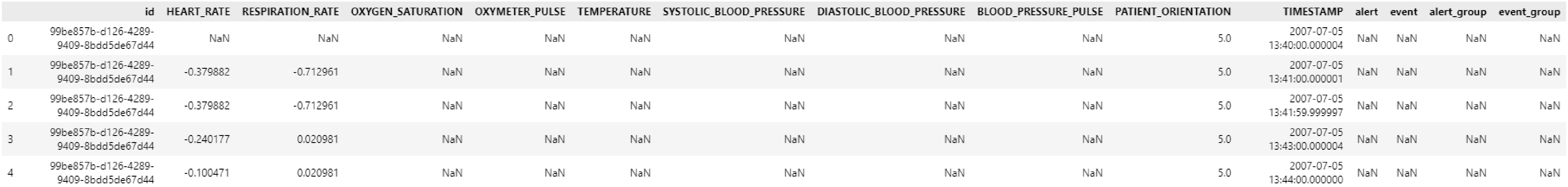
Ward data

A deep dive into the world of insufficient data

Carl Ivarsen Askehave and Andreas Hjortsø

What is the Ward dataset

- Biometric measurements from 1300+ real patients in the Danish healthcare system – 3 probes
- Timeseries covering 2-5 days with one minute resolution ~ 6M datapoints
- 721 severe adverse events (SAE) across all patients
  - What is an SAE? - 5 groups
- Ward alerts – the competition
- Real people -> messy data



	id	HEART_RATE	RESPIRATION_RATE	OXYGEN_SATURATION	OXYMETER_PULSE	TEMPERATURE	SYSTOLIC_BLOOD_PRESSURE	DIASTOLIC_BLOOD_PRESSURE	BLOOD_PRESSURE_PULSE	PATIENT_ORIENTATION	TIMESTAMP	alert	event	alert_group	event_group
0	99be857b-d126-4289-9409-8bdd5de67d44	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	5.0	2007-07-05 13:40:00.000004	NaN	NaN	NaN	NaN
1	99be857b-d126-4289-9409-8bdd5de67d44	-0.379882	-0.712961	NaN	NaN	NaN	NaN	NaN	NaN	5.0	2007-07-05 13:41:00.000001	NaN	NaN	NaN	NaN
2	99be857b-d126-4289-9409-8bdd5de67d44	-0.379882	-0.712961	NaN	NaN	NaN	NaN	NaN	NaN	5.0	2007-07-05 13:41:59.999997	NaN	NaN	NaN	NaN
3	99be857b-d126-4289-9409-8bdd5de67d44	-0.240177	0.020981	NaN	NaN	NaN	NaN	NaN	NaN	5.0	2007-07-05 13:43:00.000004	NaN	NaN	NaN	NaN
4	99be857b-d126-4289-9409-8bdd5de67d44	-0.100471	0.020981	NaN	NaN	NaN	NaN	NaN	NaN	5.0	2007-07-05 13:44:00.000000	NaN	NaN	NaN	NaN

The ward vision

- Imagine being able to predict every time a SAE happens, while also getting a description of the type of event, as well as relevant biometric measurements regarding the event.

- Imagine the nurses having an app with a score for every patient in their unit showing the probability of a SAE happening within the next few hours.

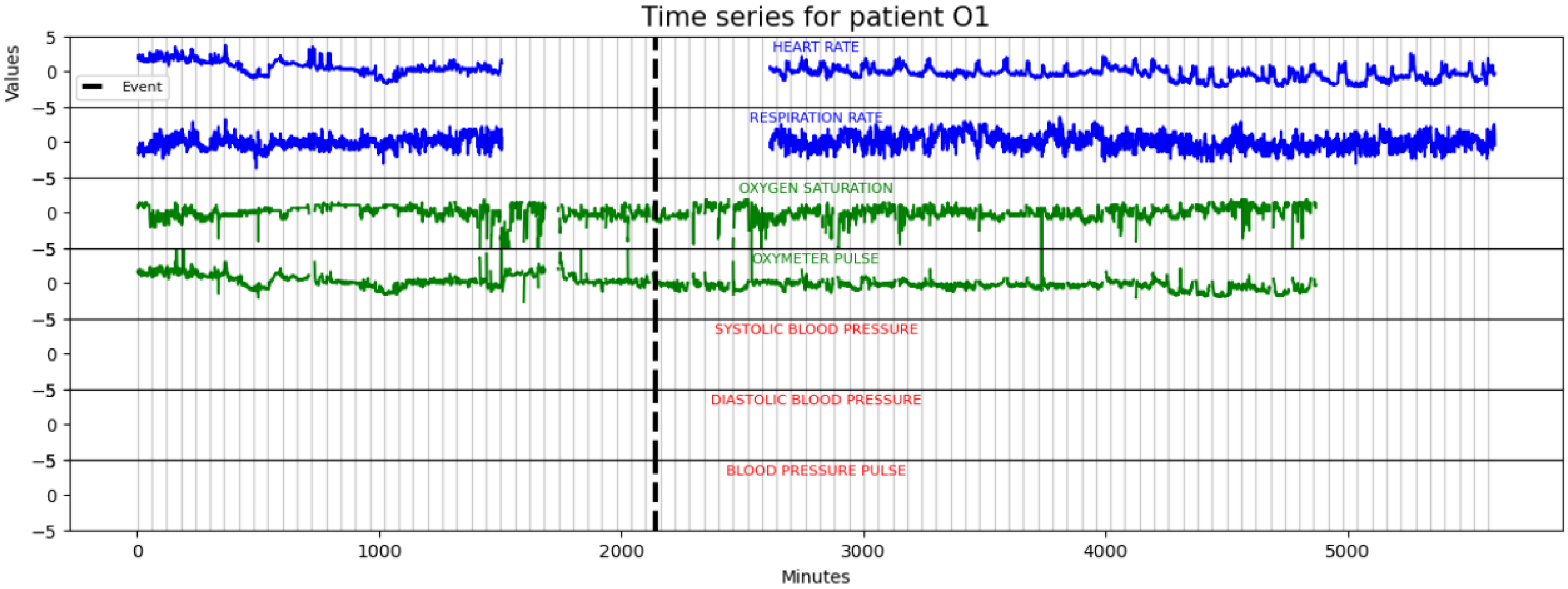
- This is (as we understood it) the vision of WARD. Norman is writing a PHD working on this, and provided the dataset shown in the previous slide.

- Our vision regarding the final project was less ambitious and began with plotting the data!

Initial look

- Patients: 1.393
- Events: 721
- Alerts : 39.311

- Patients with event: 452
- Patients with alerts: 1.307
- Minutes: 6.379.135
- Total measurements: 51.033.080
- NaNs: 32.890.617 (64 %)

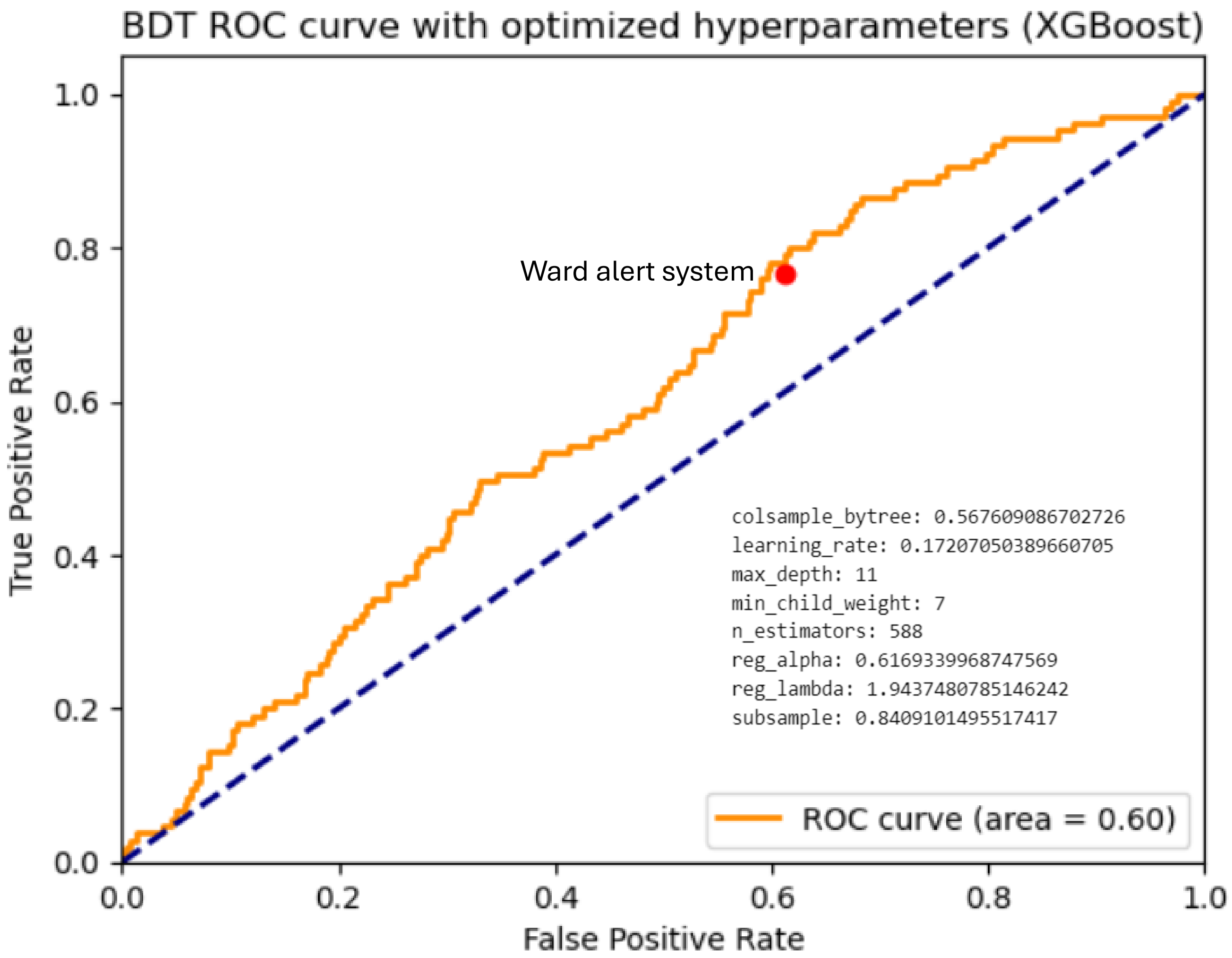


How do we account for time dependency in our models?

- Linear Regression
- Split each interval into 15-minute segments
- Perform linear regression on each segment, and store maximum slope and value as the data for the given 8-hour interval
- Gives some time dependency, but is it representative?
- This leaves us with 16 values for each 8-hour interval, and now we are ready to train a ML model!

XGBoost – the first model

- Trained on all 8-hour intervals, regardless of quality
- No regards for different event groups
- HP optimization: Randomized Search with 20 steps



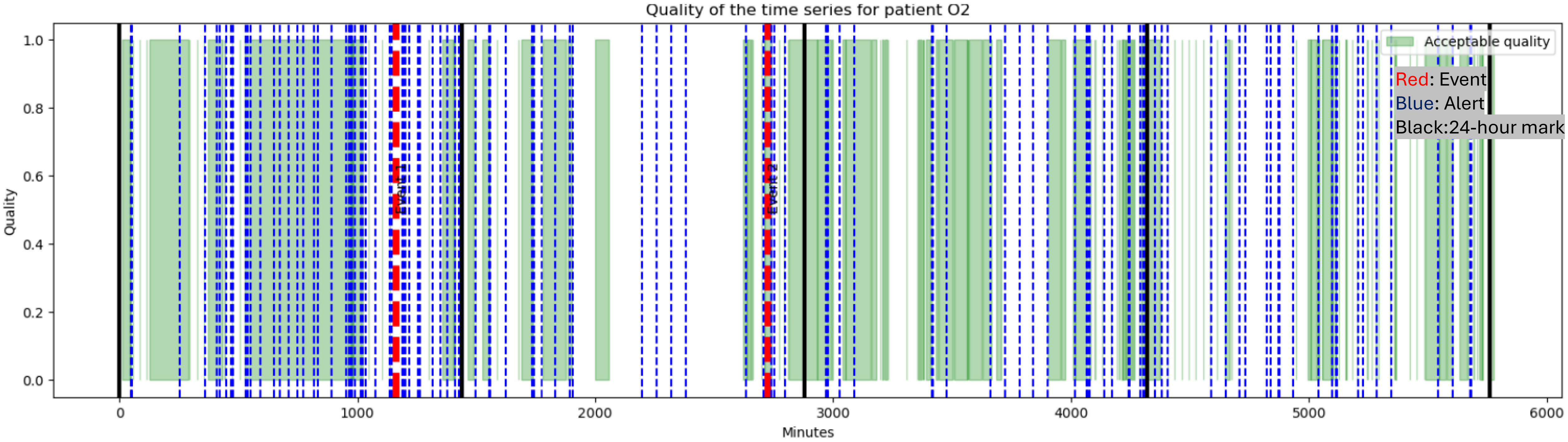
```
colsample_bytree: 0.567609086702726
learning_rate: 0.17207050389660705
max_depth: 11
min_child_weight: 7
n_estimators: 588
reg_alpha: 0.6169339968747569
reg_lambda: 1.9437480785146242
subsample: 0.8409101495517417
```

Sorting data

- Quality control
- 8-hour intervals with acceptable quality
- In the ones containing events,
  they are positioned at 7.5 h prior – 0.5 h after
- NaN value handling

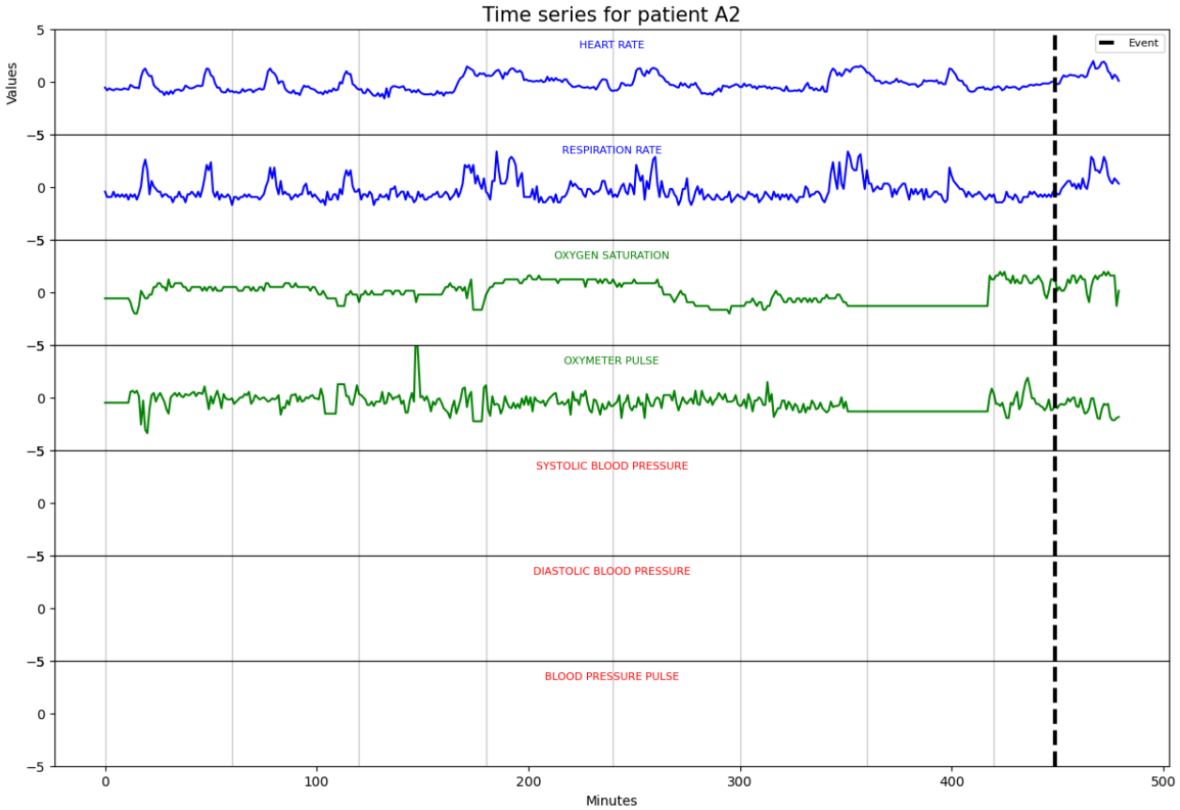
Grouped quality intervals with event:

- Group 1: 9
- Group 2: 59
- Group 3: 6
- Group 4: 14
- Group 5: 105



Data Augmentation

- Looking at event groups individually
- This is very low statistics - how do we fix it?
  - Augmented patients!
- Mix and match the biometric data for intervals with the same event group
- Every combination of 2 patients becomes 6 augmented patients

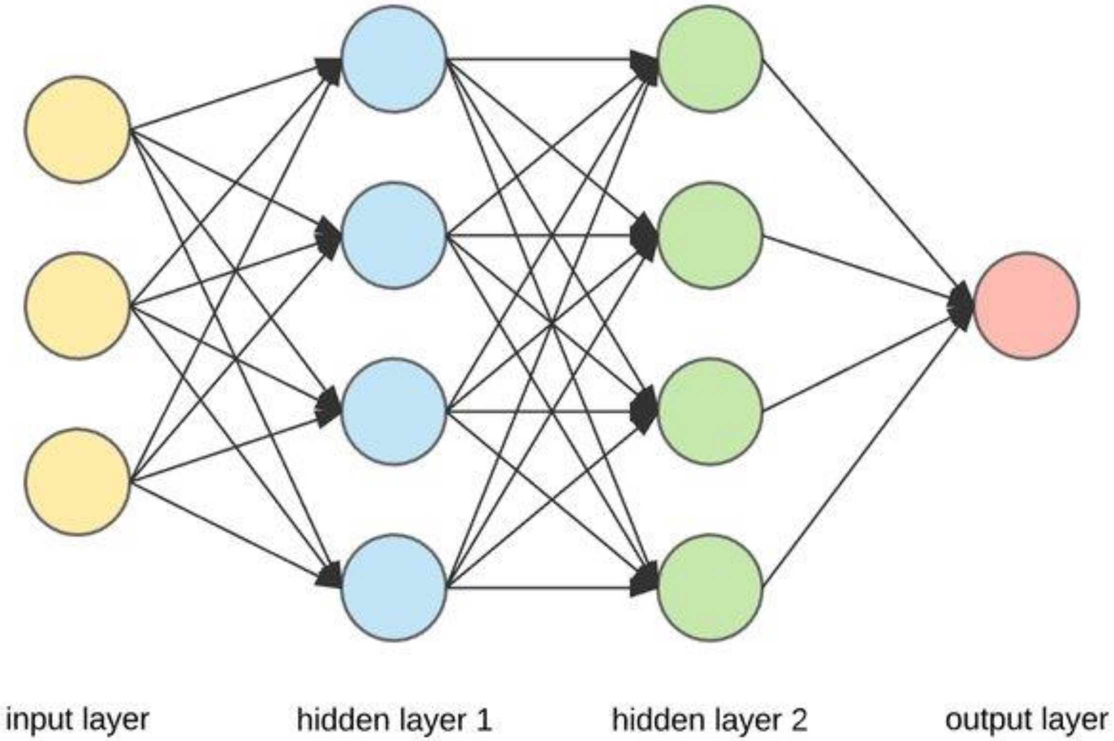


XGBoost for individual groups

- Augmented patients
- Loss function: LogLoss
- Hyperparameters
    - n_estimators
    - max_depth
    - reg_alpha
    - reg_lambda
    - scale (pos_weight)
- Bayesian Optimization with AUC criterion

Pytorch NN for individual groups

- Augmented patients
- Simple NN with 2 hidden layers
- Hyperparameters:
  - batch_size
  - hidden_size_1
  - hidden_size_2
  - learning_rate
  - scale (pos_weight)
- Loss: Binary Cross Entropy
- Bayesian Optimization with AUC criterion



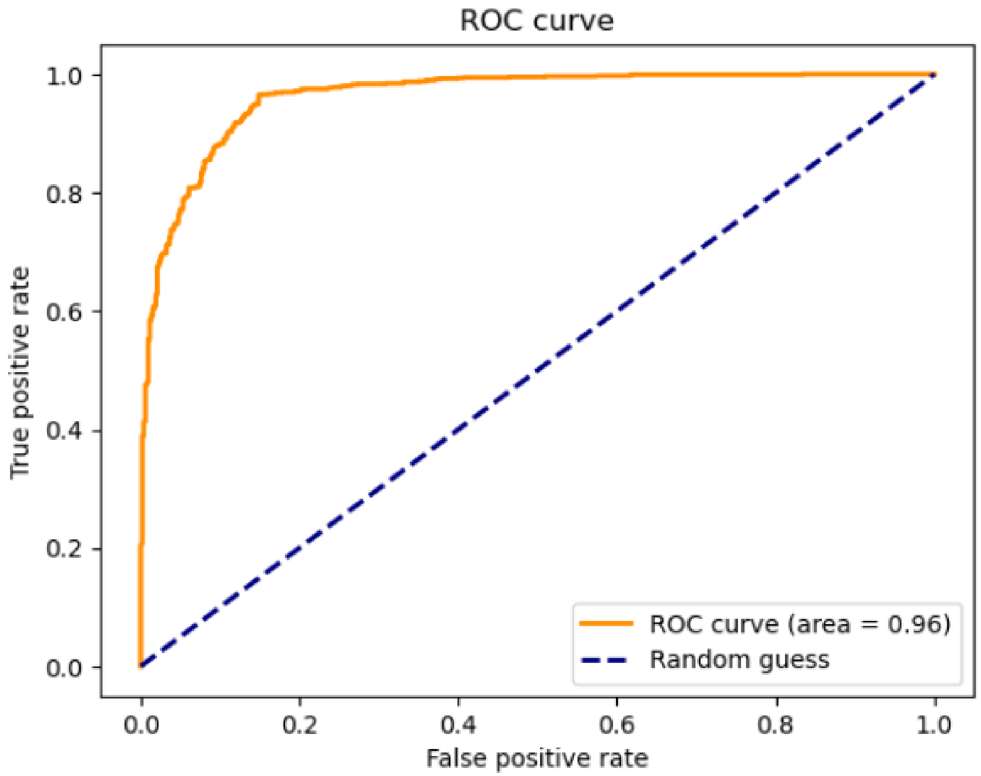
Correlation between test and training data

- Training on augmented patients
- Testing on original patients
- Wow! That's great... Something's probably wrong

- Correlation

- Solution: Separate original patients before augmenting

XGB

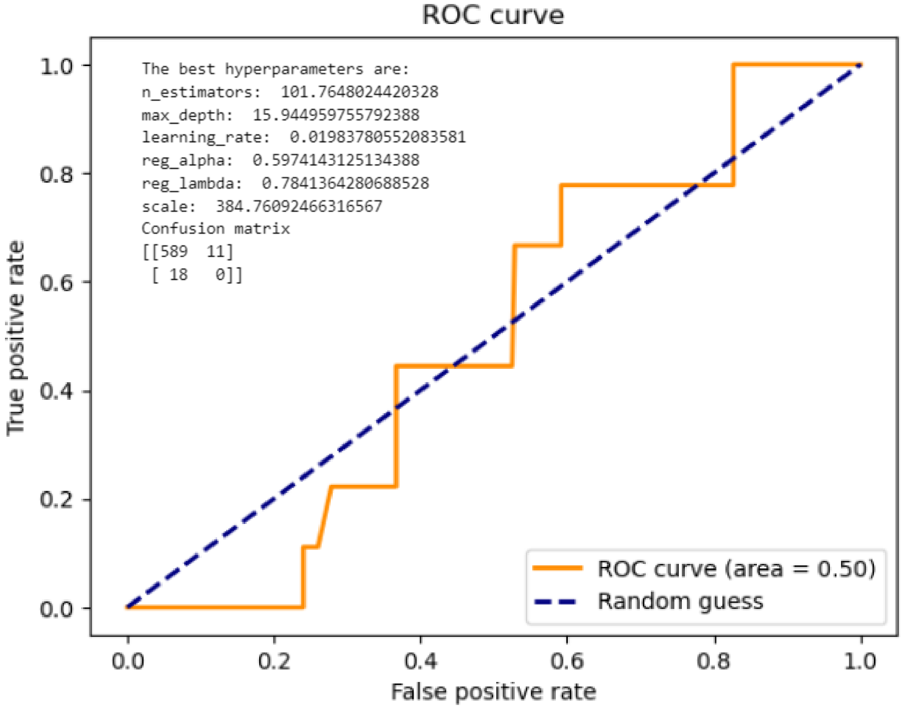


Results

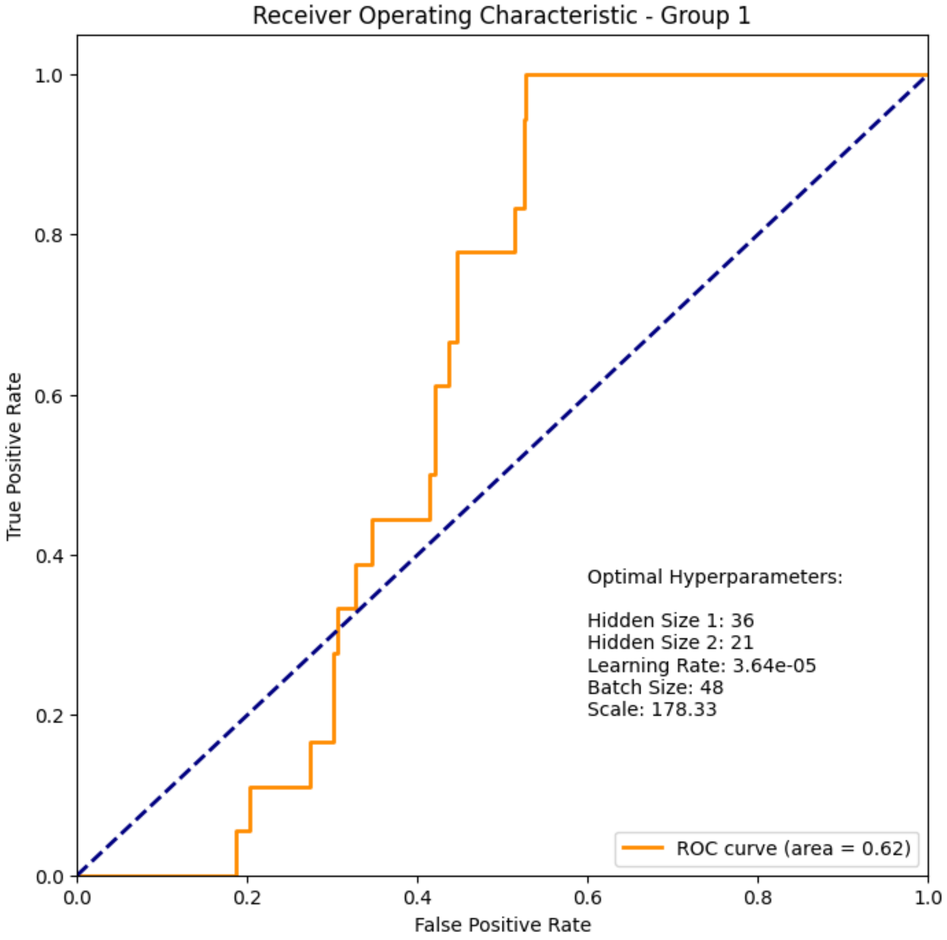
Group 1 – 108 events

NN (auc = 0.62)

XGBoost (auc = 0.50)



```
The best hyperparameters are:
n_estimators:  101.7648024420328
max_depth:  15.944959755792388
learning_rate:  0.01983780552083581
reg_alpha:  0.5974143125134388
reg_lambda:  0.7841364280688528
scale:  384.76092466316567
Confusion matrix
[[589  11]
 [ 18   0]]
```

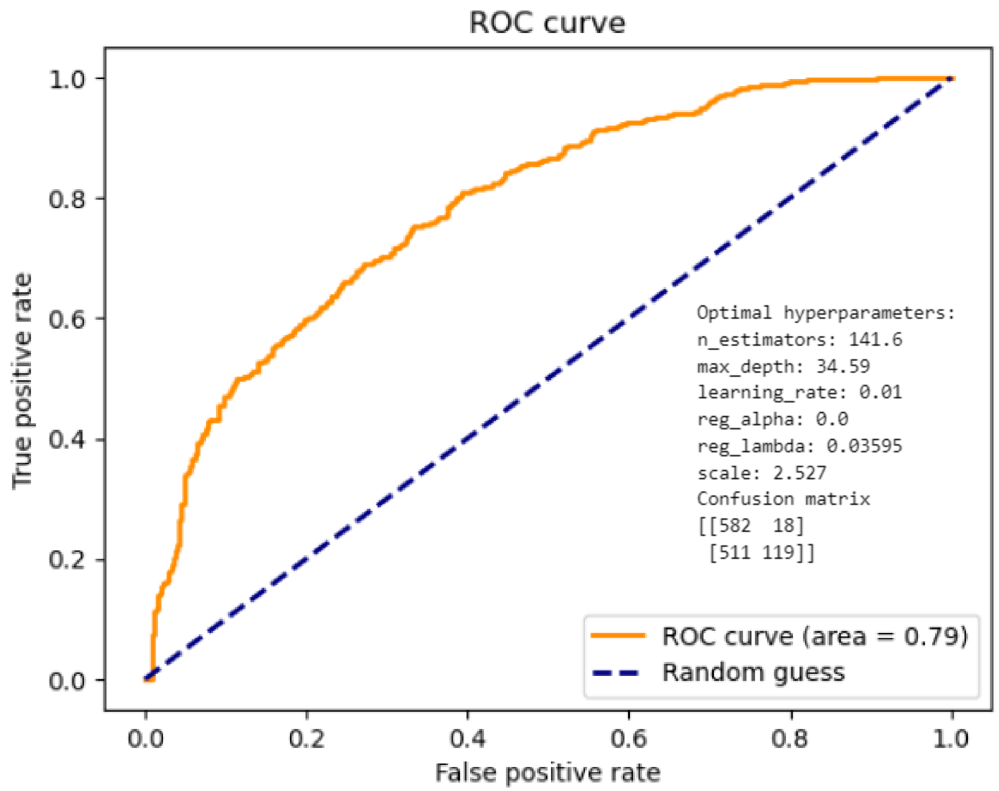


Optimal Hyperparameters:

Hidden Size 1: 36
Hidden Size 2: 21
Learning Rate: 3.64e-05
Batch Size: 48
Scale: 178.33

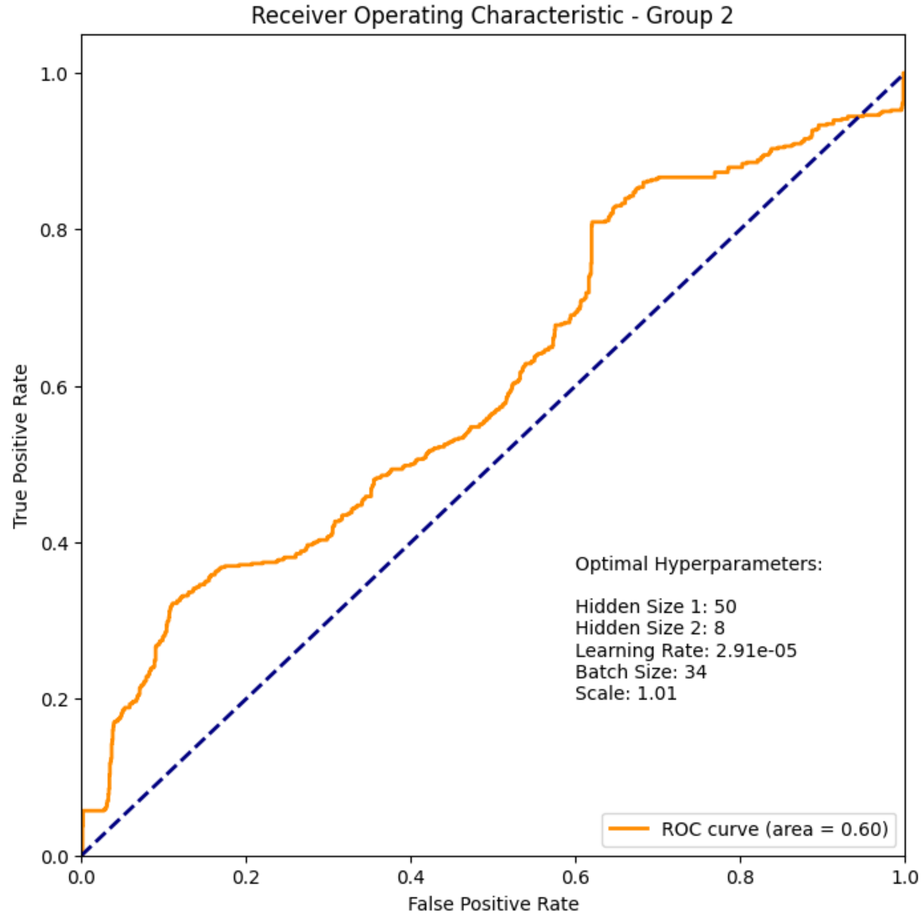
Group 2 - 6306 events

XGBoost (auc = 0.79)



```
Optimal hyperparameters:
n_estimators: 141.6
max_depth: 34.59
learning_rate: 0.01
reg_alpha: 0.0
reg_lambda: 0.03595
scale: 2.527
Confusion matrix
[[582  18]
 [511 119]]
```

NN (auc = 0.60)

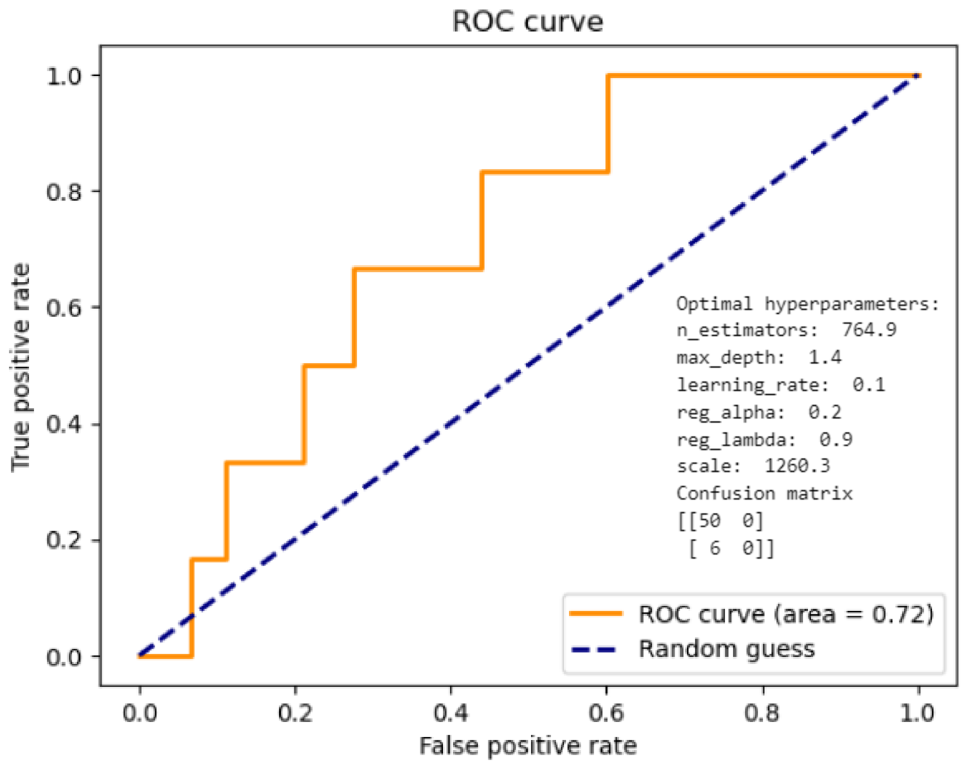


```
Optimal Hyperparameters:

Hidden Size 1: 50
Hidden Size 2: 8
Learning Rate: 2.91e-05
Batch Size: 34
Scale: 1.01
```

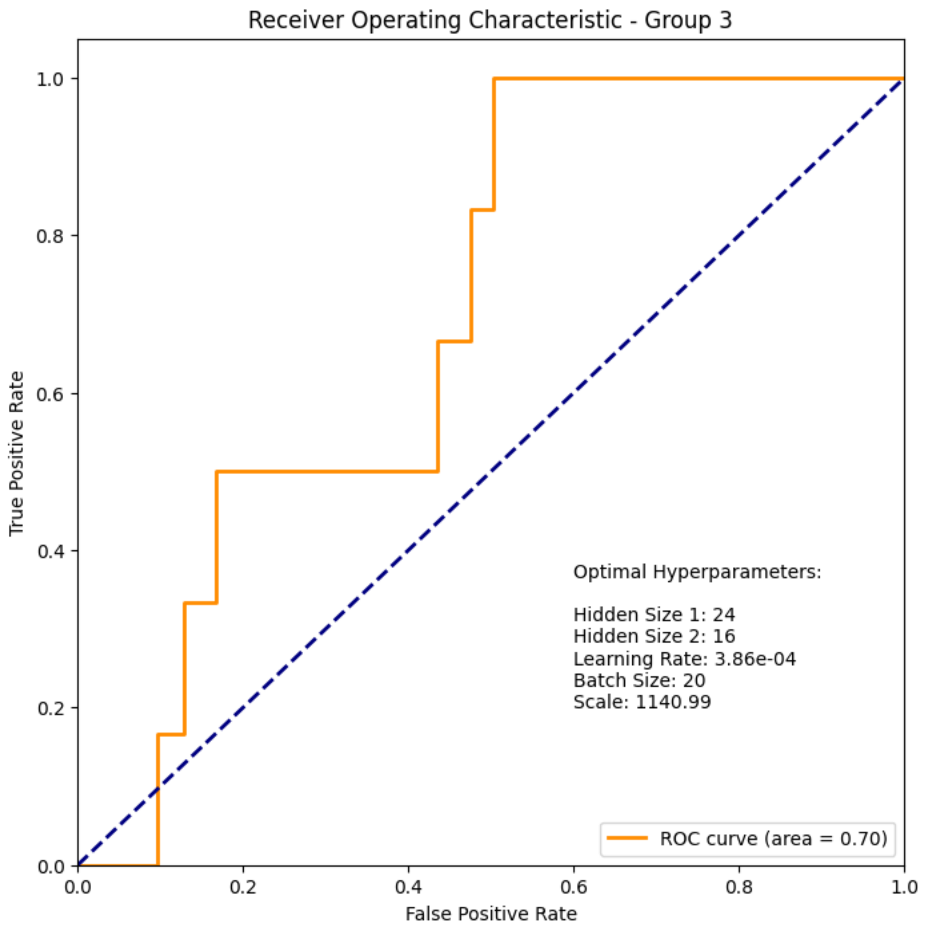
Group 3 – 42 events

XGBoost (auc = 0.72)



```
Optimal hyperparameters:
n_estimators:  764.9
max_depth:  1.4
learning_rate:  0.1
reg_alpha:  0.2
reg_lambda:  0.9
scale:  1260.3
Confusion matrix
[[50  0]
 [ 6  0]]
```

NN (auc = 0.70)

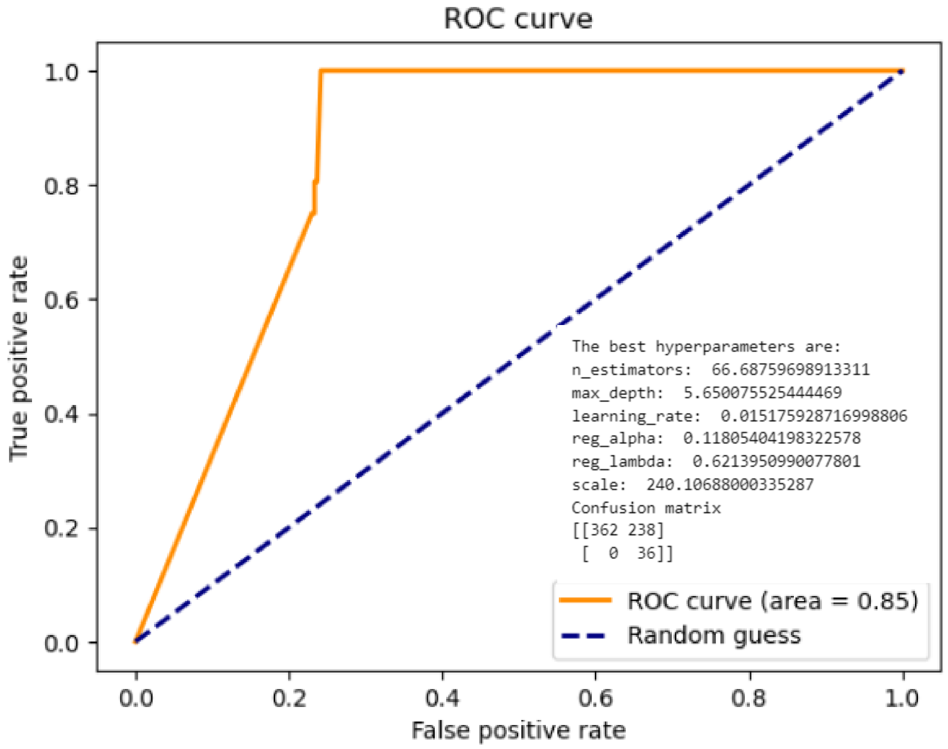


Optimal Hyperparameters:

Hidden Size 1: 24
Hidden Size 2: 16
Learning Rate: 3.86e-04
Batch Size: 20
Scale: 1140.99

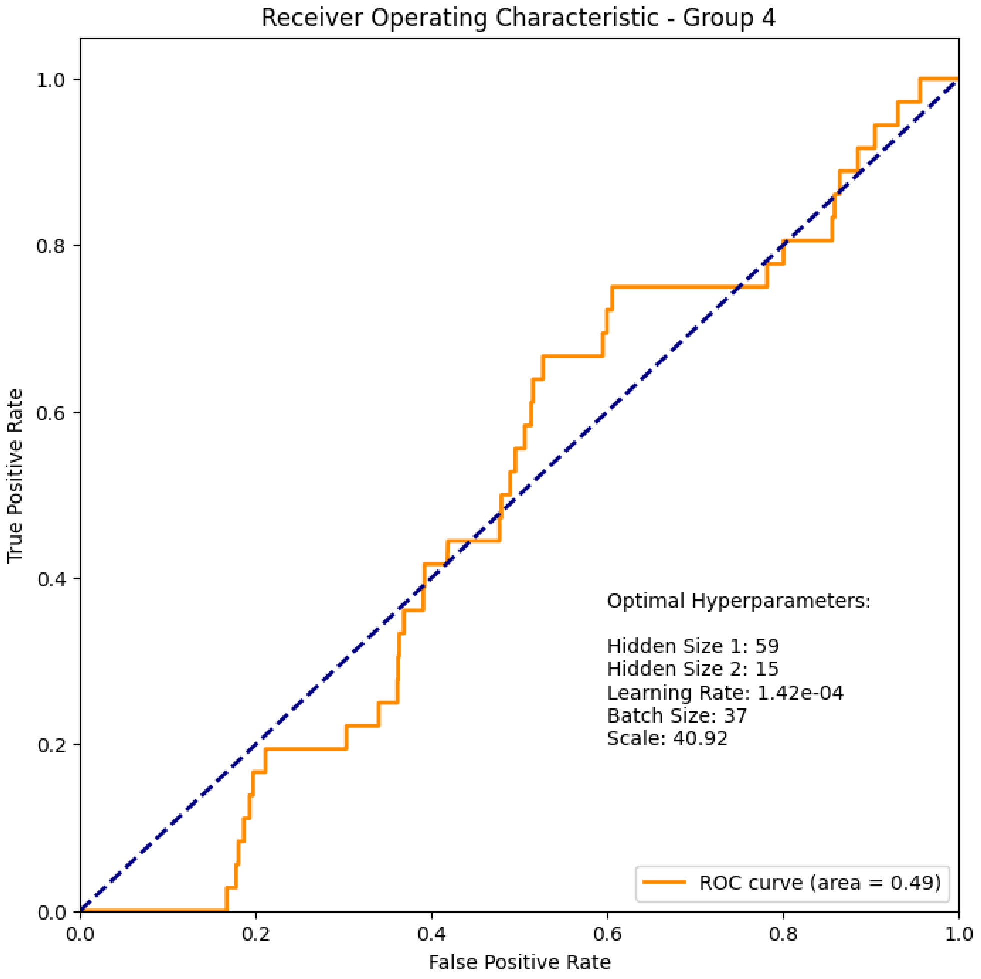
Group 4 – 306 events

XGBoost (auc = 0.85)



```
The best hyperparameters are:
n_estimators:  66.68759698913311
max_depth:  5.650075525444469
learning_rate:  0.015175928716998806
reg_alpha:  0.11805404198322578
reg_lambda:  0.6213950990077801
scale:  240.10688000335287
Confusion matrix
[[362 238]
 [  0  36]]
```

NN (auc = 0.49)

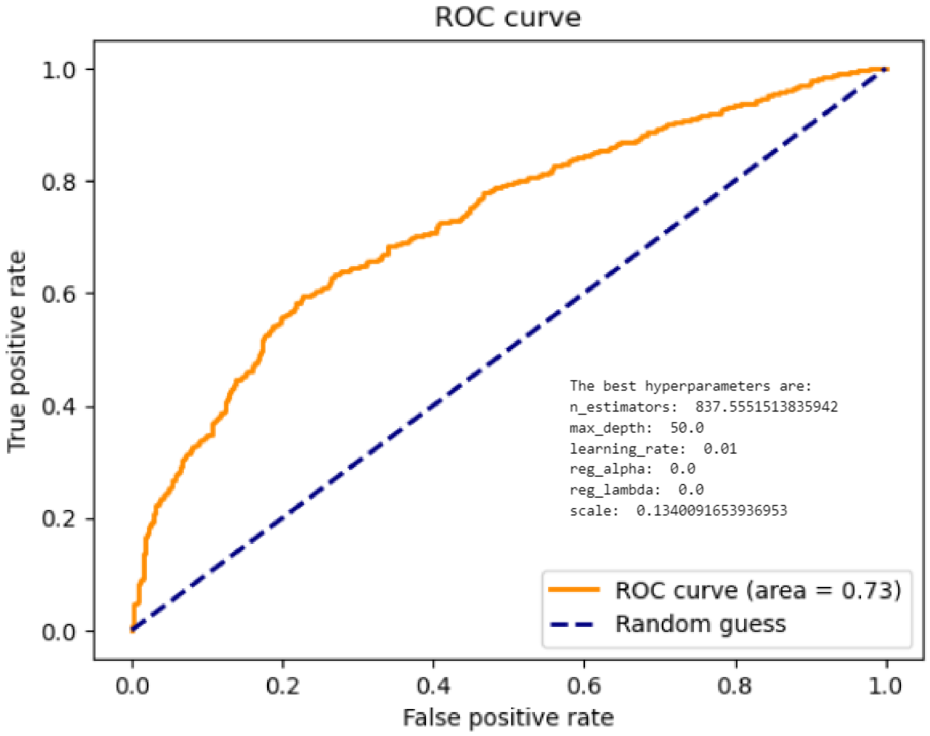


Optimal Hyperparameters:

Hidden Size 1: 59
Hidden Size 2: 15
Learning Rate: 1.42e-04
Batch Size: 37
Scale: 40.92

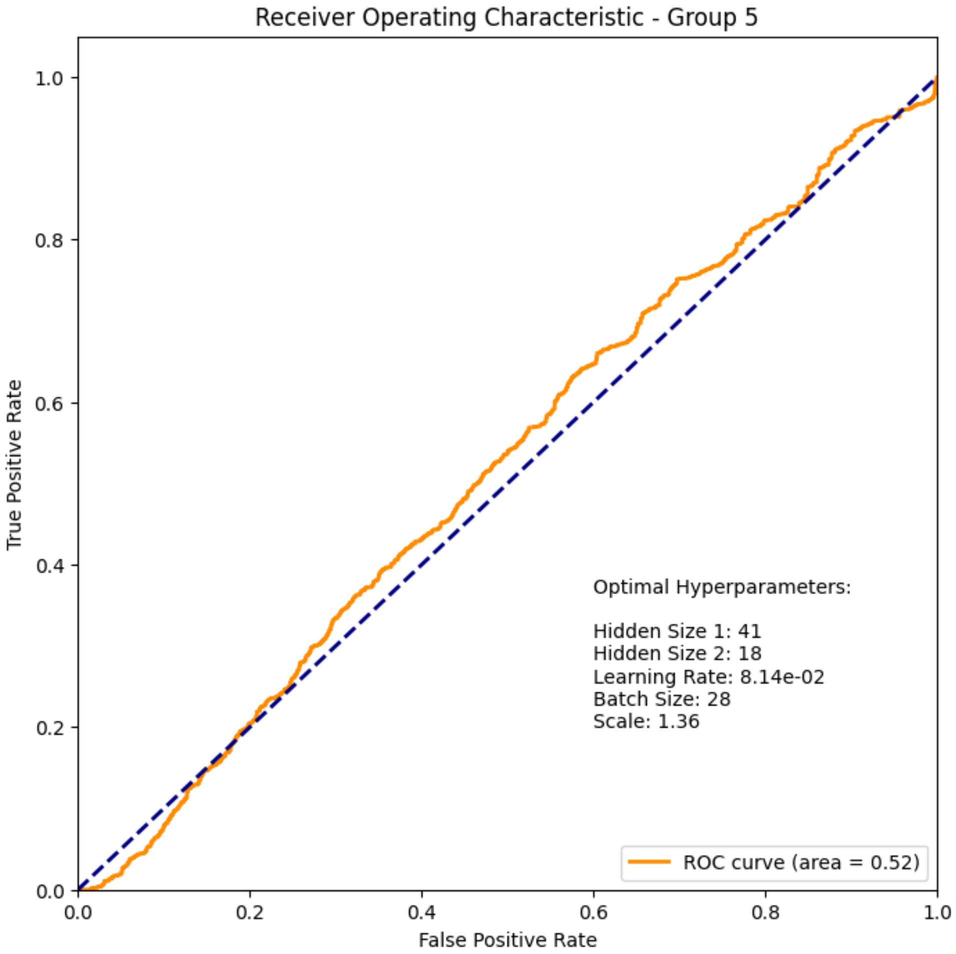
Group 5 – 20124 events

XGBoost (auc = 0.73)



```
The best hyperparameters are:
n_estimators:  837.5551513835942
max_depth:  50.0
learning_rate:  0.01
reg_alpha:  0.0
reg_lambda:  0.0
scale:  0.1340091653936953
```

NN (auc = 0.52)



Optimal Hyperparameters:

Hidden Size 1: 41
Hidden Size 2: 18
Learning Rate: 8.14e-02
Batch Size: 28
Scale: 1.36

Conclusion and future work

- Real world data can be finicky, and hard to format. Many approaches.
- XGBoost is consistently better than the NN, with group 1 as the exception.
- We improved on the Ward point.

- More sophisticated models (RNN, Echo State Networks, etc.)
- More sophisticated evaluation (not AUC, e.g. zero false negatives)

Appendix

Working on the WARD dataset has been very interesting, and working on real world data has been inspiring to us both. We don't have much for the appendix, and think we included most of our learnings in the presentation.

The project ended up mostly focused on the steps to take before running an actual ML model. Due to the complicated nature of the dataset, we wanted to fully understand it, and set up a system where we knew the augmentation inside out, and everything was properly verified before running more complicated models.

Then for each patient, we every event and store the interval from 7.5 hours prior to 0.5 hours after the event. Then we sort the intervals into 5 groups based on the event group. If 2 or more probes are active, we denote the quality of that measurement as good. If at least 80% of the measurements in the interval are good, we store the interval for future use. We remove the NaN values of each interval by forward filling and backwards filling. The we split the intervals on a group basis into training and testing sets. Performing the augmentation consisted of mixing every combination of probes for every pair of patients. This could have been done more thoroughly by utilizing the measurements in each probe. This would lead to 254 augmented patients per pair, instead of 6. Augmenting like this could be a good way of ensuring proper statistics for groups 1, 3 and 4.

We would have liked to make an RNN, but unfortunately, we didn't have the time. We would also have liked to understand the NN results better, because it seems much worse than the BDTs.