2022 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)

Classes versus Communities: Outlier Detection and Removal in Tabular Datasets via Social Network Analysis (ClaCO)

Serkan Ucer[1] Tansel Özyer[2] Reda Alhajj[3,4,5]
[1]The Scientific and Technological Research Council of Turkey (TUBITAK), Ankara, Türkiye
[2]Department of Computer Engineering, Ankara Medipol University, Ankara, Turkey
[3]Department of Computer Science, University of Calgary, Alberta, Canada
[4]Department of Computer Engineering, Istanbul Medipol University, Istanbul, Turkey
[5]Department of Health Informatics, University of Southern Denmark, Odense, Denmark

***Abstract*— In this research, we introduce a model to detect inconsistent & anomalous samples in tabular labeled datasets which are used in machine learning classification tasks, frequently. Our model, abbreviated as the ClaCO (Classes vs. Communities: SNA for Outlier Detection), first converts tabular data with labels into an attributed and labeled undirected network graph. Following the enrichment of the graph, it analyses the edge structure of the individual egonets, in terms of the class and community belongings, by introducing a new SNA metric named as 'the Consistency Score of a Node - CSoN'. Through an exhaustive analysis of the ego network of a node, CSoN tries to exhibit consistency of a node by examining the similarity of its immediate neighbors in terms of shared class and/or shared community belongings. To prove the efficiency of the proposed ClaCO, we employed it as a subsidiary method for detecting anomalous samples in the train part in the traditional ML classification task. With the help of this new consistency score, the least CSoN scored set of nodes flagged as outliers and removed from the training dataset, and remaining part fed into the ML model to see the effect on classification performance with the 'whole' dataset through competing outlier detection methods.**

We have shown this outlier detection model as an efficient method since it improves classification performance both on the whole dataset and reduced datasets with competing outlier detection methods, over several known both real-life and synthetic datasets.

Keywords*— *Social Network Analysis, supervised learning, graph-based outlier detection, structural outlier detection, downsampling of data

I. Introduction

The use of networked data is ubiquitous; particularly thanks for its help compacting data in a visual intuitive language. The power of visualization of data as a network diagram has many advantages including for humans to grasp a vast amount of information at once. While the network approach has many advantages; many of the real-world phenomena could not be able to directly expressable as a graph. As an examplary to such is the labeled tabular data, which is often used in machine learning (ML) classification tasks; where the observations described by various features and results are tagged with categorical labels. In this research, we aim to convert raw 'tabular data' to a 'network graph' and inspect this graphs' nodes by SNA techniques in order to infer their 'consistency' with respect to its raw form.

After converting tabular dataset to a network graph; one can analyze it further via graph theory and Social Network Analysis techniques. Among those techniques; finding the communities of the graph is of particular interest in this work; since we aim to compare the neighbors of nodes based on their belongings to (i) their current classes and (ii) their inferred communities. Via this comparison; we have designed a novel SNA measure works at the node level (named as 'consistency score of a node – CSoN') which assigns consistency scores to nodes; and in turn, those scores used as an outlier detection criteria. By considering the class distribution of the initial tabular dataset; the nodes of the graph are further examined by their respective CSoN scores; for removal of certain low-scored nodes. A brief pipeline is diagrammed and explained as in Fig. 1.

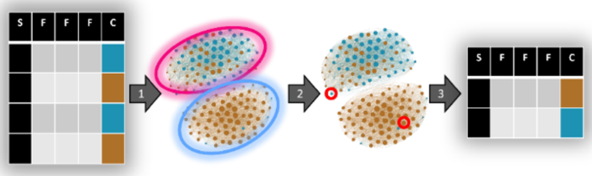


Fig. 1. – Given a tabular dataset (as illustrated by S for samples, F for features, C for class information) with categories (illustrated as in brown and turquoise colors), in step (1), we generate an undirected network graph based on vectorial similarity of nodes (hence 'samples' become 'nodes') and then we find SNA communities (illustrated as in Fuschia and blue colors). After computing CSoN for each node, (2) the least scored ones are (illustrated as in red circles) removed from the graph, finally (3) removed outliers reflected back to the initial tabular dataset.

We have designed ClaCO Model so that it may work on multi-labeled tabular datasets; which may have contain both numerical and/or categorical features. As the performance evaluation of outlier detection; we have preferred to use it in the ML classification task as a data cleansing tool over the training part. With the use of selected well-established ML classifiers; we compared the contribution of the ClaCO Model on the classification performance in comparison to full (not reduced datasets) and also to competing outlier detection methods.

In this introductory concept paper; we would like to present the current work and fundamental ideas to the community. Following the briefing on problem setting and related work in Section 2; the ClaCO Model has been described in Section 3 in detail. In section 4, we present the designed experiments in order to measure the effectiveness of

IEEE/ACM ASONAM 2022, November 10-13, 2022

the proposed method against competing outlier detection methods. Discussion on the future work to be done and future directions presented in the final Section 5.

II. Background and Problem Statement

Outliers in statistical context are samples that show dissimilar features as the general distribution of the data. The concept merely lacks a general definition, for instance, Hawkins [1] defines the outlier data as in “an observation that deviates so much from other observations as to arouse suspicion that it was generated by a different mechanism.” The presence of outliers in a classification dataset may decrease the performance of the classification performance of ML classification model. In this respect, several approaches have been devised to identify and remove outliers on the training part of a dataset, automatically.

The formal problem statement of ClaCO is; given a tabular classification dataset, can we detect outlying or anomalous samples (other than traditional methods) by analyzing it as a graph. Thus, we set the hypothesis validation as in “if we can detect anomalous samples; reducing them from the data should improve classification performance by all or on majority of ML classifiers”. In order to simplify the ClaCO process; we prepared below Fig. 2 to display where the ClaCO Model fits in the ML classification process.

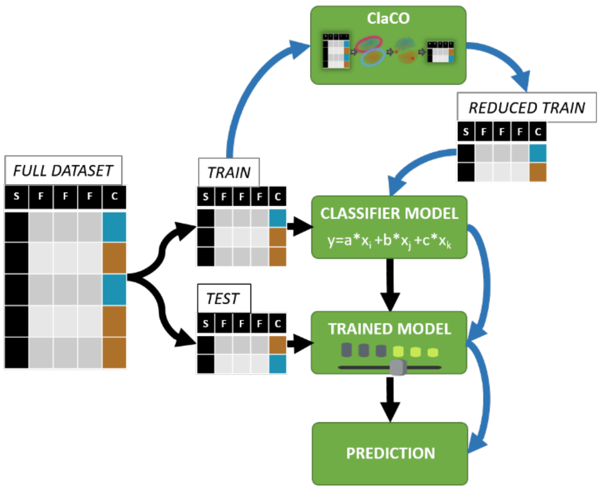


Fig. 2. - Overview of ML Classification task and intervention of ClaCO Model. Conventionally, following the dataset split into ‘train’ and ‘test’ parts, ‘train’ fed into the classifier model (as in black arrows). ClaCO intervenes this process by seeking and removing outliers in ‘train’ data before feeding it into the classifier model (as in blue arrows.)

Several research has been done on improving the classification performance using outlier detection methods over training data. Among these, in [2] researchers efficiently show the performance increase of SVM and Decision Tree classifiers over seven bi-class imbalanced datasets by introducing an outlier score for samples.

We believe our research contributes to this domain in several aspects including:

i. graph-based structural identification of outliers,
ii. superior or on par classification performance over a wide set of ML classifiers,
iii. able to work on multi-class (not only bi-class) severely imbalanced datasets, and
iv. explainable and visually expressable outlier detection process.

In the following two subsections, we present the current related work in two aspects: (A) the outlier detection methods over tabular datasets and (B) graph-based anomaly detection methods.

A. *Outlier Sample Detection Methods over Tabular Datasets*

Detection of outliers in conventional tabular data models can be grouped in 5 approaches [3] those are probabilistic, proximity based, linear, deep learning and ensemble methods. All methods belonging to those catgories define outliers as the data points that are distant from the rest of the data samples. In this section, we will briefly review selected eight outlier detection methods including classical ones from ML domain to the latest deep learning methods, and emerging algorithms like COPOD and ECOD.

1) Local Outlier Factor [4] (LOF): This technique employs the nearest neighbors-based approach for outlier detection. In short, each sample is assigned a scoring according to how isolated based on the size of its local neighborhood. Those samples with the largest score are more likely to be outliers; and removed based on the presumed contamination rate.

2) One Class Support Vector Machines [5] (OCSVM): SVMs are initially designed to classify biclass datasets. When modeling one class, the algorithm captures the density of the ‘majority class’ (i.e., the class that is heavily populated relative to other classes) and classifies samples on the extremes of the density function as outliers [6].

3) Isolation Forests [7] (iForest): This tree-based anomaly detection model the normal data in such a way as to isolate anomalies that are both few and different in the feature space. These are respectively samples that are the minority consisting of fewer frequency and ii) samples which have attribute-values that are very different from those of normal instances.

4) Minimum Covariance Detection [8] (MCDE): This model works efficiently on data that shows Gaussian distribution properties. This approach can be generalized by defining an ellipsoid that covers the normal data, and data that falls outside this shape is considered an outlier. An efficient implementation of this technique for multivariate data is the Minimum Covariance Determinant [6], [9].

5) Principal Component Analysis [10] (PCA): The very familiar PCA model has vast usage both in exploratory data analysis and classification tasks. As an outlier detection model, PCA is used as reconstruction tool for the original dataset; and distance from reconstructed data to the original data employed as an outlier score.

6) Deep One-Class Classification [11] (DeepSVDD): This recent model employs deep learning approach to outlier detection task by training a neural network while minimizing the volume of a hypersphere that encloses the network representations of the data. Outlier scores has been extracted by calculating the distance from the center of the hypersphere
.

7) Copula-Based Outlier Detection [12] (COPOD): Inspired by copulas for modeling multivariate data distribution, this model constructs an empirical copula, and then uses it to predict tail probabilities of each given data point

to determine its level of outlierness which in turn used in order to generate outlier scores.

8) Unsupervised Outlier Detection Using Empirical Cumulative Distribution Functions [13] (ECOD): This model is very similar to COPOD but uses cumulative distribution functions (instead of copulas) in order to detect outlierness of the data samples.

For all the above methods; the most important parameter is the contamination factor. In conventional outlier detection methods; the term 'contamination rate' refers to the assumption of the percentage of outliers' exists in a dataset. For automatic detection and removal of outliers; its value defaults to 10%. There is a slight exception in the OCSVM Model that, instead of defining an explicit contamination rate, it employs 'nu' factor, which approximates the ratio of outliers; but in implementation, it might exceed this rate.

In comparison, the ClaCO Model indeed fits into this range of techniques since we generate graphs on normed space; and detect outliers based on distance-based scores (ie CSoN). For this respect; we have included these techniques in "4. Experiments" section.

B. Anomalous Node Detection on Graphs

With the popularization of networked data, outlier node detection within a graph has attracted attention. Anomalous nodes broadly refer to the set of nodes that deviate from the rest of the nodes in terms of graph structural properties such as edge distribution or neighbor structures. We can divide anomalous node detection approaches into two main categories [14]:

i. traditional techniques and
ii. deep learning-based techniques.

Traditional techniques use a priori networked data (data already in the graph form) and employ attributes of the nodes to find a statistical tie by either matrix factorization techniques and/or SNA measures such as in-degree/out-degree [15] [16]. The ClaCO Model only partially (partially, since ClaCO converts tabular data into a graph in 'normalized' space where distance between nodes are meaningful, contrast to nodes which have ambigious relation as in convential graph theory.) fits into this category, since in the 'outlier detection' part, our proposed model statistically assigns a consistency score based on the structural properties of the graph. On the other hand, and more recently, deep learning techniques (ie. not the actual graph but its feature representation vectors are analyzed) are popularized for detecting anomalies [17] [18].

III. ClaCO: Pipeline of the Proposed Model

In this section, we will present the details of the process, which is briefly described in Fig. 2.

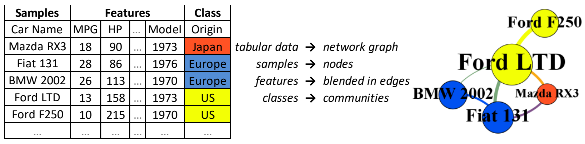
A. Preprocessing the Dataset

In this step, we split the dataset into train and test parts by stratified cross-validation. The ClaCO only uses data only from the training part in order to prevent data leakage. Further, the data is preprocessed with scaling techniques for numerical features, and one-hot-encoding for the categorical features. Since those processes are trivial, we skip the details here.

B. TD2NG: Tabular Data to Network Graph

After the train part is preprocessed; we use it to construct the undirected graph. We complete this conversion by a method presented in a recent work of us [19]; which simply is a novel Exploratory Data Analysis technique named as the 'Tabular Data to Network Graph (TD2NG)'.

Briefly speaking, in a network graph; edges between nodes represent some kind of interaction between connected ones. In our conversion procedure within TD2NG; this interaction is defined as the vectorial similarity between samples. In brief; the similarity between samples is inferred via the weighted and normed (L2 norm for numerical parts of the sample and/or L0 norm for categorical parts; where weights are obtained via feature importance methods) distances. We have presented the overview of the TD2NG Model in Fig. 3.



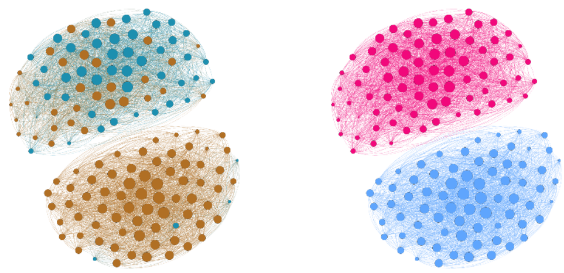
Samples	Features				Class
Car Name	MPG	HP	...	Model	Origin
Mazda RX3	18	90	...	1973	Japan
Fiat 131	28	86	...	1976	Europe
BMW 2002	26	113	...	1970	Europe
Ford LTD	13	158	...	1973	US
Ford F250	10	215	...	1970	US
...	...	...	...	...	...

Fig. 3. - Overview flow diagram of the TD2NG Model over a sample tabular datasets belonging to Car models. Diagram also displays naming conventions during conversion from tabular data into network graph.

Finally, we would like to note that (after the conversion procedure) the changes in the naming of the structures: tabular data becomes a graph, samples become nodes, data features are represented in edges, and the class information is represented as communities within the graph.

C. Community Analysis of the Network Graph

Our definition of consistency depends on the community analysis, so in this step, we carefully split the raw graph into communities. Kindly note that at this step we already have a community splitting, which is done by the class of nodes. We also employ an additional community analysis method, which is the Louvain community detection method [20], that also splits nodes into groupings.



(i) (ii)

Fig. 4. - (i) The tabular (illustrative) dataset 'Titanic' [21] converted into a network graph with TD2NG Model. Nodes colored in turquoise represent the 'survivors' of the Titanic disaster, while brown represents the 'victims'. Graph layouted with Fruchterman-Reingold [22] algorithm. Nodes are scaled in size by their weighted degree. (ii) The tabular dataset 'Titanic' splitted automatically into two communities by the Louvain community detection method. Fuschia color represents Community A and blue color represents Community B.

Louvain method iteratively optimizes the modularity of proposed communities until the modularity of that communities converges to 1 (which is defined as full modular clustering). By comparing Fig. 4 (i) and (ii), we see that the class 'victims' hugely overlaps with nodes belonging to Community B. Only four of the community B members are from the class 'survivors', which gives a hint about their outlyingness in the 'survivors' class.

D. Analysing Ego Networks & Introduction of the Degree of Consistency Measure

As we hinted in the former section, ClaCO uses a comparison-based measure to assign consistency scores to nodes, which is computed according to their belongings into classes and communities. The idea behind this proposed 'consistency' measure is that, a node is consistent as far as its first-degree neighbors share the same class and same community with that node. So, we analyze ego (sub)networks of this graph iteratively to compute CSoN for each node.

The first-degree neighbors of any node can be from or combination of:

- Same class, same community (SS),
- Same class, different community (SD)
- Different class, same community (DS), and/or
- Different class, different community (DD)

regions. We have illustrated this insight on an illustrative diagram presented in Fig. 5.

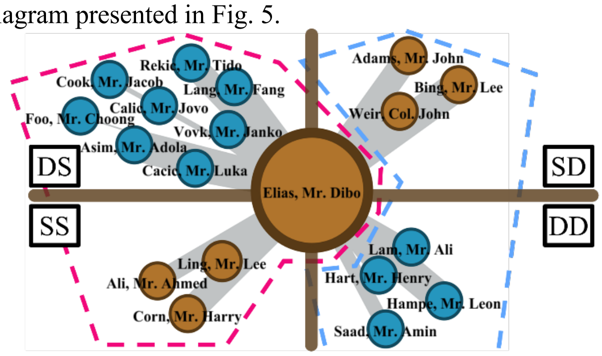


Fig. 5. - Four regions of neighbors are illustrated in the egonet of the node "Elias, Mr. Dibo" from the 'Titanic' graph. Node colors represent classes (turquoise for the 'survivors', brown for 'victims') and regions in dashed lines represent different communities found by Louvain community detection.

The weights of the edges to the neighbors are used as score components in the calculation of the consistency score. As we analyze these four types of neighbors, we claim that the most important ones are in the SS and SD regions, since we heuristically know that the driving force of consistency lies within being in the same class. Between these two, SD can be considered as beneficial towards deducing the consistency but should affect the score far less than neighbors in the SS region. For the remaining two regions, neighbors in DS and DD regions show a nodes' deviation from its class, so the (potential) neighbors in these regions should affect CSoN negatively. The penalizing effect obviously should be higher in the DD region since neighbors in this region are both from different classes and different communities. With these insights, we define CSoN measure as in:

$$CSoN(i) = W^T * (ew_i(SS),\ ew_i(SD),\ ew_i(DS),\ ew_i(DD)) \tag{1}$$

$$ew_i(REGION) = \frac{\sum_{REGION} ew_i}{\sum ew_i} \quad \text{where } REGION \in \{SS, SD, DS, DD\} \tag{2}$$

$$W = (w_{SS}, w_{SD}, w_{DS}, w_{DD}) \tag{3}$$

In the equations (1, 2, 3), i refers to the index of a specific node, W refers to the weights of the components for four regions, and e_{wi}(region) refers to the proportion of edge weights of node i to its first degree neighbors in the specified region to the total edge weights of node i. Through the selection of weight vector W, we indeed introduce a reward/penalty factor to the neighbors in these four regions. Through an exhaustive heuristic search, and by keeping up the insights from the discussions above, we conclude the selection of weight scalar vector W as in (1, 0.5, -0.5, -1) for the CSoN model. Within this, we state that the CSoN score belongs to the range [-1, 1] where negative scores indicate inconsistency whereas positive scores show a nodes' consistency.

E. Removing Inconsistent/Outlier Nodes

We will now use CSoN scores in two stages for the task of cleansing the dataset from potential outliers. As in a similar fashion, in stage 1 of the CSoN removal process, for every class category, we remove the bottom k nodes, whereas k is calculated according to the contamination rate (traditionally defaulted to 10%). In addition to the stage 1 removal, in stage 2, we further remove certain nodes from the overly populated classes, whereas CSoN is negative. This approach is useful for dealing with imbalanced datasets since it favors the least populated classes. For the class imbalanced datasets [23], the term imbalance ratio (IR) refers to the proportion of frequency of the major class (i.e., the most populated class) to the other classes. Accordingly, overly populated classes are classes that are above the preset threshold for IR. In ClaCO Model, we set the boundary of IR $\geqslant$ 2 to detect the least populated classes.

IV. Experimental Setup

In this section, we will present the details of the experiments designed to measure the effectiveness of the outlier detection by the ClaCO Model. As we stated earlier; for evaluating improvement of the classification performance of the ClaCO Model; we designed an experimental setup consisting of 11 datasets to be classified in 10 versions running over 7 selected ML classifiers.

A. OD Models and Tabular Classification Datasets

For the selection of the datasets to display proof of the concept work of the ClaCO Model, we carefully selected as many real-world diverse datasets from various domains with various feature and class structures. On the preprocessing steps; we standardized datasets before feeding them into ML classifiers since some classifiers (especially those are based on a scale sensitive distance metrics like SVM) heavily depend on the data to be scaled before. Table 1 summarizes the datasets experimented with.

Table 1 - Datasets used in experiments.

Dataset Name	Domain	Samples		Features			Classes		
		Sample Size	CV for splitting	*Numeric*	*Categoric*	Total	Class Labels	Class Distribution	Imbalance Ratio
Bacteria Types	Life Sci.	428	5	*5*	*0*	5	2	Imbalanced	3.26
Breast Cancer	Medicine	286	5	*0*	*9*	9	2	Imbalanced	2.37
Colon	Medicine	62	2	*1988*	*0*	1988	2	Imbalanced	1.81
COVID	Medicine	436	5	*36*	*1*	37	2	Imbalanced	1.64
Heart UCI	Medicine	302	5	*6*	*7*	13	2	Imbalanced	1.18
Lymphoma	Medicine	96	2	*4026*	*0*	4026	9	Imbalanced	23
Make Blobs	Syntethic	500	5	*2*	*0*	2	2	Balanced	1
PBC	Medicine	276	5	*16*	*2*	18	4	Imbalanced	9.25
Pima Diabetes	Medicine	768	5	*8*	*0*	8	2	Imbalanced	1.86
Titanic	Statistics	712	5	*4*	*3*	7	2	Imbalanced	1.47
Weather Rain	Statistics	412	5	*16*	*5*	21	2	Imbalanced	3.52

The Outlier Detection models used in this work is presented in Table 2. Following the type classification of the respective models in [3]; we selected at least one OD model per OD model category. Also, an ensemble method (iForest) has been added to this list.

Table 2 – Outlier Detection Models used in experiments..

Model	Title	Outlier Detection Type	Developed in
LOF	Local Outlier Factor	Proximity-Based	2000
MCDE	Minimum Covariance Determinant	Linear Model	1999
OCSVM	One-Class Support Vector Machines	Linear Model	2001
PCA	Principal Component Analysis	Linear Model	2003
iForest	Isolation Forest	Outlier Ensembles	2008
DeepSVDD	Deep One-Class Classification	Deep Neural Networks	2018
COPOD	Copula-Based Outlier Detection	Probabilistic	2020
ECOD	Unsupervised Outlier Detection Using Empirical Cumulative Distribution Functions	Probabilistic	2022

The contamination factor for the competing outlier detection methods and the ClaCO Model has been set to the same %10 for a fair comparison. Also, for addressing the reproducibility of the results; we have set the same seed numbers in data-splitting (cross-validation) processes.

After common preprocessing of the data; we split the dataset into train and test parts (in cross-validation) and with only using the training part; we have appropriately prepared ten versions which are listed below.

i. Original training dataset part (no reduction in samples),

ii. Reduced (with LOF model) training data,

iii. Reduced (with iForest model) training data,

iv. Reduced (with OCSVM model) training data,

v. Reduced (with MCDE model) training data,

vi. Reduced (with ECOD model) training data,

vii. Reduced (with COPOD model) training data,

viii. Reduced (with DeepSVDD model) training data,

ix. Reduced (with PCA model) training data, and

x. Reduced (with proposed ClaCO Model) training data.

A note on the supevised character of the process; as a classification problem is given with a sample data matrix X and a target categorical vector y, where X is assumed to be contaminated with outliers. Our assumption is (see Fig. 2) to clean X_{train} training part of the data from possible outliers and then use this cleaned version of training data X_{train}^{clean} to predict y_{test} . In principle, outlier detection task is categorized as an unsupervised task meaning that data is not labeled according to outlierness a priori. Within this, for the best use of data; our application of ClaCO (as well as in competing OD models) as to the data as in the joint $\{X_{train}, y_{train}\}$ matrix. This is a legal operation since, during the detection and removal of outlier in training part of the data, we do not leak any information from the test part, which should be not known at the time of prediction.

B. *Classifiers used for Benchmarking*

The selection criteria for ML classifiers were to include as diverse as much in terms of their approach for classification. All classifiers run with their default parameter set (except for ANN, maximum iteration is upgraded to 2000 from 200 since it does not usually converge with 200 iterations) . Also, we did not search for the best parameters for classifiers since aim is not to find best classifier. We have used scikit-learn [24] implementation of those classifiers except XGBoost, where we used native XGBoost Python library [25]. We present in Table 3 the list of classifiers and their parameters used for the comparison.

Table 3 - ML Classifiers used in experiments.

Classifier Name	Classifier Type	Used Parameters
AdaBoost	Boosting	base_estimator=None, n_estimators=50, learning_rate=1.0, algorithm='SAMME.R', random_state=None
Artificial Neural Networks	Function Based	max_iter=2000, hidden_layer_sizes=(100,), activation='relu', *, solver='adam', alpha=0.0001, batch_size='auto', learning_rate='constant', learning_rate_init=0.001, power_t=0.5, shuffle=True, random_state=None, tol=0.0001, verbose=False, warm_start=False, momentum=0.9, nesterovs_momentum=True, early_stopping=False, validation_fraction=0.1, beta_1=0.9, beta_2=0.999, epsilon=1e-08, n_iter_no_change=10, max_fun=15000
Decision Tree	Tree-based	criterion='gini', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_decrease=0.0, class_weight=None, ccp_alpha=0.0
Random Forest	Ensemble-Tree based	n_estimators=100, *, criterion='gini', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None, min_impurity_decrease=0.0, bootstrap=True, oob_score=False, n_jobs=None, random_state=None, verbose=0, warm_start=False, class_weight=None, ccp_alpha=0.0, max_samples=None
SVM Linear Kernel	Function Based	C=1.0, kernel='linear', degree=3, gamma='scale', coef0=0.0, shrinking=True, probability=False, tol=0.001, cache_size=200, class_weight=None, verbose=False, max_iter=- 1, decision_function_shape='ovr', break_ties=False, random_state=None
SVM RBF Kernel	Function Based	C=1.0, kernel='rbf', degree=3, gamma='scale', coef0=0.0, shrinking=True, probability=False, tol=0.001, cache_size=200, class_weight=None, verbose=False, max_iter=- 1, decision_function_shape='ovr', break_ties=False, random_state=None
XGBoost	Boosting	default parameter set

For each dataset and each configuration, we repeated the experiments 10 times with different seed numbers (see appendix for full configuration.), in order to achieve balanced and smoothed classification scores. As a last note; the MCDE model partially failed to run over the dataset "Lymphoma" since the input (for several seed numbers) was not created rank for the construction of a covariant matrix. Also, the OCSVM model failed to find outliers (flagged all samples as outliers in some folds of the data) in the PBC dataset, for some certain splits of the data. For both of these scenarios; we still preferred to include variations that were working in "5. Results and Discussion" section.

C. *Software Setup*

We implemented the ClaCO Model on the Python 3.9 platform. We have used scikit-learn [24] implementation of the LOF, iForest, OCSVM and MCDE outlier detection models whereas PyOD [3] implementation for the remaining four models namely COPOD, DeepSVDD, PCA and ECOD. For the implementation of the classification models, we preferred to use the scikit-learn library framework since it is the de facto standard platform in the ML domain. Scikit-learn also allows parameter optimization by grid or random search methods; making it useful to find the best performer set of parameters for a given task. Within this, we did not practice parameter search or optimization for this research; to present a fair comparison with other outlier detection methods. We also used other python libraries such as networkx, bokeh, pandas, and NumPy. All network images within this research are produced with social network analysis software Gephi.

V. Results and Discussion

We have set two types of benchmarks related to the contribution to the classification performance of selected classifiers in selected datasets:

A) Overall classification performance comparison over the six dataset versions,

B) Individual ML classifiers performance comparison over datasets.

In this paper, we evaluate the classification performance through several different measures, precision, recall, and F1 score. For the sake of interpretability, we prepared the reports in this section according to the weighted F1 score (average of

10 runs with different random seeds) since it efficiently indicates the classification performance, particularly over imbalanced datasets.

A. *Comparison of Overall Classification Performance Achieved by*

In this benchmark, we aim to display how much top classification performance has been achieved over several types of dataset versions (there is a total of 10 configurations for each dataset). Following Table 3, we can see that; when the ClaCO Model is used as in outlier detection; the classification performance increases up to %10 (i.e., in PBC dataset, classification performance increased to 54% from 49%) over the original dataset. When compared to all dataset versions; ClaCO achieves the best performance for 9 of the 10 datasets in all versions and ranks as the first. Only in *Bacteria* dataset, ClaCO ranks in second position after OCSVM model. Summary of the results presented in Table 4.

Table 4 - Classification performance summary over different data configurations (over two tables).

Dataset Name and Version			Ada Boost	ANN	Decision Trees	Random Forest	SVM Linear	SVM RBF	XGBoost	Maximum F1 Score	Ranking of the Model
Breast Cancer	Original Dataset		69.98%	68.87%	67.69%	**73.00%**	69.85%	61.86%	70.24%	**73.00%**	2
	Reduced dataset by	COPOD	66.70%	63.99%	65.01%	**66.81%**	65.67%	59.21%	65.57%	**66.81%**	10
		DeepSVDD	68.88%	67.16%	66.92%	**69.93%**	68.44%	60.98%	68.30%	**69.93%**	3
		ECOD	**68.19%**	63.17%	65.19%	66.05%	64.77%	58.86%	66.82%	**68.19%**	9
		iForest	68.17%	65.64%	65.29%	**68.43%**	66.77%	60.30%	68.02%	**68.43%**	8
		LOF	67.36%	64.09%	65.76%	**68.52%**	67.25%	59.91%	67.34%	**68.52%**	7
		MCDE	68.36%	65.50%	66.43%	**69.27%**	67.32%	60.72%	68.40%	**69.27%**	5
		OCSVM	68.23%	65.76%	65.93%	68.78%	**69.11%**	59.88%	66.91%	**69.11%**	6
		PCA	68.95%	68.40%	65.60%	**69.69%**	68.36%	60.32%	66.69%	**69.69%**	4
		ClaCO	72.80%	71.88%	72.87%	**75.08%**	73.24%	71.96%	72.03%	**75.08%**	1
Heart UCI	Original Dataset		78.72%	79.15%	73.57%	79.95%	**82.48%**	81.28%	79.18%	**82.48%**	2
	Reduced dataset by	COPOD	74.57%	77.88%	71.82%	79.16%	**79.32%**	75.29%	78.81%	**79.32%**	10
		DeepSVDD	78.74%	79.23%	72.78%	79.75%	**80.77%**	79.62%	78.46%	**80.77%**	6
		ECOD	76.29%	77.20%	71.87%	79.01%	**79.46%**	77.67%	78.27%	**79.46%**	9
		iForest	77.85%	78.21%	73.62%	79.38%	**80.46%**	79.12%	78.76%	**80.46%**	7
		LOF	79.47%	80.92%	74.65%	80.92%	**81.47%**	79.68%	79.97%	**81.47%**	3
		MCDE	77.43%	78.51%	73.08%	79.94%	**81.28%**	78.28%	78.59%	**81.28%**	4
		OCSVM	78.46%	80.56%	73.94%	80.05%	**81.00%**	80.30%	78.62%	**81.00%**	5
		PCA	77.52%	77.67%	72.66%	78.64%	**80.37%**	78.20%	78.68%	**80.37%**	8
		ClaCO	85.44%	85.52%	81.96%	84.27%	84.95%	**85.59%**	84.94%	**85.59%**	1
Make Blobs	Original Dataset		81.96%	84.67%	79.32%	82.45%	**84.84%**	84.65%	81.36%	**84.84%**	4
	Reduced dataset by	COPOD	80.63%	**84.05%**	78.54%	81.68%	83.83%	80.91%	80.71%	**84.05%**	10
		DeepSVDD	82.63%	**85.23%**	81.50%	83.20%	84.08%	85.02%	82.83%	**85.23%**	2
		ECOD	81.06%	**84.26%**	78.25%	81.94%	83.99%	81.37%	81.26%	**84.26%**	8
		iForest	81.81%	**84.60%**	80.81%	82.31%	84.43%	82.82%	81.92%	**84.60%**	6
		LOF	82.44%	**84.95%**	81.63%	83.03%	84.59%	83.53%	82.17%	**84.95%**	3
		MCDE	82.13%	**84.77%**	80.72%	82.78%	84.67%	83.21%	82.19%	**84.77%**	5
		OCSVM	81.11%	84.09%	80.22%	82.13%	**84.52%**	82.85%	81.76%	**84.52%**	7
		PCA	80.11%	**84.16%**	78.06%	82.00%	84.14%	81.56%	80.96%	**84.16%**	9
		ClaCO	87.47%	87.91%	87.38%	87.77%	85.93%	**87.97%**	87.43%	**87.97%**	1
Titanic	Original Dataset		79.31%	80.62%	76.15%	79.42%	77.67%	**80.67%**	79.49%	**80.67%**	3
	Reduced dataset by	COPOD	76.28%	**77.42%**	73.63%	76.40%	75.17%	76.46%	76.11%	**77.42%**	10
		DeepSVDD	78.77%	78.70%	76.53%	79.60%	77.18%	77.30%	**79.69%**	**79.69%**	5
		ECOD	76.13%	**77.58%**	73.12%	76.58%	75.54%	77.16%	76.28%	**77.58%**	9
		iForest	77.61%	**77.85%**	74.38%	77.46%	76.28%	77.30%	77.37%	**77.85%**	7
		LOF	79.10%	79.38%	77.65%	80.75%	77.30%	78.63%	**81.18%**	**81.18%**	2
		MCDE	**77.59%**	77.44%	74.57%	76.68%	74.72%	76.27%	76.20%	**77.59%**	8
		OCSVM	78.56%	78.24%	77.28%	**79.76%**	76.59%	77.31%	79.40%	**79.76%**	4
		PCA	76.97%	77.61%	74.94%	78.16%	77.21%	**78.27%**	77.71%	**78.27%**	6
		ClaCO	**84.72%**	82.62%	84.07%	83.93%	82.39%	83.01%	83.66%	**84.72%**	1
Weather Rain	Original Dataset		81.49%	81.80%	78.41%	83.19%	80.46%	81.53%	**83.44%**	**83.44%**	2
	Reduced dataset by	COPOD	78.89%	78.73%	75.66%	**80.32%**	77.12%	73.94%	80.13%	**80.32%**	10
		DeepSVDD	80.37%	78.06%	77.54%	81.48%	78.61%	73.81%	**82.08%**	**82.08%**	5
		ECOD	78.80%	78.86%	76.50%	80.13%	78.19%	73.43%	**80.47%**	**80.47%**	9
		iForest	79.10%	78.49%	78.10%	**81.35%**	78.06%	72.92%	81.08%	**81.35%**	7
		LOF	80.77%	79.38%	79.65%	82.51%	79.23%	75.84%	**83.34%**	**83.34%**	3
		MCDE	79.53%	78.20%	78.10%	81.44%	77.29%	73.18%	**82.03%**	**82.03%**	6
		OCSVM	79.83%	79.76%	77.01%	81.38%	78.94%	75.24%	**82.38%**	**82.38%**	4
		PCA	79.05%	78.15%	77.40%	**81.14%**	78.12%	71.51%	80.84%	**81.14%**	8
		ClaCO	84.89%	82.28%	82.90%	**87.07%**	81.80%	83.00%	85.74%	**87.07%**	1

Dataset Name and Version			Ada Boost	ANN	Decision Trees	Random Forest	SVM Linear	SVM RBF	XGBoost	Maximum F1 Score	Ranking of the Model
Colon Cancer	Original Dataset		76.36%	78.96%	72.17%	78.80%	**83.15%**	68.08%	78.01%	**83.15%**	2
	Reduced dataset by	COPOD	79.27%	78.88%	70.84%	**81.77%**	81.35%	68.30%	77.48%	**81.77%**	4
		DeepSVDD	72.43%	78.51%	67.47%	75.25%	**80.81%**	64.20%	72.61%	**80.81%**	8
		ECOD	77.11%	78.59%	71.27%	**81.95%**	81.49%	69.60%	78.58%	**81.95%**	3
		iForest	77.59%	78.54%	70.25%	79.53%	**81.59%**	68.69%	76.60%	**81.59%**	6
		LOF	78.24%	78.46%	72.21%	**81.74%**	80.73%	70.51%	77.87%	**81.74%**	5
		MCDE	71.26%	76.16%	67.67%	73.77%	**79.67%**	60.84%	72.40%	**79.67%**	9
		OCSVM	65.73%	74.25%	63.58%	69.20%	**77.35%**	54.02%	64.51%	**77.35%**	10
		PCA	78.16%	78.44%	70.91%	**81.40%**	80.72%	70.33%	76.64%	**81.40%**	7
		ClaCO	78.42%	82.71%	77.44%	84.39%	**86.46%**	73.29%	78.35%	**86.46%**	1
Lymphoma	Original Dataset		55.85%	87.38%	61.81%	74.31%	**91.88%**	62.39%	70.26%	**91.88%**	2
	Reduced dataset by	COPOD	51.73%	81.98%	57.09%	66.88%	**84.81%**	56.54%	64.71%	**84.81%**	9
		DeepSVDD	58.60%	86.53%	62.40%	73.04%	**89.16%**	59.49%	69.05%	**89.16%**	3
		ECOD	52.04%	82.33%	57.52%	67.27%	**85.37%**	57.27%	66.07%	**85.37%**	6
		iForest	50.93%	83.02%	57.02%	68.01%	**85.83%**	56.00%	66.61%	**85.83%**	5
		LOF	51.88%	81.93%	57.53%	67.45%	**85.22%**	56.41%	64.85%	**85.22%**	7
		MCDE	54.71%	82.77%	57.84%	69.60%	**87.57%**	55.79%	66.24%	**87.57%**	4
		OCSVM	47.48%	81.13%	56.13%	62.75%	**82.31%**	46.47%	62.79%	**82.31%**	10
		PCA	51.00%	82.44%	57.88%	67.39%	**85.01%**	56.41%	65.13%	**85.01%**	8
		ClaCO	54.74%	87.53%	62.71%	76.38%	**92.23%**	66.41%	68.74%	**92.23%**	1
Bacteria	Original Dataset		65.86%	66.17%	67.84%	68.80%	65.85%	65.78%	**69.29%**	**69.29%**	9
	Reduced dataset by	COPOD	66.45%	66.90%	68.24%	69.67%	65.86%	65.92%	**69.99%**	**69.99%**	8
		DeepSVDD	69.76%	67.36%	70.65%	71.40%	65.86%	66.01%	**71.84%**	**71.84%**	4
		ECOD	66.60%	66.98%	67.82%	69.91%	65.86%	66.11%	**70.23%**	**70.23%**	7
		iForest	69.20%	66.98%	69.57%	**71.34%**	65.86%	65.86%	71.31%	**71.34%**	5
		LOF	70.07%	67.48%	69.95%	**71.93%**	65.85%	65.92%	71.42%	**71.93%**	3
		MCDE	66.17%	67.23%	67.32%	**68.78%**	65.86%	66.10%	68.65%	**68.78%**	10
		OCSVM	71.43%	69.90%	72.21%	**74.60%**	65.86%	65.84%	74.37%	**74.60%**	1
		PCA	68.04%	66.67%	67.91%	**71.16%**	65.86%	65.93%	70.79%	**71.16%**	6
		ClaCO	71.76%	69.48%	72.40%	72.77%	66.36%	66.08%	**72.92%**	**72.92%**	2
PBC	Original Dataset		44.42%	48.94%	40.77%	47.41%	**49.19%**	49.16%	45.90%	**49.19%**	2
	Reduced dataset by	COPOD	44.06%	46.22%	39.75%	42.94%	**47.03%**	43.98%	41.83%	**47.03%**	6
		DeepSVDD	42.55%	43.30%	41.59%	**45.42%**	43.29%	40.86%	44.53%	**45.42%**	10
		ECOD	44.14%	**47.18%**	38.88%	42.60%	46.65%	43.06%	40.81%	**47.18%**	5
		iForest	43.40%	46.82%	40.39%	44.10%	**47.24%**	44.68%	43.58%	**47.24%**	4
		LOF	43.49%	42.23%	39.99%	**46.36%**	44.37%	40.29%	45.57%	**46.36%**	7
		MCDE	44.23%	46.39%	40.07%	46.48%	**48.05%**	46.80%	45.76%	**48.05%**	3
		OCSVM	43.35%	42.80%	41.51%	**45.53%**	45.12%	40.08%	45.07%	**45.53%**	8
		PCA	43.80%	44.50%	40.34%	45.32%	**45.48%**	42.15%	43.71%	**45.48%**	9
		ClaCO	36.17%	53.88%	50.72%	**54.43%**	52.82%	46.36%	52.49%	**54.43%**	1
Pima Diabetes	Original Dataset		74.46%	75.60%	69.96%	75.67%	**76.31%**	75.46%	73.48%	**76.31%**	3
	Reduced dataset by	COPOD	71.68%	73.18%	66.52%	74.25%	**76.15%**	69.39%	72.69%	**76.15%**	4
		DeepSVDD	73.71%	72.70%	69.63%	**75.46%**	67.01%	67.14%	73.27%	**75.46%**	5
		ECOD	71.42%	**72.75%**	67.34%	72.31%	67.91%	66.52%	71.45%	**72.75%**	10
		iForest	72.09%	72.60%	68.41%	74.82%	**76.38%**	70.88%	72.69%	**76.38%**	2
		LOF	72.92%	73.41%	69.26%	**74.70%**	67.42%	69.33%	72.94%	**74.70%**	7
		MCDE	73.17%	71.93%	68.39%	74.31%	**75.31%**	73.00%	72.49%	**75.31%**	6
		OCSVM	71.72%	72.76%	68.01%	**73.61%**	68.58%	66.84%	72.50%	**73.61%**	9
		PCA	70.88%	72.53%	66.96%	**74.07%**	72.15%	68.77%	72.20%	**74.07%**	8
		ClaCO	**84.05%**	75.91%	82.52%	83.54%	67.67%	70.16%	83.37%	**84.05%**	1
COVID	Original Dataset		76.70%	78.00%	71.72%	77.57%	77.61%	**78.07%**	76.89%	**78.07%**	5
	Reduced dataset by	COPOD	73.92%	78.40%	70.54%	77.66%	**78.48%**	74.85%	77.90%	**78.48%**	3
		DeepSVDD	72.90%	73.08%	68.50%	**75.55%**	73.09%	69.35%	75.53%	**75.55%**	10
		ECOD	75.20%	**78.21%**	70.19%	77.33%	77.58%	76.18%	77.74%	**78.21%**	4
		iForest	75.11%	77.85%	70.81%	77.11%	77.37%	75.88%	**77.98%**	**77.98%**	6
		LOF	78.41%	76.12%	71.85%	79.13%	73.55%	73.27%	**80.12%**	**80.12%**	2
		MCDE	75.05%	75.59%	69.99%	75.31%	74.87%	**76.17%**	75.94%	**76.17%**	9
		OCSVM	75.23%	74.76%	70.83%	76.45%	74.12%	69.90%	**77.34%**	**77.34%**	7
		PCA	75.01%	76.03%	70.05%	76.18%	75.88%	76.49%	**77.02%**	**77.02%**	8
		ClaCO	81.40%	75.36%	79.00%	81.62%	74.05%	72.95%	**81.91%**	**81.91%**	1

B. *Individual Comparison of ML Classifiers over Dataset Versions*

In this benchmark, we aim to display how much the performance of the same ML classifier is affected by the several types of dataset configurations. Following Table 4, we simply give the basis point difference between classification performance with ClaCO Model versus full dataset and competing outlier detection methods. We see that; on 5 of the total 11 datasets, ClaCO improved the classification performance of the same classifier over all other versions of the data for every classifier. On the remaining 6 of the datasets; still, ClaCO improved almost all ML classifiers' performance but in several cases; ClaCO couldn't improve the performance. Summary of results presented in Table 5.

Table 5 - Classification performance summary over different data configurations at individual ML Classifier levels. The color green indicates that the respective classifier improves its performance over data reduced with ClaCO; white color is for vice versa (over two tables).

Dataset Name and Version			Ada Boost	ANN	Decision Trees	Random Forest	SVM Linear	SVM RBF	XGBoost
Breast Cancer	Original Dataset		-2.82%	-3.02%	-5.18%	-2.07%	-3.39%	-10.10%	-1.79%
	Reduced dataset by	COPOD	-6.11%	-7.89%	-7.86%	-8.27%	-7.56%	-12.75%	-6.46%
		DeepSVDD	-3.92%	-4.72%	-5.95%	-5.15%	-4.79%	-10.98%	-3.73%
		ECOD	-4.61%	-8.71%	-7.68%	-9.03%	-8.46%	-13.10%	-5.21%
		iForest	-4.63%	-6.25%	-7.58%	-6.64%	-6.46%	-11.66%	-4.01%
		LOF	-5.45%	-7.79%	-7.12%	-6.56%	-5.99%	-12.05%	-4.69%
		MCDE	-4.44%	-6.38%	-6.44%	-5.81%	-5.92%	-11.24%	-3.62%
		OCSVM	-4.58%	-6.12%	-6.94%	-6.30%	-4.12%	-12.08%	-5.12%
		PCA	-3.86%	-3.49%	-7.27%	-5.38%	-4.88%	-11.64%	-5.34%
		ClaCO	**72.80%**	**71.88%**	**72.87%**	**75.08%**	**73.24%**	**71.96%**	**72.03%**
Heart UCI	Original Dataset		-6.73%	-6.36%	-8.39%	-4.32%	-2.47%	-4.31%	-5.75%
	Reduced dataset by	COPOD	-10.87%	-7.64%	-10.14%	-5.11%	-5.64%	-10.30%	-6.12%
		DeepSVDD	-6.71%	-6.29%	-9.18%	-4.52%	-4.19%	-5.97%	-6.48%
		ECOD	-9.15%	-8.32%	-10.09%	-5.26%	-5.49%	-7.92%	-6.66%
		iForest	-7.59%	-7.31%	-8.33%	-4.89%	-4.49%	-6.47%	-6.17%
		LOF	-5.97%	-4.60%	-7.31%	-3.35%	-3.48%	-5.91%	-4.97%
		MCDE	-8.01%	-7.01%	-8.88%	-4.33%	-3.67%	-7.30%	-6.35%
		OCSVM	-6.98%	-4.96%	-8.02%	-4.22%	-3.95%	-5.28%	-6.32%
		PCA	-7.92%	-7.85%	-9.30%	-5.62%	-4.59%	-7.39%	-6.26%
		ClaCO	**85.44%**	**85.52%**	**81.96%**	**84.27%**	**84.95%**	**85.59%**	**84.94%**
Make Blobs	Original Dataset		-5.50%	-3.23%	-8.06%	-5.33%	-1.09%	-3.32%	-6.07%
	Reduced dataset by	COPOD	-6.84%	-3.86%	-8.84%	-6.10%	-2.10%	-7.06%	-6.72%
		DeepSVDD	-4.84%	-2.67%	-5.88%	-4.57%	-1.85%	-2.95%	-4.60%
		ECOD	-6.41%	-3.65%	-9.13%	-5.84%	-1.94%	-6.60%	-6.17%
		iForest	-5.66%	-3.31%	-6.56%	-5.46%	-1.49%	-5.15%	-5.51%
		LOF	-5.03%	-2.96%	-5.75%	-4.74%	-1.33%	-4.44%	-5.26%
		MCDE	-5.34%	-3.13%	-6.66%	-5.00%	-1.26%	-4.76%	-5.24%
		OCSVM	-6.36%	-3.82%	-7.15%	-5.64%	-1.41%	-5.12%	-5.67%
		PCA	-7.36%	-3.75%	-9.32%	-5.78%	-1.79%	-6.42%	-6.48%
		ClaCO	**87.47%**	**87.91%**	**87.38%**	**87.77%**	**85.93%**	**87.97%**	**87.43%**
Titanic	Original Dataset		-5.41%	-2.01%	-7.92%	-4.51%	-4.72%	-2.33%	-4.18%
	Reduced dataset by	COPOD	-8.45%	-5.20%	-10.44%	-7.53%	-7.22%	-6.54%	-7.55%
		DeepSVDD	-5.96%	-3.93%	-7.55%	-4.32%	-5.21%	-5.70%	-3.98%
		ECOD	-8.59%	-5.04%	-10.95%	-7.35%	-6.85%	-5.85%	-7.39%
		iForest	-7.11%	-4.78%	-9.69%	-6.47%	-6.11%	-5.71%	-6.29%
		LOF	-5.62%	-3.24%	-6.42%	-3.17%	-5.09%	-4.38%	-2.49%
		MCDE	-7.14%	-5.18%	-9.50%	-7.25%	-7.67%	-6.74%	-7.46%
		OCSVM	-6.16%	-4.39%	-6.79%	-4.17%	-5.79%	-5.69%	-4.26%
		PCA	-7.76%	-5.01%	-9.13%	-5.77%	-5.18%	-4.74%	-5.95%
		ClaCO	**84.72%**	**82.62%**	**84.07%**	**83.93%**	**82.39%**	**83.01%**	**83.66%**
Weather Rain	Original Dataset		-3.41%	-0.48%	-4.48%	-3.88%	-1.34%	-1.47%	-2.30%
	Reduced dataset by	COPOD	-6.00%	-3.54%	-7.24%	-6.75%	-4.68%	-9.06%	-5.61%
		DeepSVDD	-4.52%	-4.21%	-5.35%	-5.59%	-3.19%	-9.19%	-3.67%
		ECOD	-6.10%	-3.41%	-6.40%	-6.94%	-3.61%	-9.57%	-5.27%
		iForest	-5.80%	-3.79%	-4.80%	-5.72%	-3.74%	-10.08%	-4.67%
		LOF	-4.12%	-2.90%	-3.24%	-4.56%	-2.57%	-7.16%	-2.40%
		MCDE	-5.36%	-4.08%	-4.79%	-5.62%	-4.51%	-9.82%	-3.71%
		OCSVM	-5.06%	-2.51%	-5.89%	-5.69%	-2.86%	-7.77%	-3.36%
		PCA	-5.85%	-4.12%	-5.50%	-5.93%	-3.68%	-11.49%	-4.90%
		ClaCO	**84.89%**	**82.28%**	**82.90%**	**87.07%**	**81.80%**	**83.00%**	**85.74%**

Dataset Name and Version			Ada Boost	ANN	Decision Trees	Random Forest	SVM Linear	SVM RBF	XGBoost
Colon Cancer	Original Dataset		-2.06%	-3.75%	-5.27%	-5.59%	-3.31%	-5.21%	-0.34%
	Reduced dataset by	COPOD	0.85%	-3.82%	-6.61%	-2.62%	-5.11%	-4.99%	-0.88%
		DeepSVDD	-5.99%	-4.19%	-9.97%	-9.14%	-5.65%	-9.10%	-5.74%
		ECOD	-1.31%	-4.12%	-6.17%	-2.44%	-4.97%	-3.69%	0.23%
		iForest	-0.83%	-4.17%	-7.20%	-4.85%	-4.87%	-4.60%	-1.75%
		LOF	-0.18%	-4.25%	-5.23%	-2.65%	-5.73%	-2.79%	-0.49%
		MCDE	-7.16%	-6.55%	-9.78%	-10.62%	-6.79%	-12.45%	-5.95%
		OCSVM	-12.70%	-8.46%	-13.86%	-15.19%	-9.11%	-19.27%	-13.85%
		PCA	-0.27%	-4.27%	-6.54%	-2.99%	-5.75%	-2.96%	-1.71%
		ClaCO	**78.42%**	**82.71%**	**77.44%**	**84.39%**	**86.46%**	**73.29%**	**78.35%**
Lymphoma	Original Dataset		1.12%	-0.15%	-0.89%	-2.08%	-0.35%	-4.02%	1.51%
	Reduced dataset by	COPOD	-3.00%	-5.55%	-5.62%	-9.51%	-7.42%	-9.87%	-4.03%
		DeepSVDD	3.86%	-1.00%	-0.30%	-3.34%	-3.07%	-6.92%	0.31%
		ECOD	-2.70%	-5.20%	-5.19%	-9.12%	-6.86%	-9.14%	-2.67%
		iForest	-3.81%	-4.52%	-5.69%	-8.37%	-6.40%	-10.41%	-2.13%
		LOF	-2.86%	-5.60%	-5.17%	-8.93%	-7.01%	-10.00%	-3.89%
		MCDE	-0.03%	-4.76%	-4.87%	-6.78%	-4.66%	-10.62%	-2.51%
		OCSVM	-7.26%	-6.40%	-6.57%	-13.63%	-9.92%	-19.94%	-5.95%
		PCA	-3.74%	-5.09%	-4.82%	-8.99%	-7.22%	-10.00%	-3.61%
		ClaCO	**54.74%**	**87.53%**	**62.71%**	**76.38%**	**92.23%**	**66.41%**	**68.74%**
Bacteria	Original Dataset		-5.90%	-3.31%	-4.56%	-3.98%	-0.51%	-0.30%	-3.63%
	Reduced dataset by	COPOD	-5.30%	-2.58%	-4.16%	-3.10%	-0.50%	-0.16%	-2.93%
		DeepSVDD	-2.00%	-2.12%	-1.75%	-1.37%	-0.50%	-0.08%	-1.08%
		ECOD	-5.15%	-2.50%	-4.58%	-2.86%	-0.50%	0.02%	-2.69%
		iForest	-2.56%	-2.50%	-2.83%	-1.43%	-0.50%	-0.22%	-1.62%
		LOF	-1.68%	-2.00%	-2.45%	-0.84%	-0.51%	-0.17%	-1.51%
		MCDE	-5.59%	-2.25%	-5.08%	-4.00%	-0.50%	0.01%	-4.27%
		OCSVM	-0.32%	0.41%	-0.20%	1.83%	-0.50%	-0.24%	1.45%
		PCA	-3.71%	-2.82%	-4.49%	-1.61%	-0.50%	-0.16%	-2.13%
		ClaCO	**71.76%**	**69.48%**	**72.40%**	**72.77%**	**66.36%**	**66.08%**	**72.92%**
PBC	Original Dataset		8.24%	-4.94%	-9.95%	-7.02%	-3.63%	2.79%	-6.59%
	Reduced dataset by	COPOD	7.89%	-7.66%	-10.97%	-11.49%	-5.79%	-2.38%	-10.66%
		DeepSVDD	6.38%	-10.58%	-9.13%	-9.01%	-9.54%	-5.50%	-7.95%
		ECOD	7.97%	-6.70%	-11.84%	-11.82%	-6.17%	-3.31%	-11.68%
		iForest	7.23%	-7.07%	-10.33%	-10.32%	-5.59%	-1.69%	-8.90%
		LOF	7.32%	-11.66%	-10.74%	-8.07%	-8.45%	-6.07%	-6.92%
		MCDE	8.06%	-7.49%	-10.65%	-7.95%	-4.77%	0.44%	-6.73%
		OCSVM	7.18%	-11.09%	-9.21%	-8.90%	-7.70%	-6.28%	-7.41%
		PCA	7.63%	-9.38%	-10.38%	-9.11%	-7.34%	-4.21%	-8.78%
		ClaCO	**36.17%**	**53.88%**	**50.72%**	**54.43%**	**52.82%**	**46.36%**	**52.49%**
Pima Diabetes	Original Dataset		-9.58%	-0.31%	-12.56%	-7.86%	8.64%	5.29%	-9.89%
	Reduced dataset by	COPOD	-12.36%	-2.73%	-16.01%	-9.29%	8.48%	-0.78%	-10.68%
		DeepSVDD	-10.34%	-3.20%	-12.90%	-8.07%	-0.66%	-3.02%	-10.10%
		ECOD	-12.63%	-3.15%	-15.18%	-11.23%	0.24%	-3.64%	-11.91%
		iForest	-11.95%	-3.31%	-14.11%	-8.72%	8.71%	0.72%	-10.68%
		LOF	-11.13%	-2.49%	-13.27%	-8.84%	-0.25%	-0.84%	-10.43%
		MCDE	-10.87%	-3.97%	-14.14%	-9.23%	7.64%	2.83%	-10.88%
		OCSVM	-12.32%	-3.14%	-14.52%	-9.93%	0.90%	-3.33%	-10.87%
		PCA	-13.16%	-3.37%	-15.57%	-9.47%	4.48%	-1.39%	-11.17%
		ClaCO	**84.05%**	**75.91%**	**82.52%**	**83.54%**	**67.67%**	**70.16%**	**83.37%**
COVID	Original Dataset		-4.70%	2.64%	-7.29%	-4.05%	3.56%	5.13%	-5.02%
	Reduced dataset by	COPOD	-7.48%	3.05%	-8.46%	-3.97%	4.42%	1.90%	-4.01%
		DeepSVDD	-8.50%	-2.28%	-10.51%	-6.07%	-0.97%	-3.59%	-6.38%
		ECOD	-6.20%	2.85%	-8.81%	-4.30%	3.53%	3.23%	-4.17%
		iForest	-6.29%	2.49%	-8.20%	-4.52%	3.32%	2.93%	-3.93%
		LOF	-3.00%	0.76%	-7.16%	-2.50%	-0.50%	0.33%	-1.79%
		MCDE	-6.36%	0.23%	-9.01%	-6.32%	0.81%	3.23%	-5.98%
		OCSVM	-6.17%	-0.59%	-8.17%	-5.17%	0.07%	-3.05%	-4.57%
		PCA	-6.39%	0.68%	-8.95%	-5.44%	1.82%	3.54%	-4.89%
		ClaCO	**81.40%**	**75.36%**	**79.00%**	**81.62%**	**74.05%**	**72.95%**	**81.91%**

VI. Discussion and Future Directions

In this research paper, we have introduced a novel and efficient graph-based outlier detection model that can work on multiclass tabular datasets. We have found that the ClaCO model improves the classification performance of many well-established ML classifiers over the full dataset and reduced datasets with competing methods. In addition to performance superiority, ClaCO also presents an explainable process for detecting and removing outliers. Within these, since graph data structures are inefficient, time-complexity is an issue for higher sized datasets, and hinders the scalability of the method. As in the current configuration, a dataset having 500 samples, in 5 CV split, takes around ~3 minutes for ClaCO to find outliers.

We believe this work will contribute to data sampling and outlier detection domains in the future. We will continue to seek efficient ways to convert tabular data into graphs, hence achieving also in the scalability of the proposed model.

References

[1] D. M. Hawkins, Identification of Outliers. Dordrecht: Springer Netherlands, 1980.

[2] J. Kong, W. Kowalczyk, S. Menzel, and T. Bäck, “Improving Imbalanced Classification by Anomaly Detection,” in Parallel Problem Solving from Nature – PPSN XVI, vol. 12269, T. Bäck, M. Preuss, A. Deutz, H. Wang, C. Doerr, M. Emmerich, and H. Trautmann, Eds. Cham: Springer International Publishing, 2020, pp. 512–523. doi: 10.1007/978-3-030-58112-1_35.

[3] Y. Zhao, Z. Nasrullah, and Z. Li, “PyOD: A Python Toolbox for Scalable Outlier Detection,” p. 7.

[4] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, “LOF: Identifying Density-Based Local Outliers,” p. 12.

[5] B. Schölkopf, J. C. Platt, J. C. Shawe-Taylor, A. J. Smola, and R. C. Williamson, “Estimating the Support of a High-Dimensional Distribution,” Neural Comput., vol. 13, no. 7, pp. 1443–1471, Temmuz 2001, doi: 10.1162/089976601750264965.

[6] J. Brownlee, “4 Automatic Outlier Detection Algorithms in Python,” Machine Learning Mastery, Jul. 07, 2020. https://machinelearningmastery.com/model-based-outlier-detection-and-removal-in-python/ (accessed Feb. 25, 2022).

[7] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation Forest," in 2008 Eighth IEEE International Conference on Data Mining, Dec. 2008, pp. 413–422. doi: 10.1109/ICDM.2008.17.

[8] P. Rousseeuw and K. Driessen, "A Fast Algorithm for the Minimum Covariance Determinant Estimator," Technometrics, vol. 41, pp. 212–223, Aug. 1999, doi: 10.1080/00401706.1999.10485670.

[9] M. Hubert, M. Debruyne, and P. J. Rousseeuw, "Minimum Covariance Determinant and Extensions," WIREs Comp Stat, vol. 10, no. 3, May 2018, doi: 10.1002/wics.1421.

[10] M.-L. Shyu, S.-C. Chen, K. Sarinnapakorn, and L. Chang, "A Novel Anomaly Detection Scheme Based on Principal Component Classifier," p. 8.

[11] L. Ruff et al., "Deep One-Class Classification," in Proceedings of the 35th International Conference on Machine Learning, Jul. 2018, pp. 4393–4402. Accessed: Apr. 11, 2022. [Online]. Available: https://proceedings.mlr.press/v80/ruff18a.html

[12] Z. Li, Y. Zhao, N. Botta, C. Ionescu, and X. Hu, "COPOD: Copula-Based Outlier Detection," 2020 IEEE International Conference on Data Mining (ICDM), pp. 1118–1123, Nov. 2020, doi: 10.1109/ICDM50108.2020.00135.

[13] Z. Li, Y. Zhao, X. Hu, N. Botta, C. Ionescu, and G. H. Chen, "ECOD: Unsupervised Outlier Detection Using Empirical Cumulative Distribution Functions," arXiv:2201.00382 [cs, stat], Mar. 2022, Accessed: Apr. 11, 2022. [Online]. Available: http://arxiv.org/abs/2201.00382

[14] X. Ma et al., "A Comprehensive Survey on Graph Anomaly Detection with Deep Learning," arXiv:2106.07178 [cs], Oct. 2021, Accessed: Feb. 14, 2022. [Online]. Available: http://arxiv.org/abs/2106.07178

[15] L. Akoglu, M. McGlohon, and C. Faloutsos, "oddball: Spotting Anomalies in Weighted Graphs," in Advances in Knowledge Discovery and Data Mining, vol. 6119, M. J. Zaki, J. X. Yu, B. Ravindran, and V. Pudi, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 410–421. doi: 10.1007/978-3-642-13672-6_40.

[16] N. Liu, X. Huang, and X. Hu, "Accelerated Local Anomaly Detection via Resolving Attributed Networks," pp. 2337–2343, 2017.

[17] K. Ding, J. Li, R. Bhanushali, and H. Liu, "Deep Anomaly Detection on Attributed Networks," 2019, pp. 594–602. doi: 10.1137/1.9781611975673.67.

[18] S. Bandyopadhyay, L. N, S. V. Vivek, and M. N. Murty, "Outlier Resistant Unsupervised Deep Architectures for Attributed Network Embedding," in Proceedings of the 13th International Conference on Web Search and Data Mining, New York, NY, USA: Association for Computing Machinery, 2020, pp. 25–33. Accessed: Feb. 14, 2022. [Online]. Available: https://doi.org/10.1145/3336191.3371788

[19] Serkan ÜÇER, Tansel ÖZYER, and Reda ALHAJJ, "Tabular Data To Network Graph (TD2NG): A Visual Exploratory Data Analysis Technique For Supervised Learning," presented at the International Informatics Congress (IIC2022) Batman, Turkey, Feb. 2022.

[20] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," J. Stat. Mech., vol. 2008, no. 10, p. P10008, Oct. 2008, doi: 10.1088/1742-5468/2008/10/P10008.

[21] "Titanic - Machine Learning from Disaster." https://kaggle.com/c/titanic (accessed Feb. 14, 2022).

[22] T. M. J. Fruchterman and E. M. Reingold, "Graph drawing by force-directed placement," Software: Practice and Experience, vol. 21, no. 11, pp. 1129–1164, 1991, doi: 10.1002/spe.4380211102.

[23] G. Lemaıtre and F. Nogueira, "Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning," p. 5.

[24] F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," J. Mach. Learn. Res., vol. 12, no. null, pp. 2825–2830, Nov. 2011.

[25] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 785–794, Aug. 2016, doi: 10.1145/2939672.2939785.