Rodrigo Faria Gonçalves

Desenvolvimentos para a generalização dos modelos de Frustração Celular para deteção de anomalias

Towards a general Cellular Frustration approach to anomaly detection

Universidade de Aveiro
2021

Rodrigo Faria Gonçalves

Desenvolvimentos para a generalização dos modelos de Frustração Celular para deteção de anomalias

Towards a general Cellular Frustration approach to anomaly detection

"*You can call me anything you want, but don't ever call me a self-made man.*"

— Arnold Schwarzenegger, *UH Commencement Address*

Universidade de Aveiro
2021

Rodrigo Faria Gonçalves

Desenvolvimentos para a generalização dos modelos de Frustração Celular para deteção de anomalias

Towards a general Cellular Frustration approach to anomaly detection

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Computacional, realizada sob a orientação científica do Doutor Fernão Rodrigues Vístulo de Abreu, Professor auxiliar do Departamento de Física da Universidade de Aveiro.

Dedico este trabalho aos meus pais, pois a sua concretização deve-se em grande parte ao seu esforço e sacrifício para me darem essa oportunidade na vida.

o júri / **the jury**

presidente / president	Professor Doutor Tomás António Mendes Oliveira e Silva Professor Associado, Universidade de Aveiro
vogal / examiner	Doutor Bruno Filipe dos Santos Faria Cientista de Dados, Bosch Car Multimédia Portugal, S.A.
orientador / supervisor	Professor Doutor Fernão Rodrigues Vístulo de Abreu Professor Auxiliar, Universidade de Aveiro

agradecimentos / **acknowledgements**

O maior agradecimento de todos terá de ir para os meus pais, que me apoiaram desde o início e tiveram sempre confiança que seria capaz de ultrapassar quaisquer desafios para concluir o curso, tendo sido uma constante fonte de inspiração e motivação.

Devo um agradecimento importante ao meu orientador Professor Fernão Abreu pelo interesse e ajuda constante ao longo do projeto, mostrando-se sempre disponível não só para discutir assuntos pertinentes ao trabalho, mas também temáticas variadas da vida, sobre as quais deu conselhos que levo para a vida.

Durante o último ano em que a vida de todos sofreu muitas reviravoltas, pude contar com a amizade e convívio dos meus amigos da minha terra natal, que sem dúvida tornaram mais agradável a experiência de fazer uma dissertação remotamente, longe do ambiente académico a que estava acostumado, merecendo por isso um muito obrigado. Em especial agradeço à minha amiga de longa data, que fez questão de acompanhar o progresso do meu trabalho, por ter sido uma fonte de motivação e animação quando estas mais me faziam falta.

Um muito obrigado a Aveiro e à Universidade de Aveiro pelas inesquecíveis experiências de vida proporcionadas.

De modo geral agradeço aos meus colegas e amigos de engenharia pelo convívio, tanto em eventos sociais como em sessões de estudo, que tornou a experiência académica muito melhor.

palavras-chave

frustração celular, aprendizagem automática, aprendizagem sem supervisão, deteção de anomalia, *one-class support vector machines*, *isolation forest*, *k-means*.

resumo

Os Sistemas de Frustração Celular modelam interações entre agentes apresentadores e detetores, com o objetivo de concretizar deteções de anomalias em *data sets*. Estes dois tipos de agentes seguem uma dinâmica frustrada (i.e., instável), na qual continuamente trocam de agente do outro tipo com o qual estão emparelhados, quando uma amostra normal é apresentada pelos agentes apresentadores. De forma a que os SFCs consigam fazer deteções, os apresentadores têm de mostrar amostras anómalas que tenham uma ou mais características anómalas, o que leva a que os agentes emparelhem durante mais tempo, levando a emparelhamentos estáveis e consequentemente deteções.

Este trabalho melhora as versões anteriores do modelo ao permitir que os detetores vejam duas regiões do espaço das características como anómalas, com uma região normal entre elas. A técnica de *clustering K-means* também foi utilizada para agrupar dados nos *data sets*, para que os detetores consigam particionar o espaço das características e sejam atribuídos a uma certa região que verão como normal, enquanto que o resto verão como anómala. Mostra-se que não existe necessidade de treinar populações de detetores separadamente para fazer deteções em *data sets* que tenham amostras anómalas entre amostras normais.

A versão atual do modelo é comparada com versões prévias do modelo de Frustração Celular, e também com dois métodos bem conhecidos de deteção de anomalias, o *One-Class Support Vector Machines*, e o *Isolation Forest*. Os resultados mostram que tem desempenho equiparável em relação aos métodos concorrentes, enquanto que em relação às versões prévias, é capaz de atingir resultados equivalentes sendo um modelo mais robusto aplicável em mais situações com menos esforço.

Por fim, algumas ideias sobre trabalho futuro são discutidas de forma a que o modelo seja melhorado, pois ainda revela alguns problemas quando certas condições pouco favoráveis surgem em *data sets*.

keywords cellular frustration, machine learning, unsupervised learning, anomaly detection, one-class support vector machines, isolation forest, k-means.

abstract Cellular Frustrated Systems model the interactions between presenter and detector agents, with the goal of detecting anomalies in data sets. These two types of agents follow a frustrated dynamic (i.e., unstable), in which they continuously change the agent of the other type they are paired with, when a normal sample is presented by the presenter agents. In order for CFSs to make detections, presenters must show abnormal samples that have one or more abnormal features, which leads the agents to pair for longer times, leading to stable pairs, hence detections.

This work improves upon the previous versions of this model by allowing detectors to see two regions of feature space as abnormal, with a normal region of feature space in-between. The K-means clustering technique is also used to cluster data in data sets, so that detectors are able to partition the feature space and be assigned to a certain region which they will see as normal, with the rest being seen as abnormal. It is shown that there is no need to train separate populations of detectors in order to make detections with data sets that have abnormal samples in-between normal ones.

The current version of the model is compared with previous versions of the Cellular Frustration model, and also with two well known anomaly detection methods, the One-Class Support Vector Machines, and the Isolation Forest. The results show that it has comparable performance in relation to the competing methods, whereas regarding the previous versions, it is able to achieve the same results while being a more robust model applicable in more situations with less effort.

Finally, some ideas for future work are discussed in order to further improve the model, which still has some issues when certain unfavorable conditions arise in data sets.

Contents

List of Tables

List of Figures

List of Algorithms

CHAPTER 1

Introduction

Nowadays, machine learning (ML) is very popular for developing powerful algorithms that solve complex tasks and real world problems. These often involve processing large amounts of data to uncover useful information, which is a process called data mining. Applications of ML can be found in almost every professional field, from business to sports, academia to industry, robotics to healthcare.

Specifically, ML is a field of research that develops mathematical models capable of producing predictions. Part of the development process of a model requires adjusting parameters, and in order to do that, a lot of information must be fed to the model. This information comes from the real world in the form of data sets.

Since in the real world there are different kinds of situations, problems, tasks, that need to be dealt with, the need for different ML models arises. Some models are better for categorical prediction, and some for numerical prediction. Categorical prediction means trying to identify something as being part of a class of objects. This is the case, for instance, when trying to recognize different fruits while sorting them [1]. Numerical prediction tries to make a prediction regarding the evolution of a trend of data points by means of a mathematical technique called regression. For example, if a model can be adjusted to fit the data from the stock prices of some stock in the stock market, it will be able to predict the closing price of that stock [2].

One field of ML of particular interest is anomaly detection, which tries to tackle the problem of recognizing patterns in data that exhibit unexpected behavior. These patterns are often called anomalies or outliers, and these terms are used interchangeably. The importance of this field comes from the fact that anomalies in data are important information that can be acted upon in many application domains. [3]

There are many possible applications of anomaly detection. Some practical examples are: detecting malignant tumors through an MRI image [4]; recognizing an abnormal network traffic pattern because a hacked computer in a network of computers is sending information to an unauthorized destination [5]; preventing credit card fraud by detecting irregular transactions [6].

This work's contribution to the anomaly detection field consists in improving the cellular frustration (CF) model described in [7], by making it more general and accurate. Although this work builds on previous attempts in the same direction, more work is still re-

quired. CF algorithms still remain poorly studied despite good results for detecting anomalies in data sets, and for this reason, it is important to make these studies.

This work is organized as follows. In chapter 2 an overview of the most relevant techniques and concepts in ML related to this work are provided, particularly regarding anomaly detection. In chapter 3 the CF model will be explained in detail, setting clearly the terminology, definitions and concepts, in order to provide a clear picture of its inner workings and how it can be applied in anomaly detection scenarios. In chapter 4 the CF model developed in this work will be compared to its previous versions, and to other anomaly detection methods. To compare their performances, synthetic and real data sets will be used, with these results being discussed in detail. Finally, in chapter 5 a summary of the work done, and results obtained, will be made, along with a brief description of possible future work.

CHAPTER 2

Machine Learning Methods And Concepts

Machine learning is a multidisciplinary field with many fundamental concepts that must first be understood in order to apply ML techniques successfully, and some of these will be discussed in this chapter.

Although the focus of this work will be on anomaly detection, which is mostly associated with the ML subfield known as unsupervised learning, a brief overview of supervised learning and relevant techniques thereof will be provided.

Two of the most popular anomaly detection algorithms are explained in this chapter, and these are the One-Class Support Vector Machines (OCSVMs), and Isolation Forest (IF). The popular K-means clustering technique will also be explained. Finally, an overview of the most common ML techniques performance evaluation methods and concepts, will be described.

2.1 Data Sets, Samples, And Features

All measured real world information is stored in what is called a data set, which is a collection of related sets of information organized as a matrix, $\mathcal{X} = \{x_N = (v_1, \ldots, v_n) \forall S_u : u = 1, 2, \ldots, N\}$, as seen in Figure 2.1. Each set of information is called a sample, S, which has features, $F_f \forall f = 1, \ldots, n$, that characterize a measurement.

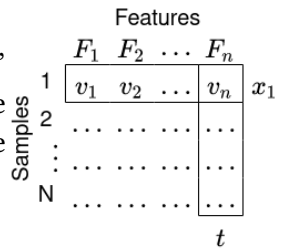


Figure 2.1: Visual representation of a data set, and how samples' feature values are stored in it. A data set is a matrix with each sample on a row, and values from different features on each column. The total number of samples is N_S.

There are two ways to store data in data sets. Labeled data has a column output vector of dependent target variables, $t = (t_1, \ldots, t_N)$, each with a corresponding input vector of independent variables, x_N. This column vector is the rightmost column of values in a data set. Unlabeled data does not have the labels t, having only the input vectors x_N.

2.2 Data Normalization

In ML it is often required to normalize data stored in a data set before using it. This aims to give equal importance to features and avoid biases regarding the way an algorithm weighs the importance of features in a data set. This is particularly important when there is no information beforehand regarding the order of magnitude of the data.

For instance, if a data set has features with values with very different orders of magnitude, perhaps with differences in the hundreds or thousands, when calculating how alike two data points are based on the distance between them, the algorithm would consider the values with greater orders of magnitude with a bigger weight, which would affect the result immensely. This is especially true in algorithms involving distance measurements for classification, regression or clustering. This can be solved with two normalization techniques commonly used in ML called min-max normalization and z-score normalization, although there are others. The description of these methods references the ones in [8].

The min-max normalization computes a linear transformation on a data set by mapping feature values, v_u, to a new range of values as follows $v_u \in [min_F, max_F] \rightarrow v'_u \in [min'_F, max'_F]$, where min_F and max_F are the minimum and maximum values of the original range, and min'_F and max'_F are the new range values, respectively. This is done by calculating

$$v'_u = \frac{v_u - min_F}{max_F - min_F}(max'_F - min'_F) + min'_F. \tag{2.1}$$

Notice there's never any issue while mapping the original values, but if a new value not initially present in the data set is mapped, there is a chance it will provoke an out-of-bounds error if it is less than min'_F or greater than max'_F. To solve this, one can simply map any values less than min'_F to min'_F and any values greater than max'_F to max'_F.

Another issue is that outliers could create distortions in the mapping of the normalized values. This happens because a lot of values are compressed in a small range because the normalization maps the outliers to the edge of the range and therefore must be farther from the majority of points. This required separation between points is what is considered a distortion, because it gives a sense that values are more closely related than they really are.

The z-score normalization is useful when the minimum and maximum values of a feature F are unknown, while also being robust to outliers if they are present. It requires the mean, μ, and standard deviation, σ, of each feature to normalize each feature accordingly. It normalizes a feature F's value, v_u, by calculating

$$v'_u = \frac{v_u - \mu_F}{\sigma_F}. \tag{2.2}$$

One issue with this normalization is that the range of the new values cannot be defined, but it will be around zero. This means that the order of magnitude of the values in different features can vary a little, when in reality they should all have the same order of magnitude.

2.3 Supervised Learning

Supervised learning is an approach to ML usually applied in classification or regression problems. Supervised methods rely on human input for labelling data sets correctly so that

the model can learn from it, and produce correlations between the labeled output vector of target variables, $t = (t_1, \ldots, t_n)$, and the corresponding set of input vectors, $\{x_n\}$, where $n = 1, \ldots, N$ is the n^{th} row of the data set used for training a model.

In regression and classification problems the common approach is to extract the input and corresponding output variables from a training set, which will be used to train a model that fits the training data, so that when a new observation is provided to the model it can accurately make a prediction. In regression a prediction is a real continuous value, while in classification it is a discrete binary value that labels the class of the prediction.

Some real world applications of supervised learning are: computer vision regarding object recognition [9]; classifying applicants for life insurance by doing a risk assessment [10]; predicting healthcare costs of individuals to help prioritize the allocation of care management resources [11].

Some problems with this approach are: human expertise required to set up models properly; higher chance of poorly trained algorithms due to human error when building training sets; unable to independently cluster or classify data. [12]

A summary of some fundamental concepts and methods of supervised learning will be provided, which describe how ML methods in general work, especially regarding the mathematical concepts behind them.

2.3.1 Linear Regression

The linear regression model coupled with the least-mean-squares (LMS) algorithm is probably one of the simplest and most well known approach to a regression problem. Its description references [13].

This model is given by linear combinations of nonlinear functions as

$$y(x, \omega) = \omega_0 + \sum_{j=1}^{M-1} \omega_j \phi_j(x),$$

where M is the number of parameters, $\phi_j(x)$ are basis functions, and ω_0 is a bias parameter that allows offsetting data. To simplify this formula, one can define an extra dummy basis function, $\phi_0(x) = 1$, so that it becomes

$$y(x, \omega) = \sum_{j=0}^{M-1} \omega_j \phi_j(x) = \omega^T \phi(x), \tag{2.3}$$

where $\omega = (\omega_0, \ldots, \omega_{M-1})^T$ and $\phi = (\phi_0, \ldots, \phi_{M-1})^T$. In this case the vector $\phi(x)$ of basis functions can be written as $\phi(x) = x$. Usually the $y(x, \omega)$ function is called a hypothesis function and will be used to make predictions regarding a target variable t.

To make predictions the model must be trained to compute values close to the ones in the output vector of the training set, which is to say $y(x, \omega) \approx t$ is the goal. To figure out how far a prediction is from the real value, the mean squared error (MSE) function is used, which is one of many possible cost functions, and it is given by

$$E(\omega) = \frac{1}{2N} \sum_{n=1}^{N} (t_n - \omega^T x_n)^2, \tag{2.4}$$

where N is the number of observations in the training set, and $\frac{1}{2}$ is a convenience factor.

The LMS algorithm is a sequential learning algorithm based on the technique of stochastic gradient descent that minimizes the cost function, which means it tries to approach the real value during training when computing the hypothesis function.

First the gradient of the cost function is needed, and it is given by

$$\nabla E(\omega) = \frac{\partial}{\partial \omega} E(\omega) = \frac{1}{N}(\omega^T x - t)x. \tag{2.5}$$

The idea behind this algorithm is that the model parameters should be continually updated until the cost function converges to a local or global minimum. This is done by updating the parameters vector ω using

$$\omega^{(\tau+1)} = \omega^{(\tau)} - \eta \nabla E(\omega) = \omega^{(\tau)} + \eta \frac{1}{N}(t_n - \omega^{(\tau)^T} x_n)x_n, \tag{2.6}$$

where τ is the iteration number, and η is the learning rate parameter. Note that the learning rate should be chosen adequately so that the algorithm converges fast. While a larger learning rate will lead to faster convergence, if it is too large the algorithm will diverge, therefore some trial and error may be required to find an optimal learning rate.

2.3.2 Logistic Regression

In classification several of the steps seen in linear regression are also taken. Although the idea here is to take an input vector and assign it a class in a set of discrete classes, C_k, where $k = 1, \ldots, K$. These classes are disjoint and each input is only assigned to one output class. The input space is divided into decision boundaries, which means that decisions are made based on where the hyperplane splits the data. The description of this method references [14].

When dealing with a probabilistic model in a two-class problem, it is common for the target variable to take the form of a binary value such as $t_n \in \{0, 1\}$, such that $t = 1$ represents the positive class, and $t = 0$ represents the negative class. The value of t can also represent the probability of the prediction being one or the other class.

In this situation, the hypothesis function used in the linear regression model no longer suits the model, mainly because it predicts values in a continuous range, well beyond the binary values considered here. Therefore, a new function called sigmoid, or logistic, function is chosen, because it outputs values between 0 and 1, which will be very useful for making probabilistic predictions regarding classes that are represented by binary values. This function is given by

$$\sigma(\omega^T x) = \frac{1}{1 + e^{-\omega^T x}}. \tag{2.7}$$

In this case, instead of minimizing a cost function the goal is to maximize it, therefore the stochastic gradient ascent algorithm will be used, which is quite similar to the previously discussed stochastic gradient descent algorithm in every aspect.

To get to the cost function, first the likelihood function must be defined, and it is given by

$$p(t_n | x_n, \omega) = \prod_{n=1}^{N} \sigma(\omega^T x_n)^{t_n} (1 - \sigma(\omega^T x_n))^{1-t_n}, \tag{2.8}$$

where $\sigma(\omega^T x_n) = p(t = 1|x_n, \omega)$. Then the cost function will be derived as

$$E(\omega) = \ln(p(t_n|x_n, \omega)) = \sum_{n=1}^{N}(t_n \ln(\sigma(\omega^T x_n)) + (1 - t_n) \ln(1 - \sigma(\omega^T x_n))), \tag{2.9}$$

and its gradient as

$$\nabla E(\omega) = \sum_{n=1}^{N}(\sigma(\omega^T x_n) - t_n)x_n. \tag{2.10}$$

Finally, just as in the linear regression case, this gives the rule that updates the parameters of the model as

$$\omega^{(\tau+1)} = \omega^{(\tau)} + \eta\nabla E(\omega) = \omega^{(\tau)} + \eta(t_n - \omega^{(\tau)^T} x_n)x_n. \tag{2.11}$$

2.3.3 Support Vector Machines

The Support Vector Machines (SVMs) method is an ML approach typically used in classification of both linear and nonlinear data, although it can also be used for regression.

Linearly Separable Data

For this description, [15] is referenced and the two-class classification problem is considered, which can be approached with the linear model

$$y(x) = w^T \phi(x) + b, \tag{2.12}$$

where $\phi(x)$ is a fixed feature space transformation, and b is the bias parameter. The training set has N input vectors $x_1, \ldots, x_N$, with corresponding target values $t_1, \ldots, t_N$, where $t_n \in \{-1, 1\}$, and $n = 1, \ldots, N$.

Given a training data set is linearly separable in feature space, there is at least one set of parameters w and b such that $y(x_n) > 0$ for points with $t_n = 1$, and $y(x_n) < 0$ for points with $t_n = -1$, and $t_n y(x_n) > 0$ for all training points. Since there can be many solutions, the SVM finds the best one by finding the one that will give the minimum classification error on new observations.

The way the SVM finds the best solution is by searching for the maximum margin hyperplane (MMH). This is the decision boundary with the largest distance between samples from different classes. First, one must know how to compute the distance from any point x_n to the decision surface, which is given by $\frac{t_n y(x_n)}{\|w\|}$. Note that all points satisfy the constraint $t_n(w^T \phi(x_n) + b) \geq 1$.

Since the distance between any support vector and the decision surface is $\frac{1}{\|w\|^2}$, the maximal margin is given by $\frac{2}{\|w\|^2}$. This makes it obvious that in order to maximize the maximal margin, $\|w\|^2$ must be minimized. Therefore, the objective function will be $\frac{1}{2}\|w\|^2$, still subject to the constraints mentioned previously, which is an example of a quadratic programming problem.

To solve this optimization problem, Lagrangian multipliers $a_n \geq 0$ will be used, which gives the following Lagrangian function

$$L(w, b, a) = \frac{1}{2} \|w\|^2 - \sum_{n=1}^{N} a_n(t_n(w^T \phi(x_n) + b) - 1), \tag{2.13}$$

where $a = (a_1, \ldots, a_N)^T$. By equating the derivatives of $L(w, b, a)$ with respect to w and b equal to zero, the following conditions emerge

$$w = \sum_{n=1}^{N} a_n t_n \phi(x_n), \tag{2.14}$$

$$0 = \sum_{n=1}^{N} a_n t_n. \tag{2.15}$$

By eliminating w and b from $L(w, b, a)$ using these conditions, what is left is the dual representation of the maximum margin problem, given by

$$\tilde{L}(a) = \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m t_n t_m k(x_n, x_m), \tag{2.16}$$

which will be maximized with respect to a, subject to the constraints

$$a_n \geq 0, \tag{2.17}$$

$$\sum_{n=1}^{N} a_n t_n = 0. \tag{2.18}$$

Also, the kernel function introduced here is defined by $k(x, x') = \phi(x)^T \phi(x')$.

Finally, new data points can be classified by the trained model by evaluating the sign of $y(x)$, which can be expressed in terms of $\{a_n\}$ and the kernel function, resulting in

$$y(x) = \sum_{n=1}^{N} a_n t_n k(x, x_n) + b. \tag{2.19}$$

The resulting sign of this calculation tells on which side of the hyperplane the data point lies, and consequently its class.

For every data point either $a_n = 0$ or $t_n y_n(x_n) = 1$, and for those that $a_n = 0$ they will not be used to make predictions for new observations. The relevant data points are called support vectors, and since they satisfy $t_n y(x_n) = 1$ they will lie on the MMH. See Figure 2.2 for an example. This is crucial for the practicability of SVMs, because after the model is trained many of the data points will not be used in calculations, and only the support vectors will be used for those calculations that return predictions.

After solving the optimization problem and finding the values for a, only the parameter b is left to be determined. Considering Equation 2.19 with the understanding that any support vector x_n satisfies $t_n y(x_n) = 1$, this gives

$$t_n \sum_{m \in S} (a_m t_m k(x_n, x_m) + b) = 1, \tag{2.20}$$

where S is the set of indices of the support vectors. Finally, the parameter b is given by

$$b = \frac{1}{N_S} \sum_{n \in S} \left(t_n - \sum_{m \in S} a_m t_m k(x_n, x_m) \right), \tag{2.21}$$

where N_S is the total number of support vectors.

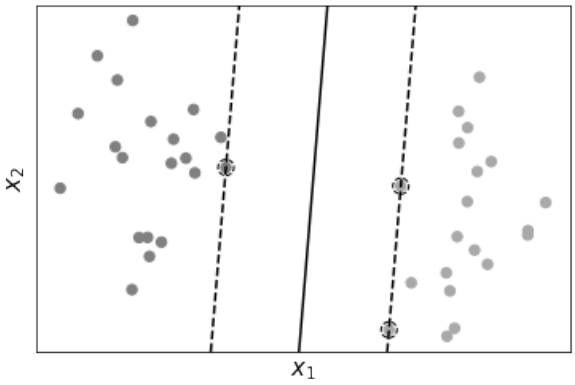


Figure 2.2: SVM decision boundary with margins defined by support vectors. Here the MMH separates data points into two classes. The margins, represented by dashed lines, are defined based on the support vectors that the SVM found, which in turn are represented by the points circled by dashed lines, while the distance between the margins is called the maximal margin.

As mentioned in the beginning, this approach assumes linearly separable data, which might not always be the case, especially when the class distributions overlap due to outliers or noise. When this is the case, the decision boundary reached after training may lead to poor generalization. Therefore, during training there must be some leeway for the misclassification of some training points. For this next part of the description, [16] is referenced.

The idea is then to allow data points to be misclassified with a penalty proportional to the distance from the decision boundary. For this a linear function of the distance will be used, and slack variables $\xi_n \geq 0$ with one slack variable for each training point will be introduced. For data points on or inside the correct margin $\xi_n = 0$, whereas for the remaining points $\xi_n = |t_n - y(x_n)|$. For instance, a point on the decision boundary $y(x_n) = 0$ will have $\xi_n = 1$, while points beyond that will be misclassified with $\xi_n > 1$. Points with $0 < \xi_n \leq 1$ are correctly classified but are between the decision boundary and the margin. This results in the classification constraint being

$$t_n y(x_n) \geq 1 - \xi_n, \tag{2.22}$$

where the slack variables satisfy $\xi_n \geq 0$.

This results in what are called Soft-Margin SVMs, and the corresponding objective function is

$$C \sum_{n=1}^{N} \xi_n + \frac{1}{2} \|w\|^2, \tag{2.23}$$

where $C > 0$ is a user defined parameter that controls the width of the maximum margin and the penalty for misclassifications. When $C \to \infty$ the previous SVM is recovered. If C is small some misclassifications are allowed. The goal here is to minimize this objective function in order to maximize the margin.

The Lagrangian is then

$$\tilde{L}(a) = \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m t_n t_m k(x_n, x_m), \tag{2.24}$$

which is equal to the one in the linearly separable case, although now the constraints are

$$0 \leq a_n \leq C, \tag{2.25}$$

$$\sum_{n=1}^{N} a_n t_n = 0. \tag{2.26}$$

Nonlinearly Separable Data

When data are not linearly separable, the previously discussed SVM would not be able to define a decision boundary that clearly separates the data into different classes, therefore no proper solution would be found. To solve this problem two things must be done to get a nonlinear SVM. First the original data are transformed into a higher dimensional space by means of a nonlinear mapping. This makes it more likely that the data become linearly separable, and therefore the next step is again solving a quadratic optimization problem that tries to find a linear separating hyperplane in this new higher dimensional space. The MMH found corresponds to a nonlinear separating decision surface in the original space. This case's description references [17].

One example of this mapping would be mapping a 2-D input vector $x = (x_1, x_2)$ into a 3-D space, z, using the mappings $\phi_1(x) = x_1$, $\phi_2(x) = x_2$, and $\phi_3(x) = x_1^2$. Now the decision surface would be given by

$$y(z) = w_1 x_1 + w_2 x_2 + w_3 x_1^2 + b = w_1 z_1 + w_2 z_2 + w_3 z_3 + b.$$

The linear decision surface in z space effectively corresponds to a nonlinear second-order polynomial in the original space. The issues now are choosing an appropriate mapping and the computational cost of doing all the dot products during training and testing.

To solve the new issues a mathematical trick called the kernel trick will be used. Since during training the training vectors appear as dot products, $\phi(x)^T \phi(x')$, instead of calculating the dot product on the transformed vectors, it is equivalent to apply a kernel function to the original data. This way the mapping is avoided and all the calculations can be done in the original space which is likely to be of a lower dimensionality.

Two popular kernels are the Polynomial kernel of degree h

$$k(x, x') = (x^T x' + 1)^h, \tag{2.27}$$

and the Gaussian radial basis function (RBF) kernel

$$k(x, x') = e^{-\frac{\|x - x'\|^2}{2\sigma^2}} = e^{-\gamma \|x - x'\|^2}. \tag{2.28}$$

Although there is no simple rule for determining the best kernel for any given situation, in practice, choosing different kernels does not usually make a big difference in the model's accuracy, and during training the SVM will always find a solution.

Training this model can take some time if a large data set is used for training, but it is an accurate model capable of dealing with complex nonlinear decision boundaries. Additionally, the support vectors found provide a compact description of the trained model, while being less prone to overfitting compared to other methods.

2.4 Unsupervised Learning

Unsupervised learning is an approach to ML that does not require human input, which means data are unlabeled. Models created using this technique are capable of learning simply by processing huge amounts of data. They do this to find hidden structures and patterns in data, that allow them to understand how data are correlated and consequently make predictions based on that.

Some real world applications of unsupervised learning are: sentiment analysis in social media to correlate people's emotions with what they post online [18]; detecting faulty equipment in time to prevent economic loss [19]; recommending news to users online through a recommendation engine [20].

Some problems with this approach are: inaccurate results are likelier; human intervention is required to validate results; less clear how data are clustered based on hidden relations. [21]

In this work, unsupervised learning will be used to perform anomaly detection and clustering. What follows are descriptions of the methods K-means, used for clustering, OCSVM, and IF, used for anomaly detection.

2.4.1 K-means

Clustering is an ML technique used to group unlabeled data based on their similarity. A clustering algorithm takes in raw data and tries to find patterns in that data so that it can output labeled groups of data that are correlated in some way. The one used in this work was the K-means clustering algorithm, which is one of the most popular and simpler algorithms that provides adequate results, hence why it was chosen. This algorithm's description references [22].

This algorithm aims to partition a data set $\{x_1, \ldots, x_N\}$ with N observations, with x_i having D dimensions, into K clusters, where K is a user defined parameter that predefines the number of desired clusters. The idea is that distances between points in a cluster are small compared to those outside their cluster, so it makes sense to group them. To use distances as a metric in this algorithm, the data set has to first be normalized according to Equation 2.2.

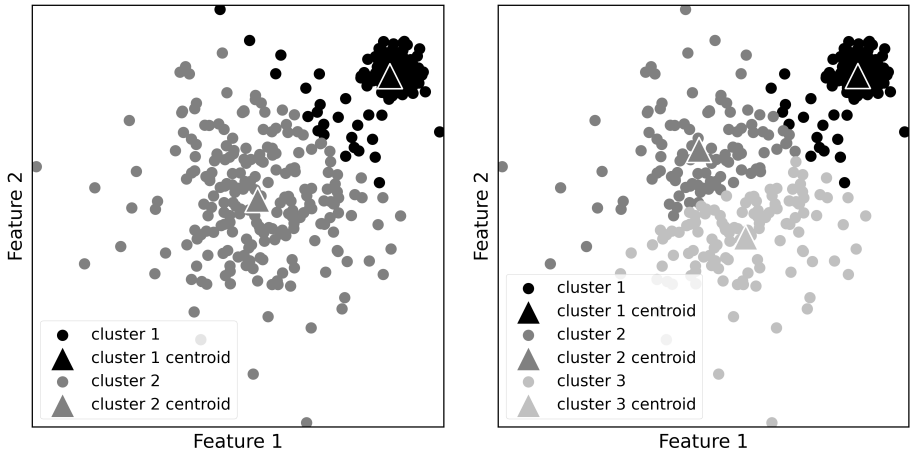
A cluster can essentially be identified by its center, therefore it is useful to formalize this concept as a set of D dimensional vectors $\{\mu_k\}$, where $k = 1, \ldots, K$, and each μ_k represents the k^{th} cluster. In the end there will be a set of vectors $\{\mu_k\}$ with data points assigned to them, where the sum of the squared distances of each point to the closest cluster center is a minimum.

During the process of assigning data points to clusters, each point x_n will have a corresponding set of binary indicator variables $r_{nk} \in \{0, 1\}$ that tells which cluster k the point x_n is assigned to. If point x_n is assigned to cluster k, $r_{nk} = 1$, otherwise if assigned to some other cluster $j \neq k$, $r_{nj} = 0$, and this is known as the 1-of-K coding scheme.

The objective function in this scenario is given by

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \left\| x_n - \mu_k \right\|^2 . \tag{2.29}$$

Now the goal is to minimize J by finding the values $\{r_{nk}\}$ and vectors $\{\mu_k\}$ that do so. This is done by means of an iterative process with two successive steps that perform optimizations with respect to r_{nk} and μ_k. To kick-start this process, initial values must be chosen for each μ_k. Then in the first step, J is minimized with respect to r_{nk} while μ_k is fixed, and in the second step, J is minimized with respect to μ_k while r_{nk} is fixed. This is repeated until the algorithm converges.



(a) **(b)**

Figure 2.3: K-means clustering when (a) $K = 2$ and (b) $K = 3$. In this example synthetic data was generated from two separate Gaussian distributions with different parameters. Note how the algorithm always manages to provide a solution even if it does not fit the data adequately, as seen in (b). Nevertheless, when the K parameter is correctly chosen the algorithm usually does a good job of clustering data, as seen in (a). This is a trivial example, but in more complicated cases, the elbow method is typically used to find an adequate K.

To determine r_{nk} first note that terms with different n are independent, therefore it is possible to optimize for each n individually by setting $r_{nk} = 1$ for any value of k that minimizes $\|x_n - \mu_k\|^2$. This means the n^{th} point is assigned to the closest cluster center. Formally,

$$r_{nk} = \begin{cases} 1 & \text{if } k = \arg\min_j \|x_n - \mu_j\|^2 \\ 0 & \text{otherwise.} \end{cases} \tag{2.30}$$

To optimize μ_k with r_{nk} fixed, first note that J can be minimized by equating to zero its derivative with respect to μ_k, which gives

$$2\sum_{n=1}^{N} r_{nk}(x_n - \mu_k) = 0, \tag{2.31}$$

and solving for μ_k gives

$$\mu_k = \frac{\sum_n r_{nk} x_n}{\sum_n r_{nk}}. \tag{2.32}$$

In Equation 2.32 the denominator represents the number of points assigned to cluster k, therefore μ_k is equal to the mean of all points x_n in cluster k. This is why this algorithm is called K-means.

As mentioned, this two-step procedure is performed until the algorithm either converges or simply stops due to a limiting condition regarding the number of iterations allowed. Note that this algorithm always converges because J is systematically minimized, although it is possible that convergence will occur at a local minimum rather than a global minimum. In Figure 2.3 a visual representation of the results produced by K-means clustering can be seen.

When it is unclear what a good K value would be when applying clustering to a data set, the elbow method, as seen in Figure 2.4, can be used to plot a distortion score against a range of K values, which visually clarifies what an adequate K value would be for that data set. This distortion score is essentially the objective function J.

The main goal of the elbow method is decreasing the sum of within-cluster variance of each cluster, which happens the more clusters there are. Although this allows grouping data with more granularity, after a certain point, splitting a cohesive cluster only reduces this variance slightly, while producing clusters that group information in a potentially worse way. [23]

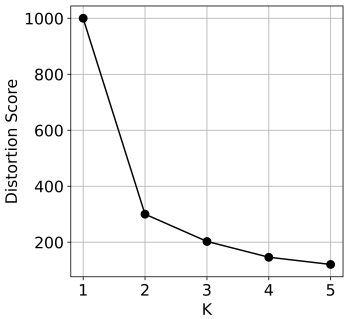


Figure 2.4: Elbow method applied to the clustering example in Figure 2.3. Note that for $K = 2$ the curve suddenly shifts direction, similar to a human arm, where the highest score is the shoulder, the point where the curve suddenly shifts direction is the elbow, and the remaining curve is the forearm. This shift signals the optimal K for the data set.

2.4.2 One-Class Support Vector Machines

In the original SVM method, the goal was to differentiate points by labeling them with different classes, and there could be simply two classes or many classes. But in some situations the interest lies in distinguishing normal observations in a training set from abnormal observations that do not belong to the distribution of those training points. Specifically this is called outlier detection, when the training data has observations far from the regions with more densely packed observations, or novelty detection, when a new observation is presented to the model so that it can be classified as being part of the normal observations or not. This is equivalent to having only one class and labeling normal data with that class. To solve this issue the OCSVM method was developed. This method's description references [24].

To separate data from the origin the following quadratic program problem must be solved,

$$\min_{w \in F, \xi \in \mathbb{R}^N, b \in \mathbb{R}} \quad \frac{1}{2} \|w\|^2 + \frac{1}{\nu N} \sum_i \xi_i - b \tag{2.33}$$

$$\text{s.t.} \quad w^T \phi(x_i) \geq b - \xi_i, \quad \xi_i \geq 0, \tag{2.34}$$

where N is the number of observations, $x_1, \ldots, x_N$ is the training data, F is a dot product space, $i = 1, \ldots, N$, and $\nu \in [0, 1]$ represents the fractions of support vectors and outliers.

The decision function that computes on which side of the hyperplane a point is, is given by

$$f(x) = sgn\left(\sum_i a_i k(x_i, x) - b\right), \tag{2.35}$$

where the coefficients are found as the solution of the dual problem

$$\min_a \quad \frac{1}{2}\sum_{i,j} a_i a_j k(x_i, x_j) \tag{2.36}$$

$$\text{s.t.} \quad 0 \leq a_i \leq \frac{1}{\nu N}, \quad \sum_i a_i = 1, \tag{2.37}$$

where $j = 1, \ldots, N$. Also, for any a_i which is not at the upper or lower bound, the corresponding x_i satisfies $b = w^T\phi(x_i) = \sum_j a_j k(x_j, x_i)$.

Essentially this method estimates a function, f, that outputs 1 in a small region of space where most data points are concentrated, and -1 everywhere else. Since it uses the kernel trick it transforms the data according to the kernel used, and separates it from the origin with maximum margin. When a new observation is computed with f, its value determines which side of the hyperplane it lies on.

2.4.3 Isolation Forest

This method uses binary trees to efficiently isolate anomalous data points in a data set. Since these points are prone to isolation they ought to be closer to the root, contrary to normal points which should be further along the tree. See Figure 2.5b for an example of a tree. The only two parameters considered in this method are the number of trees, t, used, and the sub-sampling size, ψ. This method's description references [25].

Since an isolation tree is a full binary tree, a node, T, can be an external node with zero children nodes or an internal node with one test and two children nodes, (T_l, T_r). A test involves a split value p between the minimum and maximum value of a feature q, such that the test $q < p$ splits points into T_l and T_r.

Given a data set and a corresponding subset $X = \{x_1, \ldots, x_N\}$, where N is the number of observations, an isolation tree can be made by recursively dividing X by randomly selecting a feature q and a split value p until one of three conditions is met: a tree height limit is reached; $|X| = 1$, which means only one point is left to partition; all the remaining points in X have the same values.

A tree structure represents this recursive partitioning, and the path length from the root to an external node, where an isolated point lies, equals the number of partitions needed to isolate that point. Figure 2.5 depicts this process clearly.

If all points are different, then when an isolation tree is fully grown, each one will be isolated in an external node. This means there will be N external nodes and $N - 1$ internal nodes, with the total of nodes being $2N - 1$.

The path length, $h(x)$, is equal to the number of edges between x's external node and the root of the isolation tree. For N observations in a data set, the average path length is given by

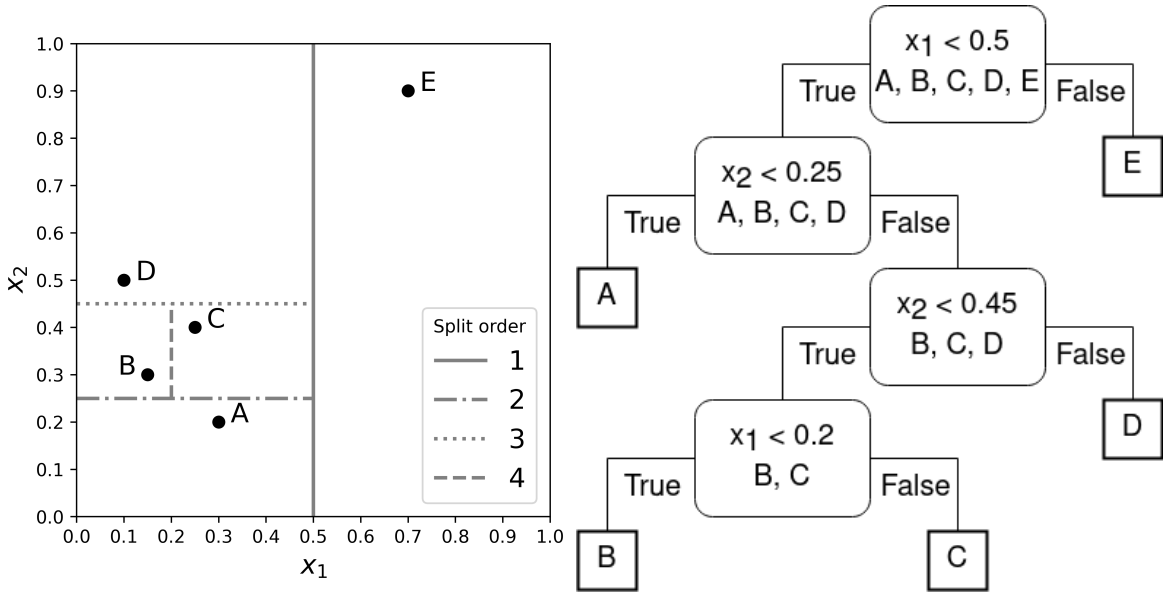
$$c(N) = 2H(N-1) - \left(\frac{2(N-1)}{N}\right), \tag{2.38}$$

where $H(a)$ is the harmonic number, and $c(N)$ is the average of $h(x)$, which is used to normalize $h(x)$.

The anomaly score, s, of a point, x, is given by

$$s(x, N) = 2^{-\frac{E(h(x))}{c(N)}}, \tag{2.39}$$

where $E(h(x))$ is the average of $h(x)$ in an ensemble of isolation trees. Note s's lower and upper bounds are respectively given by $E(h(x)) \to N - 1 \implies s \to 0$ and $E(h(x)) \to 0 \implies s \to 1$, resulting in $0 < s \leq 1$, with the understanding that $0 < h(x) \leq n - 1$. The middle ground between definitely being an anomaly and definitely being a normal observation is given by $E(h(x)) \to c(N) \implies s \to 0.5$. This means that when a point has $s \approx 1$ it is an anomaly, whereas if $s \ll 0.5$ it is probably a normal observation. If all observations score $s \approx 0.5$ then most likely there are no anomalies in the data set.



(a) **(b)**

Figure 2.5: IF's (a) feature space partitioning and (b) resulting isolation tree that detects an outlier. In (a) the split order gives a sense of how easy it is for an outlier, such as point E, to be partitioned early on in the process. In (b) the isolation tree visually represents all the isolated points and their depths in relation to the root, which is represented by the top of the tree. Note how the outlier point E has $depth = 1$, because one edge has to be traversed to reach it, and therefore is immediately recognized as an outlier, whereas the remaining points require further partitioning to be isolated.

To use IF for detecting anomalies, two things must be done. First the isolation trees must be trained, each with a different sub-sample of the training data. Then in the detection stage, each observation of the test set is passed through all the isolation trees, and an anomaly score is calculated for each one. During this last stage, when each observation x is passed through an isolation tree, $h(x)$ is calculated by counting the number of edges from the root to the external node, and subsequently $E(h(x))$ is derived, which is necessary to calculate the aforementioned anomaly score, as seen in Equation 2.39. On average anomalies should be closer to the root. One way to rank how outlying a point is, is by sorting

points according to their path lengths or anomaly scores, where the ones at the top will be considered anomalies.

2.5 Performance Evaluation Metrics

There are many ways to evaluate the performance of an ML technique given some data, but only a few that are most commonly used will be mentioned. In regression problems the two most popular metrics are the mean absolute error (MAE) and MSE. In classification problems some of the most used metrics are the confusion matrix, accuracy, recall, precision, specificity, fallout, miss rate, the receiver operating characteristic (ROC) curve, and the area under curve (AUC).

The MAE is given by

$$\text{MAE} = \frac{1}{N}\sum_{n=1}^{N}|t_n - x_n|\,, \tag{2.40}$$

where x_n is the prediction, t_n the target value, and N the number of observations [26]. Some properties of this error function are: it is slower to converge with a fixed learning rate, therefore a dynamic learning rate will most likely be required; it is robust to outliers since it does not increase the error dramatically when they appear; although it measures the distance from the predicted value to the target value it does not tell the direction of the error.

The MSE is given by

$$\text{MSE} = \frac{1}{N}\sum_{n=1}^{N}(t_n - x_n)^2, \tag{2.41}$$

where the different variables represent the same as in Equation 2.40 [27]. Some properties of this error function are: an adequate fixed learning rate will allow a fast convergence; since its gradient is higher when the error is high and lower when the error is low, it will give a more precise value when converging; it is sensitive to outliers since it will square the error when they appear, giving more weight to them.

To describe the performance evaluation metrics used in classification problems, [28] was referenced.

When dealing with a classification problem, the way the accuracy of predictions is measured is essentially by calculating the ratio of correct predictions to total predictions, and this takes the form of

$$\text{Accuracy} = \frac{\text{number of correct predictions}}{\text{number of predictions}} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}, \tag{2.42}$$

where the different kinds of predictions made are true positives (TP), true negatives (TN), which represent the correct predictions made regarding the positive and negative classes, respectively, and false positives (FP), false negatives (FN), which represent the wrong predictions made regarding the positive and negative classes, respectively, although sometimes the model simply outputs a correct or wrong prediction as seen with the logistic regression for instance. If the accuracy is preferred in terms of a percentage, then the resulting ratio

just needs to be multiplied by 100. Note that when using accuracy as a metric it is important to have a balanced data set. A balanced data set requires an approximately equal number of samples from each class.

The confusion matrix in Table 2.1 takes the different kinds of predictions TP, TN, FP, and FN, and presents them in a tabular visualization to make it clear how much the prediction model is confusing the two classes, positive and negative.

Table 2.1: Confusion matrix used in classification problems. The sum of the diagonal with TP and TN represents the total of correct predictions, while the sum of the negative diagonal with FP and FN represents the total of incorrect predictions. The sum of all four cells represents the total of predictions made regarding the provided data set.

Real Class \ Predicted Class	Positive Class	Negative Class
Positive Class	TP	FN
Negative Class	FP	TN

Recall, or true positive rate (TPR), is the ratio of predicted positive cases that are indeed real positives in relation to false negatives, and is also an important metric for the ROC curve. It is given by

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}. \tag{2.43}$$

Precision, or positive predictive value (PPV), is the ratio of predicted positive cases that are indeed real positives in relation to false positives, and it is given by

$$\mathrm{PPV} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}. \tag{2.44}$$

Specificity, or true negative rate (TNR), is the ratio of predicted negative cases that are indeed real negatives in relation to false positives, and it is given by

$$\mathrm{TNR} = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}}. \tag{2.45}$$

Fallout, or false positive rate (FPR), is the ratio of predicted positive cases that in reality are negatives in relation to true negatives, and is also an important metric for the ROC curve. It is given by

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}}. \tag{2.46}$$

Miss rate, or false negative rate (FNR), is the ratio of predicted negative cases that in reality are positives in relation to true positives, and it is given by

$$\mathrm{FNR} = \frac{\mathrm{FN}}{\mathrm{FN} + \mathrm{TP}}. \tag{2.47}$$

The ROC curve plots TPR against FPR, which are normalized metrics, at various detection thresholds, and it allows the comparison between classifiers with different models or

parameters by visually showing how a classifier performed. Figure 2.6 shows the different performances that classifiers can have and how those are plotted in the ROC curve.

The AUC is calculated with numerical integration methods that solve definite integrals, which in this case was the trapezoidal rule with non-uniform grid given by

$$\text{AUC}_{b-a} = I(f) = \int_a^b f(x)\,dx \approx \sum_{k=1}^{N} \frac{f(x_{k-1}) + f(x_k)}{2} \Delta x_k, \tag{2.48}$$

where a is the first point x_0, b is the last point x_N, and $\Delta x_k = x_k - x_{k-1}$ is the width of a segment of the grid [29]. In the context of this work, the AUC is always calculated in a finite interval $[a, b]$, where $a = 0$ and $b = 1$.

Since the ROC curve is inside the unit square, the AUC is always $0 \leq \text{AUC} \leq 1$. For the best classifier $\text{AUC} = 1$, while for the worst classifier $\text{AUC} = 0$. When $\text{AUC} = \frac{1}{2}$ three scenarios are possible: the classifier is a random classifier; some parts of the ROC curve are below the diagonal that defines a random classifier, while others are above it, compensating each other; half of the ROC curve has $\text{TPR} = 0$ while the other half has $\text{TPR} = 1$. [30]

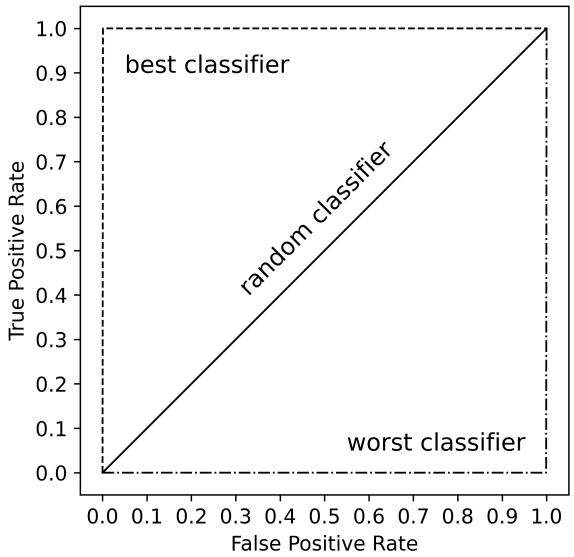


Figure 2.6: Generic ROC curve that shows possible performances for a classifier. Points above the diagonal means the performance of the classifier is better than simply random chance, while being below the diagonal means the performance is worse than chance. When points lie exactly on the diagonal, FPR = TPR throughout the curve, and this means the classifier outputs correct results at the same rate as incorrect ones, therefore it is a random classifier. The best possible classifier has $(\text{FPR} = 0, \text{TPR} = 1)$ in the top left corner, which means that every classification is correct. The worst possible classifier has $(\text{FPR} = 1, \text{TPR} = 0)$ in the bottom right corner, which means that every classification is incorrect. When comparing different classifiers with this curve, the goal is always to choose the best possible classifier, which means choosing the classifier with the performance curve that is above the diagonal and farthest from it, in other words, the one closest to having $(\text{FPR} = 0, \text{TPR} = 1)$ in the top left corner.

CHAPTER 3

Cellular Frustration Model

The algorithms that solve the Stable Marriage Problem (SMP) create arrangements where all agents are paired in stable bonds. Stable bonds are created when paired agents cannot be destabilized by interacting with other agents. Therefore, solving the Stable Marriage Problem amounts to finding such stable arrangements. [31] The inspiration for the CF model was in part taken from this model, but it operates differently.

With CF algorithms the goal of the dynamic is the opposite. Instead, agents should be designed to engage in a perpetual unstable dynamic, therefore never reaching configurations where all agents are in stable pairings. This dynamic depends on the information presented by one set of agents and contained in a sample of a data set. The principle is that if agents have been prepared to engage in a very unstable dynamic when samples from a normal class are presented, then when samples from an abnormal class are displayed the population should engage in a less frustrated dynamic.

Based on this principle, these algorithms give good results in anomaly detection tasks when compared to currently popular anomaly detection algorithms. More interestingly, since these algorithms derive from models appropriate to describe cellular interactions in the real immune system, this work can gain relevance for building a deeper understanding of how the immune system works [32].

In this chapter, the concepts and terminology that define the CF model are presented. These build upon those presented in [7] and [33].

3.1 Terminology And Definitions

Definition 1 (Agent). *An agent* $a_i \in A \; \forall i \in \{0, 1, 2, \ldots, N_A\} \colon N_A = |A|$*, is defined as a tuple* $a_i = \{s_i, l_i, K_i, m_i\}$*, where:*

- s_i *is the signal shown by the agent;*
- l_i *is a sequence of signals called the global preference list of the signals shown by the agents of the opposite type;*
- K_i *is the set of agents with which an agent* a_i *can interact with;*

- m_i *is the state of an agent* a_i *and contains the index* i *of the agent with which it is matched, or the value* -1 *if it is unmatched.*

Definition 2 (Agent types). *The set of agents* A *is made up of two subsets of agents, where in one exist presenters* P*, and in the other, detectors* D*, such that* $A = P \cup D$ *and* $P \cap D = \emptyset$*. The total number of presenters and detectors is given by* $N_P = |P|$ *and* $N_D = |D|$*, respectively. Hence, the total number of agents is given by* $N_A = N_P + N_D$*.*

Definition 3 (Subtypes). *In this model there can be subtypes for each type of agent. The subtypes of presenters are defined by the ordering of their list of signals* l_i*, where the first signal* s_i *of that list determines its subtype. On the other hand, the subtypes of detectors are defined by the signal* s_i *that they show. A pair of matched agents of the same subtype is called* M*, whereas if their subtypes are different it is called* $\overline{M}$*.*

Definition 4 (Signals). *The set of signals shown by presenters is given by* $X_P = \{s_i \colon s_i \in X \subset \mathbb{R} \wedge a_i \in P\}$*, whereas the set of signals shown by detectors is given by* $X_D = \{s_i \colon s_i \in X \subset \{1, 2\} \wedge a_i \in D\}$*. While presenters can show many signals, detectors only show one signal, which is equal to their subtype. To clarify when one or the other is being referenced, the notation* s_i^P *will be used when mentioning the presenters' set of signals and* s_i^D *when mentioning the detectors' set of signals.*

Definition 5 (Rankings). *The preference lists* l_i*, are ordered according to the preferences of their agents'* a_i*, towards the signals contained in them, with each signal* s_i *having a ranking* $r_{l_i}(s_i)$ *associated with it. Each ranking is simply the index of a signal in* l_i*. A lower index rank is preferred over a higher index one.*

Definition 6 (Detectors' Domains). *Each detector has a normal domain,* $\mathcal{I}$*, and an abnormal domain,* $\mathcal{O}$*, for each feature's distribution of a cluster of samples. A randomly generated threshold probability,* ν_i*, sets how many samples are included in* $\mathcal{I}$*, as* $N_S = 100 - 100\nu_i\%$*, with* $\mathcal{O}$ *having the remaining.*

In statistics the normal domain would be the acceptance region of a distribution, while the abnormal domain would be the rejection region.

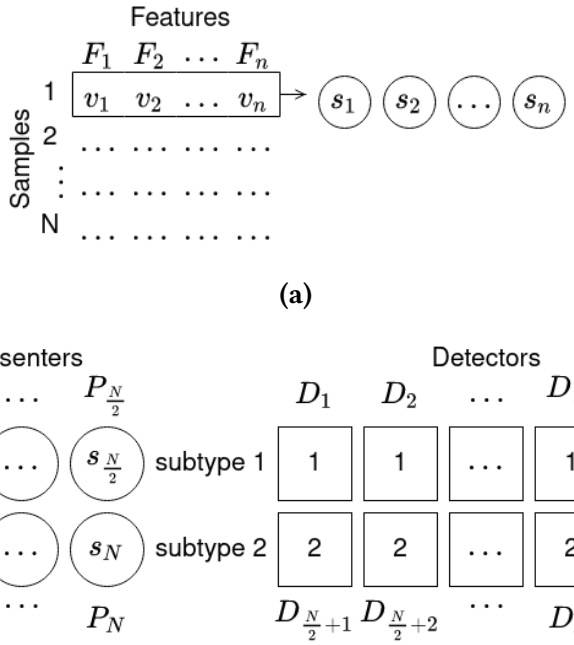
Definition 7 (Detectors' Critical Values). *Each* $\mathcal{I}$ *has symmetric critical values* $z_{\nu_i/2}$ *calculated from the normalized features',* F_f*, distributions of values* $v_f \forall S_u$*, of samples,* S*, within it, that give it lower and upper bounds. The lower bound is given by* $z_{\nu_i/2} = \min\{v_f \forall S_u\}$*, while the upper bound is given by* $z_{1-\nu_i/2} = 1 - z_{\nu_i/2}$*.*

See Figure 3.3 and Figure 3.4 to understand how Definition 6 and Definition 7 relate to the way detectors see signals s_i as either being inside or outside their normal domains.

Definition 8 (Signals Mapping). *Detectors establish a mapping,* M*, of signals,* s_i^P*, as* $M : s_i \to \{\iota_i, o_i\} \forall a_i \in P$*, where signals* $\iota_i \in \mathcal{I} \subset X_P \wedge o_i \in \mathcal{O} \subset X_P : \mathcal{I} \cup \mathcal{O} = X_P \wedge \mathcal{I} \cap \mathcal{O} = \emptyset$*. Therefore, detectors see the continuous values shown by the signals* s_i^P *as binary signals.*

See Figure 3.1, Figure 3.3 and Figure 3.4 for a visual representation of how signals are mapped.

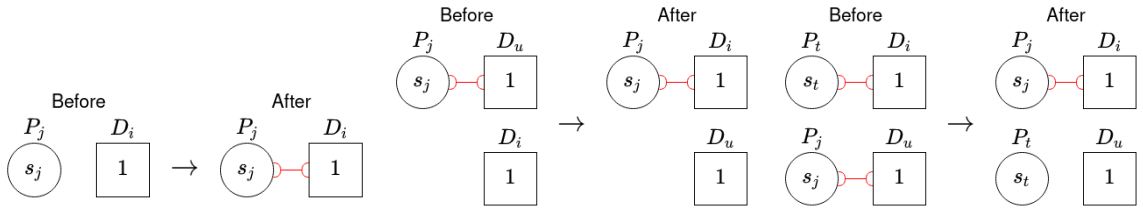
Definition 9 (Interactions Between Types Of Agents). *Only interactions between agents of different types are allowed, such that* $K_i \subseteq D \; \forall \; a_i \in P$ *and* $K_i \subseteq P \; \forall \; a_i \in D$. *Furthermore, there can only be pairs of agents, e.g., one agent bound to another.*



(a)

(b)

Figure 3.1: CF model's mapping of information from the samples in a data set onto presenters and ultimately detectors' preference lists. (a) Shows how each sample's information is mapped onto a set of presenters signals. (b) Shows the preference lists of both presenters and detectors and how the signals they show are mapped onto those lists. Note that presenters show a wide range of signals while detectors only show the signals 1 and 2, which are ranked in the presenters' preference lists according to their subtype, meaning presenters will always prefer detectors that have the same subtype as them. The signals shown by presenters are mapped onto a binary set of signals that represent the signal either being inside the normal domain of a detector or outside it.



(a) **(b)** **(c)**

Figure 3.2: The three fundamental ways agents interact when deciding who to pair with. (a) When both agents are unpaired. (b) When all agents except one are paired. (c) When all agents are paired.

Definition 10 (Decision Rules). *To update the states* m_j, m_t, *of agents* $a_j, a_t \in P$, *and* m_i, m_u, *of agents* $a_i, a_u \in D$, *when these interact with each other, it is necessary to follow the*

update rules R:

1. *if* $m_i = -1 \wedge m_j = -1 \Rightarrow m_i = j, m_j = i$.
2. *if* $m_i = t \wedge m_j = -1 \wedge r_{l_i}(s_j) < r_{l_i}(s_t) \Rightarrow m_i = j, m_j = i, m_t = -1$.
3. *if* $m_i = -1 \wedge m_j = u \wedge r_{l_j}(s_i) < r_{l_j}(s_u) \Rightarrow m_i = j, m_j = i, m_u = -1$.
4. *if* $m_i = -1 \wedge m_j = u \wedge r_{l_j}(s_i) = r_{l_j}(s_u) \wedge s_j \in \mathcal{I}_i \wedge s_j \in \mathcal{O}_u \Rightarrow m_i = j, m_j = i, m_u = -1$.
5. *if* $m_i = t \wedge m_j = u \wedge r_{l_i}(s_j) < r_{l_i}(s_t) \wedge r_{l_j}(s_i) < r_{l_j}(s_u) \Rightarrow m_i = j, m_j = i, m_t = -1, m_u = -1$.
6. *if* $m_i = t \wedge m_j = u \wedge r_{l_i}(s_j) < r_{l_i}(s_t) \wedge r_{l_j}(s_i) = r_{l_j}(s_u) \wedge s_j \in \mathcal{I}_i \wedge s_t \in \mathcal{O}_i \wedge s_j \in \mathcal{O}_u \Rightarrow m_i = j, m_j = i, m_t = -1, m_u = -1$.

It becomes apparent that the agents follow a greedy strategy, since each agent swaps pair whenever they have the opportunity to pair with a preferred agent.

It is important to note the distinction between these rules and the ones considered in the original CF model used as reference in this work. Originally, only rules 1, 2, 3, and 5, existed, with no rule ever mentioning the states of the signals as either being inside or outside a detector's normal domain.

Lemma 1. *Only the R rules, 1, 5, and 6, change the number of unmatched agents.*

Proof. From Definition 10 it can be taken that when all the rules have had their conditions fulfilled, only rule 1 lowers by two the number of unmatched agents and only rules 5, and 6, equally increase that same number, whereas the remaining rules do not change the number of unmatched agents. □

Definition 11 (Stable Matching)**.** *A matching between two agents, a_i and a_j, is said to be stable, when the successive use of rules, R, on every other agent does not change the states, m_i and m_j, of those two agents.*

Definition 12 (Stochastic Iteration)**.** *During a stochastic iteration and given a sequence in which all N_A agents are included, each one of these agents will be sequentially randomly chosen to interact with another agent of the opposite type, which itself was chosen randomly. Next, the states, m_i, of these agents will be updated according to rules, R.*

Definition 13 (Population Configuration)**.** *A population configuration, C, is given by the set of matched agents along with the set of unmatched agents: $C = \{a_i, a_j \colon m_i = j \wedge m_j = i \colon i, j = 1, \ldots, N_A\} \cup \{a_i \colon m_i = -1 \colon i = 1, \ldots, N_A\}$.*

Definition 14 (Stable Configuration)**.** *A configuration, C, is said to be stable when all agents that are within it are in a stable matching.*

Despite being possible to achieve a stable configuration, it is important to note that the probability of a population of agents reaching such a configuration is tiny.

Lemma 2. *The probability of a population, with equal number of presenters and detectors, staying in an unstable configuration by the end of an iteration is* $\frac{N_A}{2} - 1$ *greater than ending up in a stable configuration.*

Proof. In the worst case scenario, consider a population that is just one iteration away from ending up in a stable configuration. This can be achieved by unmatching two agents bound to each other in a population already in a stable configuration. Furthermore, assume that the next iteration involves one of these unmatched agents.

The probability of that agent choosing the only other unmatched agent and therefore returning to a stable configuration is $\frac{1}{\frac{N_A}{2}} = \frac{2}{N_A}$. Therefore, the probability of the population remaining unstable is $1 - \frac{2}{N_A}$. Since $\frac{1-\frac{2}{N_A}}{\frac{2}{N_A}} = \frac{N_A}{2} - 1$ this means the probability of the population remaining unstable is $\frac{N_A}{2} - 1$ greater than ending up in a stable configuration, with this probability increasing the more agents that are involved. □

Definition 15 (Matching Lifetime)**.** *The lifetime of a matching between two agents is determined by the number of iterations that elapsed between the iteration when they matched and the iteration when they unmatched. This is represented by* $\tau_i \in \mathbb{N}$.

3.2 Procedure And Mechanisms

The CF model has several steps and mechanisms that allow it to train detectors and consequently perform detections on data sets. The most important mechanisms will be explained in some detail in order to give a deeper and clearer understanding of the inner workings of the model.

3.2.1 Feature Space Normalization

It is important to normalize feature values in data sets in order to facilitate the mapping of the signals as explained previously. The min-max normalization described in Equation 2.1 was used in this model and the values were mapped to the range $[0, 1] \in \mathbb{R}$.

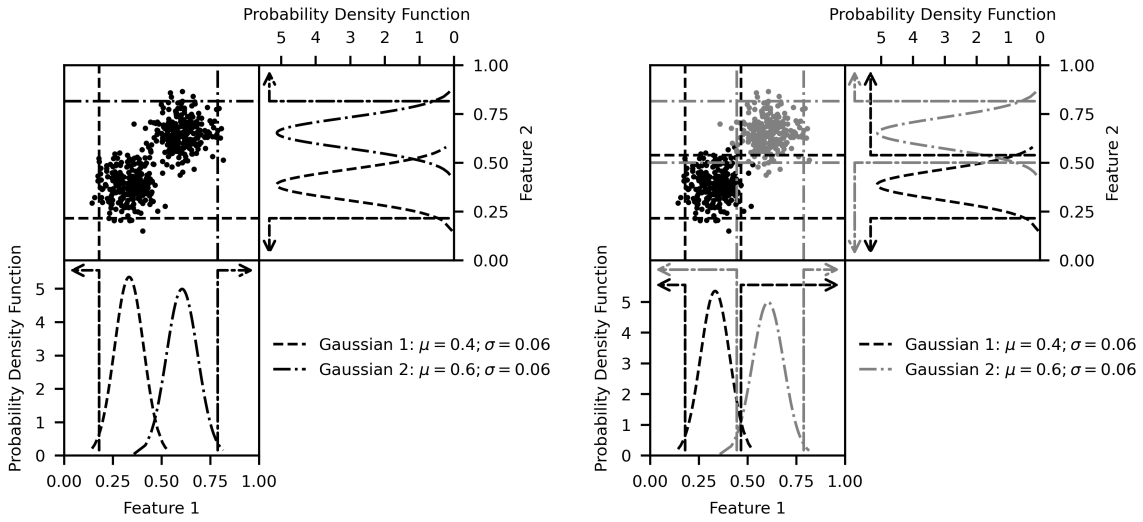
3.2.2 Feature Space Partitioning

One major problem with CF is that detectors require an accurate partition of the feature space, otherwise there will be many false positives or false negatives, hence no detection at all. One way to solve this is by partitioning the feature space into clusters. This way detectors see less feature space as being normal. Note that since detectors take the distribution of values within each feature and divide it into normal and abnormal domains, it becomes crucial to ensure these domains are properly defined.

Figure 3.3 shows the difference between a clustered and non-clustered data set and how this impacts a detector's ability to detect anomalies. In this example K-means clustering with $K = 2$ was used.

After this partitioning, detectors are then able to correctly map the continuous feature values to the binary signals shown by presenters during training, as either being inside or

outside their normal domains. Figure 3.4 describes this in terms of the cumulative distribution function $F_i(z_{\nu_i/2})$, estimated from the data available for training, that is considered when looking at the distribution of a feature's values.



(a) **(b)**

Figure 3.3: Training samples distributions projections (a) without and (b) with clustering. In (a) there is only one cluster, therefore all detectors base their view of the feature space on the left tail of Gaussian 1 and the right tail of Gaussian 2. This has the consequence of making the detectors see a lot of the abnormal feature space as normal, which is the case in [7]. In (b) detectors are assigned to each cluster, therefore they are able to partition the feature space better. This way they are able to better distinguish normal regions of space from abnormal ones. In tests discussed later on, detectors were distributed evenly across all clusters considered during training.

Note how detectors in this work have two tails, which means they should be more robust to false negatives when abnormal feature values show on either side of the feature space distribution, but this might also mean that more normal values will be considered abnormal during testing, hence more false positives. In [7] detectors only have one tail, hence one abnormal domain with which to detect abnormal samples. This means that even if a very obvious abnormal feature value appears in a sample inside the normal domain, it will be considered as a normal value, leading to a false negative. It becomes apparent that the approach in this work is more aggressive than the one in [7] regarding the mapping of feature values into normal and abnormal signals.

During testing, it was noticed that detectors specialized to their assigned cluster (i.e., normal domains based solely on the features' distributions of samples from their cluster), had difficulty dealing with samples from others clusters, because to them anything outside their cluster was abnormal, therefore many false positives would ensue. This led to the development of the shuffling procedure, which is quite simply randomly swapping detectors' pairs of left and right critical values of each feature with other detectors. This led to more robust detections, since now when detectors saw samples from different clusters than the one they were assigned to originally, they would see those as only partly abnormal, which means they would see some features as normal, leading to less false positives.

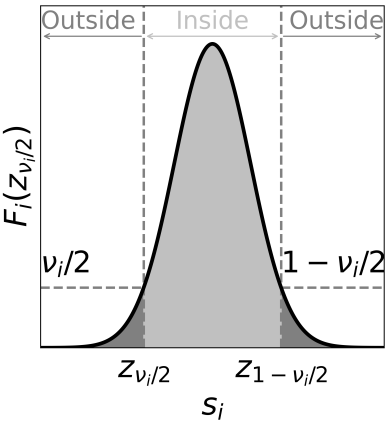


Figure 3.4: Mapping of normalized feature values s_i into binary signals ι_i and o_i. In order for detectors to map signals s_i onto signals inside or outside their normal domain, they must calculate $F_i(z_{\nu_i/2})$ for each feature. When s_i lies in the left, or right, tail of the distribution, $F_i(z_{\nu_i/2}) \leq \nu_i/2$ and $F_i(z_{1-\nu_i/2}) \geq 1 - \nu_i/2$, respectively, and it will be mapped onto a signal o_i, otherwise it will be mapped onto a signal ι_i. $\nu_i \sim U(0, \nu_{max})$, where typically $\nu_{max} \leq 0.2$.

3.2.3 Cellular Frustration Dynamic

There are three main stages when using the CF model, and these are training, calibration, and monitoring. Although they have different purposes and functions, the main loop is the same. This loop is described in Algorithm 3.1. Note that the total number of iterations the algorithm executes is n_{max} and the period between changing samples is T, which typically was $T = 100$.

Algorithm 3.1 Cellular frustration dynamic.

```
Initialize detectors' global preference lists with signals ι_i and o_i randomly ranked
{τ_i} ← 0
for all n ∈ [1, n_max] do
    if n mod T is 0 then
        Change sample shown by presenters
    for all a_j ∈ P ∪ a_i ∈ D do
        Pick a random presenter a_j
        Pick a random detector a_i
        decision(A, a_i, a_j) - apply decision rules in Definition 10
    for all a_i ∈ A do
        if a_i ∈ A is paired then
            τ_i ← τ_i + 1
```

3.2.4 Sample Rotation

Presenters show many samples from the data set throughout the CF dynamic, therefore, there has to be some method of selecting samples from it. Some different methods are: sequentially in the same order as they are stored in the data set; if clustering was applied to the data set, sequentially but on a per-cluster basis, selecting all samples from each cluster before moving on to the next cluster; completely random; randomly selecting a cluster and then randomly selecting a sample from that cluster. The second method was chosen.

3.2.5 Agent Dissociation

To prevent two agents from being paired for too long, possibly even during the entirety of the dynamic, when an agent is selected to take its turn in an iteration, there is a chance that it will become unmatched. This does not significantly alter the results since the probability of this dissociation happening is fairly low.

3.2.6 Detectors' Preference Lists

Detectors have two preference lists, one global and one local. The global list has all the possible binary signals each presenter can show, as defined in Definition 1. The local list only has the binary signals currently shown by presenters given the selected sample, as seen from each detector's perspective regarding its normal and abnormal domains, as seen in Figure 3.5 with reference to Figure 3.6. This list is used for computational reasons, to decrease access times to signals.

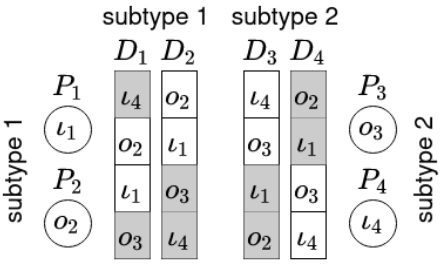


Figure 3.5: Detectors' local preference lists for a sample.

The global preference list of every detector is initially randomized, such that the binary signals corresponding to each presenter are scattered throughout the list without any ordering. Later on there will be some ordering of these signals when education is introduced. These unordered lists can be seen as per the example in Figure 3.6.

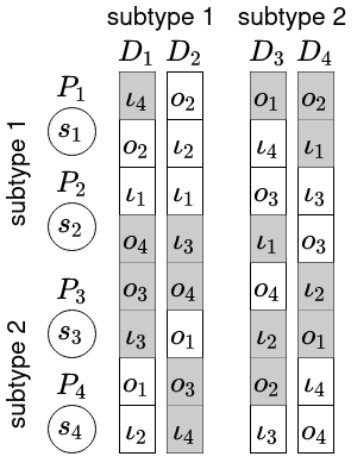


Figure 3.6: Detectors' unordered global preference lists l_i, before training. In this case $N_P = N_D = 4 \implies N_A = 8$, which means there are 4 signals shown in any given moment, while the total number of possible binary signals is 8, which will have to be represented in detectors' preference lists. Signals from presenters of the opposite subtype to each detector are colored in gray.

3.2.7 Training

During training, when agents are paired for too long, this indicates that they both have each other high on their preference lists, therefore detector agents must be trained to avoid this attachment to presenters with subtype equal to theirs that show signals ι_i. The goal is to train detectors to prefer signals o_i, so that when an abnormal sample is shown, detectors will have high affinity with presenters' signals and become very attached to them, leading to long matching lifetimes. Since presenters of the same subtype as detectors will generate the longest matching lifetimes, after training, these presenters will have their signals lowest in detectors' preference lists. This will result in a frustrated dynamic.

Training a batch of detectors is only done every few thousand iterations E_n, in order to allow statistically relevant detectors' matching lifetimes to appear. The condition check that sends a detector to training compares a detector's current τ_i with the threshold for training $\tau_{threshold}$, which is updated to a new lower value based on the highest τ_i amongst all detectors, but only if no detectors were trained after the last E_n iterations, which typically was $E_n = 1500$.

Algorithm 3.2 Detector education.

```
τ_max ← 0
trained ← false
for all a_i ∈ D do
    if τ_i > τ_max then
        τ_max ← τ_i
    if τ_i > τ_threshold then
        a_j is paired with a_i
        trained ← true
        newRank ← random integer greater than rank_{l_i}(s_j)
        In l_i swap signal at rank rank_{l_i}(s_j) with signal at rank newRank
        m_i, m_j ← −1
        τ_i, τ_j ← 0
if trained is false then
    τ_threshold ← τ_max
```

Algorithm 3.3 Training.

```
Initialize detectors' global preference lists with signals ι_i and o_i randomly ranked
{τ_i} ← 0
τ_threshold ← n_max
for all n ∈ [1, n_max] do
    if n mod T is 0 then
        Change sample shown by presenters
    for all a_j ∈ P ∪ a_i ∈ D do
        Pick a random presenter a_j
        Pick a random detector a_i
        decision(A, a_i, a_j) - apply decision rules in Definition 10
    for all a_i ∈ A do
        if a_i ∈ A is paired then
            τ_i ← τ_i + 1
    if n mod E_n is zero then
        education(A, τ_threshold)
```

Algorithm 3.2 describes how the detectors' preference lists are manipulated in order to produce an educated population of detectors capable of recognizing abnormal samples when presented. Algorithm 3.3 is essentially the same as Algorithm 3.1 with the added function that educates detectors.

The education mechanism is visually described in Figure 3.7, and the final educated lists for an example case are shown in Figure 3.8.

3.2.8 Calibration

Before detection can be achieved, detectors must be calibrated with the normal samples used for testing in a two-step procedure. First an activation tau, τ_{act}, is defined as follows. When running the CF dynamic, a sorted vector of taus, $c(\tau_i)$, will count the number of times any detector leaves a pairing with a lifetime τ_i. When the dynamic ends, a cumulative sum is performed on this vector in order to get the number of times each $\tau \geq \tau_i$ occurred. Then these new values in $c(\tau \geq \tau_i)$ are averaged as follows $c_{avg}(\tau \geq \tau_i) = c(\tau \geq \tau_i)/N_S/N_D$, where N_S is the total of all calibration samples. Finally, $\tau_{act} = \max\{c_{avg}(\tau \geq \tau_i) < 1\}$. This is visually shown in Figure 3.9.

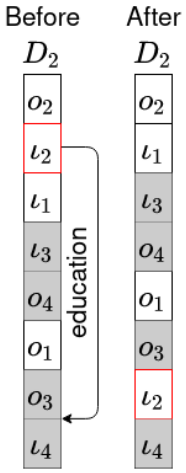


Figure 3.7: Detector being educated with an education operation applied to its global preference list l_i. This operation consists in swapping the signal outlined in red with a signal lower ranked in the l_i. This is triggered by a long matching lifetime $\tau_i > \tau_{threshold}$, which is undesirable when detectors are interacting with samples they perceive as normal, therefore they must learn to dislike these samples. This stable matching happens because D_2 ranks highly the signal shown by P_2, and since they are both of the same subtype, P_2 also ranks D_2 highly in its preference list. On top of this, D_2 also sees the signal s_2 as being inside its normal domain, which according to the decision rules further promotes this stable matching.

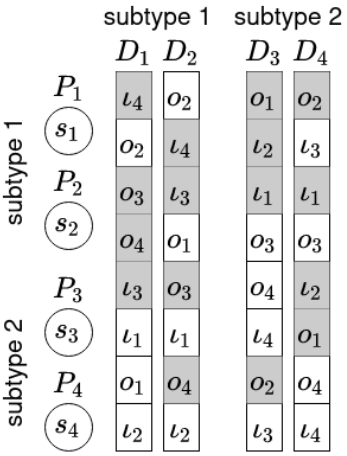


Figure 3.8: Trained detectors and their ordered global preference lists l_i with reference to the lists in Figure 3.6. Some signals were correctly educated to lower ranks while some were not. All lists now have mostly gray colored signals at the top, which means detectors learned to prefer signals from presenters of the opposite subtype, which leads to lower matching lifetimes when presenting normal samples, hence cellular frustration is achieved. In general all lists are trained well. D_2's l_i has one signal on top of the list shown by a presenter of the same subtype, because it never appeared during training. In D_4's l_i, an incorrect education occurred, where a previously correctly ranked signal ι_1 now has a lower rank, and signal ι_3 climbed the list, which is bad, because it leads to long pairings with normal samples. This can happen when the population's configuration is very stable at some point and triggers a long matching. Although very problematic this sort of event is rare.

The idea is that the highest $\tau \geq \tau_i$, where on average detectors have a low $c(\tau \geq \tau_i)$ when shown normal test samples, should be a τ where response is minimum for normal samples but still above 0. Therefore, for abnormal samples it should be easy for even weak responses to trigger detections. It is important to select an activation tau where on average detectors still have $c(\tau \geq \tau_i) > 0$, because if $c(\tau \geq \tau_i) = 0$, when a normal sample is shown during testing it easily triggers a detection, leading to false positives.

The CF dynamic is run for w_{max} iterations for each sample, where $w_{max} = 5000$ in this case. Each sample needs to be evaluated for a few thousand iterations to avoid the influence of statistical fluctuations when registering matching lifetimes.

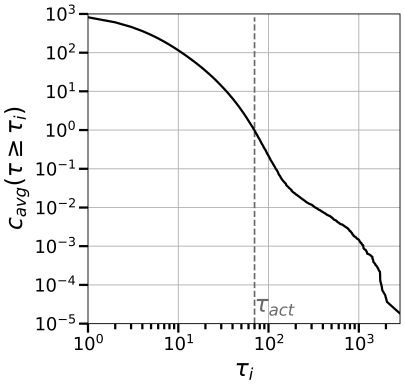


Figure 3.9: Calibration of an activation tau τ_{act}.

Finally, an activation threshold $n_i(\tau_{act}) \forall a_i \in D$ is defined as follows. The same CF dynamic is run again, and a sorted vector $c_{i,j}(\tau_{act}) \forall j$, where j corresponds to a normal sample used for calibration, will register the number of times a detector a_i establishes a matching lifetime $\tau_i \geq \tau_{act}$, for each sample j. Then each detector's activation threshold is calculated as $n_i(\tau_{act}) = c_{i,x}(\tau_{act})$, where $x = N_S \times f$, with $f \in [0, 1]$. In this case $f = 0.05$, which means the 5% largest number of pairing lifetimes greater than τ_{act} were considered.

3.2.9 Detection

The last stage when applying this model is the monitoring stage, where both normal and abnormal samples from the test set are monitored in order to extract the responses from detectors towards each sample. From these responses a ROC curve is plotted to see the performance of the model when trying to distinguish abnormal samples from normal ones.

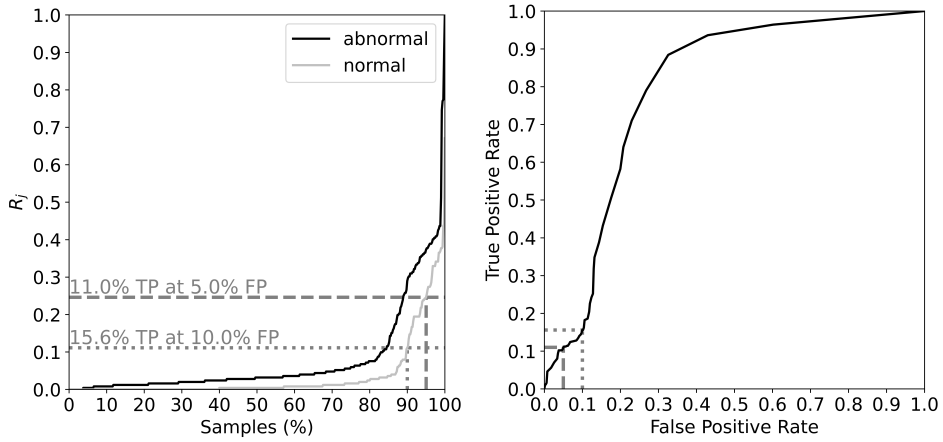


Figure 3.10: Translation of detectors' responses to a typical ROC curve. (a) Shows the normalized responses obtained for the normal and abnormal samples, with two threshold examples for 90% and 95% of the normal samples, represented by the dotted and dashed lines, respectively. All the responses towards normal and abnormal samples above the dotted or dashed lines are considered detections, be it true positives or false positives. (b) Shows the example thresholds translated into the TPR for a given FPR seen in a typical ROC curve.

There should be a frustrated dynamic when a normal sample is presented, whereas when an abnormal sample is presented more stable pairings should occur, leading to long pairing lifetimes, which in turn produce a response from detectors towards these abnormal samples much higher than towards normal samples. Therefore, there should be a clear difference between these responses, hence result in a good detection by the cellular frustrated system (CFS).

In this stage, the same CF dynamic considered in the calibration stage is also run here, for the same w_{max} iterations per sample. Finally, the total response towards a sample j by the collection of detectors, is based on the number of times each detector established a long

pairing lifetime relative to its activation threshold, and it is calculated as follows

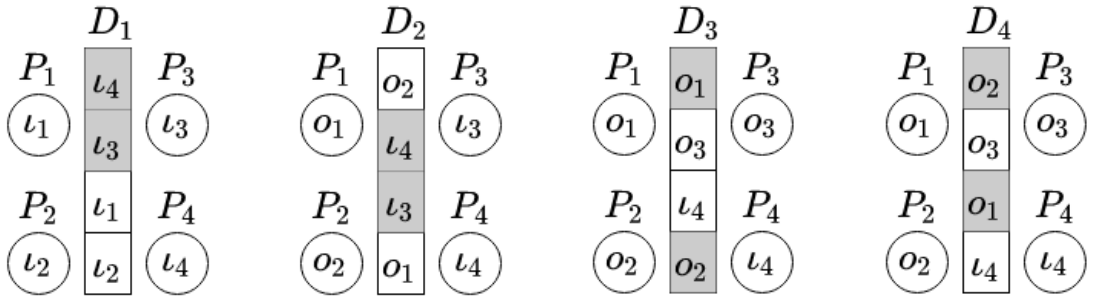
$$R_j = \sum_{i}^{N_D} (c_{i,j}(\tau_{act}) - n_i(\tau_{act}))H(c_{i,j}(\tau_{act}) - n_i(\tau_{act})), \tag{3.1}$$

where H is the Heaviside step function. See Figure 3.10 to better understand how the ROC curve is plotted based on detectors' responses.

The Heaviside step function guarantees a result has a lower bound of zero if the calculation it takes as argument results in a negative value, and it is given by

$$H(x) = \begin{cases} 1 & \text{x > 0} \\ 0 & \text{x < 0.[34]} \end{cases} \tag{3.2}$$

CFSs have the ability to detect three different abnormal patterns in trained detectors' local preference lists, which allow them to make detections when an abnormal sample is shown by presenters. These are the presence of abnormal signals o_i that were rarely or never shown during training, the absence of too many normal signals ι_i compared to how common they were during training, and the absence of combinations of signals ι_i frequently shown together during training. These are more clearly explained in Figure 3.11.



(a) **(b)** **(c)** **(d)**

Figure 3.11: Mechanisms of local preference lists that allow detectors to make detections, with reference to Figure 3.8. These four examples show the local preference lists of detectors when a certain combination of signals is shown by presenters, and how the order of these signals in preference lists leads to detections. Since each presenter can only show one of its two possible binary signals at any given time, there is no purpose in representing the remaining signals that are not being evaluated at that time, therefore the local preference list representation is more appropriate. (a) Shows how a properly educated list prevents erroneously detections when during monitoring a normal sample is presented, and detectors should not be making long pairings. Since the signals at the top of the list belong to presenters with the opposite subtype to D_1, frustration is achieved. (b) During training the signal o_2 never appeared because it is an abnormal signal, and no sample contained any feature value that could be mapped to it, so it remained in its original randomly generated rank in D_2's global preference list. Therefore, when a sample appears in the monitoring stage with some feature value that can be mapped to o_2, it appears at the top, which provokes a response from D_2, leading to a detection. (c) When the combination of two normal signals at the top of D_3's list become absent, the abnormal signal o_3 is able to climb the list and induce a stable matching between D_3 and P_3. Since both have the same subtype, P_3 becomes very attached to D_3, leading to a detection. This sort of absent combinations can happen when certain features are strongly correlated. (d) When many sequential signals ι_i in D_4's list become absent, the signals o_i are able to climb the list provoking a strong response and triggering a detection. It is important to note that in all these situations, a strong response only occurs when a collection of detectors share the same view of a sample. This is especially important for the detection mechanism that relies on absent combination of signals ι_i, where only a few might disappear in each l_i for an abnormal sample hard to detect.

CHAPTER 4

Results

In this chapter the CF model will be tested alongside the anomaly detection methods mentioned in chapter 2. These methods and K-means clustering, will be implemented using the Scikit Learn library version 1.0 in [35]. The tests in this chapter will consider two types of data sets, synthetic and real data sets. The synthetic data sets will come from Gaussian distributions with parameters based on the ones considered in [36]. The real data set considered here will be the one in [37] provided by [38], which contains red and white wines with their qualities labeled. In this work only the white wines were considered.

To evaluate the performance of the anomaly detection models tested in this chapter, the ROC curve and AUC described in chapter 2 will be used. This will be particularly useful to compare the results obtained in this work for the CF model, with the ones obtained in [7] and [36], for the white wines data set. Additionally, to better explain the results of the CF model, the detectors' responses and how they perceive the features of samples in each test, will be shown and discussed.

4.1 Tests With Synthetic Data Sets

To test any model methodically, one important requirement is having control over the test environment, in order to be able to tweak parameters and see how these affect results, allowing drawing conclusions. This is why before testing an ML model with real data, testing it with synthetic data is an important step. The results from these tests should give some indication of how the model will behave with real world data, hence how useful it will be and in which scenarios it can be applied.

The synthetic data sets used were generated according to the Gaussian distribution, given by

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/(2\sigma^2)}, \tag{4.1}$$

where μ is the mean, σ is the standard deviation, and x is a random variable such that $X \sim \mathcal{N}(\mu, \sigma^2)$ [39]. Six different data sets, corresponding to six different test cases, were generated with each one having $N_f = 11$ number of features. The reasoning behind this N_f is because the real data sets tested later on also have 11 features, therefore it is useful to perform these tests with that same condition. Three different Gaussian distributions were

used to generate data points for each feature, two for normal samples, and one for abnormal samples. Since each of these distributions always maintained the same parameters when generating points for each feature, all features are directly correlated.

In each test case, 1000 normal samples and 500 abnormal samples were generated. Three clusters of data points were created based on these samples. One cluster had 500 normal samples from one Gaussian distribution, a second cluster also had 500 normal samples but from a different Gaussian distribution, and finally a third cluster had the only 500 abnormal samples generated by the third Gaussian distribution. The parameters used for each test case in each Gaussian distribution are in Table 4.1. These test cases reference the ones in [36]. Each test case was run five times with random normal samples selected for training and testing, in order to average results to avoid influences from statistical fluctuations. All methods were cross validated with exactly the same samples.

Table 4.1: Parameters used in each Gaussian distribution for each test case. N_n is the total number of normal samples, N_1 is the number of normal samples in the first cluster, N_2 is the number of normal samples in the second cluster, and N_a is the total number of abnormal samples, which are in the third cluster.

Case	Samples					
	Normal ($N_n = 1000$)				Abnormal ($N_a = 500$)	
	Cluster 1 ($N_1 = 500$)		Cluster 2 ($N_2 = 500$)		Cluster 3	
	μ	σ	μ	σ	μ	σ
1	0.3	0.1	0.4	0.1	0.7	0.1
2	0.48	0.06	0.54	0.06	0.51	0.12
3	0.25	0.06	0.5	0.06	0.75	0.06
4	0.38	0.06	0.64	0.06	0.51	0.12
5	0.25	0.06	0.75	0.06	0.50	0.06
6	0.48	0.02	0.52	0.02	0.50	0.03

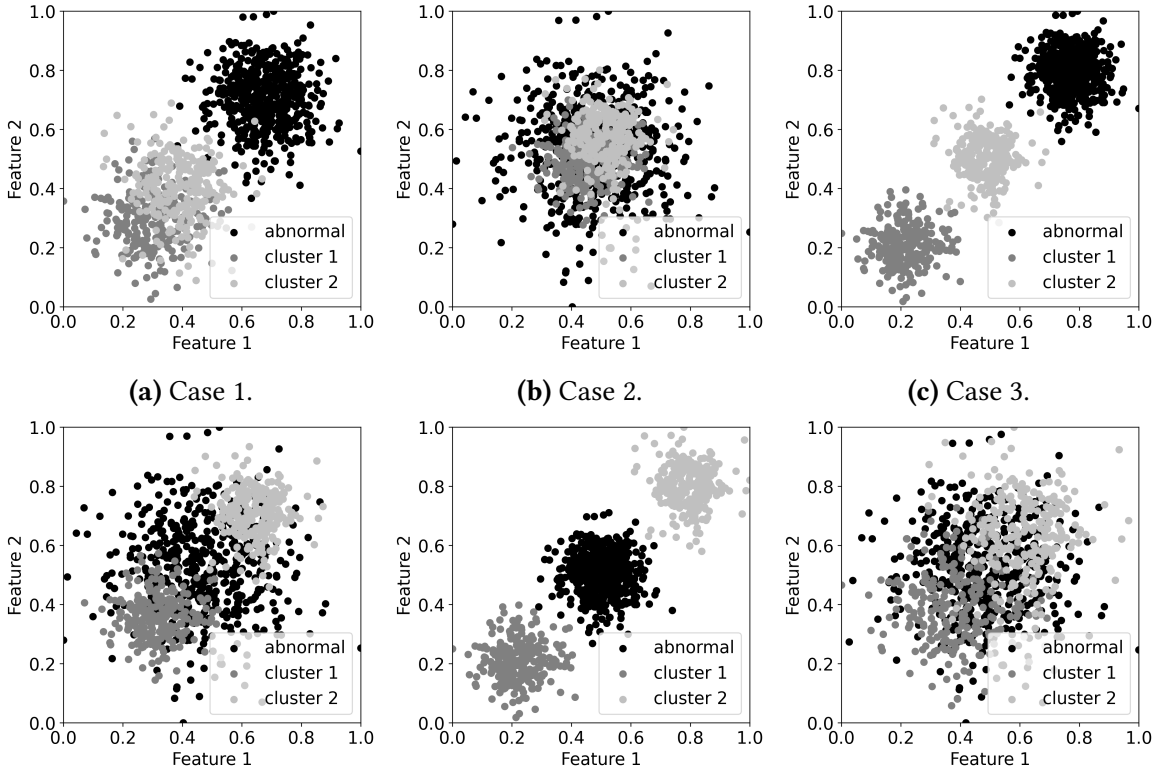
The distribution of samples from each generated cluster in feature space, for each test case, is represented in Figure 4.1 without clustering. This figure only contains the test set used in one selected test run for each case. It contains 250 samples from Cluster 1, 250 samples from Cluster 2, and 500 samples from Cluster 3. Before using the training set with the CF method, K-means clustering was applied with $K = 2$.

As stated before, two competing anomaly detection methods will be compared with the CF model presented in this work. The results for each test case using each anomaly detection method, can be seen in Figure 4.4 and Figure 4.5. For the CF model, the responses curves and the number of signals o_i detectors see in samples, can be seen in Figure 4.2 and Figure 4.3, respectively. Based on these results a brief discussion for each test case will follow.

4.1.1 Test Cases 1, 3, And 5

The test cases 1, 3, and 5, will be grouped together in this discussion because of the similarities of their results. The data sets considered in these tests have the abnormal samples well separated from the normal samples clusters. Therefore, all methods should have identical

detection performances in all cases, these being 100% detection rates with 0% false positives. These expected results are exactly what is seen in Figure 4.4a, Figure 4.4c, and Figure 4.4e. Table 4.2 confirms these results.



(a) Case 1. **(b)** Case 2. **(c)** Case 3.

(d) Case 4. **(e)** Case 5. **(f)** Case 6.

Figure 4.1: Visual representation of the distributions of the data points in the test set for each test case. Before using the generated data sets with the CF model, they are normalized with min-max normalization, as seen in Equation 2.1, with normalized values in the range $[0, 1] \in \mathbb{R}$. For a better understanding of how detectors are assigned to each cluster, and consequently see some samples as normal and some as abnormal, see Figure 3.3 and Figure 3.4.

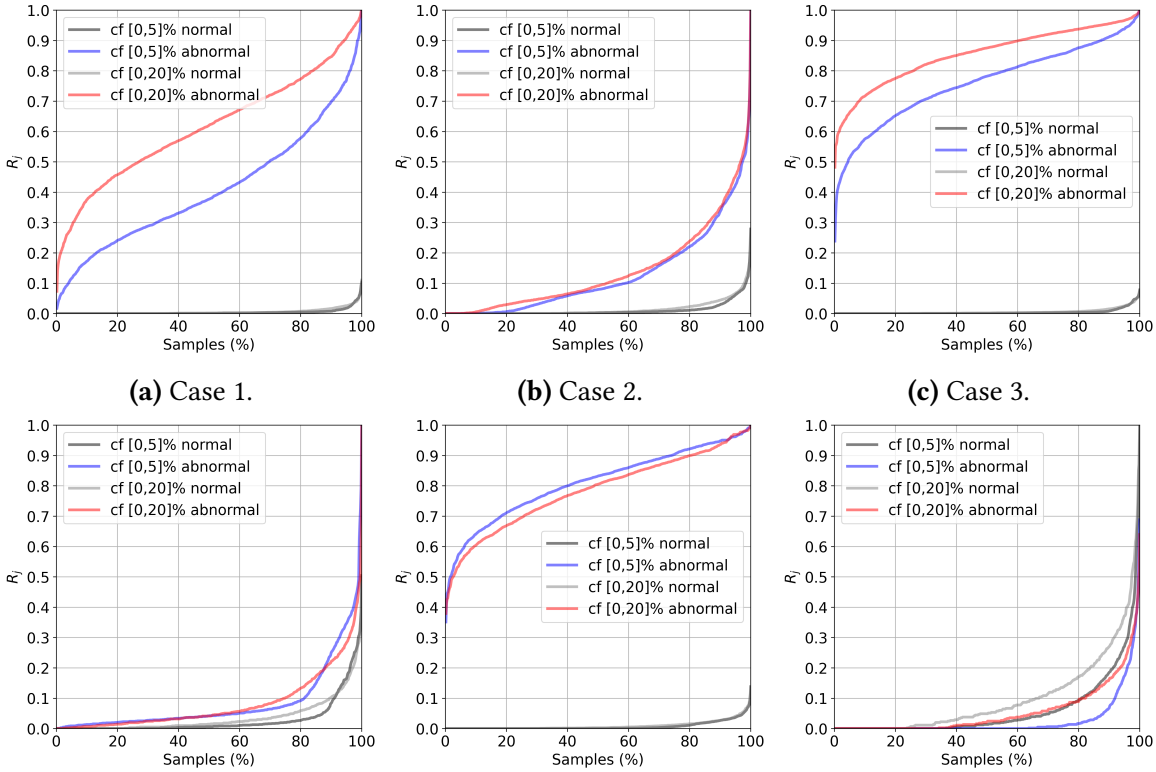
In particular for the CF model, it is worth noting the pattern seen in the responses, Figure 4.2, and signals o_i, Figure 4.3, plots. In Figure 4.2a, approximately 100% of the responses towards abnormal samples are greater than $R_j = 0.1$, and in Figure 4.2c and Figure 4.2e, all the responses towards abnormal samples are greater than $R_j = 0.1$, whereas towards normal samples all the responses are less than $R_j = 0.1$. In Figure 4.3a, Figure 4.3c, and Figure 4.3e, the ratio of number of signals o_i detectors see in the abnormal samples to the number of signals o_i they see in normal samples is approximately double or more. This is a clear indication that the model can easily make detections when more features in a sample are abnormal compared to the ones the model was trained on.

Regarding the fact that there does not seem to be a strict correlation between the average number of signals o_i and the intensity of the corresponding response, for the same averaged samples, it should be noted that this being an average, sometimes there are less than 11 signals o_i in a sample, leading to a weaker response from detectors. The model does predict several detection mechanisms that allow this, for example when a single signal ι_i from a

presenter of the opposite subtype is higher in the detectors' preference lists it will promote a frustrated dynamic, decreasing their capability to make detections, even though most signals are o_i. This being inferred from the results, more analysis is required to empirically demonstrate it.

4.1.2 Test Case 2

The data set in this case presents a difficult situation, since almost all the abnormal samples overlap the normal ones. In order to detect all the abnormal samples, some trade-off with FPR will be required. This can be seen in Figure 4.4b, where the OCSVM and IF have higher detection rates at 0% FPR than the CFSs CF $[0, 5]\%$ and CF $[0, 20]\%$. The intervals $[0, 5]\%$ and $[0, 20]\%$, refer to the randomly generated ν_i values of detectors, which can be any value in those intervals for each CFS considered. Table 4.2 also shows that all methods achieved a somewhat similar level of performance, with CF only having approximately less 10% AUC than the other methods.



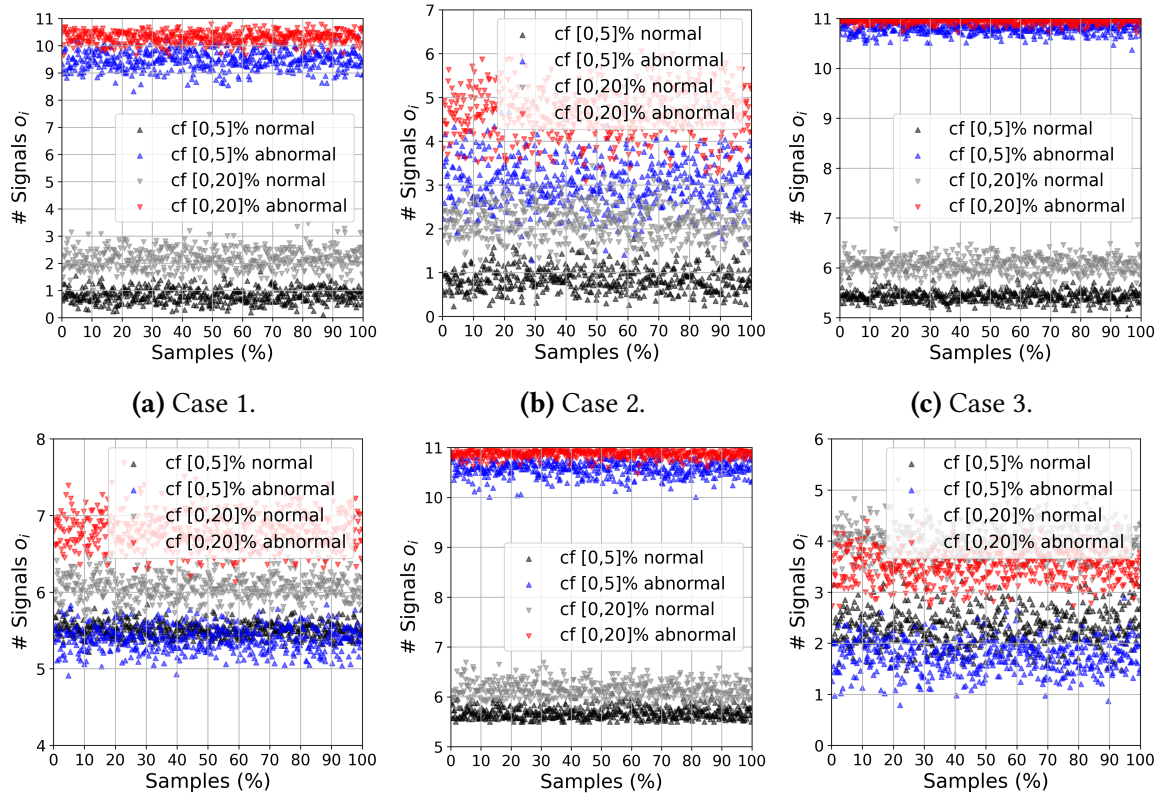
(a) Case 1. **(b)** Case 2. **(c)** Case 3.

(d) Case 4. **(e)** Case 5. **(f)** Case 6.

Figure 4.2: Detectors' normalized average responses for each test case with synthetic data. For the corresponding ROC curves see Figure 4.4. The samples in each response curve of each run for each test, were sorted prior to averaging, so that each run's highest and lowest response provoking samples were averaged accordingly. The ROC curves in Figure 4.4 were averaged considering the FPR and TPR of each test run, not the final averaged responses.

Figure 4.3b shows that abnormal samples only have half of their signals seen as o_i, in CF $[0, 20]\%$, which is half of what is seen in cases 1, 3, and 5, with normal samples still having low amounts of signals o_i, these being approximately two. This means detectors can still make detections thanks to a high ratio of number of signals o_i seen in abnormal samples

to normal samples, but just barely. In order for more detections to occur with this kind of ratio, some combination of signals ι_i must disappear in detectors' preference lists, in order to promote long pairings, hence detections. In CF $[0, 5]\%$ the same detection mechanism applies, but since there are even less signals o_i seen in abnormal samples than in CF $[0, 20]\%$, the overall performance is worse, as can be seen in Figure 4.4b when comparing the curves of CF $[0, 5]\%$ and CF $[0, 20]\%$. The reason why CF $[0, 20]\%$ is able to make more detections with less false positives, is because it has a higher ν_{max}, allowing the model to define wider abnormal domains in its detectors, which in turn will be able to recognize more abnormal samples, albeit at the cost of also seeing more signals o_i in normal samples, as is apparent in Figure 4.3b.



(a) Case 1. **(b)** Case 2. **(c)** Case 3.

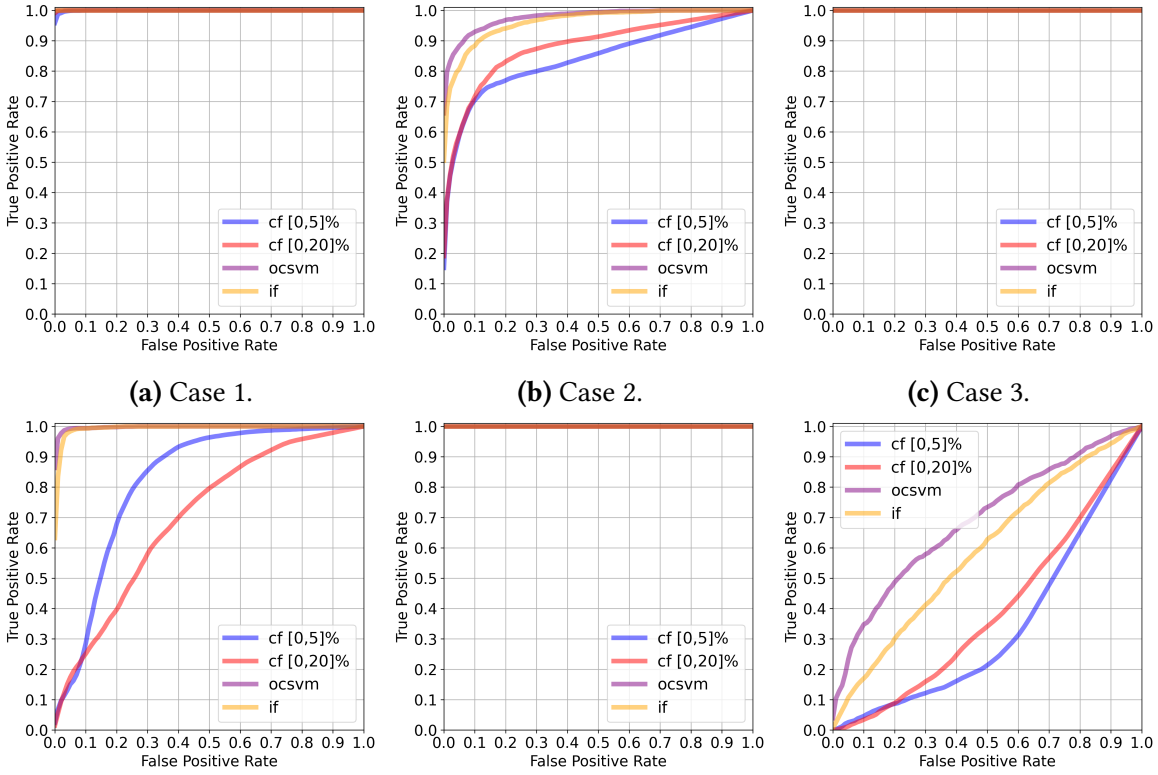
(d) Case 4. **(e)** Case 5. **(f)** Case 6.

Figure 4.3: Average number of signals out o_i seen per detector in each normal and abnormal sample for each test case with synthetic data. Each point corresponds to a sample, with all samples being sorted prior to averaging in the same order for each test as in Figure 4.2.

4.1.3 Test Case 4

The test setup in this case is very similar to the one in case 2, albeit with normal samples being further apart, requiring the CF model to learn two separate normal regions of samples overlapping the whole abnormal region, while in case 2 essentially only one big normal region needed to be learned, with many detectors sharing much of the same knowledge of the feature space. Regarding the performances of the several methods, it is clear from Table 4.2 that both OCSVM and IF were able to detect approximately 100% of the abnormal samples without false positives, with the CF model performing worse up to 30%. The

OCSVM with the RBF kernel can easily define decision boundaries around the Gaussian clusters used for training, allowing it to still separate well most of the abnormal samples from the normal ones. The IF with its techniques of sub-sampling and randomly generated forest of isolation trees, can more precisely learn the normal regions, because they sit on the edge of the abnormal region, where abnormal samples are sparser, and in addition to this, the abnormal samples are quite dispersed, which further helps it partition the feature space. The results in Figure 4.4d confirm this.



(a) Case 1. **(b)** Case 2. **(c)** Case 3.

(d) Case 4. **(e)** Case 5. **(f)** Case 6.

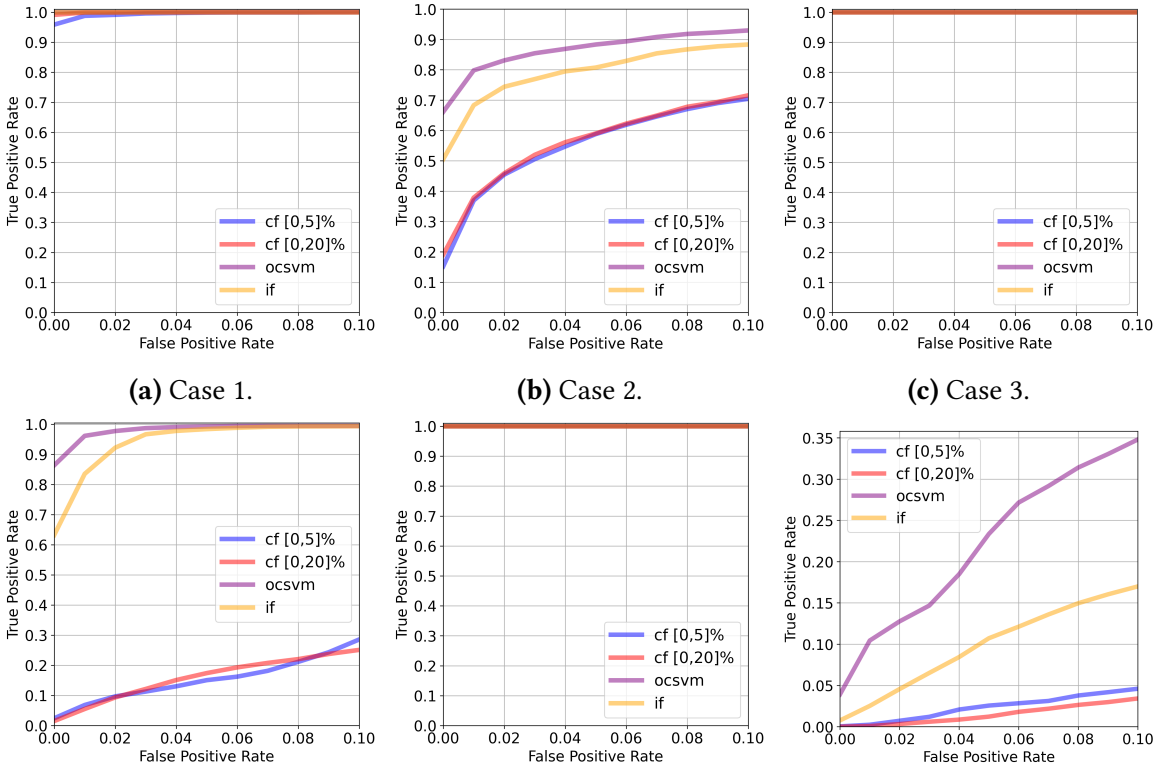
Figure 4.4: ROC curves for each test case with synthetic data. The OCSVM and IF methods were used with the default parameters of the library. For OCSVM, $k(x, x') = e^{-\gamma\|x-x'\|^2}$, $\gamma = \frac{1}{N_f \sigma^2}$, and $\nu = 0.5$. For IF, $t = 100$, and $\psi = \min\{256, |S|\}$. The CF method was used with $N_P = N_D = 110$, $n_{max} = 10^6$, and $\nu_{max} = 5\%$ for CF $[0, 5]\%$, and $\nu_{max} = 20\%$ for CF $[0, 20]\%$. For the AUC scores see Table 4.2, and for a zoomed in view between 0% and 10% FPR see Figure 4.5.

Table 4.2: Average AUC $(\%)$ and corresponding standard deviation, for each test case for each model.

Case	1		2		3		4		5		6	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
CF $[0, 5]\%$	100	0.0	84.0	1.1	100	0.0	82.2	2.5	100	0.0	33.9	0.7
CF $[0, 20]\%$	100	0.0	87.8	0.6	100	0.0	70.0	3.3	100	0.0	39.2	0.7
OCSVM	100	0.0	97.7	0.2	100	0.0	99.8	0.0	100	0.0	70.0	0.8
IF	100	0.0	96.2	0.3	100	0.0	99.4	0.1	100	0.0	59.1	1.6

Looking at Figure 4.2d and Figure 4.3d, bearing in mind the performances of both CF

$[0, 5]\%$ and CF $[0, 20]\%$ in case 2, it is important to understand why CF $[0, 20]\%$ now performs 10% worse than CF $[0, 5]\%$. Essentially, what allowed it to perform better in case 2, now hinders its performance. It is clear that CF $[0, 20]\%$ had a higher response to abnormal samples with its detectors seeing higher amounts of signals o_i in them, but due to the wider abnormal domains defined in its detectors, that also means that normal samples contained in these regions will also evoke a similar response, leading to more false positives, as seen in its ROC curve. On the other hand, it can be inferred that CF $[0, 5]\%$ was able to better educate its detectors' preference lists, because even though both abnormal and normal samples had equal amounts of signals o_i, the ones shown in normal samples were likely lower on their preference lists, with the ones in abnormal samples being higher. Not only this, but it is likely that some signals ι_i shown by presenters of the opposite subtype were also high on their preference lists, so that when a normal sample was shown, a frustrated dynamic followed, whereas when an abnormal sample was shown, those signals disappeared, pushing to the top the signals o_i that would lead to a detection. Further analysis is required to demonstrate this empirically.



(a) Case 1. **(b)** Case 2. **(c)** Case 3.

(d) Case 4. **(e)** Case 5. **(f)** Case 6.

Figure 4.5: ROC curves in Figure 4.4 cutoff at 10% FPR for each test case with synthetic data. For clarity, Table 4.3 holds the values of each model's TPR $(\%)$ at 10% FPR.

Clarifying Edge Cases

Test case 4 was chosen to illustrate the sort of edge cases that might arise in data sets, which either lead to detections with zero or some false positives.

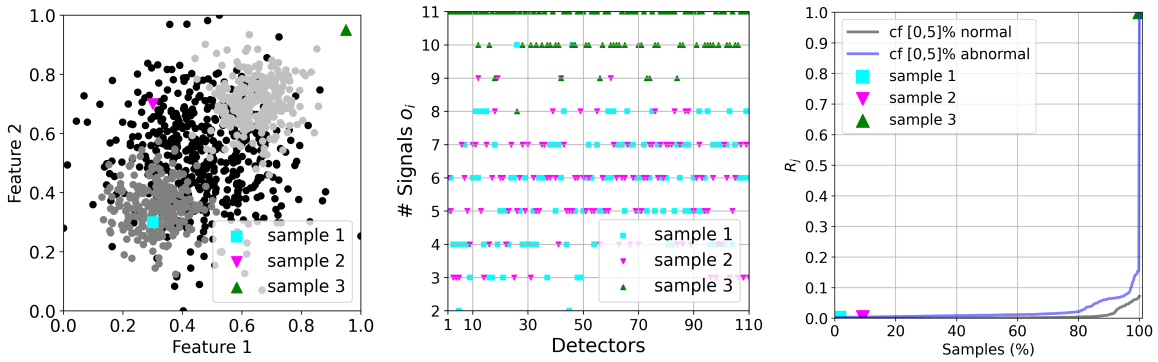
It is clear from Figure 4.6, that the samples detectors collectively see as having almost all the signals as signals o_i will generate strong responses which approach $R_j = 1.0$, as

Table 4.3: Average TPR (%) at 10% FPR and corresponding standard deviation, for each test case for each model.

Case	1		2		3		4		5		6	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
CF $[0, 5]\%$	100	0.0	70.4	1.7	100	0.0	28.6	12.4	100	0.0	4.60	0.26
CF $[0, 20]\%$	100	0.0	71.6	2.0	100	0.0	25.1	2.0	100	0.0	3.42	0.37
OCSVM	100	0.0	93.0	0.7	100	0.0	99.4	0.0	100	0.0	34.8	2.5
IF	100	0.0	88.3	1.8	100	0.0	99.3	0.2	100	0.0	17.0	2.4

seen with Sample 3, whereas if only approximately half or less of the signals are signals o_i, weaker responses will ensue, approaching $R_j = 0.0$. Since in this case the abnormal samples 1 and 2 share most of the feature space with normal samples, the calibration done using normal samples will result in weak responses towards them, with both generating identical responses.

In a ROC curve, the responses towards samples 1 and 2 would result in true positives only for a high percentage of false positives, which means they are very hard to effectively detect, whereas Sample 3 would result in a true positive for 0% false positives, which means it is very easy to detect.



(a) **(b)** **(c)**

Figure 4.6: Edge cases that arise in data sets, considering a single run of test case 4. (a) Shows the same clusters as in Figure 4.1d, but with three different abnormal samples introduced manually by substituting three abnormal samples in the original data set. Sample 1 has all its features equal and inside Cluster 1. Sample 2 has all its odd features equal and inside Cluster 1, and all its even features equal and inside Cluster 2. Sample 3 has all its features equal and outside both clusters. (b) Shows the quantity of signals o_i each detector sees in each sample. (c) Shows the responses each sample provoked.

4.1.4 Test Case 6

The scenario in this case is a very hard one to tackle, since not only do the normal and abnormal samples overlap completely, but the normal samples are also quite dispersed. It is obvious that training any model in these conditions will be difficult. Nevertheless, this test tries to show how far these methods can be taken before failing. As per Figure 4.4f, it becomes clear that all methods struggled, which means they performed similarly to a random classifier or even worse than that, with OCSVM still being able to make some

detections with low FPR. This is likely due to the fact that the RBF kernel can still easily define decision boundaries due to the Gaussian nature of the normal samples, despite this, the overlap between normal and abnormal samples in the test set does not allow a good performance, similar to the ones in cases 1, 3, and 5. The same cannot be said for the remaining methods, since it is apparent that they either performed similar to a random classifier, in the case of the IF, or even worse than that, in the case of the CF model. For the IF it is hard to distinguish abnormal samples from normal ones with its partitioning technique with so much overlap, hence the poor results. For the CF model it is clear that the detectors involved cannot accurately partition the feature space to separate abnormal and normal samples, because of the overlap between these samples. Therefore, they are not able to make any sort of relevant distinction between normal and abnormal samples. Even worse, the model starts classifying samples as opposite to their true class, which means the model learned that normal samples are in fact abnormal. Figure 4.2f and Figure 4.3f explain why this happened. Since detectors see an equal amount of signals o_i, or on average one more, in normal samples compared to abnormal ones, their responses will be stronger towards normal samples, with response curves towards normal samples being above the response curves towards abnormal samples. Approaching rates of 100% false positives all models were able to eventually make detections at rates approaching 100% true positives, but this fact is not very useful since a model that makes detections with this sort of trade-off is impractical for use.

4.2 Tests With A Real Data Set

After testing the anomaly detection methods with synthetic data, real world data was used to test their performances. The data set used was the white wines data set in [37], which has 4898 samples with 11 features, based on physicochemical tests, and 1 label, based on sensory data, with these being: 1 - fixed acidity; 2 - volatile acidity; 3 - citric acid; 4 - residual sugar; 5 - chlorides; 6 - free sulfur dioxide; 7 - total sulfur dioxide; 8 - density; 9 - pH; 10 - sulphates; 11 - alcohol; 12 - quality, which is the label of the data, more specifically being a score between 0 (worst) and 10 (best).

The data labels were not used to train the models, since as mentioned, the anomaly detection models tested are unsupervised. Nevertheless, the labeled data was useful to separate the wines into classes, therefore creating subsets of the original data set, which were used to test the models with different classes of wines.

Unfortunately this data set is not balanced, therefore training the models using individual classes of wines is impractical. This can be seen in Table 4.4, where it is clear that some wine classes have a lot more samples than others. The solution found was to simply aggregate classes of wines into new classes that better represent the data set as a whole. The new classes considered are as follows: classes 3, 4 and 5 become the Worst class; classes 4, 5 and 6, become the Bad class; classes 5, 6 and 7, become the Moderate class; classes 6, 7 and 8, become the Good class; classes 7, 8 and 9, become the Best class.

Table 4.4: Number of samples, N_S, in each original wine class.

Class	3	4	5	6	7	8	9
N_S	20	163	1457	2198	880	175	5

The total number of samples each new class of wines has, that was used for training the models, can be seen in Table 4.5, along with the number of samples used to test the models, which essentially are the remaining samples belonging to the other classes of wines with respect to the class being used for training.

These data sets were always normalized according to Equation 2.2 before using them with OCSVM, IF, and K-means. Each wine class test was run five times with random normal samples selected for training and testing, in order to average results to avoid influences from statistical fluctuations. All methods were cross validated with exactly the same samples. Before training the CFSs, the training samples were clustered using the K-means clustering method, with $K = 4$, determined through the elbow method discussed in chapter 2.

The results for each wine class test using each anomaly detection method, can be seen in Figure 4.9 and Figure 4.10. For the CF model, the responses curves and the number of signals o_i detectors see in samples, can be seen in Figure 4.7 and Figure 4.8, respectively. Based on these results a brief discussion for each wine class test will follow.

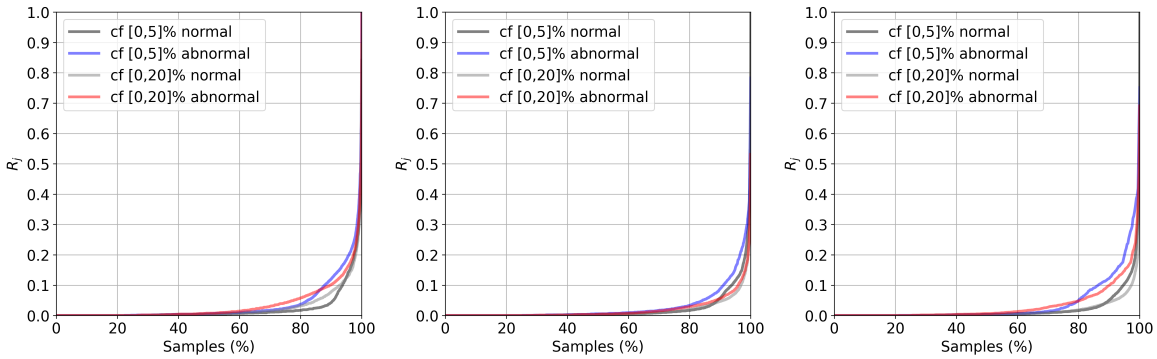
Table 4.5: Number of normal samples, N_n, in each wine class during training and testing, and number of abnormal samples, N_a, during testing.

	Class	Worst	Bad	Moderate	Good	Best
Training	N_n	500	500	500	500	500
Testing	N_n	1140	3318	4035	2753	560
	N_a	3258	1080	363	1645	3838

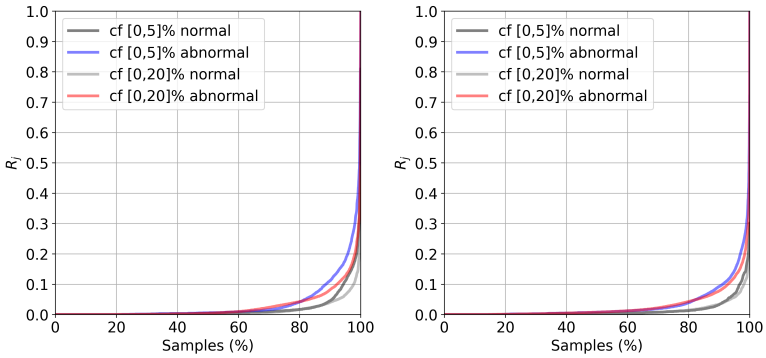
4.2.1 All Wine Classes Tests

All the classes data sets present very difficult scenarios for anomaly detection methods, since in each of them both the normal and abnormal samples overlap a lot, similar to the scenario in Figure 4.1f. This translates to poor performances, similar to a random classifier, as can be seen in Figure 4.9 and Table 4.6, with all methods performing similarly to each other. The OCSVM with the RBF kernel defines decision boundaries not very well suited for the configuration of the samples in the data set, since some of their features' distributions are very different to the Gaussian distribution. Not only this, but the overlapping nature of normal and abnormal samples also leads to false positives. Similarly, the IF has difficulty in generating forests of isolation trees that successfully partition abnormal samples closer to their roots. The CF model shares the same issues, since it is difficult to train detectors to recognize abnormal samples when they completely overlap with normal ones during testing.

Both the CF $[0, 5]\%$ and CF $[0, 20]\%$ CFSs performed similarly, regarding the responses towards normal and abnormal samples, and the number of signals o_i detectors saw in these samples. This can be seen in Figure 4.7 and Figure 4.8. In all tests the ratio of the number of signals o_i seen in abnormal samples to normal ones is approximately one. This backs the hypothesis that due to the overlapping nature of the data, training the models to distinguish between normal and abnormal samples is very hard, with any detections made coming with the trade-off of FPR. For detections to occur in these situations, the several detection mechanisms must be used, although this must be showed empirically in the future. In fact, comparing Figure 4.7 and Figure 4.8 with Figure 4.2d, Figure 4.2f, Figure 4.3d, and



(a) Worst. **(b)** Bad. **(c)** Moderate.



(d) Good. **(e)** Best.

Figure 4.7: Detectors' normalized average responses for each wine class test. For the corresponding ROC curves see Figure 4.9. Regarding the averaging of the responses and ROC curves, the same explanation in Figure 4.2 applies here.

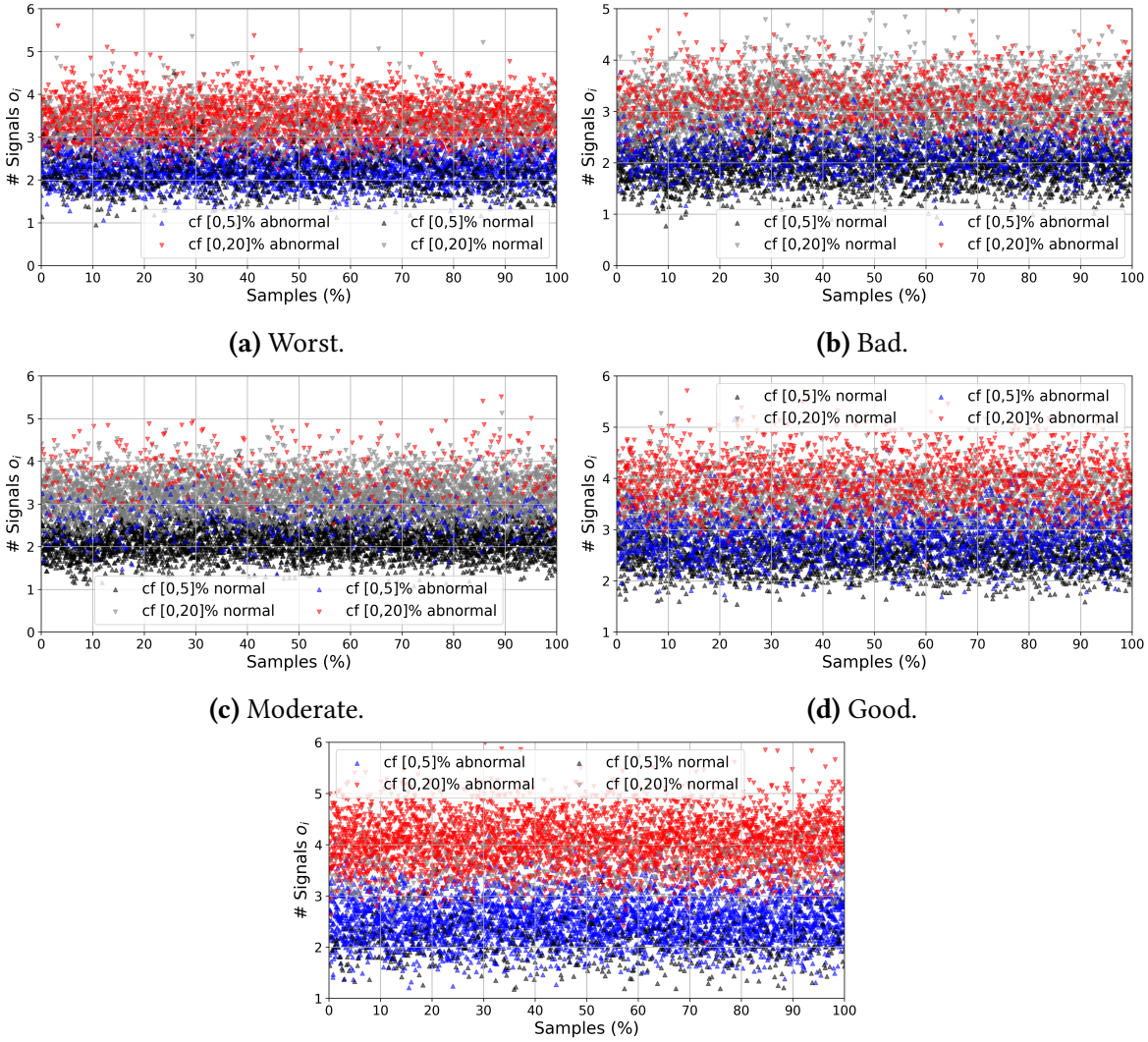
Figure 4.3f, it becomes apparent that the same issues arise in these wine classes tests as in tests 4 and 6 with synthetic data.

Table 4.6: Average AUC $(\%)$ and corresponding standard deviation, for each wine class test for each model.

Class	Worst		Bad		Moderate		Good		Best	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
CF $[0,5]\%$	54.9	4.0	55.3	1.8	53.5	0.9	55.7	3.2	62.4	2.1
CF $[0,20]\%$	55.3	0.9	55.6	0.5	59.8	1.0	59.6	1.8	64.2	1.0
OCSVM	41.2	0.5	47.1	0.5	63.3	0.3	56.2	0.3	50.5	1.2
IF	43.4	0.8	51.7	1.1	64.3	0.8	56.1	1.6	52.4	1.1

4.3 Comparison With Previous Cellular Frustration Models

In this final section, a brief comparison will be made between the detection rates achieved with each version of the CF model, at 10% FPR for each wine class test. These results can be seen in Table 4.8. Note that the standard deviations of all tests are missing for the CF in [36],



(a) Worst.

(b) Bad.

(c) Moderate.

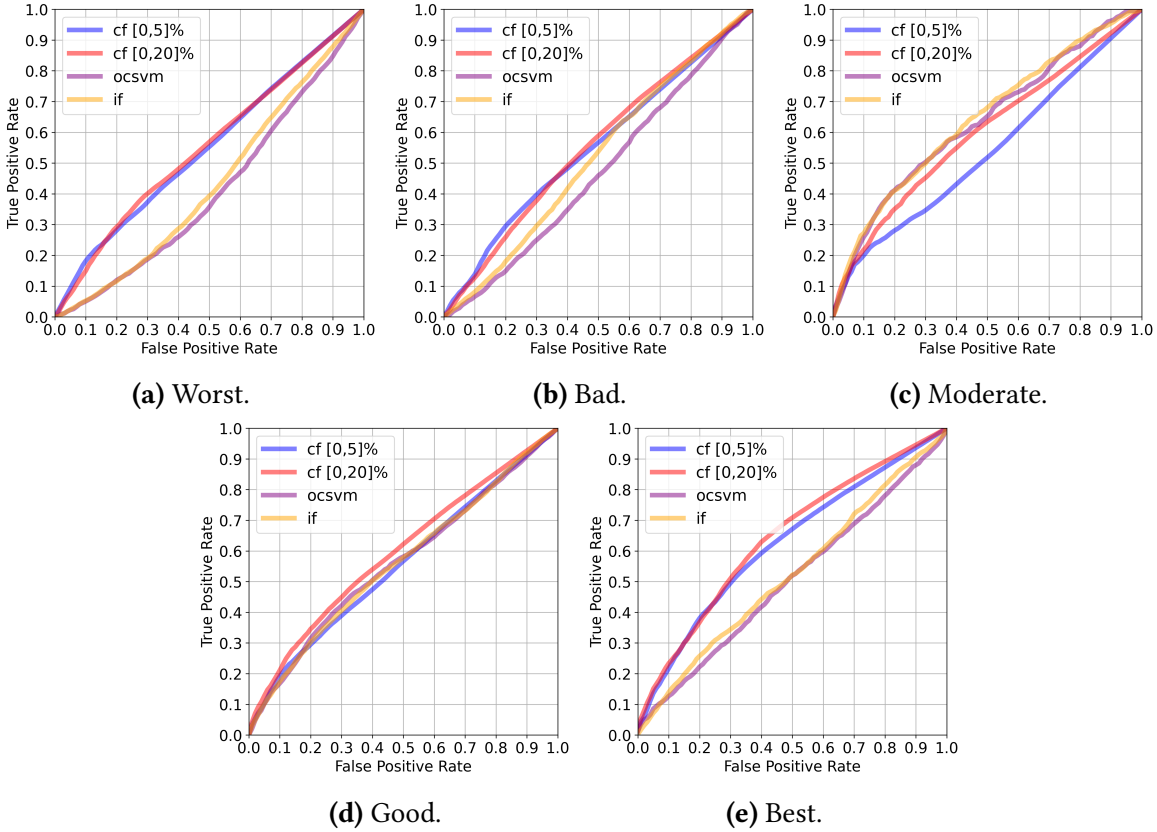
(d) Good.

(e) Best.

Figure 4.8: Average number of signals out o_i seen per detector in each normal and abnormal sample for each wine class test. Each point corresponds to a sample, with all samples being sorted prior to averaging in the same order for each test as in Figure 4.7. Note that the simple reason why some figures have visibly more points of certain colors and less of others, is because the normal and abnormal samples in the test sets are unbalanced, as is shown in Table 4.5.

which makes it hard to understand if those results are in range of the standard deviations of the other CF versions, or how consistent they were.

Overall the results across all three versions of the CF model are very similar, being within standard deviations of each other, except for the Moderate wine class test, which was 2.1% worse with the current version, considering the best case scenario regarding the standard deviation for CF $[0, 20]\%$ and worst case scenario for CF in [7]. This was expected since the current model did not remove the prior mechanisms for detection, nor decision rules that govern the CF dynamic. The current model adds new decision rules that allow the creation and use of detectors with two tails instead of one, which before limited the



(a) Worst. **(b)** Bad. **(c)** Moderate.

(d) Good. **(e)** Best.

Figure 4.9: ROC curves for each wine class test. The OCSVM and IF methods were used with the default parameters of the library. For OCSVM, $k(x, x') = e^{-\gamma\|x-x'\|^2}$, $\gamma = \frac{1}{N_f \sigma^2}$, and $\nu = 0.5$. For IF, $t = 100$, and $\psi = \min\{256, |S|\}$. The CF method was used with $N_P = N_D = 110$, $n_{max} = 10^6$, and $\nu_{max} = 5\%$ for CF $[0, 5]\%$, and $\nu_{max} = 20\%$ for CF $[0, 20]\%$. For the AUC scores see Table 4.6, and for a zoomed in view between 0% and 10% FPR see Figure 4.10.

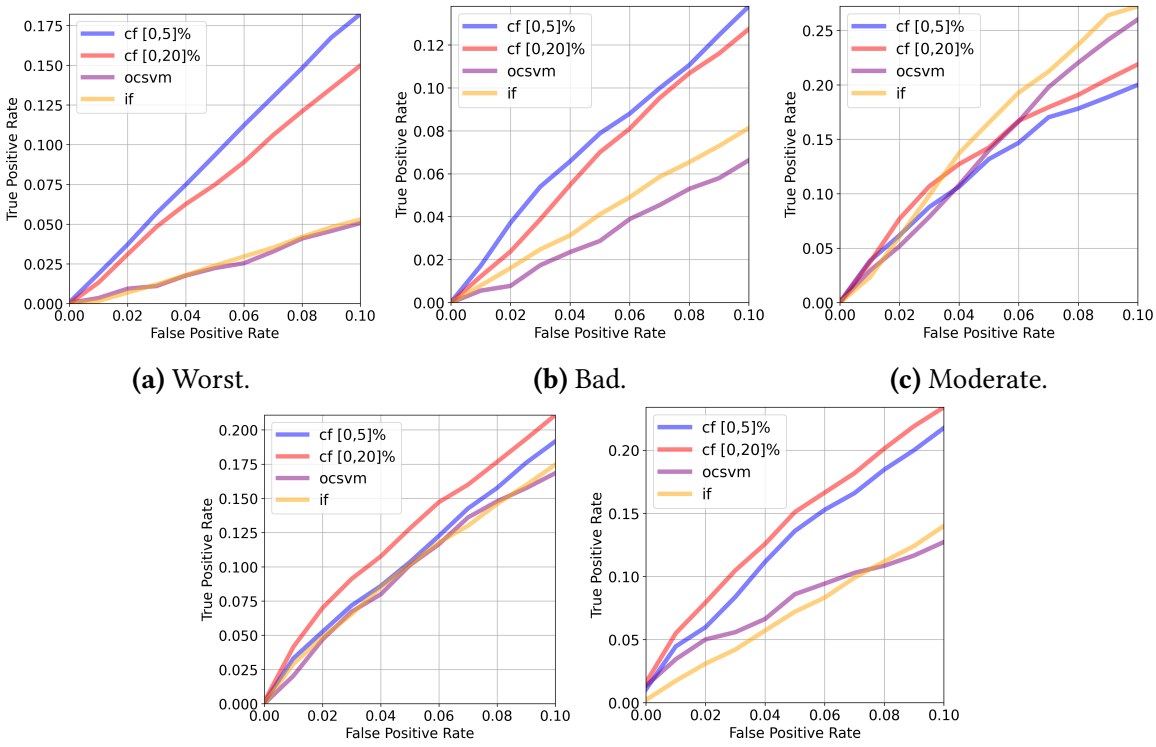
Table 4.7: Average TPR $(\%)$ at 10% FPR and corresponding standard deviation, for each wine class test for each model.

Class	Worst		Bad		Moderate		Good		Best	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
CF $[0, 5]\%$	18.2	4.6	13.8	1.6	20.0	0.8	19.2	1.7	21.8	1.7
CF $[0, 20]\%$	15.0	3.6	12.7	1.1	21.9	1.7	21.1	1.9	23.4	1.2
OCSVM	5.06	0.13	6.63	0.05	26.0	0.6	16.8	0.2	12.7	0.8
IF	5.29	0.61	8.13	0.54	27.2	1.2	17.5	1.5	14.0	2.4

application of the model to only certain data sets, or otherwise required training separate populations of detectors to achieve the same results.

The differences in results can be attributed to: the new decision rules that govern the agents' interactions; the different ways the CFSs were calibrated; the parameters considered for the CF model; the stochastic nature of the model; the randomly selected samples used for training and testing the CFSs.

Further analysis can be done to the current version of the CF model, like the ones in [7],



(a) Worst. **(b)** Bad. **(c)** Moderate.

(d) Good. **(e)** Best.

Figure 4.10: ROC curves in Figure 4.9 cutoff at 10% FPR for each wine class test. For clarity, Table 4.7 holds the values of each model's TPR $(\%)$ at 10% FPR.

Table 4.8: Average TPR $(\%)$ at 10% FPR and corresponding standard deviation, for each wine class test for each CF model version.

Class	Worst		Bad		Moderate		Good		Best	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
CF $[0, 5]\%$	18.2	4.6	13.8	1.6	20.0	0.8	19.2	1.7	21.8	1.7
CF $[0, 20]\%$	15.0	3.6	12.7	1.1	21.9	1.7	21.1	1.9	23.4	1.2
CF in [36]	16.2	—	14.1	—	26.7	—	22.3	—	23.1	—
CF in [7]	16.5	4.2	15.3	2.9	27.4	1.7	20.2	2.2	19.7	2.2

but this will be left as future work. There is some evidence that the current model has not yet reached its final form, like the fact that detectors that are too specialized in some region of the feature space, will see normal samples from other regions as abnormal, leading to false positives if a large percentage of detectors incur in that mistake. To solve this, instead of two symmetric abnormal domains, there could be two independent abnormal domains, each with its signal o_i, i.e., a signal $o_{i,l}$ in the left tail, and a signal $o_{i,r}$ in the right tail. Hopefully this will allow the detectors to more clearly distinguish abnormal samples from normal ones to their left and right, instead of identifying them as the same type of anomaly regardless of the side the sample appears on.

Regarding the K-means clustering technique used, it should be obvious that this is not the best option for clustering data sets, either for the CF model or when working with many other ML methods. It was used due to its simplicity and adequate results in this context,

but more advanced and fitting means of clustering data can be used, as was discussed in [36].

One final note regarding future improvements to the CF model is that the shuffling procedure introduced is not finalized, with this being an introductory version of it for the purpose of helping to solve the problem that arises when detectors are too specialized in a certain region of space. A more deliberate treatment of the configuration of detectors' lists of critical values must be developed in future work.

CHAPTER 5

Conclusion

This work described how improvements to the decision rules of the CF model and how detectors look at samples, allow it to be applied in a more generalized way when dealing with data sets with varying data configurations, especially if data are clustered in separate regions of the feature space. The data considered here were synthetic and real in nature, in order to test the models in a controlled environment and also in a real world environment where conditions are less predictable and have more nuance, respectively. This work also showed that the gains in the results regarding synthetic data in [36], were also maintained without resorting to a particular clustering technique and applying the CF model separately to each cluster of normal data in order to train multiple populations of detectors. Finally, as in previous works, not only was the OCSVM method tested and compared with the current CF model, but in addition to it the IF method was also tested.

The results of the synthetic data tests clearly show that in relation to the CF model in [7], the current model can handle more situations regarding the distribution of data in feature space, especially when abnormal samples lie in between normal samples. Regarding the CF model in [36] identical results were obtained, but with a more streamlined application of CFSs, since the model can now handle clustered data independently of its configuration in feature space, with only one population of trained detectors. Previously there was the need to run two independent CFSs to be able to detect abnormal samples in between regions of space with normal samples. Now, a single CFS can handle the same situation. In relation to the competing anomaly detection methods, OCSVM and IF, it also became clear that in most situations CFSs are capable of comparable results.

The results of the real data tests clearly show that overall the current CF model performs comparably to the previous ones. As mentioned, there are drawbacks to the current model, but these can be solved in future iterations as there is already some idea of the kinds of solutions that can be implemented, as discussed in the previous chapter. In relation to the OCSVM and IF methods, for the default parameters of the Scikit Learn library, the CF model performed between 1.1 to 3.6 times better regarding detection rates at 10% FPR, in four out of the five classes, with the remaining class test showing the OCSVM and IF performed approximately 1.2 times better than the CF model.

The initial goal of this work of further improving the CF model was achieved with results confirming this. Still, more research into the mechanisms of the model and improvements to the current model implementation must be done. Especially regarding the way

the partitioning of feature space is done and how detectors are assigned to certain regions of space, and how they identify normal and abnormal regions of the feature space.

Bibliography

[1] J. L. Joseph, V. A. Kumar, and S. P. Mathew, "Fruit classification using deep learning," in *Innovations in Electrical and Electronic Engineering*, S. Mekhilef, M. Favorskaya, R. K. Pandey, and S. R. Nath, Eds., Singapore: Springer Singapore, 2021, pp. 807–817, ISBN: 978-981-16-0749-3.

[2] S. Ravikumar and P. Saraf, "Prediction of stock prices using machine learning (regression, classification) algorithms," in *2020 International Conference for Emerging Technology (INCET)*, 2020, pp. 1–5. DOI: 10.1109/INCET49848.2020.9154061.

[3] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, Jul. 2009, ISSN: 0360-0300. DOI: 10.1145/1541880.1541882.

[4] C. Spence, L. Parra, and P. Sajda, "Detection, synthesis and compression in mammographic image analysis with a hierarchical image probability model," in *Proceedings IEEE Workshop on Mathematical Methods in Biomedical Image Analysis (MMBIA 2001)*, 2001, pp. 3–10. DOI: 10.1109/MMBIA.2001.991693.

[5] V. Kumar, "Parallel and distributed computing for cybersecurity," *IEEE Distributed Systems Online*, vol. 6, no. 10, 2005. DOI: 10.1109/MDSO.2005.53.

[6] E. Aleskerov, B. Freisleben, and B. Rao, "Cardwatch: A neural network based database mining system for credit card fraud detection," in *Proceedings of the IEEE/IAFE 1997 Computational Intelligence for Financial Engineering (CIFEr)*, 1997, pp. 220–226. DOI: 10.1109/CIFER.1997.618940.

[7] B. Faria and F. V. de Abreu, "Cellular frustration algorithms for anomaly detection applications," *PLOS ONE*, vol. 14, no. 7, e0218930, 2019. DOI: https://doi.org/10.1371/journal.pone.0218930.

[8] J. Han, M. Kamber, and J. Pei, "3 - data preprocessing," in *Data Mining (Third Edition)*, ser. The Morgan Kaufmann Series in Data Management Systems, J. Han, M. Kamber, and J. Pei, Eds., Third Edition, Boston: Morgan Kaufmann, 2012, pp. 113–114, ISBN: 978-0-12-381479-1. DOI: https://doi.org/10.1016/B978-0-12-381479-1.00003-4.

[9] B. Heisele, "Visual object recognition with supervised learning," *IEEE Intelligent Systems*, vol. 18, no. 3, pp. 38–42, 2003. DOI: 10.1109/MIS.2003.1200726.

[10] N. Boodhun and M. Jayabalan, "Risk prediction in life insurance industry using supervised learning algorithms," *Complex & Intelligent Systems*, Apr. 2018. DOI: https://doi.org/10.1007/s40747-018-0072-1. [Online]. Available: https://link.springer.com/article/10.1007/s40747-018-0072-1#citeas.

[11] M. A. Morid, K. Kawamoto, T. Ault, J. Dorius, and S. Abdelrahman, *Supervised learning methods for predicting healthcare costs: Systematic literature review and empirical evaluation*, Apr. 2018. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5977561/.

[12] I. C. Education, *What is supervised learning?* Aug. 2020. [Online]. Available: https://www.ibm.com/cloud/learn/supervised-learning.

[13] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006, pp. 138–144, ISBN: 0387310738.

[14] ——, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006, pp. 179–180, 197, 206, ISBN: 0387310738.

[15] ——, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006, pp. 326–330, ISBN: 0387310738.

[16] ——, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006, pp. 331–333, ISBN: 0387310738.

[17] J. Han, M. Kamber, and J. Pei, "9 - classification: Advanced methods," in *Data Mining (Third Edition)*, ser. The Morgan Kaufmann Series in Data Management Systems, J. Han, M. Kamber, and J. Pei, Eds., Third Edition, Boston: Morgan Kaufmann, 2012, pp. 413–415, ISBN: 978-0-12-381479-1. DOI: https://doi.org/10.1016/B978-0-12-381479-1.00009-5.

[18] X. Hu, J. Tang, H. Gao, and H. Liu, "Unsupervised sentiment analysis with emotional signals," in *Proceedings of the 22nd International Conference on World Wide Web*, ser. WWW '13, Rio de Janeiro, Brazil: Association for Computing Machinery, 2013, pp. 607–618, ISBN: 9781450320351. DOI: 10.1145/2488388.2488442. [Online]. Available: https://doi.org/10.1145/2488388.2488442.

[19] N. Soares, E. P. de Aguiar, A. C. Souza, and L. Goliatt, "Unsupervised machine learning techniques to prevent faults in railroad switch machines," *International Journal of Critical Infrastructure Protection*, vol. 33, p. 100 423, 2021, ISSN: 1874-5482. DOI: https://doi.org/10.1016/j.ijcip.2021.100423.

[20] S. Cao, N. Yang, and Z. Liu, "Online news recommender based on stacked autoencoder," in *2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS)*, 2017, pp. 721–726. DOI: 10.1109/ICIS.2017.7960088.

[21] I. C. Education, *What is unsupervised learning?* Sep. 2020. [Online]. Available: https://www.ibm.com/cloud/learn/unsupervised-learning.

[22] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006, pp. 424–425, ISBN: 0387310738.

[23] J. Han, M. Kamber, and J. Pei, "10 - cluster analysis: Basic concepts and methods," in *Data Mining (Third Edition)*, ser. The Morgan Kaufmann Series in Data Management Systems, J. Han, M. Kamber, and J. Pei, Eds., Third Edition, Boston: Morgan Kaufmann, 2012, p. 486, ISBN: 978-0-12-381479-1. DOI: https://doi.org/10.1016/B978-0-12-381479-1.00010-1.

[24] B. Schölkopf, R. Williamson, A. Smola, J. Shawe-Taylor, and J. Platt, "Support vector method for novelty detection," in *Proceedings of the 12th International Conference on Neural Information Processing Systems*, ser. NIPS'99, Denver, CO: MIT Press, 1999, pp. 582–588.

[25] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *2008 Eighth IEEE International Conference on Data Mining*, 2008, pp. 413–422. DOI: 10.1109/ICDM.2008.17.

[26] "Mean absolute error," in *Encyclopedia of Machine Learning and Data Mining*, C. Sammut and G. I. Webb, Eds. Boston, MA: Springer US, 2017, p. 806, ISBN: 978-1-4899-7687-1. DOI: https://doi.org/10.1007/978-1-4899-7687-1_953.

[27] "Mean squared error," in *Encyclopedia of Machine Learning and Data Mining*, C. Sammut and G. I. Webb, Eds. Boston, MA: Springer US, 2017, p. 808, ISBN: 978-1-4899-7687-1. DOI: https://doi.org/10.1007/978-1-4899-7687-1_528.

[28] D. Powers, "Evaluation: From precision, recall and f-measure to roc, informedness markedness & correlation," English, *Journal of Machine Learning Technologies*, vol. 2, no. 1, pp. 37–63, 2011, ISSN: 2229-3981.

[29] K. Atkinson, *An Introduction to Numerical Analysis*. Wiley, 1991, pp. 251–253, ISBN: 978-0-471-62489-9.

[30] "Area under curve," in *Encyclopedia of Machine Learning and Data Mining*, C. Sammut and G. I. Webb, Eds. Boston, MA: Springer US, 2017, pp. 61, 1112, ISBN: 978-1-4899-7687-1. DOI: 10.1007/978-1-4899-7687-1_918. [Online]. Available: https://doi.org/10.1007/978-1-4899-7687-1_918.

[31] D. Gale and L. S. Shapley, "College admissions and the stability of marriage," *The American Mathematical Monthly*, vol. 69, no. 1, pp. 9–15, 1962. [Online]. Available: http://www.jstor.org/stable/2312726.

[32] F. Vistulo de Abreu, E. N. M. Nolte'Hoen, C. R. Almeida, and D. M. Davis, "Cellular frustration: A new conceptual framework for understanding cell-mediated immune responses," in *Artificial Immune Systems*, H. Bersini and J. Carneiro, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 37–51, ISBN: 978-3-540-37751-1.

[33] B. Faria, "Developments of a new artificial intelligence approach for anomaly detection," Ph.D. dissertation, May 2017, pp. 43–46. [Online]. Available: http://hdl.handle.net/10773/21126.

[34] B. Davies, "The laplace transform," in *Integral Transforms and Their Applications*. New York, NY: Springer New York, 2002, p. 28, ISBN: 978-1-4684-9283-5. DOI: 10.1007/978-1-4684-9283-5_2.

[35] F. Pedregosa, G. Varoquaux, A. Gramfort, *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[36] D. B. d. Carvalho, *Clustering in cellular frustration models*, Jul. 2018. [Online]. Available: http://hdl.handle.net/10773/25202.

[37] *Wine quality data set*, Oct. 2009. [Online]. Available: http://archive.ics.uci.edu/ml/datasets/Wine+Quality.

[38] P. Cortez, A. Cerdeira, F. Almeida, T. Matos, and J. Reis, "Modeling wine preferences by data mining from physicochemical properties," *Decision Support Systems*, vol. 47, no. 4, pp. 547–553, 2009, ISSN: 0167-9236. DOI: https://doi.org/10.1016/j.dss.2009.05.016.

[39] X. Zhang, "Gaussian distribution," in *Encyclopedia of Machine Learning and Data Mining*, C. Sammut and G. I. Webb, Eds. Boston, MA: Springer US, 2017, p. 531, ISBN: 978-1-4899-7687-1. DOI: 10.1007/978-1-4899-7687-1_107. [Online]. Available: https://doi.org/10.1007/978-1-4899-7687-1_107.