Machine Learning Prediction of High-Current Disruptions with Low-Current Training Data

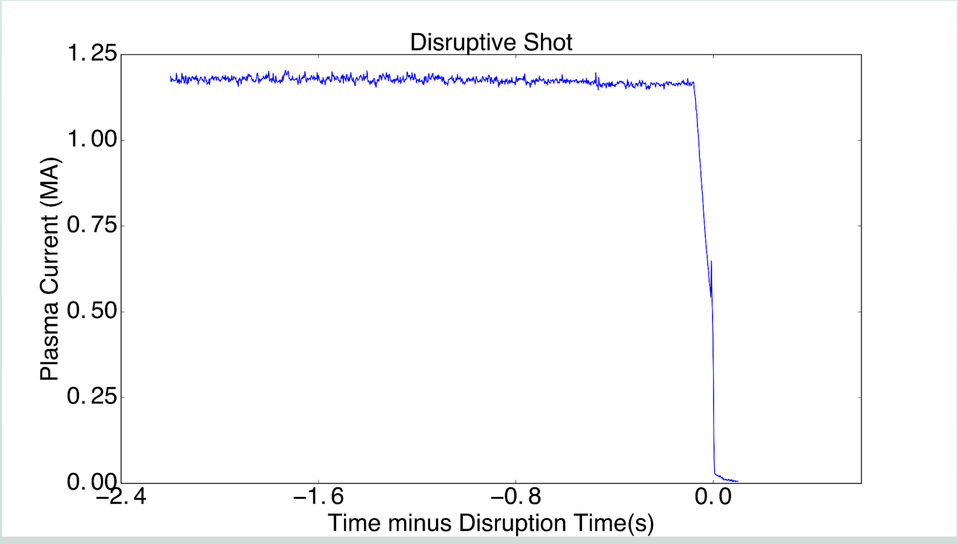
Nathaniel Barbour[1], Kornee Kleijwegt[2], Leonard Lupin-Jimenez[3], Egemen Kolemen[4]
[1]Yale University, [2]Eindhoven University of Technology, [3]Stanford University, [4]Princeton University

Objective

- **Develop a predictor of high plasma current disruptions using ensembles of regression trees trained with low plasma current data**
- **Explore methods of stacking the predictions of multiple types of regression ensembles.**

Disruptions

- Disruption events occur in tokamaks when the plasma current rapidly decreases to zero.



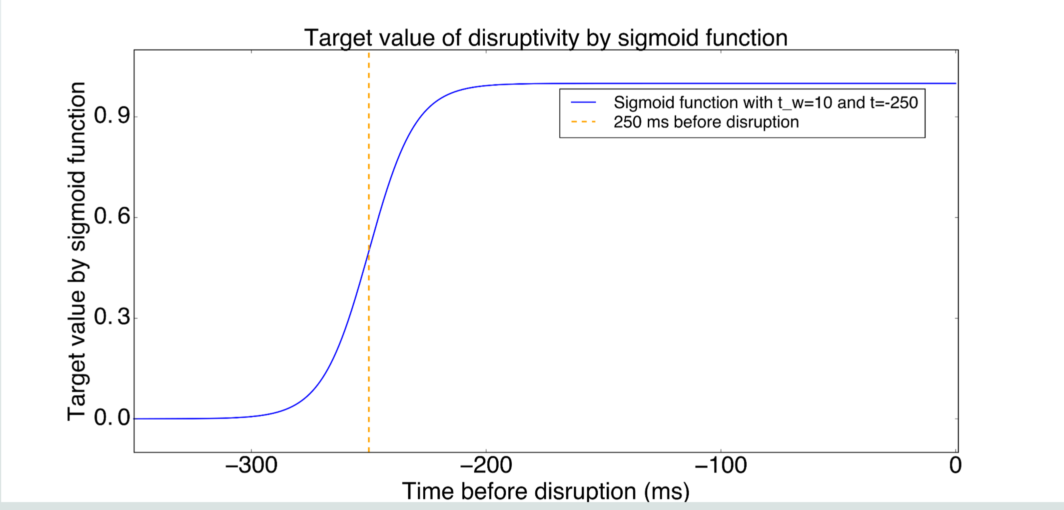
- Disruptions can lead to "runaway" electrons, which cause significant damage to plasma-facing components.

Disruption Prediction

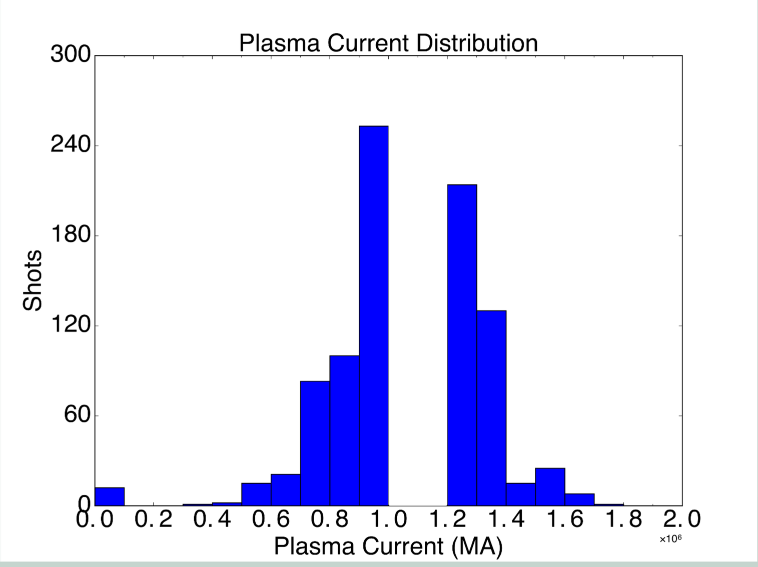
- Complex, nonlinear interactions between parameters preclude pure physics-based disruptors.
- Machine learning techniques for pattern recognition are often employed.
- Artificial Neural Networks (ANNs) used at JET [1], ASDEX [2], ADITYA [3], and DIII-D [4] classify disruptions with signals available in real-time.
- ANNs are "black boxes"; prediction methods are not transparent.
- **Classification and regression trees [5] offer a more transparent approach to prediction.**
- **Challenges to solve before ITER:**
  - Develop an accurate predictor without having existing database of high plasma current shots at ITER
  - Develop a predictor which highlights patterns in parameters of interest for disruption modeling

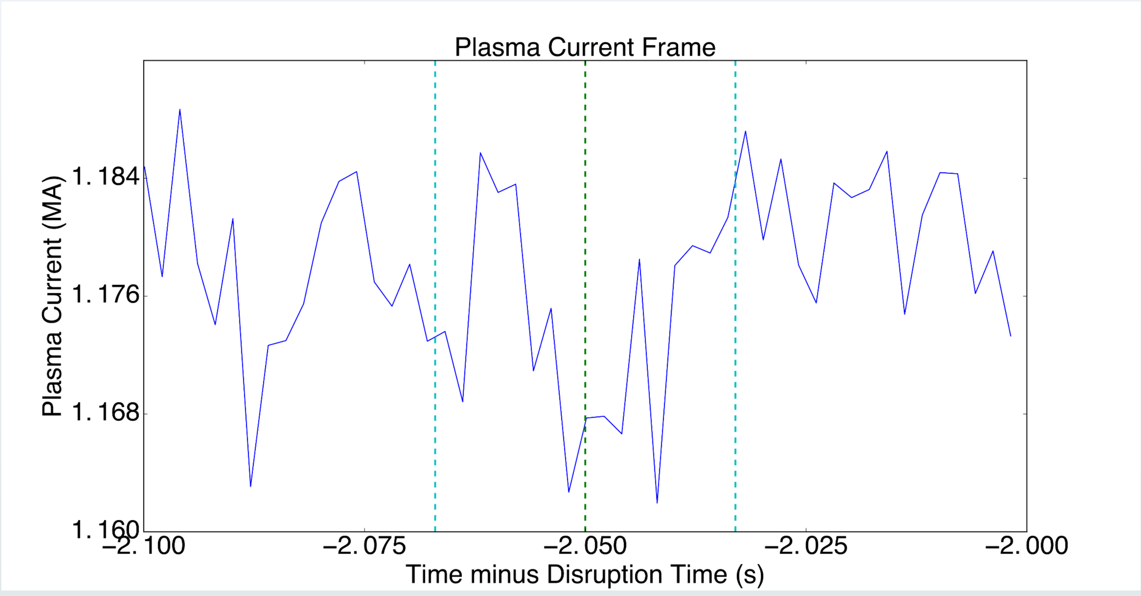
Data Reduction Process

- Time series data for 29 plasma parameters from 630 disrupting shots and 500 non-disrupting shots are collected.
- Time frames are created for each parameter by splitting the time series data into 100 ms windows.
- For disruptive shots, disruptivity target values are created using a sigmoid activation function:

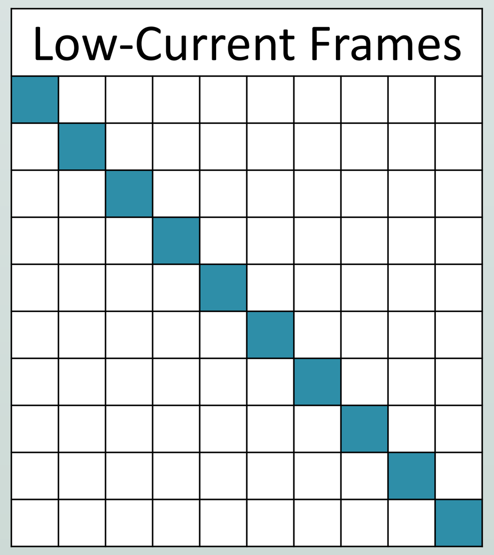


- Ensembles of regression trees are trained with low-current data using k-fold cross validation.
- A disruptivity prediction is made for each high-current frame using the trained ensembles.
- Accuracy is evaluated at a series of disruptivity thresholds between 0 and 1.





- Each frame is split into sub-frames (whole, halves and thirds).
- For each sub-frame, the mean, variance, and trend are calculated.
- The coefficients of a cubic polynomial fit are calculated from the data of the whole frame.
- Thus, one frame has 638 associated values in the dataset (29 parameters, each with 22 values).

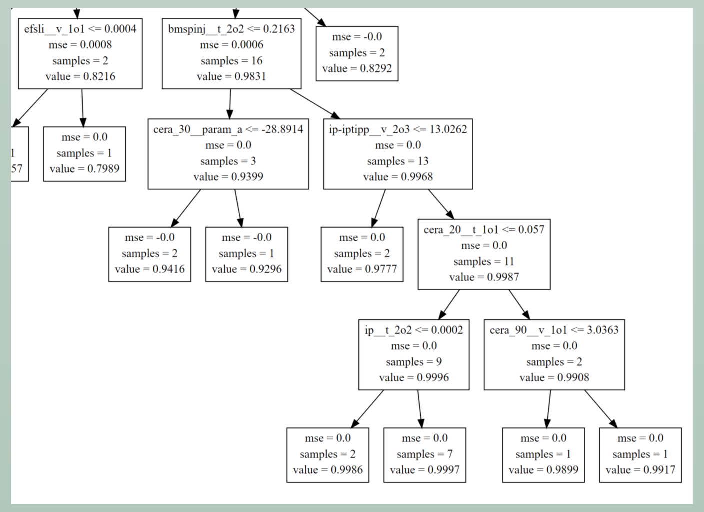


k-fold cross validation:

With k=10, low-current data is split into 10 folds. Ten ensembles are trained, each with a different fold excluded from the training set. The final prediction is the mean of the ensembles' predictions.

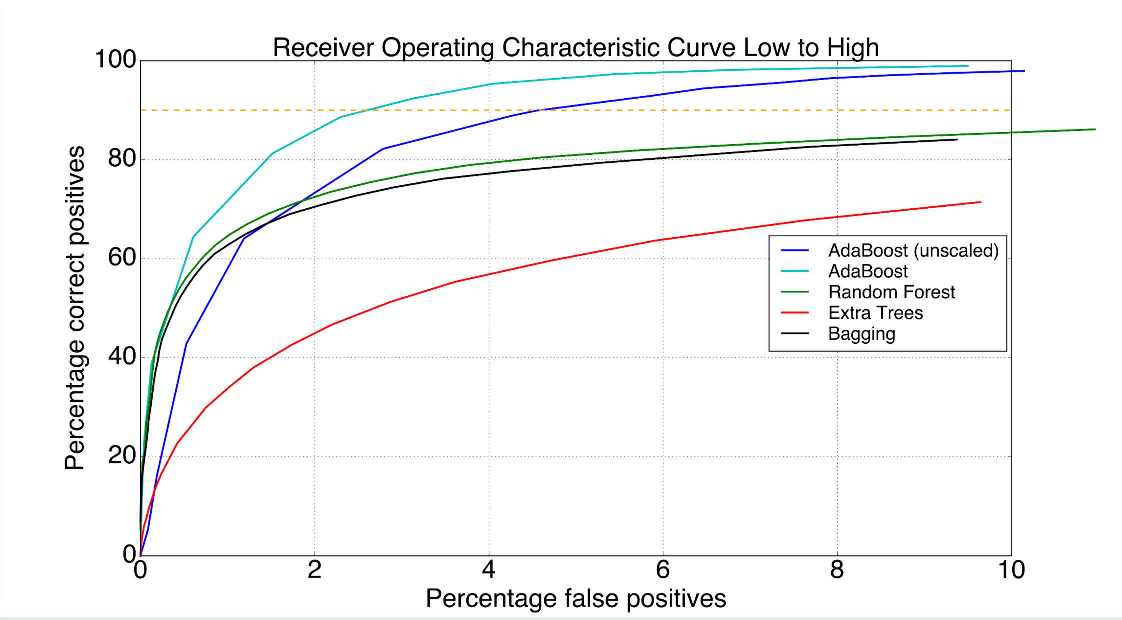
Regression Tree Ensembles

- Regression trees group frames into terminal nodes, "leaves", by a series of decisions:
- At each level, the parameter value which reduces the mean squared error of the disruptivity predictions is determined.
- In each final leaf, the disruptivity value predicted is the mean of the values in that leaf.
- Four ensemble methods employed using the scikit-learn [6] machine learning package for the Python programming language:
- Bootstrap aggregating (**Bagging**) [7] trains trees in parallel using subsets of the same size as the full training set, drawn with replacement.
- **Random Forests** [8] extend the bagging method by choosing split candidates from a random subspace of the parameters.
- Extremely Randomized Trees (**Extra Trees**) [9] further extend random forests by choosing the best split from a random set of uniform splits, from a random subspace of the parameters.



- Adaptive Boosting (**AdaBoost**) [10-11] progressively trains an ensemble of weak learners by emphasizing and improving the worst predictions in the previous iteration.

Results



- **93% success rate with 3.2% false positive predictions for AdaBoost with scaled parameters**
- **Using scaled parameters almost halved the false positive predictions at 90% success rate.**
- **Non-boosting methods were less successful than AdaBoost was, but they were more robust.**
- **Low success rates of other algorithms precluded accuracy improvements from stacking regressors.**

Future Work

- Repeat with data spanning a wider range of plasma currents.
- Develop robust weighting algorithm that combines strengths of AdaBoost and Random Forests.
- Optimize parameter list.
- Perform cross-device analysis with normalized parameters.
- Train with 1-d radial profile data.
- Study cases where predictions fail.

References

[1] B. Cannas *et al* 2004 *Nucl. Fusion* **44** 68
[2] B. Cannas *et al* 2010 *Nucl. Fusion* **50** 075004
[3] A. Sengupta and P. Ranjan 2000 *Nucl. Fusion* **40** 1993
[4] D. Wroblewski *et al* 1997 *Nucl. Fusion* **37** 725
[5] Breiman, Leo; Friedman, J. H.; Olshen, R. A.; Stone, C. J. (1984). *Classification and regression trees*. Monterey, CA: Wadsworth & Brooks/Cole Advanced Books & Software.
[6] Pedregosa *et al.*, "Scikit-learn: Machine Learning in Python", JMLR **12**, pp. 2825-2830, 2011
[7] Breiman, Leo (1996). "Bagging predictors". *Machine Learning*. **24** (2): 123-140.
[8] Breiman, Leo (2001). "Random Forests". *Machine Learning*. **45** (1): 5-32.
[9] P. Geurts, D. Ernst., and L. Wehenkel, "Extremely randomized trees", *Machine Learning*, **63**(1), 3-42, 2006.
[10] Y. Freund, R. Schapire, "A Decision-Theoretic Generalization of on-Line Learning and an Application to Boosting", 1995.
[11] H. Drucker, "Improving Regressors using Boosting Techniques", 1997.

Acknowledgements

This work was made possible by funding from the Department of Energy for the Summer Undergraduate Laboratory Internship (SULI) program. This work is supported by the US DOE Contract No.DE-AC02-09CH11466.