**Neural Network Modeling**

**Preprocessing:**

We removed data points that are computed within 10 days of the cardiac shock. Using the remaining data, we consider each feature from each day as an individual feature. Next, we split the data into a training and a testing set, where the testing set consists of 20% of the total data. We replaced missing values in the training set with the mode values in each feature. Using mode for imputation is a standrad design choice. We also replaced missing values in the testing set using the mode values we computed from the training set, thus making sure that we do not use any information that will not be available in actual deployment.

**Model Training:**

We build a feed-forward neural network model with three layers; the first layer has 128 hidden units, the second has 64 units, and the last layer has 2 units. ReLU activation [1] is used in the first and second layer, and softmax activation [2] in the last layer. Using a softmax activation in the last layer allows us to produce “probabilities” in th eoutput, which can then be used to generate a ROC curve. In total, there are about 84,000 parameters. We use the ADAM optimizer with a learning rate of 0.001 to train the model for 20 epochs. During training, we reweighed the training instances based on their label frequency.

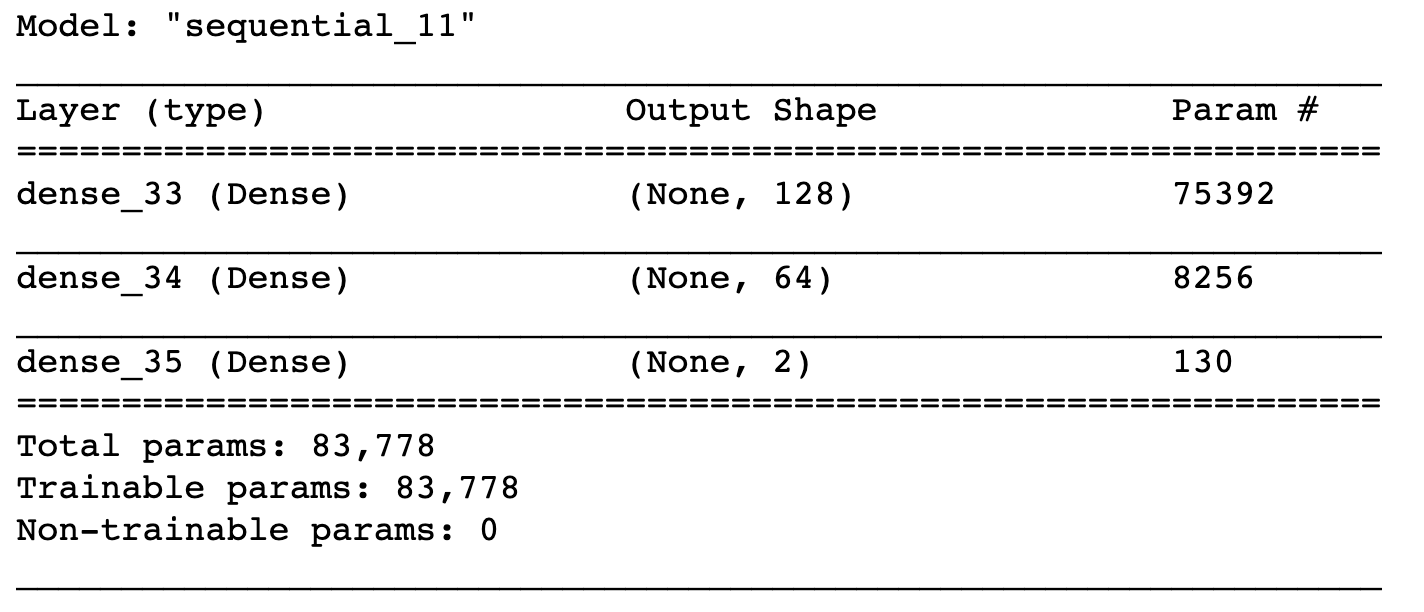
**Model Evaluation:**

Evaluating the neural network on the test data, we achieved an accuracy of 93% and an ROC-AUC of 91.3% (See Fig. 2)

We use SHAP Values (an acronym from SHapley Additive exPlanations) [3] break down a prediction to show the impact of each feature (See Fig. 1)

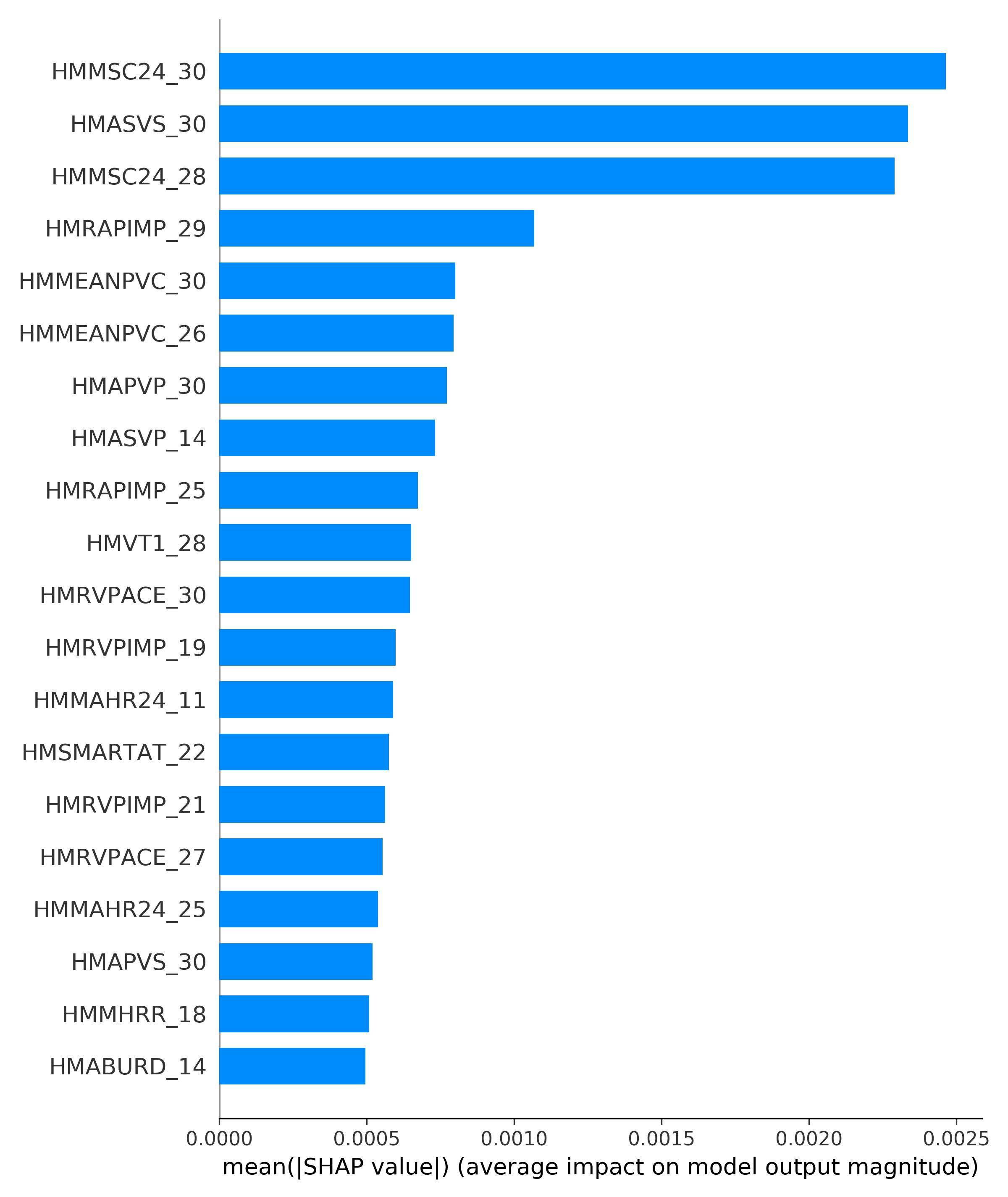
**Steps For Cardiac Shock Prediction Using Neural Networks**

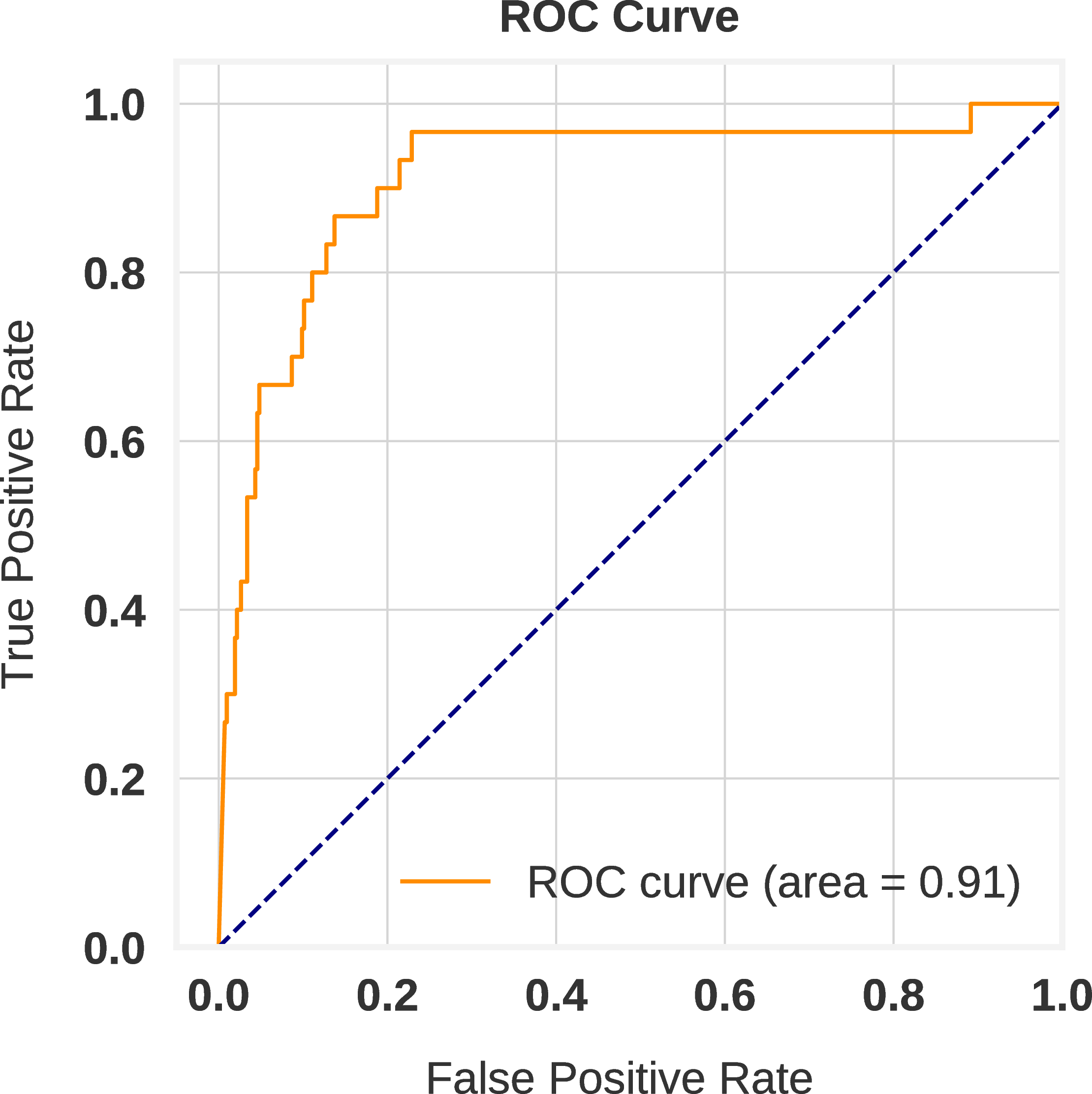
We deonote the observations as the dataset **df**, and the metadata on the samples as **meta**. We proceed as follows:

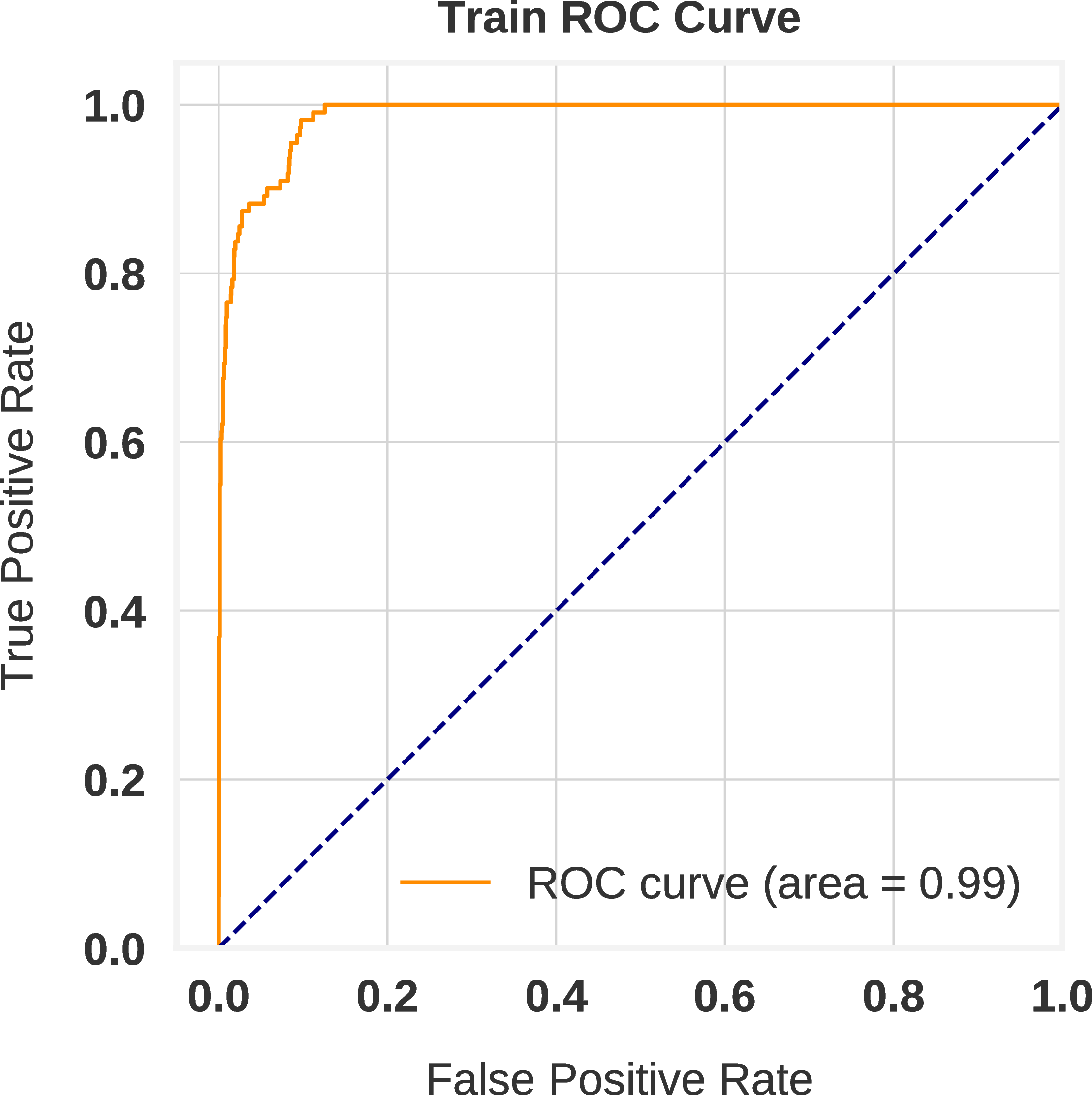
1. We remove the data for days within 10 days of the cardiac shock from df. We are left with the data for days 11 (inclusively) to 31 (inclusively).
2. For each person, we have up to 21 days of observations (rows of features, we have data for a maximum of 31 days and we removed data for the first 10 days). Each feature 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 set df2.
3. Next, we split df2 into a training set (call this train\_set) and a test set (call this test\_set), where the test set consists of 20% of the total data.
4. 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.
5. 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’.
6. We evaluated the trained model on the test set, and achieved an accuracy of 93.0% and ROC-AUC score of 91.3%.
7. We saved the model and its weights.

**Notes on software used:**

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

**Fig1. Using SHAP values to identify the importance of different features.**

****

****

**Fig 2. ROC Curves with training and test data**

**References**

1. Nair, Vinod, and Geoffrey E. Hinton. "Rectified linear units improve restricted boltzmann machines." In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 807-814. 2010.

2. Sutton, Richard S., and Andrew G. Barto. Reinforcement learning: An introduction. MIT press, 2018.

3. Lundberg, Scott M., and Su-In Lee. "A unified approach to interpreting model predictions." In *Advances in neural information processing systems*, pp. 4765-4774. 2017.