PolyPublie
Polytechnique Montréal

POLYTECHNIQUE MONTRÉAL
UNIVERSITÉ D'INGÉNIERIE

Titre: Title:	Detecting Failures in Telecommuntications Manufacture for Predictive Quality
Auteur: Author:	Paulo Victor Queiroz Correia
Date:	2024
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Queiroz Correia, P. V. (2024). Detecting Failures in Telecommuntications Manufacture for Predictive Quality [Master's thesis, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/58307/

Document en libre accès dans PolyPublie
Open Access document in PolyPublie

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/58307/
Directeurs de recherche: Advisors:	Quentin Cappart
Programme: Program:	Génie informatique

Ce fichier a été téléchargé à partir de PolyPublie, le dépôt institutionnel de Polytechnique Montréal
This file has been downloaded from PolyPublie, the institutional repository of Polytechnique Montréal

https://publications.polymtl.ca

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Detecting Failures In Telecommuntications Manufacture For Predictive Quality

PAULO VICTOR QUEIROZ CORREIA
Département de génie informatique et génie logiciel

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
Génie informatique

Avril 2024

© Paulo Victor Queiroz Correia, 2024.

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Ce mémoire intitulé :

Detecting Failures In Telecommuntications Manufacture For Predictive Quality

présenté par **Paulo Victor QUEIROZ CORREIA**
en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
a été dûment accepté par le jury d'examen constitué de :

Daniel ALOISE, président
Quentin CAPPART, membre et directeur de recherche
Firouzeh GOLAGHAZADEH, membre externe

DEDICATION

To my grandparents, Valdemiro and Helena.
May their names, efforts, and love be eternized.

ACKNOWLEDGEMENTS

The present project is the fruit of a work of two years of a partnership with Ciena. Therefore, I would like to thank all the Ciena partners I had the pleasure of working with in the past two years: Firouzeh Golaghazadeh, Marc Beauregard, Arash Iranzad and Sadaf Arezoomand.

This project would not be possible without the financial support from MITACS, and I take this moment to acknowledge their impact on this project.

I would like to also thank my supervisor Quentin Cappart for all the support on the research and the decisions made throughout the graduation, including all the personal support.

I would also like to thank the Corail team members for all the support and ideas shared throughout the whole graduation.

I also thank Professor Daniel Aloise and his students, Andressa, Clara, Emerson and Thiago, for all the company, support and for receiving me for sharing ideas at their laboratory.

I also thank Fernanda and Isabelle for receiving me and supporting me throughout the last two years.

I could not finish this without acknowledging my family, specially my parents, Celdo and Kátia, and my brother, Celdo Victor, for all the shared moments on video calls and support given by distance. I truly love you all and am the person I am because of them.

Finally, I dedicate the merits of this work for the benefit of all sentient being. By the love and compassion of the three jewels, and Padmasambhava, Chenrezig and Manjushri, may all of them, find happiness and become free of suffering.

RÉSUMÉ

Internet a révolutionné notre façon de communiquer et d'interagir avec le monde. Il a introduit de nouvelles façons de consommer l'information et de partager des idées novatrices qui ont un impact mondial, rendant les informations restreintes accessibles à l'échelle mondiale. Cependant, cela nécessite une infrastructure énorme pour soutenir la communication qui traverse le monde entier et même l'espace. Des câbles sous-marins aux serveurs, routeurs et commutateurs, l'infrastructure doit respecter certaines assurances qualité pour garantir que l'information et les services atteignent tous les endroits du monde. Et, cette qualité traverse la fabrication de l'équipement composant cette énorme infrastructure.

Les entreprises de télécommunication, telles que Ciena, sont à la pointe de la production de tels équipements. Par conséquent, leur fabrication doit respecter des normes élevées de production tout en favorisant la sécurité et les économies matérielles et économiques. Et, pour répondre à ces contraintes, les entreprises réalisent continuellement des tests pour garantir que tous les composants de leurs produits répondent à ces normes, et qu'ils fournissent un bon produit au consommateur final. Par conséquent, nous abordons ces tests d'assurance qualité dans ce projet.

Ces tests sont effectués en continu pendant le processus de fabrication, mais certains d'entre eux sont cruciaux dans le processus de production. Et, cela se produit parce qu'ils sont effectués dans les dernières étapes de production, avant d'être expédiés aux consommateurs finaux, et parce qu'ils demandent des coûts matériels, personnels et énergétiques élevés. Ils sont les *Environmental Stress Tests* (EST), qui ont deux résultats possibles: réussite ou échec. Mais, avant ces tests, les fabricants réalisent des tests système, qui sont toujours importants, mais ont différents coûts que les EST. Par conséquent, dans ce projet, nous proposons d'utiliser les tests système moins coûteux pour prédire le résultat des EST en avance, dans le but d'optimiser la production de dispositifs de télécommunication.

Nous avons sélectionné les tests système comme caractéristiques parce que les opérations de fabrication supposent que ce sont les caractéristiques les plus corrélées aux EST. Les tests système évaluent les performances des composants individuels des produits en appliquant certaines contraintes aux composants, ce qui entraîne une réponse qui doit rester entre une limite d'acceptation pour être acceptée comme réussite. De plus, ils sont réalisés avant les EST sans aucune modification. Par conséquent, nous espérons répondre à la question de savoir si ces tests sont adaptés pour servir de caractéristiques pour détecter les échecs des EST.

Pour ce faire, nous proposons trois approches différentes de prétraitement des caractéristiques. La première consiste à utiliser les résultats bruts des tests système comme caractéristiques des classificateurs pour détecter les échecs. La deuxième et la troisième, cependant, considèrent les échecs des EST comme des anomalies et utilisent un réseau neuronal autoencodeur pour traiter les données. L'autoencodeur est entraîné uniquement avec les cas de test système associés à une réussite des EST, ce qui permet d'ingénier des caractéristiques qui mettent en évidence les différences entre les échecs et les réussites des EST. Par conséquent, nous utilisons la sortie de l'encodeur de l'autoencodeur appelée espace latent comme ensemble de caractéristiques, et l'erreur de reconstruction absolue des caractéristiques d'entrée du réseau autoencodeur. Nous essayons de répondre à la question de savoir si le traitement des échecs comme des anomalies améliorerait le taux de détection des échecs en comparant à une approche sans autoencodeurs. Mais, aussi, nous employons différentes techniques de rééchantillonnage pour aborder le déséquilibre des échecs des EST et sélectionner la méthode qui a la meilleure performance.

Ensuite, nous avons effectué une optimisation bayésienne pour sélectionner la meilleure configuration d'hyperparamètres. Et, après l'ajustement des hyperparamètres, nous avons maximisé la statistique de Youden J à partir de la courbe ROC pour sélectionner le seuil de classification afin d'améliorer la détection des échecs des modèles sélectionnés.

Enfin, après avoir employé toutes ces méthodes, nous avons remarqué que l'utilisation de l'espace latent de l'autoencodeur améliorait les résultats de classification lorsqu'elle était combinée avec l'analyse du seuil. L'erreur de reconstruction absolue n'a pas pu extraire des caractéristiques qui amélioraient les résultats. Cependant, en combinant les étapes de prétraitement proposées avec l'analyse du seuil, tous les modèles sélectionnés ont amélioré les scores F1 et les scores de précision négative. Ce dernier a atteint 90,7% sur les meilleurs modèles avec l'espace latent, avec un score de rappel de 64,4% sur l'ensemble de validation. En comparaison avec l'approche sans les autoencodeurs, notre méthodologie avec l'espace latent a obtenu de meilleurs résultats sur l'ensemble de tests, avec une marge d'amélioration reposant sur l'exploration et l'ingénierie des caractéristiques.

ABSTRACT

The internet revolutionized the way we communicate and interact with the word. It introduced new ways to consume information and to share novel ideas that impact the word, making restricted information accessible on a worldwide scale. However, it needs an enormous infrastructure to support the communication that crosses the whole world and even outer space. From cables underneath the ocean to servers, routers and switches, the infrastructure must attend certain quality assurances to guarantee that information and services reach every place worldwide. And that quality passes through the manufacture of the equipment composing this enormous infrastructure.

Telecommunication companies, such as Ciena, are spearheads in the production of such equipments. Therefore, their manufacture must attend high standards of production at the same time it promotes safety and material and economic savings. And to attend these constraints, the companies continuously perform tests to guarantee that all components of their products meet those standards, and they deliver a good product to the final consumer. Consequently, we tackle these quality assurance tests in this project.

These tests are performed continuously during the manufacture process, but some of them are crucial in the production pipeline. And this happens because they are performed in the final stages of production, before being shipped to the final consumers, and because they demand high material, personal and energetic costs. They are the *Environmental Stress Tests* (ESTs), which have two possible outcomes: pass or fail. But before these tests, manufactures perform system tests, which are still important but do not have the same costs as the ESTs. Therefore, in this project, we propose using the less costly system tests to predict the outcome of the ESTs beforehand, aiming to optimize the production of telecommunication devices.

We selected the system tests as the features because the manufacture operations speculate that these are the most correlated features to the ESTs. The system tests assess the performance of individual components of the products by applying certain stress to the components, which result in a response that must remain between an acceptance boundary to be accepted as a pass. Also, they are performed before the ESTs without any type of modification. Therefore, we expect to answer if those tests are suitable to serve as features to detect EST failures.

To achieve this, we propose three different approaches of preprocessing to the features. The first one consists of using plain system test case results as the features of the classifiers to detect failures. The second and the third, however, assume the EST failures as anomalies and

use an autoencoder neural network to process the data. The autoencoder is trained only with system test cases associated with an EST pass, making it possible to engineer features that highlight the differences between EST failures and pass. Therefore, we use the autoencoder's encoder output named latent space as a set of features, and the absolute reconstruction error of the input features of the autoencoder network. We try to answer if treating the failures as anomalies would improve the failure detection rate by comparing to an approach without autoencoders. But also, we employ different resampling techniques to tackle the imbalance of the EST failures and select the method that has the best performance.

Thereafter, we performed a Bayesian optimization to select the best configuration of hyper-parameters. And after the hyperparameter tuning, we maximized the Youden J's statistic from the ROC-Curve to select the classification threshold to improve the failure detection of the selected models.

Finally, after employing all these methods, we noticed that using the autoencoder's latent space improved the classification results when combined with the threshold analysis. The absolute reconstruction error could not extract features that improved the results. However, by combining the proposed preprocessing steps with the threshold analysis, all the selected models improved the F1-Scores and the negative precision scores. The latter achieved 90.7% on the best models with the latent space, with a recall score of 64.4% on the validation set. When compared to the approach without the autoencoders, our methodology with latent space achieved better results on the test set, with the margin for improvement lying on feature exploration and engineering.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ACRONYMS

EST	Environmental Stress Test
PMA	Product Module Assemble
SMOTE	Synthetic Minority Over-sampling Technique
ADASYN	Adaptive Synthetic algorithm
UMAP	Uniform Manifold Approximation and Projection
TNR	True Negative Rate
FPR	False Positive Rate

CHAPTER 1 INTRODUCTION

The advancements in telecommunication technologies have revolutionized the way we communicate within ourselves and with the environment. It allowed the development of the internet, which opened new possibilities on how we interact with the whole planet and allowing new opportunities for creating bonds and developing collaboratively new technologies and techniques. However, all this environment calls for a high-quality infrastructure that enables massive communication. As a result, the tools we employ to construct this environment require exceptional performance and resilience to support the massive exchange of data required to construct this communal and interactive environment.

The telecommunication equipment necessary to construct the internet includes switches, routers, and optical network devices. And their manufacture is an industrial multistaged process that requires a thorough quality assurance. The quality control of production is necessary to attend the demand of users for fast and secure communication around the world. Hence, manufactures need a quality assurance protocol that makes sure the products will present minimal failures to the final user.

Telecommunication companies, such as Ciena, define a precise production line of equipment to meet those demands. They commonly design a multilayered production pipeline that produces a different type of product at each step. Each of these layers adds different components to a unit, thus transforming it into a different product. The operators then must perform different types of tests to assure the quality of the plugged parts at that stage. Therefore, they aggregate the tests into different *Process Module Assembles* (PMA), which is a set of test cases applied to the unit to test specific parts of it. A test case then is the singular unit of a test: the operators apply stress to the unit, which has an associated response to this stress. Hence, the response of a test case must lie between an acceptance boundary to be accepted as a pass. And if the response is outside this boundary, we consider that the test case failed. Having that in mind, if a PMA has any of these test cases failing, we consider that the PMA outcome is a failure. Otherwise, the outcome is a pass and the operators can carry on the production of the unit.

During the last stages of production of a unit, before shipping it to the final consumer, the operators perform general tests associated to specific PMAs to assure that the unit is working in accordance with quality standards of production. Overall, they analyze the performance of the components of the equipments, such as the electrical and optical parts. We call this set of tests the System Tests PMA. Immediately after this PMA, the engineers from the company

perform the Environmental Stress Tests (EST) PMA, which assesses the performance of the unit under extreme communication scenarios. They perform an exhaustive byte package exchange among different units, simulating extreme conditions to analyze if the unit in all its aspects behave properly under these conditions. If this PMA fails, the operators perform repairs and revisions in accordance with the EST test cases that failed, and then submit the unit again to the system tests followed by the EST PMA.

Both system and EST PMAs are utterly important for the manufacture, specially because they are the last set of tests performed in the manufacture lifecycle. Also, most of the operators perform both of these tests multiple times, even if the unit does not present any failure in any of these PMAs. They also cost plenty of resources to be performed, as well as time to conclude. However, the EST PMA presents a more critical scenario because they take a significant more amount of time to complete than the system test PMAs. And if the operators run them multiple times, it would take a significant portion of the manufacture process. Therefore, preemptively predicting the outcome of the EST PMA will potentially reduce the manufacture costs in this stage of the production by giving the operators tools to analyze the most likely outcome of the EST outcome before running them multiple times. Basically, if the operators identify an EST failure with a trained predictive model before performing it, they would not have the necessity of running the EST test multiple times before being sure that they need to perform revisions and repairs to the unit, even though they would have to run the PMA a single time to confirm or reject the predicted outcome since it is a critical PMA for manufacture.

With that in mind, we propose in this project to use the system PMA test case results to train a predictive model to detect if a unit would fail the EST PMA beforehand, and in future work use this predictive model to aid the test operators during manufacture by giving them more confidence and precision on the decisions making. We expect that by using the predictive models to aid the production, the manufacture is optimized by reducing costs and focusing the operators' attention on tasks that directly affect the quality of the final products instead of idle units running tests multiple times.

In this scenario of predicting preemptively the failures, Industrial Failure Detection refers to a set of problems of detecting behaviour that does not conform to expected behaviour. The failures define undesired modifications and critical behaviour that affect the overall performance of the system. Moreover, the unawareness of failure detection in general industry may result in environmental degradation by the unnecessary disposal of material, an increase in shutdown time and operational cost, and product degradation [2, 3].

Fortunately for the manufacturers, the failures are rare events. Since they are not the ex-

pected behaviour in a quality production line, their availability is scarce and difficult to detect. Therefore, we propose a failure detection approach for predicting the outcome of the set of ESTs performed during the manufacture of telecommunication devices and take their scarcity into account. We are going to use the system test results as the features of our detector. They are less expensive than the ESTs and engineers hypothesize they are the most correlated to the ESTs. Therefore, one of the research questions of this research lies in confirming this hypothesis with the engineers of the company.

We employ deep learning and machine learning techniques to detect these failures beforehand with the system tests, while following certain production and evaluation constraints demanded by the operators of these tests. These operator constraints' are related to the impact of an incorrect classification on the manufacturing process.

An important step in this work is defining the performance metrics to analyze the failure detection models. When analyzing the effect of wrong predictions, a false negative is more impactful than a false negative, in accordance with the design expectations from the operators. These expectations assume that the cost of running the EST PMA multiple times is much higher than the cost of reviewing and repairing the unit. Hence, a false negative means that the failure detector predicted that the unit would pass the EST PMA, but in reality it failed this PMA. A false positive is the opposite, which means that the model predicted that the unit would fail the EST PMA, but it actually passed the PMA. Hence, a false negative states that the model is in the expected standards when it is actually not. And making this assumption affects the final quality of the product by making the operators do not perform the necessary repairs and revisions. And the other way around, with a false positive, would not significantly affect the current process of rerunning the EST multiple, since the cost of reviewing and repairing is less than the cost of running the EST multiple times. Therefore, we ought to design a methodology to account this constraint while also having a good performance on predicting failures.

Therefore, we propose creating a failure detection model methodology on these crucial tests by approaching the problem with three distinct preprocessing methods. The first method consists of using the raw test data from the system tests to predict the outcome of the important tests with minimal preprocessing steps. The second and third approach, however, assumes that the EST failures are anomalous behaviour. With this assumption, the failures can be approached as outliers, which could provide us with more significant features due to the expected difference between failures and normal behaviour observations. Therefore, we are going to employ deep autoencoders trained with normal data to perform a data reconstruction of the normal behaviour input features. And with the trained network, we

are going to use either the absolute reconstruction error from the output of the network and the latent space provided by the encoder of the network as features to perform the failure detection.

We expect that by using this approach with autoencoders, the features obtained can improve the failure detection rate of our selected models. These approaches with autoencoders consist of known methods for failure and anomaly detection for different types of neural network architectures [4–6]. Thus, we are going to compare the performance of all the classifiers for each of these engineered feature sets with autoencoders with the approach without autoencoder to test the hypothesis that considering failures as anomalies could improve failure detection performance.

Furthermore, we also employ resampling techniques to increase the performance of the models, since the failures on the crucial tests represent a minority of the samples. Without treating this imbalance properly, our models could bias towards the majority class and significantly underperform on the failure detection. We will simultaneously employ the resampling techniques of oversampling and undersampling to improve the failure detection rate in this imbalanced scenario. For over sampling, we are going to use random oversampling, *SMOTE*, and *ADASYN*. While for undersampling we are going to use random undersampling and instance hardness threshold.

We also propose to carry out a model selection with Bayesian optimization for hyperparameter tuning to obtain an optimal failure detection model and to analyze the impact of the resampling parameters and preprocessing steps. This consists on realizing multiple experiments with different configurations for the classification algorithms and the resampling techniques on a search space to find the best set of hyperparameter values. This optimization consists of an initial random search for hyperparameters. However, it gradually moves towards an optimal configuration that maximizes a performance metric through assuming the optimization to be a Bayesian process. Therefore, it reduces the search time for an optimal model.

With the selected models at hand, we are going to perform a threshold analysis of the resulting models to maximize the performance under the defined constraints by the operators in the industry. Since most probabilistic models classify a sample as a class if the classification probability is greater or equal than a decision boundary of 50%, changing this boundary would modify the model's performance. It would add bias to the model towards one class. However, changing this boundary may improve the performance of classification models under certain circumstances [7–9]. Therefore, we propose using a methodology to select a different classification threshold to enhance the performance towards the failure detection in

telecommunication devices. In this project, we propose using Youden's J statistic [10] to select a classification threshold that attends to the constraints of the problem and improves the classification performance.

Finally, our research proposes an exploratory analysis and experiments to answer if the EST PMA outcomes can be predicted with a less expensive system test cases performed immediately before. To answer that, we propose three distinct preprocessing steps followed by a classification threshold analysis with the Youden J's statistic. The preprocessing approaches consist of one that uses plain system test case results to train classifiers to predict the EST outcome, and the other two based on anomaly detection methods with autoencoders. Minor objectives consist of assessing the impact of each preprocessing step on the EST failure detection, by analyzing if that using autoencoders in the context of anomaly detection would actually improve the detection than using plain system test case values. Another minor objective is to inspect the impact of the threshold analysis on the failure prediction when associated with the preprocessing steps, and if this improves the detection rate in accordance with the defined constraints. Finally, preliminary experiment results indicate a critical similarity between the normal passes and the failures from the EST tests, making it difficult to perfectly separate both classes with a classifier. But treating the failures as anomalies by preprocessing them with autoencoders creates new features that improve the classification of the EST outcomes.

The organization of this thesis follows: Chapter 2 describes the methods used for this project as well as the related work; Chapter 3 reveals a more thorough explanation of the problem at hand and describes the data used; Chapter 4 defines the methodology proposed to perform the industrial failure detection; in Chapter 5 we define the experimental setup, the results obtained and presented the discussion; and finally, in Chapter 6 we describe the conclusions drawn from the project, as well as its limitations and future work.

CHAPTER 2 BACKGROUND AND LITERATURE REVIEW

Building the full pipeline for EST failure detection demands the precise usage of different techniques to deliver a good performance, but also a thorough revision of the literature of different approaches of failure detection. In this chapter, we present the background necessary to build the full classification pipeline. Also, we present the literature review comparing different authors to find a research gap to tackle the failure detection for our approach.

2.1 Supervised Learning

Supposing a training set with N examples of input/output pairs (x_i, y_i), in which an unknown function $y = f(x)$ generated each of these pairs, the supervised learning approach allows discovering a function h that approximates the true function $f(x)$ [11]. The function h is a hypothesis drawn from a hypothesis space H that serves as a model of the data. Therefore, given a failure industrial data set, the supervised learning approaches give us tools to create a failure detection model to predict failures beforehand given only the testing data obtained during manufacture.

To create a useful hypothesis, we split the available dataset into three sets. The training set instructs the model to approximate the $f(x)$ function. The validation set allows us to evaluate the performance on unseen data to pick the most promising hypothesis. And finally, the testing set to evaluate the feasibility and efficacy of a hypothesis on unobserved data for real-world issues. However, modelling such models from the hypothesis space requires attention to both the bias imposed, and the variance produced [11].

The bias reflects the tendency in which a hypothesis deviates from the expected value when averaged from different training datasets from the same source. It imposes restrictions to the hypothesis space and limits how well we can approximate our model to the ground truth by the hypothesis. The variance, on the other hand, reflects how much the training set impacts on creating the hypothesis due to the fluctuation in the training data. Hence, different sets of training data may significantly change the performance of the model in certain cases. In the case of rare event classification, the rare sample trained might be in distinct regions in such a way that the sampling of the training data affects the variance of the model.

Finally, the bias-variance tradeoff represents the modelling choice for machine learning models. We ought to build a model between complex models with low-bias that fit training data well and low-variance models that generalizes better. With that in mind, we selected Logistic

Regression, Random Forests and XGBoost classifiers to create a failure detection model.

2.1.1 Logistic Regression

The logistic regression classifier is an extension of linear regression for classification. A linear regression model is a combination of linear equations given by Equation 2.1. It associates a weight to each attribute of a data set plus a bias to give a corresponding output. This restricts the hypothesis space to linearly separable problems.

$$h_{\mathbf{w}}(\mathbf{x}_j) = w_0 + \sum_i w_i x_{j,i} \tag{2.1}$$

In Equation 2.1, j represents the j-*th* attribute of a dataset, i represents the i-*th* observation of the dataset, w_i and w_0 are the weights and bias associated with each entry x_j. However, for logistic regression, we apply the sigmoid function given by Equation 2.2 to soften the threshold for classification and transform into a probabilistic output.

$$Z(\mathbf{x}_j) = \frac{1}{1 + e^{-h_w(\mathbf{x}_j)}} \tag{2.2}$$

Moreover, we use the binary cross entropy loss to compute how close the model approximates the ground truth in Equation 2.3. In this equation, N represents the number of observations in the data set, y_i represents the ground truth for the i-*th* observation.

$$H = -\frac{1}{N}\sum_i^N y_i \cdot log(\mathrm{Z}(\mathbf{x}_j)) + (1 - y_i) \cdot log(1 - \mathrm{Z}(\mathbf{x}_j)) \tag{2.3}$$

Finally, we optimize the coefficients of the logistic regression by performing a stochastic gradient descent. It consists in computing the gradient based on the input batch and moving the coefficients towards a configuration that minimizes the Equation 2.3.

$$\theta_{new} = \theta_{old} - \alpha \mathbf{x}_j (y_{true} - Z(\mathbf{x}_j)) \tag{2.4}$$

Hence, Equation 2.4 describes the coefficients θ update. The updated θ_{new} is defined by the old θ_{old} minus the learning rate α times the product between the entry x_j and the difference between the actual labels and the predicted outcome by the model.

2.1.2 Random Forests

Random Forest are part of a specific type of ensemble learner in machine learning: the Bagging classifiers. First, Bagging classifiers create K different hypothesis or models from the training data set. For each model, it creates a subset from the training data set by sampling n samples from the N samples available with reposition, which $n << N$. Then, with the K classifiers at hand, the prediction comes from majority voting: the class that obtained the most predictions from the classifiers is selected as the final prediction. This approach tends to reduce the variance of the classifier and diminishes the tendency to overfitting the training data and generalizing better [11]. Overfitting hence means that the classifier does not generalize and cannot perform well on unseen data during the training.

However, when using decision tree classifiers to build Bagging models, it results in K trees highly correlated. This correlation comes from using one attribute with a high information gain, which makes the hypothesis less diverse because they would most likely have the same root and select the same attributes for splitting the branches of the tree. For creating decision tree classifiers, the training algorithm selects features with a high information gain to split the data into segments with enough information to make an accurate prediction and branch a decision based on that variable's values. And if we train Bagging classifiers with the same attributes continuously, we will obtain very similar K models and do not reduce significantly the variance. Therefore, Random Forests introduce a new step to guarantee reduce the variance and improve the generalization capabilities.

Random Forests include the random selection of features at each split when training the decision forest classifiers, instead of just training each model with a different subset of observations. This means that for each time the training branches the tree, it will select from a different subsample of features from the data set. This approach reduces the variance and is efficient to create because it considers fewer attributes while being created, skips the pruning step in singular decision trees, decreasing the tendency to overfitting, and we can build the trees in parallel [11].

2.1.3 XGBoost

Another popular method of ensemble learning is Boosting. Differently from the Bagging, this strategy introduces the notion of weight to the training set. Hence, a sample with weight equals to 1 would be equal to only having a single copy of it in the data set, whereas a weight of 5 would be equivalent to having 5 copies of it in the training data set. Moreover, contrary to Bagging, the hypothesis in Boosting are created sequentially, which inhibits the

parallelization as an acceleration to the training step [11].

At the beginning of a Boosting algorithm, we train the first model with the available data, assuming that all the samples have the same weight equal to 1. Then, we train the second model, allowing it to correct the mistakes committed by the previous model by reweighting the wrong predictions such that that their weight increases while the correct predictions reduce the weights. Therefore, the Boosting trains K models sequentially following this reweighting rule. The main concern of Boosting is training weak models sequentially until to reduce the bias of the predictor, assuming that each of these models are at least better than random guessing the observations.

Gradient boosting, however, uses a similar approach to sequentially train weaker models, but it introduces the gradient descent to minimize the parameters of a model. For binary classification, it minimizes the cross entropy loss from Equation 2.3 by updating the parameters of the next model as well as reweighting the samples.

Finally, *XGBoost* (eXtreme Gradient Boosting) is a scalable end-to-end boosting system [12] that utilizes this gradient Boosting approach to train weaker models such as Decision Trees to perform the classification task. It does the Gradient Boosting with regularization and pruning of the trees and allows parallel computing with a GPU and multiple machines [11].

2.2 Unsupervised Learning

A possible definition for unsupervised learning is a set of methods and algorithms that learn mostly from unlabelled inputs, which are usually more abundant than labelled data. They try to learn new representations of the same data, such as feature learning and dimensionality reduction, and learn a generative model in the form of a probability function to generate newer samples [11].

In this section, we will focus on unsupervised methods used in this project for both performing data visualization and to generate newer ways to represent our data such that we can enhance the performance of our failure detector. Those methods are UMAP and Denoising Autoencoders, respectively.

2.2.1 Denoising Autoencoders

Autoencoders are an unsupervised learning approach to Multilayer Perceptron Networks. Their main goal is to reconstruct the input data with the minimal error possible. Their design encodes the input to a latent space, a space which then is passed through a decoding

process to replicate the input. They learn through uncovering the intrinsic structures of the data by removing sources of variation within the data [13].

To achieve this, we divide an autoencoder network into two parts: the encoder, which maps the input x to a latent space and reduced dimensionality representation $\hat{z}$, and the decoder, which maps $\hat{z}$ back to x and reconstructs the input data in the form of $\hat{x}$ [11]. Equations 2.5 and 2.6 generalizes how the autoencoders perform the reconstruction of the input data.

$$\hat{z} = f(x) = Wx \tag{2.5}$$

$$\hat{x} = g(\hat{z}) = W^T \hat{z} \tag{2.6}$$

Figure 2.1 displays an example of a linear autoencoder. It contains a single layer on both encoder and decoder. When this autoencoder is trained with mean squared error function, it approaches the *Principal Component Analysis* (PCA), another unsupervised approach to compress and reconstruct the input data [11].

However, denoising autoencoders propose a different approach on which type of data to reconstruct. Drawn from anomaly detection methods [14], denoising autoencoder are trained with a single flavour of data: the samples corresponding to the expected normal behaviour of the production pipeline [4, 6, 14, 15]. Therefore, the hypothesis is that these networks will specialize in reconstructing the majority class and struggle to reconstruct the minority.

Finally, after reconstructing the original data and considering that the autoencoder will struggle to reconstruct the anomalous data, we perform the anomaly detection by either using the absolute reconstruction error for each of the features and observations of the data set, or the latent space resulting from the encoder.

2.2.2 UMAP

The *UMAP* (Uniform Manifold Approximation and Projection) is a semi-supervised manifold technique for dimensionality reduction. It is constructed on assumptions of Riemannian geometry and algebraic topology for constructing a scalable method applicable to real-world data. Similarly to the t-SNE (t-distributed Stochastic Neighbour Embedding), the algorithm projects high dimensionality data while preserving a more accurate representation of the global structure and maintains a superior runtime performance [16]. However, it requires more memory to finish the execution.

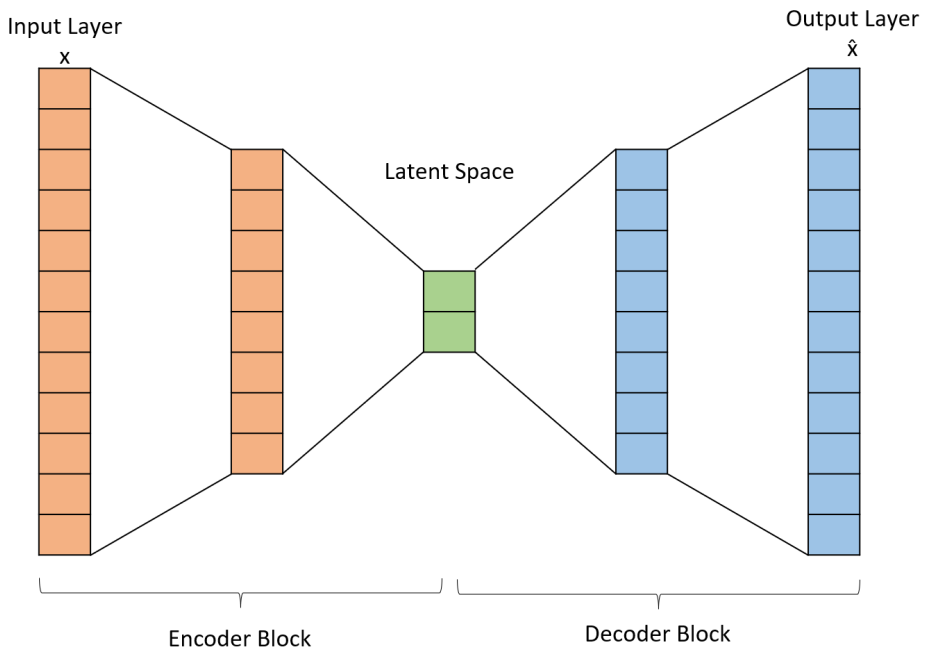


Figure 2.1 Autoencoder diagram used to reconstruct input data. It contains an encoder to reduce the dimensionality of the input into a latent space, and a decoder to reconstruct from the latent space the input of the network.

The algorithm converts the data into a topological structure with graphs based on the neighbourhood around each entry. Hence, it projects this topological structure into a reduced dimensionality while maintaining the distances found in these structures. The user has the option to feed the algorithm with each sample's label, and the algorithm will project the data while maximizing the separation of samples based on the target labels, similarly to Fischer's Linear Discriminant [17]. This linear discriminant enables the projection of the input data into a reduced dimensionality where the segregation between the classes is maximized, even though it may present overlapping. Therefore, UMAP is a useful tool for both visualization of data and a dimensionality reduction for classification algorithms.

In this project, we used this algorithm to provide a preliminary observation of the data set we are working with. Hence, it will not be used for classification purposes.

2.3 Imbalanced Learning and Datasets

Imbalanced problems have been widely explored in the literature due to its applications in real-world problems [18–21]. In binary detection, they are problems which have one class representing the expected normal behaviour and an abnormal, unexpected or even undesired

behaviour. The first is commonly referred to as the majority or negative class, while the second is the minority or positive class. Some problems may account to extreme proportions, such as 100:1 (1 positive sample for 100 negative samples), 1000:1 or even 10000:1, making these problems extremely difficult to deal with orthodox machine learning methods.

The unexpected behaviour in these problems have multiple origins. The behaviour could be *intrinsic*, which means they are a direct outcome of the nature of the data or phenomena. For example, in industry, industrial machines tend to naturally degrade due to their interaction with the environment, temperature, friction, and vibrations [15, 22]. And this degradation could result in unexpected behaviour of the machinery over time. These types of predictions are important to make because they might reduce the material losses of manufacture and reduce the risk of human error, which accounts for a significant percentage of accidents and losses [23, 24].

Another type of unexpected behaviour comes from *extrinsic* origins. They usually come from variable reasons such as time and storage of the data, not directly from the data space itself. For example, in a stream of balanced data, the source has sporadic interruptions which not data is transmitted. Hence, the system storage system might miss multiple instances of a class instead of the other, causing the final dataset to be imbalanced after the stream comes online again [25].

Nonetheless, there are multiple ways to overcome imbalanced datasets. In this project, however, we considered only resampling the dataset with oversampling or undersampling.

2.3.1 Undersampling

In imbalanced classification tasks, under sampling consists of removing samples from the majority class to prevent the classification model to simply memorize such class. It stops the model from overfitting the majority class and naively classifying all the samples as the negative, affecting the overall performance of the model on the positive class.

In this section, we will provide the basis of the undersampling methods utilized in this project. Those undersampling methods are: random under sampling and instance hardness threshold.

Random Under Sampling

Random under sampling removes samples randomly from the majority class. It randomly removes samples from the majority class to have the same amount as the minority class. However, the approach selected for this project consists of undersampling to a factor of the minority class, not exactly the original quantity of samples in the minority class.

Figure 2.2 shows an example of random undersampling on an imbalanced dataset, as well as the decision boundaries before and after applying the method. We can see that resampling directly affects the decision boundary of a Logistic Regression classifier in a 3 class problem. We also compare the decision boundary of a classifier without resampling and notice that the classification is more biased to the purple class.

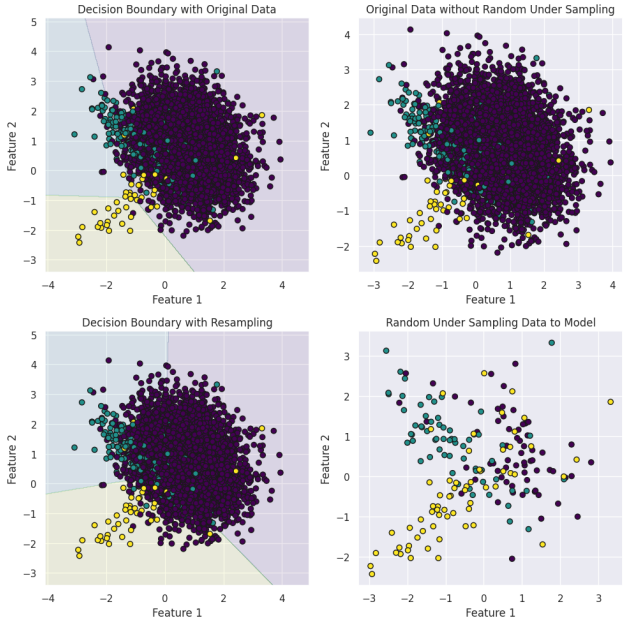


Figure 2.2 Random Under Sampling example.

Instance Hardness Threshold Under Sampling

On the other hand, Instance Hardness Threshold proposes to use a probabilistic classifier to select the samples to remove [26]. It first trains a classification algorithm that has a probabilistic output between 0 and 1. Then it selects the samples from the majority class whose probability of pertain to the minority class are the lowest. In other words, it selects only the easier samples from the majority class to remain on the final sample.

Figure 2.3 displays an example of Instance Hardness Threshold performed on an imbalanced dataset. We can observe that the algorithm selects only the easy samples from the majority class, further from the transition region between the classes. It leaves a region with no samples selected, which changes the final decision boundary of the same dataset.

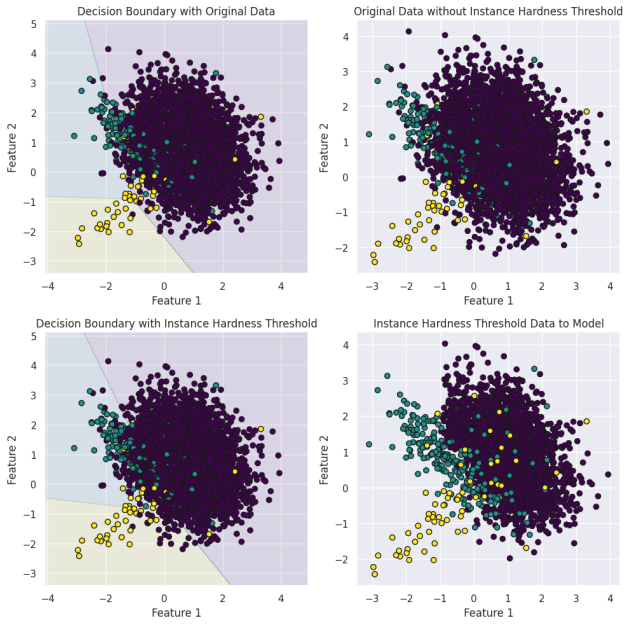


Figure 2.3 Instance Hardness Threshold under sampling example.

2.3.2 Oversampling

Oversampling techniques, on the contrary to undersampling, affect only the minority class. It tries to augment the number of positive samples available to the classifier, either by simply duplicating randomly selected samples or by artificially generating newer samples. These types of techniques also help to reduce overfitting on the majority class and potentially increase the performance of the minority class.

In this section, we will focus on the following resampling techniques: random oversampling, SMOTE, and ADASYN.

Random Oversampling

Random oversampling selects samples from the minority class and replicates them to the final resampled dataset. This replication is with replacement, and we can add small perturbations to the selected positive samples to increase variability of the dataset.

Figure 2.4 displays the decision boundary of an imbalanced dataset with and without random oversampling. We also include a small perturbation to the replicated samples.

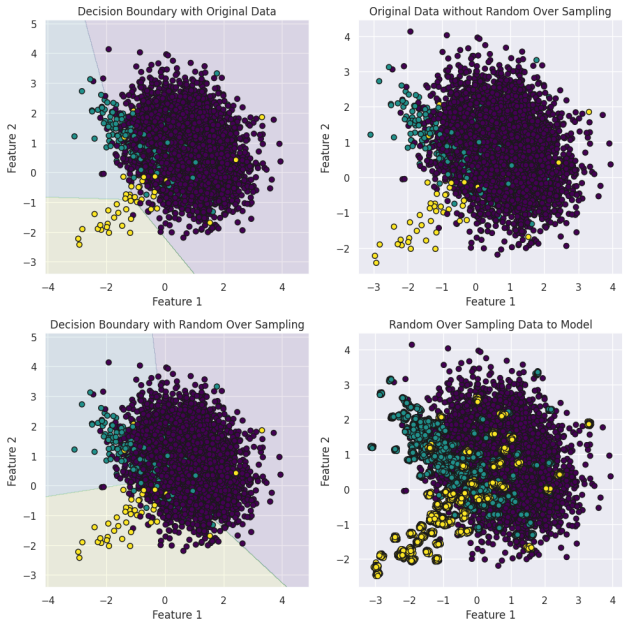


Figure 2.4 Random Oversampling decision Boundary comparison between not applying and applying the technique.

SMOTE and ADASYN oversampling

The Synthetic Minority Oversampling Technique creates synthetic data differently from randomly oversampling. It takes two random minority samples from the minority line and creates a linear function between those two points. With the functions at hand, it creates samples by selecting points between the two samples, interpolating the function [27].

Similarly, the ADASYN method creates new samples by interpolating between two minority samples. However, it selects the samples to create newer ones by using a k-Nearest Neighbours classifier (kNN), and those who are wrongly classified by the kNN are selected to over sample with a similar approach to the SMOTE method [28].

Figure 2.5 and 2.6 shows the effect of those algorithms on the decision boundaries of a classifier in an imbalanced setting. We can see the interpolation regions on each of the algorithms. However, ADASYN samples are concentrated in regions between the classes, thus harder samples to classify with the kNN.

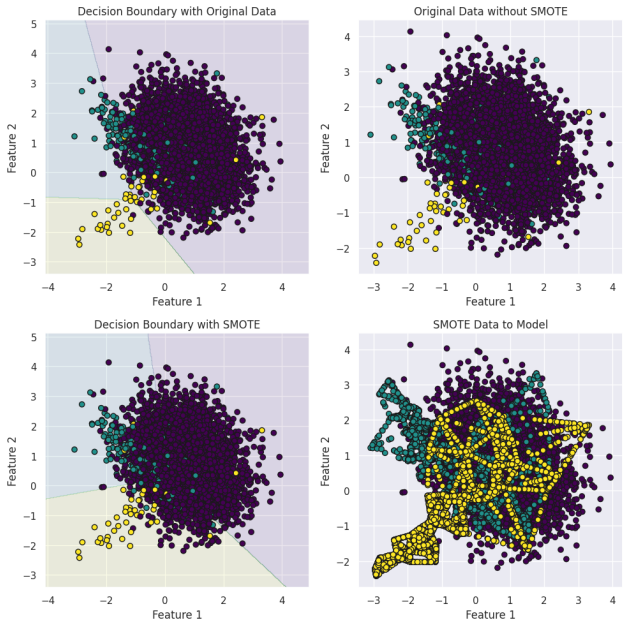


Figure 2.5 SMOTE oversampling techniques and their impact on the decision boundary of a classifier.

2.4 Model Selection

2.4.1 Hyperparameter Tuning with Bayesian Optimization

Model selection is an important step for creating any machine learning pipeline. It enables the user to test different parameters throughout the process and select the model that attends better the demands of the user. However, the model selection must make sense to the user, and it needs to define an objective function which will be either minimized or maximized. Therefore, Hyperparameter tuning stands for the process of selecting the hyperparameters of a classifier or any preprocessing steps in a machine learning pipeline to minimize or maximize a predefined metric.

The model selection in a machine learning pipeline begins with defining the data for training the classifiers and for evaluating the trained model. We selected the approach to split data into three segments: training, validation, and testing. We use the training set to make the model learn, and the validation to assess the trained model with a given set of hyperparameters for a defined metric and select the models with better performance on this set. And the testing set is used only on the final selected model. Furthermore, we define the hyperparameter configuration space, which we draw the hyperparameter values to train a model that later can be selected. We call each of the trained models during the search for optimal hyperparameters as trials.

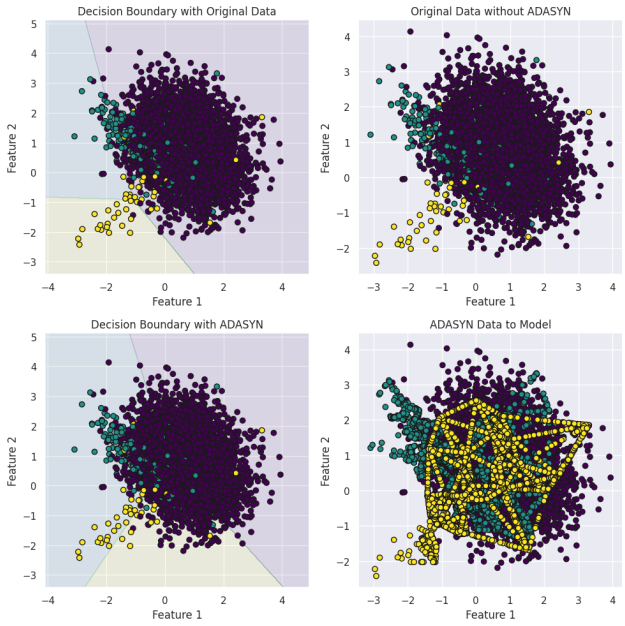


Figure 2.6 ADASYN oversampling techniques and their impact on the decision boundary of a classifier.

There are multiple approaches to handling model selection. The most common are grid search, randomized search, and Bayesian optimization. The grid search is a brute force searching method that tests all the possible configurations in a given search space. It is simple, it allows parallelism and results in the best configuration in the search space, but it is exhaustively time-consuming to give a useful configuration. The randomized search, on the other hand, takes fewer trials to give a useful response, also allows parallelism and can be combined with early stopping methods. It randomly selects configurations from the configuration space of size S for a maximum of $s << S$ trials. However, the randomized search does not guarantee an optimal model in the end, exactly because it does not explore the whole configuration space [29]. It searches naively through multiple configurations. Therefore, the Bayesian optimization comes to tackle this problem and searches through fewer configurations but more confident that the resulting model is optimal than pure randomized search.

The Bayesian Optimization is an old effective approach to solve for solving computational costly extrema functions, solving functions without closed-form expressions, difficult to evaluate derivatives and non-convex functions [30]. Basically, Bayesian optimization minimizes the loss function of a given objective function $f(\theta)$ with respect to the search space A and the configuration of hyperparameters θ. Equation 2.7 defines the objective function for this approach.

$$\theta = \arg\min_{x \in A} f(\theta) \tag{2.7}$$

Also, this approach is derived from the Bayes' theorem, which states that for evidence X, the posterior probability $P(\theta|X)$ of the model with hyperparameters θ is equal to the probability $P(X|\theta)$ times the marginal probability of $P(\theta)$. Equation 2.8 defines the general equation for the Bayes problem.

$$P(\theta|X) = P(X|\theta)P(\theta) \tag{2.8}$$

The key idea of this type of optimization is first estimate $P(\theta|X)$ with initial random trials to estimate a function that, based on the hyperparameters, can estimate the objective before computing it. Then, the algorithm enables the minimization of the cost function $f(\theta)$ for each new trial attempted. Each new trial updates $P(\theta|X)$ and gives a more precise estimation of the objective function. Therefore, each new trial walks towards the direction of minimizing $f(\theta)$ while updating the estimation to find a combination of hyperparameters that accomplish this objective [30]. Therefore, the selected model has more chances having the optimal hyperparameter values to solve the problem.

The main advantages of performing Bayesian optimization lies in being more reliable than randomized search to give better results in $n << N$ attempts. However, the algorithm does not allow parallelization and is memory expensive in comparison to grid and randomized search. A possible solution to this is estimating the objective before executing a trial by giving it an admissibility boundary [29].

2.4.2 Evaluation Metrics

The performance evaluation metrics used in this project consist of accuracy, precision, recall, F1-Score, negative precision and ROC-AUC metrics. Most of these metrics are mathematical formulations of the following variable definitions:

- True Positives (TP): The model predicts a sample as a failure and is actually a failure.
- True Negatives (TN): The model predicts a sample as normal and is actually normal.
- False Positive (FP): The model predicts a sample as a failure, but is actually normal.
- False Negative (FN): The model predicts a sample as normal, but is actually a failure.

In a confusion matrix used to evaluate the performance, these definitions would be displayed as in Table 2.1.

	Actual Value	
Predicted Value	TN	FP
	FN	TP

Table 2.1 Display with the description of each position of a confusion matrix.

The accuracy is the proportion of corrected classified samples over all samples available. Equation 2.9.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FN + FP} \tag{2.9}$$

Precision is the proportion of True Positives over all samples that the model classified in the positive class. Equation 2.10 shows the precision equation.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2.10}$$

Another desired performance metric is the precision of the negative class. This metric is important in a context where wrongly classifying failures as normal classes are more critical. Equation 2.11 describes how to compute the precision of the negative class.

$$\text{Negative Precision} = \frac{TN}{TN + FN} \tag{2.11}$$

Recall, however, is the proportion of all True Positives over all samples originally from the positive class. Hence, Equation 2.12 describes the computation of the recall. We can also name this metric as *True Positive Rate*.

$$\textit{Recall} \ = \ \frac{TP}{TP + FN} \tag{2.12}$$

The specificity measures the proportion of the True Negatives over the sum of True Negatives and False Positives. In other words, is the recall of the negative class. Equation 2.13 displays the computation necessary to find the specificity.

$$\textit{Specificity} \ = \ \frac{TN}{TN + FP} \tag{2.13}$$

The *FalsePositiveRate* is given by Equation 2.14. It can also be rewritten as one minus the specificity, and we are going to use it to perform the threshold analysis.

$$FalsePositiveRate = \frac{FP}{FP + TN} = 1 - Specificity \tag{2.14}$$

The F1-Score is the geometric mean between precision and recall, defined in Equation 2.15.

$$F1\text{-}Score = 2 \cdot \frac{Recall \cdot Precision}{Recall + Precision} \tag{2.15}$$

Finally, we have the Receiver Operating Characteristic Curve as another evaluation metric for classification algorithms. It is a two-dimensional classification performance measurement that explicitly differentiates good and bad models, but struggles to differentiate two good models. Whereas, the area under this curve represents a scalar measurement of performance of the classifier [31].

When we train a classifier for a given data set, we obtain a confusion matrix as a result from Table 2.1. However, the results come from an operating point: we classify samples that have at least 50% of probability to be from the positive class in probabilistic approaches. Therefore, by changing this operating point or threshold to another percentage, we would have another confusion matrix in the end because the model would classify the samples differently. Therefore, we measure the performances of the model on the *TruePositiveRate* and *FalsePositiveRate* of the binary classifier for different thresholds and compile it to a single value to determine the performance of a model in differentiating the two classes. And that is precisely what the ROC-Curve measures [32].

Therefore, the ROC-AUC metric is useful for selecting the best model in a model selection. This happens because the measure evaluates how well the model classifies each sample, without necessarily biasing towards a single class. Thus, we selected this measure for model selection in the hyperparameter tuning.

Figure 2.7 displays a ROC-Curve of a trained random forest classifier compared to a naive classifier that randomly classifies a sample. The trained classifier has an area under the curve greater than 0.5, indicating that the model is better than randomly assigning samples, as the naive classifier curve indicates.

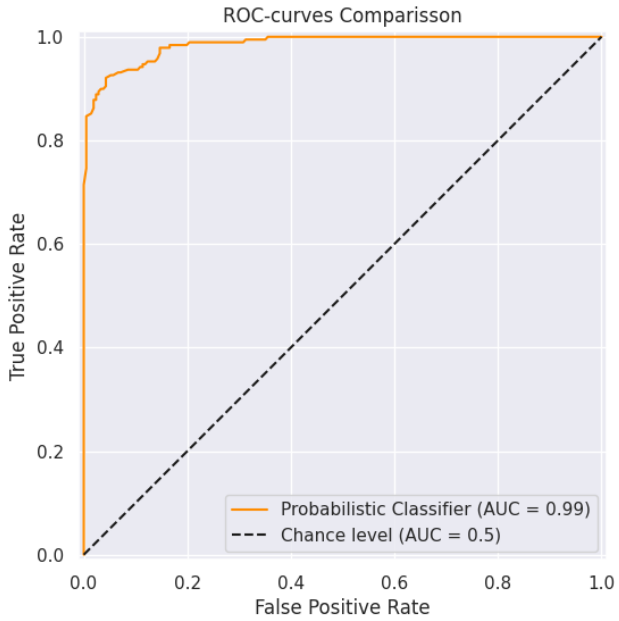


Figure 2.7 ROC-curves of a random forest classifier and a naive random classifier.

2.5 Related Work

2.5.1 Imbalanced Classification Problems in Industry

The failure detection problem disposed in this project is an imbalanced problem. A common way to deal with imbalanced problems is using resampling techniques of oversampling and under sampling. The resampling allows the machine learning algorithms to potentially generalize better and actually detect the failures from the minority class more often. However, a common mistake is how to combine the resampling techniques with other preprocessing steps such as feature selection to achieve a better performance on imbalanced problems. Exploring this issue, [33] performs a wide study on the effects of resampling and making feature selection on different types of classifiers and datasets. They use as resampling strategies random oversampling, SMOTE and random undersampling. For the feature selection, the authors prefer to use these seven techniques: statistical tests on features (T-test, ANOVA, chi-squared and information gain), random forest importance, RelieF [34], and finally SVM recursive feature elimination (SVM-RFE) [35]. Finally, after thorough experiments, the authors conclude that in the majority of cases, performing resampling after preprocessing the features results in better performances.

Generating new samples with oversampling is the most common approach for imbalanced learning in industrial setting. [19] utilizes test data to predict failures in hard drive disks

manufacturing. The data features a high dimensionality and imbalanced datasets. They perform feature selection with gradient boosting to tackle the high dimensionality. Thereafter, they use various techniques to handle the imbalance: SMOTE [27] combined with other resampling techniques. Finally, they use XGBoost and Support Vector Machine algorithms to classify failures. They also check how well their models do by using ROC-AUC and Precision-Recall area under curve (PRC-AUC).

Other authors propose a combination of feature selection and oversampling to create an imbalanced learning pipeline focused on the diesel engine manufacturing industry [20]. The problem consists of an imbalanced multi-class classification problem with high-dimensional features, consisting of one unqualified class and other three qualified classes while the features consist of an assembly line with 100 stations and 172 assembly quality characteristic detectors used as the features. Consequently, the authors propose a hybrid feature selection and oversampling method by using an application of Particle Swarm Optimization in accordance with feature importance evaluation and selecting the minority class samples that are difficult to classify, obtaining a geometric mean accuracy of 98.2% for all predictions.

However, [36] proposes a methodology for imbalanced datasets in industrial settings. It uses a combination of Mahalanobis SMOTE and Generative Adversarial Networks to produce new samples to balance the datasets. They compare their MSMOTE-GAN model to bootstrapping and tableGAN on the water-heater-liner production and other public datasets, with experimental results assessing that their proposal significantly improves the performance of the classification models with Neural Networks, XGBoosting, Random Forests and Decision Trees.

Other authors propose resampling techniques for quality and failure prediction in industry. [37] proposes a pipeline for manufacturing quality prediction on high-dimensional imbalanced datasets by resampling with SMOTE and TomeLinks and selecting features with the features importance provided by the random forest algorithm. In the first step, they generate the samples for the minority class with SMOTE and remove duplicates using TomeLinks algorithm. Then, apply the random forest to compute the feature importance of each column of the dataset, ranking them from most to least important. Finally, they select the best number of features to obtain the optimal quality predictor with the Fine Gaussian SVM classifier. Their experimental results indicate that their methodology was more accurate while shifting the classification bias towards the minority class.

Similarly to our approach with a multistaged industrial production using testing data performed during manufacture, [21] proposes a two-stage detection framework to predict failures in manufactures with anonymized features available on Kaggle and imbalanced target fea-

tures. They first cluster the data into smaller groups based on the manufacture process involved, and then apply a supervised learning algorithm to each group to perform the prediction of the failures. They obtained an AUC ROC value of 0.692, which they claim that is an expected outcome for industrial failure detectors. [38] proposes as well the development of a machine learning tool to predict failures and trends associated with an antenna testing system. They use data from the tests applied during manufacture of antennas to predict failures that occur at the end-of-line tests after certain stages of production. This kind of testing validates the functioning of the antenna and can be used to predict future malfunctions and avoid failures on products before they are delivered.

Also, [39] discusses a possible framework for detecting failures and patterns in production lines, with no practical implementation whatsoever. Although it is similar to [21] in a way of preprocessing the data with unsupervised learning, it proposes a semi-supervised framework to monitor quality in manufacturing systems. It first uses clustering algorithms to define labels of some available samples, with a hierarchical clustering algorithm, and then uses an SVM model to predict and detect those patterns.

A common approach to designing failure detection systems in industry involves using sequential data to extract features to make the detection possible. [1] suggests a Wide-Deep-Sequence Model-Based quality prediction method in industrial process analysis from different types of industrial processes. They extract time-domain features from time series by extracting the statistics in Figure 2.8. They are the mean value, absolute mean value, variance, max-min difference, skewness, and kurtosis from *tf1* to *tf6* respectively. Their target quality variable again is a continuous value, and they achieve decent results with the methodology proposed. Authors also claim that feature selection with random forest classifiers is suitable in high dimensional data. Meanwhile, [40] also extracts statistical features from torque measurements presented as time series. The statistics for each torque measurement consist of the mean, standard deviation, skewness, and kurtosis. The authors obtain the torque measurements from a milling machine used on the production of hydraulic valves to predict the diameter of drilled and reamed bores of the valves. Finally, they use the random forest algorithm to perform the regression with the statistical features from the torque measurements. They achieve precise results with these features and achieve a much faster and confident quality control in their setup.

Another common approach to maximize the performance of classifiers in imbalanced settings is by using threshold moving methods. These methods aim to change the classification threshold of classification algorithms to maximize the detection of failures without changing hyperparameters of the classification models. Numerous authors use Youden's J statistic

T-D Feature	Description
tf_1	$\frac{1}{n}\sum_{k=1}^{n} x_i^{(k)}$
tf_2	$\frac{1}{n}\sum_{k=1}^{n} \lvert x_i^{(k)} \rvert$
tf_3	$\frac{1}{n}\sum_{k=1}^{n} (x_i^{(k)} - \bar{\boldsymbol{x}}_i)^2$
tf_4	$max(\boldsymbol{x}_i) - min(\boldsymbol{x}_i)$
tf_5	$\frac{\frac{1}{n}\sum_{k=1}^{n}(x_i^{(k)}-\bar{\boldsymbol{x}}_i)^3}{[\sqrt{\frac{1}{N}\sum_{k=1}^{n}(x_i^{(k)}-\bar{\boldsymbol{x}}_i)^2}]^3}$
tf_6	$\frac{\frac{1}{n}\sum_{k=1}^{n}(x_i^{(k)}-\bar{\boldsymbol{x}}_i)^4}{[\sqrt{\frac{1}{N}\sum_{k=1}^{n}(x_i^{(k)}-\bar{\boldsymbol{x}}_i)^2}]^4}$

Figure 2.8 Time-domain features. Source: [1]

to maximize the performance on imbalanced classification problems in biomedical problems [7, 8, 41, 42], which this method was originally designed to attend [10]. In industrial settings, many authors utilize this approach to optimize the failure and anomaly detection methods with probabilistic classifiers [9, 43, 44], even though less frequently. Therefore, we propose using this statistic to optimize the threshold of probabilistic classification algorithms for failure detection systems in industrial settings.

2.5.2 Neural Networks for Failure Detection

Still on the matter of performing industrial failure detection with sequential data, [45] and [46] propose utilizing transformer architectures [47] to perform time series classification on multiple industrial settings. The first proposes using a two-tower scheme with two transformer encoders that process time and channel features from multivariate time series. After processing the entire sequence on each tower, the authors combine the results using a gated structure that weights each tower according to the impact on the classification of each type of feature. Finally, the latter model proposes using a single transformer encoder block with an embedding input layer for each time step. The authors then initialize the model with an unsupervised training by reconstructing the input features with some missing values on the sequence. And so, the models train the model with a supervised approach to classify entire time sequences at once. Both transformer models keep up with the state-of-the-art models for time sequence classification for most of the datasets evaluated. However, the transformers are suboptimal in some tasks compared to the state-of-the-art models. Thus, the main advantage of using transformers for sequence classification models is their computational efficiency, allowing to scale these models to much more complex settings [48].

A common approach in the unsupervised anomaly detection field is using autoencoders to highlight the anomalies and possibly the failures to facilitate their detection. [15] proposes using deep autoencoder architectures to perform outlier and anomaly detection on machine learning problems. Inspired by the Robust Principal Component Analysis method, the authors extend current deep autoencoders by using an anomaly regularized penalty based on lasso and ridge regularization. They achieved results with 30% improvement in scenarios where there is no clean nor noise-free data available. Their highest achievement was to create non-linear representations on noisy environments of the available data, and for the sake of failure detection, these representations are valuable specially when there is not much failure data available.

Other authors extend the usage of autoencoders for failure detection. [14] also proposes a deep autoencoder structure to reconstruct input data and use it to build a classifier. It uses the reconstruction error between the decoder output and the original data without anomalies as the features to build. With this reconstruction error, they perform anomaly detection techniques of the data, such as Enhanced Threshold Method (ETM), Threshold Classification Method (TCM) and Naive Threshold Method (NTM) [14]. Also, they randomly introduce Gaussian noise to the data to the data to enhance the detection. [49] uses a wire production dataset to assess the quality of production in an industrial pipeline. It performs parameter exploration to obtain a neural network that predicts better the quality of three eccentricity parameters by using 30 input features. The target variables consist of three continuous variables, and the neural network performs a regression. Their approach improves the results by up to 50.26%.

Autoencoders are also used in Internet of Things (IoT) contexts. [6] proposes a device type identification through network traffic for the detection of anomalies in the IoT. They use autoencoders and other methods to perform dimensionality reduction on traffic data to feed classification algorithms. But also, the authors utilize the reconstruction error to feed the classification algorithms to detect anomalies. The anomalies, however, were simulations performed by the authors and do not represent real-world attacks to the IoT devices. Even though, they obtain a performance of 0.981 in the F1-Score metric for the best model, though the autoencoder approach does have a comparable result.

Anomaly detection autoencoders are also used for anomaly detection with *SCADA* (Supervisory Control and Data Acquisition) data for wind turbines. [5] proposes using autoencoders on healthy and normal behaviour of wind turbines. They use the reconstruction error data to predict failures in multiple components. Finally, their approach also proposes certain explainability on when and why the failures or anomalies occurred.

Some authors use autoencoders but with other neural network architectures, such as *Long Short-Term Memory* (LSTM) [50], to perform anomaly detection. [4] uses LSTM autoencoders to train the model on normal traffic patterns to learn a compressed version of the data. They use, instead of the reconstruction error, the latent features obtained after the encoder to train an SVM model to detect the anomalies on the sequences. They claim that their approach enhances the detection rate, and it reduces the processing time significantly. Hence, they secure more confidence in guarding networks from malicious traffic.

After reviewing the literature, we noticed a common approach to industrial failure detection. Most of the papers suggest dealing with imbalanced datasets with resampling techniques, specially SMOTE for oversampling. But, there are not many papers focused on telecommunication manufacture failure detection and the tests performed during the fabrication. Therefore, our research gap lies in applying exploratory and failure detection methods in this telecommunication context. Also, an important approach is feature extraction from the current features. If the authors are dealing with sequences, they extract statistical features from the data or use proper neural network architectures to deal with sequences. However, an approach useful for the non-sequential data present in this project is reconstructing the input features with denoising autoencoders. Therefore, we propose comparing the performance of three distinct approaches: one that uses reconstructed data from autoencoders, one that uses that autoencoder's encoder latent space output, and another that uses plain test case results with minimal preprocessing.

CHAPTER 3 PROBLEM AND DATA SET DESCRIPTION

Describing the context of the project is an important step before defining the methodology to tackle the problem. In this chapter, we are going to contextualize the problem of detecting EST failures with system test case values, and describe how we built the dataset.

3.1 Introduction

Keeping a standard of production is utterly important to maintain the quality of industrial production. This predefined standard of expected behaviour allows us to observe when a unit does not conform with the industrial standards, but only in later stages of production, when the errors and failures already propagated through the production pipeline. Hence, if we can detect these anomalous units in production with a predictive model before propagating to next stages, we can optimize the production pipeline by performing repairs and revisions before committing to the costly stages of manufacture. Therefore, in this project, we focus our attention on maintaining the quality of manufacturing telecommunication devices by analyzing and monitoring a portion of tests performed during their production.

These telecommunication devices include products such as routers, switches, optical devices and other products used for fast and secure communication through the internet. It requires multiple steps to build a final product, such as assembling different electrical and optical components on a circuit board and guaranteeing that all parts are working accordingly, for example. Therefore, it is necessary to conceptualize a high-level and multistaged production pipeline to aggregate multiple parts in a way that it allows creating new products by aggregating new parts to existing units. The concepts necessary to build this multistaged production pipeline are:

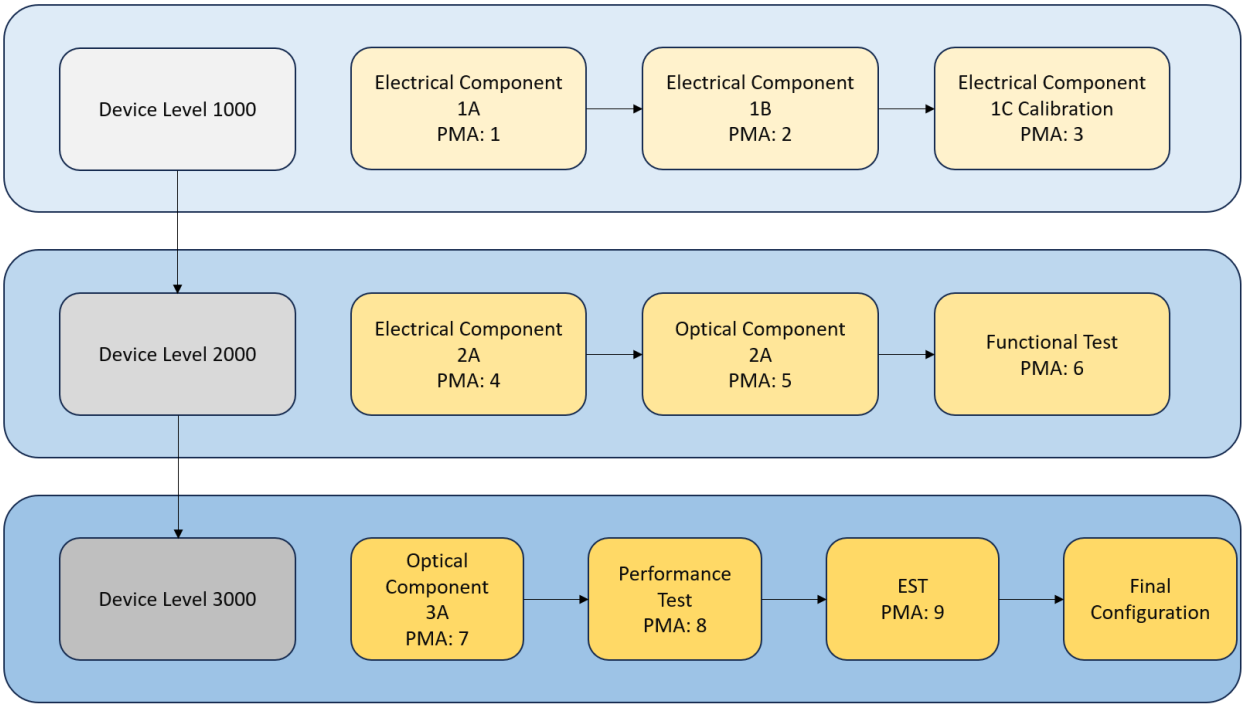
- *Product*: The product is a set of Part Numbers or the blueprint defining the steps the manufacturer submits a unit until its final form.
- *Unit*: A unit is the physical manifestation of a product in the production pipeline.
- *Part Number*: The Part Number is the identification number of a product at a specific moment of its lifecycle.
- *Serial Number*: The Serial Number identifies a unit at each step of the production line and changes on each step.

- *TLSN*: The Top-Level Serial Number (TLSN) consists of the Serial Number of the final form of a unit.
- *Test Flow*: The Test Flow is the set of Process Module Assemblies (PMA) that a product will go through during its lifecycle.
- *PMA*: Process Module Assembly (PMA) are test schemas designed to examine the functioning of specific components of a unit at each stage of its production cycle.
- *Test Cases*: The tests compose each PMA, and they examine responses of each module to certain stress submitted to the product.
- *Test Runs*: The test runs are a batch of test cases from a PMA applied to a unit in a given moment of time. We identify each test run by their *TestRunID*.

Figure 3.1 displays a simplified example of a blueprint of a device manufacture. It shows the production pipeline of a product from the perspective of adding components at each step. On each stage, we add components to the device and perform quality tests to assure their quality. Each set of tests is called a *Process Module Assembly* (PMA). Thus, for each PMA ran at a specific time, we identify it with the TestRunID. The chart divides the production of a device into three levels. We identify the stage of production of a product being made at each level by the PartNumber, whereas we identify each unit at that level with the SerialNumber. At the final form of each level, we have the Top-Level Serial Number (TLSN) to identify the fabricated unit.

During the manufacturing process, the company's operators perform quality tests within a test flow to guarantee that the products follow a certain quality pattern at a specific stage of production. Each level of production contains a set of PMAs, which each PMA contains its set of test cases. For a regular TestRunID, if all test cases within a PMA pass, we consider the PMA outcome as passed and move forward to the next PMA. At the end of each level, if all PMAs passed, we move to the next production level. If the PMA fails for a specific TestRunID, the operator either performs a rework on the unit and reruns the PMA, or runs the same PMA immediately after without any rework. Most of the tests are done with software, even though some test cases require visual inspections on the products. Moreover, most of the important tests result in numerical outputs according to the operators, even though some tests result in discrete and categorical outputs. The latter were discarded for this analysis due to the operator's advice, since they mostly do not assess the performance of the assembled components. The numerical outputs are responses of the unit to a certain stress applied to it. If we apply a certain voltage to specific components during a specific

Figure 3.1 General process flow of a unit in Ciena. Each level adds new components and functionalities to the product.



amount of time, we may obtain the maximum and minimum current during that time for this specific test, for example. These values obtained in a test must lie between an acceptance range, defined by a lower and upper bound. If the test case value lies within this range, we consider that the unit passes the test case. Otherwise, it fails the test.

An important type of experiment in the manufacture is the Environmental Stress Tests (EST). In Figure 3.1, these tests are performed during the last stages of production for the given unit. They are frequently the last layer between the manufacturer and the final consumer. These tests emulate stressful operations in the telecommunication devices by performing exhaustive byte package exchanges among different units and assessing the performance of the components of the product under these types of operations. They usually take multiple hours to complete, and operators run them multiple times before moving forward in the manufacture.

Besides the EST, the System tests are also important steps to complete the production. They are performed right before the ESTs, and do not demand as much material cost as the ESTs and do not take multiple hours to complete. The operators have a special interest in those tests because the EST tests are performed immediately after them without any

direct modification to any of the unit's components. However, they can perform repairs and modifications to the unit if the EST fails and do the system and EST tests again sequentially. Therefore, the operators hypothesize by their experience that the system tests are the most correlated to the EST and can be used to predict the outcome of the EST immediately after.

Currently, the engineers do not perform any prediction with a supervised model of the outcome of the EST during the production. Thus, we can say that they currently use a naive predictive model that assumes that every unit would have an EST failure outcome. And the delay in production is caused by the lengthy processing time required to complete multiple runs of the ESTs. Therefore, we intend to build a classification model to perform failure detection for an EST test right after the operator performed a System test. Assuming that our model confidently predicts a failure or even a pass on the EST tests, we would significantly reduce the number of times the EST tests performed by the operators. Thus reducing the production bottleneck caused by the ESTs because we assume in this project that the cost of performing EST tests multiple times is much higher than performing revisions and repairs.

However, due to the current production pipeline and the way the engineers deal with the tests, we have more EST passes available to train our classification models than we have EST failures. This promotes an imbalance between the number of failed and passed test runs. Therefore, we propose adding to our classification pipeline approaches to deal with this imbalance and promote a proper failure detection of the ESTs beforehand.

Finally, our project proposes three distinct approaches to deal with Environmental Stress Test failure detection. The first approach consists of using the raw system test case results with minimal preprocessing to train a classifier algorithm to predict the failures. The other approaches propose an anomaly detection preprocessing step by training an autoencoder neural network to reconstruct the input system test cases corresponding to EST passes. Thus, we expect that by training the autoencoders with this data, the reconstruction errors and latent space provide enough information to enhance the failure detection. All of these preprocessing approaches lead to a binary classification problem, whose difference lies in the input features given to the classifiers.

3.2 Dataset

During the manufacturing process, the technicians perform several test runs on the same unit. They do this until the test passes to move forward to the next production phase. Hence, a set of test cases compose a PMA at a given stage of production, and modelling the test runs is crucial to determine the best approach to predict the failures. Therefore, an important

step of our project is to define what are the features and, most importantly, what are the target outputs.

Initially, we were given two sets of data from the company's database. One corresponds to the outcome of tests, while the other presents the results of the test cases. The first table contains the number of tests that have passed or failed for that run, the time it took to complete all the tests and the unique identifier of the ensemble of test-runs ran at that moment, as well as the outcome of the test run. Table 3.1 presents the possible outcomes of a PMA test run, which are the possible target labels. Whereas, the second table available in the database contains the test case value outputs of the ensemble of tests in a PMA. Thus, with the test run identifier at hand, we could extract the feature test cases used to identify the failures, as well as the target labels.

Table 3.1 Possible unit test outcomes for each product given a PMA.

Unit Test Outcome	Description
0	Unit failed a significant number of tests.
1	Unit passed all the tests.
2	Unit waive-passed a significant number of tests and operator approved.
3	Unit did not complete all tests.

For this project, we considered only the units with EST outcomes 0 and 1 from the Table 3.1, and discarded the units with the other outcomes 2 and 3. We discarded these units to reduce the scope of the project to a binary classification problem, and because the engineers were not interested in these specific outcomes. However, we swapped the final dataset labels so that the value 0 corresponds to a unit that passed all the EST test cases, while the value 1 corresponds to a unit that failed a single or multiple EST test cases. Hence, Table 3.2 demonstrates the final disposition of the labels used in this project.

Table 3.2 Unit test outcomes for each product given a PMA used in this project.

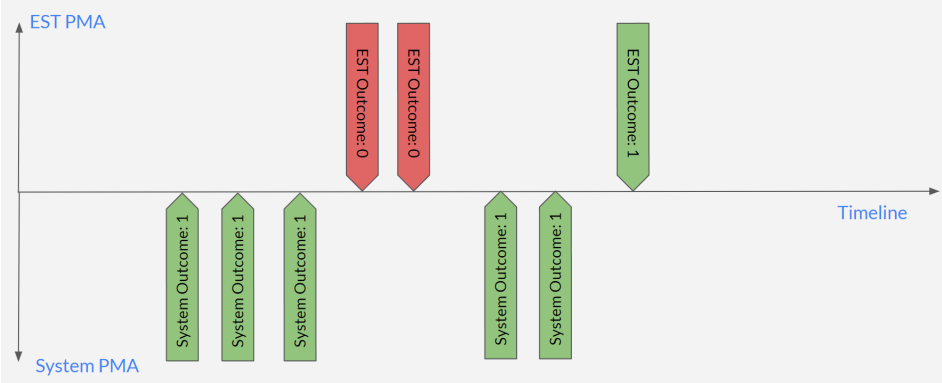
Unit Test Outcome	Description
0	Unit passed all the EST test cases.
1	Unit failed any EST test case and is considered a failure.

The next step consists of filtering the data used to perform the task at hand. First, we separate the consecutive System Test PMA for feature extraction and the EST PMA for the target label selection. With this at hand, we have the lifecycle of a given product that has

these two PMAs in their lifecycle in Figures 3.2 and 3.3. We can observe that both cases have multiple runs of both System and EST tests before finishing the process flow.

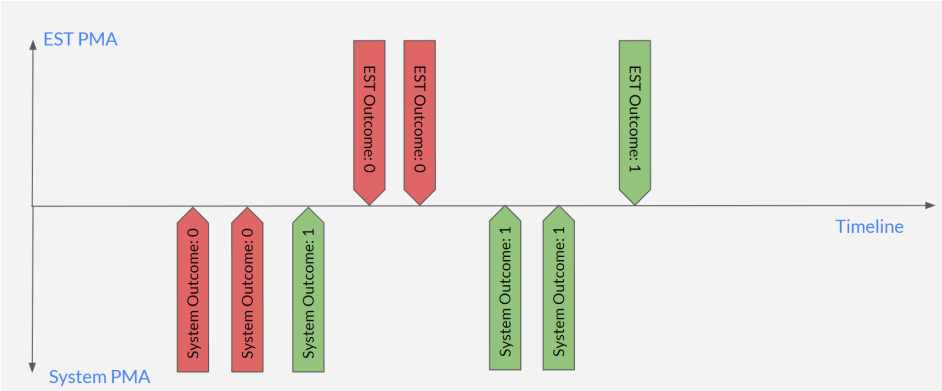
Figure 3.2 describes the lifecycle of a product, which the engineers desire to predict the EST outcomes. All the instances of the system tests passed for this unit. However, some EST instances failed, and the operators require predicting these failures beforehand because no System runs failed. Therefore, we selected units that attended to these conditions to define the feature-target pairs.

Figure 3.2 Lifecycle diagram of a desired product to select for prediction.



On the other hand, Figure 3.3 describes a unit that is not selected for the classification dataset exactly because it has system PMA runs that fail. The engineers consider that a unit failing the system tests is critical already. Therefore, they consider units like this to be not in conform with production standards and put them in a different category than the desired units. Finally, we do not consider these units in the failure detection process for this project.

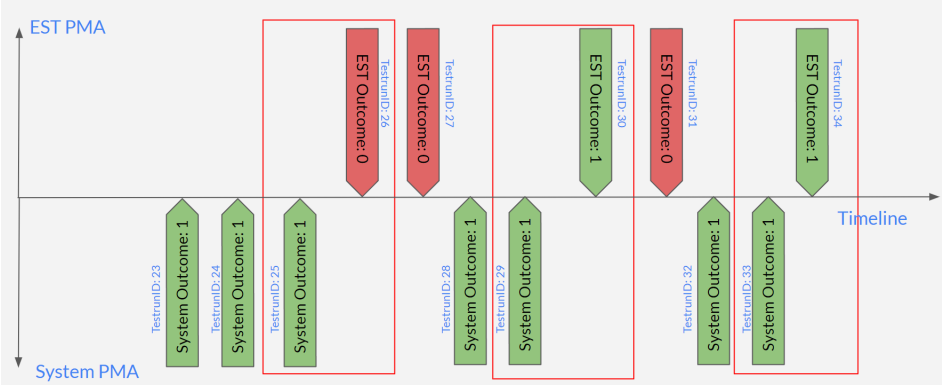
Figure 3.3 Lifecycle diagram of an undesired product to select for prediction.



We have Figure 3.4 describing how we extracted the pair features and labels to perform the

classification of failures. The diagram unveils the lifecycle of a desired product, but we are highlighting the system test run immediately before a selected EST run. We take the last system run before a target EST because operation experts state that these system tests might have a direct correlation with those ESTs, since they happen one after another without any significant changes to the unit.

Figure 3.4 Lifecycle diagram of a desired product to select for prediction, highlighting the selected units.

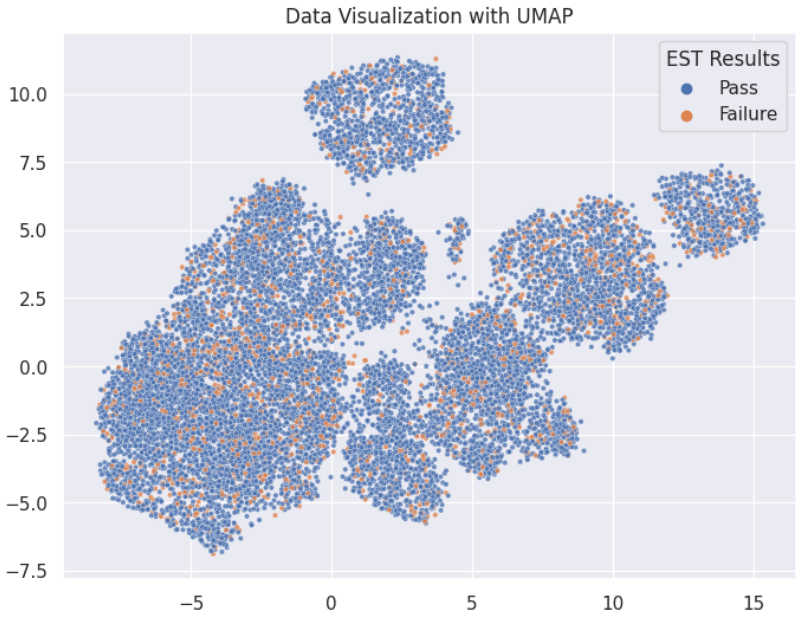


Such a way of test selection results in a single unit possibly feeding the final dataset for classification with more than one observation. Therefore, we have more observations than units analyzed. We can also induce by observing these scenarios and the available data that passing the system test does not automatically mean that the EST will pass. Therefore, this scenario most likely contains information to predict the outcome of the costly ESTs.

With the dataset at hand, Figure 3.5 displays the data distribution of the system test cases on a 2D space with UMAP, segregated by the EST outcome. We applied the system test case runs normalized in the form of a 2D matrix to the UMAP algorithm, which returned two arrays that could be plotted for analysis purposes. The plot indicates that the failure and the pass EST outcomes do not come from two distinct sources: we cannot trace a linear decision boundary that separates both classes perfectly. The failures are most likely anomalous behaviour of the system tests, but are intrinsically distributed among passes. This demonstrates the difficulty of our problem and dictates the methods that we selected to detect the failures with system test cases.

Furthermore, we built our dataset for failure prediction that consists of these pairs of System test case extracted features and the EST outcome label immediately after. The System test case features consist of tests performed on the components of a device for a specific amount of time. For example, some system test cases consist of the operator applying a tension in a component to measure the current that passed through that component for

Figure 3.5 UMAP visualization of the data.



a period of time. Then, they extract the maximum and minimum values of the current, and we define these values as features of our classification model. Other system test cases, however, might measure the potency, tension, optical measurements and other aspects of the plugged components. Therefore, after assembling all the available continuous system test cases, we have a data set containing 165 system test case features and 22620 observations. We have 2961 samples account for the failed EST tests with 13.1% of all observations while 19659 account for the normal passed EST tests with 86.9% of all observations available. Ultimately, some units have distinct system test cases applied than other units, even if the test cases are from the same PMA because these units come from distinct versions of the product. This results in missing values in the final data set, which accounts for no more than 10% of the available observations for each system test case in the worst scenario. And for these missing values, we imputed them with the median of the available system test case values.

Table 3.3 Example of the dataset table defined by previous descriptions.

TestRunID	Power_0	PowermW maximum value	PowermW minimum value	CurrX	...	EST Outcome
25	-9002	0.1298	0.1297	0.131	...	1
29	-9000	0.1297	0.1296	0.135	...	0
33	-8997	0.1299	031298	0.134	...	0
...	...	...	...	...	...	

Finally, Table 3.3 represents an example of the dataset used to perform the EST failure detection. The column *TestRunID* represents the unique identifier of the system test case features of that observation, and are not used for classification. The EST outcome represents the target label outputs of the ESTs. And the other columns are examples of the test cases used as the features of our projects. Note that we can have two types of features. The first one is a unique output to assess the performance of a component (Power_0). The second one are minimum and maximum values of a stress applied to a component for a period of time. These two types of features exemplify the types of features used for this project.

CHAPTER 4 FAILURE DETECTION METHODOLOGY

After defining the background and context of this project, we ought to define the approaches we used to perform EST failure detection. In this chapter, we describe the methodology proposed to create the full classification pipeline for detecting the EST failures. We begin with the common preprocessing steps followed by the three preprocessing approaches proposed in this project: the absolute reconstruction error and latent space with autoencoders preprocessing, and the other approach without autoencoders. Moving on, we elaborate on the methodology employed for hyperparameter tuning using Bayesian optimization, and how it was merged with resampling and classification techniques. Lastly, we describe how we perform the threshold analysis to change the classification thresholds of the classifiers.

4.1 Early Preprocessing

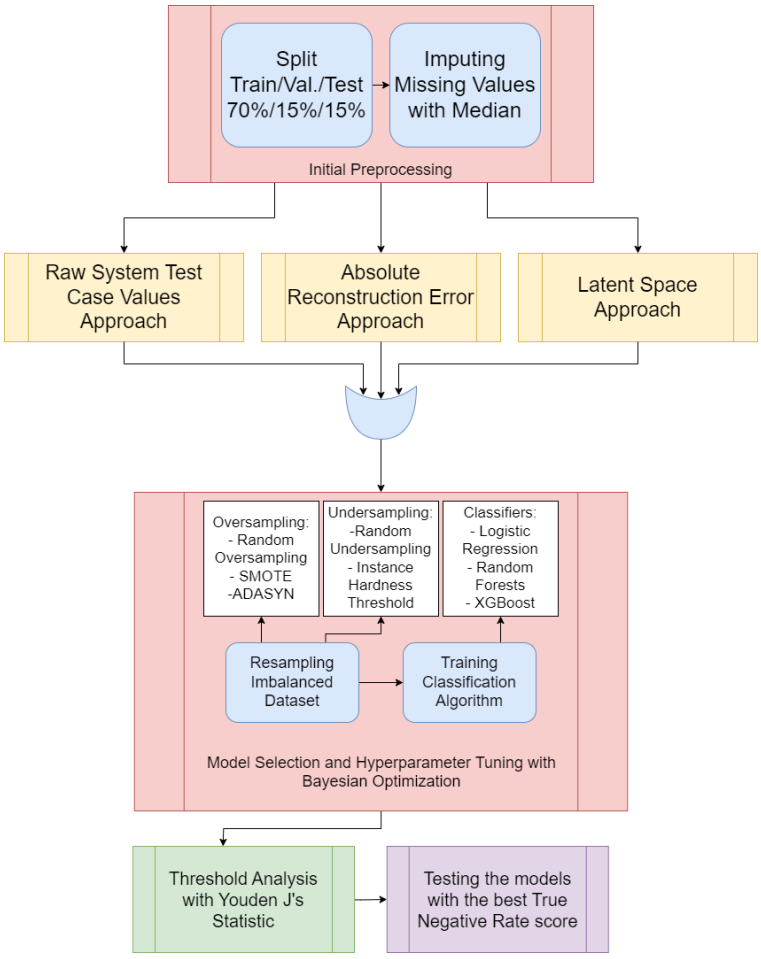
After defining the dataset with its nuances, considering the imbalance present and how we drew the pairs of features and targets, we ought to define the machine learning pipeline for creating an EST failure detector. Figure 4.1 defines the machine learning pipeline for creating a failure detector in our imbalanced dataset. It begins with a common preprocessing with data splitting and imputing missing values with the median to reduce the bias of outliers. Thereafter, we propose three different approaches for preprocessing the data towards failure detection: feature reconstruction with denoising autoencoders, using the latent space of the same autoencoder, and using the test case features purely to feed classification algorithms.

4.2 Absolute Reconstruction Error Autoencoder Preprocessing

The purpose of using autoencoders lies in appropriating from anomaly detection [14] methods to perform feature engineering and extract features that potentially have a better differentiation than using the original data. In this project, we train an autoencoder to reconstruct the EST passed data with the system test case values associated with it. We expect that, by training the autoencoder like this, the network will struggle to reconstruct failures. Thus, the difference between the reconstructed error and the original system case values would produce features that make the classifiers detect EST failure more easily. Therefore, the first approach with autoencoders consists of using the Absolute Reconstruction Error of the autoencoder as features.

The absolute reconstruction error consists in the absolute difference between the original

Figure 4.1 The Methodology pipeline developed for this project.



input x and the reconstructed input $\hat{x}$. Equation 4.1 describes how we computed the absolute reconstruction error.

$$RE = |\hat{x} - x| \tag{4.1}$$

We first normalize the original input features with the standard scaler to set all features in comparable scales even after the reconstruction. Equation 4.2 describes the normalization performed on the original data. This method assures that all the features have the same scale, with a mean equals to 0 and a standard deviation of 1. μ and σ refers to the mean and standard deviation of the i-th feature.

$$x^i_{normalized} = \frac{x^i - \mu}{\sigma} \tag{4.2}$$

4.3 Autoencoder Latent Space Preprocessing

The latent space approach, however, takes a step back into the autoencoder. We use the same network trained with the normalized input features. But instead, we use the encoded data from the encoder block, normalize it again with the standard scaler, since it does not come from any activation function from the network and to guarantee that all features from the reduced dimensionality have a similar scale and do not affect the classification algorithms. And finally, train the classification algorithms with the normalized latent space features.

The latent representation features associated with the autoencoder training with only EST passes are an important method of extracting features for training classifiers. They replace extensive feature engineering processes and have a performance comparable to handcrafted features extracted from other algorithms [13]. But also, we can interpret the use of latent space as a dimensionality reduction method, which projects the original data into a reduced space that distinct the EST failures and passes better. Therefore, we used this representation on our project to extract features from the system test case values that are useful for predicting the EST failures.

4.4 Minimal Preprocessing

The other approach consists of preprocessing the system test cases with the standard scaler only, without any autoencoder preprocessing. Afterwards, we used this data to train the classification algorithms to detect the EST failures. With that, we moved on to the next step to deal with class imbalance on the dataset.

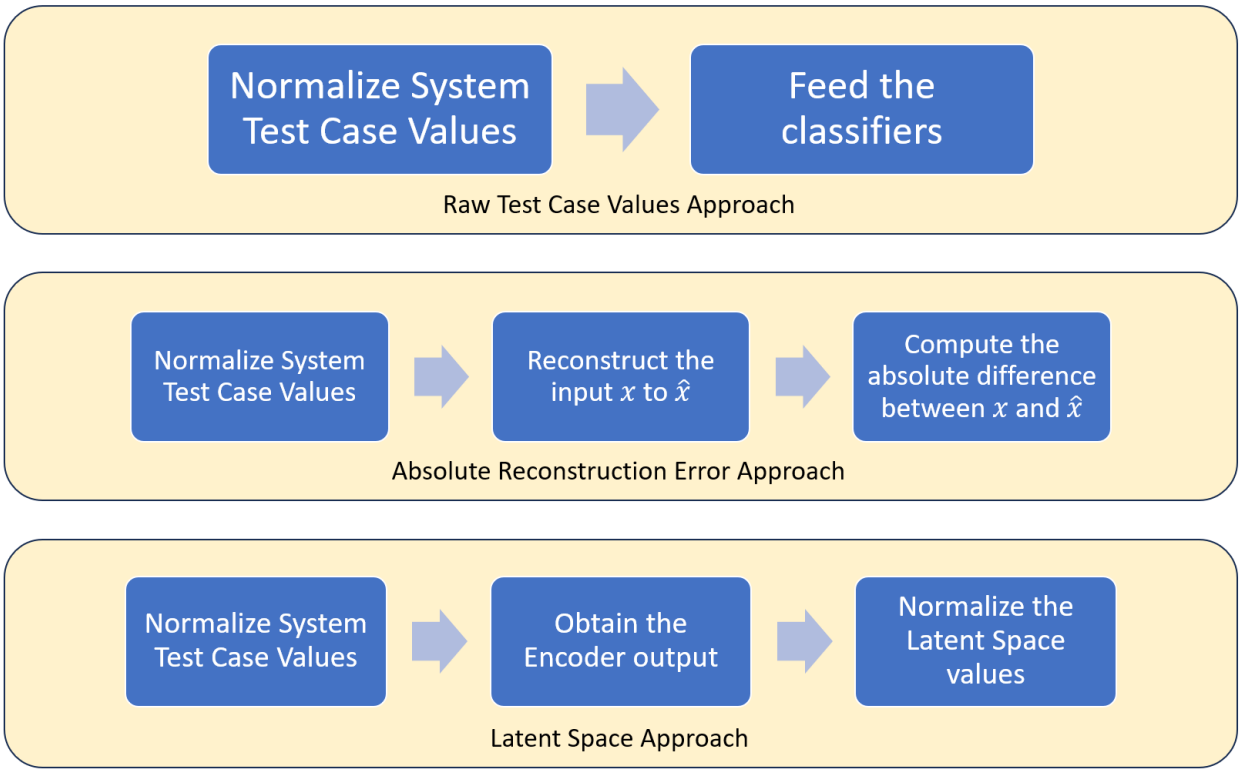
The main objective of using this approach is to have a performance baseline to compare with the autoencoders processing. And with that, gather enough evidence to estimate if our proposed methods with autoencoder are useful for improving the failure detection rate of EST outcomes.

Finally, Figure 4.2 highlights the steps of each of the proposed preprocessing approaches.

4.5 Imbalanced Failure Detection and Hyperparameter Tuning

Before feeding any data points to the classifiers, we resample our data to tackle the imbalance issue from the EST failures. We perform the resampling to balance the failure and pass classes and reduce the bias towards the EST passes, since they are the majority class. It consists of a process of first undersampling the majority class to a factor of the minority class, and then oversampling the minority class to that specified factor. We used this approach to

Figure 4.2 Proposed preprocessing approaches and the steps necessary to complete them.



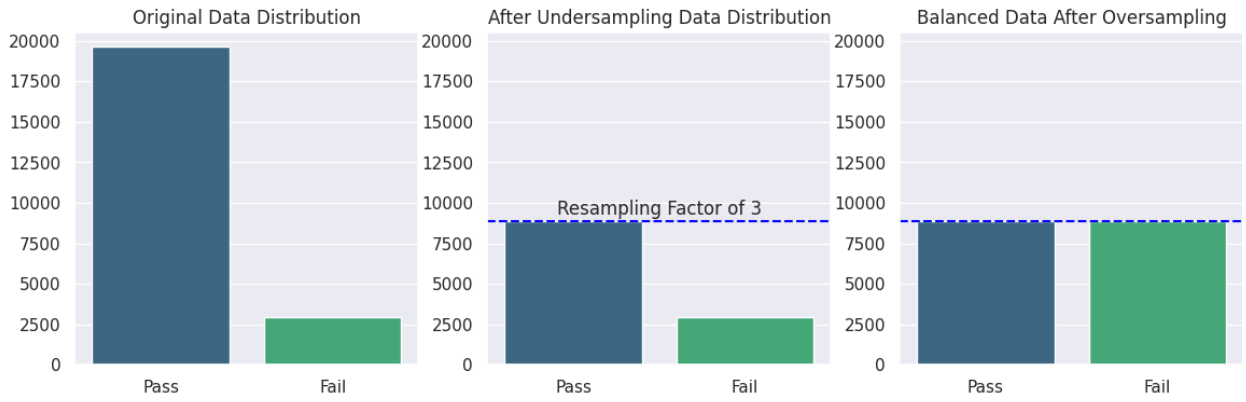
compensate for the imbalance of 13.1% of failures present in the data.

This resampling factor consists of the proportion of samples of the minority class used as the number of samples to down sample from the majority class. Hence, if we have β samples from the minority class and the resampling factor is equal to 3, we would have 3β as the number of samples from the majority class with $\alpha >> \beta$ samples, thus reducing the number of samples from α to 3β. After undersampling, we propose to oversample the minority class to this factor of 3β.

Figure 4.3 describes the resampling process to balance the data, with a resampling factor of 3, $\alpha = 19659$ and $\beta = 2961$. By the end, we would have 8883 samples from each class to train the classifiers.

Then, we use the resampled data with this method to train a classifier. We selected for this project the classifiers *Logistic Regression*, *Random Forest* and *XGBoost*. We selected these models due to their simplicity and usability in failure prediction models. Finally, we selected the best performing model after a hyperparameter tuning and moved on to the final inference step.

Figure 4.3 Resampling methodology for balancing the data. It consists of a process of under sampling the majority class first, and then over sampling the minority class.



4.6 Hyperparameter Tuning

The next step consists of performing a hyperparameter tuning with Bayesian optimization. We ought to find a configuration of hyperparameters that produce a classifier that maximizes the performance of the ROC-AUC score on the validation dataset. We search for hyperparameters on the resampling and classification processes only. Hence, we chose the ROC-AUC score to maximize during the model selection phase because it reflects how well the classifiers are confident in predicting if a product will fail or pass without the need of a predefined fixed threshold value.

In this phase, we compare the performances of the models for each one of the three preprocessing approaches. This will be our comparison base to analyze the impact of the later threshold analysis. The performances to analyze are the metrics: precision, recall, F1-Score, negative precision and accuracy.

4.7 Threshold Analysis

Furthermore, we need to consider the impact of predictions of the selected model and the errors associated with the predictions, and the selection of the evaluation metric on the final model is utterly important. A false positive represents wrongly classifying a unit as a failure when in fact it would pass the EST PMA, while a false negative represents predicting that a product would pass when in fact it would fail the EST PMA. A false positive would result in the operators performing unnecessary repairs and revisions, and the false negatives result in not performing the necessary revision and repairs in the case of a failure, thus

possibly propagating the error forward to the final consumer and causing even more costs to the manufacture than performing revisions and repairs without necessity. Therefore, the purpose of producing the predictor is aiding the operators during production to increase their productivity by giving them more confidence in performing less EST PMA runs to guarantee the final quality of the telecommunication products. And to attenuate the impact of the false negative, we ought to create a model whose most important task is to confidently predict normal behaviour more often than correcting classifying a failure with a high negative precision.

However, simply maximizing the negative precision while making model selection may result in inconsistent models. For instance, if a model assumes that the majority of the validation samples are failures without any False Positives, it would possess an undefined negative precision or a negative precision equal to 1. This would also affect the other performance metrics, such as precision and recall. Hence, it would not be much different than the current approach of naively assuming that all the samples are failures. We need to define an approach that enhances the performance of the classification models on the negative precision without affecting significantly the other metrics and not changing the hyperparameter values of the classifiers.

A common approach to optimizing the classification performance is changing the decision threshold of a classifier. This involves manually changing the threshold value of a probabilistic classifier to maximize the performance of the model in binary classification settings. For example, if we have a EST pass unit classified as a failure with 55% of chance, a threshold of 50% would make this sample classified as a failure, a False Positive observation. However, if we change the threshold to 60%, the same unit would be classified as a normal observation, thus a True Negative. Hence, deciding the threshold is utterly important to maximize failure detection performance on our proposal.

A popular method of selecting a threshold is based on the Youden's J statistic [10]. This approach consists in selecting a threshold that maximizes the sum between the precision and the specificity minus one from different threshold extracted from a ROC-curve. Equation 4.3 describes the J-statistic computation.

$$J = Recall + Specificity - 1 \tag{4.3}$$

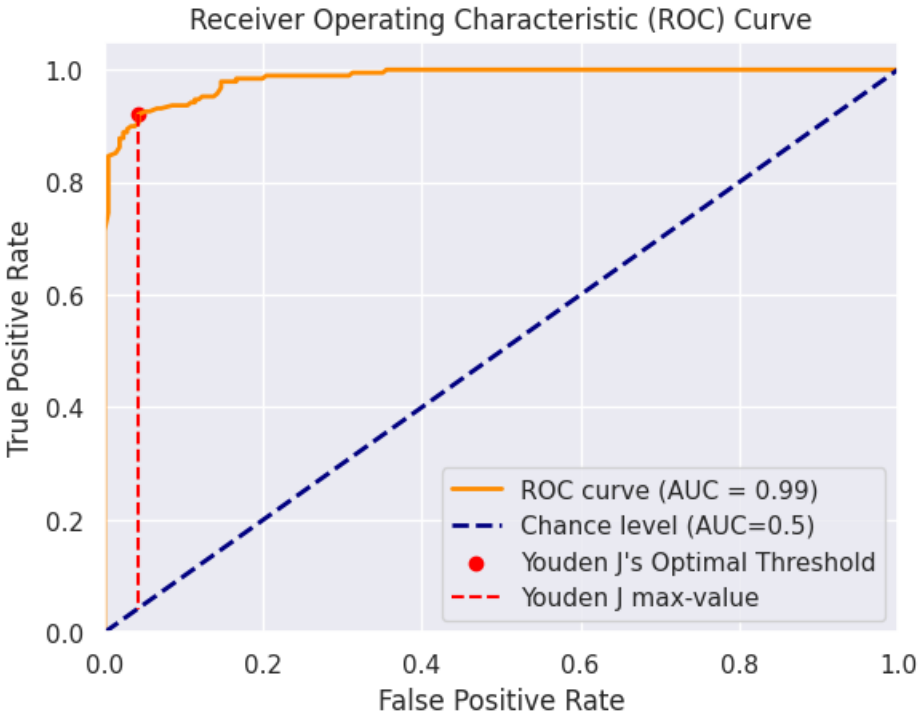
The specificity, in fact, can be rewritten as $1 - FalsePositiveRate$, and the recall as *True Positive Rate*. These two metrics are calculated when computing the ROC-curve, which gives these two metrics and the thresholds as vectors of the same size. For each False Positive Rate and True

Positive Rate, we have a threshold associated to it. This allows us to compute the Youden's J statistic for each threshold and re-write Equation 4.3 as:

$$J = TruePositiveRate - FalsePositiveRate \tag{4.4}$$

Therefore, our approach consists in finding the classification threshold from the ROC-Curve that maximizes J. We can interpret J as the vertical distance between a random predictive model and a trained classification model ROC curves at a specific threshold [8]. Figure 4.4 demonstrates the maximal Youden J's value in a ROC curve of a random forest classifier. Therefore, we expect that this approach for defining the threshold achieves the best performance in detecting failures [10].

Figure 4.4 Youden J's statistic demonstration on a ROC curve.



Furthermore, we propose performing the threshold analysis for each one of the approaches selected to preprocess the data. After obtaining the optimal hyperparameters with the Bayesian optimization, we retrain the models with this optimal configuration. And we use the validation set to define optimal classification thresholds with the Youden J's approach, followed by selecting the models for each preprocessing with the highest Negative Precision value. Finally, we assess the performance of the selected models for each preprocessing approach with the test set data.

CHAPTER 5 EXPERIMENTS, RESULTS AND DISCUSSION

After a thorough explanation of the context of the project, the dataset and the methodology employed, in this chapter we are going to present the experimental setup and discuss the results obtained.

5.1 Experimental Setup

The first step of our experimental configuration consists of splitting the data into training, validation, and test sets, considering the stratification of the EST failures proportion on the data. The training set consisted of 70% of the data, and we used it for training the models. The validation set, used for evaluating a trained model while performing the hyperparameter tuning, composed 15% of the data. Finally, the test data to evaluate the final performance of the selected models of each one of the three approaches also composed 15% of the data.

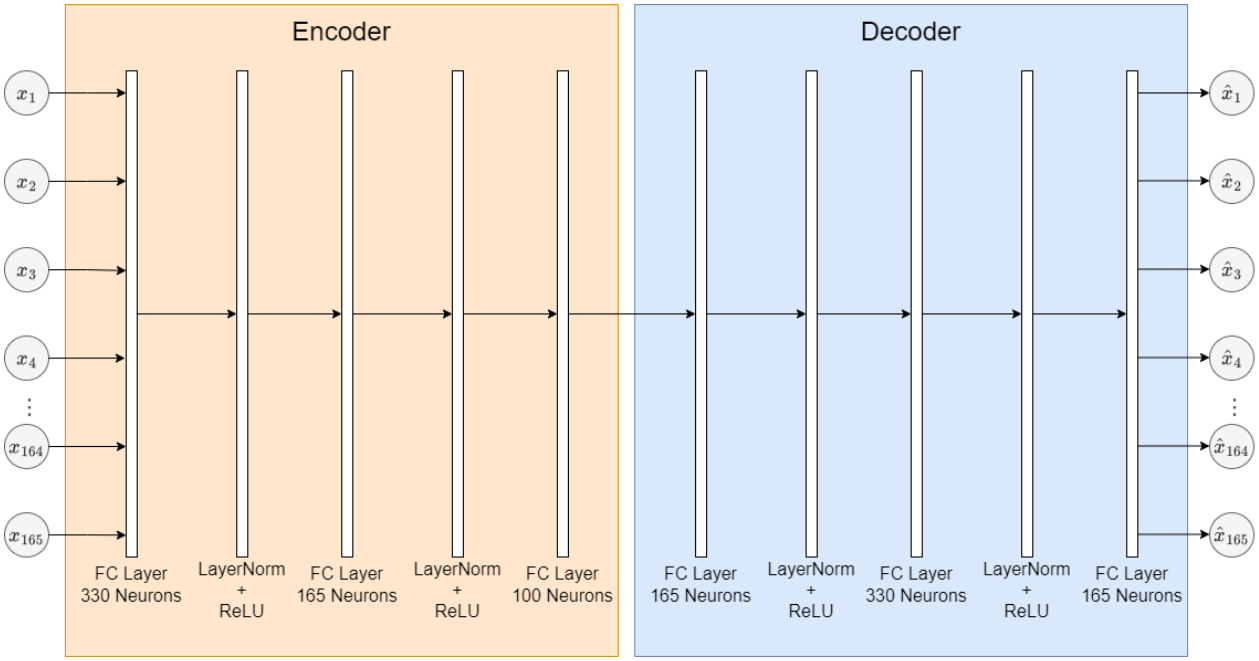
The next step consists of filling the missing values of each split with the median values of the training data. For each feature, we selected the median value of the non-missing values from the train set and filled the remaining ones for all three splits. Therefore, we now follow for the configuration of each approach.

5.1.1 Autoencoder Approaches

Before feeding the autoencoder network, we normalized the data with the standard scaler approach to have all features with the mean of 0 and a standard deviation of 1. Figure 5.1 displays the autoencoder configuration trained with only EST pass data. The autoencoder receives the normalized inputs x_1 up to x_{165} and feeds a *Fully Connected Layer* (FC Layer) with 330 neurons, the double amount of input features. Subsequently, the data undergoes a regularization procedure utilizing *LayerNorm*, followed by an activation function using *ReLU*. We then feed another FC Layer, followed the regularization and activation functions. Finally, we feed another FC Layer, but with only 100 neurons. Therefore, the Latent Space contains 100 features only. This sums up the encoder block of the autoencoder.

The decoder block takes the encoder output of 100 features and feed to another FC Layer, but with 165 neurons, and followed by the regularization and activation. Afterwards, we have an FC Layer of 330 neurons followed by regularization and activation. Finally, we reconstruct the input data with an FC Layer from $\hat{x}_1$ to $\hat{x}_{165}$. This sums up the decoder block and summarizes the architecture of the autoencoder network used in this project.

Figure 5.1 Autoencoder configuration for the preprocessing steps of Absolute Reconstruction Error and Latent Space.



The FC Layer is a perceptron layer based on Rosemblatt's Perceptron that performs the feedforward operation. Each neuron calculates a function of their inputs and moves forward the results to its successors [11]. In the proposed architecture, the results pass through the *LayerNorm* regularization and the *ReLU* activation function. The *ReLU* activation sets the neuron output to zero if their output is less or equal than zero, while uses an identity function for values greater than zero. Furthermore, the *LayerNorm* regularization performs a normalization over a batch of inputs for the outputs of a layer of the network.

We trained a single PyTorch network with these configurations to reconstruct the EST pass data for both approaches that use autoencoders. Furthermore, we trained the network for 200 epochs, with a batch-size of 64, and a learning rate of 0.0001. The main idea of these approaches lies in making the network reproduce perfectly normal pass data while increasing the reconstruction error of the system test cases associated with an EST failure. We set these parameters after exploring other paper architectures [6,14,49] and performing empirical trials with different model architectures, and selected the approach the one that most minimized the loss function value.

Since the reconstruction with autoencoders is a regression problem, we used the *Mean Squared Error Loss* function (*MSELoss*). Equation 5.1 displays the equation, which takes the input

feature x_j minus the reconstructed features $\hat{x}_j$ and squares it. Finally, it computes the average reconstruction for that entry for a batch of size B.

$$MSELoss = \sum_{i=1}^{B} \sum_{j=1}^{165} (x_{ij} - \hat{x}_{ij})^2 \tag{5.1}$$

5.1.2 Raw Test Case Values Approach

The only preprocessing required for using the classical approach in our project consists of normalizing the input data with the standard scaler method. And then feed the resampling and classification methods with this normalized data.

5.1.3 Hyperparameter Tuning Configuration

The hyperparameter tuning with Bayesian optimization enables to find the best configuration of the resampling and classification algorithms. Based on a metric, we select a configuration that maximizes this metric among different trials of the Bayesian optimization search. In this project, we selected the ROC-AUC score metric to select the best configuration during model selection. This metric measures how confident the selected model predicts whether a batch of samples is a failure or not based on the probabilistic output of the classifier. Values greater than 0.5 indicate that the models' predictions are better than random guessing.

Table 5.1 defines the hyperparameter search-space for the resampling algorithms. We selected the undersamplers: random undersampling and instance hardness threshold undersampler. Also, we searched through the over sampling algorithms of random over sampling, *SMOTE*, and *ADASYN*. Finally, we searched for a resampling factor value on a uniform distribution between 2.5 and 4.5.

Table 5.1 Configuration space of resampling hyperparameters.

Hyperparameter	Type	Search Space
Under Sampler	Categorical	['random', 'instance']
Over Sampler	Categorical	['random', 'smote', 'adasyn']
Resampling Factor	Float	[2.5, 4.5]

We used *SMAC3* package to perform Bayesian optimization in this project. It is a robust and flexible framework that it allows to improve performance with fewer trials. It hides unnecessary complexity to allow easy-to-use methods for different types of optimization, including hyperparameter tuning for model selection [51].

We selected the random forest and XGBoost classifiers due to their simplicity and higher performance in machine learning competitions. But we also selected the logistic regression as a baseline classifier. For each of the classifiers selected, we made an individual search of hyperparameters. We set the search time limit to **two** hours, with a maximum of 200 trials per classifier. Furthermore, we made a hyperparameter search through all three preprocessing approaches for the selected classifiers, accounting for a total of 9 Bayesian searches. This strategy allowed us to conduct a more balanced search for each of the classification algorithms based on the preprocessing step, thereby more confidently assuming that the configuration selected is globally optimal.

Table 5.2 defines the hyperparameter search space of the random forest classifier models. It contains the name of the hyperparameter, the type, and the search range of the variable. Therefore, Table 5.3 defines the search space for the XGBoost classifier models. Finally, the only hyperparameter searched for the logistic regression method was the $L2$ regularization term through the uniform interval of 0.0001 to 100.

Table 5.2 Configuration space of random forest classifier hyperparameters.

Hyperparameter	Type	Search Space
n_estimators	Integer	[100, 600]
max_depth	Integer	[6, 64]
criterion	Categorical	['gini', 'entropy']
min_samples_split	Integer	[2, 64]
min_samples_leaf	Integer	[1, 64]
max_samples	Float	[0.25, 1.0]

Table 5.3 Configuration space of XGBoost classifier hyperparameters.

Hyperparameter	Type	Search Space
n_estimators	Integer	[200, 1000]
max_depth	Integer	[5, 16]
eta	Float	[0.0001, 1.0]
gamma	Float	[0.0001, 1.0]
red_alpha	Float	[0.1, 200]
reg_lambda	Float	[0.1, 200]

5.2 Results and Discussion

Table 5.4 presents the results of the Bayesian search optimization for model selection. The values are the ROC-AUC results for the validation of the best hyperparameter configuration of each classifier according to each one of the three approaches for failure detection. Also, these values reflect the model's power in differentiating an EST failure and a pass on the validation dataset. For the current results of ROC-AUC on the validation set, the Latent Space approach performs better than the other approaches on this metric.

Table 5.4 ROC-AUC validation values of selected models from the Bayesian optimization.

Model	Raw Test Cases	Absolute Reconstruction Error	Latent Space
Logistic Regression	0.587	0.569	0.591
Random Forest	0.616	0.581	0.617
XGBoost	0.605	0.574	0.606

Table 5.5, however, displays the ROC-AUC values on the training set. We notice a discrepancy between the training and the validation scores, specially on the random forest models. The other models did not overfit as much to the training data. This difference reflects on the difficulty of the problem of identifying failures on the ESTs.

Table 5.5 ROC-AUC training values of selected models from the Bayesian optimization.

Model	Raw Test Cases	Absolute Reconstruction Error	Latent Space
Logistic Regression	0.623	0.646	0.625
Random Forest	0.996	0.712	0.954
XGBoost	0.785	0.705	0.823

The validation scores close to 0.5 also reflect on the difficulty of identifying the failures. This reinforces the hypothesis that the EST failures are not strongly co-related to the system test outcome values. Moreover, the validation score close to 0.5 reveals the performance close to random of our selected models. Again, this reinforces the difficulty of segregating the classes with a significant number of samples with a prediction probability close to 50% of being from either class.

Table 5.6 displays the results of the precision score for the selected models with the Bayesian optimization. Overall, the selected models with a threshold of 50% had a low precision on validation data, specially the Logistic Regression Models. When analyzing solely the

precision, the Absolute Reconstruction Error approach with autoencoders had the worst performance for any of the classification algorithms.

Table 5.6 Precision validation values of selected models from the Bayesian optimization.

Model	Raw Test Cases	Absolute Reconstruction Error	Latent Space
Logistic Regression	0.155	0.151	0.153
Random Forest	0.226	0.156	0.193
XGBoost	0.183	0.156	0.172

The Table 5.7 presents the recall results of the model selection. These results reflect on the capability of correctly classifying a failure as a failure. And the Logistic Regression Models performed the best. This happens exactly due to the difficulty of classifying the samples and the linear decision boundaries of the logistic regression. Thus, the classification regions of the logistic regression are not as tight as the other classifiers, increasing the number of samples classified as failures and less as normal passes. The random forest and XGBoost compensate for the lower recall by classifying correctly more samples as failures.

Table 5.7 Recall validation values of selected models from the Bayesian optimization.

Model	Raw Test Cases	Absolute Reconstruction Error	Latent Space
Logistic Regression	0.641	0.639	0.657
Random Forest	0.234	0.590	0.396
XGBoost	0.490	0.600	0.528

Table 5.8 displays the F1-Score results of the model selection, and represents the harmonic mean between the precision and the recall. Overall, the tree-based models had a better performance on this metric than the logistic regression for most of the preprocessing methods. By now, it's worth highlighting that using raw test case values or the latent space representation of the autoencoder trained with normal data results in a better performance. Though, the latent space had a slightly worse performance than the raw test cases.

Table 5.8 F1-Score validation values of selected models from the Bayesian optimization.

Model	Raw Test Cases	Absolute Reconstruction Error	Latent Space
Logistic Regression	0.250	0.244	0.248
Random Forest	0.230	0.247	0.260
XGBoost	0.266	0.248	0.260

Table 5.9 displays the negative precision results of the selected models with the Bayesian optimization. The negative precision values remained constant for all the preprocessing approaches and all the classification algorithms. With most of the values lying between 88% and 90%, our model strongly classifies an EST normal sample as an EST normal sample. Therefore, if the sample is classified as a normal sample, our approaches indicate that it has an 89% on average probability of being correctly classified, even before the threshold analysis.

Table 5.9 Negative Precision validation values of selected models from the Bayesian optimization.

Model	Raw Test Cases	Absolute Reconstruction Error	Latent Space
Logistic Regression	0.898	0.894	0.897
Random Forest	0.884	0.894	0.892
XGBoost	0.897	0.894	0.897

Finally, Table 5.10 unveils the accuracy results of the model selection on the validation dataset. These results indicate that the best approaches for a fixed classification threshold of 50% are using the raw test case values and using the latent space values. Furthermore, the best classification algorithm under those conditions is the random forest, achieving 79.4% and 70.4% for the first and third approaches respectively. The logistic regression models could not perform an accuracy better than 50%. And the XGBoost classifiers had a consistent performance for all preprocessing approaches, but not as good as the random forest.

Table 5.10 Accuracy validation values of selected models from the Bayesian optimization.

Model	Raw Test Cases	Absolute Reconstruction Error	Latent Space
Logistic Regression	0.497	0.482	0.477
Random Forest	0.794	0.529	0.704
XGBoost	0.645	0.523	0.606

5.3 Threshold Analysis Results

With the selected models at hand, we performed the threshold analysis. We used the *roc_curve* method from the *scikit-learn* package to obtain the TruePositiveRates, the FalsePositiveRates, and the thresholds associated with each of the first two metrics. Then, we obtained the Youden J's statistic value for each of the thresholds and selected the threshold that maximized the statistic. This will be the new threshold of each of the nine classifiers selected with the Bayesian optimization.

Table 5.11 contains the new threshold values for the classifiers after the maximization of the Youden J's statistic in percents. Note that for the logistic regression classifiers, the thresholds changed for a value above 50%, while the tree-based methods had their thresholds reduced to below the standard 50%. This means that the tree-based methods based on ensemble need fewer trees to determine a failure in a way that maximizes the difference between the precision and the false positive rate. Moreover, the logistic classification requires more confidence to identify a failure, thus classifying more samples as EST passes.

Table 5.11 Threshold value update after maximizing the Youden J's statistic.

Model	Raw Test Cases	Absolute Reconstruction Error	Latent Space
Logistic Regression	61.83%	59.39%	70.50%
Random Forest	40.35%	49.09%	44.68%
XGBoost	37.11%	49.51%	44.58%

Tables 5.12 display the optimized validation results of the selected models with the Bayesian optimization with the modified classification threshold compared to the standard results of the standard threshold. The precision of the Absolute Reconstruction Error with autoencoders increased for all the classifiers. Moreover, note that the performance of the logistic regression classifiers significantly increased for all the preprocessing approaches. However, the tree-based classifiers reduced the performance of the approaches using the raw test case values and the latent space with autoencoders.

Table 5.12 Precision values on the validation dataset after moving the threshold with the Youden J's analysis compared to the standard threshold.

Model	Raw Test Cases		Absolute Reconstruction Error		Latent Space	
	Standard	Optimized	Standard	Optimized	Standard	Optimized
Logistic Regression	0.155	0.171	0.151	0.164	0.153	0.181
Random Forest	0.226	0.171	0.156	0.161	0.193	0.170
XGBoost	0.183	0.168	0.156	0.159	0.172	0.168

Table 5.13 displays the standard and the optimized with Youden J's statistic recall score. The recall scores after moving the threshold also had a different behaviour to the precision scores. All the recall values from the logistic regression models reduced. While the random forest and the XGBoost classifiers increased their scores for all the preprocessing methods.

Table 5.14 displays the F1 scores before and after the threshold analysis. These values reflect the impact of performing the threshold analysis. The selected approach of using the maximized Youden J's statistic improved the F1-Score for all the preprocessing approaches.

Table 5.13 Recall values on the validation dataset after moving the threshold with the Youden J's analysis compared to the standard threshold.

Model	Raw Test Cases		Absolute Reconstruction Error		Latent Space	
	Standard	Optimized	Standard	Optimized	Standard	Optimized
Logistic Regression	0.641	0.506	0.639	0.556	0.657	0.455
Random Forest	0.234	0.609	0.590	0.612	0.396	0.644
XGBoost	0.490	0.663	0.600	0.612	0.528	0.624

Where the precision was reduced, the recall increased, and where the recall increased, the precision increased. Even though, the confidence of the resulting models on actually detecting failures improved.

Table 5.14 F1-Score values on the validation dataset after moving the threshold with the Youden J's analysis compared to the standard threshold.

Model	Raw Test Cases		Absolute Reconstruction Error		Latent Space	
	Standard	Optimized	Standard	Optimized	Standard	Optimized
Logistic Regression	0.250	0.255	0.244	0.253	0.248	0.259
Random Forest	0.230	0.268	0.247	0.255	0.260	0.269
XGBoost	0.266	0.268	0.248	0.253	0.260	0.264

Even though, we also need to analyze the performance on detecting actual normal EST passes with the negative precision, since False Negatives are more critical to the production pipeline. Table 5.15 displays the negative precision scores before and after the threshold analysis. The first thing we notice is that the improvement is not as significant as in the other metrics. We notice that using raw test case values and the latent space reduced the performance by a small amount for the logistic regression classifier. However, the performances for all the other preprocessing steps and classifiers improved, reaching even 90% in some cases. This means that those classifiers have approximately 90% probability of having a correct prediction of an EST pass when they classify a EST pass.

Table 5.15 Negative Precision values on the validation dataset after moving the threshold with the Youden J's analysis compared to the standard threshold.

Model	Raw Test Cases		Absolute Reconstruction Error		Latent Space	
	Standard	Optimized	Standard	Optimized	Standard	Optimized
Logistic Regression	0.898	0.894	0.894	0.895	0.897	0.894
Random Forest	0.884	0.904	0.894	0.899	0.892	0.907
XGBoost	0.897	0.908	0.894	0.897	0.897	0.904

The accuracy, in the end, was the most affected metric by the threshold analysis. Table 5.16 displays the comparison between the accuracies before and after the threshold analysis. The models that originally had an accuracy below 50% significantly improve the results, specially for the logistic regression classifiers. The results for the Absolute Reconstruction Error with autoencoders indicate that this approach was the one that benefitted the most, specially because for most of the classifiers the performance for all the metrics were improved. However, it was not better than any of the classifiers with the other preprocessing approaches.

Table 5.16 Accuracy values on the validation dataset after moving the threshold with the Youden J's analysis compared to the standard threshold.

Model	Raw Test Cases		Absolute Reconstruction Error		Latent Space	
	Standard	Optimized	Standard	Optimized	Standard	Optimized
Logistic Regression	0.497	0.613	0.482	0.570	0.477	0.659
Random Forest	0.794	0.563	0.529	0.532	0.704	0.542
XGBoost	0.645	0.525	0.523	0.525	0.606	0.544

However, the random forest and XGBoost classifiers were not affected as much. The reason possibly lies in how these algorithms compute the prediction probability of being from a class. Since they are ensemble methods, the most common way to classify in those methods is through majority voting. Therefore, originally, at least 50% of the estimators must classify a sample as an EST failure to cause the ensemble classifier to classify it as a failure. And by moving this threshold to bellow 50% we define that the model does not need to win the majority of votes of the estimators to be classified as a failure.

Furthermore, the threshold analysis significantly modified the behaviour of the selected models with the Bayesian optimization. It increased the F1-Score for all the classifiers in all the preprocessing approaches, which is the harmonic average between the precision and recall. When analyzing the metric of negative precision, performing this operation of moving the thresholds improved the metric and made our models less prone to wrongly classify an EST pass.

In this stage of the pipeline, we will select the best models for each approach based on the negative precision and analyze the results on the test dataset. We propose to analyze the performance of each preprocessing approach on the test set by the best classifier and define if our proposed preprocessing approaches had a significant impact on novel data used in production. By these definitions, we will use the raw test case values with the XGBoost classifier, the Absolute Reconstruction Error with autoencoder with the random forest classifier, and the latent space from the autoencoder with the random forest classifier as well.

Table 5.17 displays the results of all metrics for the best detectors on each preprocessing approach. Overall, using the latent space of an autoencoder trained only with EST pass data marginally improved the results from our baseline with only the normalized test case values. Only the recall metric got worse, with a difference of 0.007. But the other metrics were either equal or slightly improved. Therefore, we could achieve competitive results by this anomaly detection approach.

Table 5.17 Testing results on the best classifier of each approach according to the validation negative precision score.

Preprocessing	Precision	Recall	F1-Score	Neg. Precision	Accuracy
Raw Test Cases	0.175	**0.656**	0.276	**0.911**	0.549
Absolute Reconstruction Error	0.156	0.573	0.246	0.892	0.539
Latent Space	**0.176**	0.649	**0.277**	**0.911**	**0.556**

However, the Absolute Reconstruction error approach could not achieve competitive results. Our models could not grasp the reconstruction error features to differentiate an EST pass to a failure. Therefore, is not the most suitable approach to tackle our problem of detecting EST failures.

Finally, our results indicate that treating our problem as an anomaly detection problem with defined labels improved our failure detection performance. By training autoencoders with expected behaviour data, we could extract features from the original raw dataset and use it to train machine learning classifiers to predict EST failures. And, combined with a threshold analysis to change the classification threshold, we could achieve better results than using plain system test case values.

CHAPTER 6 CONCLUSION AND FUTURE WORK

In this project, we proposed a failure detection approach to predict failures during the manufacture of telecommunication devices. The project is a result of a partnership between *Ciena* and *Corail* (Combinatorial Optimization and Reasoning in Artificial Intelligence Laboratory) to perform intelligent data processing on the production pipeline and optimize the processes. Hence, we defined a failure detection approach to detect EST failures during the manufacture of Ciena devices. The interest in this type of test comes from the high material and time-consuming costs of performing EST tests. And since the failures are rare events, our approach had to take dataset imbalance into account.

First, we defined our dataset, which consists of system test run result values done exactly before an EST as the features. The target label consists of the outcome of those ESTs, and we define it as a binary classification problem of failure and pass. With the dataset, we performed common preprocessing steps: we separated the dataset into training, validation and testing, and we imputed missing numbers with the median of each feature of the training dataset.

With that, we proposed three approaches to perform EST failure detection. The first one consists of using the raw test case feature values normalized with a standard scaler. The second and third approaches, however, assume the failures as anomalies. Therefore, we trained an autoencoder to reconstruct the input features associated with an EST pass only. And with the trained autoencoders, we used two distinct ways to extract features: using the encoder output known as latent space, or the absolute reconstruction error of the outputs of the network.

We then selected the classifiers of logistic regression, random forest and XGBoost to perform the failure detection. And to deal with the data imbalance, we used resampling techniques to balance the data and allow the identification of the rare failures more often. Therefore, we performed a Bayesian optimization to search for the best hyperparameter configuration of the classifier and the resampling techniques, selecting the model with the best ROC-AUC validation score in the end.

The results indicate that raw system test cases and the latent space had competitive results, with a worse performance from the latter. One of the possible interpretations for latent space is to consider it as a dimensionality reduction approach of the input features. However, by training the autoencoder with EST pass samples, we extracted useful features and applied it to a classification algorithm. We noticed that the latent space approach had a better

performance on the recall and F1-Score metrics, but were inconsistent on the other metrics when compared to the raw system test case features. Therefore, without any modification to the classification threshold of the models, the first approach with system test case values performed better.

Following with the threshold analysis, we selected a classification threshold from a ROC curve that maximizes the Youden J's statistic value. By maximizing this statistic, we attenuated the number of either false positives or false negatives, and improved the performance of the classifiers for all the preprocessing proposals. The most affected by this methodology was the absolute error reconstruction with autoencoder, which had all the performance metrics improved with few exceptions. But even with this improvement on the results of the absolute reconstruction error, the classification threshold was the least affected when compared to the other methods. And had the final threshold closer to 50% for all the classification models.

However, even after the threshold optimization with the Youden J's statistic, the absolute reconstruction error with autoencoders was not the best model in the end. The latent space and the raw system test case approaches were the best approaches on the validation and testing approaches, with a small advantage of the latent space approach. The negative precision metric results, important for the good workflow of the production pipeline, were competitive. And after applying the threshold moving methodology, the results were even better while also achieving decent results on actually detecting EST failures.

Finally, the testing results demonstrate that using the latent space with an autoencoder trained with EST passes achieves better results for our problem. By training this way, we could highlight to our classifiers the differences between failures and passes clearer than using raw system test cases, even though the improvement was marginal. Thus, we gathered more evidence that treating the failures as anomalies and using anomaly detection methods is a feasible way to achieve better results in imbalanced settings for detecting environmental stress test outcomes.

6.1 Limitations and Future Work

The most noticeable limitation is the final performance of the classification models, no matter the preprocessing approach. This comes from the difficulty of detecting EST failures on this problem. As noticed in Figure 3.5, the EST failures are not easily separable from the EST passes, which makes the classification even harder. Therefore, we need more descriptive features to improve the failure detection rate.

Another limitation is the number of failures available on the dataset. The failures account

for only 13.1%, and even with the resampling techniques employed, the performances were still not competitive with other real-world problems.

Therefore, future work would need to make use of more features from the production pipeline to make more descriptive features of the environmental stress tests. Possible feature usages include the time it takes to complete each system test case, and the PartNumber associated with that unit at that time of production.

With the new features at hand, the curse of dimensionality would most likely affect the performance of the model. Therefore, future work includes employing feature selection methods, such as using feature importance from tree-based methods, recursive feature selection methods, or selecting features that maximize mutual information statistics with the target labels.

Furthermore, explainability is another approach to tackle. Understanding the system case features more correlated to the EST outcome is a crucial procedure on understanding the production pipeline and finding ways to improve it. Therefore, we ought to explore different methods to find explainability globally and locally. Some methods include using Shapley values derived from game theory to obtain explainability, as using classification models with feature importance or coefficient values associated to the system test cases.

Another approach to tackle is the threshold analysis. Currently, our approach uses the ROC-curve to select the optimal classification threshold with the Youden J's statistic. Another approach to explore is to use the Precision-Recall curve to select a threshold that maximizes the F1-Score. Moreover, we could include the threshold selection in the Bayesian optimization process, setting the threshold value as a possible hyperparameter of the search.

REFERENCES

[1] L. Ren *et al.*, "A Wide-Deep-Sequence Model-Based Quality Prediction Method in Industrial Process Analysis," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 9, pp. 3721–3731, 2020.

[2] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Reference Format*, vol. 41, no. 15, 2009. [Online]. Available: http://doi.acm.org/10.1145/1541880.1541882

[3] A. A. A. Mohd Amiruddin *et al.*, "Neural network applications in fault diagnosis and detection: an overview of implementations in engineering-related systems," *Neural Computing and Applications*, vol. 32, no. 2, pp. 447–472, 2020. [Online]. Available: https://doi.org/10.1007/s00521-018-3911-5

[4] M. Said Elsayed *et al.*, "Network anomaly detection using lstm based autoencoder," in *Proceedings of the 16th ACM Symposium on QoS and Security for Wireless and Mobile Networks*, 2020, pp. 37–45.

[5] N. Renström, P. Bangalore, and E. Highcock, "System-wide anomaly detection in wind turbines using deep autoencoders," *Renewable Energy*, vol. 157, pp. 647–659, 2020.

[6] C.-W. Tien *et al.*, "Using autoencoders for anomaly detection and transfer learning in iot," *Computers*, vol. 10, no. 7, p. 88, 2021.

[7] S. Debroy and A. Sil, "An apposite transfer-learned dcnn model for prediction of structural surface cracks under optimal threshold for class-imbalanced data," *Journal of Building Pathology and Rehabilitation*, vol. 7, no. 1, p. 83, 2022.

[8] H. Hellström *et al.*, "Classification of head and neck cancer from pet images using convolutional neural networks," *Scientific Reports*, vol. 13, no. 1, p. 10528, 2023.

[9] B. Lu, D. Xu, and B. Huang, "Deep-learning-based anomaly detection for lace defect inspection employing videos in production line," *Advanced Engineering Informatics*, vol. 51, p. 101471, 2022.

[10] W. J. Youden, "Index for rating diagnostic tests," *Cancer*, vol. 3, no. 1, pp. 32–35, 1950.

[11] S. J. Russell and P. Norvig, *Artificial Intelligence: a modern approach*. Pearson, 2021, ch. 21, pp. 653–656,696–702,776–779.

[12] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785–794.

[13] W. H. L. Pinaya *et al.*, "Autoencoders," in *Machine learning*. Elsevier, 2020, pp. 193–208.

[14] S. Kauschke, M. Muhlhauser, and J. Furnkranz, "Towards Semi-Supervised Classification of Event Streams via Denoising Autoencoders," *Proceedings - 17th IEEE International Conference on Machine Learning and Applications, ICMLA 2018*, no. 2014, pp. 131–136, 2018.

[15] C. Zhou and R. C. Paffenroth, "Anomaly detection with robust deep autoencoders," *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, vol. Part F129685, pp. 665–674, 2017.

[16] L. McInnes, J. Healy, and J. Melville, "Umap: Uniform manifold approximation and projection for dimension reduction," *arXiv preprint arXiv:1802.03426*, 2018.

[17] C. C. Aggarwal, *Data Mining: The Textbook*. Cham: Springer, 2015.

[18] G. Du *et al.*, "Joint imbalanced classification and feature selection for hospital readmissions," *Knowledge-Based Systems*, vol. 200, p. 106020, 2020. [Online]. Available: https://doi.org/10.1016/j.knosys.2020.106020

[19] A. Sridee and P. Chongstitvatana, "Machine Learning Techniques to Detect Failure in Hard Disk Drive Test Process," *6th International Conference on Information Technology, InCIT 2022*, pp. 152–156, 2022.

[20] W. Qin *et al.*, "A hybrid multi-class imbalanced learning method for predicting the quality level of diesel engines," *Journal of Manufacturing Systems*, vol. 62, no. August 2020, pp. 846–856, 2022. [Online]. Available: https://doi.org/10.1016/j.jmsy.2021.03.014

[21] D. Zhang, B. Xu, and J. Wood, "Predict failures in production lines: A two-stage approach with clustering and supervised learning," *Proceedings - 2016 IEEE International Conference on Big Data, Big Data 2016*, pp. 2070–2074, 2016.

[22] Q. Xu *et al.*, "Imbalanced fault diagnosis of rotating machinery via multi-domain feature extraction and cost-sensitive learning," *Journal of Intelligent Manufacturing*, vol. 31, no. 6, pp. 1467–1481, 2020. [Online]. Available: https://doi.org/10.1007/s10845-019-01522-8

[23] V. Venkatasubramanian *et al.*, "A review of process fault detection and diagnosis," *Computers & Chemical Engineering*, vol. 27, no. 3, pp. 293–311, Mar. 2003.

[24] A. A. A. M. Amiruddin *et al.*, "Neural network applications in fault diagnosis and detection: an overview of implementations in engineering-related systems," *Neural Computing and Applications*, vol. 32, no. 2, pp. 447–472, Dec. 2018.

[25] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[26] M. R. Smith, T. Martinez, and C. Giraud-Carrier, "An instance level analysis of data complexity," *Machine Learning*, vol. 95, no. 2, pp. 225–256, 2014.

[27] N. V. Chawla *et al.*, "SMOTE: Synthetic Minority Over-sampling Technique Nitesh," *Journal of Artificial Intelligence Research*, vol. 16, no. Sept. 28, pp. 321–357, 2002. [Online]. Available: https://arxiv.org/pdf/1106.1813.pdf%0Ahttp://www.snopes.com/horrors/insects/telamonia.asp

[28] H. He *et al.*, "ADASYN: Adaptive synthetic sampling approach for imbalanced learning," *Proceedings of the International Joint Conference on Neural Networks*, no. 3, pp. 1322–1328, 2008.

[29] T. Yu and H. Zhu, "Hyper-Parameter Optimization: A Review of Algorithms and Applications," pp. 1–56, 2020. [Online]. Available: http://arxiv.org/abs/2003.05689

[30] J. Wu *et al.*, "Hyperparameter optimization for machine learning models based on Bayesian optimization," *Journal of Electronic Science and Technology*, vol. 17, no. 1, pp. 26–40, 2019. [Online]. Available: https://doi.org/10.11989/JEST.1674-862X.80904120

[31] C. Marzban, "The roc curve and the area under it as performance measures," *Weather and Forecasting*, vol. 19, no. 6, pp. 1106–1114, 2004.

[32] A. P. Bradley, "The use of the area under the roc curve in the evaluation of machine learning algorithms," *Pattern recognition*, vol. 30, no. 7, pp. 1145–1159, 1997.

[33] I. Ramos-Pérez *et al.*, "When is resampling beneficial for feature selection with imbalanced wide data?" *Expert Systems with Applications*, vol. 188, p. 116015, 2022. [Online]. Available: https://doi.org/10.1016/j.eswa.2021.116015

[34] I. Kononenko, "Estimating attributes: Analysis and extensions of relief," in *European conference on machine learning*. Springer, 1994, pp. 171–182.

[35] I. Guyon *et al.*, "Gene selection for cancer classification using support vector machines," *Machine learning*, vol. 46, pp. 389–422, 2002.

[36] F. Liu and Y. Dai, "Product Processing Quality Classification Model for Small-Sample and Imbalanced Data Environment," *Computational Intelligence and Neuroscience*, vol. 2022, 2022.

[37] H. Zhou *et al.*, "A Hybrid Feature Selection Method RFSTL for Manufacturing Quality Prediction Based on a High Dimensional Imbalanced Dataset," *IEEE Access*, vol. 9, pp. 29 719–29 735, 2021.

[38] C. Nunes *et al.*, "A Machine Learning Tool to Monitor and Forecast Results from Testing Products in End-of-Line Systems," *Applied Sciences (Switzerland)*, vol. 13, no. 4, 2023.

[39] T. Wuest, C. Irgens, and K. D. Thoben, "An approach to monitoring quality in manufacturing using supervised machine learning on product state data," *Journal of Intelligent Manufacturing*, vol. 25, no. 5, pp. 1167–1180, 2014.

[40] S. Schorr *et al.*, "Quality prediction of drilled and reamed bores based on torque measurements and the machine learning method of random forest," *Procedia Manufacturing*, vol. 48, pp. 894–901, 2020. [Online]. Available: https://doi.org/10.1016/j.promfg.2020.05.127

[41] H. C. Rhim *et al.*, "Defining clinically meaningful cut points for leg power impairment using physical performance in older adults: A secondary analysis from boston rise," *Archives of Physical Medicine and Rehabilitation*, 2023.

[42] I. R. de Vries *et al.*, "Fetal electrocardiography and artificial intelligence for prenatal detection of congenital heart disease," *Acta obstetricia et gynecologica Scandinavica*, vol. 102, no. 11, pp. 1511–1520, 2023.

[43] H. Darvishi *et al.*, "Sensor-fault detection, isolation and accommodation for digital twins via modular data-driven architecture," *IEEE Sensors Journal*, vol. 21, no. 4, pp. 4827–4838, 2020.

[44] S. Dengler *et al.*, "Applied machine learning for a zero defect tolerance system in the automated assembly of pharmaceutical devices," *Decision Support Systems*, vol. 146, p. 113540, 2021.

[45] M. Liu *et al.*, "Gated Transformer Networks for Multivariate Time Series Classification," mar 2021. [Online]. Available: http://arxiv.org/abs/2103.14438

[46] G. Zerveas *et al.*, "A Transformer-based Framework for Multivariate Time Series Representation Learning," *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 2114–2124, 2021.

[47] A. Vaswani *et al.*, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon *et al.*, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[48] J. G. Zhang, J. P. Li, and H. Li, "Language Modeling with Transformer," *2019 16th International Computer Conference on Wavelet Active Media Technology and Information Processing, ICCWAMTIP 2019*, pp. 249–253, 2019.

[49] J. Kwon, M. G. Seok, and D. Park, "Neural Network-based Approximate Quality Prediction for Parameter Exploration in Industrial Manufacturing," *2022 International Symposium on Intelligent Signal Processing and Communication Systems, ISPACS 2022*, pp. 1–4, 2022.

[50] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[51] M. Lindauer *et al.*, "SMAC3: A Versatile Bayesian Optimization Package for Hyperparameter Optimization," *Journal of Machine Learning Research*, vol. 23, pp. 1–9, 2022.