UNIVERSIDADE D
COIMBRA

Bernardo Silva Carvalho

SAFETY MONITORING FOR MACHINE LEARNING IN CRITICAL APPLICATIONS

Dissertation in the context of the Master in Informatics Engineering, specialization in Software engineering, advised by Professor Raul Barbosa and presented to the Department of Informatics Engineering of the Faculty of Sciences and Technology of the University of Coimbra.

July of 2023

Acknowledgements

Here, I thank the people that help me during this journey.

I will start by thanking Professor Raul Barbosa, for his guidance all throughout this dissertation and for pushing me so that this thesis was the best it could be.

Thank you to the people that are a part of the CardioId project, in particular to João Vitorino for helping me do the feature extraction of the DROZY database and to Frederico Cerveira for giving me access to their dataset.

A personal thank you to my closer friends that I made while in the university, not only for your friendship but also for all the help during this course.

A final word to my family for all the help that they gave me, for always believing in me and keeping me motivated.

This work is supported by Project Reference ECSEL/0017/2019 and 876852-ECSEL-RIA-VALU3S, financed by Fundação para a Ciência e a Tecnologia, I.P./MCTES through national funds (PIDDAC) and funding from the ECSEL Joint Undertaking (JU) under grant agreement No 876852. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Sweden, Italy, Spain, Portugal, Czech Republic, Germany, Austria, Ireland, France and Turkey.

Abstract

Artificial Intelligence (AI) is becoming more sophisticated and more widely used in our day-to-day lives, from virtual assistants like Alexa or Siri to self-driving cars. The increase in use and capabilities of AI leads to it being more frequently used in critical systems, creating the need to have some type of system that can detect malfunctions. Despite all the research done to improve the performance of AI, the problem of error detection is still yet to be resolved, so we will propose an approach to overcome this problem. Our approach is based on the use of monitors to supervise the critical model and omit when it makes an error. We will have three different approaches for the monitor, from which will result five different monitors, one approach will be a Machine learning model to detect errors, another will calculate the distance to the classification boundary, and the last one will use clustering to know which points are covered by the training set. If this approach proves to be successful in detecting errors, it is expected that it starts to be used in critical systems so that it can improve Machine Learning (ML) safety.

Keywords

Monitoring, Critical systems, Dependability, Fault tolerance, Error detection, Selective Prediction, Artificial Intelligence, Trustworthy AI, Machine Learning, ROC, True Positive Rate, False Positive Rate

Resumo

Inteligência Artificial (IA) está cada vez mais sofisticado e mais presente nas nossas vidas, desde assistentes virtuais como a Alexa e a Siri até carros que conduzem sozinhos. Este aumento do uso e capacidades da inteligência artificial leva a que seja utilizado cada vez mais em situações críticas e para tal é preciso ter algum tipo de mecanismo que permite saber se houve algum error por parte da IA. Apesar de toda a investigação feita para melhorar a performance de IA, o problema de deteção de erros ainda se encontra por resolver, por isso apresentamos um método para resolver este problema. O nosso método baseia-se no uso de um monitor para supervisionar o modelo crítico e informar quando este faz um erro. Vamos ter três abordagens diferentes de monitores das quais vão resultar cinco monitores no total, uma das abordagens vai utilizar inteligência artificial para detetar erros, outra vai calcular a distância a fronteira de decisão e a última vai utilizar *clustering* para saber se os pontos estão cobertos pelo conjunto de treino. Se este método tiver sucesso em detetar erros espera-se que comece a ser utilizada em sistemas criticos pois pode melhorar a segurança de modelos de *machine learning*.

Palavras-Chave

Monitorização, Sistemas críticos, Fiabilidade, Tolerância a falhas, Deteção de erros, Previsão seletiva, Inteligência Artificial, IA Fidedigno, *Machine learning*, *ROC*, Taxa de positivos verdadeiros, Taxa de positivos falsos

Contents

Acronyms

AI Artificial Intelligence.

AUC Area Under the Curve.

FP False Positive.

FPR False Positive Rate.

ML Machine Learning.

NN Neural Network.

OOD Out-of-Distribution.

PCA Principal Component Analysis.

PPV Positive Predictive value.

ROC Receiver Operating Characteristic.

SVM Support Vector Machine.

TN True Negative.

TP True Positive.

TPR True Positive Rate.

XGB Extreme Gradient Boosting.

List of Figures

List of Tables

Chapter 1

Introduction

Artificial Intelligence (AI) is a technology that is expanding rapidly, and it is believed that it will have an immense role in the future of humanity and impact various industries, from the automotive industry to the healthcare industry.

The increased in data and complexity in healthcare led to an acceptance of AI which is expected to be extensively used in 10 years [1]. The organizations that already have started to embrace AI, are making use of Machine Learning, and deep learning [2]. Both these applications are beneficial for making predictions either with data or images. These capabilities can be paradigm-shifting in some areas of healthcare, such as radiology which is one of the areas that makes the most out of them. In this field of medicine, AI can be used to make imaging of the thorax, brain, and abdominal, as well as help, assess mammography or a colonic polyp [3].

The primary intent for the AI imaging is to increase efficacy and efficiency in clinical care [3] so that radiologists' productivity increases. This can be achieved when there is a seamless integration of AI with the radiologist workflow because he only needs to examine the images that already have the features identified, this can help in most phases of radiologist work, which are detecting, characterizing, and monitoring. It is worth reminding that the Machine Learning models are supposed to be used to support clinical decision-making [3], so it is a human-on-the-loop situation [4].

Another area that is being developed due to the capabilities of making predictions based on images and data of both Machine Learning and Deep Learning is the self-driving cars industry. There are already some semi-automated cars in the automotive industry, such as Teslas and Mercedes, and strides are being made to have fully automated cars.

With the development of Deep Learning and unsupervised learning, predictions and diagnoses will become more accurate due to better data utilization which means that it will be more widely used. However, this increase in performance is not enough to achieve trustworthy and robust AI, to achieve trustworthy AI, a vast number of parameters must be fulfilled, from the ethical and lawful side to the technical side [5].

1.1 Motivation

Although AI and Machine Learning can outperform humans in some tasks, there needs to be an assurance that the systems are safe since their impact on people's lives can be significant, and if mistakes were to happen, the consequences could be fatal. Machine learning models mainly deal with example-based supervised learning algorithms, which can lead to mistakes of receiving a case that is not prepared to handle, because of this there is a need to design an environment that can guarantee the models' safety. In order to guarantee a safe environment, we need to overcome the problem that is no error detection for machine learning models.

To fight this need, we will make use of software techniques that are used to achieve fault tolerance, more specifically by adding a supervisor that can monitor the model during its execution. With this supervisor we will be able to not only detect errors made by machine learning models but also prevent their propagation.

1.2 Research Objectives

The main goal of this thesis is to create monitors that guarantee the safety of Machine Learning models that are used in critical systems. This means that we have to create a system that can detect errors made by the model.

Error detection is a part of fault tolerance, a common way for it being introduced into a system is with the addition of a monitor. Adding this technique to a machine learning model is something that has been discussed about but the results of its capability are still unknown which leads us to our first research question:

RQ1: Can monitors precisely detect when Machine Learning models make incorrect predictions?

We have the following hypothesis, errors have patterns that allow for prediction of whether a machine learning model will provide a correct result or not. There is a wide range of possible ways to detect these patterns we will make use of three different approaches, some will be distance based and others will also include machine learning models, this leads us to the following:

RQ2: Out of the three techniques which one produces the best monitor?

We will have to compare the model with and without the monitor to evaluate if the addiction of the monitor was a worthwhile one. To know this, we will be comparing the overall performance of the model which will show if the monitor was beneficial or not, if the overall performance improves then we can assume that the model is safer, this leads us to our third research question:

RQ3: Does the model accuracy improve with the presence of the monitor?

As we mentioned error detection is a part of fault tolerance the second part is

recovery, where the errors are handled. Despite there being a variety of ways of doing this part we decided to enforce fail-silent behaviour which basically suppresses wrongful outputs, this means that if the model produces output it most likely is correct. From this arises our last research question.

RQ4: Can we enforce fail-silent behaviour on machine learning models?

After answering these questions, we should be able to have a better understanding if it is possible to have error detection in machine learning models, as well as if it the makes the model safer.

1.3 Contributions

The main contributions from this work are the following:

- Creation of five different monitors out of the three techniques presented;
- Experimentation of our approach in nine different datasets, including one that is being used by CardioID;
- A scientific paper that still is in preparation which can be viewed in the Appendix B;
- Evaluation of the performance of the monitors.

1.4 Document outline

In this section, we will portray the structure of this thesis.

In **Chapter 1**, it is presented the problem, motivation, and research objectives for this thesis.

Chapter 2 presents vital concepts such as what is trustworthy AI and also what is dependability concentrating on fault tolerance. We will also detail some of the most common errors that lead to Machine Learning models making mistakes. It will also demonstrate some of the uses of AI in critical systems and what techniques are being used to increase the performance of Machine Learning models. There is also a comparison between the different algorithms that can be used for a ML model. Finally, we present the metrics that will be used to evaluate our monitors performance and explain in more detail the techniques we will be using, like fault-silent behaviour.

Chapter 3 gives insight into what are the difficulties of having safety in machine learning when compared with more standard software. We also detail what techniques have been used to improve machine learning safety.

In **Chapter 4**, we will explain our approach to deal with the problem and explain the different ways that it can be achieved.

Chapter 5 presents results of the performance of all our different monitors to all the different datasets that we will be making use and analyse which one had the best results.

Chapter 6 will be the conclusion to this work and some thoughts for future work.

Chapter 2

State of the art

Due to the increased use of AI, there has been an increase in research to make it safer and more reliable. In this chapter, we are going to talk about the impact of AI on critical systems such as self-driving cars or healthcare and the need to have trustworthy AI. We will also detail what reliable AI is, what techniques have been used by the AI community to improve performance, and how dependability and safety can help in increasing reliability in Machine Learning (ML).

This chapter will also provide a better look at some of the classifiers that can be used in Machine Learning and demonstrate their advantages and disadvantages. It will explain what clustering and feature reduction are and why they are needed for this thesis. We will also define what metrics will be used to classify the performance of our monitors.

2.1 AI in critical systems

Although AI is still a novel concept, it is starting to be used in health and self-driving cars. There are already some applications in health care, such as diagnosing cancer and strokes, for health research, drug development, and disease prevention.

In health, AI can be used as a support tool, for example, aiding radiologists in making a diagnosis. It is also used to help in research about genomics and accelerating drug development. There are also some potential uses for public health surveillance to prevent diseases and have a better outbreak response. Despite the most apparent and impactful use of AI to have it detect anomalies, it can also be used in an administrative way to improve processes [1].

There are plenty of usages for AI in the automobile industry, from autonomous vehicles to virtual assistants. There are different levels of automation in a car, it can be a lower level with assistance to the driver, such as blind spot alerts or lane changing technology, or it can be a higher level like a fully autonomous driving car [6]. Regarding virtual assistants, it can be beneficial to monitor the emotional state and mood of the driver so that recommendations can be made, for example,

detect fatigue or an unstable state and suggesting a break which also has the purpose of increasing road safety and preventing accidents.[7]

Regardless of the context where AI is being used, it has the principal goal of automating and improving the systems by reducing human errors and increasing efficiency.

2.2 Trustworthy AI

Trustworthy AI has three main components according to the European commission [8] these components are that they should comply with all the applicable legislation and regulations, even though there is not a lot legislation in place for AI it will be developed as the technology advances, they should be **safe** and robust from a technical side which is what we are going to try to achieve in this thesis, lastly it also has the need to be ethical.

The importance of trustworthy AI in critical systems is paramount because failures can impact multiple people in a short time by making wrong predictions about people that have the same health conditions, whereas humans despite being capable of making mistakes their impact will be smaller [5]. This importance is particularly relevant in critical systems because of the more severe consequences of it not working correctly. For example, if an error occurs in an AI that makes suggestions of what music you might like to hear the result is not that harmful however if an error occurs when making a recommendation of which drug to use to treat a patient the outcome can be deadly.

The main focus on increasing the safety of AI is so that it does not cause unintentional harm. For this to be achieved, there is a need not only to create a safe and secure environment so that the system can perform in an intended manner but also need protection to guarantee that there is no harm done if an error happens to occur. Which is necessary so that we can enforce the principle of prevention of harm. This principle is strongly connected to the technical side of robustness which is to behave as intended, minimize unintentional harm and prevent it from happening. To have technical robustness and safety the following requirements must be realized, according to the AI European Commission[8]:

- **Resilience to attacks and security**

  There needs to be a secure environment so that our system cannot be coerced by malicious actions, this is what it is trying to be achieved with the adversarial training.

- **Fallback plan and safety**

  There must be a fallback plan in place in case an error occurs. This plan can be to ask a human for input before continuing its operation.

- **Accuracy**

  The accuracy of a machine learning model is the likelihood of it making a correct prediction, and it is given by the percentage of correct predictions in the test dataset. Since we are talking about critical systems, their accuracy is required to be high. However, it is not feasible to have a hundred percent accuracy which means that the model should have some way to indicate how likely it is to have made a mistake or even remove the prediction if it is not a confident one.

- **Reliability and Reproducibility**

  It is necessary that the results of an AI system are reproducible and reliable.

2.3 Causes of errors in machine learning

Even though we will not detect the root cause of the errors made by models it is worth knowing what might cause a model to make wrongful predictions. Some of the causes are described below:

- **Concept Drift**: This is a problem that can occur in real-time machine learning models, where there is hidden context that is not necessarily express through features. Changes in this hidden context have an impact in the ability of the model to make predictions [9], a good example for this can be a value prediction model for houses, even if the features have all the specification of a house the price of it is heavily dependent of what is the state of the market, which is impacted by inflation, new legislation, etc. There are two types of drift, it can be sudden or gradual.

- **Overfitting**: Is when a model does not generalize well from observed data to unseen data, this usually happens when the model memorizes all the data, including noise on the training set, instead of learning the patterns present in the data. This leads to the model performing perfectly on the training set and performing poorly on the testing set. [10]

- **Underfitting**: Is when a model fails to adequately capture the relationships between the variables in the data, it is the opposite of what happens in an overfitted model. The reasons for this happening might be the wrong choice of model, incomplete training [11] or due to data being imbalanced.

- **Data quality**: This is an overall problem that encapsulates many others, such as, missing features to represent the problem correctly, noise present in the data which might be attribute noise or class noise this can worsen the performance and efficiency of the model [12], not enough training data and class imbalance which might lead to underfitting.

- **Adversarial attacks**: They are small changes that have the goal of altering the predictions of models which can be problematic if done with malicious intent. There are a wide range of adversarial attacks like one pixel attack, fast gradient sign method or FGSSM, amongst others [13].

2.4 Techniques used in AI

In the search for more reliable and accurate machine learning models, strides are being made by the AI community in developing various techniques to increase the performance of machine learning models, and thus also increasing their trust.

The construction of a Machine Learning model typically revolves around a pipeline like the one depicted in the figure below.

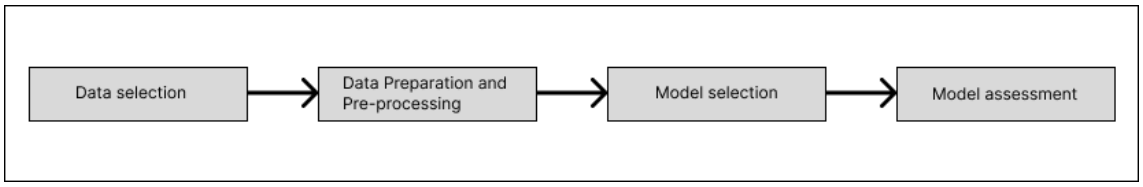


Figure 2.1: Machine learning pipeline

Other than different model algorithms there are other techniques that are used to improve the performance of machine learning models. Those techniques can be used in the different stages of the machine learning pipeline, here we have some of the techniques that are used for each stage of the pipeline:

- **Data preparation and Pre-processing**

  The techniques utilized in this stage have the goal of transforming raw data into a format that can be used by the machine learning model. Some of the techniques used are **features normalization**, which helps to deal with some linear regression problems when two features have entirely different scales. Some techniques to mitigate this problem are standardization, centering, and unit range[14], .**Feature reduction** is used when there is a need to lower the dimensional space of the original dataset. The dimensional reduction can be achieved with the use of techniques such as Principal Component Analysis (PCA) and T-SNE.[15] One technique that is often used to pass from categorical data to numerical is **data transformation**.

- **Model selection**

  In this stage we need to select which algorithm yields the best results for that there are several techniques, such as, **K-fold cross-validation** which can estimate the expected prediction error. This technique aims to have validation without creating a validation set [16]. The K-fold technique divides

the data into equal parts, the data until K-1 is used for training the model, and the rest is used to test the model. This method is repeated K times to calculate the estimated error. The typical values for K are 5 or 10. A particular case of K-fold technique where K=N is the **Leave-one-out cross-validation**, this technique only use one sample for estimating the error. So, it will have a low bias at the cost of high variance[16]. It is also during this stage where hyperparameter tuning happens to find what are the best possible configuration for the model.

- **Model Assessment**

  This stage is responsible for evaluating performance of the model. There are a lot of ways to evaluate the model such as metrics like sensitivity and specificity.

2.4.1 Clustering and Feature reduction

Clustering is a method that aims to divide data into separate groups or clusters that have significant similarities between themselves and great dissimilarity with the other clusters [17]. Clustering is one of the most used approaches when it comes to unsupervised learning, which is when the data that we want to use in the model does not have labels. One of the most well-known and used algorithms for clustering is K-means. Although we are not going to explain in detail how the k-means algorithm operates, it is significant to note that in this algorithm, it is necessary to pass the number of clusters in its initialization. This value has a vital role in the correctness of the algorithm. So, to improve clustering performance to the maximum, we will be using the elbow method to select the number of clusters [18].

It is common to use feature reduction in clustering to reduce the problem of dimensionality. Although it might worsen our clusters [19], it will allow the visualization of the clusters and enable all the calculations we will need to perform in calculating the distances of the points to the centroid, it essential to calculate the distance between categorical data. One of the most used feature reduction methods is PCA.

This method will prove vital for calculating distances for both the clustering and classification boundary monitor, because without the dimensionality reduction, the amount of time to calculate distances would take too much to have any real-world usage, in particular for the classification boundary monitor. After all, the time would increase with the number of features of the data, so it would only have acceptable performance on datasets with few features limiting its use.

2.4.2 Ensemble learning

Ensemble learning envelops methods that use multiple inducers, also known as base-learner, to make a decision. This methodology is based on human nature where we gather a range of opinions in order to make a complex decision, the

main idea being that weighing and grouping multiple opinions wields better results than only taking an opinion from a single individual [20]. Some requirements have to be fulfilled so that this method works better than a single inducer, they need to be independent, they need to be capable of making decision with local information, they also need to have a diversity of opinions and some mechanism to turn these private decisions into a collective decision.

There are a lot of advantages with this method, it helps avoiding overfitting, computational advantage because a single learner has a higher risk of getting stuck, one final big upside is that it can reach optimal hypothesis that cannot be reached by any single model [20], by combining multiple solutions to find the best option. Apart from these advantages it can also help solve various nontrivial machine learning challenges that we mentioned in Section 2.3 like, class imbalance and concept drift, as well as, tackling the curse of dimensionality, which is when there is an overwhelming number of features, and the models cannot generalize them.

There are two main frameworks that ensembles methods can fall into those being dependent framework where the output of each inducer affects the construction of the next one, and the other being independent framework where each inducer is built independently from other inducers [20]. Some of the most used methods in the dependent framework are AdaBoost and gradient boosting machines like XGBoost . When it comes to the independent framework the most used ones are random forest and bagging.

2.4.3 Knowledge distillation

One technique that is used mostly for its gain in efficiency although it can also improve the overall performance of models is **knowledge distillation**, also known as teacher-student model.

Knowledge distillation is about transferring the knowledge of a large model, the teacher, to a smaller one, the student. There are three different distillations schemes that can be used according to [21], offline distillation, which is when the knowledge is transferred from a pre-trained teacher model into a student one, then there is online distillation, that was developed to overcome some of the limitations of the previous scheme, it differs from it because both the teacher model and student model are updated simultaneously, lastly there is self-distillation where both the teacher and the student used the same network, and the knowledge from deeper sections of the network is distilled to shallower sections. All of these different schemes work for model compression, and both the online knowledge distillation and self-knowledge distillation can improve the performance of the deep model. Although this technique might be interesting for our context because it generates a smaller model which is usually simpler to evaluate, it is not what we are trying to achieve with our approach because the fundamental goal of a teacher/student model is the creation of a model that is more lightweight that serves the same purpose of the teacher, whereas we want to create a model that can detect the errors of another model, which means that their goal will be

wildly different.

Although these techniques are useful to enhance the performance of the model, they will ultimately suffer from the fact that machine learning models are highly dependent on their training data set, **which means that its reliability, precision, and recall are good if the training set is good and poor if the training set is mediocre** so basically, it is "garbage in garbage out". As pointed out previously most errors occur due to data, either it be that is does not explain the problem correctly or that is overfits, amongst other problems. Despite the fact that these techniques help mitigate some of these errors they do not outright remove them so there has to be different ways other than improving a model's performance to increase the dependability of machine learning models.

2.5 Dependability

The original definition of **dependability** is the ability to deliver service that can justifiably be trusted[22]. There is also another definition that says a system is dependable when the system is capable of avoiding service failures that are more frequent and severe than is acceptable. A common way of putting it is that the system should be as dependable as the trust placed in that system. Since **dependability** is such a vast topic, it encloses availability, **reliability**, **safety**, integrity, and maintainability. For this thesis, the most significant ones are reliability and safety because they guarantee that the service operates correctly for a given time frame and the absence of unexpected consequences for both users and the environment, respectively.[22]

There are several ways of achieving **dependability**, such as fault prevention, **fault tolerance**, fault removal, and fault forecasting. Fault tolerance and fault prevention both strive to deliver a service that can be trusted, so both of them are needed to achieve trustworthy AI.

2.5.1 Fault Tolerance

Fault tolerance aims to ensure that there are no service failures in the presence of faults, and it is carried out by **error detection** and system recovery, all the different fault tolerance techniques can be seen the figure below that was obtain from [22].

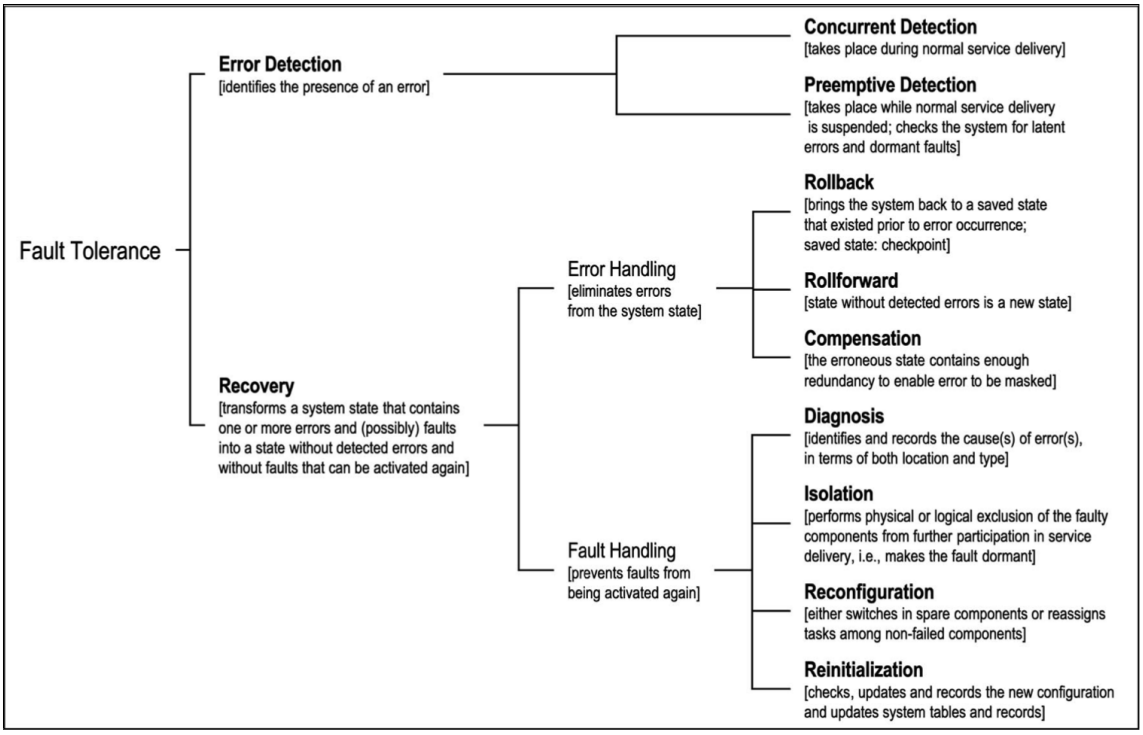


Figure 2.2: Fault tolerance techniques

Fault tolerance can suffer from a lack of fault tolerance coverage, and this can be either because of faults in the fault tolerance mechanism or due to fault assumptions that do not represent reality. There are two different ways to detect errors, preemptive detection, which is detecting errors before an error occurs, and concurrent detection which is detecting errors while the system is operating. Since our goal is to have runtime error detection we will be focusing on the latter. In order to have fault tolerance, we will use a specialized support system for fault tolerance, such as a **monitor**.[22]

Regarding the recovering portion of fault tolerance, we will be enforcing **fail silent behavior** in the ML models.

2.5.2 Online monitoring

Online monitoring can provide increased robustness, security, fault tolerance, adaptability to the system they are monitoring, and detect **runtime errors** [23]. In this case, the Machine Learning model will be considered a piece of software, meaning that a software monitor will be used. Since this technique is external to the model, it is able to complement other techniques to create safer Machine Learning models.

2.5.3 Fail silent behaviour

Since the goal is to implement fail-silent v in machine learning, it is necessary to define what fail-silent behaviour is. According to [24] fail-silent is when the

system only produces correct results, if an error happens to occur, the system does not output them. To enforce this behaviour in machine learning there is a need to detect the errors made by the model and then omit their output. This will be particularly valuable in a critical environment because it guarantees that every prediction of the model is correct, and the ones that the model did not produce output might have been wrong. This is technique that is used often in critical systems.

2.6 Classifiers comparison

Since we will use machine learning models further along in this thesis, we need to have an idea of what types of classifiers exist and what their advantages are, so we are going to make use of the following table adapted from [25].

Classifiers	**Advantages**	**Disadvantages**
Logic based algorithm: Decision Trees	Comprehensibility (easy to understand)	Difficulty/incapability to deal with lack of input information There is no common accepted algorithm to build DT (best features selection, e.g. C4.5) Requires pruning
Logic based algorithm: Learning set of rules	Comprehensibility (easy to understand)	Difficulty/incapability to deal with lack of input information
Perceptron based Techniques: Neural networks	Accuracy	Easy to occur overfitting situations No comprehensibility Incapability to deal with lack of input information
Probabilistic: Naïve Bayes (Bayesian networks)	Fast Simple Able to cope with lack of input information	Attributes' independence assumption
Instance based learning: k-Nearest neighbour	-	Incapability to deal with lack of input information Large computational time for classification Sensitive to the choice of similarity function to compare instances
Support vector machines: Support vector machine	Suitable when number of features is larger than the number of training instances	Difficulty to deal with lack of input information

Boosting: Extreme Gradient Boosting	Reduction of bias	Not scalable Dependent on outliers which can lead to overfitting

Table 2.1: Pros and Cons of classifiers

As we can analyze from this table, there are some differences between all these classifiers. A problem that may surface is the lack of information, which is not uncommon in the health industry, so classifiers that do not deal well it that lack of information might not be ideal, making a classifier like Naïve networks more appealing in this context.

Although it is outside of the goal of this thesis, it is worth mentioning that the model chosen has to be transparent so that it can be trustworthy according to the European Commission[8]. This will not only lead to the tradeoff of accuracy for better explainability as well as make the choice of which classifier to use more complex because most classifiers transform the data in ways that make it not comprehensible to humans, like neural networks and Support Vector Machine [26].

2.7 Metrics to evaluate monitors' performance

For evaluating our monitors, we will make use of some of the already established metrics in the AI community which, according to Stephen Marsland [27] and to [28], are:

- **True Positive Rate (TPR)**, or **recall**, or **sensitivity**, is the ratio of the number of correct positives (True Positive (TP)) out of the ones classified as positive.

$$TPR = \frac{TP}{TP + FN}$$

- **False Positive Rate (FPR)**, is the ratio of the number of incorrect negative classifications False Positive (FP) out of the ones that are classified as negative.

$$FPR = \frac{FP}{FP + TN}$$

- **Accuracy**, is the sum of correct classifications divided by all the classifications made, although it is not the best metric it is a simple metric that helps showing the performance of the monitor in a broader range.

$$Accuracy = \frac{TP + TN}{TP + FN + TN + FP}$$

- **Specificity**, or **TNR**, is the ratio of the number of correct negatives (True Negative (TN)) out of the ones classified as negative.

$$Specificity = 1 - FPR = \frac{TN}{TN + FP}$$

- **Positive Predictive value**, or **PPV**, is the ratio of correct positives out of the ones that are classified as positive.

$$PPV = \frac{TP}{TP + FP}$$

The **Receiver Operating Characteristic (ROC) curve** is generally used for visualizing and selecting classifiers based on their performance [29] so it will be useful to compare the performance of the monitors against each other. We will use multiple thresholds for each monitor so that we have multiple points and can create the curve. For each threshold we will calculate the TPR, which is the y-axis, and the FPR, which is the x-axis. After this step we can calculate the **Area Under the Curve (AUC)** which gives insight into the performance of the monitor. If the AUC is 0.5 it is as good as a random guess, if it is 1.0 then it is the perfect classifier and if it is 0.0 it is an anti-classifier. Specificity is the only metric that we will not specify directly in the results, but it can be easily extracted from the ROC curves since it is 1-FPR.

2.8 Conclusion

Due to everything discussed in this chapter, there is an evident need to have some way to detect when Machine Learning models make mistakes. Since that has not been achieve by the current techniques used in AI, we have to think outside the box and apply techniques used in software engineering by adding dependability strategies to Machine Learning models, more concretely adding fault tolerance through the use of a **monitor** that enforces fail-silent behaviour so that errors do not manifest themselves.

Chapter 3

Related Work

In this chapter we are going to explain the dependability limitations that exists in Machine Learning, and what existing methods and techniques are being used to get over these limitations and thus increasing the safety of machine learning models.

3.1 Safety limitations in machine learning

The main obstacle with guaranteeing safety for machine learning is that it does not comply with the traditional engineering development standards. This leads to five open challenges that compromises safety, according to [30], the first is design specification because it is not feasible to use formal design specification, another one is the inherited lack of transparency of machine learning models that makes it difficult to have traceability, one other is performance and robustness which is the most researched topic in ML which unlike code-base algorithms statistical learning algorithms tend to contain to residual error rate that they generate, which makes it harder to improve the performance, there is also the difficulty to use formal methods for verification due to the high dimensionality of the data, lastly runtime monitoring is also a challenge that has not been completed in machine learning because it naturally differs from standard software monitors that are based on a rule-set to detect hardware errors and software crashes.

3.2 Strategies for safe machine learning models

According to [30], there are three different strategies to achieve safety in Machine Learning:

1) **Inherently Safe Design**, refers to techniques that are used to design Machine Learning models that are error-free. Some of the techniques that are used for this strategy are model transparency, which is one the requirements

for trustworthy AI [8], design specification and model verification and testing, such as formal methods.

2) **Enhancing Performance and Robustness**, refers to techniques that improve the Machine Learning models performance, and their robustness, such as robust network architecture, robust training, and data sampling and augmentation. This is the strategy that is most commonly used, and it is an effective one because creating ML models that make less mistakes inherently makes them safer.

3) **Runtime Error Detection**, refers to techniques that can detect errors made by the Machine Learning model at runtime. This can prevent misclassifications, which leads to a safer model. Some of the key approaches to detect these errors are model uncertainty estimation, adversarial attack detection, out-of-distribution detection and runtime monitoring.

Out of these three strategies the one that is developed the furthest is the enhancing performance and robustness. This strategy envelops almost all the current research that is being done in the field. This is also because it envelops a lot of parts of the area, from new algorithms that have more accuracy or are more efficient to adversarial training.

One technique that is being explored is adversarial training [30], which trains the models with adversarial examples, it has the ultimate goal of increasing the model's robustness against attacks. Although, this is a great approach to fight against malicious attacks and a must have in a system where safety is required, the main problem is that it has high computational cost due to the creation of the adversarial examples, so the main research being done to overcome this problem is to make the creation of adversarial examples faster.

Despite all the techniques that are developed for increasing the robustness and performance of models it is unavoidable that there needs to exist runtime error detection to guarantee that there are no mistakes done by machine learning models. Since our approach is about detecting errors at runtime by using monitors it is important to detail what has been done in this area. The main approach for runtime error detection is **selective prediction** also known as model with **reject option** [31] [32], which has the goal of limiting the number of errors made by a model by not making a prediction when is a doubtful one, which is what we are achieving through the enforcement of fail silent behaviour. The main methods used for runtime error detection are the following according to [30]:

1) **Prediction Uncertainty**
Prediction uncertainty reports the confidence of the prediction made by the model. For example, for deep learning neural networks it is used the softmax probability of the predicted class to get this uncertainty value. This can be problematic due to the overconfidence of the model. To combat overly confident predictions of the models it is used model calibration, which consists of designing training methods so that the softmax matches the probability of a correct prediction. Another solution is uncertainty quantification

that aims to design prediction confidence measure for Machine Learning models.

2) **Out-of-Distribution Detection**
Out-of-Distribution (OOD) pertains to points that are outliers or that are not covered by the training of the model. The detection is a binary classification task to classify the ones that are covered by the training and the ones that are not. Some of the techniques for the detection are distance-based techniques, which measure the distance between the input and the training set, classification-based detection, which tries to encode normality and OOD detection scores, and density-based detection, which creates a probability density function from the source distribution to detect OOD.

3) **Adversarial Detection and Guards**
Adversarial Detection and guards have the goal of guaranteeing correct predictions of the models even in the case that there has been tampering with the input. One of the simplest solutions to detect adversarial attacks is to have a second model that is trained to be an adversarial attack detector, there is also other alternatives such as, statistical testing that compares the distribution of adversarial examples and clean examples, test-time adversarial guard which aims to predict correctly both the adversarial and normal inputs with the use of pre-processing, or by applying transformation and randomness.

One of the most used approaches falls into the **prediction uncertainty** method and is to have a reject option in a model, which means that if the probability of the prediction being correct is lower than a given threshold it abstains itself from making the prediction. There has been experiments realized to apply this behaviour in Deep neural networks by [33] and have them embedded in SVM by [34], both these experiments have been rather successful. One problem to this approach is the high confidence of prediction by NN even when the sample is not part of the training set, [35] tried to mitigate this problem for ReLu NN by proposing a robust optimization scheme through adversarial training, however the results proved that this problem of overconfidence cannot be avoid.

Chapter 4

Approach

The goal of this thesis is to present a monitor to guarantee safety for Machine Learning models that are used in critical applications. Given that our goal is to detect when the model makes an incorrect prediction, we decided to implement a monitor that will run in parallel to the model that will take in the same input of the model and will suppress the output of the model if the model's value is not trustworthy, if it is trustworthy the output will be normal, as can be seen in the following figure

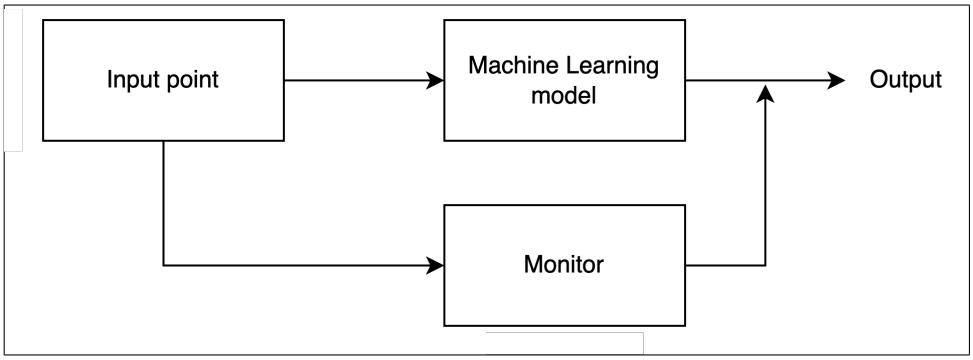


Figure 4.1: Monitor of machine learning model

Since there is a multitude of ways to create a monitor, we are going to propose three different approaches for its creation. Before going over the monitors that are proposed there are some aspects that are common to all the different approaches. The performance of the monitors will be tested for two different model algorithms, Support Vector Machine and Neural Network (NN).

4.1 Implementation and Specification

For the development of this work, we started by using python 3.10 since we wanted to use the scikit-learn library which allows us the creation of machine learning models and the creation of the ROC curves alongside matplotlib. We also used jupyter notebook which we had inside a virtual environment, where

we had all the libraries needed for the work. Some of the more important ones are numpy, mne for the drowsiness dataset, kneed for the clustering monitor, pandas, and XGBoost.

4.2 Monitor using Machine Learning

For this monitor, the idea is to use Machine Learning to create a monitor that learns to evaluate the model that is being used in a critical environment.

To achieve this approach, we are going to use the same features that are used in the critical model but instead of using the labeled values , we are going to use the difference between the prediction of the model and the real value. This will allow us to have a monitor that knows were the critical model is most likely to make wrong classifications.

For our monitor we decided to test three different algorithms, those being, Support Vector Machine, Neural Network, and Extreme Gradient Boosting (XGB).

To get the labels for the monitor we will need to compare the predictions of the model and the real value. Then we will assign zero to the classifications that the model made correctly and assign one to the classifications that the model made incorrectly, as can be seen in Figure 4.2. Due to this, we need to divide the dataset in three parts, one will be used to train the model, another will be to train the monitor and the last one will be to test both the model and the monitor. The split that was used was 40/40/20. Despite the fact that this will limit the data available to train the model it is necessary to train the monitor, because we need to make predictions with the model so that we can create the labels for the monitor, and for this we need to have data the model has not seen.

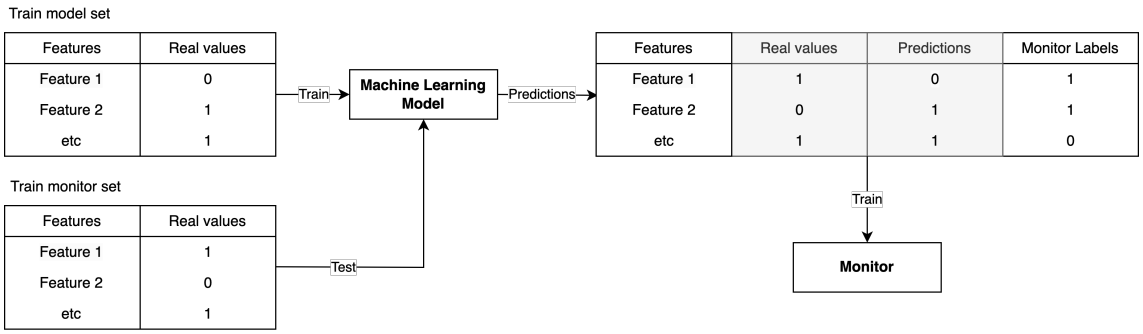


Figure 4.2: Machine learning monitor scheme

After the creation of the monitor, we will need to test different thresholds and get the TPR and FPR so that we can create the ROC curve and then calculate the AUC, these thresholds are going to be at which probability the monitor considers a prediction as positive.

4.3 Monitor using Clustering

The main idea of this monitor is to know if a given input is covered by the training dataset, the working of the monitor can be seen in Figure 4.4. This will allow us to know at runtime if the input is covered or not by the training dataset, if it is not then we cannot be sure that the prediction of the model is correct else we can assume that the prediction is valid. This technique is a distance-based technique where we check what are the points that are farther away from the **centroid**. Although we are going to be using labeled data in our test for this approach, we will ignore the labels when creating the clusters.

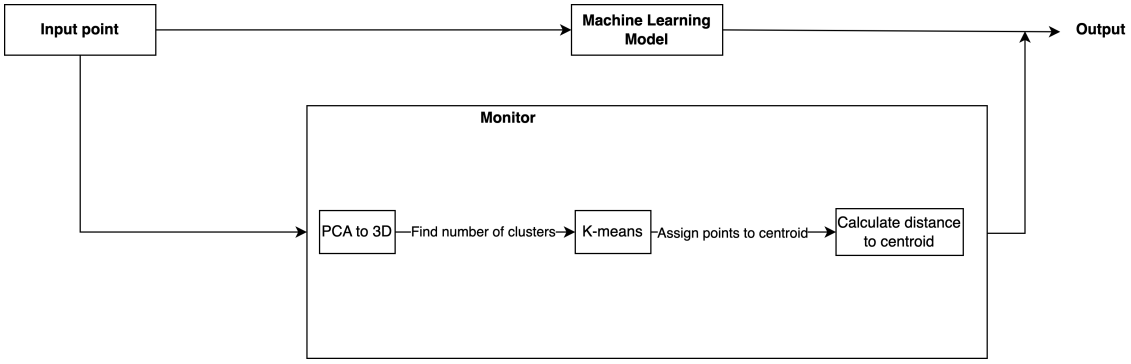


Figure 4.3: Monitor using clustering

After creating and training the model we use PCA to reduce the dimensions of our dataset to 3. We decided to use K-means as our algorithm to create the clusters, so we need to decide how many clusters we are going to have. To get this value we made use of the elbow method by using the kneed library which simplifies the method because it does not require the analysis of the graph to choose the optimal one. The next step is to create the different clusters and get their **centroids**. After this is done, we can calculate the euclidean distance of all training points to their respective centroid. Finally, we also calculate the distance of all the testing points to their respective centroid as can be seen the following figure.

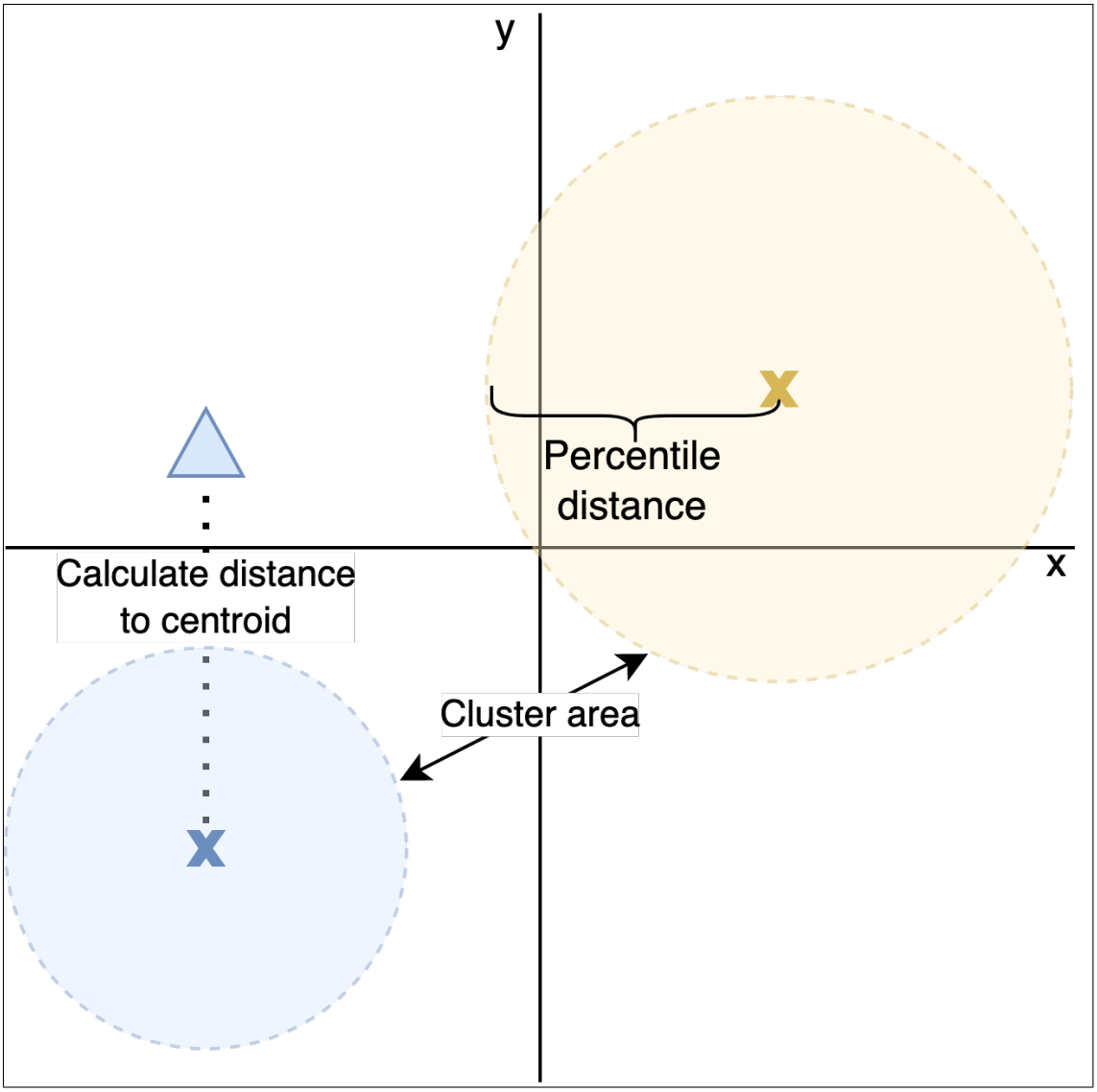


Figure 4.4: Monitor using clustering

Once we have the distance of all training points, we can calculate the distance of all the different percentiles from 0% to 99%. These distances will act as the threshold of this monitor. Any point that is further away from the centroid then the percentile distance is considered as a positive. Then we calculate the TPR and FPR for each percentile so that we can create the ROC curve and calculate the AUC.

4.4 Monitor using Classification Boundary

The goal of this monitor is to know if the input point is close to the classification boundary, which might correspond to a misclassification. This technique is also a distance-based technique and will act as a baseline for this thesis.

After creating and training the model we will once again use PCA, for this mon-

itor we will reduce the dimensional of the dataset to two, to facilitate the calculation to the classification boundary. Afterwards, we will need to check the minimum distance of each point to the decision boundary that makes the prediction changes its value. For example, if the original prediction is 1, what is the minimum distance it needs to be to become a 0. In order to make a prediction with the model we have to make the inverse of the pca to get back the features that we reduced. This can be seen in the following figure that shows how the monitor works.

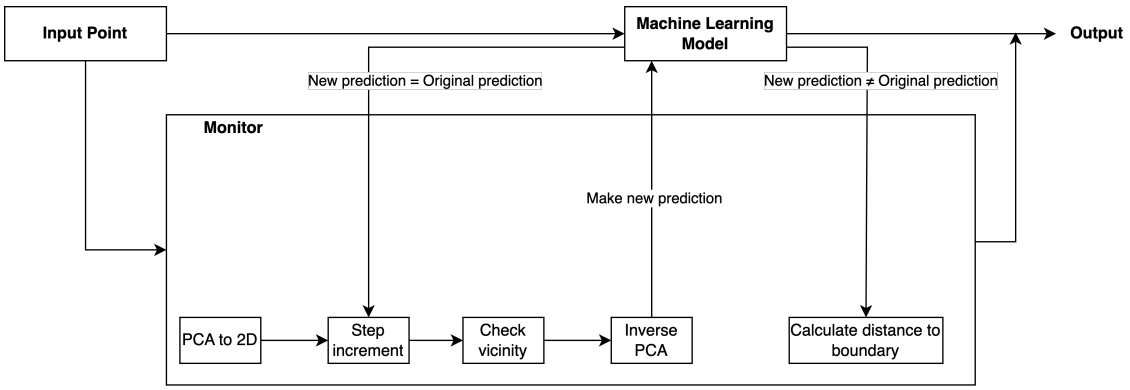


Figure 4.5: Classification boundary monitor scheme

For this approach we will calculate an approximation to the classification boundary. We start by adding a step of 0.1 at the x-axis, then if the classification boundary is not found we start by checking the vicinity from 0 to 2π with rotations of $\frac{\pi}{4}$ as can be seen in Figure 4.6, if we reach the x-axis once again and the boundary was not found we increment the step and do this all over again until finding the boundary. After finding the classification boundary for every point, we need to calculate all the euclidian distances to the classification boundary of all our training points so we can then calculate at which percentile a value can be considered doubtful. After the calculation of all the percentiles we get our thresholds.

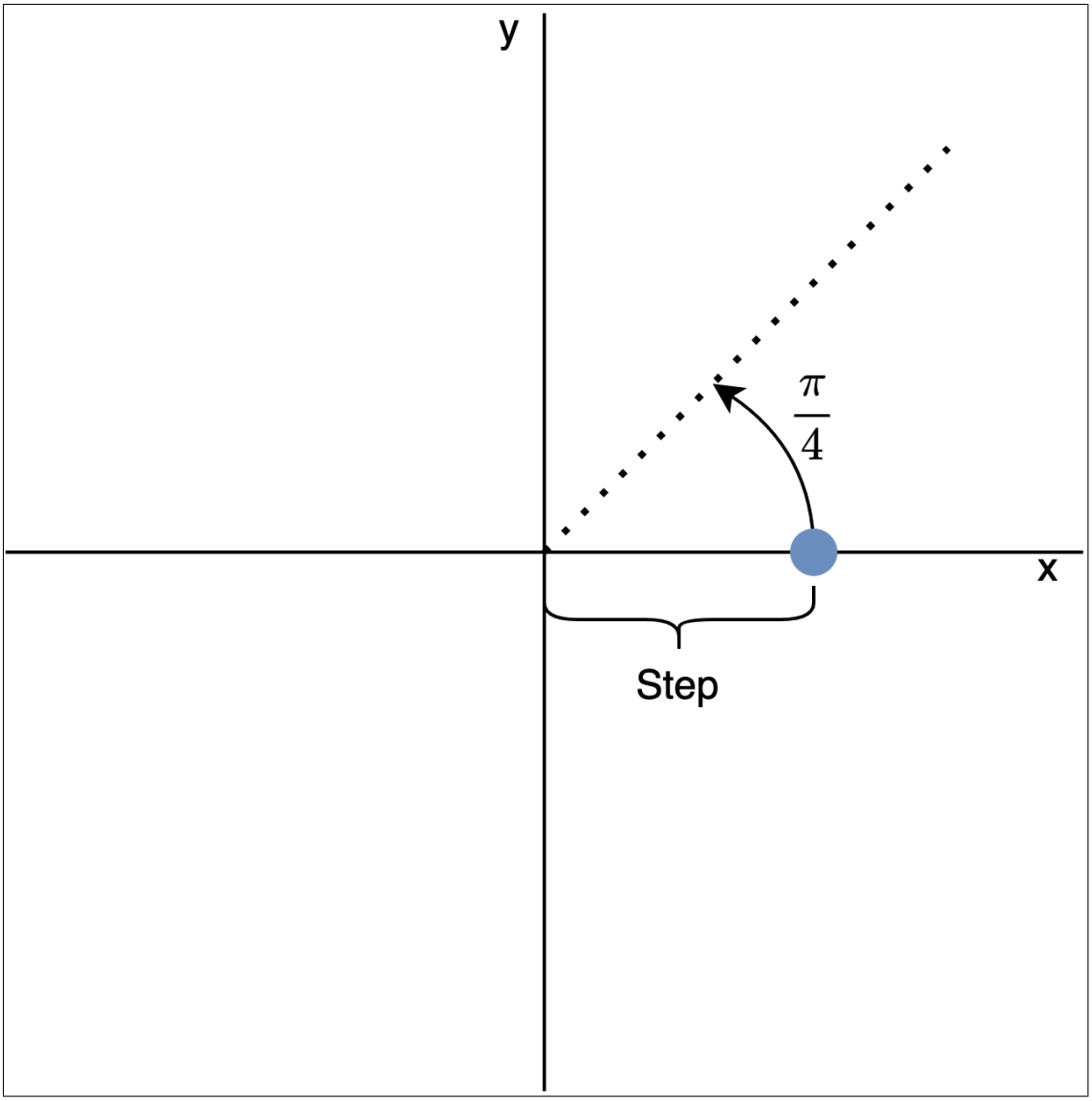


Figure 4.6: Scheme of the calculation of the Classification boundary

Following this, we also need to calculate the distance of the testing points to the decision boundary so that we can check if this distance is inside our percentile. If it is inside the percentile, which means closer to the boundary, we consider it dubious otherwise, it is considered correct. This is the opposite of what we did in the clustering monitor, the points inside the percentile are considered positives, being that a true positive is a case where our monitor flagged a point that the critical model has misclassified, and a False Positive is a point that our monitor flagged that the critical model has classified correctly. We then calculate the TPR and FPR for every percentile so that we can get the ROC curve and calculate the AUC to evaluate the performance of the monitor.

Chapter 5

Results and analysis

This chapter will show the results of all the monitors for all nine datasets and compare the performance between them. We will start by detailing the data that was used, then we explain the methodology used for the experiences, follow by the results such as the AUC and ROC curves, then the metrics we mentioned before and comparing the performance of the model with and without a monitor. To end the chapter, we discuss the results obtain that will give an answer to the research questions.

5.1 Data analysis

In this section we detail the various datasets that we used and report the accuracy for both the Support vector machine model and the neural network model, as can be seen in the following table.

Dataset	SVM model accuracy	NN model accuracy
Breast cancer	0.947	0.991
Drug classification	0.7	0.8
Orthopedic patients	0.806	0.823
Drowsiness detection	0.908	0.862
Liver disease	0.681	0.681
Heart failure	0.875	0.859
Diabetes detection	0.766	0.753
Weather prediction	0.741	0.795
Milk quality	0.92	0.967

Table 5.1: Accuracy of models across all datasets.

Most of the datasets used are from a healthcare environment with the exception of, the drowsiness detection which is the dataset that is being used by Cardioid to detect fatigue on drivers which is also a critical system, the weather prediction and also the milk quality prediction dataset are the ones that are not from a critical system. Most of the dataset are binary classification datasets, such as the breast

cancer, drowsiness detection, liver disease, heart failure and the diabetes detection dataset, the rest of the datasets have multiple classes. All the datasets, other than the drowsiness detection, which was obtain from the DROZY database, were obtain from Kaggle. It is worthy to mention that to extract the features from the drowsiness dataset we used a module that was provided by ISEP.

Having a closer look to the datasets used we have following enumeration that has the number of records in the dataset, the number of classes and their respective division and a general overview of the problem.

1. The **breast cancer** dataset [36] has 569 records (357 benign 212 malignant), it has two classes, so it is a binary classification problem those being benign and malignant, and the goal is to assign the tumor to one of the classes by using features that were obtain from an image of a breast mass.

2. The **drug classification** dataset [37] has 200 records (91 drugY 23 drugA 16 drugB 16 drugC 54 drugX), it is a multi-classification problem because it has five classes, one for each of the different type of drug, and the goal is to assign the correct drug to the patient by analyzing their blood results.

3. The **orthopedic patients** dataset [38] has 310 records (60 Hernia 100 Normal 150 Spondylolisthesis), it has three different classes them being the type of condition that the patients have, Spondylolisthesis, Normal or Hernia, and the goal is to classify the patient's condition through six biomechanical attributes.

4. The **drowsiness detection** dataset [39] has 323 records (134 awake 189 drowsy), it has two different classes, which are 0 when the subject is awake and 1 when he is drowsy, these values are obtain from the kss scale, which is a self-evaluation that was made by the subjects to evaluate their drowsiness, the scale goes from 1 to 9 and so we decided that any value above 5 is considered as drowsy and the rest are considered as awake. The goal is to see if the model can classify the state of the subject by analysing their ECG.

5. The **liver disease** dataset [40] has 579 records (414 negatives 165 positives), the goal is to detect if a patient has a liver disease or not, so it is a binary classification problem, and it achieves this classification by analyzing multiple chemical compounds present in the human body.

6. The **heart failure** dataset [41] has 918 records (410 negatives 508 positives), it has two classes which are 1 if the patient has a heart disease and 0 if it is normal, this is obtain with multiple features such as ecg, cholesterol and other values.

7. The **diabetes detection** dataset [42] has 768 records (500 negatives 268 positives), it has two classes that are 1 if the patient has diabetes and 0 if they do not, this has the goal to detect if a female has diabetes by analyzing the number of pregnancies, blood pressure and more.

8. The **weather prediction** dataset [43] has 1461 records (53 drizzle 101 fog 641 rain 26 snow 640 sun), and it has five different classes which represent the

different types of weather, drizzle, rain, sun, snow and fog, these classifications are based on the wind, the maximum temperature and other values.

9. The **milk quality** dataset [44] has 1059 records (256 high 429 low 374 medium), and it has three types of classes, which are the grade of the milk, it can be low, medium or high, this classification is based on observations such as pH, color, odor and others.

5.2 Experimental methodology

To make sure that the experimentation is fair we decided to use the same train-test spilt for every dataset and monitor to guarantee that the training data and the test data of the monitor is always the same. This cause us to divide the data into a split of 40/40/20, we need to divide it in three parts because of the machine learning monitors. The first 40 % is used for training the model, the second one is to train the monitor for the machine learning approaches, for the other approaches these 40% are ignored because they are not needed for training, the last 20% are for testing both the model and the monitor.

5.3 Monitors AUCs and ROC curves

In this section, we will present and analyse the ROC curves and respective AUC for all the different approaches and datasets. We will start by showing the AUC values followed by the ROC curves.

Monitor / Dataset	AUC of Boundary	AUC of Clustering	AUC of SVM	AUC of XGBoost	AUC of NN
Breast cancer	**0.977**	0.603	0.75	0.665	0.693
Drug classification	0.496	0.424	**0.708**	**0.708**	0.696
Orthopedic patients	0.733	0.482	0.79	0.818	**0.883**
Drowsiness detection	0.685	0.381	0.816	0.757	**0.828**
Liver disease	0.0	0.457	0.688	**0.716**	0.699
Heart failure	0.768	0.714	0.747	0.664	**0.784**
Diabetes detection	**0.681**	0.385	0.541	0.58	0.601
Weather prediction	0.586	0.542	0.752	**0.816**	0.626
Milk quality	0.747	0.495	**0.957**	0.922	0.826
Mean	0.63	0.498	**0.75**	0.738	0.738

Table 5.2: Area under the curve (AUC) of all five monitors, for the SVM model, for all datasets.

As can be seen in the Table 5.2 the machine learning approaches are the ones that provide the best results for a Support Vector Machine (SVM) model, both on average and also because they are well ahead of the other approaches in all but 2 datasets, those being the breast cancer and the diabetes detection, which has the

classification boundary as the best one. The boundary monitor has a lot of variability when compared to the machine learning ones, so despite being the best for those two datasets it has a significantly worst average, this is also because it has an AUC of 0.0 for the liver disease dataset, which means the monitor never classified anything as a positive, so the TPR is zero, this happened because the model classified everything as a negative so there is no classification boundary, this can be viewed as a limitation of this approach. The clustering monitor is consistently the poorest performing monitor, for this model the mean is even worst then a random guess, because for some datasets the errors arise near to the centroids of clusters, which means that the indicator that we used that was the proximity to the training centroid is poor for some cases, and at its best it is only a reasonable indicator.

Monitor / Dataset	AUC of Boundary	AUC of Clustering	AUC of SVM	AUC of XGBoost	AUC of NN
Breast cancer	**0.987**	0.854	0.389	0.876	0.301
Drug classification	0.582	0.309	0.625	**0.797**	0.637
Orthopedic patients	0.694	0.447	0.802	0.779	**0.884**
Drowsiness detection	0.614	0.741	**0.831**	0.722	0.645
Liver disease	0.67	0.464	0.693	0.674	**0.711**
Heart failure	0.775	0.618	0.771	0.754	**0.822**
Diabetes detection	**0.694**	0.423	0.594	0.626	0.659
Weather prediction	0.657	0.575	0.701	**0.791**	0.535
Milk quality	0.829	0.561	**0.957**	0.939	0.815
Mean	0.722	0.555	0.707	**0.773**	0.668

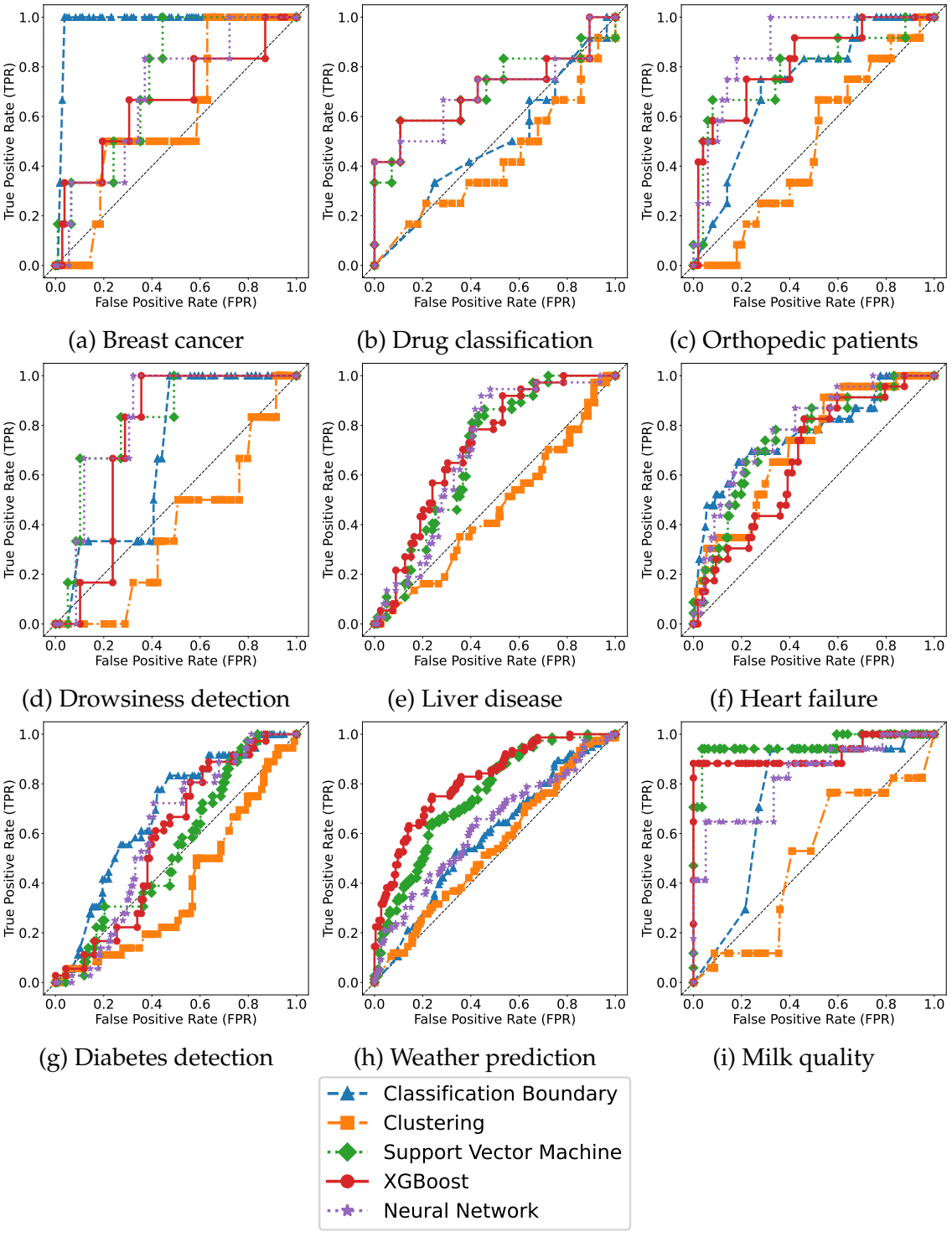
Table 5.3: Area under the curve (AUC) of all five monitors, for the NN model, for all datasets.

As can be seen in the Table 5.3 the Boundary, SVM and XGBoost are the ones that provide the best results for a NN model with fairly similar performance between 0.7 and 0.77. Both SVM and the Boundary monitor have a greater variability than XGBoost, there are two cases where the SVM monitor performs poorly and there is one where the Boundary performs poorly, though this is an improvement for the boundary monitor when compared to how it performed with the other model. The XGBoost monitor is still the most consistent one being the one that has the least variability, which explains its higher average. The NN monitor despite performing the best for more datasets than any other monitor it has one of the worst averages, this is because it has a high variability having a 0.301 and a 0.535 which negatively impacts its average performance. The clustering monitor, despite improving the performance for the NN model, still has poor performance when compared to the other ones.

In the Figure 5.1 we have the full picture to further help us understand the values of AUC that we had in the previous tables, as we can see the clustering does not work at all, with roc curve being lower then y = x (5.1b and 5.1d) which is the same as a random guess, this would suggest that perhaps the labels where switch but if we did that then we would lose the performance in the case where it performs reasonably like 5.1a, we also do not have any way of telling when they

should be inverted or not only after calculating the AUC can we know.

It is also visible that the green (SVM), red (XGB) and purple (NN) lines curves are constantly on the top which proves its high average in the Table 5.2. Although the blue line also is sometimes on top like for 5.1a, it has very negative performances like for the liver dataset 5.1e and also underperforms for the weather prediction dataset 5.1h and the drug classification dataset 5.1b.

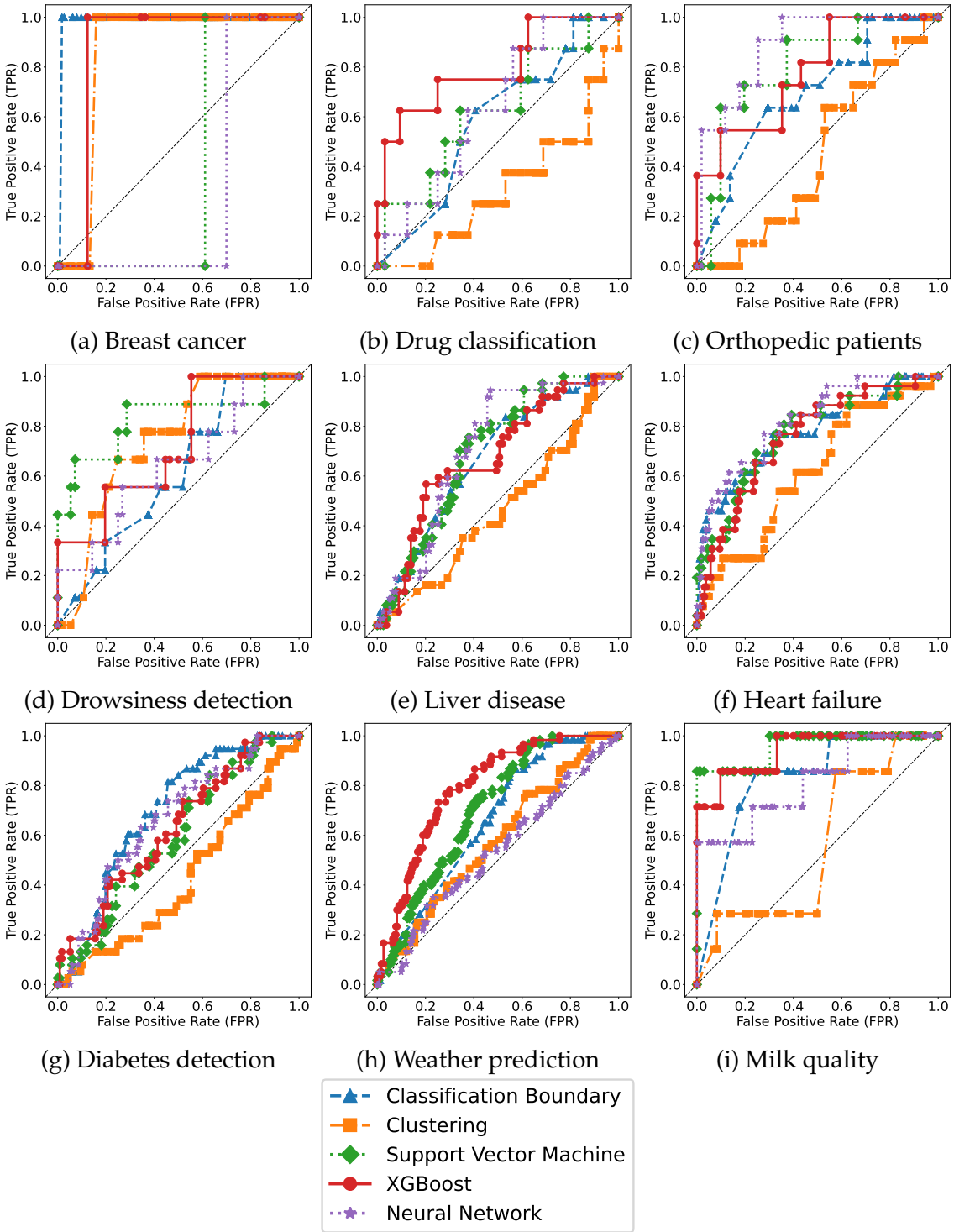


(a) Breast cancer (b) Drug classification (c) Orthopedic patients

(d) Drowsiness detection (e) Liver disease (f) Heart failure

(g) Diabetes detection (h) Weather prediction (i) Milk quality

Figure 5.1: ROC Curves of all five monitors, for the SVM model, for all datasets



(a) Breast cancer (b) Drug classification (c) Orthopedic patients

(d) Drowsiness detection (e) Liver disease (f) Heart failure

(g) Diabetes detection (h) Weather prediction (i) Milk quality

Figure 5.2: ROC Curves of all five monitors, for the NN model, for all datasets

Regarding the ROC curves for the NN model, Figure 5.2, we have a similar results, however there are some differences. We can see that there is more variability with the NN and SVM approaches, it is rather evident for the breast cancer dataset 5.2a. It is also notable that the consistency of the classification boundary is much better in this case being at its best almost the perfect classifier 5.2a and even at worst 5.2h is still similar to the other monitors. Despite the slight improvement of the clustering method it still is the worst one in almost all datasets

with the exception of the 5.2a and 5.2d.

5.4 Monitor metrics

In this section, we will show the metrics of our monitor for a particular configuration. We decided upon the configuration with the highest TPR while limiting the FPR to 15 % or less, this is what we though was reasonable for a real world scenario, of course this is dependent on the scenario and the urgency of catching these errors, if it is paramount to catch every single error than it is probably worth to increase the FPR which will lead to more mistakes caught by the monitor, at the cost of itself also making more mistakes, and so lowering the PPV of the monitor. We will show the accuracy, TPR or sensitivity, Positive Predictive value (PPV) of all the monitors and we will also compare the accuracy of the model with and without monitors.

We will start by showing the results of monitors for SVM models. Here we have a table for each metric, one for accuracy, one for sensitivity and another for PPV.

Regarding accuracy it is important to mention that it is heavily influenced by the accuracy of the model, since if the monitor does not find any error its accuracy is going to be the exact same as the model, with this mind it is not abnormal that the accuracy of the monitor is lower than the model, because to find errors the monitor makes some of his own, which means that the accuracy of the monitor will only be superior to the models if the PPV is greater than 0.5, this will be further explored once we look into the Table 5.6. As the accuracy of the monitor is that heavily correlated to the model it can even be seen as the accuracy of the model if the monitor inverted the prediction of all the points it caught.

Monitor / Dataset	Boundary Accuracy	Clustering Accuracy	SVM Accuracy	XGBoost Accuracy	NN Accuracy
Breast cancer	**0.961**	0.947	0.904	0.93	0.904
Drug classification	0.7	0.65	**0.8**	**0.8**	0.775
Orthopedic patients	0.758	0.806	**0.871**	0.855	0.839
Drowsiness detection	0.815	**0.908**	0.877	0.831	0.862
Liver disease	**0.681**	0.653	0.655	**0.681**	0.647
Heart failure	**0.821**	0.788	0.804	0.788	0.815
Diabetes detection	**0.721**	0.71	0.695	0.701	0.669
Weather prediction	0.693	0.676	0.744	**0.799**	0.724
Milk quality	0.92	0.792	0.962	**0.991**	0.925
Mean	0.786	0.77	0.812	**0.819**	0.795

Table 5.4: Accuracy of all five monitor, for SVM model, when FPR less or equal to 15%

In Table 5.4 we can see that in terms of accuracy the monitors are all rather close but with the SVM and XGBoost comfortably in front. When it comes to sensitivity the difference becomes more apparent, this being the metric that we want to see be the highest because it translates to the percentage of errors that the monitor

can detect. Ass can be seen in Table 5.5 all the machine learning monitors have similar performance with the best being SVM, detecting on average 49% of the errors made by the model. We can still see the same trend with the boundary monitor, that has excellent performance for the Breast cancer dataset with 1.0 of TPR but then there are datasets with TPR of 0.0. The clustering monitor does not even detect 10% of the errors made by the model, which is very poor, and clearly tells us that it has no ability of detecting errors.

Monitor / Dataset	Boundary Sensitivity	Clustering Sensitivity	SVM Sensitivity	XGBoost Sensitivity	NN Sensitivity
Breast cancer	**1.0**	0.0	0.333	0.333	0.333
Drug classification	0.0	0.167	**0.583**	**0.583**	0.5
Orthopedic patients	0.333	0.0	0.667	0.583	**0.75**
Drowsiness detection	0.333	0.0	**0.667**	0.167	**0.667**
Liver disease	0.0	0.051	0.216	**0.27**	0.189
Heart failure	**0.565**	0.348	0.478	0.304	**0.565**
Diabetes detection	**0.278**	0.053	0.139	0.111	0.056
Weather prediction	0.211	0.145	0.408	**0.632**	0.303
Milk quality	0.0	0.118	**0.941**	0.882	0.647
Mean	0.302	0.098	**0.492**	0.43	0.446

Table 5.5: Sensitivity of all five monitor, for SVM model, when FPR less or equal to 15%

The Table 5.6 is necessary for us to evaluate if the monitor is detecting errors because it has ability to do so or if it is sheer luck. This tables will tell us the likelihood of a positive made by the monitor is being correct.

Monitor / Dataset	Boundary PPV	Clustering PPV	SVM PPV	XGBoost PPV	NN PPV
Breast cancer	**0.6**	0	0.222	0.333	0.222
Drug classification	0	0.333	**0.7**	**0.7**	0.667
Orthopedic patients	0.364	0	**0.667**	0.636	0.562
Drowsiness detection	0.2	0	**0.4**	0.143	0.364
Liver disease	0	0.25	0.421	**0.5**	0.389
Heart failure	**0.361**	0.25	0.314	0.233	0.351
Diabetes detection	**0.37**	0.133	0.238	0.222	0.105
Weather prediction	0.348	0.268	0.508	**0.608**	0.451
Milk quality	0	0.065	0.696	**1.0**	0.524
Mean	0.249	0.144	0.463	**0.486**	0.404

Table 5.6: Positive Predictive value (PPV) of all five monitor, for SVM model, when FPR less or equal to 15%

From this table we can observe that the boundary and clustering monitor are not good because most of the positives are incorrect. The machine learning monitors are more interesting in that aspect because almost 50% of their positives were indeed an error made by the model, this despite not being the highest value be-

comes interesting when in conjunction with a recovery process. It is worth noting that the XGBoost is the best when it comes to Positive Predictive value (PPV).

Here we have the results for NN models, having once again three tables one for each metric.

Looking at Table 5.7, the results are very similar to the table of the other model. However, there are some differences and that is the fact that looking at mean for the accuracy we can observe that it increased overall for all monitors. As we mentioned before this accuracy is correlated to the accuracy of the model so it does not necessarily mean that the monitor is doing a better job for the NN model then the SVM model, it can be just that this model has better accuracy then the SVM model. Looking at accuracy the SVM monitor is the best closely followed by the XGBoost monitor.

Monitor / Dataset	Boundary Accuracy	Clustering Accuracy	SVM Accuracy	XGBoost Accuracy	NN Accuracy
Breast cancer	0.98	**0.991**	**0.991**	0.877	**0.991**
Drug classification	0.8	0.8	0.825	**0.85**	0.75
Orthopedic patients	0.774	0.823	**0.855**	0.839	0.839
Drowsiness detection	0.8	0.8	0.892	**0.908**	0.785
Liver disease	0.655	0.647	0.672	**0.698**	0.69
Heart failure	0.815	0.777	0.804	0.821	**0.832**
Diabetes detection	0.701	0.695	0.701	**0.76**	0.727
Weather prediction	0.751	0.724	0.747	**0.778**	0.72
Milk quality	0.967	0.84	**0.995**	0.901	0.986
Mean	0.805	0.788	**0.832**	0.826	0.813

Table 5.7: Accuracy of all five monitor, for NN model, when FPR less or equal to 15%

Regarding Table 5.8 there are also some notable differences and that is the fact that the XGBoost monitor performs significantly better when compare to the SVM and NN monitor, this is rather evident in the breast cancer dataset where the XGBoost monitor can detect all errors while the other two cannot detect a single one, and for the drug classification dataset where it has a TPR of 0.625 while the other two have only a TPR of 0.25, this leads to it having a much better average of detecting errors when compare to the others, managing to detect more than 50% of errors. Another reason for the difference in performance between XGBoost and the other two machine learning approaches is that the XGBoost performed better with the NN model whereas the other two performed worst. The clustering monitor had a small increase when looking at its mean while the boundary one had a small decrease when comparing to how they performed for the SVM model.

Dataset \ Monitor	Boundary Sensitivity	Clustering Sensitivity	SVM Sensitivity	XGBoost Sensitivity	NN Sensitivity
Breast cancer	**1.0**	0.0	0.0	**1.0**	0.0
Drug classification	0.0	0.0	0.25	**0.625**	0.25
Orthopedic patients	0.364	0.0	**0.636**	0.545	**0.636**
Drowsiness detection	0.111	0.444	**0.667**	0.333	0.333
Liver disease	0.216	0.054	0.27	**0.351**	0.189
Heart failure	0.538	0.269	0.423	0.385	**0.615**
Diabetes detection	0.079	0.079	0.158	0.184	**0.211**
Weather prediction	0.2	0.167	0.317	**0.483**	0.2
Milk quality	0.0	0.286	**0.857**	**0.857**	0.571
Mean	0.279	0.144	0.398	**0.529**	0.334

Table 5.8: Sensitivity of all five monitor, for NN model, when FPR less or equal to 15%

Dataset \ Monitor	Boundary PPV	Clustering PPV	SVM PPV	XGBoost PPV	NN PPV
Breast cancer	**0.333**	0.0	0	0.067	0.0
Drug classification	0.0	0.0	**0.667**	0.625	0.333
Orthopedic patients	0.364	0.0	**0.583**	0.545	0.538
Drowsiness detection	0.167	0.333	0.6	**1.0**	0.273
Liver disease	0.421	0.25	0.476	**0.542**	0.538
Heart failure	0.389	0.241	0.344	0.37	**0.432**
Diabetes detection	0.214	0.2	0.3	**0.538**	0.4
Weather prediction	0.324	0.244	0.365	**0.46**	0.261
Milk quality	0.0	0.065	**1.0**	0.231	**1.0**
Mean	0.246	0.148	0.482	**0.487**	0.42

Table 5.9: Positive Predictive value (PPV) of all five monitor, for NN model, when FPR less or equal to 15%

This table is very similar to the Table 5.6 we still have the boundary and clustering one performing the worst and the other three being better and rather close. The best monitor for this model is once again the XGBoost with an average of 0.487 which further proves the consistency of this approach.

We have two tables one for each model with all the metrics and all the monitors to make the comparison and analysis of data more compact and easier. These tables can be found in the appendix A.

For the Tables 5.10 and 5.11, we used the same configuration for the monitors that we used when we limited the FPR to 15%, this means that the configuration might not be the one that provides the best accuracy for a particular monitor. Since we are omitting the output of the monitors positives for the calculation of accuracy, we removed every positive caught by the monitor, that means that the true positives were subtracted to the incorrect model predictions and the false positives were subtracted to the correct model predictions, this way we are being fair by not counting the positives as neither good nor bad.

Monitor / Dataset	Original ACC	XGBoost ACC	SVM ACC	NN ACC	Boundary ACC	Clustering ACC
Breast cancer	0.947	0.963	0.962	0.947	**1.0**	0.947
Drug classification	0.7	**0.833**	**0.833**	0.652	0.7	0.706
Orthopedic patients	0.806	0.902	**0.92**	0.768	0.843	0.806
Drowsiness detection	0.908	0.914	0.964	**1.0**	0.927	0.908
Liver disease	0.681	**0.719**	0.701	0.716	0.681	0.676
Heart failure	0.875	0.896	0.919	0.921	**0.932**	0.901
Diabetes detection	0.766	0.765	0.767	0.776	**0.795**	0.755
Weather prediction	0.741	**0.869**	0.806	0.74	0.757	0.742
Milk quality	0.92	0.99	**0.995**	0.895	0.92	0.917
Mean differential	—	**+0.056**	**+0.056**	+0.008	+0.023	+0.002

Table 5.10: Accuracy comparison of SVM model with and without monitors across all datasets, when FPR less or equal to 15%

As we can see for the SVM model we increased the accuracy of the model for most datasets, with the ones that had the biggest increase being the drug classification one and the weather prediction with the XGBoost monitor. The values that are bold are the ones that have the best accuracy. There were some other cases that the monitor could not find any errors, so the accuracy stayed the same, and there were also a few cases that the accuracy was worse than the original one, this mostly happen with the clustering monitor. From this we can prove the trend we already saw in the other results that is that both the XGBoost and SVM monitor are the ones that prove improve the model the most, we can also see that there are some applications where the boundary monitor is the best one, however it is not as flexible as the other two.

Monitor / Dataset	Original ACC	XGBoost ACC	SVM ACC	NN ACC	Boundary ACC	Clustering ACC
Breast cancer	0.991	**1.0**	0.991	0.991	**1.0**	0.991
Drug classification	0.8	**0.906**	0.838	0.754	0.8	0.8
Orthopedic patients	0.823	0.902	**0.92**	0.829	0.863	0.823
Drowsiness detection	0.862	0.903	**0.945**	0.889	0.864	0.906
Liver disease	0.681	**0.739**	0.716	0.709	0.701	0.676
Heart failure	0.859	0.898	0.901	**0.932**	0.919	0.877
Diabetes detection	0.753	**0.78**	0.761	0.776	0.75	0.748
Weather prediction	0.795	**0.865**	0.83	0.792	0.812	0.802
Milk quality	0.967	**0.995**	**0.995**	0.949	0.967	0.972
Mean differential	—	**+0.051**	+0.041	+0.009	+0.016	+0.007

Table 5.11: Accuracy comparison of NN model with and without monitors across all datasets, when FPR less or equal to 15%

As shown in the Table 5.11 the accuracy increased with the addition of the XGBoost monitor for every dataset, the biggest increase in performance was once again in the drug classification dataset just like for the SVM model. The other machine learning monitors where able to improve the accuracy in some cases like for the drowsiness dataset and the heart failure, however they were not able to improve in a few datasets such as the breast cancer one. The boundary monitor despite having a case where he is the best there are some cases like for the diabetes dataset where it hinders the accuracy of the model. Even worst results can be observed when it comes to the clustering monitor because it rarely improves the accuracy and can actually reduce it. From all of this it becomes evident that NN monitor is clearly the worst one out of the machine learning approaches.

Both these tables have to be taken with a grain of salt because since we are removing all the positives that result from the monitor, we can be removing a lot of correct predictions from the model if we do not keep in mind the PPV. As can be seen when taking a look into the breast cancer, even though the XGBoost and boundary monitor reached the same accuracy the boundary monitor is much better for this approach as it has a greater PPV of 0.333 when compared to a PPV of 0.067 for the XGBoost.

5.5 Discussion and observations

Here we will discuss the results obtain, explain some of the observations made and emphasize their importance.

Starting with the observations we can see that there is a connection between the performance of the monitor and the performance of the model, the better the model the better the monitor is at detecting the errors made, this is especially evident when looking at the TPR for the breast cancer and milk quality dataset where some monitors were able to detect the majority of errors. The reverse also is applicable as the models that have the worst accuracy like the liver disease and diabetes dataset are the ones that have the lowest TPR across all the different monitors. There was no noticeable correlation between the problem being multiclass or binary and the performance the monitor.

We did not notice any relation between the performance of the monitor and the model it was monitoring, this becomes clear when looking into the impact of the monitors in the accuracy of the model, where the increase of accuracy is very similar for both the models and even when looking into the PPV Tables 5.6, 5.9 which also had very similar results. Where the values differ the most was in the accuracy Tables 5.4, 5.7, but as we mentioned this difference is more due to the difference in performance of the models then due to the difference in performance of the monitors.

The clustering monitor does not work it makes gross mistakes in the datasets where the errors appear near to the cluster centroid, which leads it to not being able to improve the model's accuracy for most datasets as can be seen in the Tables 5.10 and 5.11. From this we can discard this approach as one that should be used as it constantly underperformed. The same cannot be said about the other monitors, even the classification boundary that had some problems for certain datasets like the liver dataset, it had some where it was clearly the best. Which means that despite the XGBoost monitor being the most consistent one it still underperforms heavily for some cases like for the drowsiness detection dataset and diabetes, this proves that there are no silver bullets, so the decision of which monitor to use should still be analyzed for each particular model.

It is also worth noting the difference in performance between the monitors, the classification boundary one is the slowest one because it is a search-based approach which means it is not as fast as a single shot execution of a support vector machine.

Since we have monitors that can detect more than 50% of errors made by machine learning model while having a reasonable PPV we can say that our monitors can precisely detect incorrect predictions made by machine learning models, which answers **RQ1**. Regarding the **RQ2** we can say that out of the three different techniques used the best one to use for a monitor is the machine learning approach because it had the best averages in almost every category, regarding the best one out of the machine learning monitors the one that provided the best results was the XGBoost monitor, closely followed by the SVM monitor. The accuracy represents the probability that the model has of making an error occurring, since we were able to increase it with the use of monitor it answers **RQ3**, to a greater extent we can say that the safety of the model was increased by using the monitor since it made less errors. For the final research questions **RQ4**, we can say that the enforcement of fail-silent behavior is doable as it was able to increase the overall performance of the model, but a proper recovery solution can be arranged in or-

der to limit the number of incorrect detections of the monitor, which will increase the PPV of the monitors, and so improve the performance of the monitor itself.

Chapter 6

Conclusion

The increased use of machine learning models in critical systems, generates concerns over the trust of the models since it can have a big direct impact on people. To guarantee the existence of trustworthy AI there are wide range of requirements that have to be fulfilled like ethical, lawful and technical requirements.

Most the research being done by the AI community is done in this technical side of the requirements, however it is in a different area than the one we are tackling. Their efforts are more for enhancing of the performance of machine learning models because it is the biggest topic of the area as it involves new algorithms for models, distillation learning, ensemble learning, and other techniques.

The technical requirements revolve around safety and reliability, which are both a part of a bigger topic that is dependability. In our work, to achieve dependability we decided to add error detection, which is a part of fault tolerance, through the use of monitors.

We had the hypothesis that errors made by the model have patterns that allow for prediction of whether a machine learning model will provide a correct result or not.

To detect the errors made by the machine learning models we tested multiple approaches with the best one overall being the XGBoost one, although the SVM approach also provided decent results. The distance-based approaches were not as successful with the classification boundary monitor being very inconsistent, that is, it had brilliant results for some cases and was underwhelming for others, the clustering was bad all around as the distance to the training set proved that it did not have anything to do with the reason to why the model makes mistakes.

The results obtained were promising especially for the models that had better accuracy, this is even more significant when taking into account that our monitors are to be used mostly in critical environments where the performance of the models has to be high.

Most importantly we were able to improve the accuracy of the model with the use of monitors, which makes the model in itself safer and that was the main goal we had. Another noteworthy achievement in this work was the enforcement of fail-

silent behaviour on machine learning models which proved to be a decent but flawed technique.

Our work brings us one step closer to having trustworthy AI has we were able to add error detection to machine learning models, despite there being the possibility for improvements it was an important advancement in this area which was lacking in comparison with the rest of the field.

6.1 Future work

Since the best approach turned out to be the machine learning we can improve our monitors by using some of the enhancing performance techniques that are used to improve the performance of machine learning models.

We mainly focused in the area of error detection which means that for future work, we can explore better recovery techniques. Although the enforcement of fail-silent behaviour was a decent solution it can remove many correct predictions which is not ideal, with a proper way of recovering from the errors we might be able to limit the number of correct predictions suppressed, which will ultimately improve the performance of this technique and the use of monitors for error detection.

For the error recovery part, we can start by testing the possibilities of using ensembles, re-executing with a different input or having the input be executed in a different model that is tailored to specific cases.

Past that it would also be interesting to test the impact of the monitors for more complex models that are being used in the real world.

References

[1] Kalakota R Davenport T. The potential for artificial intelligence in healthcare. *Future Healthc J*, pages 94–98, 2019.

[2] Deloitte Insights. State of ai in the enterprise. 2018.

[3] Quackenbush J Schwartz LH Aerts HJWL Hosny A, Parmar C. Artificial intelligence in radiology. *Nat Rev Cancer*, pages 500–510, 2018.

[4] James Taylor Michael Ross. Managing ai decision-making tools, 2021.

[5] World Health Organization WHO. Ethics and governance of artificial intelligence for health. 2021.

[6] Yifang Ma, Zhenyu Wang, Hong Yang, and Lin Yang. Artificial intelligence applications in the development of autonomous vehicles: a survey. *IEEE/CAA Journal of Automatica Sinica*, 7(2):315–329, 2020.

[7] Giuseppe Lugano. Virtual assistants and self-driving cars. In *2017 15th International Conference on ITS Telecommunications (ITST)*, pages 1–5, 2017.

[8] High-Level Expert Group on Artificial Intelligence AI HLEC. Ethics guidelines for trustworthy ai. 2019.

[9] Alexey Tsymbal. The problem of concept drift: definitions and related work. *Computer Science Department, Trinity College Dublin*, 106(2):58, 2004.

[10] Xue Ying. An overview of overfitting and its solutions. In *Journal of physics: Conference series*, volume 1168, page 022022. IOP Publishing, 2019.

[11] Joe G Greener, Shaun M Kandathil, Lewis Moffat, and David T Jones. A guide to machine learning for biologists. *Nature Reviews Molecular Cell Biology*, 23(1):40–55, 2022.

[12] Shivani Gupta and Atul Gupta. Dealing with noise problem in machine learning data-sets: A systematic review. *Procedia Computer Science*, 161:466–474, 2019.

[13] Naveed Akhtar and Ajmal Mian. Threat of adversarial attacks on deep learning in computer vision: A survey. *Ieee Access*, 6:14410–14430, 2018.

[14] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning*. Cambridge University Press, 2014.

[15] Garreth James. *An Introduction to Statistical Learning*. Springer New York, 2013.

[16] Jerome Friedman Trevor Hastie, Robert Tibshirani. *The Elements of Statistical Learning*. Springer New York, 2009.

[17] Kristina P Sinaga and Miin-Shen Yang. Unsupervised k-means clustering algorithm. *IEEE access*, 8:80716–80727, 2020.

[18] MA Syakur, BK Khotimah, EMS Rochman, and Budi Dwi Satoto. Integration k-means clustering method and elbow method for identification of the best customer profile cluster. In *IOP conference series: materials science and engineering*, volume 336, page 012017. IOP Publishing, 2018.

[19] Ka Yee Yeung and Walter L. Ruzzo. Principal component analysis for clustering gene expression data. *Bioinformatics*, 17(9):763–774, 2001.

[20] Omer Sagi and Lior Rokach. Ensemble learning: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(4):e1249, 2018.

[21] Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129:1789–1819, 2021.

[22] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr. Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing*, 1(1):11–33, 2004.

[23] Beth A Schroeder. On-line monitoring: A tutorial. *Computer*, 28(6):72–78, 1995.

[24] D. Powell, G. Bonn, D. Seaton, P. Verissimo, and F. Waeselynck. The delta-4 approach to dependability in open distributed computing systems. In *[1988] The Eighteenth International Symposium on Fault-Tolerant Computing. Digest of Papers*, pages 246–251, 1988.

[25] S. Paredes, T. Rocha, P. de Carvalho, J. Henriques, M. Harris, and J. Morais. Long term cardiovascular risk models' combination. *Computer Methods and Programs in Biomedicine*, 101(3):231–242, 2011.

[26] Sasan Karamizadeh, Shahidan M. Abdullah, Mehran Halimi, Jafar Shayan, and Mohammad javad Rajabi. Advantage and drawback of support vector machine functionality. In *2014 International Conference on Computer, Communications, and Control Technology (I4CT)*, pages 63–65, 2014.

[27] Stephen Marsland. *Machine Learning An Algorithmic Perspective*. Chapman & Hall/CRC, 2014.

[28] Thomas F Monaghan, Syed N Rahman, Christina W Agudelo, Alan J Wein, Jason M Lazar, Karel Everaert, and Roger R Dmochowski. Foundational statistical principles in medical research: sensitivity, specificity, positive predictive value, and negative predictive value. *Medicina*, 57(5):503, 2021.

[29] Tom Fawcett. An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874, 2006.

[30] Sina Mohseni, Haotao Wang, Chaowei Xiao, Zhiding Yu, Zhangyang Wang, and Jay Yadawa. Taxonomy of machine learning safety: A survey and primer. *ACM Computing Surveys*, 55(8):1–38, 2022.

[31] Chi-Keung Chow. An optimum character recognition system using decision functions. *IRE Transactions on Electronic Computers*, (4):247–254, 1957.

[32] Ran El-Yaniv et al. On the foundations of noise-free selective classification. *Journal of Machine Learning Research*, 11(5), 2010.

[33] Yonatan Geifman and Ran El-Yaniv. Selective classification for deep neural networks. *Advances in neural information processing systems*, 30, 2017.

[34] Giorgio Fumera and Fabio Roli. Support vector machines with embedded reject option. In *Pattern Recognition with Support Vector Machines: First International Workshop, SVM 2002 Niagara Falls, Canada, August 10, 2002 Proceedings*, pages 68–82. Springer, 2002.

[35] Matthias Hein, Maksym Andriushchenko, and Julian Bitterwolf. Why relu networks yield high-confidence predictions far away from the training data and how to mitigate the problem. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 41–50, 2019.

[36] UCI Machine Learning. Breast cancer wisconsin (diagnostic) data set.

[37] Pratham Tripathi. Drug classification.

[38] UCI Machine Learning. Biomechanical features of orthopedic patients.

[39] Quentin Massoz, Thomas Langohr, Clémentine François, and Jacques G. Verly. The ulg multimodality drowsiness database (called drozy) and examples of use. In *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1–7, 2016.

[40] UCI Machine Learning. Indian liver patient records.

[41] Fedesoriano. Heart failure prediction dataset.

[42] Aman Chauhan. Predict diabities.

[43] Ananth R. Weather prediction.

[44] Shrijayan Rajendran. Milk quality prediction.

Appendices

Appendix A

Metric tables of all monitors

Monitor / Dataset	Boundary			Clustering			SVM			XGBoost			NN		
	Acc	TPR	PPV	Acc	TPR	PPV	Acc	TPR	PPV	Acc	TPR	PPV	Acc	TPR	PPV
Breast cancer	**0.961**	**1.0**	**0.6**	0.947	0.0	0	0.904	0.333	0.222	0.93	0.333	0.333	0.904	0.333	0.222
Drug classification	0.7	0.0	0	0.65	0.167	0.333	**0.8**	**0.583**	**0.7**	**0.8**	**0.583**	**0.7**	0.775	0.5	0.667
Orthopedic patients	0.758	0.333	0.364	0.806	0.0	0	**0.871**	0.667	**0.667**	0.855	0.583	0.636	0.839	**0.75**	0.562
Drowsiness detection	0.815	0.333	0.2	**0.908**	0.0	0	0.877	**0.667**	**0.4**	0.831	0.167	0.143	0.862	0.667	0.364
Liver disease	**0.681**	0.0	0	0.653	0.051	0.25	0.655	0.216	0.421	**0.681**	**0.27**	**0.5**	0.647	0.189	0.389
Heart failure	**0.821**	**0.565**	**0.361**	0.788	0.348	0.25	0.804	0.478	0.314	0.788	0.304	0.233	0.815	0.565	0.351
Diabetes detection	**0.721**	**0.278**	0.37	0.71	0.053	0.133	0.695	0.139	0.238	0.701	0.111	0.222	0.669	0.056	0.105
Weather prediction	0.693	0.211	0.348	0.676	0.145	0.268	0.744	0.408	0.508	**0.799**	**0.632**	**0.608**	0.724	0.303	0.451
Milk quality prediction	0.92	0.0	0	0.792	0.118	0.065	0.962	**0.941**	0.696	**0.991**	0.882	**1.0**	0.925	0.647	0.524
Mean	0.786	0.302	0.249	0.77	0.098	0.144	0.812	**0.492**	0.463	**0.819**	0.43	**0.486**	0.795	0.446	0.404

Table A.1: Metrics of all monitors, for SVM model, when FPR less or equal to 15%

Monitor / Dataset	Boundary			Clustering			SVM			XGBoost			NN		
	Acc	TPR	PPV	Acc	TPR	PPV	Acc	TPR	PPV	Acc	TPR	PPV	Acc	TPR	PPV
Breast cancer	0.98	**1.0**	**0.333**	**0.991**	0.0	0.0	**0.991**	0.0	0.0	0.877	**1.0**	0.067	**0.991**	0.0	0.0
Drug classification	0.8	0.0	0.0	0.8	0.0	0.0	0.825	0.25	**0.667**	**0.85**	**0.625**	0.625	0.75	0.25	0.333
Orthopedic patients	0.774	0.364	0.364	0.823	0.0	0.0	**0.855**	**0.636**	**0.583**	0.839	0.545	0.545	0.839	0.636	0.538
Drowsiness detection	0.8	0.111	0.167	0.8	0.444	0.333	0.892	**0.667**	0.6	**0.908**	0.333	**1.0**	0.785	0.333	0.273
Liver disease	0.655	0.216	0.421	0.647	0.054	0.25	0.672	0.27	0.476	**0.698**	**0.351**	**0.542**	0.69	0.189	0.538
Heart failure	0.815	0.538	0.389	0.777	0.269	0.241	0.804	0.423	0.344	0.821	0.385	0.37	**0.832**	**0.615**	**0.432**
Diabetes detection	0.701	0.079	0.214	0.695	0.079	0.2	0.701	0.158	0.3	**0.76**	0.184	**0.538**	0.727	**0.211**	0.4
Weather prediction	0.751	0.2	0.324	0.724	0.167	0.244	0.747	0.317	0.365	**0.778**	**0.483**	**0.46**	0.72	0.2	0.261
Milk quality prediction	0.967	0.0	0.0	0.84	0.286	0.065	**0.995**	**0.857**	**1.0**	0.901	**0.857**	0.231	0.986	0.571	**1.0**
Mean	0.805	0.279	0.246	0.788	0.144	0.148	**0.832**	0.398	0.482	0.826	**0.529**	**0.487**	0.813	0.334	0.42

Table A.2: Metrics of all five monitors, for NN model, when FPR is less than or equal to 15%

Appendix B

Scientific Paper

Here is the scientific paper that is being written alongside this thesis, this paper is still in preparation.

Machine Learning Monitors

Bernardo Carvalho, Raul Barbosa, and ..., *Life Fellow, IEEE*

Abstract—Artificial Intelligence (AI) is becoming more sophisticated and more widely used in our day-to-day lives, from virtual assistants like Alexa or Siri to self-driving cars. The increase in use and capabilities of AI leads to it being more frequently used in critical systems, creating the need to have some type of system that can detect malfunctions. Despite all the research done to improve the accuracy of AI, the problem of error detection is still yet to be resolved, so we will propose an approach to overcome this problem. Our approach is based on the use of monitors to supervise the critical model and omit the errors made by the model. We will have three different approaches for the monitor, from which will result five different monitors, one approach will be a Machine learning model to detect errors, another will calculate the distance to the classification boundary, and the last one will use clustering to know which points are covered by the training set. If this approach proves to be successful in detecting errors, it is expected that it starts to be used in critical systems so that it can improve Machine Learning (ML) safety.s

Index Terms—Monitoring, Critical systems, Fault tolerance, Error detection, Selective Prediction, Artificial Intelligence, Trustworthy AI, Machine Learning, ROC, True Positive Rate, False Positive Rate, Dependability

1 Introduction

AI is a technology that is expanding rapidly, and it is believed that it will have an immense role in the future of humanity and impact various industries, from the automotive industry to the healthcare industry. Both these industries have higher requirements of safety when compared to others because they generally are used in critical environments where a mistake might be deadly, which means that there is a need to have trustworthy AI. According to the European commission [1] there are three main components to achieve trustworthy AI, these components are that they should comply with all the applicable legislation and regulations, even though there is not a lot legislation in place for AI it will be developed as the technology advances, they should be **safe** and robust from a technical side which is what we are going to try to achieve in this thesis, lastly it also has the need to be ethical. To increase the safety of machine learning models we can improve the performance of the model, have inherently safe design and **runtime error detection**.

The idea of monitoring machine learning models comes from the need to detect errors made by the model, which can be achieve through fault tolerance. According to [2], fault tolerance is a mean to achieve dependability and it has the goal of avoiding service failures when a fault occurs. Fault tolerance is divided into two parts, error detection and recovery which we are not going to be undertaking in this paper. Error detection is also divided by two, concurrent detection which mean during runtime and preemptive detection which means detecting the error before it occurs. There are multiple ways to introduce fault tolerance to a system, we decided to have a monitor because it is a specialized support system of fault tolerance.

There is already some work that has been done by the AI community, by [3] and [4] which aims to limit the number of errors made by machine learning models by having models with a reject option also known as selective prediction which makes the model not make prediction when it is uncertain that it is correct.

2 Related work

Due to the increased use of AI, there has been an increase in research to make it safer and more dependable. In this chapter, we will go over what is dependability and how it can improve the safety of machine learning models, we will describe some of the causes of errors in machine learning and what strategies to increase the safety of machine learning models are being used.

2.1 Dependability

The original definition of **dependability** is the ability to deliver service that can justifiably be trusted [2]. There is also another definition that says a system is dependable when the system is capable of avoiding service failures that are more frequent and severe than is acceptable. A common way of putting it is that the system should be as dependable as the trust placed in that system. Since **dependability** is such a vast topic, it encloses availability, **reliability**, **safety**, integrity, and maintainability. For this work, the most significant ones are reliability and safety because they guarantee that the service operates correctly for a given time frame and the absence of unexpected consequences for both users and the environment, respectively. [2]

There are several ways of achieving **dependability**, such as fault prevention, **fault tolerance**, fault removal, and fault forecasting. Fault tolerance and fault prevention both strive to deliver a service that can be trusted, so both of them are needed to achieve trustworthy AI.

Fault tolerance aims to ensure that there are no service failures in the presence of faults, which is carried out by **error detection** and system recovery. Regarding the error detection part there are two different ways to detect errors, preemptive detection, which is detecting errors before an error occurs, and concurrent detection which is detecting errors while the system is operating. Since our goal is to have runtime error detection, we will be focusing on the latter. In order to have fault tolerance, we will use a

specialized support system for fault tolerance, such as a **monitor** [2].

Fail silent is according to [5] when the system only produces correct results, if an error happens to occur, the system does not output them. To enforce this behaviour in machine learning there is a need to detect the errors made by the model and then omit their output. This will be particularly valuable in a critical environment because it guarantees that every prediction of the model is correct, and the ones that the model did not produce output might have been wrong.

2.2 Causes of errors in machine learning

Even though this approach will not detect the root cause of the errors made by models it is worth knowing what might cause a model to make wrongful predictions. Some of the causes are described below:

- **Concept Drift**: This is a problem that can occur in real-time machine learning models, where there is hidden context that is not necessarily express through features. Changes in this hidden context have an impact in the ability of the model to make predictions [6], a good example for this can be a value prediction model for houses, even if the features have all the specification of a house the price of it is heavily dependent of what is the state of the market, which is impacted by inflation, new legislation, etc. There are two types of drift, it can be sudden or gradual.
- **Overfitting**: Is when a model does not generalize well from observed data to unseen data, this usually happens when the model memorizes all the data, including noise on the training set, instead of learning the patterns present in the data. This leads to the model performing perfectly on the training set and performing poorly on the testing set. [7]
- **Underfitting**: Is when a model fails to adequately capture the relationships between the variables in the data, it is the opposite of what happens in an overfitted model. The reasons for this happening might be the wrong choice of model, incomplete training [8] or due to data being imbalanced.
- **Data quality**: This is an overall problem that encapsulates many others, such as, missing features to represent the problem correctly, noise present in the data this might be attribute noise or class noise which can worsen the performance and efficiency of the model [9], not enough training data and class imbalance which might lead to underfitting.
- **Adversarial attacks**: They are small changes that have the goal of altering the predictions of models which can be problematic if done with malicious intent. There are a wide range of adversarial attacks like one pixel attack, fast gradient sign method or FGSSM, amongst others [10].

2.3 Strategies for safe machine learning models

According to [11], there are three different strategies to achieve safety in machine learning:

1) **Inherently Safe Design**, refers to techniques that are used to design ML models that are error-free. Some of the techniques that are used for this strategy are model transparency, which is one the requirements for trustworthy AI [1], design specification and model verification and testing, such as formal methods.
2) **Enhancing Performance and Robustness**, refers to techniques that improve the ML models performance, and their robustness, such as robust network architecture, robust training, and data sampling and augmentation. This is the strategy that is most commonly used, and it is an effective one because by creating ML models that make less mistakes inherently makes them safer.
3) **Runtime Error Detection**, refers to techniques that can detect errors made by the ML model at runtime. This can prevent misclassifications, which leads to a safer model. Some of the key approaches to detect these errors are model uncertainty estimation, adversarial attack detection, out-of-distribution detection and runtime monitoring.

Out of these three strategies the one that is developed the furthest is the enhancing performance and robustness. This strategy envelops almost all the current research that is being done in the field. This is also because it envelops a lot of parts of the area, from new algorithms that have more accuracy or are more efficient to adversarial training.

Despite all the techniques that are developed for increasing the robustness and performance of models it is unavoidable that there needs to exist runtime error detection to guarantee that there are no mistakes done by machine learning models. Since our approach is about detecting errors at runtime by using monitors it is important to detail what has been done in this area. The main approach for runtime error detection is **selective prediction** also known as model with **reject option** [3] [4], which has the goal of limiting the number of errors made by a model by not making a prediction when is a doubtful one, which is what we are achieving through the enforcement of fail silent behavior. The main methods used for runtime error detection are the following according to [11]:

1) **Prediction Uncertainty**
   Prediction uncertainty reports the confidence of the prediction made by the model. For example, for deep learning neural networks it is used the softmax probability of the predicted class to get this uncertainty value. This can be problematic due to the overconfidence of the model. To combat overly confident predictions of the models it is used model calibration, which consists of designing training methods so that the softmax matches the probability of a correct prediction. Another solution is uncertainty quantification that aims to design prediction confidence measure for ML models.
2) **Out-of-Distribution Detection (OOD)**
   OOD pertains to points that are outliers or that are not covered by the training of the model. The detection is a binary classification task to classify the ones that are covered by the training and the ones that are not. Some of the techniques for the detection are distance-based techniques, which measure the distance between the input and the training set, classification-based detection, which tries to encode normality and OOD detection scores, and density-based detection, which

creates a probability density function from the source distribution to detect OOD.

3) **Adversarial Detection and Guards**
Adversarial Detection and guards have the goal of guaranteeing correct predictions of the models even in the case that there has been tampering with the input. One of the simplest solutions to detect adversarial attacks is to have a second model that is trained to be an adversarial attack detector, there is also other alternatives such as, statistical testing that compares the distribution of adversarial examples and clean examples, test-time adversarial guard which aims to predict correctly both the adversarial and normal inputs with the use of pre-processing, or by applying transformation and randomness.

One of the most used approaches falls into the **prediction uncertainty** method and is to have a reject option in a model, which means that if the probability of the prediction being correct is lower than a given threshold it abstains itself from making the prediction. There has been experiments realized to apply this behaviour in Deep neural networks by [12] and have them embedded in SVM by [13], both these experiments have been rather successful. One problem to this approach is the high confidence of prediction by NN even when the sample is not part of the training set, [14] tried to mitigate this problem for ReLu NN by proposing a robust optimization scheme through adversarial training, however the results proved that this problem of overconfidence cannot be avoid.

3 METHOD

3.1 Clustering

This approach has the goal to calculate the test coverage of a given dataset and then check if the input points that the model receives are covered by the test coverage or not, this would allow us to detect if the prediction made by the model is valid or dubious at run time.

Once we have our model, we reduce the dimensions of the dataset to 3 using PCA. The clustering algorithm that we decided to use for our approach is K-means, which means that we need to decide how many clusters we will have. To get this value we made use of the elbow method through the use of the kneed library which simplifies the method because it does not require the observation of the graph to choose the optimal number of clusters. After this step we can then create the different clusters and calculate their **centroids**. Following this we calculate the euclidean distance of all the training points to their respective centroid, has can be seen in Figure 1.

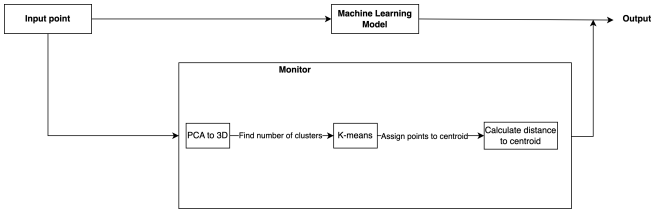


Fig. 1: Clustering monitor scheme

Using the training points distance, we can calculate the distance of all the different percentiles from 0% to 99%, these will act as our thresholds for the creation of the ROC curves that we will use to evaluate the performance of the monitors. Lastly, we also calculate the distance of all the testing points to their centroid, then we just need to check if they are inside or outside the percentile, points inside are covered by the test coverage and so are considered as negatives, and points outside are not covered are considered as positives.

3.2 Machine learning

For this monitor, the idea is to use machine learning to create a monitor that learns to evaluate the model that we desire.

To achieve this approach, we are going to use the same features that were used for the creation of the model except we will be changing the labeled values for the difference between the prediction of the model and the ground truth. This will allow us to have a monitor that is trained to detect when the model is likely to make wrong predictions. We will use two different learning algorithms for our monitor, those being Support Vector Machines or SVM and Extreme Gradient Boosting most commonly known as XGBoost.

To get the labels for the monitor we need to compare the predictions of the model and the ground truth, the ones that have the same value we assign the value of 0 and the ones that differ we assign the value of 1, as can be seen in figure 2. In order to do this, we need to divide the dataset in three parts, one will be used to train the model, another will be used to train the monitor and the other will be used to test both the model and the monitor. Since we need to divide the dataset in three parts, we decided that the best split is a 40/40/20 split. This will limit the data available, but it is necessary so that we can train the monitor, because we need to make predictions on unseen data with the model to create the labels for the monitor.

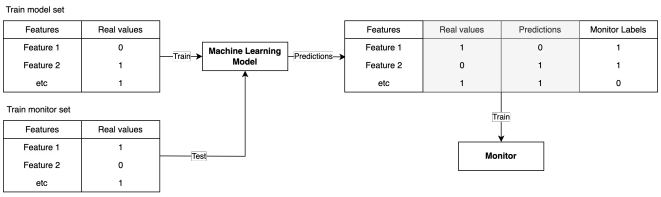


Fig. 2: Machine learning monitor scheme

To get the ROC curve we will need to test different thresholds so that we have multiple TPR and FPR, these thresholds are going to be at which value does the monitor consider a prediction as positive.

3.3 Distance to Classification Boundary

The goal of this monitor is to know if the input point is too close to the classification boundary, which might correspond to a misclassification. This technique is also a distance-based technique and will act as a baseline for this work.

After creating and training the model we will once again use PCA to facilitate the calculation to the classification boundary. Afterward, we will need to check the minimum distance of each point to the decision boundary that makes the prediction changes its value. For example, if the original

prediction is 1, what is the distance it needs to be to become a 0. In order to make a prediction with the model we have to make the inverse of the PCA to get back the features that we reduced, this can be seen in the Figure 3.

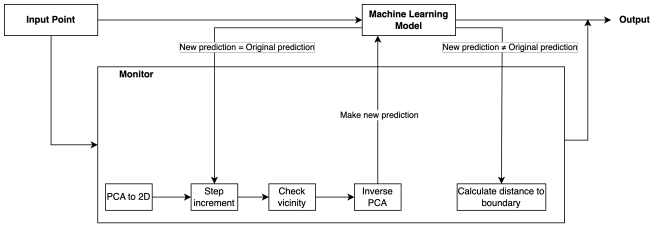


Fig. 3: Classification boundary monitor scheme

For this approach we will calculate an approximation to the classification boundary. We start by adding a step of 0.1 at the x-axis, then if the classification boundary is not found we start by checking the vicinity from 0 to 2π with rotations of $\frac{\pi}{4}$, if we reach the x-axis once again and the boundary was not found we increment the step and do this all over again until finding the boundary. After finding the classification boundary for each point, we then need to calculate all the euclidian distances to the classification boundary of all our training points so we can then calculate at which percentile a value can be considered doubtful. After the calculation of all the percentiles we get our thresholds. For this approach any point inside the percentile, which means closer to the boundary, is a positive.

3.4 Monitor Metrics

For evaluating our monitors, we will make use of some of the already established metrics in the AI community which, according to Stephen Marsland [15] and to [16], are:

- **TPR**, or **recall**, or **sensitivity**, is the ratio of the number of correct positives (TP) out of the ones classified as positive.
$$TPR = \frac{TP}{TP + FN}$$
- **FPR**, is the ratio of the number of incorrect negative classifications FP out of the ones that are classified as negative.
$$FPR = \frac{FP}{FP + TN}$$
- **Accuracy**, is the sum of correct classifications divided by all the classifications made, although it is not the best metric it is a simple metric that helps showing the performance of the monitor in a broader range.
$$Accuracy = \frac{TP + TN}{TP + FN + TN + FP}$$
- **Positive Predictive value**, or **PPV**, is the ratio of correct positives out of the ones that are classified as positive.
$$PPV = \frac{TP}{TP + FP}$$

The **ROC curve** is useful to compare the performance of the monitors against each other. We will use multiple thresholds for each monitor so that we have multiple points and so can create the curve. For each threshold we will calculate the TPR, which is the y-axis, and the FPR, which is the x-axis. After this step we can calculate the **AUC** which gives insight into the performance of the monitor. If the AUC is 0.5 it is as good as a random guess, if it is 1.0 then it is the perfect classifier and if it is 0.0 it is an anti-classifier.

4 Experimental evaluation

This chapter will show the results of all the monitors for all nine datasets and compare each monitors performance. We will start by detailing the data that was used, then we explain the methodology used for the experiences, follow by the results such as the AUC and ROC curves, then the metrics we mentioned before and comparing the performance of the model with and without a monitor.

4.1 Datasets

Here we have a table that shows the accuracy of the model of the various datasets that will be used for this paper.

Dataset	SVM model accuracy	NN model accuracy
Breast cancer	0.947	0.991
Drug classification	0.7	0.8
Orthopedic patients	0.806	0.823
Liver disease	0.681	0.681
Heart failure	0.875	0.859
Diabetes detection	0.766	0.753
Weather prediction	0.741	0.795
Milk quality	0.92	0.967
Risk stratify	0.894	0.91

TABLE 1: Accuracy of models across all datasets.

Most of these datasets are from a healthcare environment with the exception of, the weather prediction and also the milk quality dataset. All of the datasets were obtained from Kaggle, with the exception of the risk stratify. Detailing them a bit further we have:

1) The **breast cancer** dataset [17] has 569 records (357 benign 212 malignant), it has two classes, so it is a binary classification problem those being benign and malignant, and the goal is to assign the tumor to one of the classes by using features that were obtain from an image of a breast mass.
2) The **drug classification** dataset [18] has 200 records (91 druguY 23 drugA 16 drugB 16 drugC 54 drugX), it is a multi-classification problem because it has five classes, one for each of the different type of drug, and the goal is to assign the correct drug to the patient by analyzing their blood results.
3) The **orthopedic patients** dataset [19] has 310 records (60 Hernia 100 Normal 150 Spondylolisthesis), it has three different classes, them being the type of condition that the patients have, Spondylolisthesis, Normal or Hernia, and the goal is to classify the patient's through six biomechanical attributes.
4) The **liver disease** dataset [20] has 579 records (414 negatives 165 positives), the goal is to detect if a patient has a liver disease or not, so it is a binary classification problem, and it achieves this classification by analyzing multiple chemical compounds present in the human body.

5) The **heart failure** dataset [21] has 918 records (410 negatives 508 positives), it has two classes which are 1 if the patient has a heart disease and 0 if it is normal, this is obtain with multiple features such as ecg, cholesterol and other values.
6) The **diabetes detection** dataset [22] has 768 records (500 negatives 268 positives), it has two classes that are 1 if the patient has diabetes and 0 if they do not, this has the goal to detect if a female has diabetes by analyzing the number of pregnancies, blood pressure and more.
7) The **weather prediction** dataset [23] has 1461 records (53 drizzle 101 fog 641 rain 26 snow 640 sun), and it has five different classes which represent the different types of weather, drizzle, rain, sun, snow and fog, the classifications are based on the wind, the maximum temperature and other values.
8) The **milk quality** dataset [24] has 1059 records (256 high 429 low 374 medium), and it has three types of classes, which are the grade of the milk, it can be low, medium or high, this classification is based on observations such as pH, color, odor and others.
9) The **risk stratify** dataset has 2400 records (1148 0s 1252 1s), it has two different classes, which are 1 if the patient has Acute coronary syndrome or 0 if does not, this classification is based on observations such as age, heart rate, etc.

4.2 Experimental methodology

To make sure that the experimentation is fair we decided to use the same train-test spilt for every dataset and monitor to guarantee that the training data and the test data of the monitor is always the same. This cause us to divide the data into a split of 40/40/20, we need to divide it in three parts because of the machine learning monitors. The first 40 % is used for training the model, the second one is to train the monitor for the machine learning approaches, for the other approaches these 40% are ignored because they are not needed for training, the last 20% are for testing both the model and the monitor.

4.3 Results

In this section, it will be presented the results of all the different monitors for all the different datasets. We will start by showing their AUC, followed by their ROC curves, then we will have a table showing the monitors accuracy, TPR or sensitivity and PPV. We will have one final table that compares the performance of the model with and without the monitors.

As can be seen in Table 2 the machine learning approaches are the ones that provide the best results for a SVM model, both on average and also because they are well ahead of the other approaches in all but 3 datasets, those being the breast cancer, heart failure and the diabetes detection, which has the classification boundary as the best one. The boundary monitor has a lot of variability when compared to the machine learning ones, so despite being the best for those three datasets it has a significantly worst average, this also because it has an AUC of 0.0 for the liver disease dataset, which means the monitor never classified anything as a positive, so the TPR is zero, this happened because the model classified everything as a negative so there is no classification boundary, this can be viewed as a limitation of this approach. The clustering monitor is consistently the poorest performing monitor for this model, the mean is even worst then a random guess, because for some datasets the errors arise near to the centroids of clusters, which means that the indicator that we used that was the proximity to the training centroid is poor for some cases, and at its best it is only a reasonable indicator.

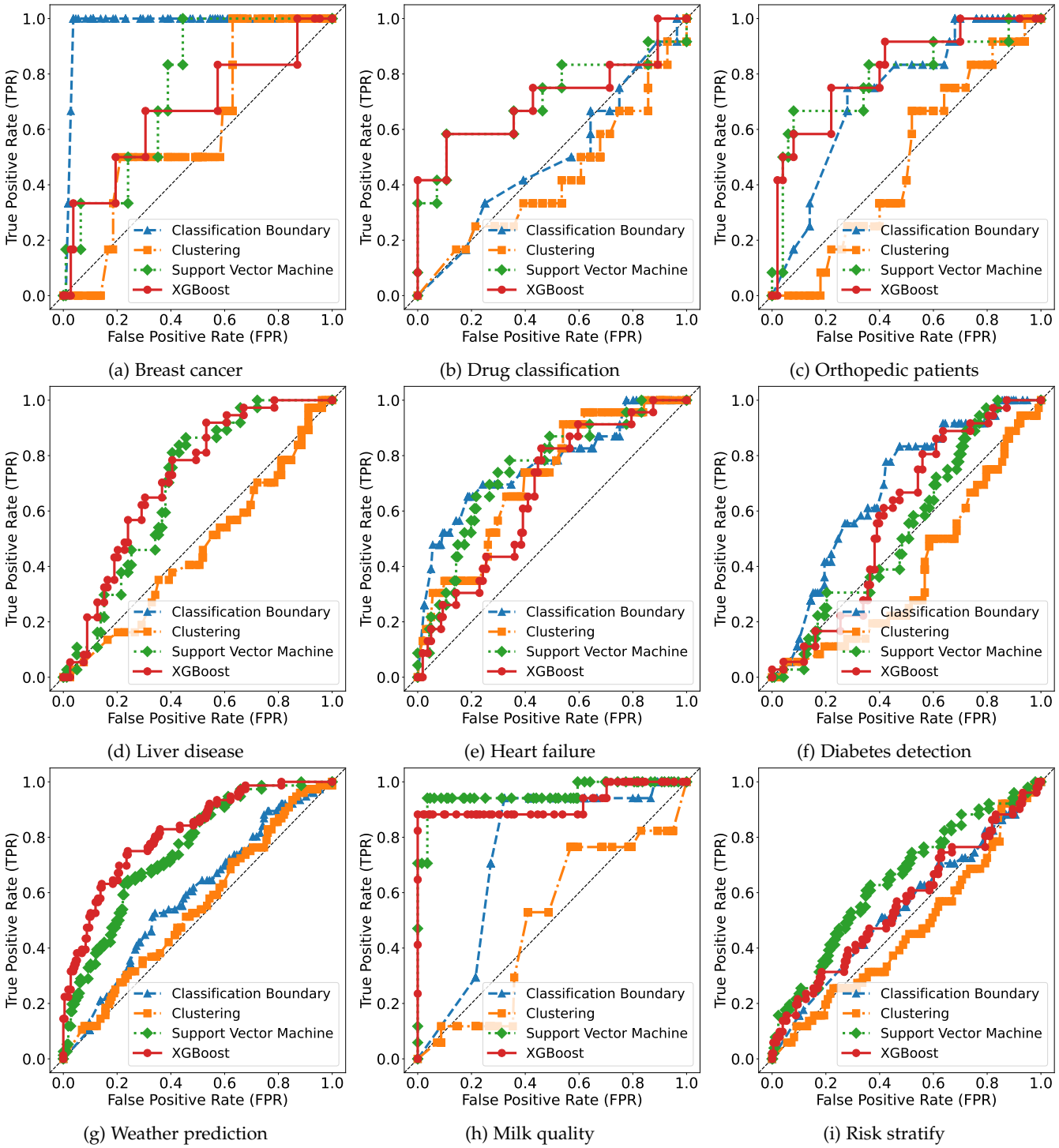
As can be seen in the Table 3 the Boundary, SVM and XGBoost are the ones that provide the best results for a NN model. Both SVM and the Boundary monitor have a greater variability than XGBoost, there are three cases where the SVM monitor performs poorly and there is two where the Boundary performs poorly, though this is an improvement for the boundary monitor when compared to how it performed with the other model. The XGBoost monitor is still the most consistent one being the one that has the least variability, which explains its higher average. It is also relevant to note that the difference of performance between the SVM and XGBoost monitors is significantly bigger when compared to the previous table. The clustering monitor, despite improving the performance for the NN model, still has poor performance when compared to the other ones.

In the Figure 4 we have the full picture to further help us understand the values of AUC that we had in the previous tables, as we can see the clustering does not work at all, with ROC curve being lower then y = x, for 4b and 4d, which is the same as a random guess, this would suggest that perhaps the labels where switch but if we did that then we would lose the performance in the case where it performs reasonably like 4a, we also do not have any way of telling when they should be inverted or not, only after calculating the AUC can we know if they should be inverted or not.

It is also visible that the green (SVM) and red (XGBoost) lines curves are constantly on the top which proves its high average in the Table 2. Although the blue (Boundary) line also is sometimes on top like for 4a, it has very negative performances like for the liver dataset 4d, where there is no line because it cannot detect a single error, and also underperforms for the weather prediction dataset 4h and the drug classification dataset 4b. The clustering is always below the monitors with the exception of the heart failure dataset.

Regarding the ROC curves for the NN model, Figure 5, we have a similar results, however there are some differences. We can see that there is more variability with the SVM approach, it is rather evident for the breast cancer dataset 5a. It is also notable that the consistency of the classification boundary is much better in this case being at its best almost the perfect classifier 5a and even at its worst 5i it is still similar to the other monitors. Despite the slight improvement of the clustering method, it still is the worst one in almost all datasets with the exception of the 5a and 5i.

Now taking a look to the metrics of our monitor for a particular configuration. We decided upon the configuration with the highest TPR while limiting the FPR to 15 % or less, this is what we though was reasonable for a real world scenario, of course this is dependent on the scenario and

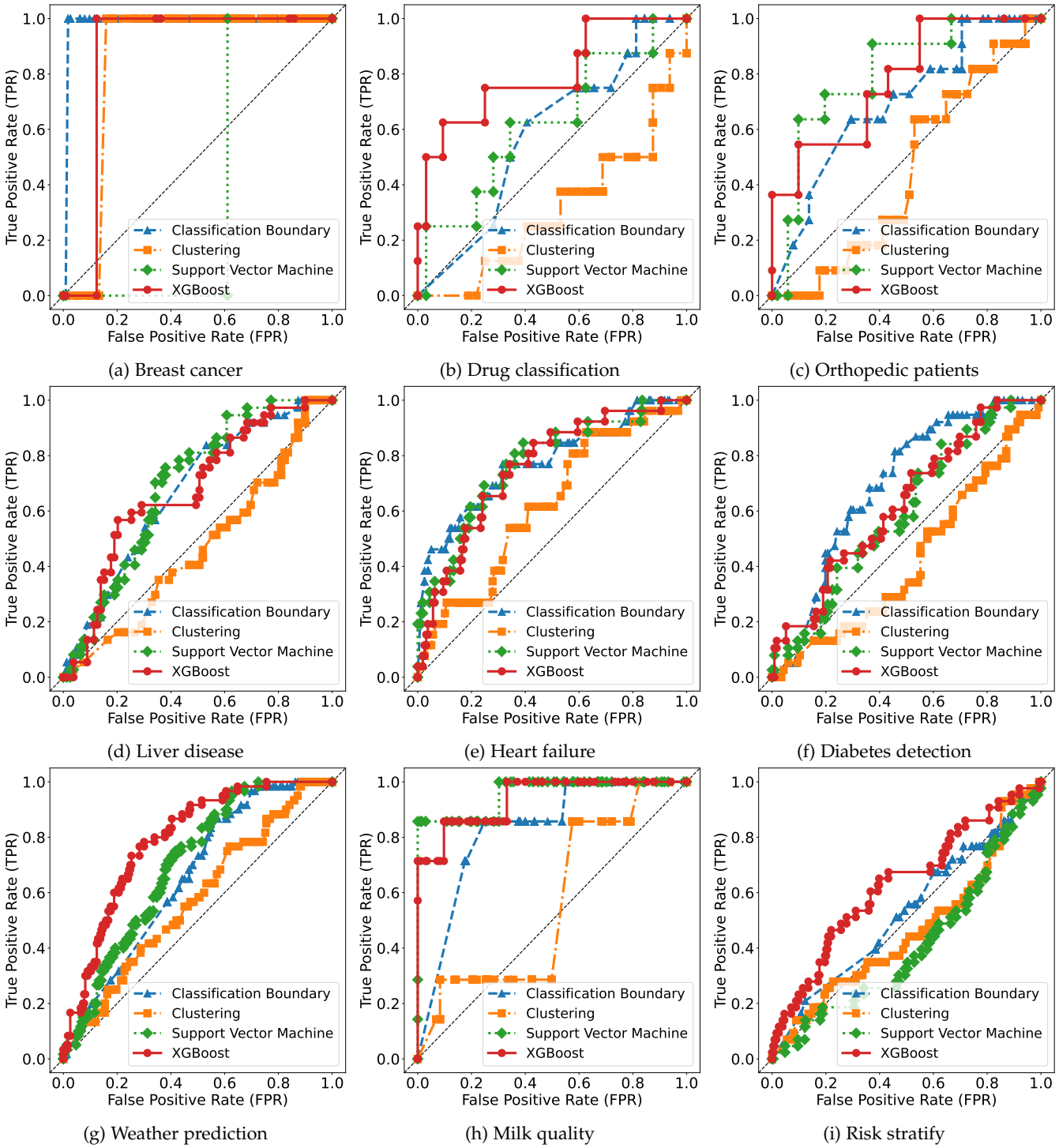


(a) Breast cancer (b) Drug classification (c) Orthopedic patients

(d) Liver disease (e) Heart failure (f) Diabetes detection

(g) Weather prediction (h) Milk quality (i) Risk stratify

Fig. 4: ROC Curves of all four monitors, for the SVM model, for all datasets



(a) Breast cancer (b) Drug classification (c) Orthopedic patients

(d) Liver disease (e) Heart failure (f) Diabetes detection

(g) Weather prediction (h) Milk quality (i) Risk stratify

Fig. 5: ROC Curves of all four monitors, for the NN model, for all datasets

Dataset	AUC of Boundary monitor	AUC of Clustering monitor	AUC of SVM monitor	AUC of XGBoost monitor
Breast cancer	**0.977**	0.603	0.75	0.665
Drug classification	0.496	0.424	**0.708**	**0.708**
Orthopedic patients	0.733	0.482	0.79	**0.818**
Liver disease	0.0	0.457	0.688	**0.716**
Heart failure	**0.768**	0.714	0.747	0.664
Diabetes detection	**0.681**	0.385	0.541	0.58
Weather prediction	0.586	0.542	0.752	**0.816**
Milk quality	0.747	0.495	**0.957**	0.922
Risk stratify	0.543	0.462	**0.645**	0.56
Mean	0.615	0.507	**0.731**	0.717

TABLE 2: Area under the curve (AUC) of all four monitors, for the SVM model, for all datasets.

Dataset	AUC of Boundary monitor	AUC of Clustering monitor	AUC of SVM monitor	AUC of XGBoost monitor
Breast cancer	**0.987**	0.854	0.389	0.876
Drug classification	0.582	0.309	0.625	**0.797**
Orthopedic patients	0.694	0.447	**0.802**	0.779
Liver disease	0.67	0.464	**0.693**	0.674
Heart failure	**0.775**	0.618	0.771	0.754
Diabetes detection	**0.694**	0.423	0.594	0.626
Weather prediction	0.657	0.575	0.701	**0.791**
Milk quality	0.829	0.561	**0.957**	0.939
Risk stratify	0.528	0.462	0.408	**0.631**
Mean	0.713	0.524	0.66	**0.763**

TABLE 3: Area under the curve (AUC) of all four monitors, for the NN model, for all datasets.

the urgency of catching these errors, if it is paramount to catch every single error than it is probably worth to increase the FPR which will lead to more mistakes caught by the monitor, at the cost of itself also making more mistakes, and so lowering the PPV of the monitor. We will show the accuracy, TPR or sensitivity, of all the monitors and we will also compare the accuracy of the model with and without monitors.

In both Tables 4 and 5 we have the accuracy, TPR or sensitivity, and Positive Predictive value (PPV) of all our monitors for each single dataset. Regarding accuracy it is important to mention that it is heavily influenced by the accuracy of the model, since if the monitor does not find any error its accuracy is going to be the exact same as the model, with this mind it is not abnormal that the accuracy of the monitor is lower than the model, because to find errors the monitor makes some of his own which means that the accuracy of the monitor will only be superior to the models if the PPV is greater than 0.5.

In the Table 4 we can see that in terms of accuracy the monitors are all rather close but with SVM and XGBoost comfortably in front. When it comes to sensitivity the difference become more apparent, this being the metric that we want to see be the highest because it translates to the percentage of errors that the monitor can detect. Both the machine learning monitors have similar performance with the best being SVM, detecting on average almost 45% of the errors made by the model. We can still see the same trend with the boundary monitor, that has excellent performance for the Breast cancer dataset with 1.0 of TPR but then there are datasets with TPR of 0.0. The clustering monitor can only detect around 11% of the errors made by the model, which is very poor, and clearly tells us that it has no ability of detecting errors.

It is necessary for us to have the PPV so we can evaluate if the monitor is detecting errors because it has ability to do so or if it is sheer luck. Looking at the PPV it becomes obvious that the boundary and clustering monitor are not good because most of the positives are incorrect. The machine learning monitors are a more interesting in that aspect because almost 50% of their positives were indeed an error made by the model in the case of the XGBoost monitor, this despite not being the highest value becomes interesting when in conjunction with a recovery process.

Regarding Table 5, the results are very similar to the previous table. However, there are some notable differences and that is the fact that the XGBoost monitor performs significantly better when compare to the SVM, this is rather evident in the breast cancer dataset where the XGBoost monitor can detect all errors and the SVM cannot detect a single one, and for the drug classification dataset where it has a TPR of 0.625 while the SVM only has a TPR of 0.25, this leads to it having a much better average of detecting errors when compare to the others, managing to detect more then 52% of errors. In regard to accuracy the SVM monitor is the best closely followed by the XGBoost monitor. The clustering monitor and classification boundary monitor have very similar performance when compared with the other model. The PPV for the NN model is slightly worst overall to the PPV for the SVM model, in particular for the XGBoost monitor, we still have the boundary and clustering one performing the worst and the other two being better and rather close. The best monitor when it comes to PPV is the SVM monitor with an average of 0.426.

For the Tables 6 and 7, we used the same configuration for the monitors that we used when we limited the FPR to 15%, this means that the configuration might not be the one that provides the best accuracy for a particular monitor. Since we are omitting the output of the monitors positives for the calculation of accuracy, we removed every positive caught by the monitor, that means that the true positives were subtracted to the incorrect model predictions and the false positives were subtracted to the correct model predictions, this way we are being fair by not counting the

Dataset \ Monitor	Boundary			Clustering			SVM			XGBoost		
	Acc	TPR	PPV	Acc	TPR	PPV	Acc	TPR	PPV	Acc	TPR	PPV
Breast cancer	**0.961**	**1.0**	**0.6**	0.947	0.0	0	0.904	0.333	0.222	0.93	0.333	0.333
Drug classification	0.7	0.0	0	0.65	0.167	0.333	**0.8**	**0.583**	**0.7**	**0.8**	**0.583**	**0.7**
Orthopedic patients	0.758	0.333	0.364	0.806	0.0	0	**0.871**	**0.667**	**0.667**	0.855	0.583	0.636
Liver disease	**0.681**	0.0	0	0.653	0.051	0.25	0.655	**0.216**	0.421	**0.681**	0.27	**0.5**
Heart failure	**0.821**	**0.565**	**0.361**	0.788	0.348	0.25	0.804	0.478	0.314	0.788	0.304	0.233
Diabetes detection	**0.721**	**0.278**	**0.37**	0.71	0.053	0.133	0.695	0.139	0.238	0.701	0.111	0.222
Weather prediction	0.693	0.211	0.348	0.676	0.145	0.268	0.744	0.408	0.508	**0.799**	**0.632**	**0.608**
Milk quality	0.92	0.0	0	0.792	0.118	0.065	0.962	**0.941**	0.696	**0.991**	0.882	**1.0**
Risk stratify	0.808	0.176	0.153	0.775	0.137	0.099	**0.825**	**0.255**	**0.22**	0.806	0.235	0.182
Mean	0.785	0.285	0.244	0.755	0.113	0.155	0.807	**0.447**	0.443	**0.817**	0.437	**0.491**

TABLE 4: Metrics of all monitors, for SVM model, when FPR less or equal to 15%

Dataset \ Monitor	Boundary			Clustering			SVM			XGBoost		
	Acc	TPR	PPV	Acc	TPR	PPV	Acc	TPR	PPV	Acc	TPR	PPV
Breast cancer	0.98	**1.0**	**0.333**	**0.991**	0.0	0	**0.991**	0.0	0	0.877	**1.0**	0.067
Drug classification	0.8	0.0	0	0.8	0.0	0	0.825	0.25	**0.667**	**0.85**	**0.625**	0.625
Orthopedic patients	0.774	0.364	0.364	0.823	0.0	0	**0.855**	**0.636**	**0.583**	0.839	0.545	0.545
Liver disease	0.655	0.216	0.421	0.647	0.054	0.25	0.672	0.27	0.476	**0.698**	**0.351**	**0.542**
Heart failure	0.815	**0.538**	**0.389**	0.777	0.269	0.241	0.804	0.423	0.344	**0.821**	0.385	0.37
Diabetes detection	0.701	0.079	0.214	0.695	0.079	0.2	0.701	0.158	0.3	**0.76**	**0.184**	**0.538**
Weather prediction	0.751	0.2	0.324	0.724	0.167	0.244	0.747	0.317	0.365	**0.778**	**0.483**	**0.46**
Milk quality	0.967	0.0	0	0.84	0.286	0.065	**0.995**	**0.857**	**1.0**	0.901	**0.857**	0.231
Risk stratify	**0.817**	0.209	0.143	0.792	0.163	0.099	0.81	0.14	0.1	0.812	**0.279**	**0.169**
Mean	0.807	0.29	0.243	0.787	0.113	0.122	**0.822**	0.339	**0.426**	0.815	**0.523**	0.394

TABLE 5: Metrics of all monitors, for NN model, when FPR less or equal to 15%

positives as neither good nor bad.

As we can see in Table 6 for the SVM model we increased the accuracy of the model for most datasets, with the ones that had the biggest increase being the drug classification and the weather prediction with the XGBoost monitor. The values that are bold are the ones that have the best accuracy. There were some other cases that the monitor could not find any errors, so the accuracy stayed the same, and there were also a few cases that the accuracy was worse than the original one, this mostly happen with the clustering monitor. From this we can prove the trend we already saw in the other results that is that both the XGBoost and SVM monitor are the ones that prove improve the model the most, we can also see that there are some applications where the boundary monitor is the best one, however it is not as flexible as the other two.

As shown in the Table 7 the accuracy increased with the addition of the XGBoost monitor for every dataset, the biggest increase in performance was once again in the drug classification dataset just like for the SVM model. The other machine learning monitors where able to improve the accuracy in some cases like for the drowsiness dataset and the heart failure, however they were not able to improve in a few datasets such as the breast cancer one. The boundary monitor despite having a case where he is the best there are some cases like for the diabetes dataset where it hinders the accuracy of the model. Even worst results can be observed when it comes to the clustering monitor because it rarely improves the accuracy and can actually reduce it.

Both these tables have to be taken with a grain of salt because since we are removing all the positives that result from the monitor, we can be removing a lot of correct predictions from the model if we do not keep in mind the PPV. As can be seen when taking a look into the breast cancer, even though the XGBoost and boundary monitor reached the same accuracy the boundary monitor is much better for this approach as it has a greater PPV of 0.333 when compared to a PPV of 0.067 for the XGBoost.

4.4 Discussion and observations

Starting with the observations we can see that there is a connection between the performance of the monitor and the performance of the model, the better the model the better the monitor is at detecting the errors made, this is especially evident when looking at the TPR for the breast cancer and milk quality dataset where some monitors were able to detect the majority of errors. The reverse also is applicable as the models that have the worst accuracy like the liver disease and diabetes dataset are the ones that have the lowest TPR across all the different monitors. There was no noticeable correlation between the problem being multiclass or binary and the performance the monitor.

We did not notice any relation between the performance of the monitor and the model it was monitoring, this becomes clear when looking into the impact of the monitors in the accuracy of the model, where the increase of accuracy is very similar for both the models and even when looking into the PPV that can be seen in the Tables 4, 5 they also had very similar results. Where the values differ the most was in the accuracy, but as we mentioned this difference is more due to the difference in performance of the models then due to the difference in performance of the monitors.

The clustering monitor does not work it makes gross mistakes in the datasets where the errors appear near to the cluster centroid, which leads it to not being able to improve the model's accuracy for most datasets as can be seen in the Tables 6 and 7. From this we can discard this approach as one that should be used as it constantly underperformed. The same cannot be said about the other monitors, even the classification boundary that had some problems for certain datasets like the liver dataset, it had some where it was

Dataset \ Monitor	Original Accuracy	XGBoost Accuracy	SVM Accuracy	Boundary Accuracy	Clustering Accuracy
Breast cancer	0.947	0.963	0.962	**1.0**	0.947
Drug classification	0.7	**0.833**	**0.833**	0.7	0.706
Orthopedic patients	0.806	0.902	**0.92**	0.843	0.806
Liver disease	0.681	**0.719**	0.701	0.681	0.676
Heart failure	0.875	0.896	0.919	**0.932**	0.901
Diabetes detection	0.766	0.765	0.767	**0.795**	0.755
Weather prediction	0.741	**0.869**	0.806	0.757	0.742
Milk quality	0.92	0.99	**0.995**	0.92	0.917
Risk stratify	0.894	0.906	**0.91**	0.9	0.892
Mean differential	—	**+0.057**	+0.054	+0.022	+0.002

TABLE 6: Accuracy comparison of SVM model with and without monitors across all datasets, when FPR less or equal to 15%

Dataset \ Monitor	Original Accuracy	XGBoost Accuracy	SVM Accuracy	Boundary Accuracy	Clustering Accuracy
Breast cancer	0.991	**1.0**	0.991	**1.0**	0.991
Drug classification	0.8	**0.906**	0.838	0.8	0.8
Orthopedic patients	0.823	**0.902**	0.92	0.863	0.823
Liver disease	0.681	**0.739**	0.716	0.701	0.676
Heart failure	0.859	0.898	0.901	**0.919**	0.877
Diabetes detection	0.753	**0.78**	0.761	0.75	0.748
Weather prediction	0.795	**0.865**	0.83	0.812	0.802
Milk quality	0.967	**0.995**	**0.995**	0.967	0.972
Risk stratify	0.91	**0.924**	0.912	0.918	0.912
Mean differential	—	**+0.048**	+0.032	+0.017	+0.003

TABLE 7: Accuracy comparison of NN model with and without monitors across all datasets, when FPR less or equal to 15%

clearly the best. Which means that despite the XGBoost monitor being the most consistent one it still underperforms heavily for some cases like for the drowsiness detection dataset and diabetes, this proves that there are no silver bullets, so the decision of which monitor to use should still be analyzed for each particular model.

It is also worth noting the difference in performance between the monitors, classification boundary one is the slowest one because it is a search-based approach which means it is not as fast as a single shot execution of a support vector machine.

5 Conclusion

The increased use of machine learning models in critical systems, generates concerns over the trust of the models since it can have a big direct impact on people. To guarantee the existence of trustworthy AI there are wide range of requirements that have to be fulfilled like ethical, lawful and technical requirements.

Most the research being done by the AI community is done in this technical side of the requirements, however it is in a different area than the one we are tackling. Their efforts are more for enhancing of the performance of machine learning models because it is the biggest topic of the area as it involves new algorithms for models, distillation learning, ensemble learning, and other techniques.

The technical requirements revolve around safety and reliability, which are both a part of a bigger topic that is dependability. In our work, to achieve dependability we decided to add error detection, which is a part of fault tolerance, through the use of monitors.

We had the hypothesis that errors made by the model have patterns that allow for prediction of whether a machine learning model will provide a correct result or not.

To detect the errors made by the machine learning models we tested multiple approaches with the best one overall being the XGBoost one, although the SVM monitor also provided decent results. The distance-based approaches were not as successful with the classification boundary monitor being very inconsistent, that is, it had brilliant results for some cases and was underwhelming for others, the clustering was bad all around as the distance to the training set proved that it did not have anything to do with the reason to why the model makes mistakes.

The results obtained were promising especially for the models that had better accuracy, this is even more significant when taking into account that our monitors are to be used mostly in critical environments where the performance of the models has to be high.

Most importantly we were able to improve the accuracy of the model with the use of monitors, which makes the model in itself safer and that was the main goal we had. Another noteworthy achievement in this work was the enforcement of fail-silent behaviour on machine learning models which proved to be a decent but flawed technique.

Our work brings us one step closer to having trustworthy AI has we were able to add error detection to machine learning models, despite there being the possibility for improvements it was an important advancement in this area which was lacking in comparison with the rest of the field.

5.1 Future work

Since the best approach turned out to be the machine learning we can improve our monitors by using some of the enhancing performance techniques that are used to improve the performance of machine learning models.

We mainly focused in the area of error detection which means that for future work, we can explore better recovery techniques. Although the enforcement of fail-silent behaviour was a decent solution it can remove many correct predictions which is not ideal, with a proper way of recovering from the errors we might be able to limit the number of correct predictions suppressed, which will ultimately improve the performance of this technique and the use of monitors for error detection.

For the error recovery part, we can start by testing the possibilities of using ensembles, re-executing with a different input or having the input be executed in a different model that is tailored to specific cases.

Past that it would also be interesting to test the impact of the monitors for more complex models that are being used in the real-world.

ACKNOWLEDGEMENT

This work is funded by national funds through the FCT - Foundation for Science and Technology, I.P., within the scope of the project CISUC - UID/CEC/00326/2020 and by European Social Fund, through the Regional Operational Program Centro 2020. The work is also supported by the FCT, Portugal Research Grant SFRH/BD/151322/2021.

This work is supported by Project Reference ECSEL/0017/2019 and 876852-ECSEL-RIA-VALU3S, financed by Fundação para a Ciência e a Tecnologia, I.P./MCTES through national funds (PIDDAC) and funding from the ECSEL Joint Undertaking (JU) under grant agreement No 876852. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Sweden, Italy, Spain, Portugal, Czech Republic, Germany, Austria, Ireland, France and Turkey.

REFERENCES

[1] H.-L. E. G. o. A. I. AI HLEC, "Ethics guidelines for trustworthy ai," 2019.

[2] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Transactions on Dependable and Secure Computing*, vol. 1, no. 1, pp. 11–33, 2004.

[3] C.-K. Chow, "An optimum character recognition system using decision functions," *IRE Transactions on Electronic Computers*, no. 4, pp. 247–254, 1957.

[4] R. El-Yaniv *et al.*, "On the foundations of noise-free selective classification." *Journal of Machine Learning Research*, vol. 11, no. 5, 2010.

[5] D. Powell, G. Bonn, D. Seaton, P. Verissimo, and F. Waeselynck, "The delta-4 approach to dependability in open distributed computing systems," in *[1988] The Eighteenth International Symposium on Fault-Tolerant Computing. Digest of Papers*, 1988, pp. 246–251.

[6] A. Tsymbal, "The problem of concept drift: definitions and related work," *Computer Science Department, Trinity College Dublin*, vol. 106, no. 2, p. 58, 2004.

[7] X. Ying, "An overview of overfitting and its solutions," in *Journal of physics: Conference series*, vol. 1168. IOP Publishing, 2019, p. 022022.

[8] J. G. Greener, S. M. Kandathil, L. Moffat, and D. T. Jones, "A guide to machine learning for biologists," *Nature Reviews Molecular Cell Biology*, vol. 23, no. 1, pp. 40–55, 2022.

[9] S. Gupta and A. Gupta, "Dealing with noise problem in machine learning data-sets: A systematic review," *Procedia Computer Science*, vol. 161, pp. 466–474, 2019.

[10] N. Akhtar and A. Mian, "Threat of adversarial attacks on deep learning in computer vision: A survey," *Ieee Access*, vol. 6, pp. 14 410–14 430, 2018.

[11] S. Mohseni, H. Wang, C. Xiao, Z. Yu, Z. Wang, and J. Yadawa, "Taxonomy of machine learning safety: A survey and primer," *ACM Computing Surveys*, vol. 55, no. 8, pp. 1–38, 2022.

[12] Y. Geifman and R. El-Yaniv, "Selective classification for deep neural networks," *Advances in neural information processing systems*, vol. 30, 2017.

[13] G. Fumera and F. Roli, "Support vector machines with embedded reject option," in *Pattern Recognition with Support Vector Machines: First International Workshop, SVM 2002 Niagara Falls, Canada, August 10, 2002 Proceedings*. Springer, 2002, pp. 68–82.

[14] M. Hein, M. Andriushchenko, and J. Bitterwolf, "Why relu networks yield high-confidence predictions far away from the training data and how to mitigate the problem," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 41–50.

[15] S. Marsland, *Machine Learning An Algorithmic Perspective*. Chapman & Hall/CRC, 2014.

[16] T. F. Monaghan, S. N. Rahman, C. W. Agudelo, A. J. Wein, J. M. Lazar, K. Everaert, and R. R. Dmochowski, "Foundational statistical principles in medical research: sensitivity, specificity, positive predictive value, and negative predictive value," *Medicina*, vol. 57, no. 5, p. 503, 2021.

[17] U. M. Learning. Breast cancer wisconsin (diagnostic) data set. [Online]. Available: https://www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data

[18] P. Tripathi. Drug classification. [Online]. Available: https://www.kaggle.com/datasets/prathamtripathi/drug-classification

[19] U. M. Learning. Biomechanical features of orthopedic patients. [Online]. Available: https://www.kaggle.com/datasets/uciml/biomechanical-features-of-orthopedic-patients

[20] ——. Indian liver patient records. [Online]. Available: https://www.kaggle.com/datasets/uciml/indian-liver-patient-records

[21] Fedesoriano. Heart failure prediction dataset. [Online]. Available: https://www.kaggle.com/datasets/fedesoriano/heart-failure-prediction

[22] A. Chauhan. Predict diabities. [Online]. Available: https://www.kaggle.com/datasets/whenamancodes/predict-diabities

[23] A. R. Weather prediction. [Online]. Available: https://www.kaggle.com/datasets/ananthr1/weather-prediction

[24] S. Rajendran. Milk quality prediction. [Online]. Available: https://www.kaggle.com/datasets/cpluzshrijayan/milkquality