Comparison of five outlier detection methods in case of OpenClick data set

Vladimir Mitrović[1],Dragan Mišić[2], Miroslav Trajanović[3], Nikola Vitković[4]

[1,2,3,4] Faculty of Mechanical Engineering, Department of Production Information Technologies, University of Niš, Serbia

[1] vlada.bmm@gmail.com
[2] misicdr@gmail.com
[3] miroslav.trajanovic@gmail.com
[4] nikola.vitkovic@masfak.ni.ac.rs

***Abstract* — In data analysis, outlier detection is very important for tasks such as filtering data or finding points of special interest. Finding outliers can be quite challenging task, especially when data set represents human behavior or certain action, which can be hard to describe or predict. This is also case in OpenClick project, a platform which provides various tests where human behavior can be analyzed through human-computer interaction. For a single test type on OpenClick project, an individual test can be invalid for various reasons, e.g., a disturbance occurred during a test (such as ringing of a phone) or perhaps a test subject cheated in some way. One way how could we distinguish invalid test from valid is through detection of outliers in data set using outlier detection methods. For this purpose, we chose five outlier detection methods, which we use to detect suspicious data points and to evaluate tests in terms of how much outliers bring disturbance in test results. In addition, we give some observations on the form of the data set of a single test and propose possible way on how to approach a task of labeling tests as valid or not.**

I. INTRODUCTION

OpenClick project aims to discover physical status, skills and abilities of people based on human – computer interaction tests [1]. Simple tests are performed just by using mouse and/or keyboard and provide readily achievable way of conducting tests and research from various fields, such as the ones presented in [2] and [3]. In addition, tests are available not only to researchers, but also to any person who decides to do tests on their own. For every test there is an option to log test results, making OpenClick a database exhibiting continuous growth. There is total of nine test types implemented, and for one of these types, mouse click, we want to analyze data that has been recorded so far.

Mouse click is one of the test types available on OpenClick project. The goal of the test is to press left mouse button as fast as possible, over time period of ten seconds. Within one test, for each click, software is logging two different measurements. One is the "Elapsed time between two clicks", a time that passed from the previous click. The other measurement is "Ordinal number of click". For last measurement in test, ordinal number of click also represents a total number of clicks in that test. From these measurements, we introduce basic evaluation score for the test, its mean value of all elapsed times between two clicks.

Whether or not such a simple action such as mouse clicking is valid or not is actually a question already present in today's world. One of the most popular examples can be found in video gaming community. Many video games reward faster clicking, e.g., if mouse click produces a swing of a melee weapon, by faster clicking, a player could inflict more damage to virtual opponent. It is no surprise that over time many different clicking methods were introduced, such as jitter, butterfly or drag clicking [4], autoclick software for multiplying number of clicks are made [5], and a number of online tools is made available which can be used to test and practice speed of mouse clicking, such as on [4] and [6]. Game developers often find high speed of mouse clicking suspicious, and some may issue a restriction to gamers if they detect that autoclick software is being used [7]. Aside from gaming world, mouse clicking was also addressed in health research field. Example of this can be found in [8], where speed of mouse clicking is used as measure of anxiety.

In our work, we try to deal with a question if a test can be considered valid or not, based on the acquired measurements. There are several ways of how single test can be invalid. One of the ways is described above and considers an unusually high number of clicks. Other type of invalid test could be if a person got distracted during test (e.g., a phone ring occurred). So, low or inconsistent number of clicks could also be an indication of invalid test.

Looking at the scatter plots of these tests, such as the given example in Fig. 1, we often visually find some data points that are more or less dislocated from the majority of points. These points, which are commonly named outliers, represent our area of interest in this work. Namely, the question is how to detect them, how much impact can these outliers have on test results, and furthermore, is test valid with outliers being present.

II. METHODS

As said in introduction, for a single mouse click test, which was performed in unsupervised manner, we want to know if test is valid based on its outliers. Fig. 1 shows example scatter plot of measurements from single test. Looking at the example scatter plot, question arises, can single measurement be considered as an outlier just by its

relation to other measurements, or we need to take into account that data represents human behavior which can be hard to predict or explain. For example, if we take a look of a highlighted measurement in red dot, we could visually notice that it deviates from the rest of the data. But can we be sure that its value of nearly 400 ms really tells us that it wasn't likely for person to achieve it, even without any disturbance?

This is why our analysis is consisted of two segments. First segment is supposed to utilize different outlier detection methods to detect outliers as deviations from the rest of the data. Second segment is supposed to tell us how much detected outliers affect test result, for which purpose we introduce evaluation parameters, which description is to follow.

A. Outlier detection methods

As a first step in choosing methods, we wanted to set a basic understanding of a data set for single test. Data set has two features, with number of samples which can be rather low, having at least a few dozen measurements, and in most cases less than hundred in total. In other words, data set for single test is fairly simple. For these reasons, we decided for a start to go with well-known and proven industry standard methods. In addition, for each chosen method we wanted it to have different algorithm idea than others, and possibly to find justification in how applying particular method can be beneficial on our data set. Following these guidelines, and also looking at the works of others, such as [9] or [10], we chose following five outlier detection methods:

- Interquartile range (IQR). IQR is determined based on 25th and 75th percentile of data points, which is then used to set lower and upper whisker (threshold). IQR doesn't rely on easily compromised mean of test measurements, while it also considers each data point as an independent measurement. Since data set for single test is simple, IQR is chosen as a simple statistics method, which could also serve as a reference point for comparison of other methods.
- Linear regression (LR). Idea behind using regression as an outlier detection method is that a mathematical model (e.g., linear, polynomial or exponential) potentially can be used to describe data. Since all tests are performed by human beings, one would expect to see that a person is experiencing fatigue over time, resulting in greater elapsed time between two clicks. So, LR is used as the simple form of regression which tries to explain this potential behavior, and residual errors from fitted model are used as outlier scores. For this method, we also log information about slope of the fit line and R-squared value in order to see how well linear model can represent data.
- K nearest neighbors (kNN). We chose kNN as a distance-based method, trying to detect an outlier by how close it is located to other data points in its *k* neighborhood, having in mind that during a certain test period(s), a person could have exhibited a change in behavior, but the one(s) which does not necessarily mean it is caused by some unwanted disturbance. Such change could reflect in difference of elapsed time measurements, which do not have to be outliers. Outlier scores are given as an average distance of a data points from its *k* neighbors.
- Local outlier factor (LOF). Along with the idea behind choosing kNN as a method for outlier detection, LOF, as a density-based method, adds that the LOF score of each data point is affiliated with scores of its *k* neighbors (through local reachability density values), implying that data might be even more dependent than the other methods are taking into consideration.
- Isolation forest (IF). Usage of randomly generated isolation trees and averaging of obtained outlier scores could provide more confidence in isolating data points which are stronger outliers, at least in terms of deviations from the rest of the data set.

For implementation of these methods, we use Python programming language with included libraries for machine learning, math and statistics. As explained, all these methods (except IQR) as an output assign each data point an outlier score. From that, there is an option to use contamination parameter to label each data point as outlier or inlier. Contamination parameter represents a percentage of data points to be labeled as outliers based on ranking by outlier score. Since we do not know in advance how many points in each data set are outliers, we decided to go with another approach. As suggested and implemented in [9], we also decided to use IQR on outlier scores, in order to set threshold values for outliers labeling. In this way, not every data set is going to have the same ratio of outliers and inliers. In fact, it is possible that certain test has no detected outliers at all, which is also expected to happen.

B. Evaluation parameters

Before describing evaluation parameters, we introduce additional parameters we needed to keep track of in order to calculate evaluation parameters:

- Outlier Value – Elapsed time between two clicks for detected outlier, given in ms.
- Outliers Count – Total number of detected outliers.
- MAX Outlier Value – Maximum elapsed time between two clicks of all detected outliers.
- Test Median – Median value of entire data set for single test.
- Outlier Score – A ratio of Outlier Value and Test Median (LR method uses predicted value of fitted model). Not to be mixed with score each method is assigning.
- Mean Original – Mean value of entire data set for single test.
- Mean Trimmed – Mean value of data set for single test calculated with removed detected outliers.

Now we can list evaluation parameters:

- MAX Outlier Score – This parameter shows highest deviation from test median of all detected outliers.
- MEAN Outlier Score - An arithmetic mean of all Outlier Scores.
- Test Mean Change – A relative change of test mean after removal of outliers.

$$Test\ Mean\ Change\ = \frac{|Mean\ Original - Mean\ Trimmed|}{Mean\ Original} \quad (1)$$

- Test Time Trimmed – An amount of time that outliers take in test duration. Here we assume that if outlier did not occur, measurement value would be test median.

$$Test\ Time\ Trimmed = \quad (2)$$
$$= \sum_{i=1}^{Outliers\ Count} i_{th}\ Outlier\ Value - Outliers\ Count * Test\ Median$$

As mentioned before, for LR method we also log slope of fitted linear model and R-squared value as a measure of fit quality. Similarly, as with main evaluation parameters, we want to see how much outliers affect these values, so we calculate following:

- Slope (Original) - Slope of linear model calculated from entire data set for single test, given in degrees.
- Slope (Trimmed) - Slope of linear model calculated from data set with removed detected outliers, given in degrees.
- R2 (Original) - R-squared of linear model calculated from entire data set for single test, given in %.
- R2 (Trimmed) - R-squared of linear model calculated from data set with removed detected outliers, given in %.

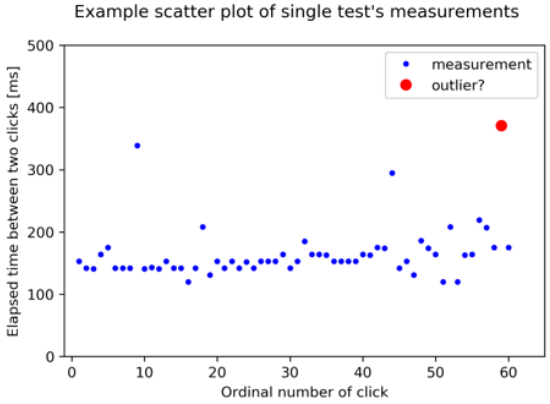

Figure 1. Scatter plot of example of a single test

III. EXPERIMENTAL SETUP

After establishing outlier detection methods and evaluation parameters, we apply all methods on each of 1068 tests we extracted from OpenClick database. According to several unofficial information we found, such as the ones in [4] and [6], world record for fastest mouse clicking goes up to 16 cps (clicks per second), which is little above 60 ms in time between two clicks. In our eye inspection of tests, we also found that regular fast clicks could be close to this value. So, in our work, we decided not to include tests with measurements smaller than 60 ms in our analysis. Also, if any method reported outlier which value is bellow mean test value, we did not consider that data point as an outlier. Therefore, our analysis is mostly oriented towards tests in which is more likely that some disturbance occurred, rather than someone was cheating and boosting its click speed.

Fig. 2 shows results of outlier detection methods performed on example scatter plot from Fig. 1. Before moving to more detailed analysis of results obtained by using these methods, we wanted to see how efficient are chosen method in terms of segregating outliers, considered as deviations from the rest of the data.

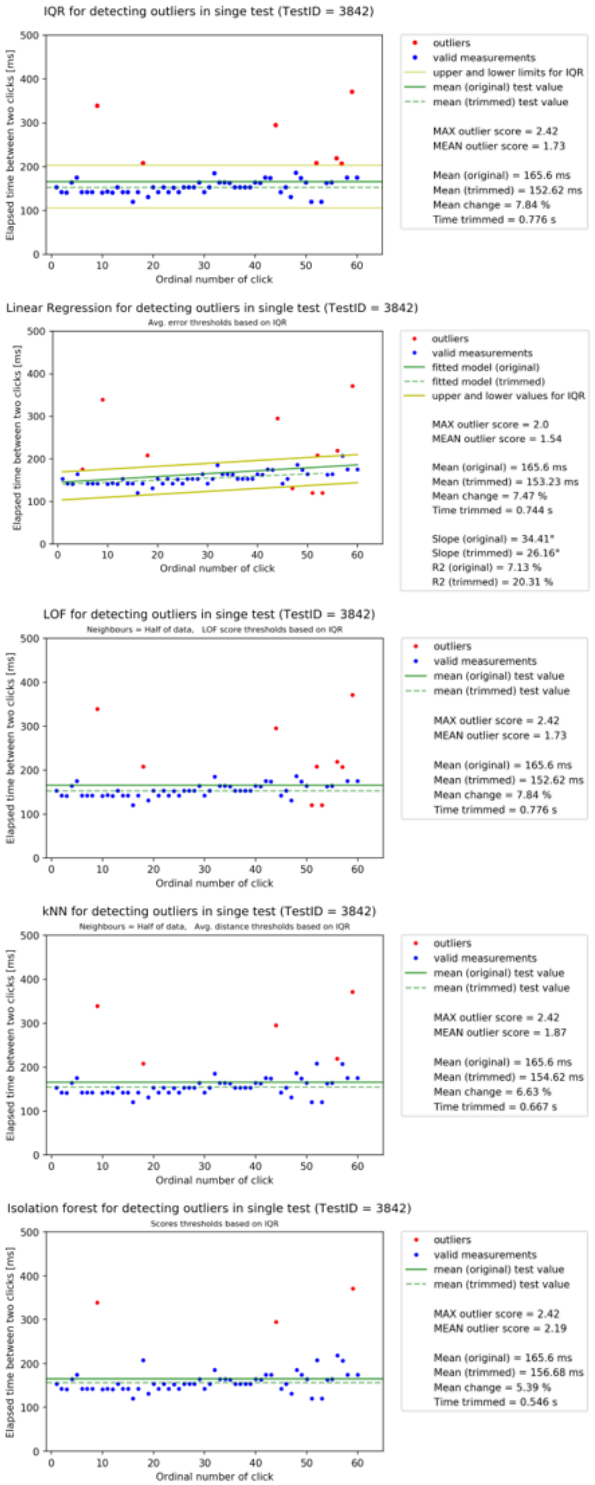

Figure 2. Outlier detection methods applied on example of a single test

Since data set for single test is unlabeled, we are not sure if outliers are correctly detected. Consequentially, outlier detection actually divides data into outliers and valid measurements, so we decided to perform classification of these two obtained classes. In general, detected outliers are in great minority, making two classes very imbalanced. To deal with this, we implemented a combination of data oversampling and undersampling in order to get balanced

classes. For undersampling of majority class (valid measurements), we used simple random undersampling. For oversampling of minority class (outliers), we used SMOTE (Synthetic Minority Oversampling TEchnique) algorithm, as it creates additional class points on lines between original points [11], rather than just duplicating existing class points. After this process, we get balanced data set where half of data are valid measurements, and the other half are outliers. This is shown on our example scatter plot in Fig. 3.

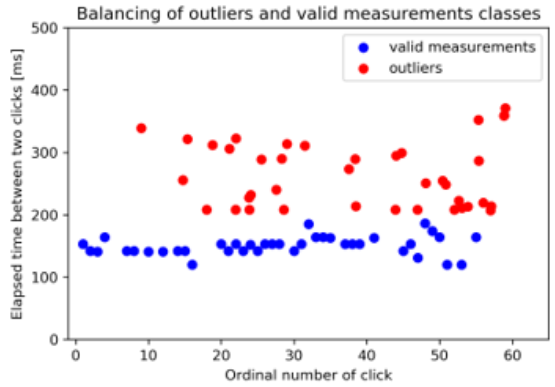

Figure 3. Balancing of outliers and valid measurements classes

After balancing, stratified five-fold cross validation with decision tree as classifier is performed, and mean efficiency parameters (precision, recall and F1 score) are calculated for all 1068 tests. Results are given in Table 1. Efficiency parameters show that all methods have very high and similar efficiency in detecting outliers, as long as they are considered as deviations from the rest of the data set.

Table 1. Efficiency of outlier detection methods

	Efficiency parameter		
Method	**Precision (mean)**	**Recall (mean)**	**F1 score (mean)**
IQR	0.9971	0.99909	0.99794
LR	0.99278	0.99505	0.99342
LOF	0.99316	0.99291	0.9925
kNN	0.99292	0.99268	0.99223
IF	0.9865	0.98681	0.98573

IV. Results and Discussion

Looking at the plots in Fig 2, we can see that all methods were able to detect certain number of outliers. What comes to attention is that IF detected the least number of outliers, and not only that, but detected outliers are the strongest ones, in terms of outlier scores. We can confirm this in general, if we take a look at metrics we devised from evaluation parameters of all tests, given in Table 2. We can see that IF detects on average the least number of outliers per test, compared to other methods. With lower number of detected outliers comes that "Test Mean Change" parameter is also lower, while "Mean Outlier Score" is higher, as only stronger outliers were detected. From this we can conclude that IF is the least restrictive method, more prone to detect only stronger outliers and possibly able to filter out lesser inconsistencies in measurements, which could be attributed to person not being able to be ideally constant during test.

From Table 2, we can see that LOF and kNN methods have the highest average of detected outliers per test. Reason for this may lie in fact that these methods are dependent on number of neighbors *k*, which can be challenging to optimize for every test, since we don't know in advance how many outliers to expect. In our trials, we experimented with different values for *k*. For the moment, we chose *k* to be half of the data, expecting that at least that number of points would be valid measurements. LOF method is more prone to detect higher number of outliers, probably due to nature of its algorithm (comparing local reachability distances between point and its neighbors), and the fact that outliers don't have to be grouped together (e.g., one outlier can occur at the beginning of the test, while some other can occur at the end of the test). On the other hand, kNN method on average detects less outliers than LOF. This can also be seen on plots in Fig. 3. Finding more optimal value for *k* could be a task we handle in future work.

Table 2. Performance of outlier detection methods

	Outlier detection method				
	IQR	**LR**	**LOF**	**kNN**	**IF**
Tests with no detected outliers	114	96	72	63	76
Average of detected outliers per test (%)	3.756	3.432	4.325	4.090	2.475
Tests where "Test Mean Change" >= 5 %	345	348	374	367	257
Tests where "MAX Outlier Score" >= 2	457	379	457	459	450
Tests where "MEAN Outlier Score" >= 2	254	180	196	205	320

In general, all methods are able to detect strongest outliers. But, looking at the Table 2, it is evident that LR method has lower "MAX Outlier Scores" and "MEAN Outlier Scores" than other methods. As mentioned before, this is because LR scores outliers differently, it uses predicted value of fitted model instead of test median. Interesting side of using LR method for detecting outliers is that we can take into account possibility that person doing test is not constant. To a certain extent, it is expectable for person to experience physical (hand tiresome) or mental (loss of concentration) fatigue. Also, it is not impossible to see someone boosting its performance as the test goes on. That is why a linear model, as a simple form of regression, could also provide such information. As said in section II, for LR method we keep track of slope of fitted model and R-squared value. Negative side of using LR is that fitted model can be compromised from the presence of outliers. For that reason, we also calculate these parameters after removal of outliers. In Table 3. we give basic statistics for these parameters.

Due to tendency of R-squared value to approach zero value as fitted linear model approaches horizontal line, we

divided slope values into three groups, negative slope, positive slope and near horizontal slope. It shows that most of the people have positive slope, indicating that a person is getting slower, possibly due to fatigue. Also, large number of people have a near horizontal slope, indicating a more constant performance. And finally, very small number of people have negative slope value, indicating that they are getting faster during test. Evaluation of fit quality with R-squared value indicates that 6 % of the tests have R-squared value greater than 50 %, a limit which could indicate a moderate fit, according to some authors, e.g., in [12]. As expected, in near horizontal slope region, mean R-squared value is very low. It tends to have greater value as slope is becoming steeper, going up to a maximum of 81.29 %.

Table 3. Slope and R-squared value of fitted model obtained with LR

	Slope after removing outliers (α)					
	Negative slope		**Near horizontal slope**		**Positive slope**	
	$\alpha \leq -30°$	**$-30° < \alpha \leq -15°$**	**$-15° < \alpha \leq 0°$**	**$0° < \alpha \leq 15°$**	**$15° < \alpha \leq 30°$**	**$30° < \alpha$**
Tests count	29	19	76	250	363	237
R2 Mean (%)	22.95	7.21	1.45	5.82	21.56	37.8
R2 Max (%)	81.29	18.44	18.92	30.61	60.97	77.24
R2 Min (%)	0.6	0.71	0.0	0.0	0.1	1.06
Tests with R2 ≥ 50%	64 (6% of all tests)					

Table 4. Correlation coefficient of evaluation parameters

	MEAN Outlier Score		**MAX Outlier Score**		**Test Time Trimmed**	
	Pearson	**Spearman**	**Pearson**	**Spearman**	**Pearson**	**Spearman**
Test Mean Change	0.6276	0.7340	0.8108	0.8592	0.9987	0.9988
Test Time Trimmed	0.6366	0.7451	0.8181	0.8672		
MAX Outlier Score	0.8433	0.8972				

At the end of this section, in Table 4, we give correlation coefficients (Pearson and Spearman) between evaluation parameters. Table is given for IQR method, which is according to Table 3 in the middle in terms of sensitivity of outlier detection. Similar correlation coefficients are obtained for other methods, with LOF and k-NN methods being close to IQR, IF having slightly higher, and LR having lower correlation values. Table 4 show that "Test Time Trimmed" and "Test Mean Change" parameters are almost ideally correlated, for all methods, and therefore one of them could be dropped out from future work, focusing on the remaining three.

V. Conclusions and Future Work

From OpenClick data base, we extracted 1068 tests, and all five outlier detection methods were run on each test. On selected example in Fig. 2, we show side by side performance of each method. In Table 2, we give performance of each method in terms of introduced evaluation parameters. We conclude that all methods can effectively detect outliers (considered as deviations from the rest of the measurements), as it is shown in Table 1. IF method is distinguishing itself from the others in a way that it is the least restrictive method. This “filter” characteristic of IF could be used as advantage to isolate only stronger outliers, which have the most impact on test results.

LOF and kNN methods can be sensitive to chosen number of neighbors, and it could be challenging to optimize *k* value from test to test. These two methods tend to report more outliers, especially LOF which, as shown in Fig. 2, can report as outlier measurements that are on closer eye inspection more obvious to be valid. In other words, reliability of these methods in this particular case is something to be cautious about.

Of all methods, LR stand out as it scores outliers differently and offers some insights of trend in human behavior over duration of test, given in Table 3. As it is expected, most of the people who performed test exhibit deterioration in click times, possibly because of some form of fatigue. Furthermore, R-squared values show that fitted linear model could explain this human behavior to a certain extent. Downside of LR is that fitted model used for outlier detection is calculated from all data points, including outliers. In future work, if there is a need to use LR to describe single test, what could be done is to remove outliers using some other method, e.g., IF, and then analyze data with LR.

When it comes to making a decision if a single test is valid based on its outliers and calculated evaluation parameters, we have to have in mind that all of these tests were performed in unsupervised manner, meaning that we do not know physical and mental state of the person, nor we know anything about surrounding conditions in which the test was done. Choosing some threshold value for any of the evaluation parameters might be harsh and based only on individual sense for what does this threshold value has to be, in order for test to be invalid. For example, looking at the Table 2, if we set that threshold for “MAX Outlier Score” is two, it would mean that almost half of all tests would be invalid, which is not very likely the case. What could be done in future work is to perform tests in controlled environment with and without simulated disturbance (e.g., sound or some other physical, impulse or continuous distractions). Then we could label tests and perform machine learning technique, which might even help us to detect invalid test as they are being done online on OpenClick platform. For the end of this work, we also take a note that for now we only started with analysis of mouse click test, and that there are eight more tests which could possibly provide more ground for comprehensive research.

REFERENCES

[1] D. Misić and M. Trajanović, Faculty of Mechanical Engineering Nis. Retrieved December 1, 2020. http://openclick.masfak.ni.ac.rs

[2] J. Arandjelović, P. Drasković, R. Turudija, M. Dimitrov, N. Bozić, N. Korunović, D. Misić and M. Trajanović, (2018a), "Trial experimental determination of the average times of actions executed in a CAD application", In 37th International Conference on Production Engineering, Serbia, 2018.

[3] J. Arandjelović, P. Drasković, R. Turudija, M. Dimitrov, N. Bozić, N. Korunović, and M. Trajanović, (2018b), "Towards a Methodology for CAD program Efficiency Assessment", Proceedings of 4th International Conference Mechanical Engineering in XXI Century, Serbia, 2018, pp. 155-161.

[4] Retrieved January 20, 2021. https://www.clickspeedtester.com/

[5] Retrieved January 20, 2021. https://en.wikipedia.org/wiki/Auto_clicker

[6] Retrieved January 20, 2021. https://clickspeedtest.com/

[7] Retrieved January 20, 2021. https://hypixel.net/threads/guide-allowed-modifications.345453/

[8] M. Macaulay, "The speed of mouse-click as a measure of anxiety during human-computer interaction," *Behaviour & Information Technology*, 23.6, pp. 427-433, 2004.

[9] F. Falcão, et al., "Quantitative comparison of unsupervised anomaly detection algorithms for intrusion detection," Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing. 2019.

[10] G. O. Campos et al., "On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study," *Data mining and knowledge discovery* 30.4, pp. 891-927, 2016.

[11] N. V. Chawla, K. W. Bowyer, L. O. Hall and W. P. Kegelmeyer, "SMOTE: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, 16, pp. 321-357, 2002.

[12] D. S. Moore, W. I. Notz, M. Fligner, *The Basic Practice of Statistics*, 8th ed., New York-United States, United States, Macmillan Publishers, 2017.