**Combining models ( MobilenetV2 & DSC)**

* Building models: (Mobilenet base model)

|  |
| --- |
| # Load MobileNet base model (excluding top layers)  base\_model = MobileNetV2(weights='imagenet', include\_top=False, input\_shape=(150, 150, 3))  base\_model.trainable = False |

* Mobilenet base model + DSC (Depthwise Separable Convolution)

|  |
| --- |
| model = models.Sequential([  base\_model,  DepthwiseConv2D(kernel\_size=4, strides=1, padding='same', activation=None),  BatchNormalization(),  layers.GlobalAveragePooling2D(),  layers.Dense(2, activation='softmax')  ]) |

* Train Model

|  |
| --- |
| # Compile the model  model.compile(optimizer=Adam(learning\_rate=0.0001),  loss='categorical\_crossentropy',  metrics=['accuracy'])  # Callbacks  callbacks = [  EarlyStopping(patience=5, restore\_best\_weights=True),  ReduceLROnPlateau(factor=0.3, patience=3)  ]  # Train the model  history = model.fit(  train\_generator,  epochs=20,  validation\_data=val\_generator,  callbacks=callbacks  ) |

* Result

|  |
| --- |
| Epoch 1/20  **13/13** ━━━━━━━━━━━━━━━━━━━━ **9s** 347ms/step - accuracy: 0.5704 - loss: 0.6702 - val\_accuracy: 0.6400 - val\_loss: 0.6779 - learning\_rate: 1.0000e-04  Epoch 2/20  **13/13** ━━━━━━━━━━━━━━━━━━━━ **3s** 215ms/step - accuracy: 0.7921 - loss: 0.4957 - val\_accuracy: 0.6400 - val\_loss: 0.6626 - learning\_rate: 1.0000e-04  Epoch 3/20  **13/13** ━━━━━━━━━━━━━━━━━━━━ **3s** 216ms/step - accuracy: 0.8738 - loss: 0.3960 - val\_accuracy: 0.7000 - val\_loss: 0.6500 - learning\_rate: 1.0000e-04  Epoch 4/20  **13/13** ━━━━━━━━━━━━━━━━━━━━ **3s** 215ms/step - accuracy: 0.9164 - loss: 0.3346 - val\_accuracy: 0.7600 - val\_loss: 0.6374 - learning\_rate: 1.0000e-04  Epoch 5/20  **13/13** ━━━━━━━━━━━━━━━━━━━━ **3s** 216ms/step - accuracy: 0.8811 - loss: 0.3343 - val\_accuracy: 0.7600 - val\_loss: 0.6271 - learning\_rate: 1.0000e-04  Epoch 6/20  **13/13** ━━━━━━━━━━━━━━━━━━━━ **3s** 219ms/step - accuracy: 0.8947 - loss: 0.2969 - val\_accuracy: 0.8000 - val\_loss: 0.6160 - learning\_rate: 1.0000e-04  Epoch 7/20  **13/13** ━━━━━━━━━━━━━━━━━━━━ **3s** 215ms/step - accuracy: 0.9250 - loss: 0.2664 - val\_accuracy: 0.8200 - val\_loss: 0.6056 - learning\_rate: 1.0000e-04  Epoch 8/20  **13/13** ━━━━━━━━━━━━━━━━━━━━ **3s** 220ms/step - accuracy: 0.9514 - loss: 0.2478 - val\_accuracy: 0.8200 - val\_loss: 0.5953 - learning\_rate: 1.0000e-04  Epoch 9/20  **13/13** ━━━━━━━━━━━━━━━━━━━━ **3s** 219ms/step - accuracy: 0.9237 - loss: 0.2186 - val\_accuracy: 0.8300 - val\_loss: 0.5832 - learning\_rate: 1.0000e-04  Epoch 10/20  **13/13** ━━━━━━━━━━━━━━━━━━━━ **3s** 223ms/step - accuracy: 0.9294 - loss: 0.2280 - val\_accuracy: 0.8300 - val\_loss: 0.5717 - learning\_rate: 1.0000e-04  Epoch 11/20  **13/13** ━━━━━━━━━━━━━━━━━━━━ **3s** 216ms/step - accuracy: 0.9079 - loss: 0.2621 - val\_accuracy: 0.8300 - val\_loss: 0.5541 - learning\_rate: 1.0000e-04  Epoch 12/20  **13/13** ━━━━━━━━━━━━━━━━━━━━ **3s** 227ms/step - accuracy: 0.9378 - loss: 0.1909 - val\_accuracy: 0.8300 - val\_loss: 0.5431 - learning\_rate: 1.0000e-04  Epoch 13/20  **13/13** ━━━━━━━━━━━━━━━━━━━━ **3s** 222ms/step - accuracy: 0.9422 - loss: 0.1848 - val\_accuracy: 0.8300 - val\_loss: 0.5302 - learning\_rate: 1.0000e-04  Epoch 14/20  **13/13** ━━━━━━━━━━━━━━━━━━━━ **3s** 225ms/step - accuracy: 0.9392 - loss: 0.2025 - val\_accuracy: 0.8300 - val\_loss: 0.5131 - learning\_rate: 1.0000e-04  Epoch 15/20  **13/13** ━━━━━━━━━━━━━━━━━━━━ **3s** 217ms/step - accuracy: 0.9447 - loss: 0.1966 - val\_accuracy: 0.8300 - val\_loss: 0.5038 - learning\_rate: 1.0000e-04  Epoch 16/20  **13/13** ━━━━━━━━━━━━━━━━━━━━ **3s** 223ms/step - accuracy: 0.9278 - loss: 0.1958 - val\_accuracy: 0.8400 - val\_loss: 0.4877 - learning\_rate: 1.0000e-04  Epoch 17/20  **13/13** ━━━━━━━━━━━━━━━━━━━━ **3s** 215ms/step - accuracy: 0.9179 - loss: 0.2031 - val\_accuracy: 0.8300 - val\_loss: 0.4701 - learning\_rate: 1.0000e-04  Epoch 18/20  **13/13** ━━━━━━━━━━━━━━━━━━━━ **3s** 217ms/step - accuracy: 0.9358 - loss: 0.1929 - val\_accuracy: 0.8300 - val\_loss: 0.4556 - learning\_rate: 1.0000e-04  Epoch 19/20  **13/13** ━━━━━━━━━━━━━━━━━━━━ **3s** 217ms/step - accuracy: 0.9756 - loss: 0.1423 - val\_accuracy: 0.8300 - val\_loss: 0.4497 - learning\_rate: 1.0000e-04  Epoch 20/20  **13/13** ━━━━━━━━━━━━━━━━━━━━ **3s** 225ms/step - accuracy: 0.9610 - loss: 0.1659 - val\_accuracy: 0.8400 - val\_loss: 0.4294 - learning\_rate: 1.0000e-04 |

Rata rata train speed 225ms

* Train Accuracy Graph

A graph of a graph with blue and orange lines

AI-generated content may be incorrect.

* Loss

A graph of loss and validation

AI-generated content may be incorrect.

* Confusion matrix

A blue squares with white text

AI-generated content may be incorrect.

* Classification report

Accuracy = 86%

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | precision | recall | f1-score | support |
| Drowsy | 0.78 | 1.00 | 0.88 | 76 |
| Nondrowsy | 1.00 | 0.72 | 0.83 | 74 |
| accuracy |  |  | 0.86 | 150 |
| macro avg | 0.89 | 0.86 | 0.86 | 150 |
| weighted avg | 0.89 | 0.86 | 0.86 | 150 |

**Combining models ( MobilenetV2 + DSC + bottleneck)**

* Building models: (Mobilenet base model)

|  |
| --- |
| # Load MobileNet base model (excluding top layers)  base\_model = MobileNetV2(weights='imagenet', include\_top=False, input\_shape=(150, 150, 3))  base\_model.trainable = False |

* Mobilenet base model + DSC (Depthwise Separable Convolution) + Bottleneck

|  |
| --- |
| mobilenet\_model = Sequential([  base\_model,  GlobalAveragePooling2D(),  Dense(4, activation='relu'),  Dropout(0.3),  Dropout(0.5),  Dense(3, activation='softmax')  ]) |

* Train Model

|  |
| --- |
| # Compile the model  mobilenet\_model.compile(optimizer=Adam(), loss='sparse\_categorical\_crossentropy', metrics=['accuracy'])# Callbacks  callbacks = [  EarlyStopping(patience=5, restore\_best\_weights=True),  ReduceLROnPlateau(factor=0.3, patience=3)  ]  # Train the model  history\_v2 = mobilenet\_model.fit(train\_data, train\_labels,  epochs=10,  batch\_size=14,  validation\_data=(val\_data, val\_labels)) |

* Result

|  |
| --- |
| Epoch 1/20  **22/22** ━━━━━━━━━━━━━━━━━━━━ **8s** 143ms/step - accuracy: 0.4290 - loss: 1.1287 - val\_accuracy: 0.5000 - val\_loss: 0.7193  Epoch 2/20  **22/22** ━━━━━━━━━━━━━━━━━━━━ **2s** 70ms/step - accuracy: 0.5870 - loss: 0.9479 - val\_accuracy: 0.5000 - val\_loss: 0.7475  Epoch 3/20  **22/22** ━━━━━━━━━━━━━━━━━━━━ **2s** 68ms/step - accuracy: 0.5019 - loss: 0.9334 - val\_accuracy: 0.5000 - val\_loss: 0.7127  Epoch 4/20  **22/22** ━━━━━━━━━━━━━━━━━━━━ **2s** 70ms/step - accuracy: 0.5676 - loss: 0.9008 - val\_accuracy: 0.5000 - val\_loss: 0.7316  Epoch 5/20  **22/22** ━━━━━━━━━━━━━━━━━━━━ **2s** 71ms/step - accuracy: 0.5223 - loss: 0.8884 - val\_accuracy: 0.5000 - val\_loss: 0.7032  Epoch 6/20  **22/22** ━━━━━━━━━━━━━━━━━━━━ **2s** 70ms/step - accuracy: 0.4797 - loss: 0.9167 - val\_accuracy: 0.5000 - val\_loss: 0.6970  Epoch 7/20  **22/22** ━━━━━━━━━━━━━━━━━━━━ **2s** 69ms/step - accuracy: 0.5013 - loss: 0.8783 - val\_accuracy: 0.5323 - val\_loss: 0.6966  Epoch 8/20  **22/22** ━━━━━━━━━━━━━━━━━━━━ **2s** 70ms/step - accuracy: 0.5027 - loss: 0.8834 - val\_accuracy: 0.5000 - val\_loss: 0.6635  Epoch 9/20  **22/22** ━━━━━━━━━━━━━━━━━━━━ **2s** 69ms/step - accuracy: 0.5038 - loss: 0.8803 - val\_accuracy: 0.6129 - val\_loss: 0.6820  Epoch 10/20  **22/22** ━━━━━━━━━━━━━━━━━━━━ **2s** 71ms/step - accuracy: 0.4875 - loss: 0.8667 - val\_accuracy: 0.6855 - val\_loss: 0.6865  Epoch 11/20  **22/22** ━━━━━━━━━━━━━━━━━━━━ **2s** 68ms/step - accuracy: 0.5411 - loss: 0.8187 - val\_accuracy: 0.6532 - val\_loss: 0.6575  Epoch 12/20  **22/22** ━━━━━━━━━━━━━━━━━━━━ **2s** 70ms/step - accuracy: 0.5483 - loss: 0.8412 - val\_accuracy: 0.7581 - val\_loss: 0.6500  Epoch 13/20  **22/22** ━━━━━━━━━━━━━━━━━━━━ **2s** 69ms/step - accuracy: 0.5261 - loss: 0.8222 - val\_accuracy: 0.8871 - val\_loss: 0.6709  Epoch 14/20  **22/22** ━━━━━━━━━━━━━━━━━━━━ **2s** 69ms/step - accuracy: 0.5510 - loss: 0.8067 - val\_accuracy: 0.8387 - val\_loss: 0.6339  Epoch 15/20  **22/22** ━━━━━━━━━━━━━━━━━━━━ **2s** 69ms/step - accuracy: 0.5086 - loss: 0.8081 - val\_accuracy: 0.8387 - val\_loss: 0.6239  Epoch 16/20  **22/22** ━━━━━━━━━━━━━━━━━━━━ **2s** 70ms/step - accuracy: 0.5069 - loss: 0.8451 - val\_accuracy: 0.8065 - val\_loss: 0.6004  Epoch 17/20  **22/22** ━━━━━━━━━━━━━━━━━━━━ **2s** 70ms/step - accuracy: 0.5350 - loss: 0.7704 - val\_accuracy: 0.7823 - val\_loss: 0.5865  Epoch 18/20  **22/22** ━━━━━━━━━━━━━━━━━━━━ **2s** 71ms/step - accuracy: 0.5715 - loss: 0.7774 - val\_accuracy: 0.7903 - val\_loss: 0.5808  Epoch 19/20  **22/22** ━━━━━━━━━━━━━━━━━━━━ **2s** 69ms/step - accuracy: 0.5707 - loss: 0.7621 - val\_accuracy: 0.9274 - val\_loss: 0.6099  Epoch 20/20  **22/22** ━━━━━━━━━━━━━━━━━━━━ **2s** 69ms/step - accuracy: 0.5843 - loss: 0.7849 - val\_accuracy: 0.9032 - val\_loss: 0.5825 |

Rata rata train speed 73ms

* Train Accuracy Graph

A graph with red line and blue line

AI-generated content may be incorrect.

* Loss

A graph of a graph showing a loss

AI-generated content may be incorrect.

* Confusion matrix

A blue squares with white text

AI-generated content may be incorrect.

* Classification report

Accuracy = 93%

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | precision | recall | f1-score | support |
| Drowsy | 0.88 | 1.00 | 0.94 | 76 |
| Nondrowsy | 1.00 | 0.87 | 0.93 | 74 |
| accuracy |  |  | 0.93 | 150 |
| macro avg | 0.94 | 0.93 | 0.93 | 150 |
| weighted avg | 0.94 | 0.93 | 0.93 | 150 |

**Combining models ( MobilenetV2 & DSC)**

* Building models: (Mobilenet base model)

|  |
| --- |
| # Load MobileNet base model (excluding top layers)  base\_model = MobileNetV2(weights='imagenet', include\_top=False, input\_shape=(150, 150, 3))  base\_model.trainable = False |

* Bottleneck with expansion function

|  |
| --- |
| # Expansion Bottleneck Function  def expansion\_bottleneck(input\_tensor, filters, expansion\_factor=6):  # Expansion layer  expanded\_filters = filters \* expansion\_factor  x = layers.Conv2D(expanded\_filters, 1, padding='same')(input\_tensor)  x = layers.BatchNormalization()(x)  x = layers.ReLU()(x)    # Depthwise convolution  x = layers.DepthwiseConv2D(3, padding='same')(x)  x = layers.BatchNormalization()(x)  x = layers.ReLU()(x)    # Projection layer  x = layers.Conv2D(filters, 1, padding='same')(x)  x = layers.BatchNormalization()(x)    # Skip connection if input and output shapes match  if input\_tensor.shape[-1] == filters:  x = layers.Add()([x, input\_tensor])    return x |

* Hybrid model creation

|  |
| --- |
| # Build the hybrid model  inputs = layers.Input(shape=(150, 150, 3))  x = base\_model(inputs, training=False)  # Add expansion bottleneck blocks  x = expansion\_bottleneck(x, filters=128)  x = layers.MaxPooling2D(2)(x)  x = expansion\_bottleneck(x, filters=64)  x = layers.MaxPooling2D(2)(x)  # Final layers  x = layers.GlobalAveragePooling2D()(x)  x = layers.Dense(130, activation='relu')(x)  x = layers.Dropout(0.55)(x)  outputs = layers.Dense(2, activation='softmax')(x)  hybrid\_model = models.Model(inputs, outputs) |

* Train Model

|  |
| --- |
| # Compile the model  hybrid\_model.compile(optimizer=optimizers.Adam(learning\_rate=0.001),  loss='categorical\_crossentropy',  metrics=['accuracy'])  # Callbacks  callbacks = [  EarlyStopping(patience=5, restore\_best\_weights=True),  ReduceLROnPlateau(factor=0.3, patience=3)  ]  # Train the model  history = hybrid\_model.fit(  train\_generator,  epochs=30,  validation\_data=val\_generator,  callbacks=callbacks  ) |

* Result

|  |
| --- |
| Epoch 1/30  **13/13** ━━━━━━━━━━━━━━━━━━━━ **15s** 460ms/step - accuracy: 0.6093 - loss: 0.7828 - val\_accuracy: 0.4600 - val\_loss: 0.6937 - learning\_rate: 0.0010  Epoch 2/30  **13/13** ━━━━━━━━━━━━━━━━━━━━ **3s** 265ms/step - accuracy: 0.8690 - loss: 0.3372 - val\_accuracy: 0.5400 - val\_loss: 0.6881 - learning\_rate: 0.0010  Epoch 3/30  **13/13** ━━━━━━━━━━━━━━━━━━━━ **4s** 275ms/step - accuracy: 0.8978 - loss: 0.2047 - val\_accuracy: 0.5400 - val\_loss: 0.6820 - learning\_rate: 0.0010  Epoch 4/30  **13/13** ━━━━━━━━━━━━━━━━━━━━ **4s** 280ms/step - accuracy: 0.9321 - loss: 0.1561 - val\_accuracy: 0.5400 - val\_loss: 0.6783 - learning\_rate: 0.0010  Epoch 5/30  **13/13** ━━━━━━━━━━━━━━━━━━━━ **4s** 276ms/step - accuracy: 0.9499 - loss: 0.1086 - val\_accuracy: 0.5400 - val\_loss: 0.6752 - learning\_rate: 0.0010  Epoch 6/30  **13/13** ━━━━━━━━━━━━━━━━━━━━ **4s** 265ms/step - accuracy: 0.9511 - loss: 0.1184 - val\_accuracy: 0.5400 - val\_loss: 0.6723 - learning\_rate: 0.0010  Epoch 7/30  **13/13** ━━━━━━━━━━━━━━━━━━━━ **4s** 269ms/step - accuracy: 0.9565 - loss: 0.0967 - val\_accuracy: 0.5400 - val\_loss: 0.6670 - learning\_rate: 0.0010  Epoch 8/30  **13/13** ━━━━━━━━━━━━━━━━━━━━ **3s** 263ms/step - accuracy: 0.9829 - loss: 0.0613 - val\_accuracy: 0.5400 - val\_loss: 0.6658 - learning\_rate: 0.0010  Epoch 9/30  **13/13** ━━━━━━━━━━━━━━━━━━━━ **3s** 263ms/step - accuracy: 0.9608 - loss: 0.1271 - val\_accuracy: 0.5400 - val\_loss: 0.6570 - learning\_rate: 0.0010  Epoch 10/30  **13/13** ━━━━━━━━━━━━━━━━━━━━ **3s** 265ms/step - accuracy: 0.9509 - loss: 0.1062 - val\_accuracy: 0.5400 - val\_loss: 0.6460 - learning\_rate: 0.0010  Epoch 11/30  **13/13** ━━━━━━━━━━━━━━━━━━━━ **4s** 266ms/step - accuracy: 0.9603 - loss: 0.1232 - val\_accuracy: 0.5400 - val\_loss: 0.6396 - learning\_rate: 0.0010  Epoch 12/30  **13/13** ━━━━━━━━━━━━━━━━━━━━ **4s** 268ms/step - accuracy: 0.9675 - loss: 0.0919 - val\_accuracy: 0.5400 - val\_loss: 0.6132 - learning\_rate: 0.0010  Epoch 13/30  **13/13** ━━━━━━━━━━━━━━━━━━━━ **3s** 264ms/step - accuracy: 0.9736 - loss: 0.0892 - val\_accuracy: 0.5500 - val\_loss: 0.6030 - learning\_rate: 0.0010  Epoch 14/30  **13/13** ━━━━━━━━━━━━━━━━━━━━ **3s** 275ms/step - accuracy: 0.9871 - loss: 0.0548 - val\_accuracy: 0.5400 - val\_loss: 0.5974 - learning\_rate: 0.0010  Epoch 15/30  **13/13** ━━━━━━━━━━━━━━━━━━━━ **3s** 264ms/step - accuracy: 0.9752 - loss: 0.0607 - val\_accuracy: 0.5500 - val\_loss: 0.5864 - learning\_rate: 0.0010  Epoch 16/30  **13/13** ━━━━━━━━━━━━━━━━━━━━ **4s** 266ms/step - accuracy: 0.9870 - loss: 0.0544 - val\_accuracy: 0.5600 - val\_loss: 0.5742 - learning\_rate: 0.0010  Epoch 17/30  **13/13** ━━━━━━━━━━━━━━━━━━━━ **3s** 264ms/step - accuracy: 0.9727 - loss: 0.0533 - val\_accuracy: 0.5800 - val\_loss: 0.5593 - learning\_rate: 0.0010  Epoch 18/30  **13/13** ━━━━━━━━━━━━━━━━━━━━ **3s** 265ms/step - accuracy: 0.9958 - loss: 0.0266 - val\_accuracy: 0.5800 - val\_loss: 0.5401 - learning\_rate: 0.0010  Epoch 19/30  **13/13** ━━━━━━━━━━━━━━━━━━━━ **4s** 280ms/step - accuracy: 0.9873 - loss: 0.0397 - val\_accuracy: 0.6000 - val\_loss: 0.5359 - learning\_rate: 0.0010  Epoch 20/30  **13/13** ━━━━━━━━━━━━━━━━━━━━ **3s** 264ms/step - accuracy: 0.9883 - loss: 0.0280 - val\_accuracy: 0.6100 - val\_loss: 0.5203 - learning\_rate: 0.0010  Epoch 21/30  **13/13** ━━━━━━━━━━━━━━━━━━━━ **3s** 265ms/step - accuracy: 0.9813 - loss: 0.0324 - val\_accuracy: 0.6600 - val\_loss: 0.5028 - learning\_rate: 0.0010  Epoch 22/30  **13/13** ━━━━━━━━━━━━━━━━━━━━ **3s** 264ms/step - accuracy: 0.9834 - loss: 0.0771 - val\_accuracy: 0.7500 - val\_loss: 0.4531 - learning\_rate: 0.0010  Epoch 23/30  **13/13** ━━━━━━━━━━━━━━━━━━━━ **3s** 272ms/step - accuracy: 0.9742 - loss: 0.0807 - val\_accuracy: 0.7700 - val\_loss: 0.4611 - learning\_rate: 0.0010  Epoch 24/30  **13/13** ━━━━━━━━━━━━━━━━━━━━ **4s** 266ms/step - accuracy: 0.9821 - loss: 0.0518 - val\_accuracy: 0.7900 - val\_loss: 0.4144 - learning\_rate: 0.0010  Epoch 25/30  **13/13** ━━━━━━━━━━━━━━━━━━━━ **3s** 264ms/step - accuracy: 0.9763 - loss: 0.0653 - val\_accuracy: 0.8500 - val\_loss: 0.3973 - learning\_rate: 0.0010  Epoch 26/30  **13/13** ━━━━━━━━━━━━━━━━━━━━ **3s** 264ms/step - accuracy: 0.9857 - loss: 0.0532 - val\_accuracy: 0.9100 - val\_loss: 0.3463 - learning\_rate: 0.0010  Epoch 27/30  **13/13** ━━━━━━━━━━━━━━━━━━━━ **4s** 266ms/step - accuracy: 0.9895 - loss: 0.0386 - val\_accuracy: 0.9500 - val\_loss: 0.3077 - learning\_rate: 0.0010  Epoch 28/30  **13/13** ━━━━━━━━━━━━━━━━━━━━ **4s** 266ms/step - accuracy: 0.9923 - loss: 0.0295 - val\_accuracy: 0.9200 - val\_loss: 0.2874 - learning\_rate: 0.0010  Epoch 29/30  **13/13** ━━━━━━━━━━━━━━━━━━━━ **3s** 265ms/step - accuracy: 0.9904 - loss: 0.0361 - val\_accuracy: 0.9700 - val\_loss: 0.2627 - learning\_rate: 0.0010  Epoch 30/30  **13/13** ━━━━━━━━━━━━━━━━━━━━ **3s** 265ms/step - accuracy: 0.9968 - loss: 0.0227 - val\_accuracy: 0.9900 - val\_loss: 0.1898 - learning\_rate: 0.0010 |

Rata rata train speed 292ms (0.292 s)

* Train Accuracy Graph

A graph of a line

AI-generated content may be incorrect.

* Loss

A graph of loss and validation

AI-generated content may be incorrect.

* Confusion matrix

A diagram of a confusion matrix

AI-generated content may be incorrect.

* Classification report

Accuracy = 98%

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | precision | recall | f1-score | support |
| Drowsy | 1.00 | 0.96 | 0.98 | 76 |
| Nondrowsy | 0.96 | 1.00 | 0.98 | 74 |
| accuracy |  |  | 0.98 | 150 |
| macro avg | 0.98 | 0.98 | 0.98 | 150 |
| weighted avg | 0.98 | 0.98 | 0.98 | 150 |