

Exploratory Machine Learning studies for disruption prediction on DIII-D

by

C. Rea, R.S. Granetz

MIT Plasma Science and Fusion Center, Cambridge, MA, USA

Presented at the

**2017 Theory and Simulation of Disruptions
Workshop**

PPPL

Princeton, NJ, USA

July 17th – 19th, 2017



the successful avoidance of disruption events is extremely relevant for fusion devices and in particular for ITER

- the problem complexity has motivated the recent efforts for development of **data-driven predictors** and **Machine Learning** studies to successfully **predict** disruption events with **sufficient warning time**
- SQL databases created, gathering time series of relevant plasma parameters: tables available on **DIII-D**, EAST and C-Mod (**cross-device analysis**)
 - more than 40 available parameters, ~500k samples per parameter
 - both disrupted and non-disrupted discharges, focus on 2015 campaigns for now
- **exploratory studies** to gain insights on the DIII-D dataset before addressing the problem of a disruption-warning algorithm
 - binary and multi-class classification analysis through Machine Learning algorithms
 - variable importance ranking
 - accuracy metrics and comparison between different applications

we chose a subset of features and samples for ML applications to the DIII-D database for disruption prediction

10 features out of ~40 available parameters

mainly **dimensionless** or **machine-independent** parameters

li	Ip_error_fraction	Vloop
q95	radiated_fraction	n1amp
beta_p	dWmhd_dt	
n/nG	Te_HWHM	

focus on **flattop disruptions: 195 flattop disruptions** complemented by an analogous number of discharges that did not disrupt, possibly extracted from the same experiments (similar operational space) \Rightarrow **392 discharges**

~70,000 samples for each of the **10** chosen **input variables**

reliable knowledge base capable of highlighting the underlying physics :

- validated signals and EFIT reconstructions
- avoided intentional disruptions
- avoided hardware-related disruptions by specifically checking for feedback control on plasma current or UFOs events

a plethora of ML algorithms is available in literature – already tested on other devices for disruption prediction

- ML supervised and unsupervised algorithms, mainly developed at JET and also studied in real-time environment, “**black box**” approach:
 - Artificial Neural Networks[1-2], multi-tiered Support Vector Machines[3], Manifolds and Generative Topographic Maps[4]

[1] G. Pautasso et al. Nuclear Fusion 42 (2002) 100-108

[2] B. Cannas et al. Nuclear Fusion 44 (2004) 68-76

[3] J. Vega et al. Fusion Engineering and Design 88 (2013)

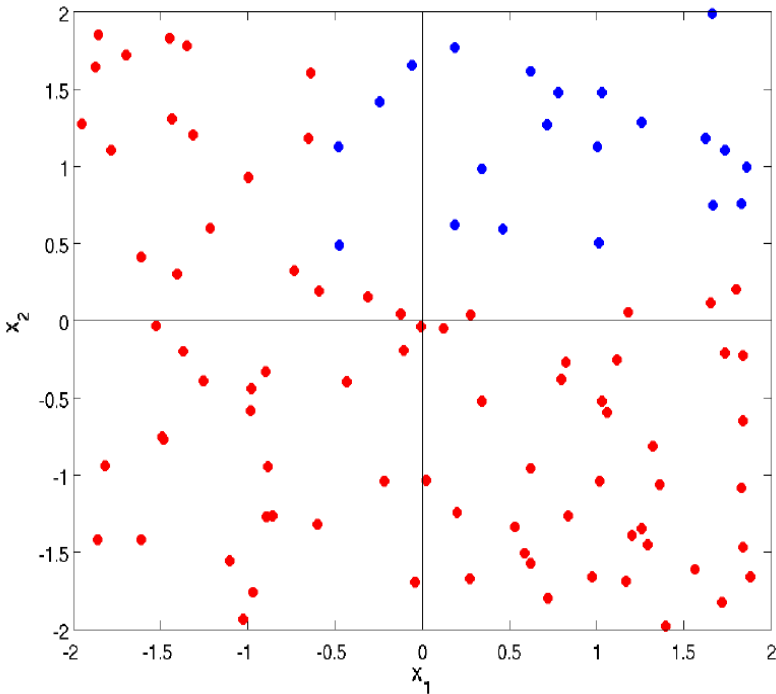
[4] B. Cannas et al. Nuclear Fusion 57 (2013) 093023

- to better understand the dataset: “**white box**” approach
 - inner components and logic are available for inspection
 - importance of individual features can be determined
- **Random Forests**[5]: a large collection of randomized de-correlated **decision trees** that can be used to solve both **classification** and regression problems

[5] L. Breiman, “Random Forests”, Machine Learning, 45(1), 5-32, 2001

decision trees are hierarchical data structures implementing the *divide-and-conquer* strategy: 2D classification example

- **CART** (Classification and Regression Trees) algorithms repeatedly partition the input space, implementing a test function at each split (node), to build a tree whose nodes are as pure as possible
- 2 features (x_1, x_2) and 2 classes (**red**, **blue**)



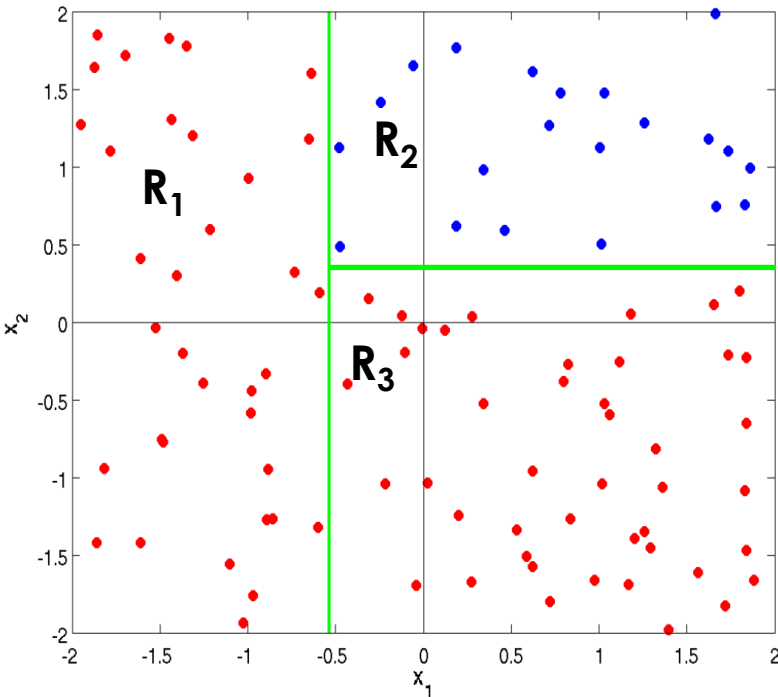
the algorithm selects a splitting value to partition the dataset, by **minimizing** an **impurity** measure:

$$I'_m = - \sum_{j=1}^n \frac{N_{mj}}{N_m} \sum_{i=1}^K p_{mj}^i \log_2 p_{mj}^i$$

E. Alpaydin, "Introduction to Machine Learning", 2nd edition, MIT Press

decision trees are hierarchical data structures implementing the *divide-and-conquer* strategy: 2D classification example

- **CART** (Classification and Regression Trees) algorithms repeatedly partition the input space, implementing a test function at each split (node), to build a tree whose nodes are as pure as possible
- 2 features (x_1, x_2) and 2 classes (**red**, **blue**)



the algorithm selects a splitting value to partition the dataset, by **minimizing** an **impurity** measure:

$$\mathcal{I}'_m = - \sum_{j=1}^n \frac{N_{mj}}{N_m} \sum_{i=1}^K p_{mj}^i \log_2 p_{mj}^i$$

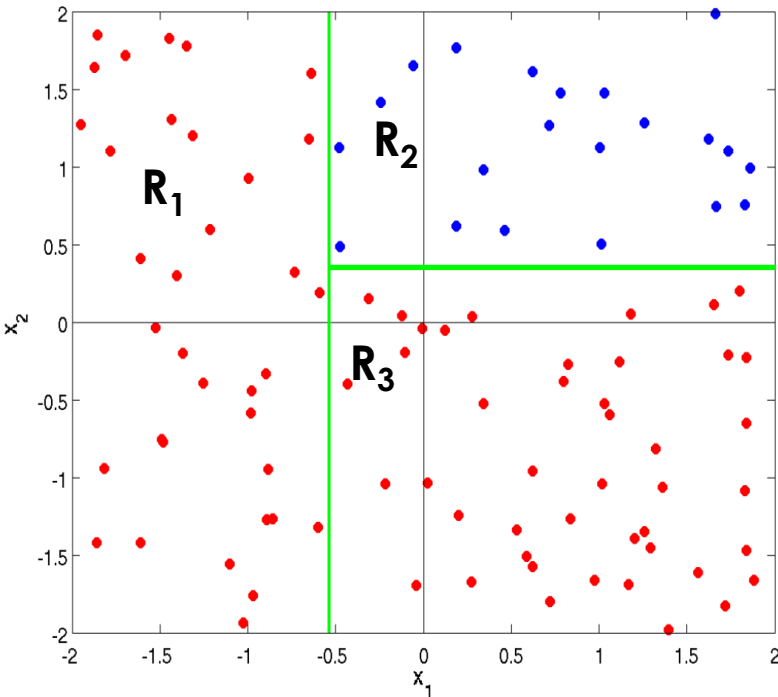
E. Alpaydin, "Introduction to Machine Learning", 2nd edition, MIT Press

all the subdivided regions are **pure** red or blue data

- a **set of rules** is defined and can be visualized as a **tree structure**

decision trees are hierarchical data structures implementing the *divide-and-conquer* strategy: 2D classification example

- **CART** (Classification and Regression Trees) algorithms repeatedly partition the input space, implementing a test function at each split (node), to build a tree whose nodes are as pure as possible
- 2 features (x_1, x_2) and 2 classes (**red**, **blue**)



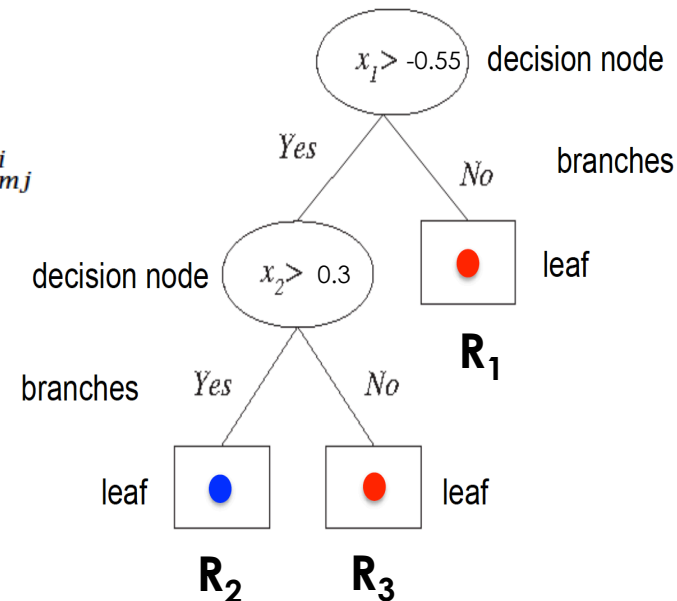
the algorithm selects a splitting value to partition the dataset, by **minimizing** an **impurity** measure:

$$\mathcal{I}'_m = - \sum_{j=1}^n \frac{N_{mj}}{N_m} \sum_{i=1}^K p_{mj}^i \log_2 p_{mj}^i$$

E. Alpaydin, "Introduction to Machine Learning", 2nd edition, MIT Press

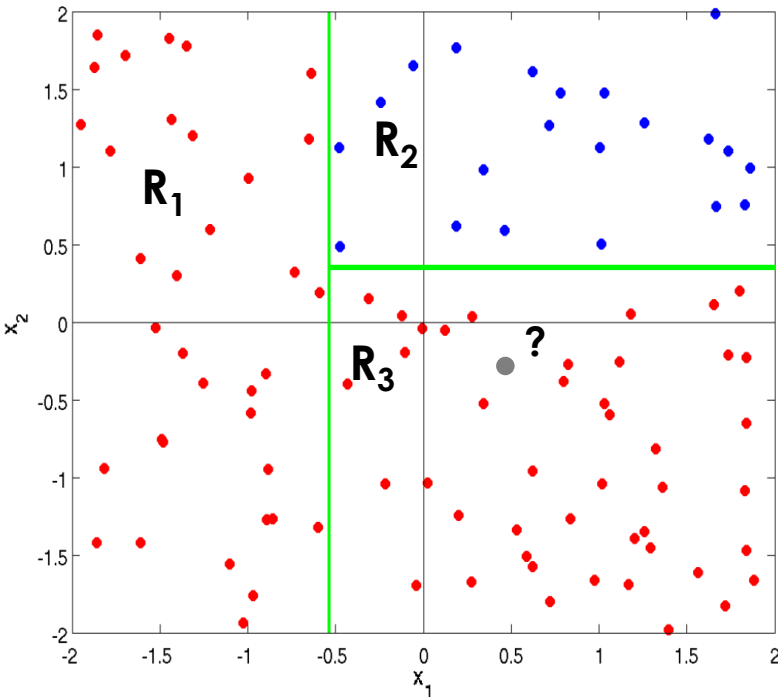
all the subdivided regions are **pure** red or blue data

- a **set of rules** is defined and can be visualized as a **tree structure**



decision trees are hierarchical data structures implementing the *divide-and-conquer* strategy: 2D classification example

- **CART** (Classification and Regression Trees) algorithms repeatedly partition the input space, implementing a test function at each split (node), to build a tree whose nodes are as pure as possible
- 2 features (x_1, x_2) and 2 classes (**red**, **blue**)



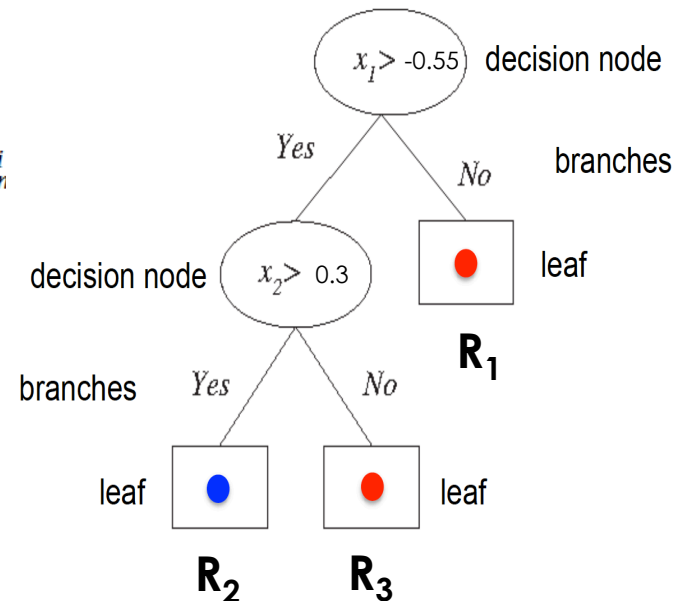
the algorithm selects a splitting value to partition the dataset, by **minimizing** an **impurity** measure:

$$I'_m = - \sum_{j=1}^n \frac{N_{mj}}{N_m} \sum_{i=1}^K p_{mj}^i \log_2 p_{mj}^i$$

E. Alpaydin, "Introduction to Machine Learning", 2nd edition, MIT Press

all the subdivided regions are **pure** red or blue data

- this **set of rules** is used to classify a new, unseen (test) sample



recursive binary trees have a key feature: interpretability, but they tend to overfit – pruning needed

- decision trees have advantages and limitations, as well as other ML algorithms - **Random Forests** seems a promising algorithm

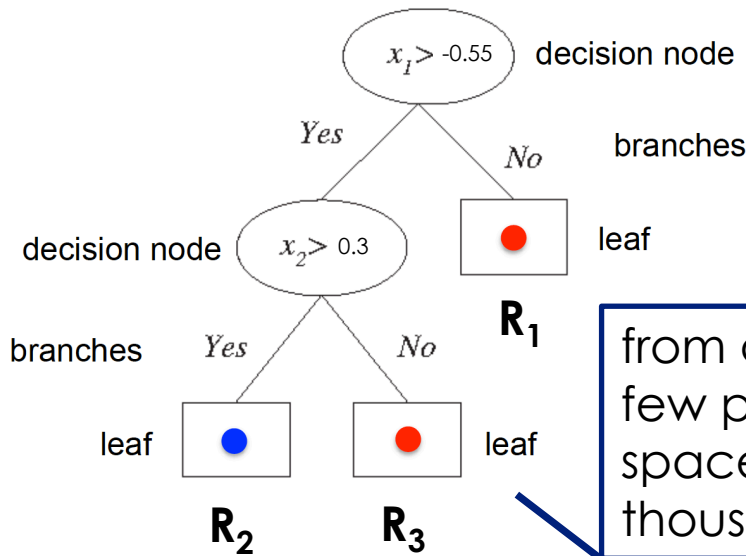
for classification purposes:	RF	Tree	Neural Nets	SVMs
no overfitting	●	●	●	●
intrinsic feature selection and robustness to outliers	●	●	●	●
parameter tuning	●	●	●	●
non-parametric models (no a-priori assumptions)	●	●	●	●
interpretability	●	●	●	●
natural handling of mixed type data	●	●	●	●
prediction accuracy	●	●	●	●
time handling	●	●	●	●

Random Forests is an ensemble method, leveraging bootstrapping and averaging techniques (*bagging*)

main steps of the algorithm:

- grow many trees (e.g., 500 trees) on **bootstrap samples** of the original training set
 - random sampling with replacement from the training set
- trees are fully grown - **minimize bias** (no pruning needed)
- **reduce variance** of noisy but unbiased (fully grown) trees by **averaging** (regression) or **majority voting** (classification) for the final decisions on test samples
- very **fast**, highly **parallelized** algorithm

binary classification problem: labeling a time slice as disrupted/non-disrupted



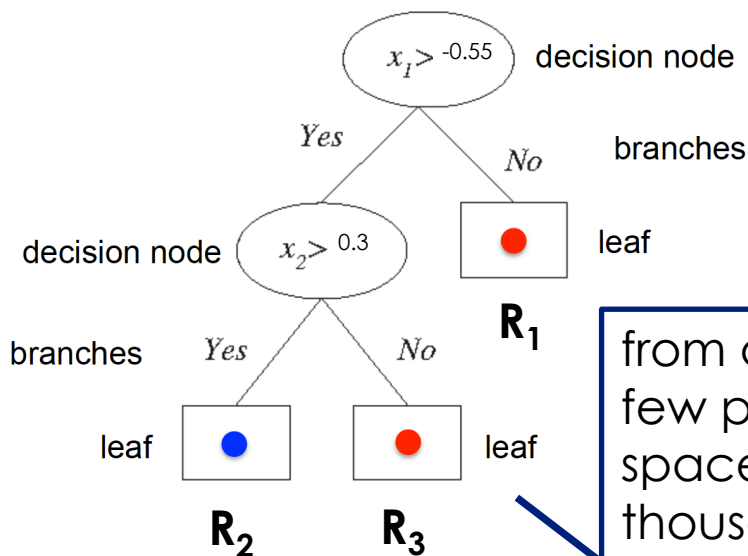
$(x_1, x_2) \rightarrow$

li	lp_error_fraction	Vloop
q95	radiated_fraction	n1amp
beta_p	dWmhd_dt	
n/nG	Te_HWHM	

from a 2D space with few points to a 10D space with many thousands of points

binary classification problem: disrupted/non-disrupted

graphical depiction of a single tree in a Random Forests

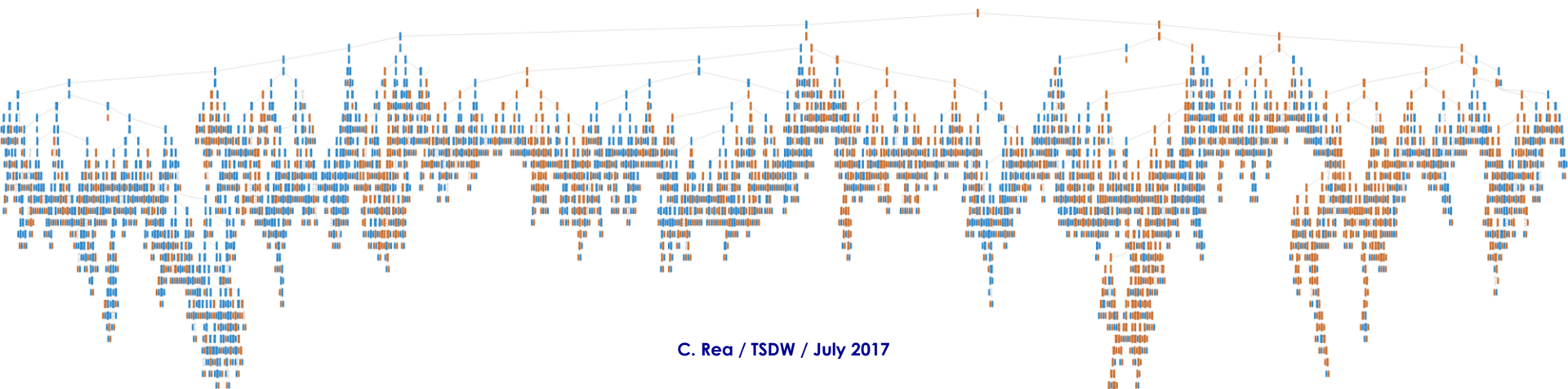


$(x_1, x_2) \rightarrow$

li	lp_error_fraction	Vloop
q95	radiated_fraction	n1amp
beta_p	dWmhd_dt	
n/nG	Te_HWHM	

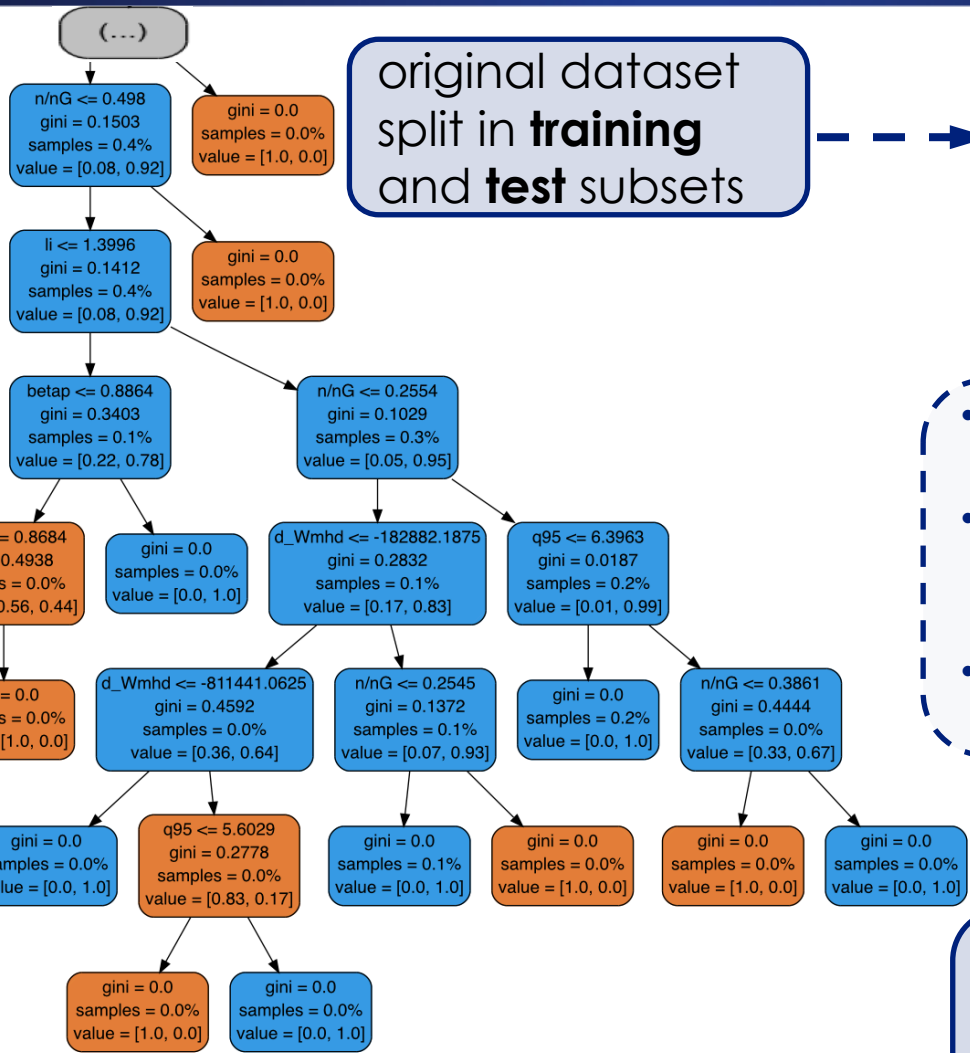
from a 2D space with few points to a 10D space with many thousands of points

10 variables
 ~ 70,000 time slices
 394 discharges
 195 flattop disruptions
 500 decision trees



binary classification problem: disrupted/non-disrupted

no time dependency – 10D feature vectors



original dataset split in **training** and **test** subsets

500 trees built on **training data** to define a set of rules

- features are **not scaled a-priori**;
- at each node, **random sub-selection** of features;
- impurity minimization by choosing the **best split**;

set of rules is used to decide if new, unseen samples in our **test set**, belong to one of the two classes

confusion matrix is used as an accuracy metrics to assess the model's capability to discriminate between class labels

binary classification -
no time dependency -
15ms **black window**
before disruption event

non-disrupted

disrupted

confusion matrix is used as an accuracy metrics to assess the model's capability to discriminate between class labels

binary classification -
no time dependency -
15ms **black window**
before disruption event

non-disrupted disrupted

class accuracy:

- **disrupted** **94.1%** (TPR)
- **non-disrupted** **94.4%** (TNR)

overall accuracy: **~94.3%**

successfully
predicted
non-disrupted
samples

false
positive
alarms

Confusion matrix

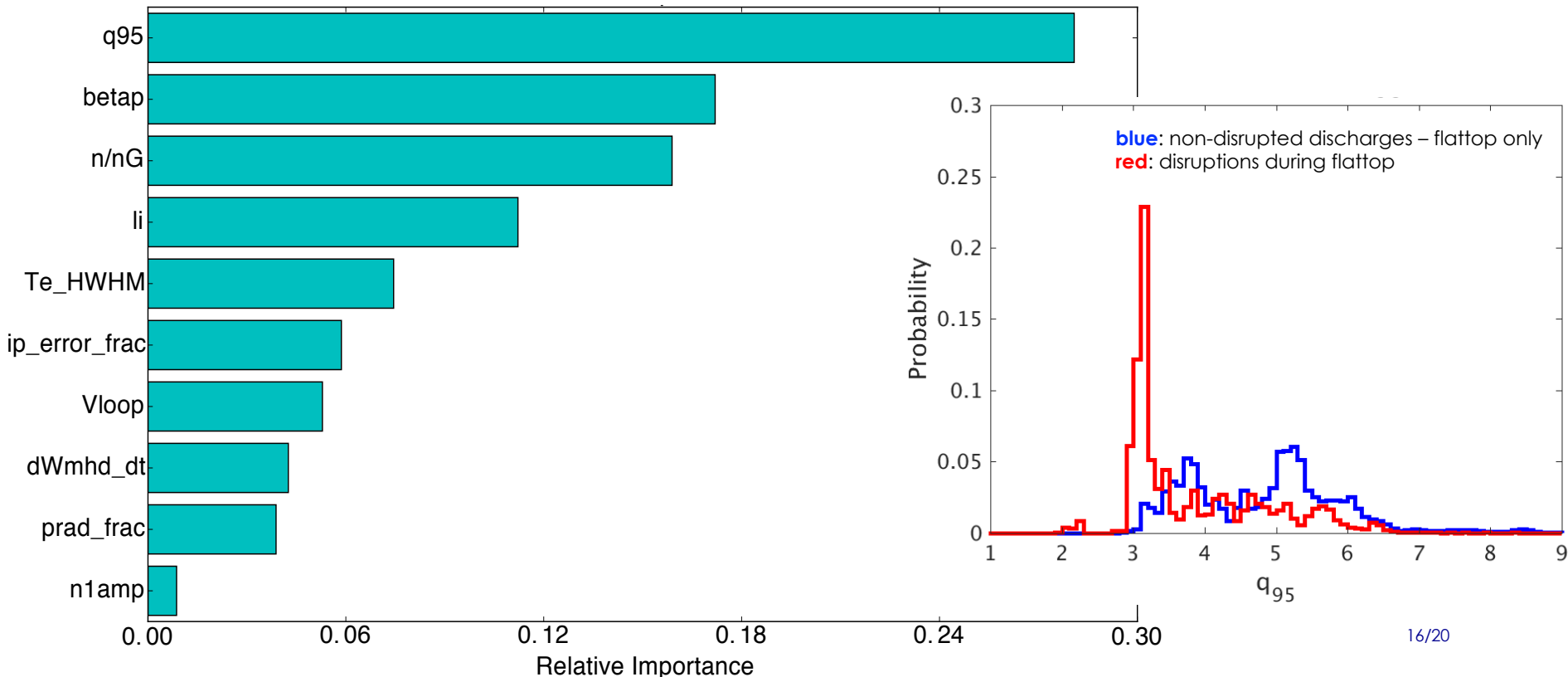
	non-disrupted	disrupted
True label non-disrupted	4940 51.4%	294 3.06%
True label disrupted	258 2.68%	4119 42.86%
	non-disrupted	disrupted
	Predicted label	

missed
detections
(false
negatives)

successfully
predicted
disrupted
samples

relative importance ranking can be extracted from the Random Forests – ‘white box’ approach

- relative variable importance wrt label predictability is defined as mean decrease impurity and can give indications on the underlying physics
- q95 is the relatively most important variable
 - probability distributions for disrupted and non-disrupted samples show different behaviors



binary classification – train/test split on the basis of the whole discharges and not the individual samples

- information pertaining to the same discharges should not be shared with the training set: how can we assess the model's **generalization** capabilities?
- reliable test set, capable of correctly highlighting the model's **predictive power**

class accuracy:

- **disrupted** 93.1%
- **non-disrupted** 87.3%

overall accuracy: ~90%

Confusion matrix

True label	non-disrupted	4540 46.75%	661 6.81%
	disrupted	309 3.18%	4201 43.26%
		non-disrupted	disrupted
		Predicted label	

in the multi-class classification, the time dependency is introduced through the definition of different class labels

multi-class classification

→ **non-disrupted** : time slices for non disruptive shots

→ **far from disr** :
time_until_disrupt > 0.35s

→ **close to disr** :
time_until_disrupt ≤ 0.35s

→ ...

class accuracy:

- **non-disrupted** **88.6%**
- **far from disr** **84.4%**
- **close to disr** **86.6%**

overall accuracy: **~87%**

Confusion matrix

		non-disrupted	far from disr	close to disr
True label	non-disrupted	4607 47.44%	579 5.96%	15 0.15%
	far from disr	326 3.36%	2594 26.71%	152 1.57%
	close to disr	71 0.73%	121 1.25%	1246 12.83%
		non-disrupted	far from disr	close to disr
		Predicted label		

in the multi-class classification, the time dependency is introduced through the definition of different class labels

multi-class classification

→ **non-disrupted** : time slices for non disruptive shots

→ **far from disr** :
time_until_disrupt > 0.35s

→ **close to disr** :
time_until_disrupt ≤ 0.35s

→ ...

class accuracy:

- **non-disrupted** **88.6%**
- **far from disr** **84.4%**
- **close to disr** **86.6%**

overall accuracy: **~87%**

grouping **non-disrupted** and **far from disr** classes, thus defining two new classes: **stable** and **disruptive** (phases), gives an overall accuracy **~96%**

Confusion matrix

	non-disrupted	far from disr	close to disr
True label	4607 47.44%	579 5.96%	15 0.15%
non-disrupted			
far from disr	326 3.36%	2594 26.71%	152 1.57%
close to disr	71 0.73%	121 1.25%	1246 12.83%
	non-disrupted	far from disr	close to disr
	Predicted label		

conclusions and future work

- ML **classification** gives promising results, with optimal performances (**Random Forests**) and possibility of gaining insights on the dataset
- address the **disruption-proximity problem**: “how much time until the discharge is going to disrupt” - possibly evaluating different algorithms that could best perform in case of **regression** problems
- **real-time** integration with the PCS
- dimensionless and machine-independent features enable **cross-device analysis**: comparison with EAST and C-Mod data and possible extrapolation to **ITER**

