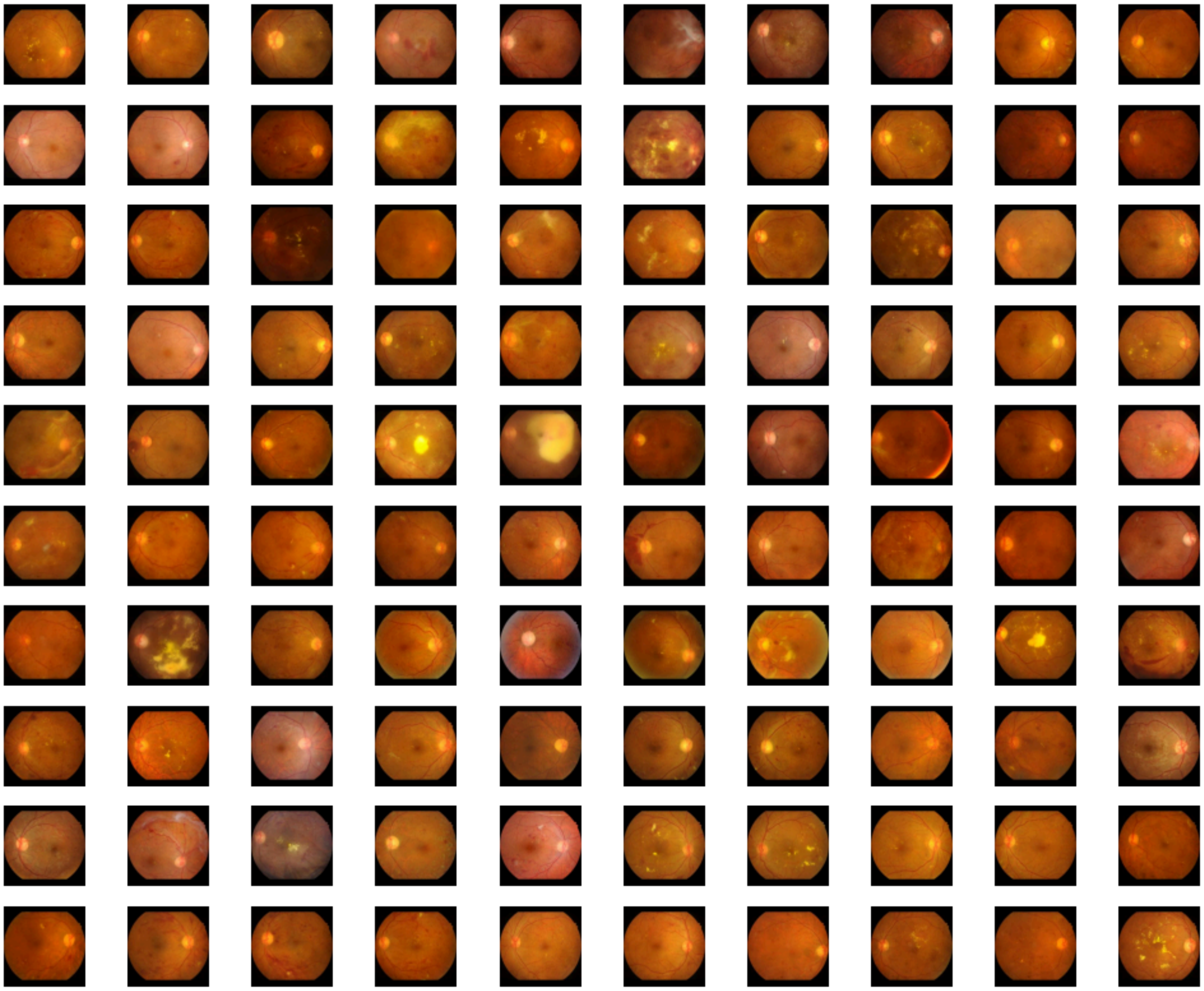
In [301…

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import os

# - Separate images in train and test into two groups as DR and nonDR:
# o NonDR : Label 0
# o DR : Label 3 & Label 4 (You don’t need to use Label 1 and Label 2)

image_files = [f for f in os.listdir("Test") if os.path.isfile(os.path.join(

testnonDR = []
testDR = []

for image in image_files:
  label = image.split("-")[1].split('.')[0]
  # nonDR
  if (label == "0"):
    testnonDR.append((os.path.join("Test",image), 0))
  elif (label == "3" or label == "4"):
    testDR.append((os.path.join("Test",image), 1))

image_files = [f for f in os.listdir("Train") if os.path.isfile(os.path.joir

trainnonDR = []
trainDR = []

for image in image_files:
  label = image.split("-")[1].split('.')[0]
  # nonDR
  if (label == "0"):
    trainnonDR.append((os.path.join("Train",image), 0))
  elif (label == "3" or label == "4"):
    trainDR.append((os.path.join("Train",image), 1))
```

Montage Train DR

In [302…

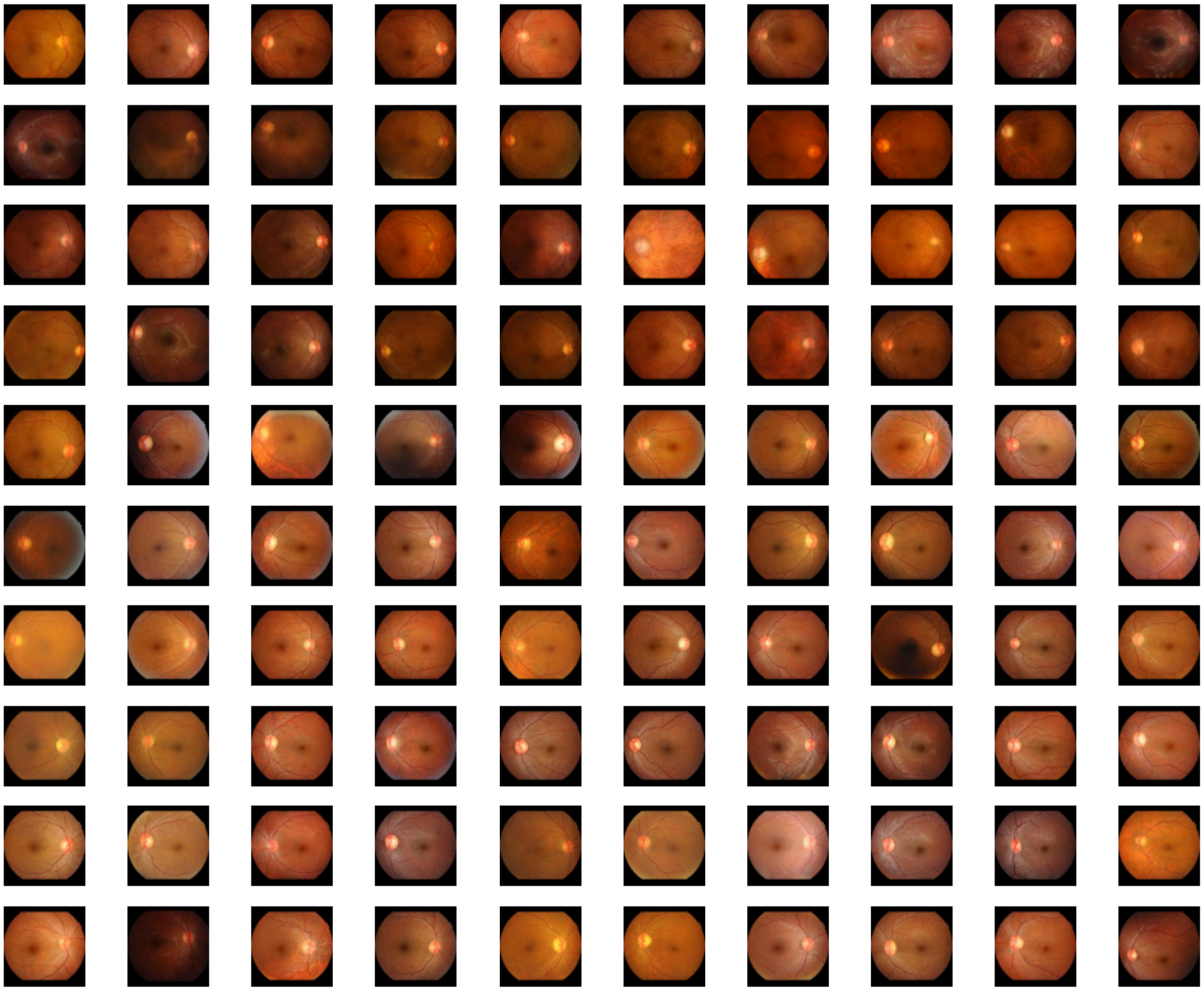
```
fig, axes = plt.subplots(10, 10, figsize=(10, 8))

for i, ax in enumerate(axes.flat):
  ax.imshow(mpimg.imread(trainDR[i][0]), cmap='gray')
  ax.axis('off')

plt.tight_layout()
plt.show()
```



Montage Train non DR

In [303…

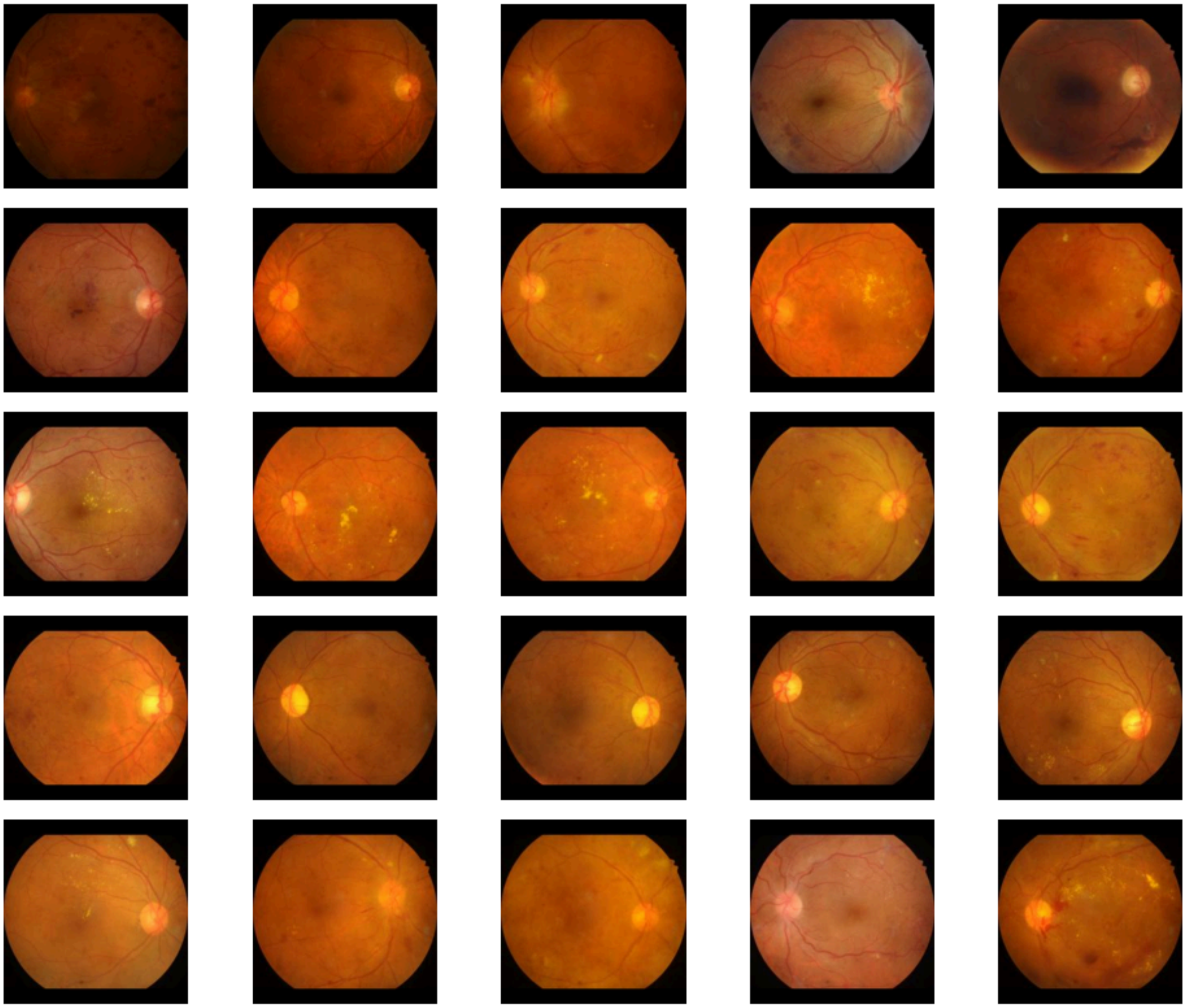
```python
fig, axes = plt.subplots(10, 10, figsize=(10, 8))

for i, ax in enumerate(axes.flat):
  ax.imshow(mpimg.imread(trainnonDR[i][0]), cmap='gray')
  ax.axis('off')

plt.tight_layout()
plt.show()
```



Montage Test DR

In [304…

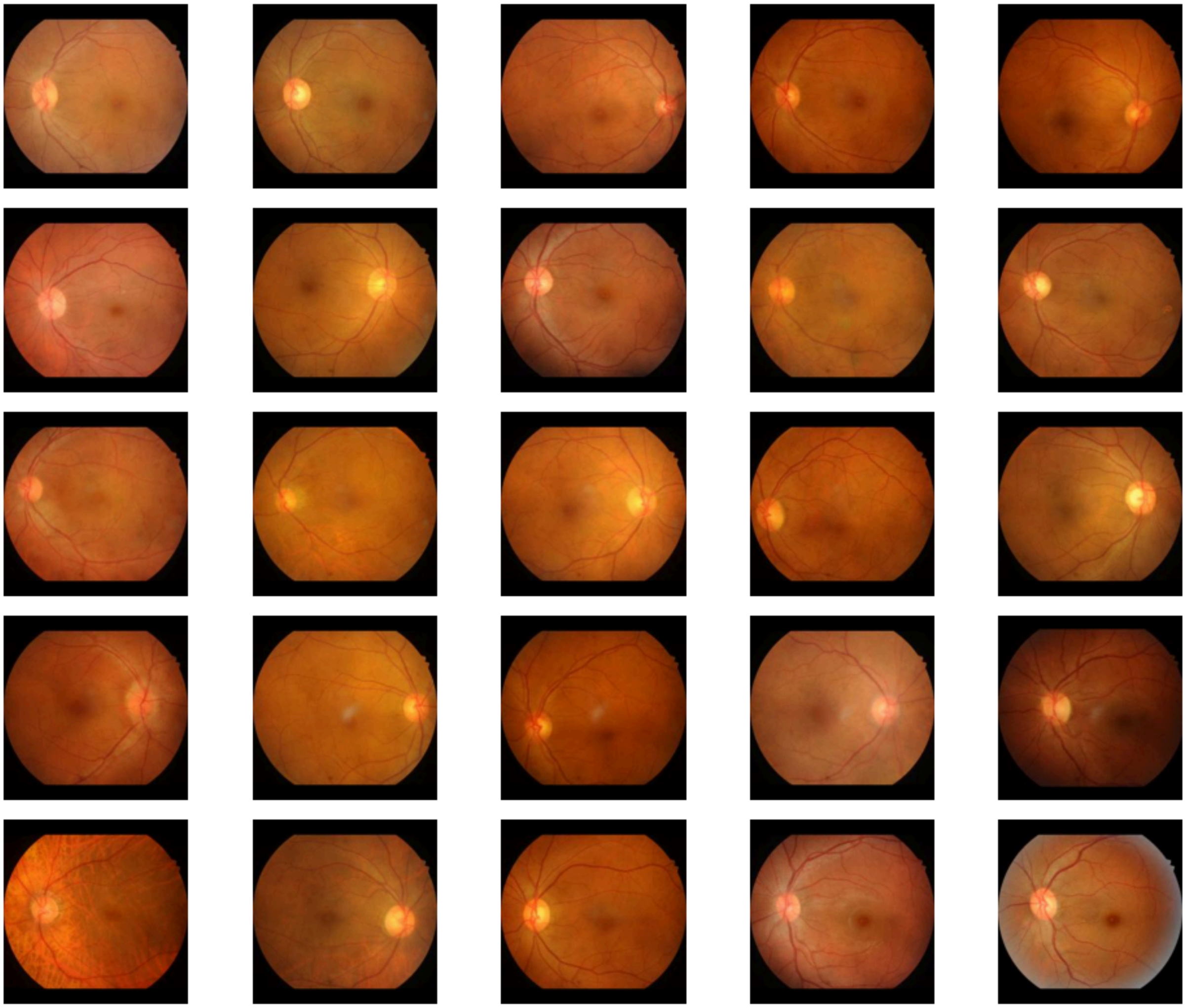
```
fig, axes = plt.subplots(5, 5, figsize=(10, 8))

for i, ax in enumerate(axes.flat):
  ax.imshow(mpimg.imread(testDR[i][0]), cmap='gray')
  ax.axis('off')

plt.tight_layout()
plt.show()
```



Montage Test non Dr

In [305…

```
fig, axes = plt.subplots(5, 5, figsize=(10, 8))

for i, ax in enumerate(axes.flat):
  ax.imshow(mpimg.imread(testnonDR[i][0]), cmap='gray')
  ax.axis('off')

plt.tight_layout()
plt.show()
```



Merge label 0 and label 3-4

In [306…

```
import random

trainSet = trainDR + trainnonDR
random.shuffle(trainSet)

testSet = testDR + testnonDR
random.shuffle(testSet)
```

Download Pretained AlexNet from pytorch

In [307…

```
import torch
import torchvision.models as models

model = models.resnet18(weights=models.ResNet18_Weights.DEFAULT)
model.fc = torch.nn.Linear(model.fc.in_features, 2)
```

In [308…

```
from torchvision import transforms
from torch.utils.data import DataLoader, Dataset
from PIL import Image
```

```
transform = transforms.Compose([
  transforms.Resize((224,224)),
  transforms.ToTensor(),
])

for i in range(len(trainSet)):
  img = Image.open(trainSet[i][0])
  img = transform(img)

  trainSet[i] = (img, trainSet[i][1])
```

In [309…

```
for i in range(len(testSet)):
  img = Image.open(testSet[i][0])
  img = transform(img)

  testSet[i] = (img, testSet[i][1])
```

In [310…

```
train_loader = DataLoader(trainSet, batch_size=32, shuffle=True)
test_loader = DataLoader(testSet, batch_size=32, shuffle=True)
```

resnet18 0.00001 learning rate 50 epochs

In [311…

```
import torch.optim as optim

criterion = torch.nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.00001)

for epoch in range(50):
  model.train()

  running_loss = 0

  for inputs, labels in train_loader:
    outputs = model(inputs)
    loss = criterion(outputs, labels)
    loss.backward()
    optimizer.step()

  running_loss += loss.item()

  print(f'Epoch {epoch+1}, Loss: {running_loss/len(trainSet)}')
```

```
Epoch 1, Loss: 0.003979040954827335
Epoch 2, Loss: 0.003750155640019517
Epoch 3, Loss: 0.002033684272246602
Epoch 4, Loss: 0.0030683944661329694
Epoch 5, Loss: 0.002543058608756455
Epoch 6, Loss: 0.002931990280225583
Epoch 7, Loss: 0.0028892753189175974
Epoch 8, Loss: 0.003243433594239825
Epoch 9, Loss: 0.003143631530642973
Epoch 10, Loss: 0.0027715327210927287
Epoch 11, Loss: 0.0024575523829181833
Epoch 12, Loss: 0.0023910129580516294
Epoch 13, Loss: 0.0024291942555616803
Epoch 14, Loss: 0.0023222514627508615
Epoch 15, Loss: 0.002478572413151366
Epoch 16, Loss: 0.0033562958008584347
Epoch 17, Loss: 0.003717732568659207
Epoch 18, Loss: 0.0038745278050463486
Epoch 19, Loss: 0.001667392508992889
Epoch 20, Loss: 0.0036924970752998084
Epoch 21, Loss: 0.0013029080188691848
Epoch 22, Loss: 0.0014445296985166082
Epoch 23, Loss: 0.0013958163530446211
Epoch 24, Loss: 0.004078942514115271
Epoch 25, Loss: 0.003914051018800253
Epoch 26, Loss: 0.006035046354805913
Epoch 27, Loss: 0.001207933583612108
Epoch 28, Loss: 0.0029593492760268633
Epoch 29, Loss: 0.004643747314868734
Epoch 30, Loss: 0.003944213288303479
Epoch 31, Loss: 0.002857551500491131
Epoch 32, Loss: 0.002119366992772321
Epoch 33, Loss: 0.002048092825403473
Epoch 34, Loss: 0.0019086494983866057
Epoch 35, Loss: 0.0015877702124851686
Epoch 36, Loss: 0.0013154096872426192
Epoch 37, Loss: 0.0011424048865351696
Epoch 38, Loss: 0.00545952663347415
Epoch 39, Loss: 0.0007470308110871667
Epoch 40, Loss: 0.007014095551308955
Epoch 41, Loss: 0.0005497945189939863
Epoch 42, Loss: 0.0004468760253854299
Epoch 43, Loss: 0.0006195369397619818
Epoch 44, Loss: 0.006771795480631669
Epoch 45, Loss: 0.0003972805253726499
Epoch 46, Loss: 0.006668717016969673
Epoch 47, Loss: 0.0066763756340115915
Epoch 48, Loss: 0.0003646231810870338
Epoch 49, Loss: 0.0007930591644480071
Epoch 50, Loss: 0.0007511932794222108
```

In [312…

```
model.eval()

correct = 0
total = 0
```

```
with torch.no_grad():
  for inputs, label in test_loader:
    outputs = model(inputs)
    _, predicted = torch.max(outputs.data, 1)
    total += len(label)
    print(predicted)
    correct += (predicted == label).sum().item()

accuracy = 100 * correct / total
print(f'Accuracy of the model on the test images: {accuracy:.2f}%')
```

```
tensor([0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0,
0,
        1, 0, 1, 0, 0, 1, 1, 0])
tensor([1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0,
0,
        0, 0, 1, 1, 1, 1, 1, 1])
tensor([0, 0])
Accuracy of the model on the test images: 87.88%
```

resnet18 0.00001 learning rate 100 epochs

In [317…

```
# reset model
model = models.resnet18(weights=models.ResNet18_Weights.DEFAULT)
model.fc = torch.nn.Linear(model.fc.in_features, 2)
```

In [318…

```
optimizer = optim.Adam(model.parameters(), lr=0.00001)

for epoch in range(100):
  model.train()

  running_loss = 0

  for inputs, labels in train_loader:
    outputs = model(inputs)
    loss = criterion(outputs, labels)
    loss.backward()
    optimizer.step()

  running_loss += loss.item()

  print(f'Epoch {epoch+1}, Loss: {running_loss/len(trainSet)}')

model.eval()

correct = 0
total = 0

with torch.no_grad():
  for inputs, label in test_loader:
    outputs = model(inputs)
    _, predicted = torch.max(outputs.data, 1)
    total += len(label)
    print(predicted)
    correct += (predicted == label).sum().item()
```

```python
accuracy = 100 * correct / total
print(f'Accuracy of the model on the test images: {accuracy:.2f}%')
```

```
Epoch 1, Loss: 0.003367181418006986
Epoch 2, Loss: 0.003274787724713871
Epoch 3, Loss: 0.0024337158592758474
Epoch 4, Loss: 0.002631116701935052
Epoch 5, Loss: 0.0024942651333048184
Epoch 6, Loss: 0.0032738906400212983
Epoch 7, Loss: 0.003392056964250854
Epoch 8, Loss: 0.0018793405957722943
Epoch 9, Loss: 0.0017234047563159513
Epoch 10, Loss: 0.0017093696019065056
Epoch 11, Loss: 0.0038820461076521226
Epoch 12, Loss: 0.004237439381937109
Epoch 13, Loss: 0.001403989263081829
Epoch 14, Loss: 0.004784800199219225
Epoch 15, Loss: 0.004576498432382071
Epoch 16, Loss: 0.0013413256476361463
Epoch 17, Loss: 0.0013709554180560872
Epoch 18, Loss: 0.004447110895980657
Epoch 19, Loss: 0.0012905172104965387
Epoch 20, Loss: 0.004211737024181084
Epoch 21, Loss: 0.0012847804373804232
Epoch 22, Loss: 0.004273584380687907
Epoch 23, Loss: 0.004004898238274838
Epoch 24, Loss: 0.0017216785408643433
Epoch 25, Loss: 0.003612162770000413
Epoch 26, Loss: 0.003189810055239191
Epoch 27, Loss: 0.002124352909711548
Epoch 28, Loss: 0.0020302752112600127
Epoch 29, Loss: 0.002300867542682455
Epoch 30, Loss: 0.002020842370356103
Epoch 31, Loss: 0.002518472040673638
Epoch 32, Loss: 0.0018096794646073873
Epoch 33, Loss: 0.0022945216193737223
Epoch 34, Loss: 0.0020184836962807502
Epoch 35, Loss: 0.001950434905545721
Epoch 36, Loss: 0.002171667865278192
Epoch 37, Loss: 0.001413072825405848
Epoch 38, Loss: 0.0013511730539195732
Epoch 39, Loss: 0.0033430672805133035
Epoch 40, Loss: 0.0010220384087544006
Epoch 41, Loss: 0.003815468183287387
Epoch 42, Loss: 0.004571317234855682
Epoch 43, Loss: 0.0018786171067085712
Epoch 44, Loss: 0.002661397716878453
Epoch 45, Loss: 0.002938705195712672
Epoch 46, Loss: 0.0029372605368321045
Epoch 47, Loss: 0.001893326930034949
Epoch 48, Loss: 0.0014511368849861946
Epoch 49, Loss: 0.001963229262875212
Epoch 50, Loss: 0.001782058046021814
Epoch 51, Loss: 0.003239985569905678
Epoch 52, Loss: 0.0027092398372605617
Epoch 53, Loss: 0.0012772668892307504
Epoch 54, Loss: 0.0025727843959971624
Epoch 55, Loss: 0.002167096165831451
Epoch 56, Loss: 0.001331825432610419
Epoch 57, Loss: 0.001035366772677648
Epoch 58, Loss: 0.001263096174841261
Epoch 59, Loss: 0.0020750626515785545
Epoch 60, Loss: 0.000963825312106062
Epoch 61, Loss: 0.002535745095650046
Epoch 62, Loss: 0.0009920889301522696
Epoch 63, Loss: 0.0025461258127531653
Epoch 64, Loss: 0.002004599060993714
Epoch 65, Loss: 0.0024581117852652584
Epoch 66, Loss: 0.0005004923747207404
Epoch 67, Loss: 0.002348156986533436
Epoch 68, Loss: 0.00058033291235972
Epoch 69, Loss: 0.0008152813763006188
Epoch 70, Loss: 0.0025840536165794045
Epoch 71, Loss: 0.0036668656864982637
Epoch 72, Loss: 0.0013322542613582388
Epoch 73, Loss: 0.0012766197033893274
Epoch 74, Loss: 0.002972202542227066
Epoch 75, Loss: 0.0023778150517652936
Epoch 76, Loss: 0.0007962611166883536
Epoch 77, Loss: 0.0019156459704447349
Epoch 78, Loss: 0.0005165083978890445
Epoch 79, Loss: 0.0005869578990490983
Epoch 80, Loss: 0.0015899667712037202
Epoch 81, Loss: 0.0015033819796046394
Epoch 82, Loss: 0.0005019799976497309
Epoch 83, Loss: 0.001694286263870358
Epoch 84, Loss: 0.001847614110211918
Epoch 85, Loss: 0.0005271538570233356
Epoch 86, Loss: 0.0017420828110513056
Epoch 87, Loss: 0.0010861120103398186
Epoch 88, Loss: 0.00048551456241756097
Epoch 89, Loss: 0.003065701588582436
Epoch 90, Loss: 0.0022045698611188953
Epoch 91, Loss: 0.0019066457850459948
Epoch 92, Loss: 0.0002024741372245759
Epoch 93, Loss: 0.0024761794142222127
Epoch 94, Loss: 0.001933122539334724
Epoch 95, Loss: 0.0032409086301632894
Epoch 96, Loss: 0.003112450879835433
Epoch 97, Loss: 0.0002458204845046255
Epoch 98, Loss: 0.001515407854480966
Epoch 99, Loss: 0.00020010665001108489
Epoch 100, Loss: 0.0009982742456146716
tensor([0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1,
1,
        1, 1, 0, 1, 1, 0, 0, 1])
tensor([1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0,
0,
        0, 1, 1, 0, 1, 0, 1, 1])
tensor([0, 0])
Accuracy of the model on the test images: 75.76%
```

resnet18 0.00001 learning rate 75 epochs

In [321…

```
# reset model
model = models.resnet18(weights=models.ResNet18_Weights.DEFAULT)
model.fc = torch.nn.Linear(model.fc.in_features, 2)

optimizer = optim.Adam(model.parameters(), lr=0.00001)

for epoch in range(75):
  model.train()

  running_loss = 0

  for inputs, labels in train_loader:
    outputs = model(inputs)
    loss = criterion(outputs, labels)
    loss.backward()
    optimizer.step()

  running_loss += loss.item()

  print(f'Epoch {epoch+1}, Loss: {running_loss/len(trainSet)}')

model.eval()

correct = 0
total = 0

with torch.no_grad():
  for inputs, label in test_loader:
    outputs = model(inputs)
    _, predicted = torch.max(outputs.data, 1)
    total += len(label)
    print(predicted)
    correct += (predicted == label).sum().item()

accuracy = 100 * correct / total
print(f'Accuracy of the model on the test images: {accuracy:.2f}%')
```

```
Epoch 1, Loss: 0.0018907897899123017
Epoch 2, Loss: 0.0019058022749563136
Epoch 3, Loss: 0.003579891145461264
Epoch 4, Loss: 0.0035323795177593306
Epoch 5, Loss: 0.0034616777405200765
Epoch 6, Loss: 0.0021617301243288508
Epoch 7, Loss: 0.002977751804233061
Epoch 8, Loss: 0.0025622900357970004
Epoch 9, Loss: 0.0023845186029426784
Epoch 10, Loss: 0.002884398638506344
Epoch 11, Loss: 0.003306969826323513
Epoch 12, Loss: 0.003290341521979306
Epoch 13, Loss: 0.003259401377073058
Epoch 14, Loss: 0.0021203767928631855
Epoch 15, Loss: 0.002807232656367558
Epoch 16, Loss: 0.002547218178032901
Epoch 17, Loss: 0.0024815872021686242
Epoch 18, Loss: 0.0020970409946219
Epoch 19, Loss: 0.001546251518717072
Epoch 20, Loss: 0.004600640400838295
Epoch 21, Loss: 0.004342324538917393
Epoch 22, Loss: 0.0032634718872693726
Epoch 23, Loss: 0.0017464726815427788
Epoch 24, Loss: 0.0029364374361149533
Epoch 25, Loss: 0.00247048798238257
Epoch 26, Loss: 0.0025541144586258826
Epoch 27, Loss: 0.0018507766352553311
Epoch 28, Loss: 0.0015308782748211219
Epoch 29, Loss: 0.0014709971758178236
Epoch 30, Loss: 0.0009501302752513366
Epoch 31, Loss: 0.005831047254777604
Epoch 32, Loss: 0.0071284283923731704
Epoch 33, Loss: 0.007279001321310199
Epoch 34, Loss: 0.007603735775335289
Epoch 35, Loss: 0.006457943860658876
Epoch 36, Loss: 0.0061243016432231505
Epoch 37, Loss: 0.0014601447238996334
Epoch 38, Loss: 0.0014407621979249591
Epoch 39, Loss: 0.003408723072319179
Epoch 40, Loss: 0.0025749046514934137
Epoch 41, Loss: 0.00316934873157902
Epoch 42, Loss: 0.0036358582834325414
Epoch 43, Loss: 0.004039306121113699
Epoch 44, Loss: 0.0043626230514467
Epoch 45, Loss: 0.0046626640201078776
Epoch 46, Loss: 0.0013814997580264793
Epoch 47, Loss: 0.0021028059465875884
Epoch 48, Loss: 0.003533430136595255
Epoch 49, Loss: 0.0021230160958108273
Epoch 50, Loss: 0.0018321470527797358
Epoch 51, Loss: 0.002129700165314433
Epoch 52, Loss: 0.0034246340336038907
Epoch 53, Loss: 0.00213578447757528
Epoch 54, Loss: 0.0017616233937007445
Epoch 55, Loss: 0.0039184566601705
Epoch 56, Loss: 0.001847662350546989
Epoch 57, Loss: 0.003178523440305361
Epoch 58, Loss: 0.00405645092173773
Epoch 59, Loss: 0.002046503445517692
Epoch 60, Loss: 0.004179865469728462
Epoch 61, Loss: 0.001491960500464829
Epoch 62, Loss: 0.001986188183498754
Epoch 63, Loss: 0.0030856457201887196
Epoch 64, Loss: 0.0019872800849291138
Epoch 65, Loss: 0.004164688318156083
Epoch 66, Loss: 0.003228315583462845
Epoch 67, Loss: 0.00318464509244095
Epoch 68, Loss: 0.003078308086914775
Epoch 69, Loss: 0.0019532460646870535
Epoch 70, Loss: 0.0023852605300190848
Epoch 71, Loss: 0.001778661508968368
Epoch 72, Loss: 0.003341324134559483
Epoch 73, Loss: 0.0017906103848483311
Epoch 74, Loss: 0.0033439163567954926
Epoch 75, Loss: 0.0037508432967189686
tensor([1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1,
1,
        1, 0, 0, 1, 0, 1, 1, 0])
tensor([0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0,
0,
        1, 0, 0, 1, 0, 0, 0, 0])
tensor([1, 1])
Accuracy of the model on the test images: 86.36%
```

resnet18 0.0001 learning rate 50 epochs

In [322…

```python
# reset model
model = models.resnet18(weights=models.ResNet18_Weights.DEFAULT)
model.fc = torch.nn.Linear(model.fc.in_features, 2)

optimizer = optim.Adam(model.parameters(), lr=0.0001)

for epoch in range(50):
  model.train()

  running_loss = 0

  for inputs, labels in train_loader:
    outputs = model(inputs)
    loss = criterion(outputs, labels)
    loss.backward()
    optimizer.step()

  running_loss += loss.item()

  print(f'Epoch {epoch+1}, Loss: {running_loss/len(trainSet)}')

model.eval()

correct = 0
total = 0
```

```
with torch.no_grad():
  for inputs, label in test_loader:
    outputs = model(inputs)
    _, predicted = torch.max(outputs.data, 1)
    total += len(label)
    print(predicted)
    correct += (predicted == label).sum().item()

accuracy = 100 * correct / total
print(f'Accuracy of the model on the test images: {accuracy:.2f}%')
```

```
Epoch 1, Loss: 0.0014524684805814394
Epoch 2, Loss: 0.002649517838593123
Epoch 3, Loss: 0.001128183148714355
Epoch 4, Loss: 0.008624348658995869
Epoch 5, Loss: 0.010659116715308756
Epoch 6, Loss: 0.01055364942736199
Epoch 7, Loss: 0.007389268522596545
Epoch 8, Loss: 0.001744859181489462
Epoch 9, Loss: 0.0035044803693600668
Epoch 10, Loss: 0.0010896535003231658
Epoch 11, Loss: 0.0007680510152638654
Epoch 12, Loss: 0.007218558500712948
Epoch 13, Loss: 0.0005039646351847667
Epoch 14, Loss: 0.009306877039749799
Epoch 15, Loss: 0.00872166611341187
Epoch 16, Loss: 0.0007290429062416582
Epoch 17, Loss: 0.0013301045514266315
Epoch 18, Loss: 0.002674480356594932
Epoch 19, Loss: 0.004941692612050573
Epoch 20, Loss: 0.000767583397112004
Epoch 21, Loss: 0.0006581556472333024
Epoch 22, Loss: 0.0007491728675040753
Epoch 23, Loss: 0.005106643480085677
Epoch 24, Loss: 0.0024840161494243933
Epoch 25, Loss: 0.001332726575985029
Epoch 26, Loss: 0.008167193557501767
Epoch 27, Loss: 0.00030043341422359305
Epoch 28, Loss: 0.0103153477383031
Epoch 29, Loss: 0.008656885837302598
Epoch 30, Loss: 0.005264028037104625
Epoch 31, Loss: 0.002562110062239235
Epoch 32, Loss: 0.0009896705354697974
Epoch 33, Loss: 0.0003240911819127747
Epoch 34, Loss: 0.01409124771444714
Epoch 35, Loss: 0.01670229017502603
Epoch 36, Loss: 3.9155768948074445e-05
Epoch 37, Loss: 4.122990404585456e-05
Epoch 38, Loss: 0.01700399450754841
Epoch 39, Loss: 0.014857337632531786
Epoch 40, Loss: 0.00013902638440929963
Epoch 41, Loss: 0.011077393817530532
Epoch 42, Loss: 0.005466240853187175
Epoch 43, Loss: 0.0013141884877987872
Epoch 44, Loss: 0.004005145933841453
Epoch 45, Loss: 0.0052103852483549006
Epoch 46, Loss: 0.008439271830399213
Epoch 47, Loss: 0.0008265522783368479
Epoch 48, Loss: 0.0010077176390918777
Epoch 49, Loss: 0.0007389164620336392
Epoch 50, Loss: 0.00060172106505368
tensor([1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1,
1,
        1, 0, 1, 1, 1, 0, 1, 1])
tensor([0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1,
1,
        0, 0, 1, 0, 1, 0, 0, 1])
```

```
tensor([0, 0])
Accuracy of the model on the test images: 66.67%
```

Plot ROC curve for the best result and show the confusion matrix.

resnet18 0.000001 learning rate 50 epochs

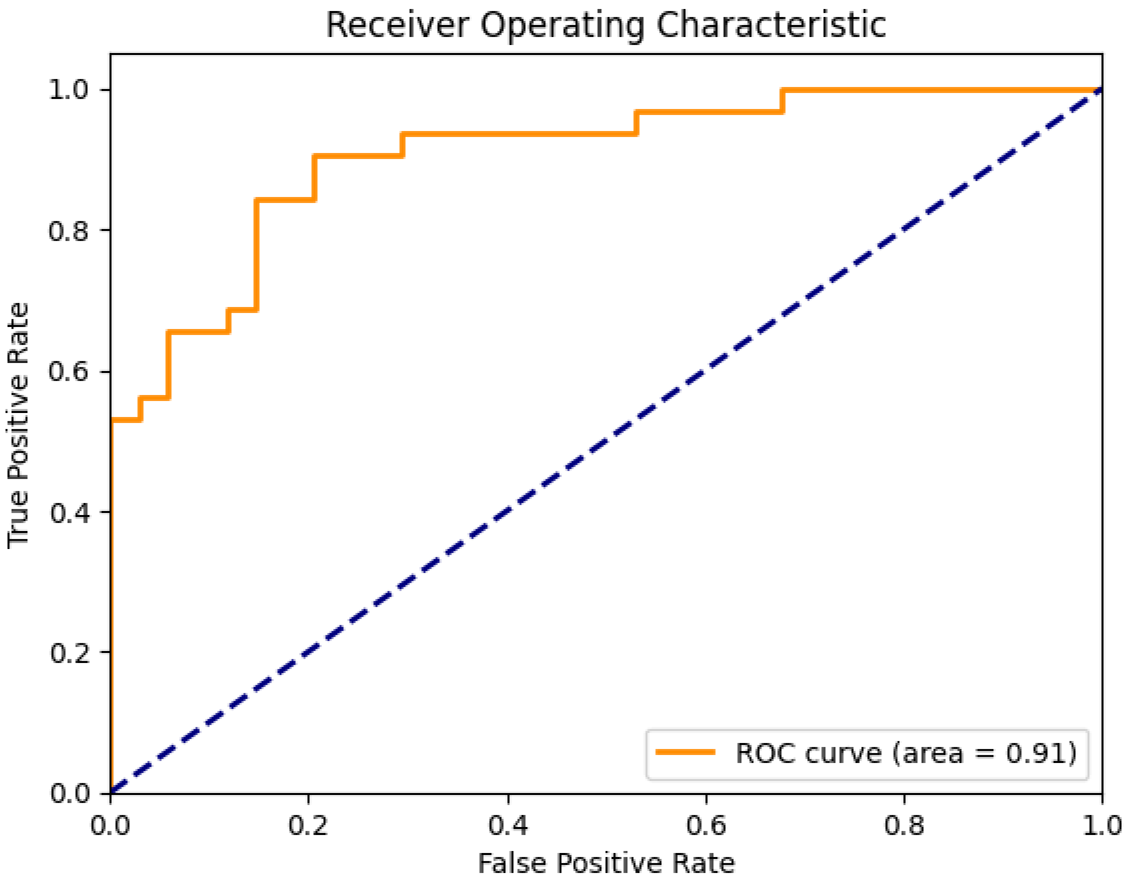
In [324…

```
# reset model
model = models.resnet18(weights=models.ResNet18_Weights.DEFAULT)
model.fc = torch.nn.Linear(model.fc.in_features, 2)

optimizer = optim.Adam(model.parameters(), lr=0.000001)

for epoch in range(50):
  model.train()

  running_loss = 0

  for inputs, labels in train_loader:
    outputs = model(inputs)
    loss = criterion(outputs, labels)
    loss.backward()
    optimizer.step()

  running_loss += loss.item()

  print(f'Epoch {epoch+1}, Loss: {running_loss/len(trainSet)}')

model.eval()

y_true = []
y_scores = []

# Set model to evaluation mode
model.eval()
with torch.no_grad():
  for inputs, labels in test_loader:
    outputs = model(inputs)
    probabilities = torch.softmax(outputs, dim=1)

    y_true.extend(labels.numpy())
    y_scores.extend(probabilities.numpy()[:, 1])
```

```
Epoch 1, Loss: 0.002972601452690154
Epoch 2, Loss: 0.0029478003542710834
Epoch 3, Loss: 0.0029348519989488655
Epoch 4, Loss: 0.002885742873997076
Epoch 5, Loss: 0.0029010049100052057
Epoch 6, Loss: 0.002549519566710357
Epoch 7, Loss: 0.002767976156004672
Epoch 8, Loss: 0.0029683122375132043
Epoch 9, Loss: 0.002568285057053028
Epoch 10, Loss: 0.002715041433326929
Epoch 11, Loss: 0.0025690838056779556
Epoch 12, Loss: 0.002724224955191408
Epoch 13, Loss: 0.0026795622903549253
Epoch 14, Loss: 0.0027320484706863817
Epoch 15, Loss: 0.002633232087012859
Epoch 16, Loss: 0.0026438825325279386
Epoch 17, Loss: 0.0026442271726140718
Epoch 18, Loss: 0.0026930860508276797
Epoch 19, Loss: 0.0026550946996369714
Epoch 20, Loss: 0.002700044023387627
Epoch 21, Loss: 0.0025589714254386693
Epoch 22, Loss: 0.002870504958156482
Epoch 23, Loss: 0.0027255455343639804
Epoch 24, Loss: 0.002935999098454932
Epoch 25, Loss: 0.002911213307065259
Epoch 26, Loss: 0.0028979555642094593
Epoch 27, Loss: 0.0025856877579299394
Epoch 28, Loss: 0.002904387763502069
Epoch 29, Loss: 0.0024562065239546365
Epoch 30, Loss: 0.0028896981176235333
Epoch 31, Loss: 0.0030100885996094937
Epoch 32, Loss: 0.002989376333437077
Epoch 33, Loss: 0.002445269187600696
Epoch 34, Loss: 0.0026470218651025675
Epoch 35, Loss: 0.002885935371487985
Epoch 36, Loss: 0.002916487738316161
Epoch 37, Loss: 0.002962239289562062
Epoch 38, Loss: 0.00289166947747019
Epoch 39, Loss: 0.002515865439106982
Epoch 40, Loss: 0.002480468629399162
Epoch 41, Loss: 0.0025414022026358876
Epoch 42, Loss: 0.002823966486444733
Epoch 43, Loss: 0.0025408724866488566
Epoch 44, Loss: 0.0027609701750343414
Epoch 45, Loss: 0.00232171965944164
Epoch 46, Loss: 0.002865936969504746
Epoch 47, Loss: 0.0028457414315368416
Epoch 48, Loss: 0.0027812661363920813
Epoch 49, Loss: 0.0027923449468056053
Epoch 50, Loss: 0.0025079869574609896
```

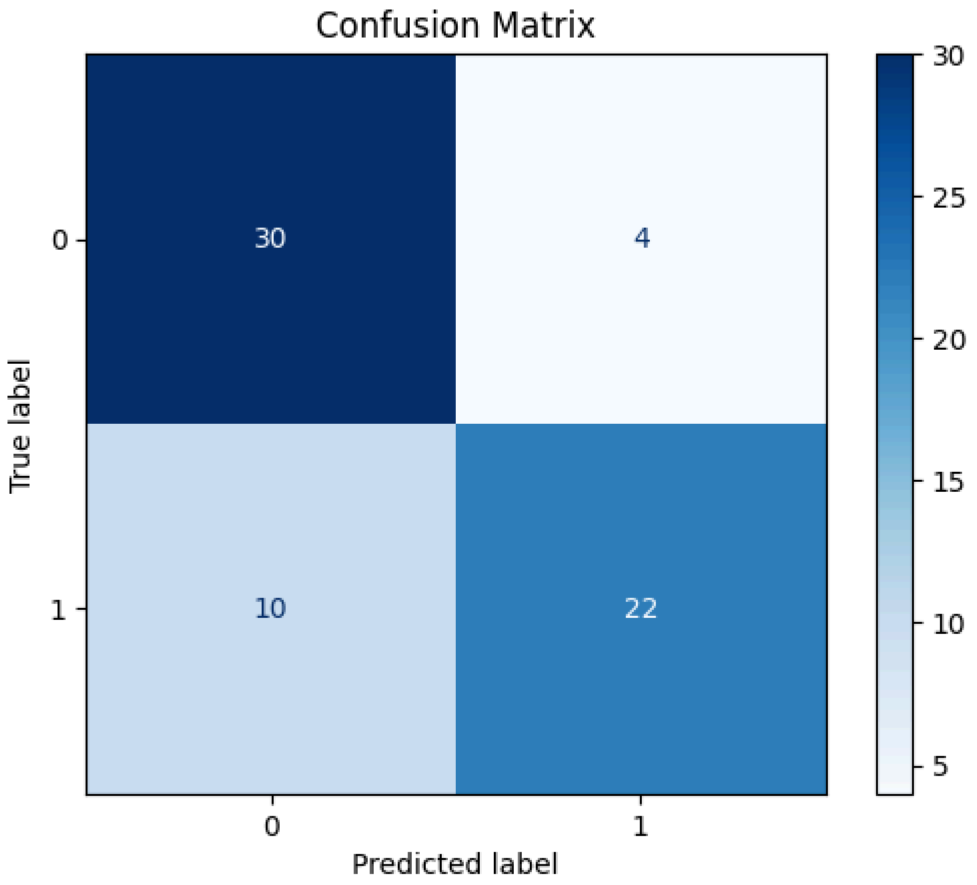
In [329…

```
from sklearn.metrics import roc_curve, auc, confusion_matrix, ConfusionMatri
fpr, tpr, thresholds = roc_curve(y_true, y_scores)
roc_auc = auc(fpr, tpr)

plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = %0.2f)
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()
```



In [327…

```
y_pred = np.array([1 if score >= 0.5 else 0 for score in y_scores])
cm = confusion_matrix(y_true, y_pred)

# Plot confusion matrix
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=[0, 1])
disp.plot(cmap=plt.cm.Blues)
plt.title('Confusion Matrix')
plt.show()
```


Confusion Matrix
True label
Predicted label
0
1
0
1
30
4
10
22
30
25
20
15
10
5