ANOMALY DETECTION in MOBILE ROBOTICS

REPORT AND RESULTS

LUIGI PALLADINO - 13/04/2022

Several Datasets

1. Boat
2. Lutra
3. Pepper
4. Kdd
5. Paper
6. Swat
7. Wadi

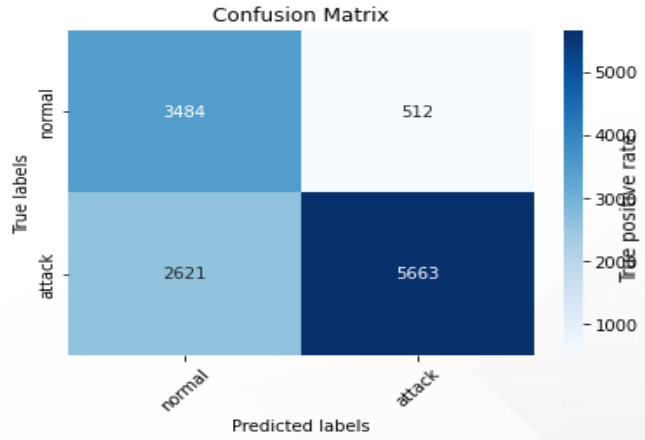
- Pepper is the most used in the notebook shared
- Boat and Swat are also nominated
- Boat and Pepper used in the paper

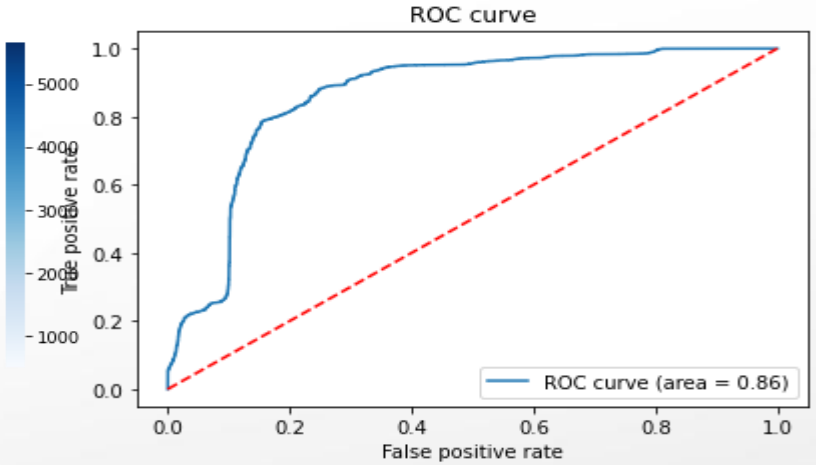
➢ 4 new datasets available from kairos?

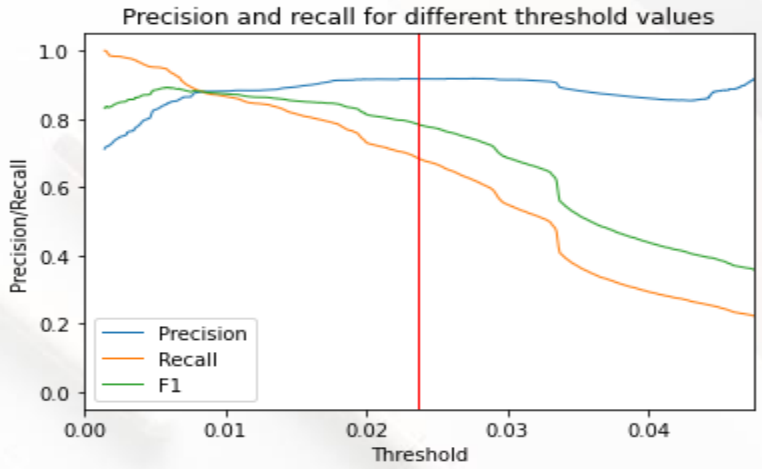
Regression models used

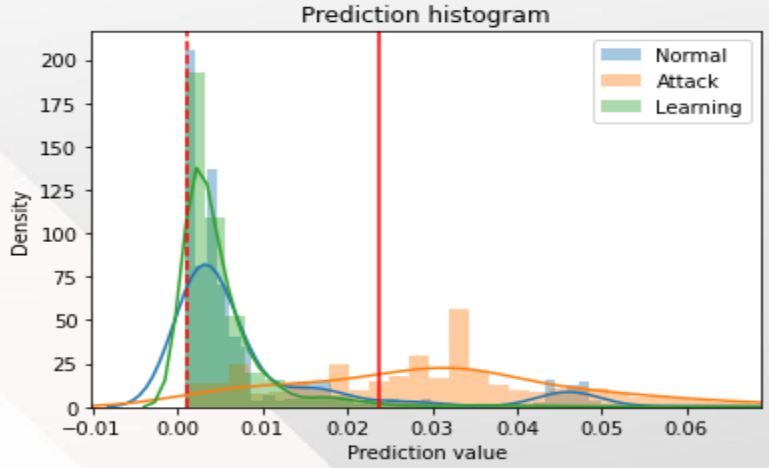
1. RNN bptt: Residual Neural Network with back-propagation through time
2. My Linear: probably “Dense”, as said in the paper (?)
3. TCN: Temporal Convolutional Network (1D)
4. Various ScikitLearn solutions explored

1. Results of RNN with bptt on *pepper*









Results with MSE on all sensors:

Total accuracy: 0.744870

Total precision: 0.917085

Total recall: 0.683607

Total f1: 0.783318

Total f2: 0.783318

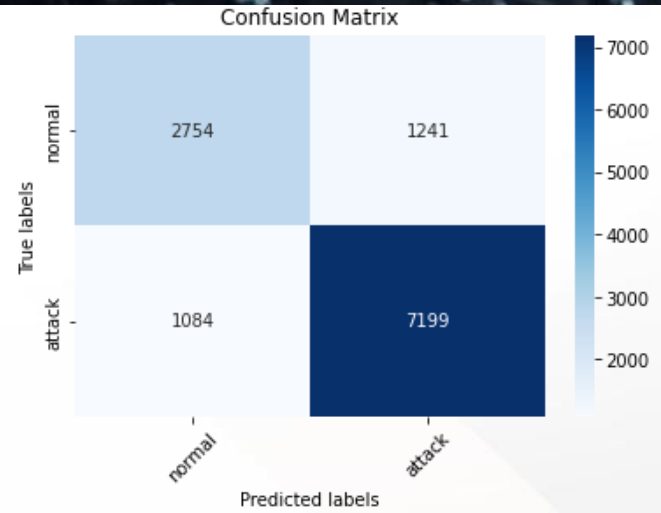
Total true negative: 3484

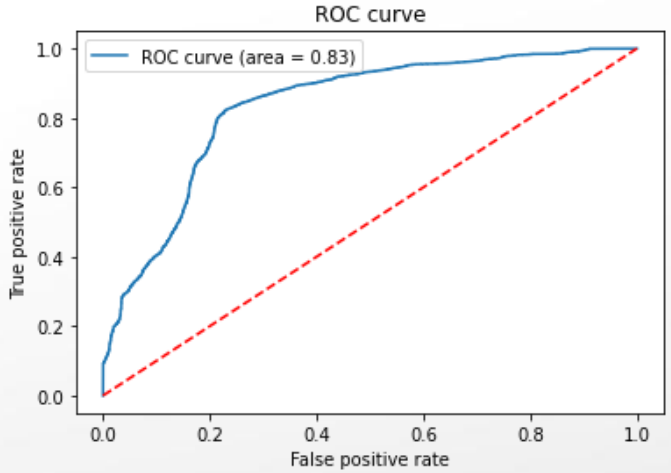
Total false positive: 512

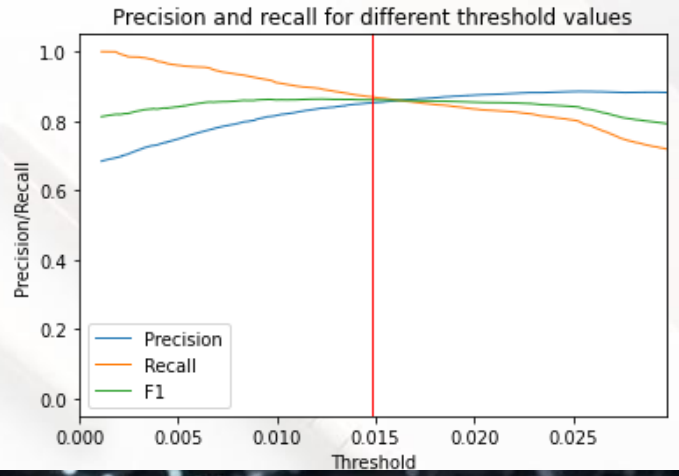
Total false negative: 2621

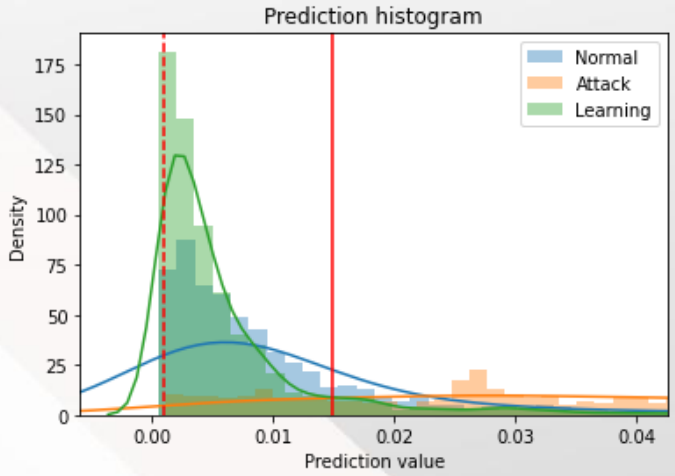
Total true positive: 5663

2. Results of Dense on *pepper*









Results with MSE on all sensors:

Total accuracy: 0.810637

Total precision: 0.852962

Total recall: 0.869130

Total f1: 0.860970

Total f2: 0.860970

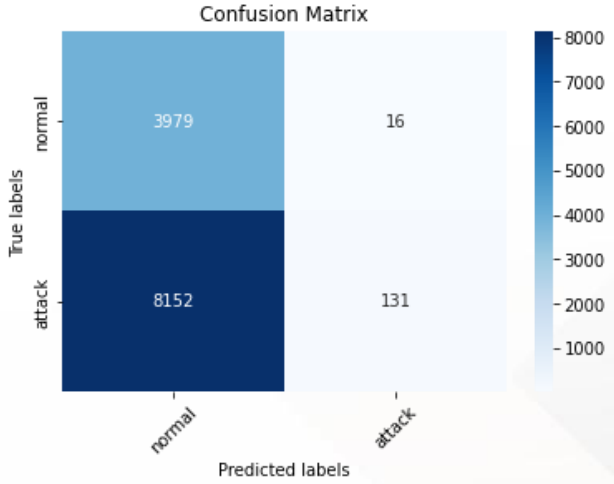
Total true negative: 2754

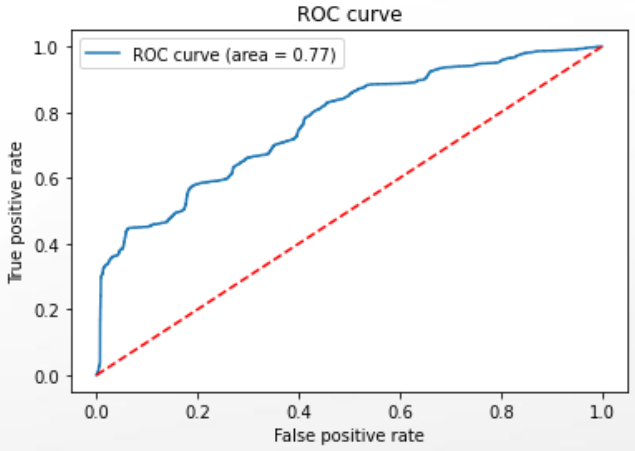
Total false positive: 1241

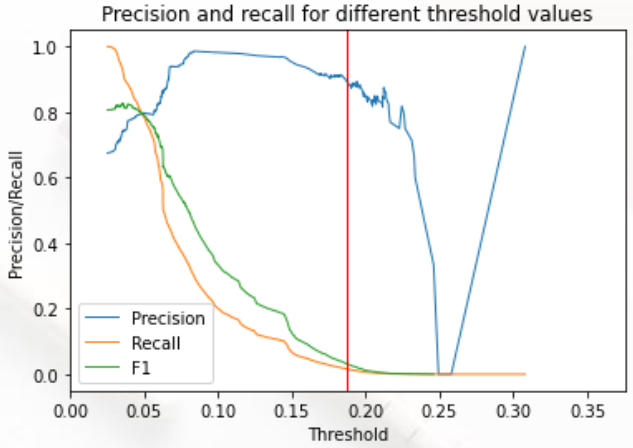
Total false negative: 1084

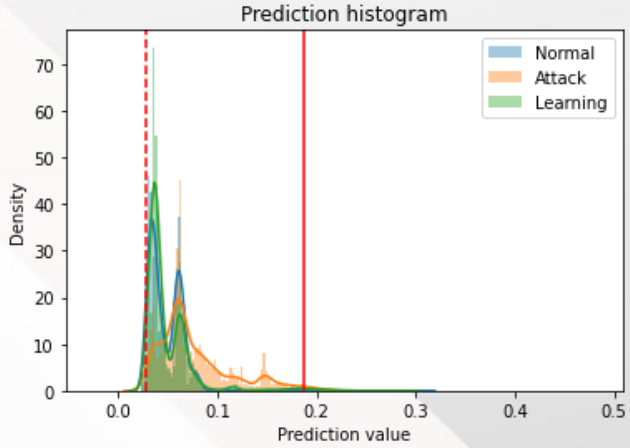
Total true positive: 7199

3. Results of TCN on *pepper*









Results with MSE on all sensors:

Total accuracy: 0.334745

Total precision: 0.891156

Total recall: 0.015816

Total f1: 0.031079

Total f2: 0.031079

Total true negative: 3979

Total false positive: 16

Total false negative: 8152

Total true positive: 131

Conclusion & Future Works

- Dense seems to be the best model in binary classification on *pepper* dataset (81% accuracy)
- RNN seems to be the best model for open-class classification, it have a better performance in regression (AUC of 0.86)

Possible future works:

1. Try these models on the new datasets from kairos
    - Try to use transfer learning from previous trainings (?)
2. Try new state of the art architectures
3. Include True Skill Score in evaluation of the models

ANOMALY DETECTION in MOBILE ROBOTICS

1ST KAIROS UPDATE

LUIGI PALLADINO - 20/05/2022

Workflow of Development

1.Data Acquisition → 2.Data Cleaning → 3. Data Labeling → 4.Model Training

1. Francesco Trotti: recording of ROS packages during simulation

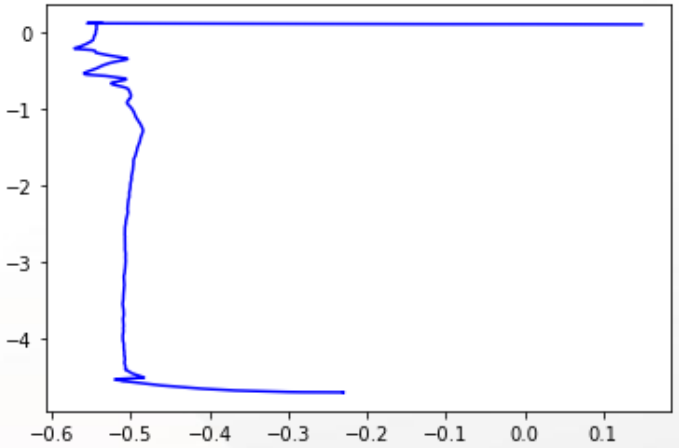
2. Selection of time-series using Pandas:
   - Syncronization of series at different frequencies

3. Division of "normal" behaviour time-series for training and data with anomalies for testing.

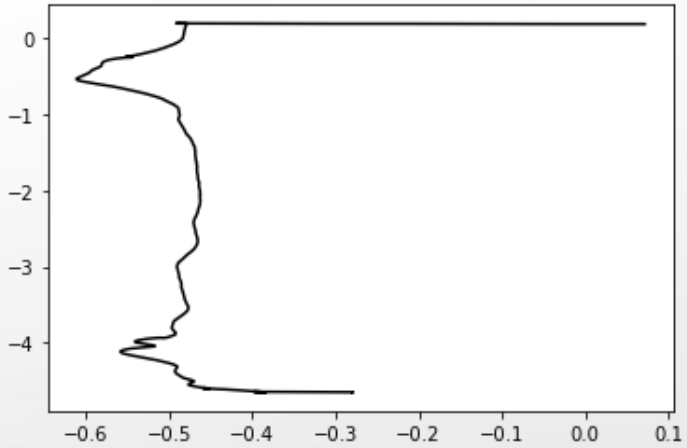
4. Training of Autoencoder architecture: Selection of threshold for anomalies

1.Data Aquisition

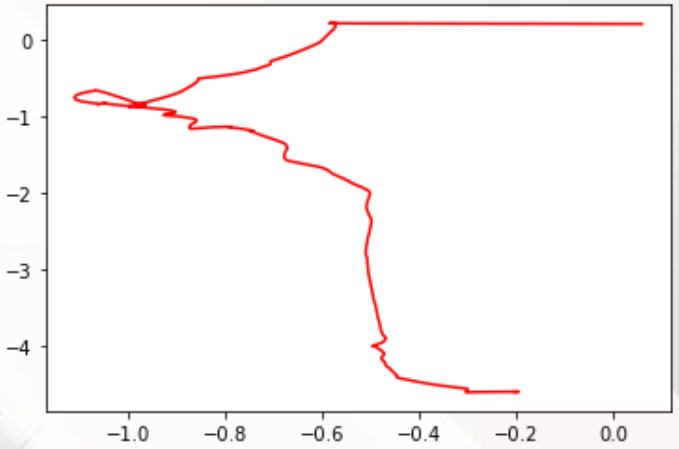
Normal plan 0



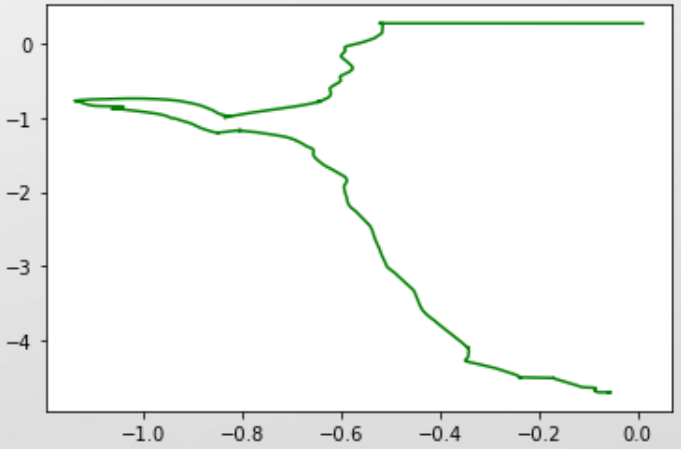
Normal plan 1



Anomaly plan 0



Anomaly plan 1

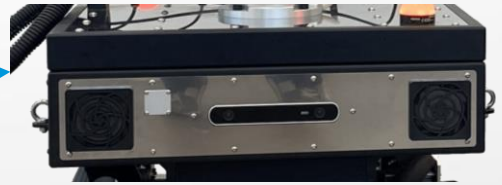


2.Data Cleaning

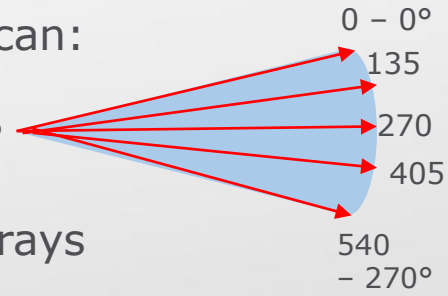
From "interview" to Francesco Trotti:

csv file	description
fufi-front_laser-scan.csv	front laser data on robot chassis
fufi-imu-data.csv	linear/angular velocities/accelerations from imu sensor inside robot chassis
fufi-joint_states.csv	motors encoder (of wheels) → efforts
fufi-rear_laser-scan.csv	rear laser data on robot chassis → not used
fufi-robotnik_base_control-cmd_vel.csv	velocity commands in robot's controller
fufi-robotnik_base_control-odom.csv	vector for estimating the position of the robot relative to a starting location
fufi-robot_pose.csv	position on the map (x,y,z γ→ quaternion)
fufi-vectornav-imu-temperature.csv	internal temperature

Personal solution to laser data:



Laser scan:



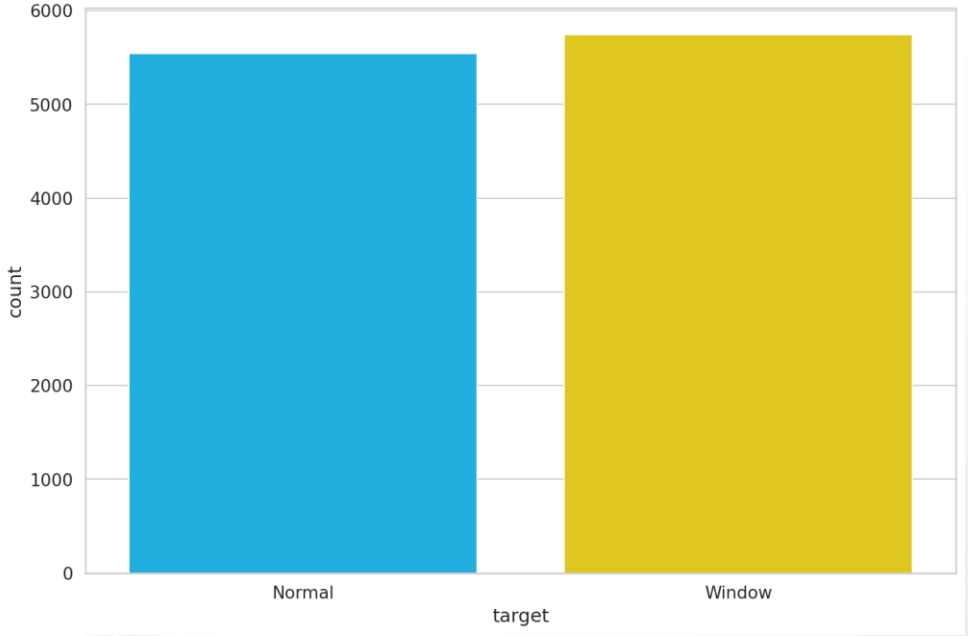
- 270°
- 540 rays
- 0.5° span per ray

Not used

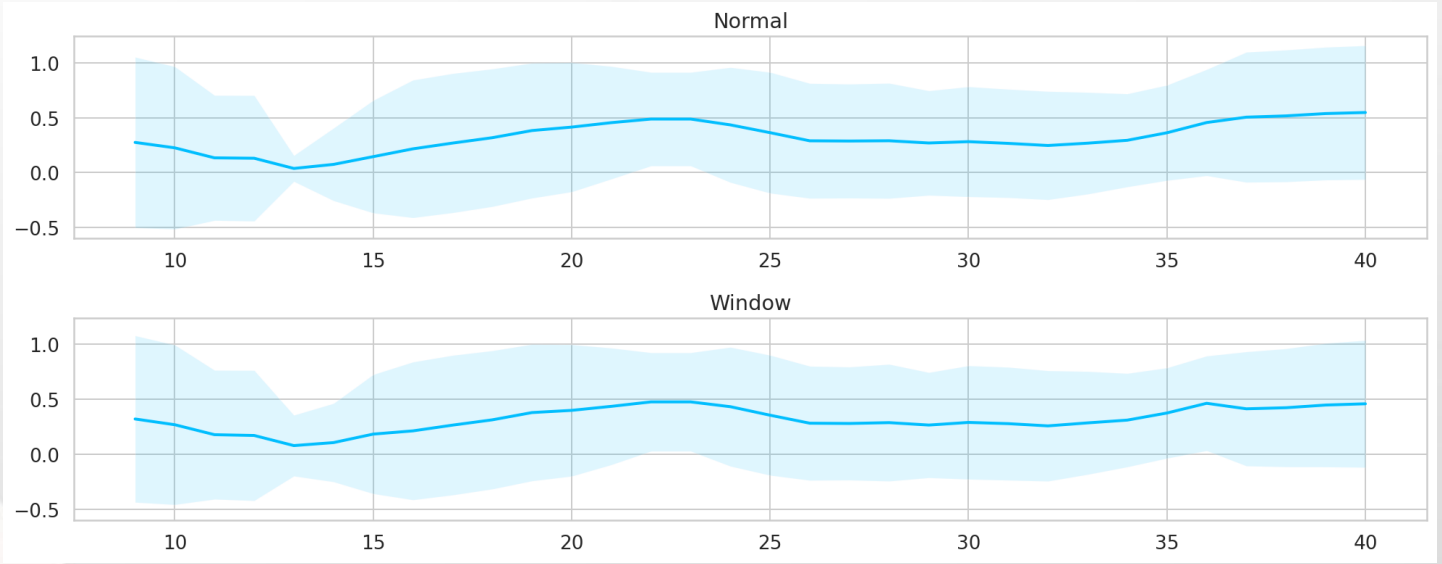
Selected 41 features/signals

1. Time
2. position.x
3. position.y
4. orientation.z
5. ranges_0
6. intensities_0
7. ranges_135
8. intensities_135
9. ranges_270
10. intensities_270
11. ranges_405
12. intensities_405
13. ranges_540
14. intensities_540
15. orientation_imu.x
16. orientation_imu.y
17. orientation_imu.z
18. angular_velocity.x
19. angular_velocity.y
20. angular_velocity.z
21. linear_acceleration.x
22. linear_acceleration.y
23. linear_acceleration.z
24. position_0
25. position_1
26. position_2
27. position_3
28. velocity_0
29. velocity_1
30. velocity_2
31. velocity_3
32. effort_0
33. effort_1
34. effort_2
35. effort_3
36. linear_cmd_vel.x
37. linear_cmd_vel.y
38. linear_cmd_vel.z
39. angular_cmd_vel.x
40. angular_cmd_vel.y
41. angular_cmd_vel.z

Exploratory Data Analysis



The normal class, has by far, the same number of examples. Normal series will be used to train the model



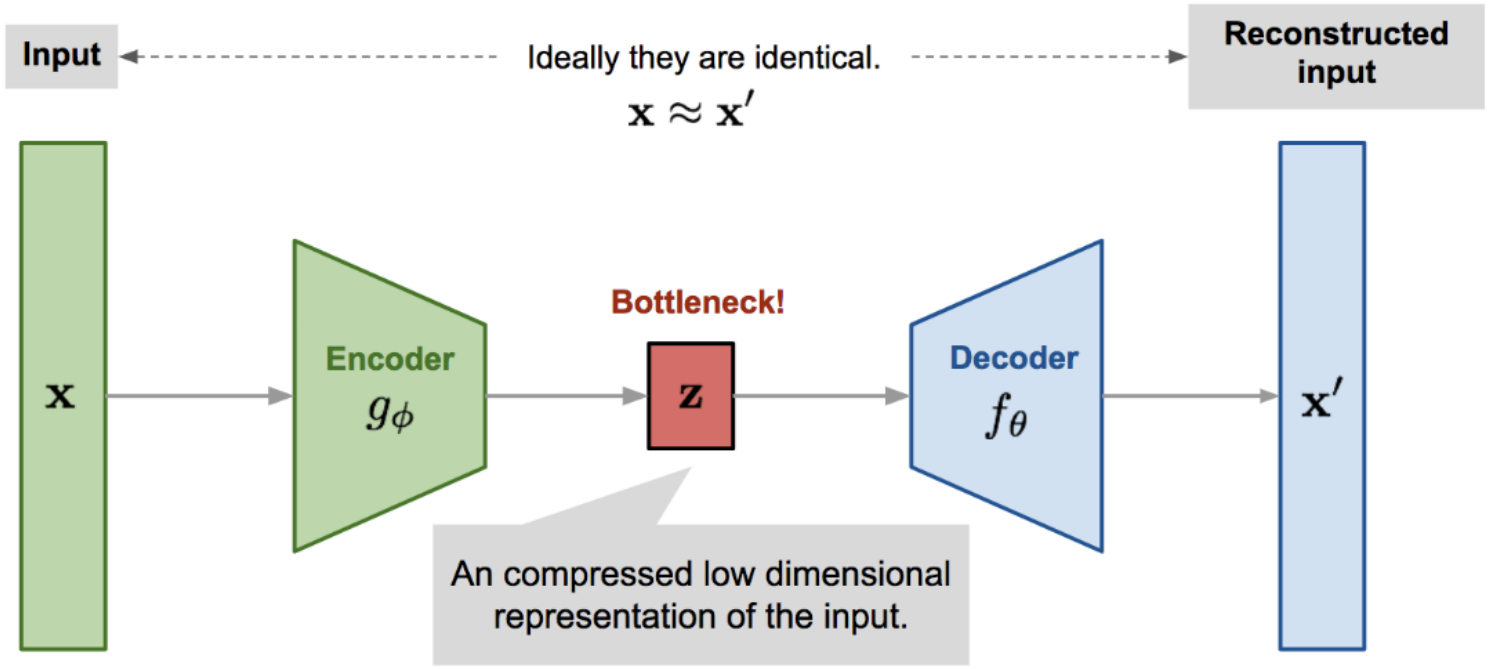
Display a (smoothed out with one standard deviation on top and bottom of it) Time Series for each class

Dataset Splitting

- Training Set: 80% of normal_plan + normal_plan1 (3992, 41)
- Validation Set: 10% of Training Set (444, 41)
- Test set: 20% Training set (1109, 41) + window_plan + window_plan1 (5742, 41)

4. Model Training

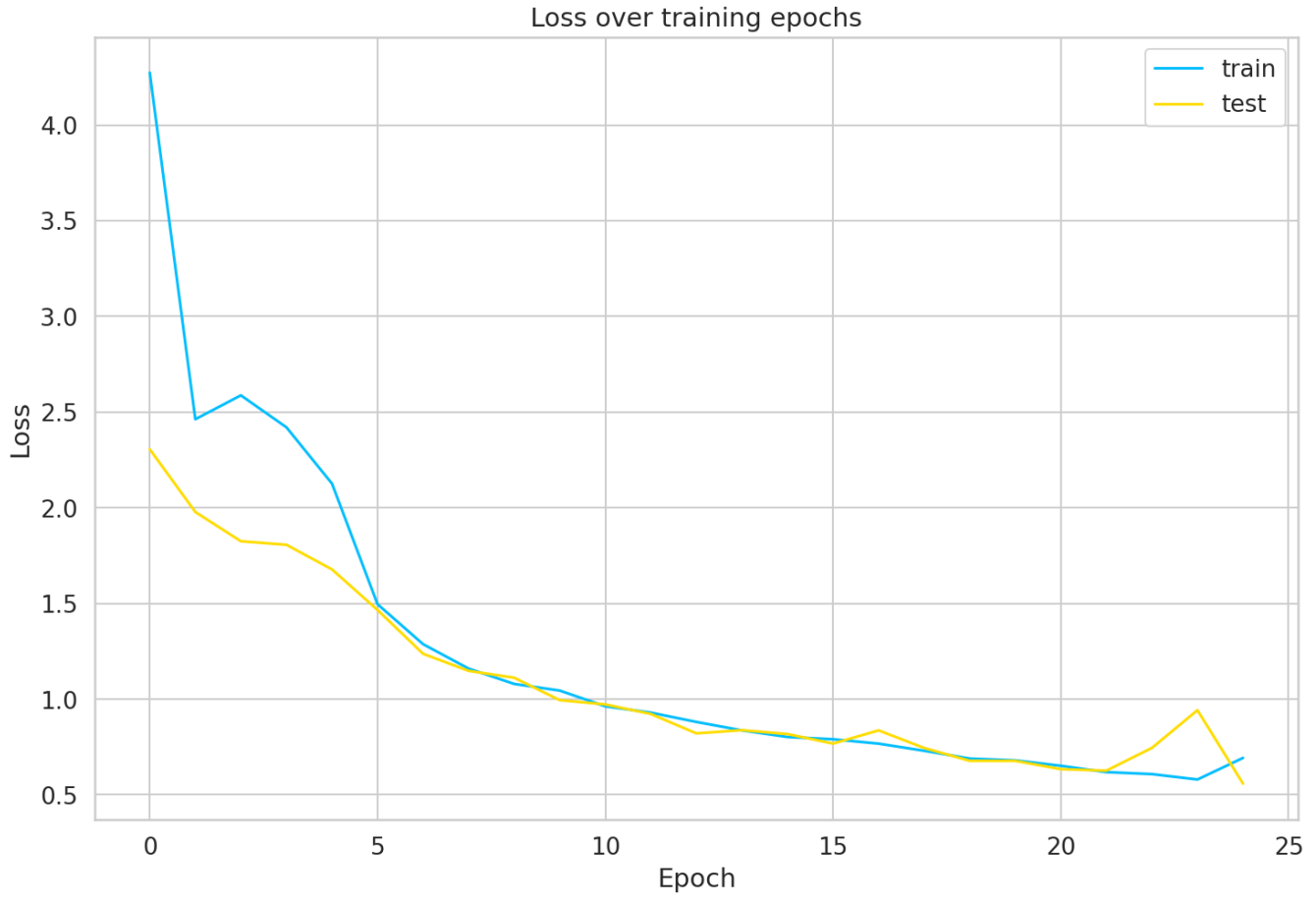
Model Definition: LSTM Autoencoder




```
==========================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================
RecurrentAutoencoder                     --                        --
├─Encoder: 1-1                           [1, 128]                  --
│    └─LSTM: 2-1                         [1, 41, 256]              265,216
│    └─LSTM: 2-2                         [1, 41, 128]              197,632
├─Decoder: 1-2                           [41, 1]                   --
│    └─LSTM: 2-3                         [1, 41, 128]              132,096
│    └─LSTM: 2-4                         [1, 41, 256]              395,264
│    └─Linear: 2-5                       [41, 1]                   257
==========================================================================
```

```
==========================================
Total params: 990,465
Trainable params: 990,465
Non-trainable params: 0
Total mult-adds (M): 40.61
==========================================
Input size (MB): 0.00
Forward/backward pass size (MB): 0.25
Params size (MB): 3.96
Estimated Total Size (MB): 4.21
==========================================
```

Training Loss

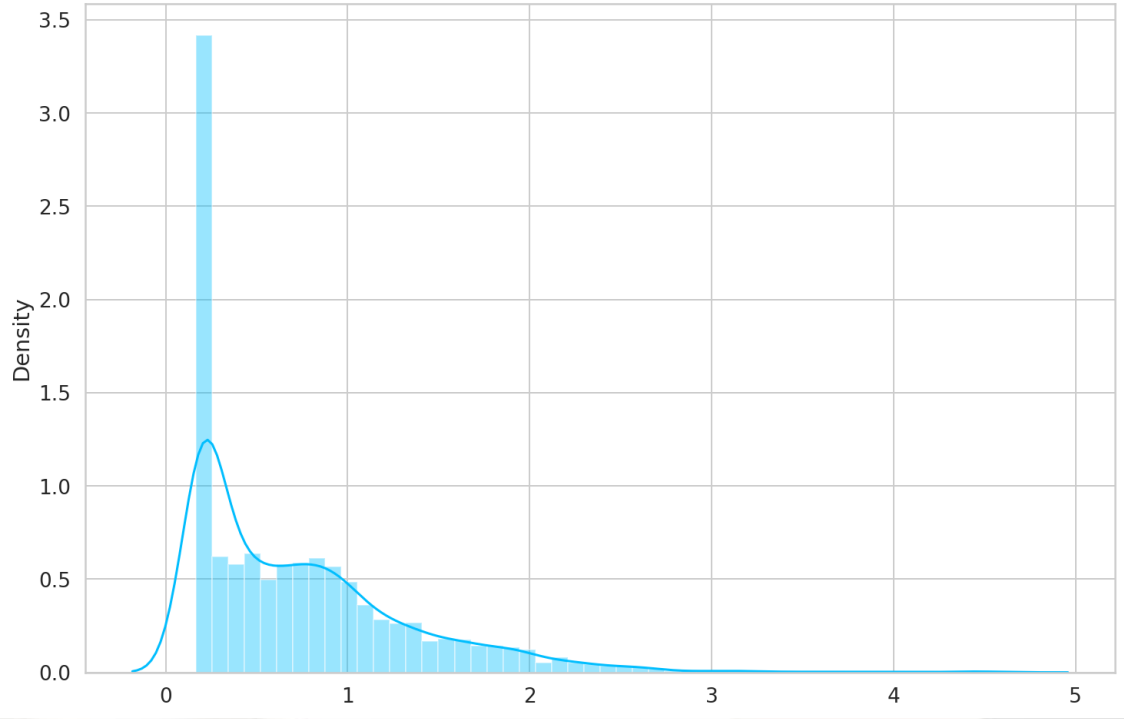


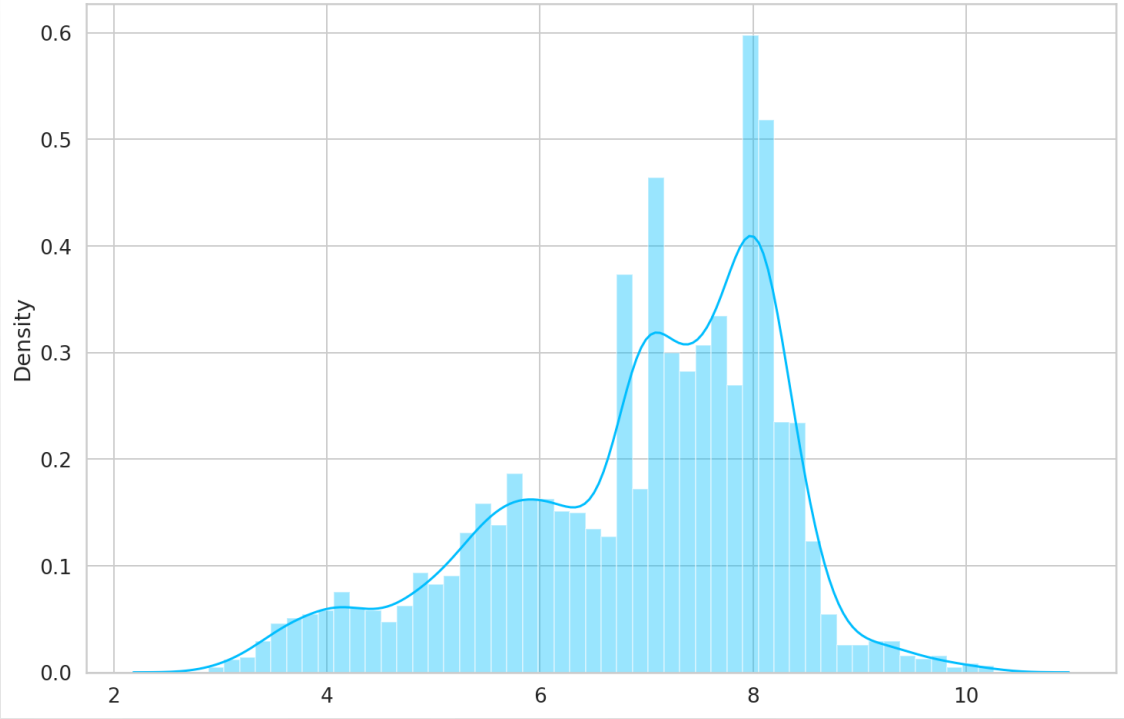
Minimize L1Loss

Creates a criterion that measures the mean absolute error (MAE) between each element in the input x and target y

Optimizer: Adam with LR=10e-3

Display of prediction loss

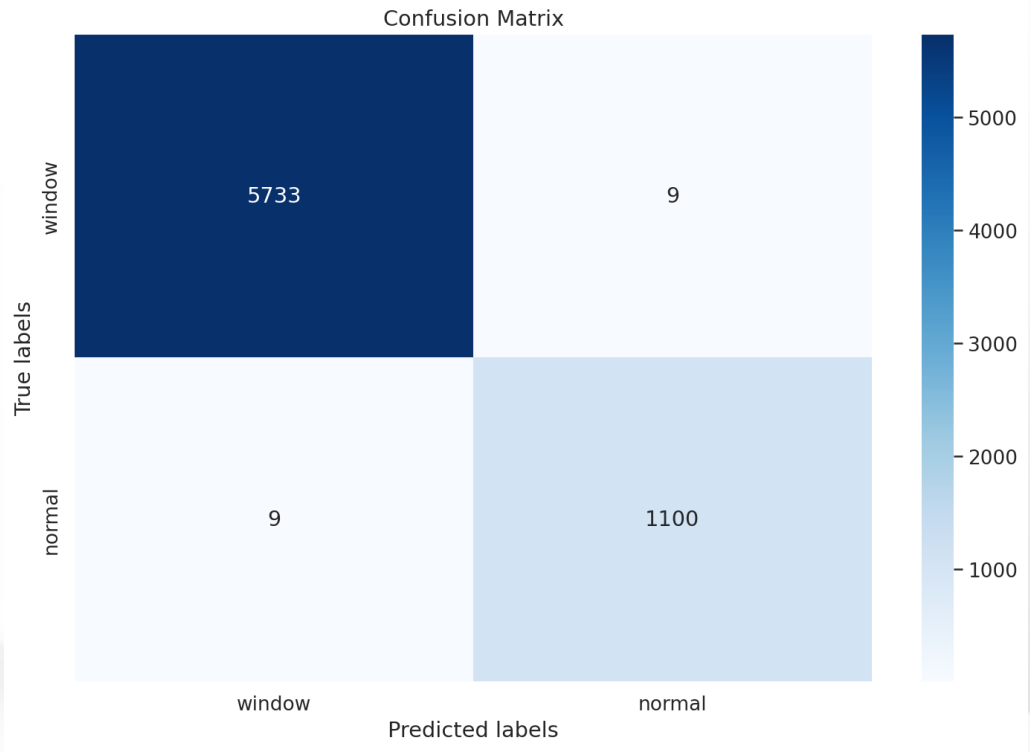


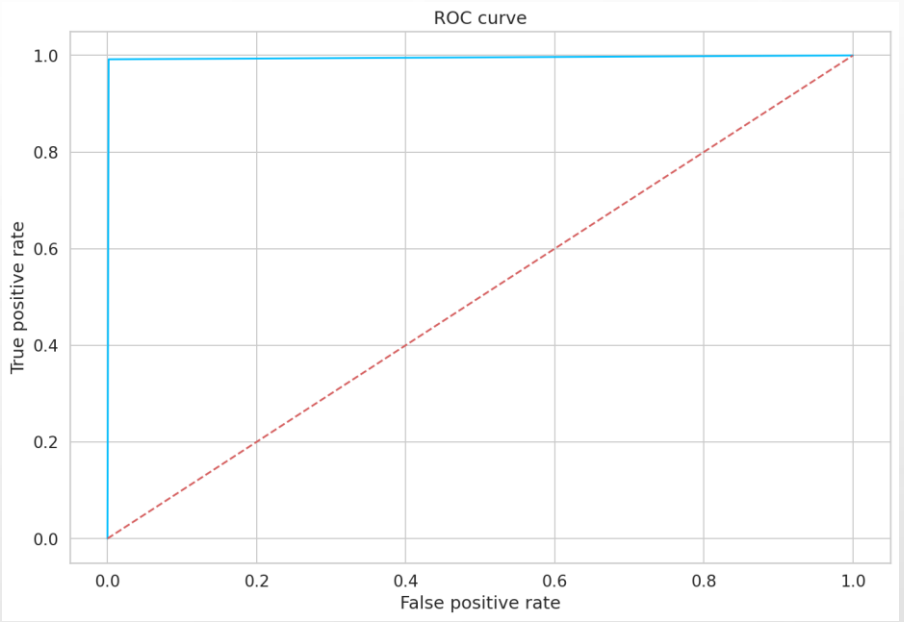


TRESHOLD DEFINITION: 3.1 (?)

Analysis of results:

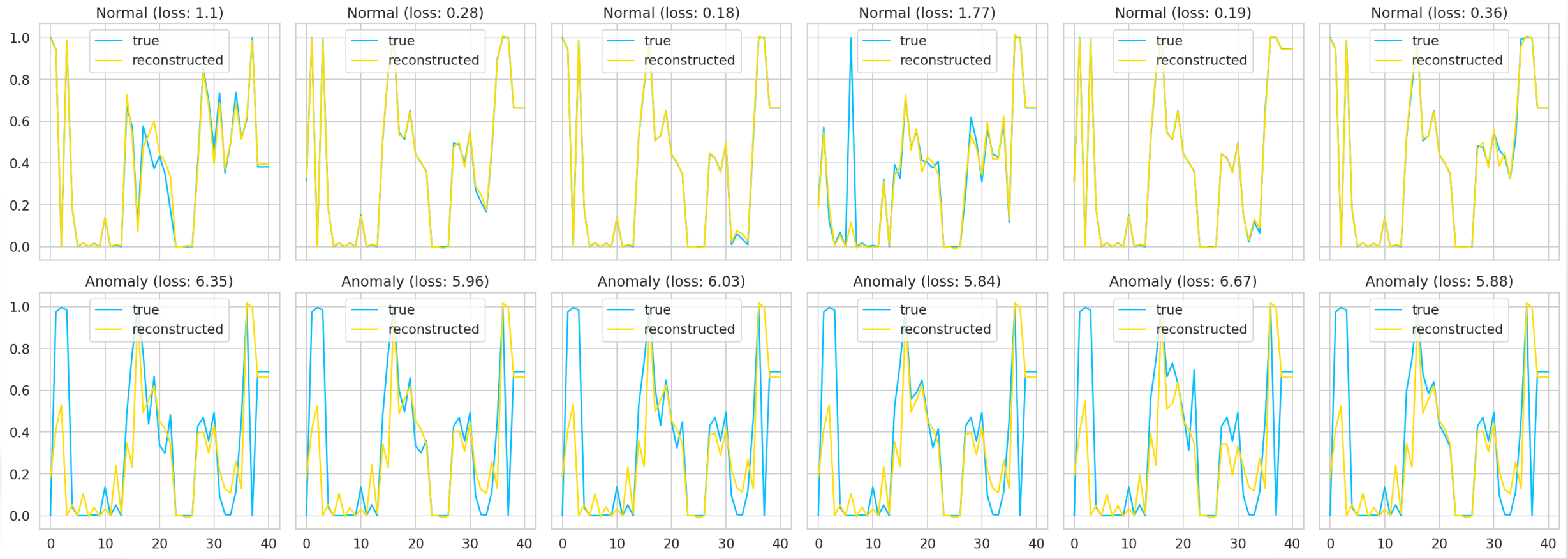
Threshold = 3.1





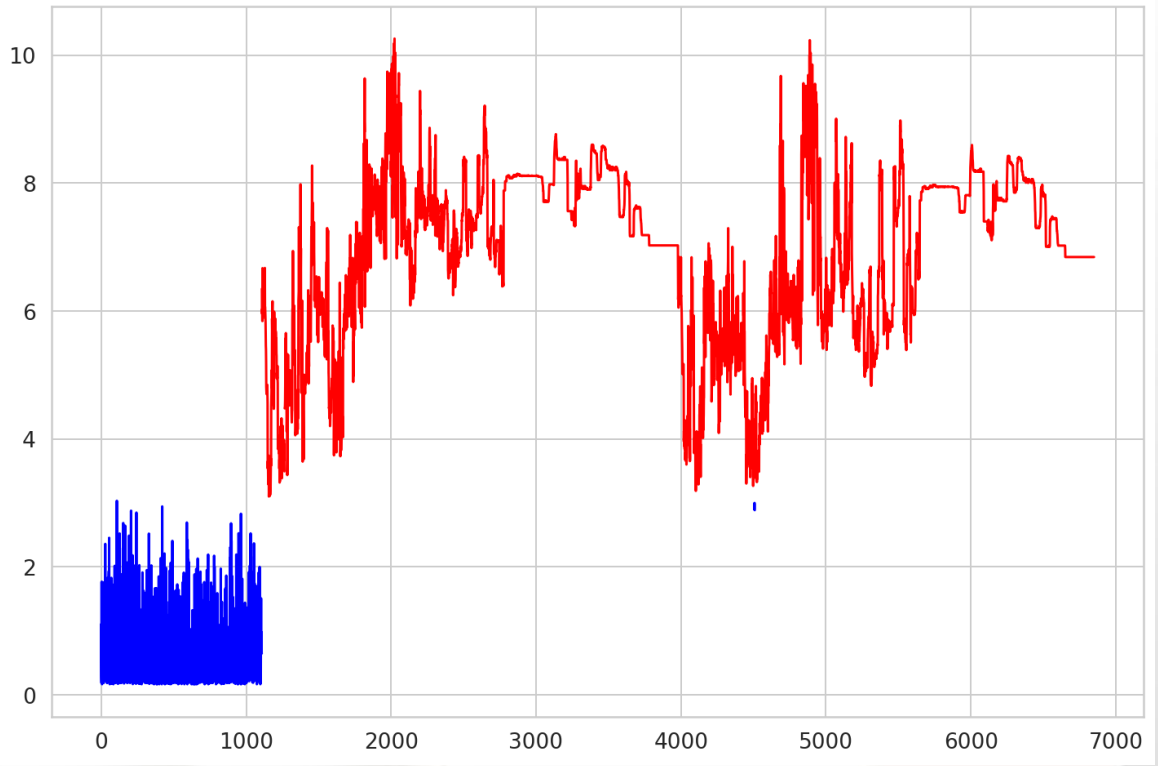
Precision: 0.991885
Recall: 0.991885
F1 score: 0.991885
Accuracy: 0.997373
Roc auc score: 0.995159

Reconstruction Error

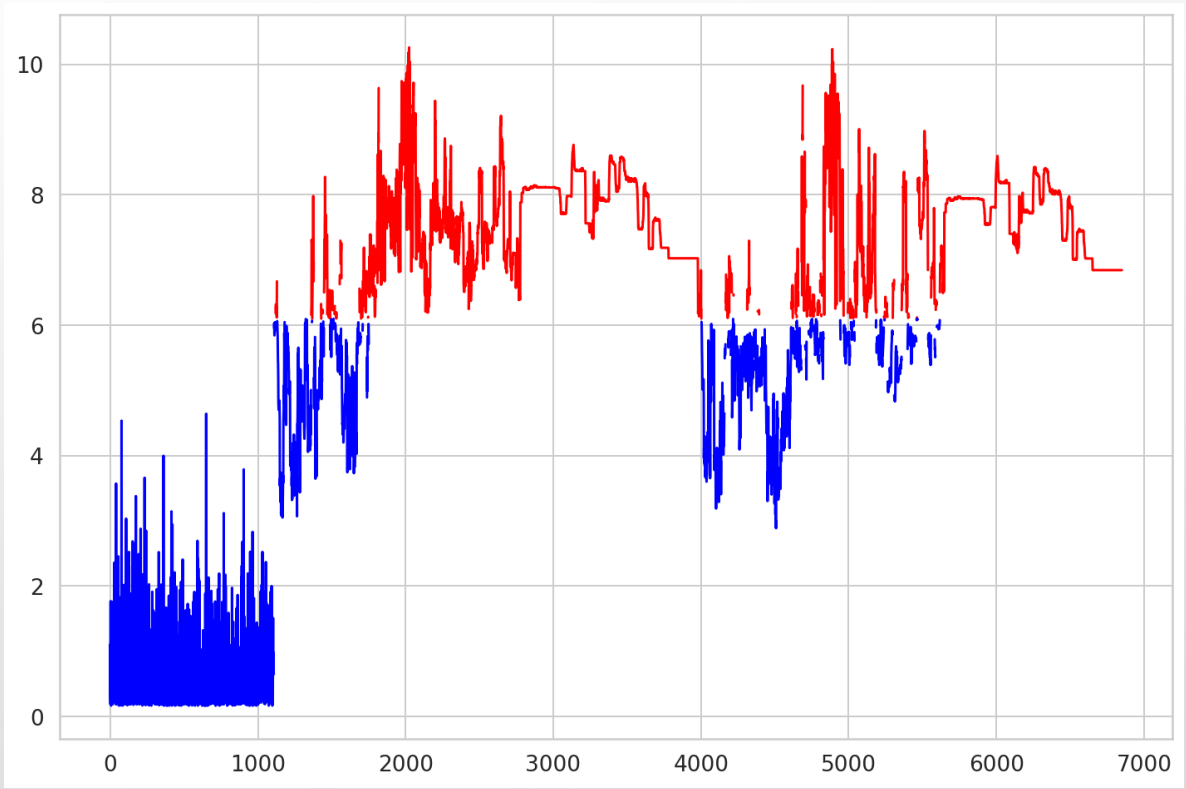


Treshold tuning

Treshold 3.1

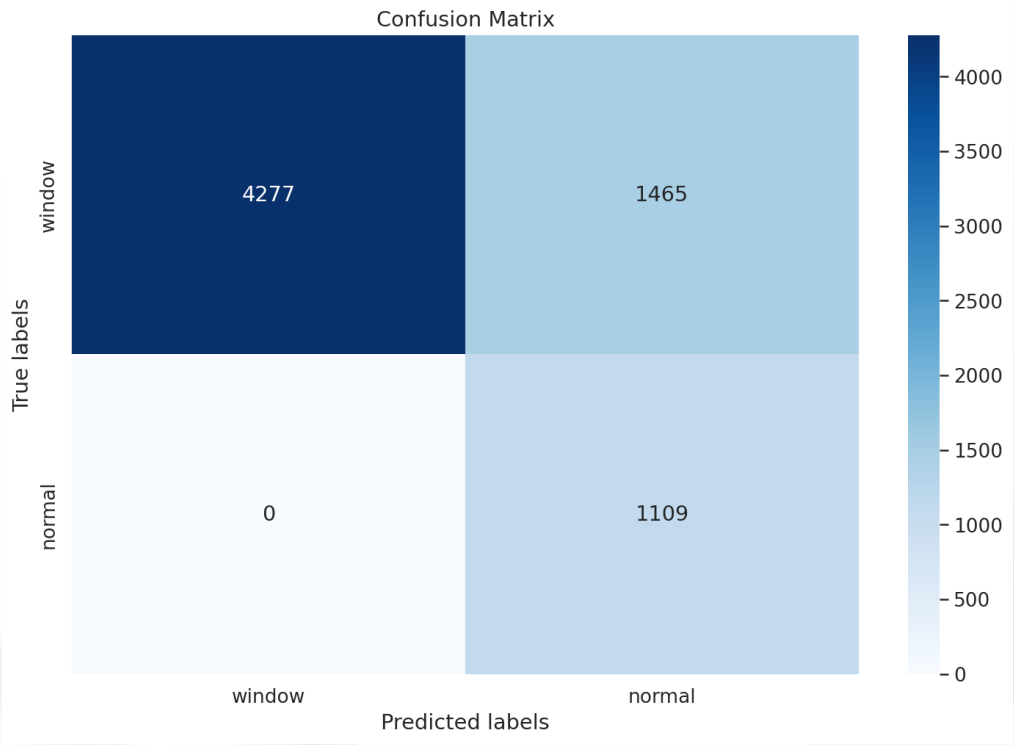


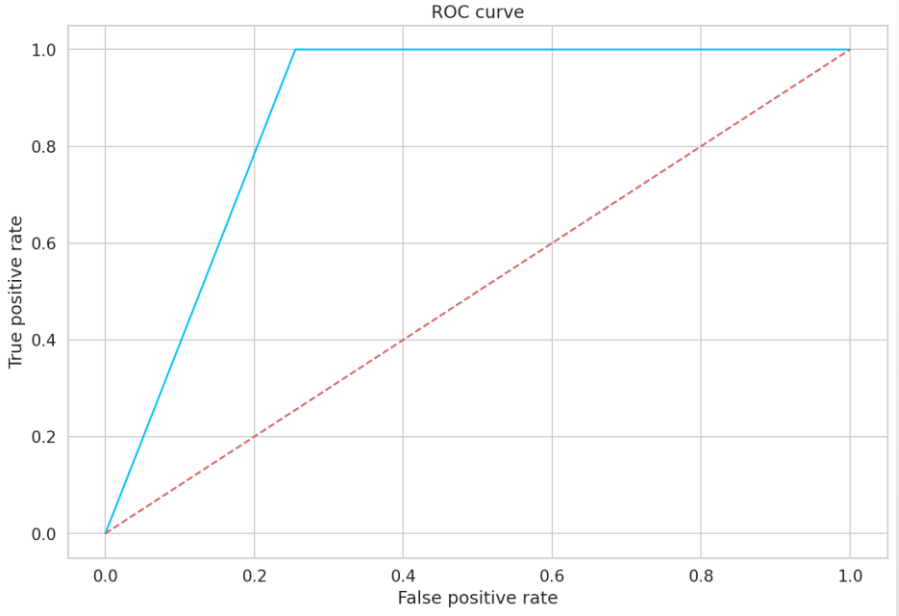
Treshold 6.1



Results of tuning:

Threshold = 6.1





Precision: 0.430847
Recall: 1.000000
F1 score: 0.602226
Accuracy: 0.786163
Roc auc score: 0.872431