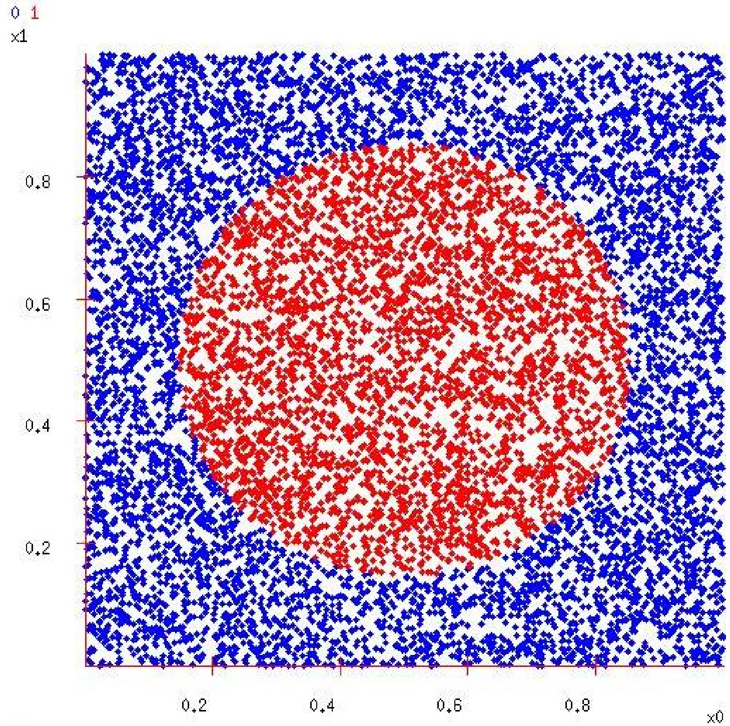


Autonomous Visualization

Khalid El-Arini
Andrew W. Moore

December 15, 2005

Motivation

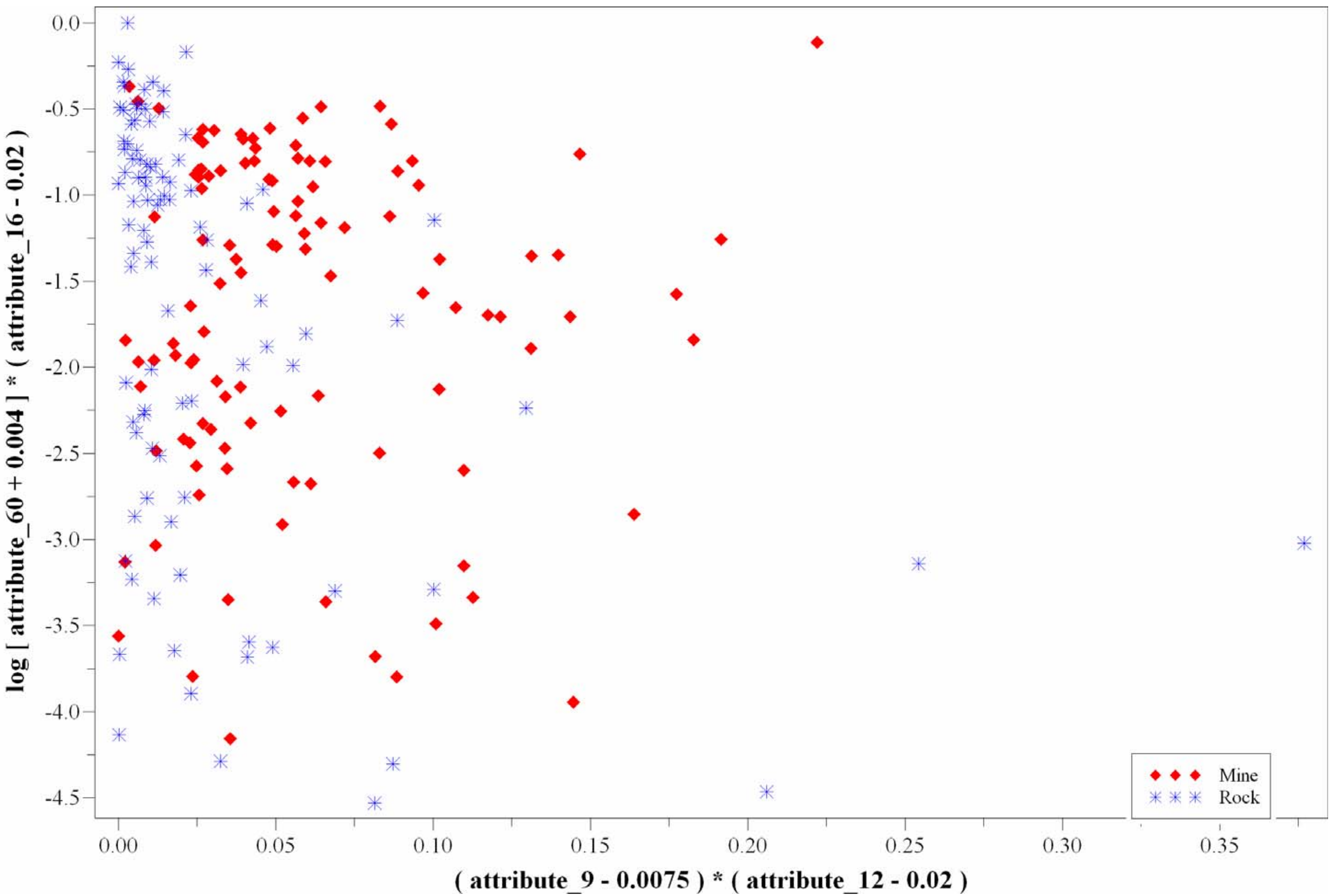
- Many classification algorithms suffer from a lack of human interpretability
- Output is often just a symbolic answer or a probability
  - “Galaxy” or “Star”
  - “77% chance of cancer”
- Without an understanding of ***why*** these decisions are made, we are often forced to accept them with blind faith in the model

Overview

- We introduce a classification algorithm that takes into account human interpretability of the results: *Autonomous Visualization (AV)*
- We present results on several real data sets
  - Classification accuracy
  - Visualization
  - Time complexity
- Concluding remarks

Interpretability

- Assume data has real-valued input attributes and a single, symbolic output
- We attempt to capture the “most relevant” snapshot of the data in the form of a two dimensional scatter plot
  - Simple arithmetic expressions
  - Up to two input attributes per axis
  - We call the pair of expressions that define the axes of a scatter plot a *pairexp*



log [attribute_60 + 0.004] * (attribute_16 - 0.02)
(attribute_9 - 0.0075) * (attribute_12 - 0.02)
Mine
Rock
0.0
-0.5
-1.0
-1.5
-2.0
-2.5
-3.0
-3.5
-4.0
-4.5
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35

pairexps

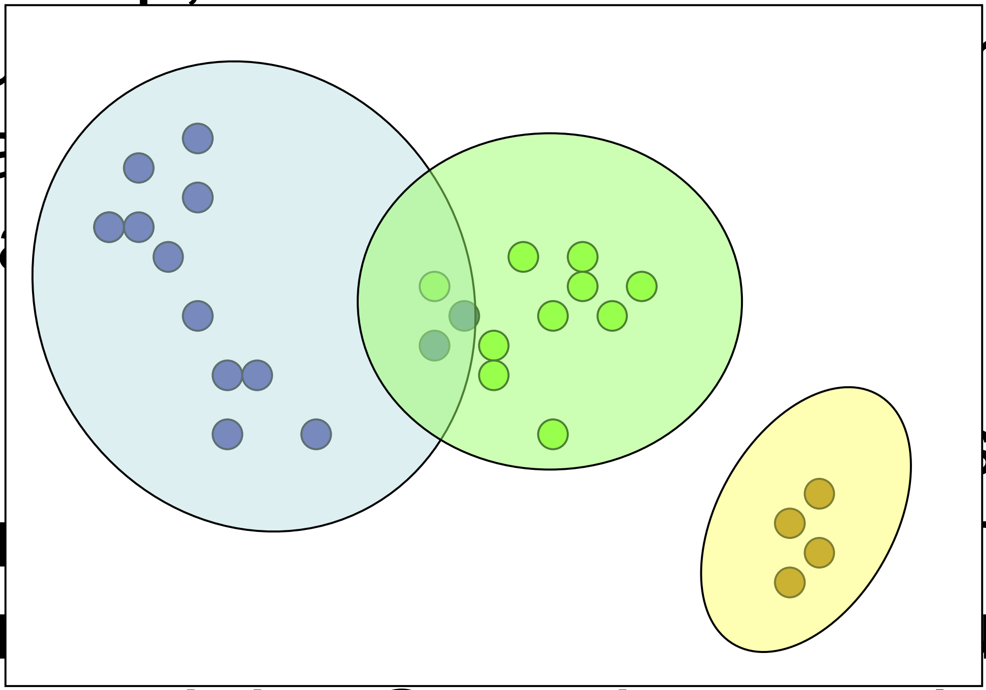
- In order for the simple expressions on each axis to be most informative, we scale and translate their operands as necessary
- For instance, before adding two attribute values, we scale both such that each is in the range [0,1]
  - Adding 0.04 to 1.3×10^8 isn't worthwhile

Relevance

- The most important aspect of AV is determining which scatter plots are better than others
- For each pairexp, we compute a Gaussian misclassification score
- We define the “most relevant” scatter plot to be the one with the **lowest** scoring pairexp

Gaussian Misclassification

- Given a pairexp, we consider the data points in ou rmed space to ha bivariate Ga per class
- We learn a ssian for each ou he number of uld occur if we used the Gaussians to classify the points in our data set



Gaussian Misclassification

$$\text{score} = \sum_i I(c_i \neq \widehat{c}_i)$$

Where c_i is the correct output class for point $\mathbf{x}_i$, and

$$\begin{aligned}
\widehat{c}_i &= \text{argmax}_k P(c_i = k | \mathbf{x}_i) \\
&= \text{argmax}_k \frac{P(\mathbf{x}_i | c_i = k) P(c_i = k)}{\sum_l P(\mathbf{x}_i | c_i = l) P(c_i = l)} \\
&= \text{argmax}_k P(\mathbf{x}_i | c_i = k) P(c_i = k)
\end{aligned}$$

Gaussian density function for class *k*

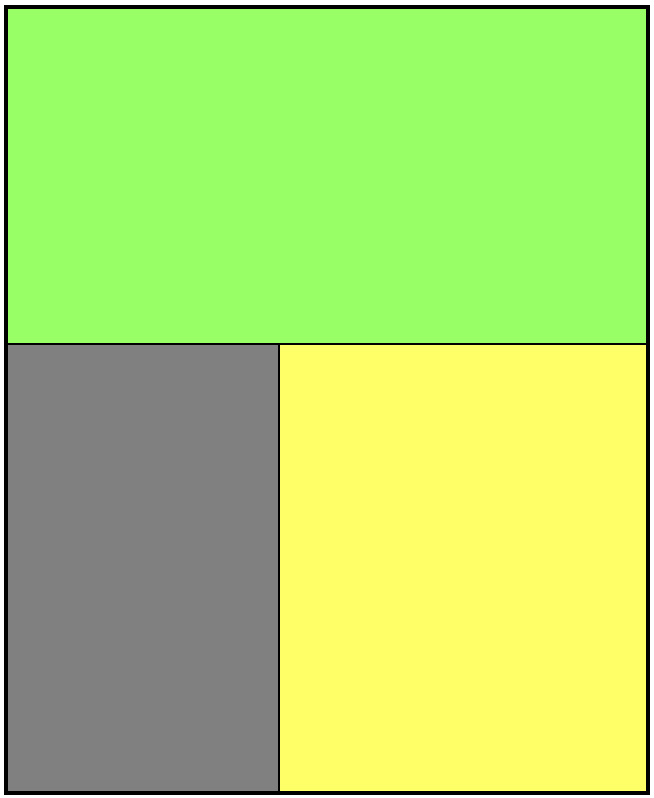
Prior probability of class *k*

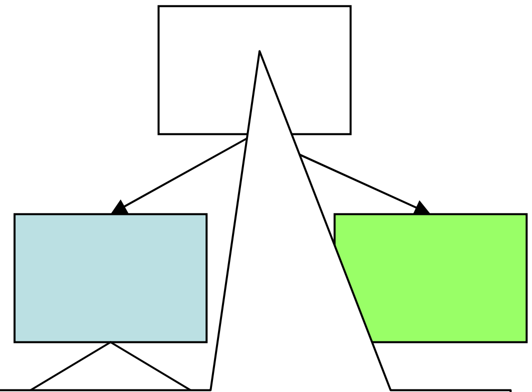
Naïve Score Computation

$$\text{score} = \sum_i I(c_i \neq \hat{c}_i)$$

- Could iterate through each point in the data set, compute its Gaussian classification, and compare to its true class label
- Clearly, this takes time linear in *n*, the size of our data set
- Instead, we can take advantage of the spatial structure of our data

kd-trees





- *K* dimensional bounding hyperrectangle
- Each node “owns” a set of data points
- Store the number of data points per class
- Store the actual data points (only at leaves)

kd-tree Score Computation

$$\text{score} = \sum_i I(c_i \neq \widehat{c}_i)$$

- A pairexp consists of at most 4 attributes, e.g. *x*,*y*,*z*,*w*
- We build a four dimensional *kd*-tree of the data over these 4 attributes
  - We use raw attributes
  - This allows us to reuse the same *kd*-tree for all possible pairexps of those 4 attributes (≈ 3,000)

kd-tree Score Computation

$$\text{score} = \sum_i I(c_i \neq \widehat{c}_i)$$

- Do breadth first search of *kd*-tree, starting at root
- At each node, try to determine the number of misclassifications among that node’s points without actually iterating through each point
  - Does any particular class **dominate** this node?

Dominance

- Recall: each *kd*-tree node defines a bounding hyperrectangle over a subset of points
- For every geometric location **x** in the bounding box, does there exist a class *k* such that $P(c_{\mathbf{x}} = k|\mathbf{x}) > P(c_{\mathbf{x}} = l|\mathbf{x})$ for all othe

We can do this efficiently—much faster than iterating through all data points

- If so, then class *k* dominates this node, and we can prune our search, since we can update the misclassification score directly

pairexp Search Algorithm

- Recap: we now know how to efficiently compute the score for a given pairexp
- Now, we need to search over pairexps in order to find the best one.
- Exhaustively searching over every possible pairexp is prohibitively slow
- Instead, we perform a hierarchical search with a greedy outer loop and an exhaustive inner loop

pairexp Search Algorithm

- Consider each pair of attributes, and find the best pairexp over two attributes
- Consider each triple of attributes that contain the best pair, and find the best pairexp over three attributes
- Consider each quadruple containing the best triple
- Consider all quadruples “one away” from best quadruple

pairexp Search Algorithm

- We build one *kd*-tree for each outer loop iteration, and reuse throughout the inner loop
- If, e.g., the best triple does not do as well as the best pair, we terminate early
- As we compute our Gaussian misclassification score for a given pairexp, we already know the best score seen so far. Thus, can prune search immediately when score gets worse than best.

Early termination pruning–our real source of computational speedup

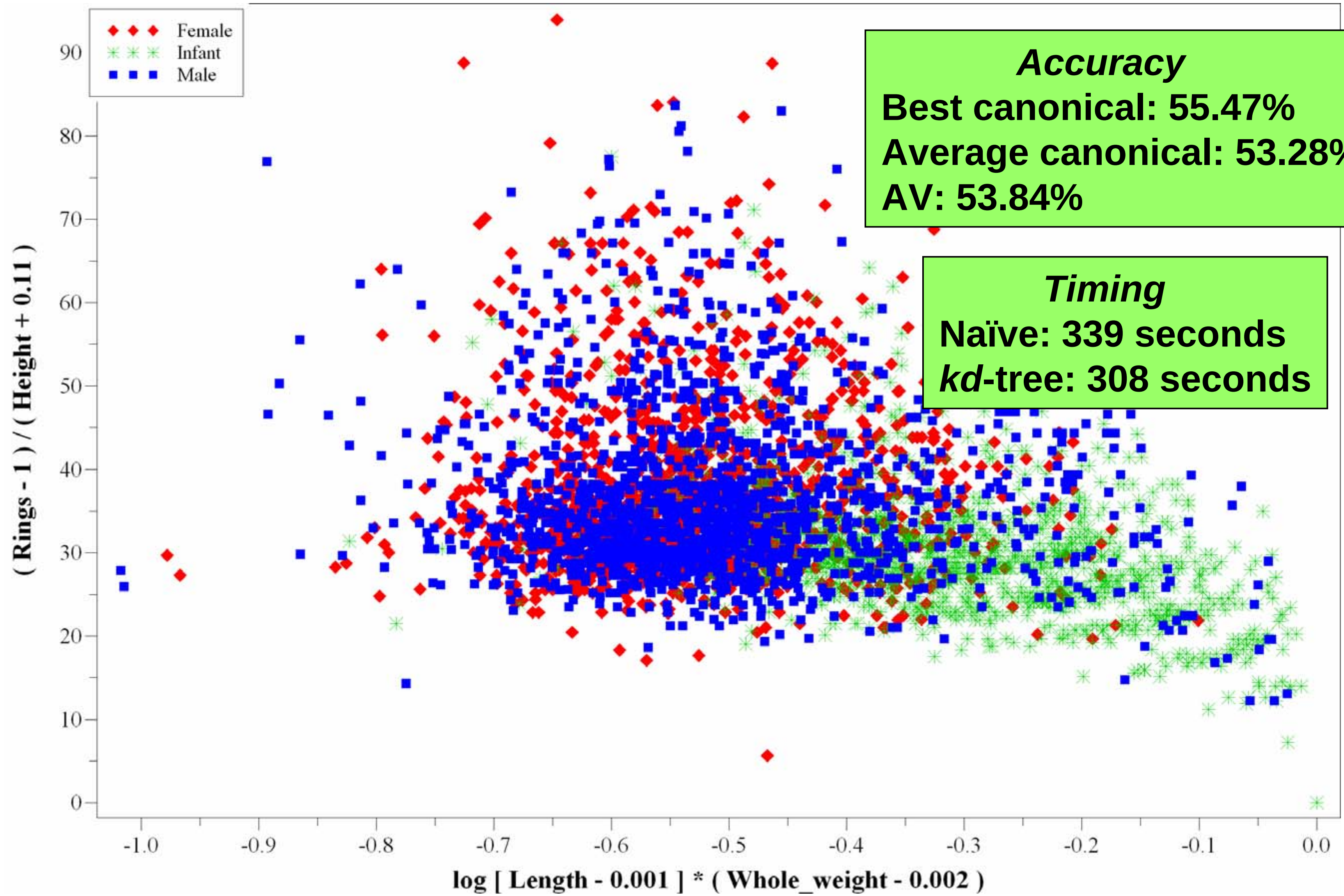
Classification

- Once we find the best pairexp, we use a Gaussian Bayes classifier in the two dimensional transformed space to classify future data points
- We performed 5-fold cross validation to obtain classification accuracy scores, and compared them to nine canonical classifiers (courtesy of Weka)
  - e.g. k-NN, SVMs, logistic regression, …

Results

Abalone

4178 records
8 real attributes
3 output values

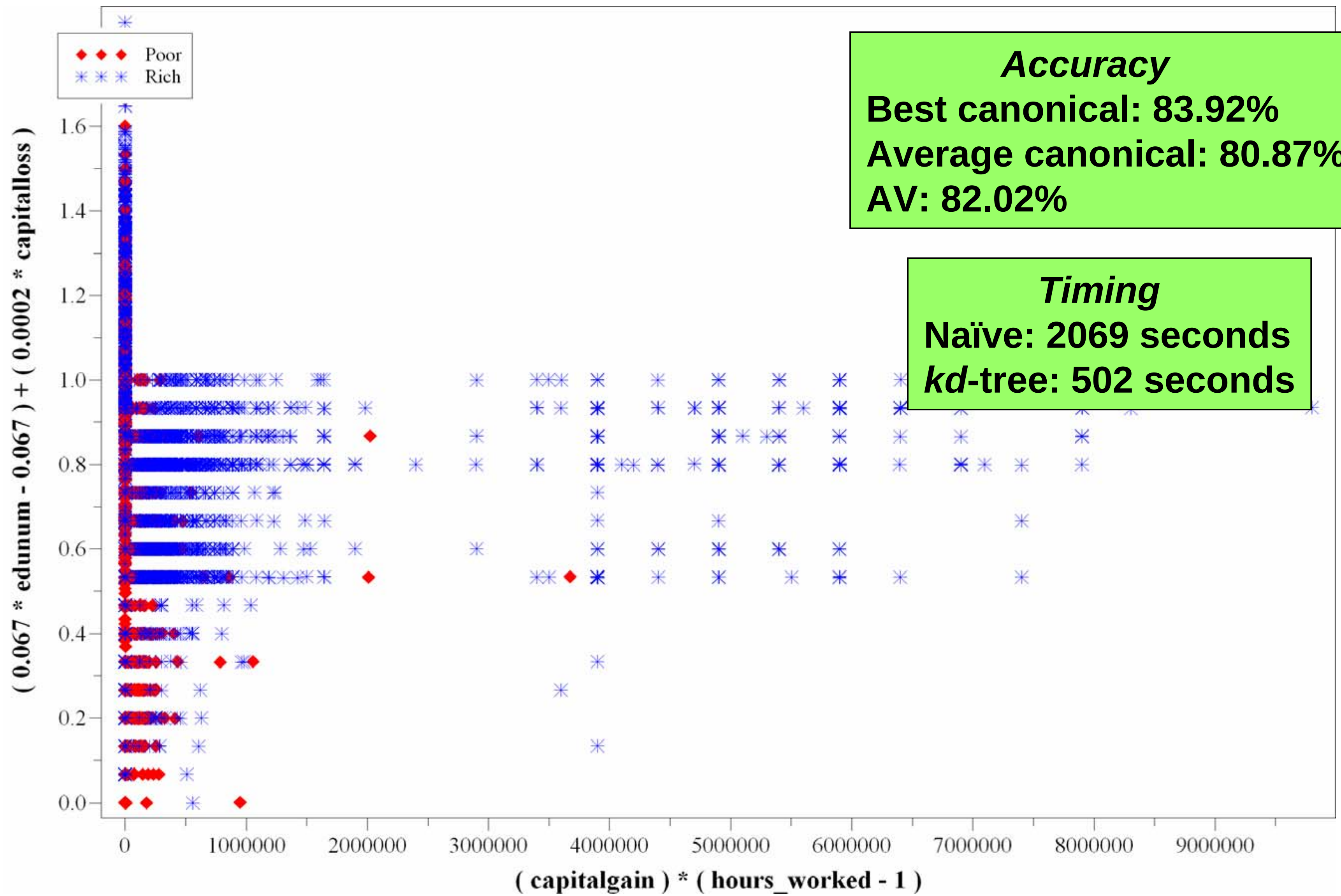


Accuracy
Best canonical: 55.47%
Average canonical: 53.28%
AV: 53.84%

Timing
Naïve: 339 seconds
***kd*-tree: 308 seconds**

Adult

48844 records
6 real attributes
2 output values

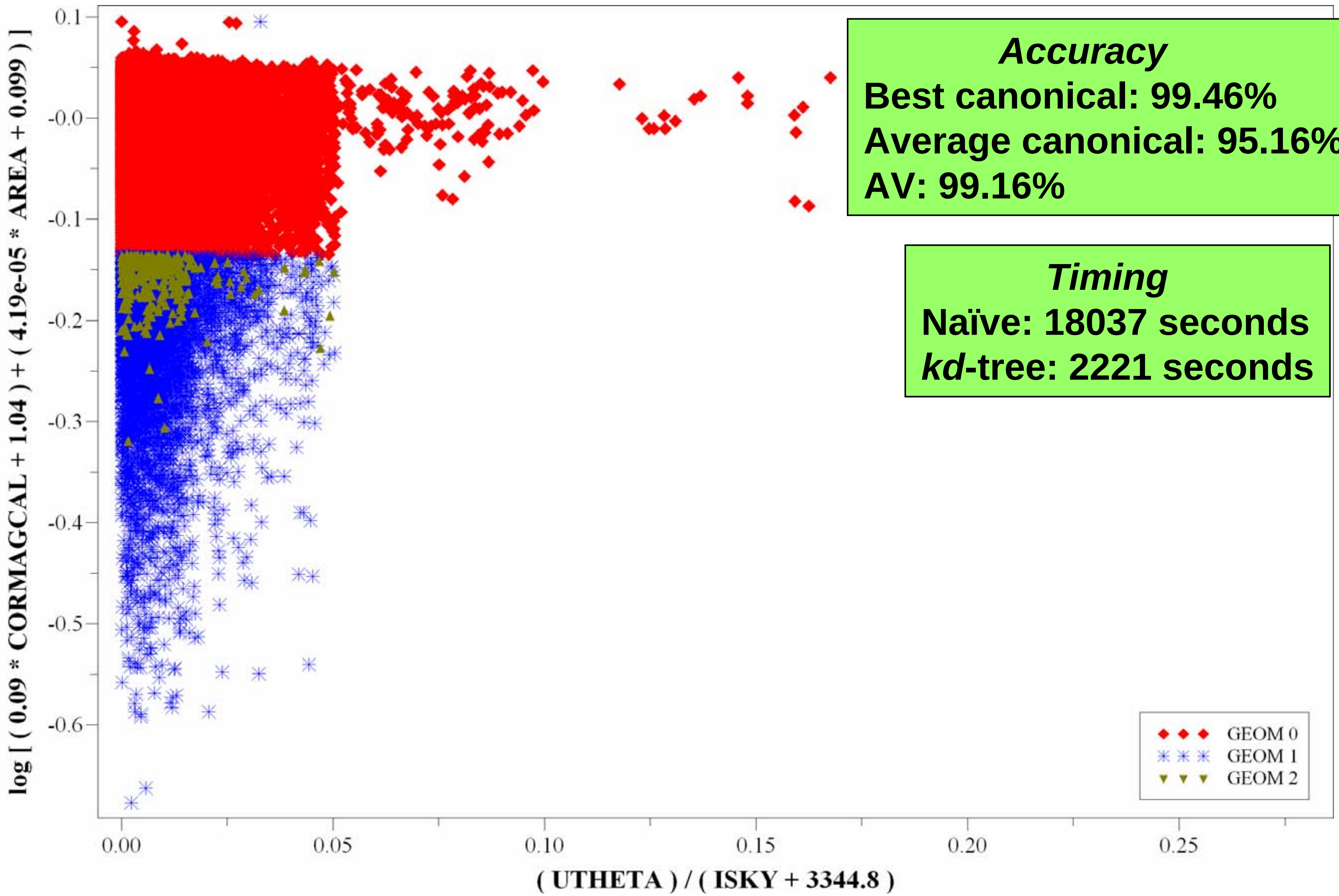


Accuracy
Best canonical: 83.92%
Average canonical: 80.87%
AV: 82.02%

Timing
Naïve: 2069 seconds
***kd*-tree: 502 seconds**

Astro

50000 records
22 real attributes
3 output values

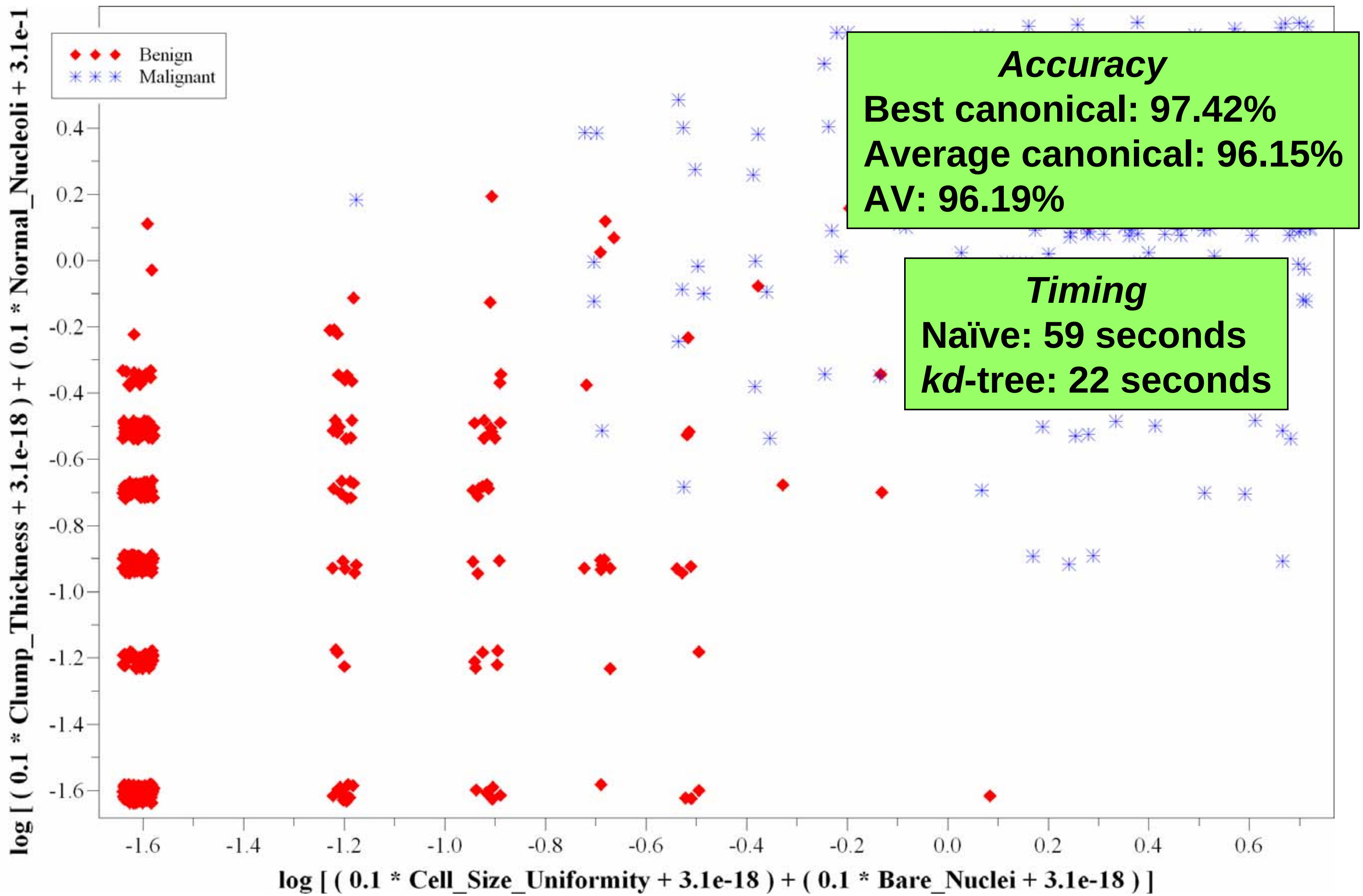


Accuracy
Best canonical: 99.46%
Average canonical: 95.16%
AV: 99.16%

Timing
Naïve: 18037 seconds
***kd*-tree: 2221 seconds**

Breast-w

701 records
9 real attributes
2 output values



Accuracy

Best canonical: 97.42%

Average canonical: 96.15%

AV: 96.19%

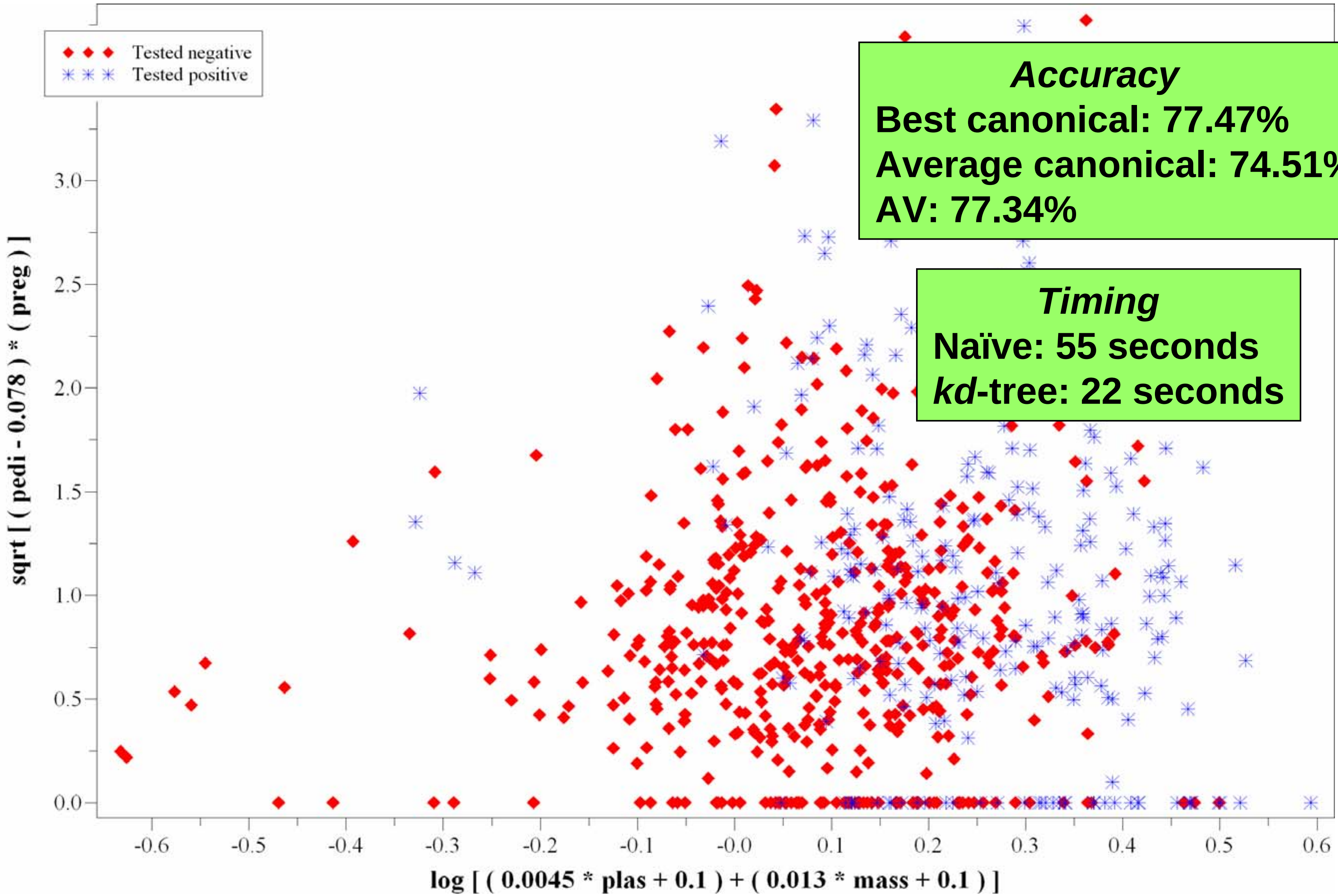
Timing

Naïve: 59 seconds

***kd*-tree: 22 seconds**

Diabetes

701 records
9 real attributes
2 output values



Accuracy

Best canonical: 77.47%

Average canonical: 74.51%

AV: 77.34%

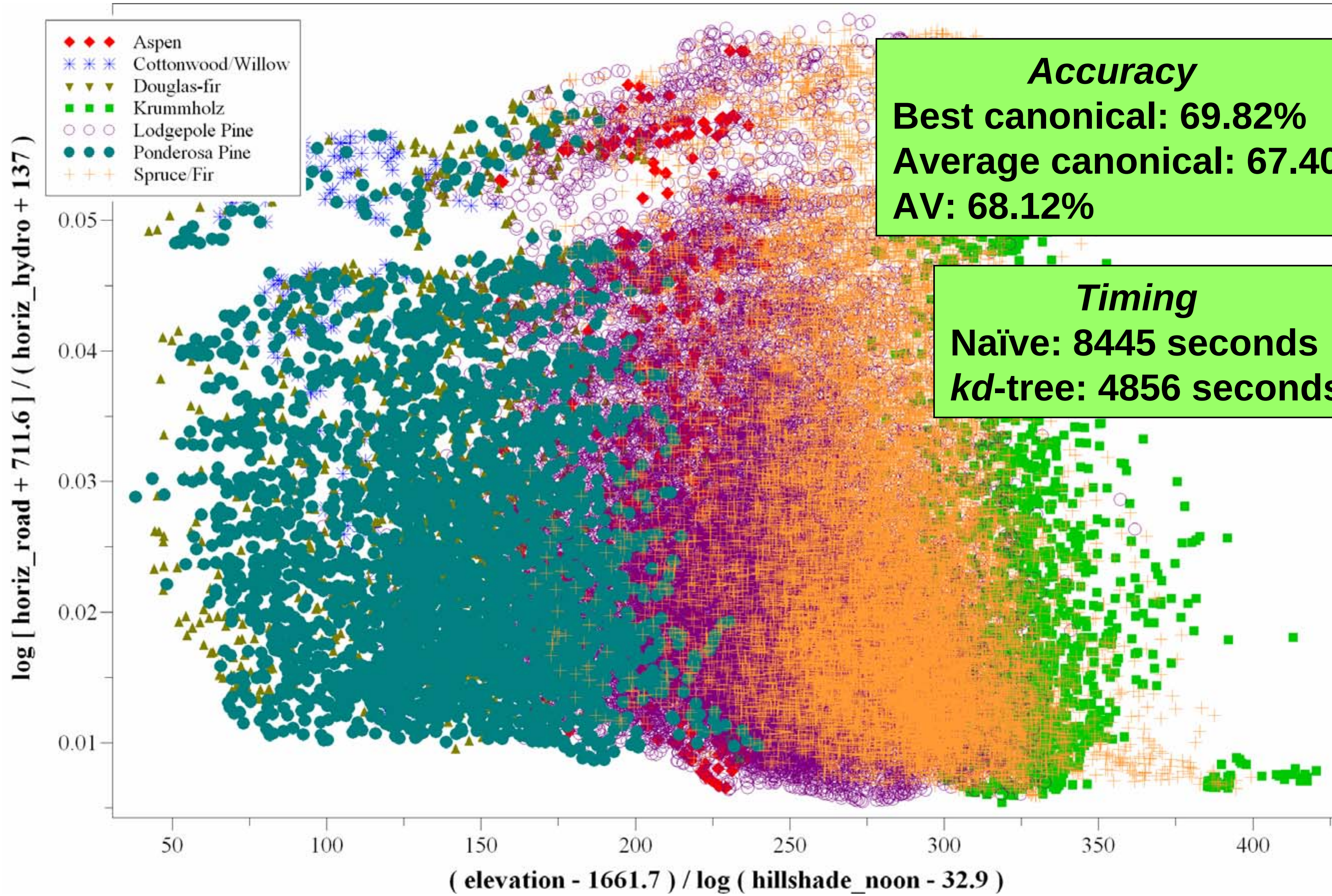
Timing

Naïve: 55 seconds

***kd*-tree: 22 seconds**

Forest

50000 records
10 real attributes
7 output values

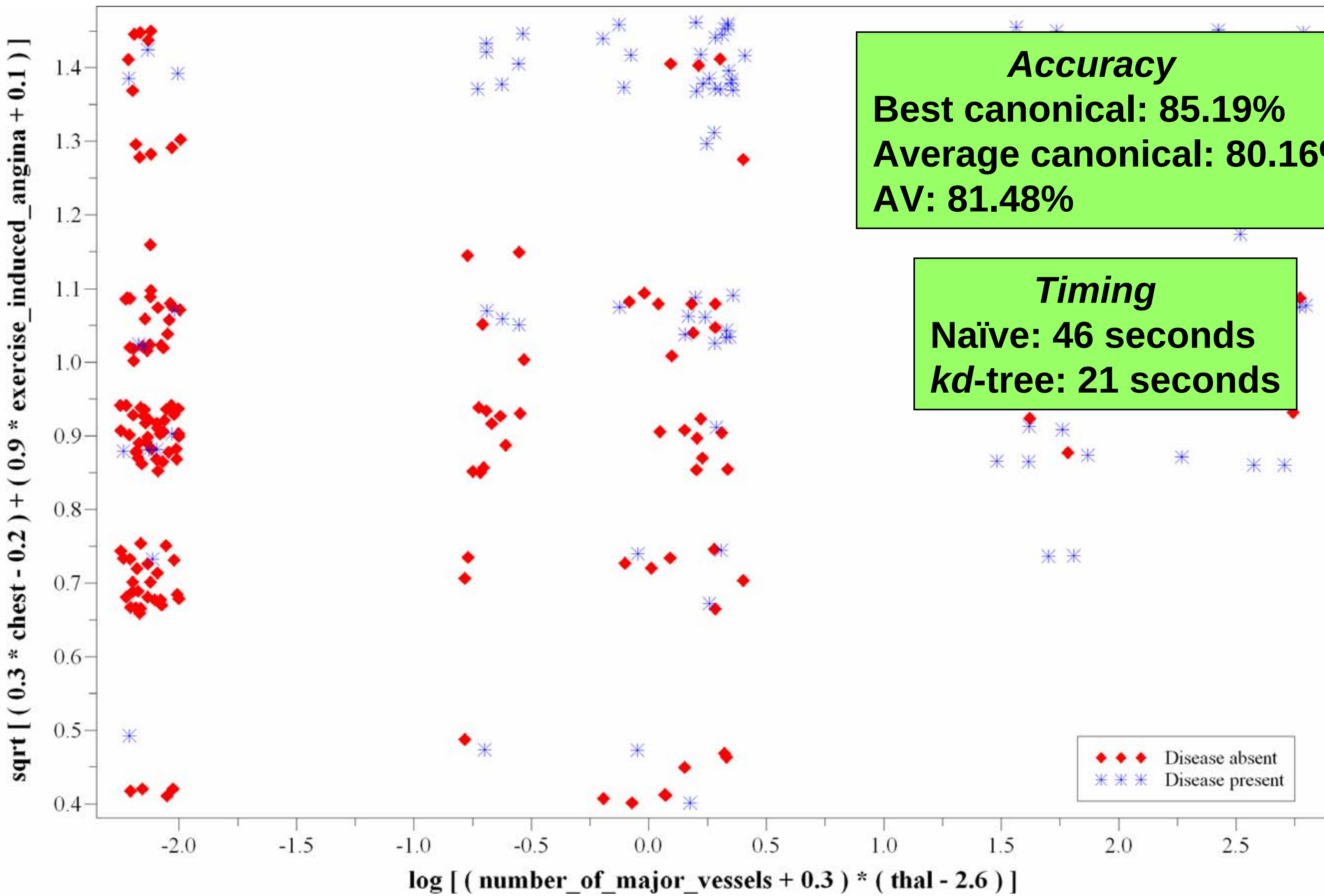


Accuracy
Best canonical: 69.82%
Average canonical: 67.40%
AV: 68.12%

Timing
Naïve: 8445 seconds
***kd*-tree: 4856 seconds**

Heart-statlog

272 records
13 real attributes
2 output values

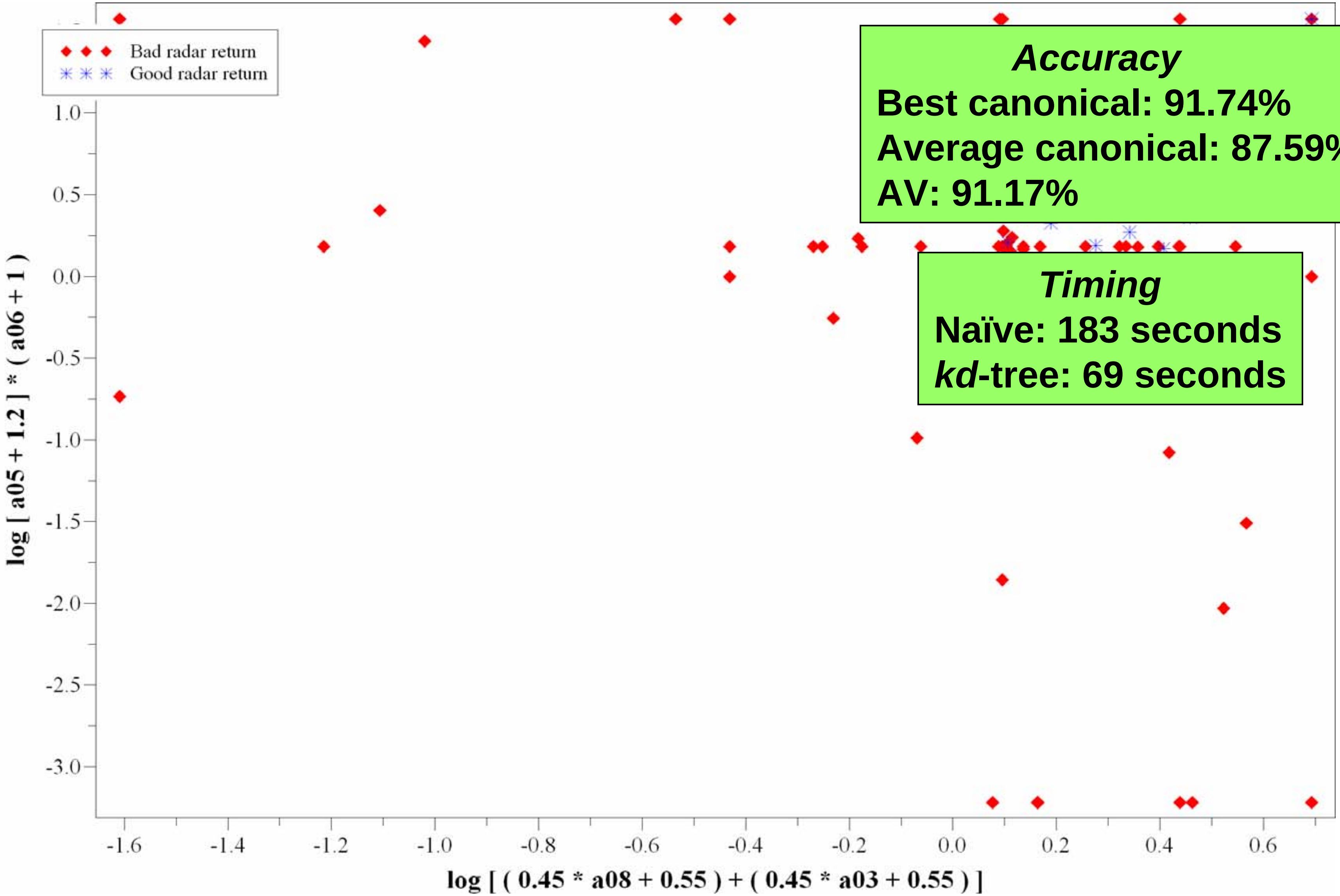


Accuracy
Best canonical: 85.19%
Average canonical: 80.16%
AV: 81.48%

Timing
Naïve: 46 seconds
***kd*-tree: 21 seconds**

Ionosphere

357 records
34 real attributes
2 output values

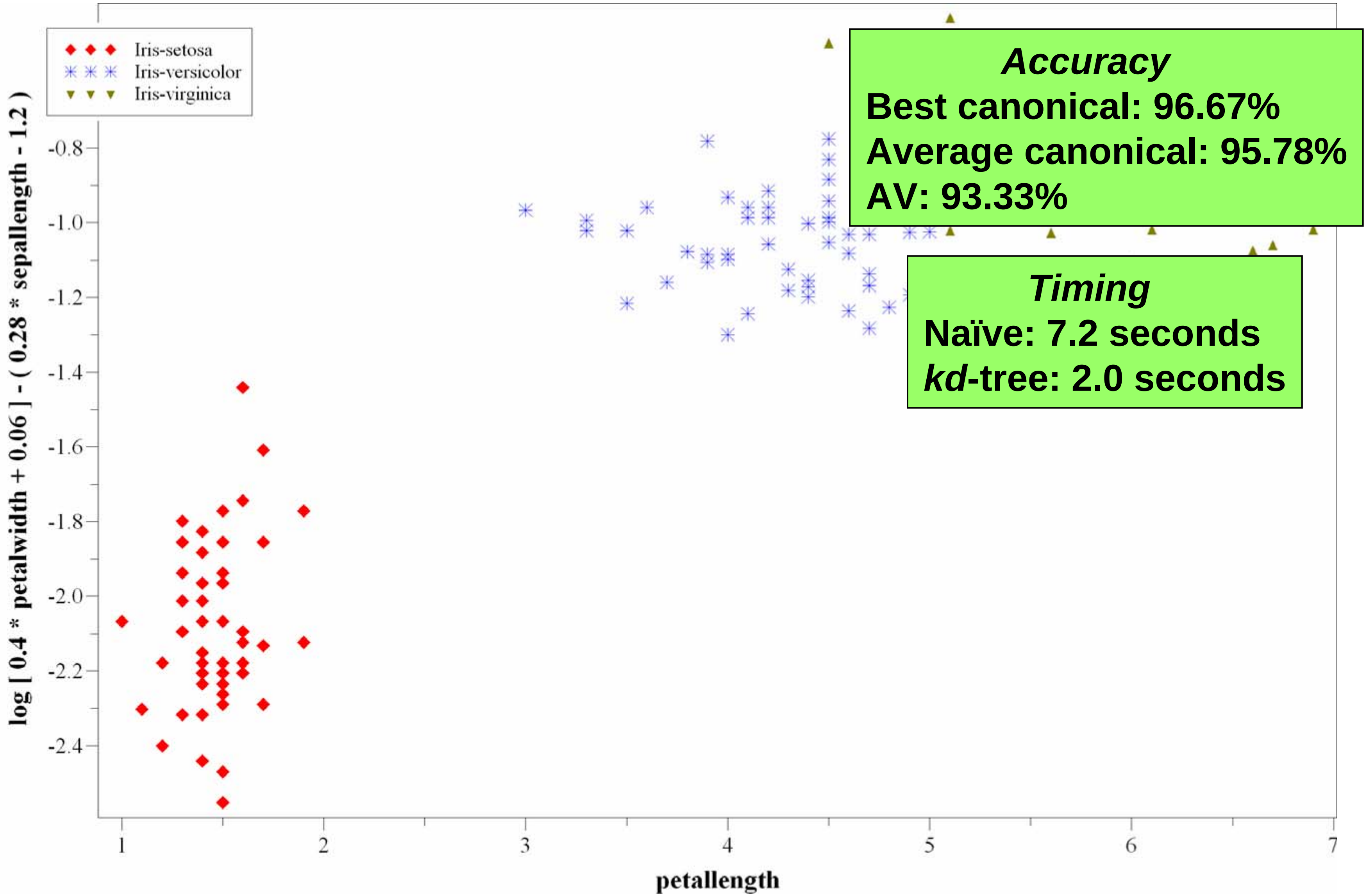


Accuracy
Best canonical: 91.74%
Average canonical: 87.59%
AV: 91.17%

Timing
Naïve: 183 seconds
***kd*-tree: 69 seconds**

Iris

155 records
4 real attributes
3 output values

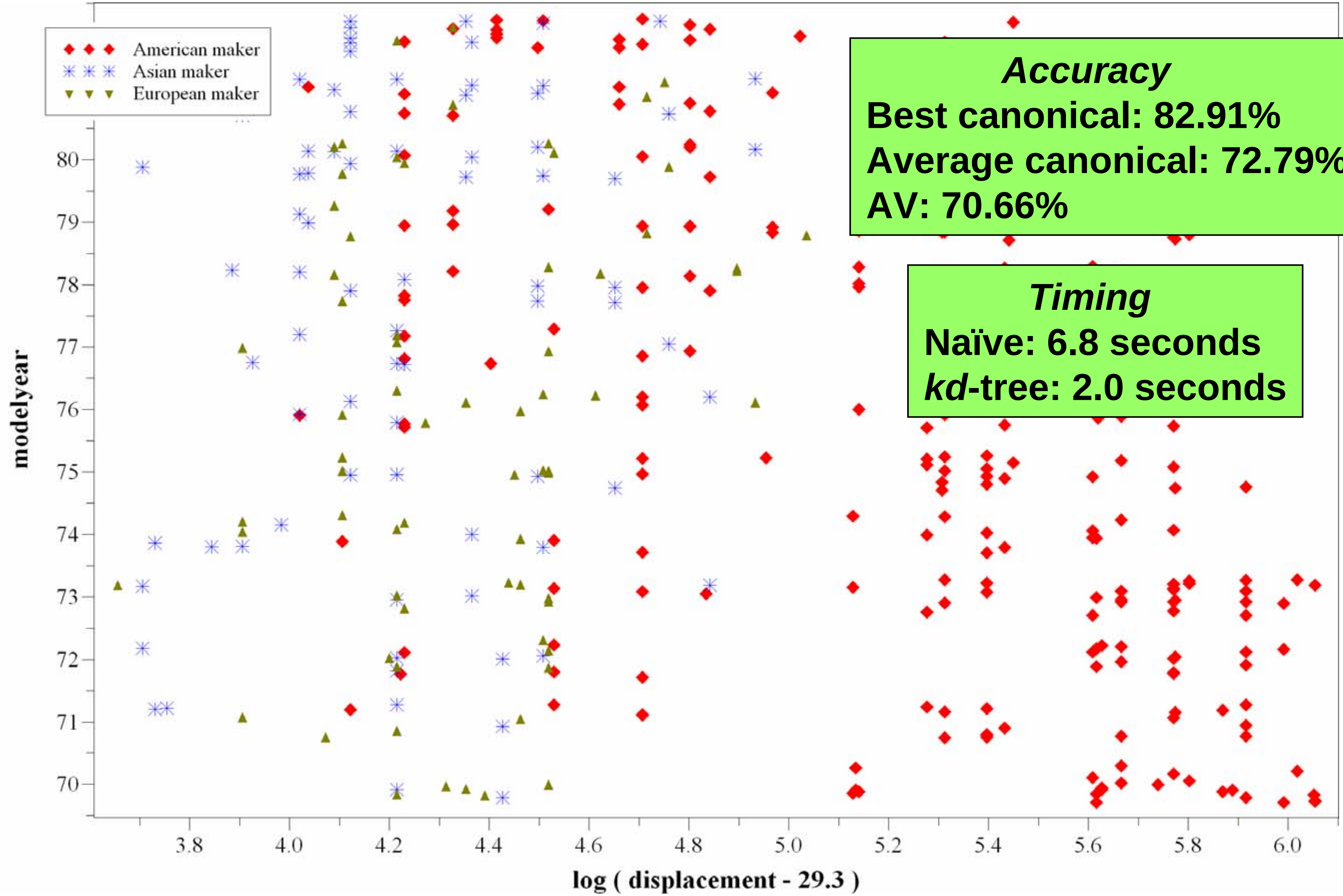


Accuracy
Best canonical: 96.67%
Average canonical: 95.78%
AV: 93.33%

Timing
Naïve: 7.2 seconds
***kd*-tree: 2.0 seconds**

Realmpg

394 records
7 real attributes
3 output values

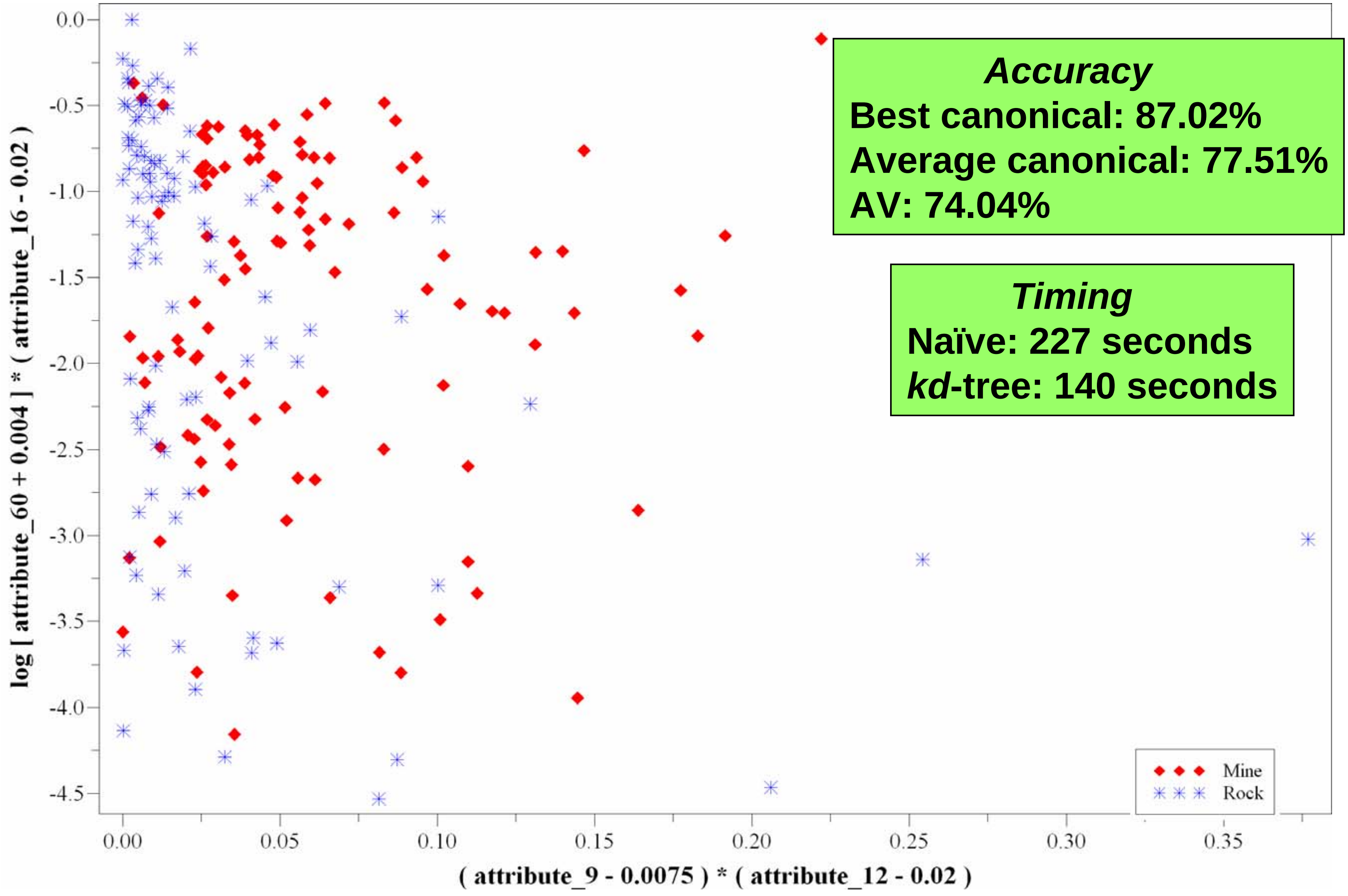


Accuracy
Best canonical: 82.91%
Average canonical: 72.79%
AV: 70.66%

Timing
Naïve: 6.8 seconds
***kd*-tree: 2.0 seconds**

Sonar

210 records
60 real attributes
2 output values

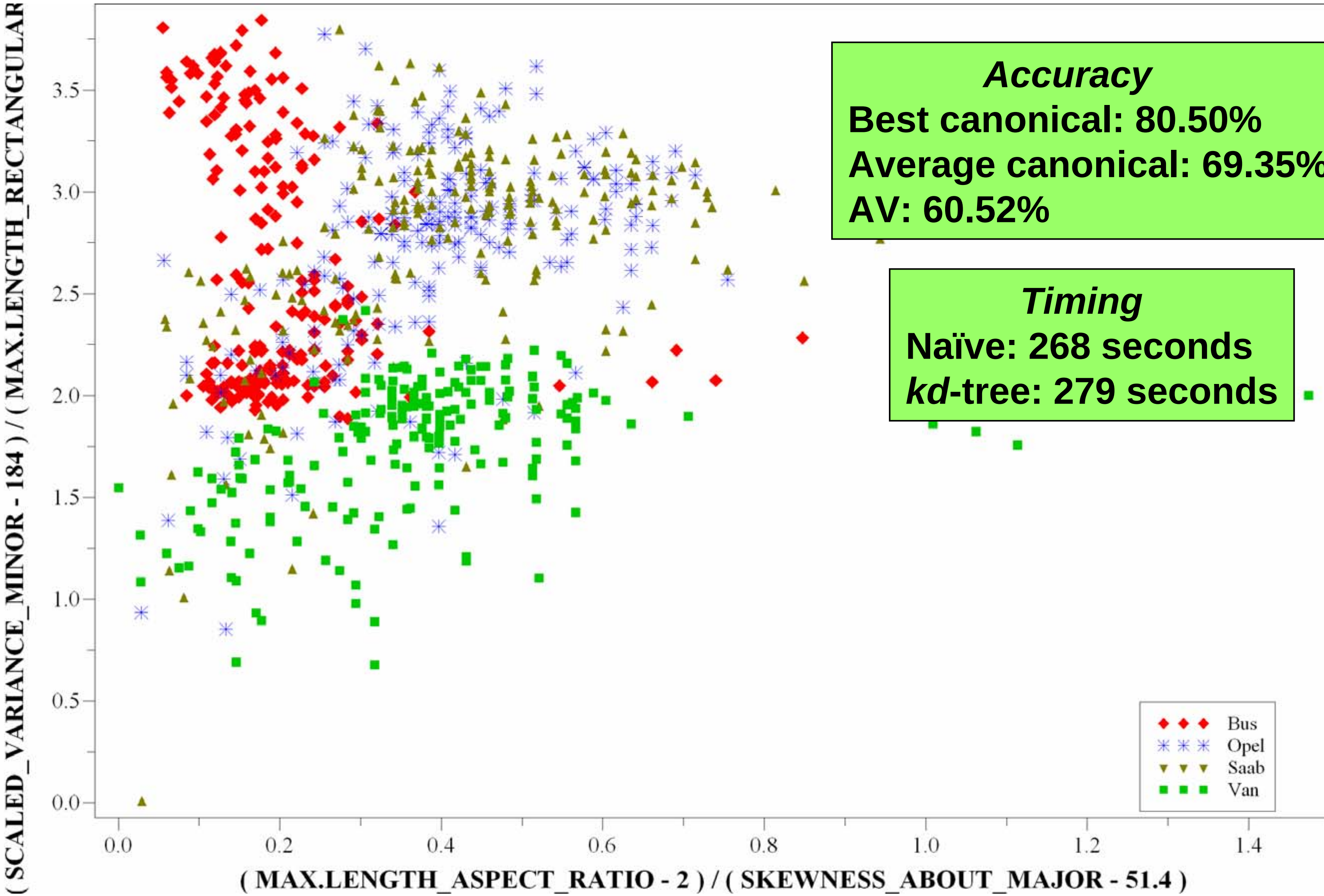


Accuracy
Best canonical: 87.02%
Average canonical: 77.51%
AV: 74.04%

Timing
Naïve: 227 seconds
***kd*-tree: 140 seconds**

Vehicle

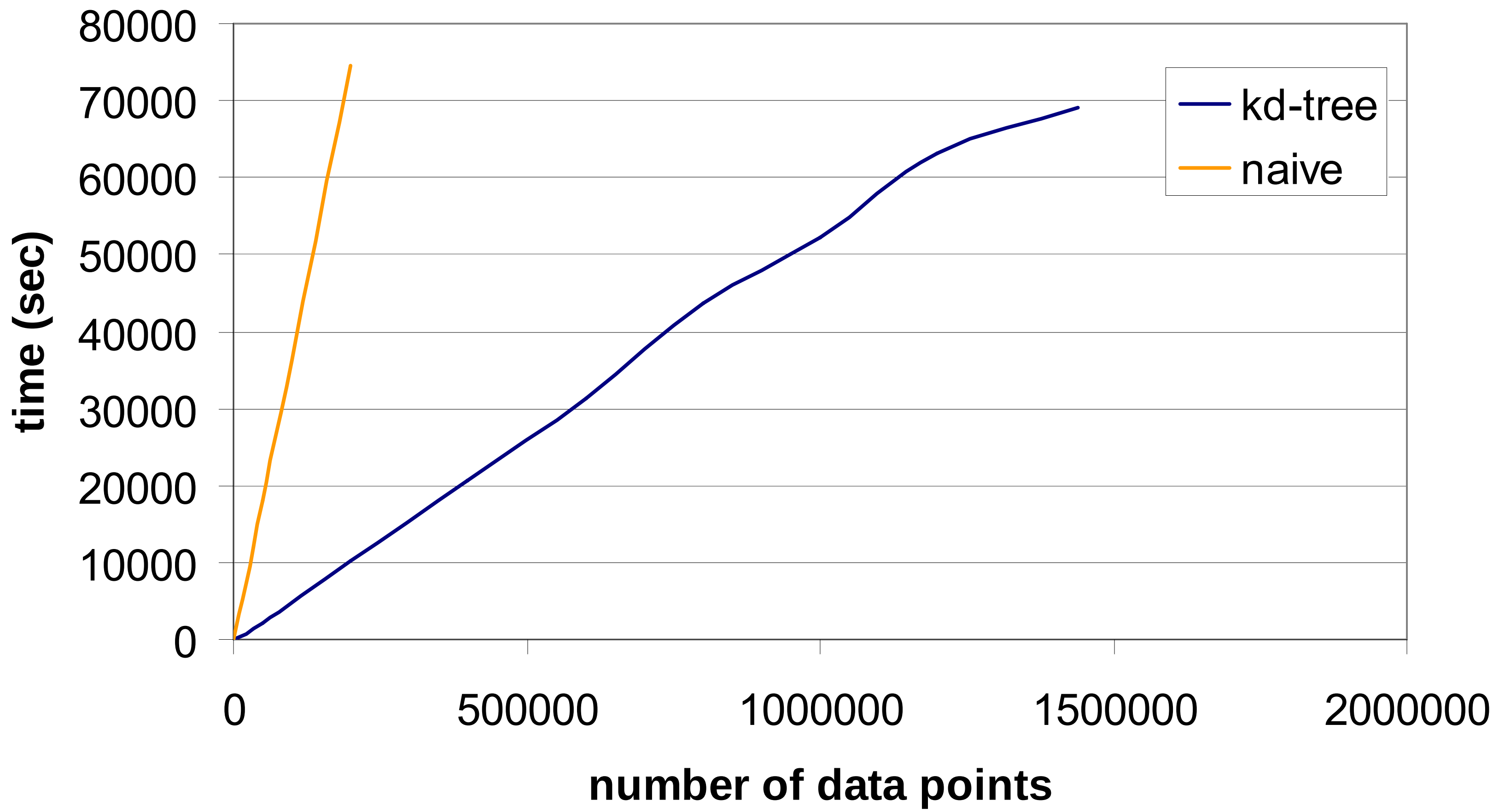
851 records
18 real attributes
4 output values



Accuracy
Best canonical: 80.50%
Average canonical: 69.35%
AV: 60.52%

Timing
Naïve: 268 seconds
***kd*-tree: 279 seconds**

Timing



AV timing analysis on expanded astro data set (i.e. full EDSGC), as number of records is increased

Conclusion

- Introduced a novel approach to classification that takes into account interpretability of the results
- Showed that there is little to be lost in terms of classification accuracy when additional interpretability is sought
  - In a pairwise comparison with nine canonical classifiers, AV does on average only 0.168 percentage points worse

Questions?

www.autonlab.org