Ideas on anomaly detection for data quality monitoring

James Catmore (Oslo)
with thanks to David Rousseau (LAL), Paul Laycock (CERN),
Katharine Leney (UCL)

Idea

- Can we use machine learning to help with data quality monitoring?
  - Scans reconstructed data lumi-block by lumi-block or run by run and alerts the shifters if the data seems to be anomalous in some way
  - Provides feedback about what exactly is wrong with the dataset
  - Allows shifters and experts to focus their attention to real problems
- This is an *anomaly detection* problem

Types of anomaly detection technique

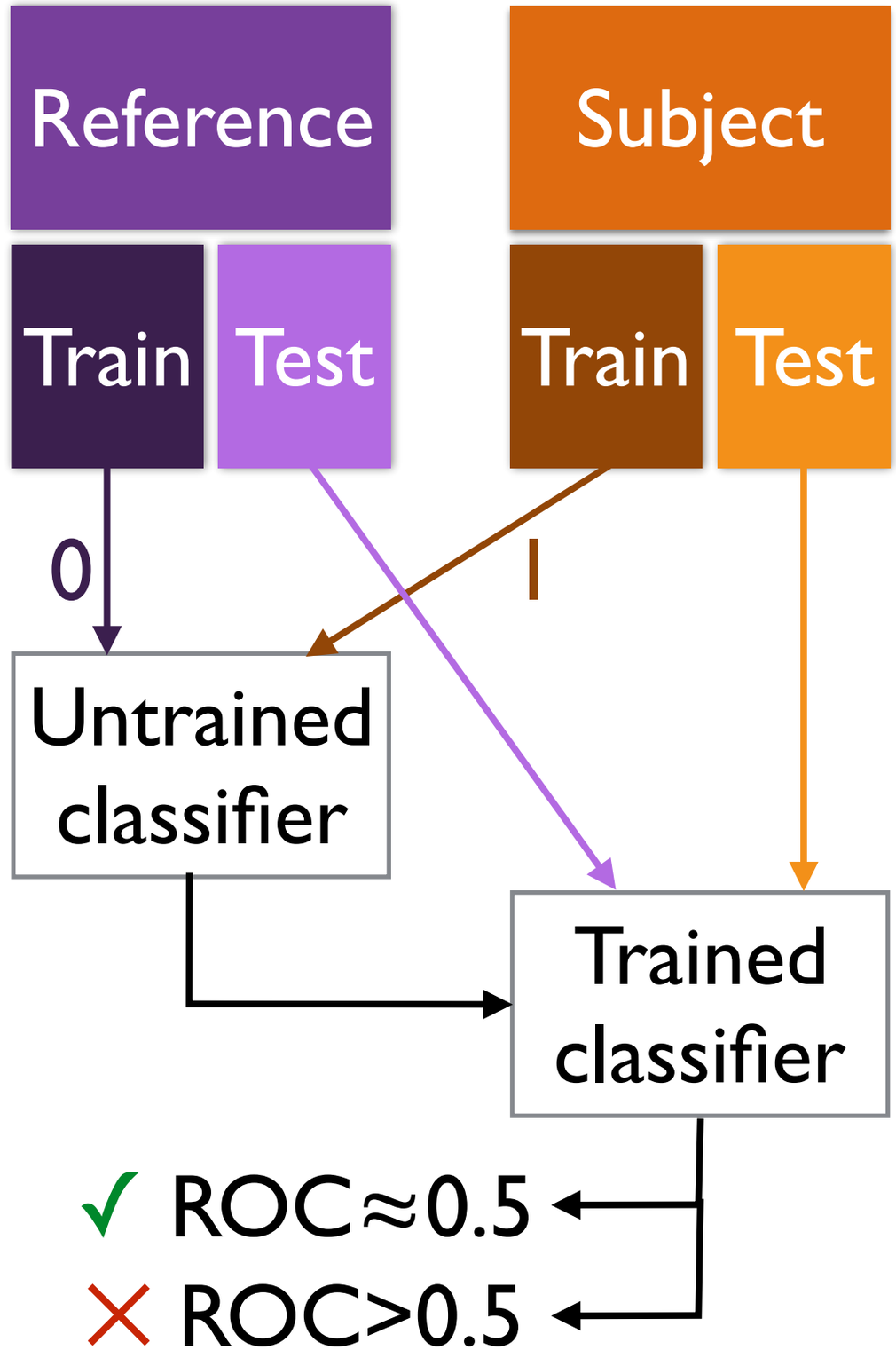
- Density methods
  - kNN, Gaussian mixture model, minimum spanning tree
- Reconstruction methods
  - Auto-encoder
- Boundary methods
  - One class support vector machine
- Classification methods
  - “Split sample” classification
  - Train on historical/simulated anomalies

This talk

- Techniques chosen for study
  - Split-sample based
  - Reconstruction based
- Description of test datasets
- Software setup
- Results using test datasets
- Sketch of how system could work

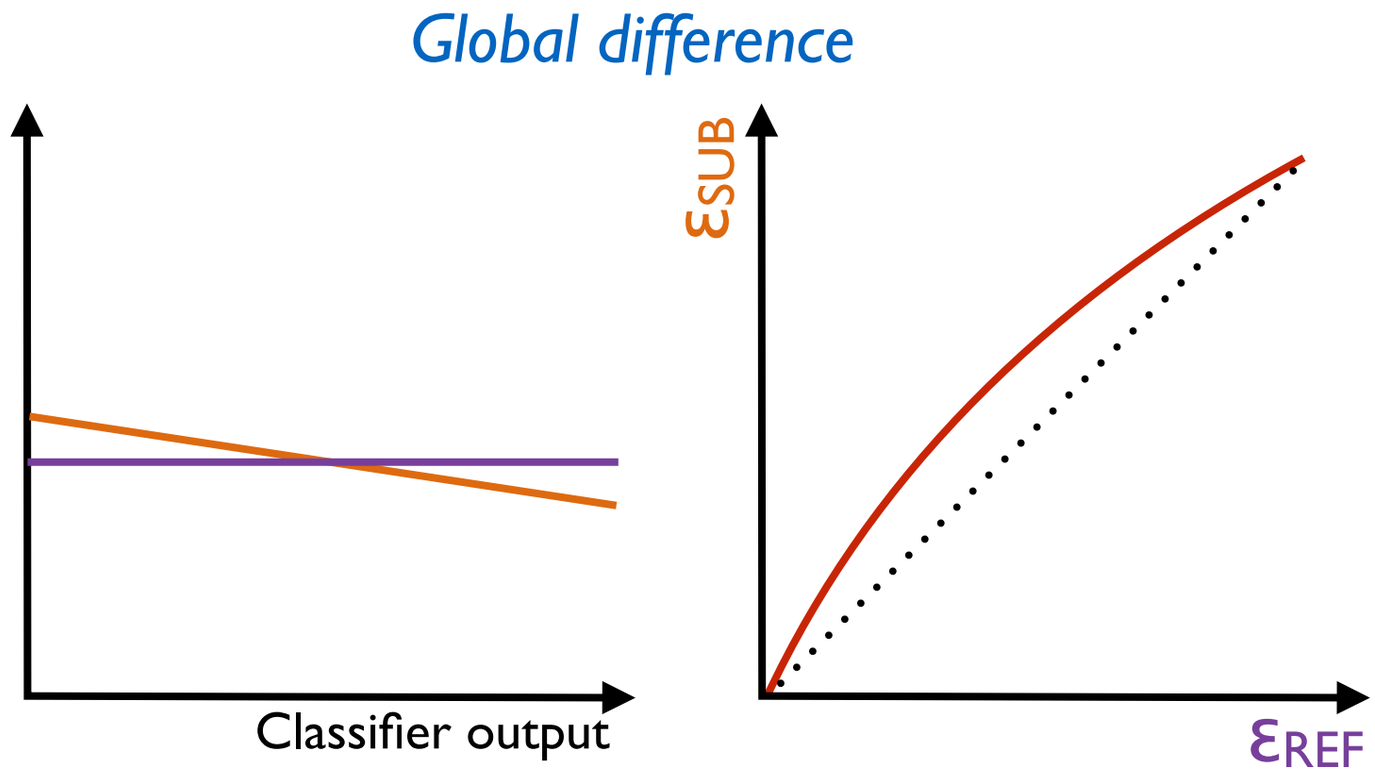
Selected approaches: split sample classifier

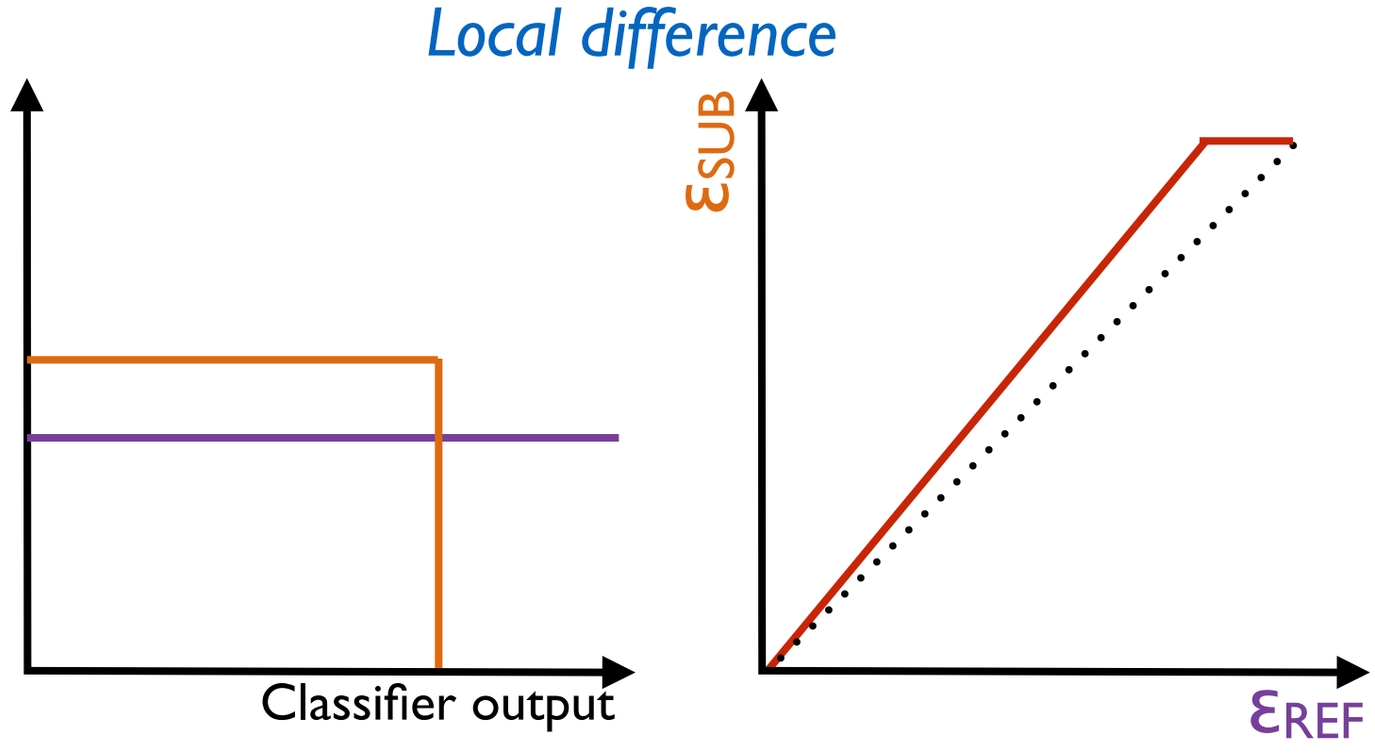
- David Rousseau's suggestion
- Two samples: reference and subject
  - We want to see if the subject is consistent with the reference
- Split both into two parts - training and testing
- Train a classifier (BDT, NN etc) to distinguish between the test and reference (e.g. as if they were "signal" and "background")
- In the testing phase, see if the classifier makes any progress in separating the reference and subject
  - If it does: there is something about the subject sample to distinguish it from the reference
  - If it is supposed to be the same as the reference, clear evidence that something is wrong



Selected approaches: split sample classifier

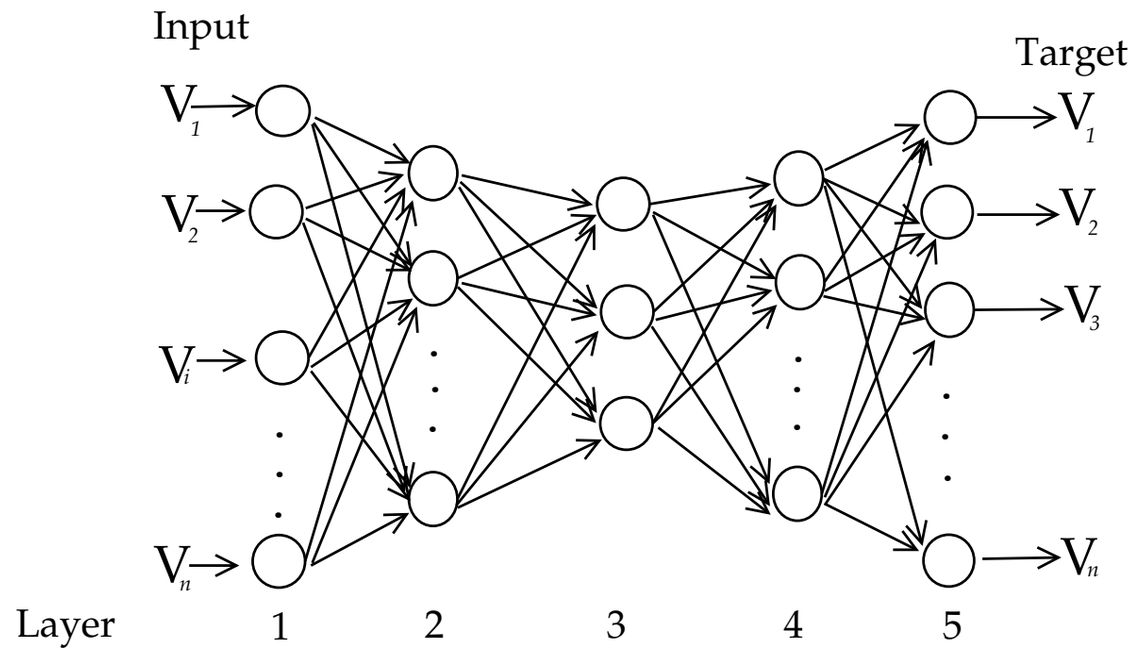
- Shape of the ROC curve may help to understand whether the problem is local or global
- Inspection of the BDT/NN weights may allow us to work out which combination of variables are leading to the separation





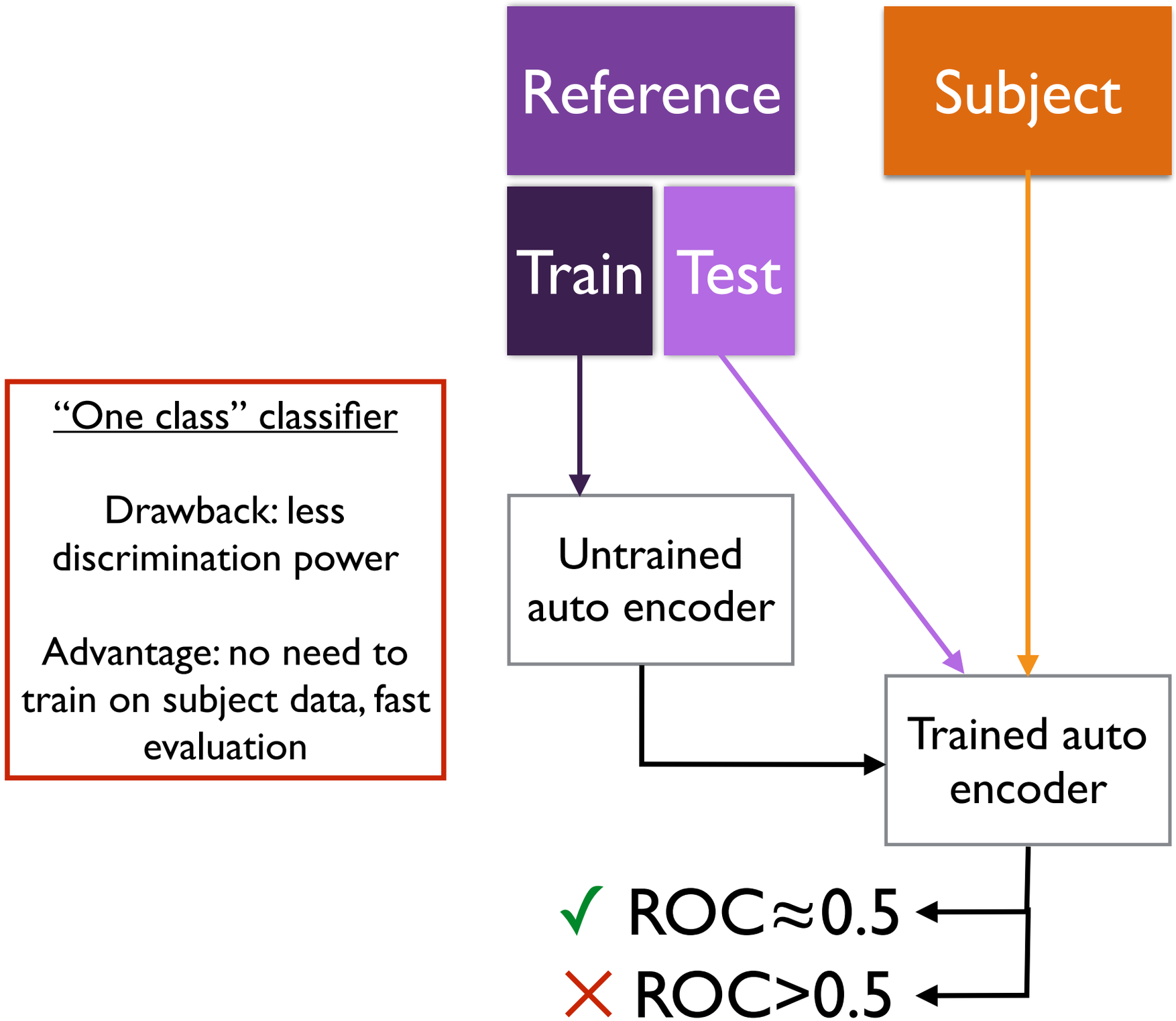
Selected approaches: auto-encoder

- Auto-encoder: NN trained on its own input
  - Usually includes a bottle-neck to compress the features of the data (e.g. PCA)



- Normally used for dimension-reduction but proposed as a means of anomaly detection
- Idea: a trained replicator neural network should reconstruct new examples taken from the bulk (normal) data with low error, but when presented with an anomalous example, will reconstruct it with a high error since it contains qualities that have not previously been encoded
  - Provides a natural measure of abnormality: the reconstruction error (difference between the input and the output)
  - Reconstruction error per event = $\sum_{i=1}^{N}(x_i^{in} - x_i^{out})^2$ N is the number of features

Selected approaches: auto-encoder



"One class" classifier

Drawback: less discrimination power

Advantage: no need to train on subject data, fast evaluation

Description of test datasets (from Paul Laycock)

- Two categories of test sample
  - **Borked**: Absolutely must see a difference (e.g. magnets are off) - for feasibility
  - **Dodgy**: Should see a difference but may be a subtle effect - understand the limitations
- Aim of study: can the algorithms
  - reliably identify LBs from the bad samples?
  - tell us something about what is wrong?

Description of test datasets (from Paul Laycock)

Category	Run	Comment	Dataset
Reference	304008	Reference	data16_13TeV.00304008.express_express.merge.AOD.x446_m1620
Borked	299278	Both magnets are off, King Bork	data16_13TeV.00299278.express_express.merge.AOD.x426_m1594
Borked	296939	First run of 2016, no HLT, lots of mucking around Queen Bork	data16_13TeV.00296939.express_express.merge.AOD.x412_m1576
Borked	299241	Toroid off, borked if you're a muon, otherwise not so borked	data16_13TeV.00299241.express_express.merge.AOD.x426_m1594
Dodgy	296942	Pixel HV scan, dodgy	data16_13TeV.00296942.express_express.merge.AOD.x412_m1576
Dodgy	299584	Pixel Layer2 out of synch until LB~200, bit dodgy	data16_13TeV.00299584.express_express.merge.AOD.x426_m1594
Dodgy	299184	Before the d0 bias was introduced, should look different to reference	data16_13TeV.00299184.express_express.merge.AOD.x425_m1588

Data setup

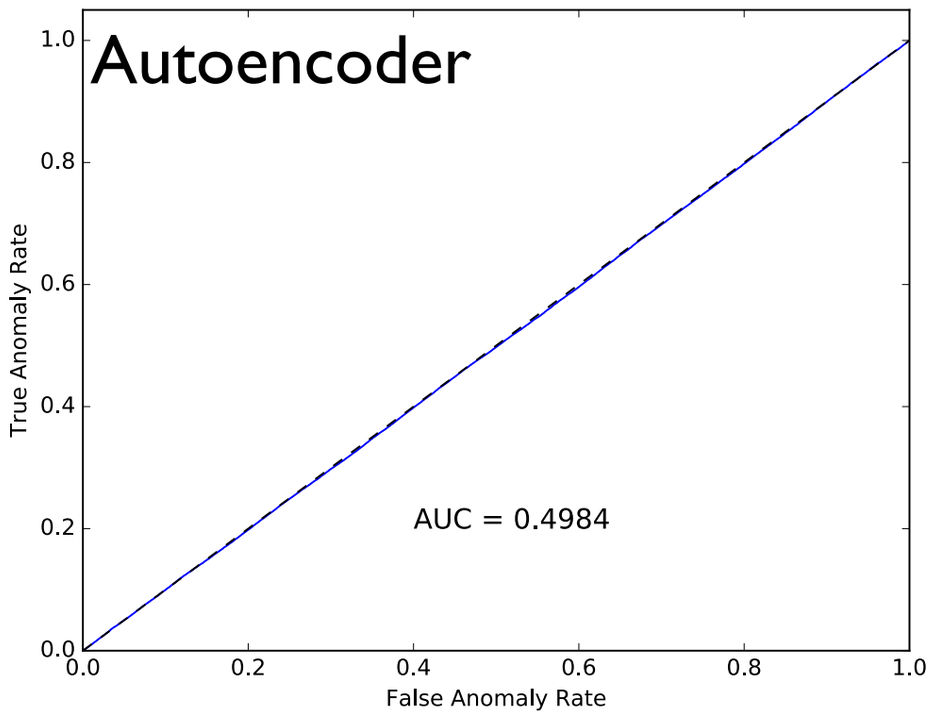
- Data unit: inner detector tracks
  - Why is the event boundary not respected?
    - Avoids having to worry about differing number of tracks per event
    - Events are largely irrelevant here
  - Why ID tracks?
    - Most obvious domain to start with - should be most sensitive to the "defects" in the various datasets listed on the table
- Input variables: all ID track variables
  - excluding {x,y,z} (avoids training on the beam spot position)
  - 71 variables in total - full list in backup
- Reference sample (304008): use GRL lumiblocks (LB range 775-784)
- Subject samples: LB selected at random since none are in GRL
- 100K tracks per sample used for training, same number for testing
  - except the auto-encoder which doesn't see the subject sample during training
- In the testing phase we are asking the algorithms to tell us whether a given track is from the reference dataset, or the subject dataset

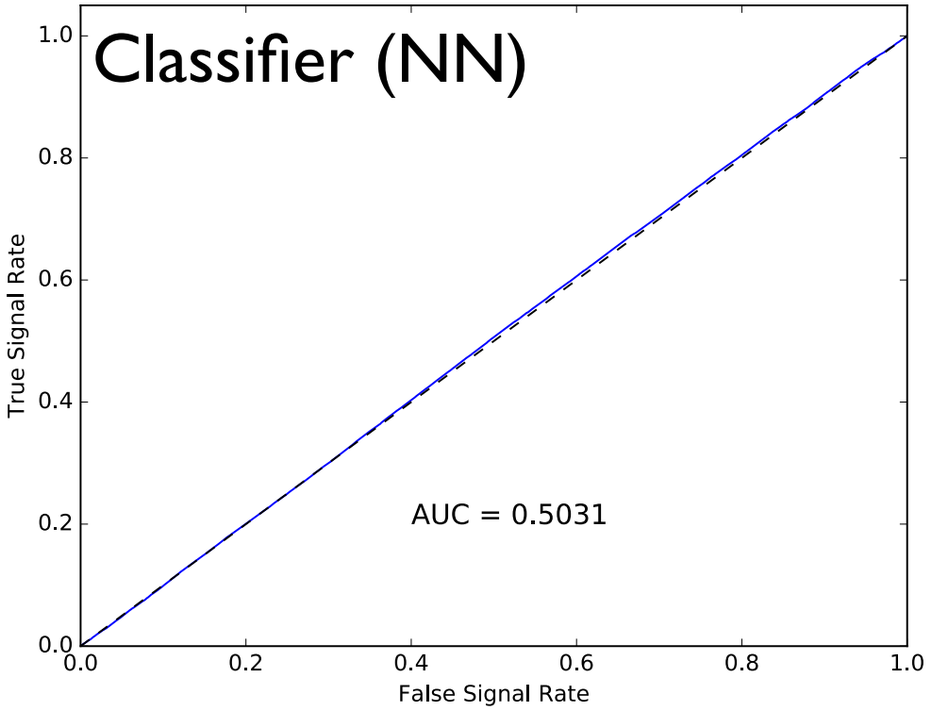
Software set-up

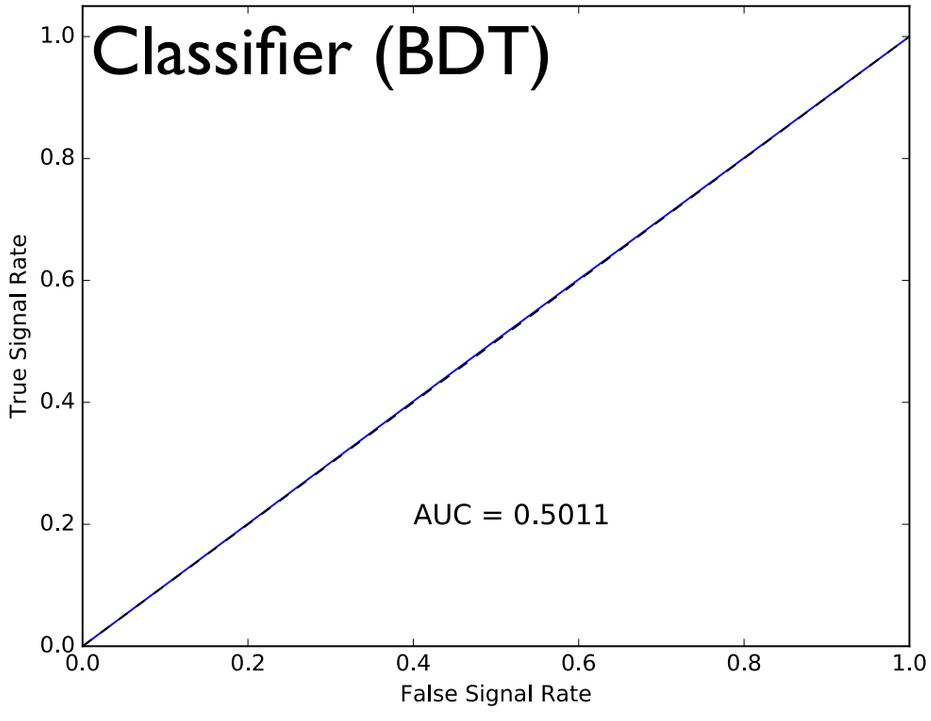
Technique	Algorithm	Software	Hyperparameters	Preprocessing
Split-sample classifier	Boosted decision tree	SciKitLearn	AdaBoost Boost cycles: 100 Max tree depth: 1	All variables scaled between 0 and 1 (SKL rescaler)
	Shallow neural network	SciKitLearn Keras TensorFlow	Structure: 71-71-1 Activation: ReLu Epochs: 100 Batch size: 100	All variables scaled between 0 and 1 (SKL rescaler)
Auto-encoder		SciKitLearn Keras TensorFlow	Structure: 71-20-71 Activation: ReLu Epochs: 100 Batch size: 100	All variables scaled between 0 and 1 (SKL rescaler)

Sanity check: reference vs reference

- Test: 304008 LB 775-784
- Subject: 304008 LB 785-794
  - Same run, adjacent LBs, both in GRL

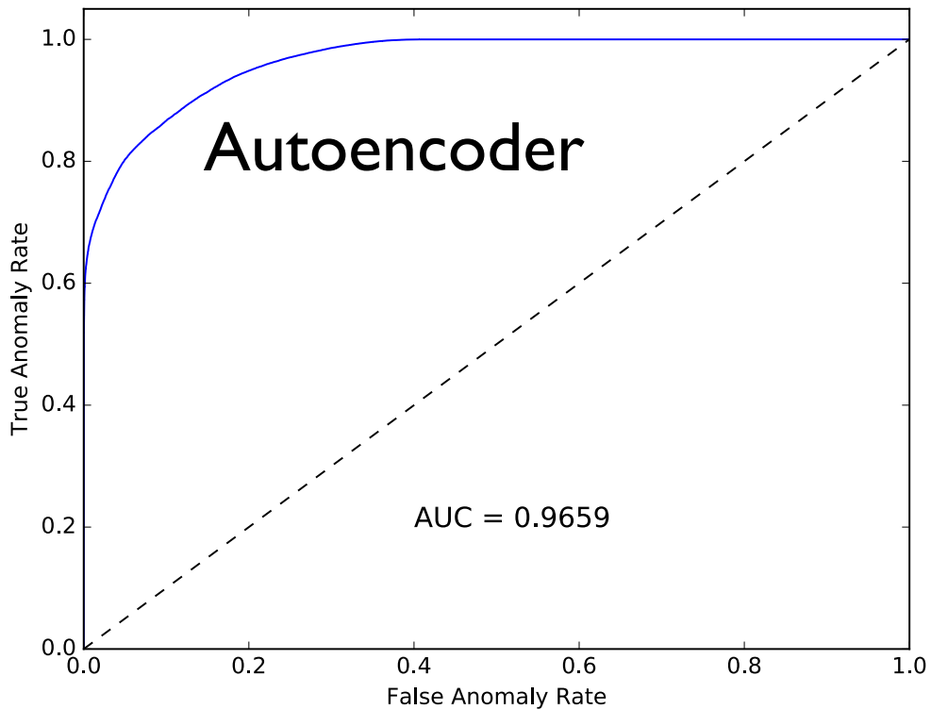


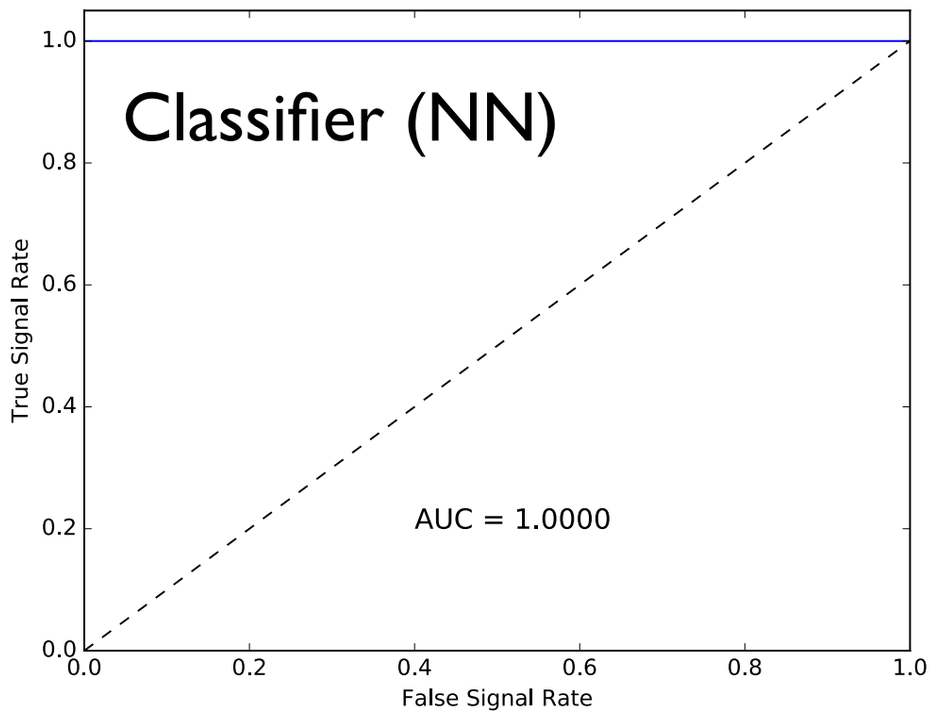


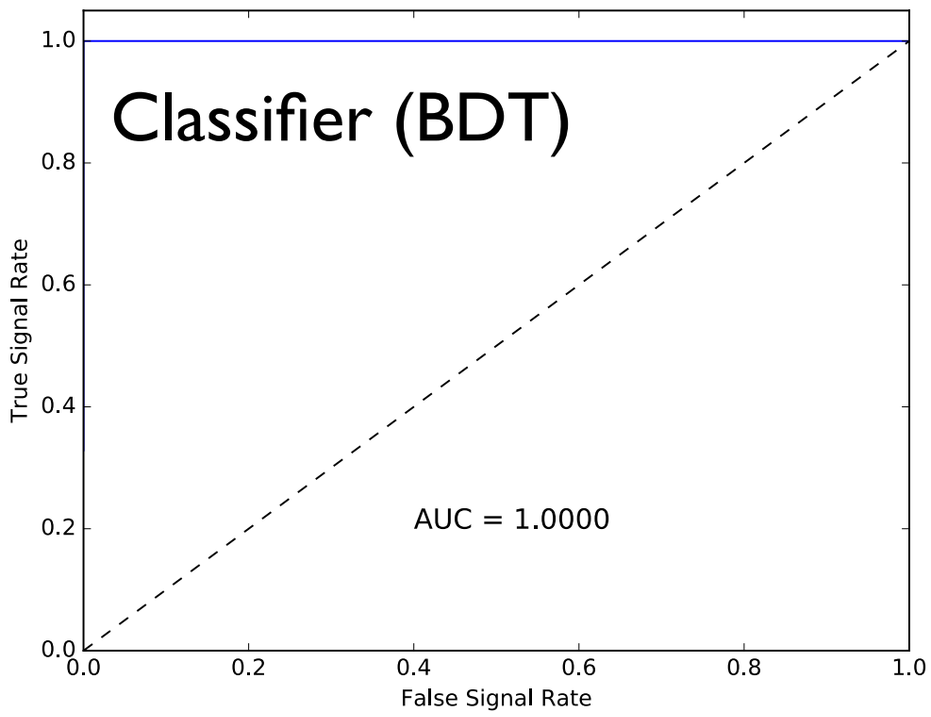


Reference vs zero field ("King Bork")

- Test: 304008 LB 775-784
- Subject: 299278

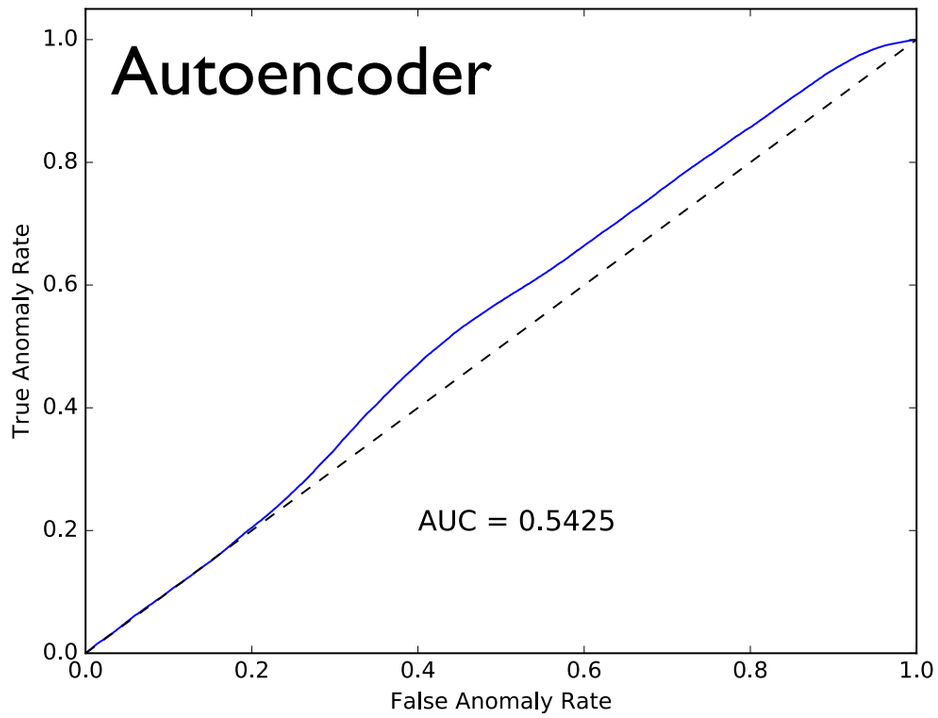


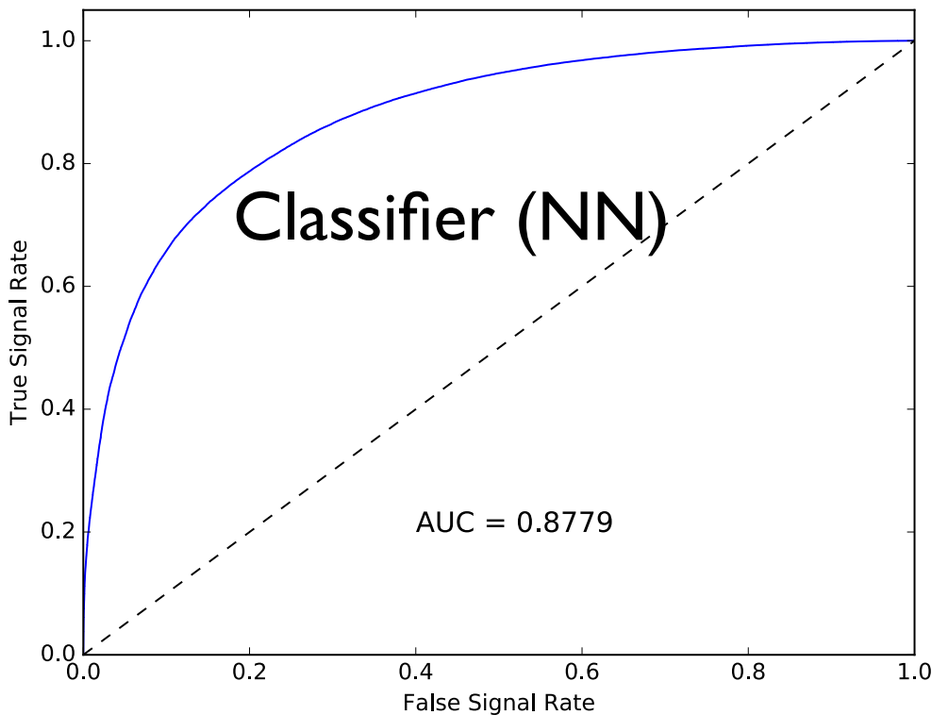


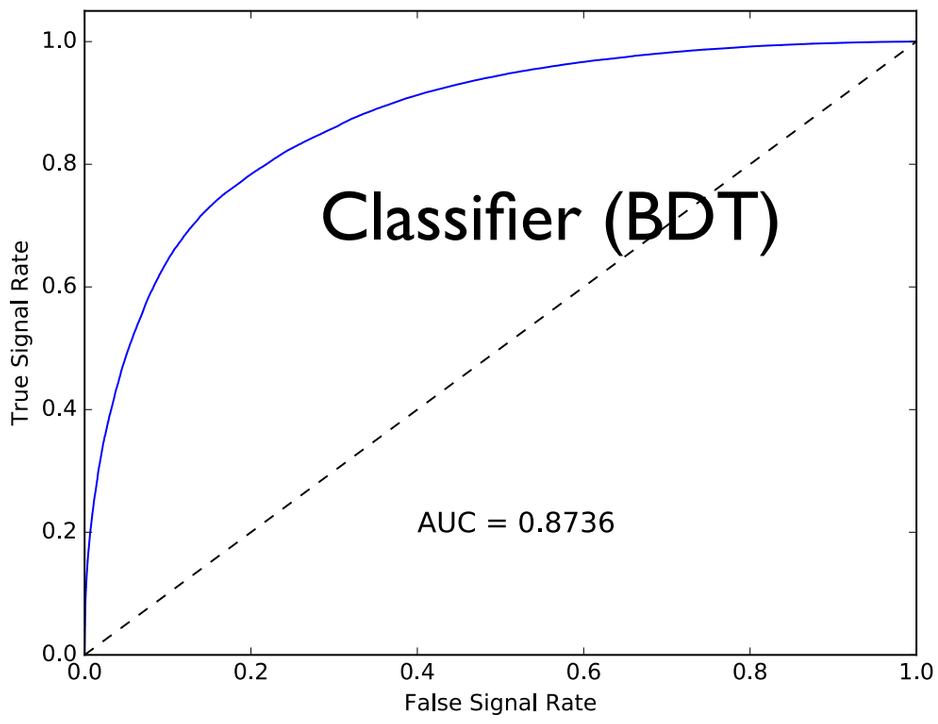


Reference versus "Queen Bork"

- Test: 304008 LB 775-784
- Subject: 296939

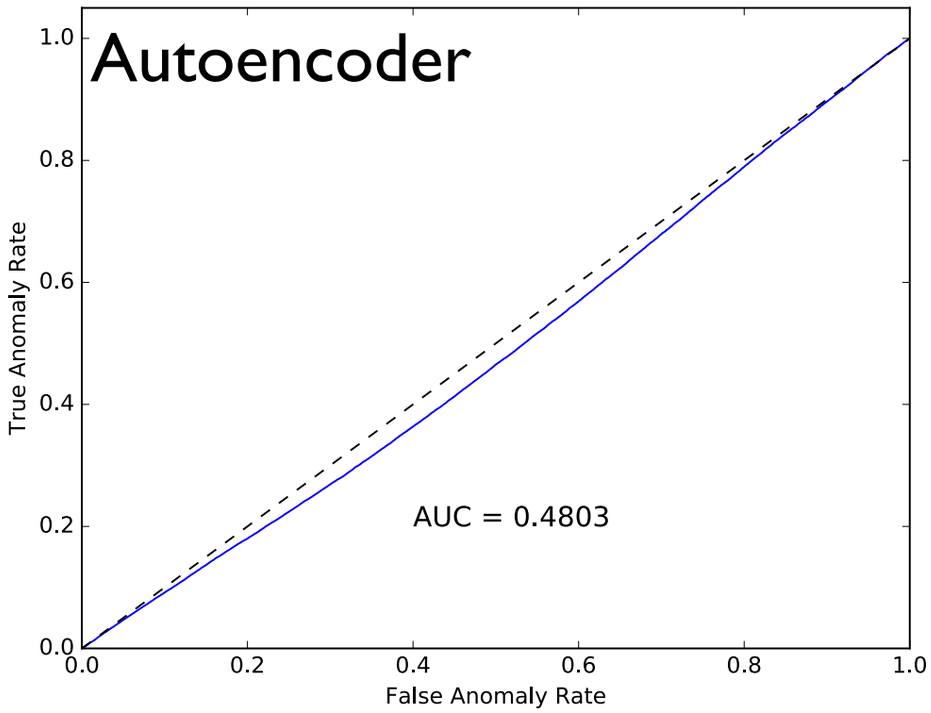


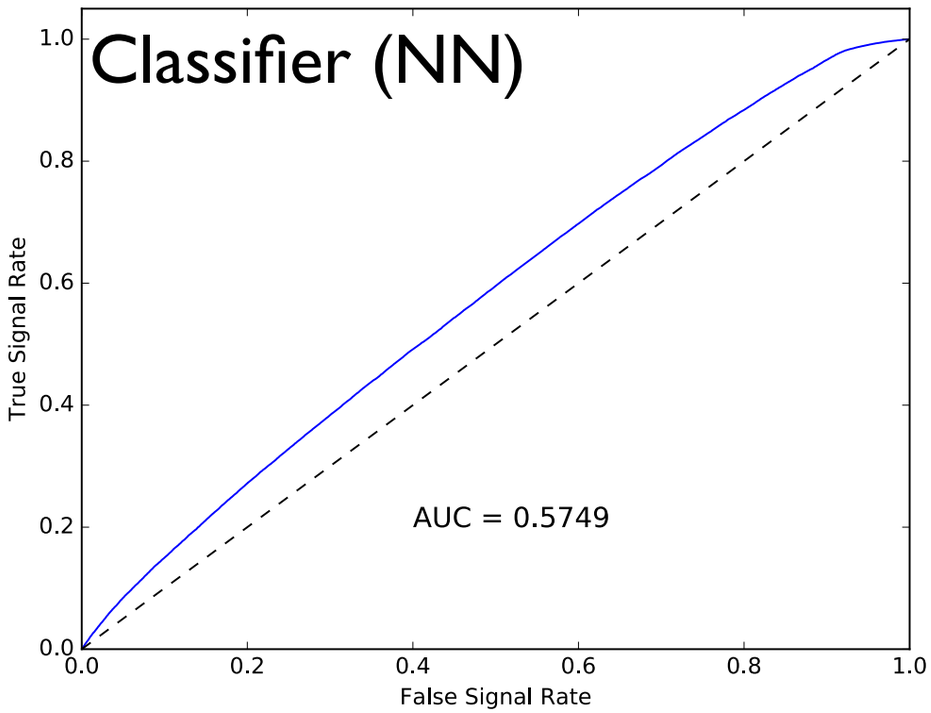


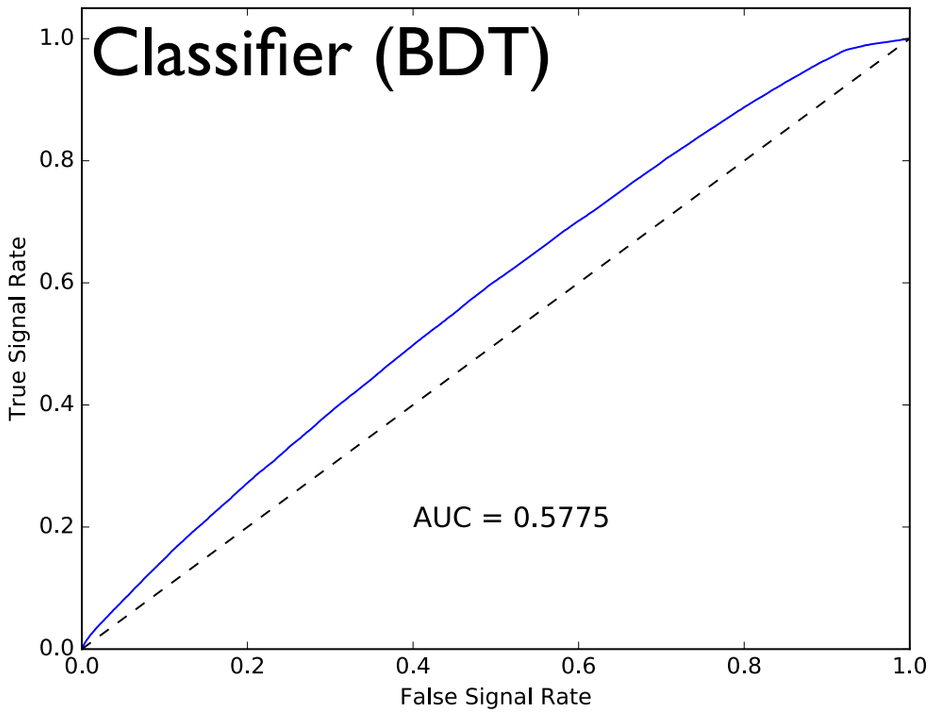


Reference vs no d0 bias

- Test: 304008 LB 775-784
- Subject: 299184

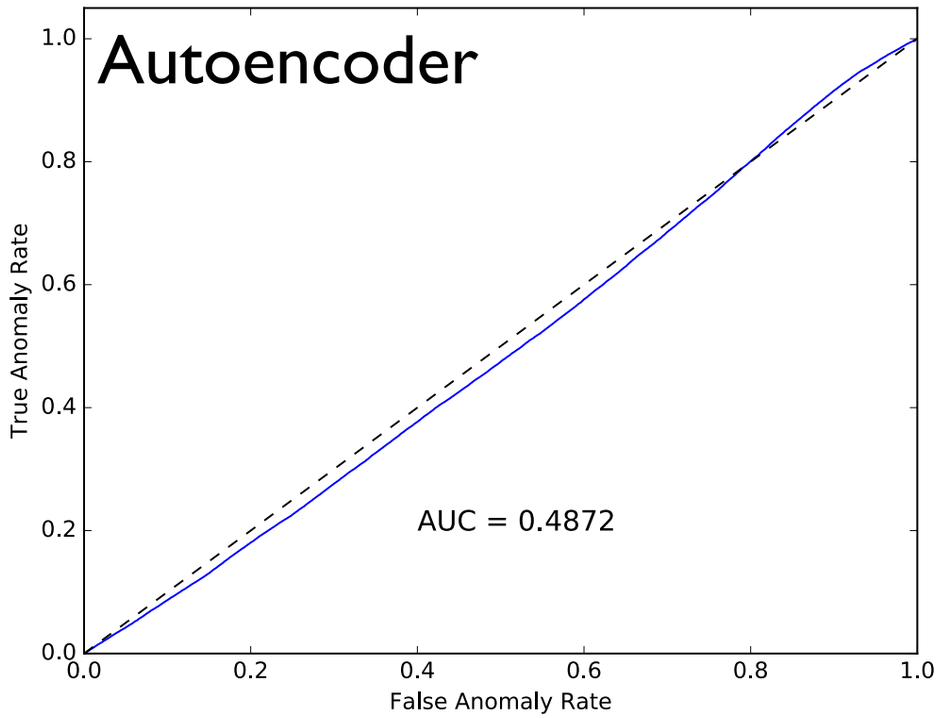


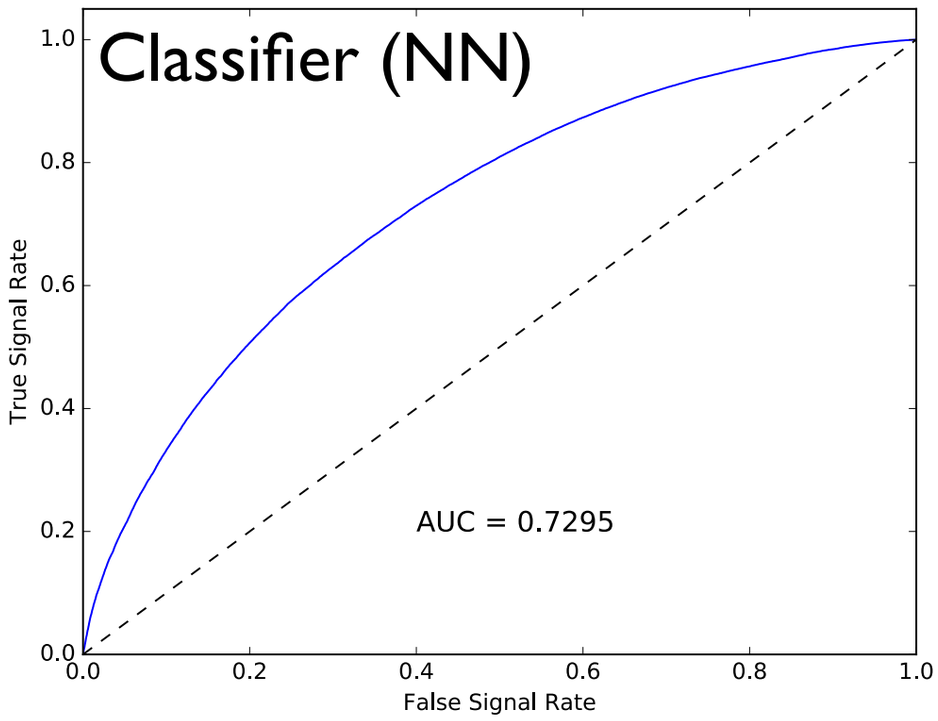


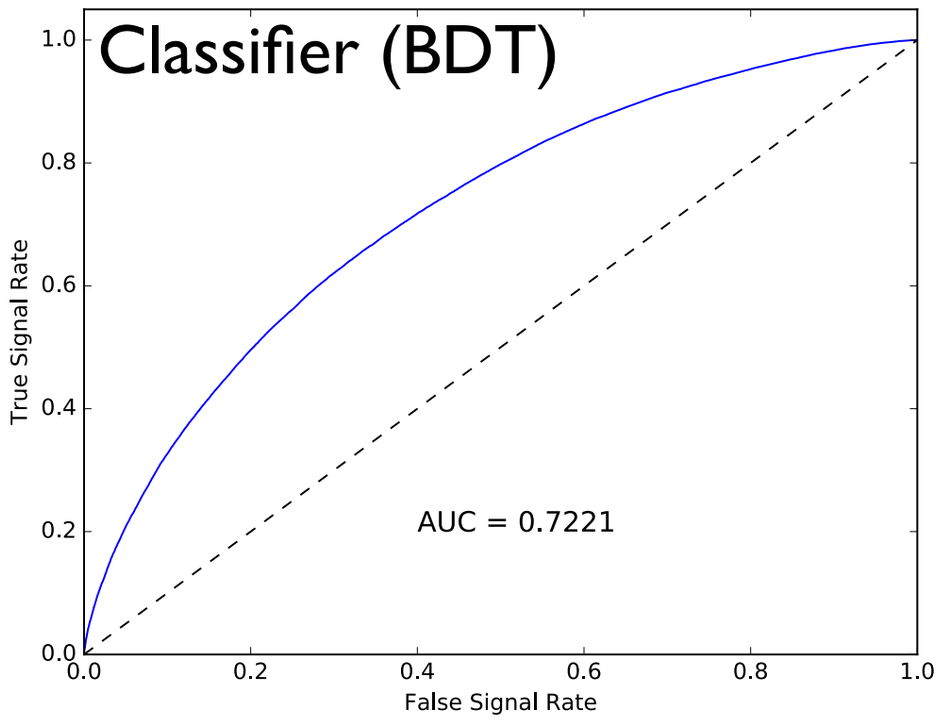


Reference vs HV pixel scan

- Test: 304008 LB 775-784
- Subject: 296942







Success Ambiguous Failure

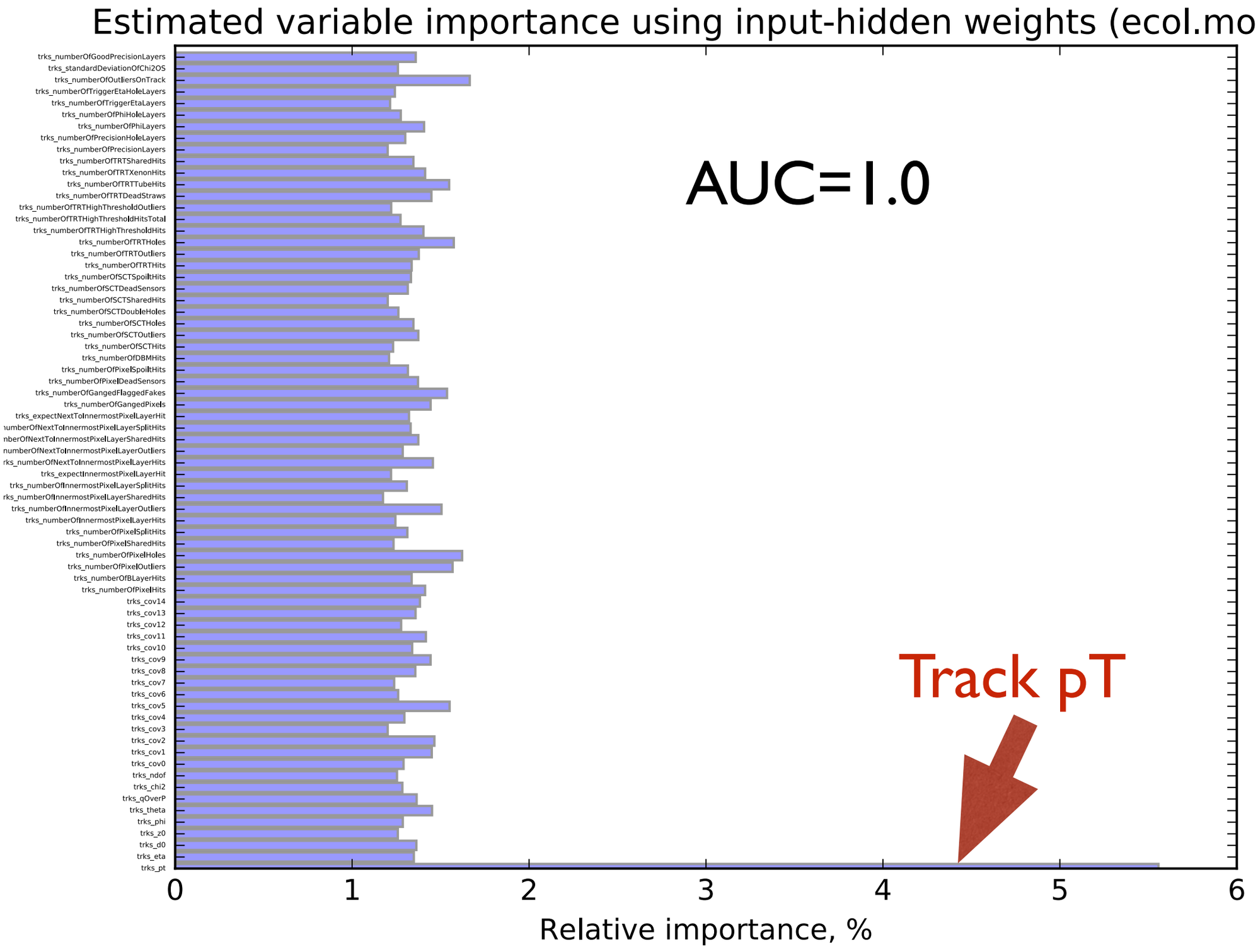
AUC results summary

Reference	Subject	Autoencoder AUC	NN AUC	BDT AUC
Reference LB 775-784 (Ref #1)	Reference LB785-794 (Ref #2)	0.50	0.50	0.50
Ref #1	Zero field (“King Bork”)	0.97	1.0	1.0
Ref #1	“Queen Bork”	0.54	0.88	0.87
Ref #1	No d0 bias	0.48	0.57	0.58
Ref #1	Pixel HV scan	0.49	0.73	0.72

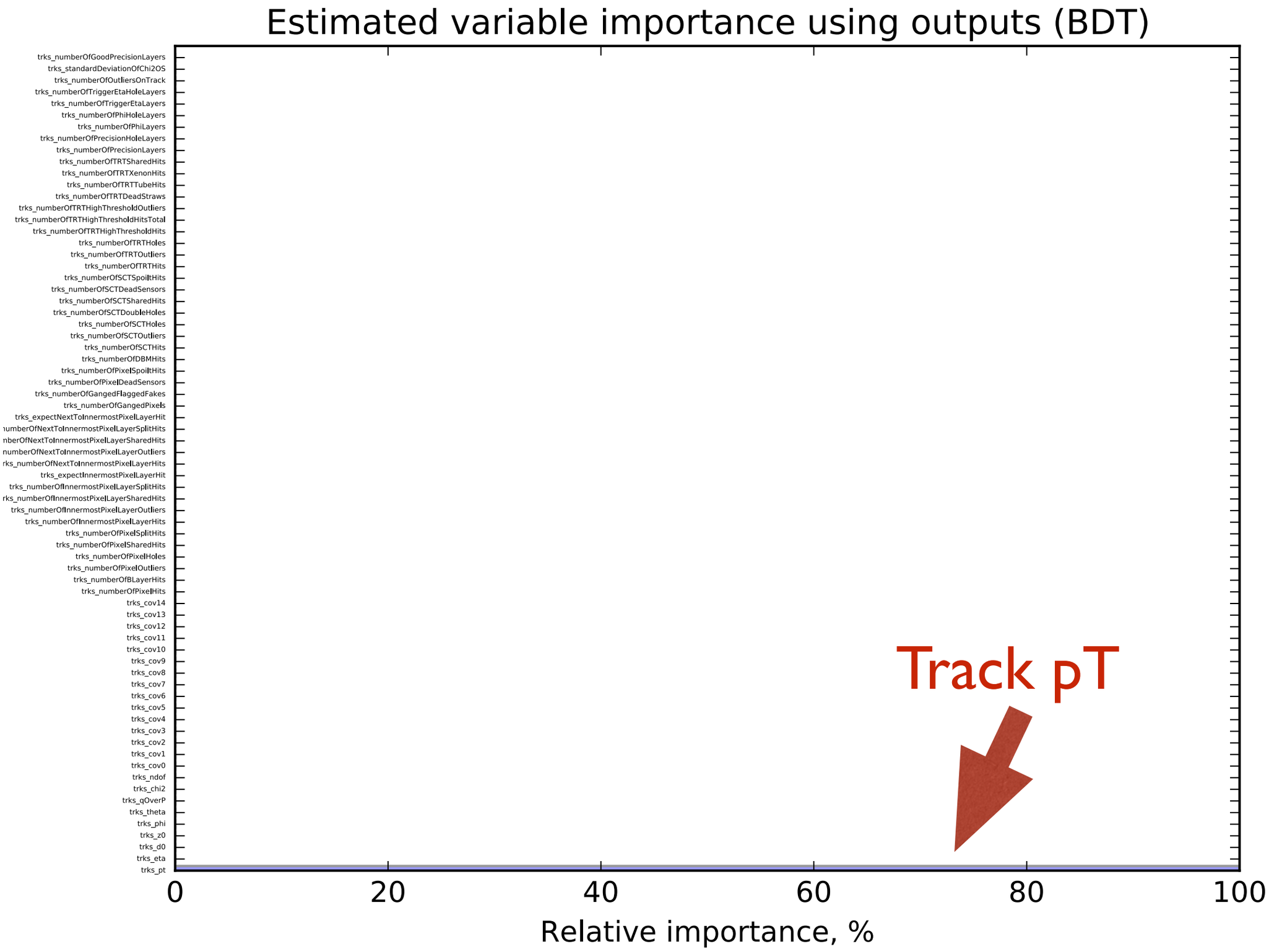
Variable importance

- Can we understand how the algorithm is separating the reference and the subject for the anomalous cases?
  - Shallow NN - "weights method"
  - Autoencoder - contribution to reconstruction error
  - BDT - ranking
- Brief descriptions in the backup slides
- How does this look with our datasets?

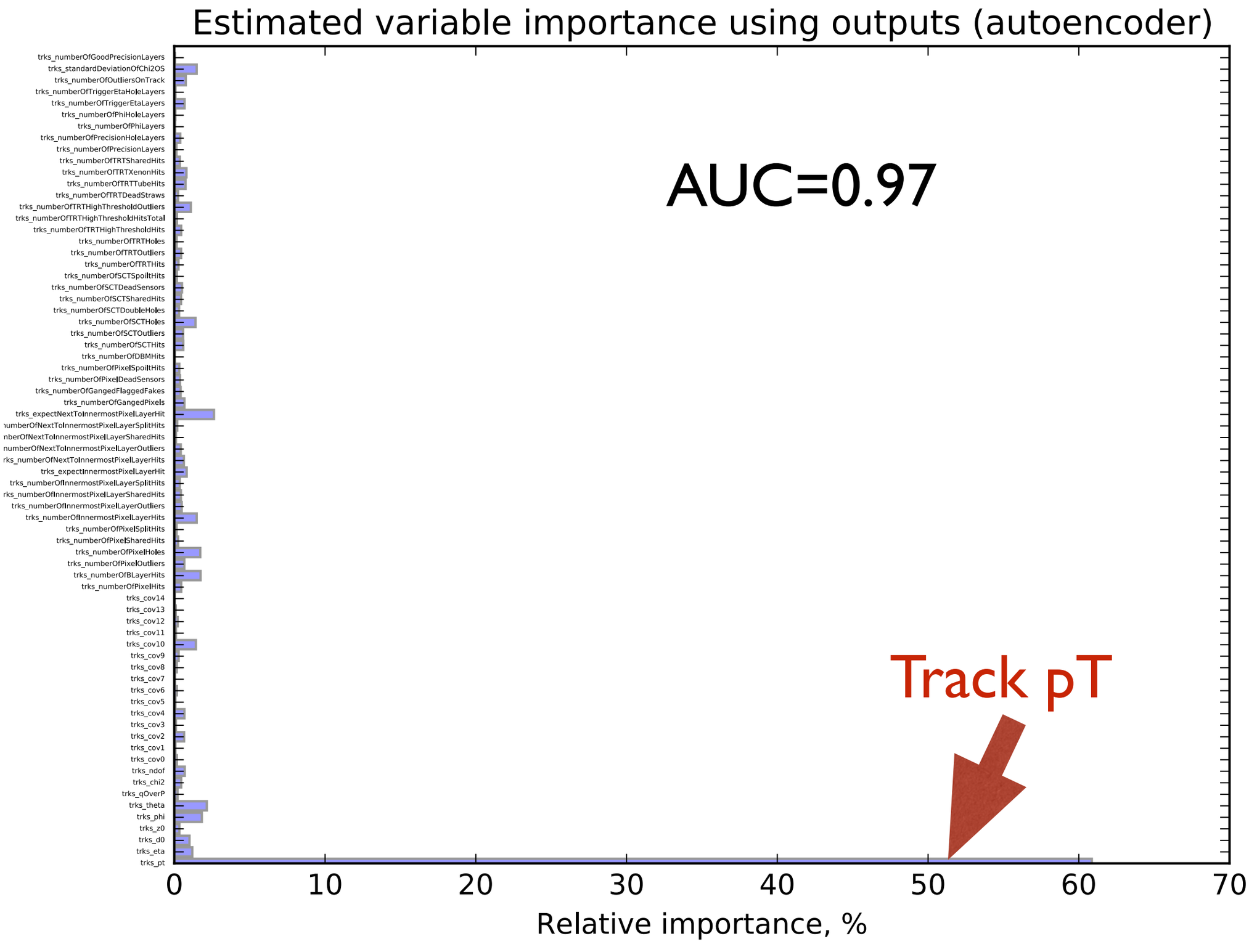
Variable importance: zero field (NN classifier)



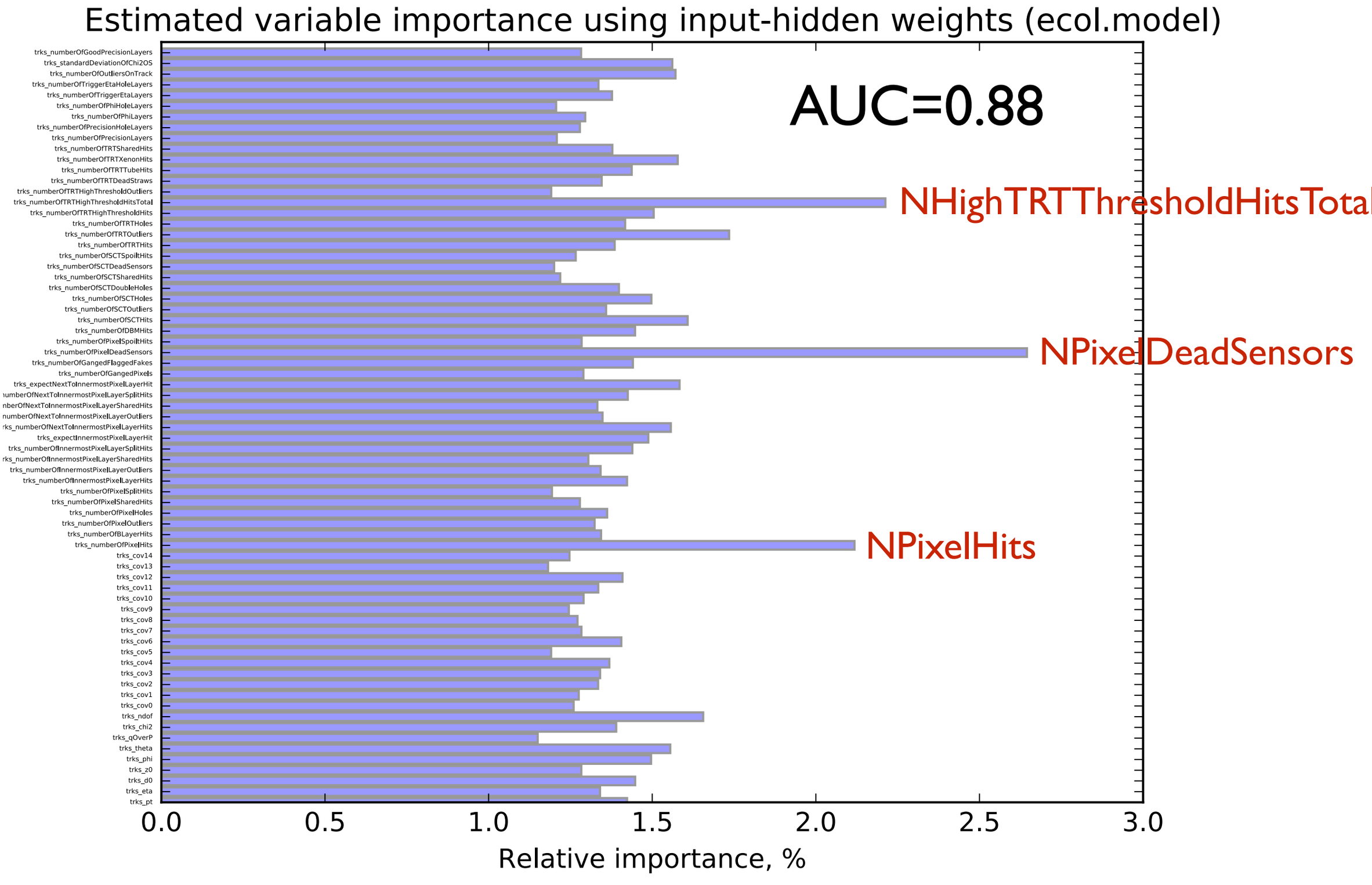
Variable importance: zero field (BDT)



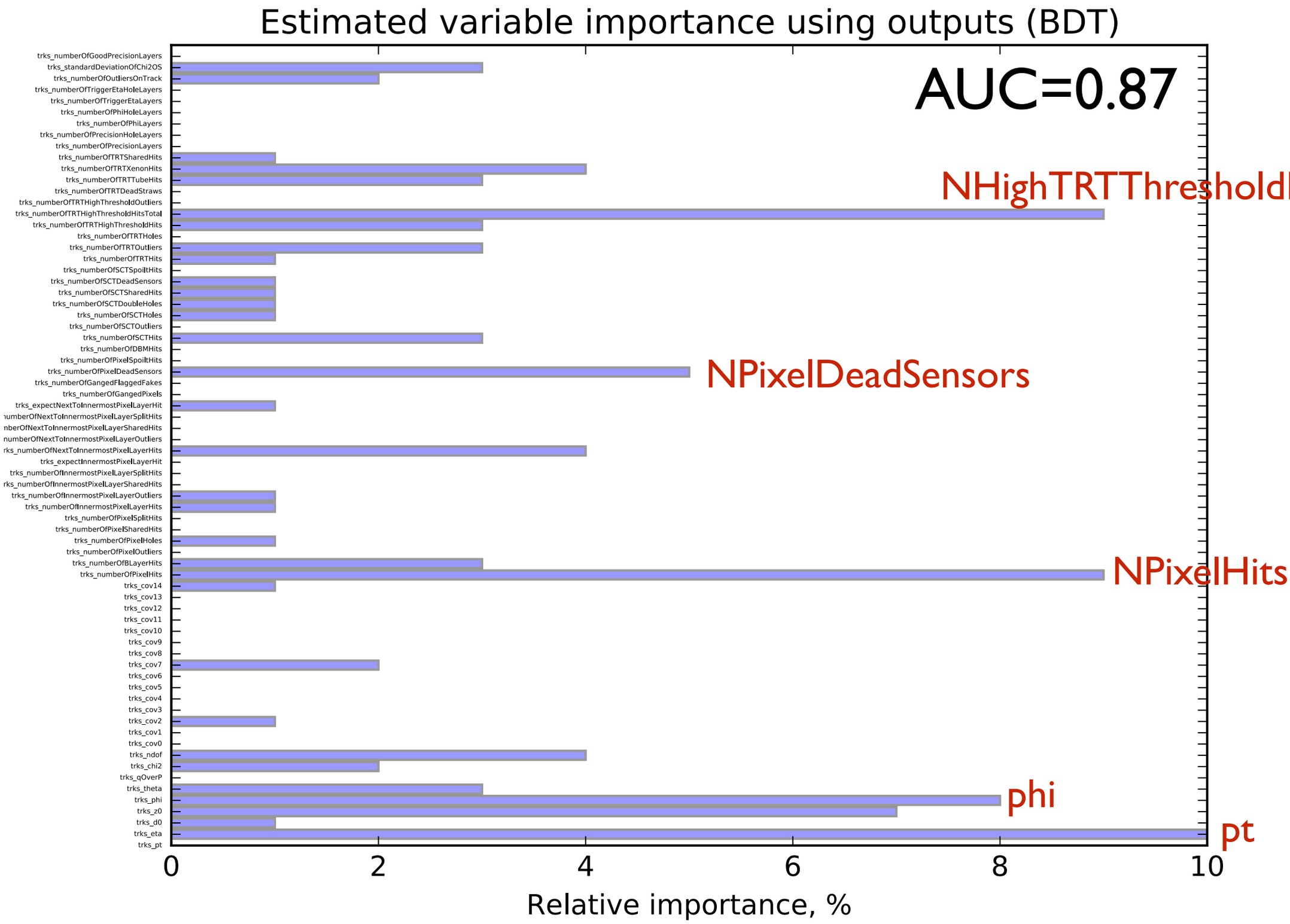
Variable importance: zero field (autoencoder)



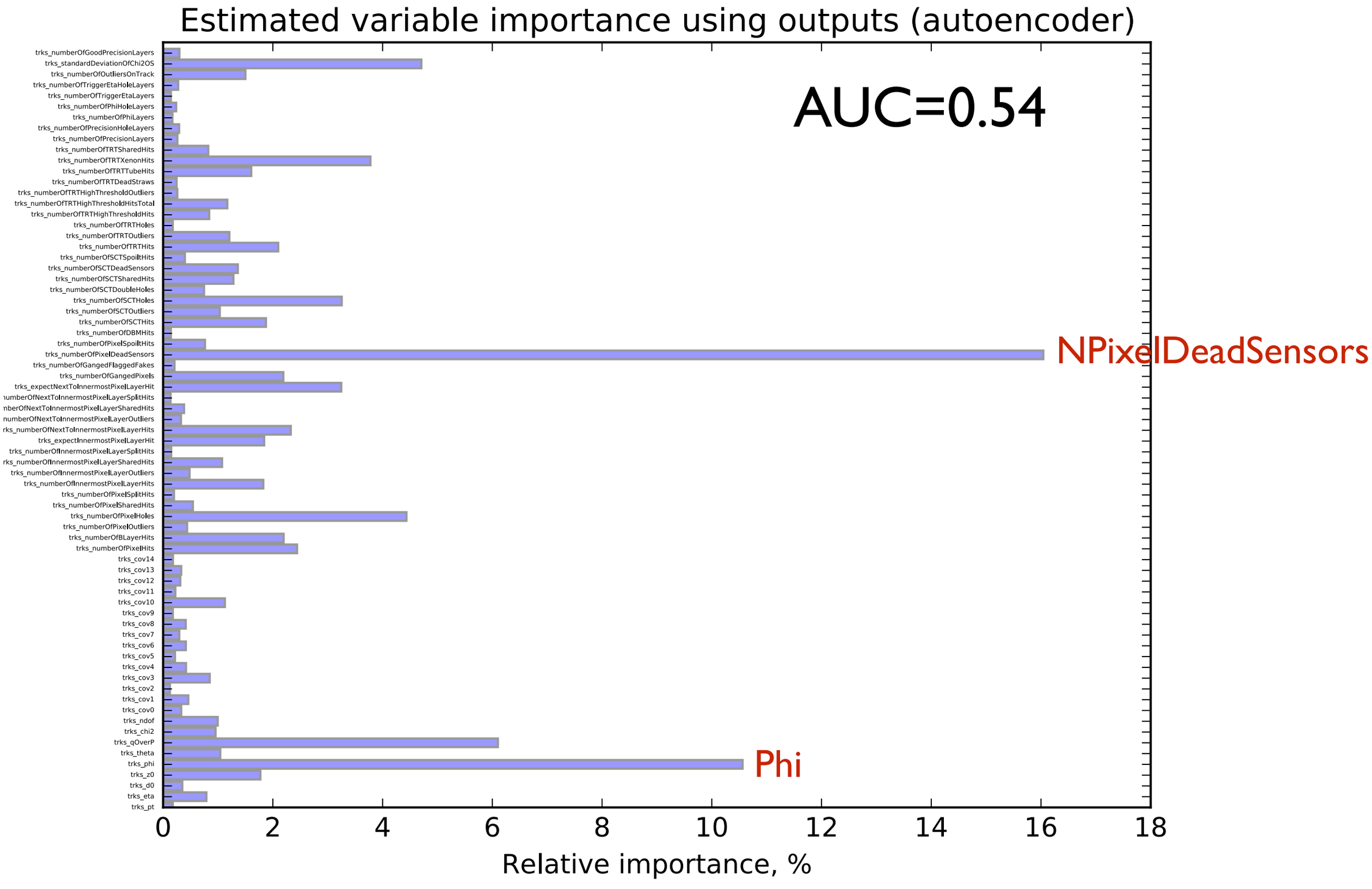
Variable importance: "Queen Bork" (NN classifier)



Variable importance: "Queen Bork" (BDT)



Variable importance: "Queen Bork" (autoencoder)



Next steps & points to think about

- Main question: is this a useful direction to go in?
  - Does it "add value" beyond simply looking at histograms
- If it is thought to be potentially useful
  - More detailed studies of the defects
  - Study other domains
  - Combined measure from all domains?
  - Consider how the mechanism could be used in real life
    - Lumi-block level?
    - Dataset level?
    - Feedback from the defects database?
  - Investigate whether the performance of the auto-encoder could be improved

Conclusions

- Using tracking variables, a prototype anomaly detection scheme based on split-sample classifiers is able to distinguish successfully luminosity blocks from "bad" runs
  - Similar scheme based on an auto encoder is not successful
- Variable importance study seems to indicate that the classifiers are not always using the same variables to distinguish the samples

Backup slides

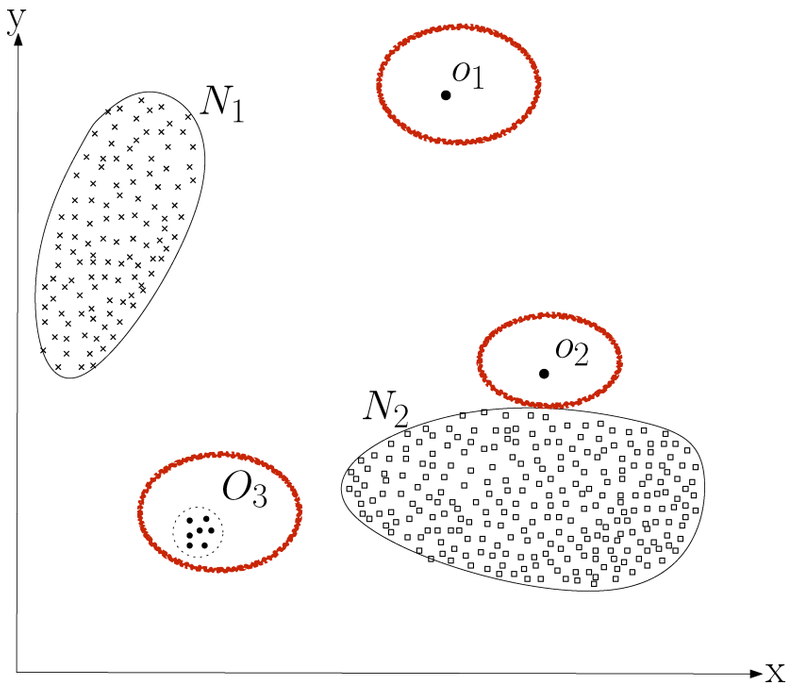
What is anomaly detection?

I think

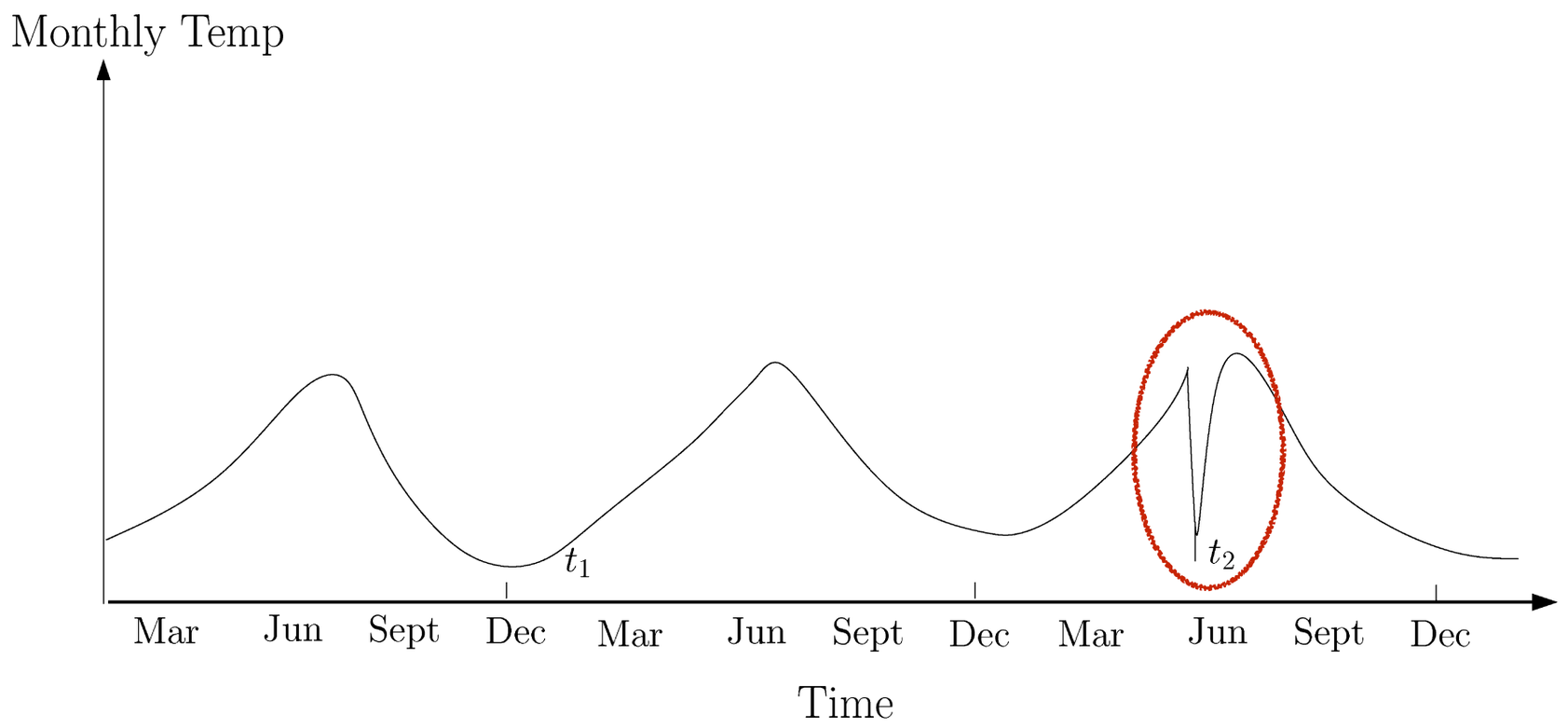
- Automatic identification of data instances (events) that are in some way different from the bulk of the data and which need detailed scrutiny by experts. Usually implied:
  - produced by a different mechanism than the bulk of the events
  - small number of anomalies w.r.t. the main part of the data
- Can be
  - supervised: train to recognise specific anomalous cases
  - semi-supervised: train only on the bulk of the data without anomalies → strong relation to one-class classification
  - unsupervised: algorithm automatically identifies the bulk by some means and thence the anomalies
- Difficult problem because in general we don't know what the anomalies look like, and there may be very few of them
  - Testing is particularly challenging: how do we evaluate the performance of an algorithm on a type of event that we have never seen before?

Types of anomaly

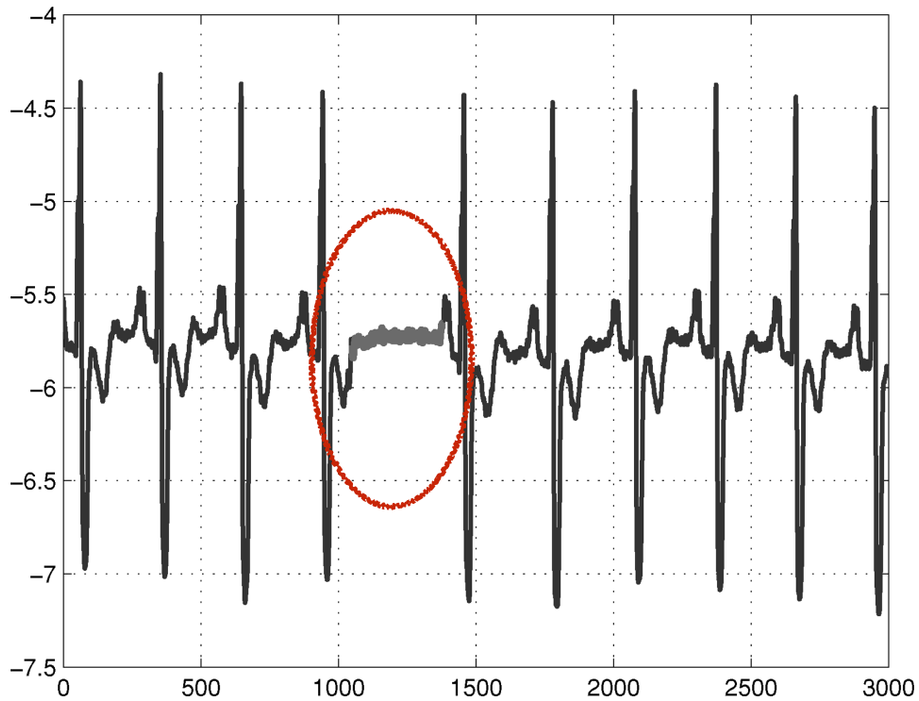
POINT



CONTEXTUAL

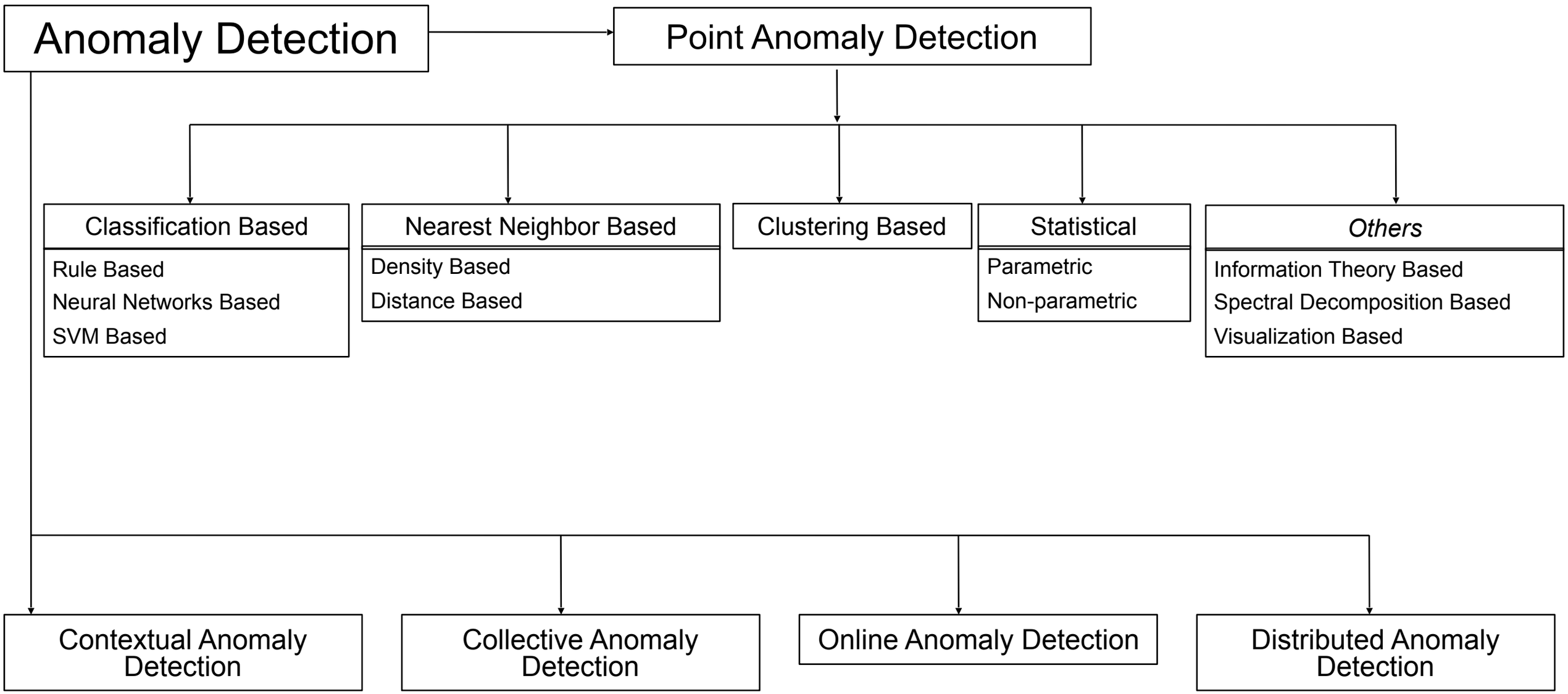


COLLECTIVE (population-level)



Images:
ACM Computing Surveys,
Vol 41, No. 3, Article 15

Taxonomy*



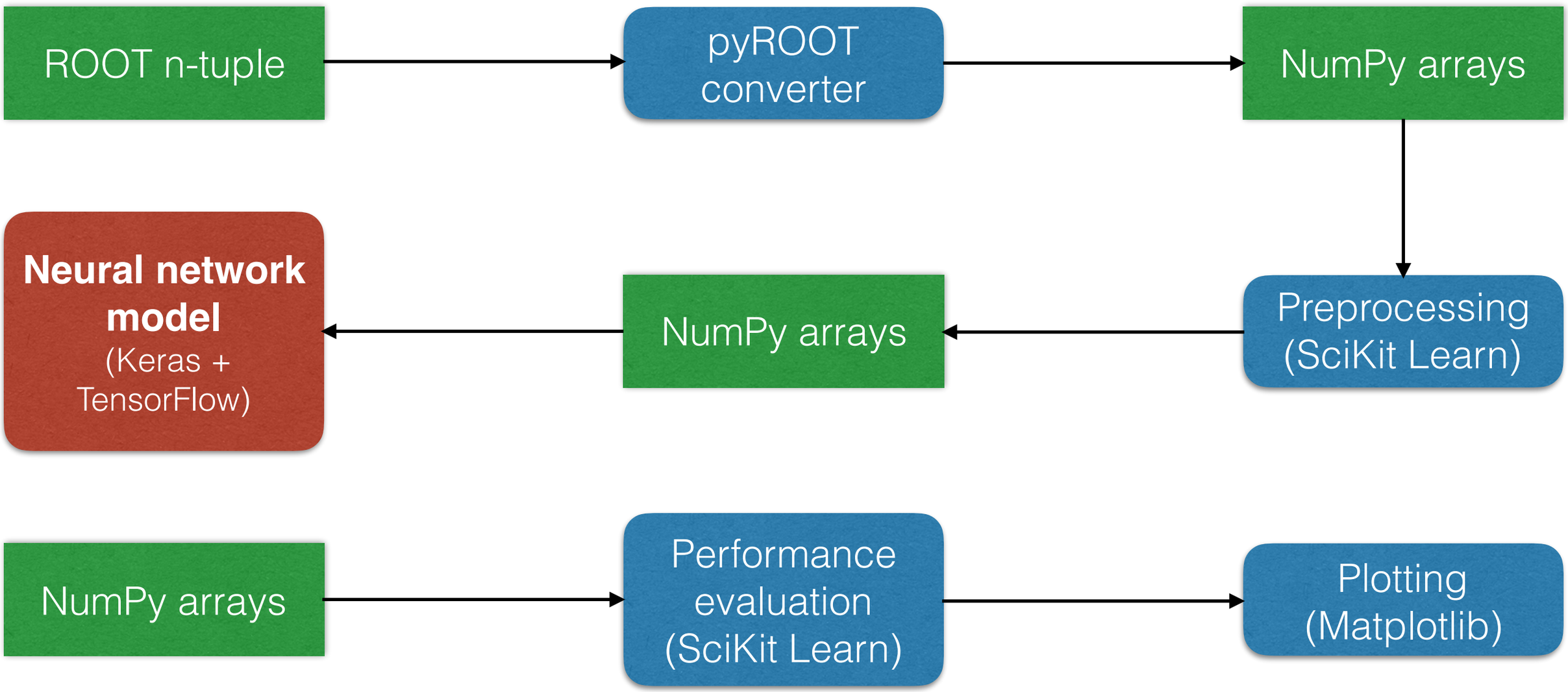
* Outlier Detection – A Survey, Varun Chandola, Arindam Banerjee, and Vipin Kumar, Technical Report TR07-17, University of Minnesota (Under Review)

Variable list

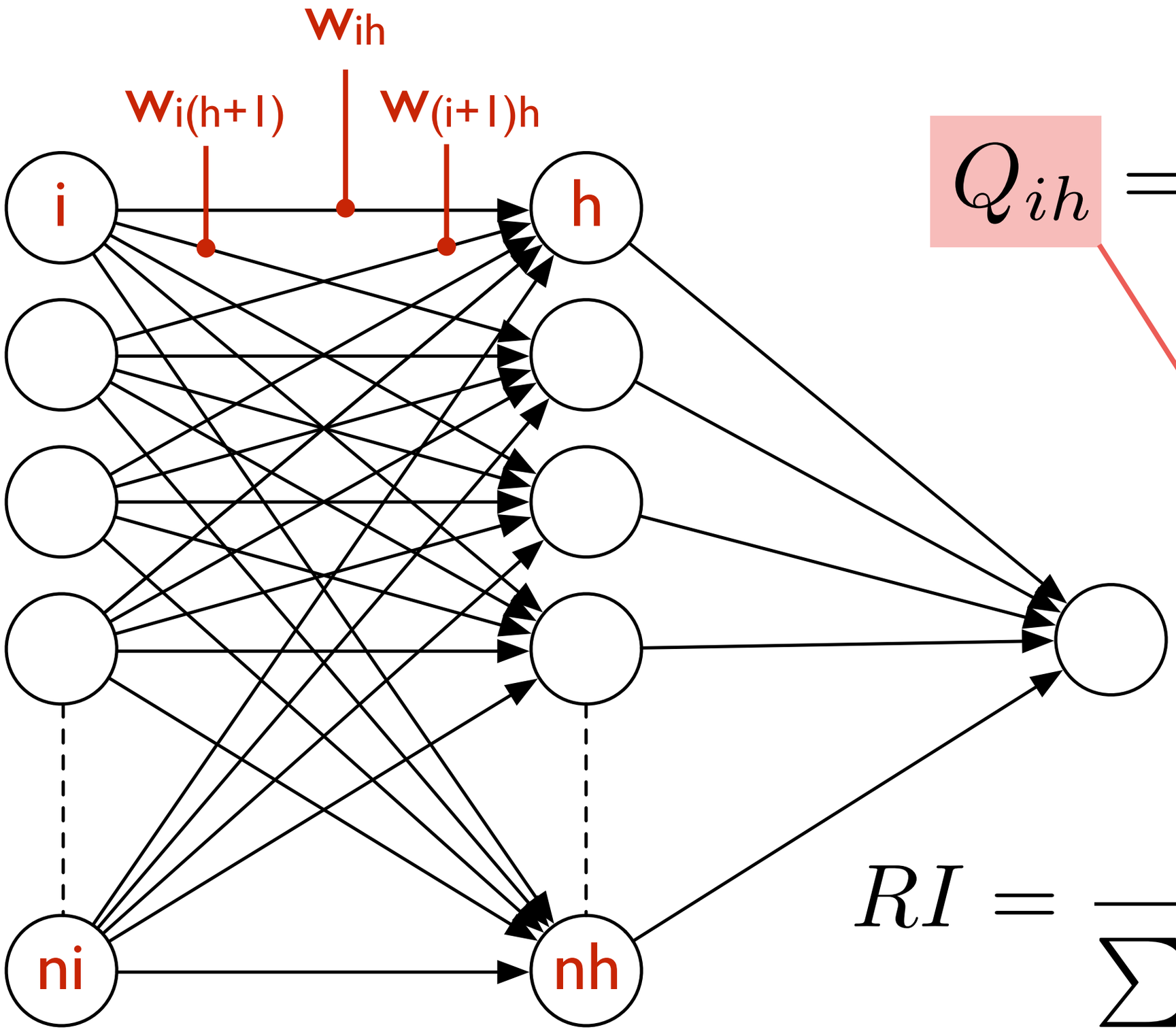

```
"trks_pt", "trks_eta","trks_d0", "trks_z0", "trks_phi", "trks_theta", "trks_qOverP",
"trks_chi2", "trks_ndof", "trks_cov0", "trks_cov1", "trks_cov2", "trks_cov3", "trks_cov4",
"trks_cov5", "trks_cov6", "trks_cov7", "trks_cov8", "trks_cov9", "trks_cov10",
"trks_cov11", "trks_cov12", "trks_cov13", "trks_cov14", "trks_numberOfPixelHits",
"trks_numberOfBLayerHits", "trks_numberOfPixelOutliers", "trks_numberOfPixelHoles",
"trks_numberOfPixelSharedHits", "trks_numberOfPixelSplitHits",
"trks_numberOfInnermostPixelLayerHits", "trks_numberOfInnermostPixelLayerOutliers",
"trks_numberOfInnermostPixelLayerSharedHits", "trks_numberOfInnermostPixelLayerSplitHits",
"trks_expectInnermostPixelLayerHit", "trks_numberOfNextToInnermostPixelLayerHits",
"trks_numberOfNextToInnermostPixelLayerOutliers",
"trks_numberOfNextToInnermostPixelLayerSharedHits",
"trks_numberOfNextToInnermostPixelLayerSplitHits",
"trks_expectNextToInnermostPixelLayerHit", "trks_numberOfGangedPixels",
"trks_numberOfGangedFlaggedFakes", "trks_numberOfPixelDeadSensors",
"trks_numberOfPixelSpoiltHits", "trks_numberOfDBMHits", "trks_numberOfSCTHits",
"trks_numberOfSCTOutliers", "trks_numberOfSCTHoles", "trks_numberOfSCTDoubleHoles",
"trks_numberOfSCTSharedHits", "trks_numberOfSCTDeadSensors", "trks_numberOfSCTSpoiltHits",
"trks_numberOfTRTHits", "trks_numberOfTRTOutliers", "trks_numberOfTRTHoles",
"trks_numberOfTRTHighThresholdHits", "trks_numberOfTRTHighThresholdHitsTotal",
"trks_numberOfTRTHighThresholdOutliers", "trks_numberOfTRTDeadStraws",
"trks_numberOfTRTTubeHits", "trks_numberOfTRTXenonHits", "trks_numberOfTRTSharedHits",
"trks_numberOfPrecisionLayers", "trks_numberOfPrecisionHoleLayers",
"trks_numberOfPhiLayers", "trks_numberOfPhiHoleLayers", "trks_numberOfTriggerEtaLayers",
"trks_numberOfTriggerEtaHoleLayers", "trks_numberOfOutliersOnTrack",
"trks_standardDeviationOfChi2OS", "trks_numberOfGoodPrecisionLayers"
```

Software workflow

https://gitlab.cern.ch/jcatmore/dqews



"Weights method" for determining variable importance in a shallow NN



$$Q_{ih} = \frac{|W_{ih}|}{\sum_{i=1}^{ni} |W_{ih}|}$$

$$RI = \frac{\sum_{h=1}^{nh} Q_{ih}}{\sum_{h=1}^{nh} \sum_{i=1}^{ni} Q_{ih}}$$

Variable importance: auto-encoder

- The auto-encoder is trained to reproduce its input, so it has as many variables in the output as in the input
- The overall output is the reconstruction error, being the sum of the squared differences between input and output variables
- So we can *directly* study the relative importance of the variables in discrimination, since the bigger the difference between input and output, the larger the reconstruction error and so the more important that variable is

Mean relative reconstruction error for jth variable

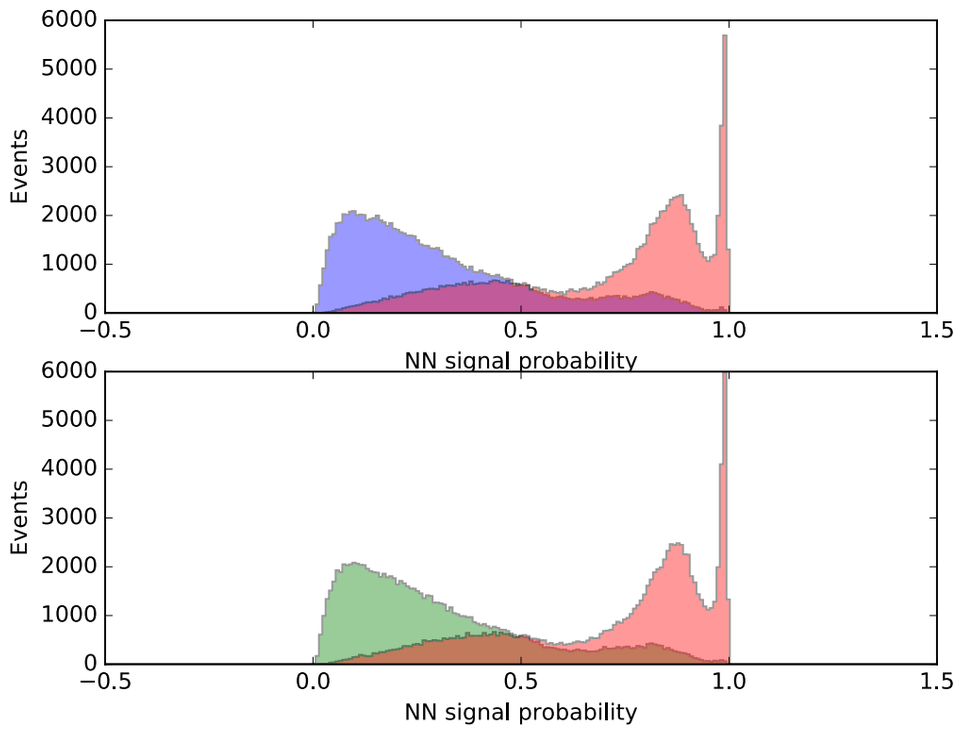
$$= \frac{1}{N} \cdot \sum_{i=1}^{N} \frac{(x_j^{in} - x_j^{out})^2}{\sum_{j=1}^{M} (x_j^{in} - x_j^{out})^2}$$

(N events, M variables)

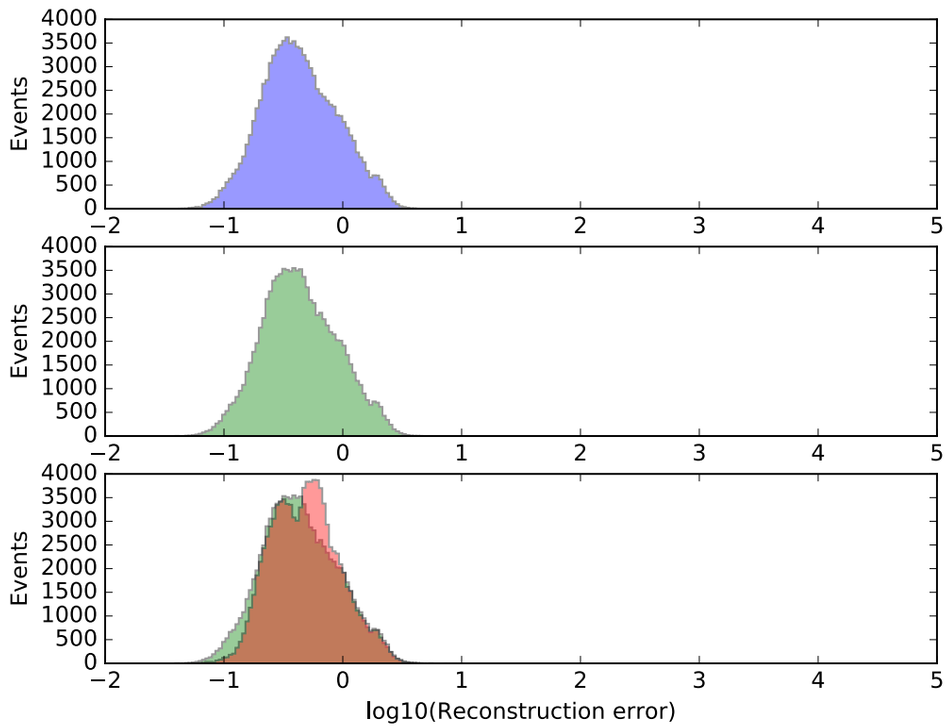
Total reconstruction error per event

Selected outputs: "Queen Bork"

NN classifier



Autoencoder



BDT classifier