* **Disclaimer:** All of the data processing and model training for this challenge was done on my personal laptop. As the laptop’s hardware is not ideal for machine learning, what is good or viable for me might not be the same for someone who has access to a better computer, or does their training on a cloud server.

This model achieves the ROC AUC score of approximately 0.63-0.65 on the validation test of 20% the training set when the prediction label is converted to 0-1 labels with 0.5 threshold. When calculating ROC AUC score with neural network probability output (not converted to 0-1 labels), ROC AUC score is around 0.70- 0.71 (depending on specific train-test split).

* **Some approaches I have tried in the challenge:**

\_ Extracting statistics features from each sample set, such as mean, minimum, maximum, standard deviation, etc. of each feature. Afterward, applying classifier to the resulting matrix (I tried random forest classifier and artificial neural network). I intended to use this approach as a base model, and not surprisingly it did not work very well. ROC AUC scores were between 0.52 and 0.54. I also tried to use SMOTE synthetic sampling to tackle the classes imbalance problem, but performance did not improve significantly.

\_ I did some further research on this feature extraction, and decided to use the Tsfresh module to automatically extract the relevant features. Unfortunately, due to the size of the dataset and my hardware limitation, I constantly ran into memory errors. I tried this method of a smaller randomized subset of the dataset (e.g. 1 million rows); it still took very long to run and did not yield great performance. It’s possible that with more computing power this method might be viable.

\_ I tried to use recurrent neural network, in particular Long-Short Term Memory (LTSM) which is recommended by a lot of literatures as a great method for time series classification. For some reason, it did not work for this problem. When trained with the original training split, the model always converged to naïve prediction (all negatives). When training data was balanced with either oversampling or undersampling, the model performance was still not great; training accuracy could not increase past around 55%.

\_ Some literature I read suggested to add a convolutional layer before LSTM. I tried and it did not work; however, I thought of building a convolutional neural network without LSTM. This approach worked the best among the ones I have tried. Because I discovered this approach quite close to the deadline, and training time is long, the model structure and parameters are probably not the most finely tuned as they could be. One parameter that I have tuned quite a bit is the learning rate. Lower or higher learning rates did not seem to work as well as the one I chose.

* **Model limitations:**

A ROC AUC score of around 0.7 is not ideal. I suspect that with some parameter tuning, I can increase the score by a few percentages. However, for a significantly better score, this problem will probably require a new approach. There are a few methods I have read about but have not tried due to time constraint. I will keep exploring this problem, and I would like to receive any suggestion for a solution.