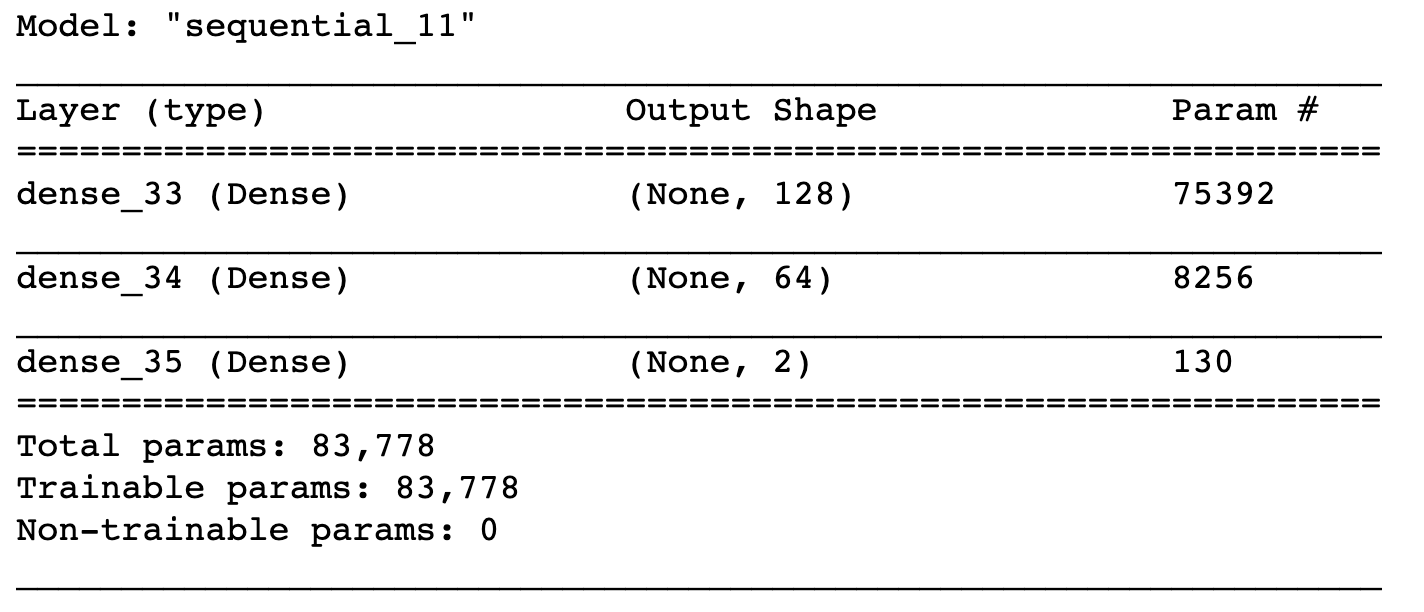
Steps For Cardiac Shock Prediction

1. We load both the data with features (let’s call this dataset df )and the metadata (let’s call this dataset meta).
2. We remove the data for days within 10 days of the cardiac shock from df. We are left with the data for days 11 (inclusively) to 31 (inclusively).
3. For each person, we have up to 21 rows of features (we have data for a maximum of 31 days and we removed data for the first 10 days). Each row has 28 columns of features. For each person, we will aggregate features by concatenating the rows for each day. As a result, for each person, we have 588 (21 \* 28) columns of features.
   1. Note that we may have missing data for certain days for a person. In that case, we will insert NaN for all features for days where there are no available data.
   2. Note that there are 2225 people. Therefore, the resulting data is of size 2225 by 588. Let’s call this data df2.
4. Next, we split df2 into a train set (call this train\_set) and a test set (call this test\_set), where the test set consists of 20% of the total data.
5. For each feature (or each column) of train\_set, we find the mode value. We replace NaNs with the mode value in train\_set. Next, we replace NaNs in the test\_set using the mode values we gathered from the train\_set.
6. We built and trained a neural network with three layers.
   1. The first layer has 128 hidden units, and the second has 64 hidden units, and the last layer has 2 hidden units. ‘Relu’ activations follow layer 1 and 2, and the ‘softmax’ activation function follows layer 3. The description of the model is shown below.
   2. 
   3. We train the model using the ‘Adam’ optimizer and we use categorical cross-entropy as the loss function. We trained for 20 epochs.
   4. In addition, we weighted the training samples based on their frequency. For example, if we have 1000 training instances with label ‘yes’ and 100 training instances with label ‘no’, then the labels with ‘no’ are weighted 10 times as much as the training instances with label ‘yes’.
7. We evaluated the trained model on the test set, and achieved an accuracy of 93.0% and ROC-AUC score of 91.3%.
8. We saved the model and its weights.

Extra Notes:

* We used Python packages:
  + Tensorflow version 2.0.0
  + Scikit-learn version 0.22
* We used Python 3.