# QNN

May 1, 2025

[1]:
```python
import pandas as pd
df1=pd.read_csv(r"D:\dataset\new_data.csv")
df1
```

[1]:
```
         Destination Port  Flow Duration  Total Length of Fwd Packets  \
0                   54865              3                           12
1                   55054            109                            6
2                   55055             52                            6
3                   46236             34                            6
4                   54863              3                           12
…                       …              …                            …
2520793                53          32215                          112
2520794                53            324                           84
2520795             58030             82                           31
2520796                53        1048635                          192
2520797                53          94939                          188

         Total Length of Bwd Packets  Fwd Packet Length Max  \
0                                  0                      6
1                                  6                      6
2                                  6                      6
3                                  6                      6
4                                  0                      6
…                                  …                      …
2520793                          152                     28
2520794                          362                     42
2520795                            6                     31
2520796                          256                     32
2520797                          226                     47

         Fwd Packet Length Min  Fwd Packet Length Mean  \
0                            6                     6.0
1                            6                     6.0
2                            6                     6.0
3                            6                     6.0
4                            6                     6.0
…                            …                       …
2520793                     28                    28.0
```

```
2520794                  42                        42.0
2520795                   0                        15.5
2520796                  32                        32.0
2520797                  47                        47.0

         Fwd Packet Length Std  Bwd Packet Length Max  \
0                      0.00000                      0
1                      0.00000                      6
2                      0.00000                      6
3                      0.00000                      6
4                      0.00000                      0
…                          …                    …
2520793                0.00000                     76
2520794                0.00000                    181
2520795               21.92031                      6
2520796                0.00000                    128
2520797                0.00000                    113

         Bwd Packet Length Min  …  Active Mean   Active Std   Active Max  \
0                            0  …          0.0          0.0            0
1                            6  …          0.0          0.0            0
2                            6  …          0.0          0.0            0
3                            6  …          0.0          0.0            0
4                            0  …          0.0          0.0            0
…                          …  …          …          …          …
2520793                     76  …          0.0          0.0            0
2520794                    181  …          0.0          0.0            0
2520795                      6  …          0.0          0.0            0
2520796                    128  …          0.0          0.0            0
2520797                    113  …          0.0          0.0            0

         Active Min  Idle Mean   Idle Std   Idle Max   Idle Min  Label  \
0                 0        0.0        0.0          0          0      1
1                 0        0.0        0.0          0          0      1
2                 0        0.0        0.0          0          0      1
3                 0        0.0        0.0          0          0      1
4                 0        0.0        0.0          0          0      1
…             …        …        …        …        …     …
2520793           0        0.0        0.0          0          0      1
2520794           0        0.0        0.0          0          0      1
2520795           0        0.0        0.0          0          0      1
2520796           0        0.0        0.0          0          0      1
2520797           0        0.0        0.0          0          0      1

        outlier
0             1
1             1
```

```
2              1
3              1
4              1
…            …
2520793        1
2520794        1
2520795        1
2520796        1
2520797        1

[2520798 rows x 62 columns]
```

[4]:
```python
#df1=df1.drop(columns=['outlier'])
df1
```

[4]:
```
          Destination Port   Flow Duration  Total Length of Fwd Packets  \
0                    54865               3                           12
1                    55054             109                            6
2                    55055              52                            6
3                    46236              34                            6
4                    54863               3                           12
…                    …             …                            …
2520793                 53           32215                          112
2520794                 53             324                           84
2520795              58030              82                           31
2520796                 53         1048635                          192
2520797                 53           94939                          188

          Total Length of Bwd Packets   Fwd Packet Length Max  \
0                                   0                       6
1                                   6                       6
2                                   6                       6
3                                   6                       6
4                                   0                       6
…                                 …                     …
2520793                           152                      28
2520794                           362                      42
2520795                             6                      31
2520796                           256                      32
2520797                           226                      47

          Fwd Packet Length Min   Fwd Packet Length Mean  \
0                             6                      6.0
1                             6                      6.0
2                             6                      6.0
3                             6                      6.0
4                             6                      6.0
```

```
…                        …                      …
2520793                 28                    28.0
2520794                 42                    42.0
2520795                  0                    15.5
2520796                 32                    32.0
2520797                 47                    47.0

          Fwd Packet Length Std  Bwd Packet Length Max  \
0                       0.00000                      0
1                       0.00000                      6
2                       0.00000                      6
3                       0.00000                      6
4                       0.00000                      0
…                        …                    …
2520793                 0.00000                     76
2520794                 0.00000                    181
2520795                21.92031                      6
2520796                 0.00000                    128
2520797                 0.00000                    113

          Bwd Packet Length Min  …   min_seg_size_forward  Active Mean  \
0                             0  …                     20          0.0
1                             6  …                     20          0.0
2                             6  …                     20          0.0
3                             6  …                     20          0.0
4                             0  …                     20          0.0
…                        …   …                   …          …
2520793                      76  …                     20          0.0
2520794                     181  …                     20          0.0
2520795                       6  …                     32          0.0
2520796                     128  …                     20          0.0
2520797                     113  …                     20          0.0

          Active Std   Active Max   Active Min  Idle Mean   Idle Std  \
0                0.0            0            0        0.0        0.0
1                0.0            0            0        0.0        0.0
2                0.0            0            0        0.0        0.0
3                0.0            0            0        0.0        0.0
4                0.0            0            0        0.0        0.0
…             …          …          …        …       …
2520793          0.0            0            0        0.0        0.0
2520794          0.0            0            0        0.0        0.0
2520795          0.0            0            0        0.0        0.0
2520796          0.0            0            0        0.0        0.0
2520797          0.0            0            0        0.0        0.0

          Idle Max   Idle Min  Label
```

```
0                 0          0      1
1                 0          0      1
2                 0          0      1
3                 0          0      1
4                 0          0      1
…             …          …     …
2520793           0          0      1
2520794           0          0      1
2520795           0          0      1
2520796           0          0      1
2520797           0          0      1

[2520798 rows x 61 columns]
```

[5]:
```python
X = df1.drop(columns=['Label'])
y = df1['Label']

# Increase range to ensure at least 5647 samples
X = X.iloc[30900:309300]
y = y.iloc[30900:309300]

print(X.shape[0])

sampled_indices = X.sample(n=300,random_state=42).index
X = X.loc[sampled_indices]
y = y.loc[sampled_indices]
X = df1.drop(columns=['Label'])
y = df1['Label']

X = X.iloc[30900:309300]
y = y.iloc[30900:309300]

print("Initial shape:", X.shape[0])

# Sample 13 points
sampled_indices = X.sample(n=300, random_state=42).index
X = X.loc[sampled_indices]
y = y.loc[sampled_indices]

# Get label distribution
print("Label distribution in sampled data:")
print(y.value_counts())
```

```
278400
Initial shape: 278400
Label distribution in sampled data:
Label
 1    182
```

```
-1    118
Name: count, dtype: int64
```

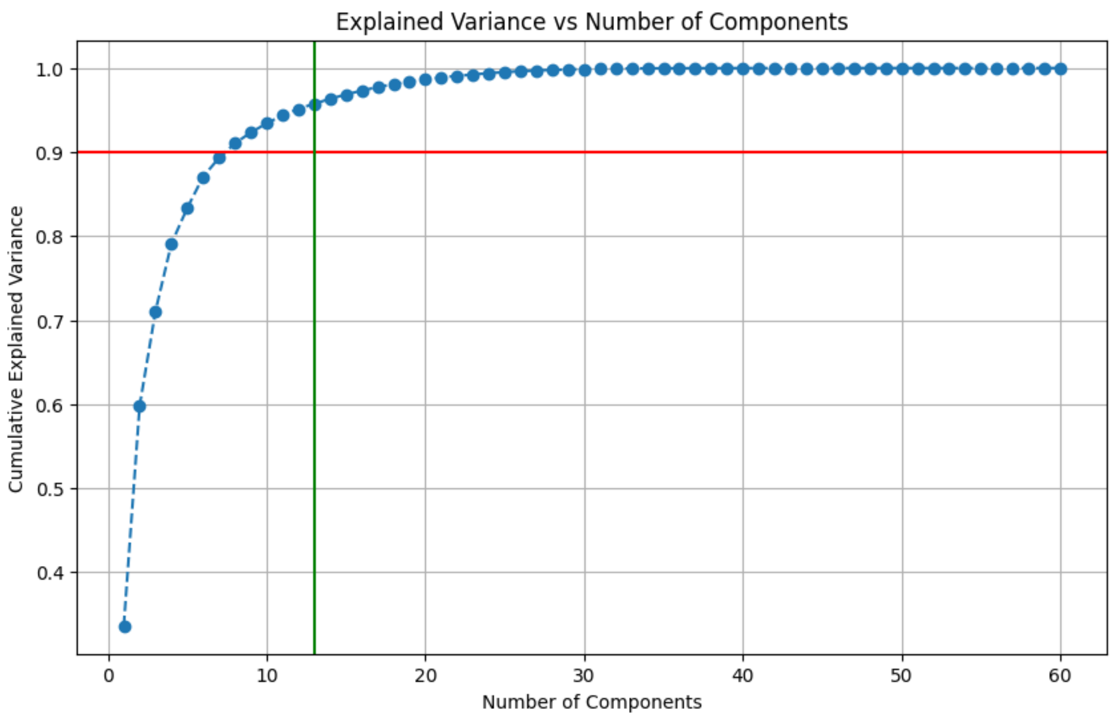
[6]:
```
# Sample 300 points
sampled_indices = X.sample(n=300, random_state=42).index
X = X.loc[sampled_indices]
y= y.loc[sampled_indices]

# Convert -1 to 0 for binary classification
Y= y.replace({-1: 0})

# Check label distribution
print("Label distribution in sampled data:")
print(Y.value_counts())
```

```
Label distribution in sampled data:
Label
1    182
0    118
Name: count, dtype: int64
```

[7]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.decomposition import PCA
scaler = MinMaxScaler(feature_range=(0, np.pi))
X = scaler.fit_transform(X)
# Standardize the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
pca = PCA(n_components=0.90)
X_pca = pca.fit_transform(X_scaled)

# Number of components selected
n_components_selected = pca.n_components_
print(f"Number of components to retain 90% variance: {n_components_selected}")

# Now let's plot how much variance each component explains

# If you want to see how each component contributes
pca_full = PCA()
pca_full.fit(X)
explained_variance_ratio = pca_full.explained_variance_ratio_
```

```python
# Cumulative variance
cumulative_variance = explained_variance_ratio.cumsum()

# Plot
plt.figure(figsize=(10,6))
plt.plot(range(1, len(cumulative_variance)+1), cumulative_variance, marker='o',␣
↪linestyle='--')
plt.axhline(y=0.90, color='r', linestyle='-')
plt.axvline(x=n_components_selected, color='g', linestyle='-')
plt.title('Explained Variance vs Number of Components')
plt.xlabel('Number of Components')
plt.ylabel('Cumulative Explained Variance')
plt.grid(True)
plt.show()
```

```
Number of components to retain 90% variance: 13
```




[8]:
```python
pca = PCA(n_components=13)
X_reduced = pca.fit_transform(X_scaled)
```

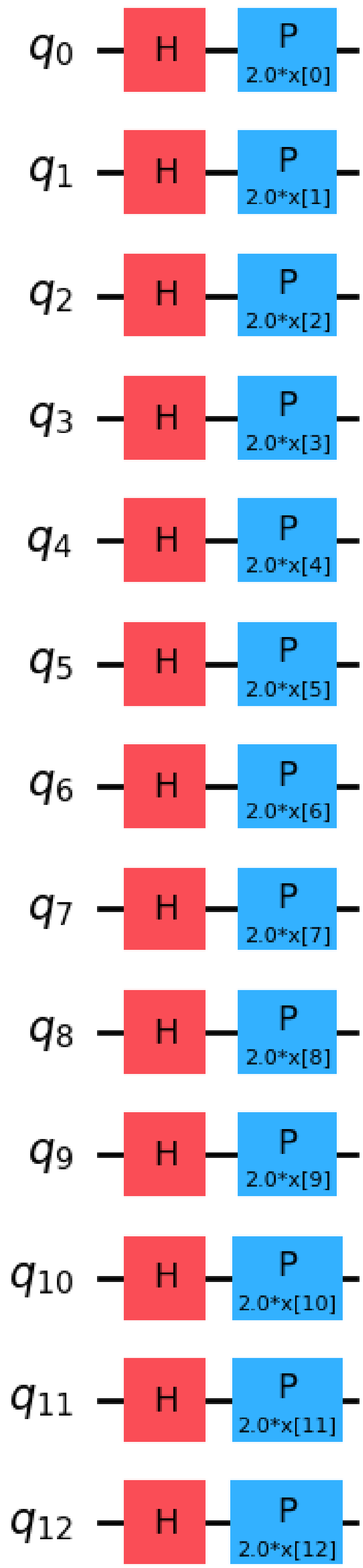
[9]:
```python
from qiskit.circuit.library import ZFeatureMap, ZZFeatureMap, PauliFeatureMap

#from qiskit_algorithms.utils import algorithm_globals
from qiskit_machine_learning.kernels import FidelityQuantumKernel
```

```
from qiskit_machine_learning.algorithms import PegasosQSVC
from qiskit_machine_learning.algorithms import QSVC
from qiskit.providers import BackendV2 as Backend
from qiskit.transpiler import Target
from qiskit.circuit.library import ZFeatureMap, ZZFeatureMap
# from qiskit_algorithms.utils import algorithm_globals
from qiskit_machine_learning.kernels import FidelityQuantumKernel
from qiskit_machine_learning.algorithms import PegasosQSVC
from qiskit_machine_learning.algorithms import QSVC
from qiskit.primitives import StatevectorSampler, Sampler
from qiskit_machine_learning.state_fidelities import ComputeUncompute
from qiskit_machine_learning.algorithms import QSVC# number of qubits is equal␣
↪to the number of features
num_qubits = 13
feature_map = ZFeatureMap(feature_dimension=num_qubits, reps=2)
sampler = Sampler()
fidelity = ComputeUncompute(sampler=sampler)
qkernel = FidelityQuantumKernel(fidelity=fidelity, feature_map=feature_map)
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_16096\3878349687.py:19:
DeprecationWarning: The class ``qiskit.primitives.sampler.Sampler`` is
deprecated as of qiskit 1.2. It will be removed no earlier than 3 months after
the release date. All implementations of the `BaseSamplerV1` interface have been
deprecated in favor of their V2 counterparts. The V2 alternative for the
`Sampler` class is `StatevectorSampler`.
  sampler = Sampler()
C:\Users\HP\AppData\Local\Temp\ipykernel_16096\3878349687.py:20:
DeprecationWarning: V1 Primitives are deprecated as of qiskit-machine-learning
0.8.0 and will be removed no sooner than 4 months after the release date. Use V2
primitives for continued compatibility and support.
  fidelity = ComputeUncompute(sampler=sampler)
```

[10]:
```
feature_map = ZFeatureMap(feature_dimension=num_qubits, reps=1)
#Decompose Circuit
feature_map.decompose().draw('mpl')
```

[10]:


q0
q1
q2
q3
q4
q5
q6
q7
q8
q9
q10
q11
q12
H
P
2.0*x[0]
2.0*x[1]
2.0*x[2]
2.0*x[3]
2.0*x[4]
2.0*x[5]
2.0*x[6]
2.0*x[7]
2.0*x[8]
2.0*x[9]
2.0*x[10]
2.0*x[11]
2.0*x[12]

[11]:
```
from qiskit_machine_learning.algorithms import QSVC
from qiskit.providers import BackendV2 as Backend
from qiskit.transpiler import Target
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report
import pandas as pd
from qiskit_aer import Aer
from qiskit.primitives import Sampler

from qiskit_aer import AerSimulator
from qiskit_ibm_runtime import QiskitRuntimeService, Sampler, Session
```

[13]:
```
backend = AerSimulator()
# Create sampler with the backend
from qiskit_ibm_runtime import SamplerV2 as Sampler

sampler = Sampler(backend)
# sampler= StatevectorSampler()
```

[14]:
```
print("Label counts:", Y.value_counts())
```

```
Label counts: Label
1    182
0    118
Name: count, dtype: int64
```

[15]:
```
# Imports (if not done already)
import torch
from torch import nn
from sklearn.model_selection import train_test_split
from qiskit.circuit.library import ZZFeatureMap, TwoLocal
from qiskit_machine_learning.neural_networks import EstimatorQNN
from qiskit_machine_learning.connectors import TorchConnector

#  Assumes: X and y already reduced, scaled, and cleaned
# Example: X_reduced = PCA(n_components=0.90).fit_transform(StandardScaler().
 ↪fit_transform(X))

# Split data
X_train, X_test, Y_train, Y_test = train_test_split(X_reduced, Y, test_size=0.
 ↪2, random_state=42)

# Convert to PyTorch tensors
X_train_tensor = torch.tensor(X_train, dtype=torch.float32)
y_train_tensor = torch.tensor(Y_train.values, dtype=torch.float32)
```

[22]:
```
unique_labels, counts = torch.unique(y_train_tensor, return_counts=True)
for label, count in zip(unique_labels, counts):
    print(f"Label {int(label.item())}: {int(count.item())} samples")
```

```
Label 0: 95 samples
Label 1: 145 samples
```

[16]:
```
# Define quantum feature map and ansatz
num_qubits = 13  # e.g., 13 if PCA gives 13 components
feature_map = ZZFeatureMap(feature_dimension=num_qubits, reps=2)
ansatz = TwoLocal(num_qubits=num_qubits, rotation_blocks='ry',␣
↪entanglement_blocks='cz')
```

[23]:
```
# Build EstimatorQNN
from qiskit import QuantumCircuit
from qiskit.circuit.library import ZZFeatureMap, RealAmplitudes
from qiskit_machine_learning.circuit.library import QNNCircuit

from qiskit_machine_learning.neural_networks import EstimatorQNN

qnn_qc = QNNCircuit(num_qubits)

qnn = EstimatorQNN(
    circuit=qnn_qc
)

qc = QuantumCircuit(num_qubits)
qc.compose(feature_map, inplace=True)
qc.compose(ansatz, inplace=True)

qnn = EstimatorQNN(
    circuit=qc,
    input_params=feature_map.parameters,
    weight_params=ansatz.parameters)
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_16096\3226022641.py:10:
DeprecationWarning: V1 Primitives are deprecated as of qiskit-machine-learning
0.8.0 and will be removed no sooner than 4 months after the release date. Use V2
primitives for continued compatibility and support.
  qnn = EstimatorQNN(
C:\Users\HP\AppData\Local\Temp\ipykernel_16096\3226022641.py:18:
DeprecationWarning: V1 Primitives are deprecated as of qiskit-machine-learning
0.8.0 and will be removed no sooner than 4 months after the release date. Use V2
primitives for continued compatibility and support.
  qnn = EstimatorQNN(
```

[24]:
```
# Wrap QNN as PyTorch model
model = TorchConnector(qnn)
```

```python
# Define loss and optimizer
criterion = nn.BCELoss()
optimizer = torch.optim.Adam(model.parameters(), lr=0.01)

# Training loop
epochs = 10
for epoch in range(epochs):
    optimizer.zero_grad()
    output = model(X_train_tensor).squeeze()  # ensure shape matches
    loss = criterion(output, y_train_tensor)
    loss.backward()
    optimizer.step()
    print(f"Epoch {epoch+1}/{epochs}, Loss: {loss.item():.4f}")
```

```
---------------------------------------------------------------------------
RuntimeError                              Traceback (most recent call last)
Cell In[24], line 13
     11 optimizer.zero_grad()
     12 output = model(X_train_tensor).squeeze()  # ensure shape matches
---> 13 loss = criterion(output, y_train_tensor)
     14 loss.backward()
     15 optimizer.step()

File c:\Users\HP\.conda\envs\QML\Lib\site-packages\torch\nn\modules\module.py:
↪1751, in Module._wrapped_call_impl(self, *args, **kwargs)
   1749     return self._compiled_call_impl(*args, **kwargs)  # type:␣
↪ignore[misc]
   1750 else:
-> 1751     return self._call_impl(*args, **kwargs)

File c:\Users\HP\.conda\envs\QML\Lib\site-packages\torch\nn\modules\module.py:
↪1762, in Module._call_impl(self, *args, **kwargs)
   1757 # If we don't have any hooks, we want to skip the rest of the logic in
   1758 # this function, and just call forward.
   1759 if not (self._backward_hooks or self._backward_pre_hooks or self.
↪_forward_hooks or self._forward_pre_hooks
   1760         or _global_backward_pre_hooks or _global_backward_hooks
   1761         or _global_forward_hooks or _global_forward_pre_hooks):
-> 1762     return forward_call(*args, **kwargs)
   1764 result = None
   1765 called_always_called_hooks = set()

File c:\Users\HP\.conda\envs\QML\Lib\site-packages\torch\nn\modules\loss.py:699␣
↪in BCELoss.forward(self, input, target)
    698 def forward(self, input: Tensor, target: Tensor) -> Tensor:
--> 699     return F.binary_cross_entropy(
```

```
    700         input, target, weight=self.weight, reduction=self.reduction
    701     )

File c:\Users\HP\.conda\envs\QML\Lib\site-packages\torch\nn\functional.py:3569,
 ↪in binary_cross_entropy(input, target, weight, size_average, reduce, reduction)
   3566     new_size = _infer_size(target.size(), weight.size())
   3567     weight = weight.expand(new_size)
-> 3569 return torch._C._nn.binary_cross_entropy(input, target, weight,
 ↪reduction_enum)

RuntimeError: all elements of input should be between 0 and 1
```

[26]:

```python
# Ensure y has only 0s and 1s
Y = y.apply(lambda v: 1 if v == 1 else 0)
Y
```

[26]:

```
205540    1
217555    1
243760    1
117487    0
287476    1
         ..
196590    1
147469    0
187926    0
153160    0
83273     0
Name: Label, Length: 300, dtype: int64
```

[33]:

```python
from sklearn.model_selection import train_test_split

X_train, X_test,Y_train, Y_test = train_test_split(X_reduced, Y, test_size=0.2,
 ↪random_state=42)
```

[38]:

```python
import torch

X_train_tensor = torch.tensor(X_train, dtype=torch.float32)
Y_train_tensor = torch.tensor(Y_train.values, dtype=torch.float32)

# If needed, clamp values to [0,1] to eliminate stray floats
Y_train_tensor = torch.clamp(Y_train_tensor, 0.0, 1.0)

# Print unique labels to be sure
print("Unique labels:", torch.unique(Y_train_tensor))
```

```
Unique labels: tensor([0., 1.])
```

[40]:
```python
# Train
```

[41]:
```python
import torch.nn.functional as F

# Training loop
epochs = 10
for epoch in range(epochs):
    optimizer.zero_grad()

    raw_output = model(X_train_tensor).squeeze()

    # Apply sigmoid to get values in [0, 1]
    output = torch.sigmoid(raw_output)

    # Match target shape
    y_target = Y_train_tensor.view_as(output)

    # Compute loss
    loss = criterion(output, y_target)
    loss.backward()
    optimizer.step()

    print(f"Epoch {epoch+1}/{epochs}, Loss: {loss.item():.4f}")
```

```
Epoch 1/10, Loss: 0.6925
Epoch 2/10, Loss: 0.6924
Epoch 3/10, Loss: 0.6922
Epoch 4/10, Loss: 0.6921
Epoch 5/10, Loss: 0.6919
Epoch 6/10, Loss: 0.6918
Epoch 7/10, Loss: 0.6917
Epoch 8/10, Loss: 0.6915
Epoch 9/10, Loss: 0.6914
Epoch 10/10, Loss: 0.6913
```

[42]:
```python
# Convert test data to tensors
X_test_tensor = torch.tensor(X_test, dtype=torch.float32)
Y_test_tensor = torch.tensor(Y_test.values, dtype=torch.float32)

# Clamp labels to ensure they are within [0, 1]
Y_test_tensor = torch.clamp(Y_test_tensor, 0.0, 1.0)
```

[43]:
```python
# Set model to evaluation mode
model.eval()

with torch.no_grad():
    # Forward pass
    raw_preds = model(X_test_tensor).squeeze()
```

```
    preds = torch.sigmoid(raw_preds)

    # Threshold to get binary predictions
    preds_class = (preds >= 0.5).float()

    # Match shape with ground truth
    y_true = Y_test_tensor.view_as(preds)
```

[44]:
```
from sklearn.metrics import accuracy_score, precision_score, recall_score,␣
↪f1_score

accuracy = accuracy_score(y_true, preds_class)
precision = precision_score(y_true, preds_class)
recall = recall_score(y_true, preds_class)
f1 = f1_score(y_true, preds_class)

print(f"Accuracy: {accuracy:.4f}")
print(f"Precision: {precision:.4f}")
print(f"Recall: {recall:.4f}")
print(f"F1-score: {f1:.4f}")
```

```
Accuracy: 0.5167
Precision: 0.6429
Recall: 0.4865
F1-score: 0.5538
```

[45]:
```
# Set model to evaluation mode
model.eval()

with torch.no_grad():
    # Training predictions
    raw_train_preds = model(X_train_tensor).squeeze()
    train_preds = torch.sigmoid(raw_train_preds)
    train_preds_class = (train_preds >= 0.5).float()

    y_train_true = Y_train_tensor.view_as(train_preds)

    # Convert to NumPy
    y_train_true_np = y_train_true.numpy()
    train_preds_class_np = train_preds_class.numpy()

    # Compute training accuracy
    from sklearn.metrics import accuracy_score
    train_accuracy = accuracy_score(y_train_true_np, train_preds_class_np)
    print(f"Training Accuracy: {train_accuracy:.4f}")
```

```
Training Accuracy: 0.6458
```

[46]:
```
# Test predictions
raw_test_preds = model(X_test_tensor).squeeze()
test_preds = torch.sigmoid(raw_test_preds)
test_preds_class = (test_preds >= 0.5).float()

y_test_true = Y_test_tensor.view_as(test_preds)

# Convert to NumPy
y_test_true_np = y_test_true.numpy()
test_preds_class_np = test_preds_class.numpy()

# Compute test accuracy
test_accuracy = accuracy_score(y_test_true_np, test_preds_class_np)
print(f"Testing Accuracy : {test_accuracy:.4f}")
```

```
Testing Accuracy : 0.5167
```

[47]:
```
import torch
import torch.nn as nn
import torch.optim as optim
from sklearn.metrics import accuracy_score
import numpy as np

# Define a simplified model with dropout and batch normalization
class Net(nn.Module):
    def __init__(self, input_dim):
        super(Net, self).__init__()
        self.fc1 = nn.Linear(input_dim, 64)
        self.bn1 = nn.BatchNorm1d(64)
        self.dropout1 = nn.Dropout(0.3)
        self.fc2 = nn.Linear(64, 32)
        self.bn2 = nn.BatchNorm1d(32)
        self.dropout2 = nn.Dropout(0.3)
        self.fc3 = nn.Linear(32, 1)

    def forward(self, x):
        x = torch.relu(self.bn1(self.fc1(x)))
        x = self.dropout1(x)
        x = torch.relu(self.bn2(self.fc2(x)))
        x = self.dropout2(x)
        x = self.fc3(x)
        return x

# Tensors from your preprocessed data
X_train_tensor = torch.tensor(X_train, dtype=torch.float32)
Y_train_tensor = torch.tensor(Y_train.values, dtype=torch.float32).view(-1, 1)
X_test_tensor = torch.tensor(X_test, dtype=torch.float32)
Y_test_tensor = torch.tensor(Y_test.values, dtype=torch.float32).view(-1, 1)
```

```python
# Initialize model
input_dim = X_train_tensor.shape[1]
model = Net(input_dim)

# Define loss and optimizer (with weight decay for L2 regularization)
criterion = nn.BCEWithLogitsLoss()
optimizer = optim.Adam(model.parameters(), lr=0.001, weight_decay=1e-4)  # L2 regularization

# Training loop
epochs = 20
for epoch in range(epochs):
    model.train()
    optimizer.zero_grad()

    output = model(X_train_tensor)
    loss = criterion(output, Y_train_tensor)
    loss.backward()
    optimizer.step()

    print(f"Epoch {epoch+1}/{epochs}, Loss: {loss.item():.4f}")

# Evaluation on training set
model.eval()
with torch.no_grad():
    train_output = torch.sigmoid(model(X_train_tensor)).squeeze()
    train_preds = (train_output >= 0.5).float()
    train_acc = accuracy_score(Y_train_tensor.numpy(), train_preds.numpy())
    print(f"Training Accuracy: {train_acc:.4f}")

    # Evaluation on test set
    test_output = torch.sigmoid(model(X_test_tensor)).squeeze()
    test_preds = (test_output >= 0.5).float()
    test_acc = accuracy_score(Y_test_tensor.numpy(), test_preds.numpy())
    print(f"Testing Accuracy: {test_acc:.4f}")
```

```
Epoch 1/20, Loss: 0.7985
Epoch 2/20, Loss: 0.7564
Epoch 3/20, Loss: 0.7441
Epoch 4/20, Loss: 0.7032
Epoch 5/20, Loss: 0.6897
Epoch 6/20, Loss: 0.6845
Epoch 7/20, Loss: 0.6519
Epoch 8/20, Loss: 0.6310
Epoch 9/20, Loss: 0.6099
Epoch 10/20, Loss: 0.6118
Epoch 11/20, Loss: 0.5788
```

```
Epoch 12/20, Loss: 0.5919
Epoch 13/20, Loss: 0.5679
Epoch 14/20, Loss: 0.5692
Epoch 15/20, Loss: 0.5505
Epoch 16/20, Loss: 0.5260
Epoch 17/20, Loss: 0.5194
Epoch 18/20, Loss: 0.5194
Epoch 19/20, Loss: 0.5040
Epoch 20/20, Loss: 0.4861
Training Accuracy: 0.9708
Testing Accuracy: 0.9333
```

[48]:
```python
from sklearn.metrics import classification_report, confusion_matrix

# Ensure model is in evaluation mode
model.eval()

with torch.no_grad():
    # Predictions on test set
    test_output = torch.sigmoid(model(X_test_tensor)).squeeze()
    test_preds = (test_output >= 0.5).float().numpy()
    y_true = Y_test_tensor.numpy()

    # Classification Report
    print("Classification Report:")
    print(classification_report(y_true, test_preds, target_names=["Normal",␣
 ↪"Anomaly"]))

    # Confusion Matrix
    cm = confusion_matrix(y_true, test_preds)
    print("Confusion Matrix:")
    print(cm)
```

```
Classification Report:
              precision    recall  f1-score   support

      Normal       0.85      1.00      0.92        23
     Anomaly       1.00      0.89      0.94        37

    accuracy                           0.93        60
   macro avg       0.93      0.95      0.93        60
weighted avg       0.94      0.93      0.93        60

Confusion Matrix:
[[23  0]
 [ 4 33]]
```