Assignment 5: Evaluating unsupervised anomaly detection algorithms

Daniela Herrera-Montes de Oca[a01350883] and Pedro Esteban Chavarrias-Solano[a00344305]

Tecnológico de Monterrey, Monterrey NL 2501, Mx.

Abstract. One-class classifiers have shown a good performance when implemented for classification tasks as faults detection, bot detection in social networks or DDOS attacks detection. In this study, a comparative statistical analysis was developed in order to evaluate the performance between four variations of an algorithm called Bagging-Random Miner, with Euclidean, Manhattan, Minkowski, and Cosine distances and three different one-class classifiers: one-class Support Vector Machine, Gaussian Mixture Model, and Isolation Forest. Each of the algorithms was evaluated over 60 Anomaly Detection datasets without pre-processing, using Min-Max scaling and using Standard normalizing, by computing the Area Under the Curve metric. The results were used to generate box plots for visualization, to perform a Friedman statistical test with a pos-thoc test in order to create a CD diagram.

Keywords: BRM, one-class classifiers, anomaly detection datasets, statistical tests.

1 Introduction

Nowadays, the development of advanced machine learning algorithms has enabled the implementation of intelligent systems capable of doing decision-making in similar or even better ways than humans. However, the number of challenges these algorithms have to face is huge. One of the main concerns is to obtain data of abnormal situations, those that are unlikely to happen, for example in masquerade detection or DDOS attacks detection. The use of one-class classifiers has shown an outstanding performance when compared against multi-class classifiers in masquerade detection problems [1]. In this work, a comparative statistical analysis was done to compare the performance of the following classifiers: Bagging-Random Miner using Euclidean distance, Bagging-Random Miner using Manhattan distance, Bagging-Random Miner using Minkowski distance, Bagging-Random Miner using Cosine distance, Gaussian Mixture Model, one-class Support Vector Machine, and Isolation Forest. These algorithms were evaluated over 60 datasets about anomaly detection using the Area Under the Curve (AUC) evaluation metric. Once all the algorithms were evaluated over the 60 datasets, a box plot was generated with the AUC values computed in order to visualize the distribution of the performances obtained by each method. Then, a Friedman statistical test with a Wilcoxon post-hoc test was done in order to establish a statistical comparison between all the algorithms. Finally, a Critical Difference (CD) diagram was created based on the Friedman statistical test. This process was repeated three times, with the original dataset, with a data pre-processing technique called Min-max scaling and with another data pre-processing technique known as Standard normalizing.

2 Methodology

2.1 Dataset preparation

Once the 60 datasets have been downloaded, a function named data2DF inputs the file name of the dataset of interest. Since the datasets are in Keel format, this function process the data in order to create a pandas dataframe that will be later used to train and test the models mentioned above. The algorithm 1 presented in this section, describes the implemented function in detail.

Algorithm 1 data2DF(pathFile)

```
data = create a list of lists containing every row of dataset file
c = save the index value of file containing @ data
columns_c = generate a list of attributes names
columns = delete ',' signs
data = create an array with the data
data = split values when a ',' is found
data_array = create an array with the data variable processed
data_df = generate a dataframe with saved columns names and processed data

for column in columns do
    if data_df [column] is a digit then
        set type of data_df [column] to float
return data_df
```

This function was created in order to work in another function called Model_test, which will be explained in subsection 2.4. Model_test will evaluate the AUC of each of the selected models over the 60 datasets and save the results in a csv file.

2.2 Modification of BRM implementation

The Bagging-Random miner implementation was modified so it could work with any dissimilarity measure. Three dissimilarity measures added were: Manhattan, Minkowski (with $p = 0.5$), and cosine distance. A method called dissimilarity was attach to the class BRM (algortihm 2). This method returns the dissimilarity measure depending on the attribute dissimilarity measure, which was also added in the constructor. The default value is the euclidean distance. The functions that obtain the distance are from the sklearn.metrics.pairwise and scipy.spatial.distance libraries.

Algorithm 2 dissimilarity(self, dissimilarity_measure, x, y)

```
if dissimilarity_measure == 'euclidean' then
    return euclidean_distances(x, y)
if  dissimilarity_measure == 'manhattan'  then
    return manhattan_distances(x, y)
if dissimilarity_measure == 'minkowski' then
    return cdist(x, y, metric = 'minkowski', p = 0.5)
if dissimilarity_measure == 'cosine' then
    return cosine_distances(x, y)
```

2.3 Data pre-processing

As mentioned before, the selected models have to be evaluated using the original dataset information and also evaluated using the data with a pre-processing technique called Min-max scaling and another one called Standard normalizing. For both cases the sklearn pre-processing library was used. In the case of Min-max scaling the datasets go through the method fit_transform which computes the minimum and maximum values and transforms the data into this feature range. For the case of Standard normalizing the resultant datasets of the Min-max scaling method, go through the method fit_transform of the StandardScaler function which computes the mean and unit variance in order to remove the mean and scale to unit variance. This procedures were implemented in a function called splitdataset, which inputs both train and test dataframes. This splitdataset function is also executed within the Model_test function that is going to be explained in the next section 2.4.

2.4 One-class classifiers evaluation

For evaluating the classifiers the function Model_test was used (algortihm 3). The inputs are the classifier(clf_classif), its name (name_classifier), and the root directory (rootDir). The function is a loop that goes through all the folders found in the root directory. For each folder that has files it assigns the train and the test file. This file are processed using the function splitdata explained in the previous section 2.3. The classifier is trained and tested, the avaluatioon is made using the Area Under the ROC Curve. The results are finally saved in a CSV file.

Algorithm 3 Model_test(classifier, name_classsifier, rootDir)

```
for files in rootDir  do
    trainFile
    testFile
    if len(fileList) > 0 then
        for i in range(1, int(len(fileList) / 2) 1) do
            train, test = data2DF(trainFile), data2DF(testFile)
            Train data with different processing = splitdataset(train, test)
            clf_classif.fit(X_train_minmax, y_train)
            y_pre_classif = prediction(X_test_minmax, classifier)
            auc = metrics.roc_auc_score(y_test, y_pred_classif)
AUC saved in csv file
```

2.5 Box plots generation

The comparison between classifiers (one-class Support Vector Machine, Gaussian Mixture Model, and Isolation Forest, Bagging-Random with euclidean, Manhattan, Minkowski and Cosine distance) AUC results was made through boxplots. The graphs were made using the seaborn library. Three boxplots were made, one for each type of data processing (raw data, minmax and Standard normalizing).

2.6 Friedman statistical with Wilcoxon posthoc test

The results from each classifier were analyzed using a Friedman with Wilcoxon posthoc test (algorithm 4). First the Friedman test is made using the AUC classifier results. This give us the statistic and the p value to know if there is a significant difference. Then a Wilcoxon test is made between each classifier, so we can know between which ones the significant difference is found. The output of this algorithm is a dataframe with the statistic and p value for each pair of classifiers.

Algorithm 4 Friedman statistical with Wilcoxon posthoc test

```
f_test = friedmanchisquare([classifiers results])
f_res = pd.DataFrame('test':'Friedman', 'statistic':f_test[0],'pvalue':f_test[1], index=[0])
wilc_test =[wilcoxon(models_df[i], models_df[j]) for i,j in itertools.combinations(models_df.columns)]
w_res = pd.DataFrame(wilc_test)
w_res['test'] = ['wilcoxon' + i +'vs'+ j for i,j in itertools.combinations(models_df.columns)]
```

2.7 Critical difference (CD) diagram

For constructing the CD diagram the code from https://github.com/hfawaz/cd-diagram was used which generates it based on the Wilcoxon method to detect pairwise significance. The input is a dataframe with three columns. The classifier used, the name of the data, and the AUC score. For comparing them the results from the seven classifiers needs to be as input. This graph was made for the three data transformations.

3 Results and discussion

3.1 Boxplots

For comparing and analyzing the results from the classifiers a boxplot was made for each data transformation. The figure 1 shows the results for the data without any transformation. The classifier that had the best performance was the Gaussian Mixture Model (GMM) which has the best average with 0.76. The Isolation Forest (ISOF) model had the worst performance with an average of 0.67. The Bagging Random miner (BRM) classifiers had similar behavior. The BRM with the best performance is the one with the default similarity measure with an AUC average of 0.72.

The datasets with MinMax scaling results are shown in figure 2. The classifier with the best performance was GMM, it maintained the same average AUC as the previous. The second best was the BRM with euclidean distance. The second average was the one class

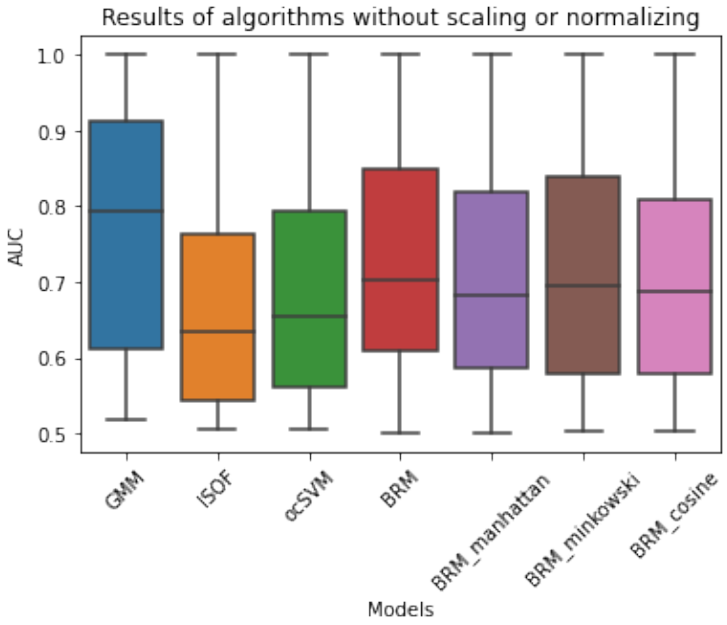

Fig. 1: Boxplot of raw data comparing classifiers AUC

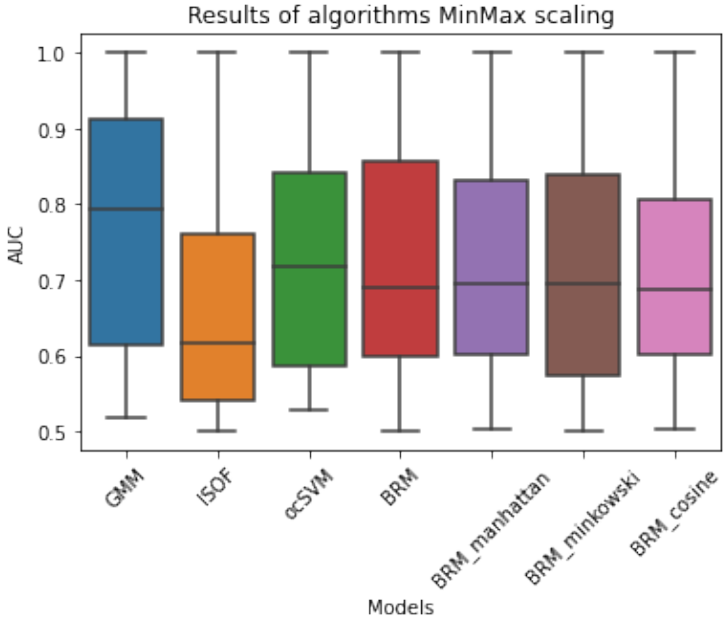

Fig. 2: Boxplot of data with MinMax transformation comparing classifiers AUC

Super Vector Machine (ocSVM), however the BRM with euclidean distance had a higher upper quartile. The worst average and lower quartile of the classifiers was the ISOF, it also maintained similar values.

The results from the data transformation with Standard normalizing are shown in figure 3. The best classifier was the GMM, the average reduced to 0.72, but the higher quartile increased in comparison to the previous cases to 0.9. The BRM had the second best average and upper quartile. The worst classiffier was the ISOF as in the previous cases. The ocSVM decreased the level of its upper quartile.

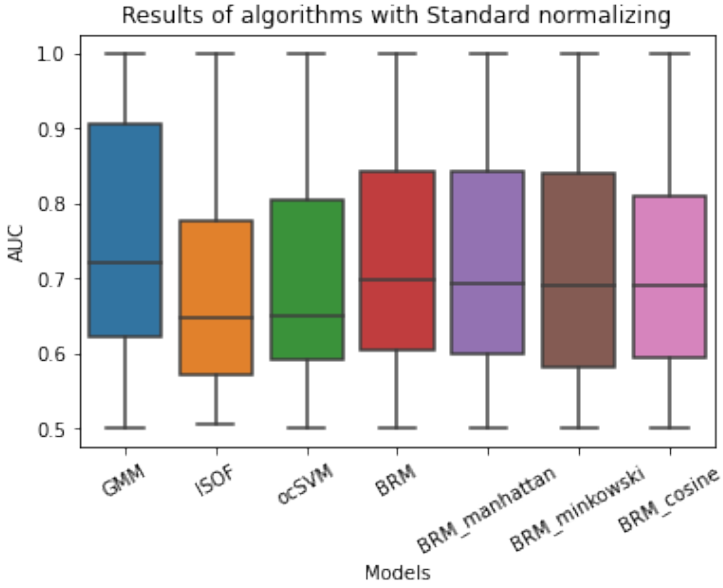

Fig. 3: Boxplot of data with using Standard normalizing transformation comparing classifiers AUC

3.2 Friedman statistical with Wilcoxon posthoc test

The results obtained were then evaluated with a Friedman statistical test in order to find if the obtained results are statistically different. Also, the Wilcoxon posthoc test with an alpha value of 0.05 was used to do a comparison among all the models and determine whether or not the results of each pair are statistically different. Tables 1 to 3 contain the statistical tests obtained for the raw dataset, Minmax scaling and Standard normalizing, respectively.

From table 1, GMM show an statistically difference with ISOF, ocSVM, and BRM with euclidean distance. ISOF and BRM with euclidean distance also describe an statistically difference between their results. Also, BRM_manhattan and BRM_cosine did show some difference between their results. The Wilcoxon posthoc tests among the others classifier did not show an statistically difference.

Table 1: Friedman statistical with Wilcoxon posthoc test over raw dataset

Test	Statistical	p-value
Friedman	23.213233	**0.000728**
wilcoxon GMM vs ISOF	361.000000	**0.000045**
wilcoxon GMM vs ocSVM	502.000000	**0.002363**
wilcoxon GMM vs BRM	510.500000	**0.007560**
wilcoxon GMM vs BRM_manhattan	711.000000	0.133157
wilcoxon GMM vs BRM_minkowski	715.000000	0.140934
wilcoxon GMM vs BRM_cosine	684.000000	0.089031
wilcoxon ISOF vs ocSVM	644.500000	0.102334
wilcoxon ISOF vs BRM	535.000000	**0.008247**
wilcoxon ISOF vs BRM_manhattan	681.000000	0.084958
wilcoxon ISOF vs BRM_minkowski	693.000000	0.102201
wilcoxon ISOF vs BRM_cosine	681.000000	0.084958
wilcoxon ocSVM vs BRM	735.000000	0.257552
wilcoxon ocSVM vs BRM_manhattan	841.000000	0.585920
wilcoxon ocSVM vs BRM_minkowski	836.000000	0.560859
wilcoxon ocSVM vs BRM_cosine	882.000000	0.808057
wilcoxon BRM vs BRM_manhattan	862.000000	0.696415
wilcoxon BRM vs BRM_minkowski	855.000000	0.658709
wilcoxon BRM vs BRM_cosine	810.000000	0.439541
wilcoxon BRM_manhattan vs BRM_minkowski	620.000000	0.865847
wilcoxon BRM_manhattan vs BRM_cosine	489.000000	**0.044946**
wilcoxon BRM_minkowski vs BRM_cosine	526.000000	0.062306

Table 2, describes that the following methods presented an statistically difference between their AUC results: GMM vs ISOF, GMM vs ocSVM, GMM vs BRM with euclidean distance, GMM vs BRM_minkowski, ISOF vs ocSVM, ISOF vs BRM with euclidean distance, ISOF vs BRM_minkowski and BRM_manhattan vs BRM_cosine. The comparisons between the other methods did not show the same behaviour.

Table 2: Friedman statistical with Wilcoxon posthoc test over Minmax scaling dataset

Test	Statistical	p-value
Friedman	29.549414	**0.000048**
wilcoxon GMM vs ISOF	296.000000	**0.000009**
wilcoxon GMM vs ocSVM	507.000000	**0.002669**
wilcoxon GMM vs BRM	439.500000	**0.002106**
wilcoxon GMM vs BRM_manhattan	734.000000	0.182712
wilcoxon GMM vs BRM_minkowski	427.500000	**0.001524**
wilcoxon GMM vs BRM_cosine	688.000000	0.094705
wilcoxon ISOF vs ocSVM	381.000000	**0.000239**
wilcoxon ISOF vs BRM	549.000000	**0.017642**
wilcoxon ISOF vs BRM_manhattan	679.000000	0.082327
wilcoxon ISOF vs BRM_minkowski	557.000000	**0.020828**
wilcoxon ISOF vs BRM_cosine	747.000000	0.216180
wilcoxon ocSVM vs BRM	869.000000	0.903875
wilcoxon ocSVM vs BRM_manhattan	853.000000	0.648088
wilcoxon ocSVM vs BRM_minkowski	857.000000	0.832619
wilcoxon ocSVM vs BRM_cosine	799.000000	0.393135
wilcoxon BRM vs BRM_manhattan	900.000000	0.912073
wilcoxon BRM vs BRM_minkowski	689.000000	0.814519
wilcoxon BRM vs BRM_cosine	853.000000	0.648088
wilcoxon BRM_manhattan vs BRM_minkowski	901.000000	0.917913
wilcoxon BRM_manhattan vs BRM_cosine	492.000000	**0.007868**
wilcoxon BRM_minkowski vs BRM_cosine	863.000000	0.701866

In table 3, the Wilcoxon posthoc test over Minmax scaling dataset shows that only GMM results when compared against those of ISOF and ocSVM and BRM_manhattan against BRM_cosine have an statistically difference. The comparisons among the other methods did not show the same behaviour.

Table 3: Friedman statistical with Wilcoxon posthoc test over Standard normalizing dataset

Test	Statistic	p-value
Friedman	14.690647	**0.022804**
wilcoxon GMM vs ISOF	483.000000	**0.003926**
wilcoxon GMM vs ocSVM	433.500000	**0.000393**
wilcoxon GMM vs BRM	612.000000	0.059395
wilcoxon GMM vs BRM_manhattan	761.000000	0.256926
wilcoxon GMM vs BRM_minkowski	746.000000	0.213459
wilcoxon GMM vs BRM_cosine	704.000000	0.120352
wilcoxon ISOF vs ocSVM	744.000000	0.287209
wilcoxon ISOF vs BRM	650.500000	0.076726
wilcoxon ISOF vs BRM_manhattan	733.000000	0.180307
wilcoxon ISOF vs BRM_minkowski	755.000000	0.238854
wilcoxon ISOF vs BRM_cosine	792.000000	0.365213
wilcoxon ocSVM vs BRM	721.000000	0.153248
wilcoxon ocSVM vs BRM_manhattan	790.000000	0.357468
wilcoxon ocSVM vs BRM_minkowski	816.000000	0.466125
wilcoxon ocSVM vs BRM_cosine	860.000000	0.685560
wilcoxon BRM vs BRM_manhattan	891.000000	0.859761
wilcoxon BRM vs BRM_minkowski	864.000000	0.707332
wilcoxon BRM vs BRM_cosine	815.000000	0.461633
wilcoxon BRM_manhattan vs BRM_minkowski	623.500000	0.154615
wilcoxon BRM_manhattan vs BRM_cosine	485.000000	**0.016946**
wilcoxon BRM_minkowski vs BRM_cosine	524.000000	0.059927

3.3 Critical difference diagram

The critical difference (CD) diagrams that were generated based on the results obtained from the statistical test are shown in figure 4, 5, and 6 for the dataset without transformation, with Minmax scaling and with Standard normalizing, respectively.

Figure 4 allows the visualization of the models performance. As it can be seen, GMM is not connected to ISOF, meaning that the results obtained from these two methods are statistically different. GMM has the better rank mean, followed by BRM with euclidean distance, BRM_minkowski and BRM_manhattan. On the other hand, the worst rank was obtained by ISOF, followed by BRM_cosine and oc-SVM.

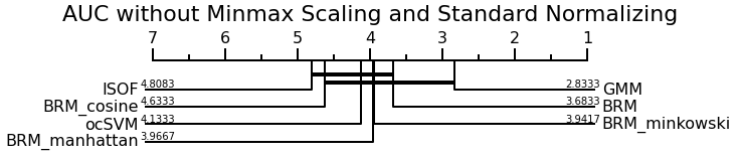

Fig. 4: CD diagram for dataset without transformation

The CD diagram shown in figure 5, refers to the models performance over the datasets with Minmax scaling pre-processing. In this case, three groupings can be observed. The first

group consists of GMM, BRM_manhattan, ocSVM, BRM_minkowski, BRM with euclidean distance and BRM_cosine. The second group is conformed by BRM_manhattan, ocSVM, BRM_minkowski, BRM with euclidean distance and BRM_cosine. Finally, the third group connects ISOF, BRM_cosine, BRM with euclidean distance, BRM_minkowski, ocSVM and BRM_manhattan.

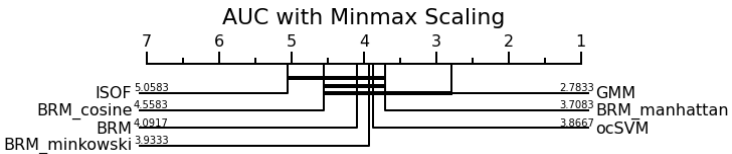

Fig. 5: CD diagram for dataset with Minmax scaling

The CD diagram in figure 6 shows the performance of the classifiers with a standard scaling. The first group includes the ISOF, BRM_cosine, ocSVM, BRM_minkowski, BRM, and BRM_manhattan, leaving the GMM. If a comparison is done between the rank mean values of figure 4, 5 and this case, it can be concluded that using standard normalizing shortens the rank mean range across the different classifiers. This means, that the results are less statistically different among all the models.

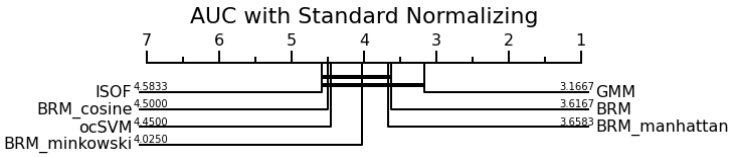

Fig. 6: CD diagram for dataset with Standard normalizing

4 Conclusions

The performance evaluation of unsupervised anomaly detection algorithms in 60 datasets through different statistical analyses and multiple data transformations, allowed us to assess and understand their behavior. The boxplot analysis showed that the GMM had the best performance with the three data transformations. The average of the AUC decreased with the Standard scaling transformation. The ISOF algorithm performed the worst. The BRM with euclidean distance was the second algorithm based on the AUC average and upper quartile. The BRM with Manhattan, Cosine, and Minkowski distance showed similar performance. The GMM showed a significant difference between the ISOF, ocSVM, BRM, and BRM with Manhattan distance using the raw data. In the three cases, the BRM algorithm with the different dissimilarity measures didn't show a significant difference. In the MinMax scaling the ISOF algorithm presented a significant difference between ocSVM, BRM, and BRM with minkowski distance. The critical difference diagrams showed that the GMM and ISOF algorithms have a significant difference with respect to the others. In general terms,

BRM showed a better performance than ISOF and ocSVM, however, GMM outperforms also BRM with every data pre-processing technique that was tested in this study.

5 Github publication

The Github repository of this project, in which the BRM modifications can be found in the following link (available from 4:00 pm on November 3):

https://github.com/Danielahmo/Evaluating-unsupervised-anomalydetection-algorithms.git

References

1. Camiña, J.B., Medina-Pérez, M.A., Monroy, R., Loyola-González, O., Villanueva, L.A.P., Gurrola, L.C.G.: Bagging-randomminer: a one-class classifier for file access-based masquerade detection. Machine Vision and Applications **30**(5), 959–974 (Jul 2019). https://doi.org/10.1007/s00138-018-0957-4