In [1]:
```
!pip install pyswarms
```

```
Collecting pyswarms
  Downloading pyswarms-1.3.0-py2.py3-none-any.whl.metadata (33 kB)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packag
es (from pyswarms) (1.13.1)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packag
es (from pyswarms) (1.26.4)
Requirement already satisfied: matplotlib>=1.3.1 in /usr/local/lib/python3.1
0/dist-packages (from pyswarms) (3.7.5)
Requirement already satisfied: attrs in /usr/local/lib/python3.10/dist-packag
es (from pyswarms) (24.3.0)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-package
s (from pyswarms) (4.67.1)
Requirement already satisfied: future in /usr/local/lib/python3.10/dist-packa
ges (from pyswarms) (1.0.0)
Requirement already satisfied: pyyaml in /usr/local/lib/python3.10/dist-packa
ges (from pyswarms) (6.0.2)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/
dist-packages (from matplotlib>=1.3.1->pyswarms) (1.3.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist
-packages (from matplotlib>=1.3.1->pyswarms) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.1
0/dist-packages (from matplotlib>=1.3.1->pyswarms) (4.55.3)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.1
0/dist-packages (from matplotlib>=1.3.1->pyswarms) (1.4.7)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/d
ist-packages (from matplotlib>=1.3.1->pyswarms) (24.2)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dis
t-packages (from matplotlib>=1.3.1->pyswarms) (11.0.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/
dist-packages (from matplotlib>=1.3.1->pyswarms) (3.2.0)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python
3.10/dist-packages (from matplotlib>=1.3.1->pyswarms) (2.8.2)
Requirement already satisfied: mkl_fft in /usr/local/lib/python3.10/dist-pack
ages (from numpy->pyswarms) (1.3.8)
Requirement already satisfied: mkl_random in /usr/local/lib/python3.10/dist-p
ackages (from numpy->pyswarms) (1.2.4)
Requirement already satisfied: mkl_umath in /usr/local/lib/python3.10/dist-pa
ckages (from numpy->pyswarms) (0.1.1)
Requirement already satisfied: mkl in /usr/local/lib/python3.10/dist-packages
(from numpy->pyswarms) (2025.0.1)
Requirement already satisfied: tbb4py in /usr/local/lib/python3.10/dist-packa
ges (from numpy->pyswarms) (2022.0.0)
Requirement already satisfied: mkl-service in /usr/local/lib/python3.10/dist-
packages (from numpy->pyswarms) (2.4.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-pac
kages (from python-dateutil>=2.7->matplotlib>=1.3.1->pyswarms) (1.17.0)
Requirement already satisfied: intel-openmp>=2024 in /usr/local/lib/python3.1
0/dist-packages (from mkl->numpy->pyswarms) (2024.2.0)
Requirement already satisfied: tbb==2022.* in /usr/local/lib/python3.10/dist-
packages (from mkl->numpy->pyswarms) (2022.0.0)
Requirement already satisfied: tcmlib==1.* in /usr/local/lib/python3.10/dist-
packages (from tbb==2022.*->mkl->numpy->pyswarms) (1.2.0)
Requirement already satisfied: intel-cmplr-lib-rt in /usr/local/lib/python3.1
0/dist-packages (from mkl_umath->numpy->pyswarms) (2024.2.0)
Requirement already satisfied: intel-cmplr-lib-ur==2024.2.0 in /usr/local/li
b/python3.10/dist-packages (from intel-openmp>=2024->mkl->numpy->pyswarms) (2
024.2.0)
```

```
Downloading pyswarms-1.3.0-py2.py3-none-any.whl (104 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 104.1/104.1 kB 3.0 MB/s eta 0:00:
00
Installing collected packages: pyswarms
Successfully installed pyswarms-1.3.0
```

In [2]:

```python
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.impute import SimpleImputer
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout
import pyswarms as ps
from pyswarms.utils.functions.single_obj import sphere
```

In [3]:
```
df = pd.read_csv("/kaggle/input/cloud-computing-performance-metrics/vmCloud_dat

df.head()
```

```
/usr/local/lib/python3.10/dist-packages/pandas/io/formats/format.py:1458: Run
timeWarning: invalid value encountered in greater
  has_large_values = (abs_vals > 1e6).any()
/usr/local/lib/python3.10/dist-packages/pandas/io/formats/format.py:1459: Run
timeWarning: invalid value encountered in less
  has_small_values = ((abs_vals < 10 ** (-self.digits)) & (abs_vals > 0)).any
()
/usr/local/lib/python3.10/dist-packages/pandas/io/formats/format.py:1459: Run
timeWarning: invalid value encountered in greater
  has_small_values = ((abs_vals < 10 ** (-self.digits)) & (abs_vals > 0)).any
()
```

Out[3]:

	vm_id	timestamp	cpu_usage	memory_usage	network_traffic	power_consumption	nu
0	c5215826-6237-4a33-9312-72c1df909881	2023-01-25 09:10:54	54.881350	78.950861	164.775973	287.808986	
1	29690bc6-1f34-403b-b509-a1ecb1834fb8	2023-01-26 04:46:34	71.518937	29.901883	NaN	362.273569	
2	2e55abc3-5bad-46cb-b445-a577f5e9bf2a	2023-01-13 23:39:47	NaN	92.709195	203.674847	231.467903	
3	e672e32f-c134-4fbc-992b-34eb63bef6bf	2023-02-09 11:45:49	54.488318	88.100960	NaN	195.639954	
4	f38b8b50-6926-4533-be4f-89ad11624071	2023-06-14 08:27:26	42.365480	NaN	NaN	359.451537	

In [4]:
```
df.columns
```

Out[4]:
```
Index(['vm_id', 'timestamp', 'cpu_usage', 'memory_usage', 'network_traffic',
       'power_consumption', 'num_executed_instructions', 'execution_time',
       'energy_efficiency', 'task_type', 'task_priority', 'task_status'],
      dtype='object')
```

In [5]:
```python
# Data Cleaning
def clean_data(df):
    # Drop irrelevant columns
    df = df.drop(columns=['vm_id','timestamp'])


    # Handle missing values
    # Numerical columns: impute with median
    num_cols = df.select_dtypes(include=np.number).columns
    num_imputer = SimpleImputer(strategy='median')
    df[num_cols] = num_imputer.fit_transform(df[num_cols])

    # Categorical columns: impute with mode
    cat_cols = df.select_dtypes(include='object').columns.drop('task_status')
    cat_imputer = SimpleImputer(strategy='most_frequent')
    df[cat_cols] = cat_imputer.fit_transform(df[cat_cols])

    # Remove duplicates
    df = df.drop_duplicates()

    return df

cleaned_df = clean_data(df)
cleaned_df.head()
```

Out[5]:

	cpu_usage	memory_usage	network_traffic	power_consumption	num_executed_instructions	e
0	54.881350	78.950861	164.775973	287.808986	7527.0	
1	71.518937	29.901883	500.007595	362.273569	5348.0	
2	50.054758	92.709195	203.674847	231.467903	5483.0	
3	54.488318	88.100960	500.007595	195.639954	5876.0	
4	42.365480	49.976089	500.007595	359.451537	3361.0	

In [6]:
```python
cleaned_df['energy_efficiency'] = cleaned_df['energy_efficiency'] * 100

cleaned_df.head()
```

Out[6]:

	cpu_usage	memory_usage	network_traffic	power_consumption	num_executed_instructions	e
0	54.881350	78.950861	164.775973	287.808986	7527.0	
1	71.518937	29.901883	500.007595	362.273569	5348.0	
2	50.054758	92.709195	203.674847	231.467903	5483.0	
3	54.488318	88.100960	500.007595	195.639954	5876.0	
4	42.365480	49.976089	500.007595	359.451537	3361.0	

In [7]:
```
cleaned_df = cleaned_df.drop(columns=['num_executed_instructions'])
cleaned_df = cleaned_df.drop(columns=['task_type','task_priority'])

cleaned_df.head()
```

Out[7]:

	cpu_usage	memory_usage	network_traffic	power_consumption	execution_time	energy_efficie
0	54.881350	78.950861	164.775973	287.808986	69.345575	55.358
1	71.518937	29.901883	500.007595	362.273569	41.396040	34.985
2	50.054758	92.709195	203.674847	231.467903	24.602549	79.627
3	54.488318	88.100960	500.007595	195.639954	16.456670	52.951
4	42.365480	49.976089	500.007595	359.451537	55.307992	35.190

In [8]:
```
# Preprocessing
def preprocess_data(df, target_column='task_status'):
    # Convert categorical columns
    le = LabelEncoder()
    for col in df.select_dtypes(include='object').columns:
        if col != target_column:
            df[col] = le.fit_transform(df[col])

    # Encode target variable
    y = le.fit_transform(df[target_column])
    X = df.drop(columns=[target_column])

    # Split data
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ra

    # Scale features
    scaler = StandardScaler()
    X_train = scaler.fit_transform(X_train)
    X_test = scaler.transform(X_test)

    return X_train, X_test, y_train, y_test, X.shape[1]

X_train, X_test, y_train, y_test, n_features = preprocess_data(cleaned_df)
```

In [9]:
```
# LSTM Model Preparation
def reshape_for_lstm(X_train, X_test):
    X_train = X_train.reshape((X_train.shape[0], 1, X_train.shape[1]))
    X_test = X_test.reshape((X_test.shape[0], 1, X_test.shape[1]))
    return X_train, X_test

X_train_lstm, X_test_lstm = reshape_for_lstm(X_train, X_test)
```

In [10]:
```
# Modified LSTM Model
def create_lstm_model(input_shape):
    model = Sequential()
    model.add(LSTM(128, input_shape=input_shape, return_sequences=True))
    model.add(Dropout(0.3))
    model.add(LSTM(64))
    model.add(Dropout(0.2))
    model.add(Dense(1, activation='sigmoid'))

    model.compile(optimizer='adam',
                  loss='binary_crossentropy',
                  metrics=['accuracy'])
    return model

lstm_model = create_lstm_model((X_train_lstm.shape[1], X_train_lstm.shape[2]))
history = lstm_model.fit(X_train_lstm, y_train,
                        epochs=100,
                        batch_size=64,
                        validation_split=0.2,
                        verbose=1)
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204: User
Warning: Do not pass an `input_shape`/`input_dim` argument to a layer. When u
sing Sequential models, prefer using an `Input(shape)` object as the first la
yer in the model instead.
  super().__init__(**kwargs)
```

```
Epoch 1/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 98s 5ms/step - accuracy: 0.3005 - loss: -45.
4481 - val_accuracy: 0.3003 - val_loss: -169.3988
Epoch 2/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 91s 5ms/step - accuracy: 0.3000 - loss: -21
2.3144 - val_accuracy: 0.3003 - val_loss: -334.4058
Epoch 3/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 92s 5ms/step - accuracy: 0.2991 - loss: -37
6.1619 - val_accuracy: 0.3003 - val_loss: -499.4401
Epoch 4/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 93s 5ms/step - accuracy: 0.3004 - loss: -53
7.1557 - val_accuracy: 0.3003 - val_loss: -664.0661
Epoch 5/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 93s 5ms/step - accuracy: 0.3009 - loss: -70
3.2362 - val_accuracy: 0.3003 - val_loss: -829.0229
Epoch 6/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 93s 5ms/step - accuracy: 0.3001 - loss: -87
7.1121 - val_accuracy: 0.3003 - val_loss: -994.2122
Epoch 7/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 93s 5ms/step - accuracy: 0.3003 - loss: -103
9.0278 - val_accuracy: 0.3003 - val_loss: -1159.2346
Epoch 8/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 93s 5ms/step - accuracy: 0.3007 - loss: -119
6.1326 - val_accuracy: 0.3003 - val_loss: -1324.3523
Epoch 9/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 93s 5ms/step - accuracy: 0.3001 - loss: -136
3.2198 - val_accuracy: 0.3003 - val_loss: -1489.3951
Epoch 10/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 90s 4ms/step - accuracy: 0.3002 - loss: -153
9.2845 - val_accuracy: 0.3003 - val_loss: -1654.7260
Epoch 11/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 92s 5ms/step - accuracy: 0.2994 - loss: -170
4.0316 - val_accuracy: 0.3003 - val_loss: -1819.4688
Epoch 12/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 95s 5ms/step - accuracy: 0.2997 - loss: -185
4.2036 - val_accuracy: 0.3003 - val_loss: -1984.3694
Epoch 13/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 93s 5ms/step - accuracy: 0.3006 - loss: -204
1.4209 - val_accuracy: 0.3003 - val_loss: -2149.3433
Epoch 14/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 94s 5ms/step - accuracy: 0.3010 - loss: -217
6.5046 - val_accuracy: 0.3003 - val_loss: -2314.0405
Epoch 15/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 92s 5ms/step - accuracy: 0.3006 - loss: -237
0.1675 - val_accuracy: 0.3003 - val_loss: -2479.5442
Epoch 16/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 93s 5ms/step - accuracy: 0.2998 - loss: -253
1.2576 - val_accuracy: 0.3003 - val_loss: -2644.7976
Epoch 17/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 93s 5ms/step - accuracy: 0.3003 - loss: -267
3.4370 - val_accuracy: 0.3003 - val_loss: -2809.6111
Epoch 18/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 93s 5ms/step - accuracy: 0.3002 - loss: -286
3.1562 - val_accuracy: 0.3003 - val_loss: -2974.2383
Epoch 19/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 93s 5ms/step - accuracy: 0.3004 - loss: -303
7.1790 - val_accuracy: 0.3003 - val_loss: -3139.3425
```

```
Epoch 20/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 92s 5ms/step - accuracy: 0.3008 - loss: -319
5.5845 - val_accuracy: 0.3003 - val_loss: -3303.9326
Epoch 21/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 94s 5ms/step - accuracy: 0.3001 - loss: -335
1.1958 - val_accuracy: 0.3003 - val_loss: -3469.1233
Epoch 22/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 93s 5ms/step - accuracy: 0.2995 - loss: -350
4.3052 - val_accuracy: 0.3003 - val_loss: -3633.9912
Epoch 23/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 94s 5ms/step - accuracy: 0.3000 - loss: -366
1.3347 - val_accuracy: 0.3003 - val_loss: -3798.9973
Epoch 24/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 94s 5ms/step - accuracy: 0.3001 - loss: -385
5.6519 - val_accuracy: 0.3003 - val_loss: -3964.1521
Epoch 25/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 93s 5ms/step - accuracy: 0.3001 - loss: -397
3.1541 - val_accuracy: 0.3003 - val_loss: -4129.8057
Epoch 26/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 93s 5ms/step - accuracy: 0.3003 - loss: -417
2.5654 - val_accuracy: 0.3003 - val_loss: -4294.6499
Epoch 27/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 95s 5ms/step - accuracy: 0.3000 - loss: -433
7.1606 - val_accuracy: 0.3003 - val_loss: -4460.0664
Epoch 28/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 94s 5ms/step - accuracy: 0.2998 - loss: -449
4.8921 - val_accuracy: 0.3003 - val_loss: -4625.1934
Epoch 29/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 93s 5ms/step - accuracy: 0.3004 - loss: -466
2.3638 - val_accuracy: 0.3003 - val_loss: -4789.9917
Epoch 30/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 94s 5ms/step - accuracy: 0.3000 - loss: -483
5.4370 - val_accuracy: 0.3003 - val_loss: -4954.8945
Epoch 31/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 94s 5ms/step - accuracy: 0.3007 - loss: -497
4.2505 - val_accuracy: 0.3003 - val_loss: -5120.0552
Epoch 32/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 93s 5ms/step - accuracy: 0.3000 - loss: -518
3.2358 - val_accuracy: 0.3003 - val_loss: -5285.1758
Epoch 33/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 91s 5ms/step - accuracy: 0.3009 - loss: -532
2.3862 - val_accuracy: 0.3003 - val_loss: -5450.6353
Epoch 34/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 93s 5ms/step - accuracy: 0.2998 - loss: -551
8.8599 - val_accuracy: 0.3003 - val_loss: -5615.5864
Epoch 35/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 94s 5ms/step - accuracy: 0.3003 - loss: -566
4.8721 - val_accuracy: 0.3003 - val_loss: -5780.6079
Epoch 36/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 94s 5ms/step - accuracy: 0.2997 - loss: -577
8.2007 - val_accuracy: 0.3003 - val_loss: -5945.6440
Epoch 37/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 93s 5ms/step - accuracy: 0.3001 - loss: -596
7.8750 - val_accuracy: 0.3003 - val_loss: -6110.9546
Epoch 38/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 93s 5ms/step - accuracy: 0.3002 - loss: -615
9.5088 - val_accuracy: 0.3003 - val_loss: -6275.9272
```

```
Epoch 39/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 93s 5ms/step - accuracy: 0.3001 - loss: -630
2.2627 - val_accuracy: 0.3003 - val_loss: -6440.9131
Epoch 40/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 93s 5ms/step - accuracy: 0.3001 - loss: -649
9.1436 - val_accuracy: 0.3003 - val_loss: -6605.8013
Epoch 41/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 94s 5ms/step - accuracy: 0.3001 - loss: -664
5.9312 - val_accuracy: 0.3003 - val_loss: -6770.9639
Epoch 42/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 95s 5ms/step - accuracy: 0.3002 - loss: -681
0.0952 - val_accuracy: 0.3003 - val_loss: -6935.9785
Epoch 43/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 89s 4ms/step - accuracy: 0.3005 - loss: -698
5.1895 - val_accuracy: 0.3003 - val_loss: -7101.1919
Epoch 44/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 93s 5ms/step - accuracy: 0.3000 - loss: -717
6.4360 - val_accuracy: 0.3003 - val_loss: -7266.4370
Epoch 45/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 94s 5ms/step - accuracy: 0.2998 - loss: -736
9.0454 - val_accuracy: 0.3003 - val_loss: -7431.7881
Epoch 46/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 94s 5ms/step - accuracy: 0.3003 - loss: -749
3.7734 - val_accuracy: 0.3003 - val_loss: -7596.9082
Epoch 47/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 94s 5ms/step - accuracy: 0.3007 - loss: -763
6.3833 - val_accuracy: 0.3003 - val_loss: -7761.9414
Epoch 48/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 93s 5ms/step - accuracy: 0.2999 - loss: -779
7.5337 - val_accuracy: 0.3003 - val_loss: -7926.8418
Epoch 49/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 94s 5ms/step - accuracy: 0.3003 - loss: -797
0.2515 - val_accuracy: 0.3003 - val_loss: -8092.0176
Epoch 50/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 94s 5ms/step - accuracy: 0.3008 - loss: -820
8.6680 - val_accuracy: 0.3003 - val_loss: -8256.9736
Epoch 51/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 94s 5ms/step - accuracy: 0.3004 - loss: -829
1.4365 - val_accuracy: 0.3003 - val_loss: -8421.8174
Epoch 52/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 93s 5ms/step - accuracy: 0.3006 - loss: -846
9.5645 - val_accuracy: 0.3003 - val_loss: -8586.8711
Epoch 53/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 93s 5ms/step - accuracy: 0.3003 - loss: -868
4.1660 - val_accuracy: 0.3003 - val_loss: -8751.6699
Epoch 54/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 94s 5ms/step - accuracy: 0.3006 - loss: -885
2.0303 - val_accuracy: 0.3003 - val_loss: -8916.9746
Epoch 55/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 93s 5ms/step - accuracy: 0.3011 - loss: -889
4.9189 - val_accuracy: 0.3003 - val_loss: -9082.0488
Epoch 56/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 93s 5ms/step - accuracy: 0.3000 - loss: -908
3.7988 - val_accuracy: 0.3003 - val_loss: -9247.0635
Epoch 57/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 95s 5ms/step - accuracy: 0.2999 - loss: -933
6.7842 - val_accuracy: 0.3003 - val_loss: -9412.1992
```

```
Epoch 58/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 93s 5ms/step - accuracy: 0.2999 - loss: -948
0.6455 - val_accuracy: 0.3003 - val_loss: -9577.1367
Epoch 59/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 93s 5ms/step - accuracy: 0.3009 - loss: -952
2.3262 - val_accuracy: 0.3003 - val_loss: -9742.0264
Epoch 60/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 93s 5ms/step - accuracy: 0.3006 - loss: -979
2.8857 - val_accuracy: 0.3003 - val_loss: -9907.0488
Epoch 61/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 94s 5ms/step - accuracy: 0.2999 - loss: -100
08.8535 - val_accuracy: 0.3003 - val_loss: -10071.7529
Epoch 62/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 94s 5ms/step - accuracy: 0.2997 - loss: -101
71.1797 - val_accuracy: 0.3003 - val_loss: -10237.0557
Epoch 63/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 93s 5ms/step - accuracy: 0.3002 - loss: -103
44.2168 - val_accuracy: 0.3003 - val_loss: -10402.5996
Epoch 64/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 94s 5ms/step - accuracy: 0.2999 - loss: -105
15.1396 - val_accuracy: 0.3003 - val_loss: -10567.4414
Epoch 65/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 93s 5ms/step - accuracy: 0.3001 - loss: -106
52.2275 - val_accuracy: 0.3003 - val_loss: -10732.3398
Epoch 66/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 93s 5ms/step - accuracy: 0.3002 - loss: -108
13.9814 - val_accuracy: 0.3003 - val_loss: -10897.1348
Epoch 67/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 94s 5ms/step - accuracy: 0.3003 - loss: -108
27.6826 - val_accuracy: 0.3003 - val_loss: -11062.1729
Epoch 68/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 94s 5ms/step - accuracy: 0.3001 - loss: -110
21.9756 - val_accuracy: 0.3003 - val_loss: -11227.4336
Epoch 69/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 94s 5ms/step - accuracy: 0.2997 - loss: -112
66.9443 - val_accuracy: 0.3003 - val_loss: -11391.9766
Epoch 70/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 94s 5ms/step - accuracy: 0.3003 - loss: -113
58.9268 - val_accuracy: 0.3003 - val_loss: -11557.6709
Epoch 71/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 94s 5ms/step - accuracy: 0.3003 - loss: -116
48.0186 - val_accuracy: 0.3003 - val_loss: -11722.6572
Epoch 72/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 93s 5ms/step - accuracy: 0.3000 - loss: -116
73.5957 - val_accuracy: 0.3003 - val_loss: -11887.8955
Epoch 73/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 93s 5ms/step - accuracy: 0.3003 - loss: -119
31.9023 - val_accuracy: 0.3003 - val_loss: -12052.8428
Epoch 74/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 94s 5ms/step - accuracy: 0.3000 - loss: -120
89.6729 - val_accuracy: 0.3003 - val_loss: -12217.7559
Epoch 75/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 94s 5ms/step - accuracy: 0.2994 - loss: -122
69.9434 - val_accuracy: 0.3003 - val_loss: -12382.8359
Epoch 76/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 90s 5ms/step - accuracy: 0.3000 - loss: -124
84.9658 - val_accuracy: 0.3003 - val_loss: -12548.0156
```

```
Epoch 77/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 93s 5ms/step - accuracy: 0.3002 - loss: -126
96.1221 - val_accuracy: 0.3003 - val_loss: -12712.9736
Epoch 78/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 94s 5ms/step - accuracy: 0.2996 - loss: -128
88.0186 - val_accuracy: 0.3003 - val_loss: -12877.7852
Epoch 79/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 94s 5ms/step - accuracy: 0.2999 - loss: -129
74.2559 - val_accuracy: 0.3003 - val_loss: -13042.9756
Epoch 80/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 94s 5ms/step - accuracy: 0.2999 - loss: -130
76.7490 - val_accuracy: 0.3003 - val_loss: -13207.7148
Epoch 81/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 94s 5ms/step - accuracy: 0.3004 - loss: -133
12.7148 - val_accuracy: 0.3003 - val_loss: -13372.5371
Epoch 82/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 94s 5ms/step - accuracy: 0.3003 - loss: -134
19.1875 - val_accuracy: 0.3003 - val_loss: -13537.8975
Epoch 83/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 93s 5ms/step - accuracy: 0.3002 - loss: -135
97.2510 - val_accuracy: 0.3003 - val_loss: -13702.7314
Epoch 84/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 94s 5ms/step - accuracy: 0.2997 - loss: -136
60.4014 - val_accuracy: 0.3003 - val_loss: -13867.3682
Epoch 85/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 94s 5ms/step - accuracy: 0.3003 - loss: -139
13.7246 - val_accuracy: 0.3003 - val_loss: -14032.7617
Epoch 86/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 92s 5ms/step - accuracy: 0.3008 - loss: -140
95.2500 - val_accuracy: 0.3003 - val_loss: -14197.9023
Epoch 87/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 94s 5ms/step - accuracy: 0.3004 - loss: -142
69.2256 - val_accuracy: 0.3003 - val_loss: -14363.0000
Epoch 88/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 95s 5ms/step - accuracy: 0.2994 - loss: -145
14.4854 - val_accuracy: 0.3003 - val_loss: -14527.8467
Epoch 89/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 94s 5ms/step - accuracy: 0.3003 - loss: -144
93.4053 - val_accuracy: 0.3003 - val_loss: -14693.2988
Epoch 90/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 94s 5ms/step - accuracy: 0.3000 - loss: -146
70.9443 - val_accuracy: 0.3003 - val_loss: -14858.3555
Epoch 91/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 94s 5ms/step - accuracy: 0.2998 - loss: -150
68.1494 - val_accuracy: 0.3003 - val_loss: -15023.7266
Epoch 92/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 94s 5ms/step - accuracy: 0.3005 - loss: -150
35.9111 - val_accuracy: 0.3003 - val_loss: -15189.2158
Epoch 93/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 93s 5ms/step - accuracy: 0.3000 - loss: -153
30.0898 - val_accuracy: 0.3003 - val_loss: -15354.0176
Epoch 94/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 94s 5ms/step - accuracy: 0.3004 - loss: -154
31.3125 - val_accuracy: 0.3003 - val_loss: -15518.9844
Epoch 95/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 94s 5ms/step - accuracy: 0.2999 - loss: -155
08.7783 - val_accuracy: 0.3003 - val_loss: -15684.1602
```

```
Epoch 96/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 93s 5ms/step - accuracy: 0.3004 - loss: -158
47.0137 - val_accuracy: 0.3003 - val_loss: -15848.6992
Epoch 97/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 94s 5ms/step - accuracy: 0.3006 - loss: -158
96.1357 - val_accuracy: 0.3003 - val_loss: -16013.6719
Epoch 98/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 94s 5ms/step - accuracy: 0.2994 - loss: -161
05.8740 - val_accuracy: 0.3003 - val_loss: -16179.2256
Epoch 99/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 93s 5ms/step - accuracy: 0.2996 - loss: -162
58.9873 - val_accuracy: 0.3003 - val_loss: -16344.1855
Epoch 100/100
20000/20000 ━━━━━━━━━━━━━━━━━━━━ 95s 5ms/step - accuracy: 0.3002 - loss: -164
49.7383 - val_accuracy: 0.3003 - val_loss: -16509.3164
```

```
In [11]: # LSTM Evaluation
lstm_pred = (lstm_model.predict(X_test_lstm) > 0.5).astype(int)
print("LSTM Classification Report:")
print(classification_report(y_test, lstm_pred))
print("Confusion Matrix:")
print(confusion_matrix(y_test, lstm_pred))
```

```
12500/12500 ━━━━━━━━━━━━━━━━━━━━ 19s 1ms/step
LSTM Classification Report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00    119845
           1       0.30      1.00      0.46    119917
           2       0.00      0.00      0.00    120400
           3       0.00      0.00      0.00     39838

    accuracy                           0.30    400000
   macro avg       0.07      0.25      0.12    400000
weighted avg       0.09      0.30      0.14    400000

Confusion Matrix:
[[     0 119845      0      0]
 [     0 119917      0      0]
 [     0 120400      0      0]
 [     0  39838      0      0]]

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:13
44: UndefinedMetricWarning: Precision and F-score are ill-defined and being s
et to 0.0 in labels with no predicted samples. Use `zero_division` parameter
to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:13
44: UndefinedMetricWarning: Precision and F-score are ill-defined and being s
et to 0.0 in labels with no predicted samples. Use `zero_division` parameter
to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:13
44: UndefinedMetricWarning: Precision and F-score are ill-defined and being s
et to 0.0 in labels with no predicted samples. Use `zero_division` parameter
to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```