AI for Healthcare Data — **SDU Summer School 2025**
Time series analysis for EEG data — Jan-Matthias Braun
Portfolio

Time-Series Analysis for Seizure Detection/Prediction

As a general note, the portfolio should provide a short description and partly a short discussion of all steps of the processing pipeline, reflecting on the lectures. Please refer to the first slide set, but try to not let yourself get carried away when writing, try to target about 8 pages max, but do not press yourself. And also do not forget to contact us when in doubt or not getting on.
Note that the templates are accompanying the lectures and are not the tasks for the portfolio. The notebooks contain a lot of additional ideas on how to look at the data or how to extend the pipeline. Due to time constraints, we unfortunately cannot include all of them into the portfolio, i.e., read the notebooks as suggestion on what to investigate, optional notebooks are not to be included. To be clear, this document contains the authoritative list of things to include for the time series module. The two optional items in this description are really optional, you can gather bonus points. Furthermore, thinking about them might provide additional insight, connecting the curriculum.
Seizure prediction is a bonus task, nonetheless, performing it is actually not that difficult, as it represents a very minimalistic approach which uses the exact prediction codes. And thus, all the relevant information on how to do it can be found in the detection notebook.

Part 3.1: Data Description

- Describe the data in file "01.edf", statistically, its stationarity and autocorrelation.
- Comparing to white noise and random walk, what do the stationarity and the autocorrelation function tell you about the underlying process?
- Please include plots of the data, with annotations, and the autocorrelation function.
- Use plots with accurate units and axis descriptions.

Part 3.2: Data Processing

- Describe possible artefacts, giving an example from the data set (zoomed plot). Discuss shortly: how can these artefacts mess up our features?
- Describe what a band-pass filter does and why we apply it. Describe the intended outcome, e.g., how can a band pass filter modify an eye-movement in the time series? Extend the previous figure with a figure of the band-pass filtered artefact.
- Optional: Try to argue in terms of the frequency spectrum. You can use the plt.specgram(...) function from the fourier notebook to create a spectrogram easily.

Part 3.3: Seizure Detection

- Describe the windows and the features you use.
- Conferring to module 1, evaluate your decoder using the relevant statistics + confusion matrix. To not get into deadline issues, do not perform feature selection and k-fold cross validation. But please think shortly about why both of them would be a good choice for a real investigation, and write down your reasoning.
- Investigate the following options:
  - How does Balancing influence the performance? Do you have an idea why it has this effect? Give a shot at explaining.
  - How does Scaling / PCA (with / without whitening) influence the performance?

  Again, do not do extensive statistics when testing these options. One or two trials should suffice to see the effect—if there.

Part 3.4: Bonus: Seizure Prediction

- Shift your labels so that each feature vector is paired with a label from its future. Pick a shift n in terms of window step sizes which could be relevant for prediction but which is not too ambitious. To implement this shift n, use

  ```
  fvs =  fvs[:-n]
  labels = labels[n:]
  ```

  to perform the shift as detailed in Figure 1.
- Except for this step, the decoder code can be completely reused. Provide the relevant statistics + confusion matrix.
- How did the performance change?
- Fun (and totally optional) extension: If you feel comfortable with python, you can do this in a loop and present a description on how the accuracy changes with an increased shift n. If in doubt, just choose **one** specific time span and use the corresponding shift n.
- This predictor is minimalistic. One could describe it as a detector with time shift. What would be your idea's for creating a more advanced predicting model?

Part 3.5: Conclusions

- Please discuss your results.
- Especially, discuss the value of the accuracy in comparison to sensitivity and specificity.
- Does the balancing affect these measures? And if so, how?
- Which performance measures are important to accurately evaluate the performance of the decoder / predictor?
- Would you use the decoder / predictor in real life? What are they suitable for?
- What are the challenges when working with this data set? How could you try to improve the training data? How important are the selected features and the annotations in the file? Which additional features would you think could improve the performance?

Hints:

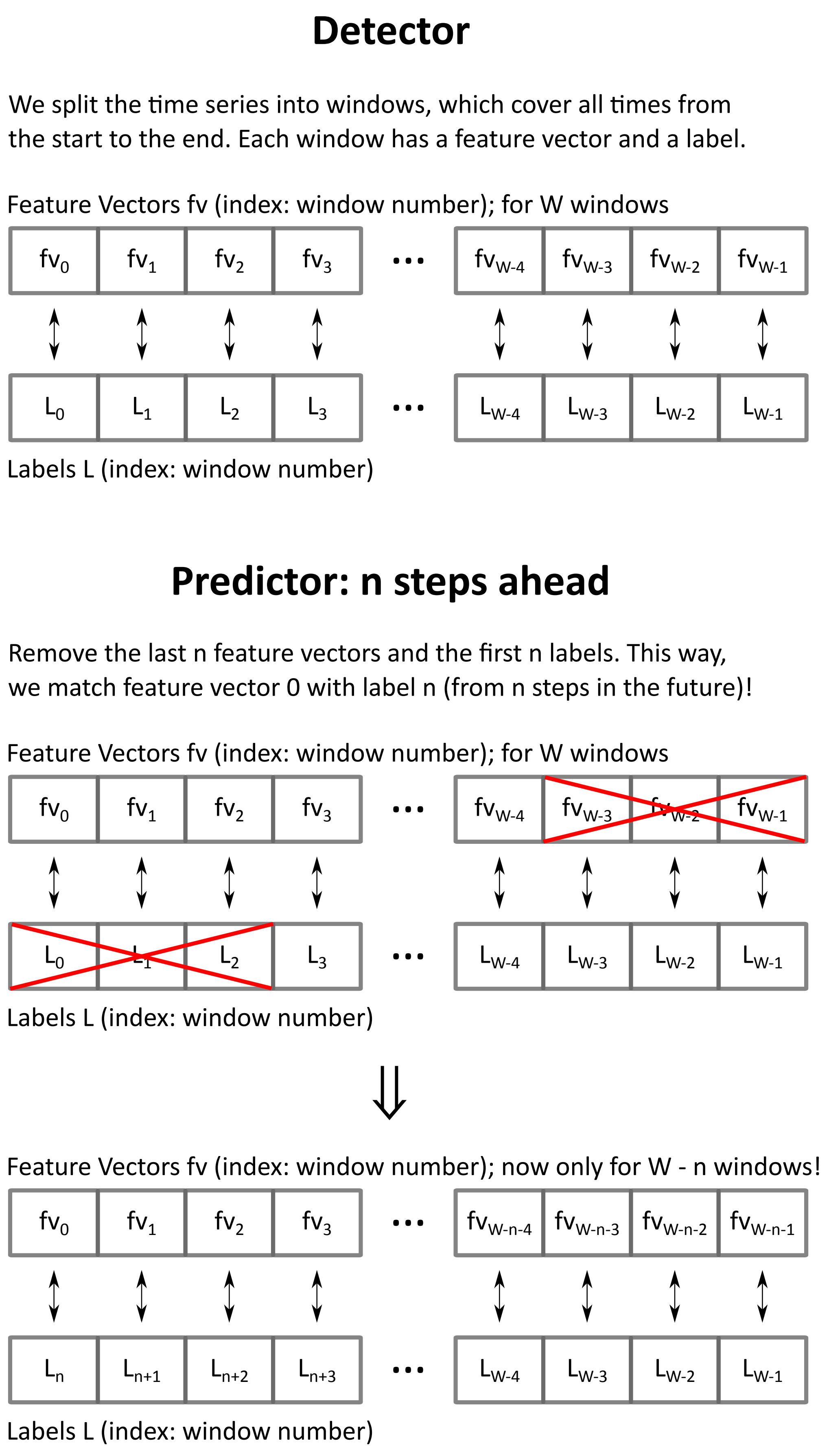
- Use plt.xlabel(…), plt.ylabel(…), plt.title(…) to provide titles for the plots and their axes.
- When plotting several graphs into the same figure, use

  ```
  plt.plot(…, label='about this graph')
  ```

  to provide a label which can then be made visible by using

  ```
  plt.legend(…)
  ```

- To create images from python, use plt.savefig('filename.ext') to save a figure to an image file. Use plt.xlim((x_{min}, x_{max})) and plt.ylim((y_{min}, y_{max})) to modify the visible part of the figure, i.e., zoom in.
- And of course, always try the documentation to get further hints on possible parameters, examples, use cases, etc.. The matplotlib gallery will give you a lot of ideas about how you can modify a plot. The seaborn gallery can also give you some ideas but is more targeted towards tabular data sets and not necessarily usable for this module.
- Whenever something is unclear or you do not know how to go on – please ask!



Detector

We split the time series into windows, which cover all times from the start to the end. Each window has a feature vector and a label.

Predictor: n steps ahead

Remove the last n feature vectors and the first n labels. This way, we match feature vector 0 with label n (from n steps in the future)!

Feature Vectors fv (index: window number); for W windows
fv_0 fv_1 fv_2 fv_3 … fv_{W-4} fv_{W-3} fv_{W-2} fv_{W-1}
L_0 L_1 L_2 L_3 … L_{W-4} L_{W-3} L_{W-2} L_{W-1}
Labels L (index: window number)
Feature Vectors fv (index: window number); for W windows
fv_0 fv_1 fv_2 fv_3 … fv_{W-4} fv_{W-3} fv_{W-2} fv_{W-1}
L_0 L_1 L_2 L_3 … L_{W-4} L_{W-3} L_{W-2} L_{W-1}
Labels L (index: window number)
Feature Vectors fv (index: window number); now only for W - n windows!
fv_0 fv_1 fv_2 fv_3 … fv_{W-n-4} fv_{W-n-3} fv_{W-n-2} fv_{W-n-1}
L_n L_{n+1} L_{n+2} L_{n+3} … L_{W-4} L_{W-3} L_{W-2} L_{W-1}
Labels L (index: window number)

Figure 1: How to prepare your data when going from a detector to a minimalistic predictor.