Monday, August 14, 2017

Report On Predictive Performance

Summary

I've recently finished running some performance tests that show interesting results.

I. Delaunay appears to be our best predictor of mean system throughput (by a factor of 2)

II. Moving forward, we may want to consider comparing the performance distributions of algorithms rather than just analyzing mean absolute error and/or mean squared error.

Data

Using original (2016) VarSys data

CFQ hypervisor scheduler
NOOP vm scheduler
file read test

All ordinal settings for:

Threads
File Size
Record Size
Frequency

Rather than selecting training / testing sets entirely at random, there are two extra criteria enforced in this comparison that fit comfortably with our real-world application scenario.

Testing points are strictly inside the convex hull of training points

Training points are selected (randomly) to be well-spaced

Comparison

We look at prediction error in mean throughput of the system using the following algorithms:

BayesTree
Delaunay
FitBoxMesh
LSHEP
MARS

There are two sets of histograms associated with this report. They are included in the report PDF, but I am also attaching the two plots in HTML format as well. I recommend viewing the HTML plots in a web browser, because they are interactive and allow you to do pairwise comparisons between algorithms more easily by (de)selecting legend entries.

"...-Dimension.html" compares the prediction performance of each algorithm in increasing dimension.

"...-Training-Percentage.html" compares the prediction performance of each algorithm when given larger amounts of training data.

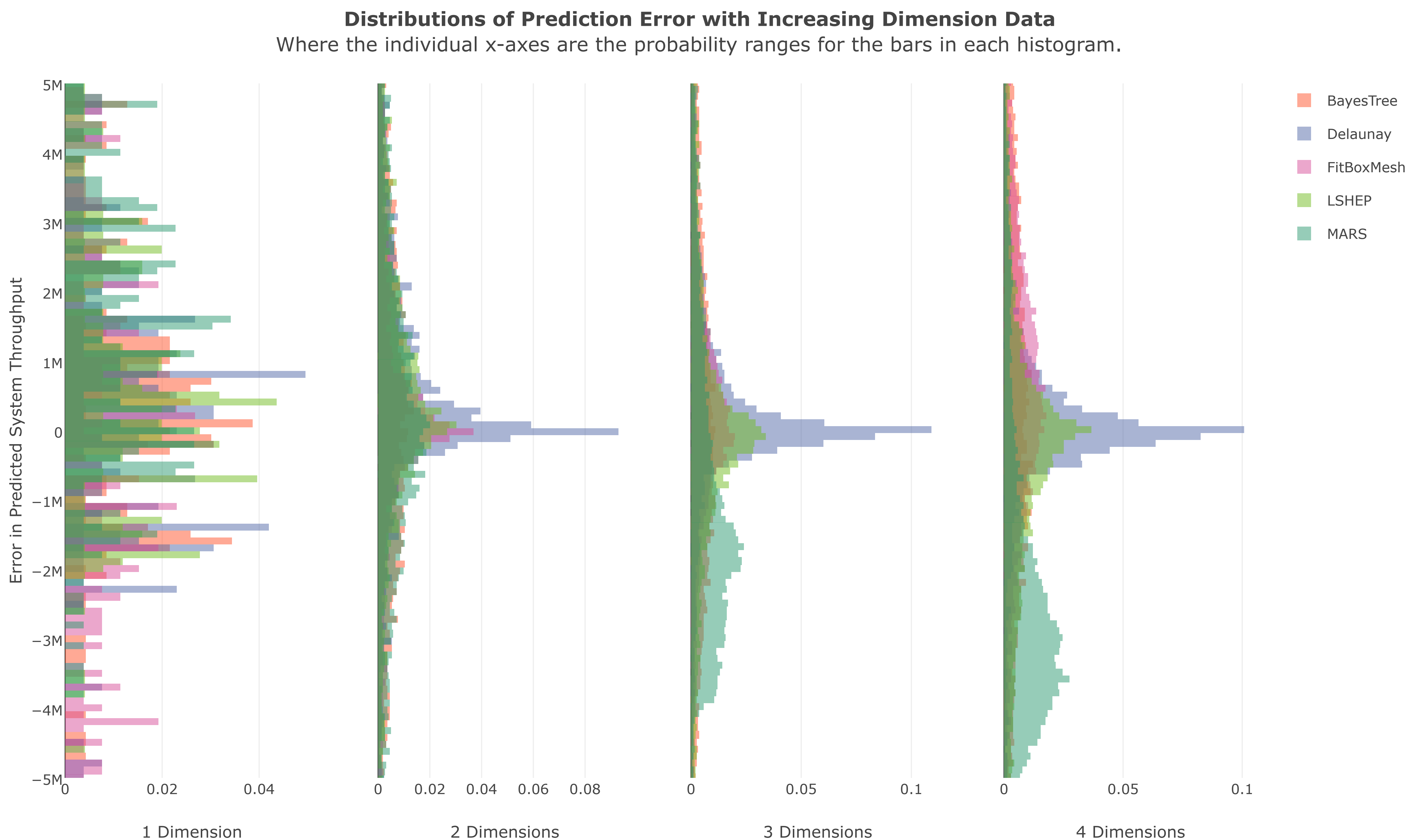
Results Summary

The Delaunay method appears to be our best predictor of system throughput in almost all scenarios. This can be seen in the following visualizations by noticing that the error for Delaunay is most tightly distributed around 0.

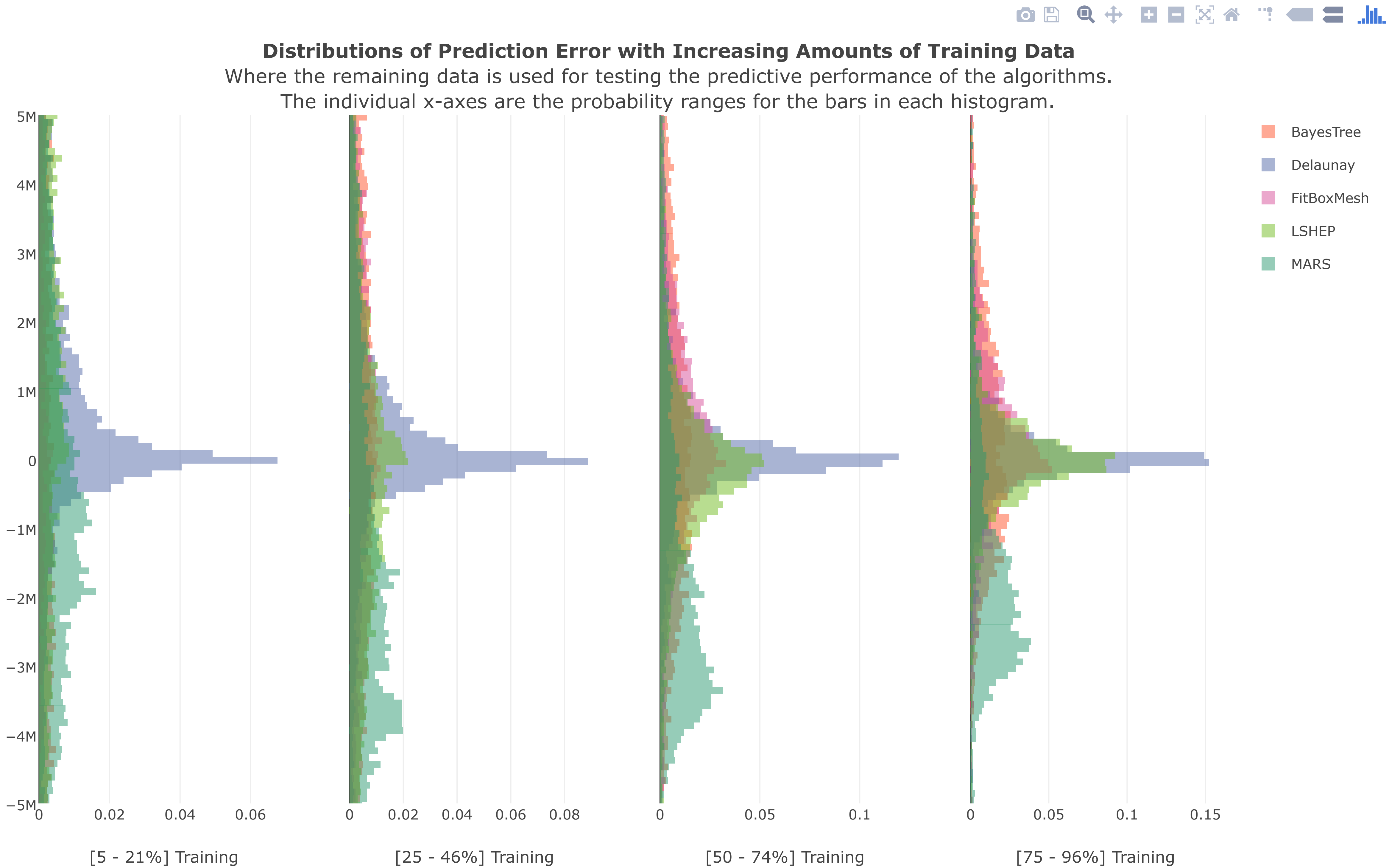
Also, notice that the distribution of errors for MARS is skewed towards under-estimating the throughput of a system. This sort of skew may be something we want to take into account when comparing algorithms in the future.

Overall Algorithm Performance

ALGORITHM	MEAN ABSOLUTE ERROR
Delaunay	2914457
MARS	5469586
LSHEP	6622584
BayesTree	7493919
FitBoxMesh	29249964



Distributions of Prediction Error with Increasing Dimension Data
Where the individual x-axes are the probability ranges for the bars in each histogram.
Error in Predicted System Throughput
5M
4M
3M
2M
1M
0
−1M
−2M
−3M
−4M
−5M
0
0.02
0.04
0.06
0.08
0.05
0.1
1 Dimension
2 Dimensions
3 Dimensions
4 Dimensions
BayesTree
Delaunay
FitBoxMesh
LSHEP
MARS



Distributions of Prediction Error with Increasing Amounts of Training Data
Where the remaining data is used for testing the predictive performance of the algorithms.
The individual x-axes are the probability ranges for the bars in each histogram.
BayesTree
Delaunay
FitBoxMesh
LSHEP
MARS
5M
4M
3M
2M
1M
0
−1M
−2M
−3M
−4M
−5M
0
0.02
0.04
0.06
0.08
0.05
0.1
0.15
[5 - 21%] Training
[25 - 46%] Training
[50 - 74%] Training
[75 - 96%] Training

Raw Results in Text Form

ALGORITHM	MEAN ABSOLUTE ERROR
1 Dimension	
Delaunay	3706736
FitBoxMesh	5254595
LSHEP	7550209
MARS	13162639
BayesTree	13873732
2 Dimensions	
Delaunay	3035326
MARS	5667880
BayesTree	9947540
LSHEP	13859288
FitBoxMesh	26508653
3 Dimensions	
Delaunay	3272200
MARS	4009267
LSHEP	5507243
BayesTree	7330992
FitBoxMesh	52092634
4 Dimensions	
Delaunay	2737639
LSHEP	4352081
MARS	4402294
BayesTree	4933948
FitBoxMesh	24850360
[5 - 21%] Training Data	
Delaunay	5309946

ALGORITHM	MEAN ABSOLUTE ERROR
MARS	6529343
LSHEP	13202393
BayesTree	13841196
FitBoxMesh	100007021
[25 - 46%] Training Data	
Delaunay	2898461
MARS	4324916
LSHEP	4936381
BayesTree	5508646
FitBoxMesh	20675219
[50 - 74%] Training Data	
Delaunay	1540992
LSHEP	2281782
BayesTree	2849707
FitBoxMesh	2874012
MARS	3156969
[75 - 96%] Training Data	
Delaunay	814356
LSHEP	1291293
BayesTree	1665588
FitBoxMesh	1737220
MARS	2503381

Moving Forward

I will be exploring the underlying reasons for the unexpectedly bad performance of "FitBoxMesh". This is the new algorithm I have been working on this summer and I had expected it to perform comparably to the Delaunay method.