M1 CPU

Summary

- **Key Question:** Can you detect if someone has COVID-19 based on their breathing patterns?. In order to answer this question, I needed to build models which can classify the six (6) breathing patterns of different samples:
  - normal breathing / Eupnea [1]
  - abnormal breathing such as:
    - Bradypnea [2]
    - Tachypnea [3]
    - Biots [4]
    - Cheyne-Stokes [5]
    - Central-Apnea [6]
- The rapid breathing pattern (Tachypnea) has been often reported as a feature of COVID-19.
- **Method:** Deep Learning techniques were trained on 120K training set that simulated respiratory patterns, and the test set (human test subjects recorded, Kinect V2 depth cameras) has 605 observations. Note, the study participants were trained to breath in the major six patterns and compared against gold-standard instruments. They are not COVID-19 patients, although one of the breathing patterns (Tachypnea) is a significant identifying symptom of the virus.
- **Techniques Used in the Original Study:** The original study focused on comparing four (4) deep learning neural networks for multiclass classification of the six repiratory patterns above. All neural networks used were Recurrent Neural Networks (RNNs), specifically: Gated Recurrent Unit (GRU), Long Short-Term Memory (LSTM), Bidirectional with Attention Layer LSTM (BI-AT-LSTM), and Bidirectional with Attention Layer GRU (BI-AT-GRU).
- **My Focus**: Attemping to reproduce results for the first three (GRU, LSTM, BI-AT-LSTM), largely utilizing Amazon Web Services (AWS) Sagemaker for high performance CPU and GPU (Tesla V100) infrastructure. Ultimately, I utilized the same neural network architecture, but had to adjust batch sizes.
- **Conclusion**: I could achieve results with similar metrics to the pre-print roughly (i.e. my results giving accuracy 77% - 83% instead of the 88% - 94%).
- **Original Pre-Print:**
  Wang, Yunlu; Hu, Menghan; Li, Qingli; Zhang, Xiao-Ping; Zhai, Guangtao; Yao, Nan. Abnormal respiratory patterns classifier may contribute to large-scale screening of people infected with COVID-19 in an accurate and unobtrusive manner. Download Original Paper

Import Libraries

- **Metrics:** SkLearn
- **Model Loading:** Keras Model Loading
- **Plotting:** MatPlotLib and SciKitPLot

In [1]:
```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report, confusion_matrix

from keras.models import load_model

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

import matplotlib.pyplot as plt
import scikitplot as skplt
```

Import Data

- Import Procedure
  - Import Test dataset of 605 observations, with the first column as the respiratory pattern class (1-6), and the next 600 columns as respiratory pattern predictors. Each predictor is a measure of respiratory pattern waveform amplitude every centi-second, thus alltogether describing a waveform over a total duration of 60 seconds and 10Hz. Prior to import by the original study authors, various data transformation steps of normalization and other signal filters are applied, as described in the papers referenced below.
- High-level notes on datasets used:
  - Training Set was generated by the original publication's MATLAB simulator code. Please see "Project 2" for details. The models loaded were trained on this training set.
  - Test Set is the original publication's test set. Please see details below.
- Dataset Details:
  - **Publication:**
    Wang, Yunlu; Hu, Menghan; Li, Qingli; Zhang, Xiao-Ping; Zhai, Guangtao; Yao, Nan. Abnormal respiratory patterns classifier may contribute to large-scale screening of people infected with COVID-19 in an accurate and unobtrusive manner. Download Original Paper
  - **Test Dataset:** Hu, Menghan (2020): Test dataset for Download Original Paper figshare. Dataset. Dowload Test Data
  - **Train Dataset**: Hu, Menghan (2019): Respiratory Pattern Simulation_ Model (RSM). figshare. Dataset. Download Training Simulator
  - Note, "main.m" of RSM code was modified from "scale = 10" to "scale = 20000" to generate the original 120,000 results.

In [2]:
```
testSource = pd.read_csv("./Breathing_TEST.csv", header = None)
testDF = testSource
x_test = testDF.iloc[:, 1:601].to_numpy()
y_test = testDF.iloc[:, 0].to_numpy()
```

```
testDF.info()
testDF.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 605 entries, 0 to 604
Columns: 601 entries, 0 to 600
dtypes: float64(600), int64(1)
memory usage: 2.8 MB
```

Out[2]:

	0	1	2	3	4	5	6
count	605.000000	605.000000	605.000000	605.000000	605.000000	605.000000	605.000000
mean	3.409917	0.486449	0.498953	0.502535	0.502367	0.500512	0.494951
std	1.719743	0.323598	0.291186	0.277246	0.276744	0.279495	0.282198
min	1.000000	0.000000	0.000000	0.002593	0.000000	0.000000	0.000000
25%	2.000000	0.210650	0.258800	0.266970	0.256240	0.251420	0.255330
50%	3.000000	0.493430	0.496110	0.507520	0.515030	0.514190	0.498510
75%	5.000000	0.761350	0.740740	0.730350	0.735880	0.733690	0.742520
max	6.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

8 rows × 601 columns

Helper Functions

- Functions to help load models that we have alread trained, utilizing the industry standard HDF5 format, see https://en.wikipedia.org/wiki/Hierarchical_Data_Format#HDF5
  - Functions include: modelSummary, confusionMatrix (absolute and normalized), plotRocAuc, and plotPrecisionRecall
  - Please see SciKitPLot, a helpful library to automatically plot ROC/AUC for multi-class problems, https://github.com/reiinakano/scikit-plot. Note, this is likely based on ideas in SciKit Learn's example on adapting ROC/AUC for multi-class classification problems, https://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html

In [3]:

```
def modelSummary(modelPath):
    model = load_model(modelPath)
    model.summary()

def confusionMatrix(modelPath, normBool):
    model = load_model(modelPath)
    y_prob = model.predict(x_test)
    predictions = y_prob.argmax(axis=-1)
    predictions = predictions.reshape(len(testDF),)
    skplt.metrics.plot_confusion_matrix(y_test, predictions, normalize=normBool)
    plt.show()

def classificationReport(modelPath):
    model = load_model(modelPath)
    y_prob = model.predict(x_test, batch_size=32)
    predictions = y_prob.argmax(axis=-1)
    predictions = predictions.reshape(len(testDF),)
    print(classification_report(y_test, predictions, labels=[1,2,3,4,5,6], zero_

def plotRocAuc(modelPath):
    model = load_model(modelPath)
    y_prob = model.predict(x_test)
    skplt.metrics.plot_roc(y_test, y_prob[:,1:7])
    plt.show()

def plotPrecisionRecall(modelPath):
    model = load_model(modelPath)
    y_prob = model.predict(x_test)
    skplt.metrics.plot_precision_recall(y_test, y_prob[:,1:7])
    plt.show()
```

Gated Recurrent Unit (GRU)

- Model Summary
- Classification Report
- Confusion Matrix (Normalized)
- Confusion Matrix (Absolute)
- Receiver Operating Characteristic (RoC) Curves and Area Under Curve (AUC)
- Precision-Recall Curvies

- Additional Info on Model Types
  - https://en.wikipedia.org/wiki/Gated_recurrent_unit
  - https://en.wikipedia.org/wiki/Recurrent_neural_network

GRU - Model Summary

- Key Hyperparameters
  - Batch Size: 128

In [17]:

```
modelSummary("./gru/best_model.h5")
```

```
Model: "respiratory"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 600, 1)]          0
_________________________________________________________________
gru (GRU)                    (None, 128)               50304
_________________________________________________________________
dense (Dense)                (None, 7)                 903
=================================================================
Total params: 51,207
Trainable params: 51,207
Non-trainable params: 0
_________________________________________________________________
```

GRU - Classification Report

- See links:
  - https://muthu.co/understanding-the-classification-report-in-sklearn/
  - https://en.wikipedia.org/wiki/Precision_and_recall

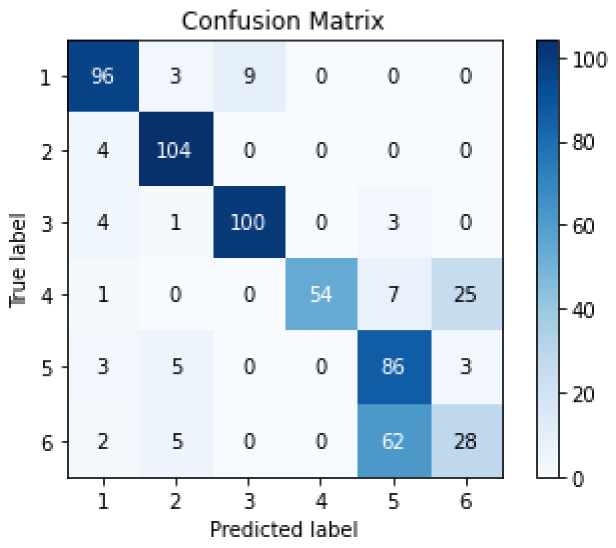
In [18]:
```
classificationReport("./gru/best_model.h5")
```

```
              precision    recall  f1-score   support

           1       0.87      0.89      0.88       108
           2       0.88      0.96      0.92       108
           3       0.92      0.93      0.92       108
           4       1.00      0.62      0.77        87
           5       0.54      0.89      0.67        97
           6       0.50      0.29      0.37        97

    accuracy                           0.77       605
   macro avg       0.79      0.76      0.75       605
weighted avg       0.79      0.77      0.76       605
```

GRU - Confusion Matrix (Absolute)

- See https://en.wikipedia.org/wiki/Confusion_matrix

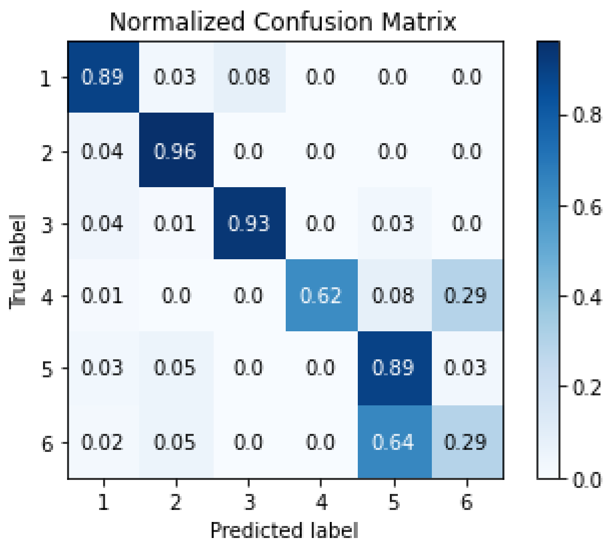
In [19]:
```
confusionMatrix("./gru/best_model.h5", False)
```



GRU - Confusion Matrix (Normalized)

- See https://en.wikipedia.org/wiki/Confusion_matrix

In [20]:
```
confusionMatrix("./gru/best_model.h5", True)
```

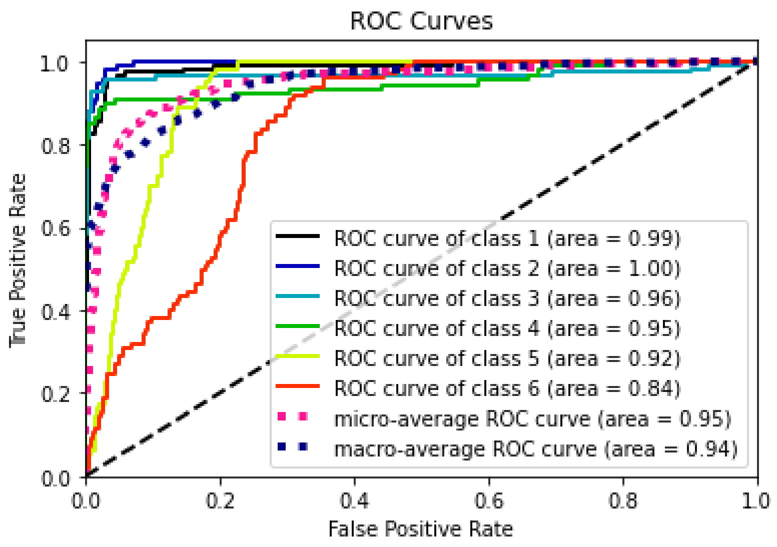


	1	2	3	4	5	6
1	0.89	0.03	0.08	0.0	0.0	0.0
2	0.04	0.96	0.0	0.0	0.0	0.0
3	0.04	0.01	0.93	0.0	0.03	0.0
4	0.01	0.0	0.0	0.62	0.08	0.29
5	0.03	0.05	0.0	0.0	0.89	0.03
6	0.02	0.05	0.0	0.0	0.64	0.29

GRU - Receiver Operating Characteristic (RoC) Curves and Area Under Curve (AUC)

- See links:
  - https://en.wikipedia.org/wiki/Receiver_operating_characteristic
  - https://en.wikipedia.org/wiki/Receiver_operating_characteristic#Area_under_the_curve
  - https://machinelearningmastery.com/roc-curves-and-precision-recall-curves-for-classification-in-python/

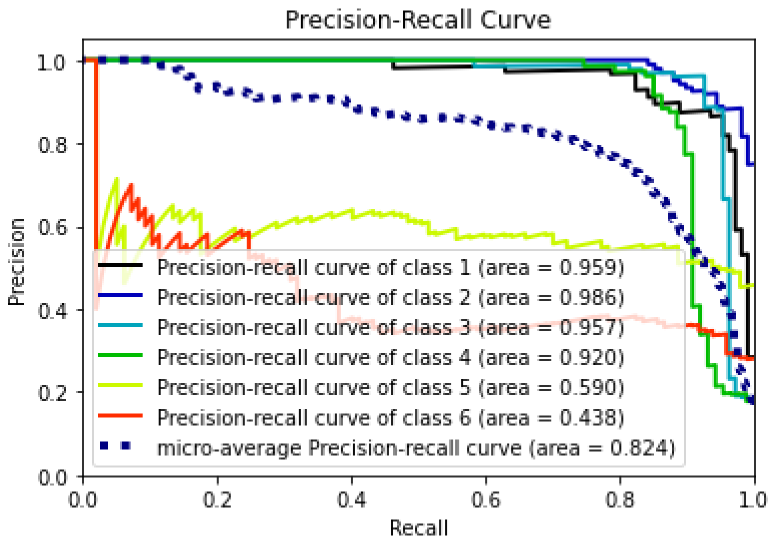
In [21]:
```
plotRocAuc("./gru/best_model.h5")
```



GRU - Precision-Recall Curve

- See links:
  - https://machinelearningmastery.com/roc-curves-and-precision-recall-curves-for-classification-in-python/
  - https://en.wikipedia.org/wiki/Precision_and_recall

In [22]:
```
plotPrecisionRecall("./gru/best_model.h5")
```



Long Short-Term Memory (LSTM)

- Model Summary
- Classification Report
- Confusion Matrix (Normalized)
- Confusion Matrix (Absolute)
- Receiver Operating Characteristic (RoC) Curves and Area Under Curve (AUC)
- Precision-Recall Curvies

- Additional Info on Model Types
  - https://en.wikipedia.org/wiki/Long_short-term_memory
  - https://en.wikipedia.org/wiki/Recurrent_neural_network

LSTM - Model Summary

- Key Hyperparameters
  - Batch Size: 32

In [23]:
```
modelSummary("./lstm/best_model.h5")
```

```
Model: "respiratory"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 600, 1)]          0
_________________________________________________________________
lstm (LSTM)                  (None, 128)               66560
_________________________________________________________________
```

```
dense (Dense)                (None, 7)                 903
=================================================================
Total params: 67,463
Trainable params: 67,463
Non-trainable params: 0
_________________________________________________________________
```

LSTM - Classification Report

- See links:
  - https://muthu.co/understanding-the-classification-report-in-sklearn/
  - https://en.wikipedia.org/wiki/Precision_and_recall

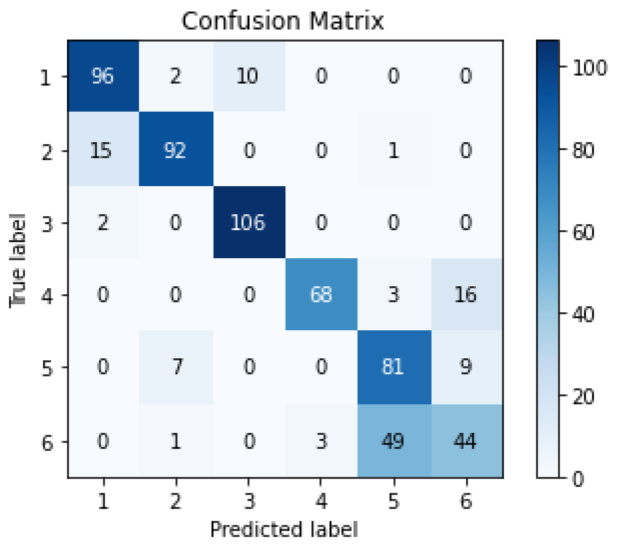
In [12]:
```
classificationReport("./lstm/best_model.h5")
```

```
              precision    recall  f1-score   support

           1       0.85      0.89      0.87       108
           2       0.90      0.85      0.88       108
           3       0.91      0.98      0.95       108
           4       0.96      0.78      0.86        87
           5       0.60      0.84      0.70        97
           6       0.64      0.45      0.53        97

    accuracy                           0.80       605
   macro avg       0.81      0.80      0.80       605
weighted avg       0.81      0.80      0.80       605
```

LSTM - Confusion Matrix (Absolute)

- See https://en.wikipedia.org/wiki/Confusion_matrix

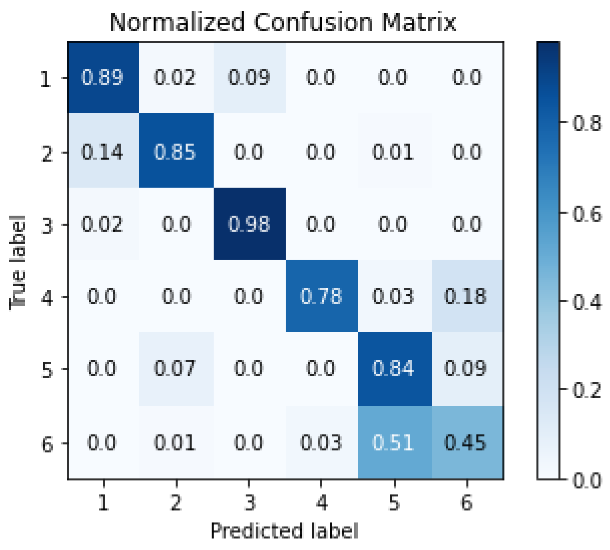
In [13]:
```
confusionMatrix("./lstm/best_model.h5", False)
```



LSTM - Confusion Matrix (Normalized)

- See https://en.wikipedia.org/wiki/Confusion_matrix

In [14]:
```
confusionMatrix("./lstm/best_model.h5", True)
```

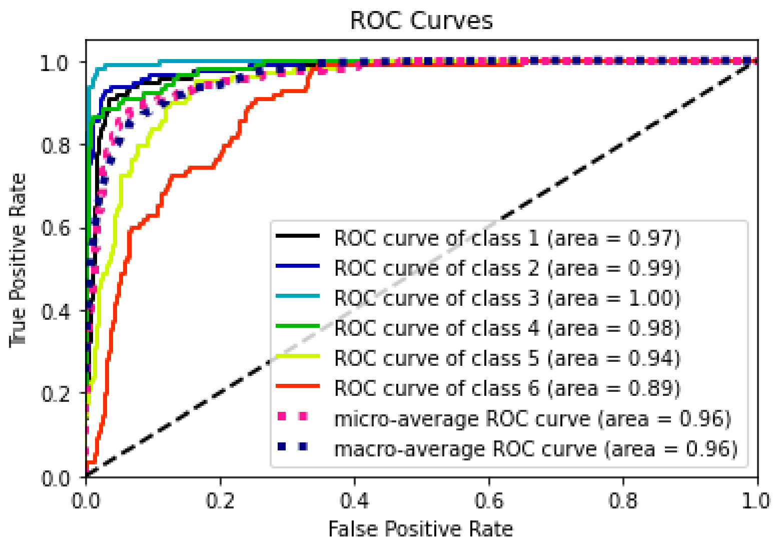


	1	2	3	4	5	6
1	0.89	0.02	0.09	0.0	0.0	0.0
2	0.14	0.85	0.0	0.0	0.01	0.0
3	0.02	0.0	0.98	0.0	0.0	0.0
4	0.0	0.0	0.0	0.78	0.03	0.18
5	0.0	0.07	0.0	0.0	0.84	0.09
6	0.0	0.01	0.0	0.03	0.51	0.45

LSTM - Receiver Operating Characteristic (RoC) Curves and Area Under Curve (AUC)

- See links:
  - https://en.wikipedia.org/wiki/Receiver_operating_characteristic
  - https://en.wikipedia.org/wiki/Receiver_operating_characteristic#Area_under_the_curve
  - https://machinelearningmastery.com/roc-curves-and-precision-recall-curves-for-classification-in-python/

In [15]:
```
plotRocAuc("./lstm/best_model.h5")
```

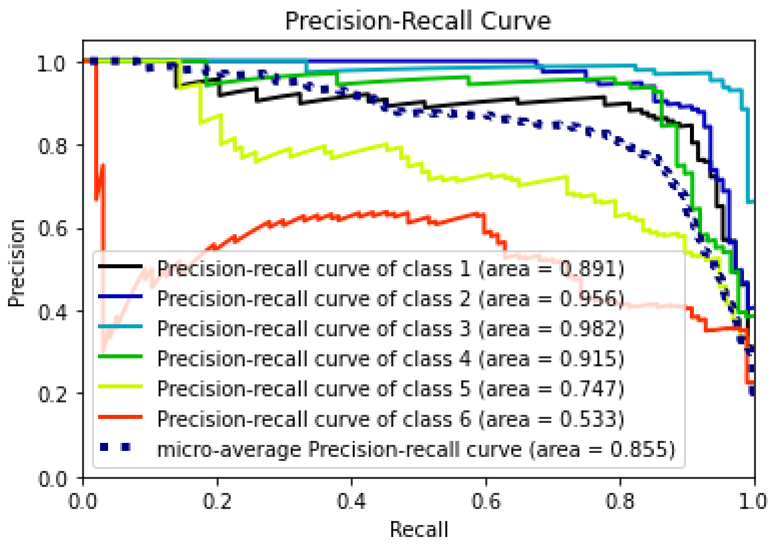


LSTM - Precision-Recall Curve

- See links:
  - https://machinelearningmastery.com/roc-curves-and-precision-recall-curves-for-classification-in-python/
  - https://en.wikipedia.org/wiki/Precision_and_recall

In [16]:

```
plotPrecisionRecall("./lstm/best_model.h5")
```


Precision-Recall Curve
Precision
Recall
Precision-recall curve of class 1 (area = 0.891)
Precision-recall curve of class 2 (area = 0.956)
Precision-recall curve of class 3 (area = 0.982)
Precision-recall curve of class 4 (area = 0.915)
Precision-recall curve of class 5 (area = 0.747)
Precision-recall curve of class 6 (area = 0.533)
micro-average Precision-recall curve (area = 0.855)