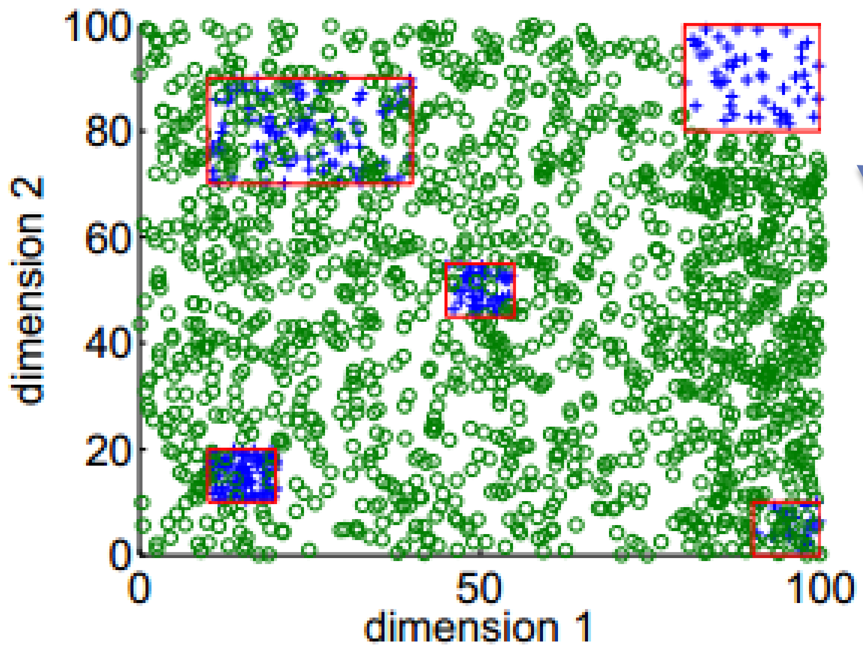
Box Drawings for Learning with Imbalanced Data

Siong Thye Goh & Cynthia Rudin
Massachusetts Institute of Technology

KDD 2014
20th ACM SIGKDD Conference on Knowledge Discovery and Data Mining
25 August 2014
New York City

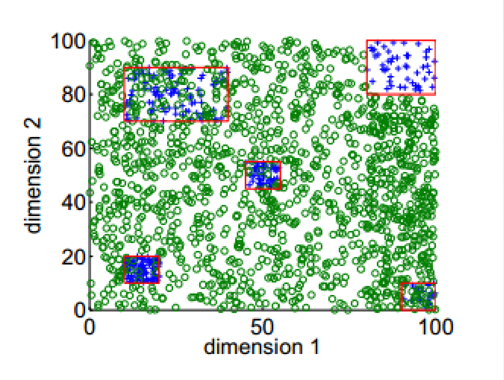
Think of…

- A few positive examples in a sea of negative examples. For modeling rare events: Machine breakdown, etc.



A box drawing classifier is a union of axis-parallel rectangles.

The usual way to do this...



Is by being greedy.

- Decision Tree (CART, C4.5 – Breiman 1984, Quinlan 1993)
  - Top down greedy: pick a features based on Gini Index or Information Gain, split data into two pieces, repeat. Prune afterwards.
- PRIM (Friedman, Fisher 1999)
  - Peel off subsets of data greedily, and if there is improvement, keep peeling off data. Occasionally put the data back.

This is too greedy for us.

In fact, greed leads to stupid answers...

- It's often hard to get anything except a trivial model that always predicts the majority class.

If our classifier says that all princesses have a thin waist, Princess Fiona from Shrek will feel unloved.

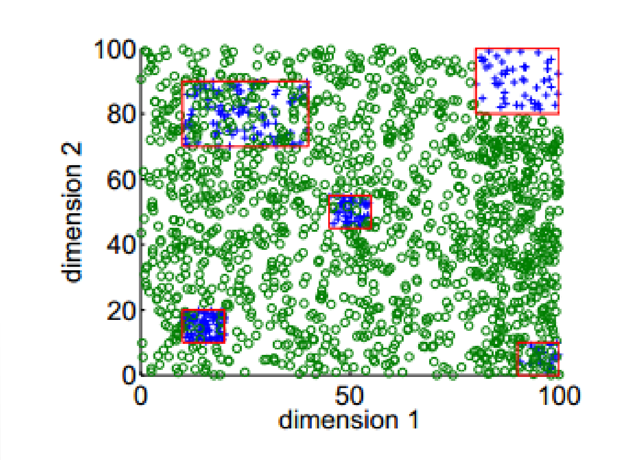
- This is true even if you fiddle like crazy with the algorithm's parameters. (We show some examples in the paper.)

- But since we are interested in the minority class, why don't we protect them?

Our Approaches

- Approach 1: The Exact Boxes algorithm
  - Optimize weighted accuracy, regularize by number of boxes
  - Mixed-Integer Programming formulation
  - Useful for not-huge problems, but solves exactly the problem we care about
  - Acts as a gold standard to compare with because it solves exactly the problem we want.

- Approach 2: The Fast Boxes algorithm
  - Approximates the solution of Exact Boxes
  - Characterize (one class learning) before discriminating
  - Requires that features are continuous.



Exact Boxes Algorithm

- Creates box drawing classifier
- A minority(positive) example is correctly classified if it resides within at least 1 box.
- A majority (negative) example is correctly classified if it does not reside in any box.
- Let *F* be the set of union of axis parallel rectangles.
- Here is Exact Boxes' objective:

$$\max_{f \in F} \sum_{i: y_i = 1} 1_{[f(x_i)=1]} + C_I \sum_{i: y_i = -1} 1_{[f(x_i)=-1]} - c_E(\#\text{ of boxes})$$

Give less weight to negative examples

Simplicity/sparsity Favors fewer boxes.

Exact Boxes Algorithm

$$\max_{l,\tilde{l},u,\tilde{u},w,z} \left[-c_e K + \sum_{i \in S_+} z_i + c_I \sum_{i \in S_-} z_i \right] \text{ subject to}$$

$$x_{ij} - l_{jk} - v \leq M\tilde{l}_{ijk}, \forall i \in S_+, \forall j, k$$
$$M(\tilde{l}_{ijk} - 1) + \epsilon \leq x_{ij} - l_{jk} - v, \forall i \in S_+, \forall j, k$$
$$u_{jk} - v - x_{ij} \leq M\tilde{u}_{ijk}, \forall i \in S_+, \forall j, k$$
$$M(\tilde{u}_{ijk} - 1) + \epsilon \leq u_{jk} - x_{ij} - v, \forall i \in S_+, \forall j, k$$
$$\textstyle\sum_{j=1}^{n} \tilde{l}_{ijk} + \sum_{j=1}^{n} \tilde{u}_{ijk} - 2n + 1 \leq w_{ik}, \forall i \in S_+, \forall j, k$$
$$2nw_{ik} \leq \textstyle\sum_{j=1}^{n} \tilde{l}_{ijk} + \sum_{j=1}^{n} \tilde{u}_{ijk}, \forall i \in S_+, \forall j, k$$
$$\textstyle\sum_{k=1}^{K} w_{ik} \leq Kz_i, \forall i \in S_+, \forall k$$
$$z_i \leq \textstyle\sum_{k=1}^{K} w_{ik}, \forall i \in S_+, \forall k$$
$$l_{jk} - v - x_{ij} \leq M\tilde{l}_{ijk}, \forall i \in S_-, \forall j, k$$
$$M(\tilde{l}_{ijk} - 1) + \epsilon \leq l_{jk} - v - x_{ij}, \forall i \in S_-, \forall j, k$$
$$x_{ij} - u_{jk} - v \leq M\tilde{u}_{ijk}, \forall i \in S_-, \forall j, k$$
$$M(\tilde{u}_{ij} - 1) + \epsilon \leq x_{ij} - u_{jk} - v, \forall i \in S_-, \forall j, k$$
$$\textstyle\sum_{j=1}^{n} \tilde{l}_{ijk} + \sum_{j=1}^{n} \tilde{u}_{ijk} - 2n + 1 \leq 2n(1 - w_{ik}),$$
$$\forall i \in S_-, \forall j, k$$
$$1 - w_{ik} \leq \textstyle\sum_{j=1}^{n} \tilde{l}_{ijk} + \sum_{j=1}^{n} \tilde{u}_{ijk}, \forall i \in S_-, \forall j, k$$
$$\textstyle\sum_{k=1}^{K} w_{ik} \leq K(1 - z_i), \forall i \in S_-, \forall k$$
$$1 - z_i \leq \textstyle\sum_{k=1}^{K} w_{ik}, \forall i \in S_-, \forall k$$
$$l_{jk} \leq u_{jk}, \forall j, k.$$

- This is a mixed-integer *linear* programming formulation
- Works nicely for not-huge problems, acts as a gold standard

Exact Boxes Algorithm

$$\max_{l,\tilde{l},u,\tilde{u},w,z} \left[-c_e K + \sum_{i\in S_+} z_i + c_I \sum_{i\in S_-} z_i\right] \text{ subject to}$$

$$x_{ij} - l_{jk} - v \le M\tilde{l}_{ijk}, \forall i \in S_+, \forall j,k$$

$$M(\tilde{l}_{ijk} - 1) + \epsilon \le x_{ij} - l_{jk} - v, \forall i \in S_+, \forall j,k$$

$$u_{jk} - v - x_{ij} \le M\tilde{u}_{ijk}, \forall i \in S_+, \forall j,k$$

$$M(\tilde{u}_{ijk} - 1) + \epsilon \le u_{jk} - x_{ij} - v, \forall i \in S_+, \forall j,k$$

$$\textstyle\sum_{j=1}^n \tilde{l}_{ijk} + \sum_{j=1}^n \tilde{u}_{ijk} - 2n + 1 \le w_{ik}, \forall i \in S_+, \forall j,k$$

$$2nw_{ik} \le \textstyle\sum_{j=1}^n \tilde{l}_{ijk} + \sum_{j=1}^n \tilde{u}_{ijk}, \forall i \in S \text{ [illegible]}$$

$$\textstyle\sum_{k=1}^K w_{ik} \le K z_i, \forall i \in \text{ [illegible]}$$

$$z_i \le \textstyle\sum_{k=1}^K \text{ [illegible] } \forall k$$

$$l_{jk} - v \text{ [illegible] }_{jk}, \forall i \in S_-, \forall j,k$$

$$M(\tilde{l} \text{ [illegible] } \le l_{jk} - v - x_{ij}, \forall i \in S_-, \forall j,k$$

$$\text{[illegible] } u_{jk} - v \le M\tilde{u}_{ijk}, \forall i \in S_-, \forall j,k$$

$$\text{[illegible]}(\tilde{u}_{ijk} - 1) + \epsilon \le x_{ij} - u_{jk} - v, \forall i \in S_-, \forall j,k$$

$$\textstyle\sum_{j=1}^n \tilde{l}_{ijk} + \sum_{j=1}^n \tilde{u}_{ijk} - 2n + 1 \le 2n(1 - w_{ik}),$$
$$\forall i \in S_-, \forall j,k$$

$$1 - w_{ik} \le \textstyle\sum_{j=1}^n \tilde{l}_{ijk} + \sum_{j=1}^n \tilde{u}_{ijk}, \forall i \in S_-, \forall j,k$$

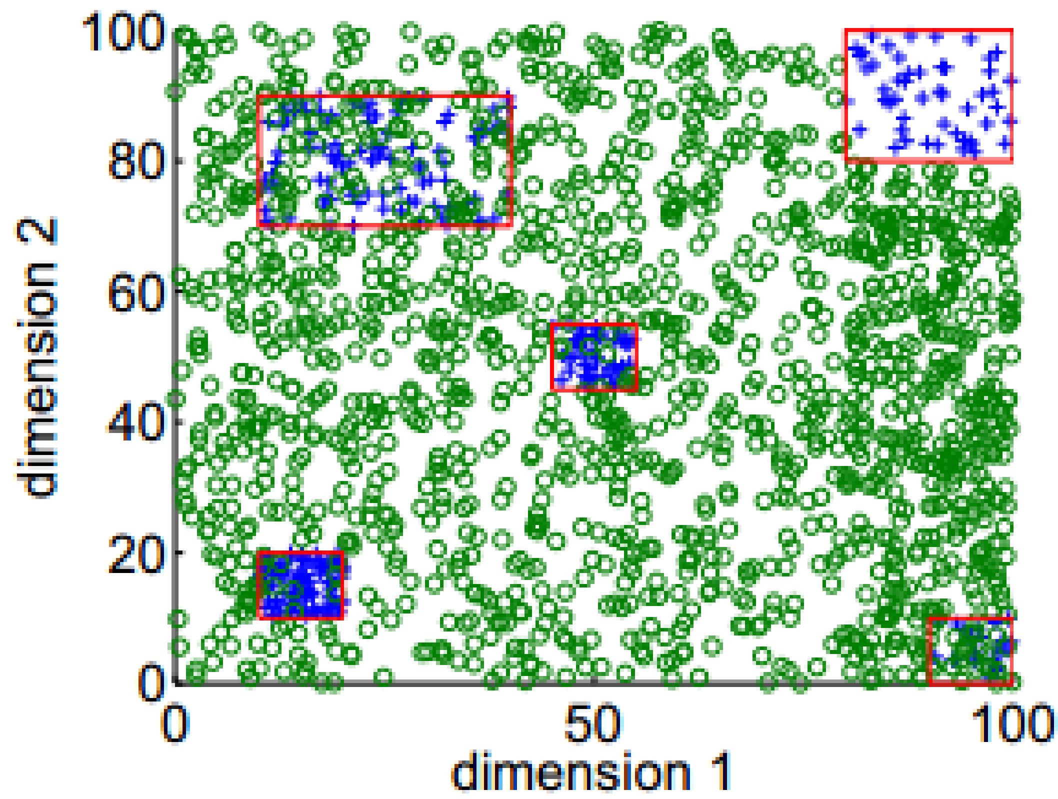
$$\textstyle\sum_{k=1}^K w_{ik} \le K(1 - z_i), \forall i \in S_-, \forall k$$

$$1 - z_i \le \textstyle\sum_{k=1}^K w_{ik}, \forall i \in S_-, \forall k$$

$$l_{jk} \le u_{jk}, \forall j,k.$$

Life is too short for details

- This is a mixed-integer *linear* programming formulation
- Works nicely for not-huge problems, acts as a gold standard

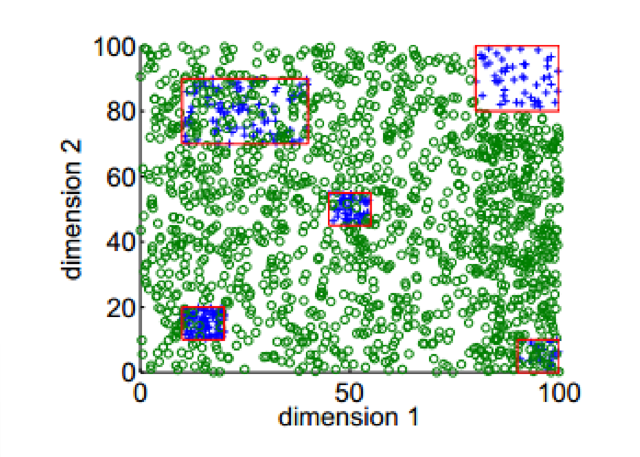


100
80
60
40
20
0
0
50
100
dimension 2
dimension 1

Our Approaches

- Approach 1: The Exact Boxes algorithm
  - Optimize accuracy, regularize by number of boxes
  - Mixed-Integer Programming formulation
  - Useful for not-huge problems, but solves exactly the problem we care about
  - Acts as a gold standard to compare with because it solves exactly the problem we want.

- Approach 2: The Fast Boxes algorithm
  - Approximates the solution of Exact Boxes
  - Characterize (one class learning) before discriminating
  - Requires that features are continuous.



Fast Boxes Algorithm

- Three main stages:

1. Clustering.
   - Characterization of positive data only.
2. Dividing space stage.
   - Decide which points will influence each boundary.
3. Boundary expansion stage.
   - Expand each box to give priority to the positive data.

Fast Boxes Algorithm

- Three main stages:

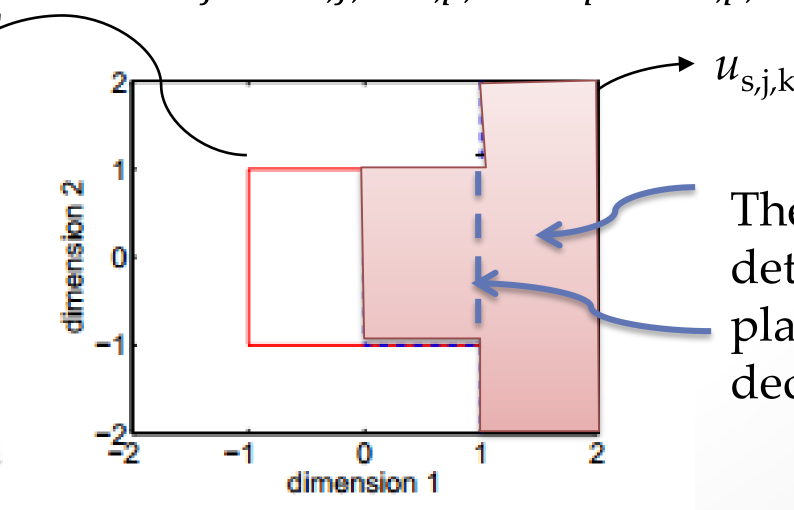
1. Clustering.
    - Characterization of positive data only.
    - This is just K means, which assumes continuous features.
2. Dividing space stage.
    - Decide which points will influence each boundary.
3. Boundary expansion stage.
    - Expand each box to give priority to the positive data.

Fast Boxes Dividing Space Stage

- Each boundary is determined by 1d supervised learning using data close to the boundary.
- We have analytical solution for each decision boundary.

$$X_{l,j,k} = \{x: x_j \le l_{s,j,k}\} \cup \left\{x: l_{s,j,k} \le x_j \le \frac{l_{s,j,k} + u_{s,j,k}}{2}, l_{s,p,k} \le x_p \le u_{s,p,k}, p \ne j\right\}$$

$$X_{u,j,k} = \{x: x_j \ge u_{s,j,k}\} \cup \left\{x: \frac{l_{s,j,k} + u_{s,j,k}}{2} \le x_j \le u_{s,j,k}, l_{s,p,k} \le x_p \le u_{s,p,k}, p \ne j\right\}$$

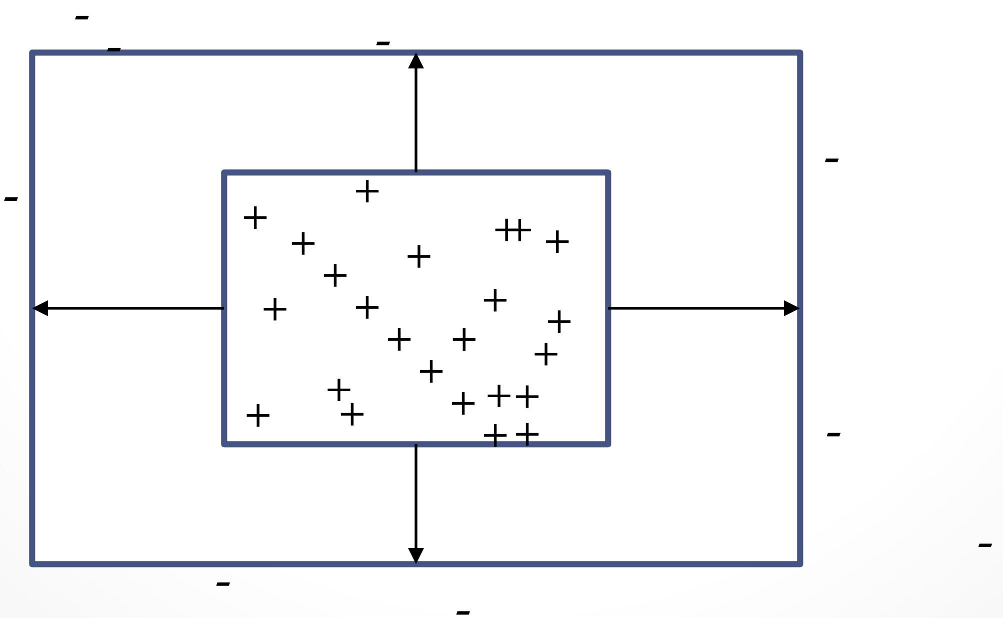


The points in here determine the placement of this decision boundary.

Analytical solutions are better than chocolate.

Boundary Expansion Stage

- We push decision boundaries for each box outwards as a form of regularization.
- This is a final optional push to almost the next negative point.



Analytical solutions are even better than cupcakes.

Summary of Fast Boxes Algorithm

1. Normalize the features to be between -1 and 1.
2. Cluster the minority data into K clusters (take standard deviation into consideration)
3. Construct the minimal enclosing box for each cluster.
4. Pick which data participate in each decision boundary calculation.
5. Compute decision boundaries analytically and separately
6. Perform expansion (optional)
7. Un-normalize by rescaling the features back.

Experiments

Baseline Algorithms

- Non-interpretable Models:
  - o Logistic Regression.
  - o SVM with RBF kernels
  - o Random Forests
  - o AdaBoost (boosted trees)
- Interpretable Models:
  - o CART
  - o C4.5
  - o C5.0
  - o Hellinger Distance Decision Tree

déjà vu, didn't you see this in the previous talk?

Performance analysis

- Compare Fast Boxes with baseline algorithms.
- It is often (but not always) one of the best performers.
- However the solution is often more interpretable.

Experimental Comparison

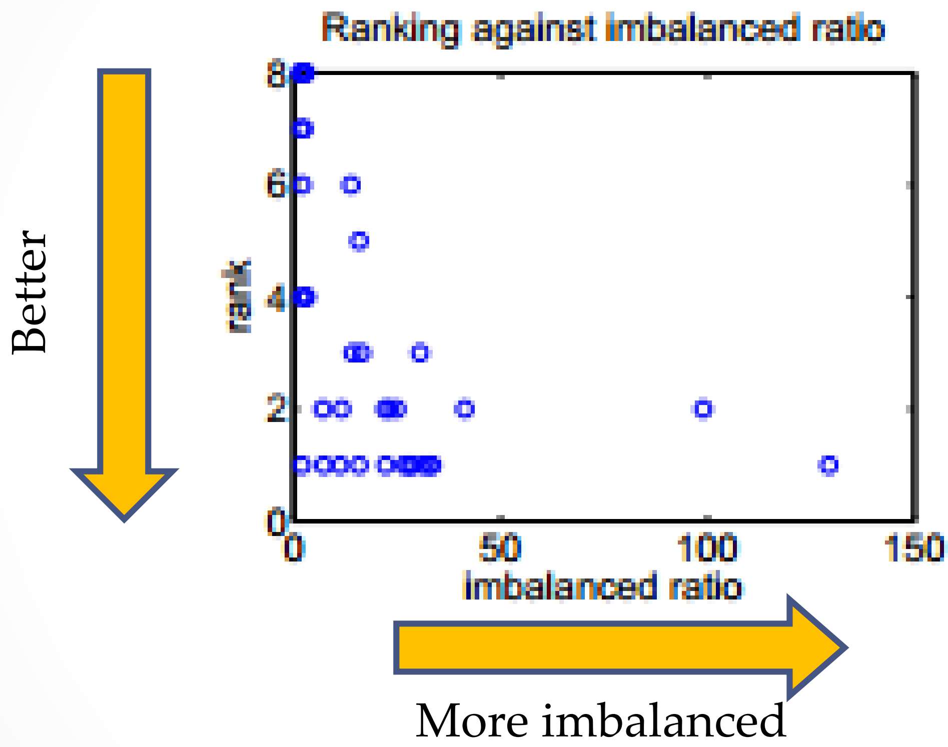
Data	Logistic	SVM	CART	C4.5	Ada-Boost	RF	C5.0	HDDT	Fast Boxes
pima	**0.8587** (0.0112)	**0.8468** (0.0126)	0.7738 (0.0123)	0.6579 (0.0347)	0.6810 (0.0218)	0.6942 (0.0126)	0.6574 (0.0353)	0.6642 (0.0274)	0.7298 (0.0241)
castle	0.5 (0)	**1** (0)	0.9941 (0.0068)	0.9947 (0.0060)	**0.9949** (0.0046)	**0.9922** (0.0079)	0.9941 (0.0060)	**0.9949** (0.0062)	**1** (0)
corner	**0.9871** (0.0129)	**0.9948** (0.0005)	0.9488 (0.2717)	0.5997 (0.1482)	0.6984 (0.0449)	0.6828 (0.0265)	0.5612 (0.1110)	0.6865 (0.0365)	0.9891 (0.0001)
diamond	0.5 (0)	**0.9980** (0.0004)	0.9585 (0.0129)	0.9328 (0.0181)	0.9460 (0.0117)	0.9433 (0.0121)	0.9311 (0.0208)	0.9364 (0.0180)	0.9744 (0.0062)
square	0.5404 (0.0718)	0.9944 (0.0001)	0.9949 (0.0051)	0.9949 (0.0043)	0.9939 (0.0033)	0.9947 (0.0033)	0.9949 (0.0043)	0.9949 (0.0027)	**0.9984** (0.0015)
flooded	0 (0)	**0.9831** (0.0010)	0.9466 (0.0157)	0.5488 (0.1074)	0.7017 (0.0231)	0.7036 (0.0252)	0.5482 (0.1077)	0.6992 (0.0208)	09638 (0.0091)
fourclass	0.8122 (0.0195)	**0.9957** (0.0176)	0.9688 (0.0176)	0.9916 (0.0296)	0.9670 (0.0265)	**0.9920** (0.0053)	0.9670 (0.0130)	0.9698 (0.0116)	0.9546 (0.0174)
castle3D	0.5449 (0.0324)	**1** (0)	0.9532 (0.0347)	0.9530 (0.0374)	0.9272 (0.0499)	**0.9455** (0.05633)	0.9439 (0.0615)	0.9530 (0.0374)	**1** (0)
corner3D	0.8448 (0.0316)	**0.9225** (0.0463)	0.8481 (0.0504)	0.5596 (0.0729)	0.6245 (0.03927)	0.5657 (0.0309)	0.5622 (0.0778)	0.6413 (0.0457)	**0.9736** (0.0091)
diamond3D	0.5449 (0.0324)	**0.7962** (0.0917)	0.7372 (0.0347)	0.5 (0.0374)	0.5492 (0.0499)	0.5957 (0.0309)	0.5622 (0.0778)	0.6883 (0.0542)	**0.9516** (0.0119)
square3D	0.5 (0)	**0.9626** (0.0156)	**0.9106** (0.0306)	0.5387 (0.1224)	0.8703 (0.01451)	0.8790 (0.0234)	0.5811 (0.1712)	0.9034 (0.0322)	**0.9578** (0.0090)
flooded3D	0.5 (0)	0.7912 (0.0781)	0.7724 (0.0902)	0.5 (0)	0.5471 (0.0329)	0.5489 (0.0440)	0.5 (0)	0.6422 (0.0749)	**0.9233** (0.0307)
breast	0.9297 (0.0230)	**0.9801** (0.0079)	0.9516 (0.0173)	0.9251 (0.0138)	0.9457 (0.0329)	0.9609 (0.0102)	0.9281 (0.0135)	0.9231 (0.0180)	0.8888 (0.0313)

Excessive amount of experiments that show:

1) Fast Boxes has the necessary complexity to produce high accuracy results (like SVM)

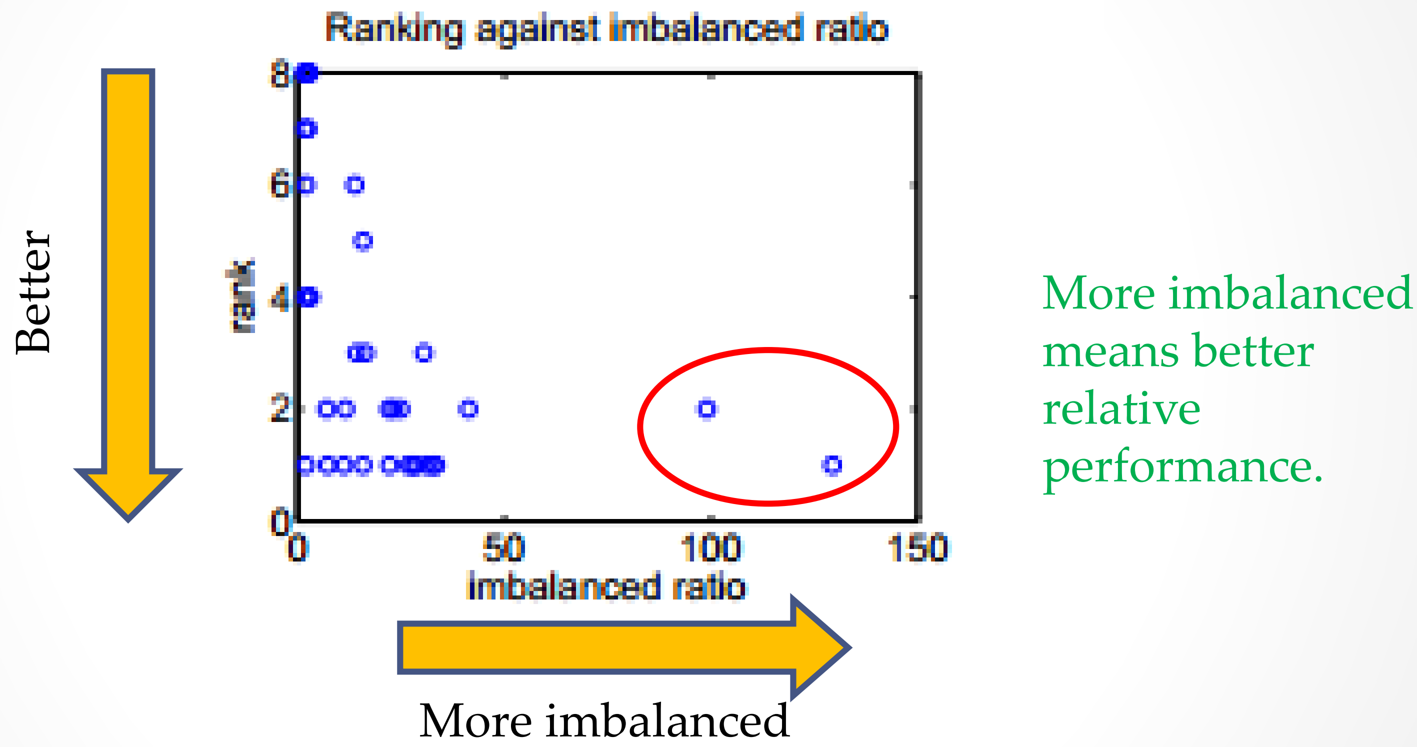
2) Interpretability doesn't hurt a bit.

When does Fast Boxes perform well? When data are more imbalanced.



- Each dot represents a dataset we tried, and the rank of Fast Boxes as compared with the other methods.
- Bottom line: if the data are more imbalanced, Fast Boxes does better.

When does Fast Boxes perform well? When data are more imbalanced.



- Each dot represents a dataset we tried, and the rank of Fast Boxes as compared with the other methods.
- Bottom line: if the data are more imbalanced, Fast Boxes
- does better.

Exact Boxes is frequently one of the Best Performers

Data	Best Performance	Fast Boxes	Exact Boxes	Exact Boxes ranking
vehicle2	0.9496 (0.015)	0.9191 (0.0242)	0.9496 (0.015)	1
haberman	0.6699 (0.0276)	0.5290 (0.0265)	**0.6632** (0.0303)	2
yeast1	0.7641 (0.0133)	0.5903 (0.0286)	0.7392 (0.0172)	2
glass0	0.8312 (0.0345)	0.7937 (0.0212)	0.7977 (0.0421)	2
iris0	1 (0)	**1** (0)	**1** (0)	1
wisconsin	0.9741 (0.0075)	0.8054 (0.1393)	**0.9726** (0.0079)	2
ecoli01	0.9840 (0.0105)	0.9433 (0.0300)	**0.9839** (0.0109)	2
glass1	0.7922 (0.0377)	0.6654 (0.0356)	**0.7922** (0.0337)	1

- Restricting the algorithm to produce a box drawing classifier does not generally seem to hinder performance.
- Selecting optimal K is difficult for Exact Boxes. Optimal K for Fast Boxes is used instead.

Summary

- Exact Boxes
  - Mixed integer Programming
- Fast Boxes
  - Characterize-then-discriminate approach.

- Take away: Aim for both interpretability and accuracy, because you can often get both.






Thank you!