```
# Import necessary libraries
import numpy as np
import pandas as pd
from keras.models import Sequential
from keras.layers import Dense, Input, BatchNormalization, Dropout
from keras.callbacks import EarlyStopping, ModelCheckpoint
from keras.optimizers import Adam
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import MinMaxScaler, RobustScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, f1_score, roc_auc_score,
classification_report
from scikeras.wrappers import KerasClassifier
from tensorflow.keras.models import load_model

np.random.seed(7)

# Load dataset
dataframe = pd.read_csv("./datasets/pca_95_cls.csv", sep=',')
X = dataframe.iloc[:, :-1]  # Selecting all columns except the last
one as input features
y = dataframe['priceUSD']   # Target variable

dataframe.head(3)

          0         1         2         3         4         5
6  \
0  0.074162  0.015329 -0.048046  0.042709  0.007321 -0.014251
0.001355
1  0.094841  0.072671 -0.077840 -0.014523  0.027039 -0.053013
0.056817
2  0.064880  0.028643 -0.038454  0.019065  0.028725 -0.014173 -
0.002313

          7         8         9  ...        41        42        43
44  \
0 -0.044263 -0.014403 -0.036199  ...  0.017701 -0.020600 -0.021125 -
0.001148
1 -0.009060  0.047423 -0.009912  ... -0.047544  0.013065  0.065670
0.006482
2 -0.031474 -0.009467 -0.034115  ...  0.020285  0.006481 -0.012896
0.008115

         45        46        47        48        49  priceUSD
0 -0.004502 -0.012360 -0.032049  0.007081  0.006557         1
1  0.020321  0.007130  0.016320  0.013705 -0.042491         1
2 -0.022120 -0.021993  0.012241  0.021045 -0.033730         1

[3 rows x 51 columns]

dataframe.shape
```

```
(735, 51)

length=dataframe.shape[1]-1

length

50

# split into input (X) and output (Y) variables
X = dataframe.iloc[:,0:length]
y = dataframe['priceUSD']

X.head(3)

          0         1         2         3         4         5
6  \
0  0.074162  0.015329 -0.048046  0.042709  0.007321 -0.014251
0.001355
1  0.094841  0.072671 -0.077840 -0.014523  0.027039 -0.053013
0.056817
2  0.064880  0.028643 -0.038454  0.019065  0.028725 -0.014173 -
0.002313

          7         8         9  ...        40        41        42
43  \
0 -0.044263 -0.014403 -0.036199  ... -0.004087  0.017701 -0.020600 -
0.021125
1 -0.009060  0.047423 -0.009912  ...  0.003421 -0.047544  0.013065
0.065670
2 -0.031474 -0.009467 -0.034115  ...  0.014521  0.020285  0.006481 -
0.012896

         44        45        46        47        48        49
0 -0.001148 -0.004502 -0.012360 -0.032049  0.007081  0.006557
1  0.006482  0.020321  0.007130  0.016320  0.013705 -0.042491
2  0.008115 -0.022120 -0.021993  0.012241  0.021045 -0.033730

[3 rows x 50 columns]

y=np.ravel(y)

y

array([1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0,
1,
       1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0,
0,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1,
0,
       0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0,
0,
       0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0,
```

```
1,
        1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0,
0,
        1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1,
1,
        0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0,
1,
        1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1,
        1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1,
        1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1,
1,
        1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1,
0,
        1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1,
0,
        0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0,
0,
        1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0,
0,
        1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0,
1,
        0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0,
0,
        0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1,
0,
        0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1,
0,
        1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0,
0,
        1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1,
0,
        0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0,
0,
        0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1,
1,
        1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1,
0,
        0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0,
0,
        0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0,
0,
        1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0,
0,
        1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1,
0,
        0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0,
0,
```

```
       1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0,
1,
       1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1,
0,
       1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0,
1,
       1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0,
1,
       0, 1, 1, 0, 1, 1, 0, 1, 1], dtype=int64)
```

```python
shape=X.shape[1]



# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=7)

estimators=[]

estimators.append(('robust', RobustScaler()))

estimators.append(('minmax', MinMaxScaler()))

scale = Pipeline(estimators, verbose=True)

scale.fit(X_train)
```

```
[Pipeline] ............ (step 1 of 2) Processing robust, total=   0.0s
[Pipeline] ............ (step 2 of 2) Processing minmax, total=   0.0s
```

```
Pipeline(steps=[('robust', RobustScaler()), ('minmax',
MinMaxScaler())],
         verbose=True)
```

```python
X_train = scale.transform(X_train)



X_test = scale.transform(X_test)

# Learning Rate Scheduler
def lr_schedule(epoch):
    """Learning Rate Schedule with updates at specific epoch
milestones"""
    lr = 1e-3
    if epoch > 180:
        lr *= 0.5e-3
    elif epoch > 160:
        lr *= 1e-3
    elif epoch > 120:
        lr *= 1e-2
    elif epoch > 80:
```

```
        lr *= 1e-1
    print('Learning rate:', lr)
    return lr

# Define the upgraded model architecture
def sequential_model(initializer='he_normal', activation='relu',
neurons=300, NUM_FEATURES=X_train.shape[1]):
    model = Sequential()
    model.add(Input(shape=(NUM_FEATURES,)))  # Input layer
    model.add(Dense(512, kernel_initializer=initializer,
activation=activation))
    model.add(BatchNormalization())            # Batch normalization
for stability
    model.add(Dropout(0.3))                    # Dropout layer for
regularization

    model.add(Dense(256, kernel_initializer=initializer,
activation=activation))
    model.add(BatchNormalization())
    model.add(Dropout(0.3))

    model.add(Dense(128, kernel_initializer=initializer,
activation=activation))
    model.add(Dense(1, activation='sigmoid'))  # Output layer for
binary classification

    # Compile the model with Adam optimizer and dynamic learning rate
    adam = Adam(learning_rate=lr_schedule(0), amsgrad=True)
    model.compile(loss='binary_crossentropy', optimizer=adam,
metrics=['accuracy'])
    return model



# Configure Model Checkpoint and Early Stopping callbacks
mcp_save =
ModelCheckpoint('trained_models/ANN_cls_interval3_pca_upgraded.keras',

                           save_best_only=True, monitor='val_loss',
mode='min')
early_stopping = EarlyStopping(monitor='val_loss', patience=100,
verbose=1, mode='min')

# Initialize the KerasClassifier without `use_multiprocessing`
classifier = KerasClassifier(
    build_fn=sequential_model,
    batch_size=32,
    epochs=1000,
    validation_split=0.1,
    shuffle=True,
```

```
    callbacks=[mcp_save, early_stopping]
)



# Train the model
classifier.fit(X_train, y_train)
```

```
C:\Users\vanda\anaconda3\Lib\site-packages\scikeras\wrappers.py:925:
UserWarning: ``build_fn`` will be renamed to ``model`` in a future
release, at which point use of ``build_fn`` will raise an Error
instead.
  X, y = self._initialize(X, y)

Learning rate: 0.001
Epoch 1/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 5s 32ms/step - accuracy: 0.5112 - loss:
0.8765 - val_accuracy: 0.4915 - val_loss: 0.6877
Epoch 2/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.5653 - loss:
0.7640 - val_accuracy: 0.5254 - val_loss: 0.6848
Epoch 3/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.5932 - loss:
0.6575 - val_accuracy: 0.5424 - val_loss: 0.6873
Epoch 4/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.6440 - loss:
0.6326 - val_accuracy: 0.5932 - val_loss: 0.6926
Epoch 5/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.6938 - loss:
0.6086 - val_accuracy: 0.5424 - val_loss: 0.7002
Epoch 6/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.6890 - loss:
0.5742 - val_accuracy: 0.5763 - val_loss: 0.6940
Epoch 7/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.7019 - loss:
0.5947 - val_accuracy: 0.5593 - val_loss: 0.6898
Epoch 8/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.7454 - loss:
0.5416 - val_accuracy: 0.5424 - val_loss: 0.6791
Epoch 9/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.6973 - loss:
0.5465 - val_accuracy: 0.5593 - val_loss: 0.6773
Epoch 10/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.7052 - loss:
0.5498 - val_accuracy: 0.5763 - val_loss: 0.6692
Epoch 11/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.6937 - loss:
0.5500 - val_accuracy: 0.5593 - val_loss: 0.6653
Epoch 12/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7201 - loss:
```

```
0.5445 - val_accuracy: 0.6102 - val_loss: 0.6295
Epoch 13/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7019 - loss:
0.5336 - val_accuracy: 0.5932 - val_loss: 0.6251
Epoch 14/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.7499 - loss:
0.5143 - val_accuracy: 0.5763 - val_loss: 0.6433
Epoch 15/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7413 - loss:
0.5139 - val_accuracy: 0.6441 - val_loss: 0.6139
Epoch 16/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7072 - loss:
0.5244 - val_accuracy: 0.6780 - val_loss: 0.6026
Epoch 17/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.7645 - loss:
0.4783 - val_accuracy: 0.6441 - val_loss: 0.6161
Epoch 18/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.7607 - loss:
0.5094 - val_accuracy: 0.6780 - val_loss: 0.6025
Epoch 19/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.7649 - loss:
0.4927 - val_accuracy: 0.6780 - val_loss: 0.6073
Epoch 20/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.7670 - loss:
0.4861 - val_accuracy: 0.6441 - val_loss: 0.6443
Epoch 21/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.7681 - loss:
0.4749 - val_accuracy: 0.6610 - val_loss: 0.6260
Epoch 22/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.7490 - loss:
0.4948 - val_accuracy: 0.6949 - val_loss: 0.6347
Epoch 23/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.7710 - loss:
0.4660 - val_accuracy: 0.6271 - val_loss: 0.6397
Epoch 24/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.7291 - loss:
0.4793 - val_accuracy: 0.6780 - val_loss: 0.6351
Epoch 25/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.7703 - loss:
0.4605 - val_accuracy: 0.6780 - val_loss: 0.6167
Epoch 26/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.7744 - loss:
0.4597 - val_accuracy: 0.6949 - val_loss: 0.6076
Epoch 27/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.7770 - loss:
0.4505 - val_accuracy: 0.6610 - val_loss: 0.6425
Epoch 28/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.7642 - loss:
0.4647 - val_accuracy: 0.6949 - val_loss: 0.6225
```

```
Epoch 29/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.7422 - loss:
0.4913 - val_accuracy: 0.6441 - val_loss: 0.6033
Epoch 30/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.7799 - loss:
0.4773 - val_accuracy: 0.6949 - val_loss: 0.6458
Epoch 31/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.7320 - loss:
0.4976 - val_accuracy: 0.6780 - val_loss: 0.6587
Epoch 32/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.7836 - loss:
0.4228 - val_accuracy: 0.7119 - val_loss: 0.7098
Epoch 33/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.7480 - loss:
0.4547 - val_accuracy: 0.6610 - val_loss: 0.7094
Epoch 34/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.7712 - loss:
0.4677 - val_accuracy: 0.7119 - val_loss: 0.7019
Epoch 35/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.7572 - loss:
0.4381 - val_accuracy: 0.6949 - val_loss: 0.7418
Epoch 36/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8105 - loss:
0.4092 - val_accuracy: 0.6780 - val_loss: 0.6817
Epoch 37/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.7657 - loss:
0.4923 - val_accuracy: 0.6441 - val_loss: 0.7196
Epoch 38/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.7807 - loss:
0.4727 - val_accuracy: 0.6949 - val_loss: 0.6773
Epoch 39/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.7952 - loss:
0.4344 - val_accuracy: 0.7119 - val_loss: 0.6443
Epoch 40/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.7756 - loss:
0.4561 - val_accuracy: 0.6949 - val_loss: 0.6338
Epoch 41/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.7887 - loss:
0.4169 - val_accuracy: 0.7119 - val_loss: 0.6488
Epoch 42/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.8099 - loss:
0.4303 - val_accuracy: 0.6780 - val_loss: 0.6743
Epoch 43/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.7676 - loss:
0.4414 - val_accuracy: 0.6610 - val_loss: 0.7213
Epoch 44/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.7976 - loss:
0.4402 - val_accuracy: 0.6610 - val_loss: 0.7168
Epoch 45/1000
```

```
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.7768 - loss:
0.4486 - val_accuracy: 0.6780 - val_loss: 0.7537
Epoch 46/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.8047 - loss:
0.3988 - val_accuracy: 0.6949 - val_loss: 0.7104
Epoch 47/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.7666 - loss:
0.4645 - val_accuracy: 0.7119 - val_loss: 0.6836
Epoch 48/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8034 - loss:
0.4090 - val_accuracy: 0.7119 - val_loss: 0.7214
Epoch 49/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.7866 - loss:
0.4072 - val_accuracy: 0.7288 - val_loss: 0.7217
Epoch 50/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.7994 - loss:
0.3946 - val_accuracy: 0.6780 - val_loss: 0.7501
Epoch 51/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.8034 - loss:
0.4034 - val_accuracy: 0.6949 - val_loss: 0.7481
Epoch 52/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.8251 - loss:
0.3986 - val_accuracy: 0.6610 - val_loss: 0.7565
Epoch 53/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.8024 - loss:
0.4000 - val_accuracy: 0.6780 - val_loss: 0.7960
Epoch 54/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8135 - loss:
0.4179 - val_accuracy: 0.6780 - val_loss: 0.8103
Epoch 55/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.8065 - loss:
0.3863 - val_accuracy: 0.6610 - val_loss: 0.7842
Epoch 56/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.8474 - loss:
0.3600 - val_accuracy: 0.6780 - val_loss: 0.8024
Epoch 57/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.7945 - loss:
0.3995 - val_accuracy: 0.6610 - val_loss: 0.8183
Epoch 58/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.7933 - loss:
0.3940 - val_accuracy: 0.6949 - val_loss: 0.7589
Epoch 59/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8129 - loss:
0.3851 - val_accuracy: 0.6949 - val_loss: 0.7550
Epoch 60/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.7959 - loss:
0.4142 - val_accuracy: 0.6780 - val_loss: 0.8083
Epoch 61/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.7755 - loss:
```

```
0.4275 - val_accuracy: 0.7119 - val_loss: 0.7930
Epoch 62/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.7902 - loss:
0.3866 - val_accuracy: 0.6780 - val_loss: 0.8007
Epoch 63/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.8734 - loss:
0.3167 - val_accuracy: 0.6780 - val_loss: 0.8228
Epoch 64/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.7950 - loss:
0.3724 - val_accuracy: 0.6610 - val_loss: 0.8251
Epoch 65/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - accuracy: 0.7914 - loss:
0.4267 - val_accuracy: 0.7119 - val_loss: 0.7911
Epoch 66/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8138 - loss:
0.4132 - val_accuracy: 0.7288 - val_loss: 0.8401
Epoch 67/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8141 - loss:
0.3878 - val_accuracy: 0.6949 - val_loss: 0.7999
Epoch 68/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8292 - loss:
0.3918 - val_accuracy: 0.7288 - val_loss: 0.8111
Epoch 69/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.7970 - loss:
0.3922 - val_accuracy: 0.7119 - val_loss: 0.8202
Epoch 70/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8176 - loss:
0.3819 - val_accuracy: 0.6949 - val_loss: 0.8413
Epoch 71/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.7912 - loss:
0.3995 - val_accuracy: 0.7119 - val_loss: 0.8461
Epoch 72/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8018 - loss:
0.3798 - val_accuracy: 0.7458 - val_loss: 0.8171
Epoch 73/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8153 - loss:
0.3729 - val_accuracy: 0.6610 - val_loss: 0.7781
Epoch 74/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.8420 - loss:
0.3366 - val_accuracy: 0.6441 - val_loss: 0.8235
Epoch 75/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8366 - loss:
0.3925 - val_accuracy: 0.6441 - val_loss: 0.7887
Epoch 76/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8216 - loss:
0.3820 - val_accuracy: 0.6780 - val_loss: 0.8305
Epoch 77/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8361 - loss:
0.3505 - val_accuracy: 0.6949 - val_loss: 0.8685
```

```
Epoch 78/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8291 - loss:
0.3873 - val_accuracy: 0.7119 - val_loss: 0.8332
Epoch 79/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8507 - loss:
0.3358 - val_accuracy: 0.7119 - val_loss: 0.7904
Epoch 80/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8145 - loss:
0.3732 - val_accuracy: 0.7627 - val_loss: 0.8061
Epoch 81/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8138 - loss:
0.3617 - val_accuracy: 0.7458 - val_loss: 0.8736
Epoch 82/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8351 - loss:
0.3718 - val_accuracy: 0.7119 - val_loss: 0.9242
Epoch 83/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8497 - loss:
0.3384 - val_accuracy: 0.7288 - val_loss: 0.8966
Epoch 84/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8450 - loss:
0.3407 - val_accuracy: 0.7288 - val_loss: 0.8941
Epoch 85/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.7965 - loss:
0.4210 - val_accuracy: 0.7119 - val_loss: 0.8984
Epoch 86/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.7926 - loss:
0.3771 - val_accuracy: 0.6949 - val_loss: 0.9507
Epoch 87/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8265 - loss:
0.3604 - val_accuracy: 0.6949 - val_loss: 0.8914
Epoch 88/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8410 - loss:
0.3270 - val_accuracy: 0.6780 - val_loss: 0.8989
Epoch 89/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8326 - loss:
0.3547 - val_accuracy: 0.6780 - val_loss: 0.8653
Epoch 90/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8533 - loss:
0.3623 - val_accuracy: 0.6610 - val_loss: 0.8421
Epoch 91/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.8185 - loss:
0.3953 - val_accuracy: 0.6780 - val_loss: 0.8616
Epoch 92/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8094 - loss:
0.3574 - val_accuracy: 0.6949 - val_loss: 0.9255
Epoch 93/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8414 - loss:
0.3311 - val_accuracy: 0.7288 - val_loss: 0.9109
Epoch 94/1000
```

```
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8242 - loss:
0.3310 - val_accuracy: 0.6780 - val_loss: 0.8331
Epoch 95/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8176 - loss:
0.3603 - val_accuracy: 0.6780 - val_loss: 0.7972
Epoch 96/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8347 - loss:
0.3500 - val_accuracy: 0.7119 - val_loss: 0.8890
Epoch 97/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - accuracy: 0.8249 - loss:
0.3564 - val_accuracy: 0.6780 - val_loss: 0.9382
Epoch 98/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8500 - loss:
0.2996 - val_accuracy: 0.6441 - val_loss: 0.8888
Epoch 99/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8296 - loss:
0.3496 - val_accuracy: 0.6780 - val_loss: 0.8679
Epoch 100/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8526 - loss:
0.3124 - val_accuracy: 0.6610 - val_loss: 0.8514
Epoch 101/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8546 - loss:
0.3106 - val_accuracy: 0.6610 - val_loss: 0.8144
Epoch 102/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.8036 - loss:
0.3613 - val_accuracy: 0.6780 - val_loss: 0.8451
Epoch 103/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8512 - loss:
0.3140 - val_accuracy: 0.7288 - val_loss: 0.8124
Epoch 104/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8599 - loss:
0.3317 - val_accuracy: 0.6610 - val_loss: 0.8488
Epoch 105/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8036 - loss:
0.3728 - val_accuracy: 0.6780 - val_loss: 0.8040
Epoch 106/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.8254 - loss:
0.3601 - val_accuracy: 0.6610 - val_loss: 0.8167
Epoch 107/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.8517 - loss:
0.3483 - val_accuracy: 0.6780 - val_loss: 0.8808
Epoch 108/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8289 - loss:
0.3480 - val_accuracy: 0.6949 - val_loss: 0.9290
Epoch 109/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8548 - loss:
0.3323 - val_accuracy: 0.6271 - val_loss: 0.8519
Epoch 110/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8482 - loss:
```

```
0.3388 - val_accuracy: 0.6441 - val_loss: 0.9083
Epoch 111/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8502 - loss:
0.3332 - val_accuracy: 0.6949 - val_loss: 0.9100
Epoch 112/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8644 - loss:
0.3409 - val_accuracy: 0.6949 - val_loss: 0.9662
Epoch 113/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8098 - loss:
0.3470 - val_accuracy: 0.6949 - val_loss: 0.9292
Epoch 114/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8102 - loss:
0.3470 - val_accuracy: 0.6780 - val_loss: 0.9605
Epoch 115/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.8257 - loss:
0.3411 - val_accuracy: 0.6780 - val_loss: 0.9495
Epoch 116/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8440 - loss:
0.3197 - val_accuracy: 0.6441 - val_loss: 0.9455
Epoch 117/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.8250 - loss:
0.3491 - val_accuracy: 0.6271 - val_loss: 0.9177
Epoch 118/1000
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.8251 - loss:
0.3462 - val_accuracy: 0.6441 - val_loss: 0.9456
Epoch 118: early stopping

KerasClassifier(
	model=None
	build_fn=<function sequential_model at 0x000002378B71C9A0>
	warm_start=False
	random_state=None
	optimizer=rmsprop
	loss=None
	metrics=None
	batch_size=32
	validation_batch_size=None
	verbose=1
	callbacks=[<keras.src.callbacks.model_checkpoint.ModelCheckpoint
object at 0x0000023789B75AF0>,
<keras.src.callbacks.early_stopping.EarlyStopping object at
0x00000237FB849250>]
	validation_split=0.1
	shuffle=True
	run_eagerly=False
	epochs=1000
	class_weight=None
)
```

```python
# Load the best model for evaluation
prediction_model =
load_model('trained_models/ANN_cls_interval3_pca_upgraded.keras',
compile=False)

# Predict and evaluate the model
y_pred = (prediction_model.predict(X_test) > 0.5).astype("int32")
print("Accuracy:", accuracy_score(y_test, y_pred))
print("F1 Score:", f1_score(y_test, y_pred, average='weighted'))
print("ROC AUC Score:", roc_auc_score(y_test, y_pred))
print(classification_report(y_test, y_pred, target_names=['Class 0',
'Class 1']))
```

```
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
Accuracy: 0.5510204081632653
F1 Score: 0.5465662455458373
ROC AUC Score: 0.5433730454207
              precision    recall  f1-score   support

     Class 0       0.52      0.44      0.48        68
     Class 1       0.57      0.65      0.61        79

    accuracy                           0.55       147
   macro avg       0.55      0.54      0.54       147
weighted avg       0.55      0.55      0.55       147
```

```python
y_prob=[prediction_model.predict(X_test).max() for i in
range(len(y_test))]
```

```
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 629us/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 0s/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 0s/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
```

```
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 0s/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 0s/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 0s/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 0s/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 50us/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
```

```
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 0s/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 0s/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 0s/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 0s/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 595us/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 0s/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 0s/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 980us/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 477us/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
```

```
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 0s/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 0s/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 0s/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 0s/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 476us/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 887us/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 0s/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 0s/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 0s/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
```

```python
# Optional: Print out the first few predictions alongside actual values for verification
predictions_df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred.flatten()})

predictions_df
```

```
     Actual  Predicted
0         1          0
1         0          1
2         1          0
3         0          1
4         1          1
..      ...        ...
142       1          1
143       0          0
144       1          0
145       0          1
146       1          0
```

```
[147 rows x 2 columns]
```