# FTP @ DEAKIN UNIVERSITY
# STREAMING ANOMALY DETECTION


## OMKAR BHANDARE (22CS30016)

**Email - omkaranu04@kgpian.iitkgp.ac.in**


**PAPER SELECTED: 3485447.3512221 (acm.org) (MemStream)**

### UNDERSTANDING THE ALGORITHM:

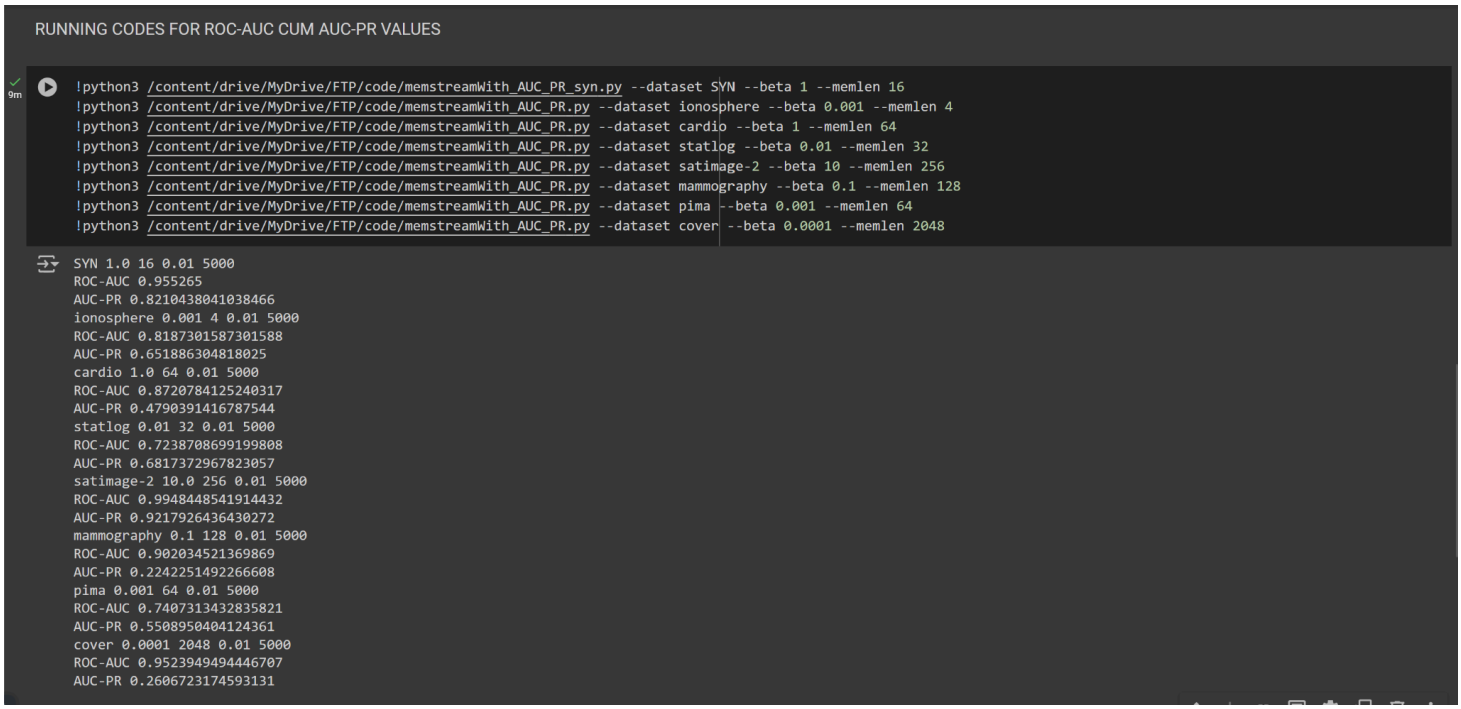
MemStream is an online anomaly detection algorithm that handles data streams by maintaining a dynamic memory of normal patterns. It employs an autoencoder to encode input data into a latent representation and reconstruct it. The difference between the input and its reconstruction is used to detect anomalies. The algorithm continuously updates its memory with new data points if their reconstruction error is below a defined threshold. This allows MemStream to adapt to changes in the data distribution over time while ensuring it effectively distinguishes between normal and anomalous data points.

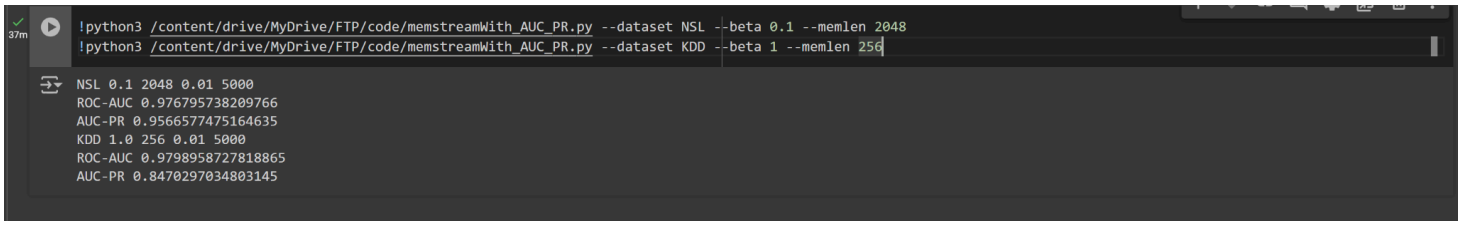
### RECREATING THE RESULTS:

Following are the results for the MemStream Algorithm along with some other algorithms on various mentioned Open Source Datasets, represented in the form of a table in the paper :

| **Method** | KDD99 | NSL | UNSW | DoS | Syn. | Ion. | Cardio | Sat. | Sat.-2 | Mamm. | Pima | Cover |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| STORM (CIKM'07) | 0.914 | 0.504 | 0.810 | 0.511 | 0.910 | 0.637 | 0.507 | 0.662 | 0.514 | 0.650 | 0.528 | 0.778 |
| HS-Tree (IJCAI'11) | 0.912 | 0.845 | 0.769 | 0.707 | 0.800 | 0.764 | 0.673 | 0.519 | 0.929 | 0.832 | 0.667 | 0.731 |
| iForestASD (ICONS'13) | 0.575 | 0.500 | 0.557 | 0.529 | 0.501 | 0.694 | 0.515 | 0.504 | 0.554 | 0.574 | 0.525 | 0.603 |
| RS-Hash (ICDM'16) | 0.859 | 0.701 | 0.778 | 0.527 | 0.921 | 0.772 | 0.532 | 0.675 | 0.685 | 0.773 | 0.562 | 0.640 |
| RCF (ICML'16) | 0.791 | 0.745 | 0.512 | 0.514 | 0.774 | 0.675 | 0.617 | 0.552 | 0.738 | 0.755 | 0.571 | 0.586 |
| LODA (ML'16) | 0.500 | 0.500 | – – – | 0.500 | 0.506 | 0.503 | 0.501 | 0.500 | 0.500 | 0.500 | 0.502 | 0.500 |
| Kitsune (NDSS'18) | 0.525 | 0.659 | 0.794 | 0.907 | – – – | 0.514 | 0.966 | 0.665 | 0.973 | 0.592 | 0.511 | 0.888 |
| DILOF (KDD'18) | 0.535 | 0.821 | 0.737 | 0.613 | 0.703 | **0.928** | 0.570 | 0.561 | 0.563 | 0.733 | 0.543 | 0.688 |
| xSTREAM (KDD'18) | 0.957 | 0.552 | 0.804 | 0.800 | 0.539 | 0.847 | 0.918 | 0.677 | **0.996** | 0.856 | 0.663 | 0.894 |
| MSTREAM (WWW'21) | 0.844 | 0.544 | 0.860 | 0.930 | 0.505 | 0.670 | **0.986** | 0.563 | 0.958 | 0.567 | 0.529 | 0.874 |
| Ex. IF (TKDE'21) | 0.874 | 0.767 | 0.541 | 0.734 | – – – | 0.872 | 0.921 | 0.716 | 0.995 | 0.867 | 0.672 | 0.902 |
| **MEMSTREAM** | **0.980** | **0.978** | **0.972** | **0.938** | **0.955** | 0.821 | 0.884 | **0.727** | 0.991 | **0.894** | **0.742** | **0.952** |

The source code of the MemStream algorithm has been implemented in https://github.com/Stream-AD/MemStream as mentioned in the paper.
Google Colab was chosen as the testing platform.
The AUC-PR values were implemented as instructed in average_precision_score — scikit-learn 1.5.0 documentation
Here are some screenshots of the recreated results:




## A SHORT OVER-VIEW ON 'ROC-AUC' AND 'ROC-PR' VALUES:

**ROC-AUC (Receiver Operating Characteristic - Area Under the Curve)**

The ROC curve is a plot of the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings.

A higher ROC-AUC value indicates a better-performing model.

**AUC-PR (Precision-Recall - Area Under the Curve)**

The PR curve is a plot of precision (positive predictive value) against recall (sensitivity) at various threshold settings.

A higher AUC-PR value indicates better performance, which is especially useful for imbalanced datasets.

A good stream anomaly detection algorithm typically aims high ROC-AUC and AUC-PR values, indicating both high true positive rates and precision:

ROC-AUC values >0.8 are desirable

AUC-PR values >0.7 are desirable, especially in imbalanced datasets where anomalies are rare

**COMPARISON WITH OTHER MODELS:**

For the comparison with different models, from the pyod source as in https://github.com/yzhao062/pyod , I used the following models:

1. KDE (Outlier Detection with Kernel Density Functions)
2. PCA (Principal Component Analysis)
3. KPCA (Kernal Principal Component Analysis)
4. LOF (Local Outlier Factor)
5. CBLOF (Clustering Based Local Outlier Factor)
6. HBOS (Histogram Based Outlier Score)
7. kNN (k Nearest Neighbours)
8. IForest (Isolation Forest)
9. LODA (Lightweight On-line Detector of Anomalies)
10. LUNAR (Unifying Local Outlier Detection Methods via Graph Neural Networks)

Here are some screenshots of the results:

```
Dataset: cardio, Algorithm: KDE, ROC-AUC: 0.7484, PR-AUC: 0.2364
Dataset: cardio, Algorithm: PCA, ROC-AUC: 0.9500, PR-AUC: 0.6084
Dataset: cardio, Algorithm: KPCA, ROC-AUC: 0.6037, PR-AUC: 0.1703
Dataset: cardio, Algorithm: LOF, ROC-AUC: 0.5458, PR-AUC: 0.1519
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:87
  warnings.warn(
Dataset: cardio, Algorithm: CBLOF, ROC-AUC: 0.8177, PR-AUC: 0.4780
Dataset: cardio, Algorithm: HBOS, ROC-AUC: 0.8377, PR-AUC: 0.4479
Dataset: cardio, Algorithm: kNN, ROC-AUC: 0.6861, PR-AUC: 0.2809
Dataset: cardio, Algorithm: IForest, ROC-AUC: 0.9232, PR-AUC: 0.5686
Dataset: cardio, Algorithm: LODA, ROC-AUC: 0.9348, PR-AUC: 0.5503
Dataset: cardio, Algorithm: LUNAR, ROC-AUC: 0.5394, PR-AUC: 0.1534
```

```
Dataset: ionosphere, Algorithm: KDE, ROC-AUC: 0.9229, PR-AUC: 0.9044
Dataset: ionosphere, Algorithm: PCA, ROC-AUC: 0.7947, PR-AUC: 0.7461
Dataset: ionosphere, Algorithm: KPCA, ROC-AUC: 0.4501, PR-AUC: 0.3494
Dataset: ionosphere, Algorithm: LOF, ROC-AUC: 0.8718, PR-AUC: 0.8252
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: F
  warnings.warn(
Dataset: ionosphere, Algorithm: CBLOF, ROC-AUC: 0.9098, PR-AUC: 0.9009
Dataset: ionosphere, Algorithm: HBOS, ROC-AUC: 0.6520, PR-AUC: 0.4140
Dataset: ionosphere, Algorithm: kNN, ROC-AUC: 0.9303, PR-AUC: 0.9333
Dataset: ionosphere, Algorithm: IForest, ROC-AUC: 0.8527, PR-AUC: 0.8070
Dataset: ionosphere, Algorithm: LODA, ROC-AUC: 0.8101, PR-AUC: 0.7515
Dataset: ionosphere, Algorithm: LUNAR, ROC-AUC: 0.9310, PR-AUC: 0.9338
```

```
Dataset: mammography, Algorithm: KDE, ROC-AUC: 0.8650, PR-AUC: 0.1938
Dataset: mammography, Algorithm: PCA, ROC-AUC: 0.8863, PR-AUC: 0.1959
Dataset: mammography, Algorithm: KPCA, ROC-AUC: 0.5399, PR-AUC: 0.0317
Dataset: mammography, Algorithm: LOF, ROC-AUC: 0.7193, PR-AUC: 0.0881
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: Fu
  warnings.warn(
Dataset: mammography, Algorithm: CBLOF, ROC-AUC: 0.8158, PR-AUC: 0.1444
Dataset: mammography, Algorithm: HBOS, ROC-AUC: 0.8299, PR-AUC: 0.1232
Dataset: mammography, Algorithm: kNN, ROC-AUC: 0.8378, PR-AUC: 0.1547
Dataset: mammography, Algorithm: IForest, ROC-AUC: 0.8675, PR-AUC: 0.2287
Dataset: mammography, Algorithm: LODA, ROC-AUC: 0.8500, PR-AUC: 0.1981
Dataset: mammography, Algorithm: LUNAR, ROC-AUC: 0.8392, PR-AUC: 0.1519
```

```
Dataset: pima, Algorithm: KDE, ROC-AUC: 0.5610, PR-AUC: 0.5940
Dataset: pima, Algorithm: PCA, ROC-AUC: 0.6322, PR-AUC: 0.4632
Dataset: pima, Algorithm: KPCA, ROC-AUC: 0.4836, PR-AUC: 0.3221
Dataset: pima, Algorithm: LOF, ROC-AUC: 0.5384, PR-AUC: 0.3666
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:87
  warnings.warn(
Dataset: pima, Algorithm: CBLOF, ROC-AUC: 0.5723, PR-AUC: 0.4334
Dataset: pima, Algorithm: HBOS, ROC-AUC: 0.6858, PR-AUC: 0.5070
Dataset: pima, Algorithm: kNN, ROC-AUC: 0.6076, PR-AUC: 0.4564
Dataset: pima, Algorithm: IForest, ROC-AUC: 0.6989, PR-AUC: 0.5172
Dataset: pima, Algorithm: LODA, ROC-AUC: 0.6134, PR-AUC: 0.4071
Dataset: pima, Algorithm: LUNAR, ROC-AUC: 0.6988, PR-AUC: 0.5197
```

```
Dataset: satimage-2, Algorithm: KDE, ROC-AUC: 0.6313, PR-AUC: 0.4323
Dataset: satimage-2, Algorithm: PCA, ROC-AUC: 0.9772, PR-AUC: 0.8721
Dataset: satimage-2, Algorithm: KPCA, ROC-AUC: 0.4817, PR-AUC: 0.0107
Dataset: satimage-2, Algorithm: LOF, ROC-AUC: 0.5326, PR-AUC: 0.0302
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: F
  warnings.warn(
Dataset: satimage-2, Algorithm: CBLOF, ROC-AUC: 0.9986, PR-AUC: 0.9713
Dataset: satimage-2, Algorithm: HBOS, ROC-AUC: 0.9716, PR-AUC: 0.7150
Dataset: satimage-2, Algorithm: kNN, ROC-AUC: 0.9296, PR-AUC: 0.3386
Dataset: satimage-2, Algorithm: IForest, ROC-AUC: 0.9919, PR-AUC: 0.9134
Dataset: satimage-2, Algorithm: LODA, ROC-AUC: 0.9932, PR-AUC: 0.9557
Dataset: satimage-2, Algorithm: LUNAR, ROC-AUC: 0.6635, PR-AUC: 0.0667
```

```
Dataset: statlog, Algorithm: KDE, ROC-AUC: 0.5632, PR-AUC: 0.5620
Dataset: statlog, Algorithm: PCA, ROC-AUC: 0.6012, PR-AUC: 0.6057
Dataset: statlog, Algorithm: KPCA, ROC-AUC: 0.4598, PR-AUC: 0.2911
Dataset: statlog, Algorithm: LOF, ROC-AUC: 0.5395, PR-AUC: 0.3715
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870
  warnings.warn(
Dataset: statlog, Algorithm: CBLOF, ROC-AUC: 0.7310, PR-AUC: 0.6838
Dataset: statlog, Algorithm: HBOS, ROC-AUC: 0.7659, PR-AUC: 0.6935
Dataset: statlog, Algorithm: kNN, ROC-AUC: 0.6701, PR-AUC: 0.5310
Dataset: statlog, Algorithm: IForest, ROC-AUC: 0.7198, PR-AUC: 0.6533
Dataset: statlog, Algorithm: LODA, ROC-AUC: 0.6047, PR-AUC: 0.5947
Dataset: statlog, Algorithm: LUNAR, ROC-AUC: 0.6353, PR-AUC: 0.4833
```

```
Dataset: SYN, Algorithm: KDE, ROC-AUC: 0.5502, PR-AUC: 0.1930
Dataset: SYN, Algorithm: PCA, ROC-AUC: 0.5183, PR-AUC: 0.1549
Dataset: SYN, Algorithm: KPCA, ROC-AUC: 0.5519, PR-AUC: 0.1860
Dataset: SYN, Algorithm: LOF, ROC-AUC: 0.5351, PR-AUC: 0.1394
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:
  warnings.warn(
Dataset: SYN, Algorithm: CBLOF, ROC-AUC: 0.5449, PR-AUC: 0.1561
Dataset: SYN, Algorithm: HBOS, ROC-AUC: 0.5497, PR-AUC: 0.1800
Dataset: SYN, Algorithm: kNN, ROC-AUC: 0.5472, PR-AUC: 0.1909
Dataset: SYN, Algorithm: IForest, ROC-AUC: 0.5488, PR-AUC: 0.1920
Dataset: SYN, Algorithm: LODA, ROC-AUC: 0.5320, PR-AUC: 0.1763
Dataset: SYN, Algorithm: LUNAR, ROC-AUC: 0.5343, PR-AUC: 0.1400
```