KU LEUVEN

Designing a Stability Metric for Assessing the Robustness of Anomaly Rankings

Connor Galvin

Daily Advisors: Lorenzo Perini and Vincent Vercruyssen

Promoter: Jesse Davis

What is stability?

- *Holistic Assessment of Structure Discovery Capabilities of Clustering Algorithms* - Hoppner and Jahnk

- Point Stability – If x and y belong to the same cluster on an algorithm executed on a random sample, what is the probability that they belong to the same cluster if the full population was clustered?

- However: The idea of stability has not yet been applied to anomaly rankings

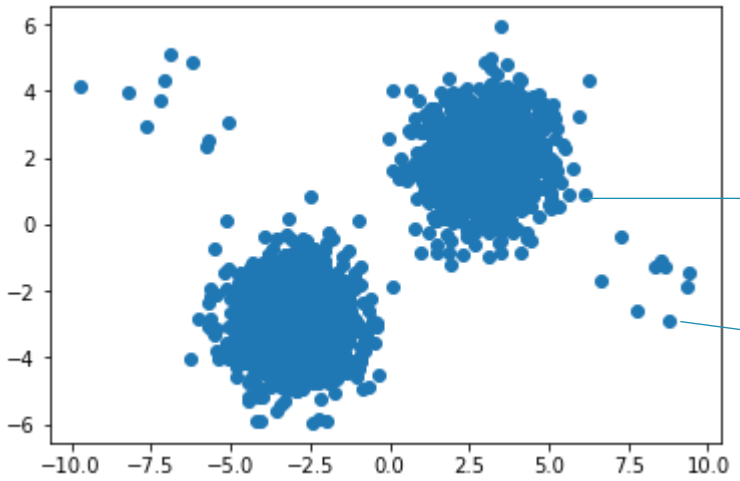
Stability for anomaly rankings

- How robust are anomaly rankings of test data to changes in training data?
- Idea: Hold test data constant. Compute anomaly rankings of test data with algorithms trained on different random subsets of training data.

Input
D, Dataset containing outliers
γ, contamination factor
h, anomaly detector model
I, number of subsample iterations

Output
S_h, stability of model h on dataset D

Pointwise Stability score



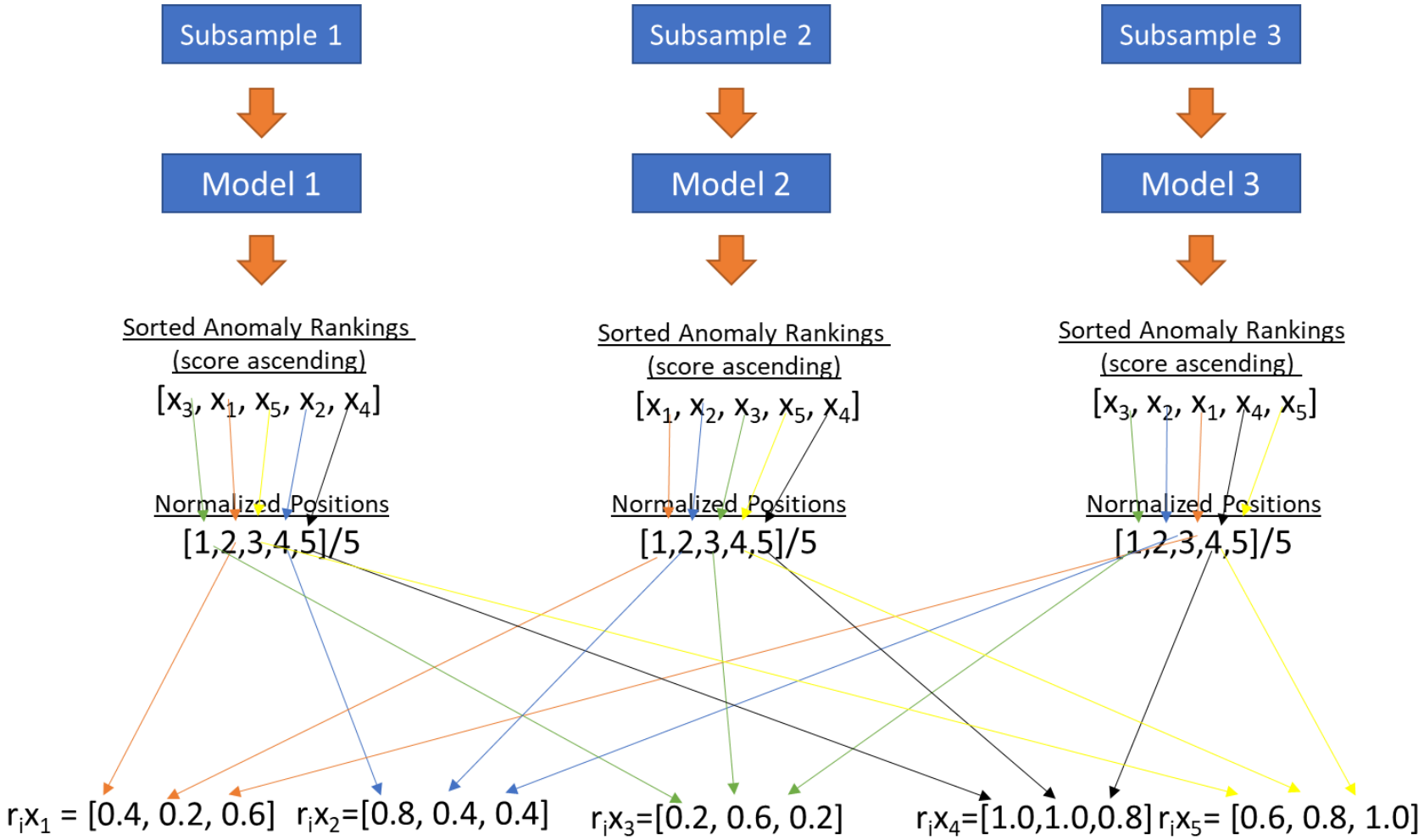
Iteration	Anomaly Ranking
1	0.11
2	0.18
3	0.53
4	0.09
5	0.22
…..	….

Iteration	Anomaly Ranking
1	0.92
2	0.97
3	0.88
4	0.94
5	0.86
…..	….

Goal

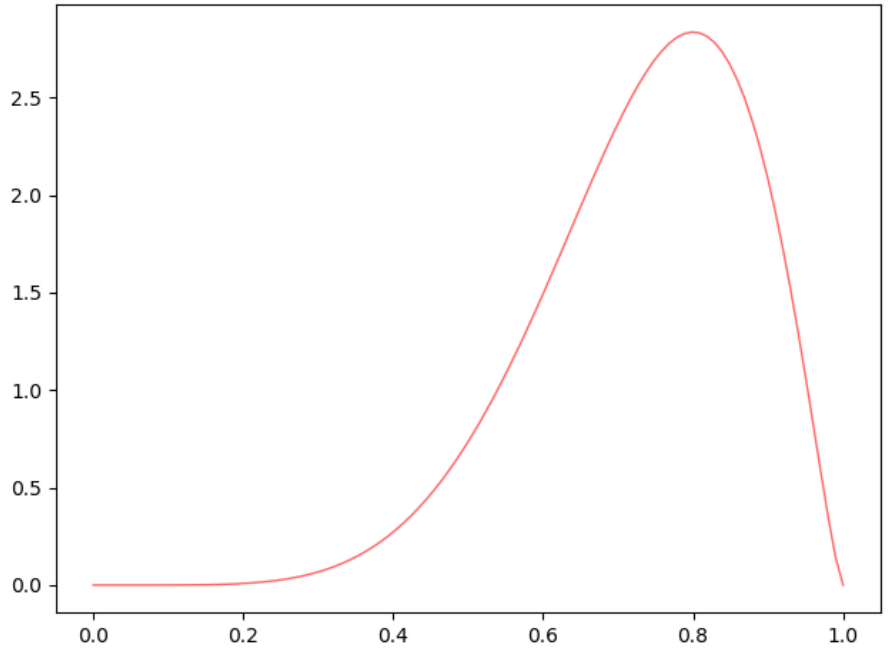
Transform this data into a single stability score for each point. Compute stability of model as average of all test points

Generating and sorting rankings



Area under Beta distribution

- Proposal: Use area under Beta distribution to measure changes in anomaly rankings
- Changes close to the anomaly threshold will receive more weight

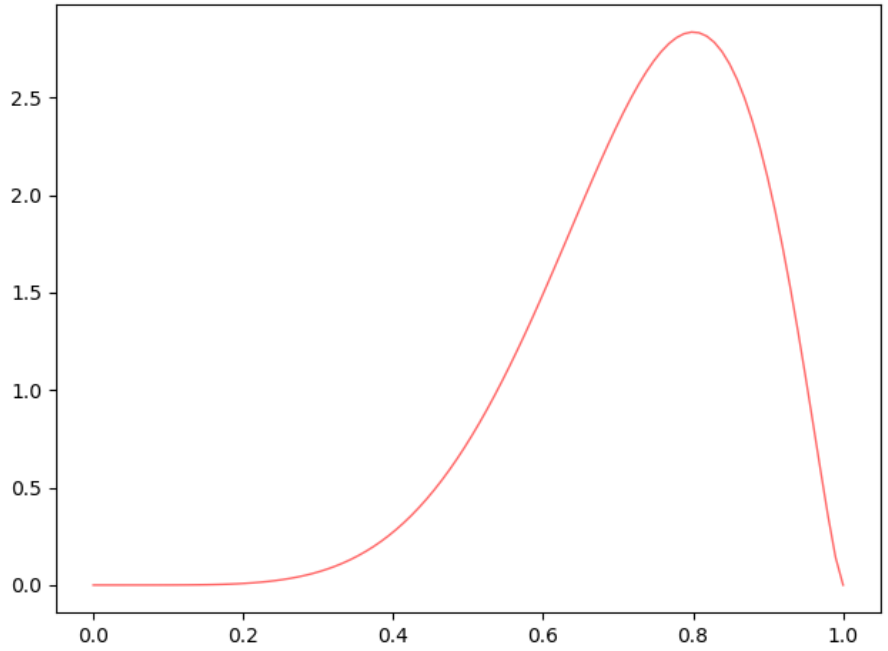


$$f(x;\alpha,\beta) = \text{constant} \cdot x^{\alpha-1}(1-x)^{\beta-1}$$

Area under Beta distribution

- Integrate between minimum and maximum relative anomaly ranking for each point
- Multiply by variance to capture the distribution of rankings

$$\sigma(r_i(x_j)) \times \int_{\min(r_i(x_j))}^{\max(r_i(x_j))} Beta$$



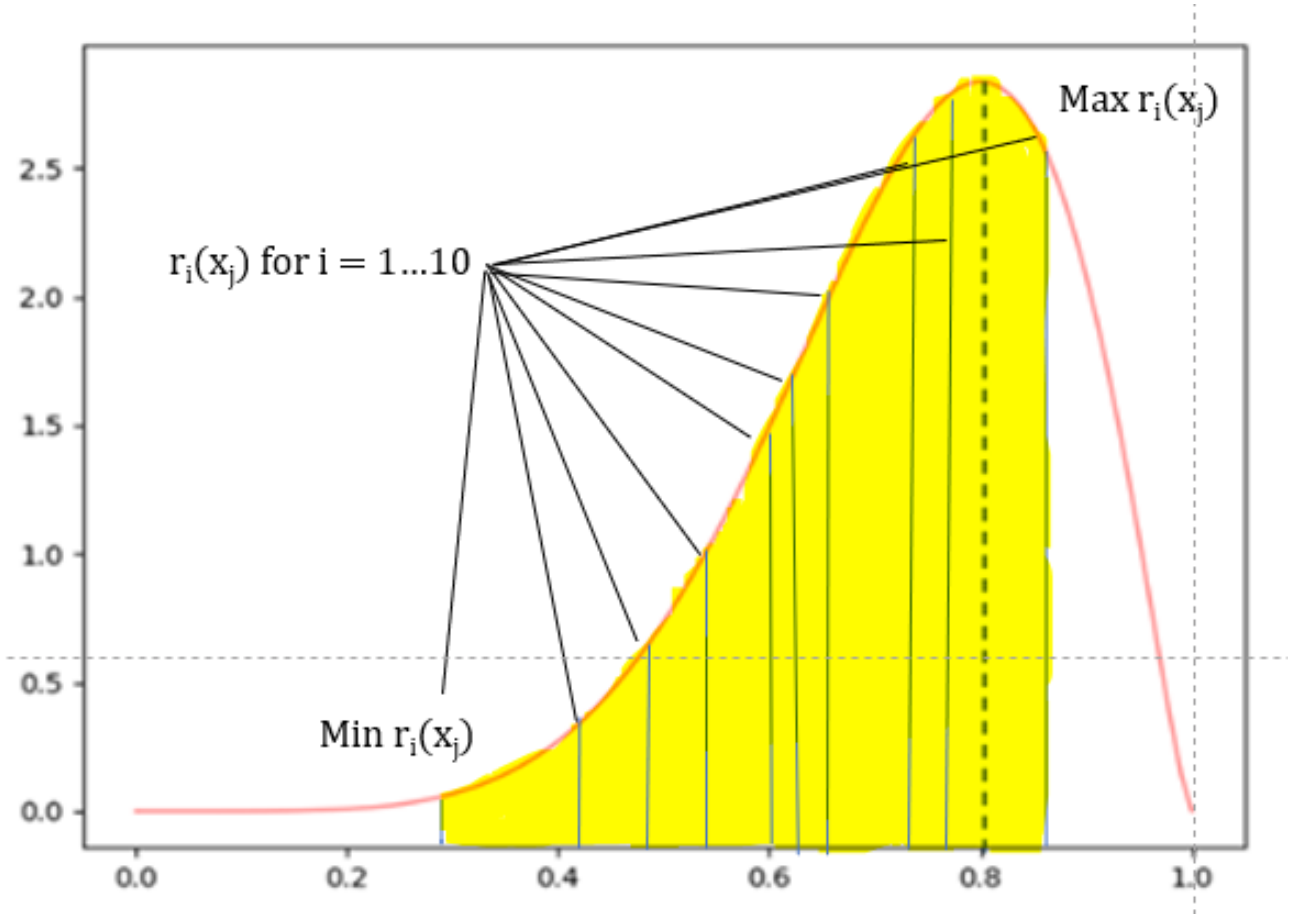
Parameters of Beta

$$Mode = \frac{\alpha - 1}{\alpha + \beta - 2}$$

$$\alpha = \beta \left(\frac{1 - \gamma}{\gamma} \right) + \frac{2\gamma - 1}{\gamma}$$

Adjust parameters to give more or less weight to changes close to anomaly threshold

Capturing area under the beta



How to make sense of this score?

Perfectly Stable

A point will always have the same anomaly ranking

$$\int_{\min(r_i(x_j))}^{\max(r_i(x_j))} Beta = 0$$

$$\sigma(r_i(x_j)) = 0$$

$$\sigma(r_i(x_j)) \times \int_{\min(r_i(x_j))}^{\max(r_i(x_j))} Beta = 0$$

Random

A point's anomaly ranking will be randomly distributed between 0 and 1

Over large t, $\int_{\min(r_i(x_j))}^{\max(r_i(x_j))} Beta \approx 1$

$$\sigma(r_i(x_j)) \approx \sqrt{\frac{(t+1)(t-1)}{12t^2}}$$

$$\sigma(r_i(x_j)) \times \int_{\min(r_i(x_j))}^{\max(r_i(x_j))} Beta \approx \sqrt{\frac{(t+1)(t-1)}{12t^2}}$$

Stability Score

Name: Stability
input : D, dataset containing outliers ;
γ, contamination factor;
h, anomaly detector model;
I, number of subsample iterations
output: Stability score $\mathcal{S}_h$ for the anomaly detector h on dataset D

Training set G, Test set T;
for $1 < i < I$ **do**
  G_i = draw a random subsample from G ;
  $h_i \leftarrow$ h.fit(G_i);
  $Scores_i \leftarrow h_i$.decisionfunction(T);
  Compose $Rankings_i$ from $Scores_i$ according to Figure 1;
end
Initialize Beta distribution;
Tune α and β parameters using Eq. 1 and 2;
for $1 < t < |T|$ **do**
  Compose $r_i(x_t)$ from $Rankings_i$ for $i = 1, \ldots, I$ according to Figure 1;
  Compute S_t from $r_i(x_t)$ for $i = 1, \ldots, I$ according to Eq. 3 and 4
end
Compute S_h from S_t for $t = 1, \ldots, |T|$ according to Eq. 8;
return S_h

Algorithm 1: Stability

Pointwise Instability Score

$$\mathcal{IS}_{x_j} = \frac{\sigma(r_i(x_j)) \times \int_{\min(r_i(x_j))}^{\max(r_i(x_j))} Beta}{\sigma_{rand}}$$

Pointwise Stability Score

$$\mathcal{S}_{x_j} = 1 - \mathcal{IS}_{x_j}$$

Model Stability Score

$$\mathcal{S}_h = \frac{1}{t} \sum_{i=1}^{t} S_{x_i}$$

Point Stability Score Calculation

Anomaly rankings for one point out of 100

Iteration	Anomaly Ranking
1	0.11
2	0.18
3	0.53
4	0.09
5	0.22

$$\int_{0.11}^{0.53} Beta \approx 0.148$$

$$\sigma(r_i(x_j)) \approx 0.159$$

$$\sigma(r_i(x_j)) * \int_{0.11}^{0.53} Beta \approx 0.081$$

$$Instability\ Score = \frac{0.081}{\sigma_{rand}} = 0.280$$

$$Stability\ Score = 1 - 0.280 = \mathbf{0.720}$$

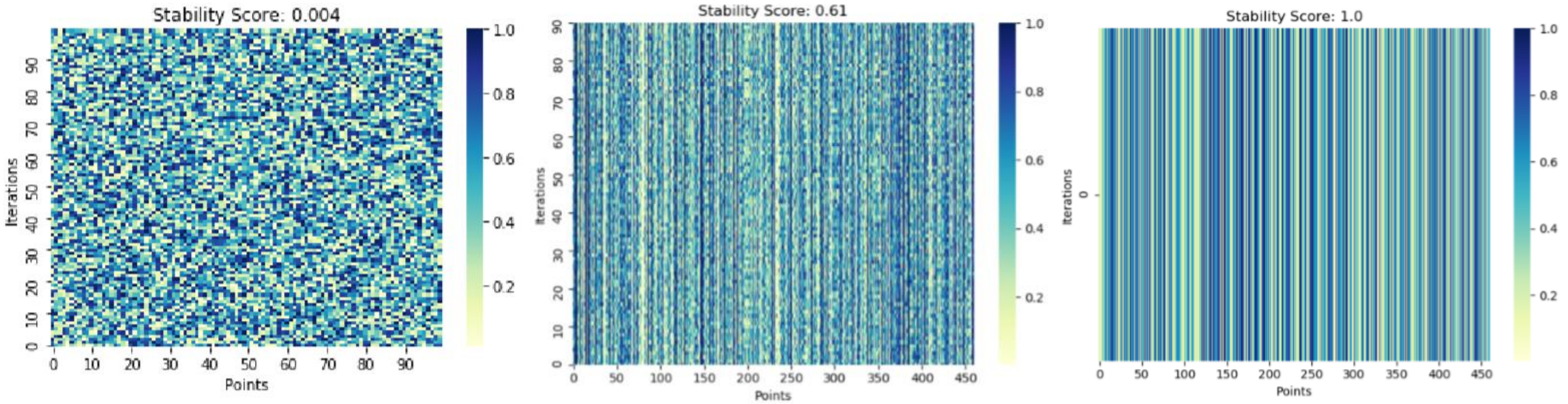
Regularization Constant

$$\sigma_{rand} = \sqrt{\frac{(t+1)(t-1)}{12t^2}}$$

where t is the number of test points. If t = 100:

$$\sigma_{rand} = 0.289$$

Heatmap - Visualization



Note: Color scale represents normalized anomaly rankings

KU LEUVEN

Stability - Results

Benchmark datasets

Table 1. Dimensionality, data size, and contamination factor of benchmark datasets

Dataset	Shuttle	Glass	WDBC	Stamps	Lympho	Iono	WBC
Dimensionality	9	7	30	9	19	32	9
Data Size	1013	214	367	340	148	351	454
Contamination	0.013	0.042	0.027	0.091	0.041	0.359	0.022

How does stability compare across different anomaly detector models?

Table 2. Comparison of stability on IF, LOF, and KNNO anomaly detection models

Model	Shuttle	Glass	WDBC	Stamps	Lympho	Iono	WBC	**Average**
IF	0.987	0.974	0.975	0.967	0.901	0.966	0.991	**0.966**
KNNO (k=20)	0.991	0.984	0.994	0.974	0.975	0.985	0.999	**0.986**
LOF (k=20)	0.944	0.941	0.987	0.905	0.973	0.951	0.995	**0.957**

Stability - Results

What is the effect of the subsample size hyperparameter on stability?

Table 4. Effect of subsample size hyperparameter on stability

Model	Subsample	Average Stability
IF	0.5 training set	**0.965**
IF	0.25-0.75 training set	**0.964**
KNNO (k=20)	0.5 training set	**0.986**
KNNO (k=20)	0.25-0.75 training set	**0.978**
LOF (k=20)	0.5 training set	**0.957**
LOF (k=20)	0.25-0.75 training set	**0.922**

What is the effect of the k hyperparameter on the stability of LOF and KNNO?

Table 5. Effect of k hyperparameter on stability in KNNO and LOF

Model	k	Average Stability
KNNO	20	**0.986**
KNNO	10	**0.972**
KNNO	3	**0.924**
LOF	20	**0.957**
LOF	10	**0.882**
LOF	3	**0.686**

Application of stability

- How can we obtain a very stable model?
- Which training points contribute the most to high stability?
- Idea: Make sample weight updates to training points based on their contribution to the model

Subsample Weight Updating

Contribution- the proportion of times that x_j takes part of the subsamples to compute the stability score

$$C_{x_j} = \frac{|\{S \subseteq G : x_j \in S\}|}{I}$$

Idea: training points with a high contribution to a high stability model are important and should have their sample weights increased. We do this by comparing different runs of stability calculation, seen here as u.

$$\Delta S_h(u) = S_h(u-1) - S_h(u-2) \quad \forall u \geq 2$$

Calculating change in stability for the model

$$\Delta C_{x_j}(u) = C_{x_j}(u-1) - C_{x_j}(u-2) \quad \forall i \geq 2$$

Calculating change in contribution for each training point

$$W_{x_j}(u) = \begin{cases} W_{x_j}(u-1) \cdot e^{\Delta C_{x_j}} & \text{if } \Delta S_h(u) \geq 0 \\ W_{x_j}(u-1) \cdot e^{-\Delta C_{x_j}} & \text{if } \Delta S_h(u) < 0 \end{cases}$$

Make Sample Weight Update

Name: UpdateSampleWeights
input : D, dataset containing outliers ;
γ, contamination factor;
h, anomaly detector model;
I, number of comparison iterations;
U, number of sample weight updates
output: W_u, Sample weights for training set at each u

Training set G, Validation set V, Test set T ;
for $1 \leq u \leq 2$ **do**
  W_u = Uniform;
  $S_h(V,u) \leftarrow Stability(V,h,\gamma,I,W_u)$;
  $S_h(T,u) \leftarrow Stability(T,h,\gamma,I,W_u)$;
  Compute $C_{x_j}(u)$ for $j = 1,2,\ldots G$ according to Eq. 9
end
for $3 \leq u \leq U$ **do**
  Compute $\Delta S_h(u)$ according to Eq. 10;
  Compute $\Delta C_{x_j}(u)$ for $j = 1,2,\ldots G$ according to Eq. 11;
  Update $W_{x_j}(u)$ for $j = 1,2,\ldots G$ according to Eq. 12;
  $S_h(V,u) \leftarrow Stability(V,h,\gamma,I,W_u)$;
  $S_h(T,u) \leftarrow Stability(T,h,\gamma,I,W_u)$;
  Compute $C_{x_j}(u)$ for $j = 1,2,\ldots G$ according to Eq. 9
end
return W_u for $u = 1,2,\ldots U$

Algorithm 2: UpdateSampleWeights

Weighted KNNO

Idea: The proposed UpdateSampleWeights algorithm gives more weight to points that contribute to high stability. What if we also gave more weight to these points when generating the anomaly scores?

For any test point $x_j \in T$ we identify its k nearest neighbors x_{jk} for k = 1, 2, … K

$$\mathcal{Z}_{x_j} = \sum_{k=1}^{K} W(x_{j_k})$$

Next, we sum the weights of a point's nearest neighbors

$$W'(x_{j_k}) = e^{W(x_{j_k})/\mathcal{Z}_{x_j}}$$

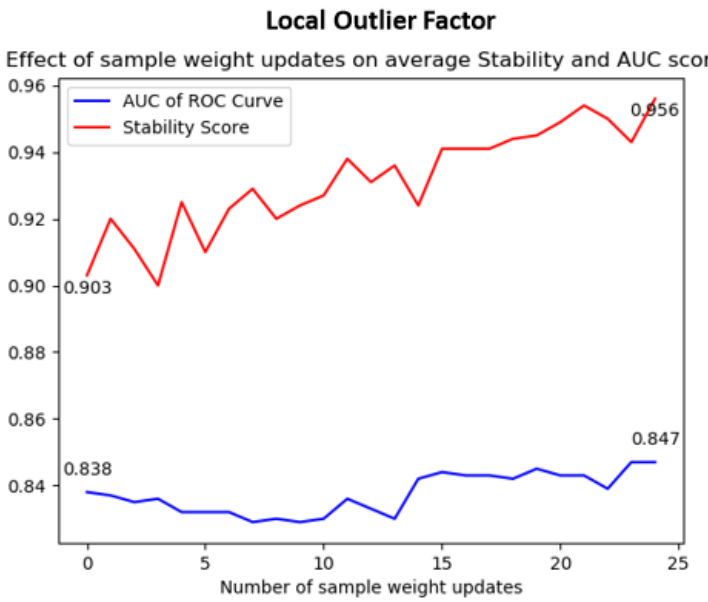
We calculate new weights in this manner to reduce the influence of very high sample weight points

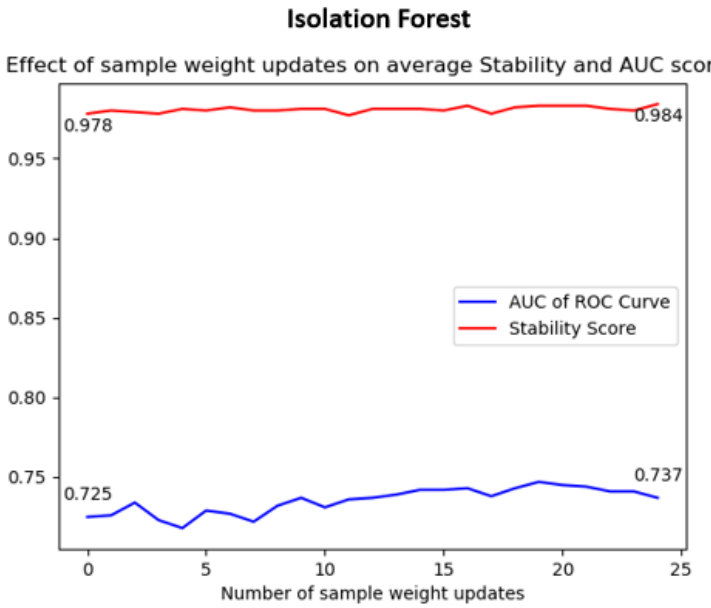
$$Score_{x_j} = \sum_{k=1}^{K} \frac{\|x_j - x_{j_k}\|_2 \times W'(x_{j_k})}{\mathcal{Z}_{x_j}}$$

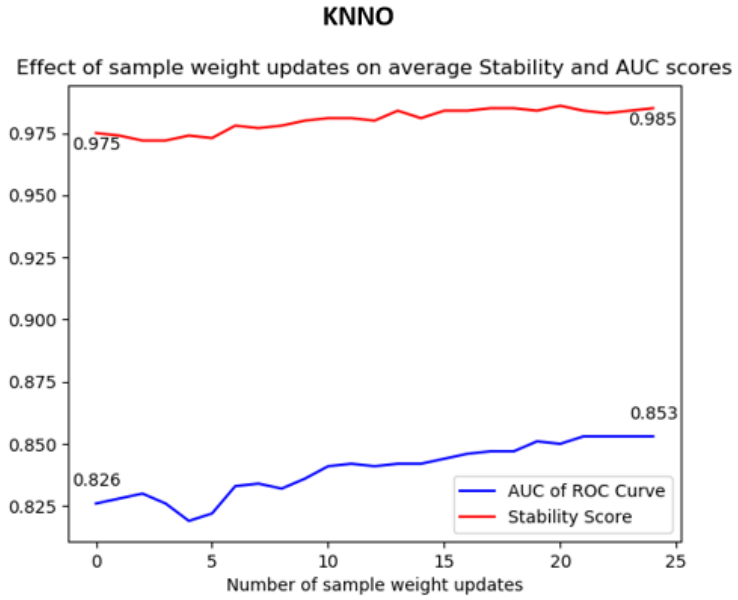
The new anomaly score is taken as the weighted average of the distance to a points nearest neighbors and their sample weights

Results- UpdateSampleWeights

Does our UpdateSampleWeights algorithm actually increase stability?





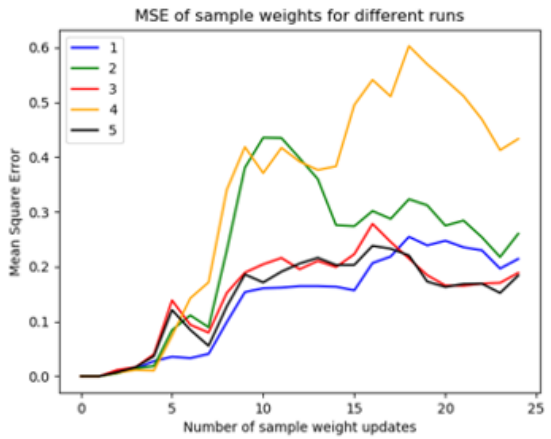


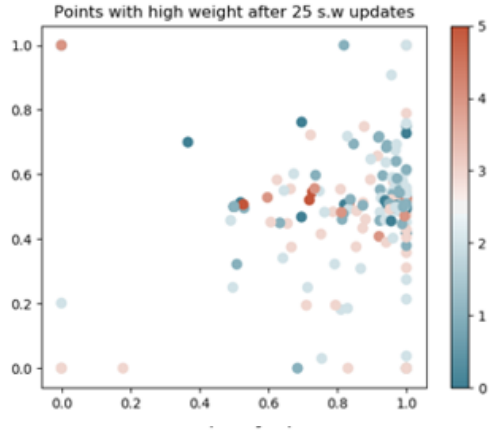
KU LEUVEN

Results- UpdateSampleWeights

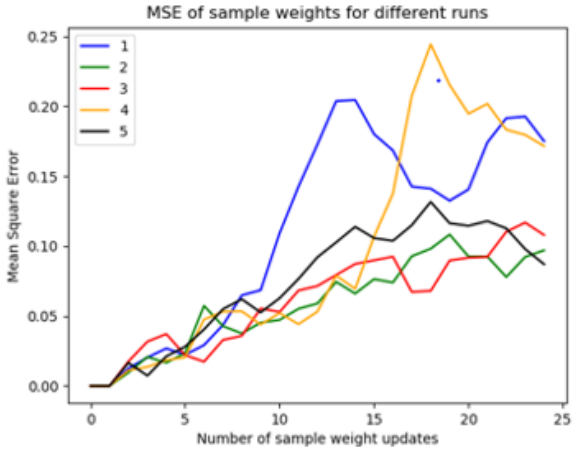
Is the algorithm consistent in which points it updates to high sample weights?

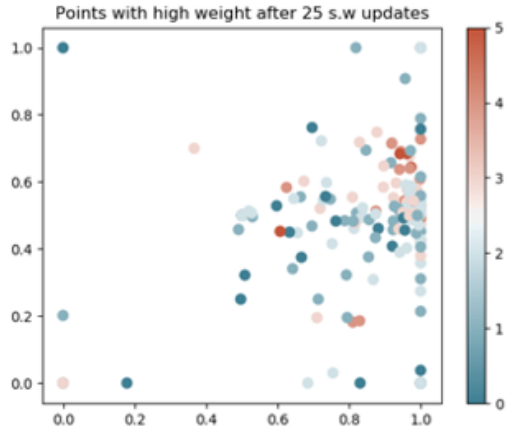
Local Outlier Factor (LOF)



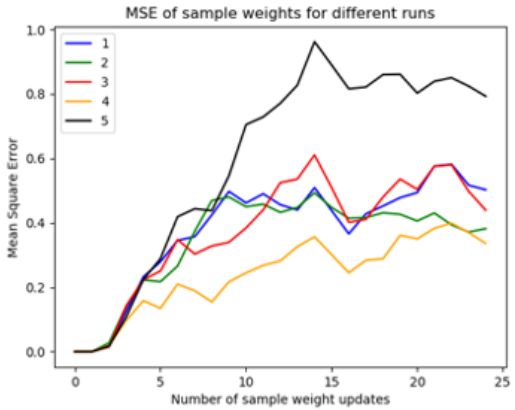


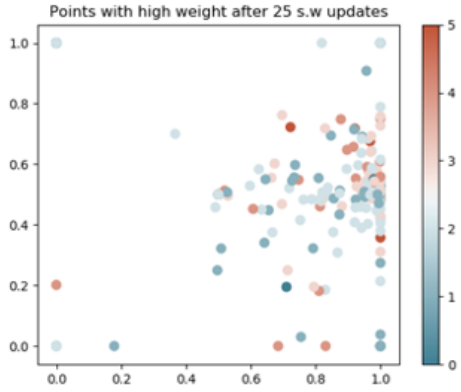
K Nearest Neighbors





Isolation Forest



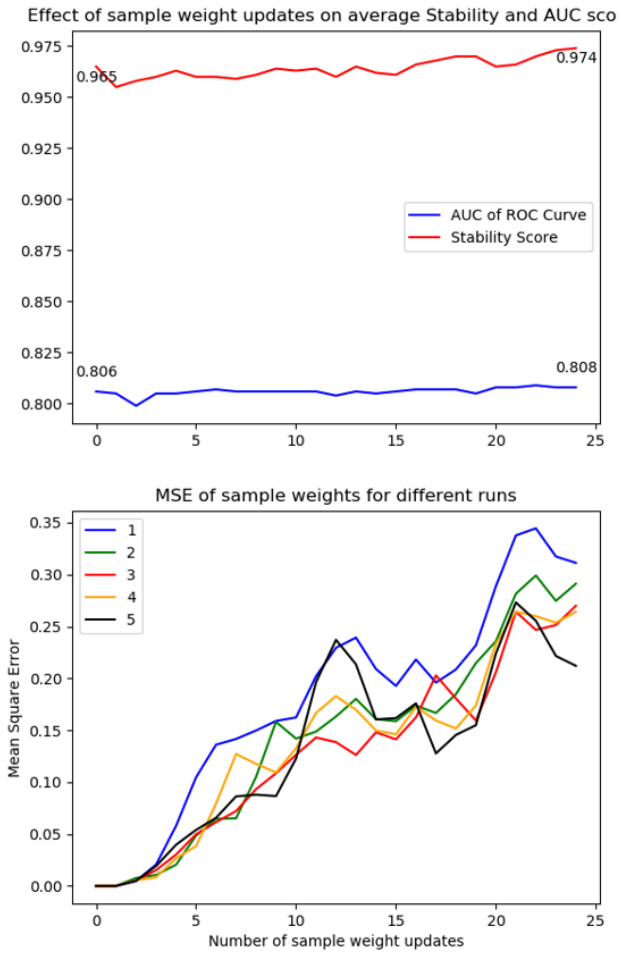


KU LEUVEN

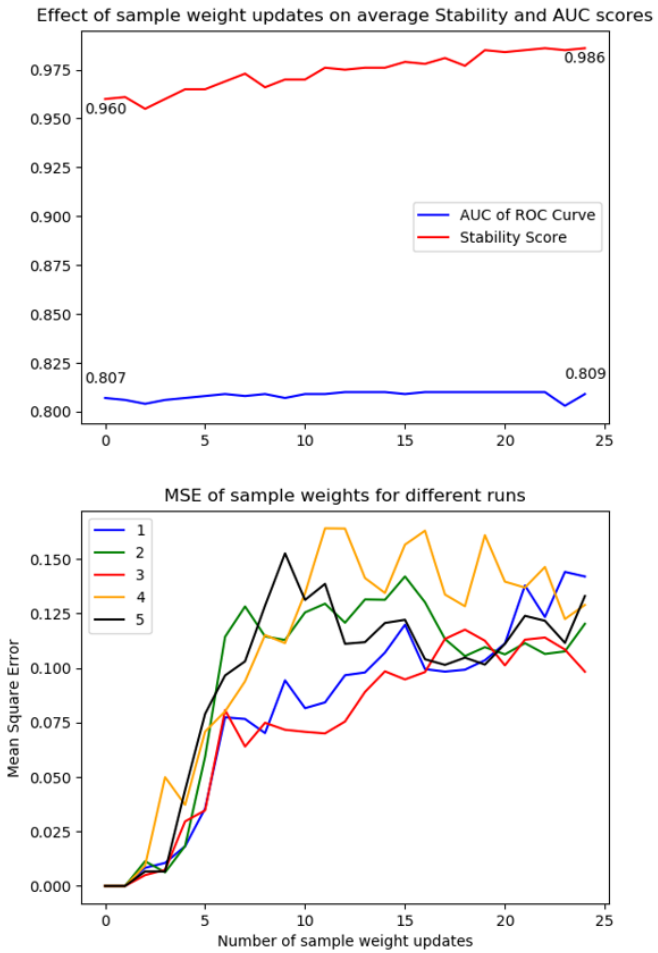
Results- UpdateSampleWeights

Is our weighted KNNO model more successful than original KNNO

Original KNN



Weighted KNN



KU LEUVEN

Conclusion/Future Work

- We have defined a stability metric scoring models from randomly generated to completely stable
- Stability of traditional anomaly detectors (KNNO, IF, LOF) on benchmarks datasets is generally much closer to perfect than random
- In addition we have developed an algorithm that increases stability by learning the most relevant training points
- Our algorithm is successful in increasing stability, however not entirely consistent in which points it converges to
- Perhaps we could get more consistent results if we framed our goal as an optimization problem, instead of using iterations to make updates.

Questions?