Local Outlier Factor Algorithm for Anomaly Detection in HPCC Systems

Arya Adesh[1], *Jyoti Shetty*[1], *Shobha G*[1], *Lili Xu*[2]
(1) RV College of Engineering , (2) LexisNexis Risk Solutions

About the project

Anomaly detection is the process of identifying unexpected abnormalities in a dataset. Local outlier factor (LOF) is a density-based method for anomaly detection where a degree of outlier-ness is assigned to every point by comparing with its neighbors. LOF can identify both Local and global outliers while other anomaly detection algorithms can find only Global Outliers. It is unsupervised and can identify anomalies even in a skewed distribution.

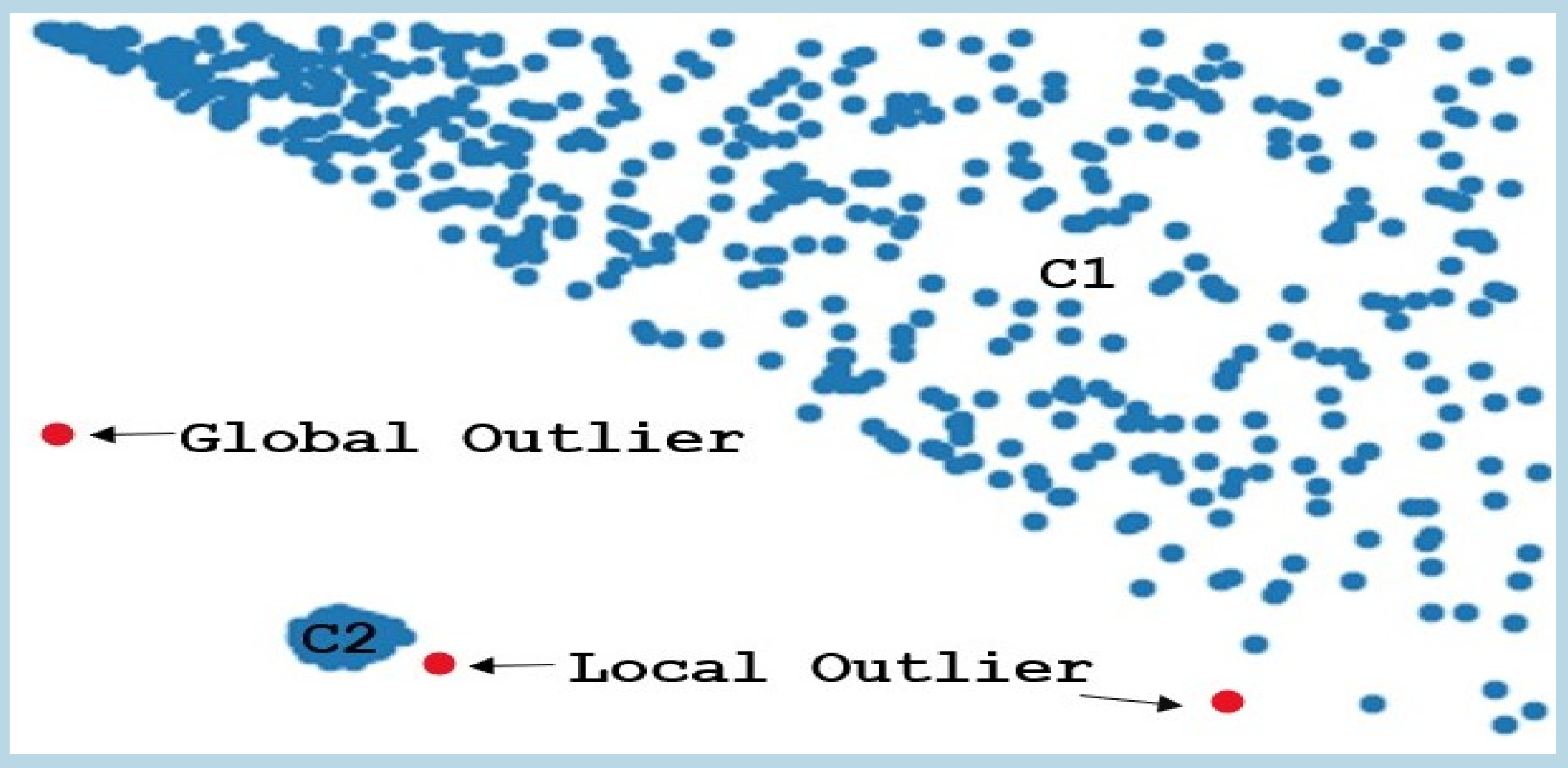

Figure 1: Types of anomalies detected by LOF

Testing with Dataset

LOF implemented in ECL was tested on various datasets. One of the datasets is: **Credit Card Fraud detection** dataset. It contains numerical input variables resulting from a PCA transformation on data from credit card transactions. It is highly imbalanced with 0.172% transactions labeled as fraudulent.

Normal	Anomalies
284,315	492

Comparison of anomaly detection algorithms

Isolated Forest (IF) and One-class SVM (1SVM) are other unsupervised Anomaly Detection algorithms, implemented in Python. They were compared with LOF algorithm implemented in ECL, based on results:

Name	Accuracy	Precision	Recall	F1
LOF	0.9965	0.1160	0.1158	0.1161
IF	0.9975	0.2703	0.2801	0.2723
1SVM	0.7046	0.0012	0.3400	0.0010

LOF performed better than One-class SVM. IF performed better than other anomaly detection algorithms.

Following results were observed on oversampled dataset where the minority class was augmented by SMOTE.

Name	Accuracy	Precision	Recall	F1
LOF	0.49884	0.1646	0.0006	0.0014
IF	0.49826	0.0120	0.0001	0.0001
1SVM	0.30231	0.0001	0.0001	0.0001

LOF performed slightly better than other algorithms. Anomalies should be rare occurrences in the dataset.

Future Work

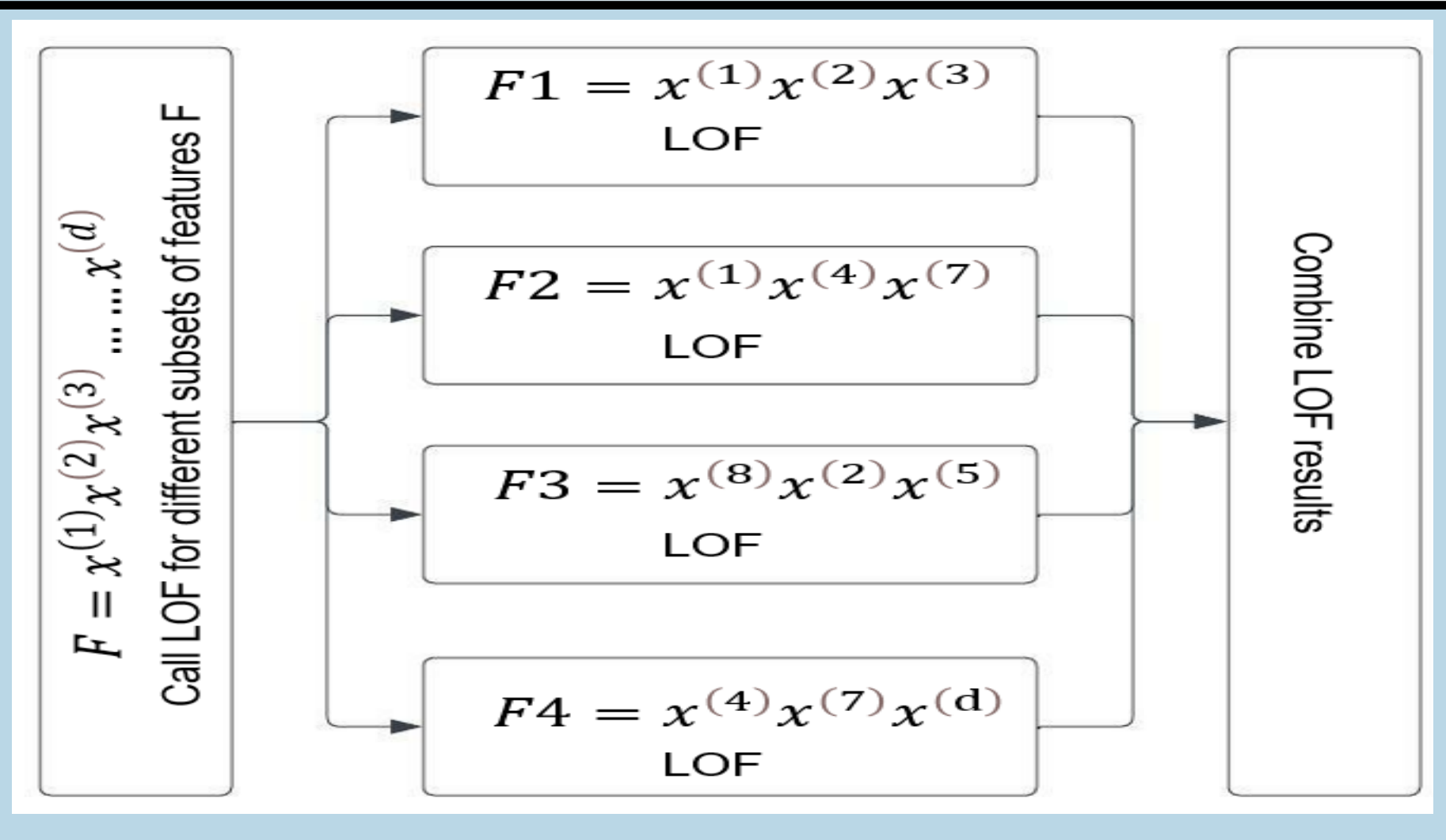
$F = x^{(1)}x^{(2)}x^{(3)}\ldots\ldots x^{(d)}$
Call LOF for different subsets of features F
$F1 = x^{(1)}x^{(2)}x^{(3)}$
LOF
$F2 = x^{(1)}x^{(4)}x^{(7)}$
LOF
$F3 = x^{(8)}x^{(2)}x^{(5)}$
LOF
$F4 = x^{(4)}x^{(7)}x^{(d)}$
LOF
Combine LOF results

Figure 2: Feature bagging for LOF

1) Feature bagging for improving performance
2) Hyperparameter tuning to improve selection of parameters for higher precision and recall.
3) Combine LOF with other anomaly detection algorithms to improve increase precision

Performance dependence on Hyperparameters

LOF algorithm uses two hyperparameters: neighborhood size(here ***neighbors***) and ***contamination***. *contamination* is the proportion of data points in the dataset to be predicted as anomalies. *Neighbors* indicates the number of nearest neighbor points,whose density should be considered to find the LOF value of each point. Data points with high LOF value have sparse neighborhoods and hence represent strong outliers. Outliers usually have LOF value of more than 1. Changing the value of *neighbors* alters LOF value of every point. However, changing *contamination* doesn't alter LOF values but only changes the threshold value for anomaly detection.

$$\forall\ p\ in\ dataset\ D,\ Sp = \{q|\ q\ \in\ neighborhood\ of\ p\},\ LOF(p) = \frac{\sum_{q\in Sp}\frac{Local\ reachability\ density(q)}{Local\ reachability\ density(p)}}{neighbors}$$

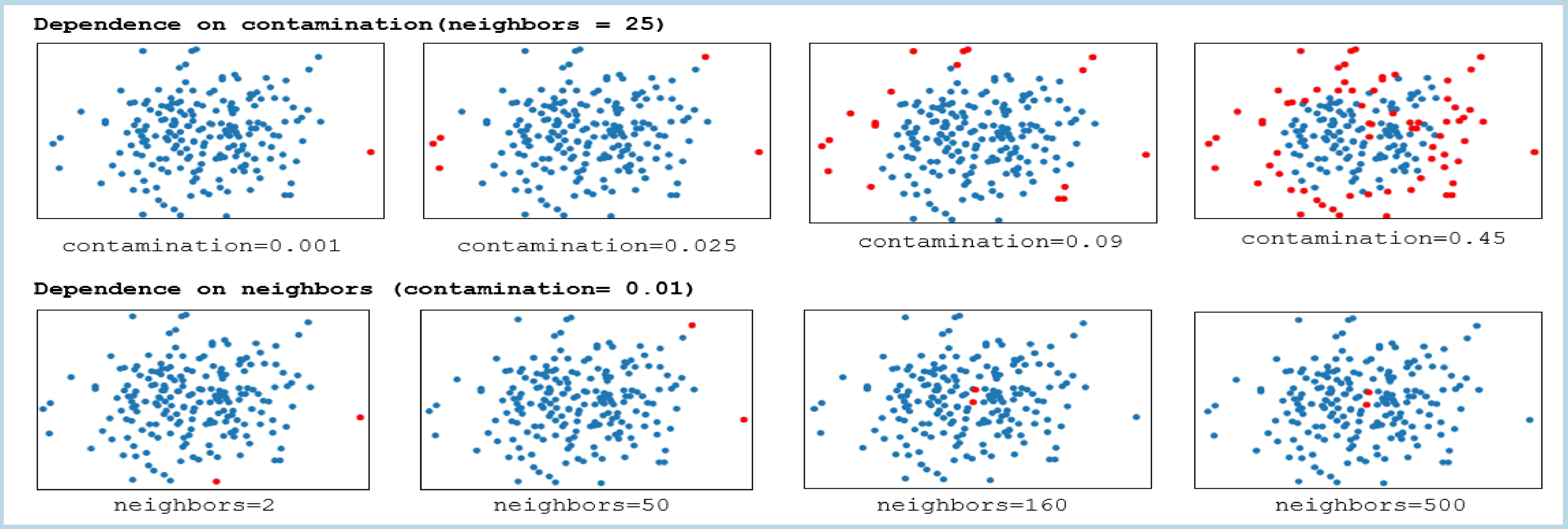

Figure 3: Anomalies detected by LOF algorithm for different values of hyperparameters

Precision (correct detection of anomalies) varies with changes in both parameters. A higher value of *neighbors* doesn't necessarily improve performance. Execution time is directly dependent on the value of *neighbors* and less dependent on *contamination* value. LOF value of every point doesn't necessarily increase with increase in *neighbors*. *Contamination* can take values from (0 ,0.5] and *neighbors* can take integer values in range [1 ,|D||].

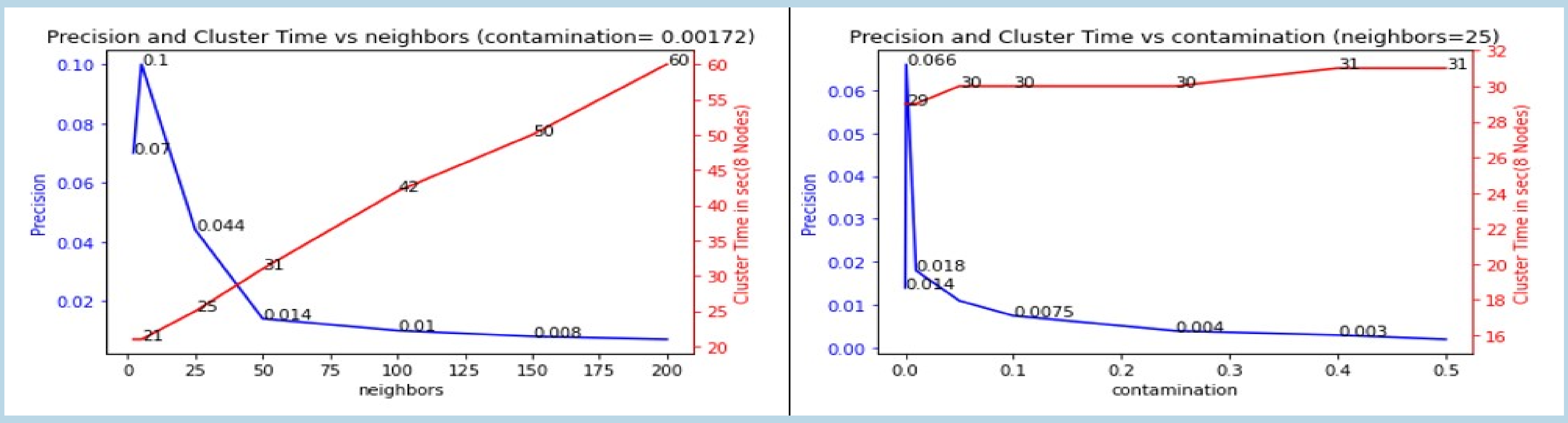

Figure 4: Dependence of Performance(Precision and Cluster time) on hyperparameters

Improved Local Outlier Factor on HPCC Systems

Normal LOF algorithm is sensitive to duplicates. If the number of duplicates of a point is more than the hyperparameter *neighbors*, then the Local Reachability density of that point becomes ∞. Normal LOF doesn't handle this efficiently, which degrades the performance. Removal of duplicates is not feasible in big data, hence Improved LOF algorithm was implemented in HPCC systems. It takes distinct neighbors and overlooks duplicates to find LOF value of points. Improved LOF algorithm uses KD Trees for this purpose. The Execution time for Improved LOF is more. However, Improved LOF algorithm can be used when the duplicates are high or for smaller values of *neighbor*

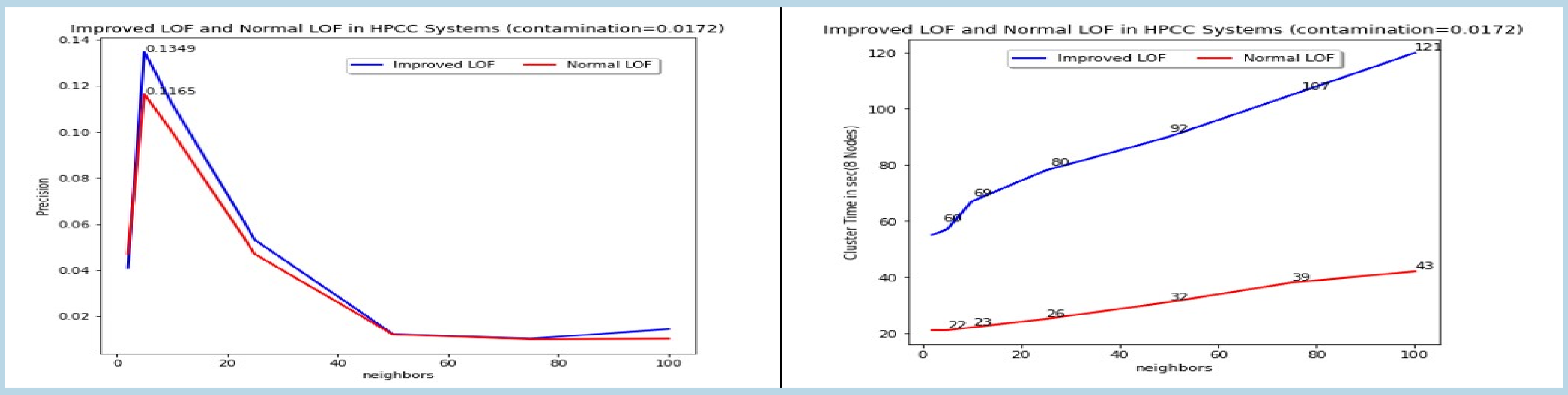

Figure 5: Performance(Precision and Cluster time) for Normal LOF and Improved LOF

Algorithm in HPCC Systems	Accuracy	Precision	Recall	F1 score	Cluster time(8 node)
Improved LOF	0.9972	0.1350	0.1342	0.1356	62 sec
Normal LOF	0.9967	0.1160	0.1158	0.1161	22 sec

Link to Video Explanation

https://youtu.be/mWPwAWyJBUQ