Data-Agnostic Local Neighborhood Generation

Experiments Details

Riccardo Guidotti
University of Pisa
Pisa, Italy
riccardo.guidotti@unipi.it

Anna Monreale
University of Pisa
Pisa, Italy
anna.monreale@unipi.it

EXPERIMENTS

We report here detailed experiments proving that the proposed approach returns efficiently instances more realistic than the state-of-the-art generators[1]. After presenting the experimental setting, we show the effectiveness of DAG and the usefulness of the evaluation framework for local neighborhood generation through *(i)* a qualitative comparison of synthetic neighborhoods, and *(ii)* a quantitative comparison of the metrics of the evaluation framework with statistical evidence of the results. In addition, we also report the average running time for the local neighborhood generation. Experiments are performed on Ubuntu 16.04 LTS, 32 GB RAM, 3.30GHz Intel Core i7 with a GPU NVidia GTX1080Ti 11G.

A. Experimental Setting

We experimented with nine tabular datasets[2], six time series datasets, three image datasets, and two text datasets. Table I reports the dataset details. All the datasets are open source and freely available.[3] We turned the RGB images of `cifar10` into black and white images. In textual datasets, we removed the punctuation, and we limited the generators with a vocabulary composed by the 1,000 most frequent words across the corpus identified by the training set. For `20news` we only considered the two classes *atheism* and *religion*.

The datasets are partitioned into training and test following the suggested partition from the literature [1], or using a 70%-30% hold-out partitioning stratified with respect to the class attribute. For each dataset we used the training partition X for *(i)* randomly selecting the support sets S, *(ii)* training the algorithms of the evaluation framework. In particular, we used X for training the GANs, for training the center-based clustering algorithm, and for calculating the outlier scores. On the other hand, we executed DAG and its competitors on 100 randomly selected instances from each test set. In the results we report the mean value obtained for each test instance.

TABLE I
DATASETS DESCRIPTION.

type	name	training	test	dimensions	labels
tabular	`ctg`	1488	638	35	10
	`avila`	14537	6231	10	10
	`ctg`	1488	638	35	10
	`diabetes`	537	231	8	2
	`ionoshpere`	245	106	34	2
	`parkinsons`	136	59	22	2
	`sonar`	145	63	60	2
	`vehicle`	592	254	18	4
	`wdbc`	398	171	18	4
time series	`arrowhead`	36	175	251	3
	`ecg200`	100	100	96	2
	`electric`	8926	7711	26	7
	`gunpoint`	50	150	150	2
	`italypower`	67	1029	24	2
	`phalanges`	1800	858	80	2
images	`cifar10`	50k	10k	32×32	10
	`fashion`	60k	10k	28×28	10
	`mnist`	60k	10k	28×28	10
text	`20news`	11k	7k	1000	2
	`imdb`	25k	25k	1000	2

We instantiate algorithms and select parameters of the evaluation framework as follows. As conditional GAN for tabular data we used the CTGAN[4] described in [2]. For time series and images we used the conditional GAN[5] described in [3]. For text we used a conditional GAN based on LSTM[6]. As discriminators we used a Random Forest (RF) classifier

[1]The source code is available at: LinkAvailableAtCameraReady.

[2]We restricted to datasets with continuous features but DAG also works for categorical attributes with appropriate distance functions or one-hot encoding.

[3]Dataset links: time series https://www.cs.ucr.edu/~eamonn/time_series_data_2018/ accessed with `tslearn`, image datasets `mnist` http://yann.lecun.com/exdb/mnist/, `fashion` https://www.cs.toronto.edu/~kriz/cifar.html, `cifar10` https://www.kaggle.com/zalando-research/ accessed with `keras`, tabular datasets downloaded from https://archive.ics.uci.edu/ml/datasets.php, textual datasets `20news` http://qwone.com/~jason/20Newsgroups/ accessed with `scikit-learn`, and `imdb` https://www.imdb.com/interfaces/ accessed with `keras`, respectively.

[4]https://github.com/Diyago/GAN-for-tabular-data/tree/master/ctgan. Parameters: epochs 300, embedding dimension 128, generator dimensions 256 (two layers), discriminator dimensions 256 (two layers), batch size 500

[5]https://github.com/eriklindernoren/Keras-GAN#cgan. For time series we used a 1 dimensional convolutional neural network with 10k epochs training while for images a 2 dimensional convolutional neural network with 20k epochs training. For both time series and images the GAN has the following properties: embedding dimension 100, generator dimensions 256, 512, 1024 with batch normalization momentum 0.8 and tanh activation function, discriminator dimensions 512 (three layers) with dropout at 0.4 and sigmoid activation function, for both generator and discriminator LeakyReLU activation for hidden layers with alpha 0.2, batch size 32.

[6]Inspired by https://keras.io/examples/lstm_text_generation/ the generator uses an LSTM with padded text with max length 1k, embedding dimension 100, softmax activation function, batch size 128, 100 epochs.

TABLE II

EXAMPLE OF LOCAL NEIGHBORHOODS FOR DIABETES DATASET GENERATED FROM THE SUPPORT SET S SUMMARIZED BY MEAN, STD DEV (STD), MIN AND MAX. THE TEST RECORD x IS IN THE FIRST ROW.

features		Age	Blood Pressure	Glucose	BMI
x		45.00	84.00	111.00	46.80
Support Set	mean	28.00	72.20	121.20	37.91
	std	6.66	13.90	30.76	5.28
	min	21.00	46.00	84.00	28.70
	max	38.00	100.00	187.00	43.40
DAG	mean	36.37	78.21	116.27	42.43
	std	8.64	10.19	20.53	5.21
	min	21.00	46.00	84.00	28.70
	max	45.00	100.00	187.00	46.80
RND	mean	36.20	77.44	116.34	41.93
	std	9.70	11.79	21.52	6.00
	min	21.00	46.00	84.00	28.70
	max	45.00	100.00	187.00	46.80
SUP	mean	36.55	78.35	116.24	42.52
	std	8.50	5.89	5.10	4.44
	min	28.00	72.20	111.00	37.91
	max	45.00	84.00	121.20	46.80
NRM	mean	36.18	77.92	116.22	42.25
	std	9.57	11.03	20.78	5.77
	min	9.38	21.27	29.63	24.51
	max	46.64	114.64	225.10	51.46
GAN	mean	23.53	63.53	105.55	39.78
	std	6.50	10.95	23.51	4.75
	min	6.20	33.82	36.23	25.58
	max	45.89	97.22	177.53	53.32

and a Neural Network (NN) as implemented in *scikit-learn*[7]. The discriminators are trained and act on the data representation used by the evaluation framework. Discriminators accuracy on test sets varies from 80% to 100%. As center-based clustering algorithm we selected k-means [4] with $k = \max(5 \cdot nbrclasses, 10)$, i.e., we assume that for each dataset there are at least ten different mechanisms generating the data present in the dataset. We leave for future work an ad-hoc selection of the number of clusters for each dataset. We underline that any other prototype based clustering algorithm can be used. As outlier detection we used LOF [5] with $k = 5$ as implemented in *scikit-learn*[8]. For every metric we report aggregated values over datasets and/or methods.

We compare DAG against the following state-of-the-art local neighborhood generators and baselines:

- RND. Given the support data S and x, it creates synthetic instances Z by changing the values of randomly selected features of x with the corresponding values of a randomly selected instance $s \in S$. It is adopted in [6].
- SUP. Given S and x it creates synthetic instances Z by randomly changing the values of randomly selected features of x with a predefined base value. It uses the mean value of the data in S for tabular data and time series, the color black for images and the empty string for text. It is adopted in [7], [8] for images and texts.
- NRM. Given S and x, it creates synthetic instances Z by changing the values of randomly selected features of x with a values extracted from a normal distribution with mean and standard deviation estimated from S. It is adopted in [7], [8] for tabular data.
- GAN. Given S and x, it trains a conditional GAN on S and generates synthetic instances according to the GAN principle. We adopt GANs with the same structure of those described above[9]. A similar approach based on adversarial autoencoder is used in [9] for images.

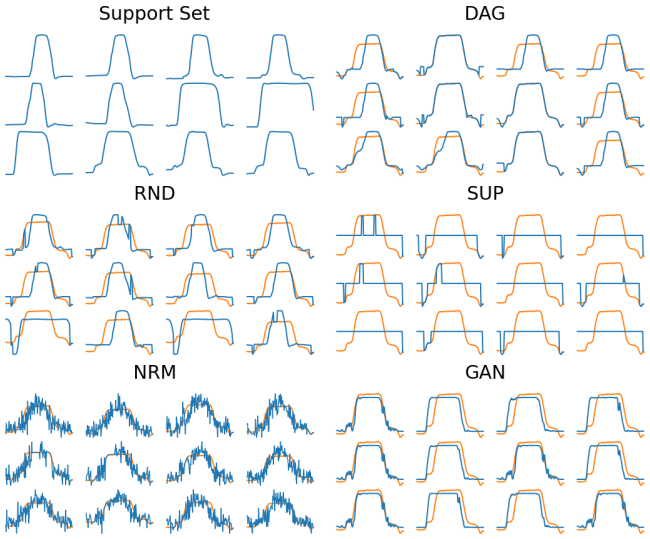

Fig. 1. Example of local neighborhoods for the `gunpoint` dataset generated from the support set S (top left). The test time series x is reported in orange under each synthetic instance.

We equip RND, SUP, and NRM with the same data transformation specified for DAG. In the experiments we vary the size $|S|$ of the support set S in $\{10, 25, 50, 100, 250, 500, 1000, 2500, 5000, 10000\}$. If not differently specified we report results for $|S| = 100$ and, for each test instance we generate $n = 1000$ synthetic neighbors. In the experiments we use by default $p = 0.5$ for the *Blend* and *MutationBlend* operators. We apply data transformation γ with $w = 4$ for time series, $w = w' = 7$ for images, and $w = 3$ for texts.

B. Qualitative Evaluation

Before presenting the quantitative evaluation, in this section we illustrate a qualitative comparison of the local neighborhood for three randomly selected test instances for the `diabetes`, `gunpoint` and `mnist` dataset. We omit an example for a text dataset for space reasons. In this experiment we fix the support set size $|S| = 12$ for every dataset.

Table II reports the example for the tabular dataset `diabetes`. For simplicity we focus on four features (Age,

[7]https://scikit-learn.org/stable/ Random Forest parameters: 100 estimators, Gini criterion, 0.02 min samples split, 0.01 min samples leaf. Neural Network with hidden layers with dimensions 32, 64, 128, ReLU activation function, adaptive learning rate and early stopping.

[8]We used $k = 5$ for LOF because it is a sufficiently small number to account for small neighbor to find outliers, and also because it is shown in [5] that 5,6 and 7 are reasonable values in many situations.

[9]The number of epochs is reduced to 300 for tabular data, 1k for time series, 2k for images and 10 for text.

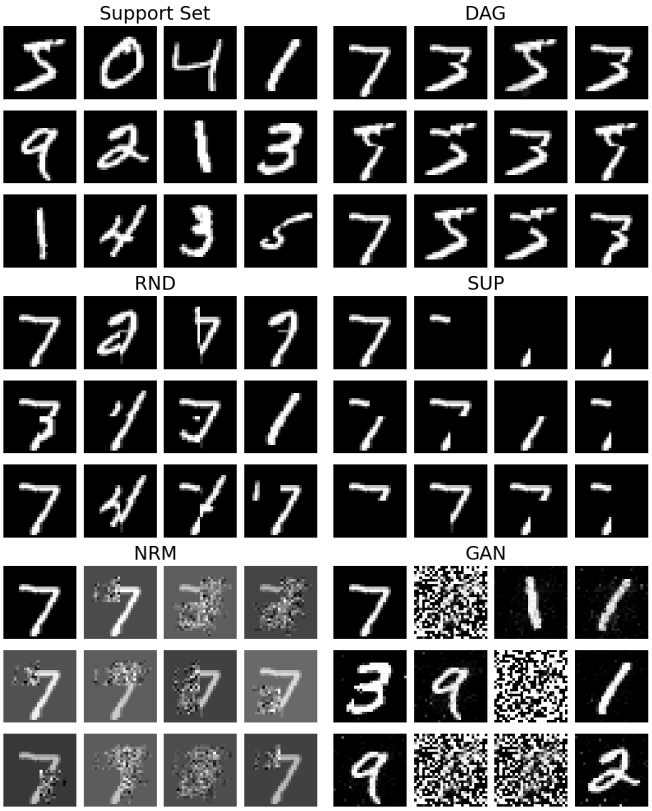

Fig. 2. Example of local neighborhoods for the `mnist` dataset generated from the support set S (top left). The test image x is on the top left corner of each synthetic neighborhood.

Blood Pressure, Glucose and BMI) out of eight, and we summarize the support set using the mean, standard deviation (std), min and max of each feature. The core record x is reported in the first row. We notice that DAG and RND have the same min and max values for all the features except Age, for which they have a different maximum value that matches with the Age of x. SUP, NRM and GAN wide these boundaries generating records which easily will be not useful for understanding aspects related to x with respect to S. In addition, DAG neighborhood seems better than the neighborhood of RND because, even though in both cases the mean values are moved towards the values of x, DAG has a lower standard deviation signaling the higher compactness and similarity with x.

Figure 1 illustrates examples of synthetic neighborhoods for the `gunpoint` dataset. We report twelve randomly selected synthetic time series for each method. The core time series x is reported in orange under every synthetic time series. The synthetic time series generated by DAG resemble real time series as much as those generated by the GAN. In particular, DAG synthetic time series retrace the core time series inserting slight modification. These slight modifications can be precious for explaining a black box behavior in the surroundings of x. The synthetic time series of RND suffers from some imperfections that them unrealistic if compared to real instances in the training set. For instance, they reveal sharp changes of the trend or a missing expected pattern in the series, i.e., it stays constant when it is expected to grow or decrease. Time series generated by SUP and NRM are clearly fake. They could be useful for some adversarial study but certainly not for locally explaining black box classifiers because a black box classifier is not trained on instances so different from the real ones. Finally, GAN tends to generate times series very similar to the original one, but we observe that among synthetic instances there is a low level of diversity that makes the neighborhood not suitable for learning local interpretable models often used for explanation.

In Figure 2 we show the synthetic neighborhoods for the `mnist` dataset. For each neighborhood we report eleven synthetic images; the core image x is the "7" reported on the top left corner. DAG synthetic images seems quite realistic if they would have been written from people with a worse calligraphy than those of the original dataset. A possible advantage from DAG neighborhood is that the synthetic images only seem 3s, 5s or 7s showing what can be achieved from the surrounding of x and from the support set S. Also the images returned by RND are not bad even though in some cases there are sudden jumps from black to white areas. The synthetic images of SUP are clearly fake as it just overlap some pixels with black areas, while those of NRM introduce too much noise in the images. Similarly to the time series, also the synthetic images generated by SUP and NRM are not useful for local black box explanation as they differ too much from the training set used by a black box classifier. Finally, GAN theoretically generates synthetic images superior to the other basic methods that works with simpler operators. However, when the support set S, which in this case corresponds with the training set for the GAN, is small, and/or the GAN is not trained for a sufficiently high number of epochs to speed-up the local generation process, then the result is not optimal. Moreover, the synthetic images are either copies of those in the support set (i.e., the GAN does not generalize), or produces random noise (i.e., perhaps the GAN it is under-trained).

C. Quantitative Evaluation

In this section we simultaneously analyze the performance of DAG against the competitors, and show the usefulness of the evaluation framework for local neighborhood generators.

In Figure 3 we report all the evaluation metrics calculated by the framework (expect *avg error* for space reasons) one for each rows, for `wdbc`, `ecg200`, `cifar10` and `imdb` datasets (one for each column), varying the support set size $|S| \in \{10, 25, 50, 100, 250, 500, 1000, 2500, 5000, 10000\}$. We selected a dataset per type but similar behaviors can be observed for other datasets of the same type. We notice that, in general, even though the size of the support set impacts on the various metrics, the trend only has small fluctuations for each method. In other words, we do not observe that different values of the support set size $|S|$ make a method to perform better or worse than another one. Regarding the *RF error*, DAG is the best performer for `wdbc` and second best for `ecg200`. For `fashion` no method can deceive

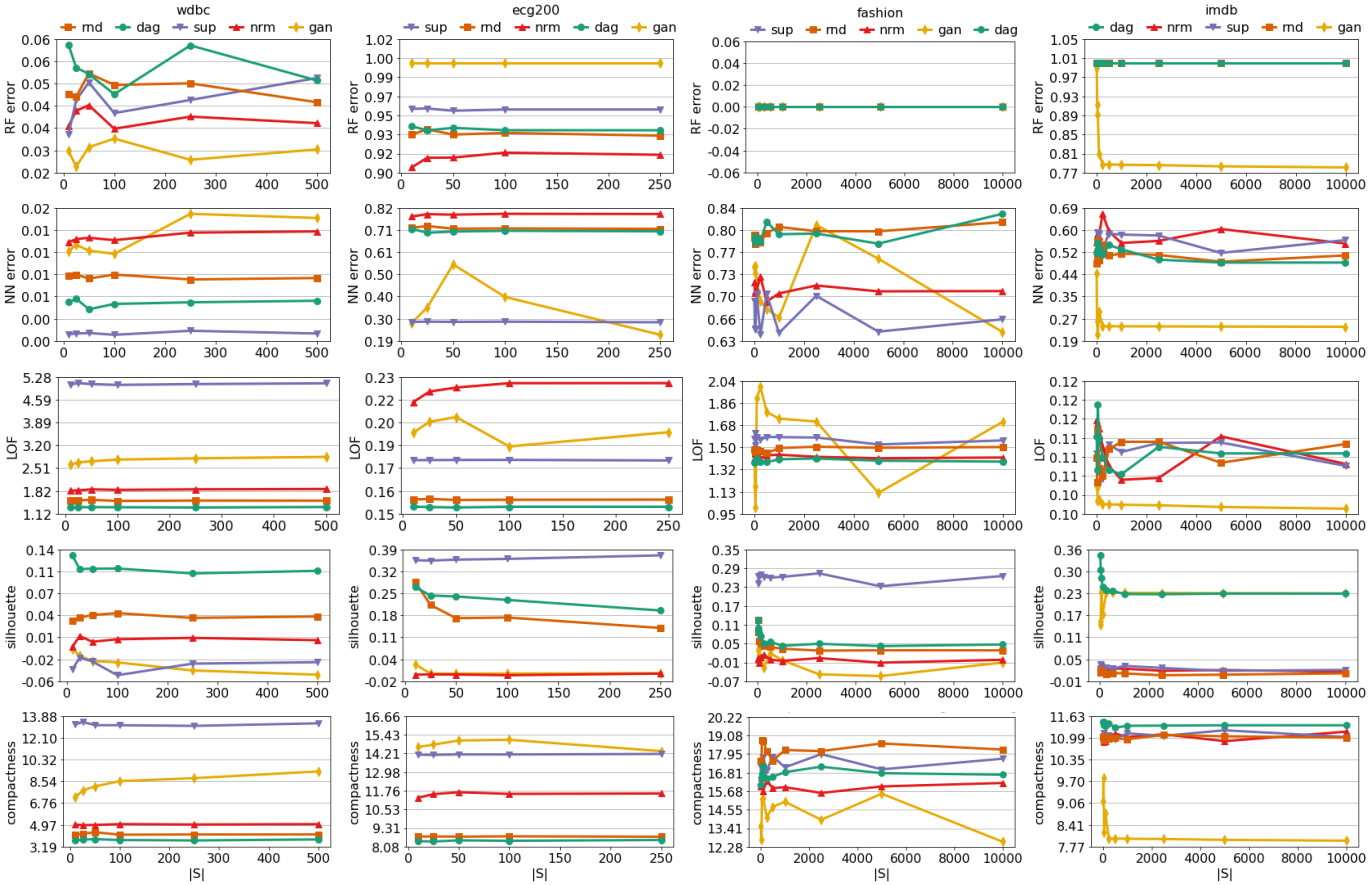

Fig. 3. Evaluation metrics (different rows) for `wdbc`, `ecg200`, `cifar10` and `imdb` datasets (different columns) varying the support set size $|S|$.

the RF discriminator, while for the `imdb` dataset all the methods completely fool the RF. The same does not happen for the *NN error* where DAG is among the bests for `ecg200` and `fashion`, and comparable for `wdbc`. Concerning the unsupervised evaluation metrics DAG behaves markedly better than the other methods. Indeed, the LOF of DAG is stably the lowest for `wdbc`, `ecg200`, and `fashion`, while for `imdb` it is comparable to the best performer NRM. However, we remind the reader the very unrealistic images generated by NRM also quantitatively proved by the lowest RF and *NN error*. DAG has the highest *silhouette* for `wdbc` and `imdb`, and has a comparable performance with the others for `ecg200` and `fashion`. For every dataset we notice a decrease in the *silhouette* of DAG with the increasing of $|S|$. Finally, DAG shows the highest compactness (lowest values) for `wdbc` and `ecg200` but the lowest compactness for `imdb`.

The mean values for each dataset type and for each metric are reported in Table III. DAG has the overall best results for the aggregated metrics with a standard deviation comparable with those of the others or often even smaller indicating higher stability. Best results are underlined. In Table IV we summarize the results by reporting the mean rank values (ranging from 1 to 5) among the different generators for a given dataset varying the support set size $|S|$ over all the metrics of the evaluation framework. The row reports the overall mean rank. It is evident that DAG ranks the best in general (p-value < 0.01 using a Wilcoxon signed rank test), and for each dataset. For the `wdbc`, `diabetes`, `ctg`, `parkinsons`, `sonar`, `avila`, `ecg200` and `imdb` datasets, DAG ranks markedly higher than the competitors. Finally, the last columns of Table III reports the average running time (in secs) for generating a local neighborhood. DAG performance are slightly worse than the other baselines for tabular data, time series and images, and slightly better than the competitors for text data. However, DAG running time is always markedly lower than GAN that is the most effective method for verisimilitude of the fake instances with the real ones for $|S|$ large enough.

Figure 4 shows the Critical Difference (CD) diagrams [10]. It displays the statistical significance of the observed paired differences in performances between pairs of local neighborhood generation methods. Two methods are tied if the null hypothesis that their performances are the same cannot be rejected using the Nemenyi test at $\alpha = 0.05$. We observe the results separately for each evaluation measure on the various datasets. DAG performs better than the compared methods with regards to *RF error*, *LOF*, *compactness* and *silhouette*, also the differences are statistically significant. For the other metrics, this does not hold, but the methods tied to DAG are always

TABLE III

MEAN VALUES AND STANDARD DEVIATION FOR EACH DATASET TYPE AND FOR EACH METRIC FOR $|S| = 100$. BEST RESULTS ARE UNDERLINED.

datatype	method	*RF error*	*NN error*	*LOF*	*compactness*	*silhouette*	*avg error*	*running time (sec)*
tabular	GAN	0.250 ± 0.38	0.152 ± 0.18	3.651 ± 3.00	9.388 ± 9.29	0.004 ± 0.05	0.211 ± 0.22	4.706 ± 1.43
	DAG	0.329 ± 0.41	0.229 ± 0.31	1.432 ± 0.36	3.253 ± 1.44	0.140 ± 0.03	0.213 ± 0.10	0.097 ± 0.02
	NRM	0.290 ± 0.39	0.179 ± 0.21	1.903 ± 0.52	4.286 ± 1.80	0.013 ± 0.04	0.267 ± 0.21	0.058 ± 0.02
	RND	0.308 ± 0.41	0.202 ± 0.25	1.637 ± 0.49	3.813 ± 1.68	0.053 ± 0.05	0.131 ± 0.14	0.028 ± 0.01
	SUP	0.291 ± 0.43	0.178 ± 0.27	5.147 ± 5.39	11.657 ± 9.34	0.009 ± 0.10	1.484 ± 1.26	0.020 ± 0.01
time series	GAN	0.393 ± 0.54	0.262 ± 0.25	0.000 ± 0.00	21.077 ± 18.63	0.004 ± 0.01	0.143 ± 0.14	46.331 ± 34.34
	DAG	0.589 ± 0.37	0.658 ± 0.41	1.498 ± 3.35	15.283 ± 14.18	0.189 ± 0.12	0.713 ± 0.97	1.467 ± 0.53
	NRM	0.564 ± 0.45	0.655 ± 0.41	1.508 ± 3.37	17.729 ± 12.24	-0.007 ± 0.04	0.716 ± 0.98	1.022 ± 0.39
	RND	0.504 ± 0.37	0.656 ± 0.40	1.252 ± 2.80	15.488 ± 13.55	0.150 ± 0.12	0.734 ± 0.99	0.911 ± 0.36
	SUP	0.397 ± 0.47	0.282 ± 0.25	0.698 ± 1.56	20.678 ± 18.42	0.089 ± 0.22	0.236 ± 0.27	0.829 ± 0.33
image	GAN	0.000 ± 0.00	0.057 ± 0.10	25.463 ± 35.63	72.973 ± 82.28	0.064 ± 0.11	1.550 ± 0.76	214.110 ± 179.27
	DAG	0.072 ± 0.12	0.390 ± 0.35	1.375 ± 0.05	17.580 ± 3.26	0.130 ± 0.03	0.075 ± 0.00	0.489 ± 0.12
	NRM	0.055 ± 0.09	0.385 ± 0.26	1.424 ± 0.06	16.191 ± 2.13	0.035 ± 0.02	0.112 ± 0.08	0.287 ± 0.08
	RND	0.060 ± 0.16	0.406 ± 0.35	1.430 ± 0.11	18.842 ± 3.65	0.156 ± 0.03	0.052 ± 0.03	0.171 ± 0.04
	SUP	0.156 ± 0.27	0.408 ± 0.33	1.536 ± 0.05	20.934 ± 9.73	0.333 ± 0.08	0.221 ± 0.24	0.192 ± 0.04
text	GAN	0.766 ± 0.07	0.338 ± 0.06	2.626 ± 3.71	10.980 ± 3.15	0.172 ± 0.00	0.019 ± 0.02	462.607 ± 30.59
	DAG	1.000 ± 0.00	0.564 ± 0.04	2.421 ± 3.42	12.788 ± 2.06	0.267 ± 0.02	0.075 ± 0.10	30.580 ± 5.67
	NRM	1.000 ± 0.00	0.563 ± 0.02	3.833 ± 5.42	11.181 ± 0.40	0.036 ± 0.00	0.105 ± 0.04	43.226 ± 3.33
	RND	0.999 ± 0.00	0.505 ± 0.02	2.591 ± 3.66	12.513 ± 2.10	0.071 ± 0.08	0.041 ± 0.05	43.704 ± 4.16
	SUP	1.000 ± 0.00	0.579 ± 0.01	3.762 ± 5.32	11.213 ± 0.30	0.032 ± 0.01	0.152 ± 0.08	43.875 ± 3.23

TABLE IV

AVERAGE RANK FOR DATASET WITH RESPECT TO EVALUATION METRICS VARYING $|S|$. BEST RESULTS ARE UNDERLINED.

dataset	DAG	RND	NRM	GAN	SUP
overall	2.445	2.913	3.263	3.969	3.213
wdbc	1.694	2.500	3.472	4.139	4.028
diabetes	1.619	2.214	3.286	3.881	4.833
ctg	1.938	2.417	3.021	4.000	4.458
ionoshpere	2.400	3.367	3.533	4.033	2.500
parkinsons	1.167	2.433	3.600	4.233	4.400
sonar	1.533	2.633	2.800	4.633	4.233
vehicle	2.405	2.619	3.286	3.690	3.833
avila	2.533	3.083	3.650	4.317	2.250
gunpoint	2.750	3.042	3.417	4.167	2.458
italypower	3.708	3.667	3.625	2.667	2.167
ecg200	2.267	2.900	4.333	3.467	2.867
phalanges	2.708	2.958	2.812	3.688	3.667
electric	3.133	3.800	4.167	2.400	2.333
arrowhead	2.389	2.167	3.667	4.333	3.278
mnist	3.115	3.117	3.783	4.600	2.567
fashion	3.400	3.683	3.333	4.200	3.050
cifar10	2.800	3.300	2.975	5.167	3.783
20news	2.952	3.214	3.262	2.778	3.381
imdb	2.048	3.233	3.317	2.100	3.217

different. Hence, DAG wins over any other method in at least four out of the six metrics.

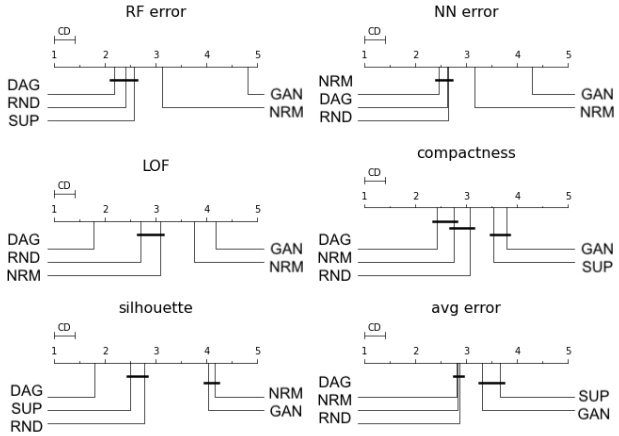

Fig. 4. CD of differences in performance with Nemenyi test at $\alpha = 0.05$.

REFERENCES

[1] D. Dua and C. Graff, "UCI machine learning repository," 2017.

[2] L. Xu *et al.*, "Modeling tabular data using conditional gan," in *NIPS*, 2019, pp. 7333–7343.

[3] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[4] P.-N. Tan *et al.*, "Introduction to data mining," 2005.

[5] M. M. Breunig *et al.*, "Lof: identifying density-based local outliers," in *SIGMOD*, 2000, pp. 93–104.

[6] R. Guidotti *et al.*, "Investigating neighborhood generation methods for explanations of obscure image classifiers," in *PAKDD*. Springer, 2019.

[7] M. T. Ribeiro *et al.*, "Why should i trust you?: Explaining the predictions of any classifier," in *KDD*. ACM, 2016, pp. 1135–1144.

[8] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *NIPS*, 2017, pp. 4765–4774.

[9] R. Guidotti *et al.*, "Black box explanation by learning image exemplars in the latent feature space," in *ECML-PKDD*. Springer, 2019.

[10] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine learning research*, vol. 7, no. Jan, pp. 1–30, 2006.