ANOMALY DETECTION

Lesson 8: Evaluating Anomaly Detection

Learning objectives

You will be able to:

- Evaluate different techniques for anomaly detection
- Perform anomaly detection on a wide variety of data types
- Explain other types of anomaly detection

Evaluation anomaly detection: labeled data

How well does our algorithm perform?

- For labeled data, anomaly detection is a unbalanced classification problem*
- Therefore, can use suitably modified performance metrics for classification*
  - Avoid metrics like accuracy that don’t work well for unbalanced data
- Many methods discussed provide a score of how anomalous each point is
  - Classification: we care about the *value* of the score
  - Anomaly detection: we care about the *rank* of the score

Labeled data: precision and recall

Measures of rank for anomaly detection

- It is common to use the score to *rank* how likely a point is an anomaly
- Typically, have the resources to look only at *n* points.
- Gives rise to metrics evaluated on the *n* “highest ranked” points:
  - Precision at *n* (P@n): Fraction of points in top *n* that are actually anomalies
  - Recall at *n* (R@n): Fraction of anomalies found in top *n* points
  - Average precision: Take average of P@k with k = 1,2,…n and k^{th} point is an anomaly (values of *k* corresponding to normal points are ignored)

Scoring process: precision at n (P@n)

Point #	Feature 1	Feature 2	Feature 3	Feature 4	Anomaly Score
1	10	10	8	10	-0.035985
2	10	5	10	3	-0.033510
3	10	5	5	3	-0.005384
4	10	6	6	3	0.000330

Points 1 and 3 (as ordered by score) are actual anomalies

P@1 = 1.0 (the lowest score is an anomaly)
P@2 = 0.5 (one of the two lowest scores is an anomaly)
P@3 = 0.667 (two of the three lowest scores are anomalies)
P@4 = 0.5 (two of the four lowest scores are anomalies)

Scoring process: average precision

Point #	Feature 1	Feature 2	Feature 3	Feature 4	Anomaly Score
1	10	10	8	10	-0.035985
2	10	5	10	3	-0.033510
3	10	5	5	3	-0.005384
4	10	6	6	3	0.000330

P@n for true anomalies: P@1 = 1.0; P@3 = 0.667

Average Precision is the average of these results:
AP = (1.0 + 0.667) / 2 = 0.833

Scoring process: recall at n (R@n)

Point #	Feature 1	Feature 2	Feature 3	Feature 4	Anomaly Score
1	10	10	8	10	-0.035985
2	10	5	10	3	-0.033510
3	10	5	5	3	-0.005384
4	10	6	6	3	0.000330

R@1 = 0.5	(one row finds half the anomalies)
R@2 = 0.5	(two rows finds half the anomalies)
R@3 = 1.0	(three rows finds all the anomalies)
R@4 = 1.0	(four rows finds all the anomalies)

Adjustment for chance: adjusted P@n

How much better are we doing than a random ordering?

- If there are *A* anomalies and *N* data points, a random ordering of points has and expected P@n of

$$\text{Expected}(P@n) = (\text{expected \# anomalies in top } n)/n = n(A/N)/n = A/N$$

- Adjusted P@n allows for effects of chance
  - Subtract effect of chance and compare with perfect detector

$$\text{Adjusted } P@n = \frac{P@n - \frac{A}{N}}{1 - \frac{A}{N}}$$

Adjustment for chance: adjusted average precision

Can also correct average precision for chance

- Adjusted AP measure the gain in average precision over the expected random score compares to the gain of a perfect anomaly detector

$$\text{Adjusted } AP = \frac{AP - \frac{A}{N}}{1 - \frac{A}{N}}$$

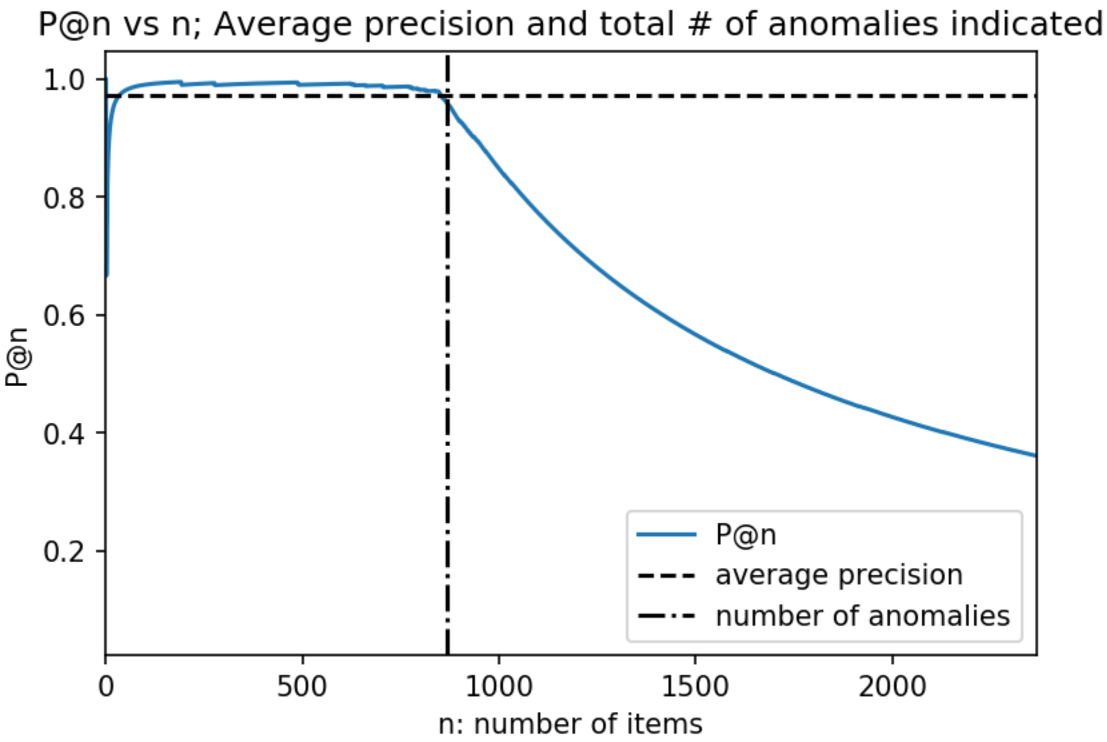
- For a good anomaly detector, the adjusted AP > 0
  - For a random ordering AP = 0
  - Orderings worse than average have AP <0

P@n and Adjusted P@n

When to use

- **Not needed**: if looking at different detection methods with the *same* metric on the *same* dataset
- **Must be used**: if looking at different datasets with different proportions of anomalies
- **In all cases**: adjusting for chances helps interpretability

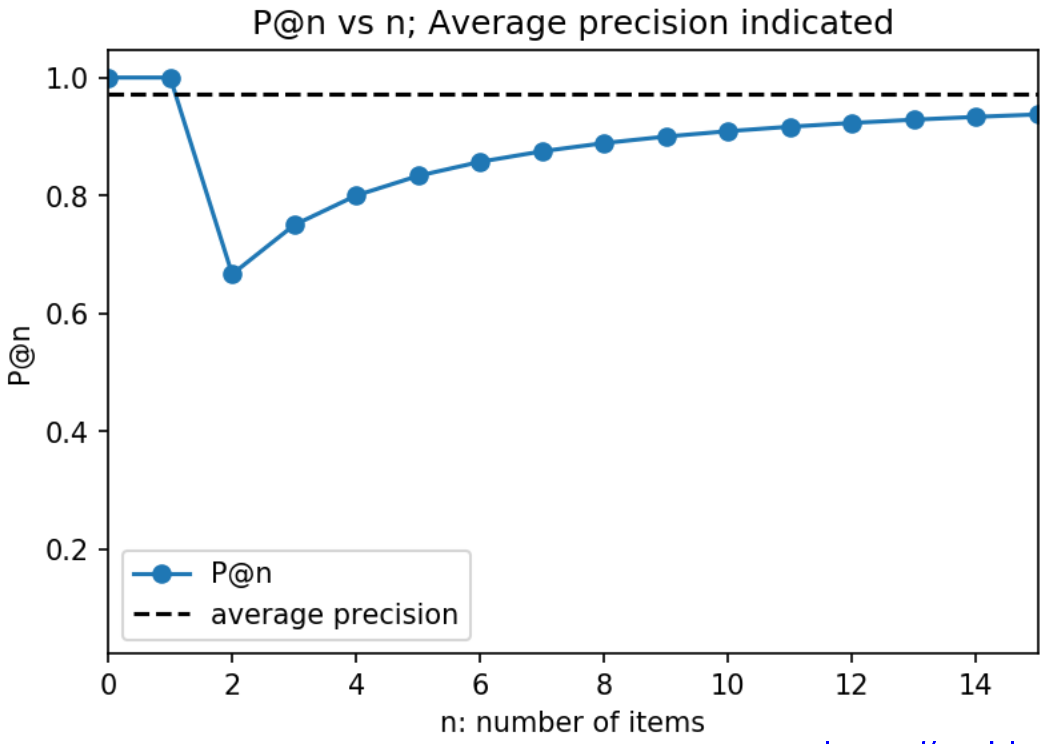
Example: Statlog Shuttle sensor dataset



- 12345 readings, each on 8 sensors
- Trying to detect abnormal operating modes
- 867 readings are in abnormal modes (i.e. anomalies)
- Isolation forest with 100 estimators used for anomaly detection.

https://archive.ics.uci.edu/ml/datasets/Statlog+%28Shuttle%29

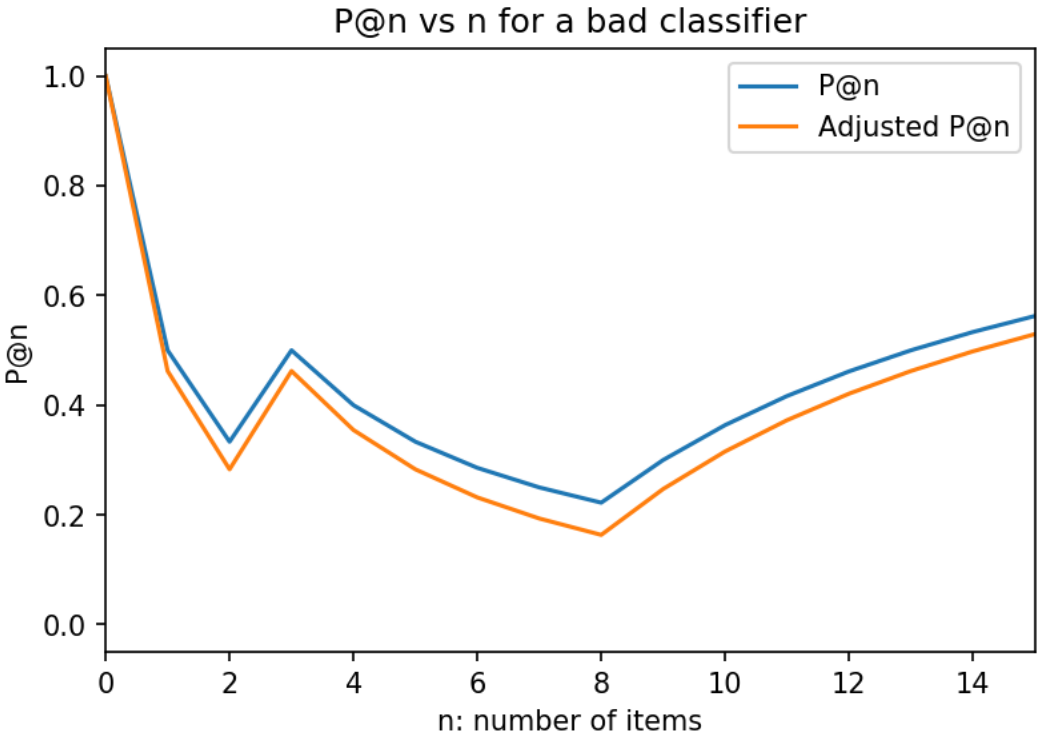
Example: Statlog Shuttle sensor data set – small n



- 12345 readings, each on 8 sensors
- Trying to detect abnormal operating modes
- 867 readings are in abnormal modes (i.e. anomalies)
- Isolation forest with 100 estimators used for anomaly detection.

https://archive.ics.uci.edu/ml/datasets/Statlog+%28Shuttle%29

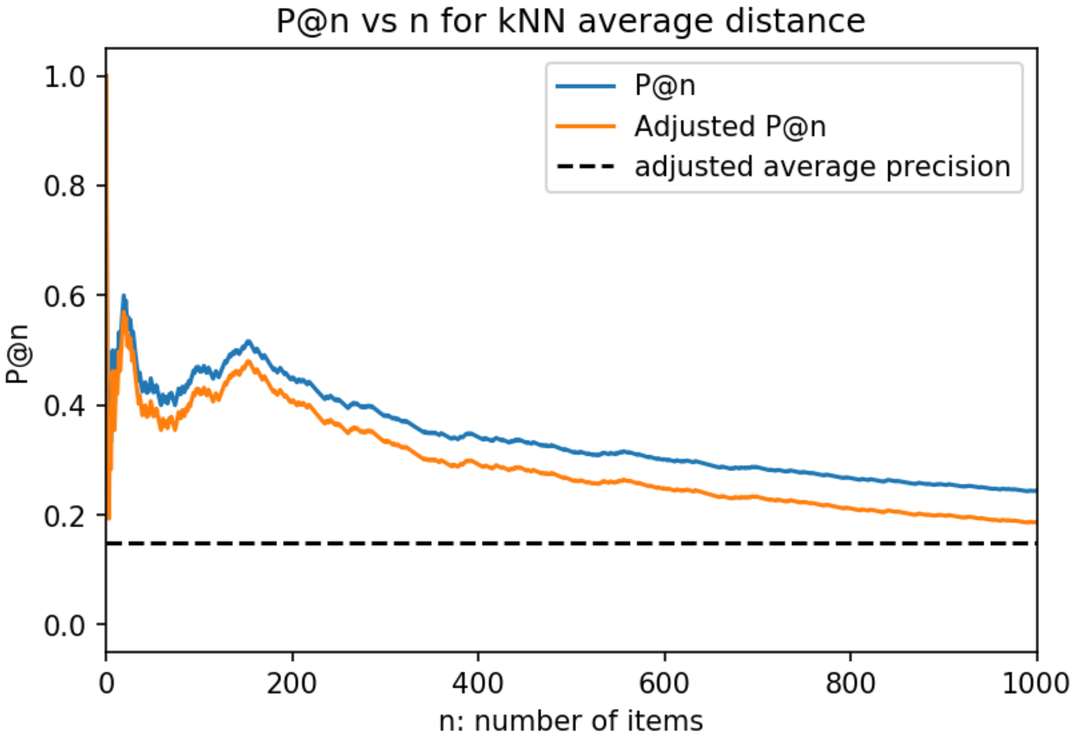
Example: small n, bad anomaly detector



- 12345 readings, each on 8 sensors
- Trying to detect abnormal operating modes
- 867 readings are in abnormal modes (i.e. anomalies)
- Isolation forest with **3** estimators used for anomaly detection.

https://archive.ics.uci.edu/ml/datasets/Statlog+%28Shuttle%29

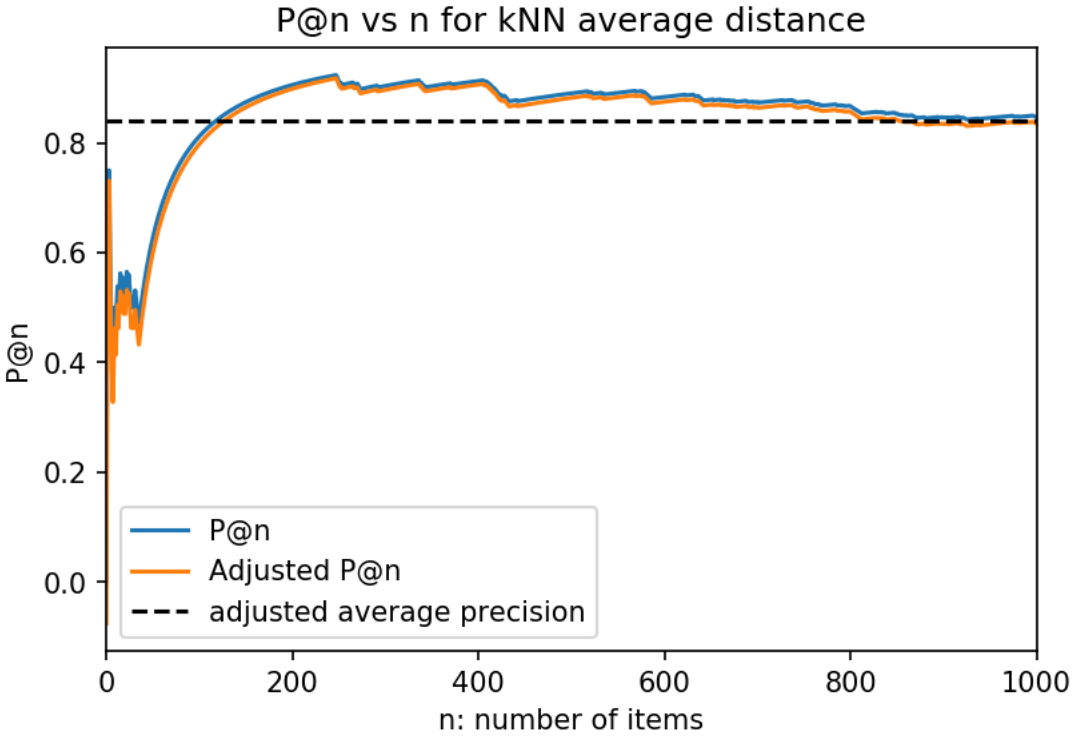
Example: same data, different classifier



- Using a KNN classifier instead with k = 20
- Parameter k has been tuned
- Note: preprocessing skipped, so poor results expected

https://archive.ics.uci.edu/ml/datasets/Statlog+%28Shuttle%29

Example: same data, different classifier



- Used best practices
- Scaled features, used PCA
- Tuned parameter k (600 NN)
- Same algorithm, same data, *much* better results

https://archive.ics.uci.edu/ml/datasets/Statlog+%28Shuttle%29

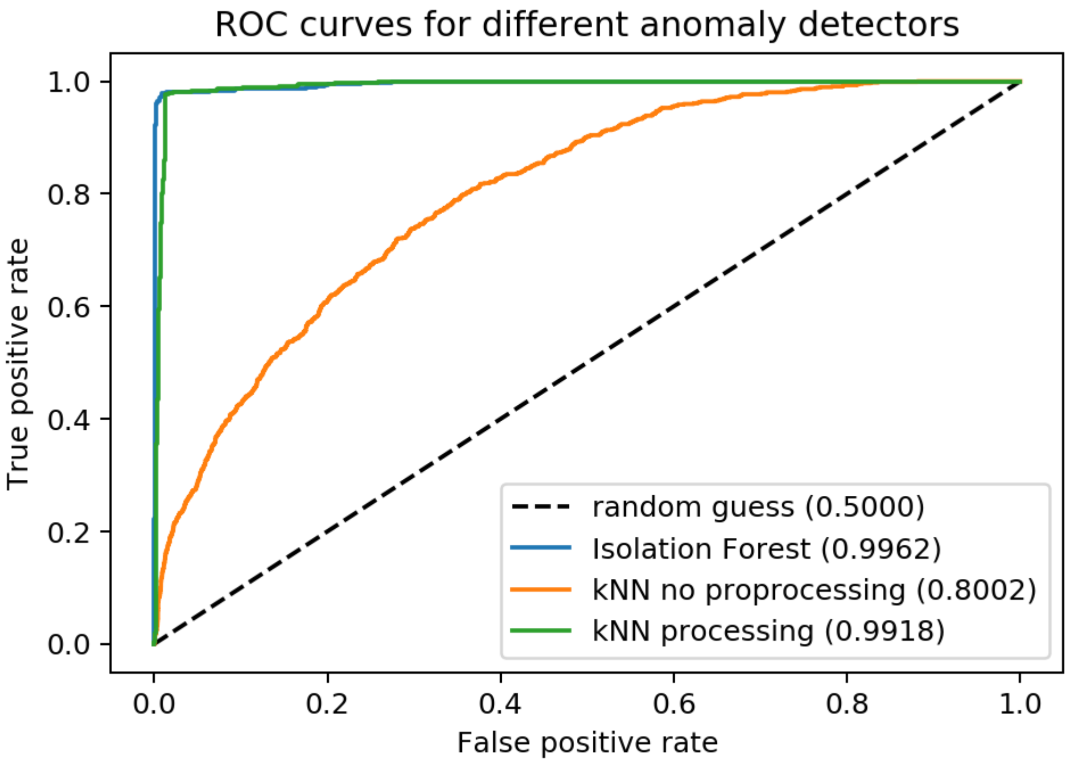
Labeled data: receiver operator characteristic score

We can also use the *Receiver Operator Characteristic* (ROC) curve to evaluate anomaly detectors

- Very common technique in classification problems
- Not sensitive to class imbalance
- Ranks the false negative rate (horizontal axis) against the true positive rate (vertical axis)
- Area under the curve (AUC) is a single number that tells us how well the detector does separating cases:

  P (randomly chosen anomaly score > randomly chosen normal score) = ROC AUC

Example: Statlog Shuttle data ROC Curve



Labeled data: summary

In a *classification problem*, the score + threshold is used to determine the class

- Precision@n (P@n): Fraction of points in the top *n* that are actually anomalies
  - Commonly used, “top heavy” (i.e. emphasizes the highest ranked point)
  - Caveat: can be sensitive to *n*
- Average Precision: Average of P@n evaluated at outlier positions
  - Commonly used, not sensitive to external parameter *n*, top heavy
- ROC AUC: Probability of a random anomaly scoring higher than random normal point
  - Commonly used, no external parameters, doesn’t require thresholds
  - Caveat: Doesn’t reward finding anomalies “early” (low *n*); score can be made up at the tail end

Unlabeled/unsupervised anomaly detection

Methods described so far don’t work when we don’t have ground truth labels

- In clustering analysis, there are internal measures of cluster “goodness” (e.g. the average distance of a point to cluster centers)
- The types of internal measure used can bias the result toward certain types of clusters
  - For example, k-means typically scores better than density-based methods if the metric is the average distance of a point to cluster centers
  - Use of internal measures in anomaly detection are generally not used; they introduce too much bias into the method

Unlabeled/unsupervised anomaly detection

Approach:

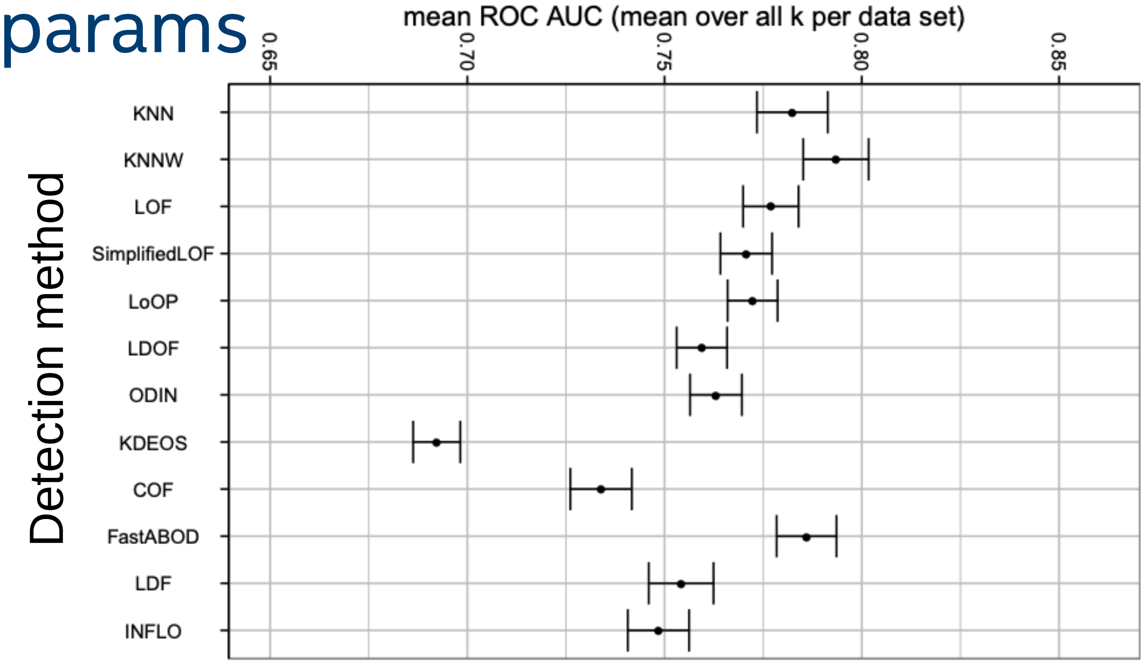
- Find a classification dataset with similar characteristics / separation
  or
  Generate labeled data with similar distributions as the expected anomalies
- Treat one of the classes as the anomaly class. Down-sample if necessary to get a reasonable normal:anomaly ratio
- Benchmark model on classification dataset
- Investigate anomalies flagged in actual data to see if model generalizes well
- Extra: try mislabeling a few of the "close" anomalies, to see what effect mislabeling has on your dataset

Unlabeled/unsupervised anomaly detection

Even though we have generated labels / used a classification problem:

- No train/test split or cross-validation
  - Doesn’t make sense to train model parameters on a different dataset
  - If we could train directly on dataset, we would use supervised methods
- Model parameters (e.g. number of neighbors) determined by evaluating metric over a “reasonable” range of parameters
- Model evaluation is given as a box-plot of the outputs on the classification problem after running over the reasonable range of parameters
- Reasonable ranges generally require subject matter expertise

Example: ranges of values while scanning hyperparams



On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study by G. O. Campos et al. (http://doi.org/10.1007/s10618-015-0444-8)

Categorical models

Datasets with discrete, unordered values

- So far we have focused on datasets with numerical data
- Many of the techniques presented can be applied to categorical data as well
- The challenges is to construct a meaningful distance function that can be used to analyze the data
  - For categorical data, distance function is often called a “similarity function”

Categorical models: similarity functions

Some examples

- Similarity = 1 if categories are the same, 0 otherwise
- Similarity = 1 if categories are the same. Otherwise

$$sim(c_1, c_2) = 1/[1 + \log(n_1)\log(n_2)]$$

  - Here n_j is the number of elements in category j (with j =1, 2)

- Similarity between two categories is $[\log(n/N)]^2$ if they are the same, 0 otherwise
  - Here n = number in category, N = number of data points

Similarities to distances

- If we have *d* numeric features and *c* categorical features

$$sim(x,y) = \sum_{i=1}^{d} x_i y_i + \sum_{i=1}^{c} sim(x_{c_i}, y_{c_i})$$

- Could use cosine distance (but we lose separation in similarity variable)

$$dist(x,y) = \cos^{-1} \frac{sim(x,y)}{\sqrt{sim(x,x) sim(y,y)}}$$

- Could combine numeric distance and similarity score:

$$squared\ dist(x,y) = \sum_{i=1}^{d} (x_i - y_i)^2 + \sum_{i=1}^{c} (1 - sim(x_{c_i}, y_{c_i}))$$

Other types of anomaly detection

Three examples

- Generative models
  - Find the probability distribution to describe the feature space and use this to detect anomalies
- Information-theoretic models
  - Anomalies increase amount of information needed to summarize the data
- Frequent pattern mining
  - If an instance of the data contains patterns found frequently, it is unlikely to be an anomaly

CONCLUSION

Use Python* for anomaly detection

Next up is a look at applying these concepts in Python*

- See notebook entitled *Evaluating_Anomaly_Detection_student.ipynb*

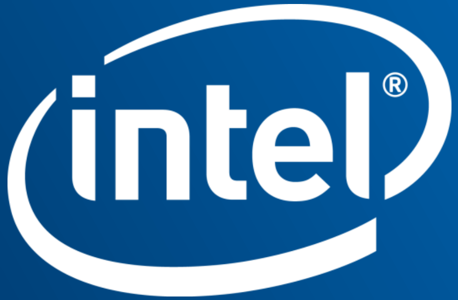
Learning objectives recap

In this session you learned how to:

- Evaluate different techniques for anomaly detection
- Perform anomaly detection on a wide variety of data types
- Explain other types of anomaly detection

References

- *Outlier Analysis* by C.C. Aggarwal (Springer 2013)
  - First chapter available free
- *On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study* by G. O. Campos et al. (Data Mining and Knowledge Discovery 2016)
  - http://doi.org/10.1007/s10618-015-0444-8
  - https://imada.sdu.dk/~zimek/InvitedTalks/TUVienna-2016-05-18-outlier-evaluation.pdf (freely available)


intel®