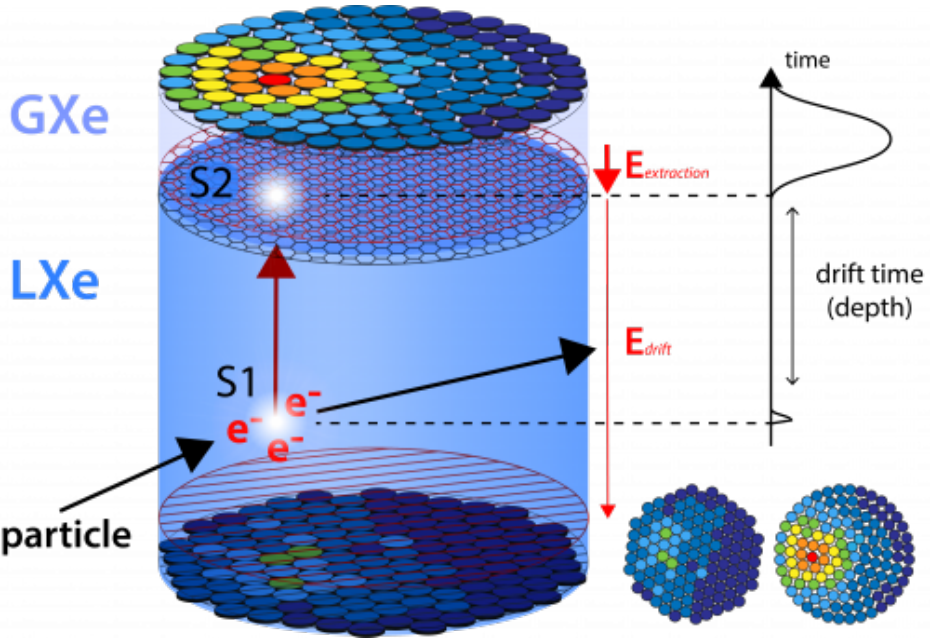
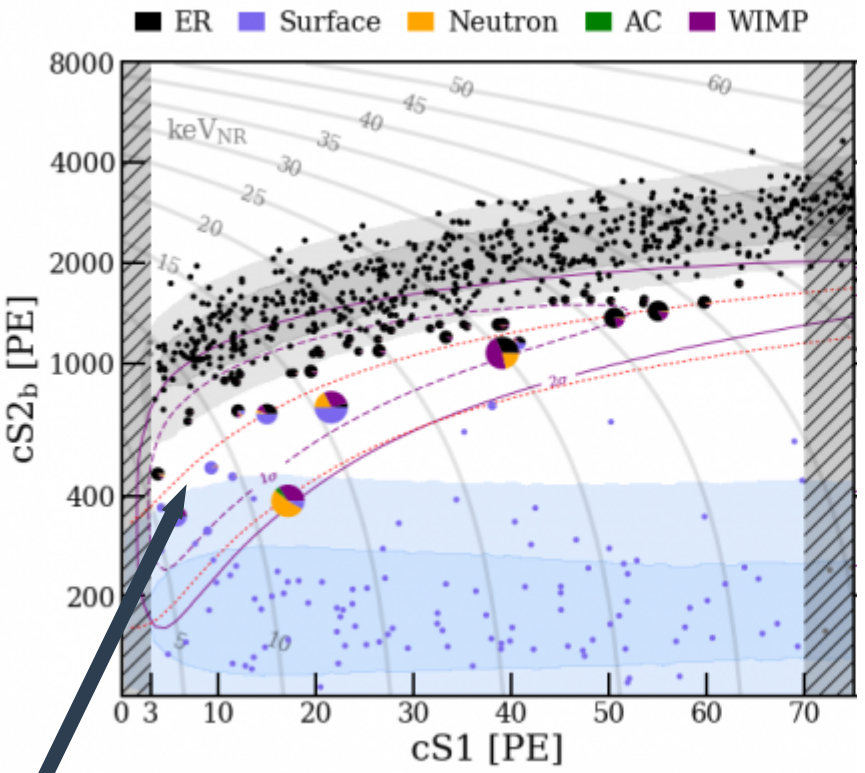
Machine Learning for Classification and Data Analysis in XENON1T

Matthew Bernstein

Dark Matter Summer School
University at Albany
7/18/18

Electronic/Nuclear Recoil Discrimination





- Red line is the NR signal reference region (Historically, throw out all events outside of it – now replaced with Profile Likelihood estimation (PL))
- Area between mean and -2σ of NR distribution
- Trade-off between decreasing WIMP signal (50% NR acceptance) and decreasing ER background (99.7% ER discrimination). These are good figures of merit for discrimination tasks

Interlude: Binary Classification

Supervised Learning

Each training example has x: “features”, y: “output”

Let y = f(x,θ), optimize over parameters θ by minimizing L(y,f(x,θ)) “loss”

Known y: “train set”, unknown y: “test set”

Assume these are statistically independent, but drawn from the same probability distribution!

Regression: y continuous

Classification: y in {1...k}

Binary classification (k=2) is probably the oldest and best understood ML task

And we happen to have a perfect instance of it

What features do we have available?

Current NR boundary is drawn and PL is fit using only 2 parameters – surely we can be more data-efficient than that!
ER and NR distributions overlap in S1/S2 space. The hope is to find more, independent basis vectors in feature space to achieve better separation. If we do, we can try to isolate them and use them as parameters in PL, etc.

Info in a typical minitree

Cuts passed/failed

Position information (x,y,z,r)

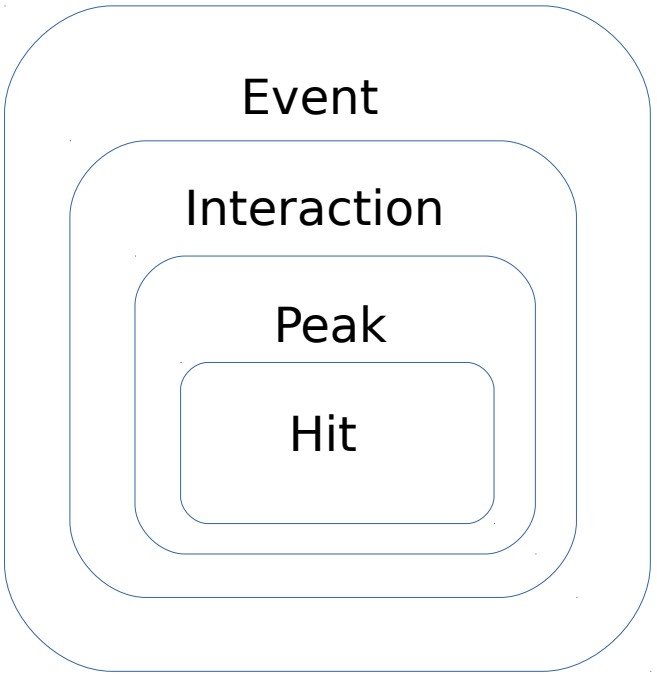
Timestamps

Nonuniform field corrections to S1,S2,x,y,z,r

Peak and pulse-level statistics

Unique IDs

Neighboring events in space and time



http://xenon1t.github.io/pax/

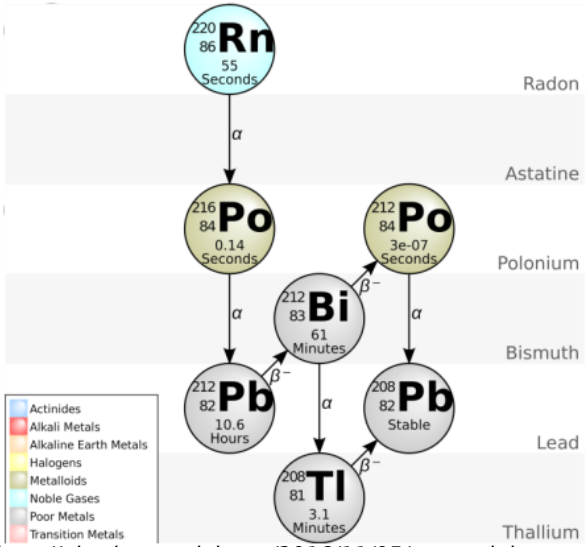
Dataset

25k events of calibration data from Nov 2016

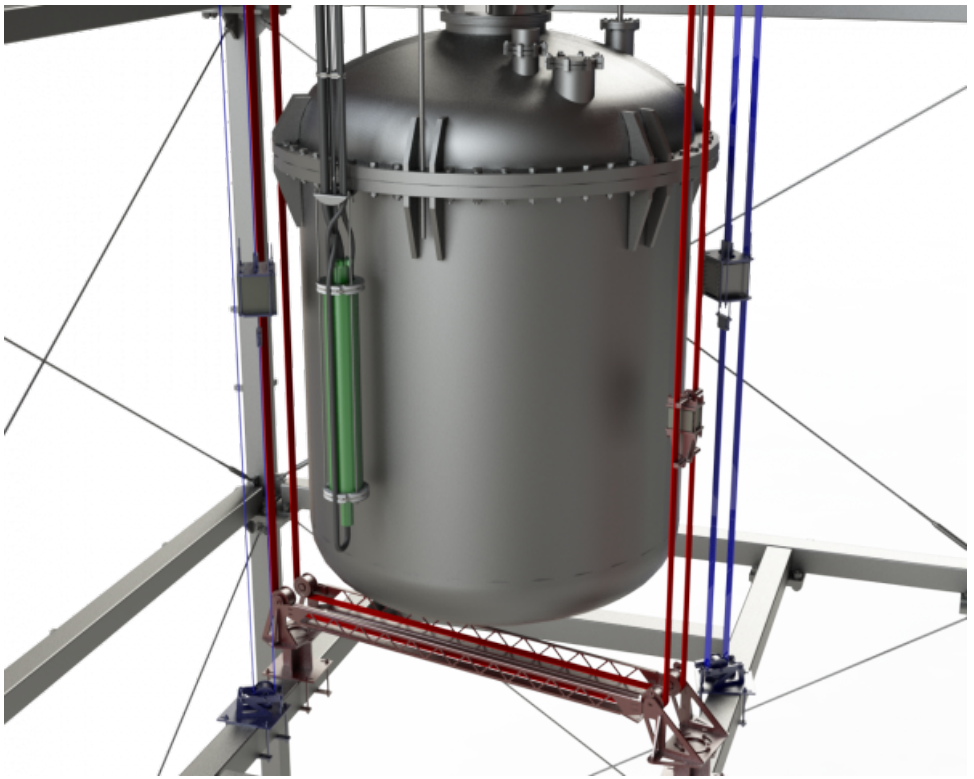
ERs: Rn220 (injected, lifetime ~10 days)

NRs: AmBe (collimated), neutron generator (uncollimated)

137 potential features from standard treemakers



http://physicsopenlab.org/2016/11/05/some-alpha-spectra/



Feature Engineering

General strategy is to only discard things that would count as data snooping, and keep all the rest, let the algorithm figure out what's useful. But how do we tell what it actually used? That comes later...

Invalid Features

137 –-> 32 features

Unique ID, timestamp

Anything that encodes position in detector

Any interactions involving decay products

Anything that distinguishes between AmBe and neutrons (NR sources)

Scaling

Fit each feature to a distribution and learn on scaled/normalized data whenever necessary (when feature range is large, but distribution localized)

Many models start with linear weights, so $|x_i| >> |x_j|$ means x_i dominates x_j in importance

Make more of the weights' range respond to bulk of x instead of outliers

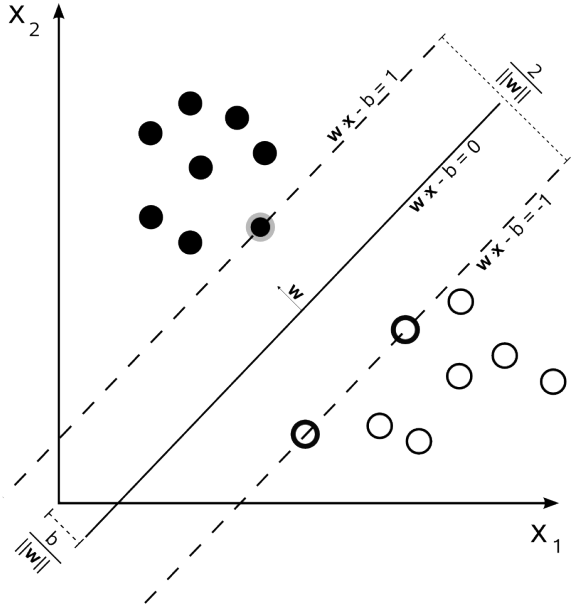
Model

The choice of model actually matters much less than the choice and processing of features

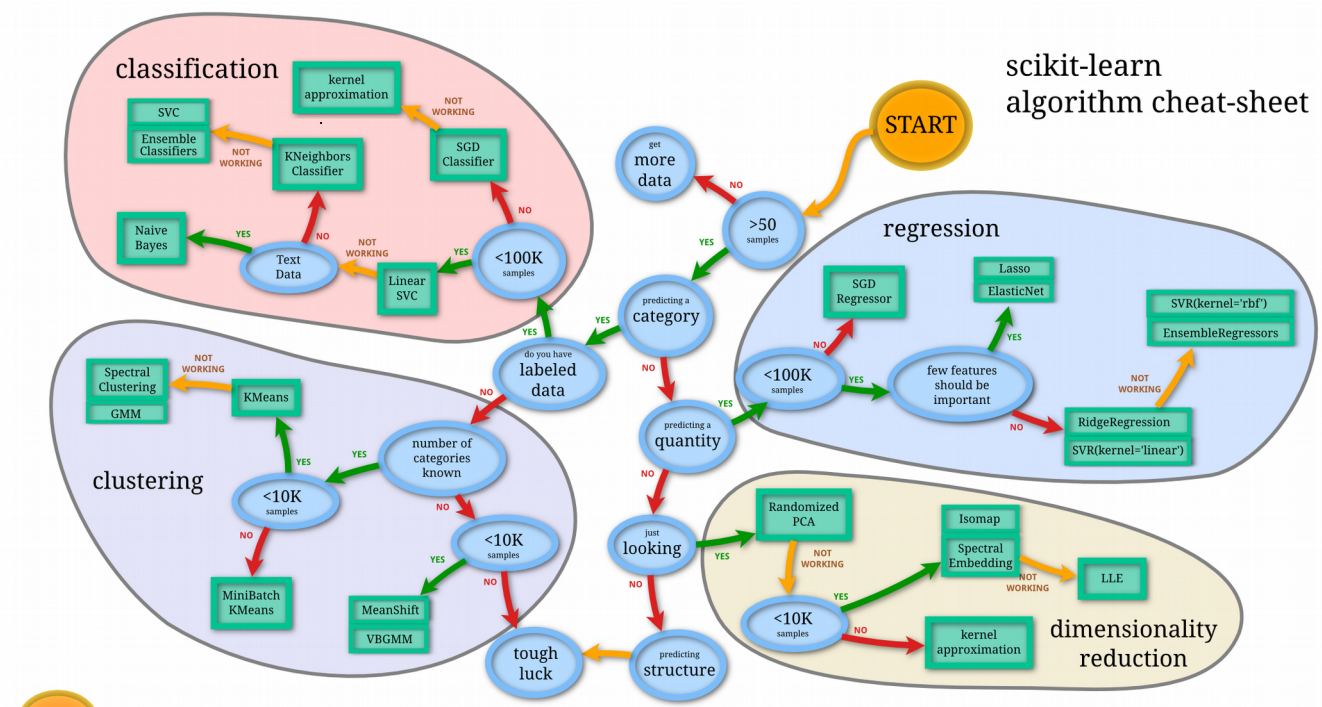
This is a small-to-medium dataset (O(10^5)) with few features (O(10))

Out of scikit-learn, best results are from a **Gaussian Support Vector Machine (SVM)**

$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma^2}\right)$$



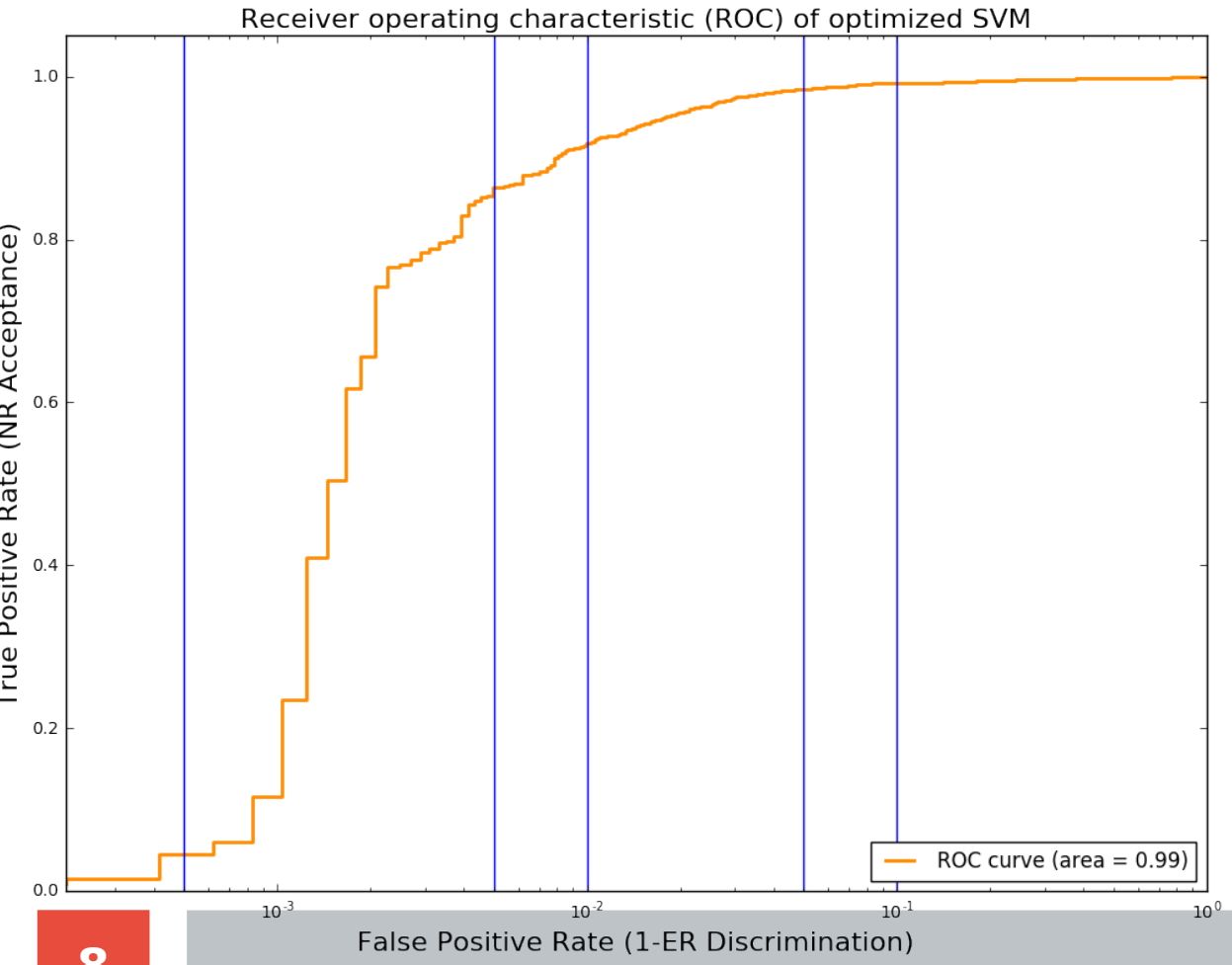
https://commons.wikimedia.org/wiki/File:Svm_max_sep_hyperplane_with_margin.png



http://scikit-learn.org/stable/tutorial/machine_learning_map/index.html

Results

ER Rejection	NR Acceptance	P(ER) Threshold
90.0%	99.0%	92.5%
95.0%	98.7%	85.2%
99.0%	94.4%	31.8%
99.5%	89.8%	13.4%
99.95%	16.1%	0.0%

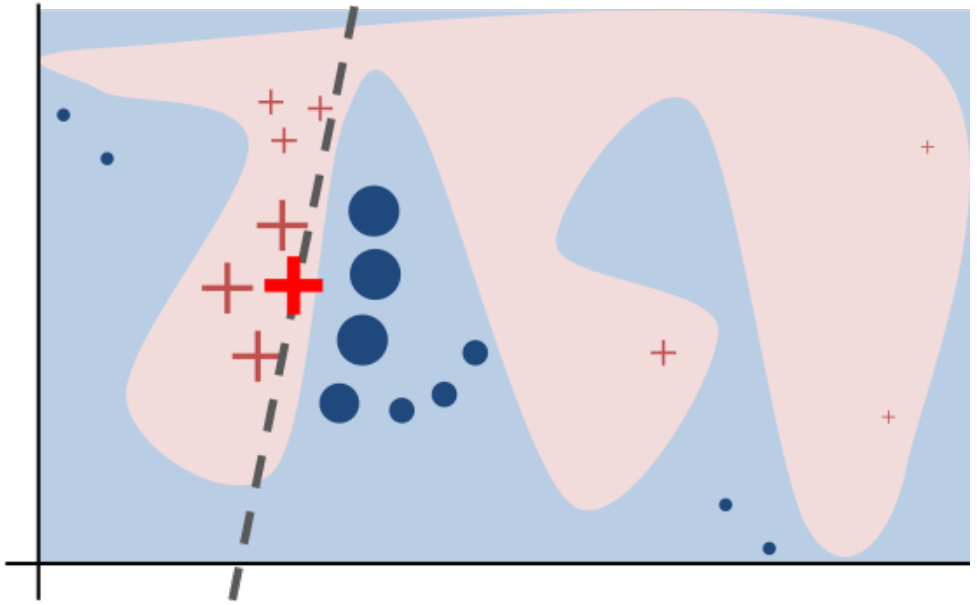


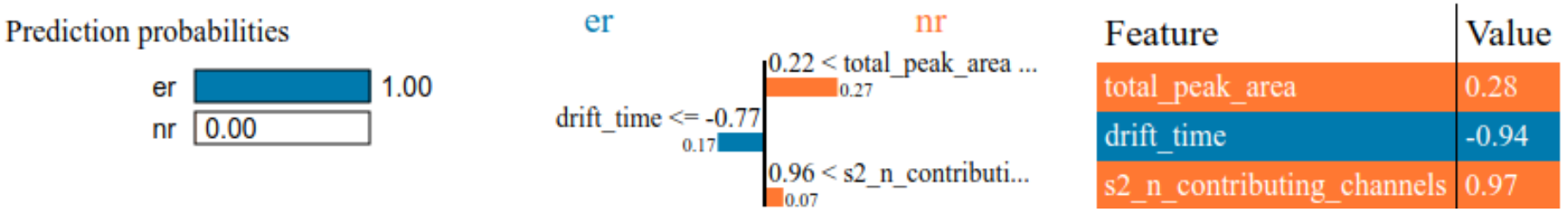
['CES', 'DISC', 'area_before_main_s2', 'cs1', 'cs1_tpf_2dfdc', 'cs2', 'cs2_bottom', 'cs2_top', 'drift_time', 'event_duration', 's1_area_fraction_top', 's1_area_fraction_top_probability', 's1_largest_hit_area', 's1_n_contributing_channels', 's1_pattern_fit', 's1_range_50p_area', 's1_range_80p_area', 's1_rise_time', 's1_tight_coincidence', 's2_area_fraction_top', 's2_area_tailcut_set_by', 's2_lifetime_correction', 's2_n_contributing_channels', 's2_over_tdiff', 's2_pattern_fit', 's2_range_50p_area', 's2_range_80p_area', 's2_rise_time', 'sum_s1s_before_main_s2', 'tailcut_set_by', 'total_peak_area']

Interpretation

Local Interpretable Model-Agnostic Explanations (LIME)

https://github.com/marcotcr/lime





ML For Analysis

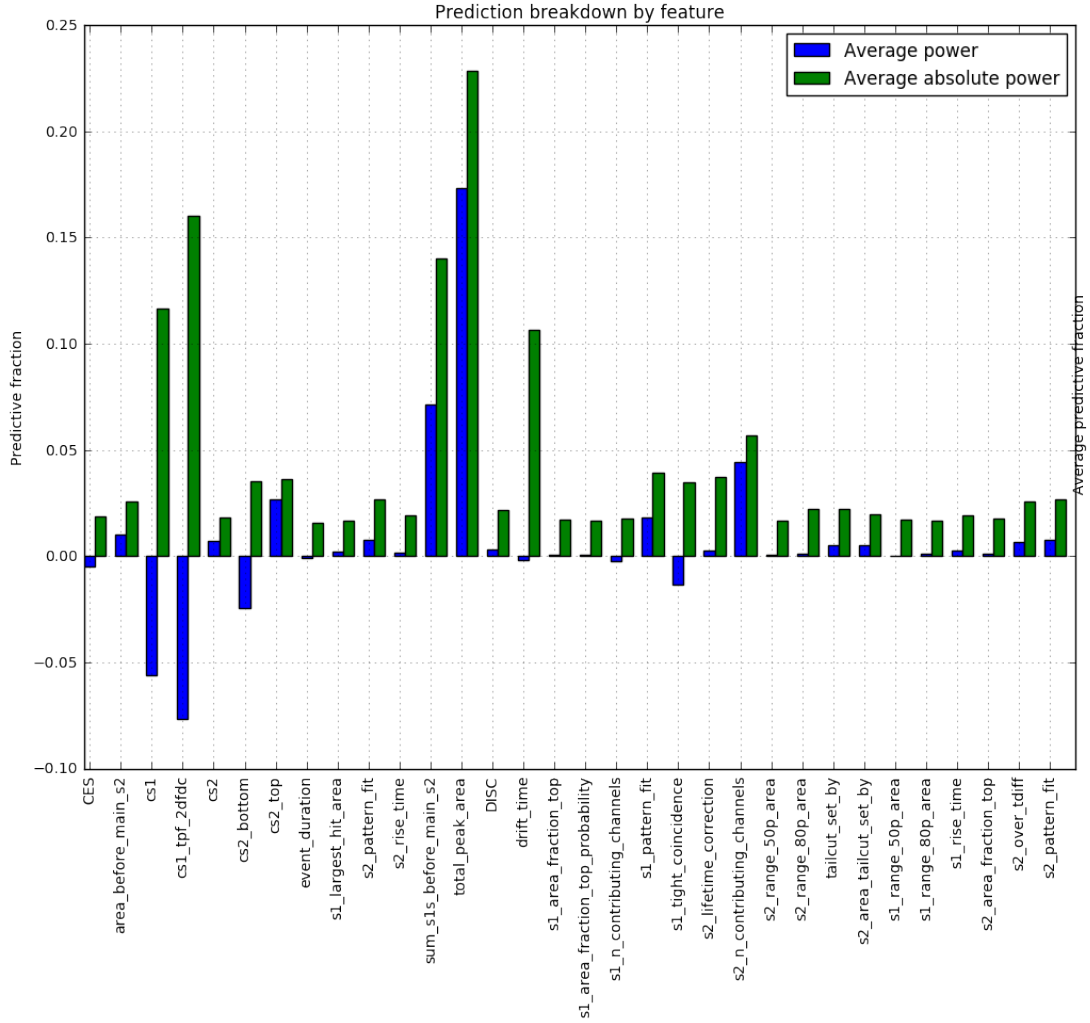
We can do other things besides classify with ML

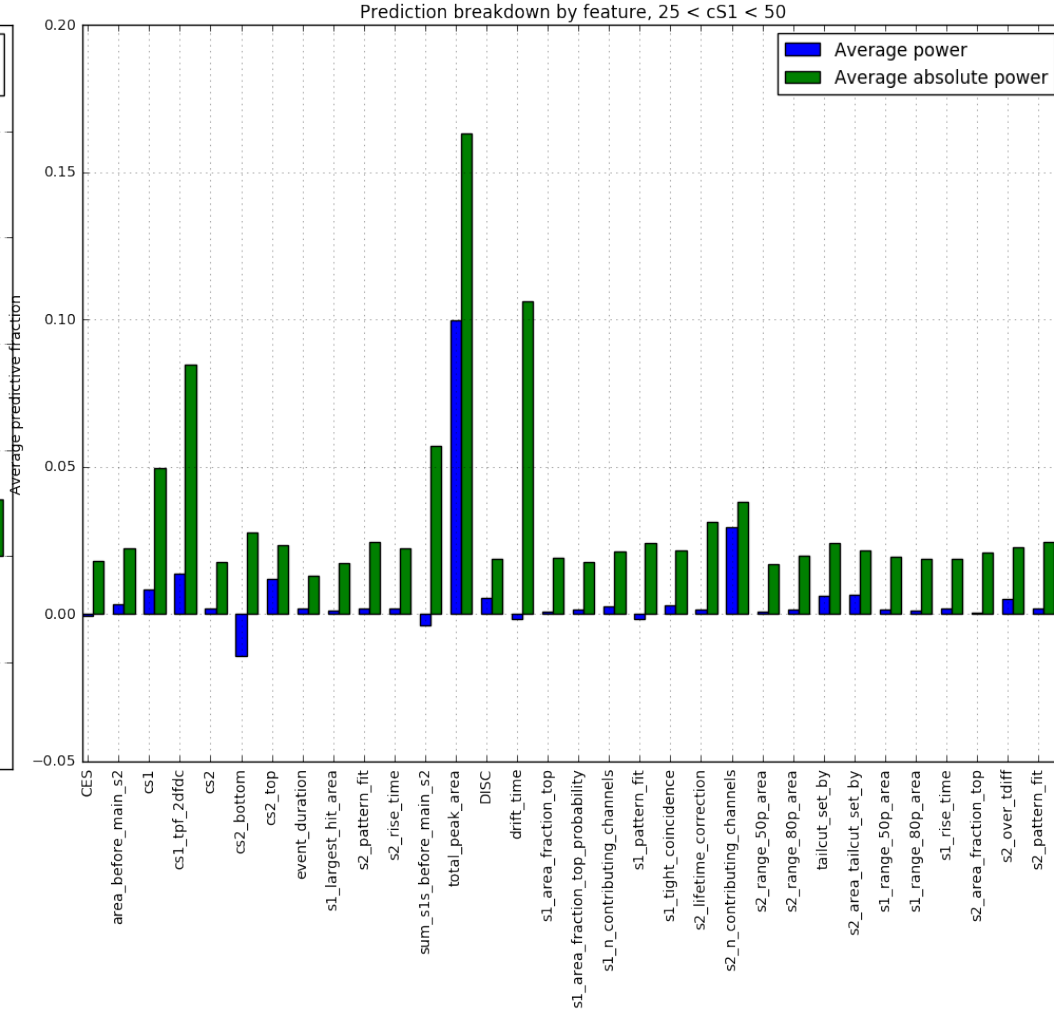
Unsupervised learning (clustering, dimensionality reduction)

Open up models to see what information is being used

Augment human analysis to characterize specific sources of background (wall events, accidental coincidence) and improve cuts targeted at them

Example: low S1 focus





Conclusions and Further Work

The only part of the ML workflow that actually uses physics domain knowledge is preparing inputs to feed to the model (the hard part).

ML has a comparative advantage when the problem is: high dimensional, lots of extraneous info of dubious relevance, poorly understood/modeled.

Although ER/NR discrimination is a textbook binary classification problem, it is none of those things, so very hard to beat humans – we are already doing this very well!

Conversely, when a background is less well understood in terms of physical motivation, ML could help decide where to look or be used directly as a cut.

We don’t collect many independent sources of information – for a single interaction, just S1 and S2 waveforms + drift time.

Models are opaque, but there are tools to open them up and help decide whether to trust them. Could apply to e.g. position reconstruction NN, motivating new classifier-based cuts.