WARD data

Christine, Marie, Vilma

Questions

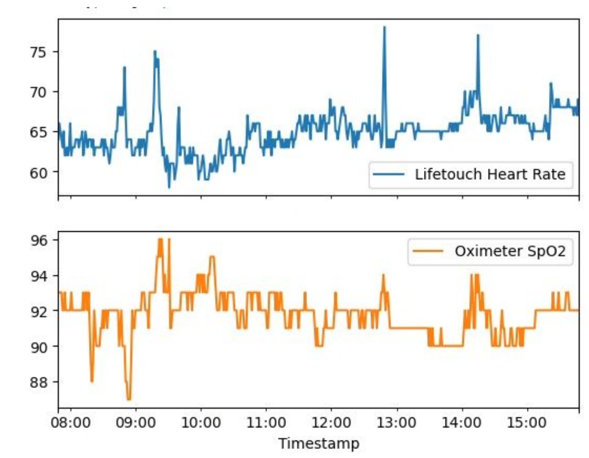
1. Can we predict significant adverse events (SAEs)?
2. Can we tell when patients are sleeping?

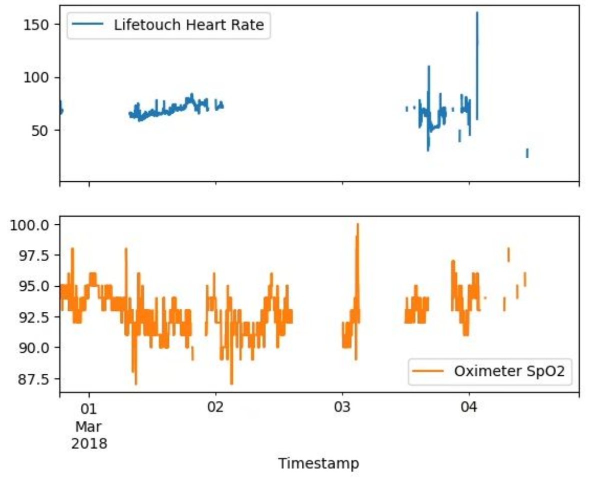
The data

- Measurements from hospital patients every minute over 4 days
- 44 values from 3 devices (Lifetouch, Oximeter, blood pressure)
- Includes e.g. derivatives, slopes & data ‘quality’ values A & B

Possible ways of dealing with the holes:

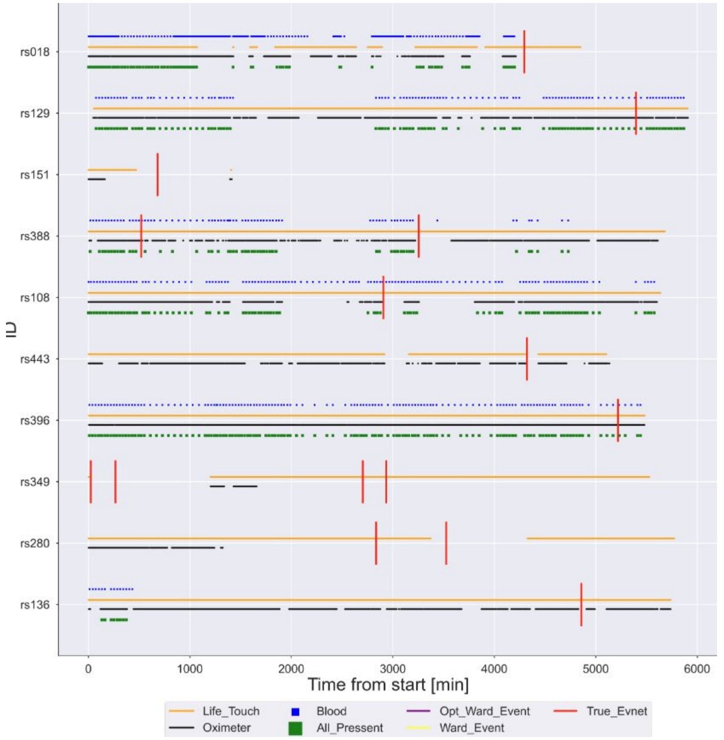
- Discarding data
- Intenterpolating between existing values
- Fourier transform
- Averaging over the data





The data: # of SAEs and their accuracy

- Not a lot of data
- 800 patients
- SAE events - 101
    - precision of event are not necessarily accurate
    - Some SAEs at the start of data, needed to discard those: ended up with 96 SAEs



The data: sae groups & overlap between them

	1: Neurologic	2: Repiratory	3: Circulation	4: Infectious	5: Other
Cases	5	12	17	15	39

SAEs in 5 different groups (Group numbers?) - biggest group ha all the ‘other’-specified SAEs

We chose to look at them collectively cause need data!

Choosing the data

- We chose 7 variables and their slopes (=14 variable model) by domain knowledge
- Good results: detecting SAE 8 h before to 2 h after event

SAE data:

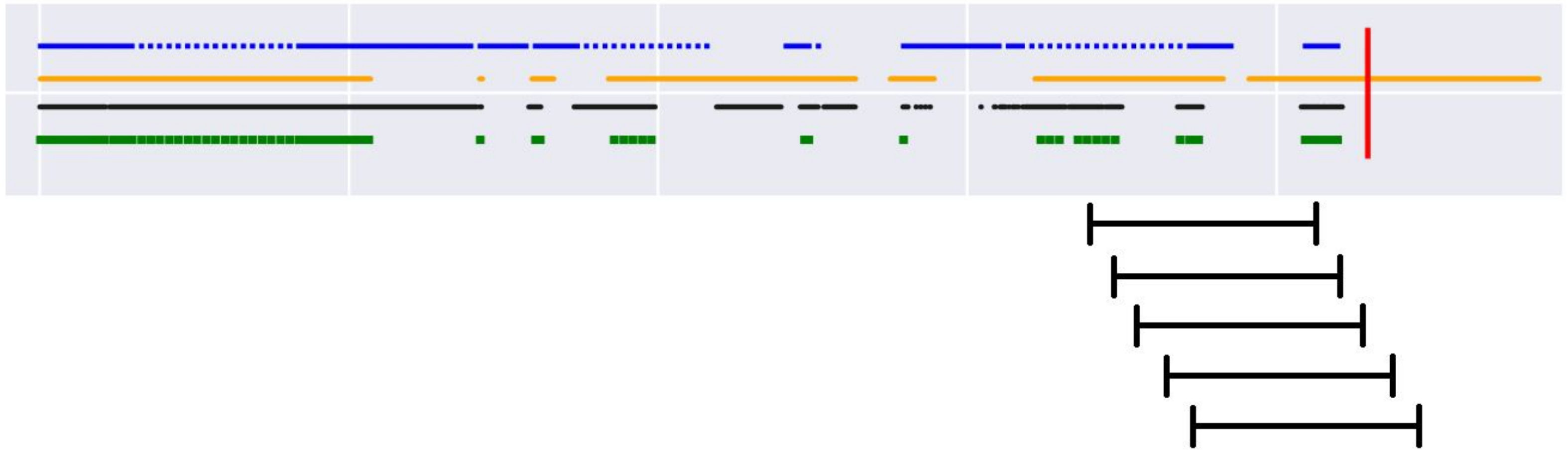
- We chose to look at 4 h leading up to and SAE
- Discarding data that’s less than 4 h from the start of measuring: 96 SAEs

non-SAE data:

- Iterating over the patients with no SAEs with continuous A & B for 4 h straight (Very slow!)
- problem: all the non-SAE data is continuous A&B while SAE data has missing values (we will ignore this for now)

Making more data

- SAEs are marked in the data **manually after the fact**
- actual position of sae not accurate, we can pretend that the SAE is anywhere +/- 1h from where it is marked in the data
- Now we can multiply the data by slightly offsetting each new sample



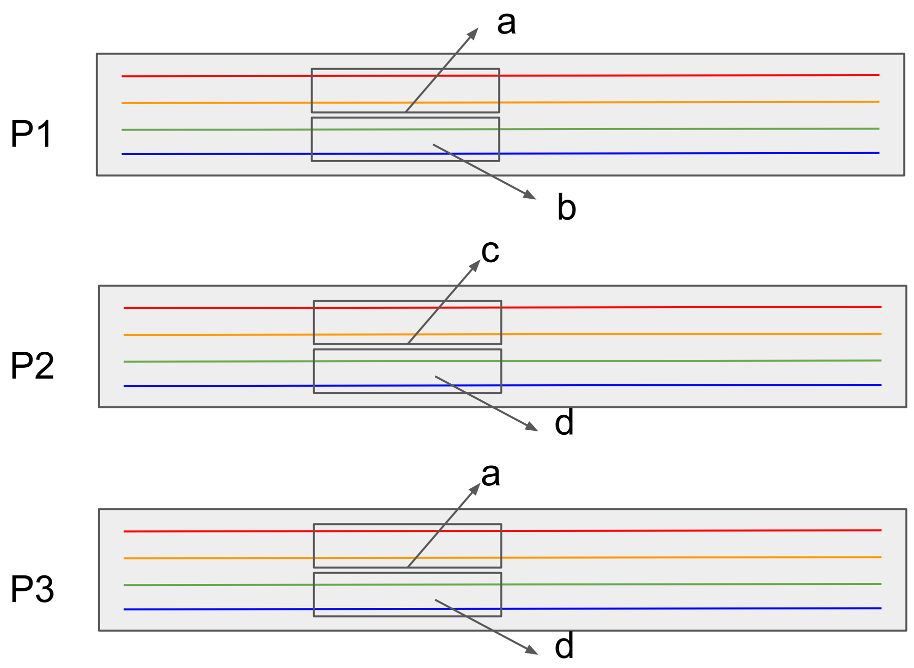
Making synthetic data

Timeframes need to be the same or discarded

Should be normalized but normalizing sucks! :(

Future work

- using ML to see if it could tell the real data from synth



ab - real data
cd - real data
cb - synthetic data
ad - synthetic data

Linear Discriminant Analysis (LDA) Marie

What is LDA

Data used

sklearn

- model classification
- fit
- predict

results

- imbalance in data?

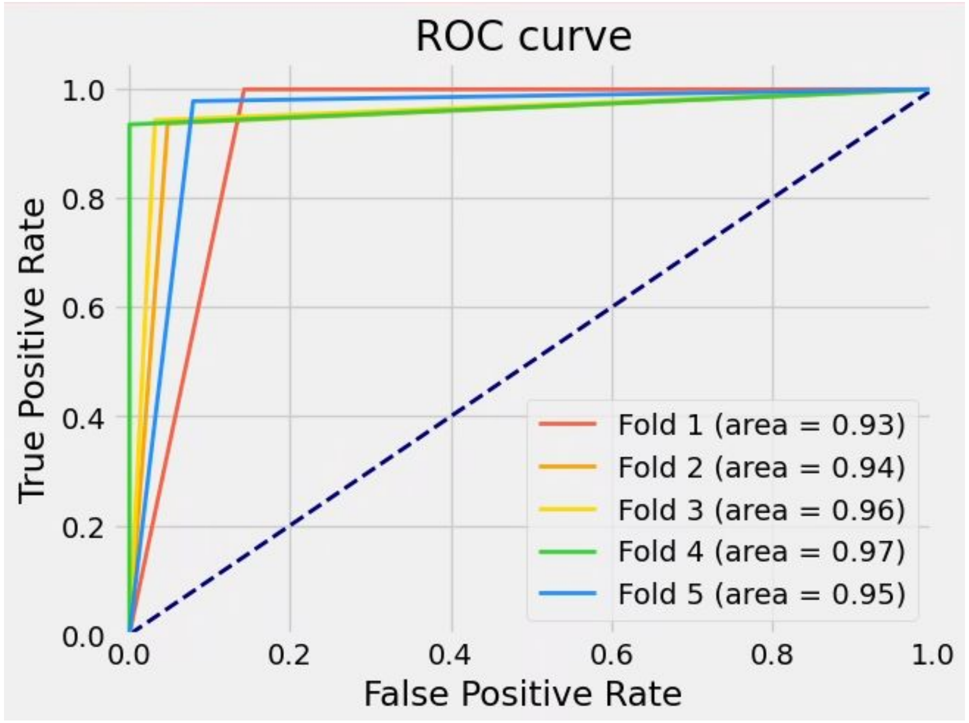
```
Accuracy: 0.9993597951344431
Precision: 0.0
Recall: 0.0
F1-score: 0.0
```

XGBoost analysis

- 14-parameter model used (averaged over 4 h)
- Nan data filled with mean
- Possible to extend model by averaging over shorter timeframes

- XGBoostClassifier - time interval leading to SAE or not
- ~300 SAEs, 350 non-SAEs
- 5-fold cross validation (we wrote our own method for this)
- Hyperparameter optimization using hyperopt
- Testing on 90/10 train/test split

XGBoost analysis



	Accuracy
crossval average:	95 %
90/10 split:	90 %

Conclusions

- Really good results but we don’t know why
- It could be that the results are genuine, but we doubt it

Questions

1. Can we predict significant adverse events (SAEs)?
2. Can we tell when patients are sleeping?

Challenges of the question

- Baseline determination
- Non-existent seasonality aka sleeping schedule
- Different people
- No labels → Clustering

Data used

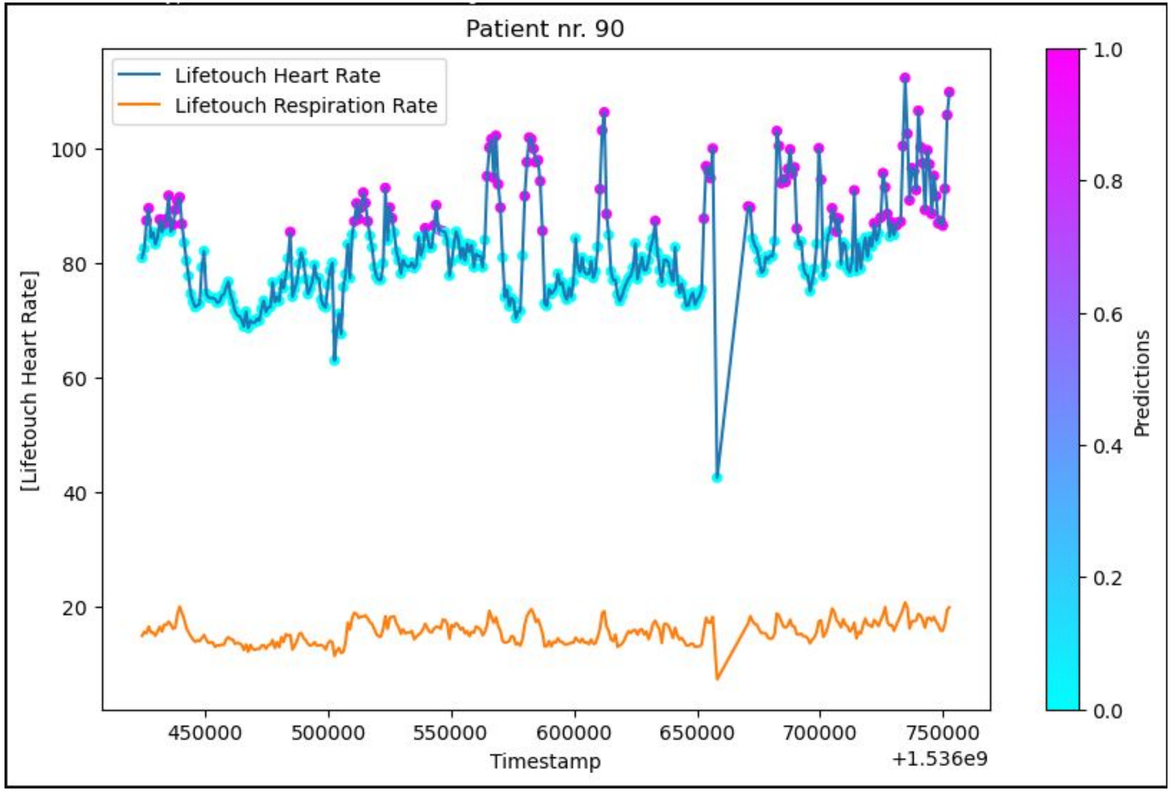
- Data Amount: Using only Lifetouch Heart Rate and Lifetouch Breathing Rate
- Netting 215312 rows after removing NaNs, before: 756670 rows
- Based on 799 patients
- Averaged over 15 minute period
- Means and standard deviations pr. patient split by ID
- Features: **ID**, Timestamp, Lifetouch Heart Rate, Lifetouch Breathing Rate
- Calculated Features: **Lifetouch Heart Rate - mean(Lifetouch Heart Rate), Lifetouch Breathing Rate - mean(Lifetouch Breathing Rate), std(Lifetouch Heart Rate), std(Lifetouch Breathing Rate)**

Different Baselines

- First approach: Ignore it
- Second Approach: Means and Standard Deviations

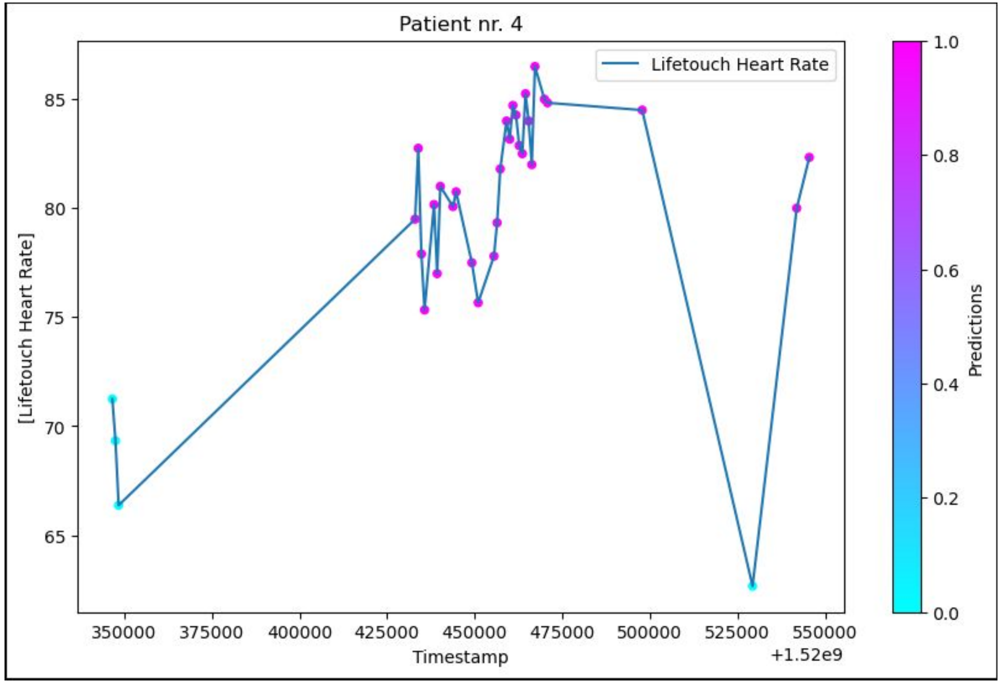
Non-existent seasonality

- Fitting on all patients vs. Fitting on individual patients
- Individual patient max data: 4 * 24 * 60 = 5760 data points
- After 15 min averaging : 1440 data points
- 1 = awake, 0 = Sleeping



Attempted Models and their failures

- KMeans
  - Issue #1: No implementation for within time series variations
  - Issue #2: No null values… well #%& ¤
  - Issue #3: Fitting on all patients raw values netting a flat line
- HDBSCAN
  - Issue #1: Still no null values…
  - Issue #2: Everything marked as noise



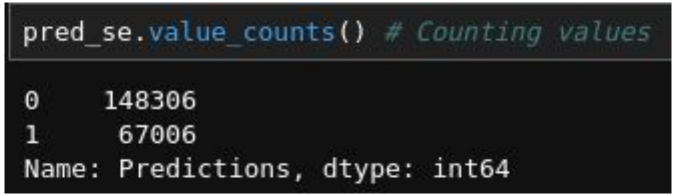
1 = Awake, 0 = Asleep… maybe?!

Kmeans - final examples and some statistics

KMeans(n_clusters=2,verbose=1)

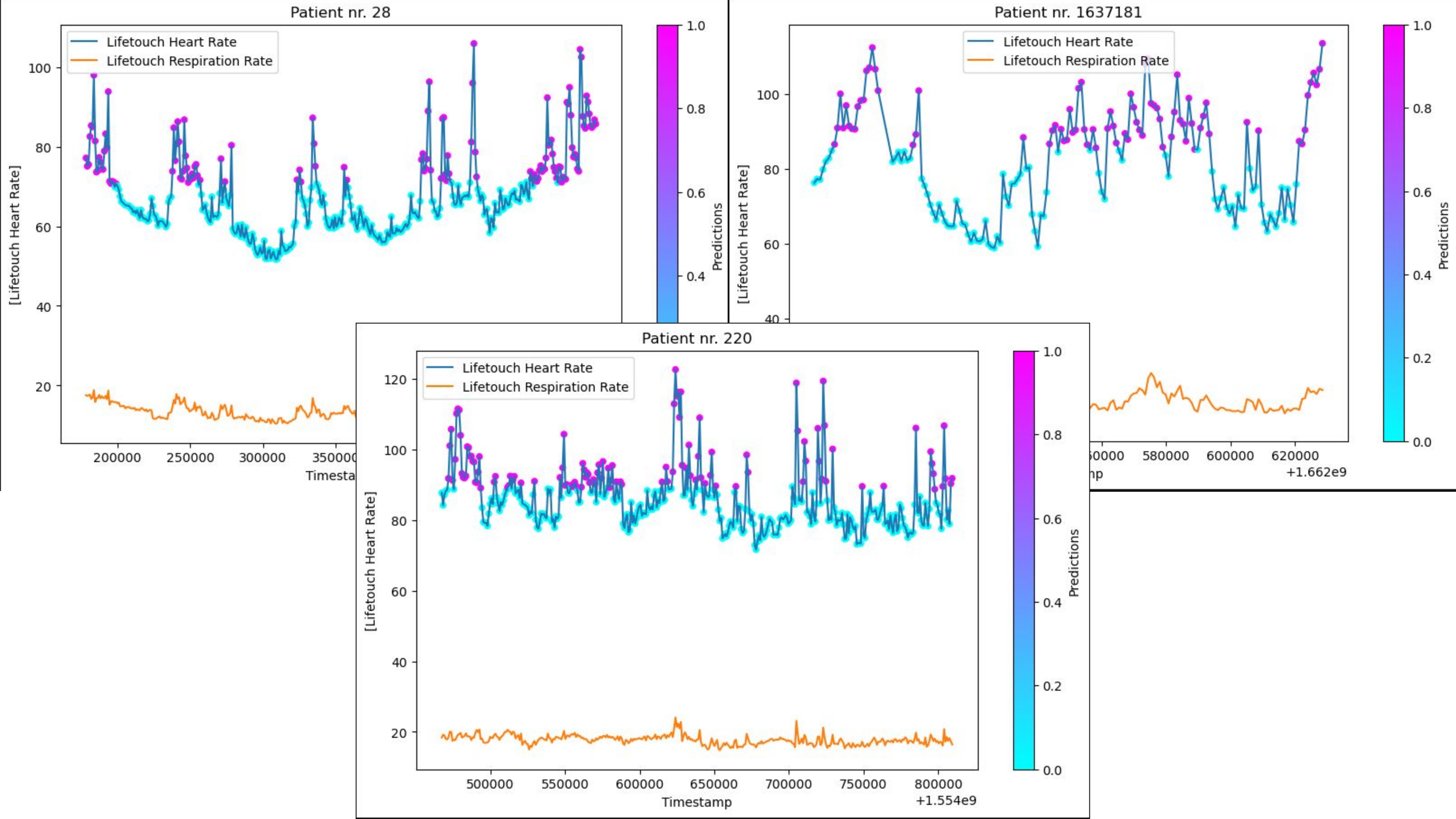
1 = awake, 0 = asleep

Reverse relation? Somewhat poor model? Not the most optimal data?



```
pred_se.value_counts() # Counting values

0    148306
1     67006
Name: Predictions, dtype: int64
```


Patient nr. 28
Lifetouch Heart Rate
Lifetouch Respiration Rate
[Lifetouch Heart Rate]
Timestamp
Predictions
Patient nr. 1637181
Lifetouch Heart Rate
Lifetouch Respiration Rate
[Lifetouch Heart Rate]
+1.662e9
Predictions
Patient nr. 220
Lifetouch Heart Rate
Lifetouch Respiration Rate
[Lifetouch Heart Rate]
Timestamp
+1.554e9
Predictions

In Conclusion…

- We can somewhat predict whether patients are awake or not
- We lack data to fully predict whether patients are awake or not, including “normal” data
- Other factors play a significant role, such as eating times, talking, medication and pain etc.

Thanks!

Appendix

- We were really careful with having the whole offset group from a single SAE in the same set (eg. train/validation/test)
- We wrote our own method for splitting the data for cross validation
- Splitting into sets was done by patient id (randomly selected) - thus, the sizes of the sets vary some but there's no bias in the validation data
- First 10 % of the data was set for testing
- The remaining 90% was split into five groups, which were used for cross validation
- The SAEs and non-SAE were sampled separately for each set (which were then shuffled, although this should not make a difference), so that we have ~equal amount of both in all the sets

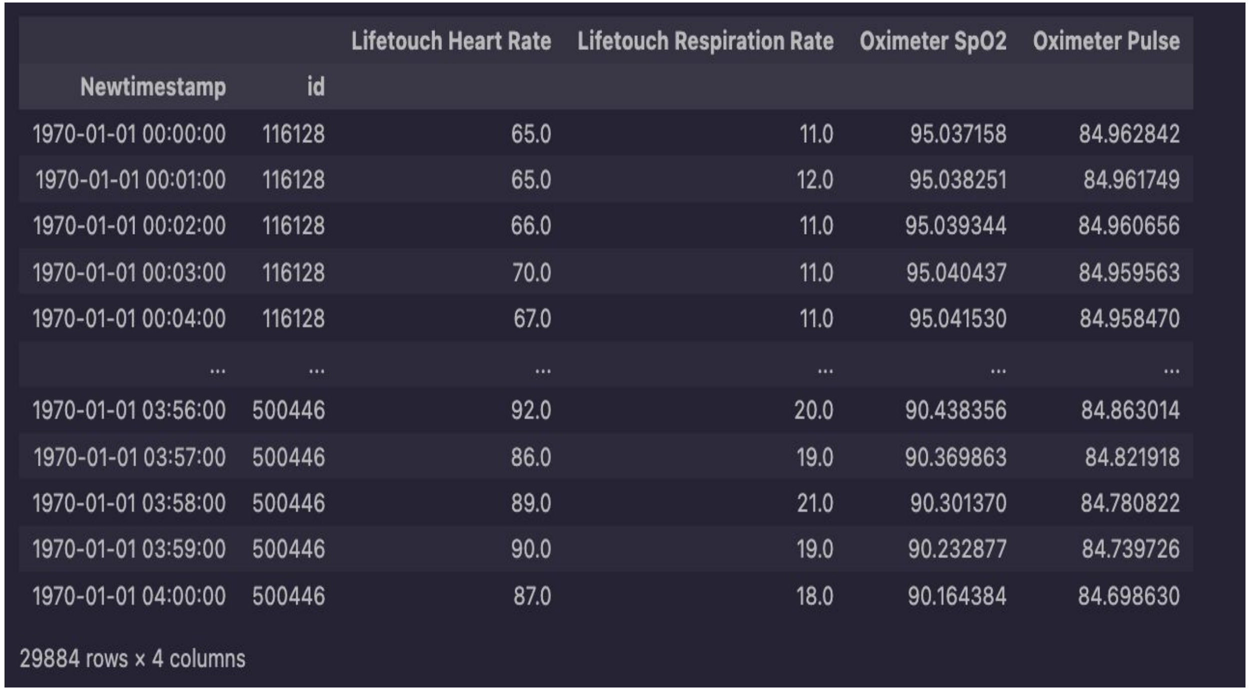
Timedata

Resampling the data to chunks of 15 minutes and 4 hours.

Combined data - Example

Combining data of first four values of two different patients. The ids are the combination of the patients

problems then with how to normalize

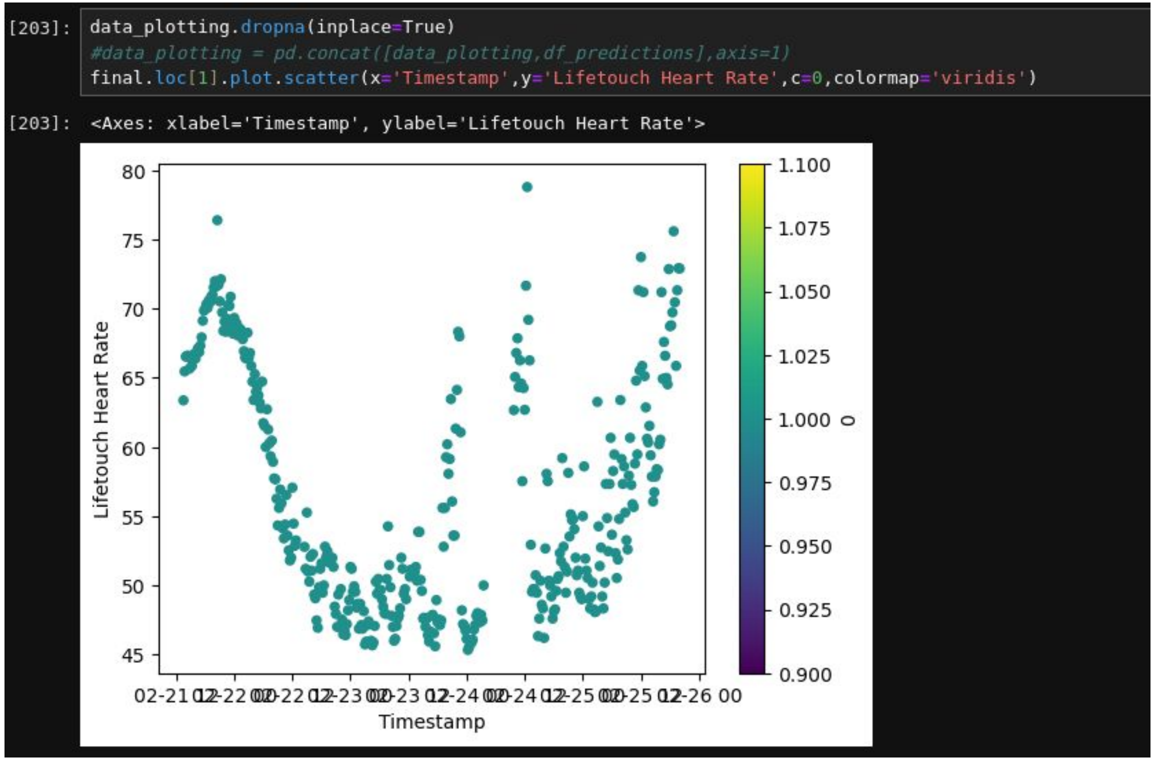


		Lifetouch Heart Rate	Lifetouch Respiration Rate	Oximeter SpO2	Oximeter Pulse
Newtimestamp	id				
1970-01-01 00:00:00	116128	65.0	11.0	95.037158	84.962842
1970-01-01 00:01:00	116128	65.0	12.0	95.038251	84.961749
1970-01-01 00:02:00	116128	66.0	11.0	95.039344	84.960656
1970-01-01 00:03:00	116128	70.0	11.0	95.040437	84.959563
1970-01-01 00:04:00	116128	67.0	11.0	95.041530	84.958470
...	...	...	...	...	...
1970-01-01 03:56:00	500446	92.0	20.0	90.438356	84.863014
1970-01-01 03:57:00	500446	86.0	19.0	90.369863	84.821918
1970-01-01 03:58:00	500446	89.0	21.0	90.301370	84.780822
1970-01-01 03:59:00	500446	90.0	19.0	90.232877	84.739726
1970-01-01 04:00:00	500446	87.0	18.0	90.164384	84.698630

29884 rows × 4 columns

Slightly more detailed Explanation for how KMeans failed

- Kmeans
  - 0th iteration: Time series not implemented for kmeans
  - First iteration: Failed due to requiring no null values
  - Second iteration: Now ignoring the holes in the data entirely, straight line model, thus some patients are constantly sleeping, others are never sleeping, 1 = sleeping, 0 = awake. Crude plot of this:



```
[203]: data_plotting.dropna(inplace=True)
       #data_plotting = pd.concat([data_plotting,df_predictions],axis=1)
       final.loc[1].plot.scatter(x='Timestamp',y='Lifetouch Heart Rate',c=0,colormap='viridis')
```

```
[203]: <Axes: xlabel='Timestamp', ylabel='Lifetouch Heart Rate'>
```

Other Considered attempts at establishing sleep pattern and why they were abandoned

- Self-written 2nd order markov-chain method with simple if statements
  - Time consuming to write, lacking idea about which level to put the if statements at
  - Could potentially have been used with the kmeans, but never got to testing that
- Dynamic Time Warping and general time series tools
  - Took a while to figure out that most of these tools work by feeding a slice of a time series and then establishing which cluster that slice belongs to, which was not applicable to what we were looking at at that time. But could be applied to the SAE detection. Problem found was that the implementation was only applicable to a single variable.
- HDBSCAN
  - These are bottom-to-top clustering and both also cluster noise. Given the nature of the data all of it got flagged as noise when this was attempted on the raw data. Another thing considered was agglomerative clustering, namely top-to-bottom clustering, but error of it not being implemented on time series meant it got abandoned. Could potentially work with the standardized data.