MemStream: Memory-Based Streaming Anomaly Detection

In Proceedings of the ACM Web Conference 2022, April 25-29

Siddharth Bhatia (National University of Singapore - siddharth@comp.nus.edu.sg)

Arjit Jain (IIT Bombay - arjit@cse.iitb.ac.in)

Shivin Srivastava (National University of Singapore - shivin@comp.nus.edu.sg)

Kenji Kawaguchi (Harvard University - kkawaguchi@fas.harvard.edu)

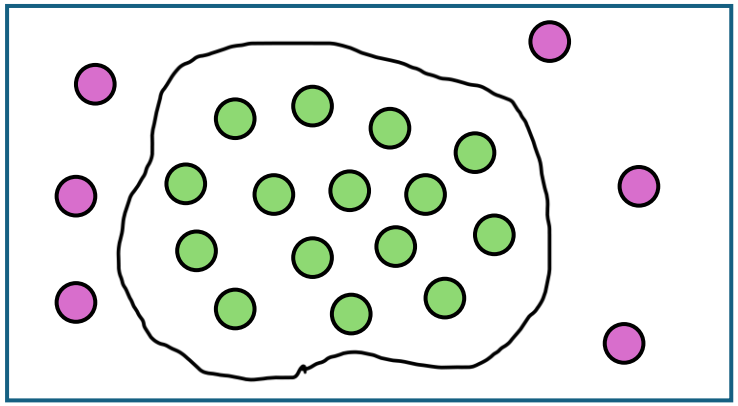
Bryan Hooi (National University of Singapore - bhooi@comp.nus.edu.sg)

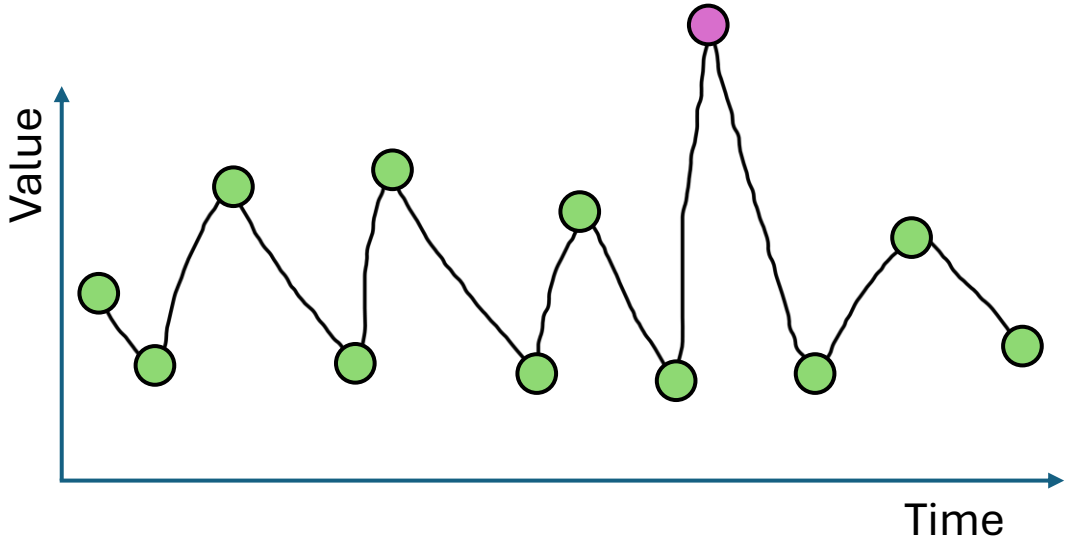
Scalable Data Stream Processing
Task 2 – Reproducible Research

Nicolas Mathias Hahn

Anomaly Detection

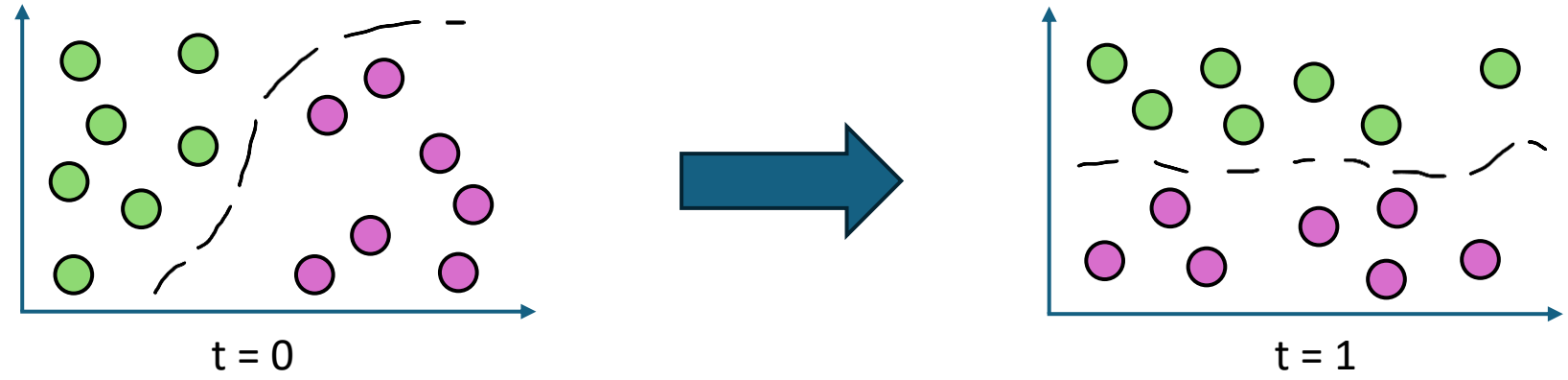
- **Anomalies**, often referred to as outliers, abnormalities, rare events, or deviants, are data points or patterns in data that do not conform to a notion of normal behavior.
- **Anomaly detection** is the task of finding those patterns in data that do not adhere to expected norms, given previous observations.



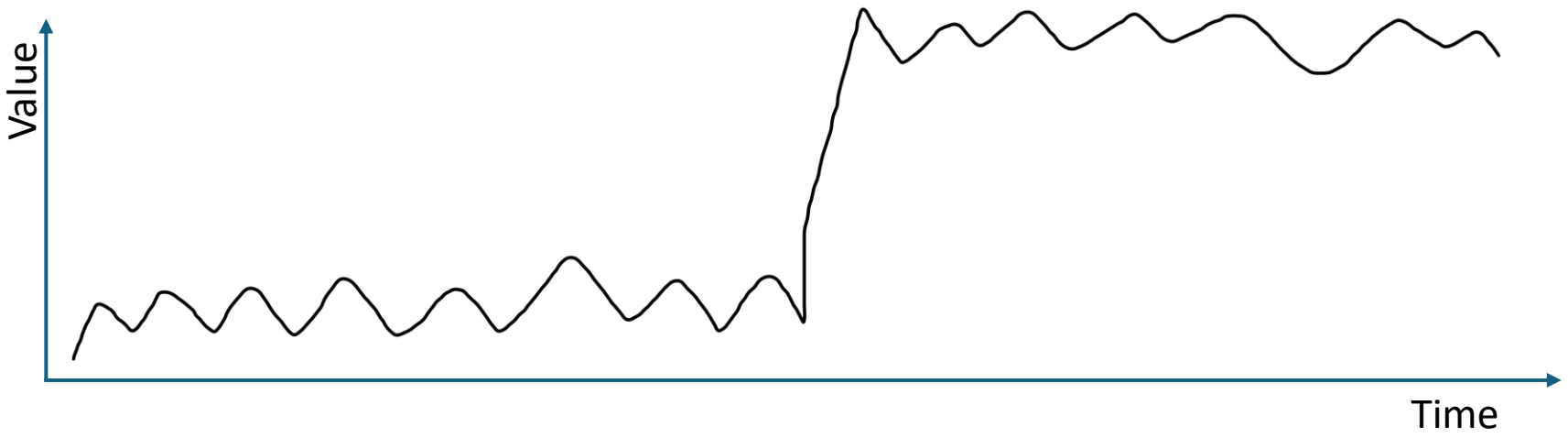


Concept Drift

Classification

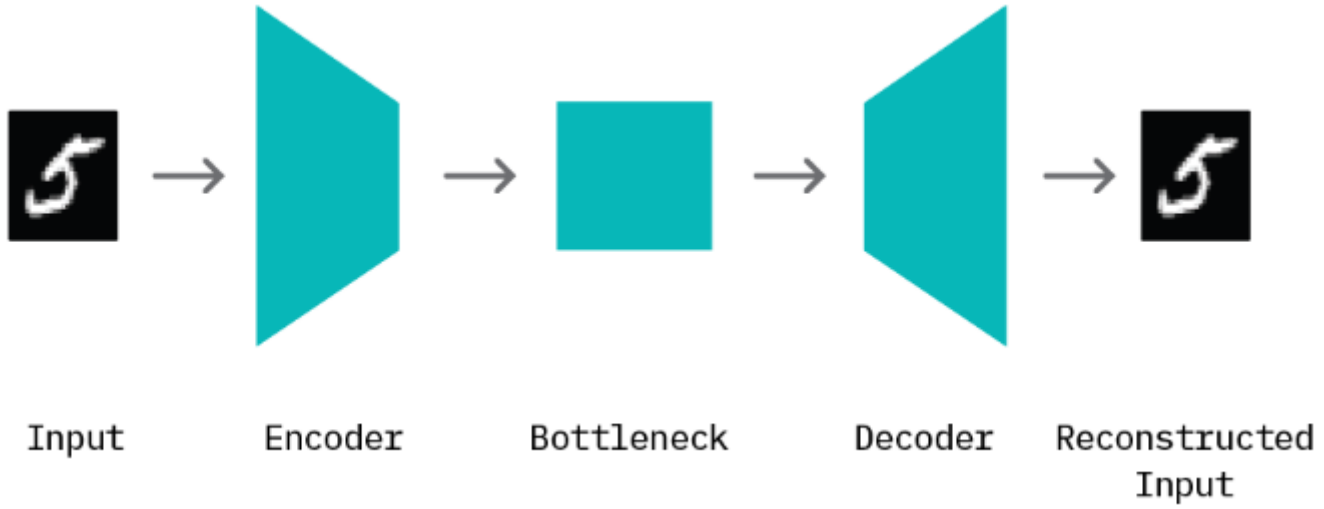


Time Series

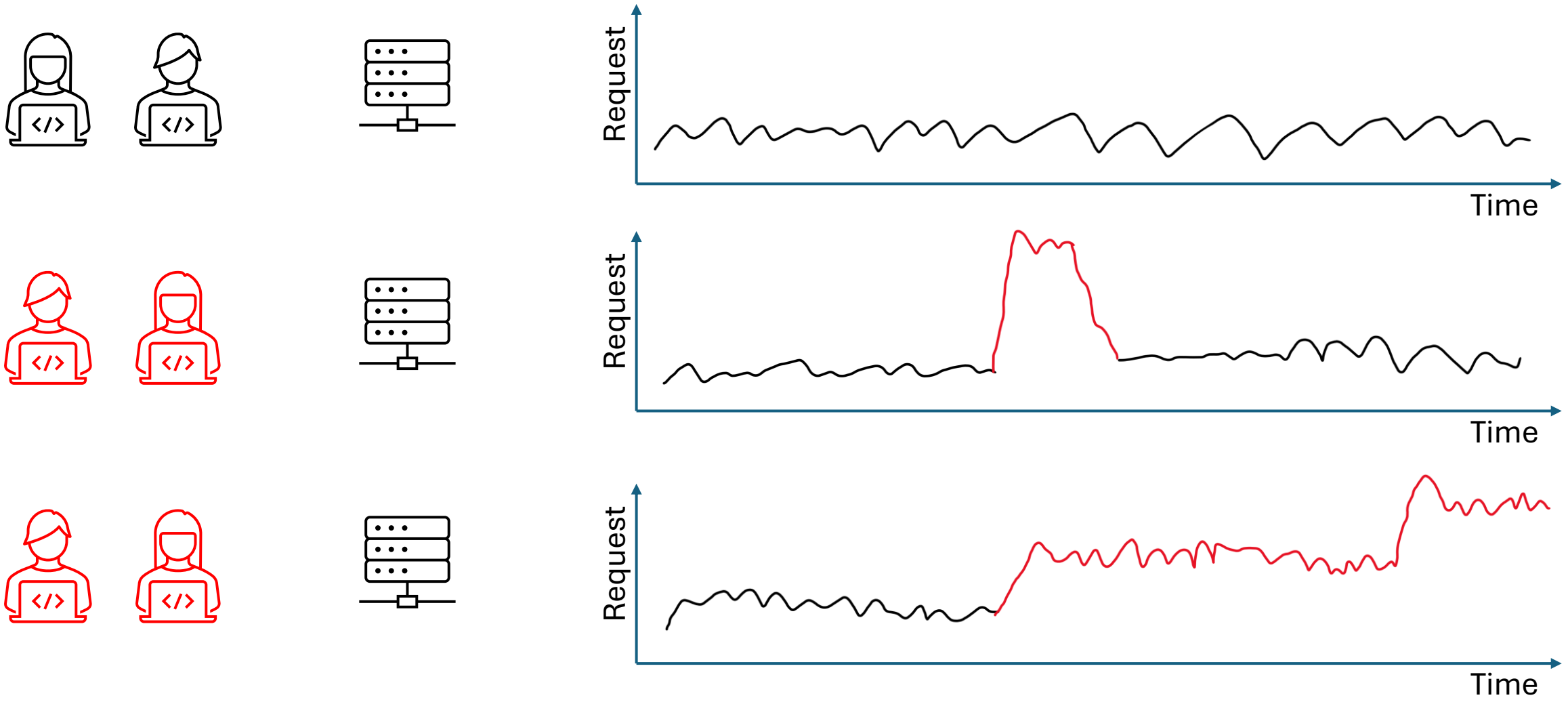


Autoencoders

- Neural networks designed to learn a low-dimensional representation, given some input data. They consist of two components:
  - **Encoder:** map input data to a low-dimensional representation (termed the *bottleneck*).
  - **Decoder:** map this low-dimensional representation back to the original input data.
- The model is trained by minimizing the reconstruction error, which is the difference (mean squared error) between the original input and the reconstructed output produced by the decoder.



MemStream: Motivation



MemStream: Overview

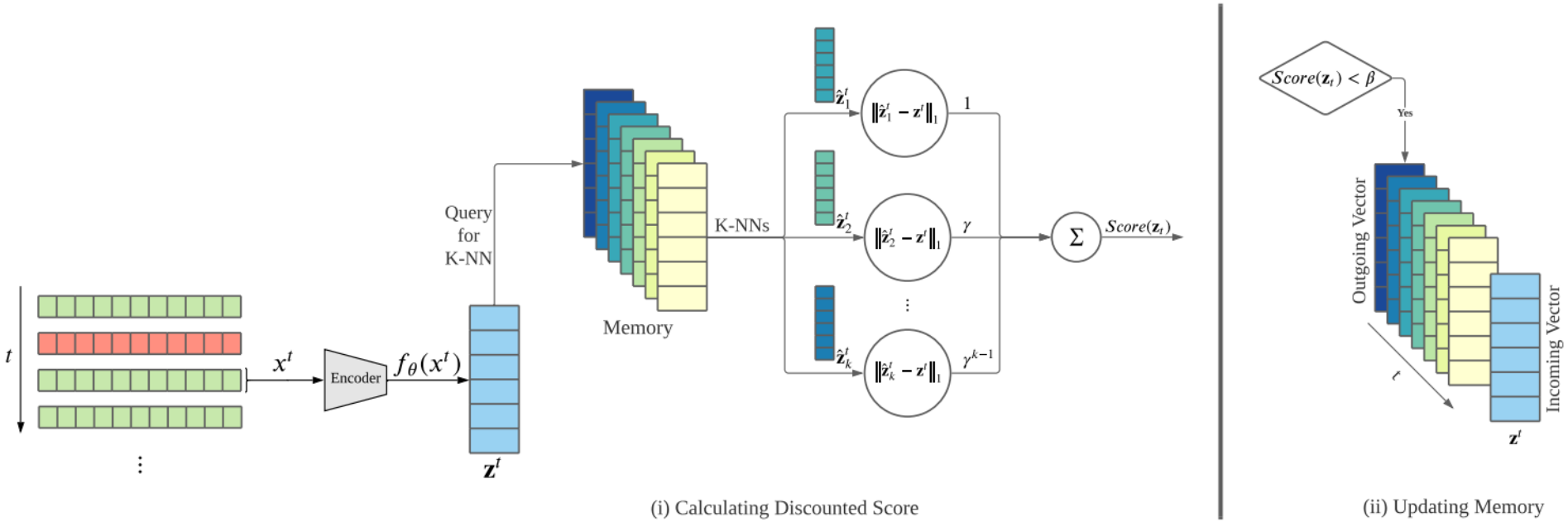


Figure 1: After an initial training of the feature extractor on a small subset of normal data, MemStream processes records in two steps: (i) It outputs anomaly scores for each record by querying the memory for K-nearest neighbours to the record encoding and calculating a discounted distance and (ii) It updates the memory, in a FIFO manner, if the anomaly score is within an update threshold β.

MemStream: Algorithm

Algorithm 1: MemStream

Input: Stream of data records
Output: Anomaly scores for each record

1 ▷ **Initialization**
2 Feature Extractor, f_θ, trained using small subset of data $\mathcal{D}$
3 Memory, M, initialized as $f_\theta(\mathcal{D})$
4 **while** *new sample* $\mathbf{x}^t$ *is received:* **do**
5 ▷ **Extract features:**
6 $\mathbf{z}^t = f_\theta(\mathbf{x}^t)$
7 ▷ **Query memory:**
8 $\{\hat{\mathbf{z}}_1^t, \hat{\mathbf{z}}_2^t ... \hat{\mathbf{z}}_K^t\} = K$-nearest neighbours of $\mathbf{z}^t$ in M
9 ▷ **Calculate distance:**
10 $R(\mathbf{z}^t, \hat{\mathbf{z}}_i^t) = ||\mathbf{z}^t - \hat{\mathbf{z}}_i^t||_1$ **for all** $i \in 1..K$
11 ▷ **Assign discounted score:**
12 $$Score(\mathbf{z}^t) = \frac{\sum_{i=1}^{K} \gamma^{i-1} R(\mathbf{z}^t, \hat{\mathbf{z}}_i^t)}{\sum_{i=1}^{K} \gamma^{i-1}}$$
13 ▷ **Update Memory:**
14 **if** $Score(\mathbf{z}^t) < \beta$ **then**
15 Replace earliest added element in $\boldsymbol{M}$ with $\mathbf{z}^t$
16 ▷ **Anomaly Score:**
17 **output** $Score(\mathbf{z}^t)$

- The autoencoder is initially trained with a small amount of data D to learn how to generate data embeddings (line 2).
- The memory is initialized with the same training dataset (line 3). We also store the mean and standard deviation of this small training dataset.
- As new records arrive, the encoder performs normalization using the stored mean and standard deviation and computes the compressed representation $\boldsymbol{z}^t$(line 6).
- It then computes the K-nearest neighbours $(\hat{\boldsymbol{z}}_1^t, \hat{\boldsymbol{z}}_2^t, \cdots, \hat{\boldsymbol{z}}_K^t)$ by querying the memory (line 8), and calculates their L1 distance with $\boldsymbol{z}^t$ (line 10).

MemStream: Algorithm

Algorithm 1: MemStream

Input: Stream of data records
Output: Anomaly scores for each record

1 ▷ **Initialization**
2 Feature Extractor, f_θ, trained using small subset of data $\mathcal{D}$
3 Memory, M, initialized as $f_\theta(\mathcal{D})$
4 **while** *new sample* $\mathbf{x}^t$ *is received:* **do**
5 ▷ **Extract features:**
6 $\mathbf{z}^t = f_\theta(\mathbf{x}^t)$
7 ▷ **Query memory:**
8 $\{\hat{\mathbf{z}}_1^t, \hat{\mathbf{z}}_2^t ... \hat{\mathbf{z}}_K^t\} = K$-nearest neighbours of $\mathbf{z}^t$ in M
9 ▷ **Calculate distance:**
10 $R(\mathbf{z}^t, \hat{\mathbf{z}}_i^t) = ||\mathbf{z}^t - \hat{\mathbf{z}}_i^t||_1$ **for all** $i \in 1..K$
11 ▷ **Assign discounted score:**
12 $$Score(\mathbf{z}^t) = \frac{\sum_{i=1}^{K} \gamma^{i-1} R(\mathbf{z}^t, \hat{\mathbf{z}}_i^t)}{\sum_{i=1}^{K} \gamma^{i-1}}$$
13 ▷ **Update Memory:**
14 **if** $Score(\mathbf{z}^t) < \beta$ **then**
15 Replace earliest added element in $\boldsymbol{M}$ with $\mathbf{z}^t$
16 ▷ **Anomaly Score:**
17 **output** $Score(\mathbf{z}^t)$

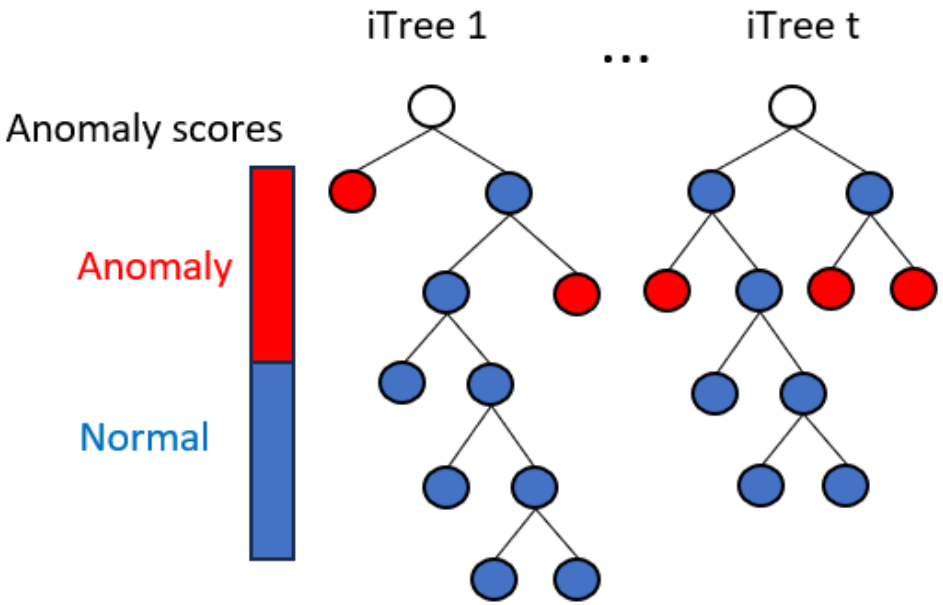
- The final discounted score is calculated as an exponentially weighted average (weighting factor γ) (line 12). This helps in making the autoencoder more robust.
- The discounted score is then compared against a user-defined threshold β (line 14) and the new record is updated into the memory in a FIFO manner if the score falls within β (line 15). This step ensures that anomalous records do not enter the memory. If the memory is updated, then the stored mean and standard deviation are also updated accordingly.
- The discounted score is returned as the anomaly score for the record $\boldsymbol{x}^t$ (line 17).

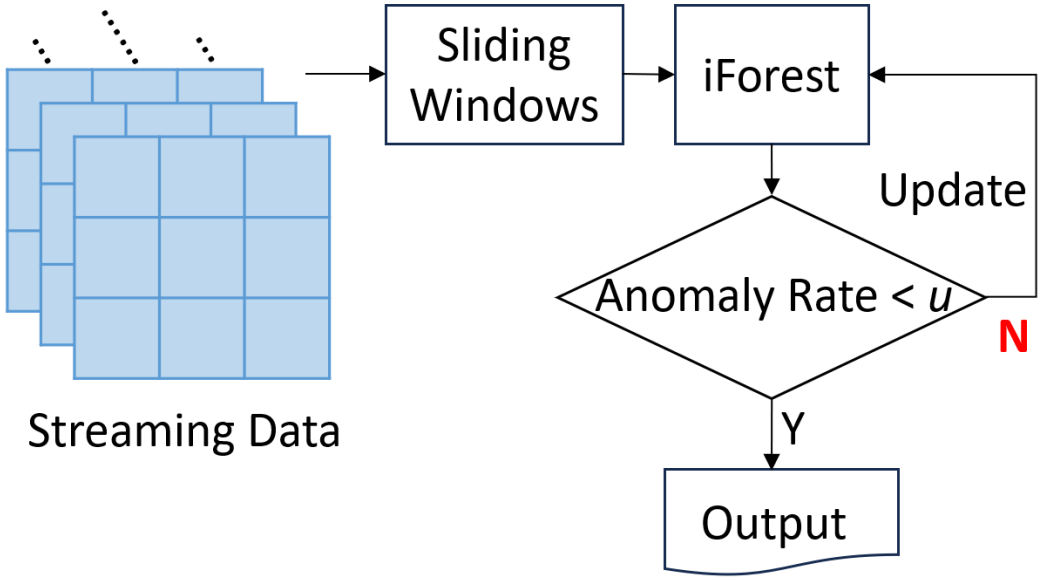
Methods

- **STORM**: Uses a sliding window to detect global distance-based outliers in data streams with respect to the current window.
- **HS-Tree**: Uses an ensemble of randomly constructed half-space trees with a sliding window to detect anomalies in evolving streaming data.
- **IForestASD**: Uses a sliding window frame scheme to handle abnormal data.
- **RSHash**: Uses subspace grids and randomized hashing in an ensemble to detect anomalies. For each model in the ensemble, a grid is constructed using subsets of features and data, random hashing is used to record data counts in grid cells, and the anomaly score of a data point is the log of the frequency in its hashed bins.
- **RCF**: Tries to further improve upon IF by creating multiple random cuts (trees) of data and constructing a forest of such trees to determine whether a point is anomalous or not.
- **LODA**: Generates several weak anomaly detectors by producing many random projections of the data and then computing a density estimation histogram for each projection. The outlier scores produced are the mean negative log-likelihood according to each histogram for each point.
- **Kitsune**: Is an ensemble of light-weight autoencoders for real-time anomaly detection.

Methods

- **IForestASD**: Uses a sliding window frame scheme to handle abnormal data.
- Images from here





Datasets

- **KDDCUP99** is amongst the most extensively used datasets for multi-aspect anomaly detection. The original dataset contains samples of 41 dimensions, 34 of which are continuous and 7 are categorical, and also displays concept drift. Using one-hot representation to encode the categorical features, we obtain a dataset of 121 dimensions. 20% of data samples are labeled as *normal* and 80% are labeled as *attack* (normal ones are anomalous).
- **NSL-KDD** solves some of the inherent problems of the KDDCUP99 dataset such as redundant and duplicate records and is considered more enhanced as compared to KDDCUP99.
- **CICIDS-DoS** was created by the Canadian Institute of Cybersecurity. Each record is a flow containing features such as Source IP Address, Source Port, Destination IP Address, Bytes, Packets. These flows were captured from a real-time simulation of normal network traffic and synthetic attack simulators. This consists of the CICIDS-DoS dataset (1.05 million records). CICIDS-DoS has 5% anomalies and contains samples of 95 dimensions with a mixture of numeric and categorical features.

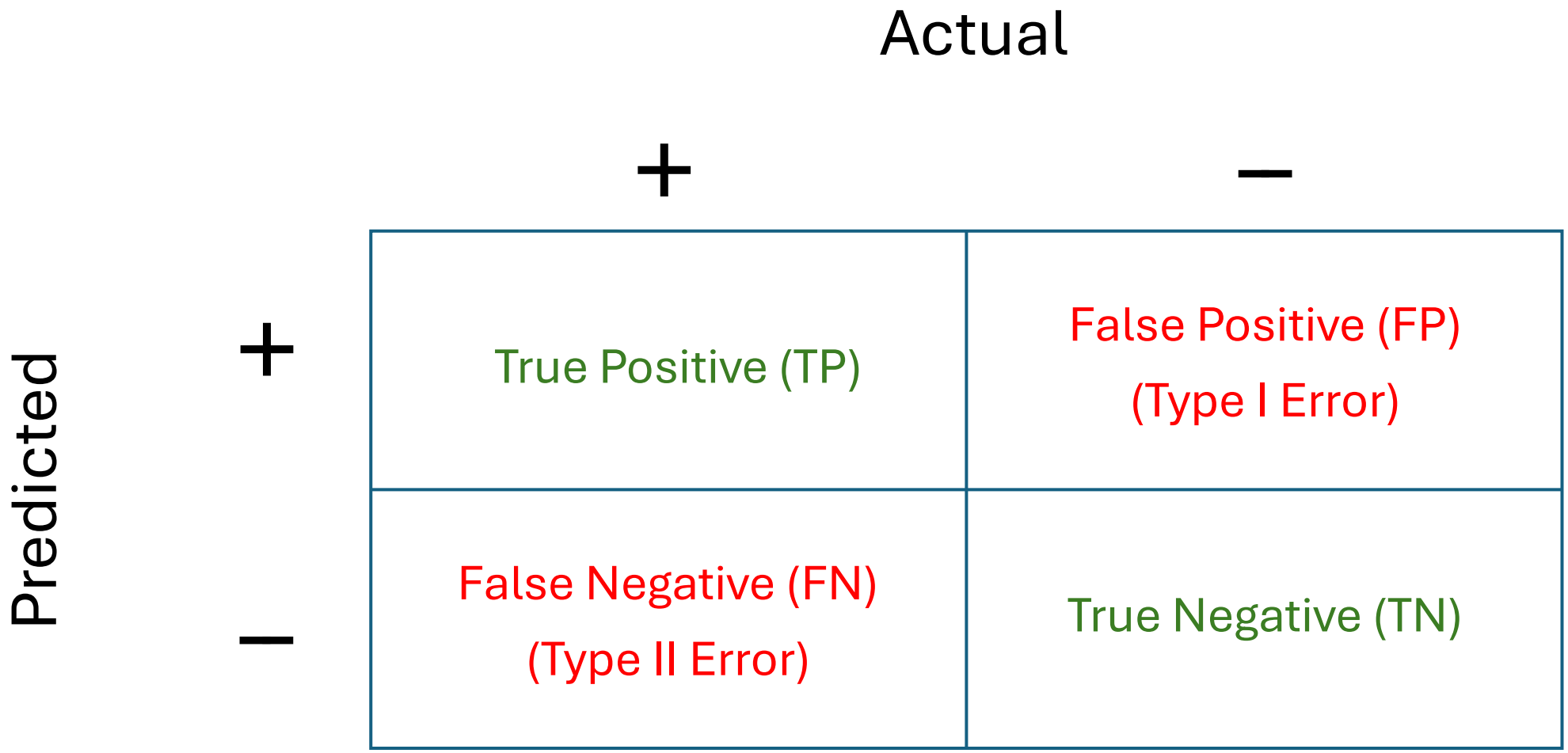
Datasets

- **UNSW-NB15** was created by the Cyber Range Lab of the Australian Centre for Cyber Security (ACCS) for generating a hybrid of real modern normal activities and synthetic contemporary attack behaviors. This dataset has nine types of attacks, namely, Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, and Worms. Also, it has 13% anomalies.
- **Syn**: 10% anomalies and T = 10000 samples. This dataset is constructed as a superposition of a linear wave with slope 2×10^3 , two sinusoidal waves with time periods 0.2T and 0.3T and amplitudes 8 and 4, altogether with an additive Gaussian noise from a standard normal distribution. 10% of the samples are chosen at random and are perturbed with uniform random noise from the interval [3, 6] to simulate anomalous data.
- **Ionosphere** is a binary classification dataset with dimensionality 33. The *bad* class is considered as outliers class and the *good* class as inliers.
- **Cardio** consists of measurements of fetal heart rate (FHR) and uterine contraction (UC) features on cardiotocograms classified by expert obstetricians. This is a classification dataset, where the classes are normal, suspect, and pathologic. For outlier detection, the normal class formed the inliers, while the pathologic (outlier) class is downsampled to 176 points. The suspect class is discarded.

Datasets

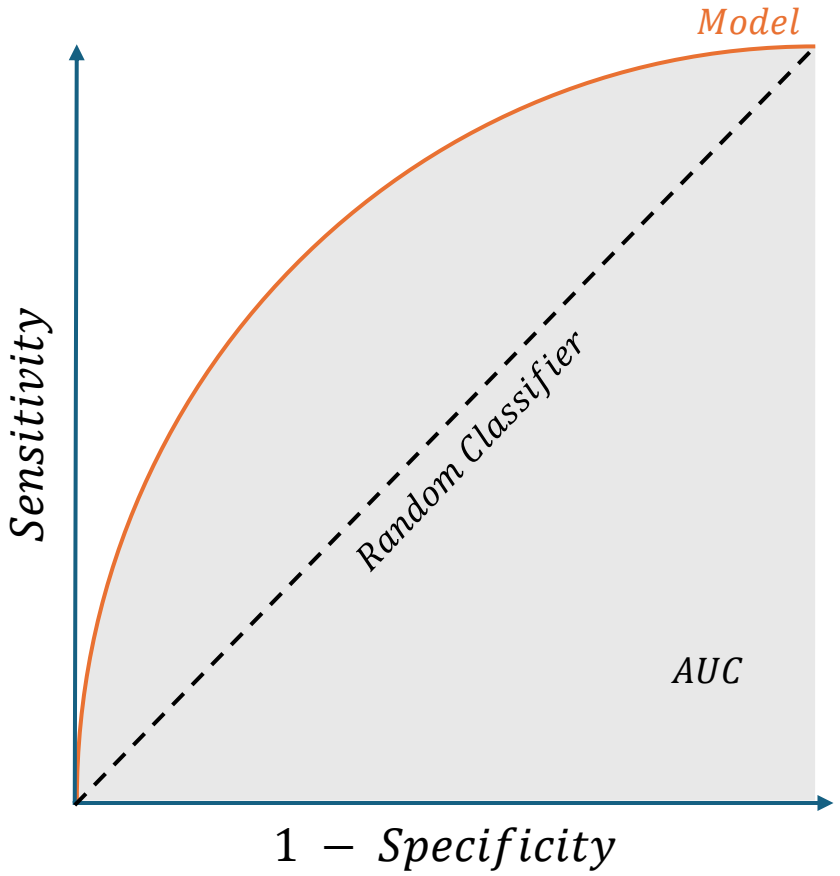
- **Satellite** is a multiclass classification dataset. The smallest three classes, i.e. 2, 4, 5 are combined to form the outliers class, while all the other classes are combined to form an inlier class.
- **Satimage-2** is also a multi-class classification dataset. Class 2 is down-sampled to 71 outliers, while all the other classes are combined to form an inlier class.
- **Mammography** has 11,183 samples with 260 calcifications. For outlier detection, the minority class of calcification is considered as the outlier class and the non-calcification class as inliers.
- **Pima** is a binary classification dataset. Several constraints were placed on the selection of instances from a larger database. In particular, all patients here are females at least 21 years old of Pima Indian heritage.
- **ForestCover** is a multiclass classification dataset. It is used in predicting forest cover type from cartographic variables only. This dataset has 54 attributes (10 quantitative variables, 4 binary wilderness areas, and 40 binary soil type variables). Here, an outlier detection dataset is created using only 10 quantitative attributes. Instances from class 2 are considered as normal points and instances from class 4 are anomalies. The anomalies ratio is 0.9%. Instances from the other classes are omitted.

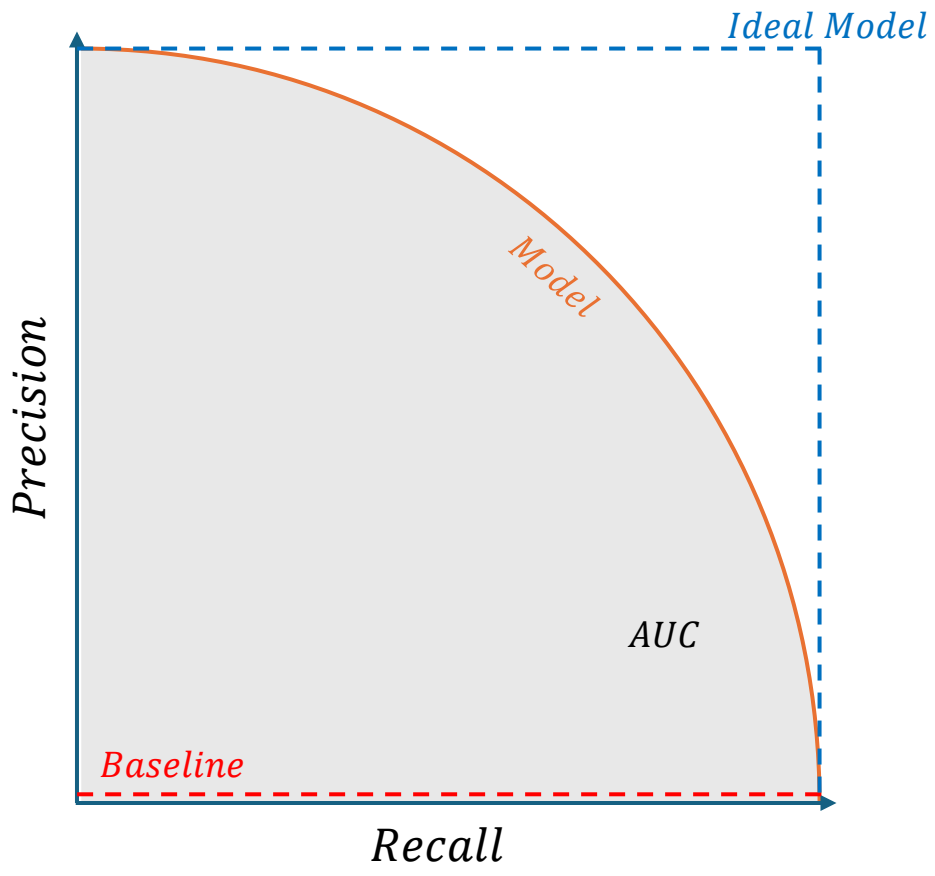
Evaluation Metrics – Confusion Matrix



Evaluation Metrics – ROC and PR Curve

$$Precision = \frac{TP}{TP + FP} \qquad Sensitivity\ or\ Recall\ = \frac{TP}{TP + FN} \qquad Specificity\ = \frac{TN}{TP + FP}$$





Experiments

- All datasets were used.
- Stratified sample of n=10,000 based on the target variable (only if records were greater).
- It was used all the baseline params for each method, as well each MemStream setup for each dataset, according to the values presented in the paper by the authors.
- Some methods from the article were ommited in this presentation, like DILOF, xStream, Mstream and Extendend Isolation Forest.

Shape of the Datasets (original)

	KDD99	NSL	UNSW	DoS	Syn.	Ion.	Cardio	Sat.	Sat.-2	Mamm.	Pima	Cover
Records	494,021	125,973	2,540,044	1,048,575	10,000	351	1,831	6,435	5,803	11,183	768	286,048
Dimensions	121	123	122	95	1	33	21	36	36	6	8	10

Experiments - Params

- **STORM**: *window_size=10000, max_radius=0.1*
- **HS-Tree**: *window_size=100, num_trees=25, max_depth=15, initial_window_X=None*
- **IForestASD**: *window_size=100, n_estimators=25, anomaly_threshold=0.5, drift_threshold=0.5*
- **RSHash**: *sampling_points=1000, decay=0.015, num_components=100, num_hash_fns=1*
- **RCF**: *num_trees=4, shingle_size=4, tree_size=256*
- **LODA**: *num_bins=10, num_random_cuts=100*
- **Kitsune**: *max_size_ae=10, learning_rate_ae=10, hidden_ratio=0.75, grace_feature_mapping= 10% of data, grace_anomaly_detector=10% of data.*

MemStream Params for each Dataset

	KDD99	NSL	UNSW	DoS	Syn.	Ion.	Cardio	Sat.	Sat.-2	Mamm.	Pima	Cover
memlen	256	2,048	2,048	2,048	16	4	64	32	256	128	64	2,048
beta	1	0.1	0.1	0.1	1	0.001	1	0.1	10	0.1	0.001	0.0001

Results – Table 2

AUC-ROC of MemStream and Streaming Baselines (datasets: sample)

Method	KDD99		NSL		UNSW		DoS		Syn		Ion.		Cardio		Sat.		Sat.-2		Mamm.		Pima		Cover	
	Article	Reproduced	Article	Reproduced	Article	Reproduced	Article	Reproduced	Article	Reproduced	Article	Reproduced	Article	Reproduced	Article	Reproduced	Article	Reproduced	Article	Reproduced	Article	Reproduced	Article	Reproduced
STORM	0.914	0.137	0.504	0.509	0.810	0.514	0.511	0.489	0.910	0.423	0.637	0.502	0.507	0.506	0.662	0.500	0.514	0.514	0.650	0.292	0.528	0.500	0.778	0.500
HS-Tree	0.912	0.777	0.845	0.858	0.769	0.282	0.707	0.605	0.800	0.800	0.764	0.780	0.673	0.624	0.519	0.511	0.929	0.914	0.832	0.846	0.667	0.654	0.731	0.745
iForest ASD	0.575	0.836	0.500	0.649	0.557	0.463	0.529	0.624	0.501	0.830	0.694	0.812	0.515	0.685	0.504	0.668	0.554	0.950	0.574	**0.856**	0.525	0.669	0.603	0.848
RS-Hash	0.859	0.067	0.701	0.607	0.778	0.692	0.527	0.335	0.921	0.079	0.772	0.246	0.532	0.425	0.675	0.310	0.685	0.244	0.773	0.143	0.562	0.445	0.640	0.384
RCF	0.791	0.954	0.745	0.194	0.512	0.601	0.514	0.835	0.774	0.781	0.675	0.664	0.617	0.627	0.552	0.547	0.738	0.668	0.755	0.833	0.571	0.566	0.586	0.787
LODA	0.500	0.500	0.500	0.500	---	0.500	0.500	0.500	0.506	0.490	0.503	0.500	0.501	0.499	0.500	0.500	0.500	0.500	0.500	0.500	0.502	0.502	0.500	0.500
Kitsune	0.525	0.860	0.659	0.719	0.794	0.437	0.907	0.612	---	---	0.514	0.485	**0.966**	**0.966**	0.665	0.335	0.973	0.973	0.592	0.691	0.511	0.439	0.888	0.701
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Mem Stream	**0.980**	**0.987**	**0.978**	**0.987**	**0.972**	**0.927**	**0.938**	**0.905**	**0.955**	**0.850**	**0.955**	**0.818**	0.884	0.872	**0.727**	0.724	0.991	**0.995**	**0.894**	0.464	**0.742**	**0.741**	**0.952**	**0.970**

Results – Table 3

AUC-PR and Time Required to run MemStream and Streaming Baselines on NSL-KDD (full)

Method	AUC-PR		Time (s)	
	Article	Reproduced	Article	Reproduced
STORM	0.681	0.479	754	1,401
HS-Tree	0.709	0.771	306	1,128
iForestASD	0.534	0.612	19,876	522
RS-Hash	0.500	0.700	892	579
RCF	0.664	0.356	665	422
LODA	0.734	0.465	2,617	1,632
Kitsune	0.673	0.673	821	700
⋮	⋮	⋮	⋮	⋮
MemStream	**0.959**	**0.957**	**55**	**218**

Results – Table 4

Effect of Memory Size on the AUC-ROC in MemStream in NSL-KDD (full)

Memory Size	AUC-ROC	
	Article	Reproduced
2^3	---	0.920
2^4	0.670	0.653
2^5	0.649	0.603
2^6	0.932	0.889
2^7	0.936	0.934
2^8	0.923	0.934

Memory Size	AUC-PR	
	Article	Reproduced
2^9	0.950	0.955
2^{10}	0.972	0.972
2^{11}	0.976	0.978
2^{12}	0.985	0.986
2^{13}	0.989	0.987
2^{14}	0.991	0.991

Results

- **Table 2:** MemStream outperforms the different streaming methods for most of the datasets tested. However, the value of AUC-ROC reproduced for Mammography dataset was lower than the value computed on the original paper. Also, there were discrepancies for all other methods tested (except LODA).
- **Table 3:** Similar to Table 2, the value of AUC-PR has discrepancies for most methods tested. Also, MemStream outperforms including on the computation time. Also, it is notable the difference on time for iForestASD that this reproduction was way faster than the time reported on the original paper.
- **Table 4:** Increasing the memory size of MemStream for the NSL-Dataset has a tendency to improve the values of AUC-ROC. The reproduced values were very close to the ones computed on the original paper. Also, the new memory length tested, 2^3, had a very good value of AUC-ROC.

Conclusion

- The paper’s conclusion that MemStream outperforms most methods was reproduced and confirmed by this work.
- It was not possible to reproduce the exact values reported:
  - The repository available for MemStream has the SEED used for the experiments, but it didn’t match the values of all the methods used.
  - Also, there isn’t a repository of all the experiments for all the methods tested.
- Using a sample of data can be an option to approximate the real metric value for all data, including the possibility to run the experiment for methods that couldn’t support the size of the dataset (e.g.: LODA with dataset UNSW).

References

- Autoencoders: https://ff12.fastforwardlabs.com/
- ConceptDrift: https://www.evidentlyai.com/ml-in-production/concept-drift
- Isolation Forest: https://arxiv.org/html/2403.10802v1

- Original Paper: https://dl.acm.org/doi/pdf/10.1145/3485447.3512221
- Paper GitHub: https://github.com/Stream-AD/MemStream?tab=readme-ov-file

- Reproduction GitHub: https://github.com/nmhahn/sdsp_memstream