Resources

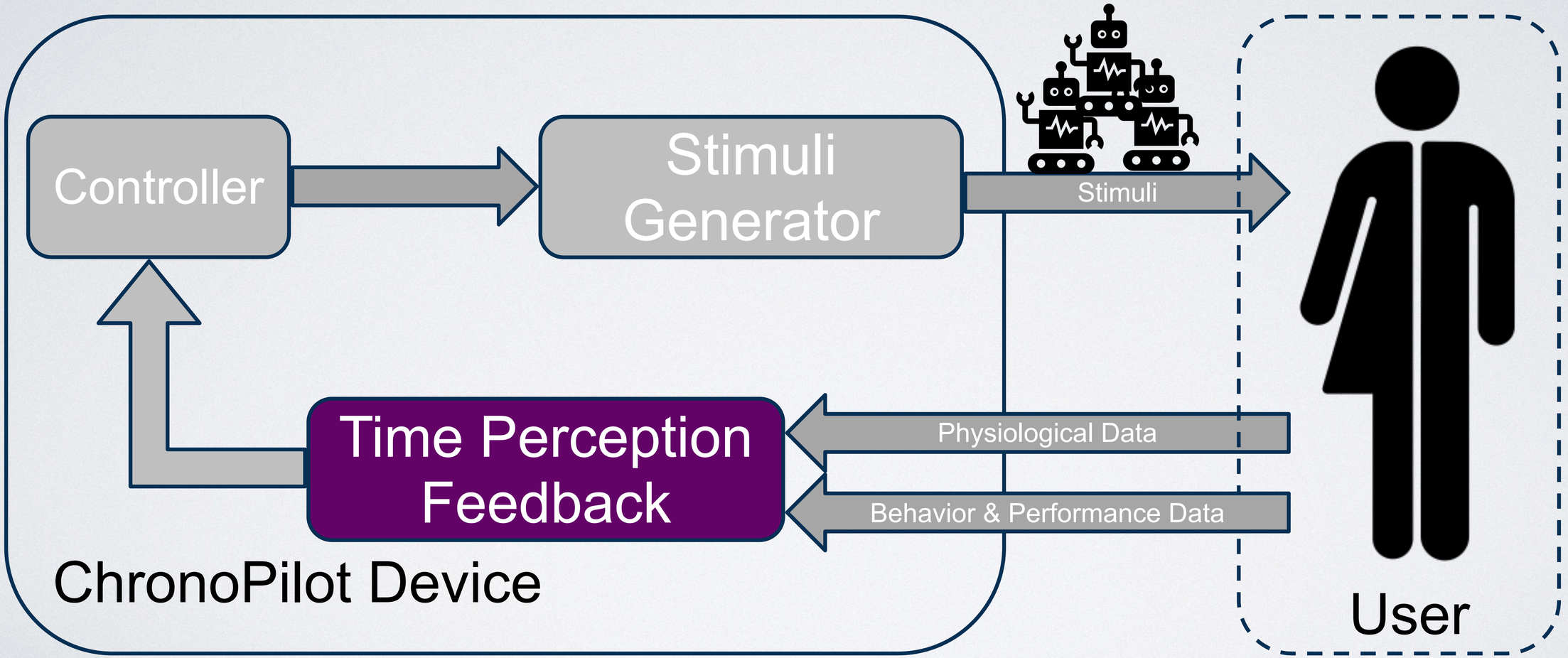
CHRONO PILOT

AUTOMATED FEEDBACK FROM PHYSIOLOGICAL DATA FOR TIME PERCEPTION STUDIES

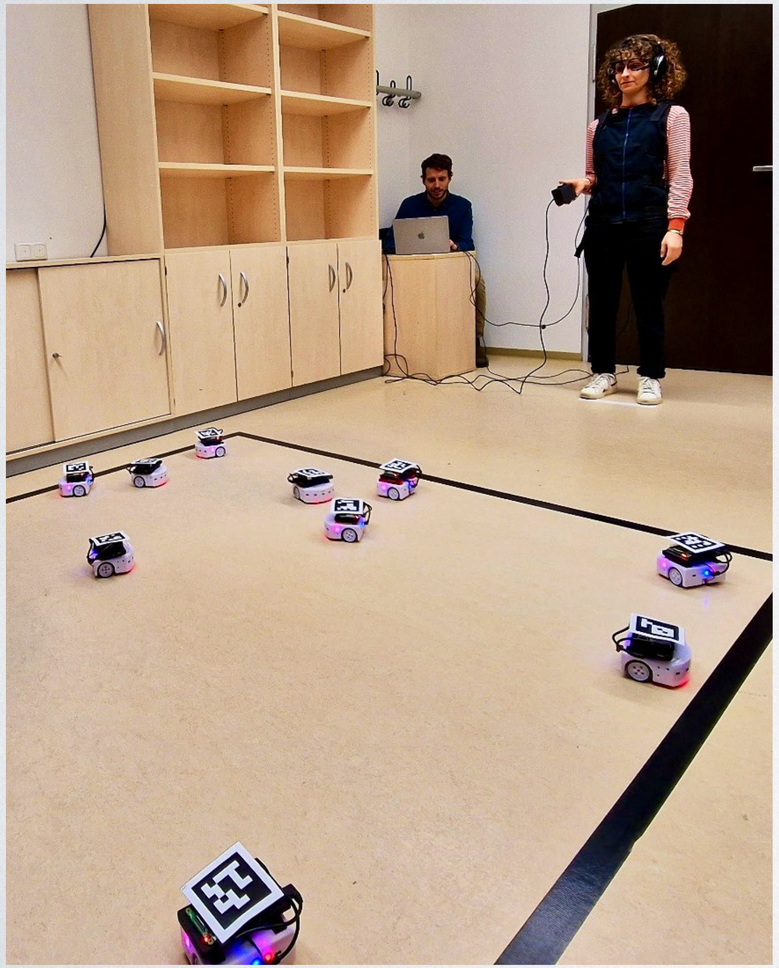
Till Aust

Funded by the European Union

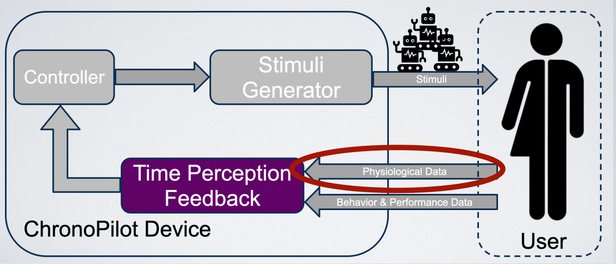
CLOSING THE LOOP

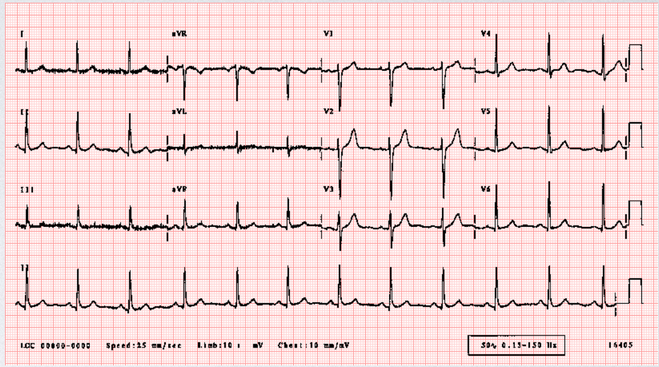


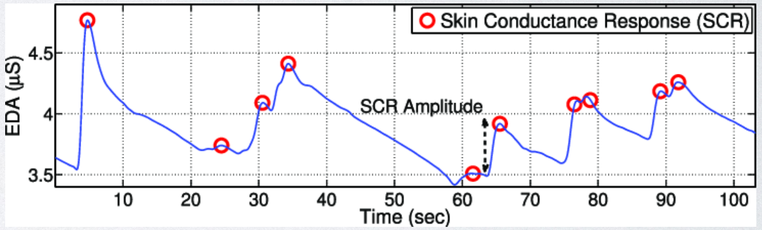
HOW?

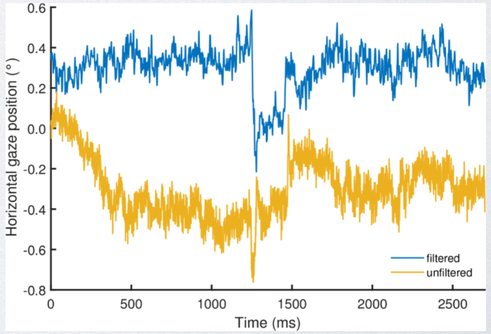



POLAR

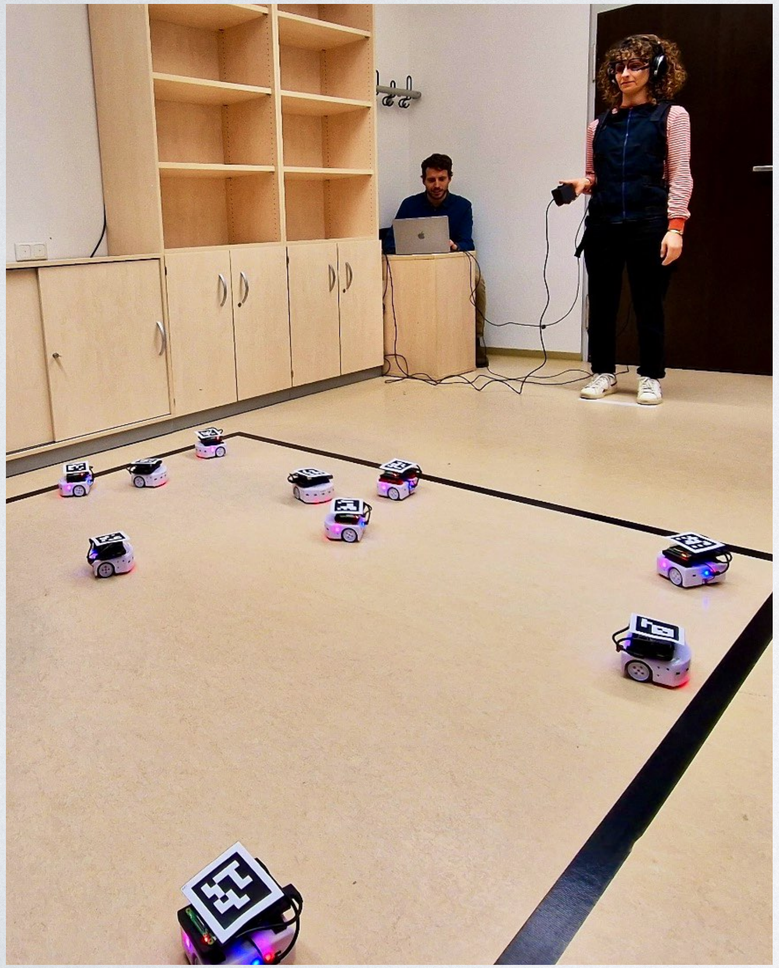

Controller
Stimuli Generator
Stimuli
Time Perception Feedback
Physiological Data
Behavior & Performance Data
ChronoPilot Device
User

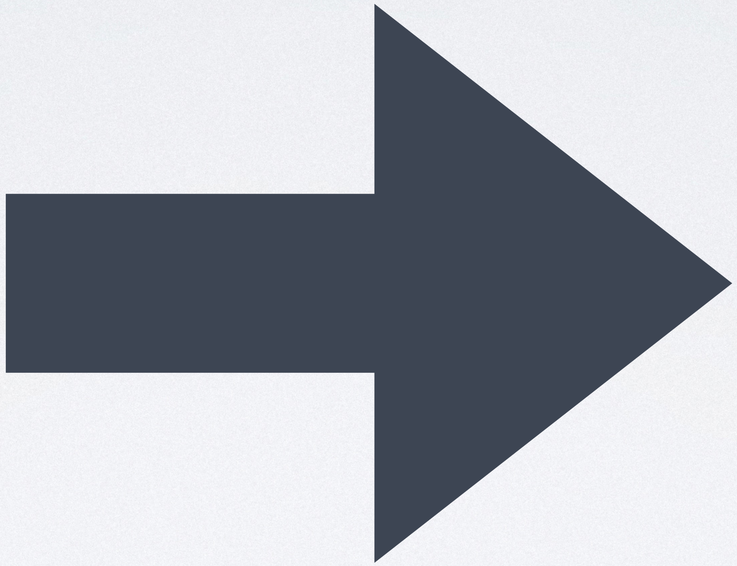


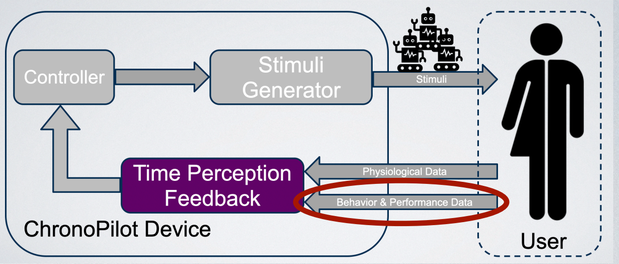

Skin Conductance Response (SCR)
SCR Amplitude
EDA (µS)
Time (sec)


Horizontal gaze position (°)
filtered
unfiltered
Time (ms)

HOW?







1. Please estimate the duration of the experiment round and indicate it on the timeline:

1:00 1:30 2:00 2:30 3:00 3:30 4:00 4:30 5:00

2. Please indicate how you perceived the time passing during the experiment round:

Very Slowly | Slowly | Neither Fast not Slowly | Fast | Very Fast

3. Please indicate your emotional valence during the experiment round:

Happy | Neutral | Sad

4. Please indicate your emotional arousal during the experiment round:

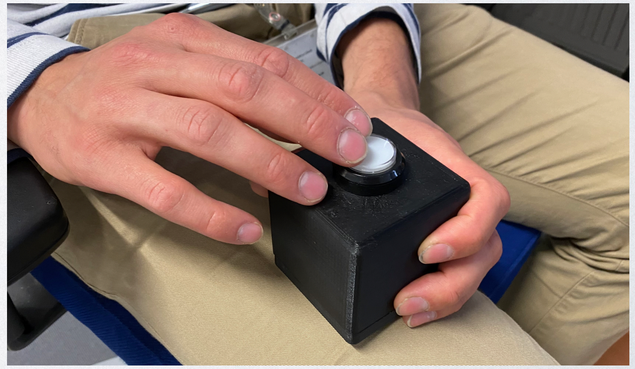
Active | Neutral | Calm

5. Please indicate your perceived level of flow during the experiment round:

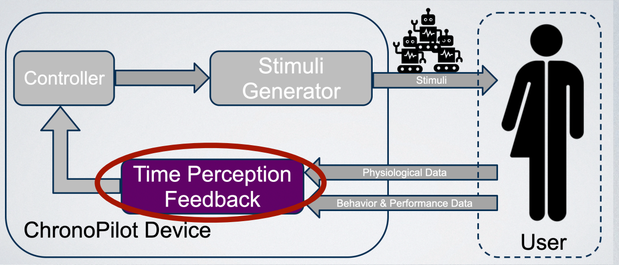
Not at All | Very Much

6. Please indicate how you perceived the task demand of the experiment round:

Too Low | Just Right | Too High

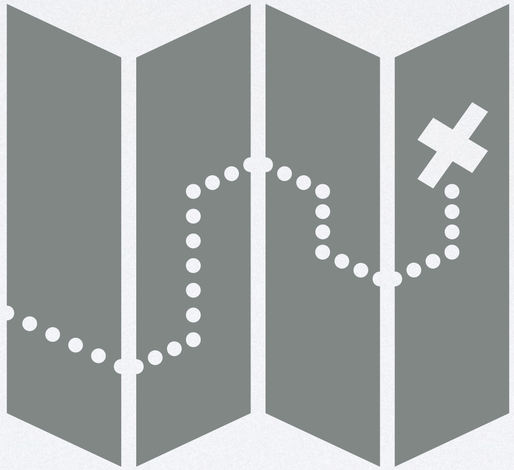


FORMALIZING



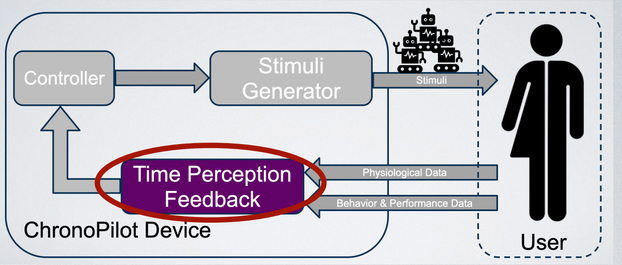
What Problem do we need to solve?

User's Physiological data

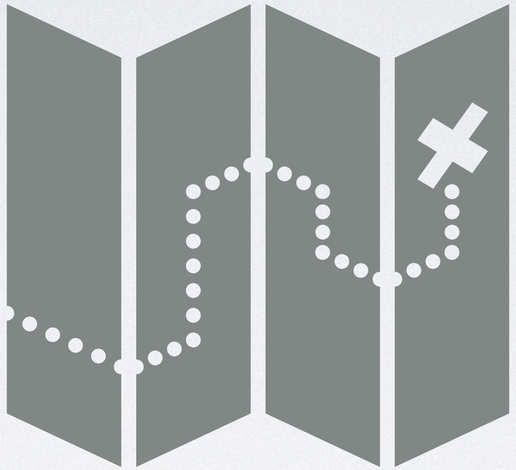


User's subjective time perception

FORMALIZING



User's Physiological data



User's subjective time perception

$$X \mapsto Y$$

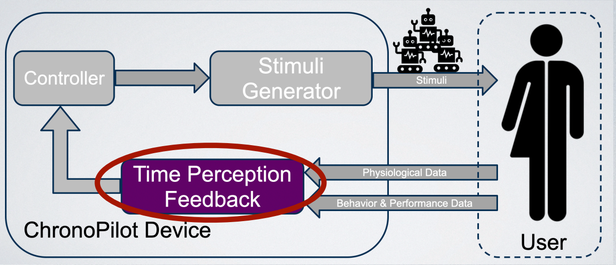
Classification

$$X \mapsto Y \in \{0,1,2,...,n\}$$

Regression

$$X \mapsto Y \in \mathbb{R}$$

FORMALIZING



Y: Subjective time perception labels from the questionnaire, e.g.

{slow, fast} {overestimation, underestimation} {slow, medium, fast}

X: **Time series data** of physiological signals

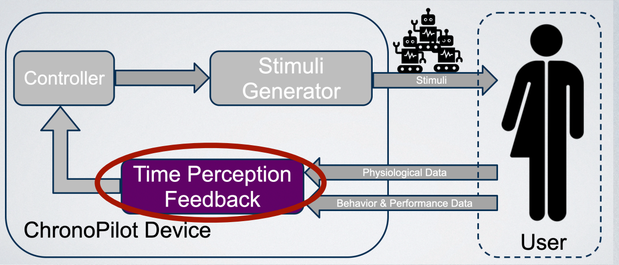
Data can be large and noisy → less descriptive

Can we **condense information** within the given data?

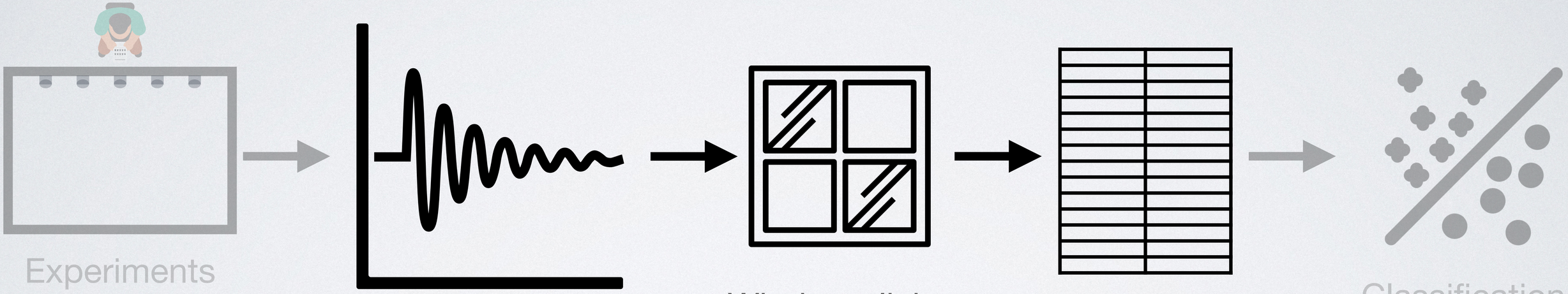
Preprocessing

$$\mathbb{R}^{t \times n} \longrightarrow \mathbb{R}^{m}$$

PREPROCESSING



$$\mathbb{R}^{t \times n} \longrightarrow \mathbb{R}^{m}$$



Experiments

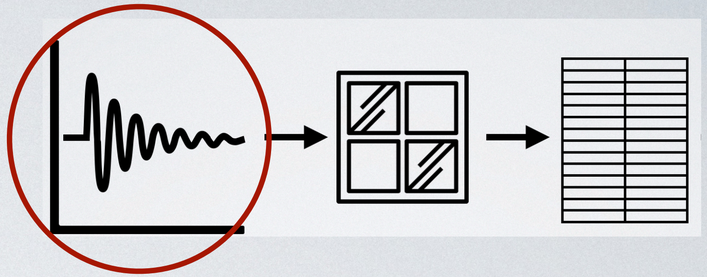
Physiological Data
- ECG (HR, RR, …)
- EDA
- SKT
- PPG
- Eye-tracking

Window slicing
Data cleaning

Feature extraction

Classification / Regression

PREPROCESSING

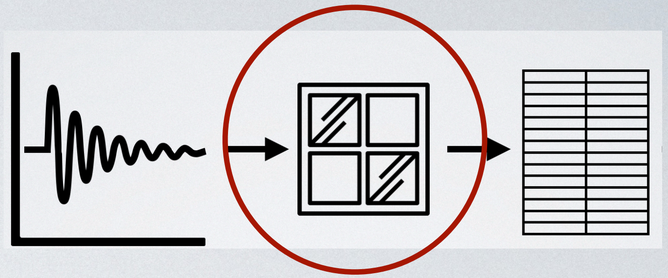


Challenges:

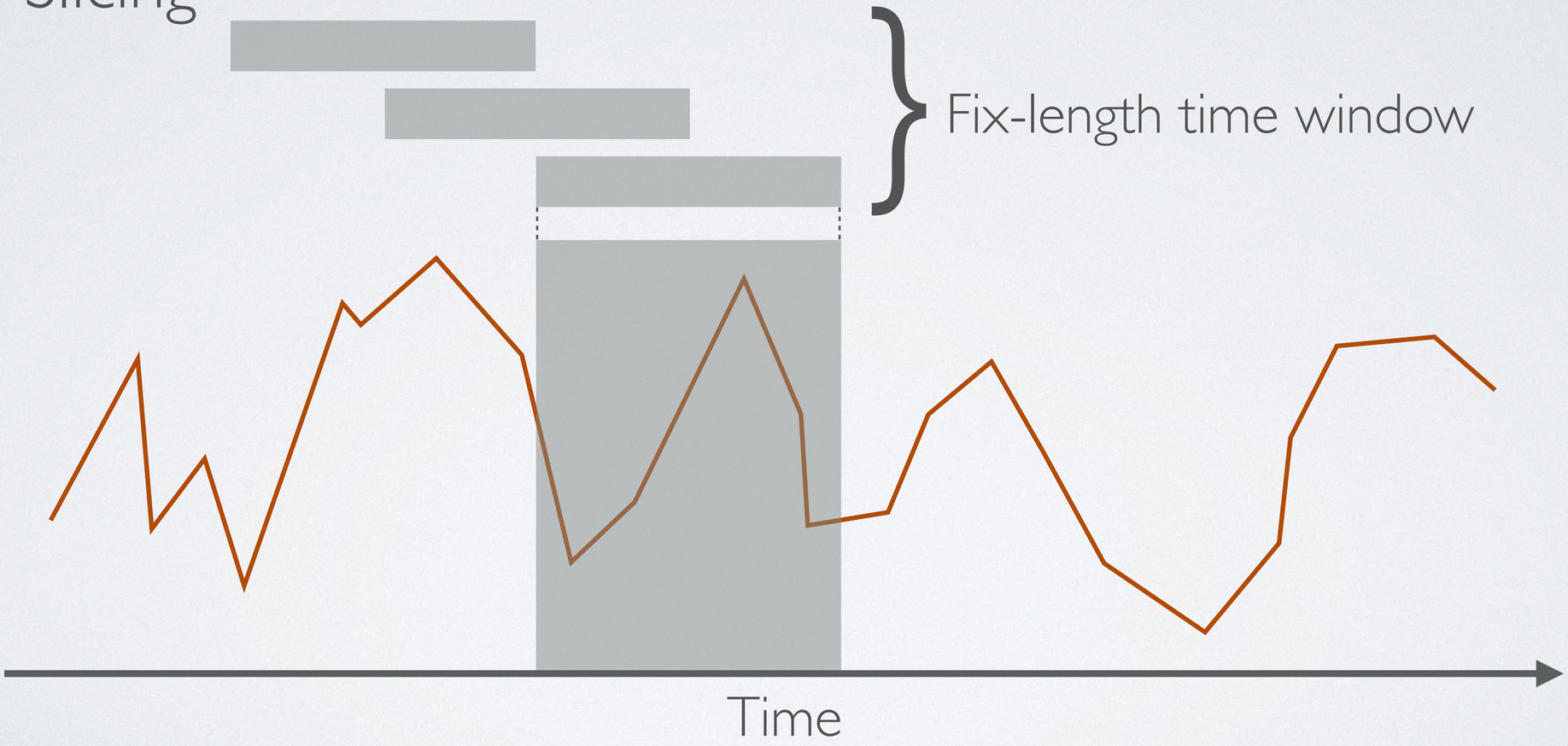
- Different sampling frequencies
- Delay in recording + alignment of different data sources
- Noise due to sensor movement

LabStreamingLayer: https://labstreaminglayer.readthedocs.io/info/getting_started.html

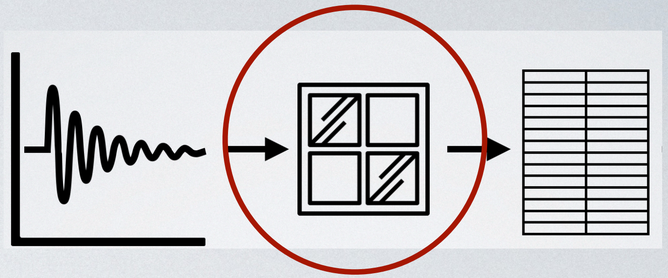
PREPROCESSING



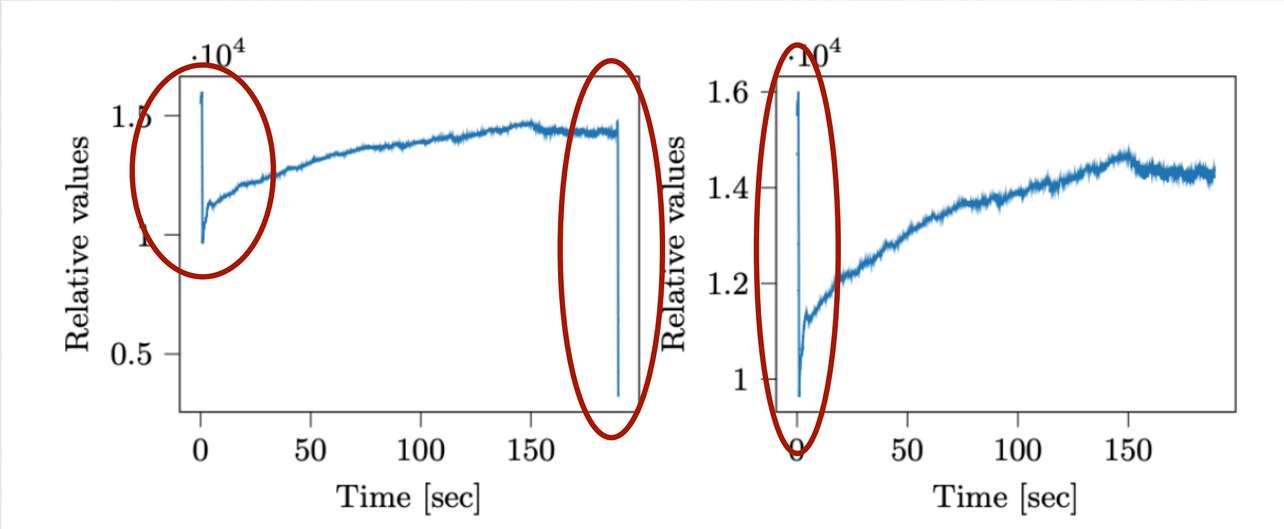
Window Slicing

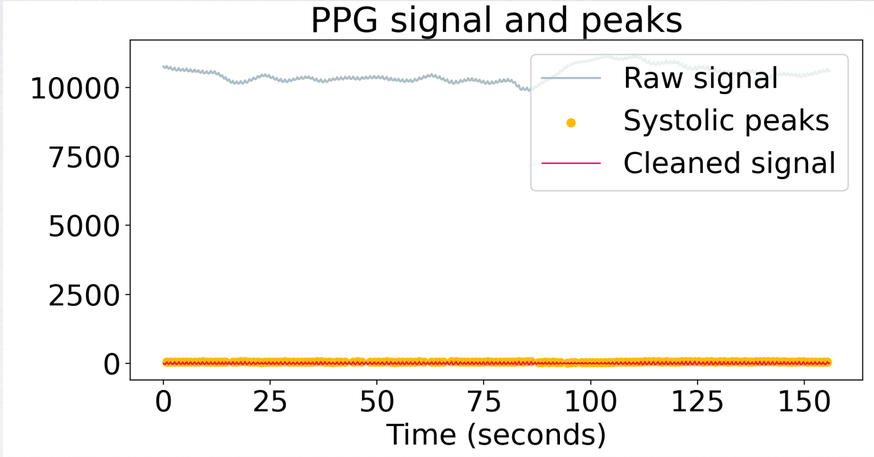


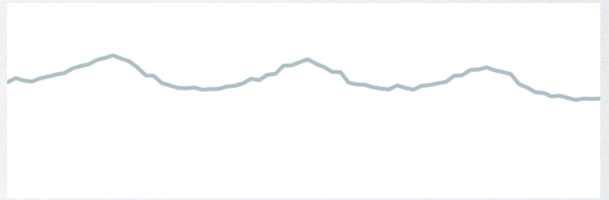
PREPROCESSING

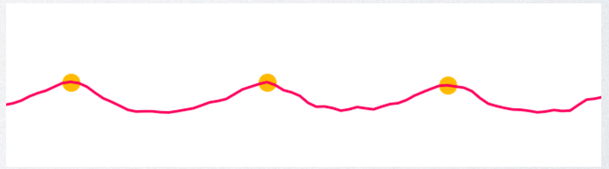


Data Cleaning

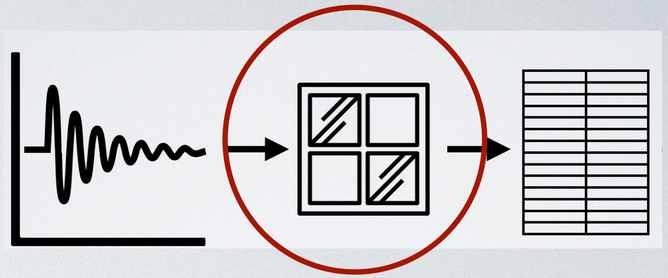








PREPROCESSING



Data Cleaning

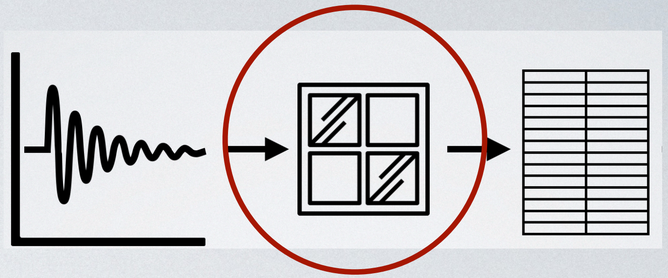
Can be very complex

In practice you can use libraries, such as, NeuroKit2 or HeartPy

heartpy 1.2.7

NeuroKit: https://neuropsychology.github.io/NeuroKit/
HeartPy:https://python-heart-rate-analysis-toolkit.readthedocs.io/en/latest/

PREPROCESSING



Data Cleaning (function call)

```
# Simulate 10 seconds of EDA Signal (recorded at 250 samples / second)
eda_signal = nk.eda_simulate(duration=10, sampling_rate=250, scr_number=3, drift=0.01)
```

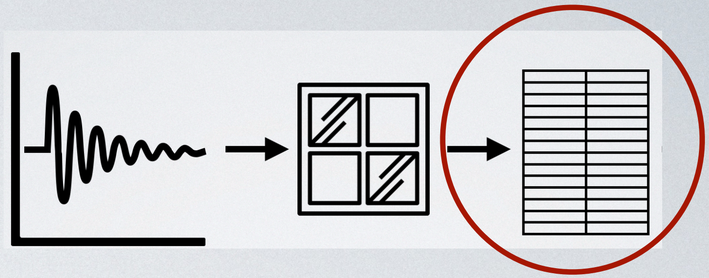
```
# Process the raw EDA signal
signals, info = nk.eda_process(eda_signal, sampling_rate=250)
```

Additional information

Cleaned and deconstructed signal

NeuroKit: https://neuropsychology.github.io/NeuroKit/

PREPROCESSING



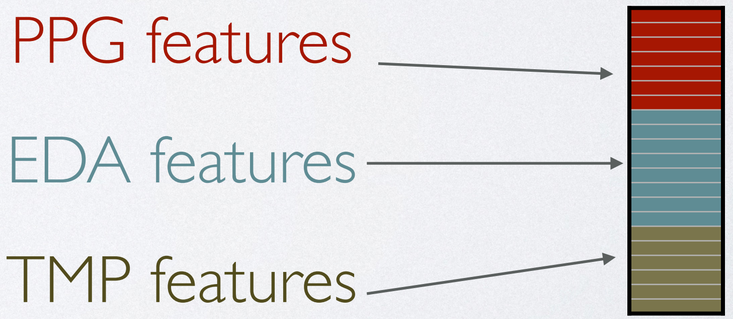
Feature extraction

Condensing information by reducing the time dimension

Extract biomarkers using established methods

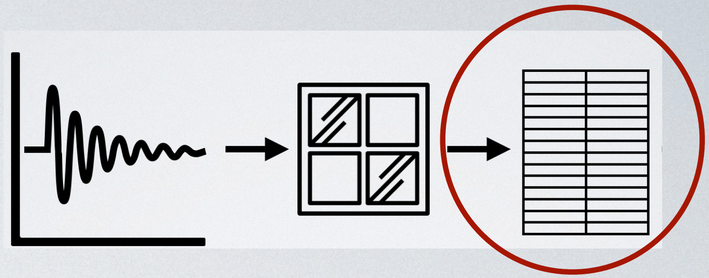
Stack biomarkers to feature vectors

Feature vector



NeuroKit: https://neuropsychology.github.io/NeuroKit/

PREPROCESSING



Feature extraction (function call)

Preprocessed data

```
# Analyze
In [11]: analyze_df = nk.eda_analyze(df, sampling_rate=100)
```

EDA features

Feature vector

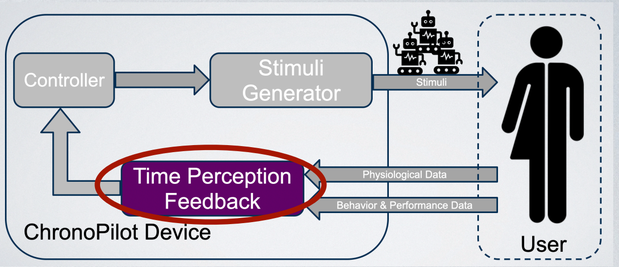
PPG features

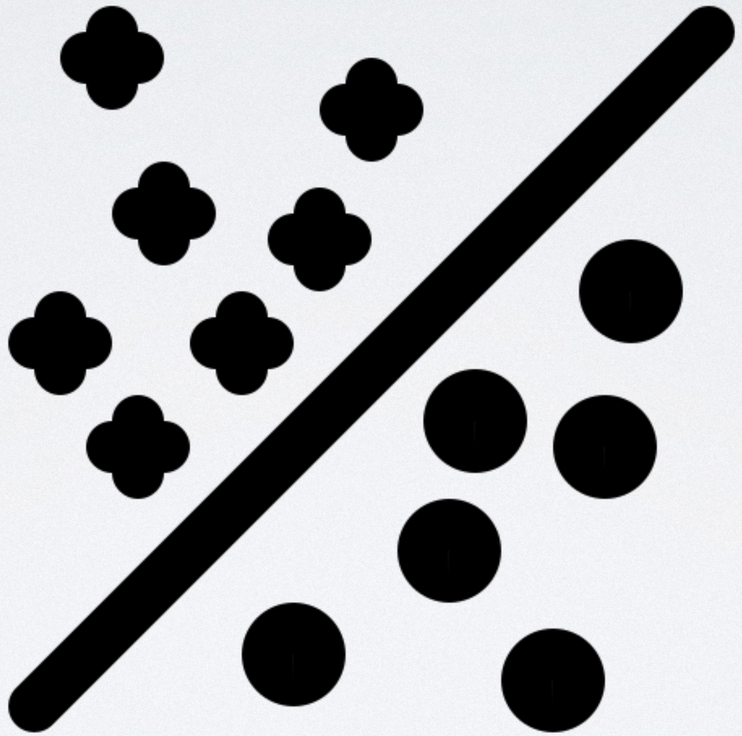
EDA features

TMP features

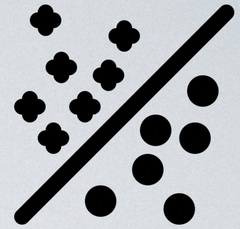
NeuroKit: https://neuropsychology.github.io/NeuroKit/

CLASSIFICATION





CLASSIFICATION



We have

Users/participants

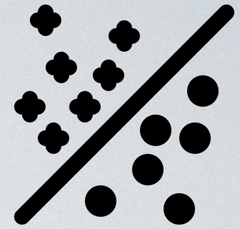
Number of classes

Number of features

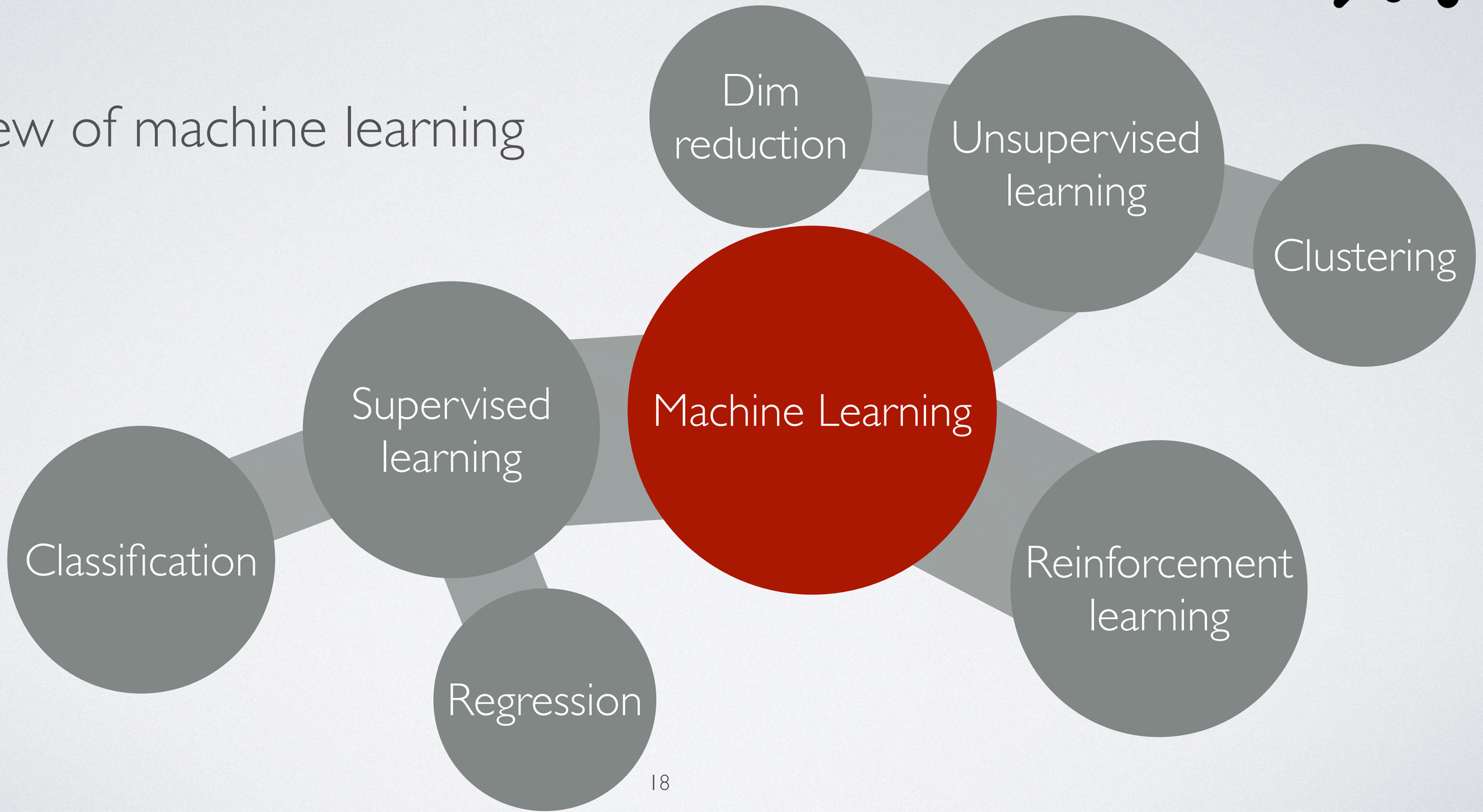
$$X = \mathbb{R}^{p \times m} \qquad y = \mathbb{R}^{p \times N}$$

How does $\mapsto$ look like?

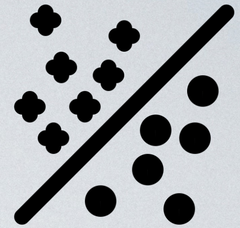
CLASSIFICATION



Overview of machine learning

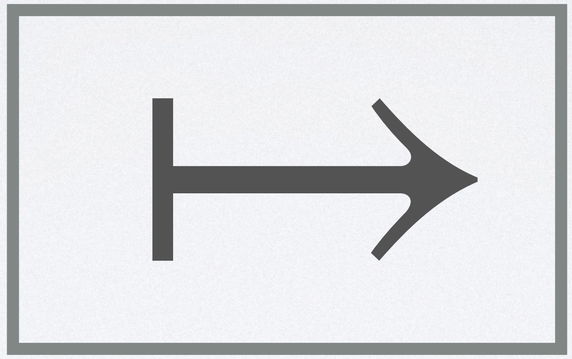


CLASSIFICATION



In our case we want to have

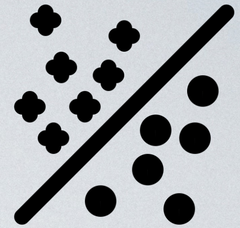
Feature vector



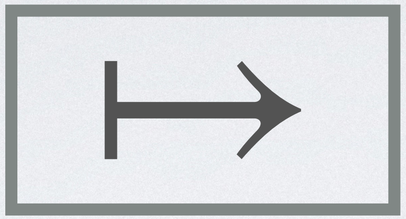
Subjective time perception

{slow, fast}

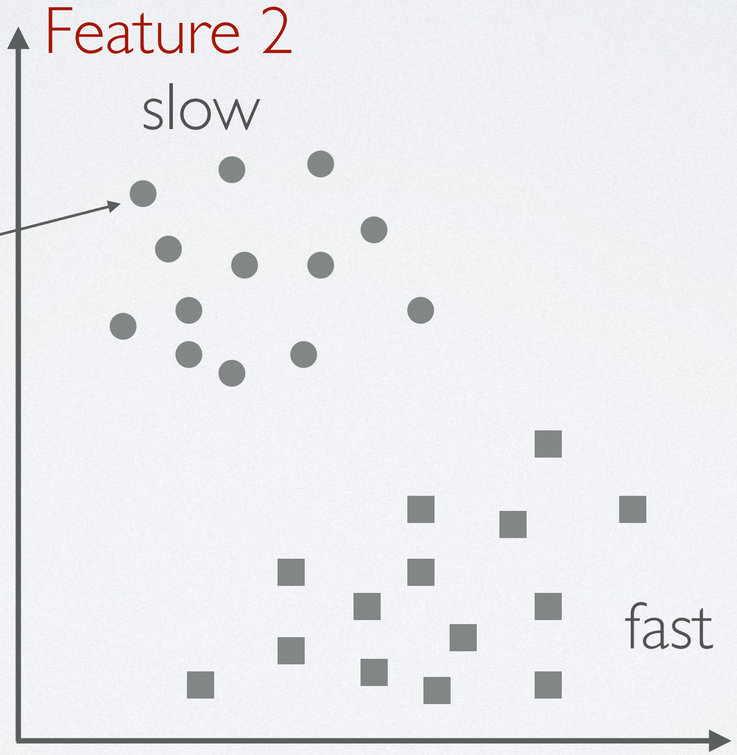
CLASSIFICATION



Possible implementation

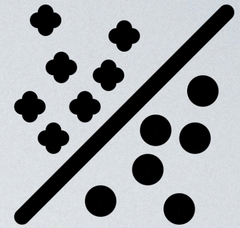


Feature vector

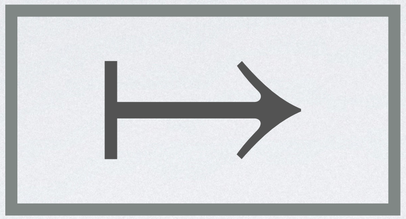


Maximum margin classifier
Support vector machine

CLASSIFICATION



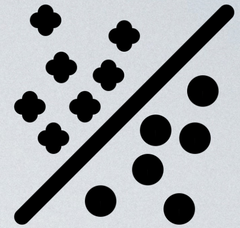
Possible implementation



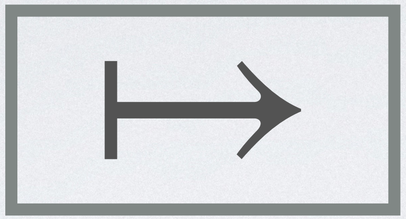
Feature vector

Maximum margin classifier
Support vector machine

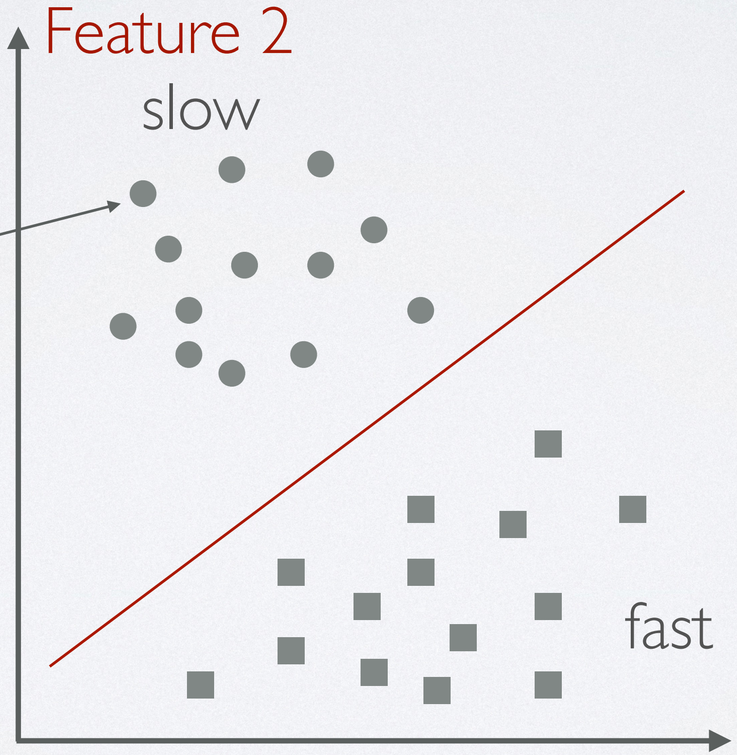
CLASSIFICATION



Possible implementation

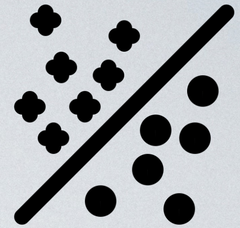


Feature vector

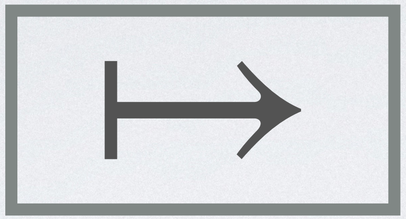


Maximum margin classifier
Support vector machine

CLASSIFICATION

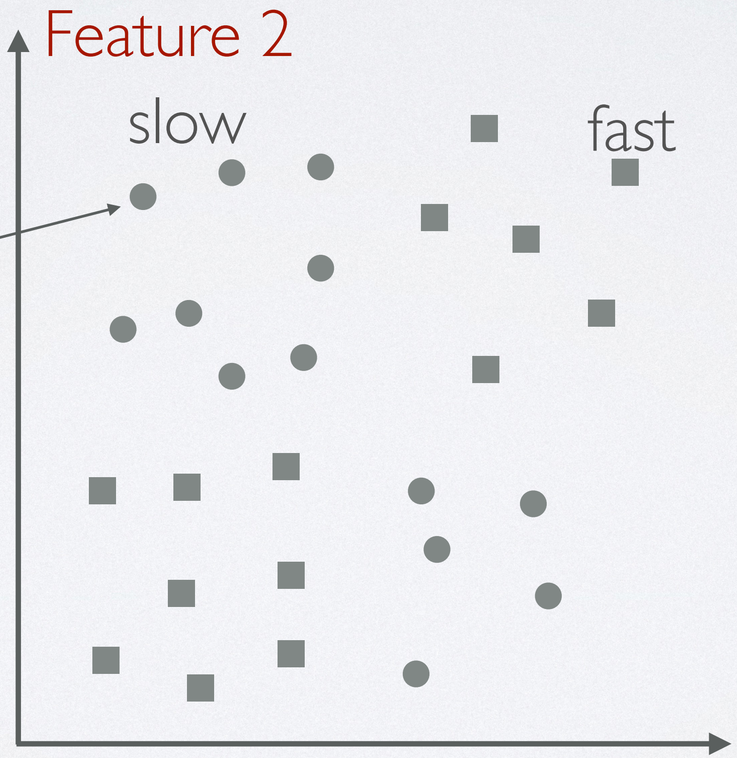


Possible implementation

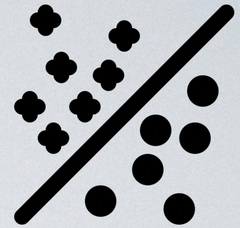


Decision trees

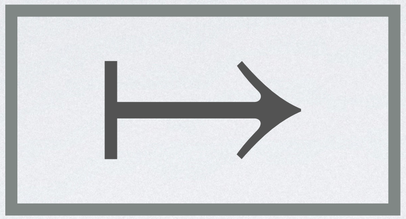
Feature vector



CLASSIFICATION



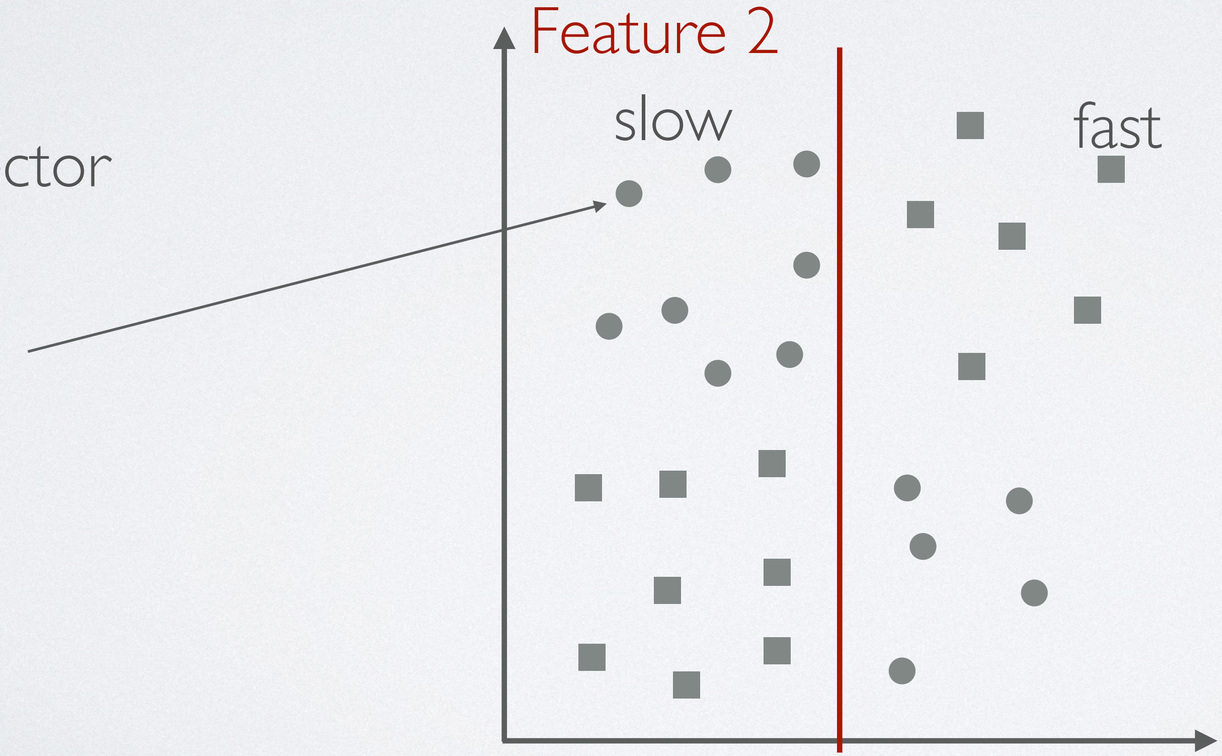
Possible implementation



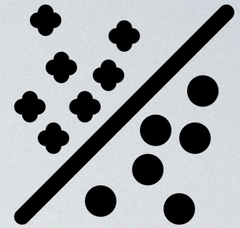
Decision trees

Feature 1 > 2

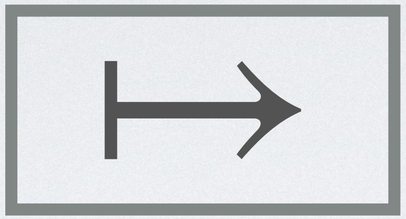
Feature vector



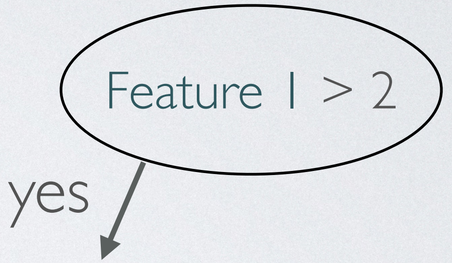
CLASSIFICATION



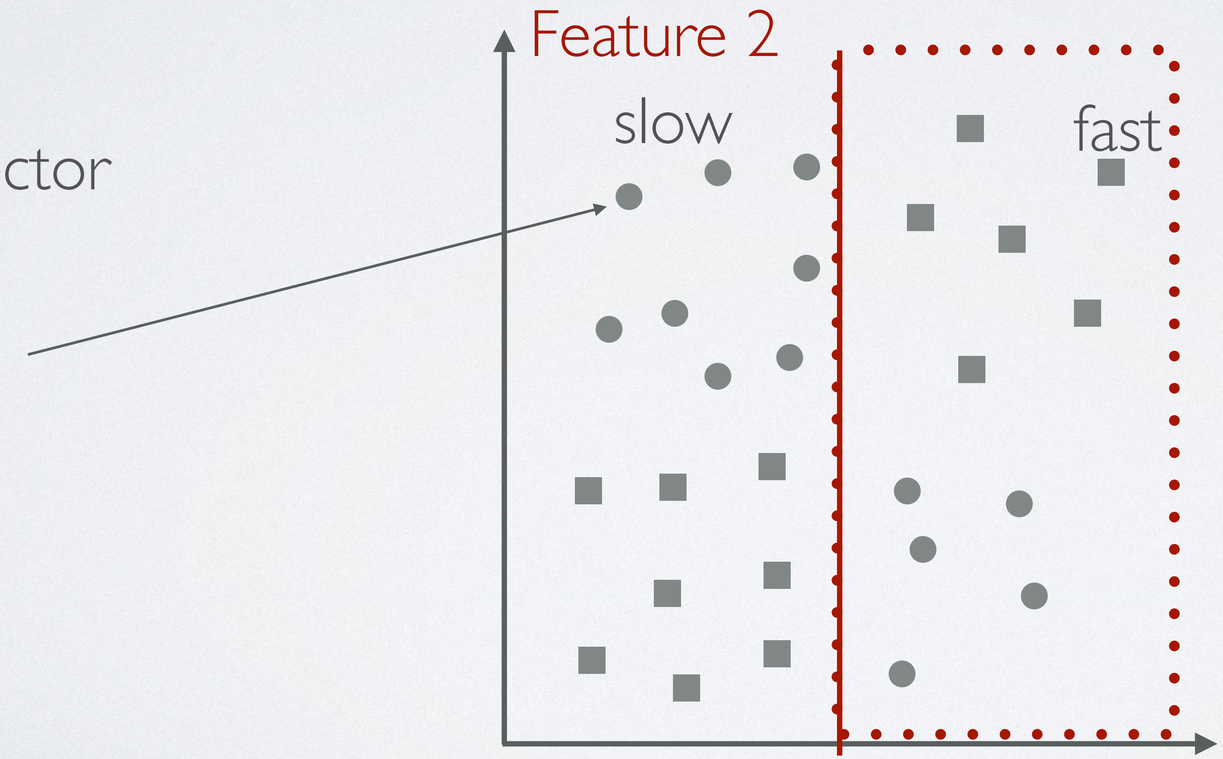
Possible implementation



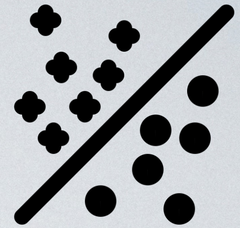
Decision trees



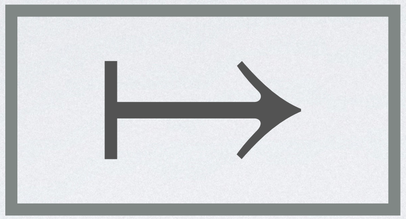
Feature vector



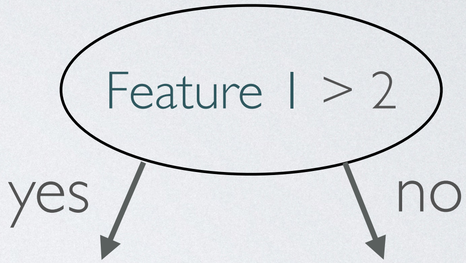
CLASSIFICATION



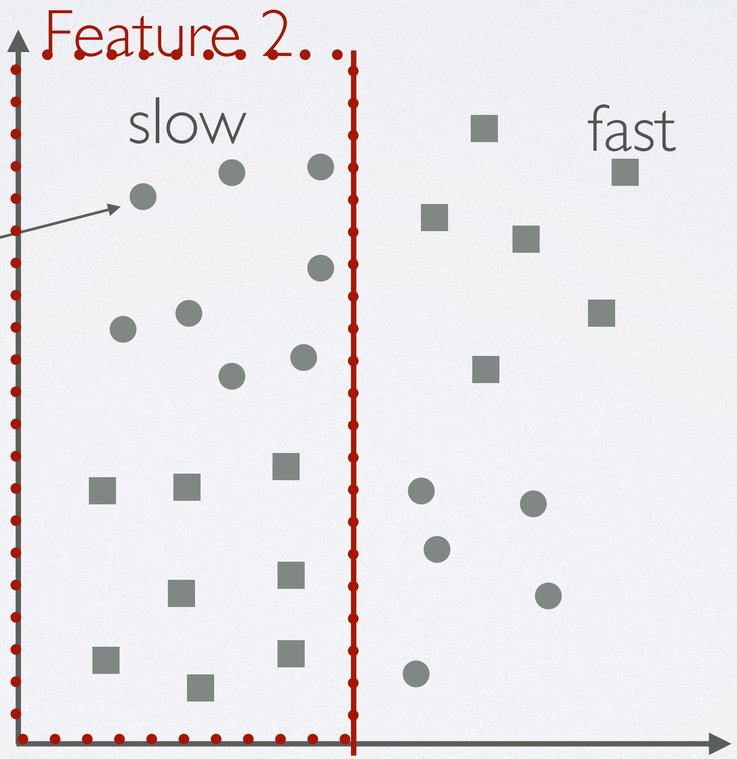
Possible implementation



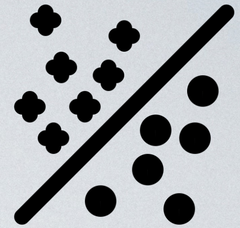
Decision trees



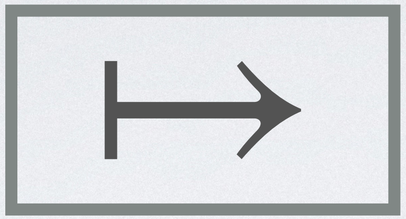
Feature vector



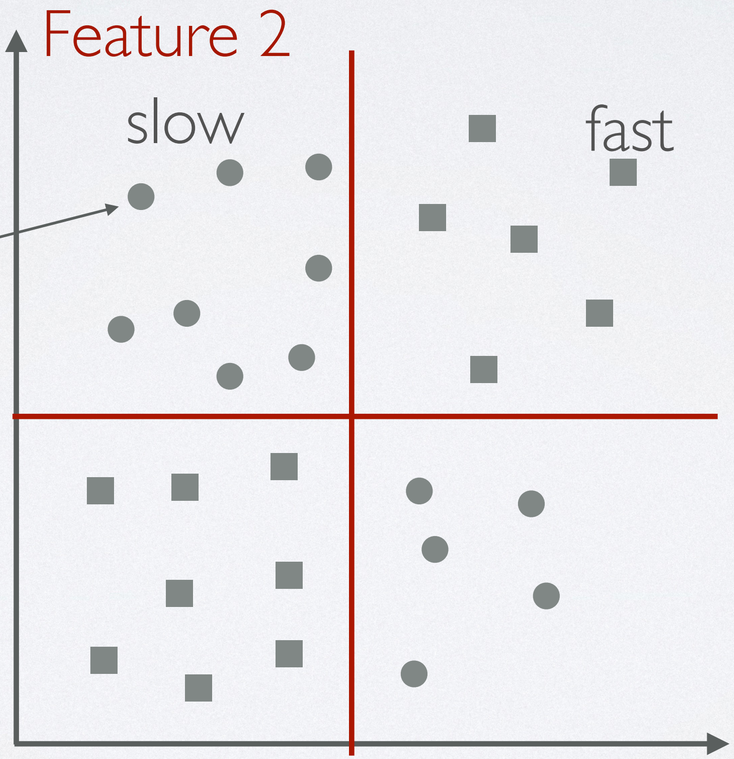
CLASSIFICATION



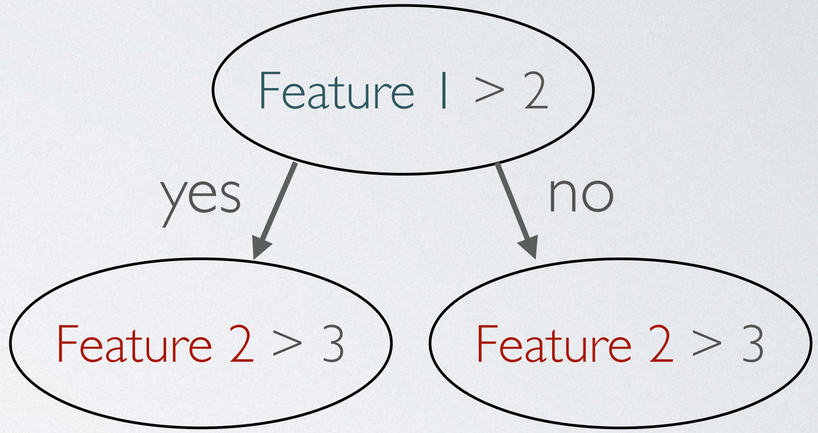
Possible implementation



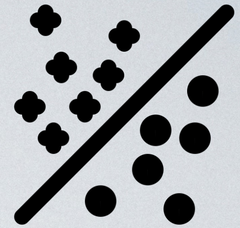
Feature vector



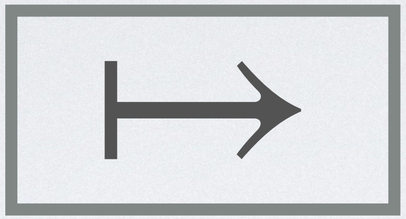
Decision trees



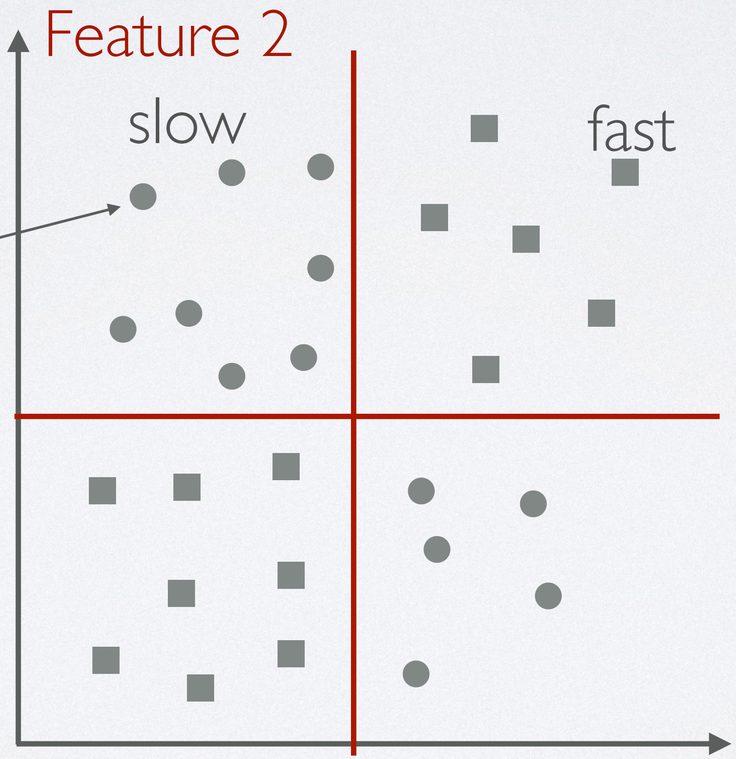
CLASSIFICATION



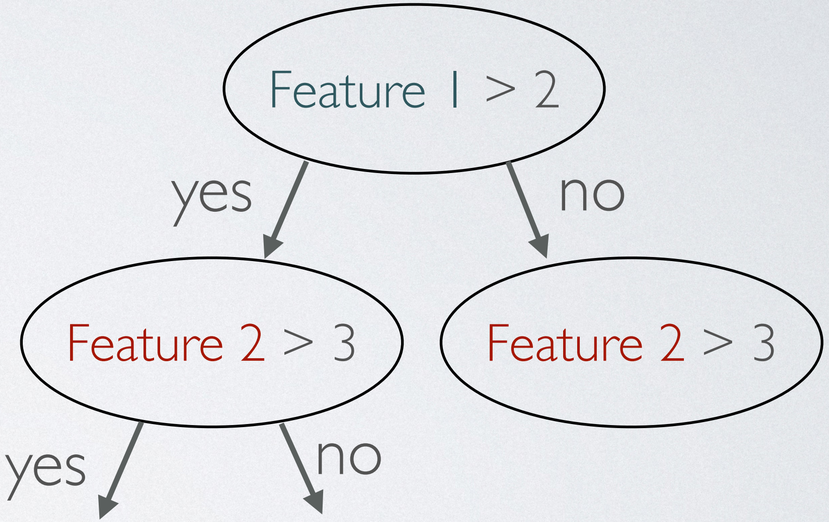
Possible implementation



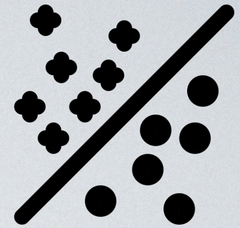
Feature vector



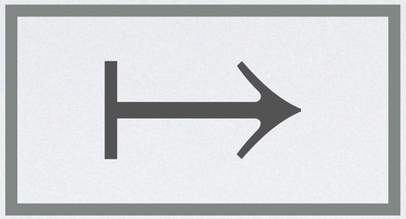
Decision trees



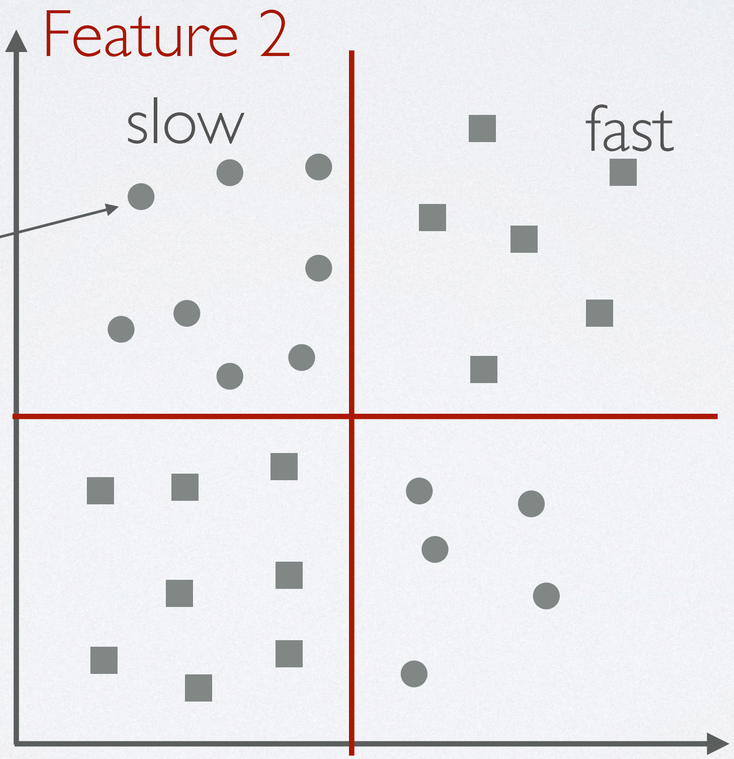
CLASSIFICATION



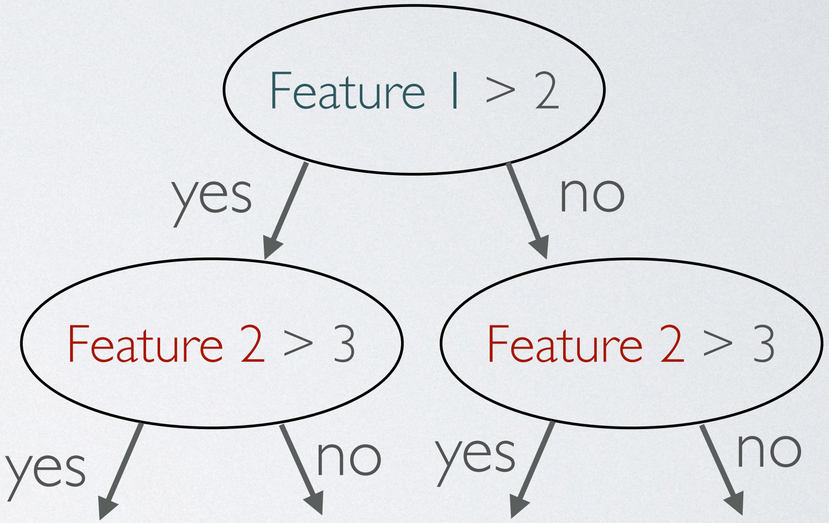
Possible implementation



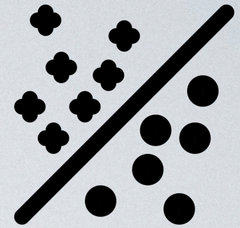
Feature vector



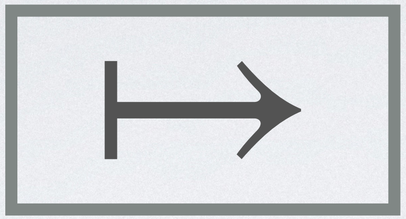
Decision trees



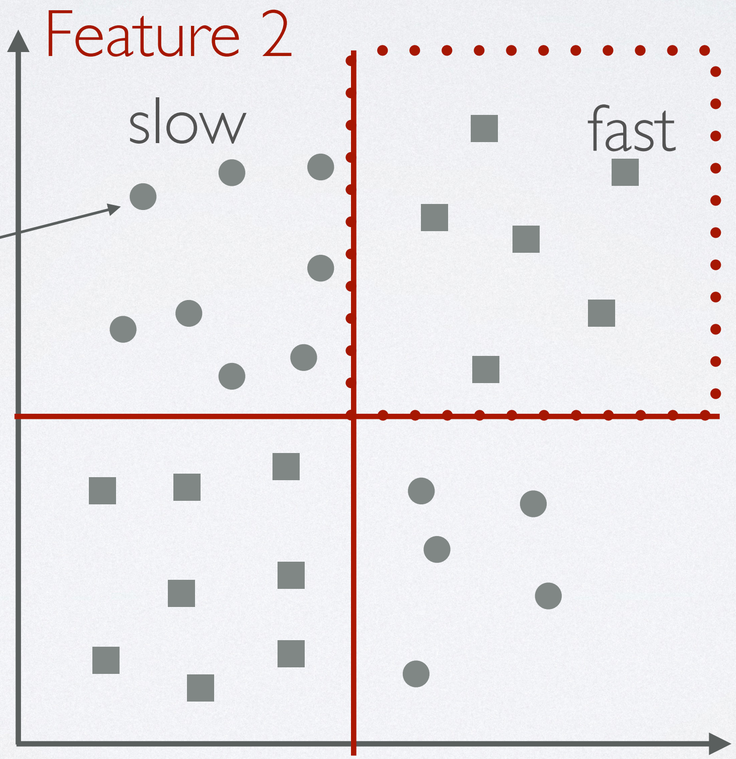
CLASSIFICATION



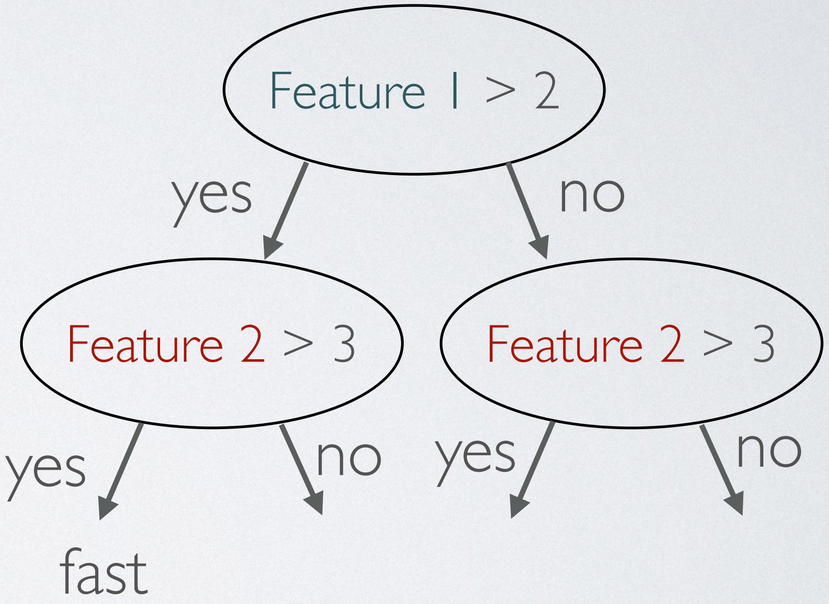
Possible implementation



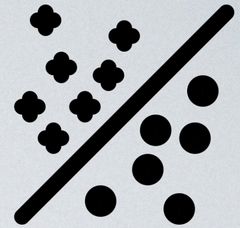
Feature vector



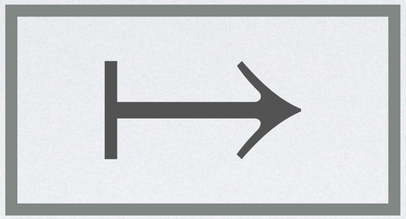
Decision trees



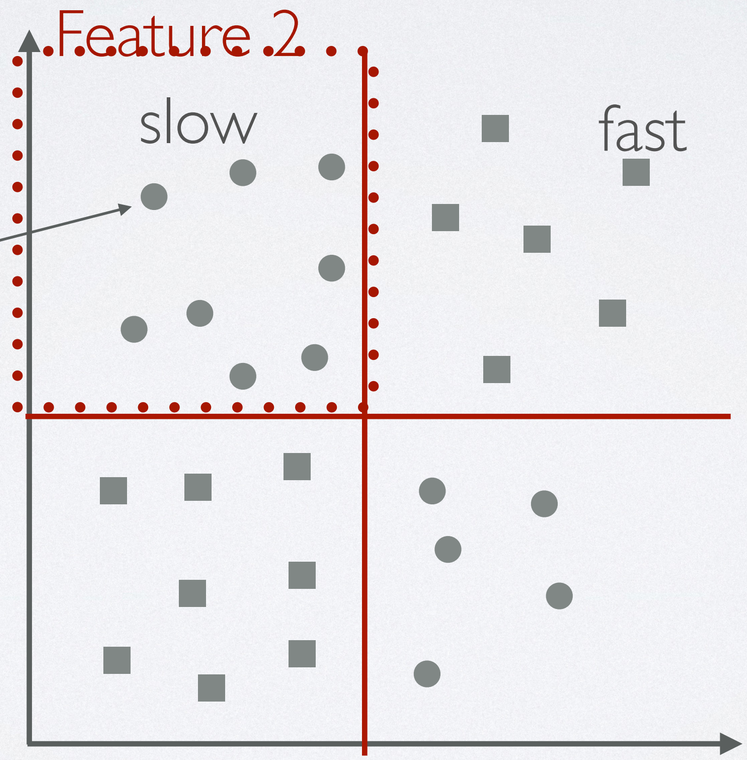
CLASSIFICATION



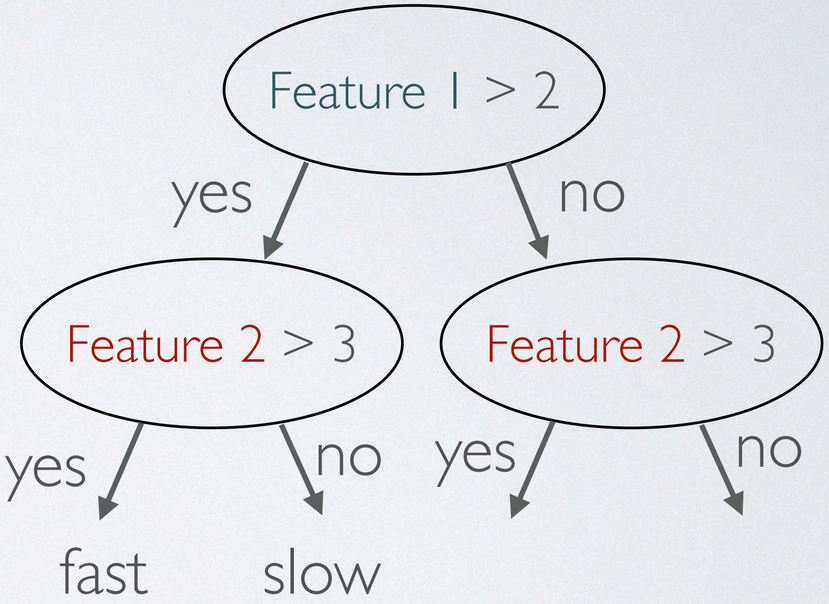
Possible implementation



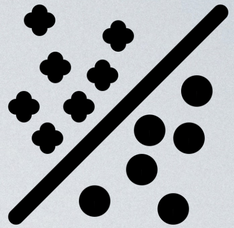
Feature vector



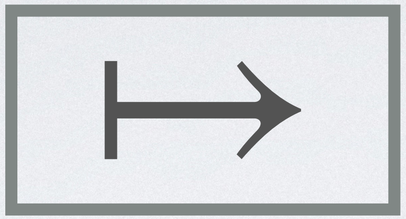
Decision trees



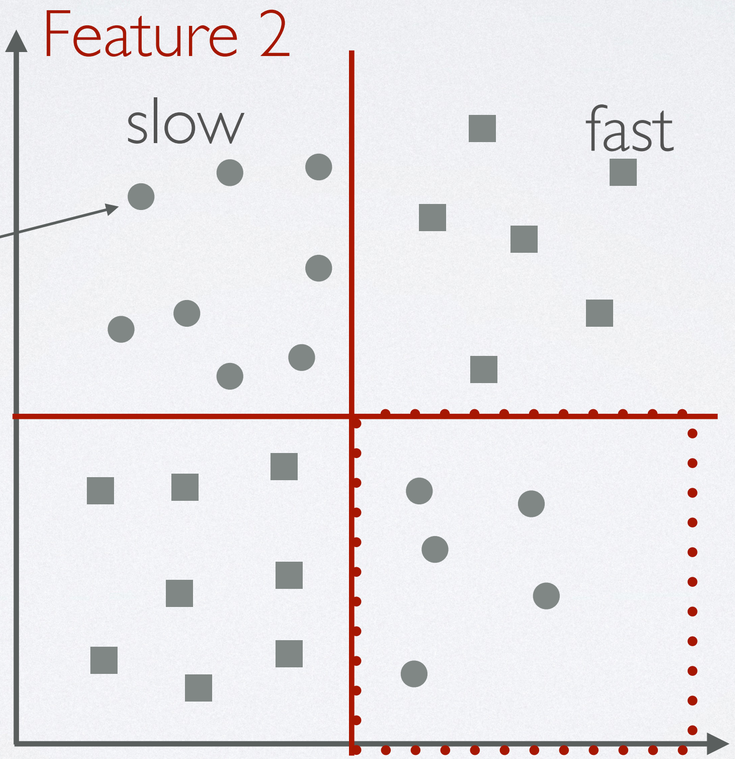
CLASSIFICATION



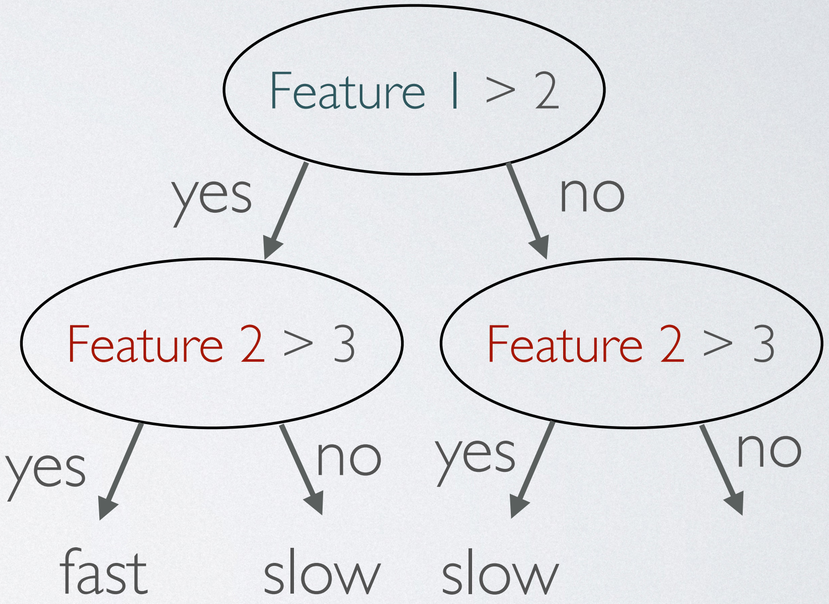
Possible implementation



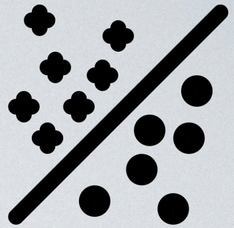
Feature vector



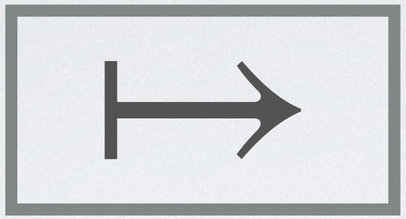
Decision trees



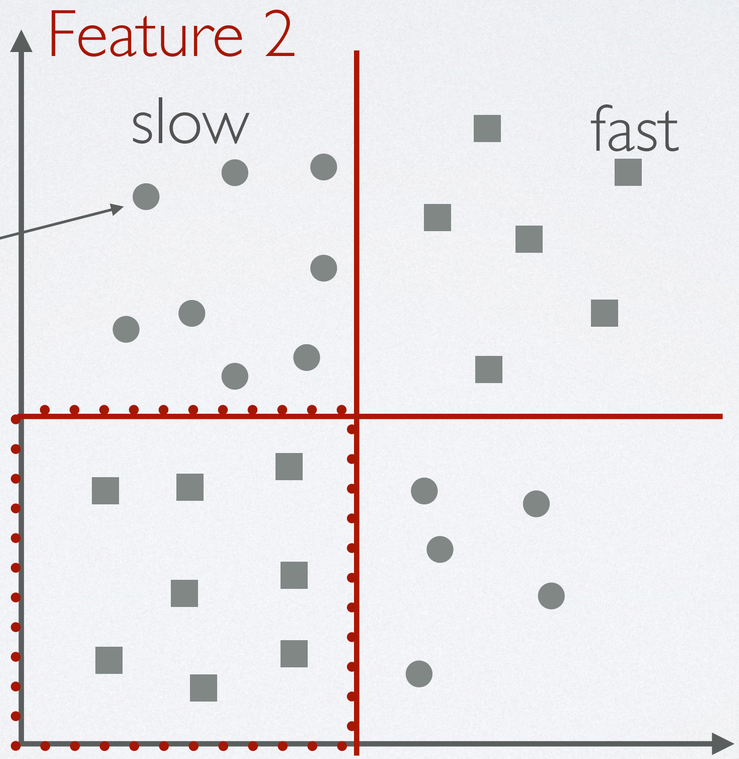
CLASSIFICATION



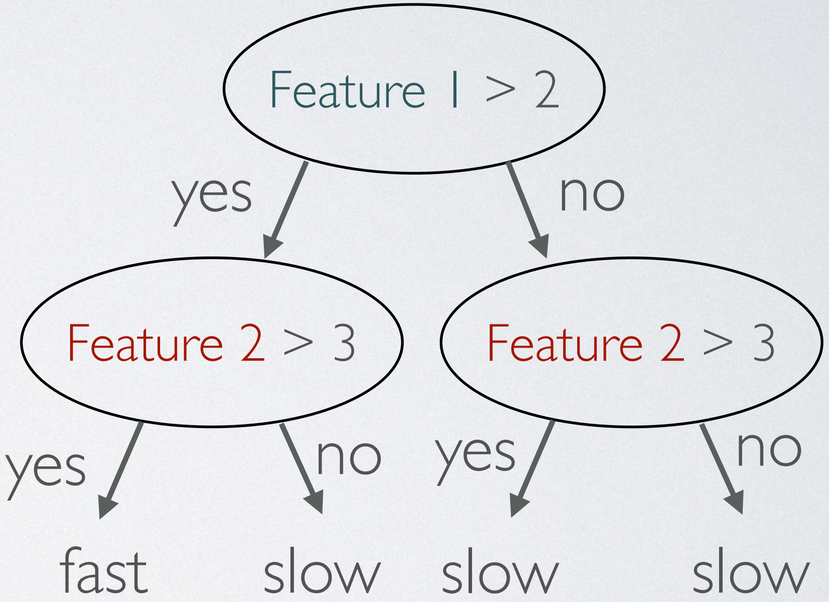
Possible implementation



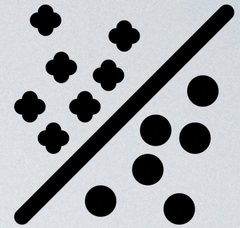
Feature vector



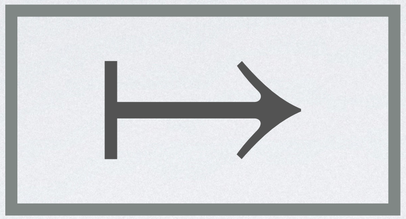
Decision trees



CLASSIFICATION

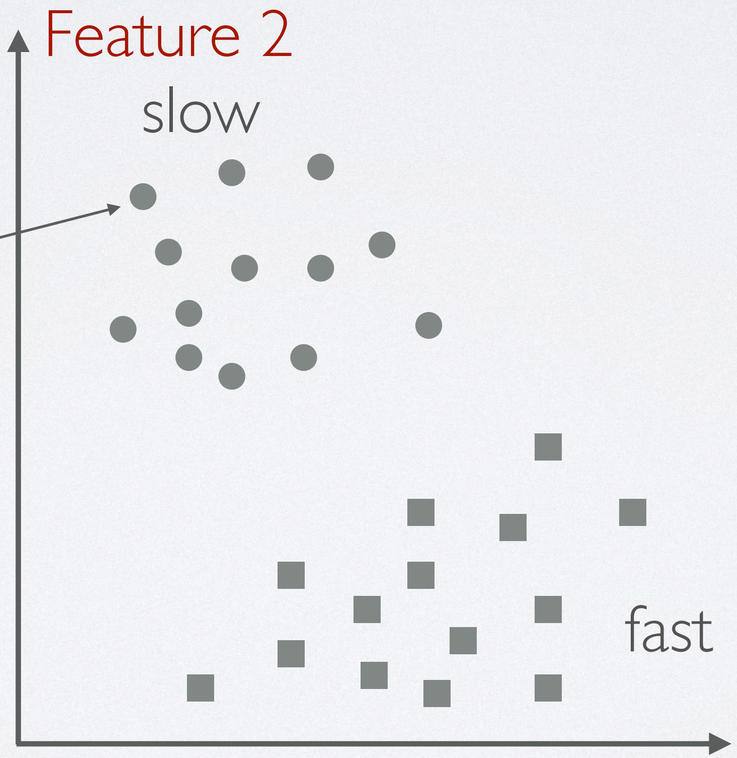


Possible implementation

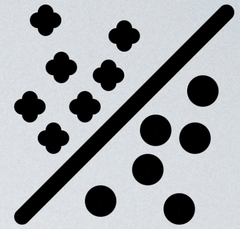


k-Nearest Neighbor

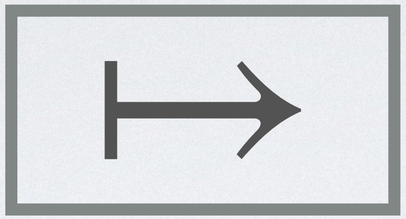
Feature vector



CLASSIFICATION



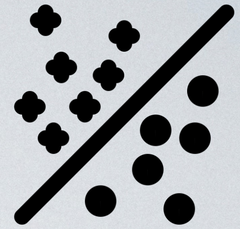
Possible implementation



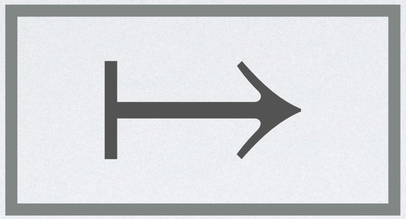
k-Nearest Neighbor

Feature vector

CLASSIFICATION



Possible implementation

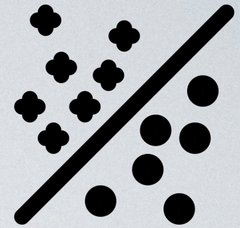


k-Nearest Neighbor

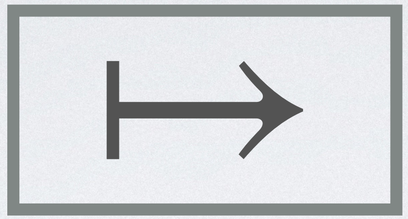
$k = 3$

Feature vector

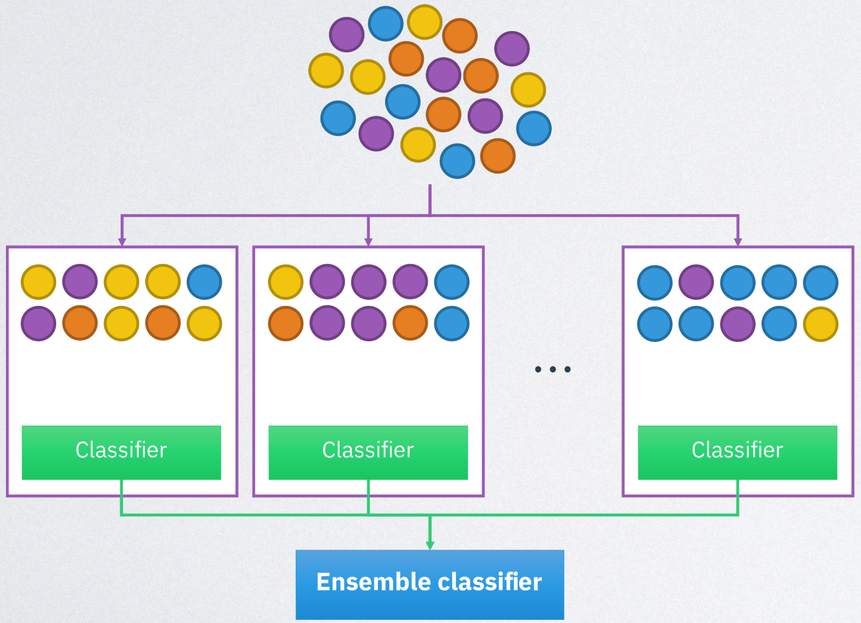
CLASSIFICATION



Possible implementation



Ensembles



https://upload.wikimedia.org/wikipedia/commons/c/c8/Ensemble_Bagging.svg

Original Data

Bootstrapping

Aggregating

Bagging

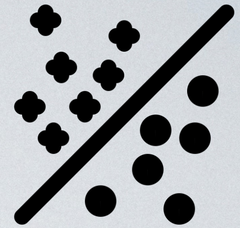
Bagging

Boosting

Random forrest

https://upload.wikimedia.org/wikipedia/commons/c/c8/Ensemble_Bagging.svg

CLASSIFICATION

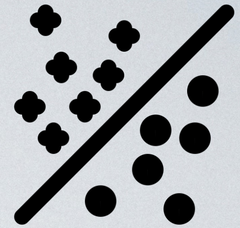


But do not worry there are also libraries that implement that for you


```
if classifier_name == "SVC":
    estimator = SVC(kernel="linear", class_weight='balanced', probability=True)  # , class_weight='balanced'
elif classifier_name == "DTC":
    estimator = DecisionTreeClassifier(criterion="entropy", splitter="best", class_weight='balanced')
elif classifier_name == "KNN":
    estimator = KNeighborsClassifier(n_neighbors=5, weights='uniform', algorithm='auto', metric='cosine', p=2)
```

see details: https://scikit-learn.org/stable/api/index.html

CLASSIFICATION



Why not deep learning?

- Amount of data
- Tabular data tend to work better with “classical” machine learning models

However, libraries exist: PyTorch or Keras

METRICS

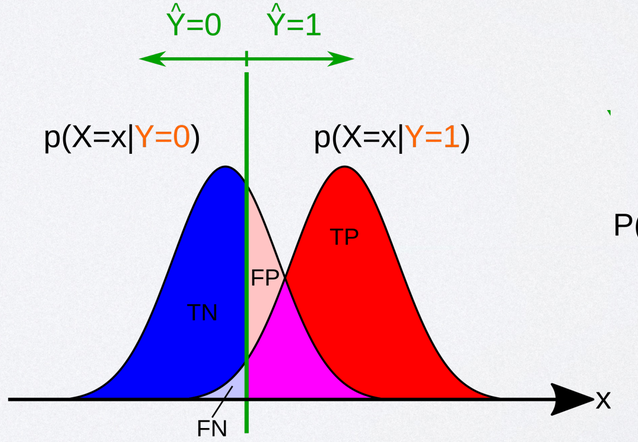
$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

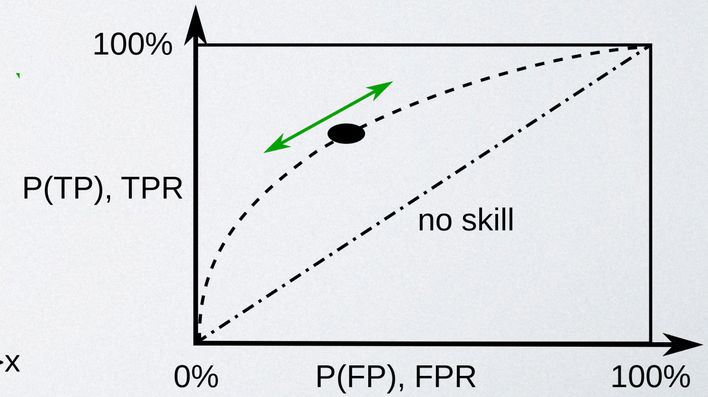
$$\text{F}_1\text{-score} = \frac{2TP}{2TP + FP + FN}$$

Receiver operating characteristics (ROC)

AUC = Area under the curve

Predicted Class \ True Class	Positive	Negative
Positive	True Positive (TP)	False Positive (FP)
Negative	False Negative (FN)	True negative (TN)





WHAT CAN YOU DO IF YOUR CLASSIFICATION DOES NOT WORK?

Feature selection

Feature importance

Analysis on when features make sense

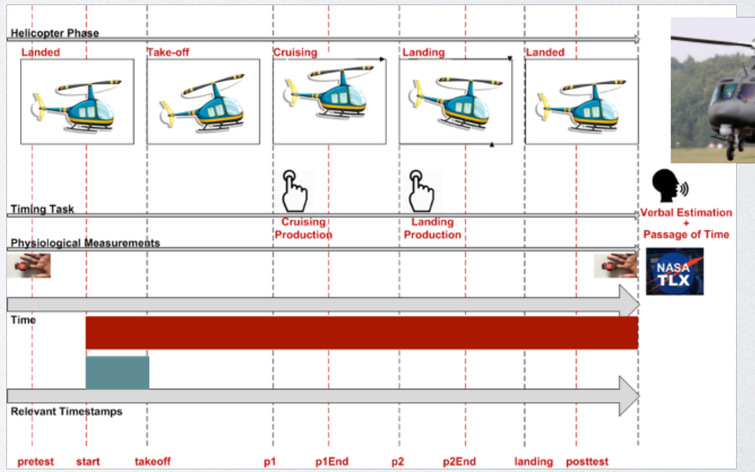
AutoML: automatically find the best model (select correct hyperparameters)

CASE STUDY

Air traffic controllers had to control one or two helicopters at the same time (english or greek)

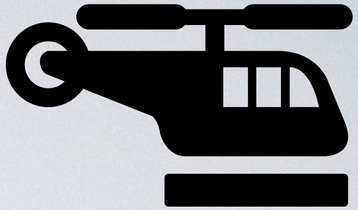
Afterwards they were asked how they did perceive their time

12 participants

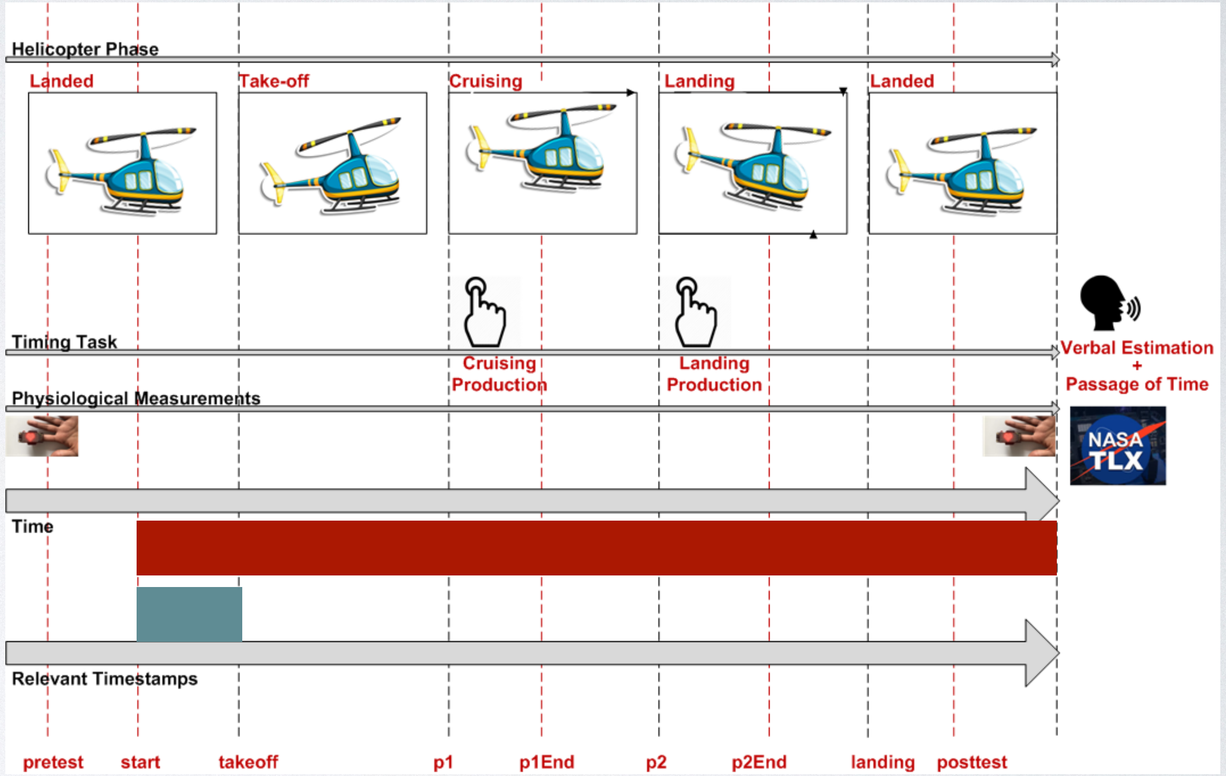


Aust et al. (2024) Automatic Classification of Subjective Time Perception Using Multi-modal Physiological Data of Air Traffic Controllers, arXiv preprint arXiv:2404.15213

PROCESSING FOR FLIGHT CONTROLLERS



- Used signals: PPG, EDA, thermopile (temperature)
- Data cleaning, automated by *HeartPy* and *NeuroKit2*
- *Feature extraction:*
  - *PPG: 13*
  - *EDA: 6*
  - *TMP: 5*
- Background subtraction: red_feature - blue_feature = used_feature



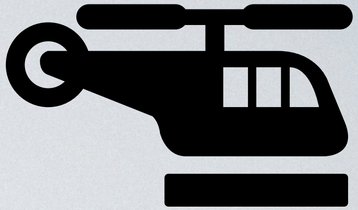
HeartPy:

P. van Gent, H. Farah, N. Nes, and B. van Arem, "Heart rate analysis for human factors: Development and validation of an open source toolkit for noisy naturalistic heart rate data," in *Proceedings of HUMANIST publications*, 2018.

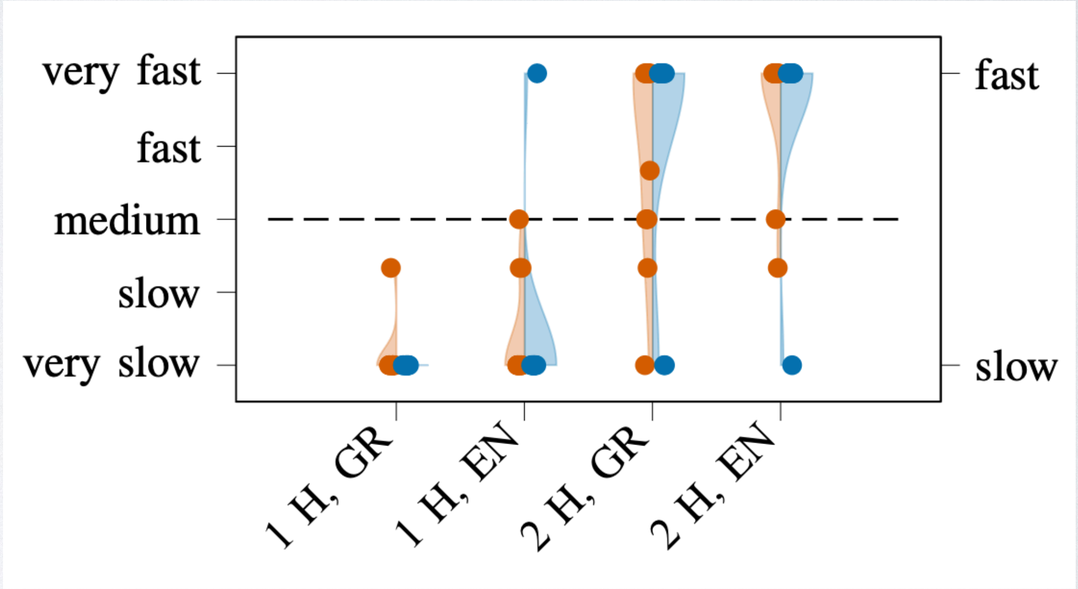
NeuroKit2:

D. Makowski, T. Pham, Z. J. Lau, J. C. Brammer, F. Lespinasse, H. Pham, C. Schölzel, and S. H. A. Chen, "NeuroKit2: A python toolbox for neurophysiological signal processing," *Behavior Research Methods*, vol. 53, no. 4, pp. 1689–1696, 2021.

PROCESSING FOR FLIGHT CONTROLLERS II

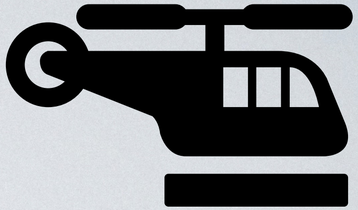


- Binary classification problem
  - Original labels: {very slow, …, very fast}
  - New labels: {slow, fast}



- #helicopters has larger influence than language

CLASSIFICATION FOR FLIGHT CONTROLLERS



- 11 classifiers (LOOCV)
- Feature selection method: none, SFS, RFECV, subsets
- SHAP analysis

SHAP:

S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 4765–4774.

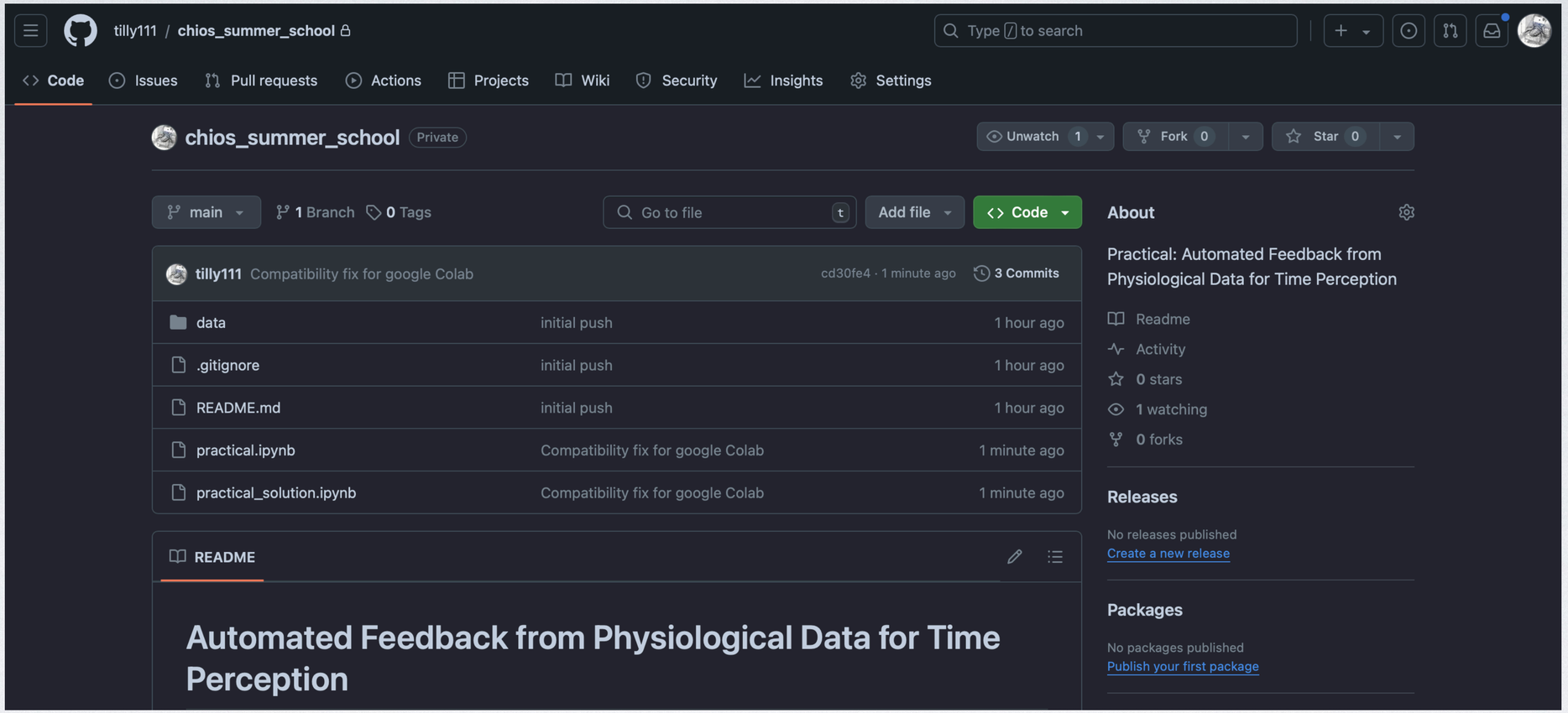
TABLE I

OVERVIEW OF THE MEAN ACCURACIES OVER ALL PARTICIPANTS OBTAINED BY OUR SELECTED STATE-OF-THE-ART CLASSIFIERS USING VARIOUS FEATURE SELECTION ALGORITHM AND MIN-MAX NORMALIZATION FOR ALL PARTICIPANTS. THE "*" MARKED CLASSIFIERS CAN HAVE VARYING RESULTS DUE TO THEIR IMPLEMENTATION.

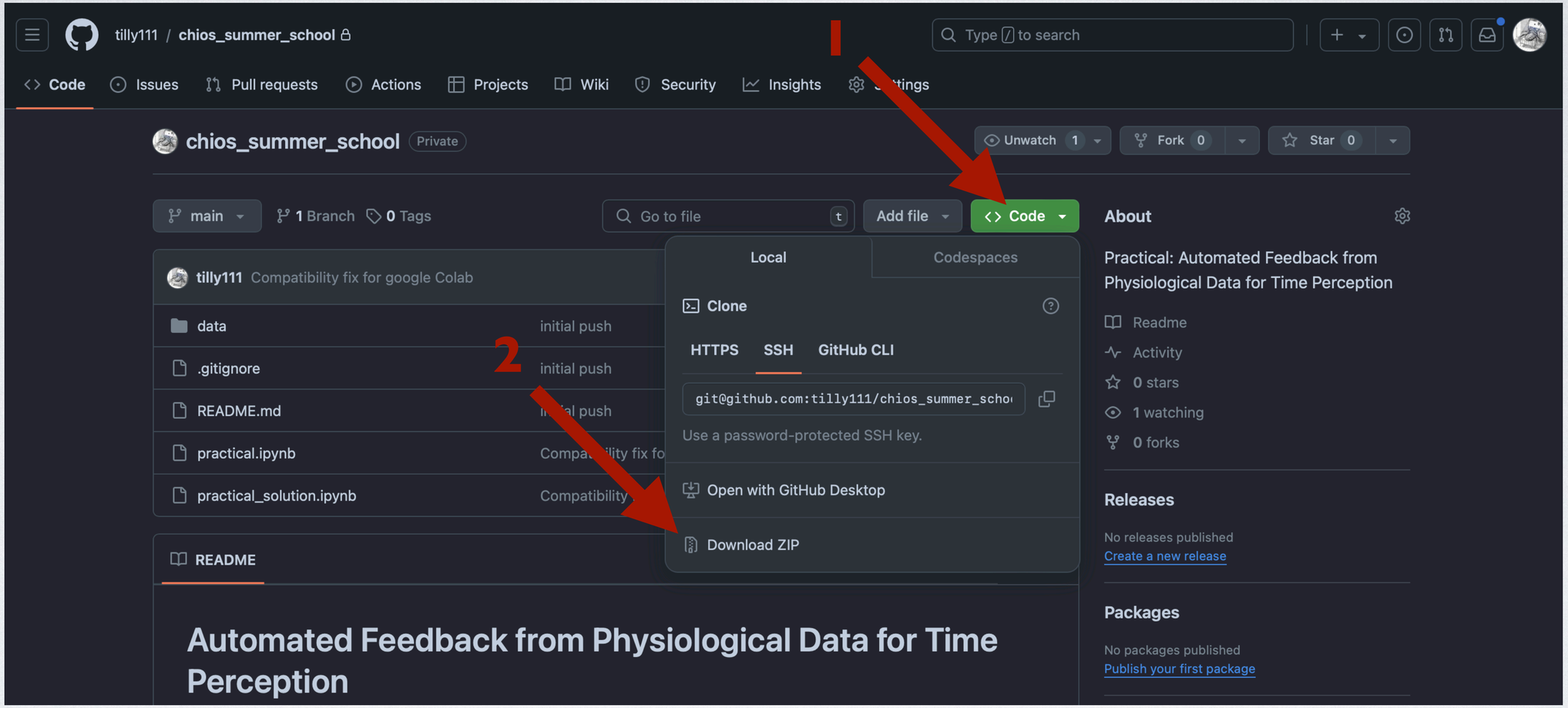
Classifier	None	SFS	RFECV	PPG	PPG+EDA
SVC	**0.7917**	**0.7708**	0.7083	0.6875	**0.7917**
DTC*	0.5417	0.5625	0.5625	0.5833	0.5625
KNN	0.75	0.6458	N.A.	0.7292	0.6458
GNB	0.6042	0.5833	N.A.	0.6667	0.6042
LR	0.5625	0.6875	0.6667	0.6875	0.7083
LDA	0.7708	0.6458	**0.75**	**0.7708**	**0.7917**
QDA	0.625	0.5625	N.A.	0.6042	0.625
RF*	0.6875	0.5208	0.7083	0.6875	0.625
GB*	0.6458	0.5417	0.625	0.6042	0.5833
AB	0.6458	0.5	0.5833	0.6458	0.6458
XGB	0.5625	0.4167	0.6458	0.6667	0.5833

HANDS-ON

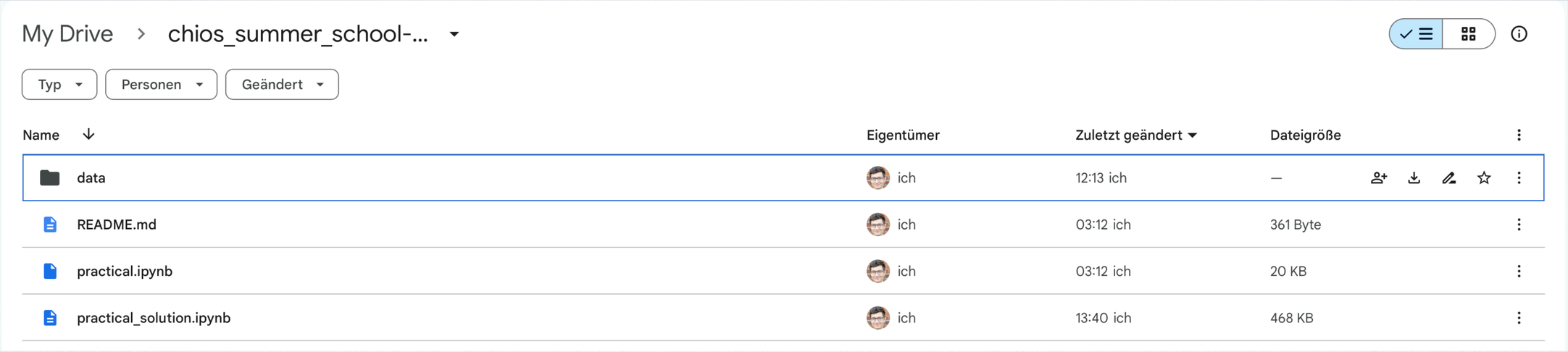
1. Go to: https://github.com/tilly111/chios_summer_school



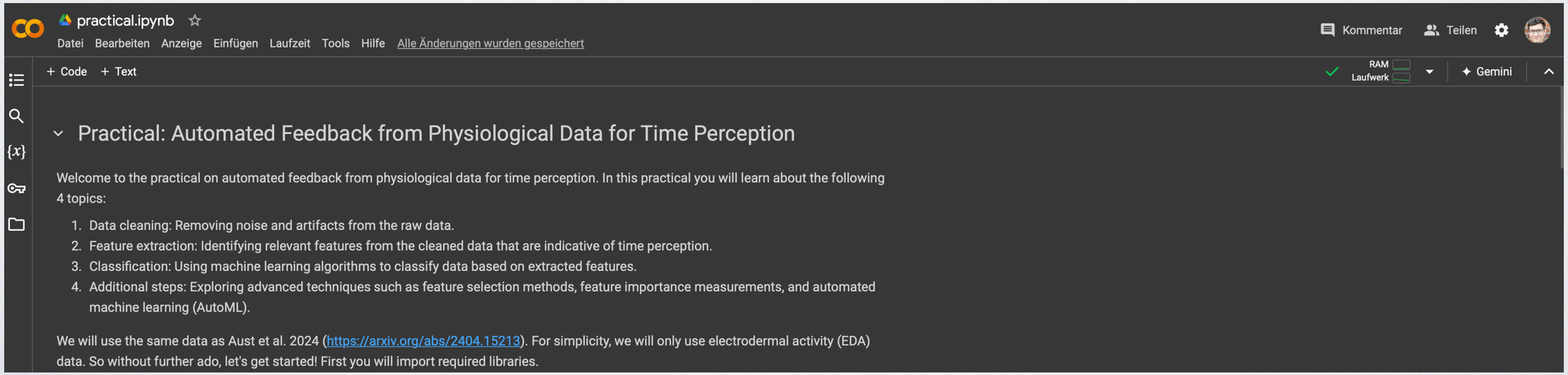
2. Download the project



3. Upload the project to your google drive



4. Double click practical.ipynb (or open it with google colab)



LET'S CODE TOGETHER