Lehrstuhl das Fach Verteilte Informations und Multimedia-Systeme

Health Status for Hard Drive Failure Detection

Masterarbeit von

Miguel Vieira Pereira

1. Prüfer	2. Prüfer
Prof. Dr. Harald Kosch	Prof. Dr. Michael Granitzer

July 8, 2025

Contents

Abstract

List of Figures

List of Tables

1 Introduction

1.1 Motivation

The amount of information produced and processed in the world has seen a steady increase over the past decades. The world internet traffic is increasing exponentially since a few decades. Edholm's Law[1] predicts that this behaviour should continue until at least 2030. It shows that telecommunications data rates are rising in a manner analogous to the one predicted by Moore's Law[2]: doubling every 18 months. Even 20 years after this prediction was done, it is still used by organizations responsible by developing the infrastructure to transmit huge amount of data [3].

As the amount of data generated by users all around the world increases, the need to store and access this data increases accordingly. This evolution is matched by an expansion on the number of digital services offered to users: musics, videos, books and other types of files and media that can be accessed from anywhere. The solution companies found to these problems was Cloud Computing. Even though Cloud Computing is a vague term, one way to define it is "a technique where IT services are provided by massive low-cost computing units connected by IP networks"[4].

However, even though Cloud Computing is sold as a way for users to access computing resources they don't have physical access to, the hardware responsible for this computing power must be placed somewhere. This is achieved by the establishment of multiple data centers all around the world to which devices from anywhere can connect in order to get access to the desired services. Microsoft, for example, has 300 data centers worldwide to provide their services to their clients[5].

A data center is a complex installation that consists of thousands of hard drives connect by kilometers of optical fiber cables. Therefore, there are massive amounts of investment done in order to create and maintain these facilities. Microsoft is investing $80 billion

in order to improve their data centers[5]. So there is a demand for services related to the maintenance and improvement of data centers.

In this context, one of the main aspects of Cloud Computing in general is the strong virtualization and reliability of the system[4], meaning that even if some of the hardware fails the service should still be provided without interruption. So, as hardware components fail, they need to be replaced to ensure that the system keeps working for a long time.

Moreover, in order to maintain a reliable system it is not enough to only replace the hardware after it fails. In order to prevent some serious issues such as data loss without having to permanently have a copy of all the data it is necessary to predict the hardware failures.

Therefore, one of the main problems faced by data centers is the need to detect which pieces of hardware are going to fail before a fatal crash occurs to allow it to be replaced. This allows both the prevention of data loss.

Out of all the components that are part of a computer, 80% of the failures occur due to the problems on the hard drives, as indicated by Table 1.1 Therefore, there is a special focus on predicting failures on disks.

Hardware vendors are well aware of this situation and the difficulties related to managing thousands of hard drives. So, in order to allow their users to better tackle these problems, they added a monitoring system to their drives. This technology is called SMART (Self-Monitoring, Analysis, and Reporting Technology).

Using this protocol, vendors can provide to users indicators of the disk status to the user. Some of these attributes are Power-On Hours, Air Flow Temperature and Reallocated Sector Count (number of sectors that had to be copied elsewhere on the disk after a failed read or write operation in order to prevent data loss)[7].

The SMART system can also include status of different attributes. However, these are limited to threshold not exceeded or threshold exceeded for each attribute, where the threshold is a reference value set by the vendor[7].

Moreover, even when there is an indication that a threshold has been exceeded, it does not mean that the drive will suffer an unrecoverable failure. Sometimes it can be a sign that the drive will work more slowly than its specifications indicate.

Device	Proportion
HDD	81.84 %
Miscellaneous	10.20 %
Memory	3.06 %
Power	1.74 %
RAID card	1.23 %
Flash card	0.67 %
Motherboard	0.57 %
SSD	0.31 %
Fan	0.19 %
HDD backboard	0.14 %
CPU	0.04 %

Table 1.1: Data center failure percentage by component - [6]

In addition to that, these status provided by the vendors consider each attribute independently. They do not take into account the fact that some problems in the hard drive can cause different indicators to increase simultaneously, so by considering the relationship between different attributes would allow problems to be more accurately detected. For example, having a huge number of Power-On cycles when compared to Power-On hours may indicate a problem in the disk that causes it to crash and restart constantly.

Finally, this ad-hoc approach uses static values and do not consider the evolution of the attributes over time. Imagine the situation of a disk that has not reallocated any disks so far, and suddenly rate at which sectors have to be reallocated increases. By taking into account how this attribute evolves over time, this problem can be more quickly detected before the threshold value is reached.

However, even though the built-in failure detection system is not very effective, it doesn't mean that the data itself cannot be used to obtain more interesting results, after all the problem of failure detection must still be solved. Current research uses mostly machine learning approaches such as SVMs and Neural Networks to tackle this problem. Further details of how these methods work will be detailed in Chapter 2.

This project has three main objectives. The first one is to create a library that implement some of the current approaches used in research. The idea is to also made it in such a way that it can be extended with new methods in order to allow them to be easily tested and compared to other ones.

After creating the library, further tests can be performed on existing methods to understand how they evolve as their parameters change. For example, the research of Zhu et al. on backpropagation neural networks for disk failure[8] uses a neural network with one single hidden layer with a fixed amount of 30 nodes on it. With our library, we can change the number of nodes of a neural network analogous to the one presented by them and study how the results change when the number of layer or nodes is altered.

The third objective is to extend existing methods to also make use of the concept of Health Status introduced by Xu et al.[9]. Their idea is to give a score from 1 to n to each sample depending on how close it is to a failure, instead of using only a pass/fail status.

So, the goal is to extend other methods to support more than two classes and study their reaction to this change. Different models can be extended this way, whether they are other neural network based methods such as [8] or even if they use other approaches such as the Random Forest one presented by Shen et al.[10].

1.2 Problem

Let $x_{i,t}$ be the vector in which the j^{th} component designs the measure of the j^{th} SMART attribute for disk i at time t. As input, we also have a value y_i that is either 0, if this disk has eventually failed or 1 if, until the end of the observation period, the disk kept working properly. Also, let the number of different smart attributes be equal to m.

Then given a series of vectors with the SMART attributes for an unseen disk, $\hat{x}_t$, we want to output a value $\hat{y}_t$. Here, $\hat{y}_t$ should be equal to 1 if the algorithm predicts that the samples $(\hat{x}_1 \dots \hat{x}_t)$ are closer to the samples in the input that did not fail than to the ones that malfunctioned, and 0 otherwise.

In a real world scenario, if the value $\hat{y}_t$ is equal to 1, then a warning should be sent to the people responsible for maintaining the data center in order to replace the concerned disk.

This corresponds to a classical classification problem. The numerical vectors as input and a binary output suggest that machine learning methods such as Classification Trees[11], Support Vector Machines[8] and Neural Networks[9] are well adapted to tackle this problem.

However, this problem has a few specific characteristics. First of all, the samples are not completely independent, since the values $(x_{i,1}, x_{i,t})$ are a time series that describe the status of disk i through time. As we will see on Chapter 3, this can be a motivation to use methods such as LSTM (Long Short-Term Memory) networks that can encode and work with time dependencies.

Moreover, even other methods such as Random Forests that can't easily deal with time dependencies can be extended to make use of the time series aspect of the data. This can be one by appending new components to $x_{i,t}$ corresponding to the value $x_{i,t}[j] - x_{i,t-\delta}[j], j \in \{1, \ldots, m\}$. In this case, δ is a constant that allows the algorithm to detect sudden changes in the value of an attribute[10].

A second point that needs to be mentioned is that the amount of disks in each class is not similar at all. Since a hard drive has a service life of around 3 to 5 years[12], and observations are performed for a few weeks, no failure will be observed for most of the disks. The ratio of failing disks over the total amount is between 0.4% and 1.9%[9]. So, any approach to this problem has to handle this imbalance in the input data.

Thirdly, the meaning of each SMART attribute is not the same for different vendors[7]. The protocol is standardized, but what it reports is not. So, an algorithm for failure prediction cannot be trained on data from different vendor disks. Moreover, in general, it is not a good idea to mix data from different disks on the same dataset, since they can have different failure causes profiles.

Most of the time, this does not pose a problem to the datasets generated by the data centers. This is due to the fact that, in practice, a data center uses hundreds of copies of the hardware of the same model and has at most a couple of different models. This allows for the replacement if failing disks do be done in a cheaper and more swiftly manner, since it simplifies stock management.

When evaluating an algorithm there are two main metrics that must be taken into account. The first one is the FDR (Failure Detection Rate). This is equal to the ratio

between the number of disks that are correctly predicted to be failing and the total amount of failing disks. Its value should be as close to 1 as possible

The second metric is the FAR (False Alarm Rate). This is equal to the ratio between the number of disks that are wrongly predicted to be failing and the total amount of disks that do not fail. Its value should be as close to 0 as possible.

The challenge is to find algorithms that find an equilibrium between this values. If it predicts every $\hat{x}_t$ to give an output $\hat{y} = 1$, the FAR will be equal to 0, but the FDR too. On the other hand, if it predicts every $\hat{x}_t$ to give an output $\hat{y} = 0$, the FDR will be equal to 1, but the FDR too. So, we notice that a trade off must be done between the FDR and the FAR.

It does not mean, however, that both metrics should be treated identically. Suppose that during a period of one month, 2% of the hard drives in a data center fail. This corresponds to a lifetime of about 4 years, which may represent a real world scenario. If the FDR decreases by 1%, then only 0.02% of the disks of the data center will fail without a warning during one month.

In contrast, if the FAR increases by the same amount, then 0.98% of the disks of the whole data center will be unnecessarily replaced during the same period. This is close to 50% of the disks that actually need to be replaced and can incur a substantial cost. So, most algorithms in the literature prefer to have a bias towards classifying a disk as working rather than failing.

Aditionnaly, it is impossible to reach perfect values for both indicators. This is due to the fact that hard drives are subject to real world conditions. For example, a short circuit on a disk may be caused by an electrical surge, but it does not depend on the internal state of the disk and thus cannot be predicted by the approach using SMART attributes.

Some additional steps can be made after executing the machine learning algorithm. A common one is the use a voting algorithm. The simplest way this can be done is as follows[10]: each sample $\hat{x}_t, t \in \{1, \ldots, T\}$, often including the change rate components, is evaluated independently. A sequence $\hat{y}_t, \{1, \ldots, T\}$ is obtained. Then, a fraction f is chosen. If more than $f \cdot T$ of $\hat{y}_t$ is equal to 0 then the output of the system is 0, otherwise it is 1.

The main advantage of this method is that it allows to easily control the trade off between the FDR and the FAR which can be desired as we have seen above. It suffices to increase f in order to decrease the FAR at a cost of also decreasing the FDR.

2 Background

Given the way the problem can be formulated according to Section 1.2, the failure detection problem can be viewed as a classification problem.

In this case the SMART attributes are the input variables. We have two classes: **good**, which corresponds to the disks that did not fail during the period that they were observed, and **bad**, which are the ones that did fail.

Therefore, machine learning approaches are the most appropriate to tackle the problem. In the literature, some of the machne learning methods used are Decision Trees, Random Forests, Recurrent Neural Network and Long Short-Term Memroy Networks.

In the next few sections we will discuss how these approaches were implemented and the results that they obtained.

2.1 Decision Tree

Decision Trees represent a flowchart in the form of if-else statements. Because of this, one of their main advantages is the fact that it is fairly easy to interpret a decision tree and to understand how it works and how it arrives at its conclusions.

Formally, a Decision Tree is a rooted tree $G = (V, E), E \subseteq V^2$. The leaves of the tree are labeled with one of the possible results of the decision problem. In our case, it will be either **good** or **bad**.

The other nodes are called decision nodes. When evaluating an unseen sample, consists in traversing the tree starting from the root and taking a left or right child of the current node depending on the conditions in it until a leaf is reached, which corresponds to an output.

For the following discussion, it may be useful to have at least a superficial understanding of the training process of a Decision Tree. The tree starts with a single node, the root, and all the training samples are placed there. The root is then added to a queue.

While the queue is not empty, the first node in it is processed. The processing step starts by choosing a splitting value c for each attribute.

Then, for each attribute i, the samples are divided in two sets depending on wheter its value is bigger or smaller than a threshold value c_i. One way to obtain the threshold value is to compute the average of the attribute i over the samples assigned to the node being processed.

Then the algorithm computes which of the partitions maximize a certain criterion function. A common criterion to use here is the information gain, which is the opposite of the Shannon Entropy [13].

Then, two new nodes and edges are created. The original node is then labeled by (i, c_i) where i is the index of the attribute that maximizes the criterion function. The training samples that were assigned to the node are they split between its two children according to the value of their attribute i.

If some condition for one of the children is met such as a certain depth or all samples are in the same node, then the is not added to the queue. Instead, it is kept as a leaf with a label that corresponds to the class that appears most frequently in the samples assigned to it.

Otherwise the node is added to the queue to be processed later.

After being trained, when the tree needs to predict the class of a certain sample x it will start traversing the tree from its root. Then, at each non-leaf node, it will read the label (i, c) and if x_i is smaller than c then the next node in the traversal is the left child, else it is the right child.

The training and the evaluation processes are illustrated by **Algorithm 1** and **Algorithm 2** respectively.

Algorithm 1: TrainDecisionTree(T: train set)

```
1  r ← Node(), q ← Queue();
2  r.samples ← T;
3  q.push(r);
4  while not q.empty() do
5      n ← q.pop(), maxi ← ∞, att ← −1;
6      c ← −1;
7      for i ← 1 to m do
8          c′ ← avg(n.samples, i);
9          s1 ← {x ∈ n.samples | x_i ≤ c′};
10         s2 ← {x ∈ n.samples | x_i > c′};
11         gain = criterion(s1, s2);
12         if gain > maxi then
13             mini ← gain;
14             att ← i;
15             c ← c′;
16             n.left.samples ← s1, n.right.samples ← s2;
17         end
18     end
19     n.attribute ← att;
20     n.threshold ← c;
21     for child of n do
22         if shouldProcess(child) then
23             q.push(child);
24         end
25         else
26             child.isLeaf ← true;
27             child.result = mostCommon(child.samples);
28         end
29     end
30 end
```

Algorithm 2: EvaluateDecisionTree(*tree*: Decision Tree, x: sample)

```
cur ← tree.root;
while not cur.isLeaf do
    if x[cur.attribute] ≤ cur.threshold then
        cur ← cur.left;
    end
    else
        cur ← cur.right;
    end
end
return cur.result;
```

The explicability of this model comes from the fact that when a sample is evaluated, the decision steps that resulted in the corresponding output can be followed, as show in **Algorithm 2**. So, it is possible to verify which parameters are taken into account and to predict what would happen if one of then was changed. Therefore the impact of each attribute can be interpreted.

Decision Trees come in two flavors: Classification and Regression Trees. The former corresponds to classifying samples in discrete, independent classes. So, Classification Trees will treat **good** and **bad** samples as different entities.

Regression Trees are concerned in predicting a continuous value. So, they allow us to use the concept of health status.

Suppose we try to give an integer score from 1 to m to each drive. The ones that do not fail correspond to a value of m while the samples of the failing drive are given values from 1 to $m - 1$ depending on how close they are to the moment of the breakdown. Then a Regression Tree will not try to simply predict an integer from 1 to m, intead it will try to find an exact value to it.

We can explain the difference between the two approaches when using multiple classes as follows: imagine we have a training sample whose desired output value is 1. Then, for the Classification Tree the error is the same when it puts it in the class 2 and in the class m.

On the other hand, a Regression Tree uses a mean squared error funcion, so even if it doesn't put the sample in the class 1 it will consider it worse the cases in which it is put

in class m compared to when it is put in class 2. It may even predict a value of 1.5 to the sample, even though there is no sample in the training set with this value for the dependent variable.

Compare this to the task of classifying a shape as a square, a triangle or a circle. Then, unless there is some additional issue specific to the context of the application, misclassifying a circle as a square is not worse than classifying it as a triangle . This fundamentally differs from the task of predicting a health status in which it is better to predict a 1.5 to a sample whose expected output is 1 than to predict a 5 for example.

Decision Trees were used by Li et al. [11] in order to obtain an algorithm capable of predicting hard drive failure. They tested both Classification and Regression trees.

In their approach, they were able to obtain an FDR of more than 95% while keeping the FAR below 1%.

Apart from the result in itself this work introduce some additional concepts that are useful for the implementation of other methods and is actually used by later researchers. First of all, they add a feature selection step that keeps only a subset of the SMART features passed as input.

More interesting though is the fact that they add columns to their table. This corresponds to the variation of the value of certain features over an interval. It allows the algorithm to take into account the fact that the training samples for a specific hard drive represent a time series.

More precisely, they choose a value for T. Let the samples for the i^{th} hard drive be the list x_i in which each entry corresponds to a sample and they are ordered in the order they were taken. Let $x_{i,j}[t]$ be the j^{th} feature of the sample taken at instant t. Then the value of the new column for the vector $x_i[t]$ is:

$$x_{i,n+j}[t] = x_{i,j}[t] - x_{i,j}[t-T], j \in \{0, \ldots, n-1\}$$

Where n is the original number of features in the vector.

This is an elegant way to include the time dependence aspect of the problem. The main advantage is that it can be applied to any method to try to improve methods that are not designed to work with time series.

In their work, they also use a voting algorithm. This way, in order to classify an unseen hard drive, they take the last N samples and evaluate each of them using the Decision Tree. If more than $\frac{N}{2}$ of them are put in the **bad** class, then the hard drive is classified as failing.

However, their research on Decision Trees does not implement some other techniques used by other research projects and that could be included. For instance, they do not test different thresholds for the voting algorithm. They always use a ratio of 0.5.

In addition to that, they always train the model with a constant ratio between good and bad disks in the training set. For each 3 **bad** ones, they include 7 **good** ones. They do not study what happens if, for example, for each **bad** HD there are 10 **good** ones, which more closely represents the real world scenario if no data is filtered.

In the end, the results they obtained are promising. They obtained FARs below 0.5% while keeping the FDR around 95% for the Classification Tress.

The Regression Tree model used a continuous value between -1 and 0 as the health status value, linearly dependent on how much time before the breakdown of the drive is. This approach was able to increase the FDR by 1% while also decreasing de FAR by around 0.2% when compared to the Classification Tree implemented by them.

2.2 Random Forest

One of the problems faced by Decision Trees, specially when they become large is overfitting [14]. This is due to the fact that the split values for each attribute used by the tree are directy taken from the values on the training set. So, when the tree is deep and there are only a few training samples in a node, the splitting values will sharply follow the ones in the training set.

Another way to explain this is that since a Decision Tree can closely follow the patterns observed in the training set because it can choose the splitting threshold independently of the ancestor nodes in the tree, it presents a small bias. But in Machine Learning there is a principle known as the Bias-Variance tradeoff that states that a model with low bias will present a high variance and vice-versa [15].

More concretely, for our problem at hand, suppose that there are only a few samples in the training set with a certain cause of failure. During training, if they are in a node **N** and are mixed with other samples, we may be able to correctly put them in a leaf that is a child of **N**.

So, the model will have a good performance on the training set which corresponds to a small bias, since the expected and predicted value will be the same. However, imagine that when evaluating an unseen sample there is an attribute, that is not important, to identify this cause of failure, is slightly different from the values observed in the training set.

Then, the path taken will not go through **N** and thus it won't go through the same evaluation process as samples that are simillar. As a result, a small change in a random attribute can cause a huge differnce on the path traversed and thus on the result. Having an algorithm that can behave very differently for slightly different inputs corresponds to a model that has a high variance.

In order to tackle this, the concept of Random Forests has been introduced [16]. The idea is to train multiple, independent Decision Trees at once, each one trained with a different subset of the training data. Notice that these subsets do not have to be disjoint, specially when the train set is relatively small.

Random Forests have performed well for a variety of tasks, ranging from image recognition to Alzheimer's disease detection and prediction [17].

The set of trees has, therefore, a smaller bias when compared to the one trained using all the data at once. This is due to the fact that we can reduce the probability of overfitting, since there will have a larger varierty of decision nodes.

If we return to the example of a specific cause of failure, the training samples corresponding to it will not even necessarily be on the same tree anymore. So, there is a larger set of nodes that have been trained with samples corresponding to this failure. Therefore, the probability of going through one of them when traversing the tree is higher even if the unseen sample does not correspond perfectly.

The drawback is that since the training samples with the same failure cause will be spread, the probability of having a node that had multiple ones when training is smaller, which can increase the error on the training set and on samples very simillar to the ones in the training set.

Once the set of trees is created and trained, it remains to decide how to regroup all of them.

The most common approach when asking for the model to predict the result for a sample is to take the value predicted by each tree and then combine them. For a classification problem, it can be returning the most common predicted class, while for a regression problem it can be done by taking the average of the outputs of the trees.

This can be done quite efficiently, since, at each node, it suffices to compare the value of a specific attribute with a threshold and go to the corresponding child. Moreover, since the traversal processes are independent, they can even be done in parallel.

A research by Shen et al. [10] used Random Forests for hard drive failure prediction. One of the most interesting aspects of their research is how they combined the results of the different tree of the forests.

Instead of simply checking whether most trees had an output of **good** or **bad** for a specific sample, they giving different trees different weights based on a clustering algorithm.

More specifically, they performed the clustering process for the good and bad samples independently. Then, when predicting the outcome for an unseen sample, the clusters c_1 and c_2 to which it would belong if it were a good or bad one are computed.

Since it is known which samples of the training set were used to train each tree, it is possible to evaluate the accuracy of each tree for the training samples in c_1 and in c_2. Based on these accuracies, it is possible to give a bigger weight to trees that better learned the samples of the clusters to which the sample being evaluate belongs.

They used a simple algorithm in which the weight of a tree is either 0 or 1 depending on whether its accuracy for its training sample belonging to either c_1 or c_2 is at least 0.5. However, it is not difficult to imagine a slightly more complex algorithm that results in a weight $w \in [0, 1]$ which is directly proportional to the accuracy.

Despite this simplicity, they were able to achieve a great performance. The FDR obtained was above 98% while the FDR was kept under 0.1% for multiple scenarios.

What allowed them to get these excellent outcomes, besides of course the algorithm, was their dataset that contained more than 2 million samples of SMART attribute snapshots.

2.3 Backpropagation Neural Network

3 Methods

Describe the method/software/tool/algorithm you have developed here

4 Results

Describe the experimental setup, the used datasets/parameters and the experimental results achieved

5 Discussion

Discuss the results. What is the outcome of your experimetns?

6 Conclusion

Summarize the thesis and provide a outlook on future work.

Bibliography

[1] Steven Cherry. "Edholm's law of bandwidth". In: *IEEE spectrum* 41.7 (2004), pp. 58–60 (cit. on p. 1).

[2] Gordon E Moore. "Cramming more components onto integrated circuits". In: *Proceedings of the IEEE* 86.1 (1998), pp. 82–85 (cit. on p. 1).

[3] Aarne Mammela and Antti Anttonen. "Why Will Computing Power Need Particular Attention in Future Wireless Devices?" In: *IEEE Circuits and Systems Magazine* 17.1 (2017), pp. 12–26. DOI: 10.1109/MCAS.2016.2642679 (cit. on p. 1).

[4] Ling Qian et al. "Cloud computing: An overview". In: *IEEE international conference on cloud computing*. Springer. 2009, pp. 626–631 (cit. on pp. 1, 2).

[5] *Microsoft Datacenters*. https://datacenters.microsoft.com/. Accessed: 2025-05-14. Microsoft (cit. on pp. 1, 2).

[6] Guosai Wang, Lifei Zhang, and Wei Xu. "What Can We Learn from Four Years of Data Center Hardware Failures?" In: *2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. 2017, pp. 25–36 (cit. on p. 3).

[7] *Communicating With Your SSD - Understanding SMART Attributes*. Tech. rep. Accessed: 2025-05-16. 2013, pp. 24–25 (cit. on pp. 2, 5).

[8] Bingpeng Zhu et al. "Proactive drive failure prediction for large scale storage systems". In: *2013 IEEE 29th symposium on mass storage systems and technologies (MSST)*. IEEE. 2013, pp. 1–5 (cit. on pp. 4, 5).

[9] Chang Xu et al. "Health status assessment and failure prediction for hard drives with recurrent neural networks". In: *IEEE Transactions on Computers* 65.11 (2016), pp. 3502–3508 (cit. on pp. 4, 5).

[10] Jing Shen et al. "Random-forest-based failure prediction for hard disk drives". In: *International Journal of Distributed Sensor Networks* 14.11 (2018) (cit. on pp. 4–6, 15).

[11] Jing Li et al. "Hard drive failure prediction using classification and regression trees". In: *2014 44th annual ieee/ifip international conference on dependable systems and networks*. IEEE. 2014, pp. 383–394 (cit. on pp. 5, 12).

[12] Kashi Venkatesh Vishwanath and Nachiappan Nagappan. "Characterizing cloud computing hardware reliability". In: *Proceedings of the 1st ACM Symposium on Cloud Computing*. SoCC '10. Indianapolis, Indiana, USA: Association for Computing Machinery, 2010, pp. 193–204 (cit. on p. 5).

[13] Claude E Shannon. "A mathematical theory of communication". In: *The Bell system technical journal* 27.3 (1948), pp. 379–423 (cit. on p. 9).

[14] Xue Ying. "An overview of overfitting and its solutions". In: *Journal of physics: Conference series*. Vol. 1168. IOP Publishing. 2019 (cit. on p. 13).

[15] Erica Briscoe and Jacob Feldman. "Conceptual complexity and the bias/variance tradeoff". In: *Cognition* 118.1 (2011), pp. 2–16 (cit. on p. 13).

[16] Tin Kam Ho. "Random decision forests". In: *Proceedings of 3rd international conference on document analysis and recognition*. Vol. 1. IEEE. 1995, pp. 278–282 (cit. on p. 14).

[17] Anjaneyulu Babu Shaik and Sujatha Srinivasan. "A brief survey on random forest ensembles in classification model". In: *International Conference on Innovative Computing and Communications: Proceedings of ICICC 2018, Volume 2*. Springer. 2019, pp. 253–260 (cit. on p. 14).

Eidesstattliche Erklärung

Hiermit versichere ich, dass ich diese Masterarbeit selbstständig und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt habe und alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, als solche gekennzeichnet sind, sowie, dass ich die Masterarbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt habe.

Passau, July 8, 2025

Miguel Vieira Pereira