Progress:

All 126 sets of predictors:

1. Only data that do not have bad in the filename were included.
2. From filename extract subject number and load his/her associated weight.
3. Missing data were replaced with next valid data per column.
4. Data were segmented as discussed, each window starts from the corresponding first non-zero value in the event to the last zero value before the next first non-zero.
5. Resulting windows' lengths varied between 17 data point per window to 337 data point per window. The total number of windows was 6667.
6. 17 data per window is a very short window length, I took only the windows that had minimum 136 data-point (Arbitrary choice). At the end I had 4835 different windows (1832 windows were discarded because they had less than 136 data-point).
7. Zero padding at the was applied on all windows to have 337 data point.
8. Resulting predictor dataset was: 4835*337*126 (126 different predictor). The outcome columns are divided by the corresponding mass.
9. Area was computed from the outcome vectors. Some statistics of the 4835 areas are given in Table 1.
10. The windows were then shuffled and split into train-test.
11. Segments were standardised to have a mean of 0 (remove dc component) and variation of standard deviation.
12. State-of-the-art model [InceptionTime] was used for training. All steps summarised in Fig. 1.
13. Later a custom 2d channel CNN was also used for training to compare with.

Table 1 Area statistics

Mean	Min value	Max value	Standard deviation
-28.85806810	-87.43672118619698	5.740925581	12.1669867

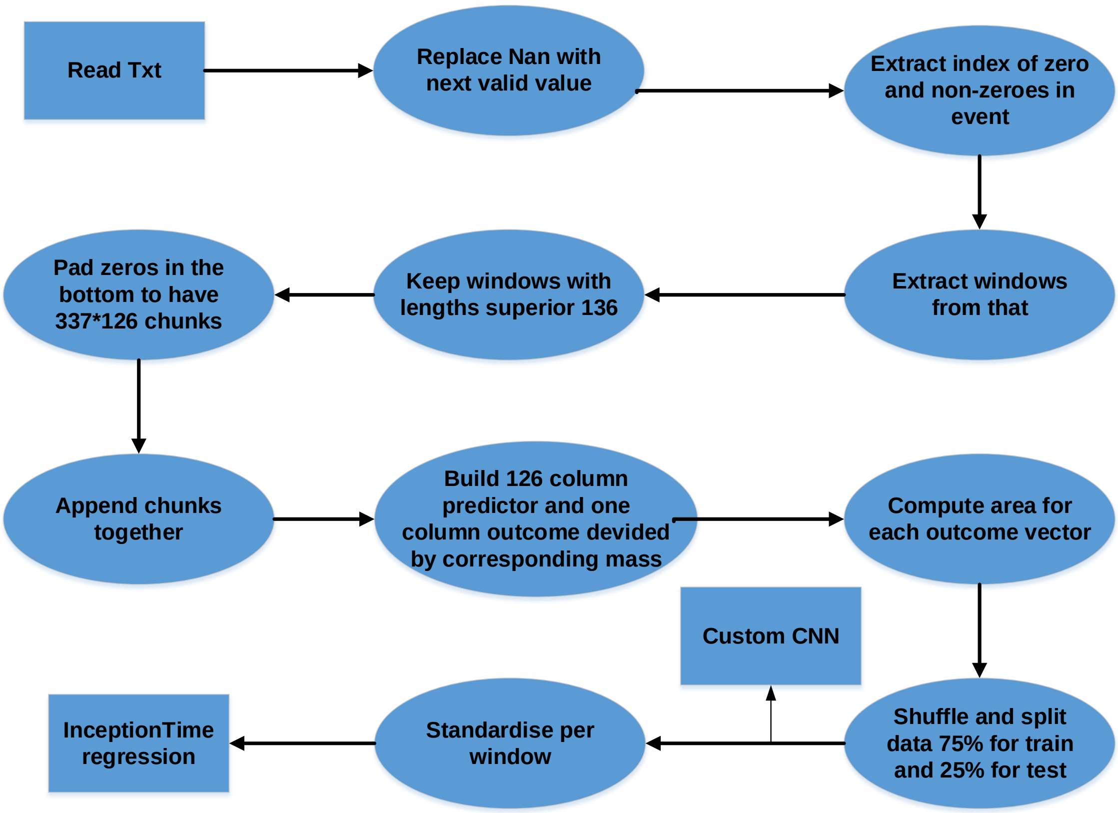


Fig. 1 flow chart of the processing

Fig. 2 Shows 9 different windows of 126-predictors each with the expected prediction above each image. You can see the zeros added all as constant curves at the right of each signal.

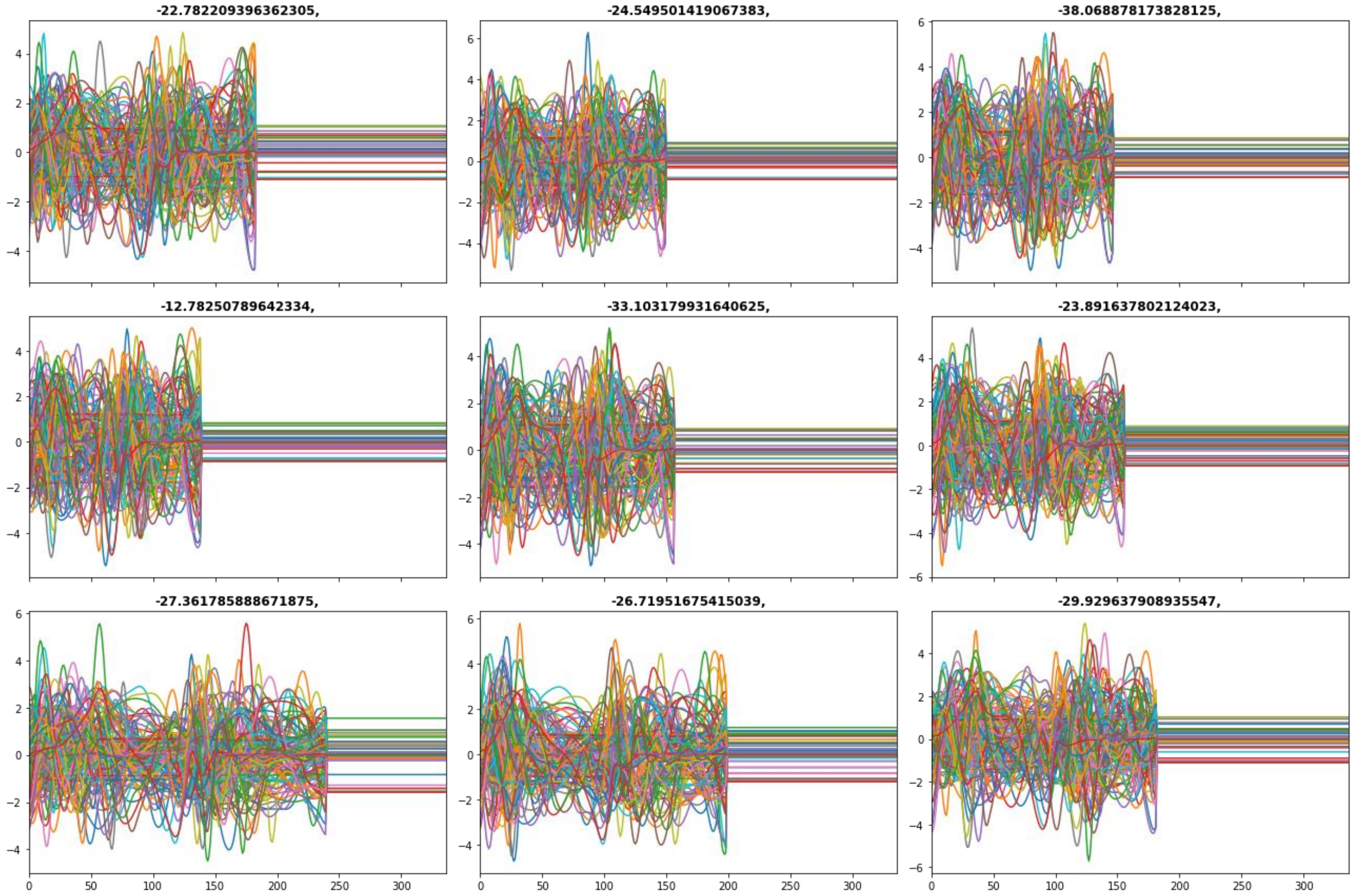


Fig. 2 Random windows with their associated area

After 100 very fast training epochs, I had the results summarised in Table 2. The metric plots are given in Fig. 3.

Table 2 Results after 100 epochs

Epoch	train_loss	valid_loss	mae	_rmse	mape	Time/epoch
100	1.694194	5.405107	1.703848	2.324889	7.6%	00:01

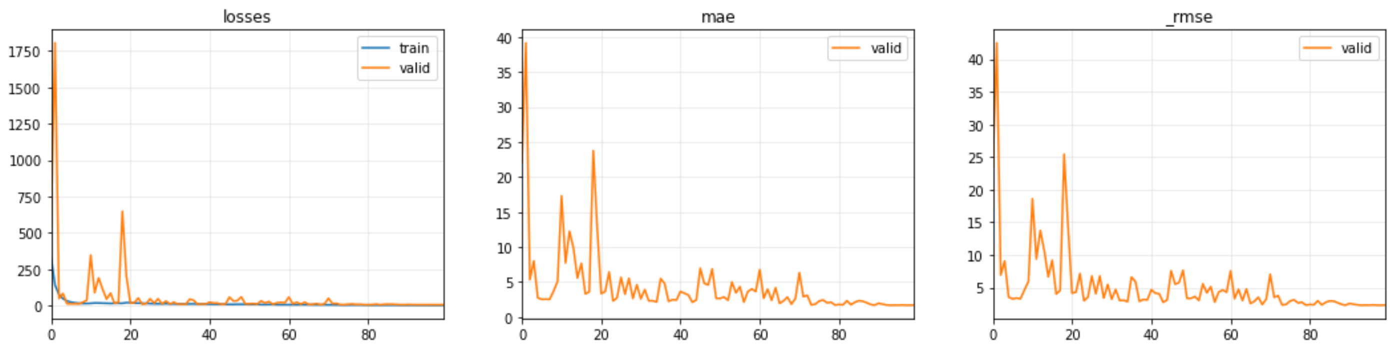


Fig. 3 Metrics after 100 epochs

Some predictions are given below, with the corresponding windows and the actual are. You can see that the values are close. Mape 7.6% is an excellent one!

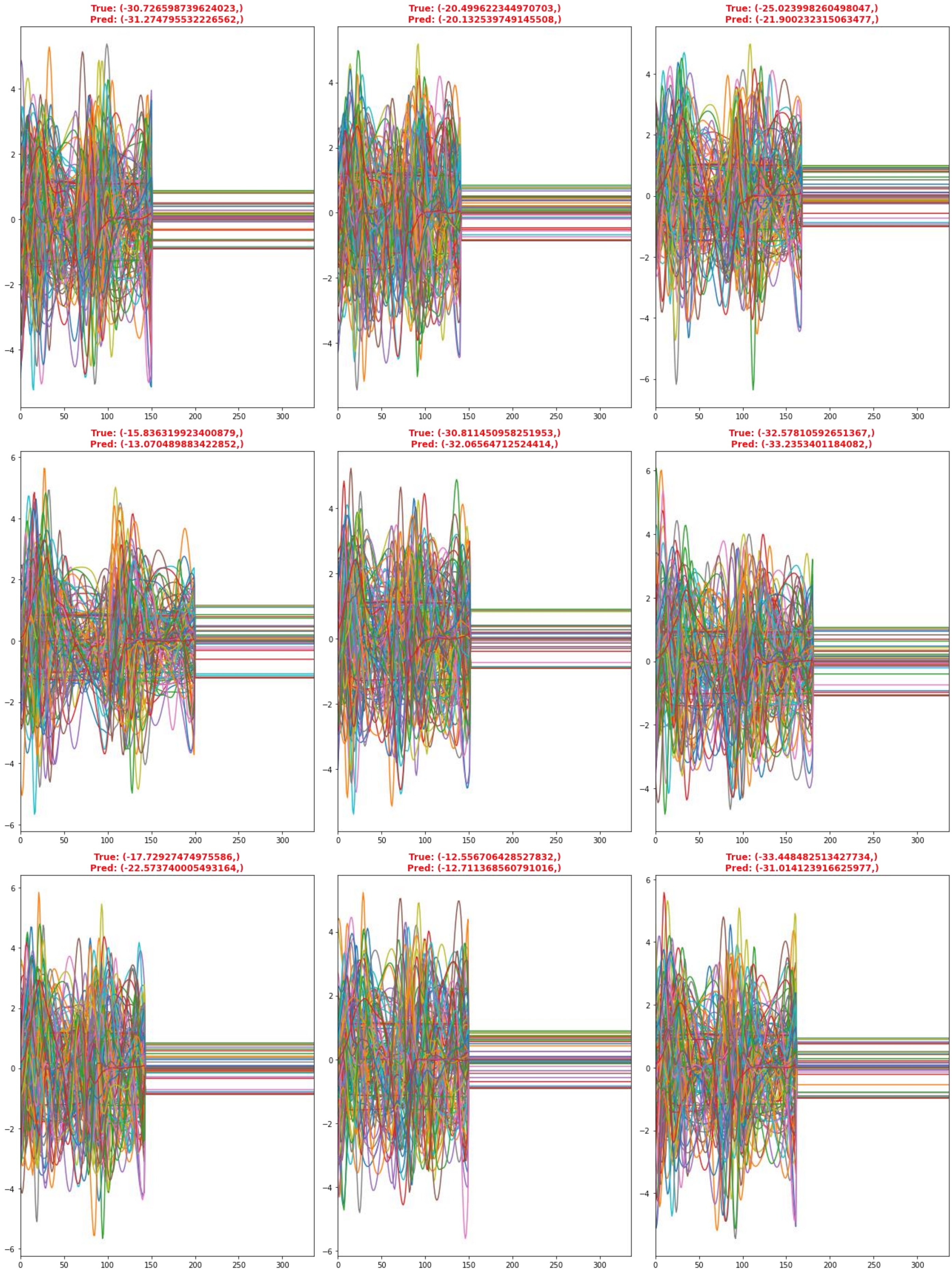


Fig. 4 predicted vs real area.

Now feeding the Raw data to a custom CNN model described below and training for 100 epochs gave accuracies given in Table 3 for epoch 39, after that the model started to overfit.

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_2 (Conv2D)            (None, 335, 124, 64)      640

conv2d_3 (Conv2D)            (None, 333, 122, 128)     73856

max_pooling2d_1 (MaxPooling2 (None, 166, 61, 128)      0

flatten_1 (Flatten)          (None, 1296128)           0

dense_4 (Dense)              (None, 512)               663618048

dropout_1 (Dropout)          (None, 512)               0

dense_5 (Dense)              (None, 256)               131328

dense_6 (Dense)              (None, 128)               32896

dense_7 (Dense)              (None, 1)                 129
```

Table 3 Best achieved custom CNN metrics

Epoch	train_loss	valid_loss	mae	_rmse	mape	Time/epoch
100	6.54	51.34	4.86	2.324889	17.87%	00:17

Furthermore other models namely: ExceptionTime, ResNet, FCN, TCN, XCM, RNN_FCN, TransformerModel, InceptionTime ,XCM were used and the results are given in Table 4 below.

Table 4 comparaison between all models trained with 126 predictors' vectors

	Standardised with 126 features					
	train_loss	valid_loss	mae	_rmse	mape	Time/epoch
InceptionTime	1.694194	5.405107	1.703848	2.324889	6.264964	00:01
Custom CNN	6.54	51.34	4.86	2.324889	17.86998	00:17
Exception	981.8322	977.89	28.91853	31.27123	106.332	00:01
ResNet	2.103211	5.681127	1.757577	2.383512	6.462523	00:00
FCN	2.749499	5.801149	1.777879	2.408557	6.537172	00:00
TCN	2.776684	6.115119	1.868325	2.472877	6.869738	00:01
XCM	1.999519	5.384712	1.696015	2.320498	6.236162	00:23
RNN_FCN	3.507069	6.116888	1.836237	2.473234	6.751752	00:01
TransformerModel	17.72151	470.6854	18.33376	21.69529	67.4123	00:00
	Raw data with 126 features					
InceptionTime	1.915389	4.798767	1.62254	2.190609	5.965998	00:01
XCM	1.911112	9.529335	2.410025	3.086962	8.861541	00:23
	Normalised GASF 337					
XResnet 18	15.746682	13.012957	2.773102	3.607347	10.696557	11:34

We see that the XCM model performed slightly better that the inceptionTime.

Feeding raw data directly to the models without standardisig improved the results on inception time and achieved 5.965998 % mape only!

Prediction are given in Fig. 6 and evolution of the metrics

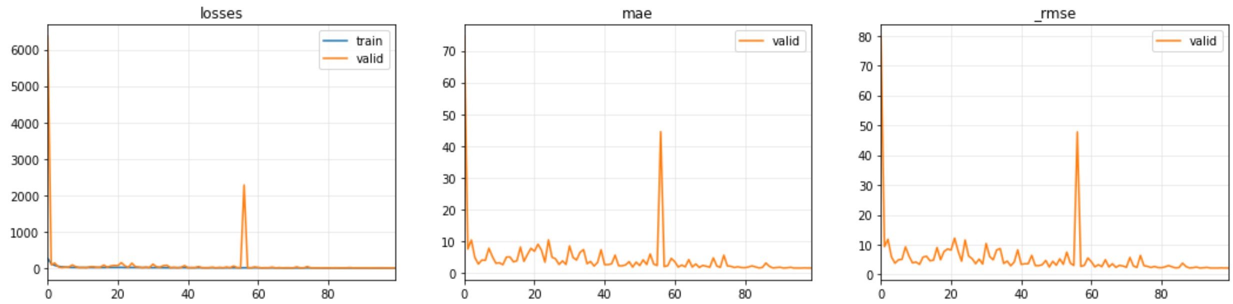


Fig. 5 evolution of metrics for best performing model

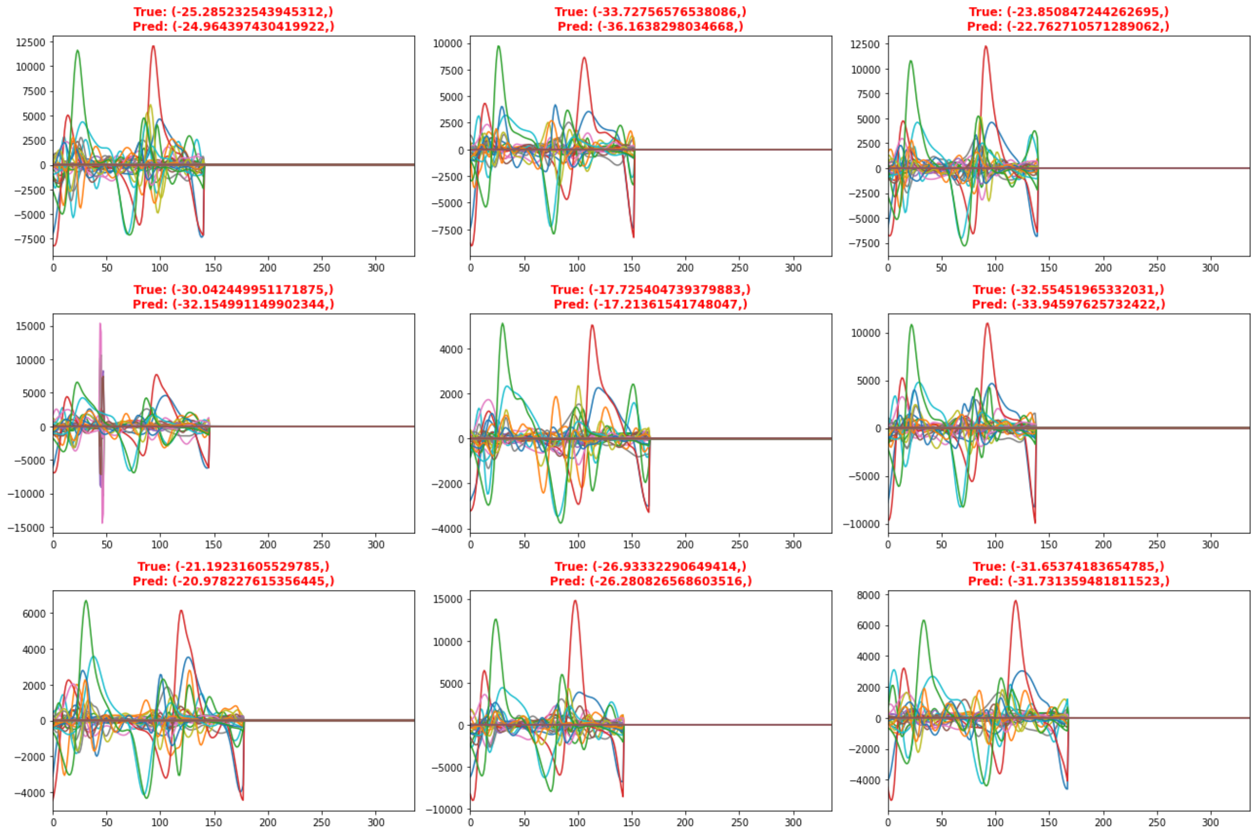


Fig. 6 Raw data InceptionTime

42 sets of predictors:

Now we merge the x,y,z components of the predictors together and compute the magnitudes, this reduces the feature vector from 126 to 42, we also tested different preprocessing: normalisation, standardisation (per sample or per variable) and raw data, results for different tested models are given in below in Table 5.

Table 5 different results for 42 feature vectors

	Raw with 42 features					
	train_loss	valid_loss	mae	_rmse	mape	Time/epoch
InceptionTime	3.580549	13.93484	2.749375	3.73294	10.10931	00:00
XCM	2.839566	11.53357	2.420306	3.39611	8.899344	00:08
	Raw with 42 features					
InceptionTime	2.374141	13.976	2.711294	3.738449	9.969292	00:01
	Normalised data with 42 features					
InceptionTime	4.164655	12.75583	2.624736	3.57153	9.651023	00:01
	Normalised per sample per variable					
InceptionTime	5.640666	11.61801	2.511483	3.40852	9.234597	00:00
GADF XRESNET18	47.686832	95.977089	7.965482	9.796789	29.2886877	03:25

For this feature space XCM performed slightly better than the others with about 8.9% mape.

Some predictions are given in Fig. 7.

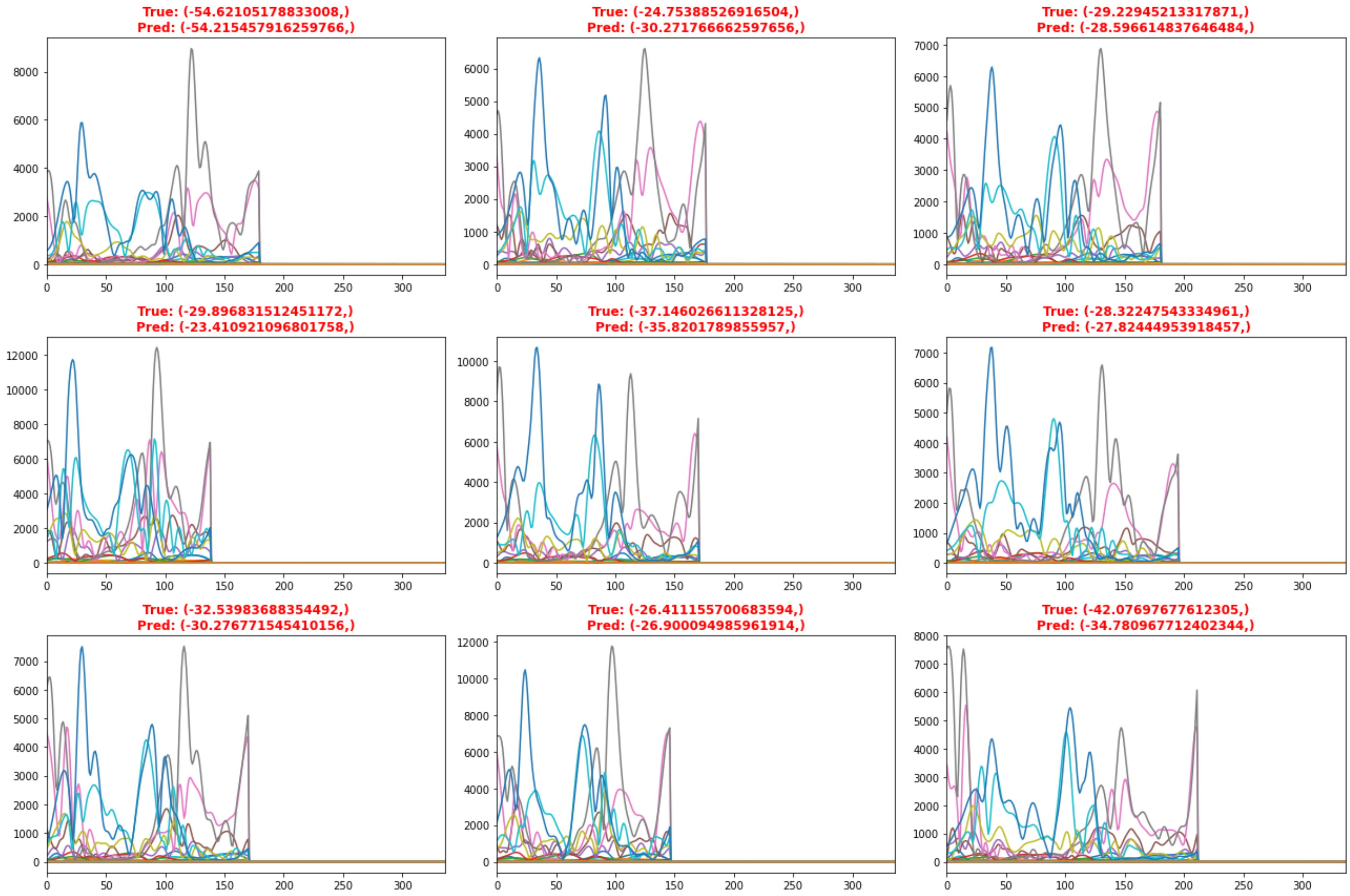
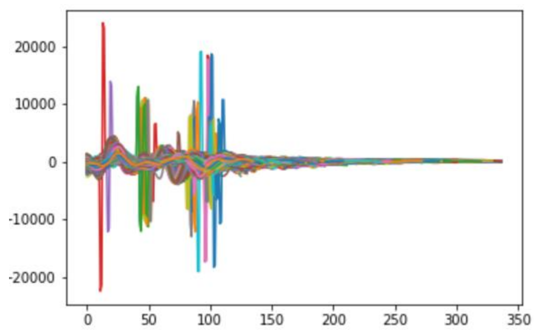


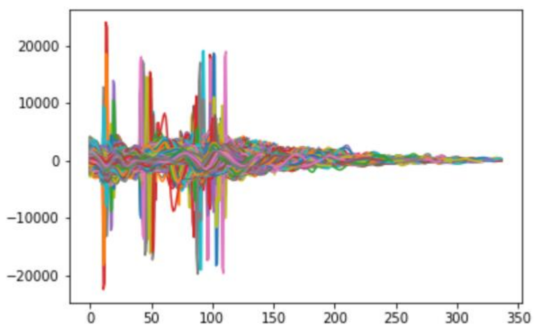
Fig. 7 predictions for 42 predictors raw XCM model

New segmentation:

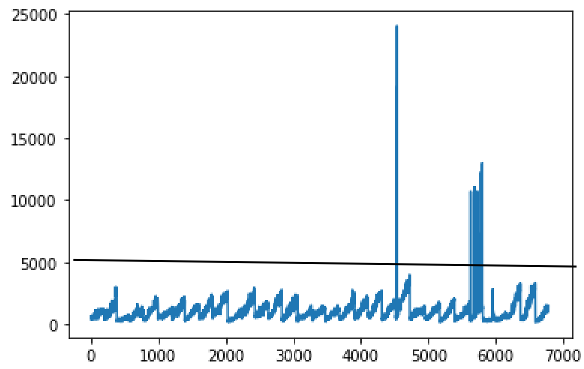
Using the event file, the data has been segmented. Results were similar to the first one. Only segments whose length are superior to 110 and inferior to 300 were included. Segments with lengths inferior to 300 were padded with zeros.

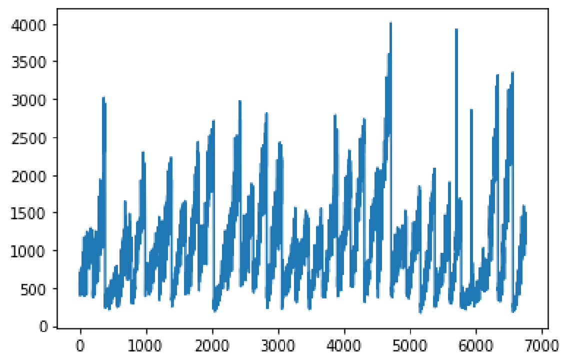
Some of the segments were noisy and had spikes (acceleration data) like bellow:



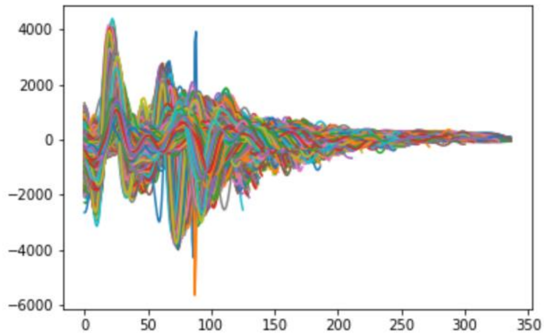


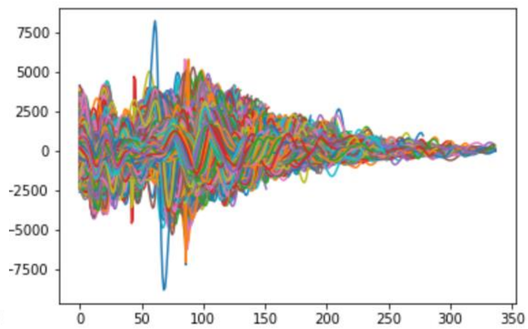
In order to remove segments that created problems, max, min, range were computed for each segment and then plotted. A magnitude threshold was set to extract the indices of corrupted segments.



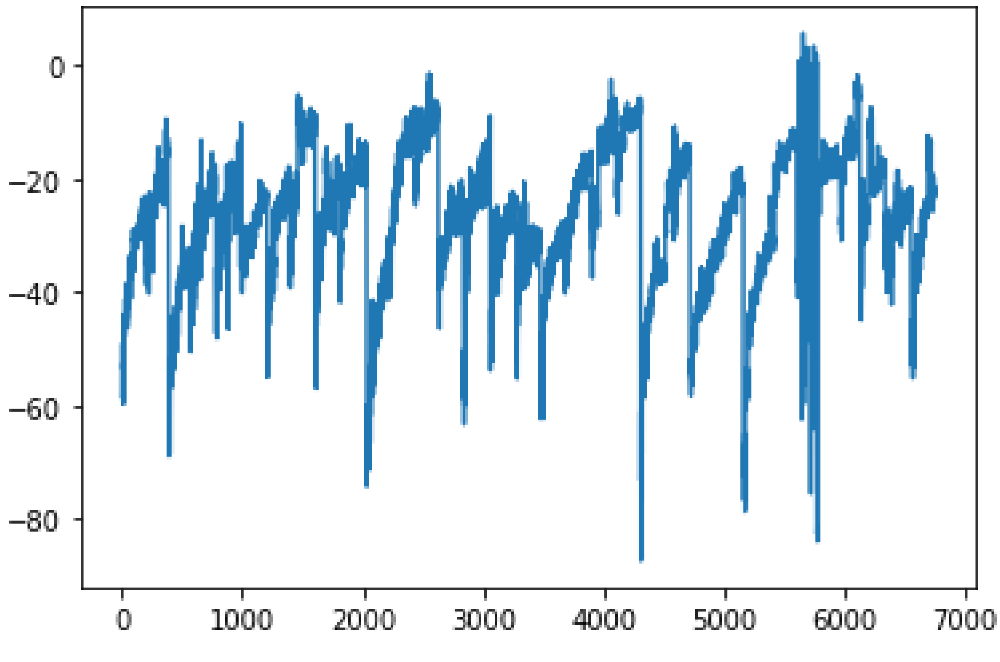


Same predictors are shown below:

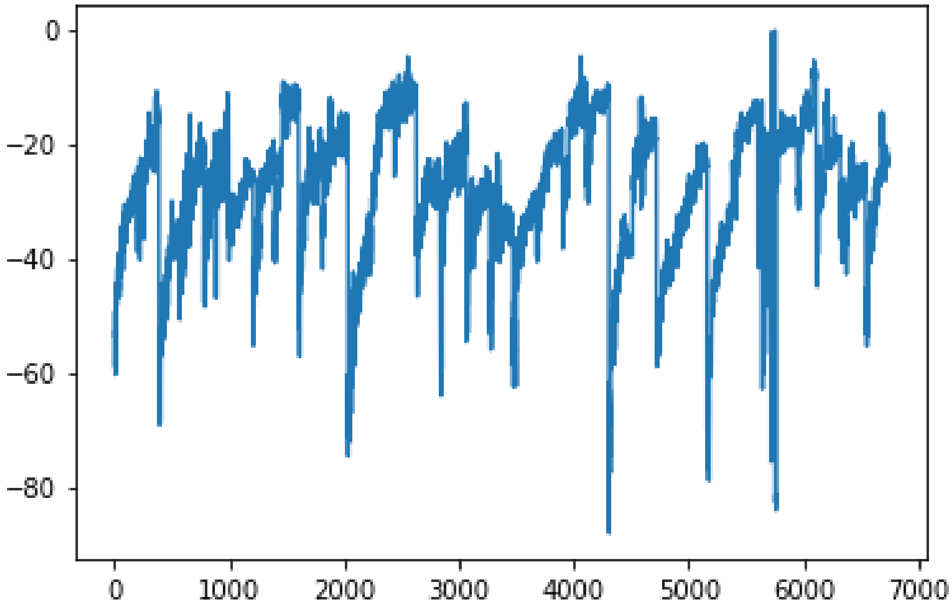




Area/ segment given bellow



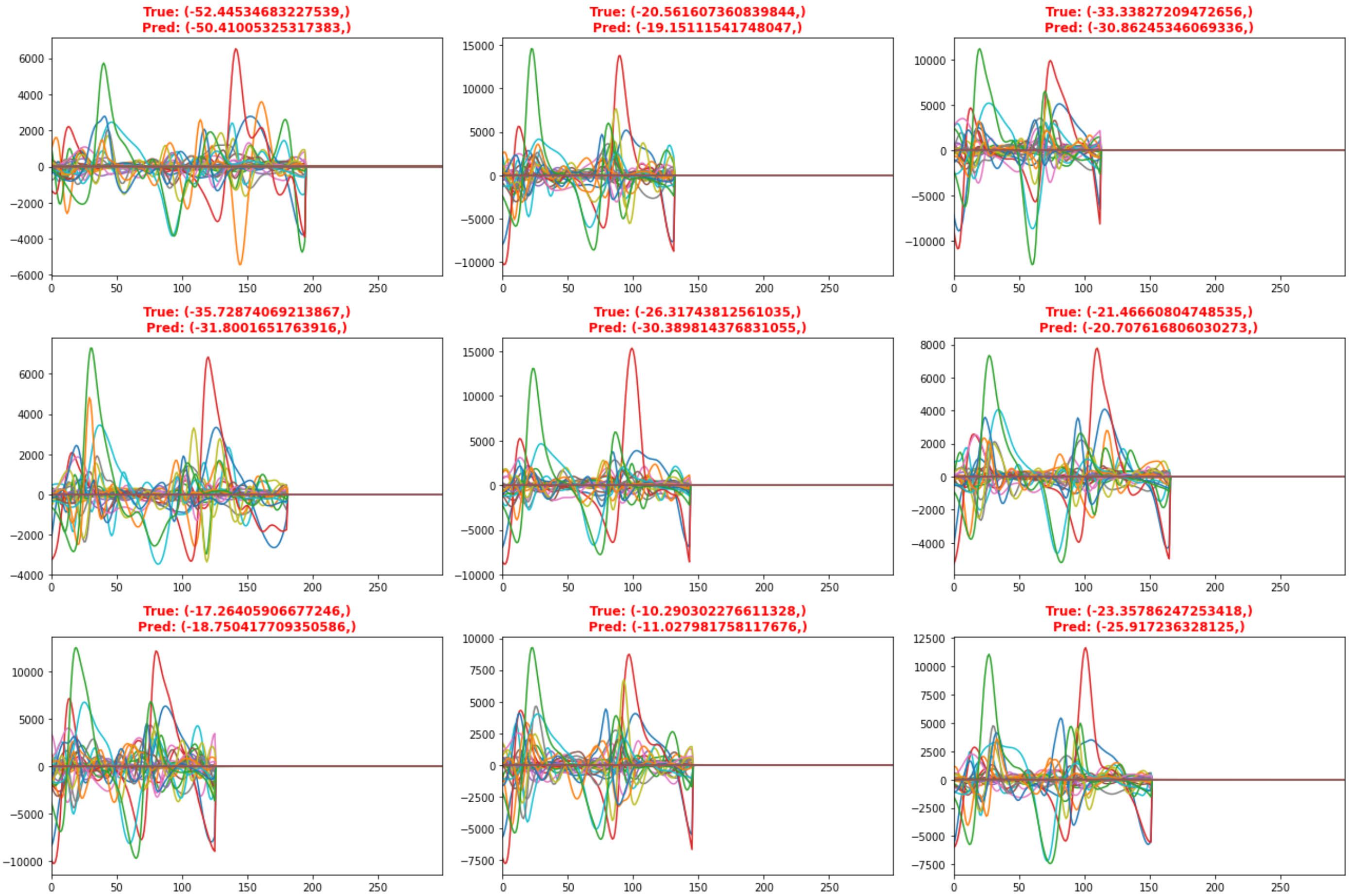
Some segments () have positive areas, those segments are noisy and were filtered



	Raw data with 126 features					
	train_loss	valid_loss	mae	_rmse	MAPE	Time/epoch (MM:SS)
InceptionTime_pre	4.282371	5.541217	1.695879	2.353979	5.935577	00:01
InceptionTime	6.701561	6.048764	1.758357	2.459424	6.15425	00:01
ResNet	5.278995	6.102412	1.768327	2.470306	6.189145	00:01
RNN_FCN	7.240803	7.355473	1.949827	2.712097	6.824395	00:01
Custom CNN	8.911	16.9749	2.7818	3.45646	9.7363	00:21
Exception	920.4557	929.74	28.28388	30.49164	98.99357	00:02
Normalised GADF 126 channels (300 X300)						
XResnet 18	19.89229	21.33288	3.447948	4.618753	12.06782	12:30
Normalised GADF 126 channels (400 X400)						
XResnet 18	21.36648	25.90076	3.866157	5.08928	13.53155	20:26

After that, different classifiers were tried as seen bellow:

Results for best performing regression:



Conclusion

Different models, feature spaces, preprocessing were experimented. InceptionTime with Raw data for a set of 126 feature vector performed best.