Exercise

- Leverage the Human Activity Recognition Using Smartphones Data Set:
  - activity (eg: walking, laying etc) records for a group of 30 people
  - raw embedded accelerometer and gyroscope signals
  - transformations of these signals (aka *engineered features*)
  - splitted into a training and a test sets (group splitting)

- To train models that predict the 6 different possible activities:
  - a non deep learning model using any data available
  - a deep learning model using the raw signals

- Produce a report containing the models results and 1 visualisation

3- group features

isolate a set of base features and create thematic groups of features

4- select features

using an L1 penalised logistic regression, determine best features among each group

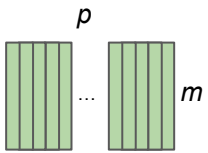
inputs

raw inertial signals

- total acceleration x, y, z
- body acceleration x, y, z
- body angular velocity x, y, z

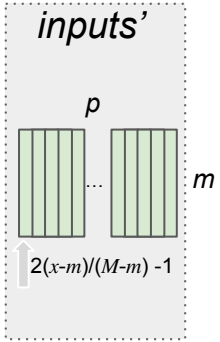
1- engineer features

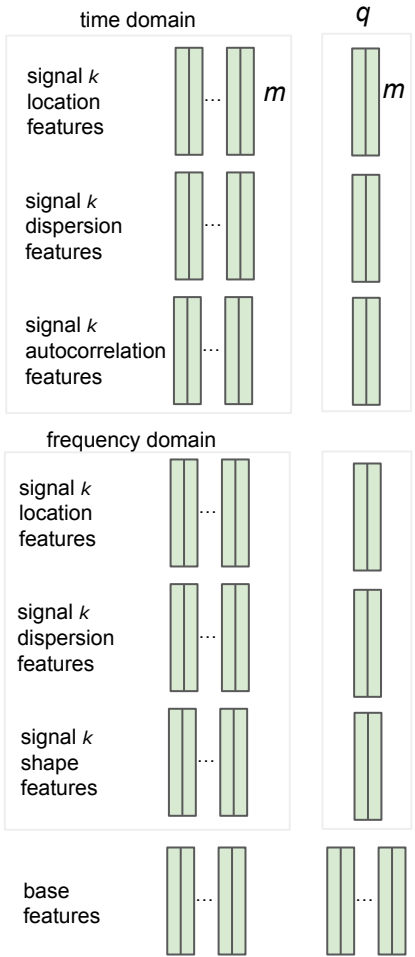
various signal processing techniques



2- preprocess

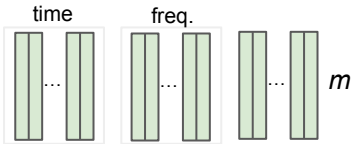
min-max scaling,
caveat - it's likely that it has been done on the union training/test sets (eg: all features which don't reach 1/-1 in the training set do reach the boundary in the test set)





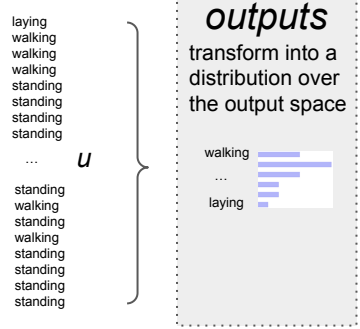
5- merge

concatenate the base features with the best features found in each thematic group



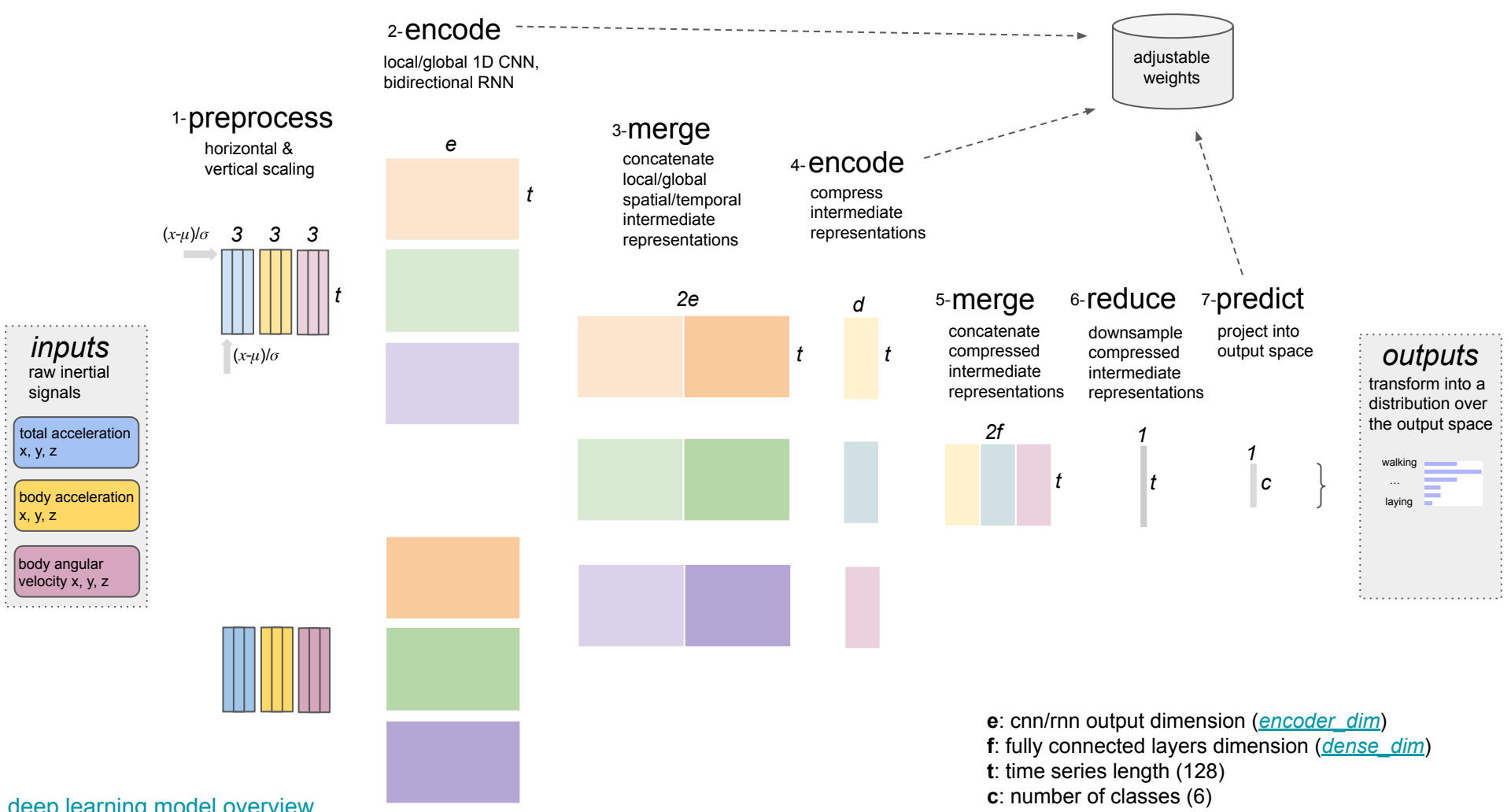
6- predict

evaluate each row against all the decision trees fitted at training time on re-sampled version of the dataset



m: training set size (7,352)
p: initial number of features (561)
q: number of features kept per group (*max_extra_feat*)
u: best number of trees (*n_estimators*) determined by cross validation

[non deep learning model overview]



e: cnn/rnn output dimension (*encoder_dim*)
f: fully connected layers dimension (*dense_dim*)
t: time series length (128)
c: number of classes (6)

deep learning model overview

Project structure

- Makefile orchestrates all the recurring tasks of the project:
  - improves reproducibility
  - make targets are good candidates to jobs in a CI pipeline (eg: GitHub Actions)
  - typically `make test` !

- TOML configuration files contain model parameters:
  - should be version controlled
  - files differences are easier to review
  - allows dictionaries (helpful to define cross validation grid)
  - needs some validation (eg: perhaps with an additional logic layer using pydantic)

- All models steps are encapsulated into a serialisable self-sufficient model:
  - mitigates the risk of training-serving skew
  - Tensorflow SavedModel format for the deep learning model
  - Joblib format for the non deep learning model

Results

- Given the labels are relatively balanced it is safe to use accuracy to compare models
- Deep learning model performs better (without feature engineering, less data & no data leakage)
- Models struggle between the sitting and standing positions

deep learning model

Model accuracy 93.62%

	1 WALKING	2 WALKING_UPSTAIRS	3 WALKING_DOWNSTAIRS	4 SITTING	5 STANDING	6 LAYING
1 WALKING	457	15	0	24	0	0
2 WALKING_UPSTAIRS	2	443	2	24	0	0
3 WALKING_DOWNSTAIRS	0	3	411	6	0	0
4 SITTING	0	3	0	431	57	0
5 STANDING	0	1	0	49	482	0
6 LAYING	0	0	0	0	2	535

non deep learning model

Model accuracy 91.45%

	1 WALKING	2 WALKING_UPSTAIRS	3 WALKING_DOWNSTAIRS	4 SITTING	5 STANDING	6 LAYING
1 WALKING	485	3	8	0	0	0
2 WALKING_UPSTAIRS	65	398	8	0	0	0
3 WALKING_DOWNSTAIRS	25	58	337	0	0	0
4 SITTING	0	0	0	426	65	0
5 STANDING	0	0	0	20	512	0
6 LAYING	0	0	0	0	0	537

error analysis deep learning model (parallel coordinate plot)