In [ ]:
```
from tensorflow import keras
from tensorflow.keras import layers
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score, precision_score, recall_score, f1_score
```

In [ ]:
```
train_raw = pd.read_csv('/Data/train.csv')
test_raw = pd.read_csv('/Data/test.csv')
```

In [ ]:
```
print(test_raw)
```

```
          MEAN_RR    MEDIAN_RR        SDRR      RMSSD       SDSD  SDRR_RMSSD  \
0      721.901897   727.267280   74.722315  12.361264  12.361069    6.044877
1      843.538633   844.407930   58.499429  19.298880  19.298795    3.031234
2      958.523868   966.671125  132.849110  21.342715  21.342653    6.224565
3      824.838669   842.485905  117.822093  11.771814  11.771248   10.008830
4      756.707933   747.941620  143.968457  13.357748  13.356388   10.777899
...           ...          ...         ...        ...        ...         ...
41028 1118.406543  1117.857050  113.955632  18.592177  18.592071    6.129225
41029  855.991173   765.221410  296.763366  20.623727  20.610745   14.389415
41030  689.388662   663.192770  133.815713  21.699038  21.696138    6.166896
41031  761.421571   761.064590   60.792671  11.280586  11.280573    5.389141
41032  868.040461   873.048735   61.301652  11.781442  11.781440    5.203239

             HR      pNN25     pNN50        SD1  ...    HF_PCT      HF_NU  \
0     84.121868   4.933333  0.000000   8.743513  ...  3.921868   9.760289
1     71.478642  21.000000  0.200000  13.650863  ...  1.123416   1.663151
2     63.874293  24.133333  1.800000  15.096571  ...  0.370208   0.766416
3     74.330531   4.733333  0.533333   8.326307  ...  0.615932   3.358652
4     82.092049   5.933333  0.666667   9.447545  ...  0.662879   6.292253
...         ...        ...       ...        ...  ...       ...        ...
41028 54.234182  18.800000  0.266667  13.150966  ...  0.006813   0.035830
41029 76.416971  15.733333  3.533333  14.578861  ...  1.383097   8.622919
41030 90.254005  13.933333  4.000000  15.346606  ...  3.559302  10.160405
41031 79.313782   2.733333  0.266667   7.979232  ...  3.124367   6.905613
41032 69.484061   3.733333  0.000000   8.333516  ...  0.580832   1.408066

               TP        LF_HF     HF_LF    sampen    higuci  datasetId  \
0      1698.605390     9.245599  0.108160  2.097342  1.243696          2
1      2358.884694    59.126832  0.016913  2.217275  1.250056          2
2      4328.633724   129.477524  0.007723  2.217136  1.144943          2
3      2854.449091    28.773854  0.034754  2.106863  1.142355          2
4      5310.027472    14.892559  0.067148  1.912191  1.128098          2
...            ...          ...       ...       ...       ...        ...
41028  5100.718213  2789.962965  0.000358  2.183460  1.082927          2
41029  8532.599635    10.597001  0.094366  1.695327  1.154370          2
41030  6074.251632     8.842128  0.113095  2.078000  1.214869          2
41031  1337.780773    13.480973  0.074179  2.191379  1.234546          2
41032  1678.332700    70.019404  0.014282  2.143131  1.142660          2

          condition  condition_code
0         no stress               0
1     time pressure               1
2         no stress               0
3         no stress               0
4      interruption               1
...             ...             ...
41028 time pressure               1
41029  interruption               1
41030     no stress               0
41031     no stress               0
41032 time pressure               1

[41033 rows x 37 columns]
```

In [ ]:
```
for feat in list(train_raw.columns)[0:-3]:
  corr = train_raw[feat].corr(train_raw['condition_code'], method='spearman')
  print(feat, ':', corr)
```

```
MEAN_RR : 0.30304308386387024
MEDIAN_RR : 0.30825705284926463
SDRR : 0.12119837769637454
RMSSD : 0.2052762377363823
SDSD : 0.20529054128502913
SDRR_RMSSD : 0.00972561257116244
HR : -0.3085077291732474
pNN25 : 0.21797361623338357
pNN50 : 0.11868409311109891
SD1 : 0.2052905412595383
SD2 : 0.12056071313013972
KURT : -0.12975565644773862
SKEW : -0.0898604874425136
MEAN_REL_RR : -0.015026702180431417
MEDIAN_REL_RR : -0.053719433439914505
SDRR_REL_RR : 0.07098585471142109
RMSSD_REL_RR : -0.019180075229286435
SDSD_REL_RR : -0.019180332582991757
SDRR_RMSSD_REL_RR : 0.1469969367399451
KURT_REL_RR : -0.12975565644773862
SKEW_REL_RR : -0.0898604874425136
VLF : 0.11440983444675058
VLF_PCT : 0.026311830845100078
LF : 0.12074506545902193
LF_PCT : -0.010989299252453013
LF_NU : 0.21880604438857557
HF : -0.16417159280741953
HF_PCT : -0.20595577045702848
HF_NU : -0.2188060445415182
TP : 0.1693233429653008
LF_HF : 0.2188060445670088
HF_LF : -0.2188060445925104
sampen : 0.13133692376620407
higuci : -0.05424684977934845
```

In [ ]:
```
x1 = train_raw['RMSSD']
x2 = train_raw['HR']
x3 = train_raw['SDRR']
x_training = np.array(list(zip(x1, x2, x3)))
y_training = train_raw['condition_code']
y_training = keras.utils.to_categorical(y_training)
```

In [ ]:
```
xt1 = test_raw['RMSSD']
xt2 = test_raw['HR']
xt3 = test_raw['SDRR']
x_test = np.array(list(zip(xt1, xt2, xt3)))
y_test = test_raw['condition_code']
y_test = keras.utils.to_categorical(y_test)
```

In [ ]:
```
#scaling x_training
mean = np.mean(x_training, axis = 0)
std = np.std(x_training, axis = 0)
x_training = np.array(list((x - mean)/std for x in x_training))

#scaling x_test
mean = np.mean(x_test, axis = 0)
std = np.std(x_test, axis = 0)
x_test = np.array(list((x - mean)/std for x in x_test))
```

In [ ]:
```
# Create the model
model = keras.Sequential()
model.add(layers.Dense(200, input_shape=[None, 3], activation='relu'))
model.add(layers.Dense(50, activation='relu'))
model.add(layers.Dense(2, activation='softmax'))

model.compile(
    optimizer=keras.optimizers.Adam(learning_rate=1e-3),
    loss=keras.losses.CategoricalCrossentropy(),
    metrics=[keras.metrics.CategoricalAccuracy(name="acc")],)
```

In [ ]:
```
model.fit(x_training, y_training, epochs=10, batch_size=128, validation_data=(x_test, y_test))
```

```
Epoch 1/10
WARNING:tensorflow:Model was constructed with shape (None, None, 3) for input KerasTensor(type_spec=TensorSpec(shape=(None, None, 3), dtype=tf.float32, name='dense_input'), name='dense_input', description="created by layer 'dense_input'"), but it was called on an input with incompatible shape (None, 3).
WARNING:tensorflow:Model was constructed with shape (None, None, 3) for input KerasTensor(type_spec=TensorSpec(shape=(None, None, 3), dtype=tf.float32, name='dense_input'), name='dense_input', description="created by layer 'dense_input'"), but it was called on an input with incompatible shape (None, 3).
2884/2886 [============================>.] - ETA: 0s - loss: 0.4480 - acc: 0.7768
WARNING:tensorflow:Model was constructed with shape (None, None, 3) for input KerasTensor(type_spec=TensorSpec(shape=(None, None, 3), dtype=tf.float32, name='dense_input'), name='dense_input', description="created by layer 'dense_input'"), but it was called on an input with incompatible shape (None, 3).
2886/2886 [==============================] - 9s 3ms/step - loss: 0.4480 - acc: 0.7768 - val_loss: 0.3746 - val_acc: 0.8251
Epoch 2/10
2886/2886 [==============================] - 8s 3ms/step - loss: 0.3389 - acc: 0.8455 - val_loss: 0.3085 - val_acc: 0.8672
Epoch 3/10
2886/2886 [==============================] - 8s 3ms/step - loss: 0.2984 - acc: 0.8672 - val_loss: 0.2941 - val_acc: 0.8777
Epoch 4/10
2886/2886 [==============================] - 8s 3ms/step - loss: 0.2742 - acc: 0.8797 - val_loss: 0.2875 - val_acc: 0.8776
Epoch 5/10
2886/2886 [==============================] - 8s 3ms/step - loss: 0.2567 - acc: 0.8891 - val_loss: 0.2564 - val_acc: 0.8857
Epoch 6/10
2886/2886 [==============================] - 8s 3ms/step - loss: 0.2436 - acc: 0.8954 - val_loss: 0.2429 - val_acc: 0.8940
Epoch 7/10
2886/2886 [==============================] - 8s 3ms/step - loss: 0.2317 - acc: 0.9012 - val_loss: 0.2285 - val_acc: 0.9026
Epoch 8/10
2886/2886 [==============================] - 8s 3ms/step - loss: 0.2211 - acc: 0.9069 - val_loss: 0.2098 - val_acc: 0.9111
Epoch 9/10
2886/2886 [==============================] - 9s 3ms/step - loss: 0.2115 - acc: 0.9110 - val_loss: 0.2115 - val_acc: 0.9092
Epoch 10/10
2886/2886 [==============================] - 10s 3ms/step - loss: 0.2031 - acc: 0.9159 - val_loss: 0.2021 - val_acc: 0.9191
```

Out[ ]:
```
<keras.callbacks.History at 0x7f18a12c2dc0>
```

In [ ]:
```
results = model.evaluate(x_test, y_test, batch_size=128)
print("test loss:", results[0])
print("test acc:", results[1])
```

```
321/321 [==============================] - 1s 2ms/step - loss: 0.2021 - acc: 0.9191
test loss: 0.2021140456199646
test acc: 0.9190651178359985
```

In [ ]:
```
predictions_prob = model.predict(x_test)
predictions_class = predictions_prob.argmax(axis=-1)
```

```
1283/1283 [==============================] - 2s 2ms/step
```

In [ ]:
```
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score, precision_score, recall_score, f1_score
```

In [ ]:
```
print(confusion_matrix(test_raw['condition_code'], predictions_class))
```

```
[[20067  2091]
 [ 1230 17645]]
```

In [ ]:
```
print(classification_report(test_raw['condition_code'], predictions_class))
```

```
              precision    recall  f1-score   support

           0       0.94      0.91      0.92     22158
           1       0.89      0.93      0.91     18875

    accuracy                           0.92     41033
   macro avg       0.92      0.92      0.92     41033
weighted avg       0.92      0.92      0.92     41033
```