# ML4DQM Updates

Pritam Palit, Shamik Ghosh, Subir Sarkar
Saha Institute of Nuclear Physics, India
11/05/2020

# Topics to report

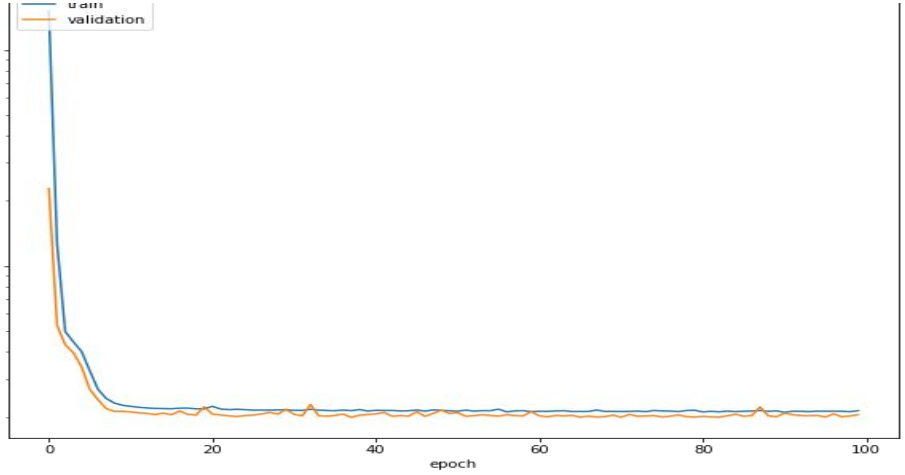
Anomaly detection with Autoencoder for 2017 data

- As a beginner, I first used Autoencoder architecture as given in Francesco's Code for : chargeInner_PXLayer_1
  - Checked the distribution of Reconstruction Error (mseTop10)
  - Checked the Precision-Recall curve , F1 score and corresponding best Threshold
  - Checked the Confusion Matrix
- Then, I checked the sequence of the consecutive bins and decided to use LSTM Autoencoder
  - Compared the results.

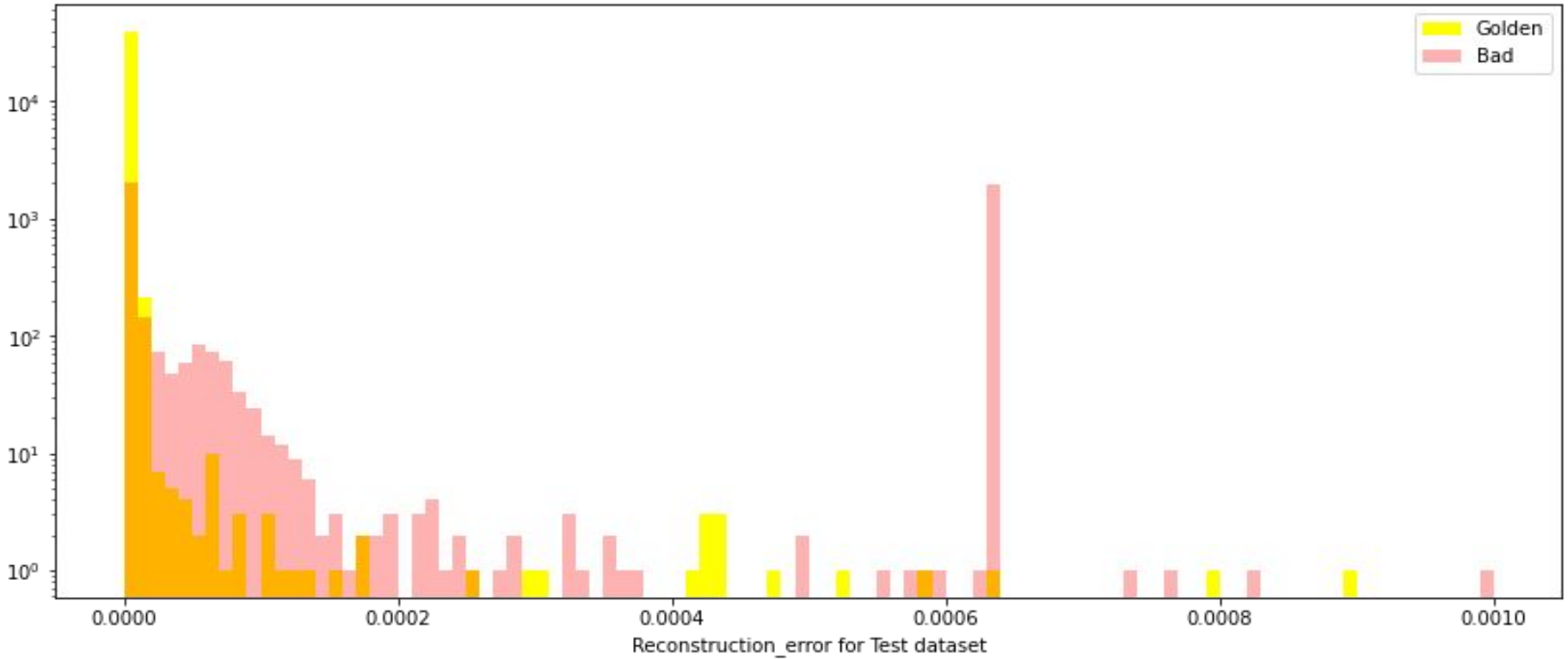
# Standard Autoencoder

- The total dataset is split into Train - Validation - Test ( 72 - 8 - 20 )%
  - We Trained only on Good runs and lumis of Train dataset
  - Good runs and lumis selected from Golden json file
  - Test is done on the whole test dataset (including both good and bad )
- Standard DNN based autoencoder with tanh activation in all layers :
  input - 10 - 3 - 10 - output
- Optimizer = adam , loss = mseTop10 (Mean of Top 10 highest error)

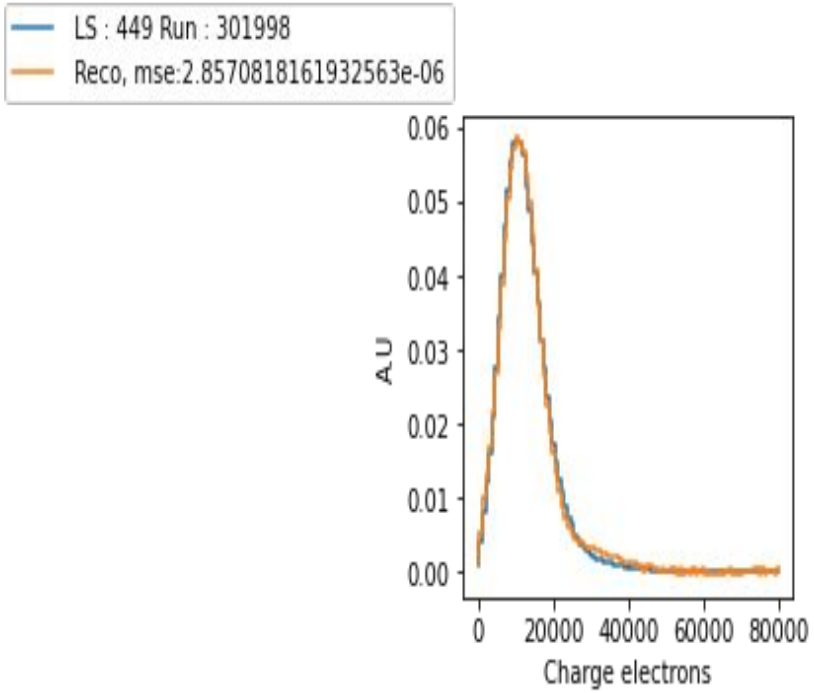
```
Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 100)               0
_________________________________________________________________
dense_1 (Dense)              (None, 10)                1010
_________________________________________________________________
dense_2 (Dense)              (None, 3)                 33
_________________________________________________________________
dense_3 (Dense)              (None, 10)                40
_________________________________________________________________
dense_4 (Dense)              (None, 100)               1100
=================================================================
Total params: 2,183
Trainable params: 2,183
Non-trainable params: 0
```

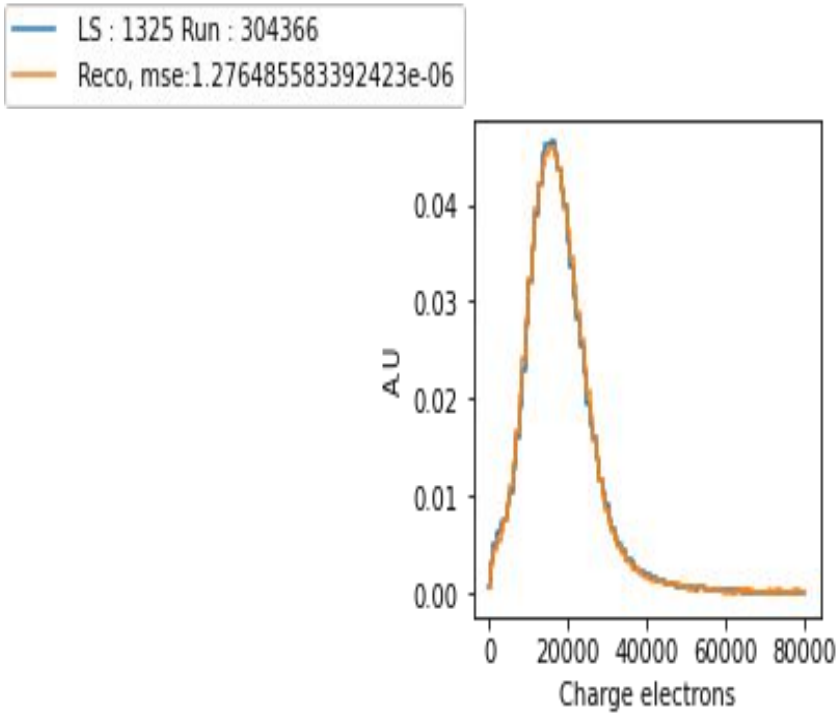


# Reconstruction error For Test Data

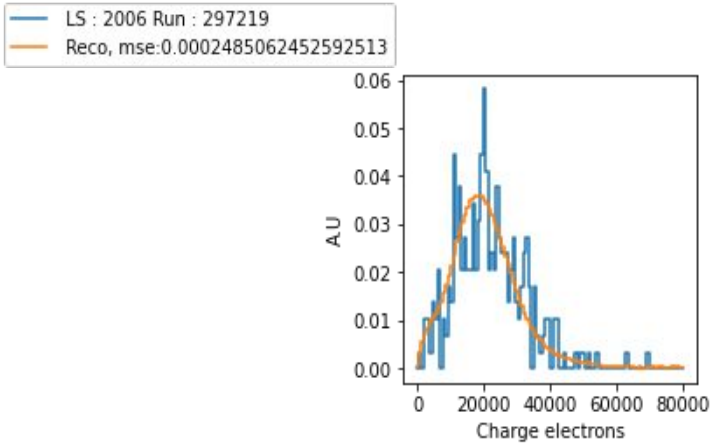


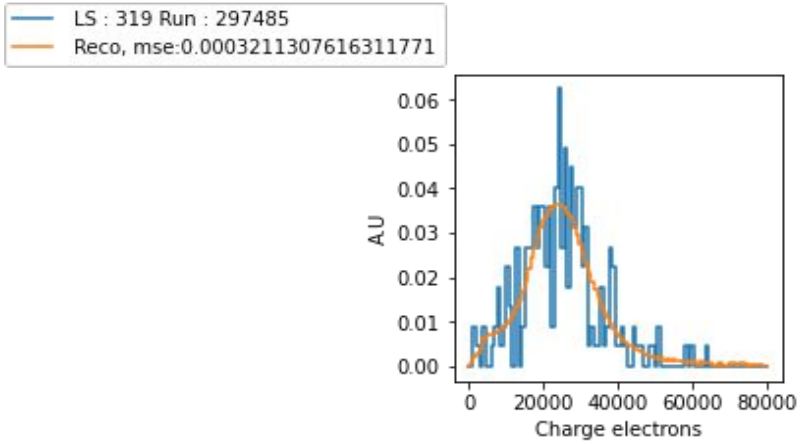
# Error < 0.0002

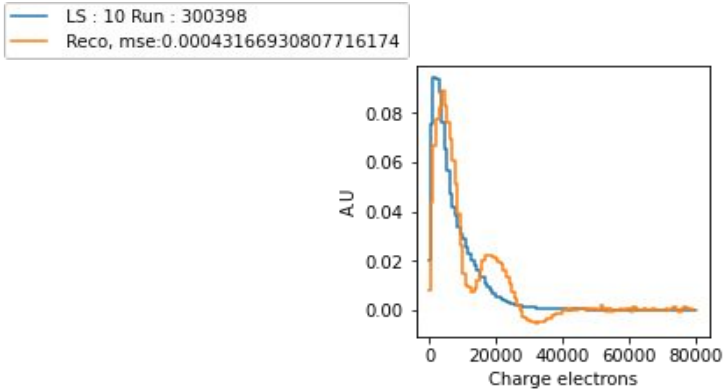

LS : 449 Run : 301998
Reco, mse:2.8570818161932563e-06
A.U
0.06
0.05
0.04
0.03
0.02
0.01
0.00
0
20000
40000
60000
80000
Charge electrons

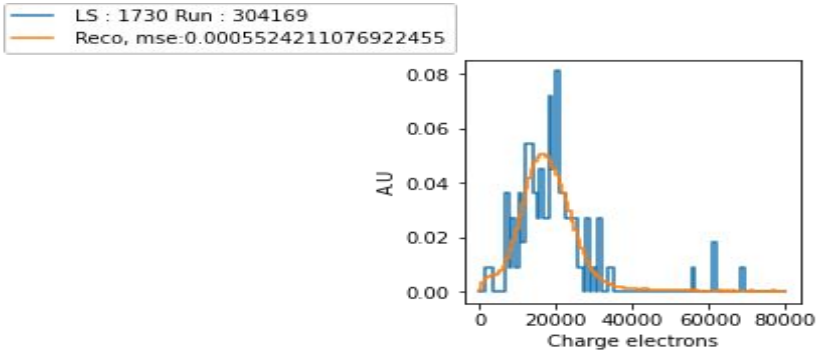

LS : 1325 Run : 304366
Reco, mse:1.276485583392423e-06
A.U
0.04
0.03
0.02
0.01
0.00
0
20000
40000
60000
80000
Charge electrons

# 0.0002 < Error < 0.0006

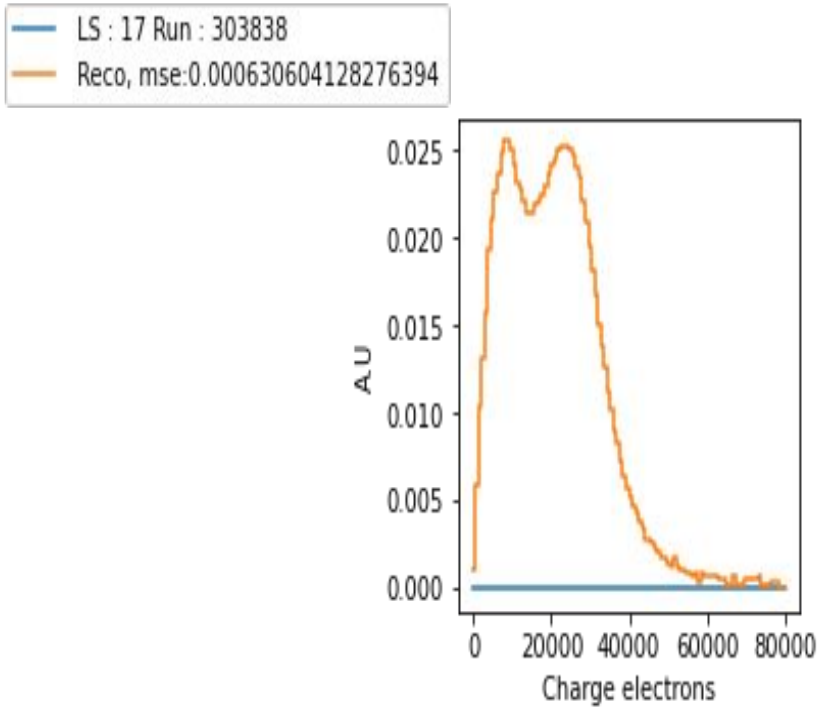


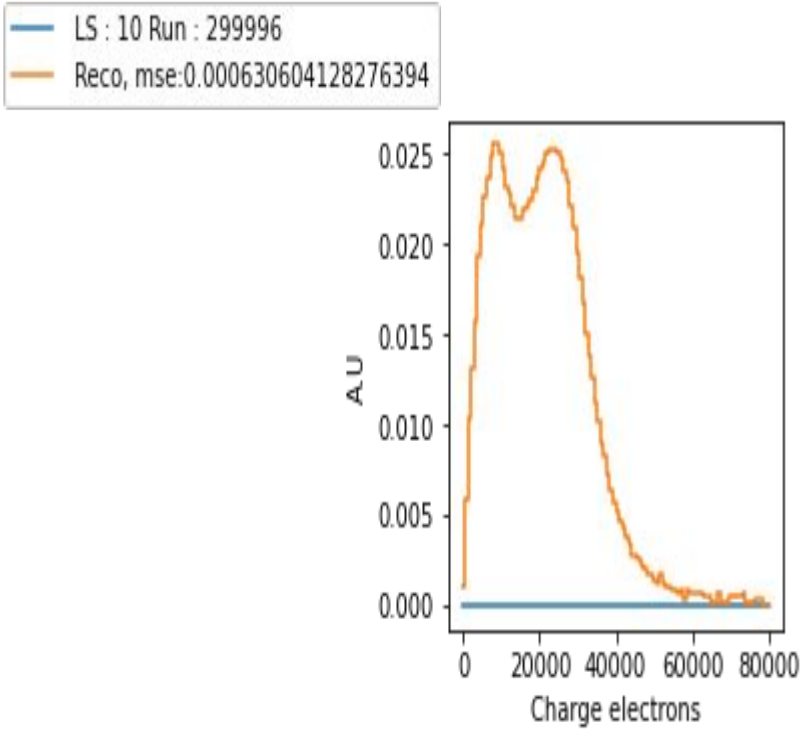



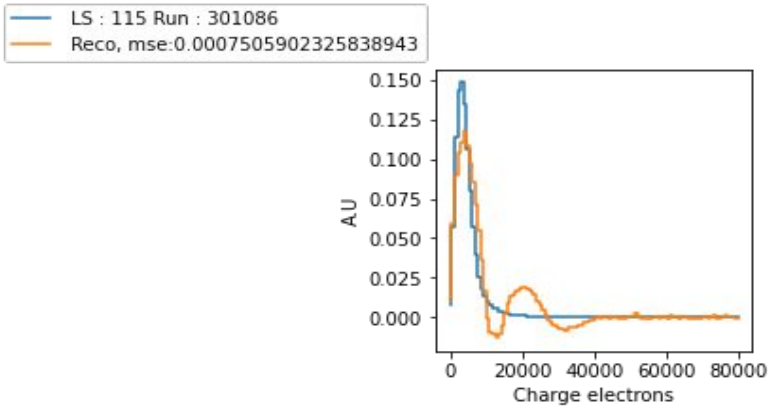



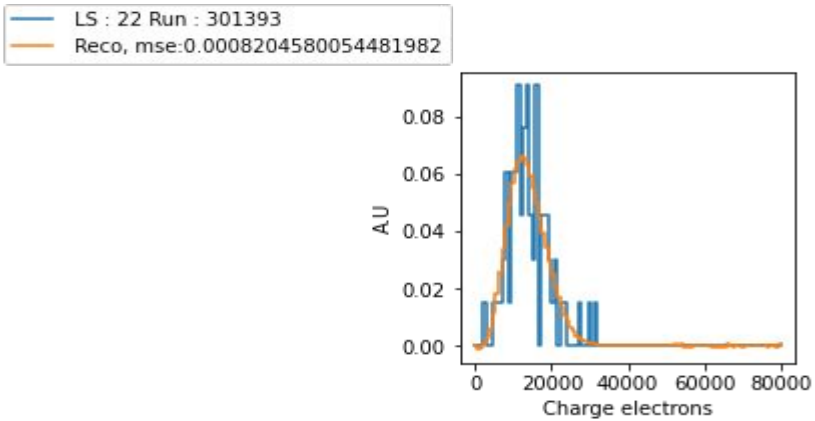
# 0.0006 < Error < 0.0007

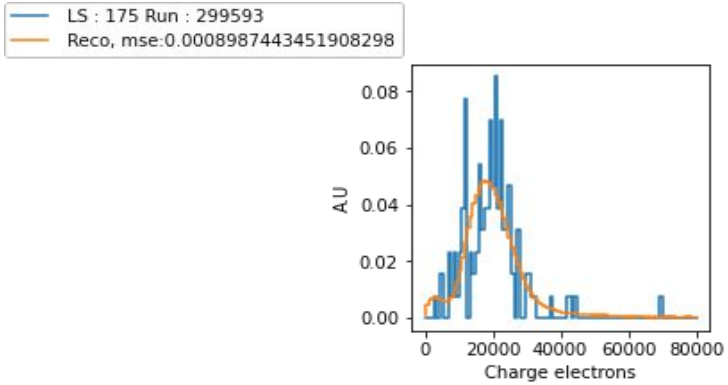


# 0.0007 < Error < 0.0009

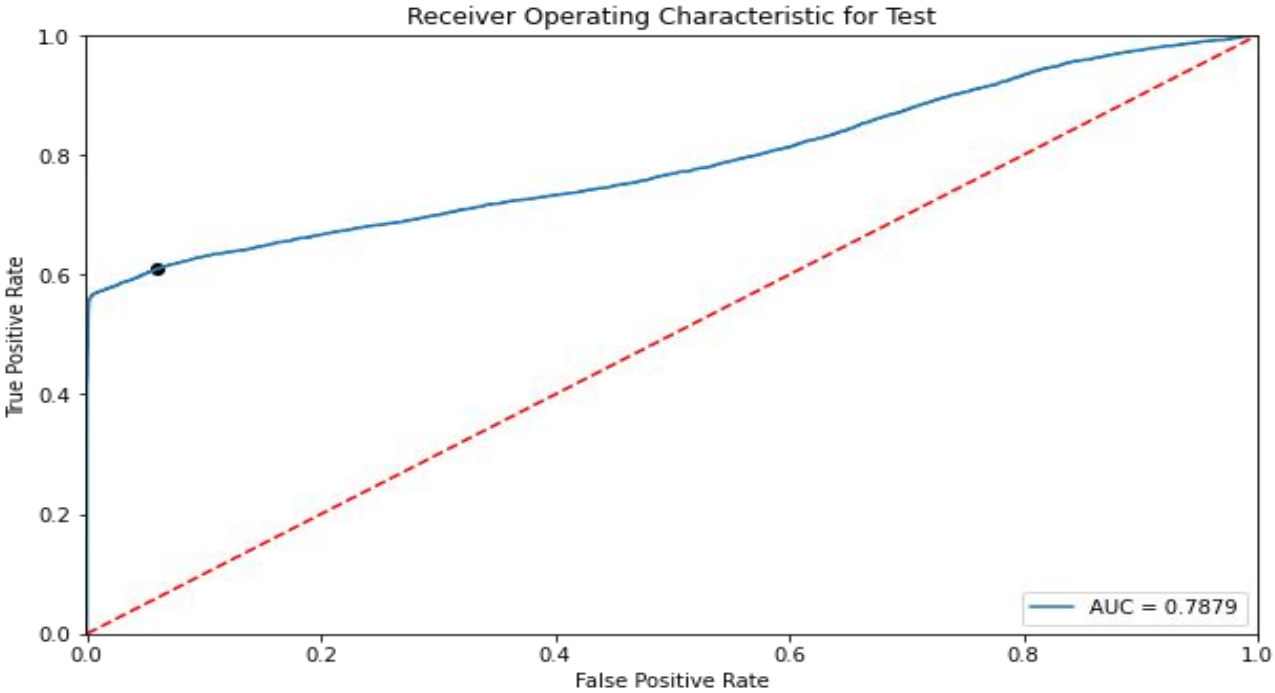


# ROC Curve

$$true\ positive\ rate = \frac{true\ positives}{true\ positives + false\ negatives} \qquad false\ positive\ rate = \frac{false\ positives}{false\ positives + true\ negatives}$$




Here :

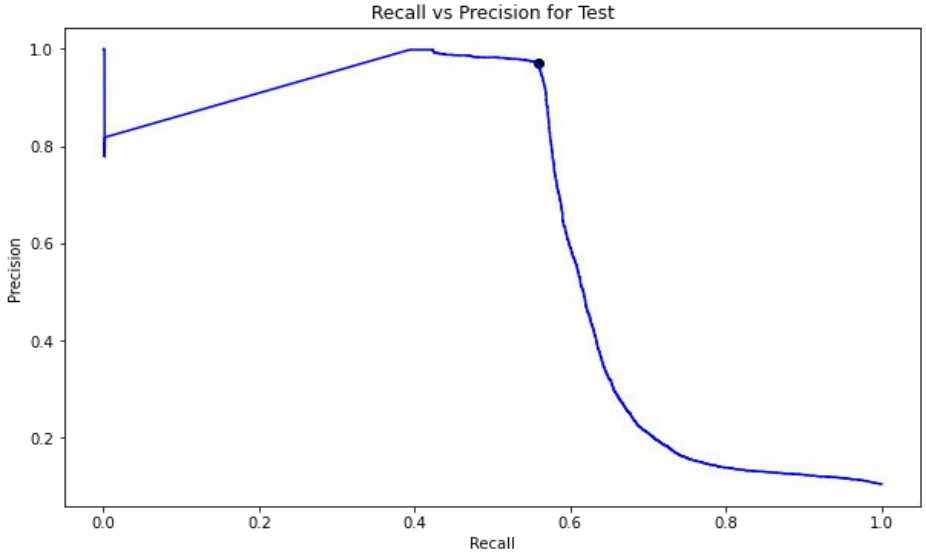
Positive = Anomaly

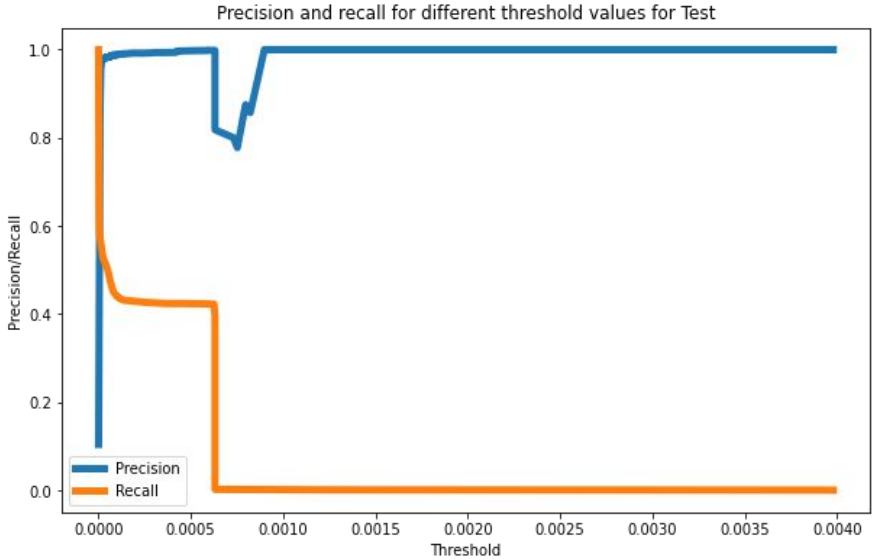
Negative = Good

# Precision - Recall (PR) Curve for Imbalanced Dataset

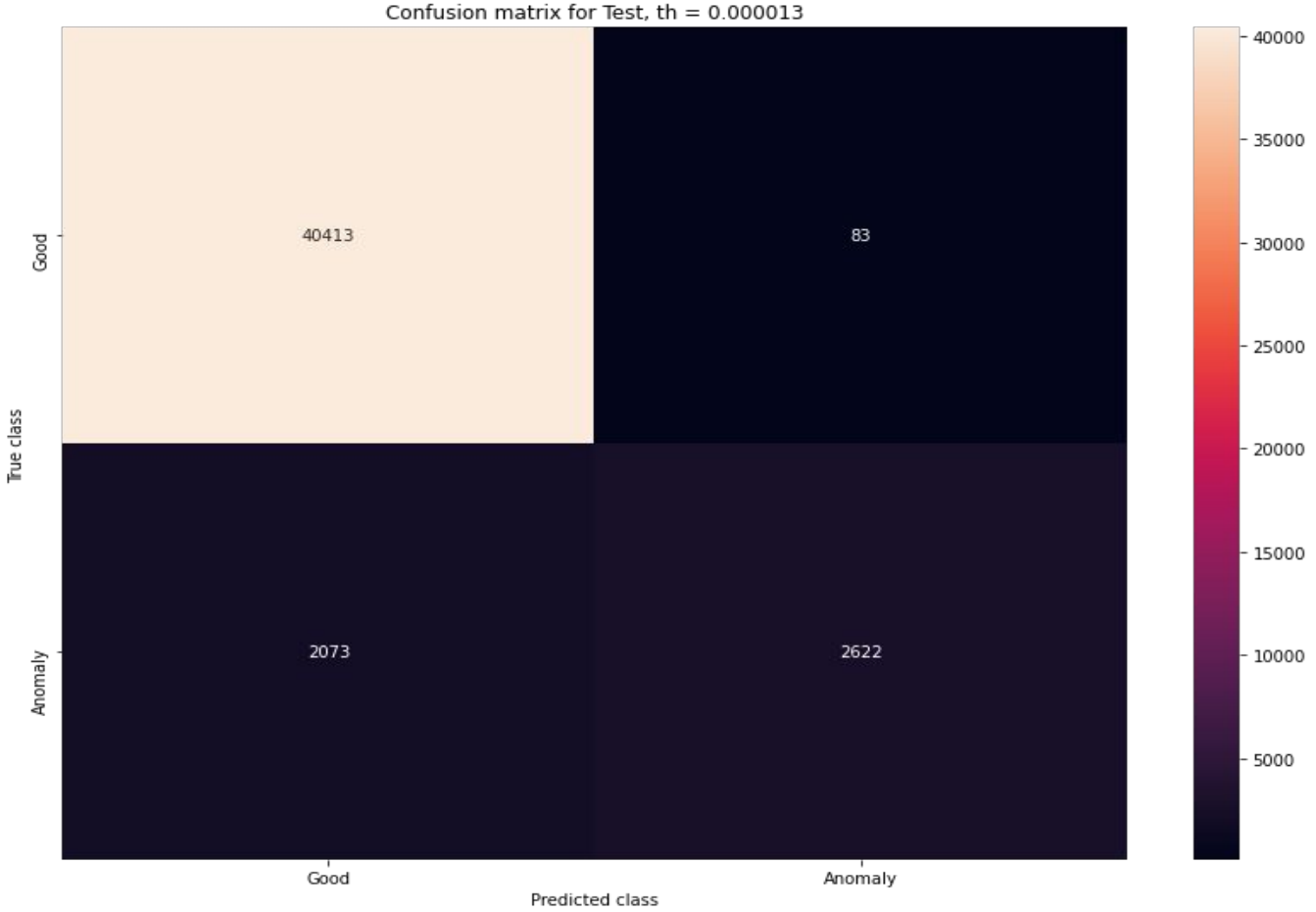
$$recall = \frac{true\ positives}{true\ positives\ + false\ negatives} \qquad precision = \frac{true\ positives}{true\ positives + false\ positives}$$

- PR curve for anomaly detection in a imbalanced dataset
  - Less interested in true negative (as well as fpr of ROC curve)
- `fscore = (2 * precision * recall) / (precision + recall)`
- **Best Threshold=0.000013 => F-Score=0.709**



# Confusion Matrix




Want to reduce both F.P & F.N

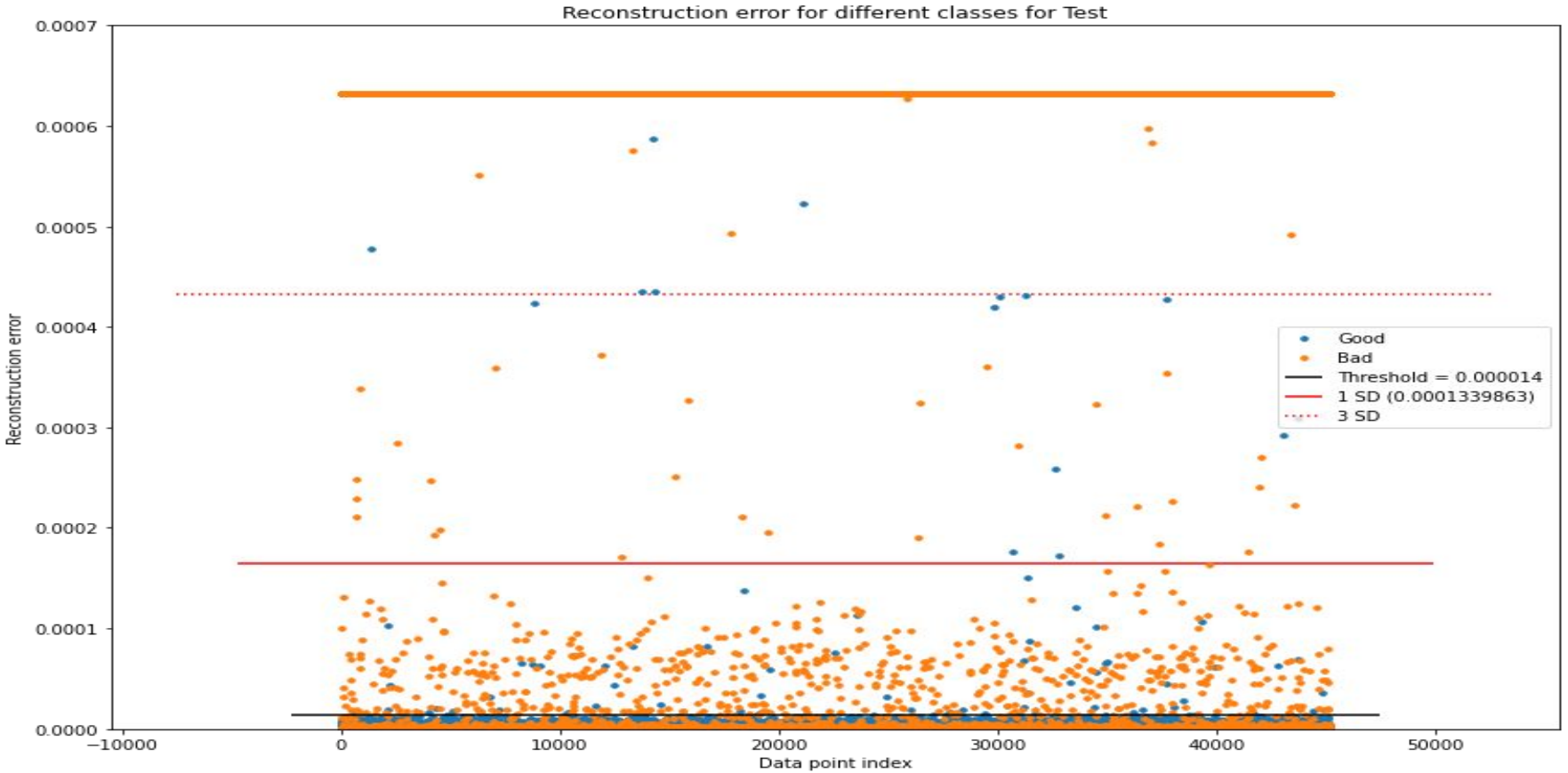
F.P = 0.21 % of N

F.N. = 44.2 % of P

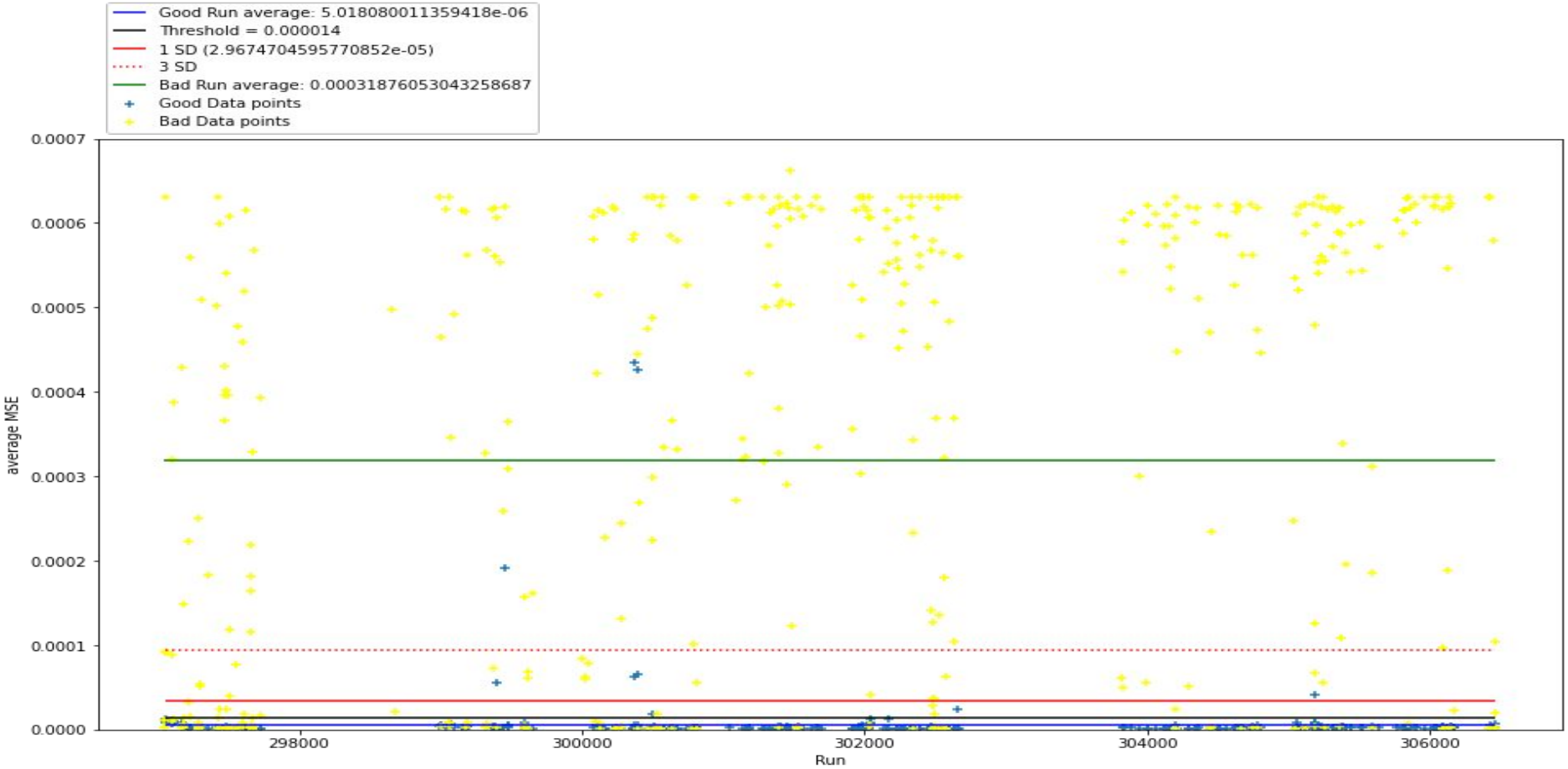
Suggestions?

Should we call it Positive or Bad lumi at all?

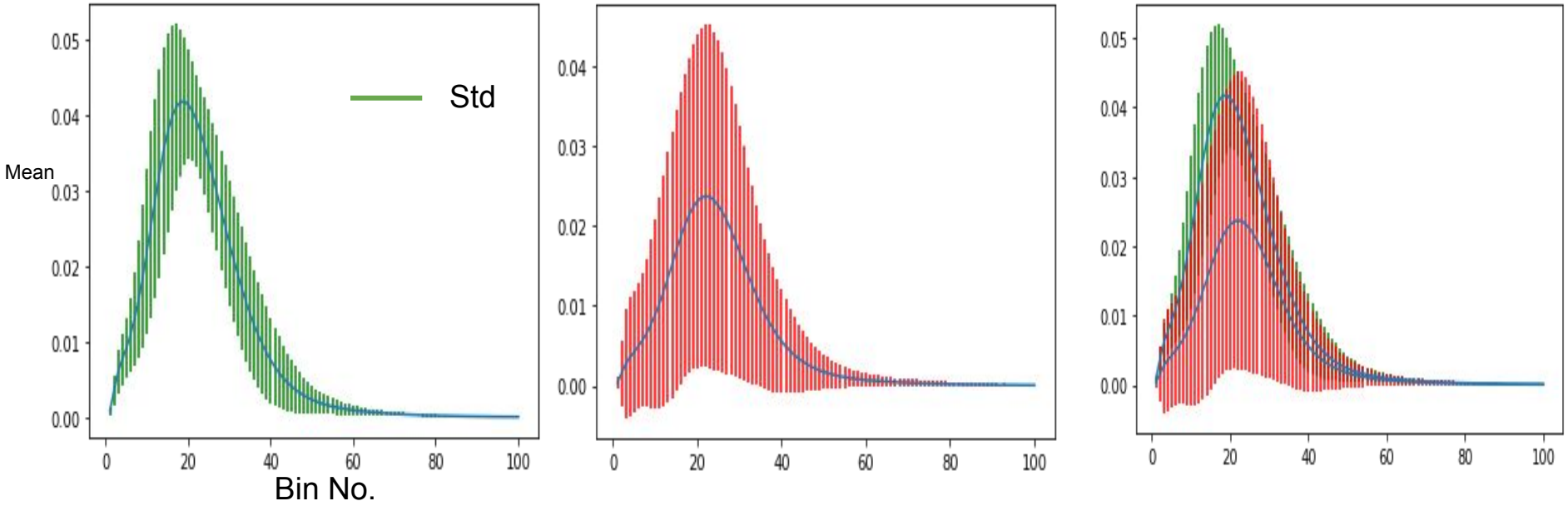
# Reconstruction error for test dataset (Good & Bad)



# Reconstruction error Global Trend over runs



# Mean and Std of Good and Bad data over all lumis




Good : Possibility of some correlation over the sequence of bins

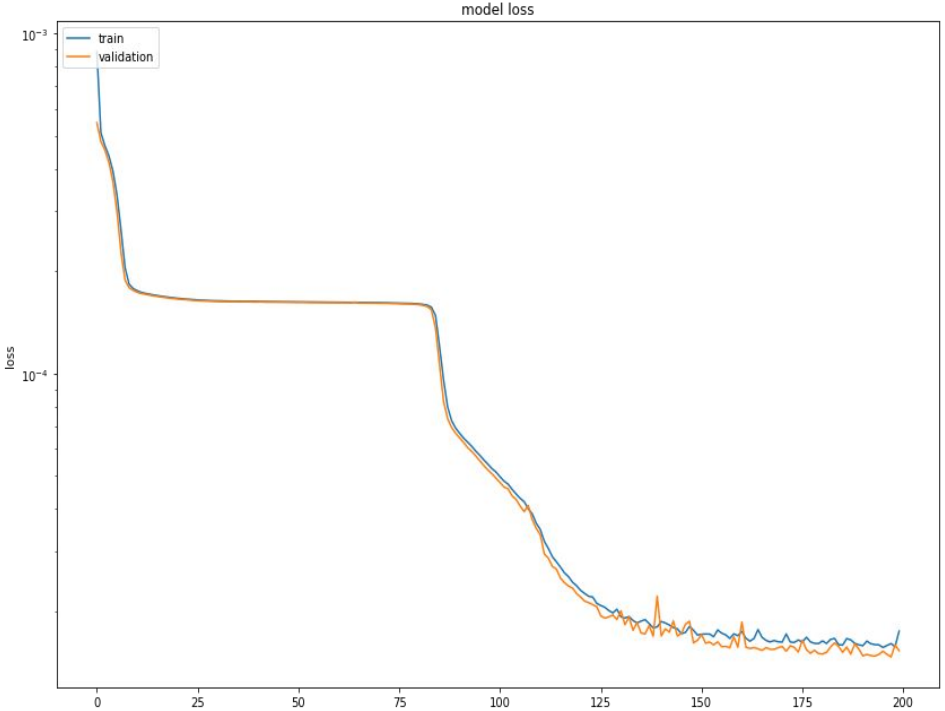
Bad : random fluctuation of std => Poor correlation

=> We try to investigate the sequential information of bins by LSTM AE

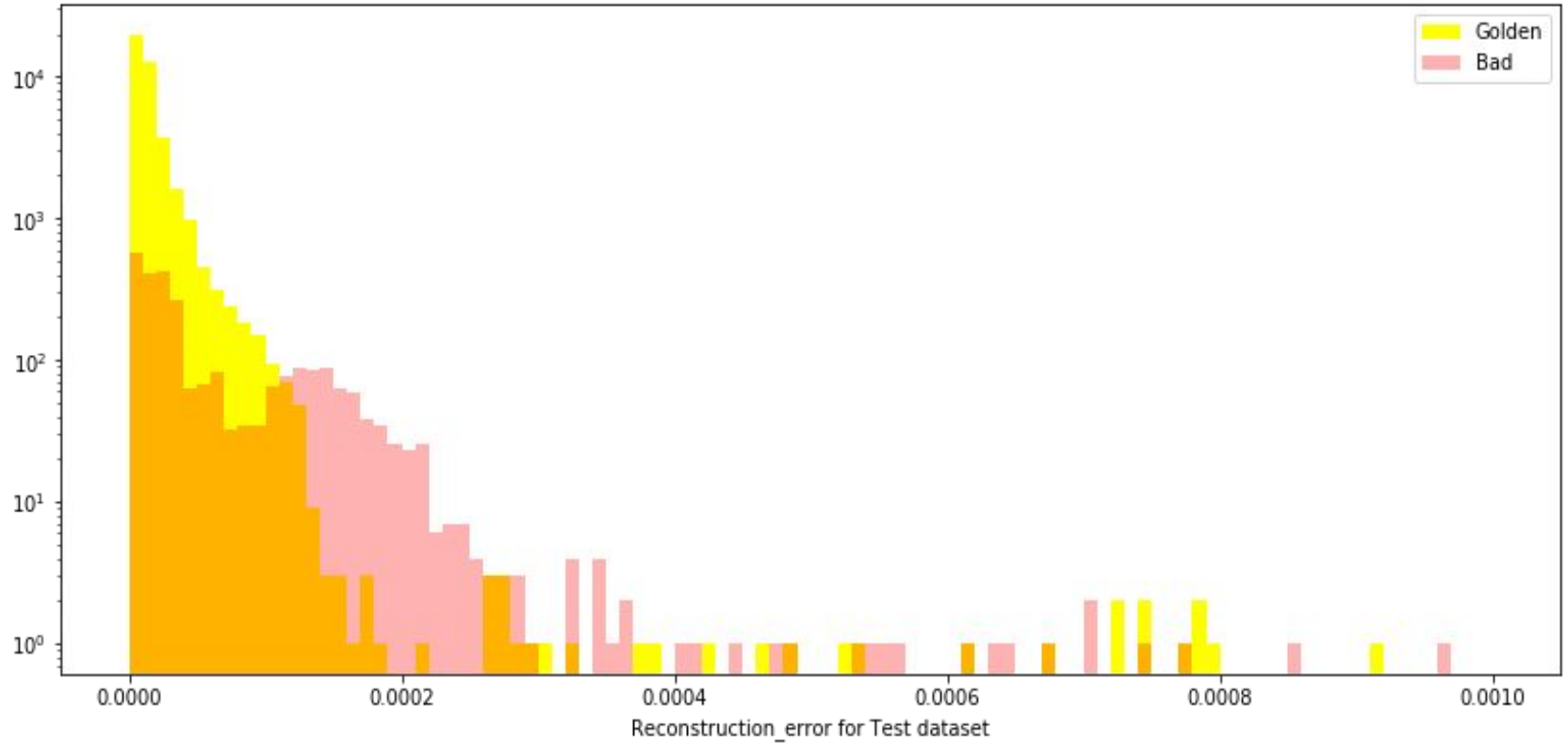
# LSTM Autoencoder

- Using individual bin as a step in a sequence, we tried to encode histogram as sequence in LSTM (timestep = 100, no, of features = 1)
- Tried to keep no. of parameters as close possible to the standard AE to compare

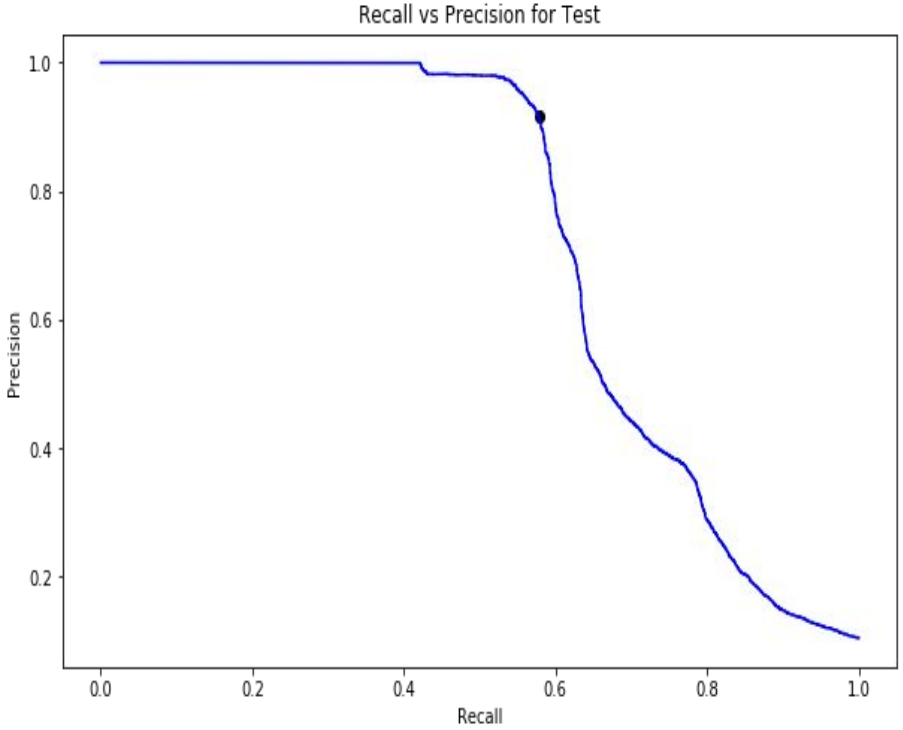
```
(145591, 100, 1)
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 100, 10)           480
_________________________________________________________________
lstm_2 (LSTM)                (None, 3)                 168
_________________________________________________________________
repeat_vector_1 (RepeatVecto (None, 100, 3)            0
_________________________________________________________________
lstm_3 (LSTM)                (None, 100, 3)            84
_________________________________________________________________
lstm_4 (LSTM)                (None, 100, 10)           560
_________________________________________________________________
time_distributed_1 (TimeDist (None, 100, 1)            11
=================================================================
Total params: 1,303
Trainable params: 1,303
Non-trainable params: 0
_________________________________________________________________
```

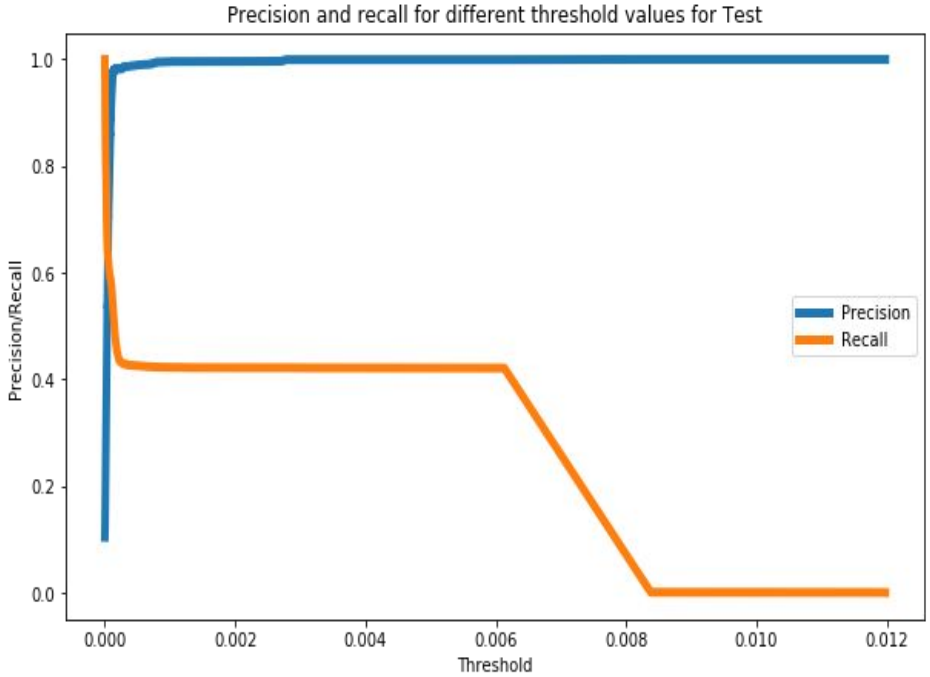


# Reconstruction error (mseTop10)



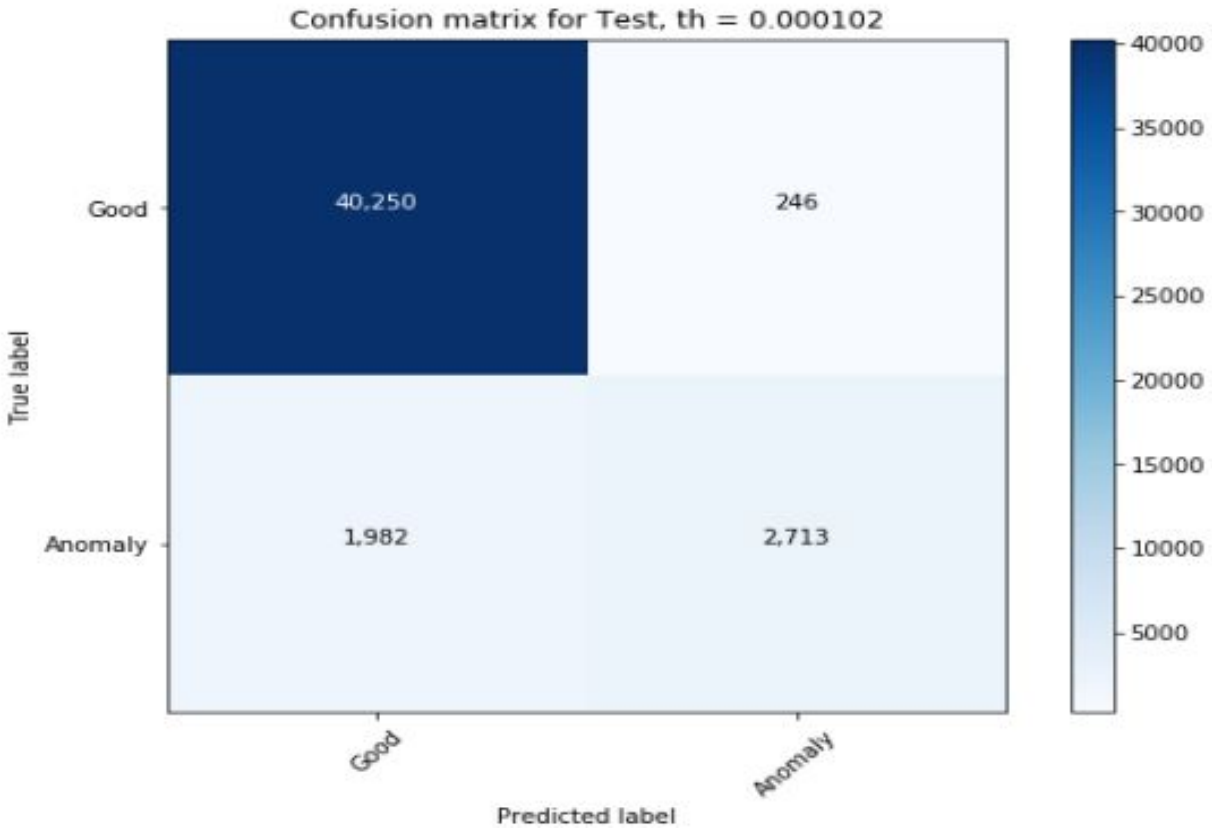
# PR Curve




Best Threshold = 0.000102, F1 Score = 0.709 (Same as Standard AE)

# Confusion Matrix



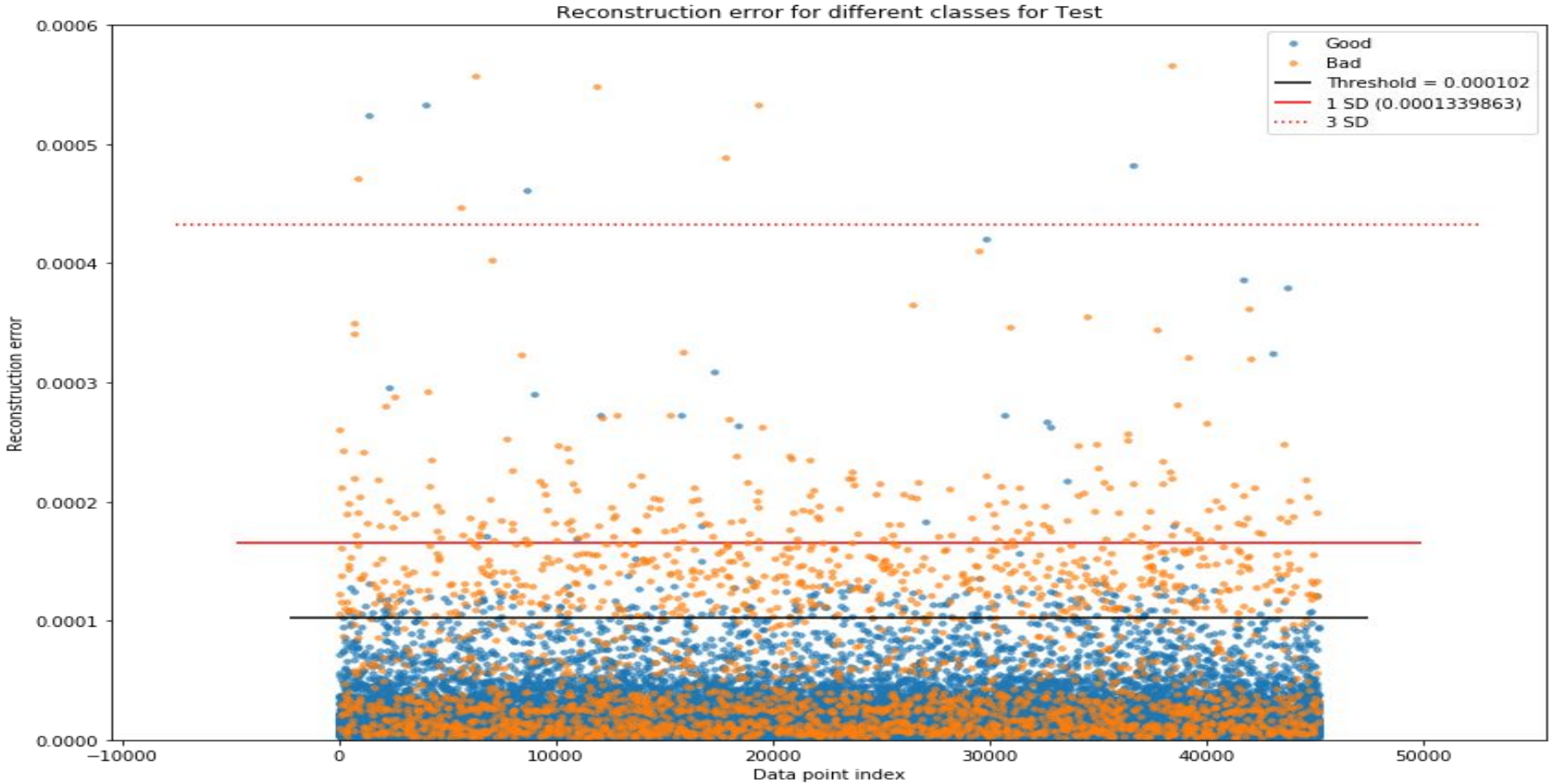

FP = 0.61% of N

FN = 42.2% of P

Comparable with standard AE

Till now, no further improvement wrt standard AE

# Reconstruction error for test dataset (Good & Bad)



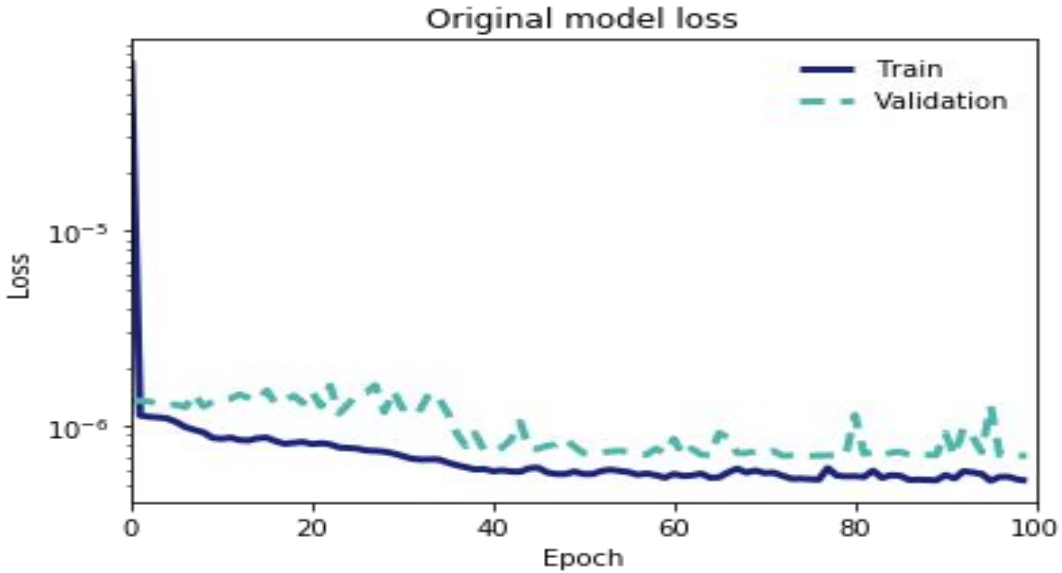
# Future plan & Acknowledgements

- We further want to investigate the models and the parameters more.
- On behalf of Tracker, we want to start looking into Hist1D Normalized HitResiduals_TIB_layer* for 2017 data.
- Reference for LSTM & PR Curve:
  - https://towardsdatascience.com/step-by-step-understanding-lstm-autoencoder-layers-ffab055b6352
  - https://machinelearningmastery.com/roc-curves-and-precision-recall-curves-for-classification-in-python/
- Thanks Francesco for sharing the code snippets! It helped me a lot as a beginner.

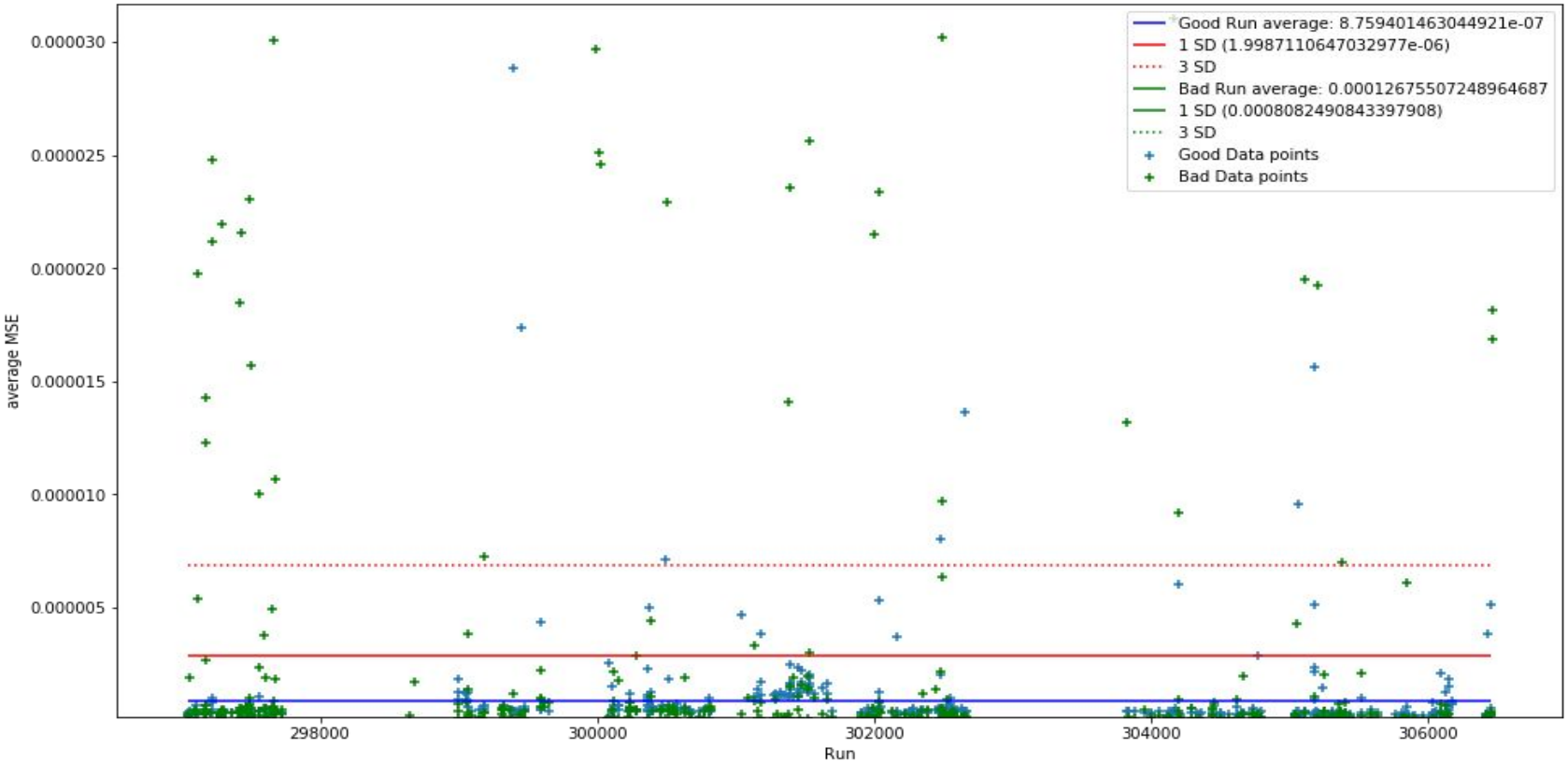
# Thank you!

# Standard Autoencoder

- Standard DNN based autoencoder with tanh activation in intermediate layers final layer. input - 10 - 3 - 10 - output
- Optimizer = adam , loss = mseTop10
  Good runs and lumis selected from Golden json file



# GlobalMSETrend

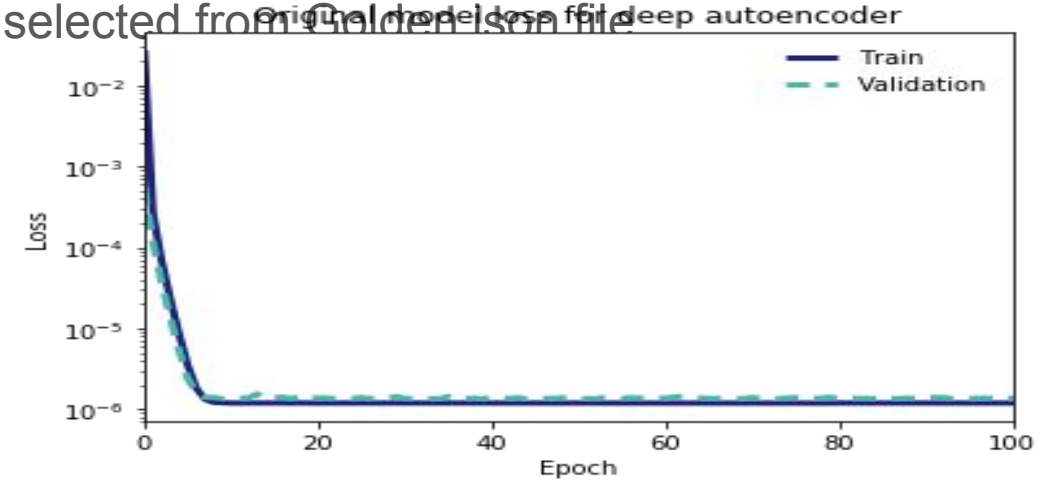


# Deep autoencoder

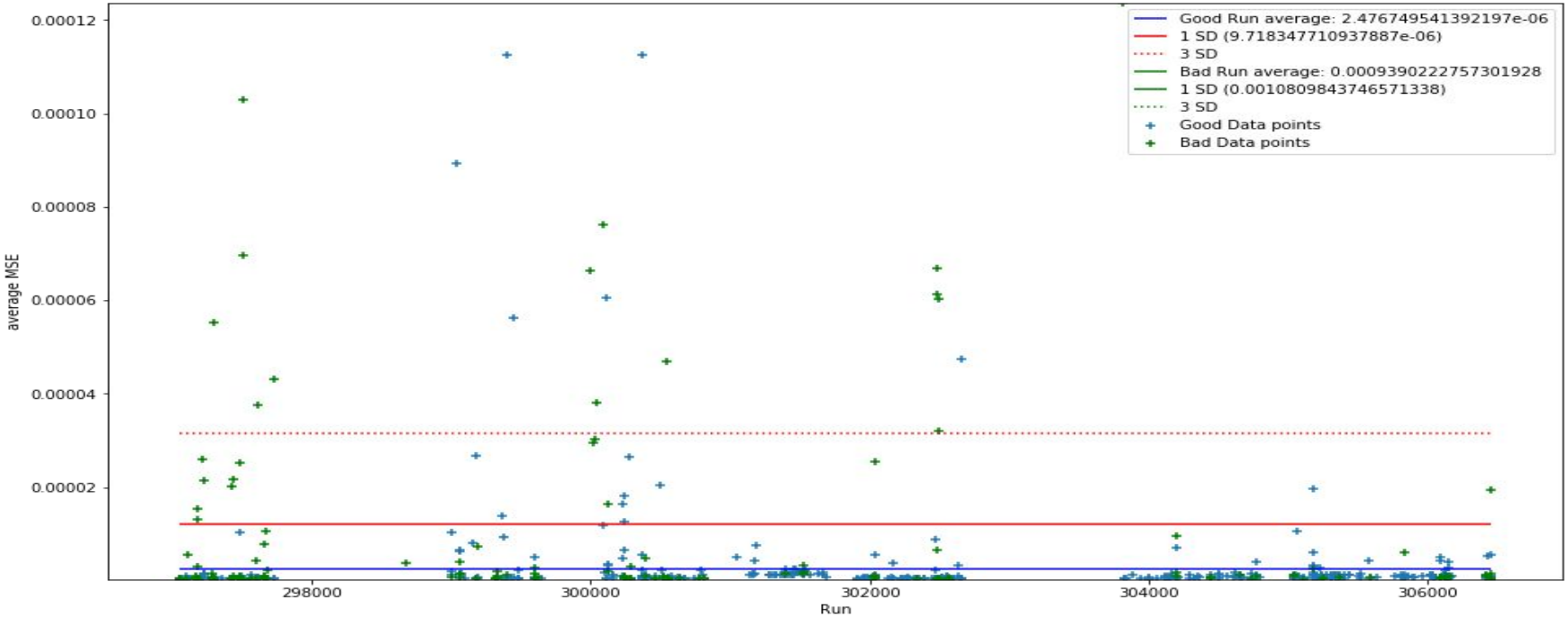
- Deeper DNN based autoencoder with relu activation in intermediate layers and tanh in the final layer. (input - 20 - 10 - 20 - output)

Optimizer = adam , loss = mseTop10

Good runs and lumis selected from Golden Json file

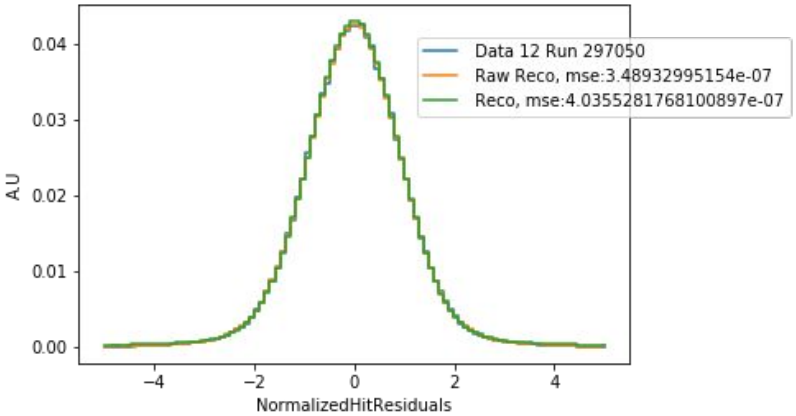


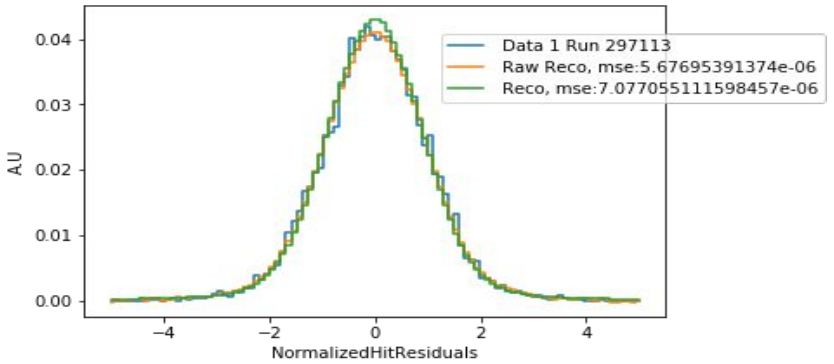
# GlobalMSETrend

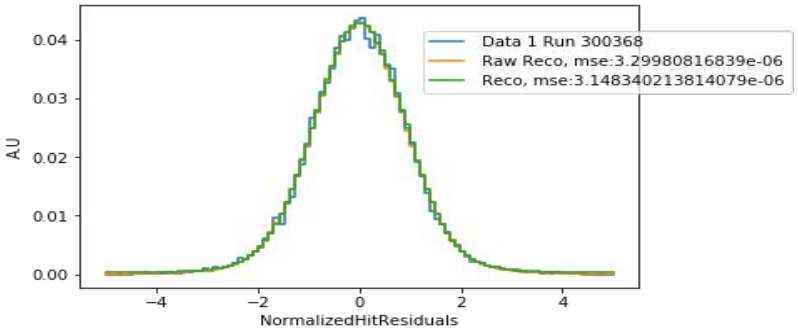


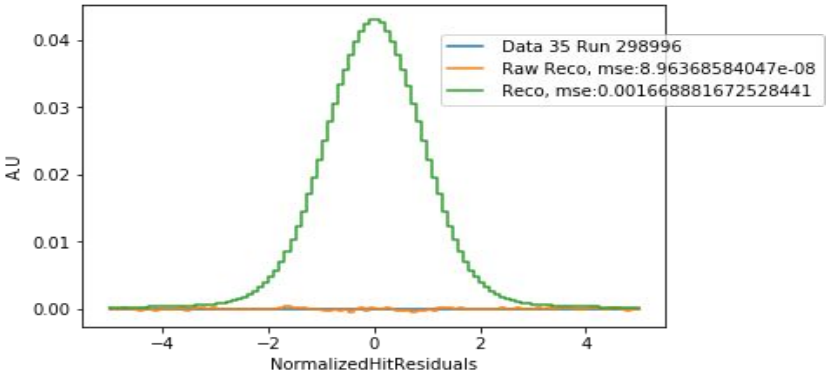

Deep AE works better in global mse trend
Good and bad lumis are not well separated.

# Predictions for good lumis [Raw reco = first AE, Reco = Deep AE]

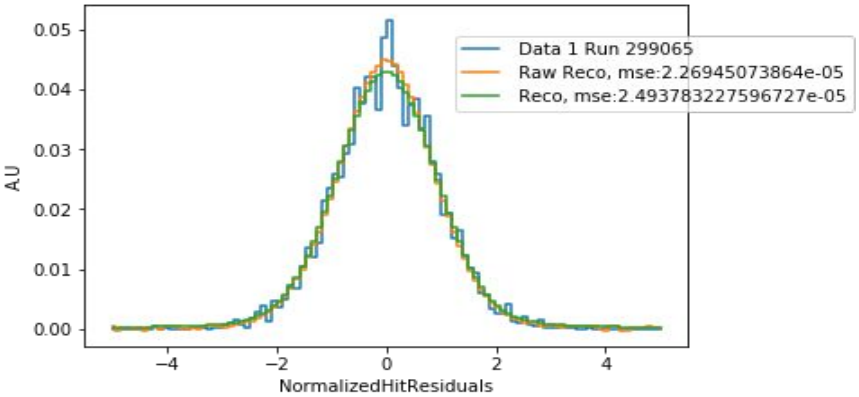


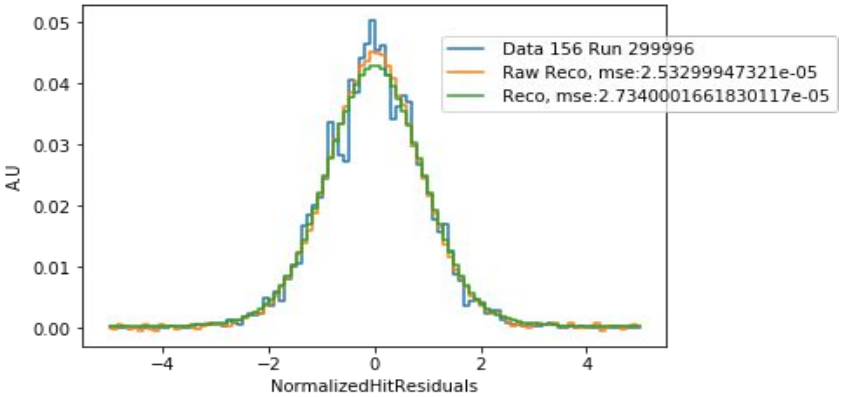



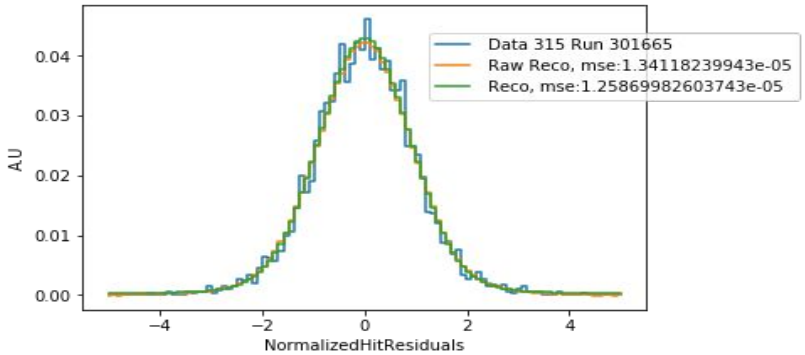



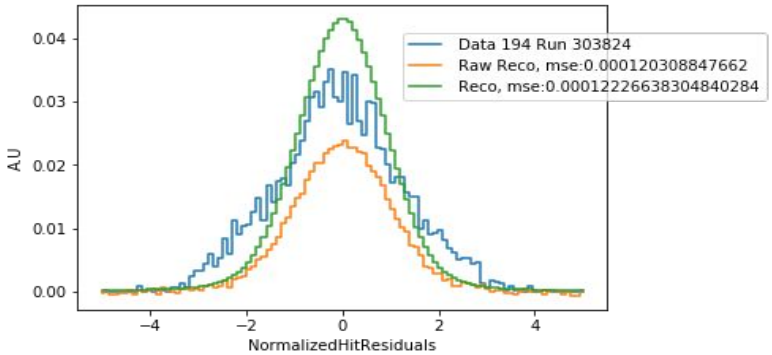



# Predictions for bad lumis [Raw reco = first AE, Reco = Deep AE]



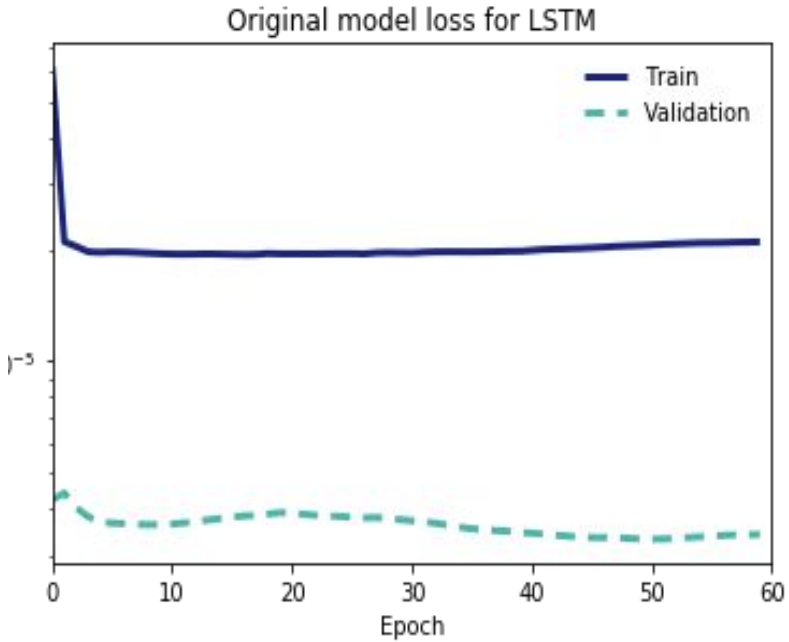



# Using LSTM autoencoder for chargeInner_PXLayer_1

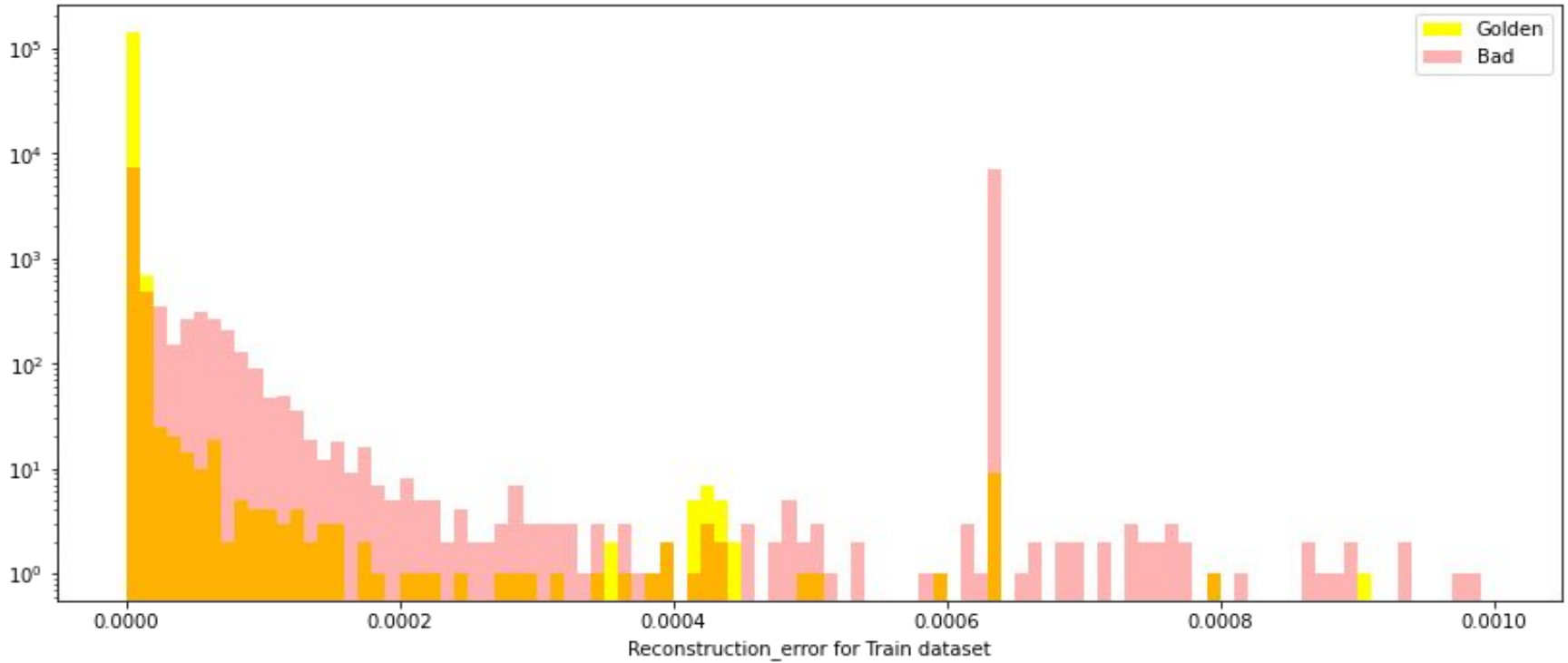
Using individual bin as a step in a sequence, we tried to encode histogram as sequence in LSTM.

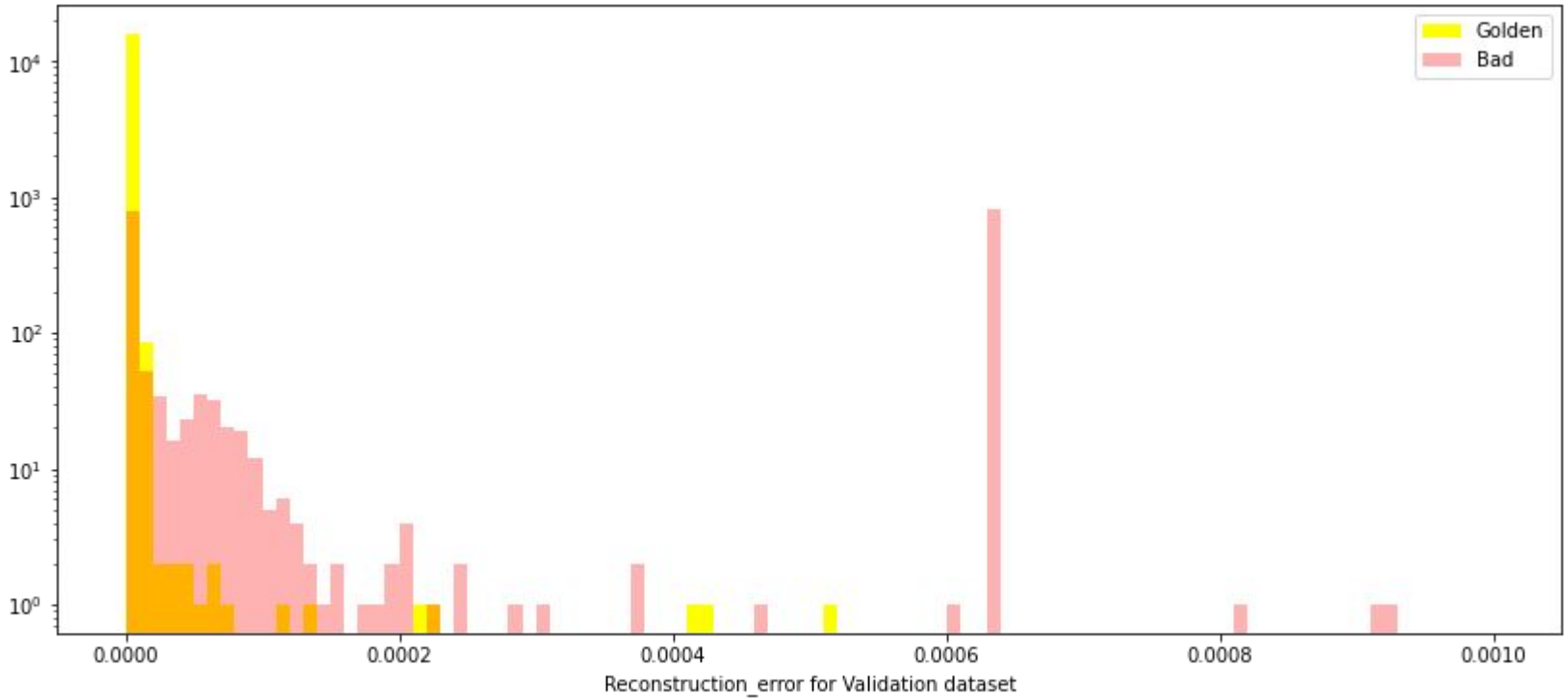
```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
layer_0 (LSTM)               (None, 10)                480
_________________________________________________________________
layer_1 (RepeatVector)       (None, 100, 10)           0
_________________________________________________________________
layer_2 (LSTM)               (None, 100, 10)           840
_________________________________________________________________
layer_3 (TimeDistributed)    (None, 100, 1)            11
=================================================================
Total params: 1,331
Trainable params: 1,331
Non-trainable params: 0
```

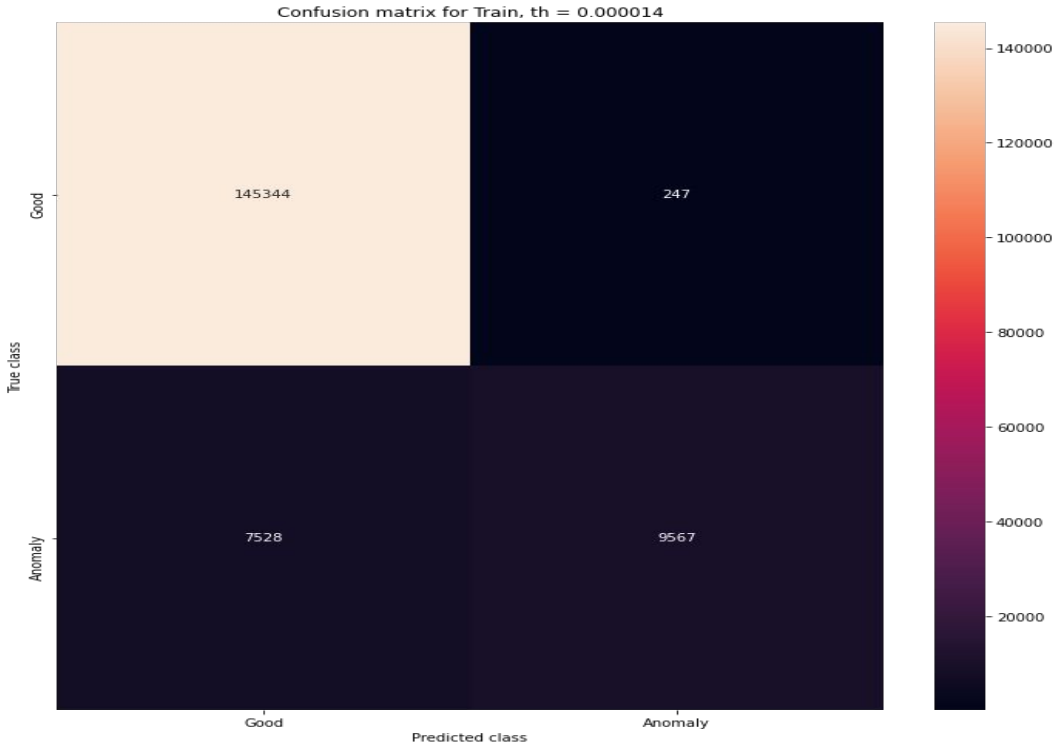


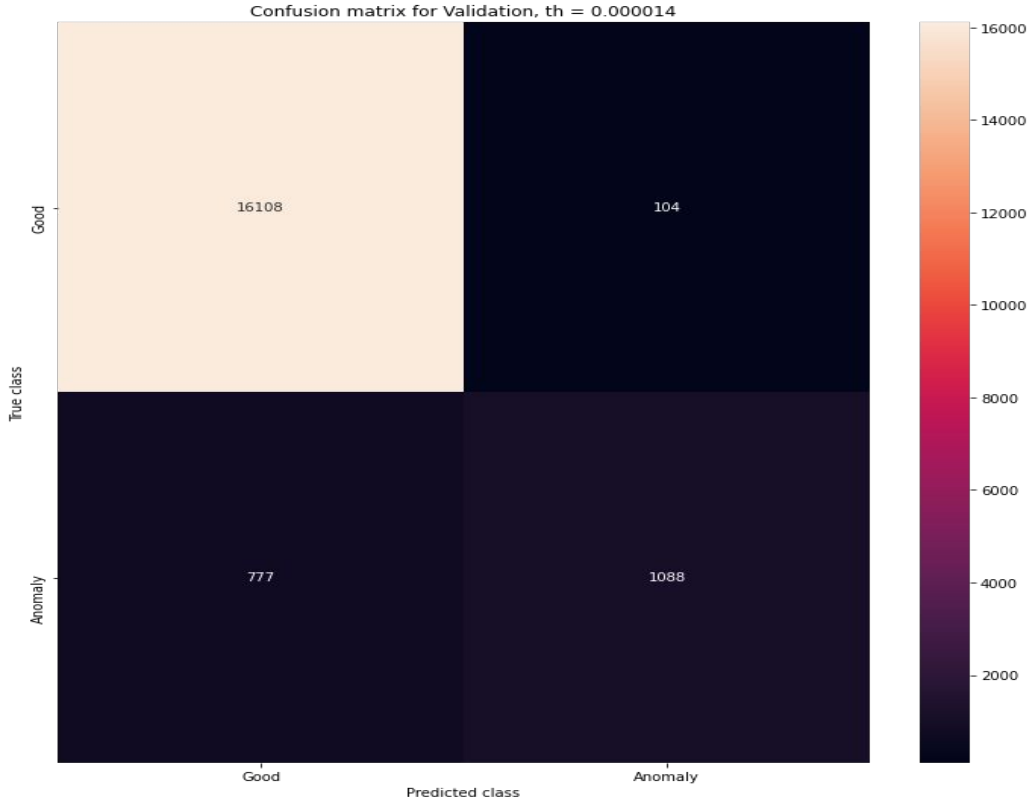

Yet to arrive at optimised network architecture.

Work in progress.


Golden
Bad
$10^5$
$10^4$
$10^3$
$10^2$
$10^1$
$10^0$
0.0000
0.0002
0.0004
0.0006
0.0008
0.0010
Reconstruction_error for Train dataset


Golden
Bad
$10^4$
$10^3$
$10^2$
$10^1$
$10^0$
0.0000
0.0002
0.0004
0.0006
0.0008
0.0010
Reconstruction_error for Validation dataset


Confusion matrix for Train, th = 0.000014
145344
247
7528
9567
Good
Anomaly
True class
Good
Anomaly
Predicted class
140000
120000
100000
80000
60000
40000
20000


Confusion matrix for Validation, th = 0.000014
16108
104
777
1088
Good
Anomaly
True class
Good
Anomaly
Predicted class
16000
14000
12000
10000
8000
6000
4000
2000

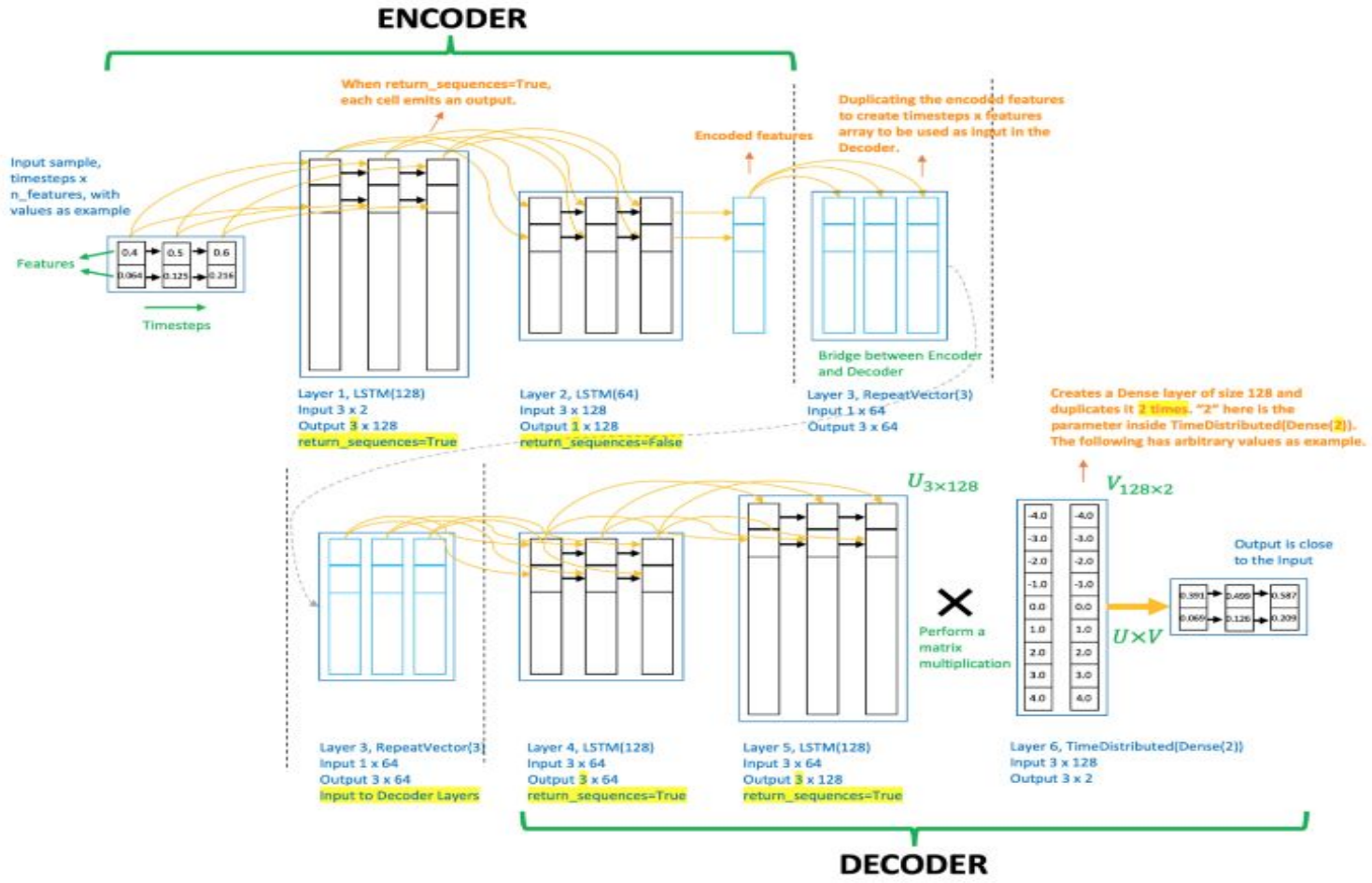
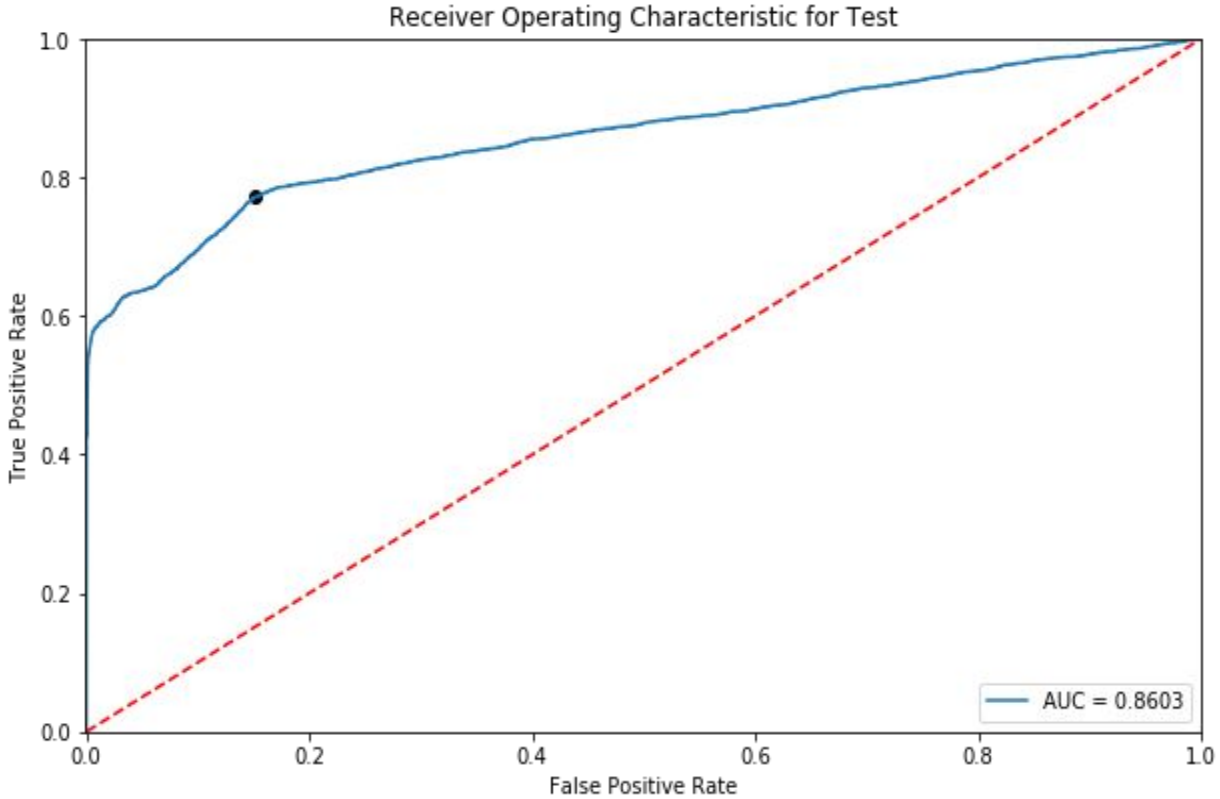
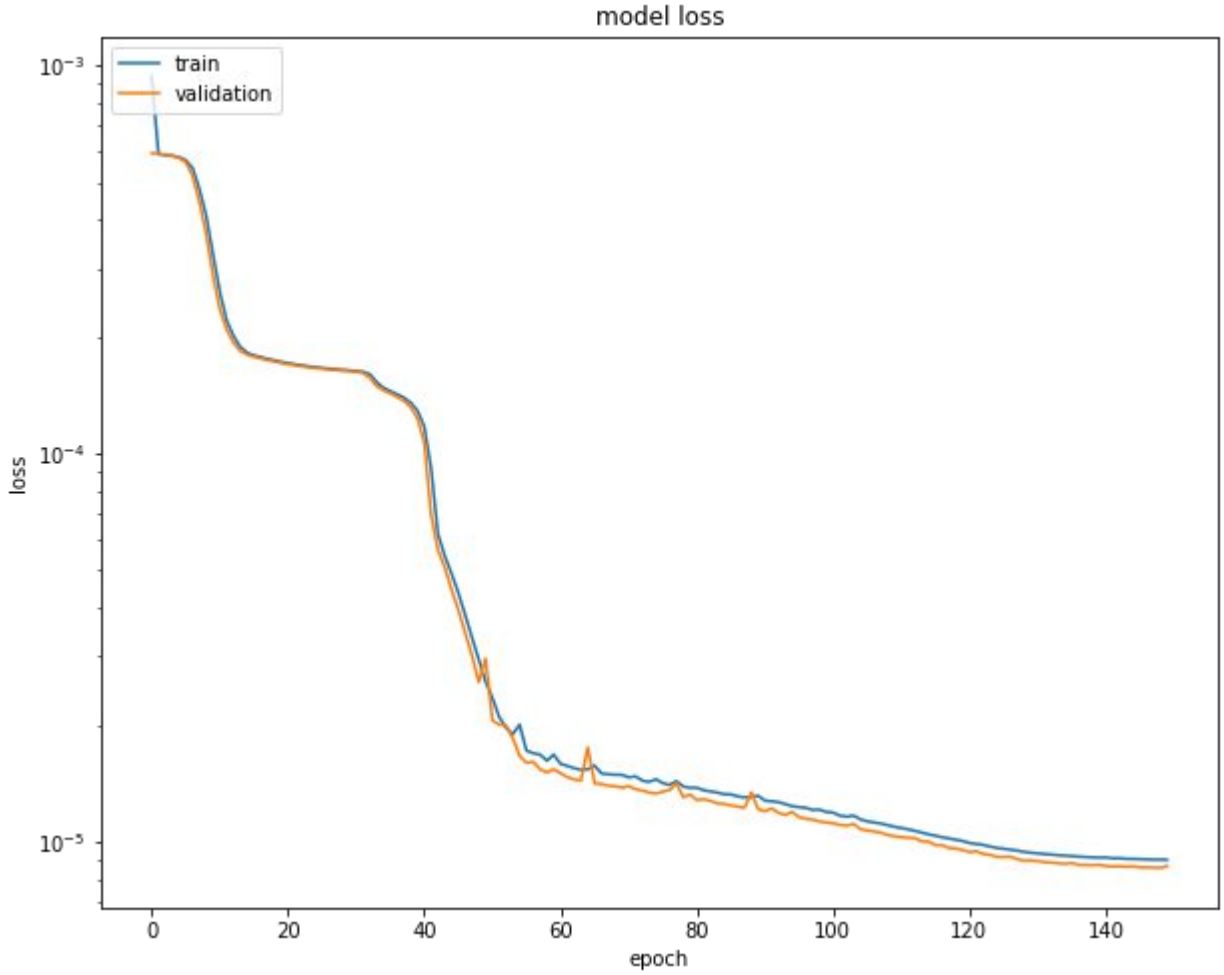
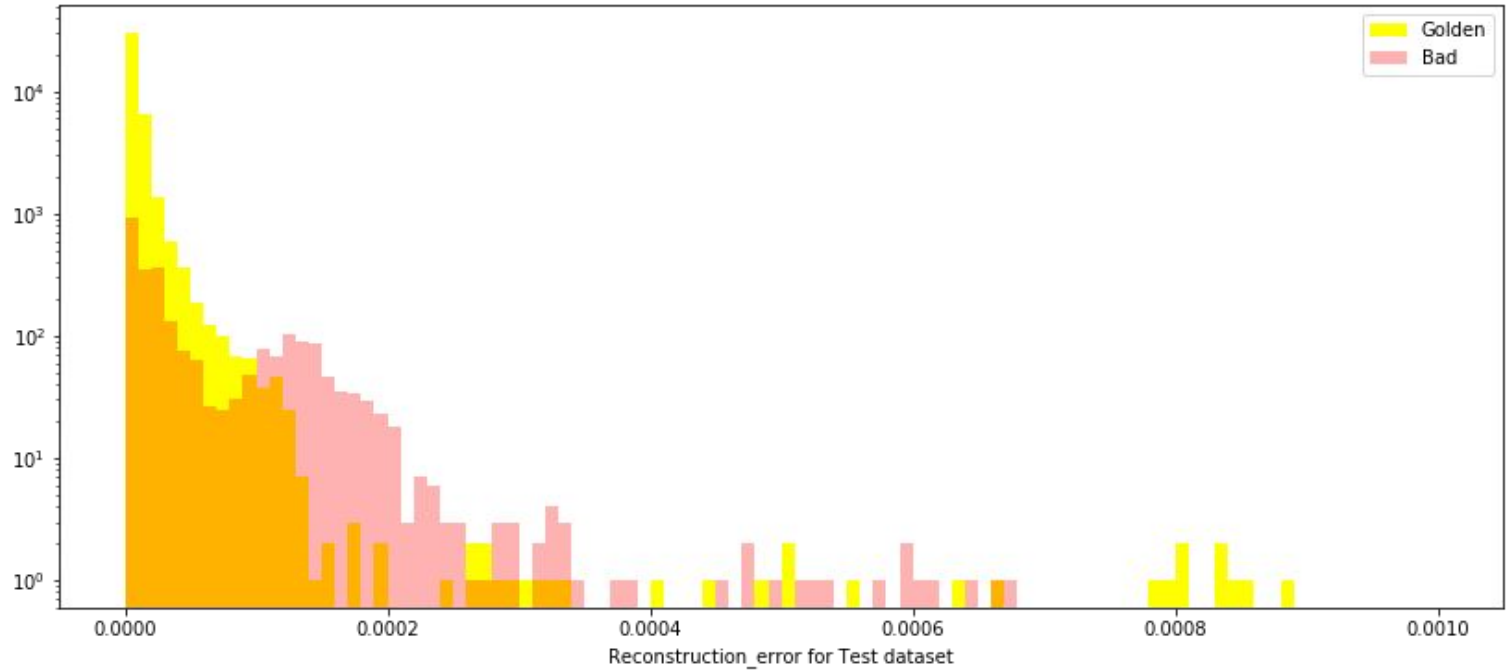


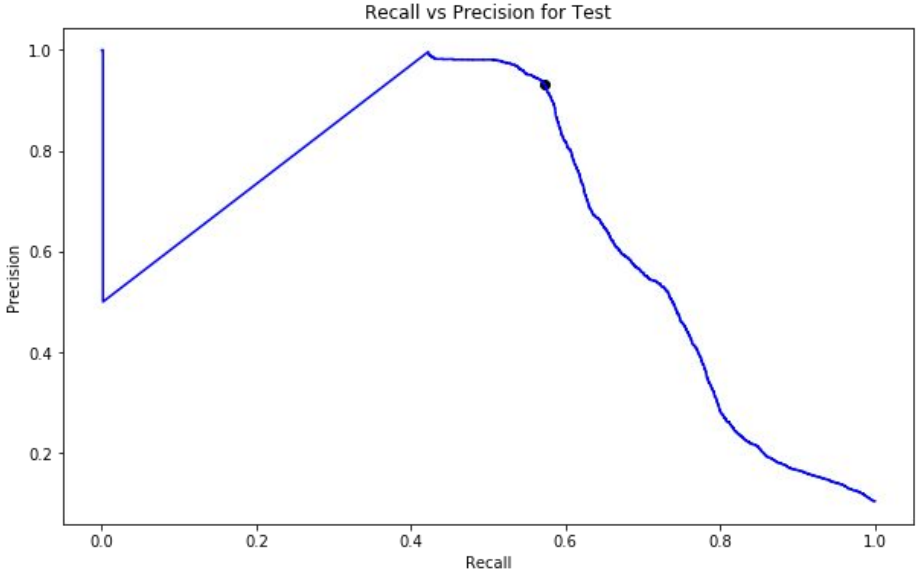

Figure 2.3. LSTM Autoencoder Flow Diagram.

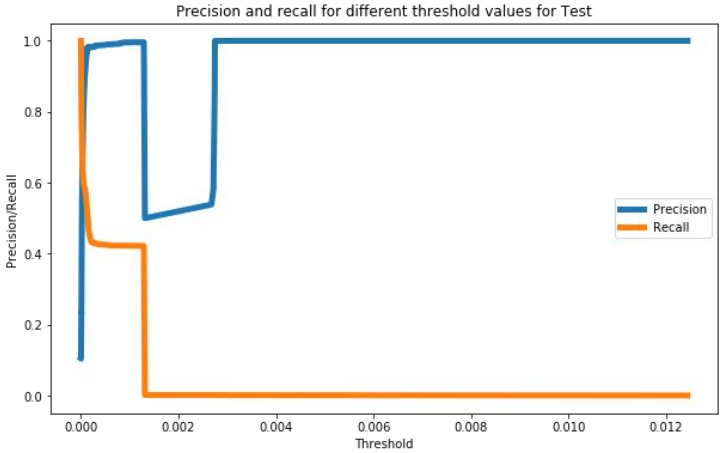
# LSTM ROC

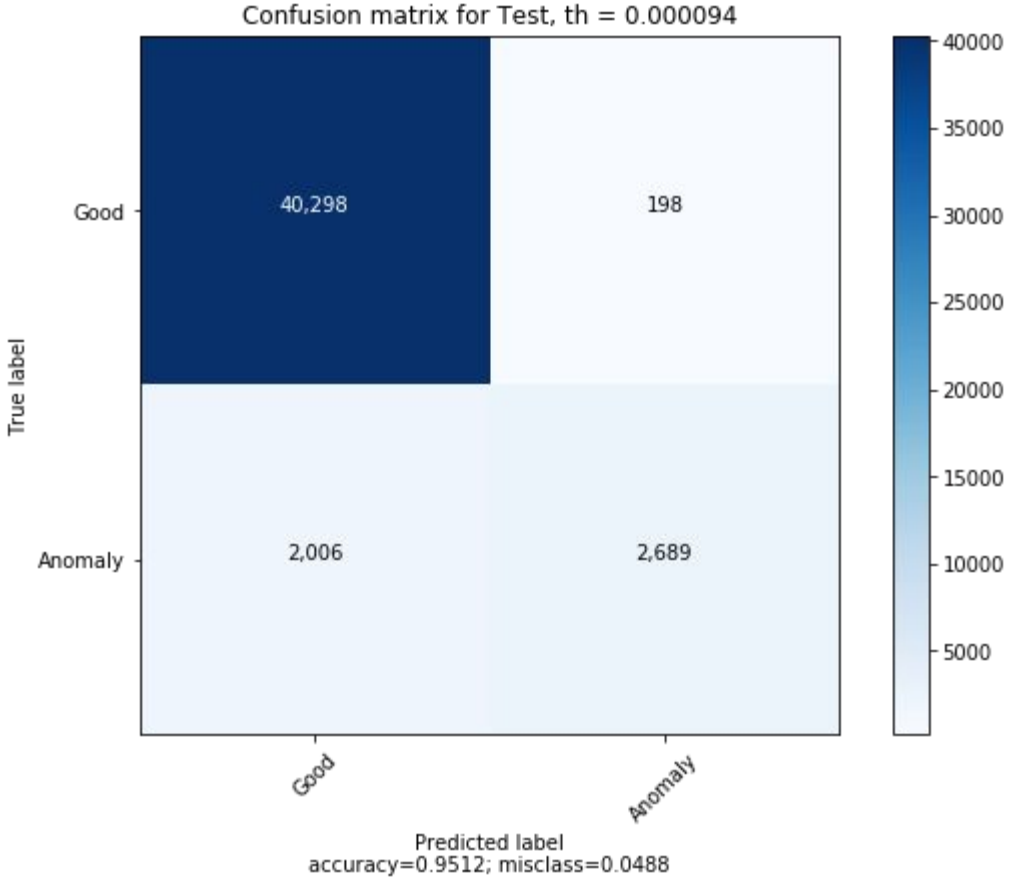

Receiver Operating Characteristic for Test
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
AUC = 0.8603

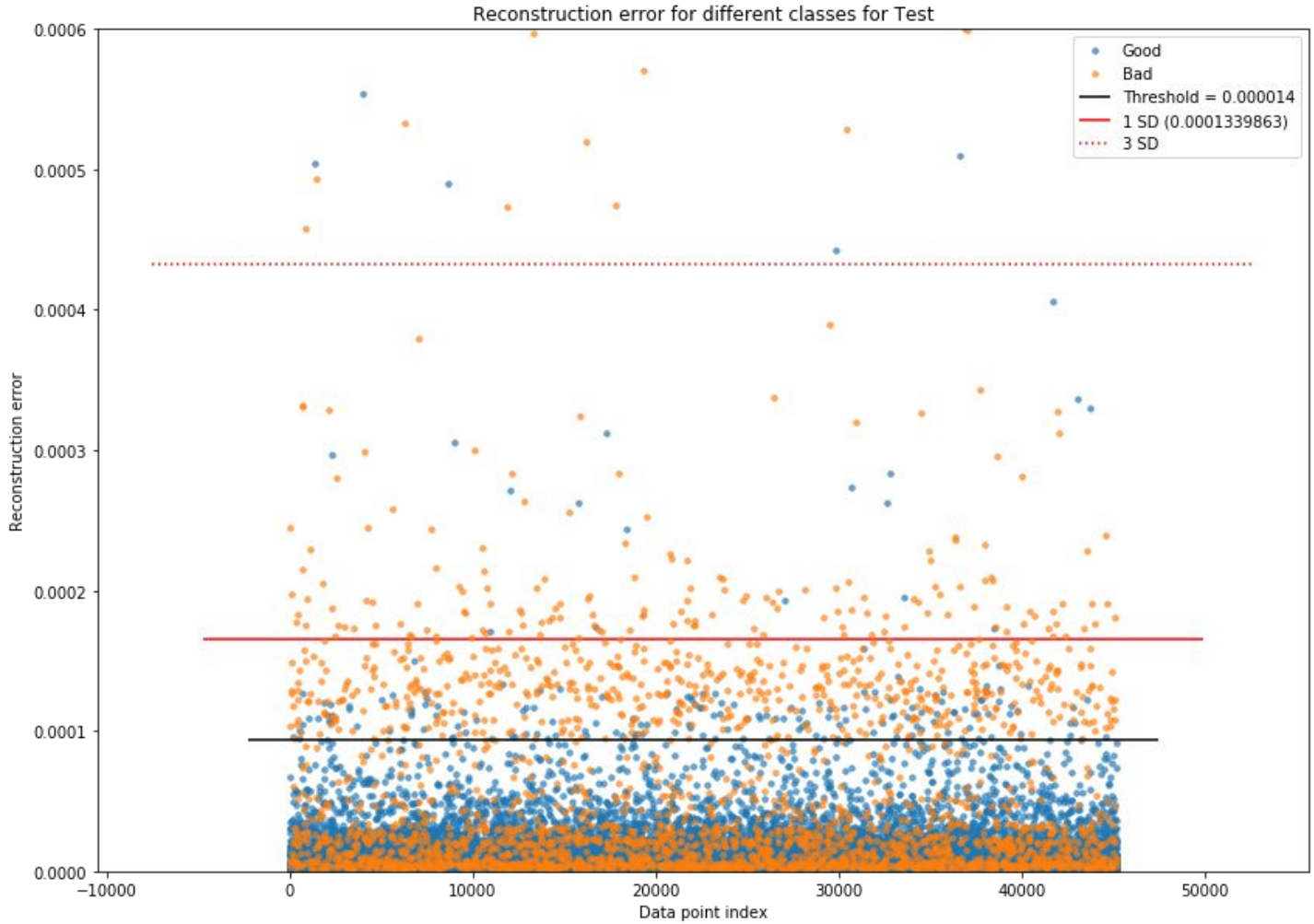

model loss
train
validation
loss
epoch
$10^{-3}$
$10^{-4}$
$10^{-5}$
0
20
40
60
80
100
120
140


Golden
Bad
$10^4$
$10^3$
$10^2$
$10^1$
$10^0$
0.0000
0.0002
0.0004
0.0006
0.0008
0.0010
Reconstruction_error for Test dataset


Recall vs Precision for Test
Precision
Recall
1.0
0.8
0.6
0.4
0.2
0.0
0.2
0.4
0.6
0.8
1.0


Precision and recall for different threshold values for Test
Precision/Recall
Threshold
Precision
Recall
1.0
0.8
0.6
0.4
0.2
0.0
0.000
0.002
0.004
0.006
0.008
0.010
0.012


Confusion matrix for Test, th = 0.000094
Good
Anomaly
True label
40,298
198
2,006
2,689
Good
Anomaly
Predicted label
accuracy=0.9512; misclass=0.0488
40000
35000
30000
25000
20000
15000
10000
5000


Reconstruction error for different classes for Test
Good
Bad
Threshold = 0.000014
1 SD (0.0001339863)
3 SD
Reconstruction error
Data point index
0.0006
0.0005
0.0004
0.0003
0.0002
0.0001
0.0000
−10000
0
10000
20000
30000
40000
50000