BOOSTING
Concept, AdaBoost and Exercises

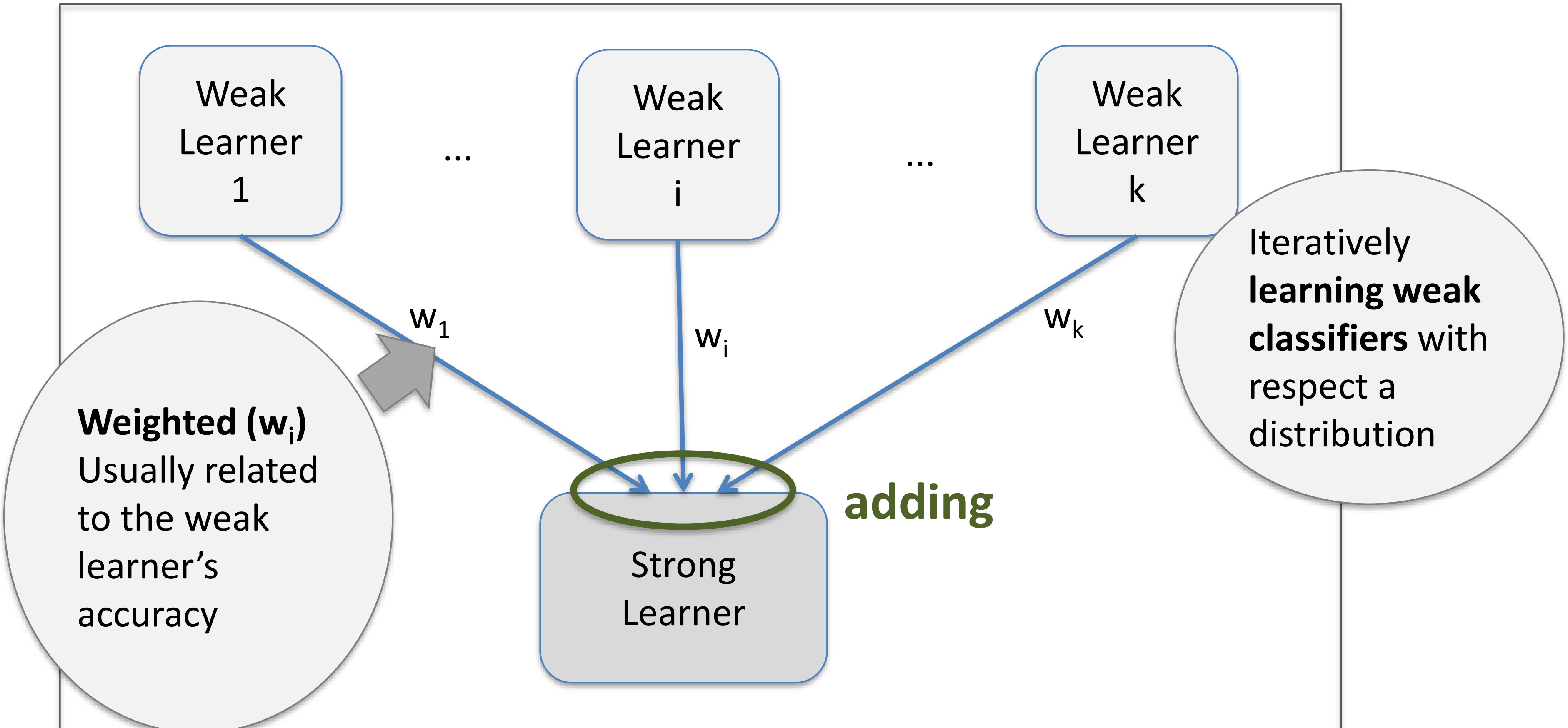
Elisabet Golobardes & Ángel Berián
La Salle – Universitat Ramon Llull

v2020.02.27

BOOSTING. Concept

- A machine learning ensemble meta-algorithm
- For primarily reducing bias and also variance
- In supervised learning
- A family of machine learning algorithms that convert weak learners to strong ones.

BOOSTING. Algorithms (1/2)



BOOSTING. A boost classifier

TRAINING

A boost classifier is a classifier in the form

$$F_T(x) = \sum_{t=1}^{T} f_t(x)$$

Where each $\boldsymbol{f_t}$ is a **weak learner** that

- takes $\boldsymbol{x}$ as input
- and returns a value indicating the **class** of the object.

Each weak learner produces an **output hypothesis**, $\boldsymbol{h(x_i)}$, for each sample in the training set. At each **iteration *t***, a weak learner is selected and assigned a **coefficient $\boldsymbol{\alpha_t}$** such that **the sum training error $\mathbf{E_t}$** of the resulting *t*-stage boost classifier is minimized.

$$E_t = \sum_i E[F_{t-1}(x_i) + f_t(x)]\,;where\;\; f_t(x) = \alpha_t h(x_i)$$

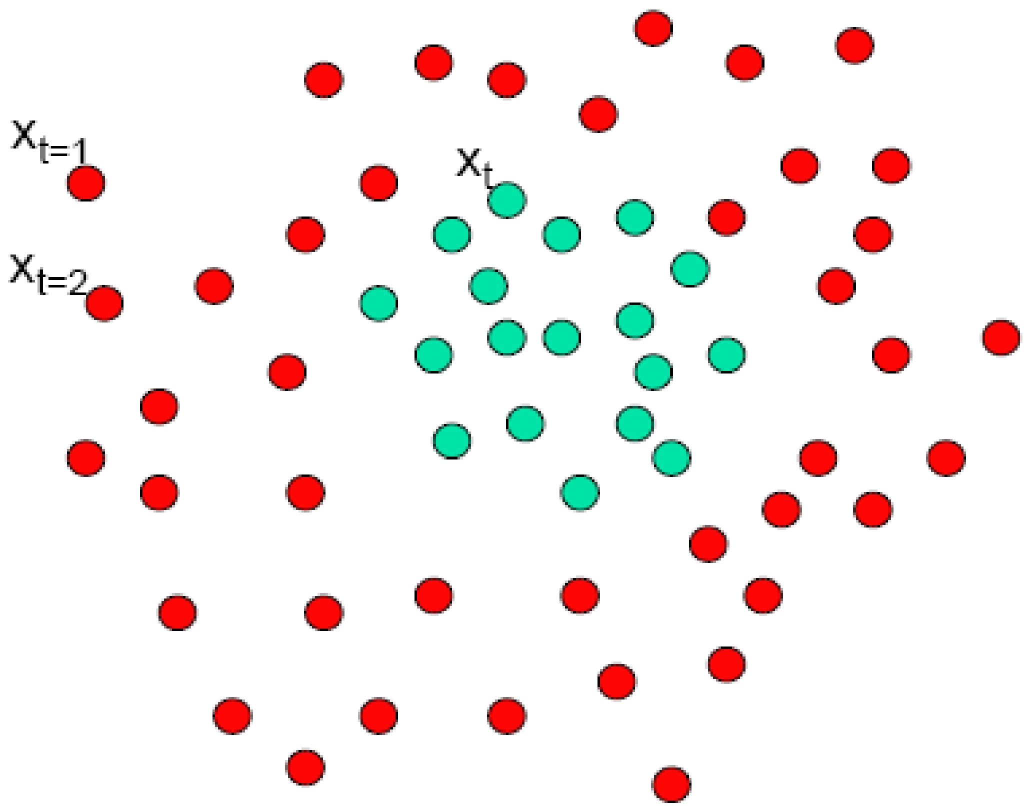
Here $F_{t-1}(x)$ is the boosted classifier that has been built up to the previous stage of training, ***E(F)*** is some **error function** and $f_t(x)$ is the weak learner that is being considered for addition to the final classifier.

WEIGHTING

At each iteration of the training process, a **weight** $\boldsymbol{w_{i,t}}$ is assigned to each sample in the training set equal to the current error $E\big(F_{t-1}(x_i)\big)$ on that sample.

These weights can be used to inform the training of the weak learner.

Boosting - Example



Each data point has

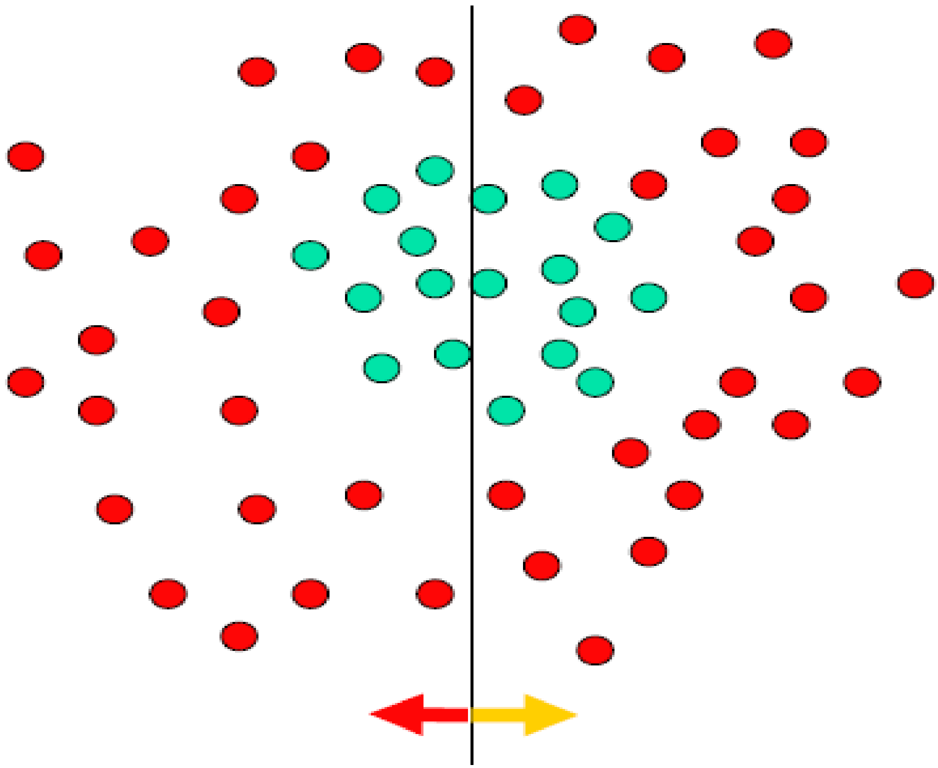
a class label:

$$y_t = \begin{cases} +1 \\ -1 \end{cases}$$

and a weight:

$$w_t = 1$$

Boosting - Example



Each data point has a class label:

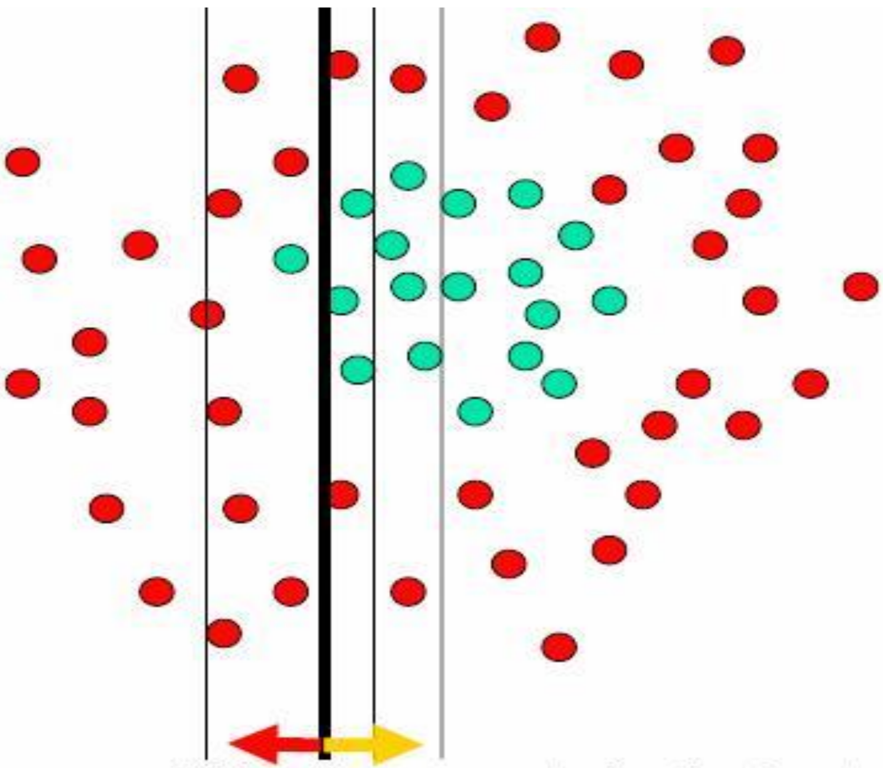
$$y_t = \begin{cases} +1 & (\text{red}) \\ -1 & (\text{green}) \end{cases}$$

and a weight:

$w_t = 1$

h => p(error) = 0.5 it is at chance

Boosting - Example



Each data point has

a class label:

$$y_t = \begin{cases} +1 \ (\ ) \\ -1 \ (\ ) \end{cases}$$

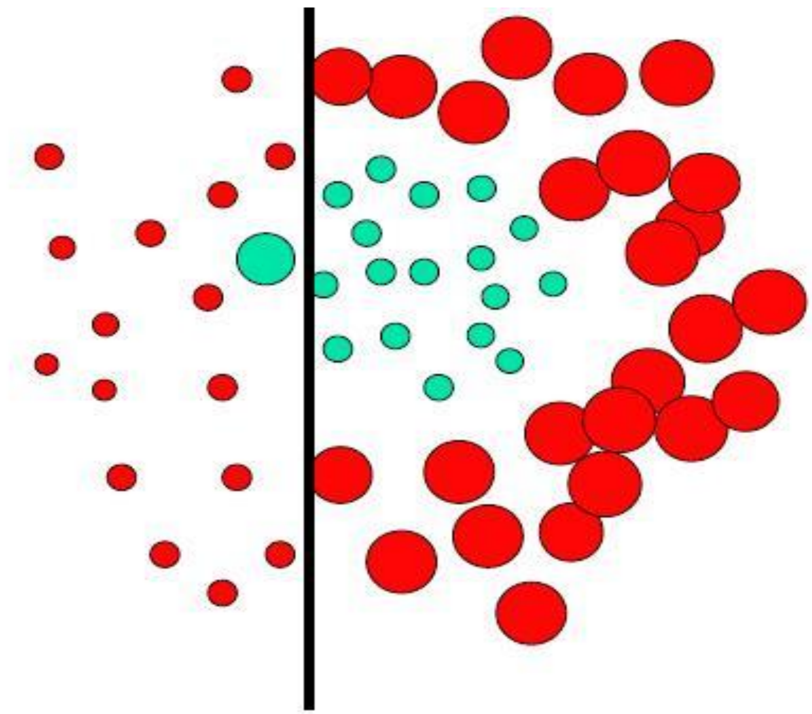
and a weight:

$w_t = 1$

This one seems to be the best

This is a ‘**weak classifier**’: It performs slightly better than chance.

Boosting - Example



Each data point has

a class label:

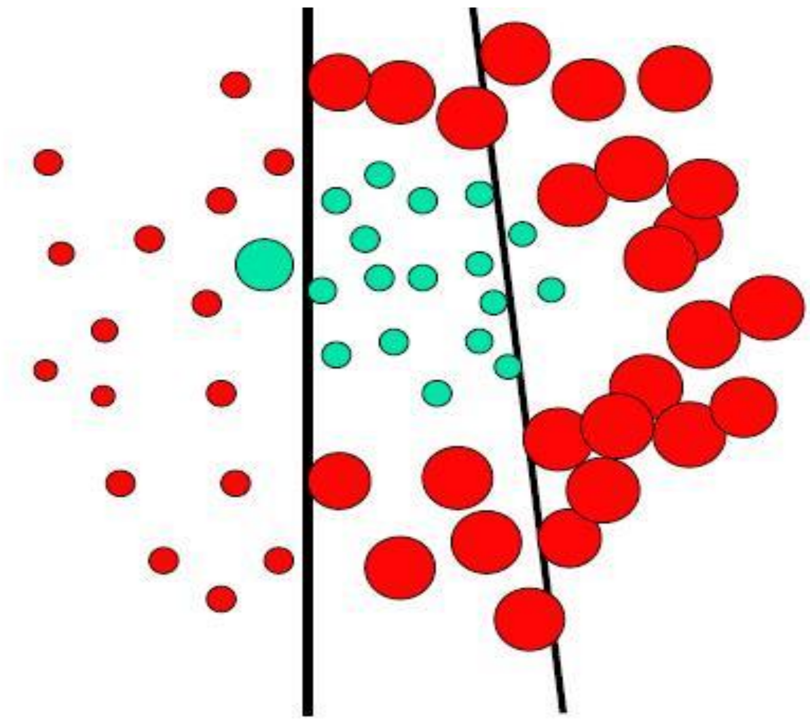
$$y_t = \begin{cases} +1 & (\text{red}) \\ -1 & (\text{green}) \end{cases}$$

We update the weights:

$$w_t \leftarrow w_t \exp\{-y_t H_t\}$$

We set a new problem for which the previous weak classifier performs at chance again

Boosting - Example



Each data point has

a class label:

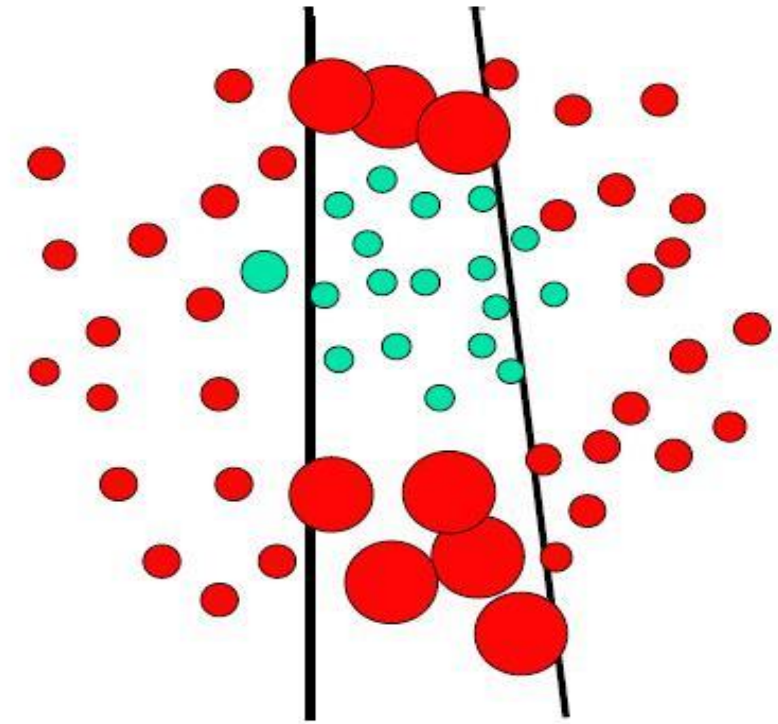
$$y_t = \begin{cases} +1 \ (\text{red}) \\ -1 \ (\text{green}) \end{cases}$$

We update the weights:

$$w_t \leftarrow w_t \exp\{-y_t H_t\}$$

We set a new problem for which the previous weak classifier performs at chance again

Boosting - Example



Each data point has a class label:

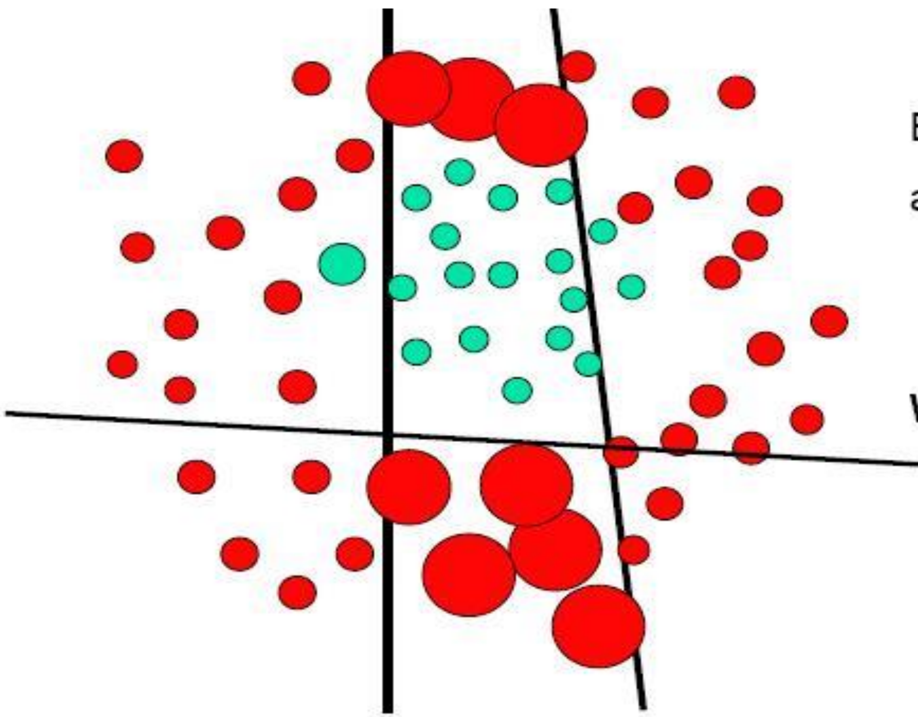
$$y_t = \begin{cases} +1 & (\text{red}) \\ -1 & (\text{green}) \end{cases}$$

We update the weights:

$$w_t \leftarrow w_t \exp\{-y_t H_t\}$$

We set a new problem for which the previous weak classifier performs at chance again

Boosting - Example



Each data point has

a class label:

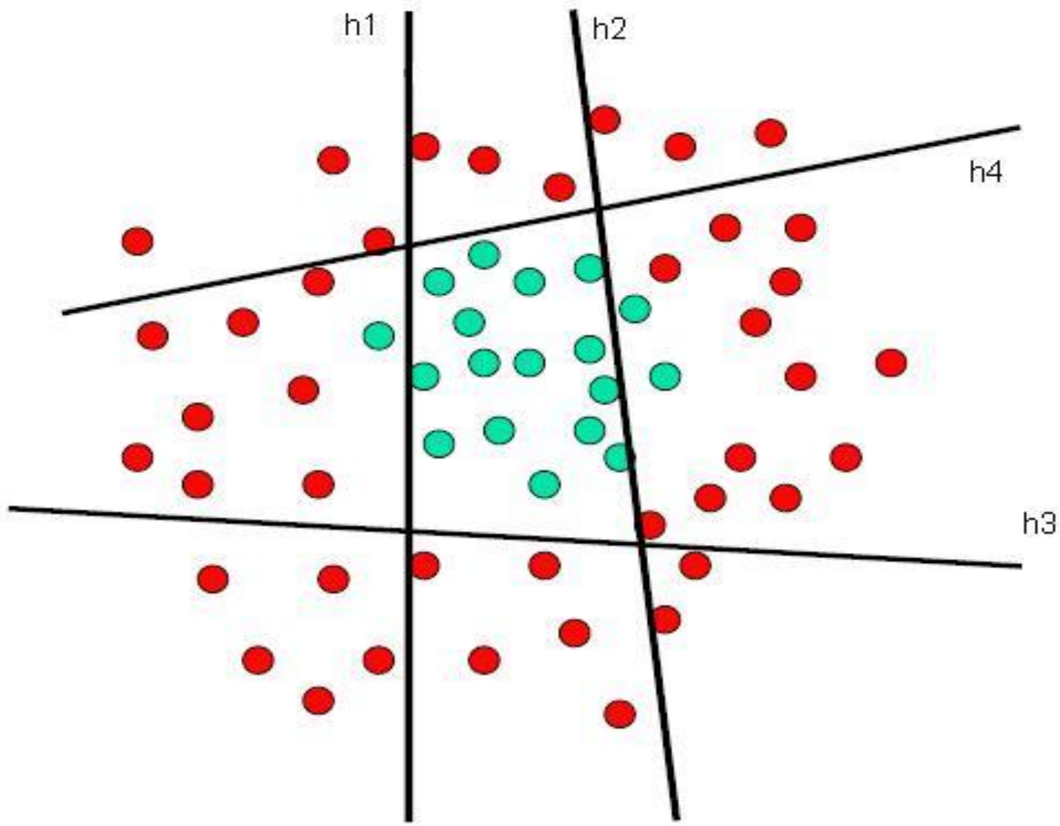
$$y_t = \begin{cases} +1 \ (\bullet) \\ -1 \ (\bullet) \end{cases}$$

We update the weights:

$$w_t \leftarrow w_t \exp\{-y_t H_t\}$$

We set a new problem for which the previous weak classifier performs at chance again

Boosting - Example



The strong (non- linear) classifier is built as the combination of all the weak (linear) classifiers.

BOOSTING. Algorithms (2/2)

- There are many boosting algorithms:
  - **AdaBoost** (Freund & Schapire, 1996)
  - LPBoost
  - TotalBoost
  - BrownBoost
  - **XGBoost**
  - MadaBoost
  - LogitBoost
  - **CatBoost**
  - **LightGBM**

- AnyBoost framework: All of them boosting performs **gradient descent** in a function space using a **convex cost function**.

- The main variation between many boosting algorithms is their **method of weighting training data points** and **hypotheses**.

AdaBoost

- AdaBoost (Adaptative Boosting)
  - Yoav Freund & Robert E. Schapire (1996);
  - Won the 2003 Gödel Prize for their work

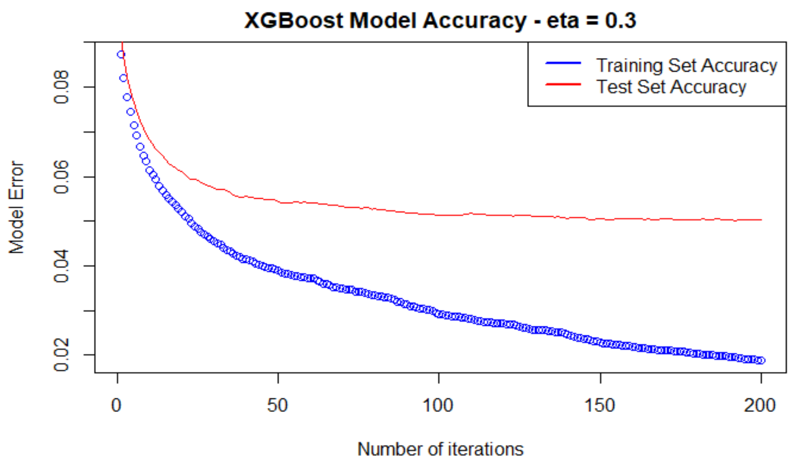
- Linear classifier with all its desirable properties
- Has good generalization properties
- Is a feature selector with a principled strategy (minimization of upper bound on empirical error)
- Close to sequential decision making

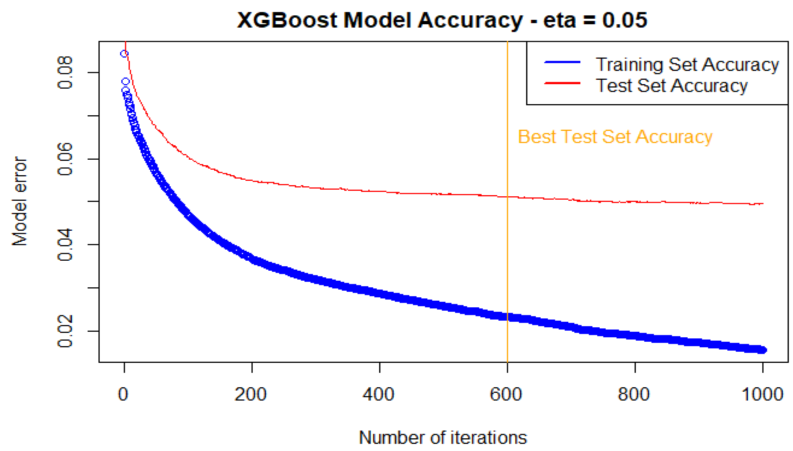
AdaBoost – Pros and Cons

- Pros:
  - Very simple to implement
  - Fairly good generalization
  - The prior error need not be known ahead of time
- Cons:
  - Suboptimal solution
  - Can over fit in presence of noise

Model Learning Rates

- Learning rates help regularizing the model and typically, the lower the value it is the better the model will learn with multiple iterations avoiding overfitting.





References

- **[Freund & Schapire, 1996]** Freund, Yoav; Schapire, Robert E (1996). *”Experiments with a New Boosting Algorithm*". Machine Learning: Proceedings of the Thirteenth International Conference.

- **[Freund & Schapire, 1997]** Freund, Yoav; Schapire, Robert E (1997). "*A decision-theoretic generalization of on-line learning and an application to boosting*". Journal of Computer and System Sciences, 55: 119-139. Article No. SS971504.

Intro to XGBoost, CATBoost & LightGBM

Elisabet Golobardes & Ángel Berián
La Salle – Universitat Ramon Llull

v2020.02.27

State of the art Boosting models

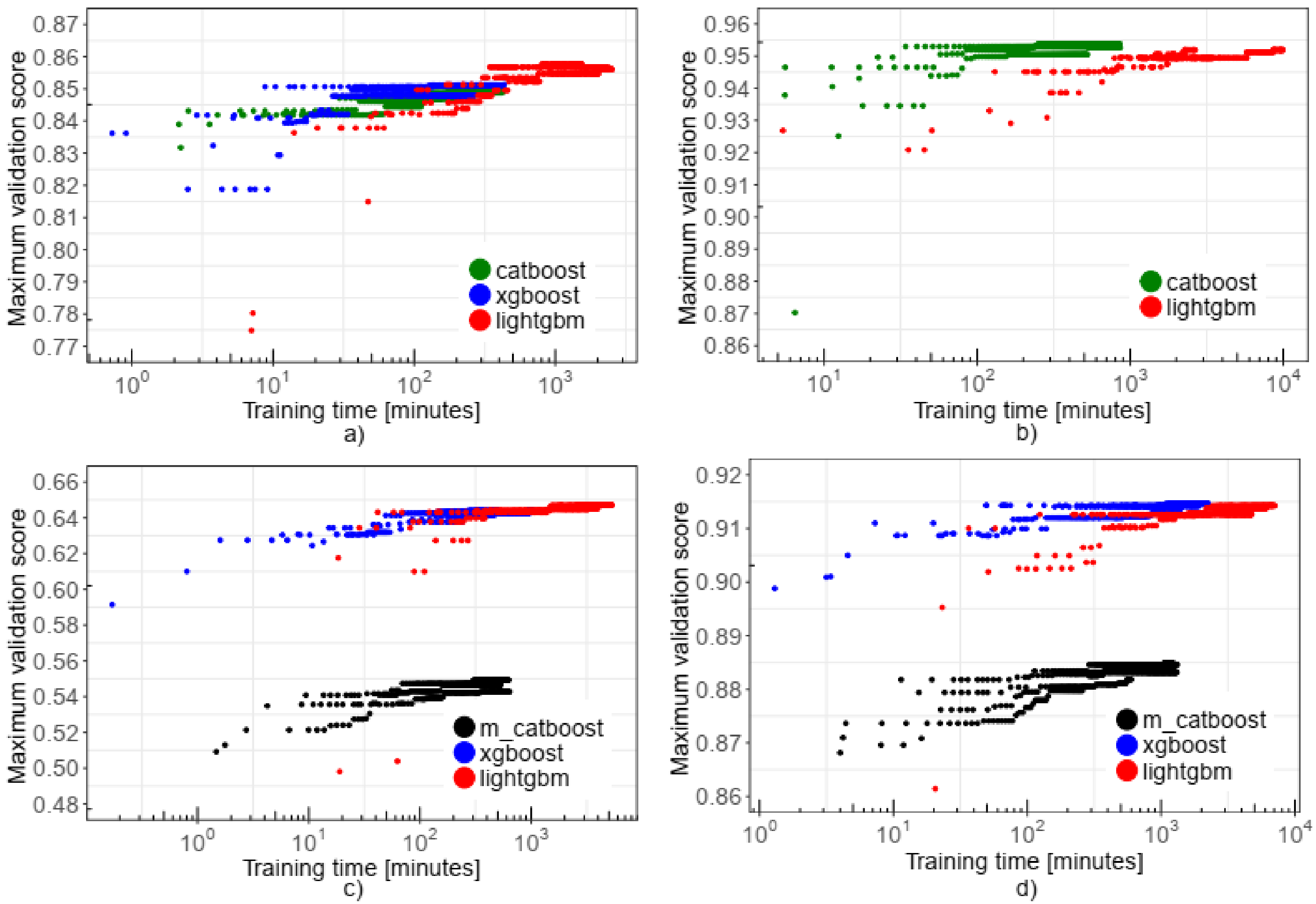
- XGBoost (eXtreme Gradient Boosting)
  - Tianqi Chen (2014);
- LightGBM (Light Gradient Boosting Machine)
  - Microsoft (2017);
- CATBoost (CATegory Boosting)
  - Yandex (2017);

Model tuning Comparison

Function	XGBoost	CatBoost	Light GBM
Important parameters which control overfitting	1. **learning_rate or eta** – optimal values lie between 0.01-0.2 2. **max_depth** 3. **min_child_weight:** similar to min_child leaf; default is 1	1. **Learning_rate** 2. **Depth** - value can be any integer up to 16. Recommended - [1 to 10] 3. No such feature like min_child_weight 4. **l2-leaf-reg**: L2 regularization coefficient. Used for leaf value calculation (any positive integer allowed)	1. **learning_rate** 2. **max_depth**: default is 20. Important to note that tree still grows leaf-wise. Hence it is important to tune **num_leaves** (number of leaves in a tree) which should be smaller than 2^(max_depth). It is a very important parameter for LGBM 3. **min_data_in_leaf**: default=20, alias= min_data, min_child_samples
Parameters for categorical values	Not Available	1. **cat_features**: It denotes the index of categorical features 2. **one_hot_max_size**: Use one-hot encoding for all features with number of different values less than or equal to the given parameter value (max – 255)	1. **categorical_feature:** specify the categorical features we want to use for training our model
Parameters for controlling speed	1. **colsample_bytree**: subsample ratio of columns 2. **subsample**: subsample ratio of the training instance 3. **n_estimators**: maximum number of decision trees; high value can lead to overfitting	1. **rsm:** Random subspace method. The percentage of features to use at each split selection 2. No such parameter to subset data 3. **iterations**: maximum number of trees that can be built; high value can lead to overfitting	1. **feature_fraction:** fraction of features to be taken for each iteration 2. **bagging_fraction:** data to be used for each iteration and is generally used to speed up the training and avoid overfitting 3. **num_iterations:** number of boosting iterations to be performed; default=100

Source: https://towardsdatascience.com/catboost-vs-light-gbm-vs-xgboost-5f93620723db

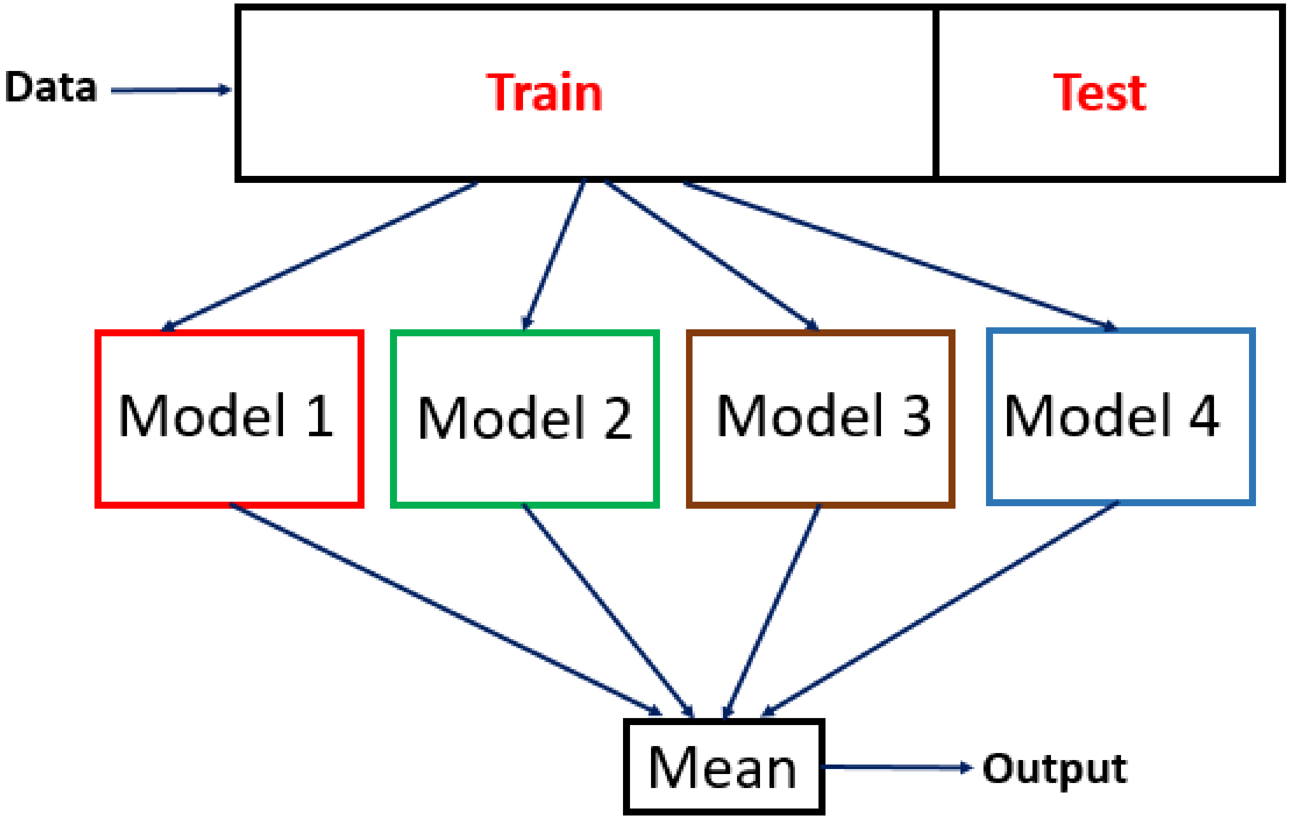
Model training time Comparison



Max validation score vs. Total HPO runtime (a) Higgs (b) Epsilon (c) Microsoft (d) Yahoo.

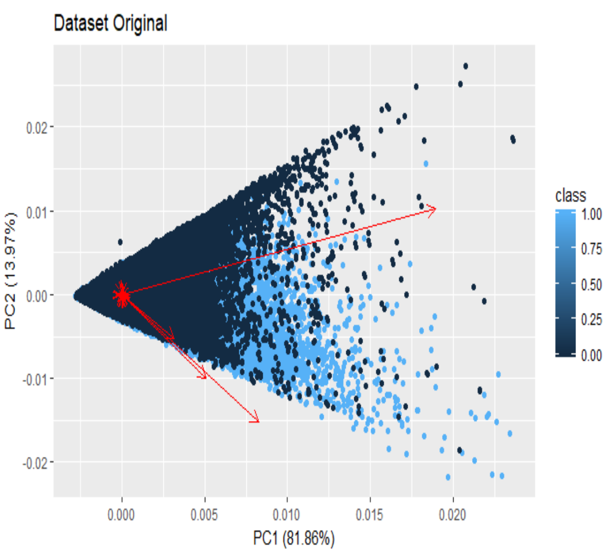
Source: https://arxiv.org/abs/1809.04559

Ensemble Methods

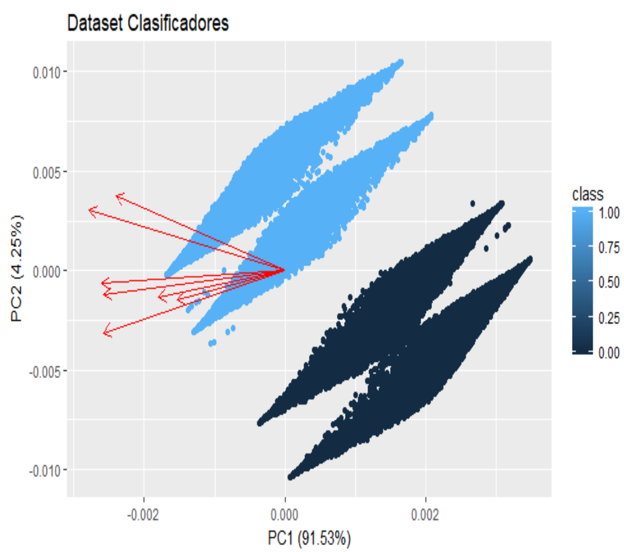


Feature engineering (FE)

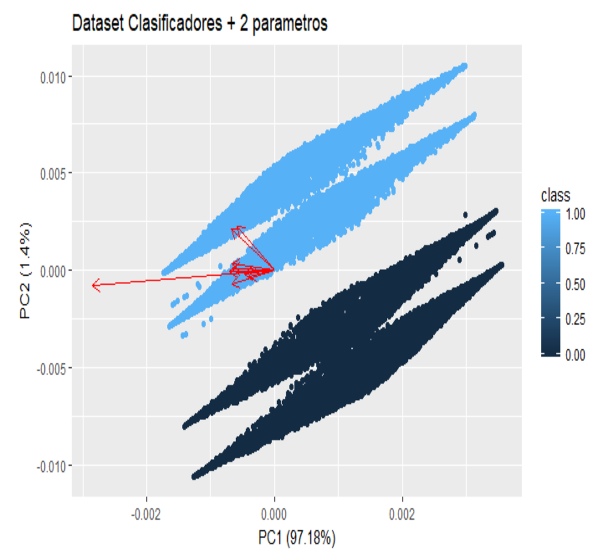
DATASET ORIGINAL



DATASET DE CLASIFICADORES



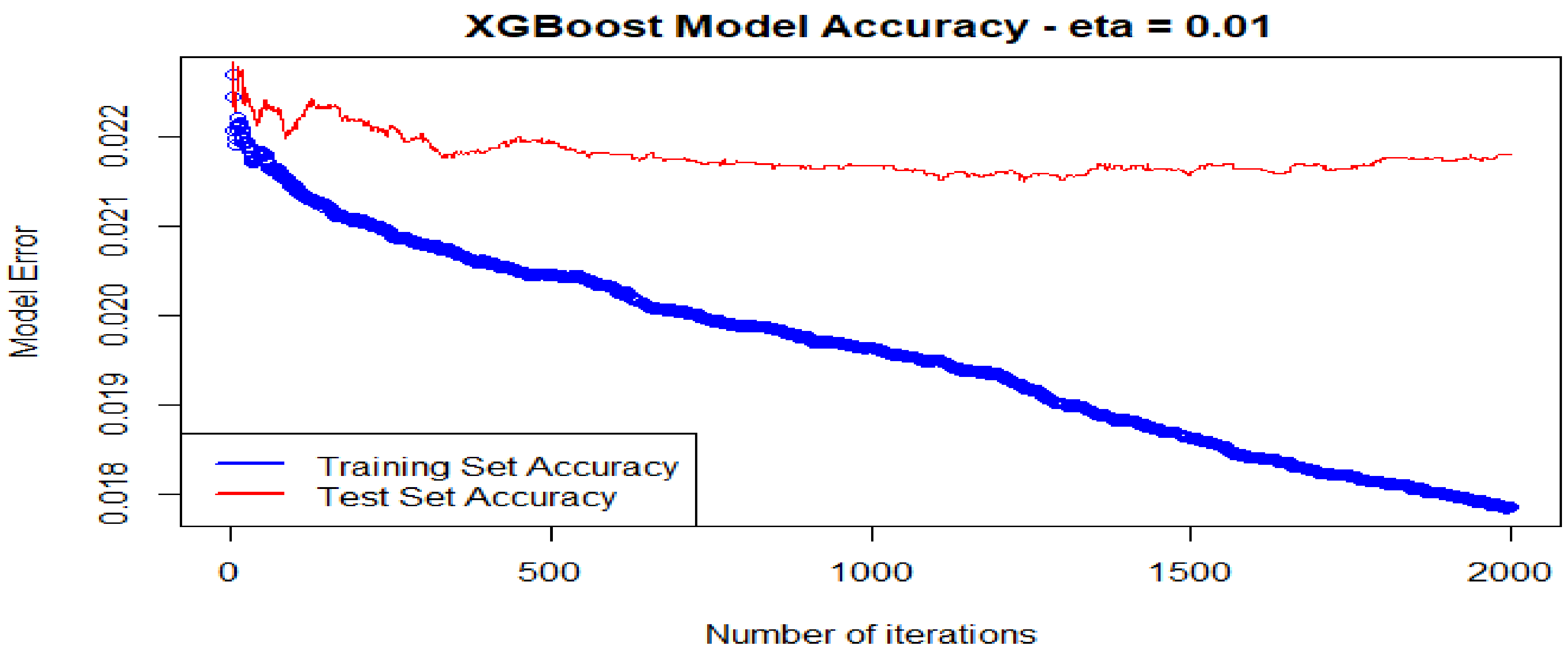
CLASIFICADORES + FE



Mediante el modelo de varias capas generamos una mayor separación entre las clases

Fuente: Elaboración Propia

Prevención de sobre-entrenamiento en el modelo final



Con un aprendizaje de 0.01 el modelo no mejora los resultados por encima de 1200 iteraciones

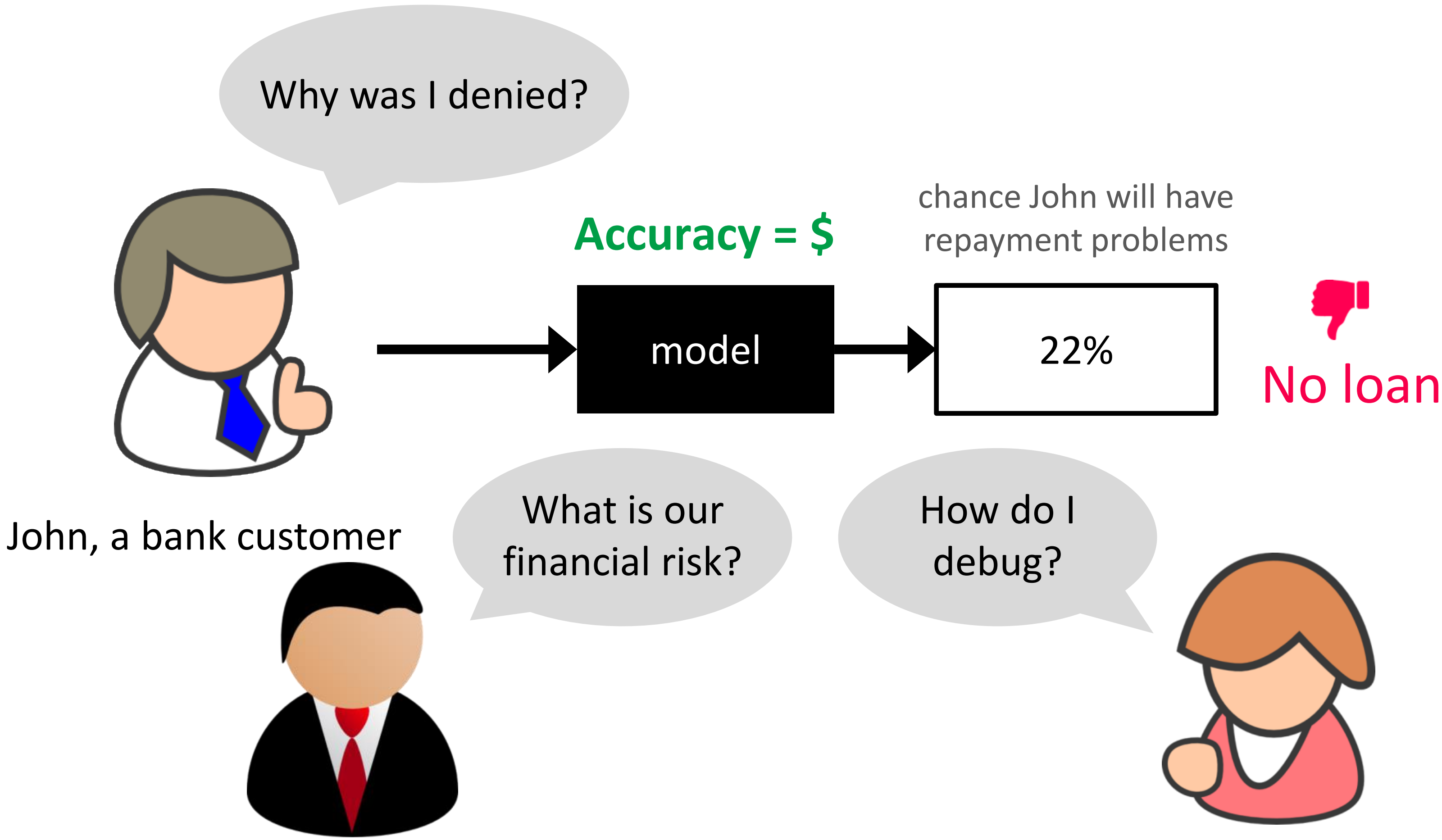
Fuente: Elaboración Propia

Shapley Values in Machine Learning

Elisabet Golobardes & Ángel Berián
La Salle – Universitat Ramon Llull
v2020.02.27

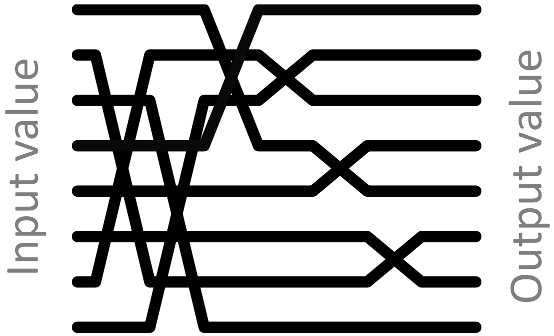
Explainable AI in practice

Model development

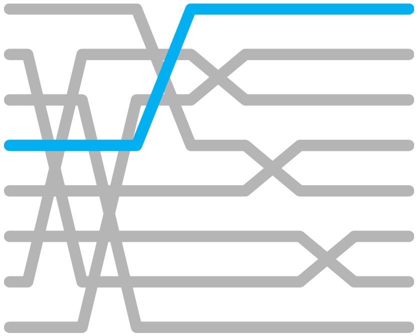

Why was I denied?
Accuracy = $
chance John will have repayment problems
model
22%
No loan
John, a bank customer
What is our financial risk?
How do I debug?

	Interpretable	Accurate
Complex model	←	?
Simple model	? →	

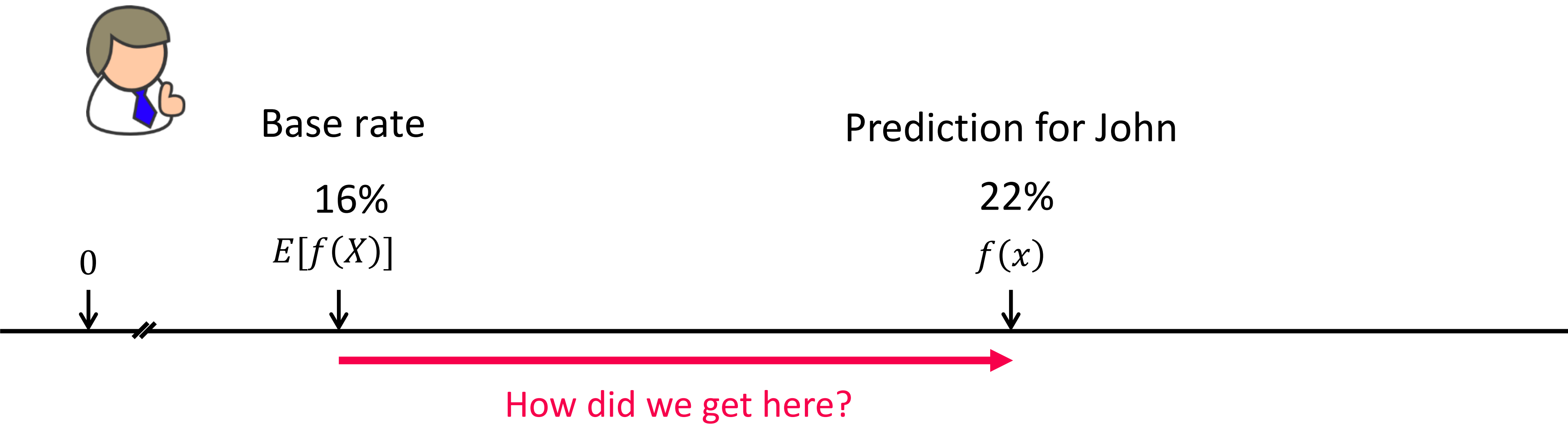
Interpretable or accurate: **choose one**.



Complex models are
inherently complex!

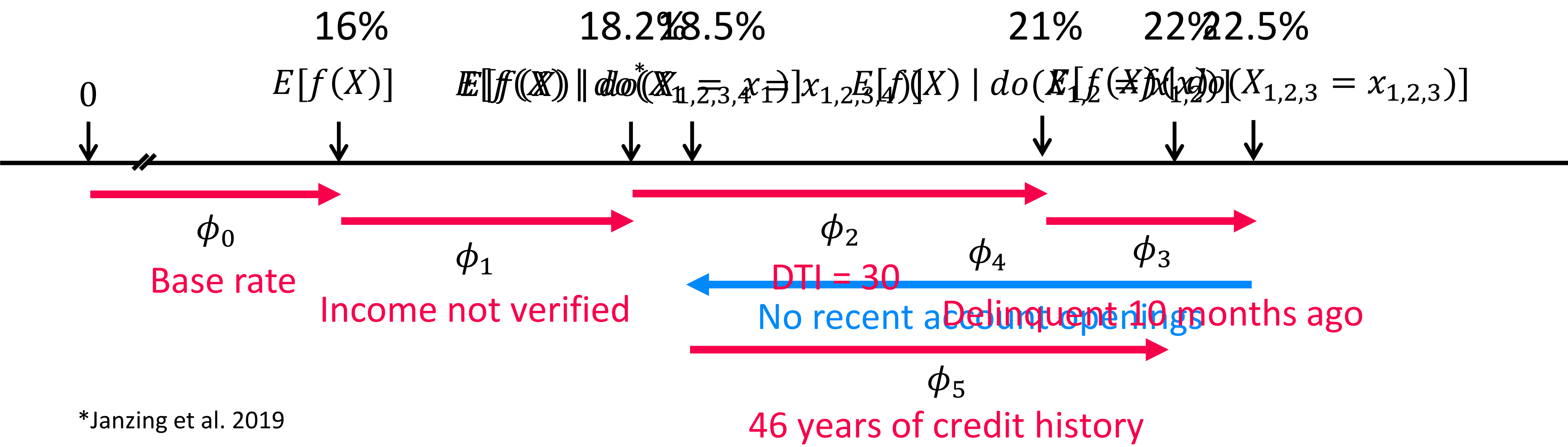


But a single prediction involves only
a small piece of that complexity.



Base rate
Prediction for John
16%
22%
0
$E[f(X)]$
$f(x)$
How did we get here?

The order matters!

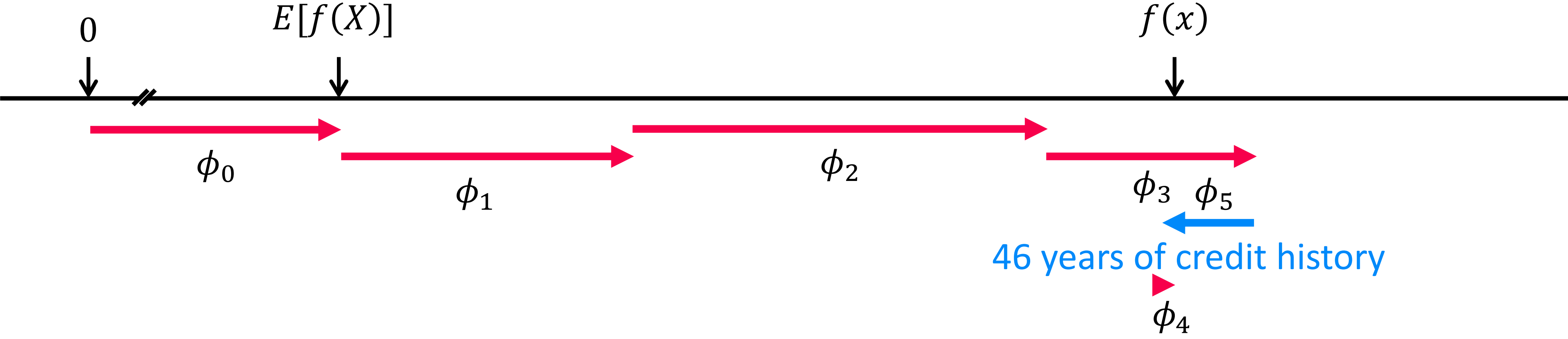


*Janzing et al. 2019

Lloyd Shapley

Nobel Prize in 2012

The order matters!

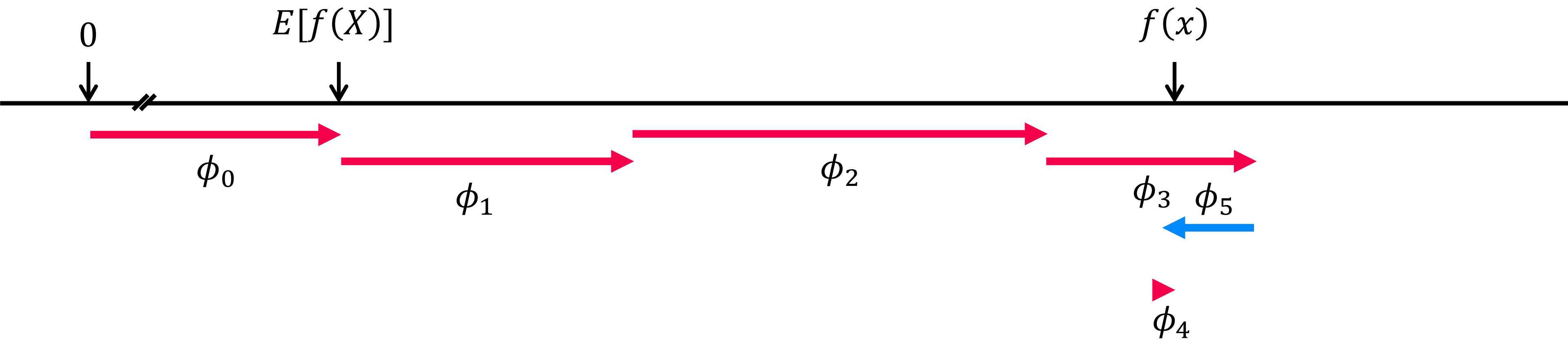


Shapley properties

1

Additivity (local accuracy) – The sum of the local feature attributions equals the difference between the base rate and the model output.

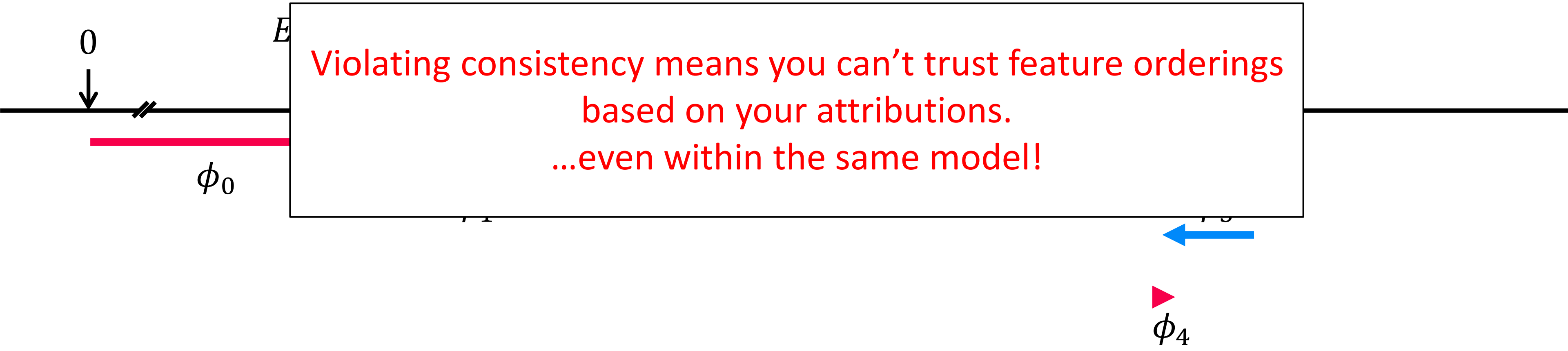
$$E[f(x)] + \sum_{i=1}^{M} \phi_i = f(x)$$



Shapley properties

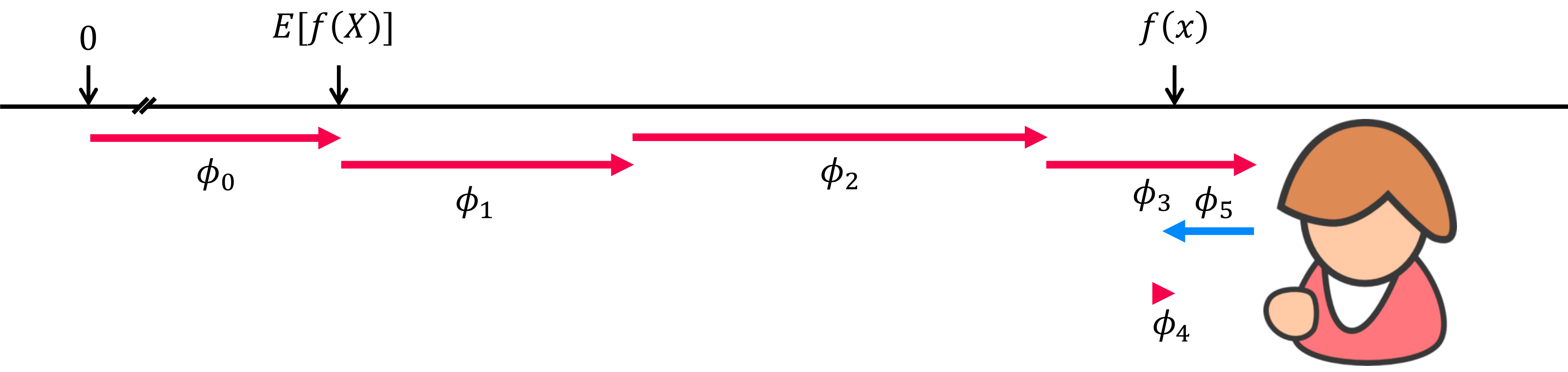
2

Monotonicity (consistency) – If you change the original model such that a feature has a larger impact in every possible ordering, then that input's attribution should not decrease.

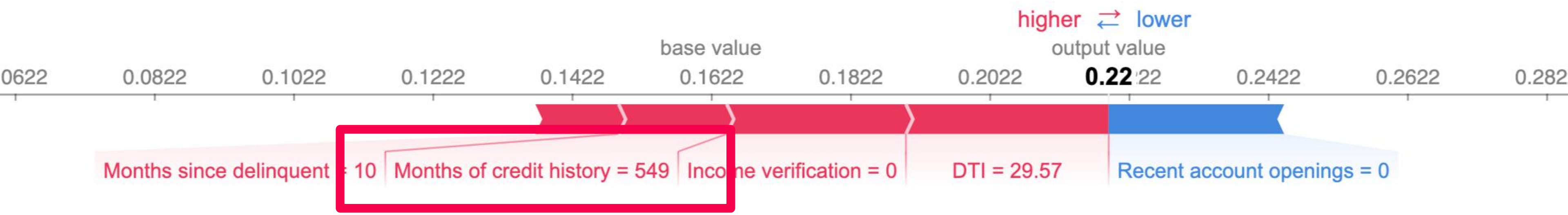


Violating consistency means you can't trust feature orderings based on your attributions.
…even within the same model!

Shapley values result from **averaging over all N! possible orderings**.



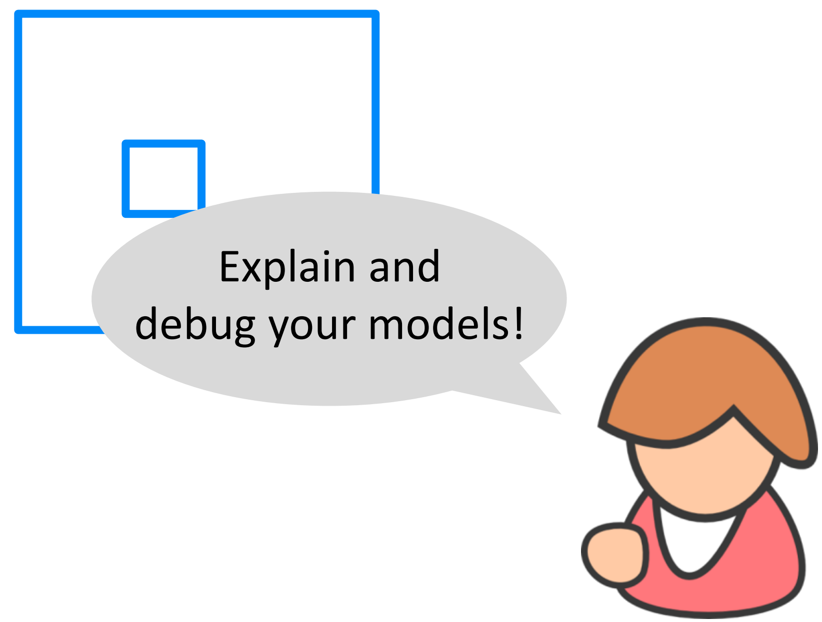
```
ex = shap.TreeExplainer(model, …)
shap_values = ex.shap_values(X)
shap.force_plot(     _          _      _       ,      _       ])
```



Why does 46 years of credit history increase the risk of payment problems?

```
shap.dependence_plot
```

The model is identifying retirement-age individuals based on their long credit histories!



Explainable AI in practice

Model development

- Debugging/exploration
- Monitoring
- Encoding prior beliefs

Human/AI collaboration

- Customer retention
- Decision support
- Human risk oversight

Regulatory compliance

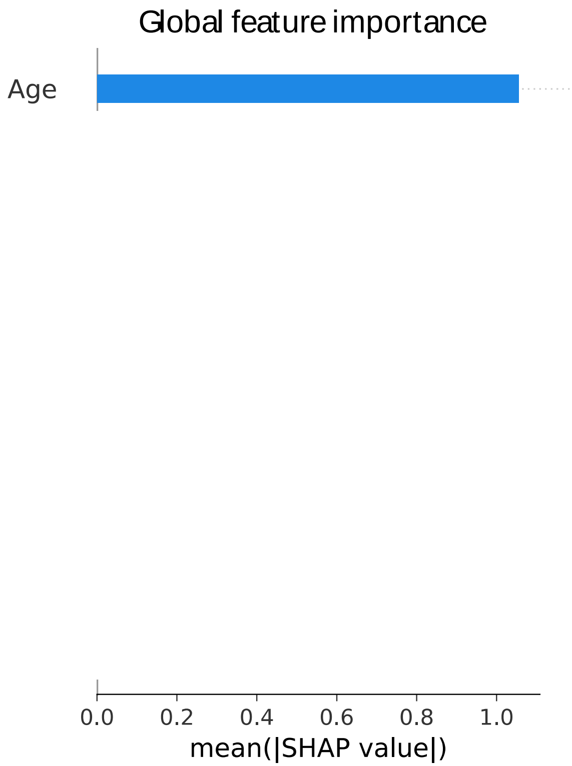
- Consumer explanations
- Anti-discrimination
- Risk management

Scientific discovery

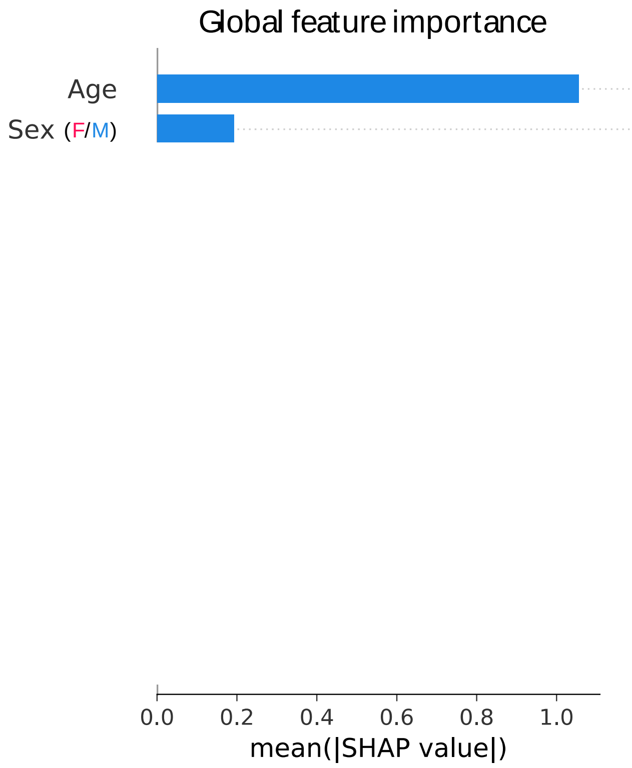
- Population subtyping
- Pattern discovery
- Signal recovery

Mortality risk model

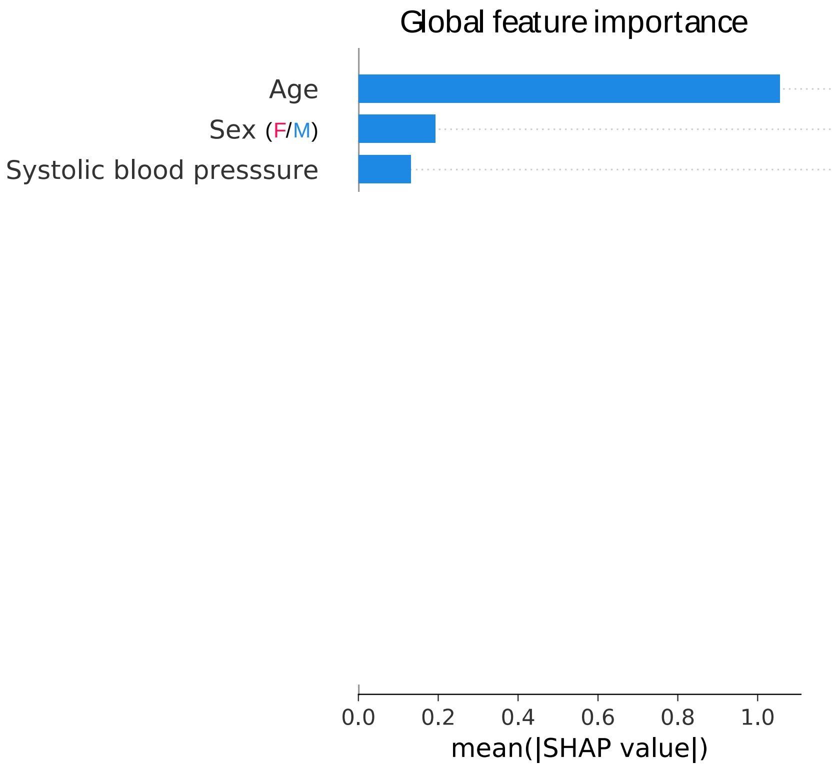
Mortality risk model



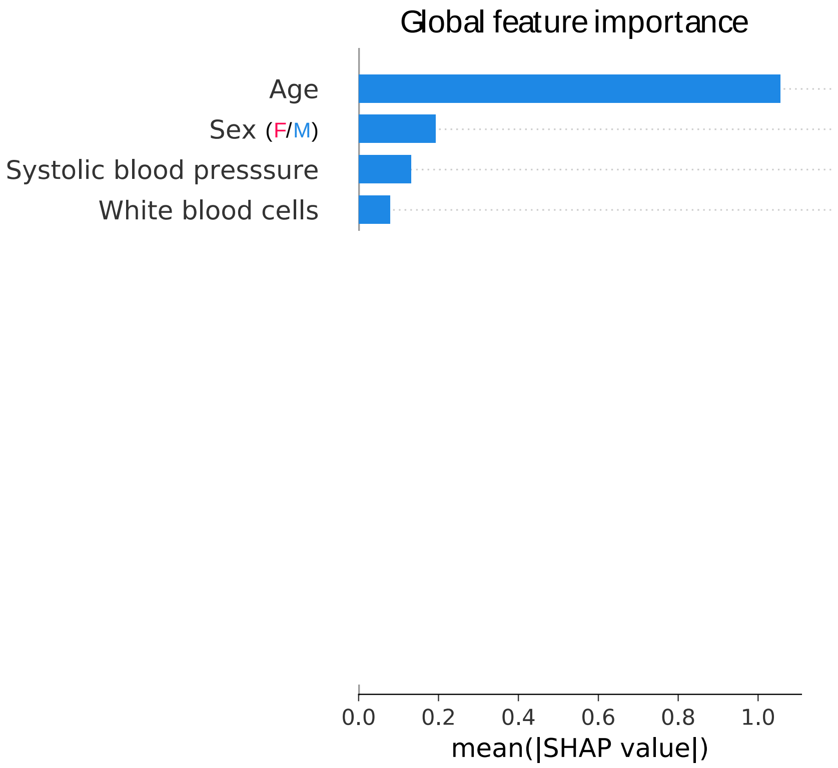
Mortality risk model



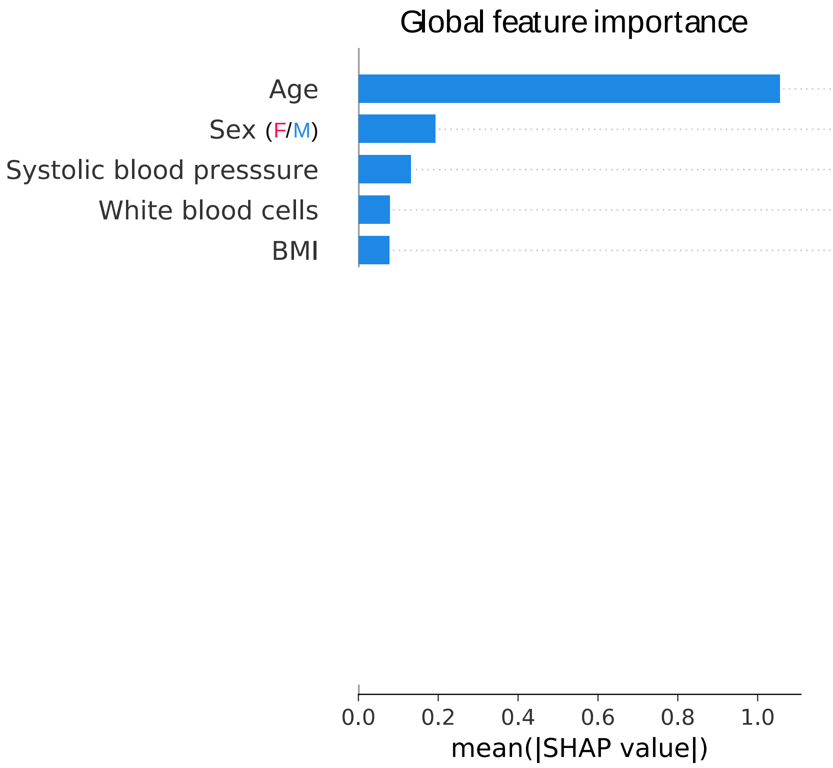
Mortality risk model



Mortality risk model

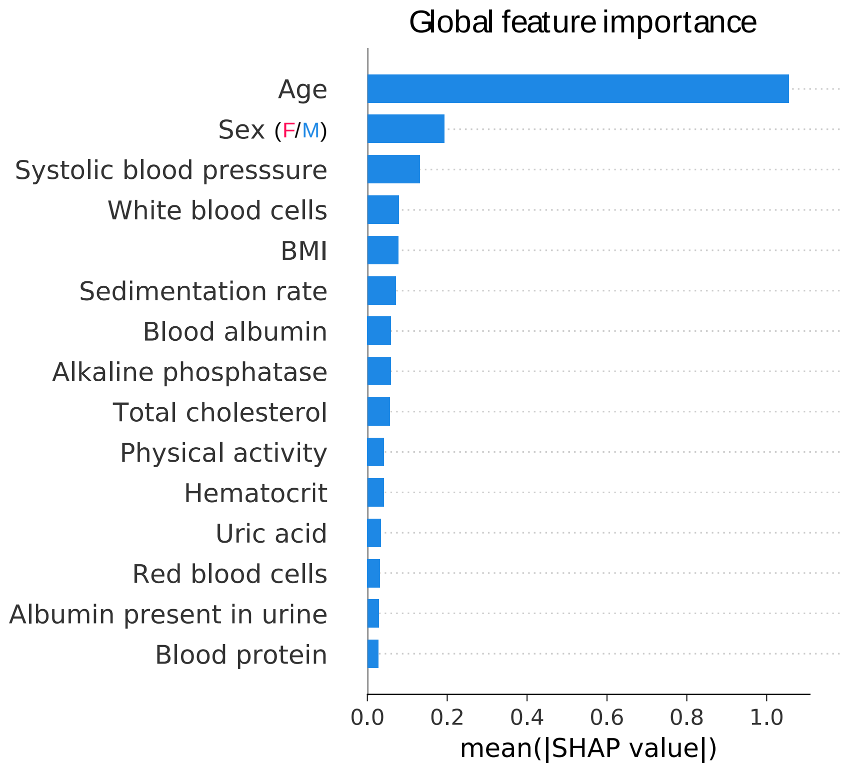


Mortality risk model



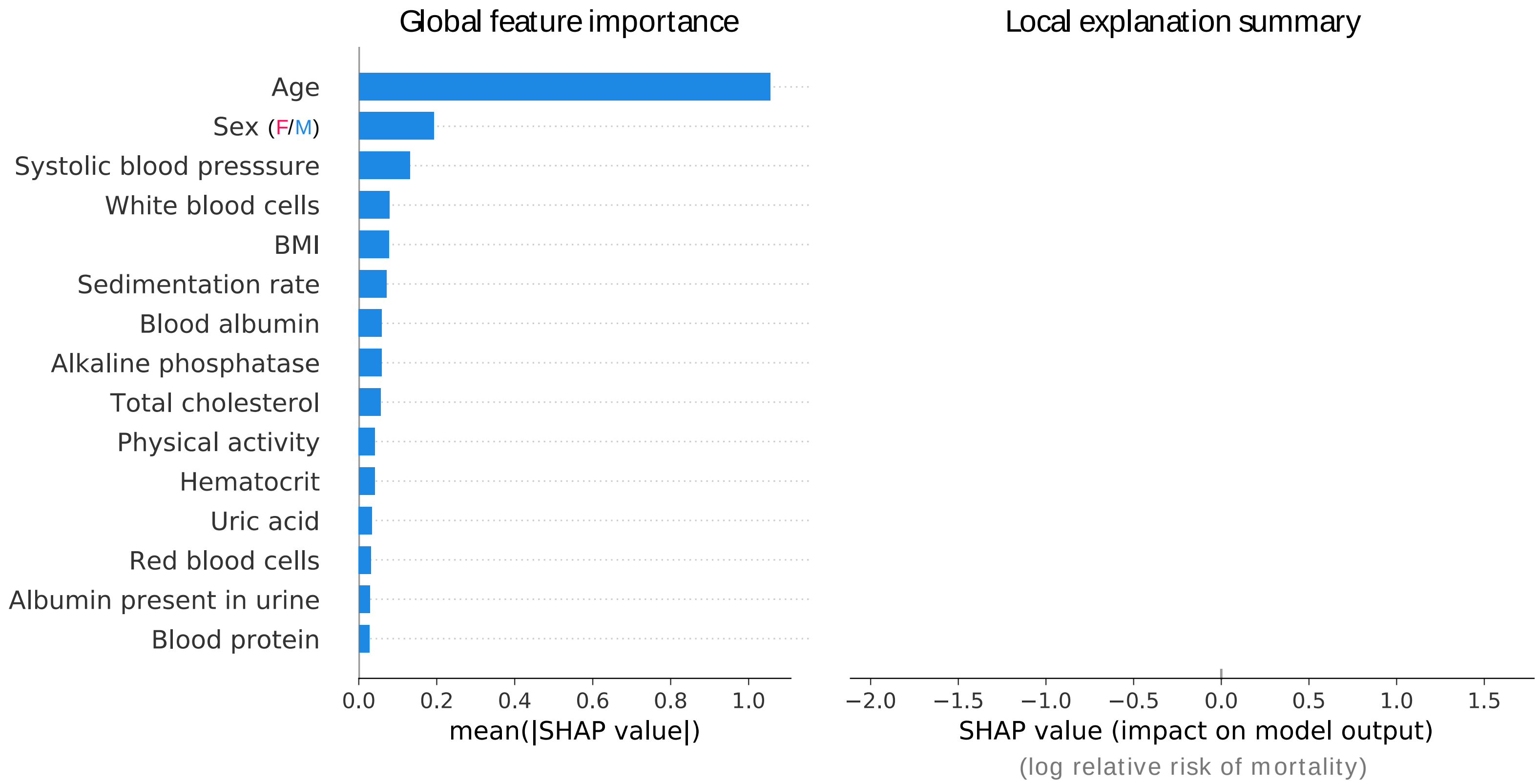
Mortality risk model

Reveal rare high-magnitude mortality effects

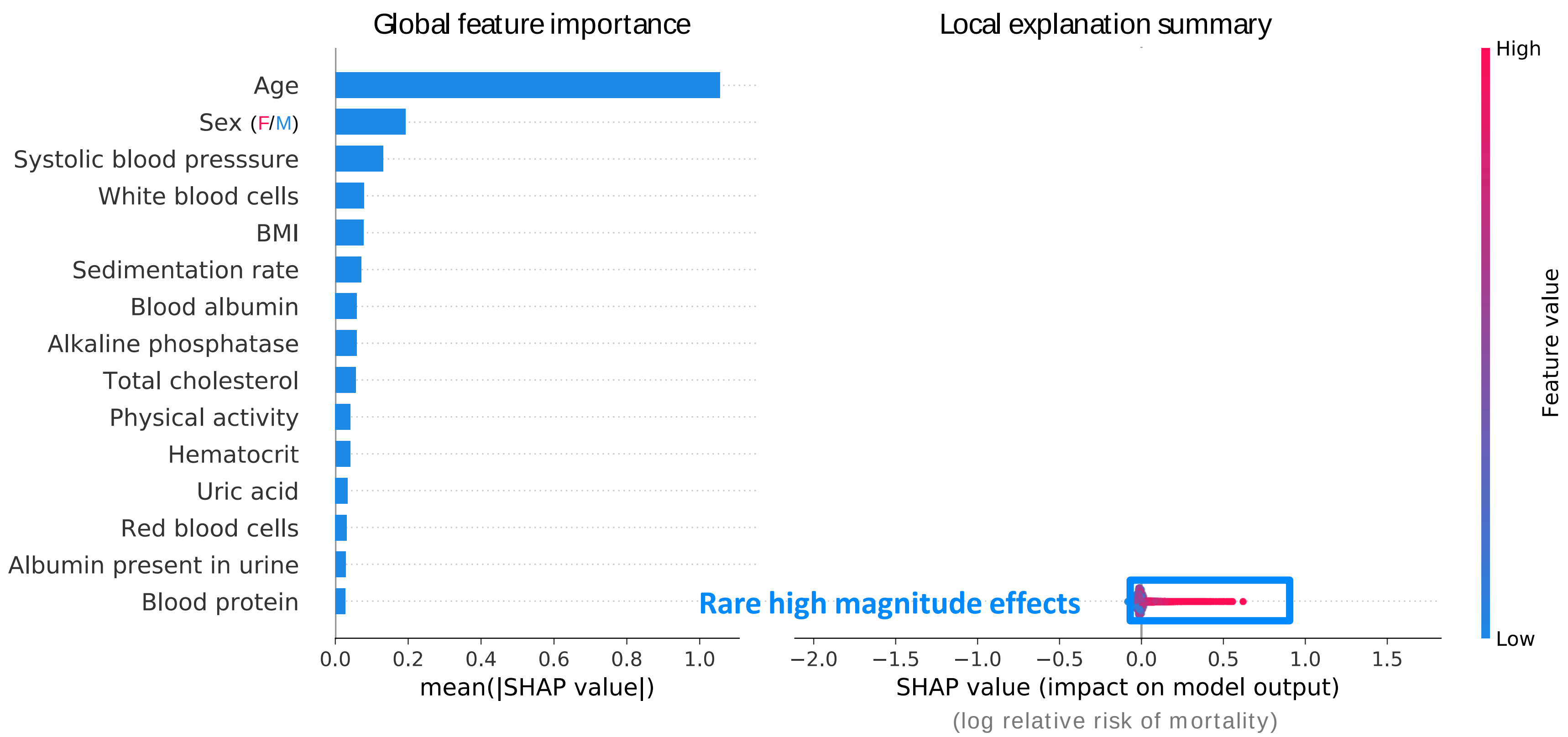


Conflates the prevalence of an effect with the magnitude of an effect

Reveal rare high-magnitude mortality effects



Reveal rare high-magnitude mortality effects



Reveal rare high-magnitude mortality effects