Unsupervised Anomaly Detection for Hard Drives

Enrico Barelli[1], Ennio Ottaviani[2]

[1,2] *On A.I.R. srl, Genova, 16129, Italy*
enrico.barelli@onairweb.com
ennio.ottaviani@onairweb.com

ABSTRACT

In the age of industry 4.0 and smart sensors, continuous monitoring of different machinery produces an enormous amount of data. Because of that, data centers are now-a-days a very important asset, not only for large-scale cloud providers, but also for medium to large enterprises which decide to store in-house the ever-increasing data collected during business operations.

An efficient method for the maintenance of the great number of hard drives housed in data centers is critical to assure availability in a cost effective manner. Since 2013, Backblaze `https://www.backblaze.com/` has published statistics and datasets for researchers to gain insights on hard drive performances and their failures, in this paper more than 2.5 million records, following hard drives S.M.A.R.T. readings for over a year, will be analysed.

The aim and main contribution of the paper is to provide a real-world example of how to build and evaluate both an online and an offline pipeline which can effectively monitor and give precursor signals of hard-drive failures. The paper is organized as follows: after a brief introduction on the problem the dataset and the experimental setting will be described. Both the offline and the online pipelines will then be discussed along with empirical results comparing them to a supervised baseline.

1. INTRODUCTION

Data is being collected at an ever-increasing rate by firms of every size, as the opportunity cost relation is becoming more and more clear. Equipping machines with an array of useful sensors has become relatively cheap, data collection and remotization are also becoming easier thanks to more accessible technology in the form of ready-to-use cloud-based solutions. Whether the data is stored in an in-house data center or in the cloud, hard drives are the base unit of storage. Effective maintenance of a large number of hard drives can as such decrease costs and reduce unexpected downtime or complications.

As reported by Vishwanath & Nagappan (2010), hardware failure can lead to a degradation in performance to end-users due to service unavailability and can result in losses to the business, both in immediate revenue as well as long-term reputation. The authors also report that 70% of all server failures is due to hard disks, 6% due to RAID controllers, 5% due to memory and the rest (18%) due to other factors. Thus, hard disks are not only the most replaced component, they are also the most dominant reason behind server failure. While the percentages may have changed from 2010, the problem is still present and requires attention. The already mentioned Backblaze provider reports a snapshot for 2019 which still reports an annualized failure rate of 1.89%, which means that a design and strategy that handle failures gracefully is still needed. Large industries in the tech sector are still committed to research in this area. A 2016 study from Botezatu, Bogojeska, Giurgiu and Wiesmann, who at the time were affiliated with IBM, proposes a machine learning-based pipeline for predicting disk optimal replacements time, about 10 to 15 days in advance. Moreover, the more reliable SSD technology is still at a price range which makes it not viable for data centers. An official 2018 blog post by Roderick Bauer (Backblaze) highlights that HDD will be the storage medium for large-scale data centers for the foreseeable future.

Datasets extracted from Backblaze data have already been used for research purposes, such as Nicolas Aussel, Samuel Jaulin, Guillaume Gandon, Yohan Petetin, Eriza Fazli & Sophie Chabridon (2017), but there is, to our knowledge, no standard benchmark on their data. The main contribution of this paper is to give a clear, real-world use-case study which combines and compare different approaches to establish the practical feasibility of a simple unsupervised process. In the next section, the dataset used in this study will be described in detail along with the experimental setting for the statistical analysis.

The objective of this paper is to show that it is possible to

Enrico Barelli et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 United States License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

build a completely unsupervised pipeline that produces an anomaly score that highly correlates to hard drives time to failure (TTF), in such a way that a decision to replace them can be made before failure with minimal waste due to false alarms. Favourable comparisons with a state of the art supervised classifier will be presented in the next sections.

A brief example of how such a pipeline can be extended for data streams and continuous sensor monitoring will be given.

2. Dataset and experimental setting

In this paper, more than 5000 hard drives will be followed through time. More precisely, a set of machines from a single model has been chosen for analysis, the Hitachi HDS5C30-30ALA630, followed in a period spanning from May 2013 to December 2014. The dataset constructed in this way contains 2 625 514 records with 53 failures, these failures have no clear definition rule but are instead reported a-posteriori by Backblaze as the day before the failure of the equipment reported by users.

The features used in the analysis are readings from Self Monitoring Analysis and Reporting Technology $(S.M.A.R.T.)$ recorded from the sensors embedded in the devices both in raw and normalized form and are recorded at a rate of one record per hard drive per day. Smart-194, smart-5, smart-1-raw, smart-197, smart-9 have been chosen as they are temperatures, uptime and error counts that are collected by almost all modern devices by different vendors. As there is no standard on which S.M.A.R.T. readings are recorded, the analysis on the chosen ones should be the most easily generalized on other models. Additional details about the chosen features can be found in table 1. S.M.A.R.T. features have thoroughly studied in the literature for example in the work by Slađana Đurašević and Borislav Đorđević (2017) which provides many statistical details about the chosen features and some other S.M.A.R.T. indicators. The features are given by Backblaze both in raw and normalized form; this work uses them in the raw form as the chosen algorithm do not use distances and thus their performance is not impaired by the difference in the scale of the feature's values.

Some records have been kept as the test set, which has been used just for performance evaluation. This evaluation set comprises records starting from 01-09-2014 to the last day available on the dataset. This choice, while somewhat arbitrary, ensures not to mix past and future in the training set and keeps 11 failures in the test set. The experimental setting also mimics a possible real-life scenario, where a solution is studied using some past data and then deployed for a period before being evaluated again.

The test set contains 555 707 records, the train/test split follow roughly a 80%-20% split strategy. The total failures are 53, in this way they are distributed evenly in both sets, with the same 80%-20% split between training and testing.

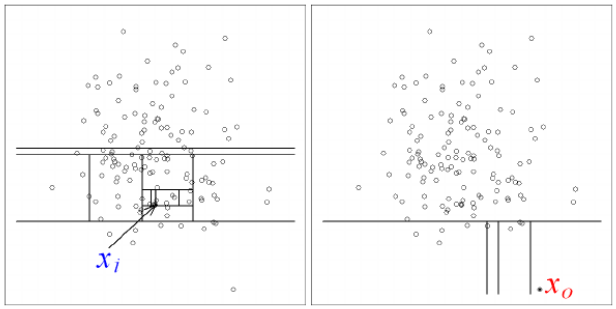
3. Unsupervised offline pipeline

The objective of this work is to present an unsupervised process that can assign a meaningful anomaly score to each data point. Such a score should correlate well with time to failure so that it can be used to make informed decisions about maintenance and/or replacements ahead of failures. The data analysed here contains some failures examples, but in other scenarios failures may not have been observed or may be both rare and of different nature, this makes it hard, if not impossible, to build a supervised system. Even when failure examples are present, it will be shown that a state of the art classifier does not necessarily produce better results than an unsupervised pipeline. Our example uses only 5 features for ease of extensibility to other hard drive manufacturers. Nonetheless, the chosen method is able, and is even more suitable, compared to traditional distance-based methods, to problems with a large number of variables. This is because distances become less meaningful in high dimensional spaces, where every point tends to be far away from the others. On the contrary the chosen method, which will be described in detail in the next section, does not use a distance metric but an isolation approach. This characteristic would enable the method to generalize well to whichever set of attributes is chosen; it must be noted that the largest superset of S.M.A.R.T. readings available considering all vendors and models could cover over 70 attributes (even if they could be very sparsely populated). Such a high number of variables could pose a significant problem because as data become sparser the true outliers can be masked by noise effects of multiple irrelevant variables. Moreover, the different kinds of meaning of the variables (counts, real numbers, boolean, possibly even categories) would require a standardization procedure and one-hot encoding, complicating the process. Defining the locality of a point in such a setting becomes a problem and algorithms that use regular metrics and distances can fail when working with all the dimensions. For example all pairs of points are almost equidistant (in the Euclidean sense) in a high dimensional space (Aggarwal, Hinneburg & Keim, 2001).

In order to handle both high dimensional cases and low dimensional ones the chosen unsupervised method belongs to a general class of algorithms called *subspace methods* as they find anomalies in the data by exploring subspaces of the volume occupied by them in the feature space. The subspace approaches thus implicitly limit the problem of having a high number of variables by effectively using few of them at a time.
The chosen anomaly detector is Isolation forest, proposed by Liu, Ting & Zhou in 2008.

Table 1. S.M.A.R.T features details

Feature code	Full name	Description
SMART 1	Read Error Rate	Indicates the rate of hardware read errors from disk surface.
SMART 5	Reallocated Sectors Count	Indicates the number of bad sectors that have been remapped to spare area.
SMART 9	Power-On Hours	Shows total count of hours in power-on state.
SMART 194	Temperature	Indicates the device temperature expressed in Celsius degrees.
SMART 197	Current Pending Sector Count	Indicates the number of unstable sectors which are waiting to be remapped.



(a) Isolating x_i (b) Isolating x_o

Figure 1. Number of partitions needed to isolate two different point. Image taken from the original paper by Liu, Ting & Zhou in 2008

3.1. Isolation Forest description

Isolation forests are an ensemble method consisting of a set of isolation trees, which are trees defined and constructed to separate anomalous data points from normal ones. The basic assumption is that in a random partitioning algorithm anomalies should be separated (isolated) from the others data points in its earliest steps. This should happen intuitively because when partitioning extreme values should be isolated earlier with respect to subtle differences. In figure 1 a visual example of this situation is given.

This method has been chosen as the mechanism under which it operates is very intuitive and explainable to a domain expert which has no prior experience of data mining techniques, moreover it has no explicit assumption about a mathematical model (i.e. a parametric model) about the data distributions which is often hard to estimate in practice. As a last remark, isolation forests use subsampling and do not compute distances. These are highly desirable characteristics when working with a large volume of data as it makes the algorithm easily scalable as it has a low memory requirement and linear time complexity. A sketch of the algorithm used to build an isolation tree is in the following:

- randomly select a partition of the dataset e.g. with bagging
- randomly select a variable q and a split value p and split binarily the dataset according to this rule
- repeat the last step until each node has only one instance or all data in the node have the same values for all variables

Note that an isolation tree (iTree) is a proper binary tree, where each node has exactly zero or two child nodes.
Assuming distinct instances each one is isolated in an external node when the tree is fully grown. The path length from the root to the isolation in an external node can be measured and is then used as an indication of the susceptibility to the isolation of the point.

Results from an ensemble of trees can be averaged and normalized, and an anomaly score computed. More precisely the path length of a point x can be defined as the number of edges the point needs to traverse in the isolation tree from the root to the external node it becomes isolated into. With a short path length the point is very easily isolated and thus is considered anomalous. A problem arises when translating this idea to a numeric score as the isolation tree maximum height grows in the order of n while the average height grows as $\log(n)$ The authors propose a normalization schema to give bounds to the average path lengths by exploiting the isolation trees' equivalent structure to the binary search trees' (BST) one. The full expression for the anomaly score, as defined in the original paper, exploits the structure of a BST and in particular the average path of an unsuccessful search which is:

$$c(n) = 2H(n-1) - (2(n-1)n)$$

where $H(i)$ is the harmonic number and it can be estimated by $\ln(i) + \gamma$, γ is the Euler-Mascheroni constant. Now, let $h(x)$ be the number of edges a point x traverse an iTree from the root node until the traversal is terminated. Since $c(x)$ is the average of $h(x)$ given n it can be used to normalize it. So the anomaly score s of an istance x can be defined as:

$$s(x,n) = 2^{-\frac{E(h(x))}{c(n)}}$$

where $E(h(x))$ is the average of $h(x)$ on the ensemble of iTrees in the forest. This normalization schema bounds effectively the score in the interval $[0,1]$, with 0.5 as a midpoint, 1 as the most anomalous score and points with an s much smaller than 0.5 as considered normal. In this work, the opposite is used, with 0 as midpoint and -1 as the most anomalous, and number greater than 0 considered as normal. This ensure the separating point between anomalies and normal points is 0 which has been found more natural. The BST structure can also be used to devise strategies based on the

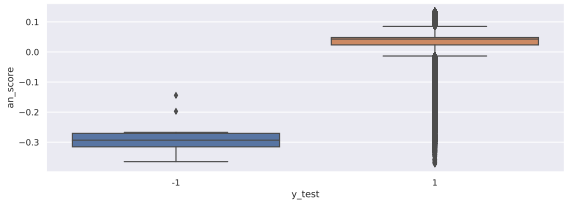


Figure 2. Anomaly scores vs failures

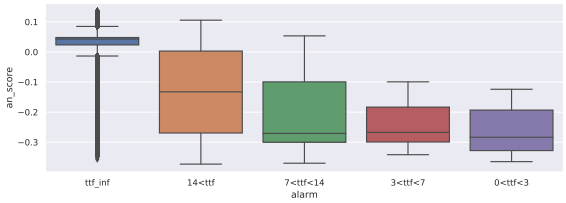


Figure 3. Anomaly scores vs time to failure classes

Figure 4. Anomaly scores vs time, in red the score median.

average height of a BST for pruning or to stop the growing process earlier.

Intuitively this method explores random subspaces of the data volume without assuming independence of the features, but sub-sampling them at random jointly. The authors note that, while anomaly detectors based on data density, such as Local Outlier Factor (Breunig, Kriegel, Ng & Sander, 2000), assume that normal points occur in dense regions to the contrary of anomalies that occur in sparse regions, and methods based on distances, such as One-Class Support Vector Machines (Scholkopf, Platt, Shawne-Taylor, Smola & Williamson, 1999), assume that a normal point is near its neighbours in some metric space, the proposed method is more adaptive as, after each split and in each tree, the subset of the data analysed is completely different and thus can detect both global and local outliers.

3.2. Empirical evaluation results

The objective of the first experiment is to explore the links between the anomaly score outputted by the Isolation forest and the time to failure of hard drives. All the following results are on the test set described above.

In order to analyse the behaviour of the anomaly score outputted by the algorithm the Isolation Forest has been trained with the default parameters indicated on the paper. This would reflect a first approach to the problem, moreover assuming not to have any example of failures the analyst would not have any mean of optimizing the choice. In such cases, the usual procedure is to use a range of parameters and then to average the resulting anomaly scores. This averaging ensures robustness to possible wrong choices of the parameter, but as it will be shown later this is not necessary in this case. This evidence supports the claim by the original authors that the default parameters should provide satisfactory results in most cases.

The first observation is about raw anomaly scores concerning the two classes: failure (-1) and non-failure (1). The anomaly score is unbounded and in a scale such that the lower the score the more anomalous the point is. The results in figure 2 boxplot show the score can highlight differences in the two classes.

The failures in the dataset are tagged as the day before the breakdown and it has been shown that on that date the anomaly score is lower. It is then clear that there is some signal that something is wrong immediately before the failure, but having more time to act would improve the reliability of the data center, giving more time for backups and/or substitutions.

Using the date information new classes are created to inspect this problem. Hard drives that do not fail in the period of observation are given the class ‘ttf-inf’ to mean that their time to failure is infinite, rows that are more than 14 days away from failure are given the class ‘14<ttf’, ‘7<ttf<14’ stands for rows that are less than 14 days from failure but more than 7 and so on.

In figure 3 it can be noted from the boxplots that there are indeed some differences in the anomaly scores for the different classes. While there is overlap the score is decreasing w.r.t. the classes in a clear way. A multi-class thresholding rule could in theory be established to try and classify each hard drive in a ‘risk’ class. Another strategy is to follow each hard drive in time with a pre-trained model, without special updates or time series processing.

This naive approach again highlights, as can be seen in figure 4, the feasibility of such a simple strategy. There are clearly identifiable change points in the anomaly scores’ time series that can be automatically detected and used to give alarms. An old but effective algorithm for change detection is CUSUM (CUmulative SUM control chart) first proposed by Page (1954). An example output of a CUSUM approach to the same hard drive, searching for changes only in the downward direction, can be seen in figure 5.

3.3. Comparison with a supervised method

The previous sections focused on establishing a link between TTF and the anomaly score, but did not provide quantitative

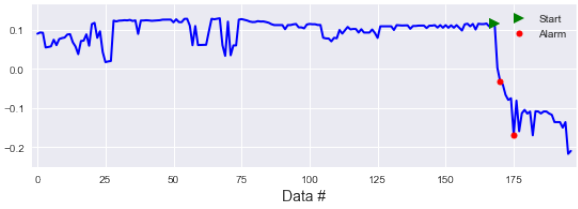


Figure 5. Anomaly score change detection

Table 2. XGBoost confusion matrix

actual/predicted	fail	ok
fail	9	2
ok	3770	551937
F1	0.0047	
Recall	0.8182	
Precision	0.0024	
False Alarm Rate	0.0067	

results. In this section, since class labels are available, a supervised algorithm (a classifier) will be used and compared with a simple thresholding strategy on the anomaly score. The chosen classifier is XGBoost, first proposed by Chen Tianqi and Guestrin Carlos in 2016, a state-of-the-art classifier which is currently widely used in industry and to win Kaggle competitions. XGBoost (eXtreme Gradient Boosting) is a very versatile algorithm that can optimize any twice differentiable objective function via gradient descent in a tree boosting setting. The idea behind functional gradient descent is quite old with the first very successful algorithm being ADAboost (Freund & Schapire, 1997). The dataset is extremely imbalanced for a classifier, accuracy is then not the right metric to optimize and to check. The chosen metric is then the F1 score. The F1 score is defined as follows:

$$F1 = \frac{2 \cdot precision \cdot recall}{precision + recall} \tag{1}$$

where

$$precision = \frac{TP}{TP + FP} \tag{2}$$

$$recall = \frac{TP}{TP + FN} \tag{3}$$

Both algorithms used the training set to choose their hyperparameters via a 3-fold cross-validation. In this case, there are two more choices to optimize in the same cross-validation, namely the threshold on the anomaly score threshold for the IF and the classes prior sample probability for XGBoost. The optimal value for all these choices has been chosen to optimize the F1-score.

The results on the test set can be seen in tables 2 and 3.

It can be noted that the performances are quite similar with the same detection rate and a very small difference (w.r.t. the number of records) in false positives. This suggests that when

Table 3. Isolation Forest confusion matrix

actual/predicted	fail	ok
fail	9	2
ok	3100	552607
F1	0.0058	
Recall	0.8182	
Precision	0.0029	
False Alarm Rate	0.0055	

the classes are so imbalanced and the anomalous class, even if known, is not representative of the anomalous population (because not every failure condition has already been observed), while a classifier can and do work well, an unsupervised algorithm can be a better choice.

4. Unsupervised online pipeline

The scenario analysed up until now assumes to have a significant amount of data from which starting a training procedure that defines the normal behaviour is possible. It also ignores the possibility of having a continuous data stream that is impossible to actually store. Moreover, it ignores the problem of *concept drift* which highlights that the process under observation can evolve and change over time. This may be a natural occurrence in the phenomenon and not an anomaly, or it can be anomalous as it first happens but it then becomes the new 'normality'. It can be interesting to be able to detect and report the change, especially if is an abrupt and massive one, this is the aim of change-detection algorithms. But after the first occurrence, it would be best if the online anomaly detection algorithm could be able to automatically adapt, and if necessary stop reportinh as anomalous the changed data stream. This can involve, for example, a new training step or a revision of the algorithm or/and its parameters.

While examples of this kind of problem can come from the most diverse domains, such as when an industry acquires a new client which requires an increase of production, or in user-modelling applications where the goal is to check for abnormal behaviour the user might just change habits, in the context of a data center it can occur when the underlying process which generates the data being stored changes.

Some algorithms can handle data streams and online learning natively by having an update mode with a memory mechanism and a fixed memory requirement. An example of a very efficient algorithm of this kind is Randomized Subspace Hashing (Sathe & Aggarwal, 2016). Alternatively, algorithms that are offline in nature can be adapted to the online case by a higher-level strategy.

4.1. A general strategy to move from offline to online

A possible general strategy to adapt an offline algorithm is to set a window length in which data is stored to later update the model, while this is not the only option, it is quite easy to

implement. The general pipeline could be:

- collect enough data to do a first comprehensive training phase, as if the algorithm was to operate in an offline mode
- choose a window length w and store new arriving data until it is filled, output an outlier score for each new datum as soon as possible
- once the window is filled, erase the first arrived w data from the current dataset (the oldest ones), add the newest w points and start a new training procedure on the updated dataset, until the new model is ready continue storing new points in the (now empty) window and output anomaly scores with the old model
- once the new model is trained delete the old one

The window acts basically as a temporary storage area where new points are accumulated until there are enough to start a new training procedure by deleting the same number of points starting from the oldest in the dataset. This procedure allows the model to keep up with concept drift by always being updated with new data while forgetting the oldest, it also manages the infinite length by storing and using a fixed-size dataset plus a fixed-size window. The window length and the starting dataset set size are critical parameters that must be chosen with care as to allow the model to be trained in time between each collection phase while having enough data to always model the non-anomalous class. Some algorithms may not require a full new training phase but could allow a partial update on just the new data.

If an anomaly is tagged by the described procedure it can either be added anyway to the window for further training, flagged and discarded, or flagged and stored in a separate window. A mix of these strategies can be used basing the choice on thresholds on the anomaly score. If all anomalies are transferred into the training set the algorithm can be made almost immune to concept drift, but it can also stop detecting useful anomalies, while if anomalies are tagged and discarded, the algorithm, while handling the infinite sequence length, is prone to be afflicted by the consequences of concept drift. Human experts can inspect flagged anomalies and decide if they should be incorporated in the training set. Mixed strategies based on automatic thresholds can, for example, allow points with a score that make them suspicious but not really alarming to be included in training, while excluding points with a really extreme score. Such a strategy, while being naive and having numerous critical choices involved in using it, makes virtually any offline algorithm able to operate in an online mode. An example of this approach for the Isolation Forest algorithm can be found in the work of Ding & Fei, 2013 or more recently in the work by Zhang, Wang & Zhang (2019). In the next section, an example of such a pipeline will be given.

Table 4. Sliding window with educated guess for the threshold

actual/predicted	**fail**	**ok**
fail	7	4
ok	1643	544876
F1	0.008	
Recall	0.636	
Precision	0.004	
False Alarm Rate	0.003	

Table 5. Sliding window, default

actual/predicted	**fail**	**ok**
fail	11	0
ok	54919	491600
F1	0.0004	
Recall	1.00	
Precision	0.002	
False Alarm Rate	0.1	

4.2. Empirical evaluation results

A pipeline similar to the one described above has been used on the same test set of the offline pipeline. More precisely all data points from the week before the day to be tested are used to fit an Isolation Forest which is then used to compute an anomaly score for the data collected at the present day (for which a label of 1 means that tomorrow the hard drive will fail). This represents a sliding window with a seven-day length and a look-ahead period of one day. The model is fitted on the whole population instead of on a per-hard-drive basis as the throughput is not high enough with a datum per day per hard drive.

The confusion matrices obtained day per day are cumulated for the whole test period. Using an informed decision for the anomaly score threshold of the Isolation Forest, based on failure occurrences in the training period, the sliding window approach is able to obtain results comparable to the offline case, albeit with a different trade-off between false positives and false negatives, as seen in table 4. If instead, the default parameters are used, simulating a situation where absolutely no information is available, the online procedure produces a lot of false positives, as can be seen in table 5, but with a detection rate of 100%. It must be noted that in a real-world scenario it is unlikely that no action would be taken to adjust the threshold for giving an alarm in more than three months and at the same time that absolutely no history is available to have at least a rough idea of how to set it.

5. CONCLUSIONS

In this work, it has been shown how an unsupervised method can be used to monitor a set of more than five thousands hard drives for more than a year and particularly how it compares in a favourable manner with respect to a state of the art classifier. Moreover, a general pipeline on how to adapt an offline

anomaly detection algorithm to the online case has been presented and tried in a simple way to the same dataset.

In general the possibility of using a whole population of different objects of the same type, in this case, hard drives, to monitor the single parts in an online fashion, without specific, per-object, training, has been studied and proven possible providing that sensible parameters choices can be made.

An interesting possibility for future work would be to start in an unsupervised setting and start to use labels as soon as they are available in order to fine-tune the parameters of the algorithm and explore the optimal policy to do so, moreover, it would be possible to combine a classifier trained on labels and an anomaly detector that works in an unsupervised way, again the optimal policy on how to combine the two outputs should be studied.

REFERENCES

Charu C. Aggarwal, *Outlier Analysis*, Springer International Publishing, 2017 ISBN 978-3-319-47577-6

Fei Tony Liu, Kai Ming Ting & Zhi-Hua Zhou, *Isolation Forest*, ICDM '08 Proceedings of the 2008 IEEE International Conference on Data Mining

Page, E.S., *"Continuous Inspection Schemes"*, Biometrika 41, 100-115 (1954)

Chen Tianqi and Guestrin Carlos, *XGBoost: A Scalable Tree Boosting System*, Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016

Saket Sathe, Charu C. Aggarwal, *Subspace Outlier Detection in Linear Time with Randomized Hashing*, 2016 IEEE 16th International Conference on Data Mining (ICDM)

Zhiguo Ding, Minrui Fei, *An Anomaly Detection Approach Based on Isolation Forest Algorithm for Streaming Data using Sliding Window*, 3rd IFAC Internaton Conference on Intelligent Control and Automation Science, 2013

Freund, Y. and Schapire, R., *A decision-theoretic generalization of online learning and an application to boosting*, 1997

Charu C. Aggarwal , Alexander Hinneburg , Daniel A. Keim, *On the Surprising Behavior of Distance Metrics in High Dimensional Space* Lecture Notes in Computer Science, 420-434, Springer, 2001

Scholkopf, Platt, Shawne-Taylor, Smola & Williamson, *Support Vector Method for Novelty Detection*, NIPS, volume 12 January 1999

Breunig, Kriegel, Ng & Sander, *LOF: Identifying Density-Based Local Outliers*, Proc. ACM SIGMOID 2000 Int. Conf. On Management of Data, Dalles, TX

K. V . Vishwanath and N.Nagappan. *Characterizing cloud computing hardware reliability. ACM Symposium on Cloud Computing*, DBLP, 2010

Rodericki Bauer, `https://www.backblaze.com/blog/ssd-vs-hdd-future-of-storage/`, 6 March 2018

Nicolas Aussel, Samuel Jaulin, Guillaume Gandon, Yohan Petetin, Eriza Fazli, Sophie Chabridon *Predictive models of hard drive failures based on operational data.* ICMLA 2017: 16th IEEE International Conference On Machine Learning And Applications, Dec 2017, Cancun, Mexico. pp.619-625

Slađana Đurašević and Borislav Đorđević *Anomaly detection using SMART indicators for hard disk drive failure prediction* IcETRAN conference proceedings, 2017

Botezatu, Bogojeska, Giurgiu and Wiesmann *Predicting Disk Replacements towards Reliable Data Centers* Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016

Tinglei Zhang, Endong Wang, Dong Zhang *Predicting failures in hard drives based on isolation forest algorithm using sliding window* IOP Journal of Physics: Conf. Series 1187 42084, 2019