Anomaly Detection

Tom Slooff
Università della Svizzera italiana

Contents

Introduction

- Which majors is everybody in?
- Who is familiar with Python?
- Who is familiar with Machine Learning?
- Who is familiar with Anomaly Detection?

Introduction

- An **Anomaly:** *Deviation or departure from the normal or common order, form, or rule.*

In a security context, security incidents are often anomalies:

- Network intrusion detection
- Physical access monitoring (keycards)
- Program behaviour: malware
- Input validation against adversarial machine learning

Security Monitoring has large overlap with Anomaly Detection

Introduction

You get computer network data. The task: train / create a model to detect attacks (e.g. port scan).
How do you do it?

Introduction

Important in AD:

- Tainted dataset
- Untainted dataset

Broadly two types of learning

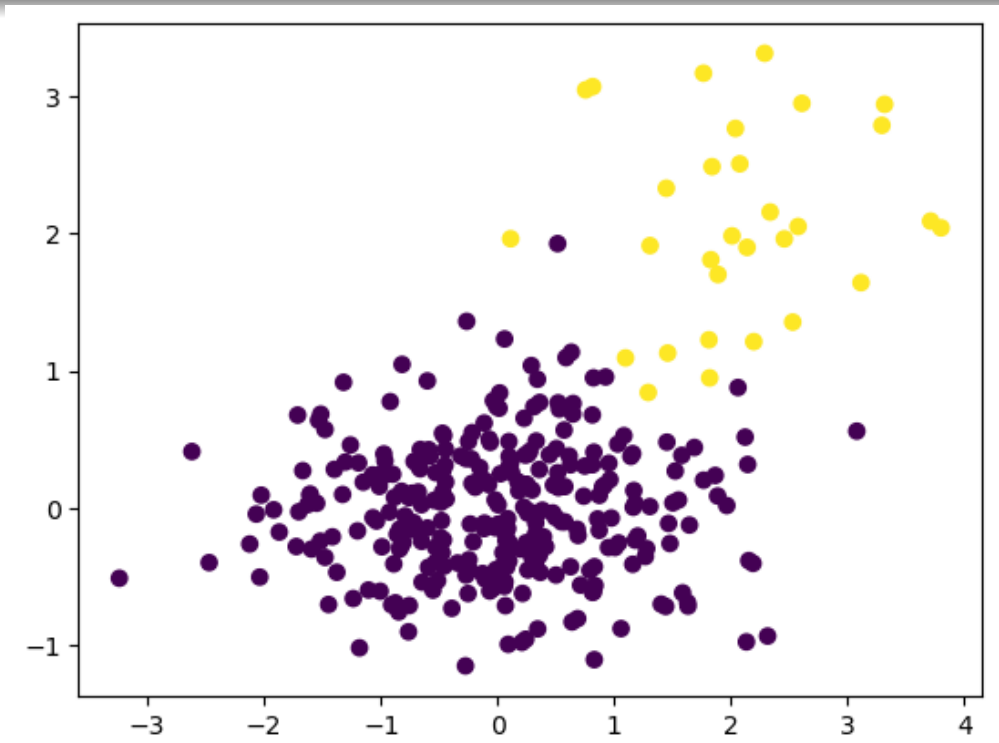
- Supervised / two class learning
- Unsupervised / one class learning

ML vs. AD

How is AD different from ML?

- AD: asymmetry. There is a reference / normal behaviour and there is data which falls outside of it.
- ML: imbalance. While one class may be severely overrepresented, the classes may overlap in some concept.

ML vs. AD



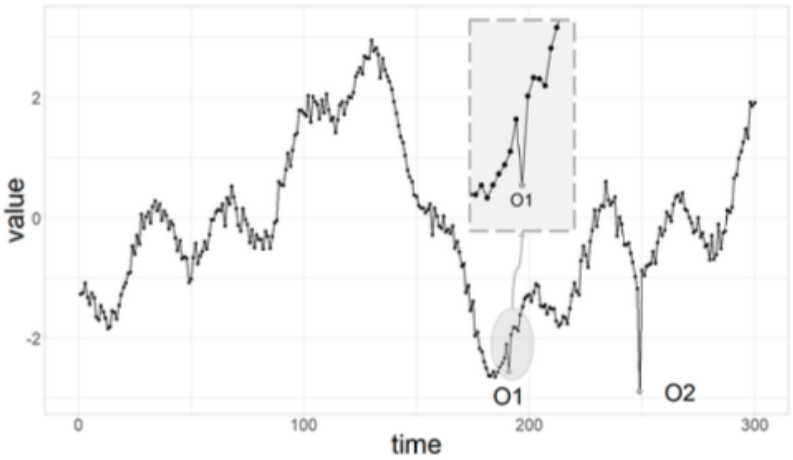
- ML model may find these clusters.
- You should not rely on an AD model to do so:
  - The underrepresented class is somewhat close to the overrepresented class
  - There is a clear structure to the underrepresented class so they are not anomalous
- And vice versa

Types of anomalies

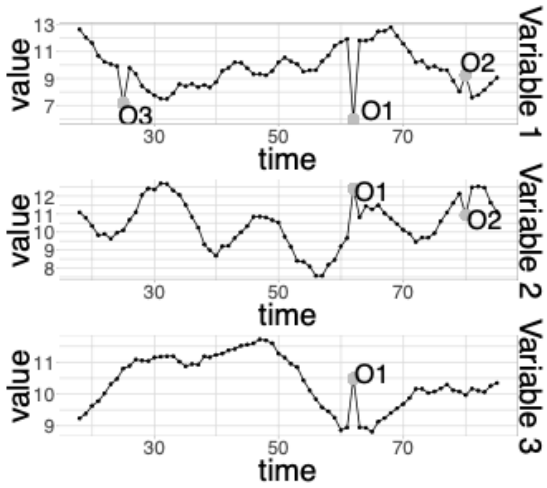
In temporal data, there are several different types of anomalies.

Types of anomalies

Point Anomalies



(a) Univariate time series.

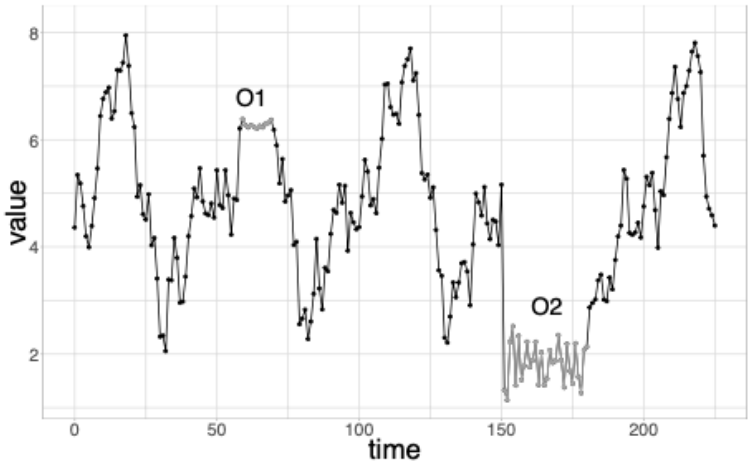


(b) Multivariate time series. [1]

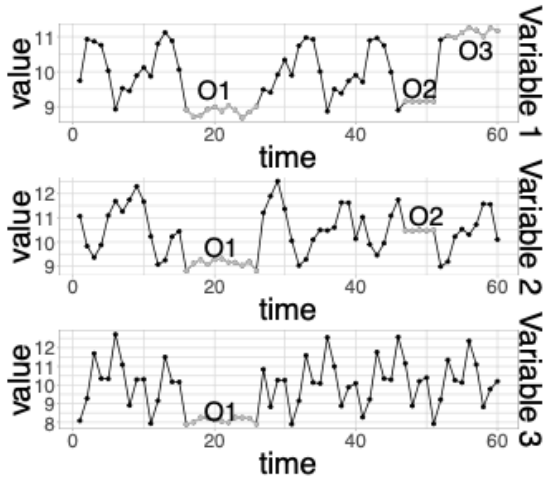
[1] From: Blázquez-García, A., Conde, A., Mori, U., & Lozano, J. A. (2021). A review on outlier/anomaly detection in time series data. ACM Computing Surveys (CSUR), 54(3), 1-33.

Types of anomalies

Sub-stream Anomalies



(a) Univariate time series.



(b) Multivariate time series.

2

[2] From: Blázquez-García, A., Conde, A., Mori, U., & Lozano, J. A. (2021). A review on outlier/anomaly detection in time series data. ACM Computing Surveys (CSUR), 54(3), 1-33.

Types of anomalies

Whole-stream Anomalies

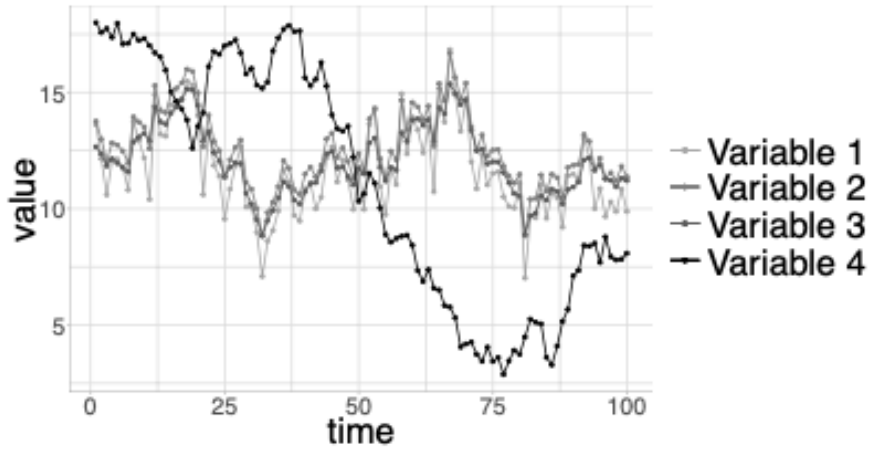


Fig. 5. Outlier time series (*Variable 4*) in a multivariate time series. [3]

[3] From: Blázquez-García, A., Conde, A., Mori, U., & Lozano, J. A. (2021). A review on outlier/anomaly detection in time series data. ACM Computing Surveys (CSUR), 54(3), 1-33.

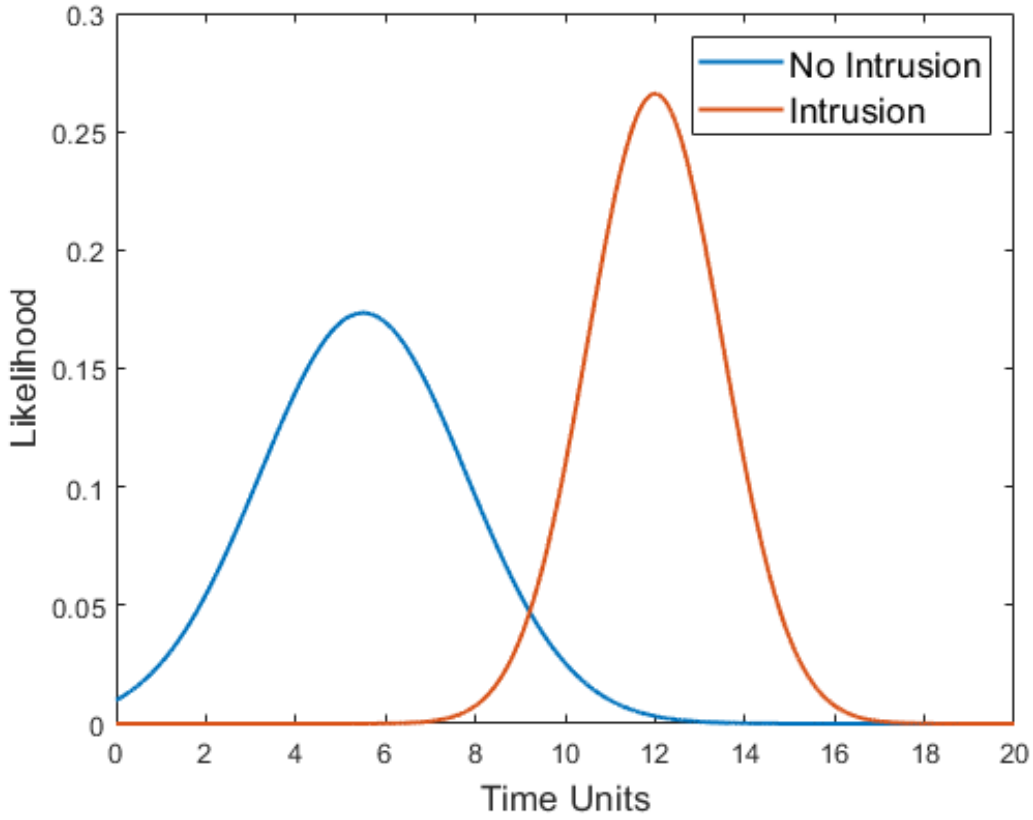
Contents

Evaluation

Suppose we have a very simple dataset which contains the connection durations of users to my website. The dataset is supervised: I know which users were maliciously connecting and which were not. I build a very simple model: two gaussian distributions. One for the normal users and one for the malicious users.

Evaluation

In the end we have something like this



Where do we put the threshold?

Evaluation

		Predicted condition	
	Total population = P + N	Positive (PP)	Negative (PN)
Actual condition	Positive (P)	True positive (TP)	False negative (FN)
	Negative (N)	False positive (FP)	True negative (TN)

Figure: Confusion Matrix. Taken from wikipedia.

Which of these is most impactful in the anomaly detection setting?

Evaluation

		Predicted condition	
	Total population = P + N	Positive (PP)	Negative (PN)
Actual condition	Positive (P)	True positive (TP)	False negative (FN)
	Negative (N)	False positive (FP)	True negative (TN)

Figure: Confusion Matrix. Taken from wikipedia.

True Positive Rate (TPR): $\frac{TP}{TP+FN}$ i.e. how many of the positives do we catch?
False Positive Rate (FPR): $\frac{FP}{FP+TN}$ i.e. how many of the negatives do we misclassify?
Precision / Positive Predictive Value (PPV): $\frac{TP}{TP+FP}$ i.e. what is the probability of a positive, given that we classify it as positive?
F1-score is the harmonic mean of PPV and TPR: $2 * \frac{PPV*TPR}{PPV+TPR}$

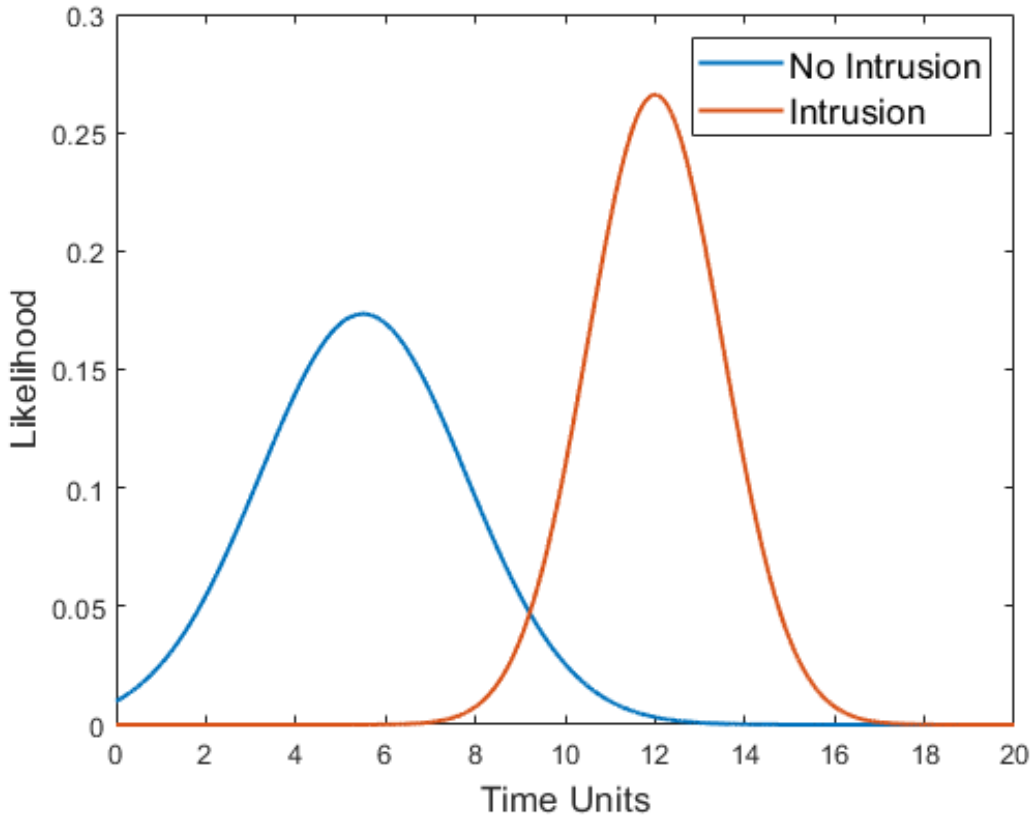
Evaluation

		Predicted condition	
	Total population = P + N	Positive (PP)	Negative (PN)
Actual condition	Positive (P)	True positive (TP)	False negative (FN)
	Negative (N)	False positive (FP)	True negative (TN)

Figure: Confusion Matrix. Taken from wikipedia.

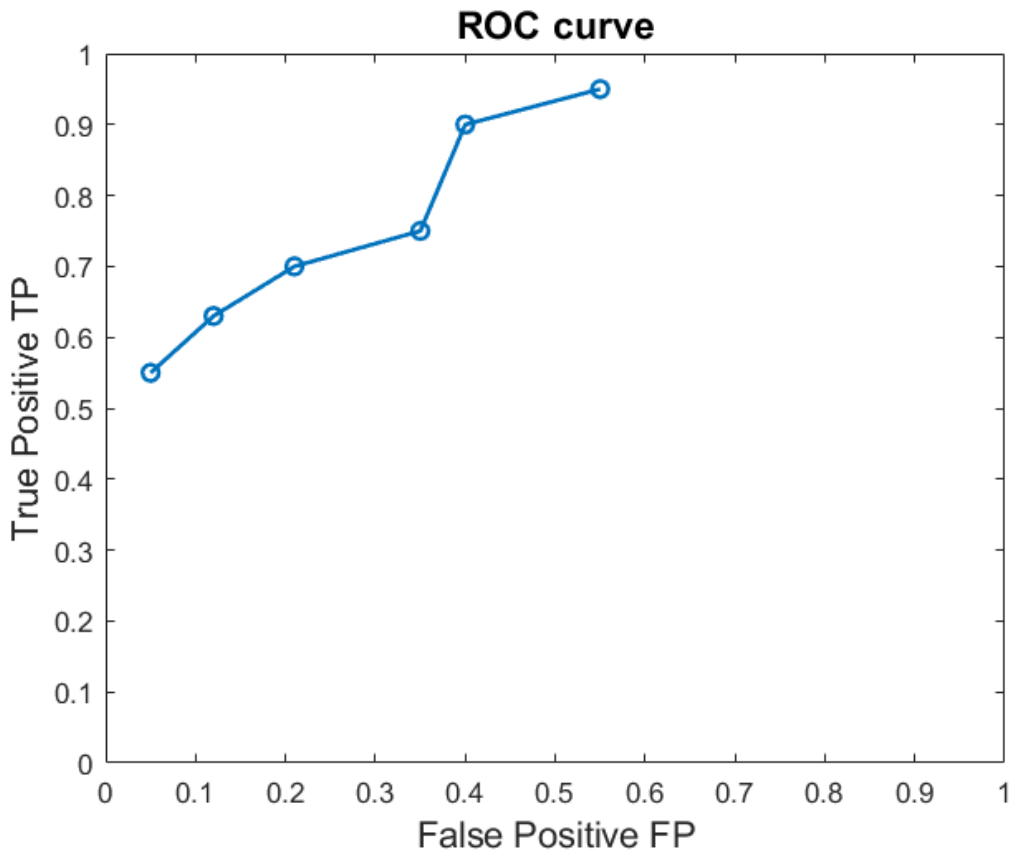
Why don't we use accuracy?
Base Rate Fallacy

ROC Curve



As we shift the threshold, our metrics change as well.
To capture this we have the Receiver Operating Characteristic (ROC) curve.

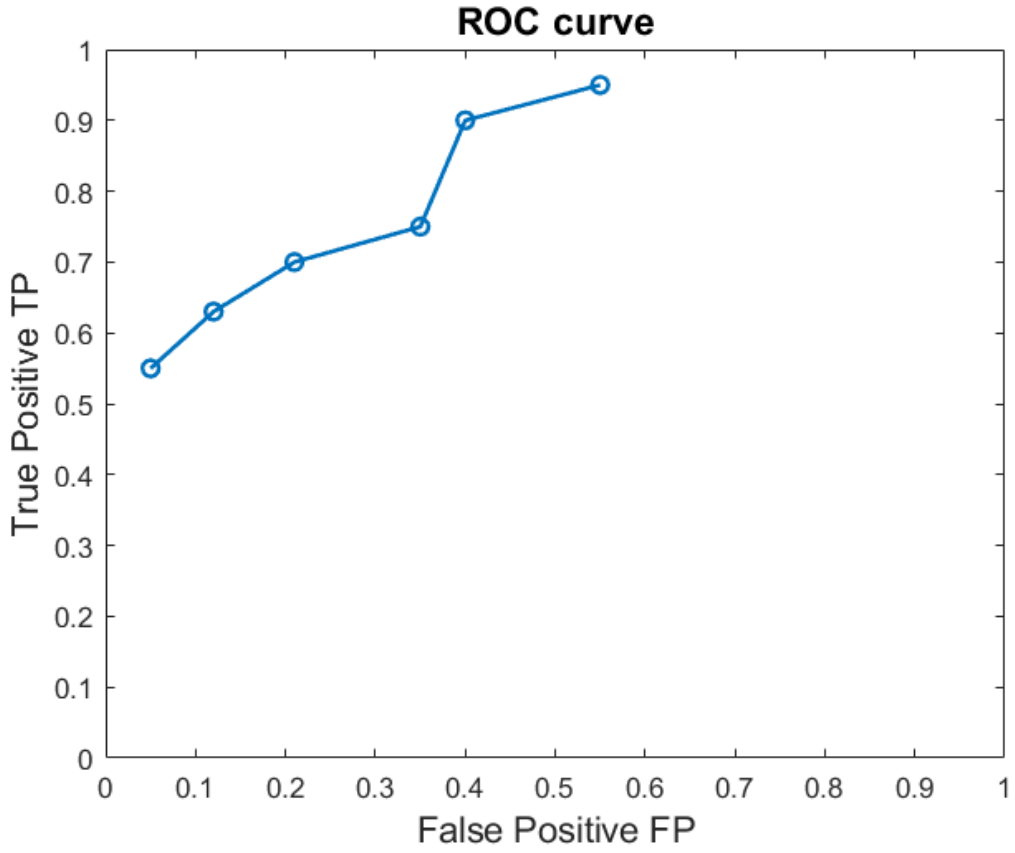
ROC Curve



The ROC curve plots the True Positive Rate against the False Positive Rate as we shift the threshold.

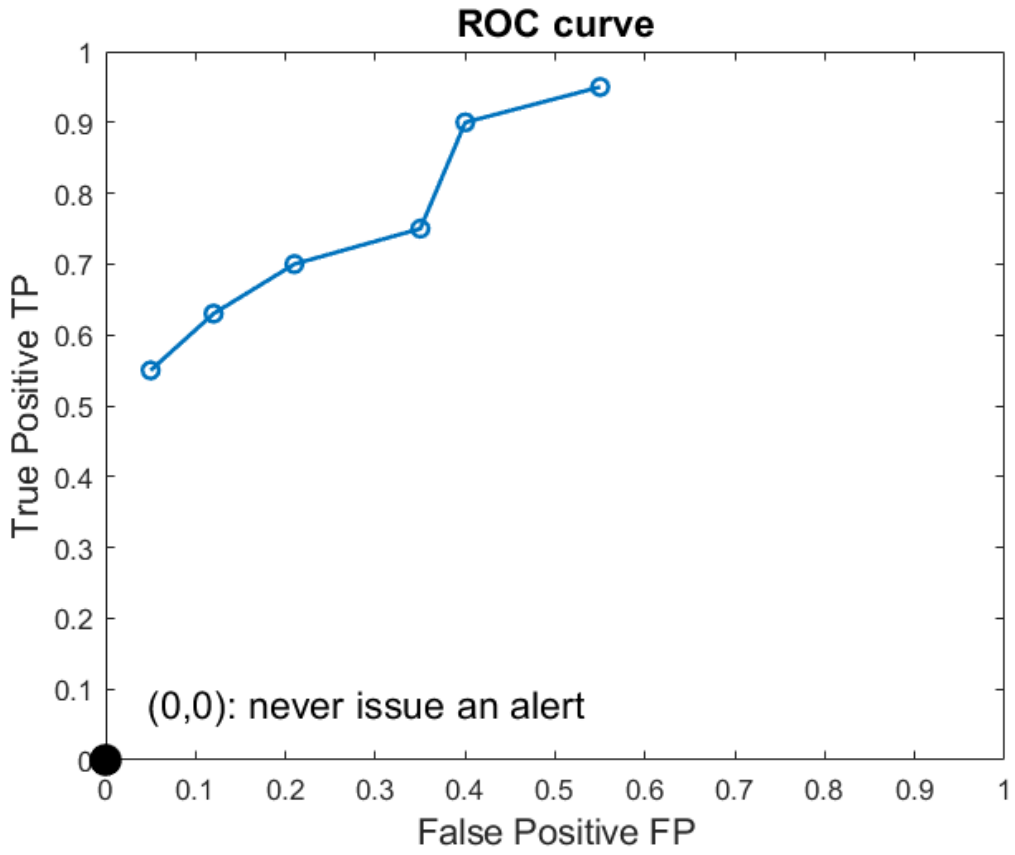
ROC Curve

Question: where would a model be which never predicts an anomaly?



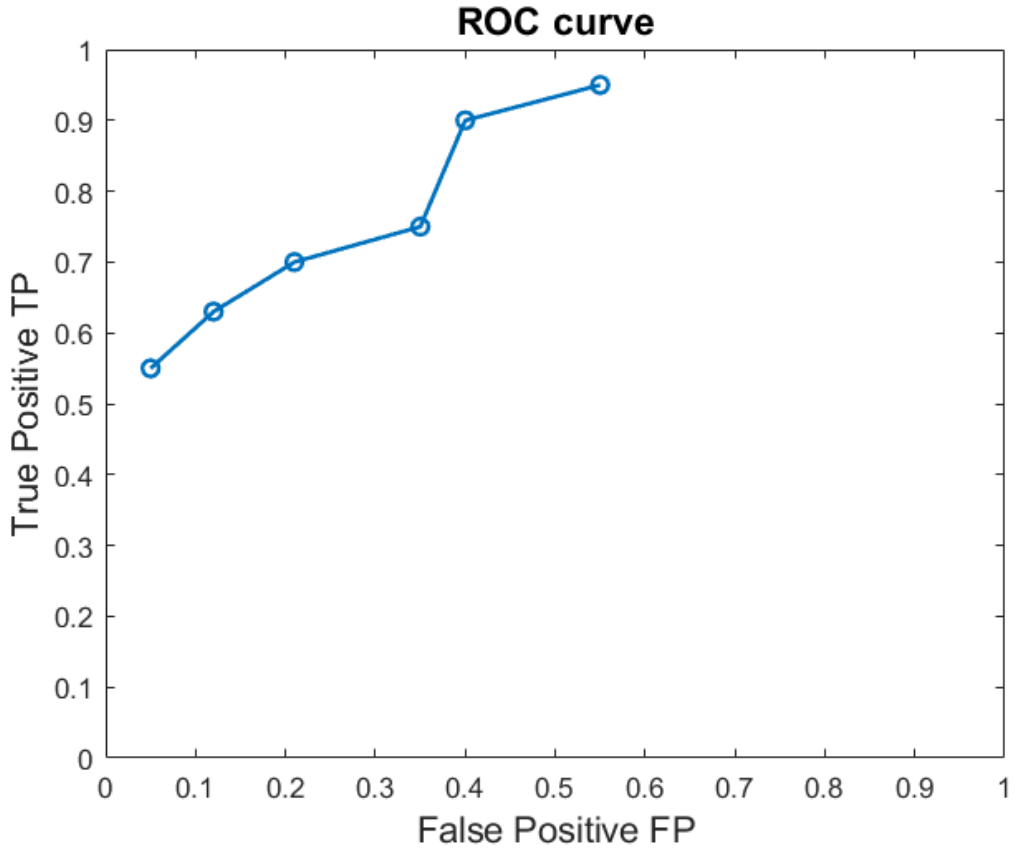
ROC Curve

Question: where would a model be which never predicts an anomaly?



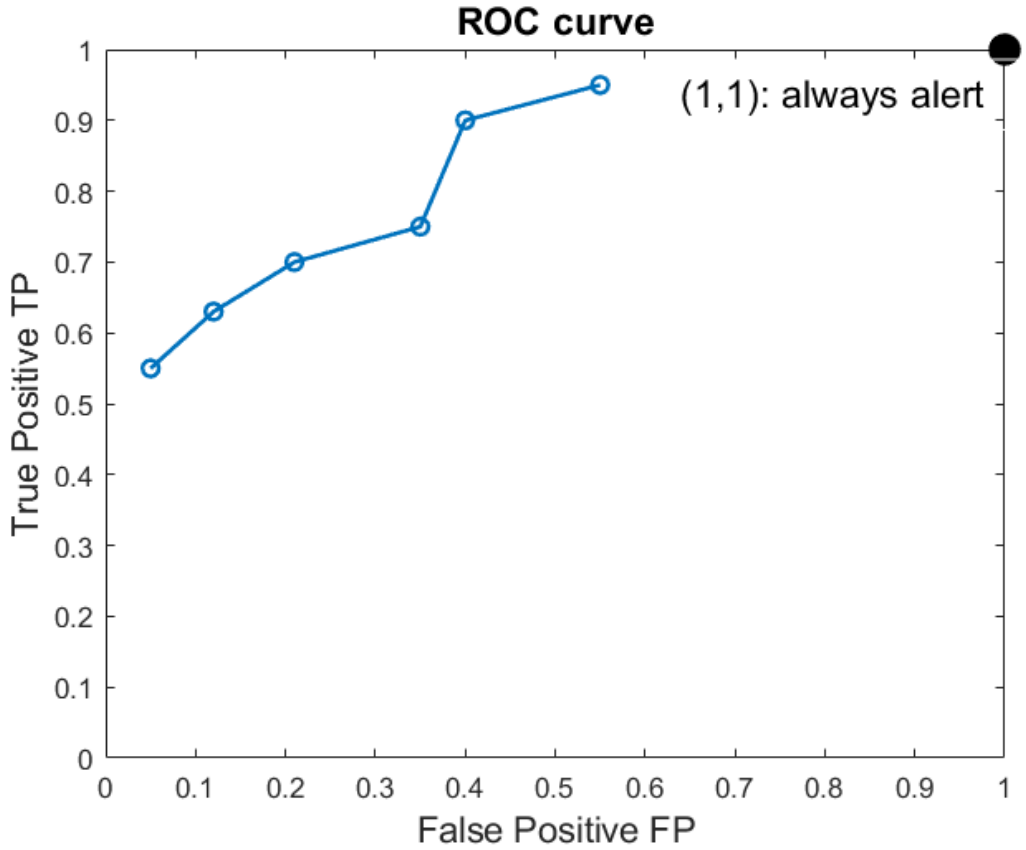
ROC Curve

Question: where would a model be which always predicts an anomaly?



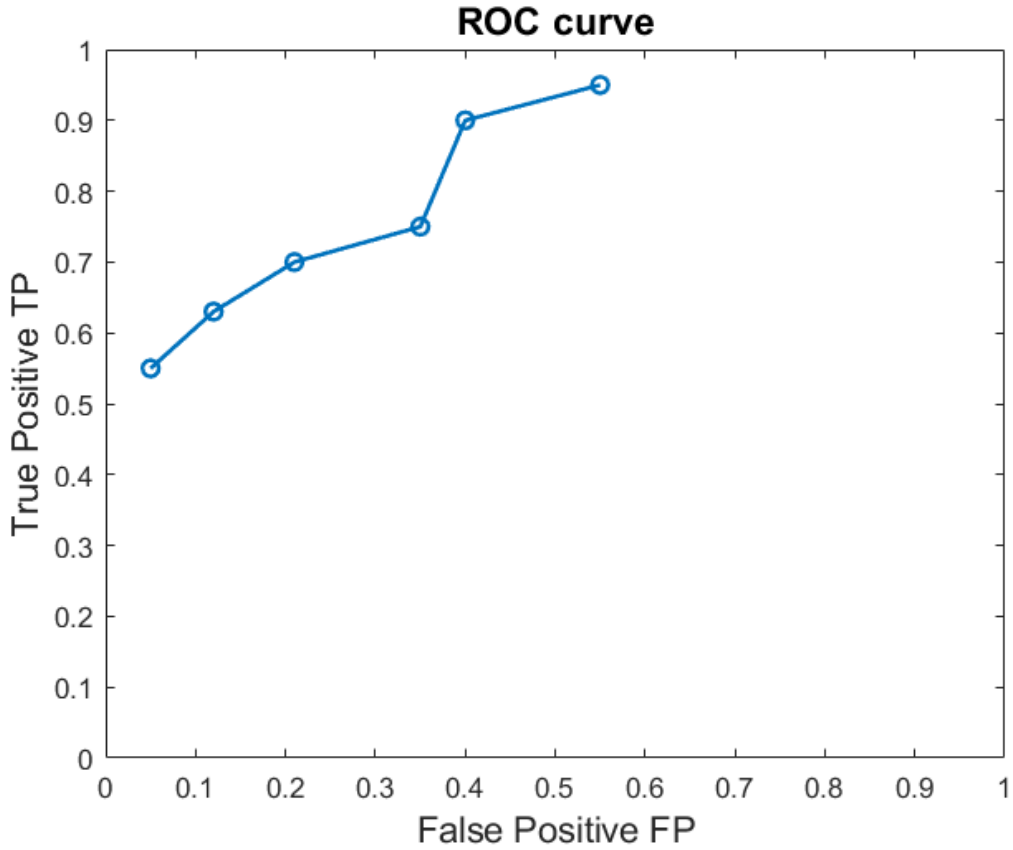
ROC Curve

Question: where would a model be which always predicts an anomaly?



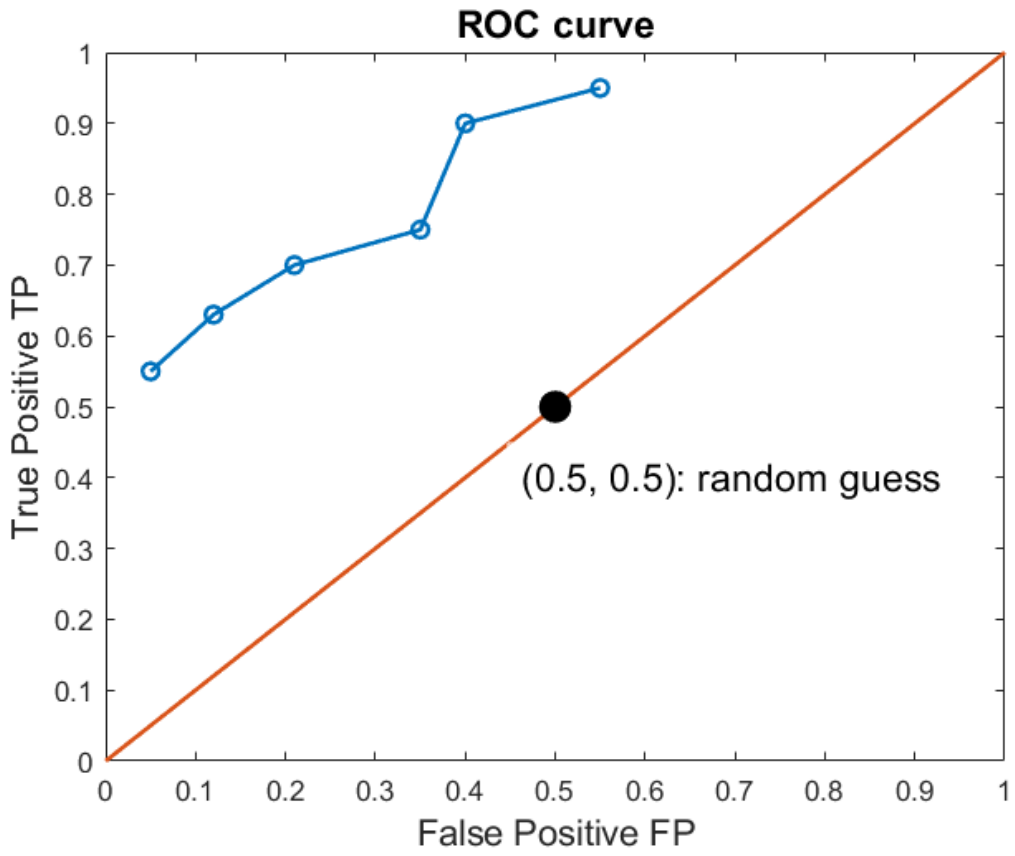
ROC Curve

Question: where would a model be which randomly guesses anomalies?



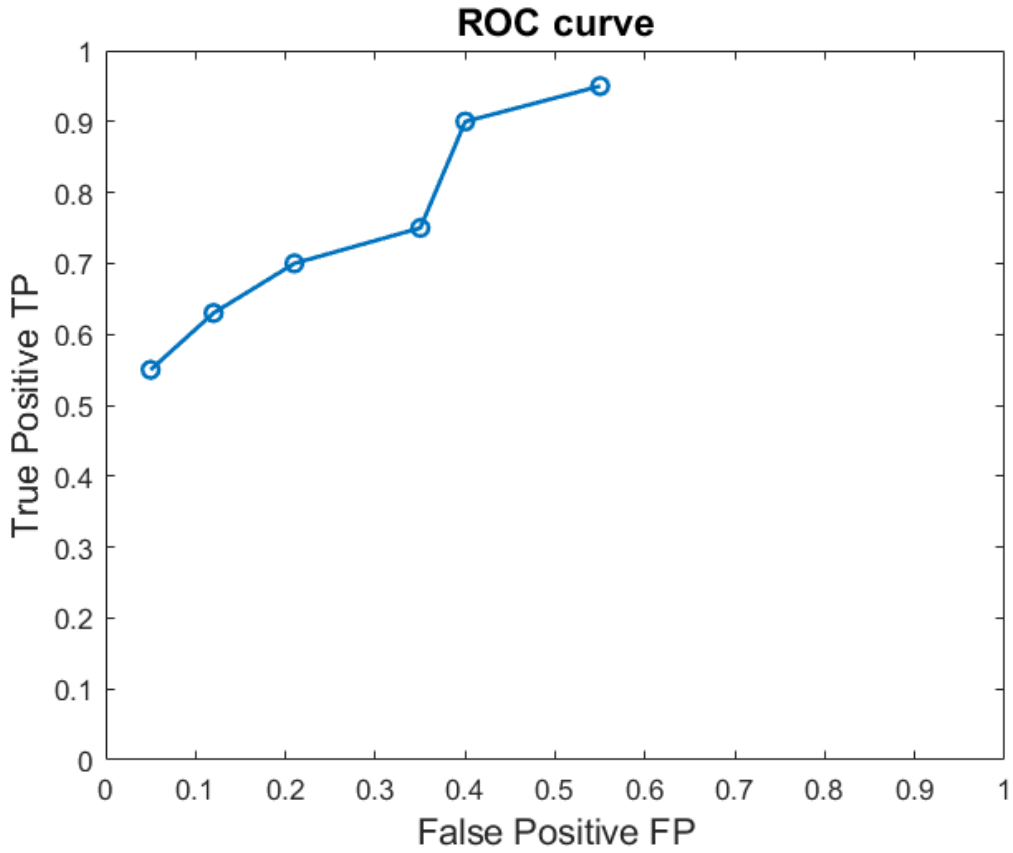
ROC Curve

Question: where would a model be which randomly guesses anomalies?



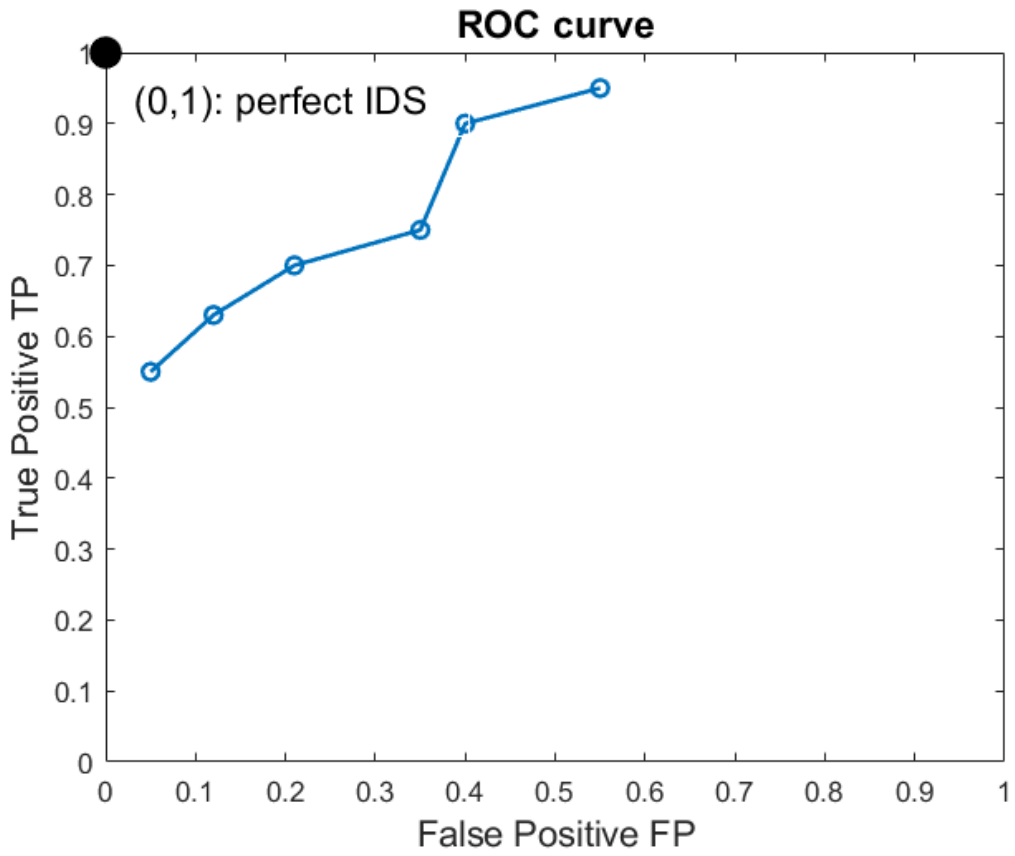
ROC Curve

Question: where would a model be which perfectly predicts the anomalies?

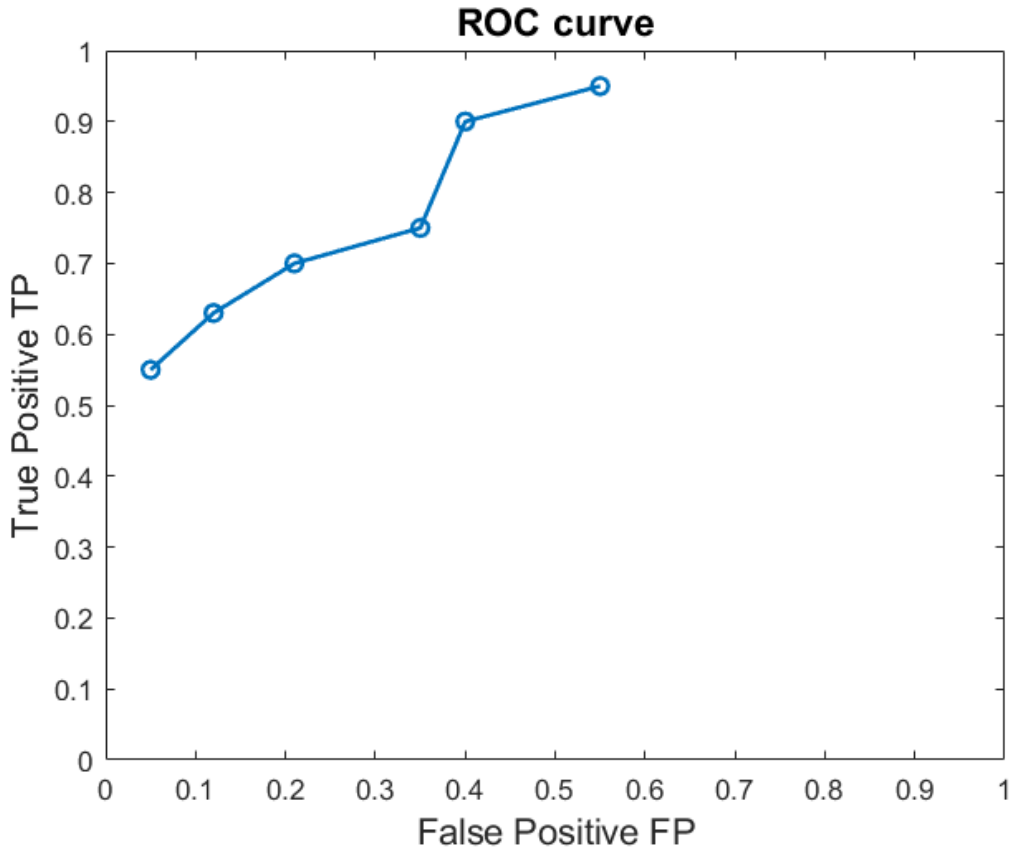


ROC Curve

Question: where would a model be which perfectly predicts the anomalies?

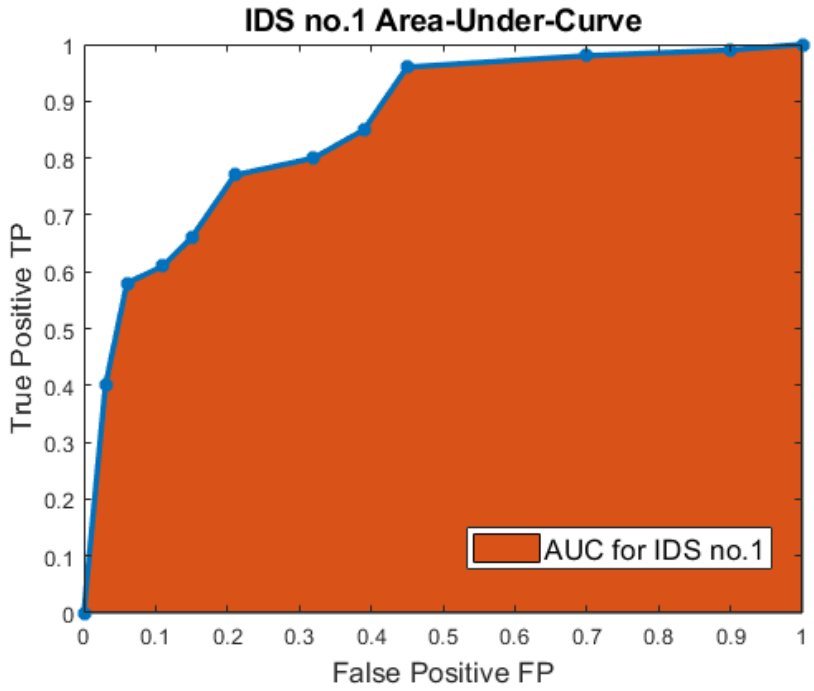


ROC Curve



How would you compare two models using the ROC curve?
How would you capture the performance of the model on the ROC curve in one metric?

ROC AUC


IDS no.1 Area-Under-Curve
True Positive TP
False Positive FP
AUC for IDS no.1
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

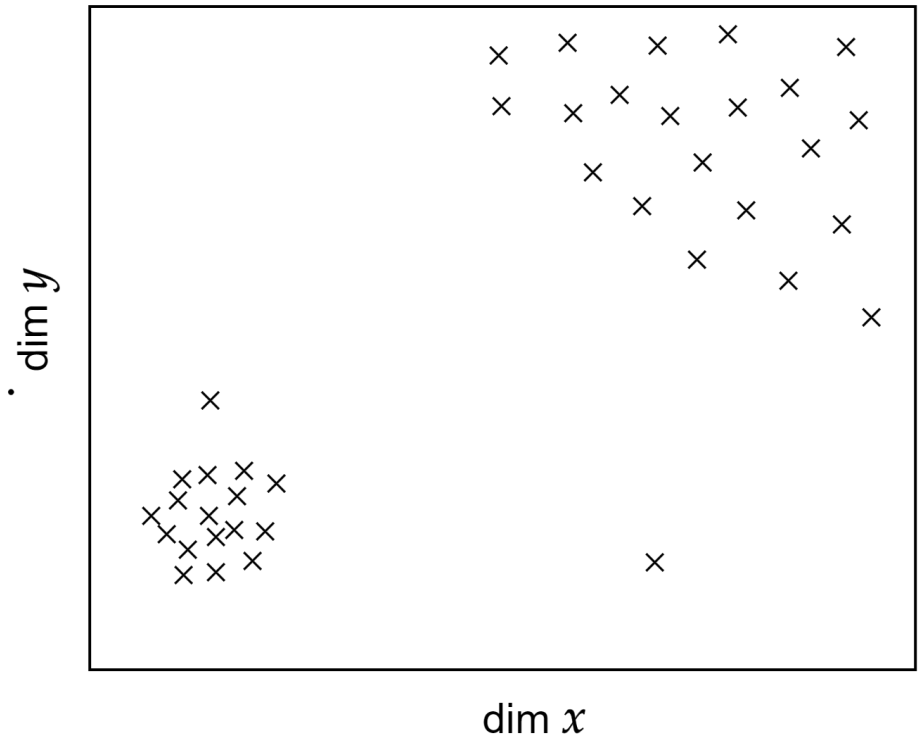
Contents

Models

- local outlier factor
- isolation forest
- Any ML model
- autoencoder

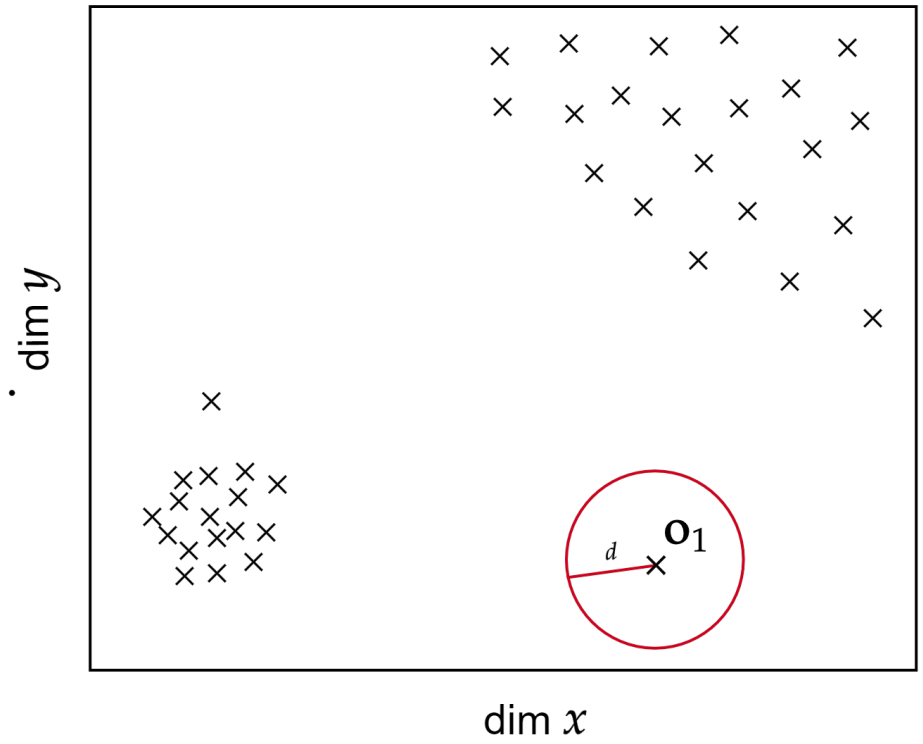
Local Outlier Factor

LOF



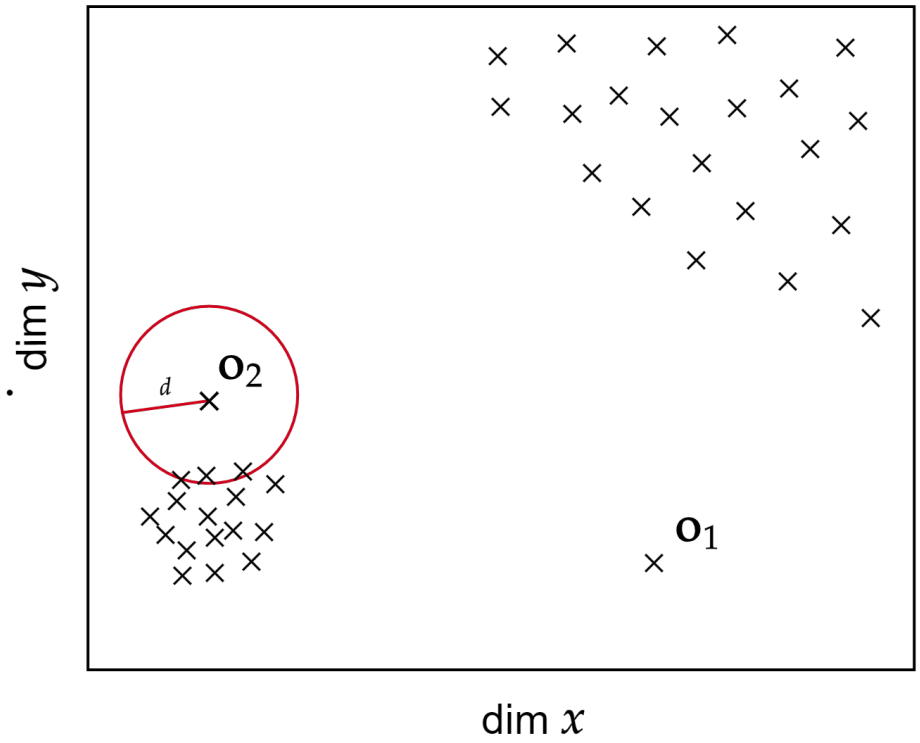
- To introduce LOF we will consider a special dataset as example
- The 2-dimensional dataset $\mathcal{D}$ has two main clusters

LOF



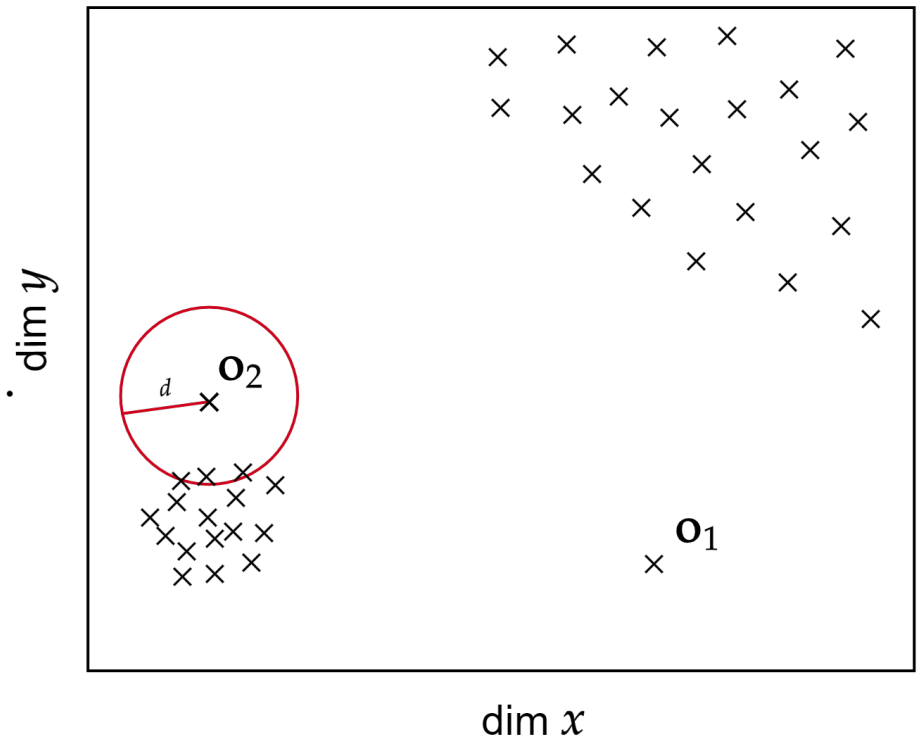
- If we search in radius d around $\mathbf{o}_1$ then we will find no other datapoints
- Thus DBO (Distance-Based Outlier) will easily mark $\mathbf{o}_1$ as an outlier

LOF



- What about datapoint $\mathbf{o}_2$ though?
- If we search in radius d around $\mathbf{o}_2$ we will find datapoints from cluster 2
- Thus, unless we pick a very small radius, DBO will not mark $\mathbf{o}_2$ as an outlier

LOF

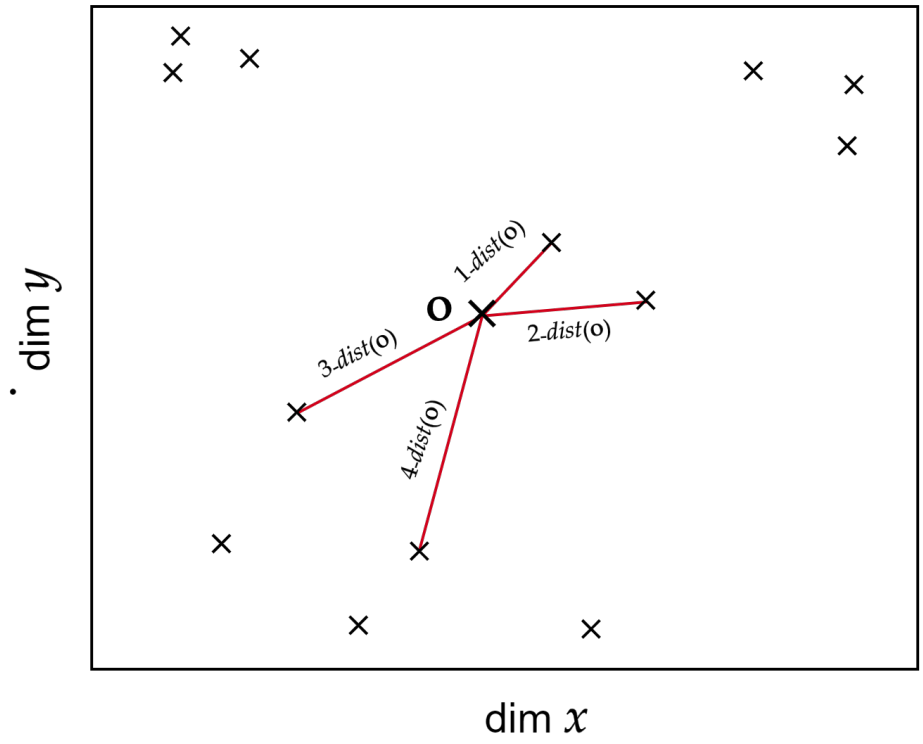


- For DBO $\mathbf{o}_1$ is a **global outlier**
- But $\mathbf{o}_2$ is not a global outlier
- Still, it is an outlier **relative to the local neighbourhood**

LOF

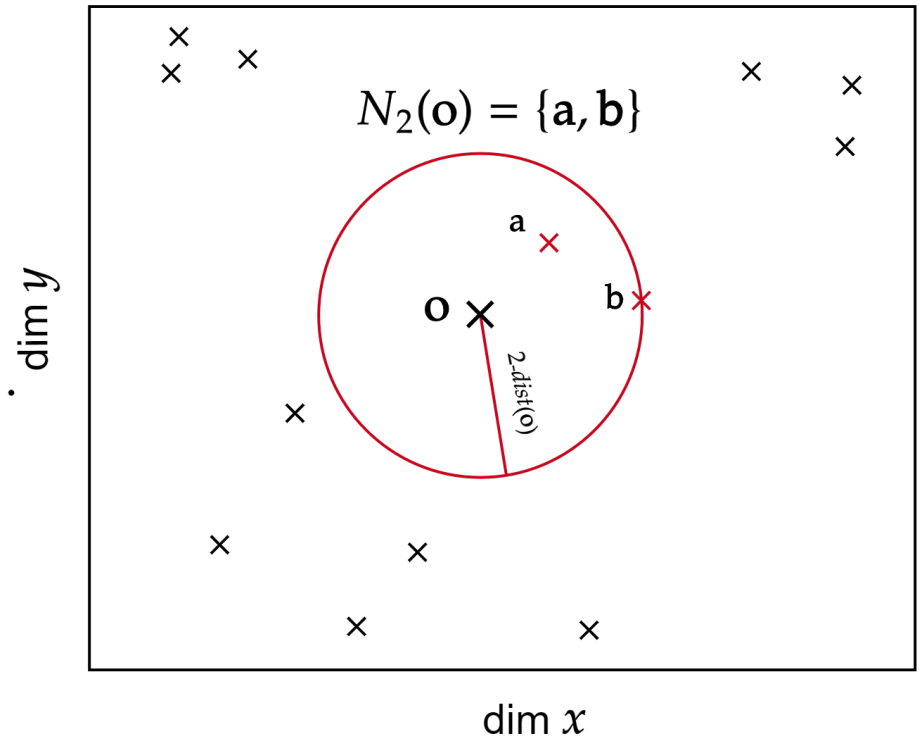
- When clusters of different densities exist, outliers can be merged to a cluster
- This results to undetected outliers and is known as **masking**
- The **Local Outlier Factor (LOF)** is a technique that can deal with such cases

k-distance. For any integer $k > 0$, we define the the k-distance of datapoint $\mathbf{o}$ as the distance of $\mathbf{o}$ to its k-th nearest neighbour. We refer to this distance as $k\text{-}dist(\mathbf{o})$



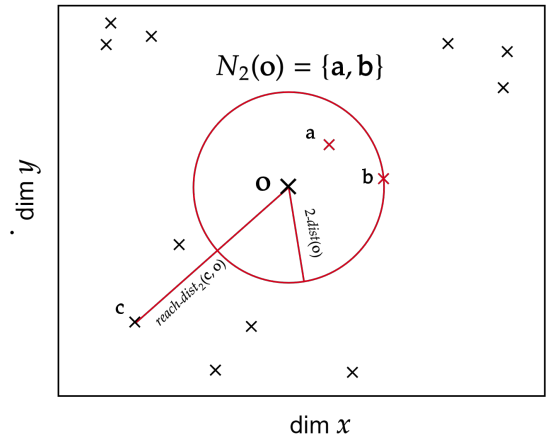
k-neighbourhood. We define the k-neighbourhood of datapoint $\mathbf{o}$ as the set of all datapoints $\mathbf{x}$ in dataset $\mathcal{D}$ (except $\mathbf{o}$) whose distance from $\mathbf{o}$ is not greater than the k-distance of $\mathbf{o}$. We refer to this set as $N_k(\mathbf{o})$.

$$N_k(\mathbf{o}) = \{\mathbf{x} \in \mathcal{D} / \{\mathbf{o}\} \text{ s.t. } dist(\mathbf{x}, \mathbf{o}) \leq k\text{-}dist(\mathbf{o})\}$$



Reachability distance. We define the reachability distance of datapoint $\mathbf{x}$ w.r.t. datapoint $\mathbf{o}$ as:

$$reach\text{-}dist_k(\mathbf{x}, \mathbf{o}) = max\{k\text{-}dist(\mathbf{o}), dist(\mathbf{x}, \mathbf{o})\}$$

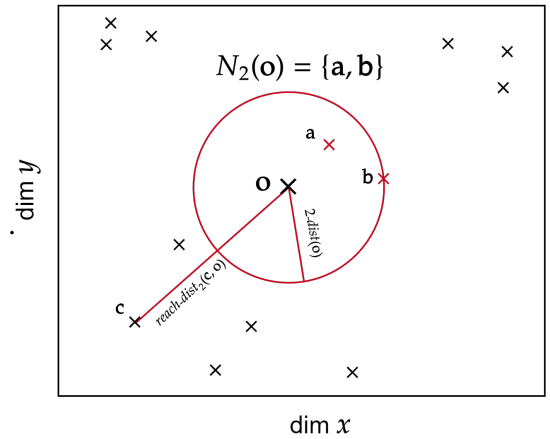


- If datapoint $\mathbf{x}$ is far away from datapoint $\mathbf{o}$ then their reachability distance is their actual distance between them

  e.g. $reach\text{-}dist_2(\mathbf{c}) = dist(\mathbf{c}, \mathbf{o})$

Reachability distance. We define the reachability distance of datapoint $\mathbf{x}$ w.r.t. datapoint $\mathbf{o}$ as:

$$reach\text{-}dist_k(\mathbf{x}, \mathbf{o}) = max\{k\text{-}dist(\mathbf{o}), dist(\mathbf{x}, \mathbf{o})\}$$



- However if the datapoints are sufficiently close then the reachability distance is the k-distance of $\mathbf{o}$.

  e.g. $reach\text{-}dist_2(\mathbf{a}) = 2\text{-}dist(\mathbf{o})$

Note that k is a hyper-parameter of LOF i.e. we need to fine-tune the algorithm by trying various values of k

Local reachability density. We define the lrd_k of datapoint $\mathbf{o}$ as the inverse of the average reachability distances for a certain k-neighbourhood.

$$lrd_k(\mathbf{o}) = \left(\frac{\sum_{\mathbf{x} \in N_k(\mathbf{o})} reach\text{-}dist_k(\mathbf{o}, \mathbf{x})}{|N_k(\mathbf{o})|} \right)^{-1}$$

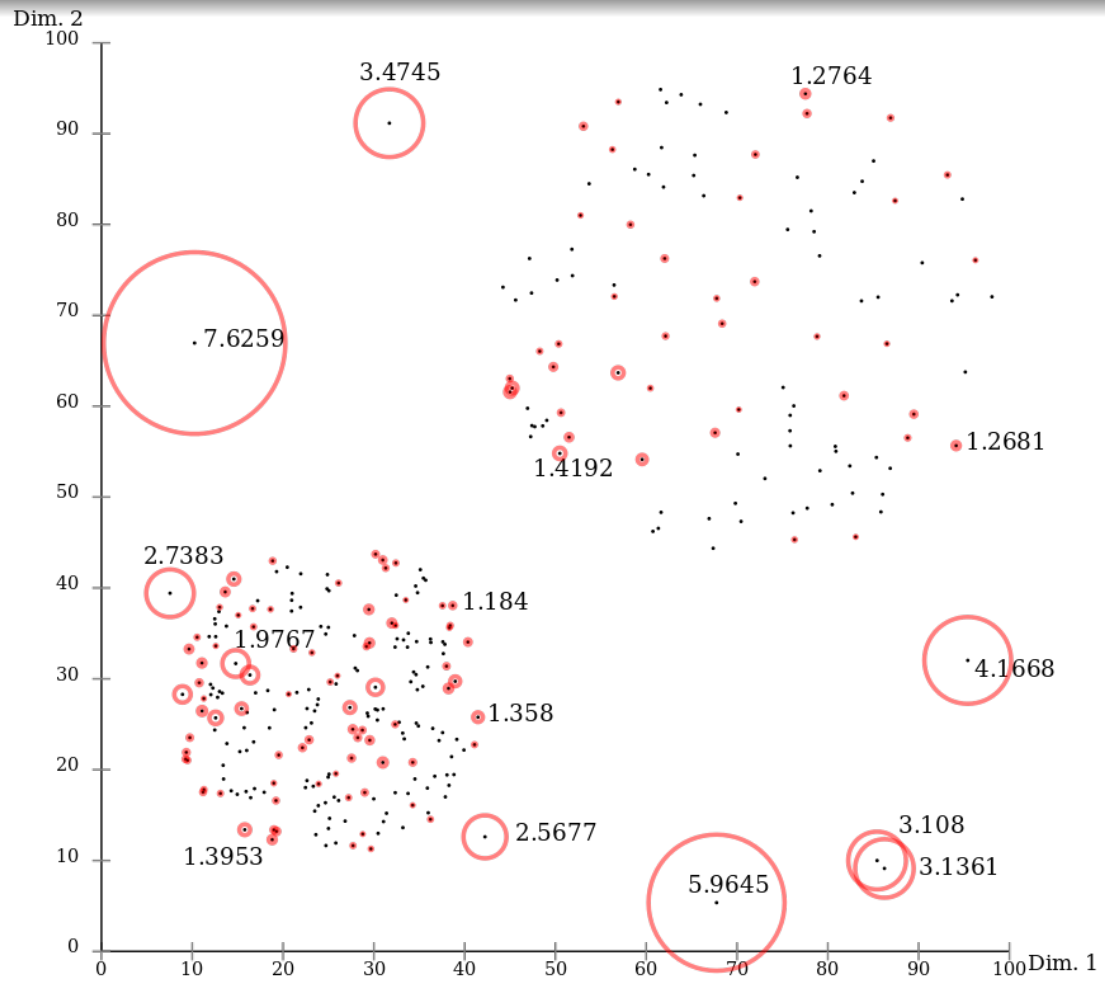
- $lrd_k(\mathbf{o})$ quantifies the density of datapoints in the neighbourhood of $\mathbf{o}$

Local Outlier Factor (LOF). We define the LOF_k as the average ratio of local reachabilities of $\mathbf{o}$ and the local reachabilities of the k-neighbourhood of $\mathbf{o}$.

$$LOF_k(\mathbf{o}) = \frac{\sum_{x \in N_k(\mathbf{o})} \frac{lrd_k(\mathbf{x})}{lrd_k(\mathbf{o})}}{|N_k(\mathbf{o})|}$$

- The LOF captures the degree to which the datapoint $\mathbf{o}$ is an outlier

LOF



- $LOF \approx 1$ means similar density as neighbours
- $LOF < 1$ means higher density than neighbours
- $LOF > 1$ means lower density than neighbours i.e. potential outlier

Isolation Forest

iForest

- Most anomaly detection techniques profile the normal behavior of a system and then identify outliers
- The isolation forest algorithm (iForest) works by isolating anomalies instead of profiling the normal behavior
- iForest takes advantage of the following two properties of anomalies:
  1. Anomalies are few compared to the number of normal instances
  2. The features of an anomaly are very different from the features of normal instances
- Since anomalies are "few and different" they are more susceptible to **isolation**, compared to normal instances

Constructing an isolation Tree (iTree)

Assume that we have the following training dataset with:

- 6 datapoints stored in vectors $\{\mathbf{x_0}, \mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3}, \mathbf{x_4}, \mathbf{x_5}\}$
- 5 features $[x^0, x^1, x^2, x^3, x^4]$ for each datapoint $\mathbf{x_i}$

$$\mathcal{D} = \begin{bmatrix} \mathbf{x_0}^\intercal \\ \mathbf{x_1}^\intercal \\ \mathbf{x_2}^\intercal \\ \mathbf{x_3}^\intercal \\ \mathbf{x_4}^\intercal \\ \mathbf{x_5}^\intercal \end{bmatrix} = \begin{bmatrix} 4.74 & 0.71 & -0.26 & 2.79 & 5.11 \\ 4.71 & 0.75 & -0.30 & 2.50 & 5.58 \\ 4.72 & 0.82 & -0.28 & 2.61 & 5.26 \\ 4.66 & 0.72 & -0.31 & 2.62 & 5.30 \\ 4.80 & 0.71 & -0.27 & 2.71 & 5.33 \\ 4.81 & 0.77 & -0.29 & 2.72 & 5.40 \end{bmatrix}$$

1. Randomly select a feature q from $\{x^0, x^1, x^2, x^3, x^4\}$

   e.g. we select feature $q = x^2$

2. Randomly select a split point p between the maximum and the minimum values of the selected feature q

   e.g. $max(x^2) = -0.26$ and $min(x^2) = -0.31$

   we select randomly $p \in \{-0.31, -0.26\}$ and we pick $p = -0.28$

$$\mathcal{D} = \begin{bmatrix} 4.74 & 0.71 & \mathbf{-0.26} & 2.79 & 5.11 \\ 4.71 & 0.75 & -0.30 & 2.50 & 5.58 \\ 4.72 & 0.82 & -0.28 & 2.61 & 5.26 \\ 4.66 & 0.72 & \mathbf{-0.31} & 2.62 & 5.30 \\ 4.80 & 0.71 & -0.27 & 2.71 & 5.33 \\ 4.81 & 0.77 & -0.29 & 2.72 & 5.40 \end{bmatrix}$$

3. We partition the dataset $\mathcal{D}$ to the left and right sets $\mathcal{D}_l$ and $\mathcal{D}_r$ such that:

$$\mathcal{D}_l = \{\mathbf{x} \in \mathcal{D} \text{ s.t. } q < p\} \quad \text{and} \quad \mathcal{D}_r = \{\mathbf{x} \in \mathcal{D} \text{ s.t. } q \geq p\}$$

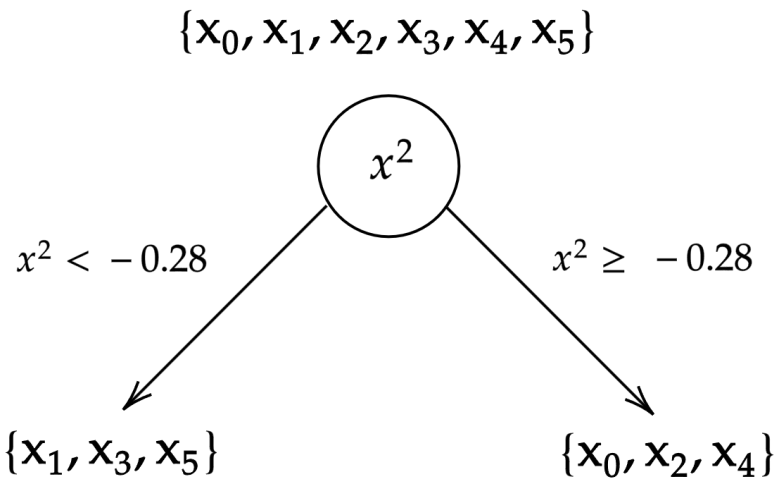
e.g. we split the dataset in the following way:

$$\mathcal{D}_l = \{\mathbf{x} \in \mathcal{D} \text{ s.t. } x^2 < -0.28\} \quad \text{and} \quad \mathcal{D}_r = \{\mathbf{x} \in \mathcal{D} \text{ s.t. } x^2 \geq -0.28\}$$

$$\mathcal{D} = \begin{bmatrix} 4.74 & 0.71 & -0.26 & 2.79 & 5.11 \\ 4.71 & 0.75 & -0.30 & 2.50 & 5.58 \\ 4.72 & 0.82 & -0.28 & 2.61 & 5.26 \\ 4.66 & 0.72 & -0.31 & 2.62 & 5.30 \\ 4.80 & 0.71 & -0.27 & 2.71 & 5.33 \\ 4.81 & 0.77 & -0.29 & 2.72 & 5.40 \end{bmatrix}$$

iForest

- We organize the partitioned sets using a binary tree structure
- We refer to this as the **isolation tree** (iTree)



selected feature q: x^2

split point p: -0.28

4. The construction of the iTree continues recursively
   - We focus on the left set $\mathcal{D}_l$

$$\mathcal{D}_l = \begin{bmatrix} 4.71 & 0.75 & -0.30 & 2.50 & 5.58 \\ 4.66 & 0.72 & -0.31 & 2.62 & 5.30 \\ 4.81 & 0.77 & -0.29 & 2.72 & 5.40 \end{bmatrix}$$

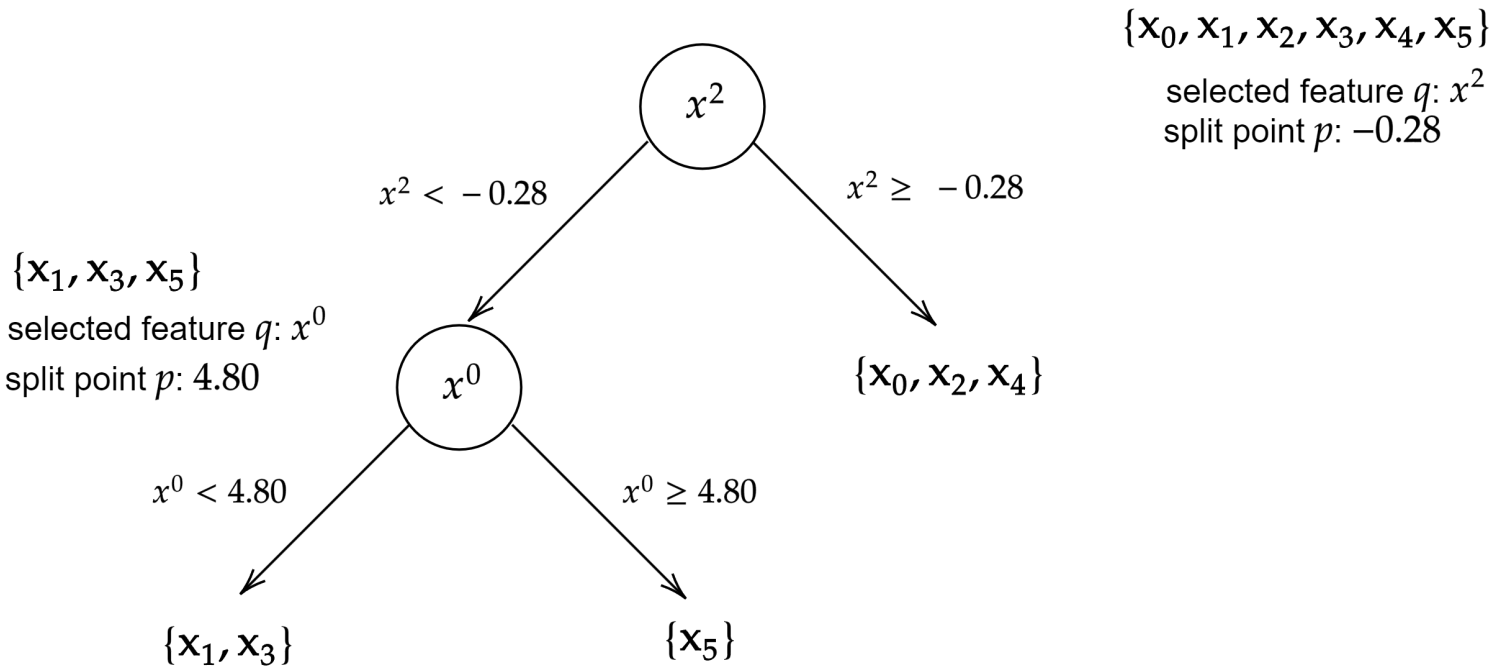
   - We select randomly feature $q = x^0$ and then select split point $p \in \{min(x^0), max(x^0)\} \iff p \in \{4.66, 4.81\}$. We pick $p = 4.80$

$$\mathcal{D}_l = \begin{bmatrix} 4.71 & 0.75 & -0.30 & 2.50 & 5.58 \\ \mathbf{4.66} & 0.72 & -0.31 & 2.62 & 5.30 \\ \mathbf{4.81} & 0.77 & -0.29 & 2.72 & 5.40 \end{bmatrix}$$

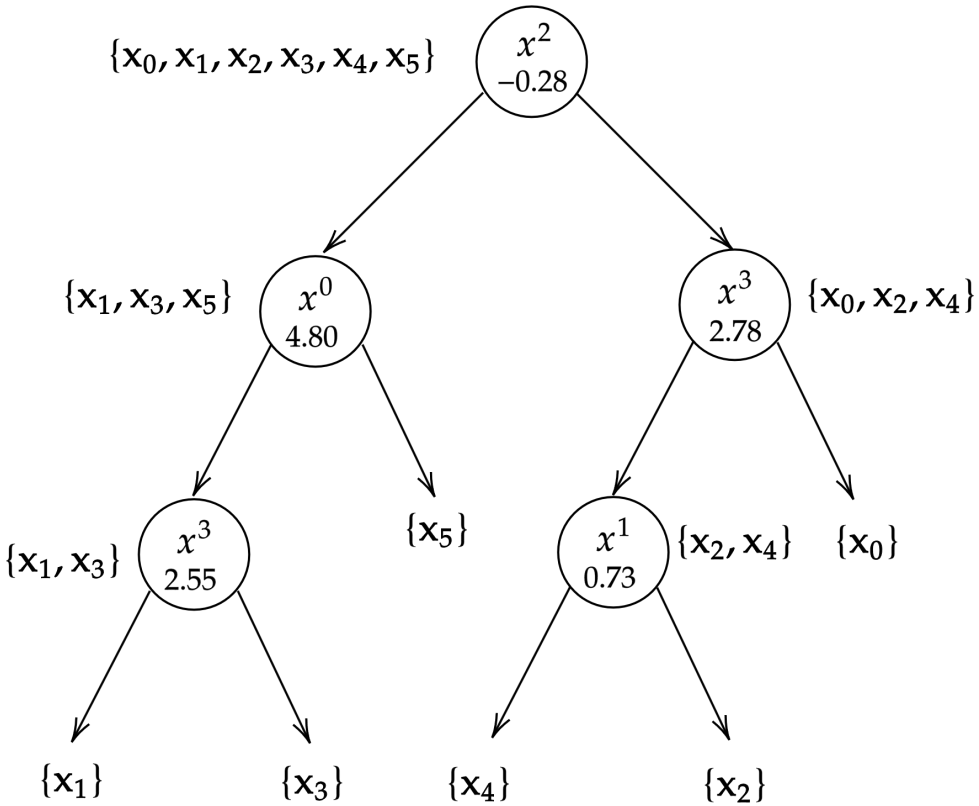
   - We split $\mathcal{D}_l$ to $\{\mathbf{x} \in \mathcal{D}_l \text{ s.t. } x^0 < 4.80\}$ and $\{\mathbf{x} \in \mathcal{D}_l \text{ s.t. } x^0 \geq 4.80\}$

$$\mathcal{D}_l = \begin{bmatrix} 4.71 & 0.75 & -0.30 & 2.50 & 5.58 \\ 4.66 & 0.72 & -0.31 & 2.62 & 5.30 \\ 4.81 & 0.77 & -0.29 & 2.72 & 5.40 \end{bmatrix}$$

- The iTree grows as we keep splitting the dataset



- The full iTree is constructed once all nodes have set size 1 or when a predefined tree height limit is reached



iForest

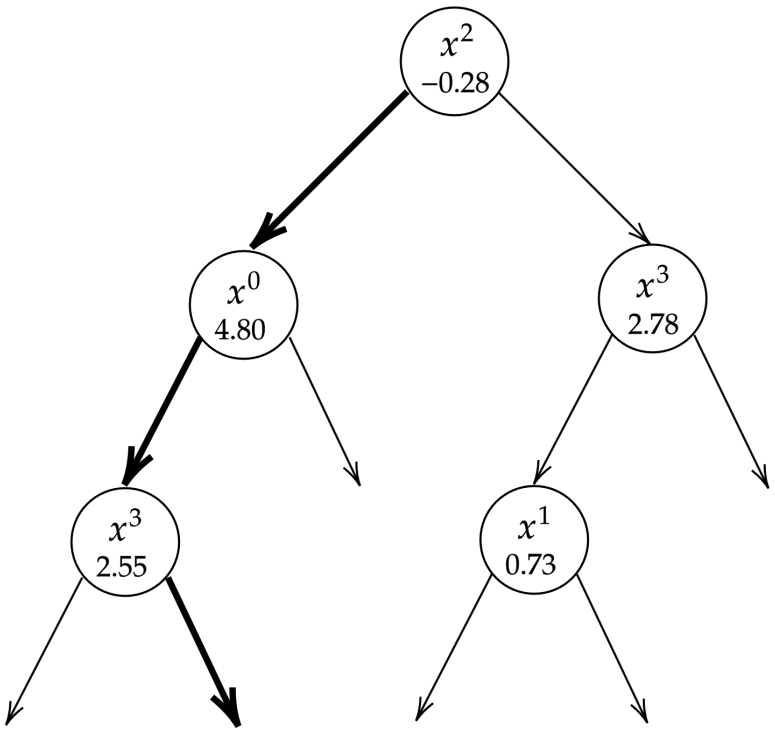
- The iTree algorithm is initialized with dataset $\mathcal{D}$ and possibly an upper limit on the tree height l. The function works recursively, keeping track of the current tree height e.
- During recursion, the function should also keep track of all selected features q and all the split values p

iTree($\mathcal{D}, e, l, q, p$)
if $e \geq l$ *or* $|\mathcal{D}| \leq 1$ **then**
  return $\emptyset$
end
select randomly feature q
select randomly split point $p \in [min(q), max(q)]$
$\mathcal{D}_l = \{\mathbf{x} \in \mathcal{D}$ s.t. $q < p\}$
$\mathcal{D}_r = \{\mathbf{x} \in \mathcal{D}$ s.t. $q \geq p\}$
return iTree($\mathcal{D}_l, e + 1, l, p, q$)
return iTree($\mathcal{D}_r, e + 1, l, p, q$)

Testing data on an iTree

- We have used the training dataset $\mathcal{D}$ to construct an iTree
- Let's test it with a datapoint $\mathbf{x}$ that is labeled as "normal behavior"

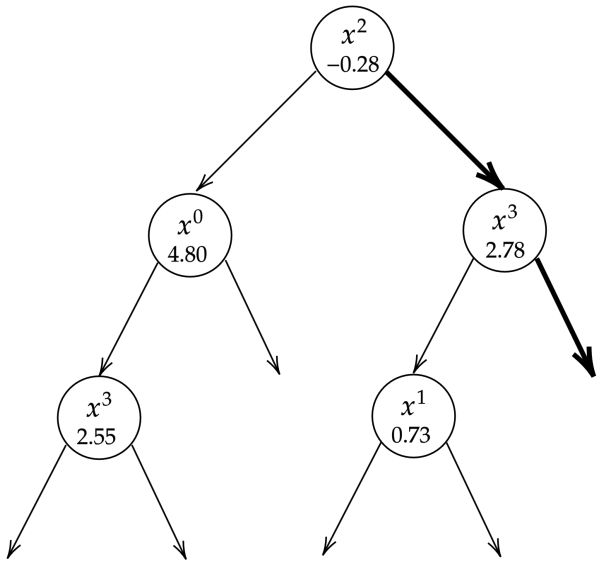
$$\mathbf{x} = \begin{bmatrix} 4.70 & 0.73 & -0.29 & 2.65 & 5.41 \end{bmatrix}$$



iForest

- Let's test it with a datapoint $\mathbf{x}'$ that is labeled as "anomaly"

$$\mathbf{x}' = \begin{bmatrix} 5.70 & 0.71 & -0.09 & 3.51 & 5.39 \end{bmatrix}$$



- Passing $\mathbf{x}'$ through the iTree results in path length equal to 2
- **iForest idea:** On average, the path length of anomalies is shorter compared to the path length of normal datapoints

Training phase

- Constructing an iTree is a probabilistic process, since we choose randomly the feature q and the split point p
- The iForest generates t iTrees (ensemble) to perform anomaly detection using the training dataset $\mathcal{D}$
- Since the goal is not to model the normal behavior, every iTree can be constructed using a sub-sample $\mathcal{D}'$ of the training dataset $\mathcal{D}$ with size $\psi < |\mathcal{D}|$

1 iForest($\mathcal{D}, t, \psi$)
2 $forest = \emptyset$
3 **for** *i=1 to t* **do**
4 $\quad \mathcal{D}' \xleftarrow{R}$ sample($\mathcal{D}, \psi$)
5 $\quad forest = forest \cup$ iTree($\mathcal{D}'$)
6 **end**
7 return $forest$

Testing phase

- Let datapoint **x** that we need to decide if it is “anomaly" or “normal"
- iForest gets the path lengths produced by **x** across all iTrees in the ensemble
- The average path length is computed and possibly normalized by factor c to produce the anomaly score

1 Score($\mathbf{x}$, $forest$)
2 **for** *i=1 to t* **do**
3 $\quad tree = forest(i)$
4 $\quad pl(i) =$ PathLength($\mathbf{x}$, $tree$)
5 **end**
6 $\overline{pl} = \sum_{i=1}^{t} pl(i)$
7 $c = 2(log(\psi - 1) + 0.5772156649) - 2(\psi - 1)/\psi$
8 $score = 2^{-\overline{pl}/c}$
9 return score

Final notes on iForest

- Simple and efficient algorithm that matches the accuracy of more complex anomaly detection methods.
- iForest does not model the normal behavior, thus it can sub-sample the training dataset and work with partial models. This reduces the computational cost.
- iForest does not use a standard distance/density metric, reducing the computational cost.
- iForest has linear time complexity and low memory requirements, thus it can scale up to process large datasets and/or datasets with a large number of dimensions.
- Works effectively with high-dimensional datasets when several dimensions are not related to the anomalies.
- Deals well with masking effects

Machine Learning for Anomaly Detection

ML for AD

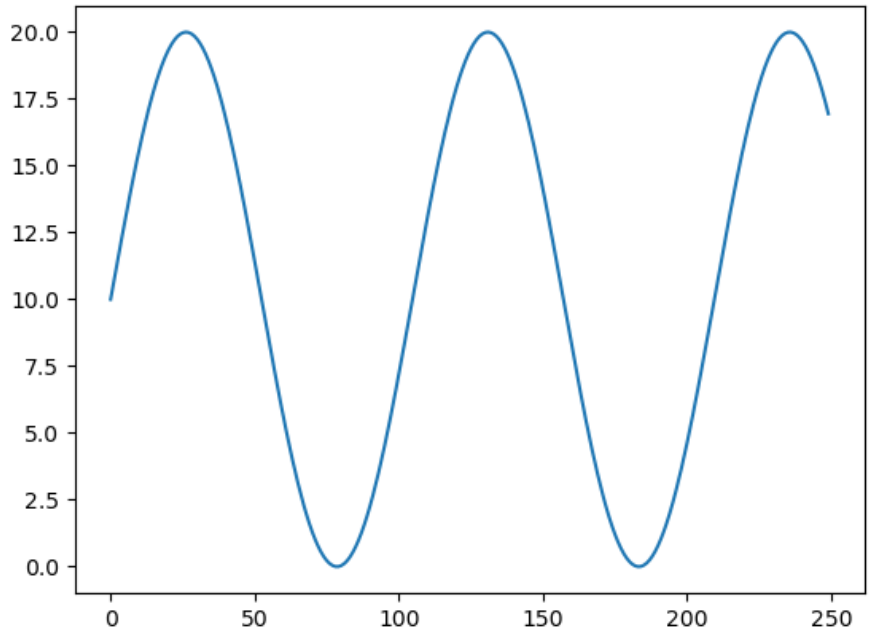
- In temporal datasets it is possible to perform anomaly detection using any conventional ML model
- Idea: anomalies are those datapoints which fall outside of the expected
- We train a ML model to predict the next value in the dataset
- Then, assuming our model is accurate, use the distance between the actual datapoint and the predicted datapoint as a measure of anomalousness

ML for AD

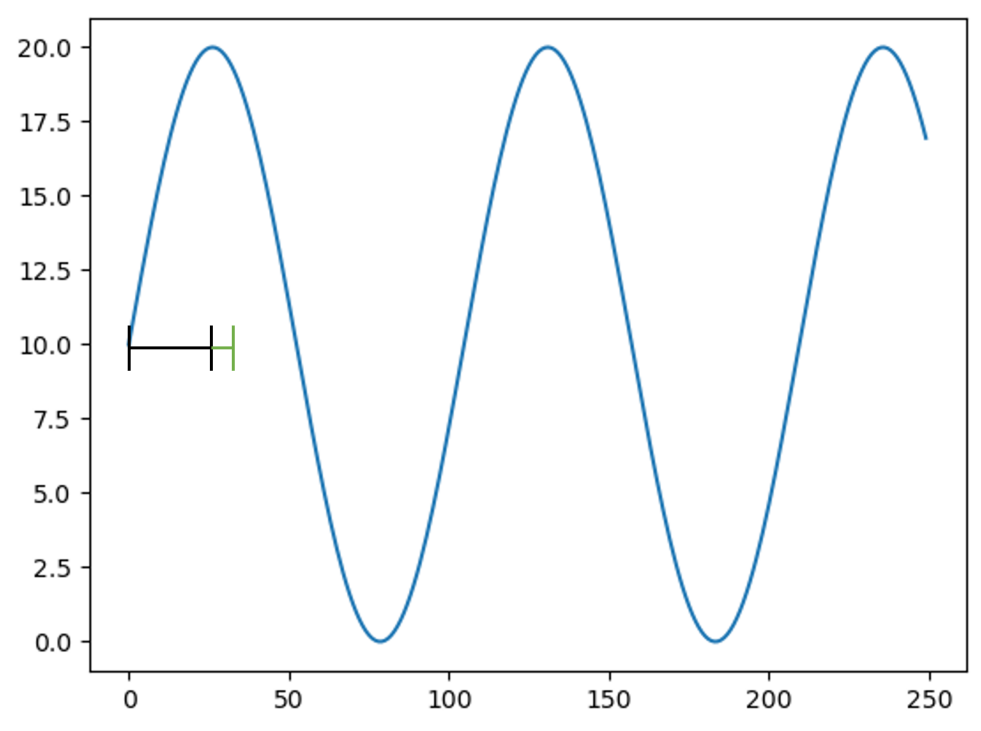
Possible models

- SVM
- Random Forest
- MLP
- CNN
- Graph Neural Networks
- ...

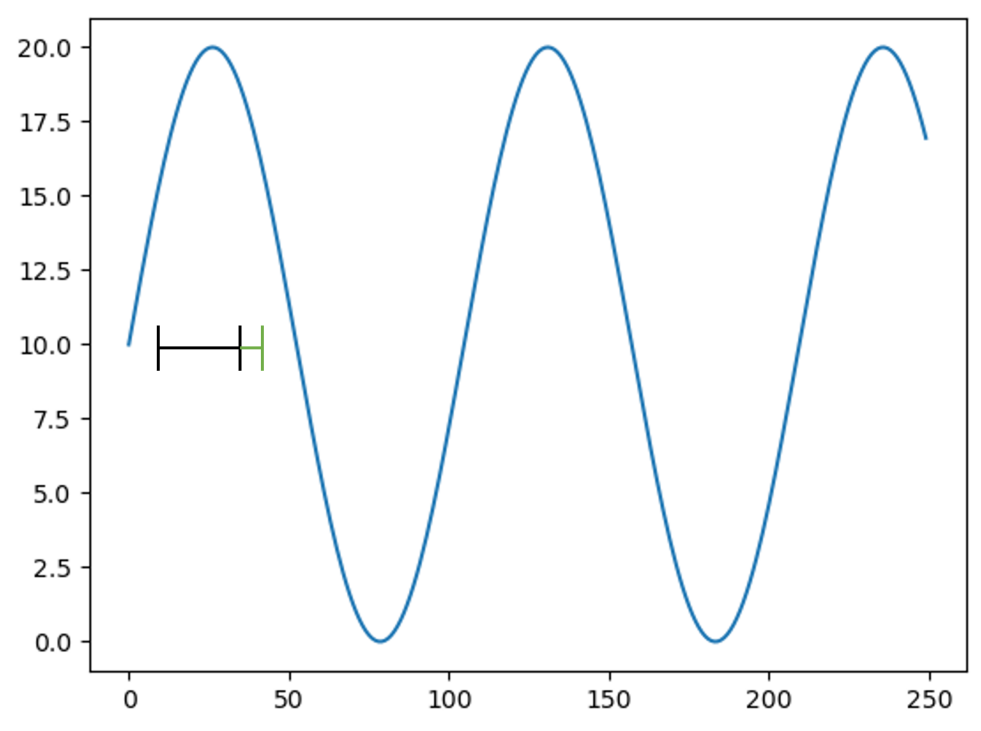
ML for AD



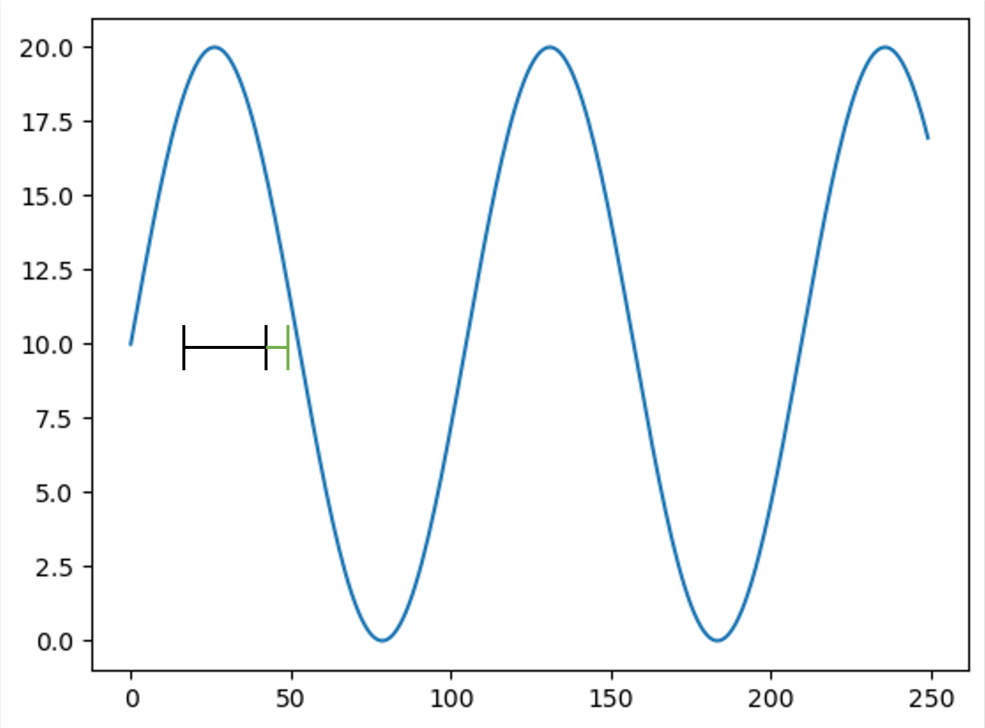
ML for AD



ML for AD



ML for AD



- Still unsupervised AD

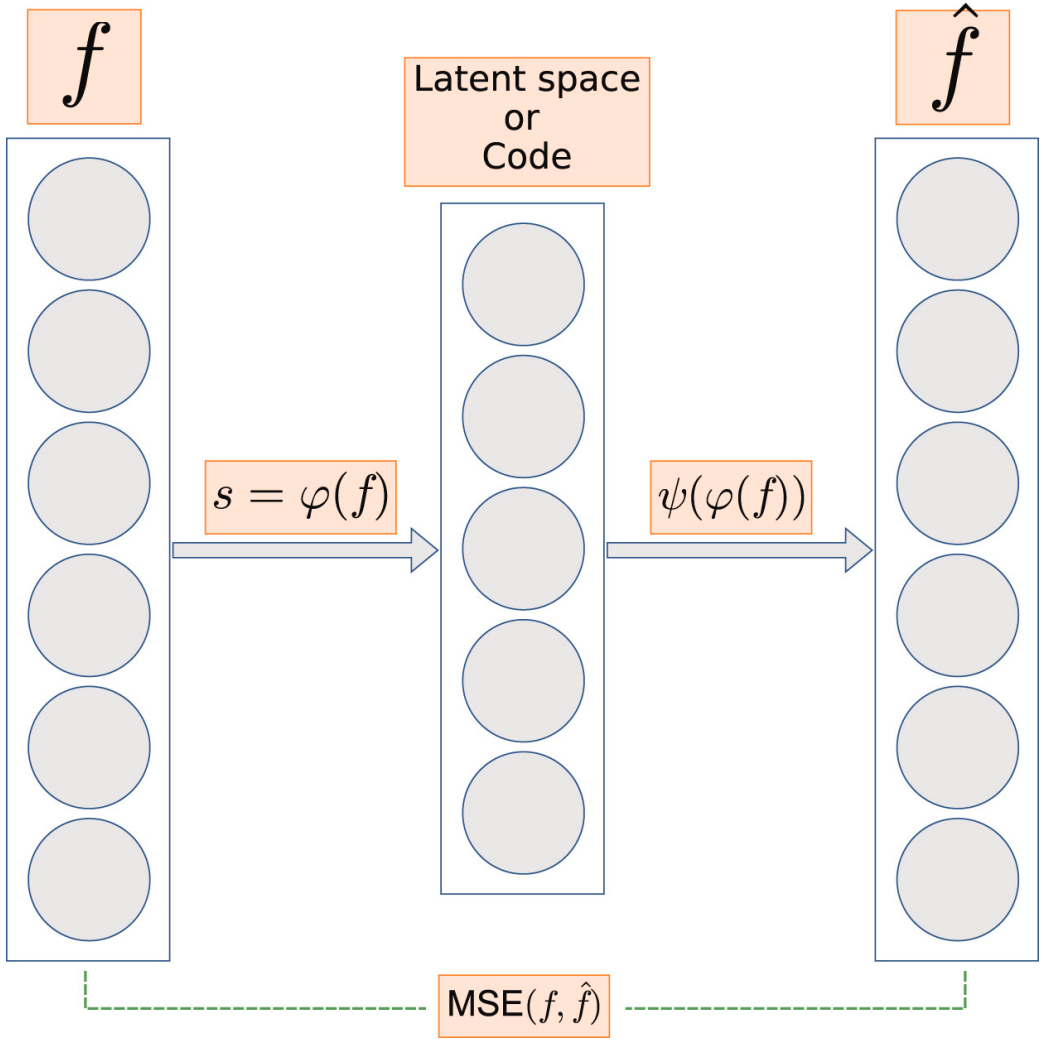
ML for AD

- In anomaly detection, response time is often important. Especially so in temporal settings as these are often live.
- You need to carefully consider your model complexity together with the hardware you have available.
- Your model needs to be updated throughout its lifetime, your environment is almost always non-stationary.

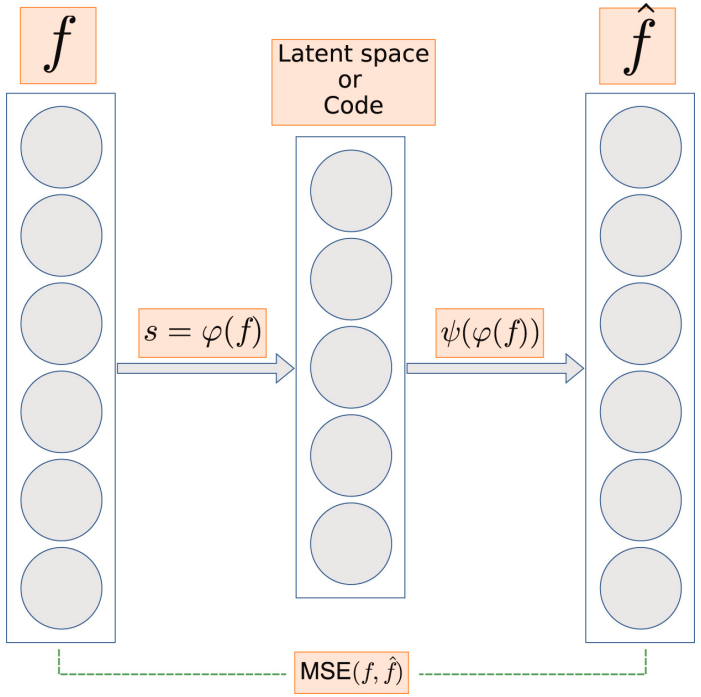
A special case concerns a specific neural network architecture:

The Autoencoder

Autoencoder for AD

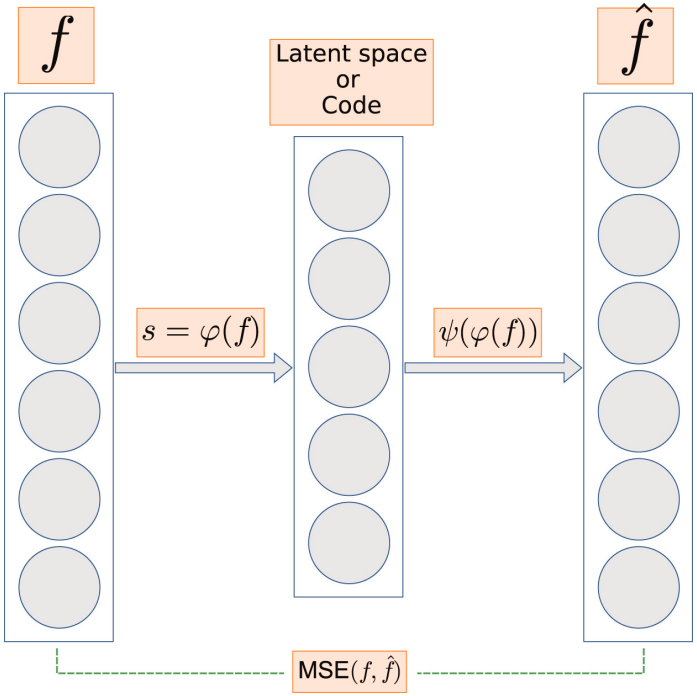


Autoencoder for AD



- An autoencoder consists of an encoder and a decoder
- The encoder encodes input into latent space
- The decoder decodes back from latent space
- By doing this the autoencoder learns to compress data, only maintaining relevant features

Autoencoder for AD



- Idea: the autoencoder learns to reconstruct data points
- Anomalies fall outside the normal data pattern
- Therefore, autoencoders are ineffective at reconstructing anomalies
- We can use the reconstruction error as a measure of anomalousness

Thanks

Special thanks to Kostas Papagiannopoulos from the University of Amsterdam, whose slides this slide deck is based on

kpcrypto.net

Thanks for your attention!