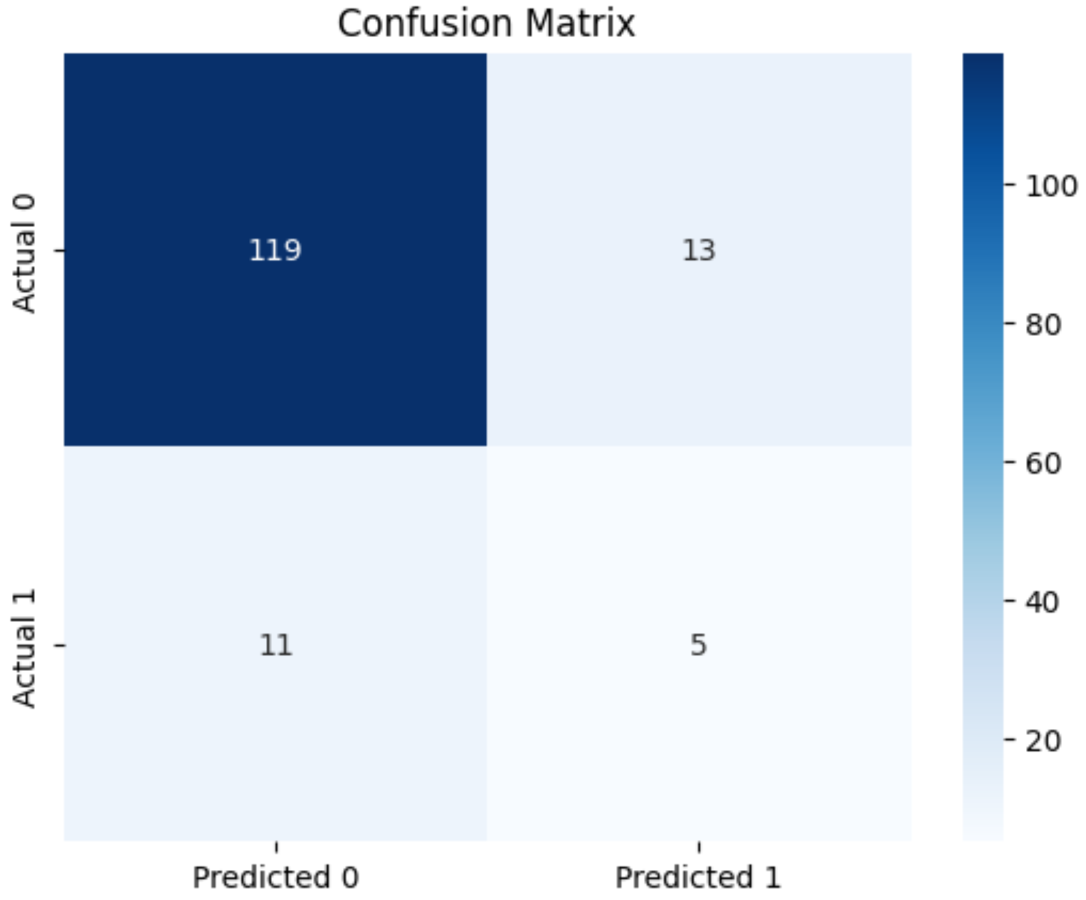


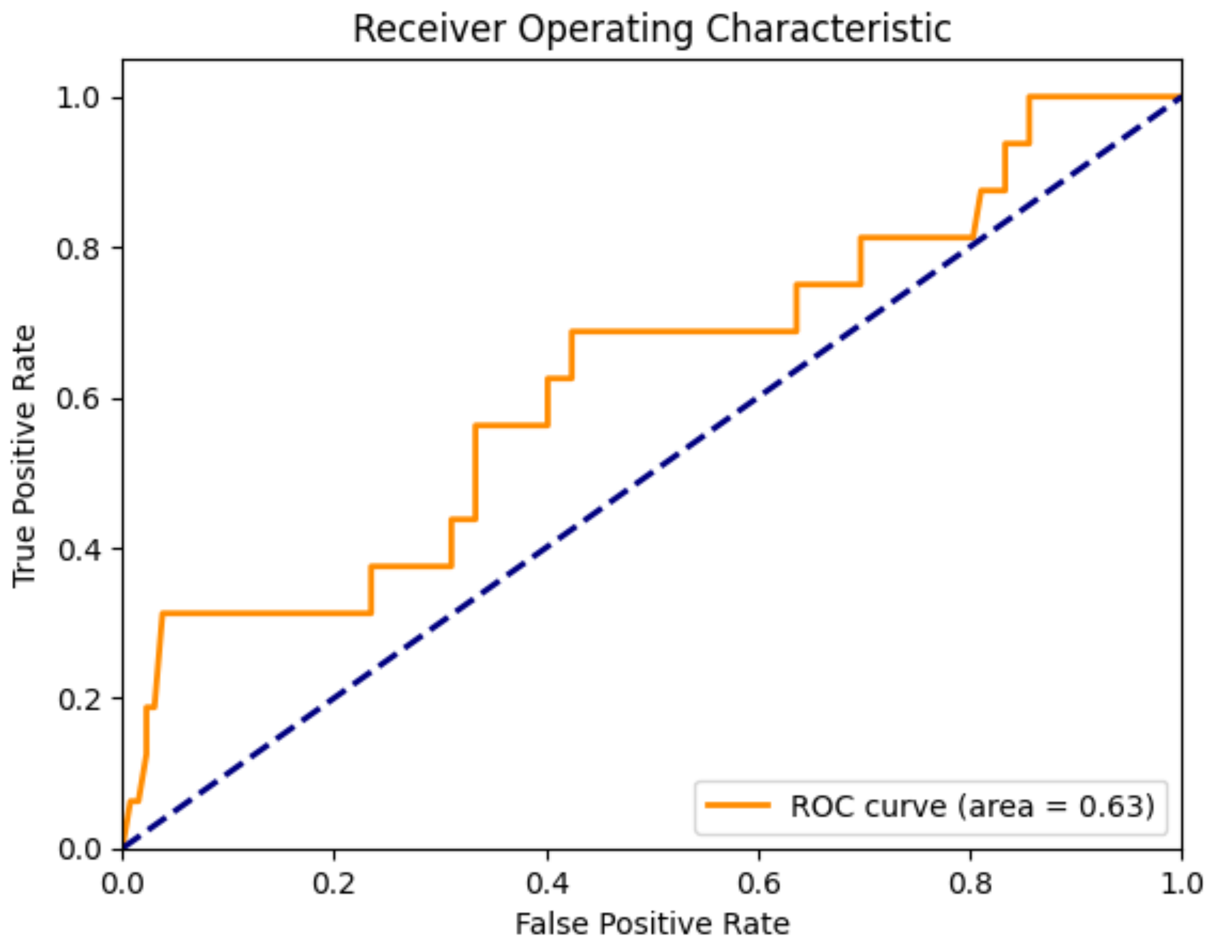




```
ROC Curve Metrics:
FPR: [0.         0.00757576 0.00757576 0.01515152 0.02272727 0.02272727
 0.03030303 0.10606061 0.10606061 0.11363636 0.12121212 0.13636364
 0.18181818 0.18181818 0.22727273 0.25       0.25       0.26515152
 0.28030303 0.31060606 0.32575758 0.33333333 0.34848485 0.37121212
 0.38636364 0.41666667 0.41666667 0.43181818 0.43939394 0.43939394
 0.53787879 0.5530303  0.56818182 0.58333333 0.59090909 0.59090909
 0.59848485 0.59848485 0.62878788 0.65151515 0.70454545 0.70454545
 0.75757576 0.75757576 0.81818182 0.82575758 1.        ]
TPR: [0.     0.     0.0625 0.125  0.125  0.1875 0.3125 0.3125 0.375  0.4375
 0.4375 0.4375 0.4375 0.5    0.5    0.5    0.5625 0.5625 0.5625 0.5625
 0.5625 0.5625 0.5625 0.5625 0.5625 0.5625 0.625  0.625  0.625  0.6875
 0.6875 0.6875 0.6875 0.6875 0.6875 0.75   0.75   0.8125 0.8125 0.8125
 0.8125 0.875  0.875  0.9375 0.9375 1.     1.    ]
ROC AUC: 0.683
```

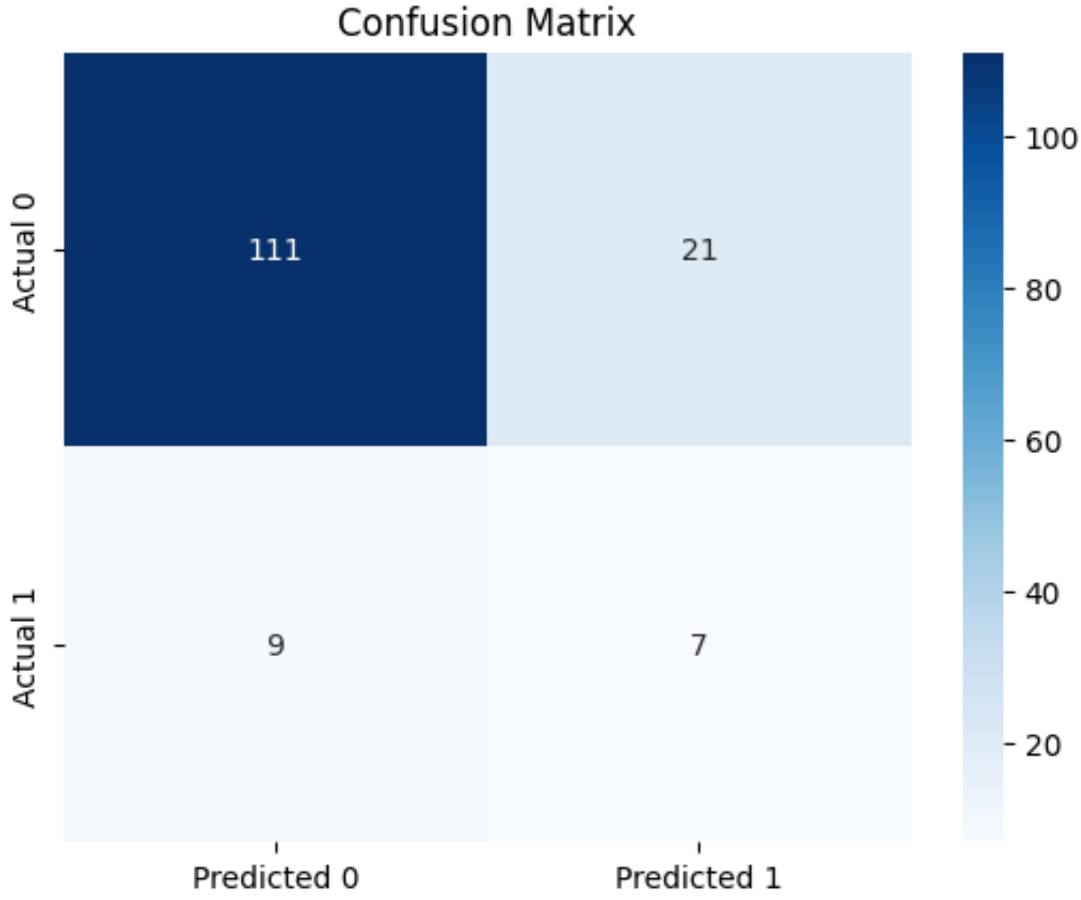
```
Running evaluation with seed 24...
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.py:87:
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step
Training - Accuracy: 0.5000, Sensitivity: 0.0000, Specificity: 1.0000, F1:
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
Threshold: 0.10, Accuracy: 0.1081, Sensitivity: 1.0000, Specificity: 0.0000
Threshold: 0.15, Accuracy: 0.3919, Sensitivity: 0.7500, Specificity: 0.3485
Threshold: 0.20, Accuracy: 0.5811, Sensitivity: 0.6250, Specificity: 0.5758
Threshold: 0.25, Accuracy: 0.6486, Sensitivity: 0.5625, Specificity: 0.6591
Threshold: 0.30, Accuracy: 0.6824, Sensitivity: 0.3750, Specificity: 0.7197
Threshold: 0.35, Accuracy: 0.7500, Sensitivity: 0.3125, Specificity: 0.8030
Threshold: 0.40, Accuracy: 0.8243, Sensitivity: 0.3125, Specificity: 0.8864
Threshold: 0.45, Accuracy: 0.8378, Sensitivity: 0.3125, Specificity: 0.9015
Threshold: 0.50, Accuracy: 0.8581, Sensitivity: 0.3125, Specificity: 0.9242
Threshold: 0.55, Accuracy: 0.8784, Sensitivity: 0.3125, Specificity: 0.9470
Threshold: 0.60, Accuracy: 0.8784, Sensitivity: 0.3125, Specificity: 0.9470
Threshold: 0.65, Accuracy: 0.8919, Sensitivity: 0.1875, Specificity: 0.9773
Threshold: 0.70, Accuracy: 0.8851, Sensitivity: 0.1250, Specificity: 0.9773
Threshold: 0.75, Accuracy: 0.8919, Sensitivity: 0.0625, Specificity: 0.9924
Threshold: 0.80, Accuracy: 0.8919, Sensitivity: 0.0000, Specificity: 1.0000
Threshold: 0.85, Accuracy: 0.8919, Sensitivity: 0.0000, Specificity: 1.0000
Threshold: 0.90, Accuracy: 0.8919, Sensitivity: 0.0000, Specificity: 1.0000
Threshold: 0.95, Accuracy: 0.8919, Sensitivity: 0.0000, Specificity: 1.0000
Threshold: 1.00, Accuracy: 0.8919, Sensitivity: 0.0000, Specificity: 1.0000
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step

Metrics for chosen threshold 0.45:
Accuracy: 0.8378, Sensitivity: 0.3125, Specificity: 0.9015, F1: 0.2941, ROC
```

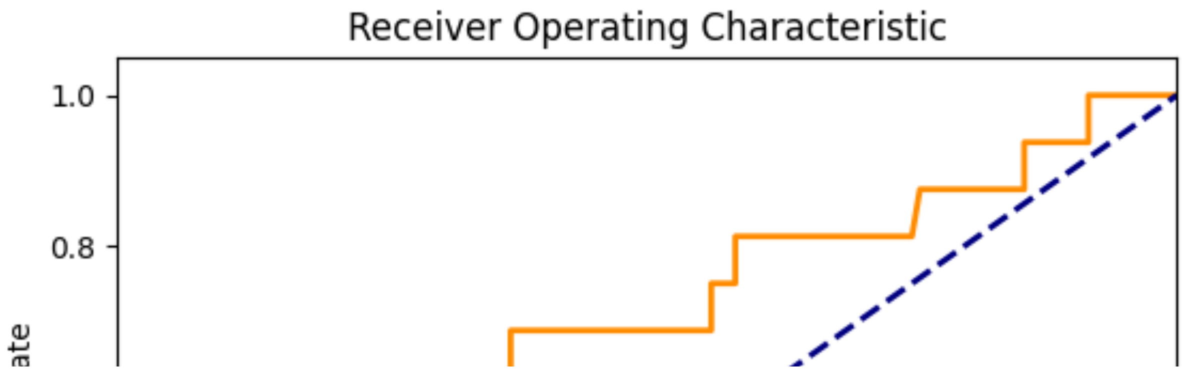


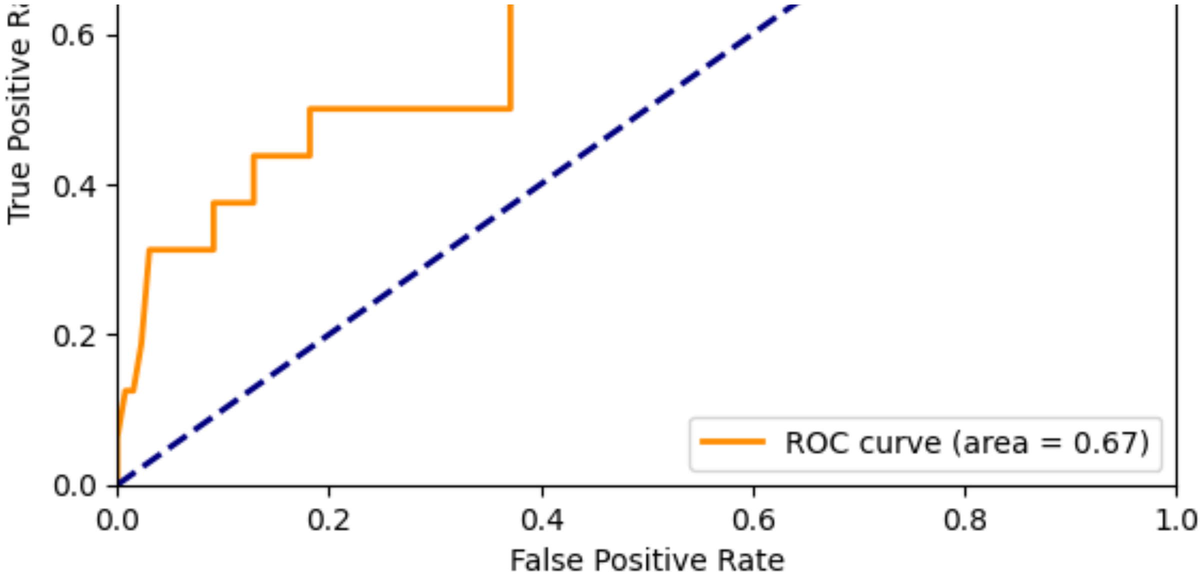


```
ROC Curve Metrics:
FPR: [0.         0.00757576 0.01515152 0.02272727 0.02272727 0.03030303
 0.03787879 0.08333333 0.09848485 0.21212121 0.22727273 0.23484848
 0.23484848 0.24242424 0.25757576 0.31060606 0.31060606 0.33333333
 0.33333333 0.36363636 0.38636364 0.40151515 0.40151515 0.42424242
 0.42424242 0.43939394 0.46212121 0.53787879 0.5530303  0.56060606
 0.57575758 0.63636364 0.63636364 0.68181818 0.6969697  0.6969697
 0.8030303  0.81060606 0.83333333 0.83333333 0.85606061 0.85606061
 0.93181818 0.9469697  1.        ]
TPR: [0.     0.0625 0.0625 0.125  0.1875 0.1875 0.3125 0.3125 0.3125 0.3125
 0.3125 0.3125 0.375  0.375  0.375  0.375  0.4375 0.4375 0.5625 0.5625
 0.5625 0.5625 0.625  0.625  0.6875 0.6875 0.6875 0.6875 0.6875 0.6875
 0.6875 0.6875 0.75   0.75   0.75   0.8125 0.8125 0.875  0.875  0.9375
 0.9375 1.     1.     1.     1.    ]
ROC AUC: 0.626

Running evaluation with seed 25...
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.py:87:
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step
Training - Accuracy: 0.5000, Sensitivity: 0.0000, Specificity: 1.0000, F1:
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
Threshold: 0.10, Accuracy: 0.1081, Sensitivity: 1.0000, Specificity: 0.0000
Threshold: 0.15, Accuracy: 0.4527, Sensitivity: 0.7500, Specificity: 0.4167
Threshold: 0.20, Accuracy: 0.5473, Sensitivity: 0.6875, Specificity: 0.5303
Threshold: 0.25, Accuracy: 0.5946, Sensitivity: 0.6875, Specificity: 0.5833
Threshold: 0.30, Accuracy: 0.6216, Sensitivity: 0.5000, Specificity: 0.6364
```

```
Threshold: 0.35, Accuracy: 0.6757, Sensitivity: 0.5000, Specificity: 0.6970
Threshold: 0.40, Accuracy: 0.7703, Sensitivity: 0.5000, Specificity: 0.8030
Threshold: 0.45, Accuracy: 0.7973, Sensitivity: 0.4375, Specificity: 0.8409
Threshold: 0.50, Accuracy: 0.8446, Sensitivity: 0.3125, Specificity: 0.9091
Threshold: 0.55, Accuracy: 0.8851, Sensitivity: 0.3125, Specificity: 0.9545
Threshold: 0.60, Accuracy: 0.8986, Sensitivity: 0.3125, Specificity: 0.9697
Threshold: 0.65, Accuracy: 0.8986, Sensitivity: 0.1250, Specificity: 0.9924
Threshold: 0.70, Accuracy: 0.8986, Sensitivity: 0.0625, Specificity: 1.0000
Threshold: 0.75, Accuracy: 0.8919, Sensitivity: 0.0000, Specificity: 1.0000
Threshold: 0.80, Accuracy: 0.8919, Sensitivity: 0.0000, Specificity: 1.0000
Threshold: 0.85, Accuracy: 0.8919, Sensitivity: 0.0000, Specificity: 1.0000
Threshold: 0.90, Accuracy: 0.8919, Sensitivity: 0.0000, Specificity: 1.0000
Threshold: 0.95, Accuracy: 0.8919, Sensitivity: 0.0000, Specificity: 1.0000
Threshold: 1.00, Accuracy: 0.8919, Sensitivity: 0.0000, Specificity: 1.0000
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step

Metrics for chosen threshold 0.45:
Accuracy: 0.7973, Sensitivity: 0.4375, Specificity: 0.8409, F1: 0.3182, ROC
```

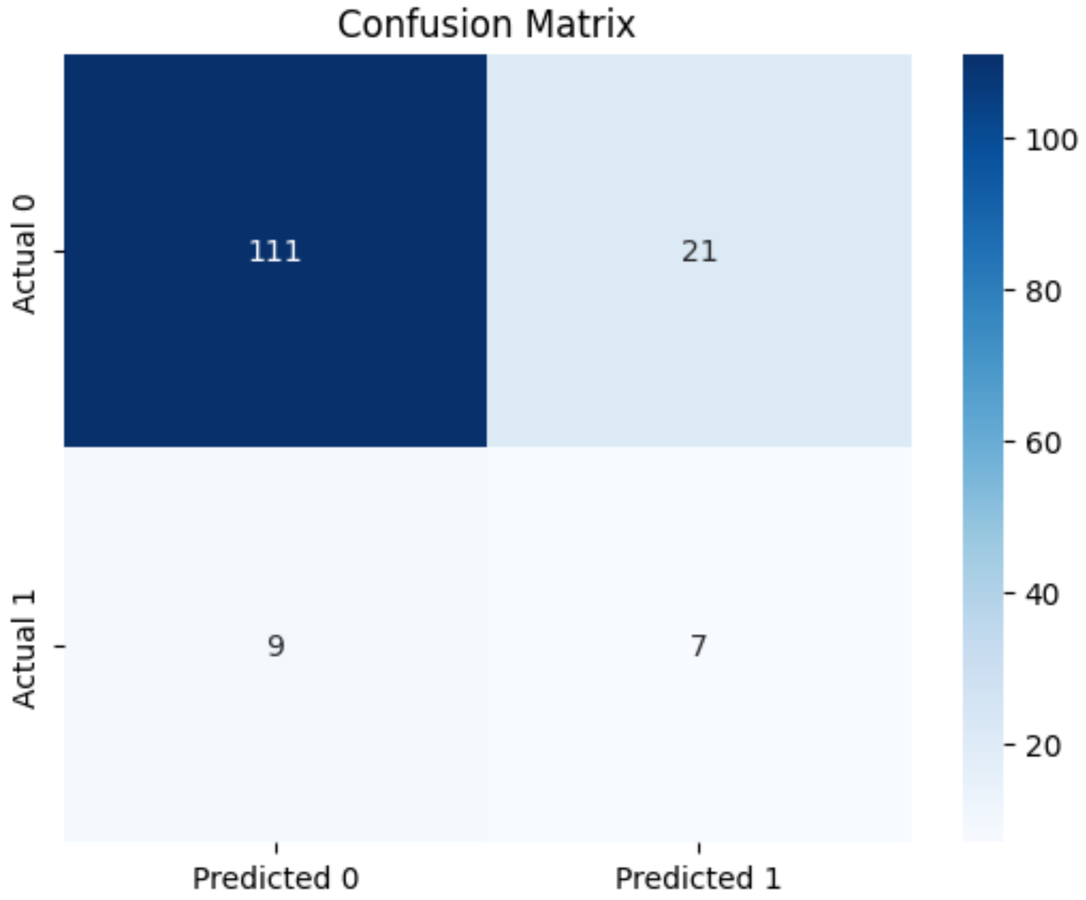


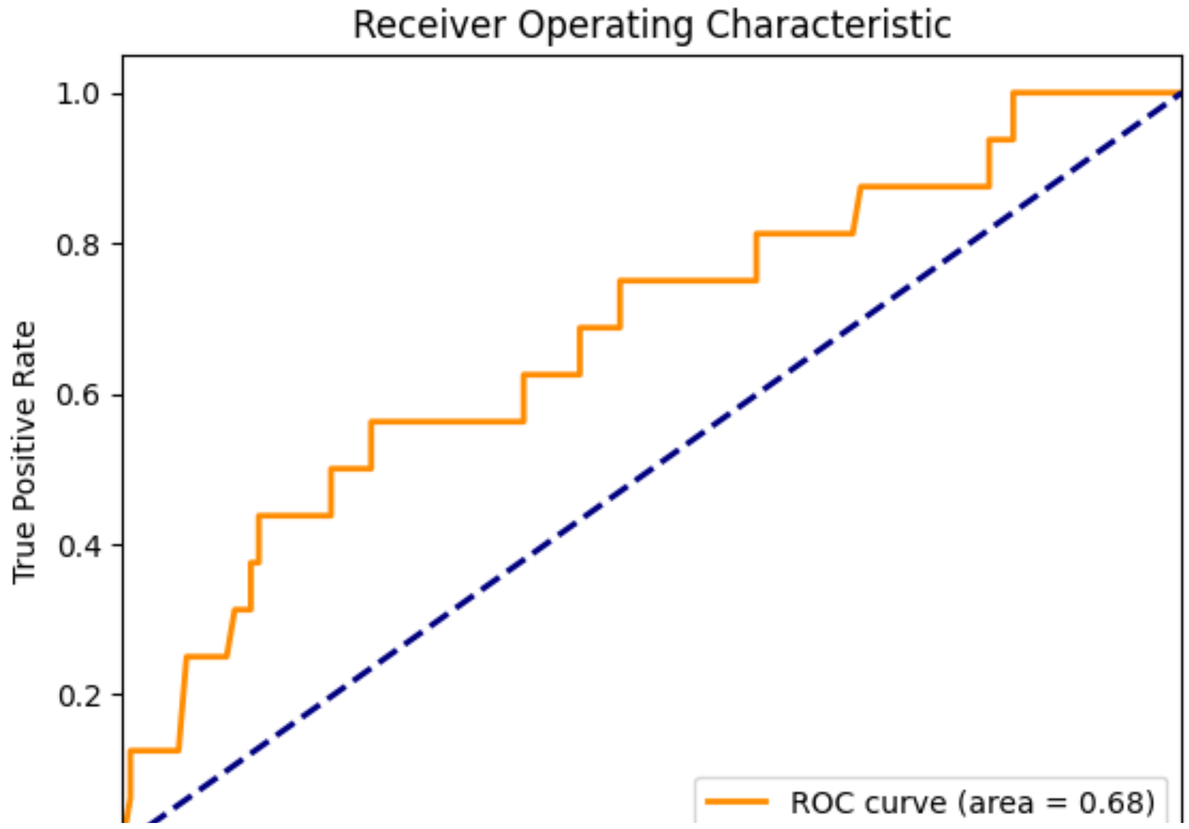


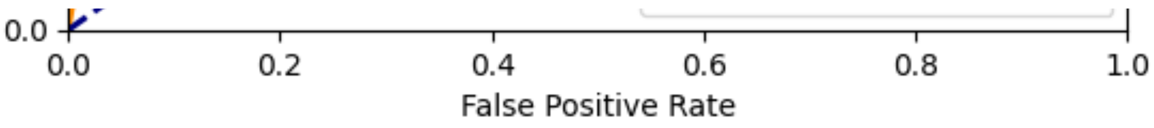


```
ROC Curve Metrics:
FPR: [0.         0.         0.00757576 0.01515152 0.02272727 0.03030303
 0.06060606 0.07575758 0.09090909 0.09090909 0.10606061 0.12121212
 0.12878788 0.12878788 0.13636364 0.15151515 0.18181818 0.18181818
 0.1969697  0.21969697 0.26515152 0.28787879 0.37121212 0.37121212
 0.56060606 0.56060606 0.58333333 0.58333333 0.59848485 0.75
 0.75757576 0.78787879 0.8030303  0.85606061 0.85606061 0.87121212
 0.91666667 0.91666667 1.        ]
TPR: [0.     0.0625 0.125  0.125  0.1875 0.3125 0.3125 0.3125 0.3125 0.375
 0.375  0.375  0.375  0.4375 0.4375 0.4375 0.4375 0.5    0.5    0.5
 0.5    0.5    0.5    0.6875 0.6875 0.75   0.75   0.8125 0.8125 0.8125
 0.875  0.875  0.875  0.875  0.9375 0.9375 0.9375 1.     1.    ]
ROC AUC: 0.671

Running evaluation with seed 26...
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.py:87:
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step
Training - Accuracy: 0.5000, Sensitivity: 0.0000, Specificity: 1.0000, F1:
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
Threshold: 0.10, Accuracy: 0.1081, Sensitivity: 1.0000, Specificity: 0.0000
Threshold: 0.15, Accuracy: 0.1081, Sensitivity: 1.0000, Specificity: 0.0000
Threshold: 0.20, Accuracy: 0.6081, Sensitivity: 0.6250, Specificity: 0.6061
Threshold: 0.25, Accuracy: 0.6486, Sensitivity: 0.5625, Specificity: 0.6591
Threshold: 0.30, Accuracy: 0.7095, Sensitivity: 0.5625, Specificity: 0.7273
Threshold: 0.35, Accuracy: 0.7635, Sensitivity: 0.5000, Specificity: 0.7955
Threshold: 0.40, Accuracy: 0.7905, Sensitivity: 0.4375, Specificity: 0.8333
Threshold: 0.45, Accuracy: 0.7973, Sensitivity: 0.4375, Specificity: 0.8409
Threshold: 0.50, Accuracy: 0.8243, Sensitivity: 0.3750, Specificity: 0.8788
Threshold: 0.55, Accuracy: 0.8311, Sensitivity: 0.3125, Specificity: 0.8939
Threshold: 0.60, Accuracy: 0.8851, Sensitivity: 0.1250, Specificity: 0.9773
Threshold: 0.65, Accuracy: 0.8919, Sensitivity: 0.0000, Specificity: 1.0000
Threshold: 0.70, Accuracy: 0.8919, Sensitivity: 0.0000, Specificity: 1.0000
Threshold: 0.75, Accuracy: 0.8919, Sensitivity: 0.0000, Specificity: 1.0000
Threshold: 0.80, Accuracy: 0.8919, Sensitivity: 0.0000, Specificity: 1.0000
Threshold: 0.85, Accuracy: 0.8919, Sensitivity: 0.0000, Specificity: 1.0000
Threshold: 0.90, Accuracy: 0.8919, Sensitivity: 0.0000, Specificity: 1.0000
Threshold: 0.95, Accuracy: 0.8919, Sensitivity: 0.0000, Specificity: 1.0000
```

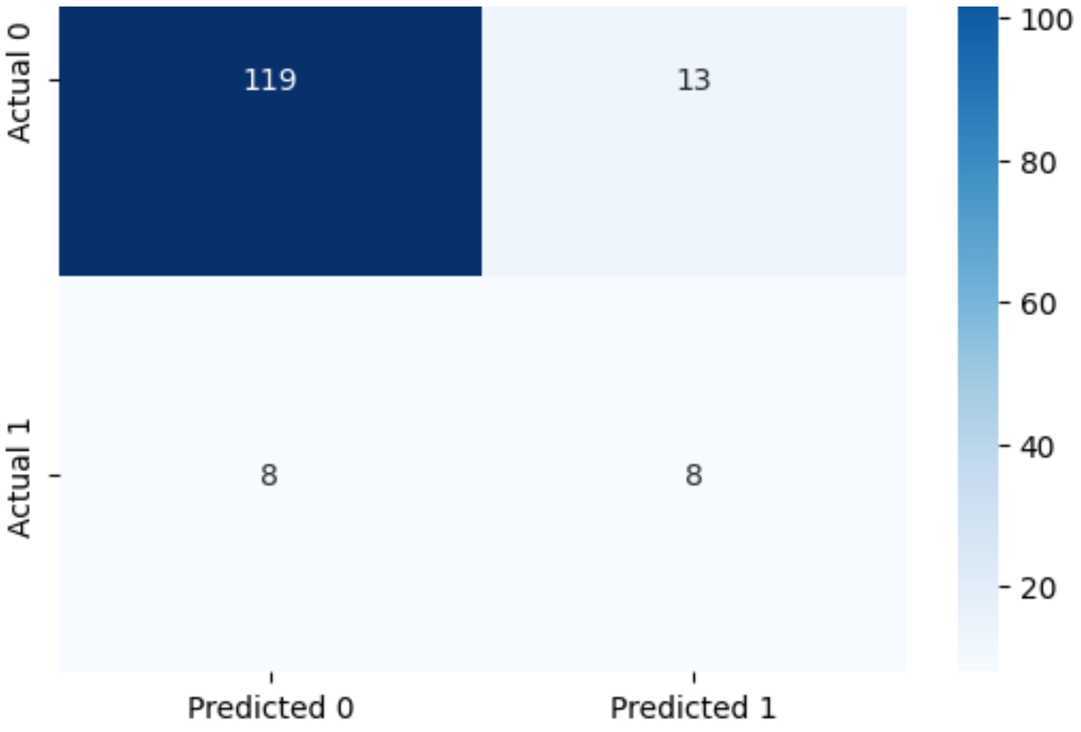
```
Threshold: 0.95, Accuracy: 0.8919, Sensitivity: 0.0000, Specificity: 1.0000
Threshold: 1.00, Accuracy: 0.8919, Sensitivity: 0.0000, Specificity: 1.0000
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step

Metrics for chosen threshold 0.45:
Accuracy: 0.7973, Sensitivity: 0.4375, Specificity: 0.8409, F1: 0.3182, ROC
```

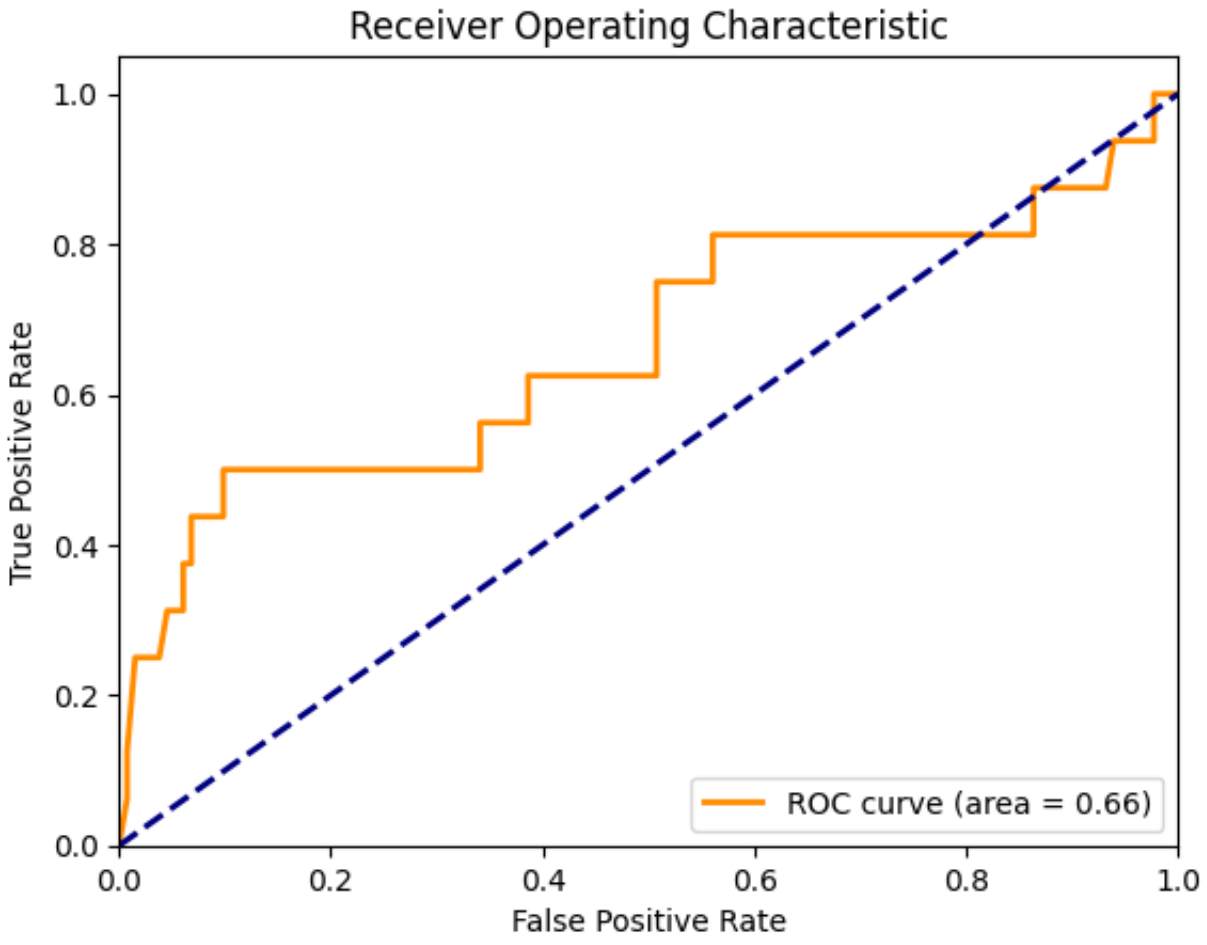




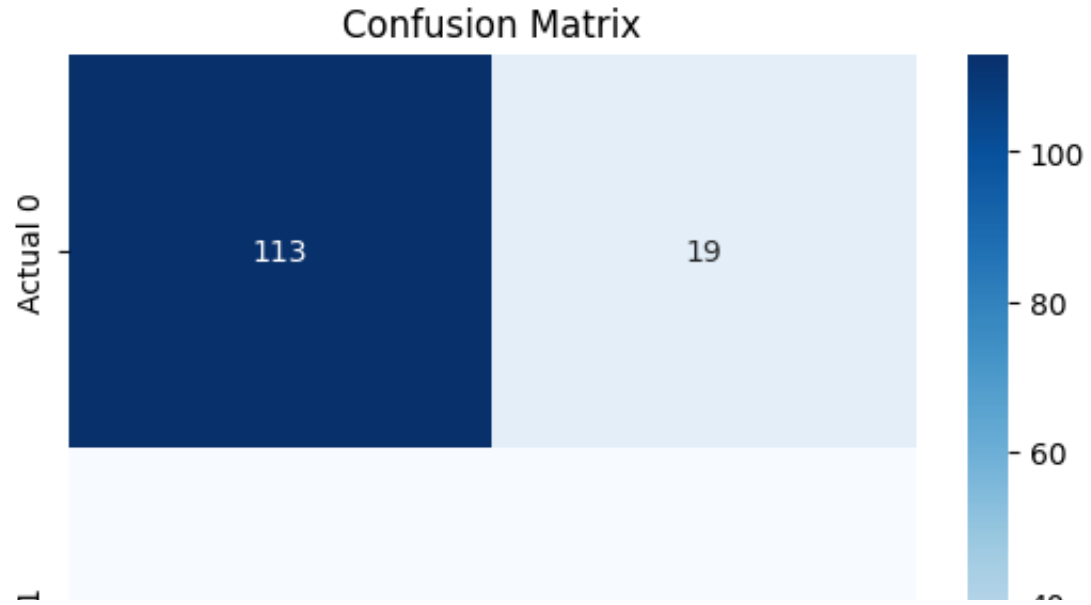


```
ROC Curve Metrics:
FPR: [0.         0.00757576 0.00757576 0.0530303  0.06060606 0.09848485
 0.10606061 0.12121212 0.12121212 0.12878788 0.12878788 0.18181818
 0.1969697  0.1969697  0.21212121 0.23484848 0.23484848 0.25
 0.29545455 0.31060606 0.37878788 0.37878788 0.40151515 0.41666667
 0.43181818 0.43181818 0.43939394 0.45454545 0.46969697 0.46969697
 0.53030303 0.54545455 0.56060606 0.57575758 0.59848485 0.59848485
 0.61363636 0.65151515 0.67424242 0.68939394 0.6969697  0.81818182
 0.81818182 0.84090909 0.84090909 1.        ]
TPR: [0.     0.0625 0.125  0.125  0.25   0.25   0.3125 0.3125 0.375  0.375
 0.4375 0.4375 0.4375 0.5    0.5    0.5    0.5625 0.5625 0.5625 0.5625
 0.5625 0.625  0.625  0.625  0.625  0.6875 0.6875 0.6875 0.6875 0.75
 0.75   0.75   0.75   0.75   0.75   0.8125 0.8125 0.8125 0.8125 0.8125
 0.875  0.875  0.9375 0.9375 1.     1.    ]
ROC AUC: 0.679

Running evaluation with seed 27...
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.py:87: 
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step
Training - Accuracy: 0.5000, Sensitivity: 0.0000, Specificity: 1.0000, F1:
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
Threshold: 0.10, Accuracy: 0.1081, Sensitivity: 1.0000, Specificity: 0.0000
Threshold: 0.15, Accuracy: 0.3649, Sensitivity: 0.8125, Specificity: 0.3106
Threshold: 0.20, Accuracy: 0.5068, Sensitivity: 0.7500, Specificity: 0.4773
Threshold: 0.25, Accuracy: 0.5743, Sensitivity: 0.6250, Specificity: 0.5682
Threshold: 0.30, Accuracy: 0.6554, Sensitivity: 0.5000, Specificity: 0.6742
Threshold: 0.35, Accuracy: 0.7500, Sensitivity: 0.5000, Specificity: 0.7803
Threshold: 0.40, Accuracy: 0.8108, Sensitivity: 0.5000, Specificity: 0.8485
Threshold: 0.45, Accuracy: 0.8581, Sensitivity: 0.5000, Specificity: 0.9015
Threshold: 0.50, Accuracy: 0.8581, Sensitivity: 0.5000, Specificity: 0.9015
Threshold: 0.55, Accuracy: 0.8716, Sensitivity: 0.3125, Specificity: 0.9394
Threshold: 0.60, Accuracy: 0.8919, Sensitivity: 0.2500, Specificity: 0.9697
Threshold: 0.65, Accuracy: 0.8986, Sensitivity: 0.1250, Specificity: 0.9924
Threshold: 0.70, Accuracy: 0.8919, Sensitivity: 0.0625, Specificity: 0.9924
Threshold: 0.75, Accuracy: 0.8919, Sensitivity: 0.0000, Specificity: 1.0000
Threshold: 0.80, Accuracy: 0.8919, Sensitivity: 0.0000, Specificity: 1.0000
Threshold: 0.85, Accuracy: 0.8919, Sensitivity: 0.0000, Specificity: 1.0000
Threshold: 0.90, Accuracy: 0.8919, Sensitivity: 0.0000, Specificity: 1.0000
Threshold: 0.95, Accuracy: 0.8919, Sensitivity: 0.0000, Specificity: 1.0000
Threshold: 1.00, Accuracy: 0.8919, Sensitivity: 0.0000, Specificity: 1.0000
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step

Metrics for chosen threshold 0.45:
Accuracy: 0.8581, Sensitivity: 0.5000, Specificity: 0.9015, F1: 0.4324, ROC
```

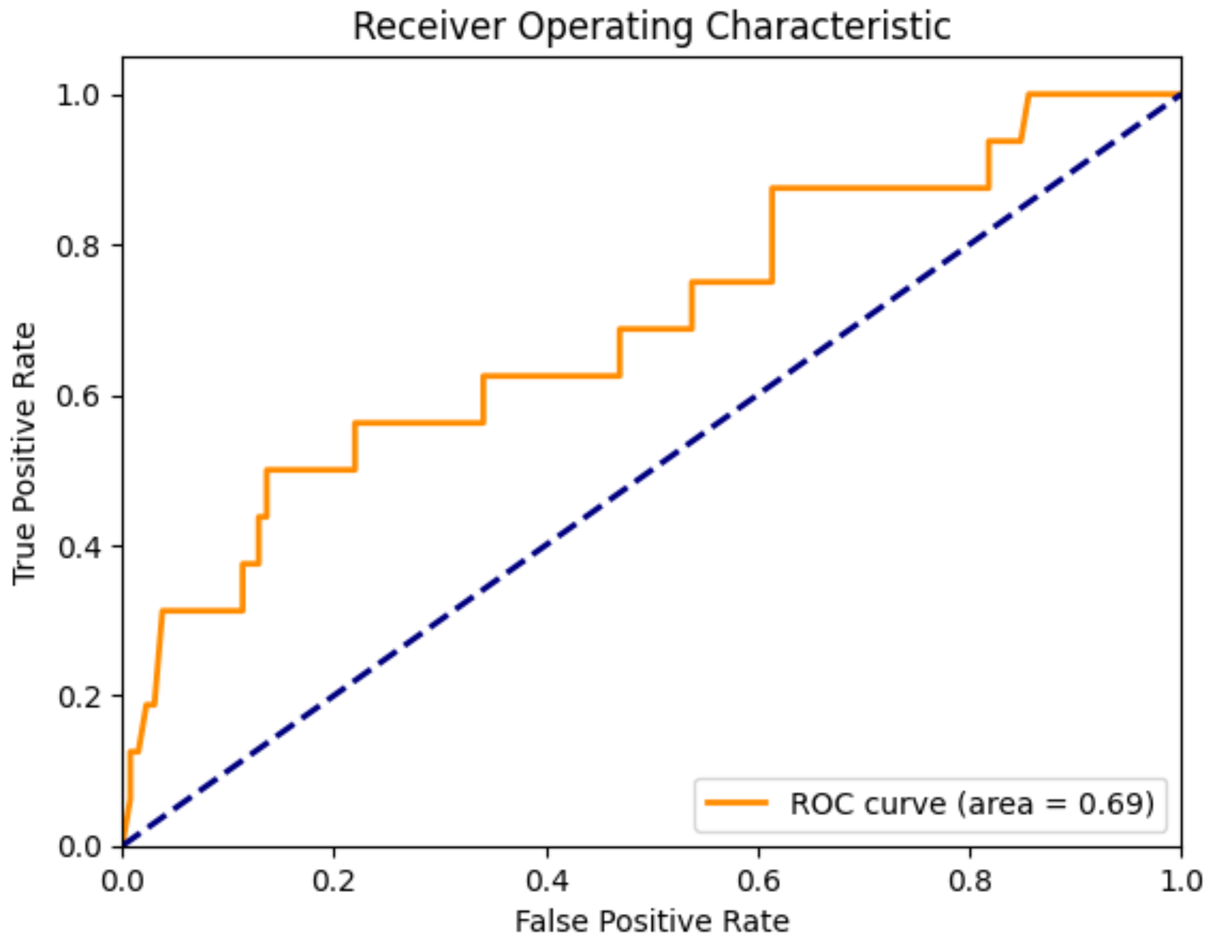




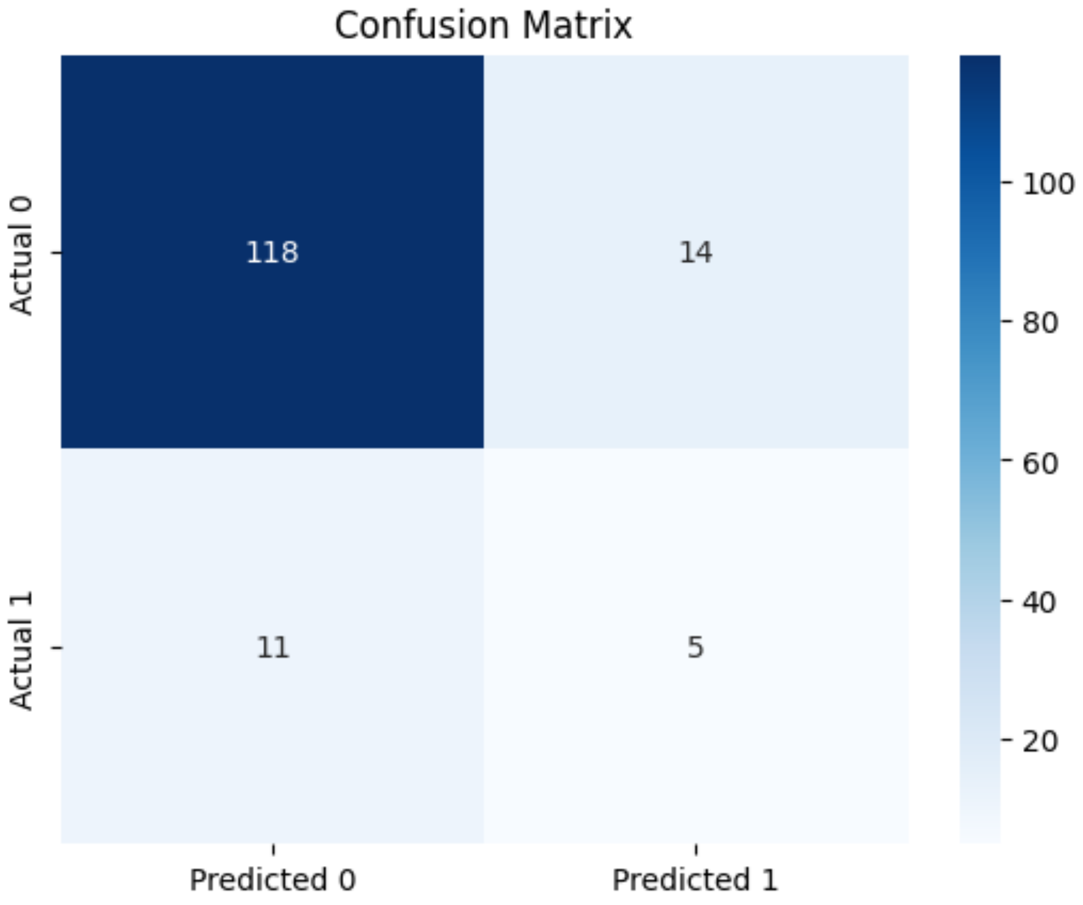
```
ROC Curve Metrics:
FPR: [0.         0.00757576 0.00757576 0.01515152 0.03787879 0.04545455
 0.06060606 0.06060606 0.06818182 0.06818182 0.08333333 0.09848485
 0.09848485 0.15151515 0.16666667 0.24242424 0.27272727 0.34090909
 0.34090909 0.36363636 0.38636364 0.38636364 0.48484848 0.5
 0.50757576 0.50757576 0.56060606 0.56060606 0.60606061 0.62121212
 0.64393939 0.66666667 0.72727273 0.74242424 0.86363636 0.86363636
 0.93181818 0.93939394 0.97727273 0.97727273 1.        ]
TPR: [0.     0.0625 0.125  0.25   0.25   0.3125 0.3125 0.375  0.375  0.4375
 0.4375 0.4375 0.5    0.5    0.5    0.5    0.5    0.5    0.5625 0.5625
 0.5625 0.625  0.625  0.625  0.625  0.75   0.75   0.8125 0.8125 0.8125
 0.8125 0.8125 0.8125 0.8125 0.8125 0.875  0.875  0.9375 0.9375 1.
 1.     ]
ROC AUC: 0.664

Running evaluation with seed 28...
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.py:87:
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step
Training - Accuracy: 0.5000, Sensitivity: 0.0000, Specificity: 1.0000, F1:
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
Threshold: 0.10, Accuracy: 0.1081, Sensitivity: 1.0000, Specificity: 0.0000
Threshold: 0.15, Accuracy: 0.4595, Sensitivity: 0.7500, Specificity: 0.4242
Threshold: 0.20, Accuracy: 0.5743, Sensitivity: 0.6250, Specificity: 0.5682
Threshold: 0.25, Accuracy: 0.6081, Sensitivity: 0.6250, Specificity: 0.6061
Threshold: 0.30, Accuracy: 0.6284, Sensitivity: 0.6250, Specificity: 0.6288
Threshold: 0.35, Accuracy: 0.6757, Sensitivity: 0.5625, Specificity: 0.6894
Threshold: 0.40, Accuracy: 0.7432, Sensitivity: 0.5625, Specificity: 0.7652
Threshold: 0.45, Accuracy: 0.8176, Sensitivity: 0.5000, Specificity: 0.8561
Threshold: 0.50, Accuracy: 0.8378, Sensitivity: 0.3125, Specificity: 0.9015
Threshold: 0.55, Accuracy: 0.8716, Sensitivity: 0.3125, Specificity: 0.9394
Threshold: 0.60, Accuracy: 0.8784, Sensitivity: 0.3125, Specificity: 0.9470
Threshold: 0.65, Accuracy: 0.8919, Sensitivity: 0.1250, Specificity: 0.9848
Threshold: 0.70, Accuracy: 0.8919, Sensitivity: 0.1250, Specificity: 0.9848
Threshold: 0.75, Accuracy: 0.8919, Sensitivity: 0.0000, Specificity: 1.0000
Threshold: 0.80, Accuracy: 0.8919, Sensitivity: 0.0000, Specificity: 1.0000
Threshold: 0.85, Accuracy: 0.8919, Sensitivity: 0.0000, Specificity: 1.0000
Threshold: 0.90, Accuracy: 0.8919, Sensitivity: 0.0000, Specificity: 1.0000
Threshold: 0.95, Accuracy: 0.8919, Sensitivity: 0.0000, Specificity: 1.0000
Threshold: 1.00, Accuracy: 0.8919, Sensitivity: 0.0000, Specificity: 1.0000
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step

Metrics for chosen threshold 0.45:
Accuracy: 0.8176, Sensitivity: 0.5000, Specificity: 0.8561, F1: 0.3721, ROC
```

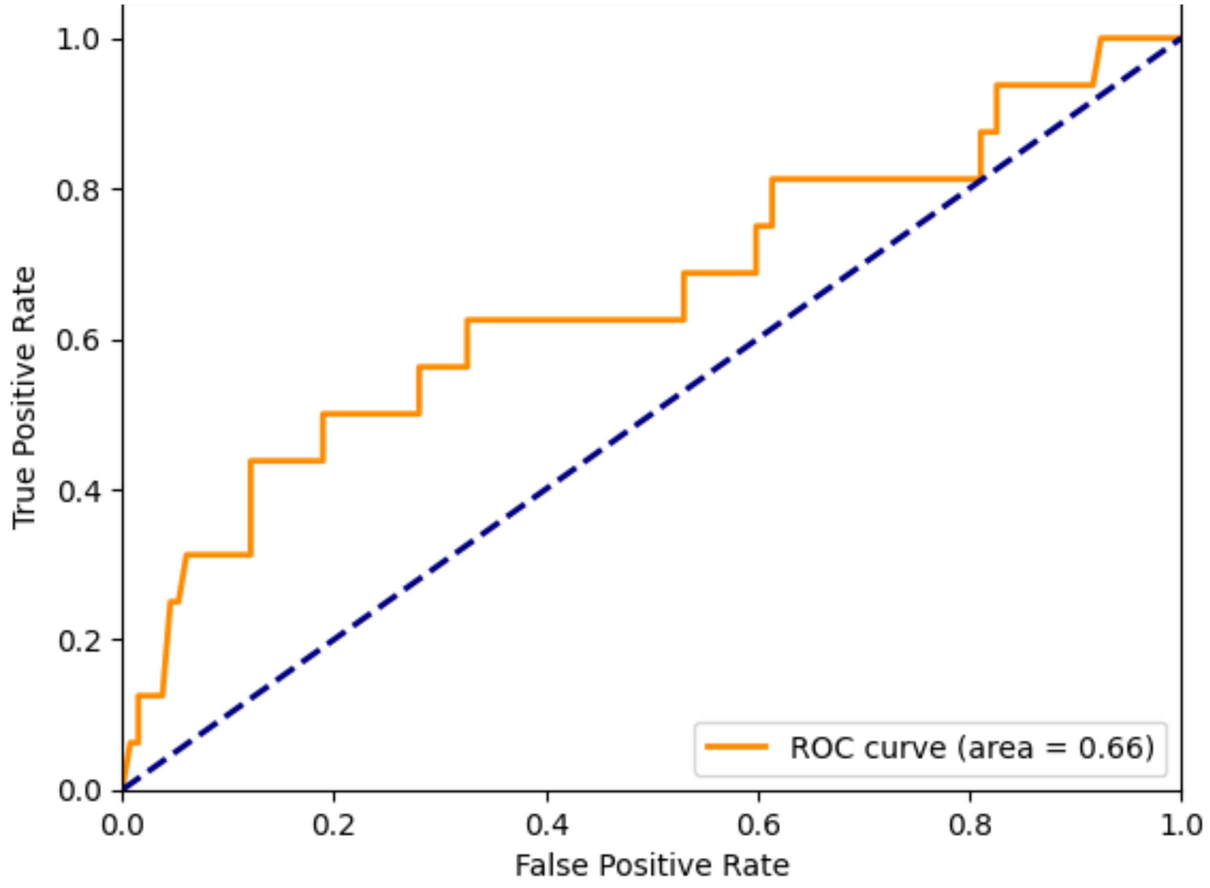


Actual 1
8
8
40
20
Predicted 0
Predicted 1



```
ROC Curve Metrics:
FPR: [0.         0.00757576 0.00757576 0.01515152 0.02272727 0.03030303
 0.03787879 0.11363636 0.11363636 0.12878788 0.12878788 0.13636364
 0.13636364 0.15909091 0.17424242 0.18939394 0.20454545 0.21969697
 0.21969697 0.25       0.26515152 0.32575758 0.34090909 0.34090909
 0.36363636 0.46969697 0.46969697 0.53787879 0.53787879 0.54545455
 0.57575758 0.59848485 0.61363636 0.61363636 0.71969697 0.73484848
 0.81818182 0.81818182 0.84848485 0.85606061 0.88636364 0.90151515
 1.        ]
TPR: [0.     0.0625 0.125  0.125  0.1875 0.1875 0.3125 0.3125 0.375  0.375
 0.4375 0.4375 0.5    0.5    0.5    0.5    0.5    0.5    0.5625 0.5625
 0.5625 0.5625 0.5625 0.625  0.625  0.625  0.6875 0.6875 0.75   0.75
 0.75   0.75   0.75   0.875  0.875  0.875  0.875  0.9375 0.9375 1.
 1.     1.     1.    ]
ROC AUC: 0.691

Running evaluation with seed 29...
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.py:87:
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step
Training - Accuracy: 0.5000, Sensitivity: 0.0000, Specificity: 1.0000, F1:
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
Threshold: 0.10, Accuracy: 0.1081, Sensitivity: 1.0000, Specificity: 0.0000
Threshold: 0.15, Accuracy: 0.1081, Sensitivity: 1.0000, Specificity: 0.0000
Threshold: 0.20, Accuracy: 0.3581, Sensitivity: 0.8125, Specificity: 0.3030
Threshold: 0.25, Accuracy: 0.5405, Sensitivity: 0.6250, Specificity: 0.5303
Threshold: 0.30, Accuracy: 0.6824, Sensitivity: 0.5625, Specificity: 0.6970
Threshold: 0.35, Accuracy: 0.7432, Sensitivity: 0.5000, Specificity: 0.7727
Threshold: 0.40, Accuracy: 0.8041, Sensitivity: 0.4375, Specificity: 0.8485
Threshold: 0.45, Accuracy: 0.8311, Sensitivity: 0.3125, Specificity: 0.8939
Threshold: 0.50, Accuracy: 0.8716, Sensitivity: 0.3125, Specificity: 0.9394
Threshold: 0.55, Accuracy: 0.8716, Sensitivity: 0.3125, Specificity: 0.9394
Threshold: 0.60, Accuracy: 0.8784, Sensitivity: 0.1250, Specificity: 0.9697
Threshold: 0.65, Accuracy: 0.8919, Sensitivity: 0.1250, Specificity: 0.9848
Threshold: 0.70, Accuracy: 0.8919, Sensitivity: 0.0000, Specificity: 1.0000
Threshold: 0.75, Accuracy: 0.8919, Sensitivity: 0.0000, Specificity: 1.0000
Threshold: 0.80, Accuracy: 0.8919, Sensitivity: 0.0000, Specificity: 1.0000
Threshold: 0.85, Accuracy: 0.8919, Sensitivity: 0.0000, Specificity: 1.0000
Threshold: 0.90, Accuracy: 0.8919, Sensitivity: 0.0000, Specificity: 1.0000
Threshold: 0.95, Accuracy: 0.8919, Sensitivity: 0.0000, Specificity: 1.0000
Threshold: 1.00, Accuracy: 0.8919, Sensitivity: 0.0000, Specificity: 1.0000
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step

Metrics for chosen threshold 0.45:
Accuracy: 0.8311, Sensitivity: 0.3125, Specificity: 0.8939, F1: 0.2857, ROC
```





```
ROC Curve Metrics:
FPR: [0.         0.00757576 0.01515152 0.01515152 0.03787879 0.04545455
 0.0530303  0.06060606 0.12121212 0.12121212 0.18939394 0.18939394
 0.20454545 0.21969697 0.26515152 0.28030303 0.28030303 0.31060606
 0.32575758 0.32575758 0.36363636 0.38636364 0.42424242 0.4469697
 0.53030303 0.53030303 0.54545455 0.59848485 0.59848485 0.61363636
 0.61363636 0.65909091 0.67424242 0.70454545 0.71969697 0.76515152
 0.78030303 0.81060606 0.81060606 0.82575758 0.82575758 0.91666667
 0.92424242 1.        ]
TPR: [0.     0.0625 0.0625 0.125  0.125  0.25   0.25   0.3125 0.3125 0.4375
 0.4375 0.5    0.5    0.5    0.5    0.5    0.5625 0.5625 0.5625 0.625
 0.625  0.625  0.625  0.625  0.625  0.6875 0.6875 0.6875 0.75   0.75
 0.8125 0.8125 0.8125 0.8125 0.8125 0.8125 0.8125 0.8125 0.875  0.875
 0.9375 0.9375 1.     1.    ]
ROC AUC: 0.656

Aggregated Test Set Metrics Across Seeds:
   accuracy  sensitivity  specificity        f1   roc_auc
0  0.824324       0.4375     0.871212  0.350000  0.654593
1  0.824324       0.3750     0.878788  0.315789  0.638021
2  0.844595       0.3750     0.901515  0.342857  0.699574
3  0.837838       0.3750     0.893939  0.333333  0.683002
4  0.837838       0.3125     0.901515  0.294118  0.626184
5  0.797297       0.4375     0.840909  0.318182  0.671165
6  0.797297       0.4375     0.840909  0.318182  0.678741
7  0.858108       0.5000     0.901515  0.432432  0.663589
8  0.817568       0.5000     0.856061  0.372093  0.691051
9  0.831081       0.3125     0.893939  0.285714  0.656487
```

```
Summary of Test Set Metrics (Mean, Standard Error, 95% Confidence Interval)
Accuracy: Mean = 0.827, SE = 0.006, 95% CI = [0.813, 0.841]
Sensitivity: Mean = 0.406, SE = 0.021, 95% CI = [0.358, 0.455]
Specificity: Mean = 0.878, SE = 0.008, 95% CI = [0.860, 0.896]
F1: Mean = 0.336, SE = 0.013, 95% CI = [0.306, 0.367]
Roc_auc: Mean = 0.666, SE = 0.007, 95% CI = [0.650, 0.683]
```

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