

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO CEARÁ
PRÓ-REITORIA DE PESQUISA, PÓS-GRADUAÇÃO E INOVAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Francisco Mateus Rocha Filho

**Logic-based explainability for reject option-based classifiers and
expanding minimal explanations for standard classifiers**

Fortaleza – CE

4 de setembro de 2024

Francisco Mateus Rocha Filho

**Logic-based explainability for reject option-based classifiers
and expanding minimal explanations for standard classifiers**

Dissertação submetida ao Programa de Pós-Graduação em Ciência da Computação do Instituto Federal de Educação, Ciência e Tecnologia do Ceará, como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Linha de Pesquisa: Inteligência Artificial

Orientador: Dr. Ajalmar Rêgo da Rocha Neto

Co-orientador: Dr. Thiago Alves Rocha

Instituto Federal de Educação, Ciência e Tecnologia do Ceará (IFCE)

Pró-Reitoria de Pesquisa, Pós-Graduação e Inovação (PRPI)

Programa de Pós-Graduação em Ciência da Computação (PPGCC)

Fortaleza – CE

4 de setembro de 2024

Dados Internacionais de Catalogação na Publicação
Instituto Federal do Ceará - IFCE
Sistema de Bibliotecas - SIBI
Ficha catalográfica elaborada pelo SIBI/IFCE, com os dados fornecidos pelo(a) autor(a)

F478l Filho, Francisco Mateus Rocha.
Logic-based explainability for reject option-based classifiers and expanding minimal explanations for standard classifiers / Francisco Mateus Rocha Filho. - 2024.
73 f. : il. color.

Dissertação (Mestrado) - Instituto Federal do Ceará, Mestrado em Ciência da Computação, Campus Fortaleza, 2024.
Orientação: Prof. Dr. Ajalmar Rêgo da Rocha Neto.
Coorientação: Prof. Dr. Thiago Alves Rocha.

1. Explicações baseadas em lógica. 2. Inteligência Artificial Explicável. 3. Máquina de vetores de suporte. 4. Perceptron multicamadas. 5. Opção de rejeição. I. Título.

CDD 004



Instituto Federal de Educação, Ciência e Tecnologia do Ceará (IFCE)
Pró-Reitoria de Pesquisa, Pós-Graduação e Inovação (PRPI)
Programa de Pós-Graduação em Ciência da Computação (PPGCC)

Francisco Mateus Rocha Filho

Esta dissertação foi julgada adequada para a obtenção do título de Mestre em Ciência da Computação, sendo aprovada pela Coordenação do Programa de Pós-Graduação em Ciência da Computação do Instituto Federal de Educação, Ciência e Tecnologia do Ceará e pela banca examinadora:

Orientador: Dr. Ajalmar Rêgo da Rocha Neto
Instituto Federal de Educação, Ciência e Tecnologia do
Ceará (IFCE)

Co-orientador: Dr. Thiago Alves Rocha
Instituto Federal de Educação, Ciência e Tecnologia do
Ceará (IFCE)

Dr. João Fernando Lima Alcântara
Universidade Federal do Ceará (UFC)

Dr. João Paulo Pordeus Gomes
Universidade Federal do Ceará (UFC)

Fortaleza – CE
4 de setembro de 2024

*This work is dedicated to God and to my family, as they are my strongest roots, anchor,
and place of rest.*

Acknowledgements

Firstly, I would like to express my gratitude to God for the opportunity to pursue my master's degree. Without him, none of my major achievements would have been possible, and they would lack true significance without his divine presence, kindness, and permission.

I want to thank the Federal Institute of Ceará (IFCE), the Cearense Foundation for Support to Scientific and Technological Development (FUNCAP), and the Postgraduate Program in Computer Science (PPGCC) for providing unwavering support and encouragement throughout this journey. They have been of crucial importance, enabling the realization of fruitful research.

I also extend my gratitude to each member of my research group, who consistently encouraged me to give my best and strive for nothing less than excellence. Special thanks to fellow master's students Francisco Acélio Cunha de Sousa and Reginaldo Pereira Fernandes Ribeiro, for engaging in conversations within and beyond the realm of research, and for alleviating the pressures of each day. In particular, I would like to express my appreciation to Dr. Ajalmar Rêgo da Rocha Neto and Dr. Thiago Alves Rocha, for their excellent advice, insightful ideas, incredible guidance, and unwavering support throughout the execution of my master's program.

*“If I make use of the tongues of men and of angels, and have not love, I am like sounding
brass, or a loud-tongued bell.”
(1 Corinthians 13:1)*

Resumo

Os modelos de aprendizagem de máquina, como a máquina de vetores de suporte, são frequentemente usados para solucionar problemas de mundo real, auxiliando em tarefas de tomada de decisão. Eles são regularmente usados em conjunto com estratégias para mitigar o custo de classificações incorretas, como a estratégia de opção de rejeição para rejeitar instâncias difíceis de classificar e delegá-las a um especialista ou até para outro classificador. Isso pode aumentar a confiança do modelo. Explicações podem ser usadas para destacar as razões para a rejeição, sendo importantes para evitar confiar cegamente nos resultados obtidos pelo modelo. Embora a maioria dos trabalhos relacionados tenha desenvolvido meios para gerar explicações para modelos de aprendizagem de máquina, há uma falta de trabalho considerando a presença de opção de rejeição. Além disso, muitos métodos não garantem a corretude e a minimalidade das explicações. Tais métodos também podem ter capacidades de generalização insuficientes, com explicações muito sucintas e que não fornecem entendimentos suficientes sobre o modelo.

Para abordar essas questões, apresentamos duas propostas principais. A primeira é uma abordagem baseada em lógica com garantias formais sobre a corretude e a minimalidade das explicações para classificadores com opção de rejeição. São geradas explicações para um classificador de vetores de suporte linear com opção de rejeição, nosso modelo-alvo. A avaliação da abordagem é feita por meio de comparação com um algoritmo heurístico para geração de explicações, chamado Anchors. Os resultados obtidos mostram que nosso método proposto fornece explicações mais concisas e com custo de tempo reduzido. Além disso, embora nossa abordagem seja demonstrada com classificadores de vetores de suporte lineares, ela pode ser facilmente adaptada para outros classificadores com opção de rejeição, como redes neurais e florestas aleatórias.

Nossa segunda proposta concentra-se em fornecer uma maior capacidade de generalização para as explicações geradas. A generalização é importante, pois explicações mais gerais podem ser aplicadas a um maior número de instâncias e são capazes de trazer uma compreensão mais aprofundada de como o modelo funciona. Isso é feito em duas abordagens: a Onestep, que gera explicações em um passo, e a Twostep, que gera explicações em dois passos. A avaliação é feita sobre os modelos perceptron multicamadas e classificador de vetores de suporte linear em um cenário de classificação, sem opção de rejeição. Os resultados mostram que as explicações geradas pelo método Twostep são válidas para mais instâncias, embora muitas vezes ao custo de tempo de computação um pouco maior.

Palavras-chave: Explicações baseadas em lógica. Inteligência Artificial Explicável. Máquina de vetores de suporte. Perceptron multicamadas. Opção de rejeição.

Abstract

Machine learning models, such as the support vector machine, are frequently used to solve real world problems, aiding in decision-making tasks. They are often used in conjunction with strategies to mitigate the cost of misclassification, such as the reject option strategy for rejecting hard-to-classify instances and delegating them to a specialist or even to another classifier. This may increase the confidence of the model. Explanations can be used to highlight the reasons for rejection, and are important to avoid blindly trusting the obtained model results. While most related work has developed means to generate explanations for machine learning models, there is a lack of work considering the presence of reject option. Moreover, many methods lack guarantees of correctness and minimality of explanations. Such methods might also have insufficient generalization capabilities, with explanations being too concise and not providing enough insights regarding the model.

To address these issues, we bring two main proposals. The first one is a logic-based approach with formal guarantees on the correctness and minimality of explanations for classifiers with reject option. We compute explanations for a linear support vector classifier with reject option, our target model. The approach is evaluated through comparison against an heuristic algorithm for generating explanations, named Anchors. The obtained results show that our proposed method gives shorter explanations with reduced time cost. Furthermore, although our approach is demonstrated with linear support vector classifiers, it can be easily adapted to other classifiers with reject option, such as neural networks and random forests.

Our second proposal focuses on providing a higher generalization capability for generated explanations. Generalization is important since more general explanations may be applied over more instances and might bring a more insightful understanding of how the model works. This is done in two approaches: the Onestep, which generates explanations in one step, and the Twostep, which generates explanations in two steps. Evaluation is done over multilayer perceptron and linear support vector classifier models in a classification scenario, without reject option. Results show that explanations generated by the Twostep method hold over more instances, albeit often at the cost of a slightly higher computation time.

Keywords: Logic-based explainable AI. Explainable Artificial Intelligence. Support vector machines. Multilayer Perceptron. Reject Option.

List of Figures

Figure 1 – Support vector classifier hyperplanes.	38
Figure 2 – Classification through features f_1 and f_2	46

List of Tables

Table 1	– Datasets details.	49
Table 2	– Reject region thresholds using training data. Classifier accuracy without reject option, accuracy with reject option, and rejection ratio for test data. Instances by class for each entire dataset.	50
Table 3	– Time comparison between our approach and Anchors (seconds). Less is better.	51
Table 4	– Explanation size comparison between our approach and Anchors. Less is better.	52
Table 5	– Datasets details.	63
Table 6	– Time results for SVM.	64
Table 7	– Time results for MLP.	65
Table 8	– Coverage results between Onestep and Twostep.	65
Table 9	– Explanation ranges results between Onestep and Twostep.	66

List of Algorithms

1	Computing a minimal explanation	48
2	Computing an Onestep expanded explanation	60
3	Computing a Twostep expanded explanation	62

List of abbreviations and acronyms

AI	Artificial Intelligence
BLDT	Blood Transfusion
BRCW	Breast Cancer Wiscounsinn
CBC	Coin-or Branch and Cut
FOL	First Order Logic
IONS	Ionosphere
IRIS	Iris
LP	Linear Programming
MILP	Mixed Integer Linear Program
ML	Machine Learning
MLP	Multilayer Perceptron
NN	Neural Network
PARK	Parkinsons
PIMA	Pima
ReLU	Rectified Linear Unit
RO	Reject Option
SONR	Sonar
SVC	Support Vector Classifier
SVM	Support Vector Machine
SVM-HM	Hard-Margin Support Vector Machine
SVM-SM	Soft-Margin Support Vector Machine
VRTC	Vertebral Column
WINE	Wine
XAI	eXplainable AI

List of symbols

$\mathcal{F}$	Set of features.
$\mathcal{K}$	Set of classes.
c	A class.
$\mathcal{N}$	Number of classes.
D	Set of domain constraints.
$\mathbf{x}$	Feature vector (instance).
$\mathcal{C}$	Classifier.
$\mathcal{L}$	Number of training patterns.
$\mathbf{w}$	Weight vector.
w	Scalar weight element.
W	Weight matrix.
W^k	Weight matrix of layer k .
$\mathbf{b}$	Bias vector.
h_o	Optimal separating Hyperplane.
$\hat{y}$	Prediction output.
w_o	Optimal hyperplane weight vector.
b_o	Optimal hyperplane bias.
$sign$	Sign function.
ξ	Slack variable.
n_k	Number of neurons in layer k .
σ	Non-linear activation function.
$\mathbf{y}^k$	Intermediate value vector of layer k .
$\mathbf{x}^k$	Output value vector of layer k .

$\mathbf{y}_j^k$	j-th scalar element in intermediate value vector of layer k .
$\mathbf{x}_j^k$	j-th scalar element in output value vector of layer k .
τ	Transpose of vector/matrix.
$argmax$	Argmax function.
$\hat{R}$	Risk of rejecting instances.
Er	Ratio of misclassified instances.
w_r	Cost (weight) of rejection.
R	Ratio of rejected training instances.
t_+	Positive class threshold.
t_-	Negative class threshold.
E	An explanation.
E'	A subset of explanation E .
φ	A quantifier-free first-order formula.
ϕ	A quantifier-free first-order formula.
s	Atomic formulas.
z	First-order variable.
$\mathcal{A}$	An assignment.
Γ	Set of formulas.
$\models$	Logical consequence.
P	First-order formula.
C	Regularization parameter.
T	Set of thresholds ranges.
s_j^k	Linear condition of $ReLU$.
G	0-1 MILP formulation.
$\mathbf{o}$	MLP Output layer.
r_j	j-th binary activation variable.

$\mathcal{R}$	Restriction for the prediction not holding.
l	Lower bound.
u	Upper bound.
E_{xp}	Expanded/Generalized explanation.
E_{in}	Intermediate explanation.
p	Feature range regularization parameter.
ϵ	Small real value.

Contents

1	INTRODUCTION	31
1.1	Motivation and Overview	31
1.2	General and Specific Objectives	34
1.3	Scientific Production	34
1.4	Dissertation Structure	35
2	BACKGROUND	37
2.1	Classification Problems	37
2.2	Support Vector Machines	37
2.3	Multilayer Perceptron	39
2.4	Reject Option Classification	40
2.5	XAI and Related Work	41
2.6	First-Order Logic	43
2.7	Conclusions	44
3	LOGIC-BASED EXPLANATIONS FOR LINEAR SUPPORT VECTOR CLASSIFIERS WITH REJECT OPTION	45
3.1	Linear Support Vector Classifiers may not be Instance-based Interpretable	45
3.2	Explanations For Linear SVMs with Reject Option	46
3.3	Experiments	49
3.4	Results	51
3.5	Conclusions	53
4	GENERALIZING LOGIC-BASED EXPLANATIONS	55
4.1	Inflated explanations	55
4.2	Multilayer Perceptron Encoding	56
4.3	Generalizing Explanations	57
4.3.1	Onestep	59
4.3.2	Twostep	60
4.4	Experiments	62
4.5	Results	64
5	CONCLUSIONS AND FUTURE WORKS	69
5.1	Conclusions	69
5.2	Future Work	70

BIBLIOGRAPHY	71
-------------------------------	-----------

1 Introduction

1.1 Motivation and Overview

The increased insertion of artificial intelligence (AI) in the daily lives of people has made more common the usage of machine learning (ML) models (LECUN; BENGIO; HINTON, 2015; BISWAS, 2023). Some of these models, such as the support vector machine (SVM) and multilayer perceptron (MLP), are used for classification problems, making decisions based on given data and returning important information to human specialists on complex decision-making tasks (RAMKUMAR et al., 2021; YANG et al., 2018). Therefore, it is important that the answers given by these models can be trusted, with guarantees that attest they are correct. However, a variety of classification models may come across instances that are difficult to correctly classify, be it due to lack of good data, to feature bias (color, size, gender) (IGNATIEV, 2020) or to noise present on the given instances (WELD; BANSAL, 2018). Concepts of eXplainable AI (XAI) can be used to better understand the answers given by such AI systems, with explainability being fundamental for delivering trustworthy models (IGNATIEV; NARODYTSKA; MARQUES-SILVA, 2019; IGNATIEV, 2020; MARQUES-SILVA; IGNATIEV, 2022; SAMEK; MÜLLER, 2019). This can be used for generating explanations that lead to a better understanding of the decision-making process of the models.

In this work, we consider instance-based explanations (IGNATIEV, 2020). These can provide interpretable insights regarding the decision logic used by the model to determine an answer, highlighting the relevant features of the given instance that influenced the output. The linking of the output to specific instances and their features enables a better understanding of how the model operates and may lead to a more comprehensive analysis of the decision-making process.

The SVC is one of the most well-known ML models. The linear support vector machine, specifically, has been used in a variety of classification problems (CHLAOUA et al., 2019; RICHHARIYA et al., 2020; YANG et al., 2018). However, this model can also fall into the same previously said pitfalls, failing to give a correct classification. As such, the reject option (RO) strategy depicted in Chow (1970) can be used to remedy such cases. Classification with RO is a paradigm that aims to improve reliability in decision support systems by handling more complex cases and avoiding a higher error rate. It involves withholding and rejecting a sufficiently ambiguous classification result, i.e. when the instance in question is situated too close to the decision boundary of the classifier. These cases are separated to be dealt with in another specialized way, be it through other models or with the assistance of a human specialist (MESQUITA et al., 2016). Since

classification with RO comprises a set of techniques, in this work we consider the strategy where the classifier is trained as usual and the rejected cases are determined after the training phase. Usually, this process requires finding a trade-off between the costs of misclassifications and rejections.

The linear SVC can already be globally interpreted to a certain extent based on the decision function, where the most important features are often associated with the highest weights (GUYON et al., 2002). Such analysis, however, is not capable of giving decisive answers on more specific cases (KRAUSE et al., 2017). This gives a margin for questionable explanations, especially when RO is present due to the added level of complexity. Therefore a need for a more thorough, instance-based explanation method.

Most of the efforts in XAI do not have formal guarantees of correctness, therefore not being able to determine whether or not the models can be fully trusted. Model-agnostic methods for generating explanations, such as LIME (RIBEIRO; SINGH; GUESTRIN, 2016), SHAP (LUNDBERG; LEE, 2017), and Anchors (RIBEIRO; SINGH; GUESTRIN, 2018), tend to have intrinsic heuristic characteristics that give a margin for incorrect model interpretations. These usually generate explanations by exploring the local instance space of the given data, which may not be thorough enough and fail to represent the true behavior of the model (IGNATIEV; NARODYTSKA; MARQUES-SILVA, 2020). Moreover, they can be often proven wrong through counterexamples, exposing their contradictions. Further evidence has been found to support the assertion that these methods lack guarantees (DIMANOV et al., 2020; IGNATIEV; NARODYTSKA; MARQUES-SILVA, 2019; IGNATIEV, 2020). Therefore, there is a need for alternative methods that ensure correctness. In recent years, there has been a surge of research dedicated to investigating logic-based XAI for ML models, such as neural networks, naive Bayes, random forests, decision trees, and boosted trees (AUDEMARD et al., 2022; IGNATIEV et al., 2022; IGNATIEV; NARODYTSKA; MARQUES-SILVA, 2019; IZZA; IGNATIEV; MARQUES-SILVA, 2020; MARQUES-SILVA et al., 2020).

Thus, the importance of trustworthy ML models increases the need for logic-based approaches to explain the decisions made by these models (IGNATIEV, 2020; IGNATIEV; NARODYTSKA; MARQUES-SILVA, 2020). The computation of explanations in these approaches rigorously explores the entire feature space. Due to this, such explanations are provably correct and hold for any point in the space, which therefore makes them trustworthy (IGNATIEV, 2020). Moreover, logic-based approaches also have guarantees of minimality. This is important since succinct explanations seem easier to interpret by humans (MILLER, 1956), leading them to focus on the important features, and are redundancy-free. Explanations without redundancies mean they are an irreducible set of features, where removing a feature alters the explanation and leads to a different model output.

An explanation can be seen as a subset of features that form a rule. Whenever the rule is applied, the model outputs will always be the same (IGNATIEV; NARODYTSKA; MARQUES-SILVA, 2019; SHIH; CHOI; DARWICHE, 2018). Most rules are a set of restricted values in the form of *feature = value* and therefore are only satisfied for a narrow set of points in the instance space. This type of explanation usually returns very limited information and may not relay much overall insight about the model. For example, stating that a medical analysis can be applied to patients with a height of 180 cm and an age of 56 is less informative than affirming it is also valid for heights between 175 cm and 185 cm and ages between 50 and 60. Therefore, an explanation that includes features associated with ranges of values is a lot more insightful, since a single explanation can be applied to a variety of cases. It is important to note that generated explanations are not unique, which enables different explanations to be valid for the same instance.

Due to the unreliability of heuristic-based explanations, many recent logic-based approaches have been proposed for a variety of ML models (AUDEMARD et al., 2022; IGNATIEV; NARODYTSKA; MARQUES-SILVA, 2019; IGNATIEV et al., 2022; IZZA; IGNATIEV; MARQUES-SILVA, 2020; MARQUES-SILVA et al., 2020; Rocha Filho et al., 2023). However, there is still a notable scarcity of work related to explanations for when reject option is present. Moreover, there is also a lack of works dealing with guarantees of correctness and minimality that also focus on more general explanations. While Anchors (RIBEIRO; SINGH; GUESTRIN, 2018) returns explanations with some degree of generalization, it is a heuristic method for generating explanations and therefore has some issues regarding giving correct answers. The approach described in Izza et al. (2023) has a focus on generating general explanations, expanding feature ranges through increments to the lower and upper bounds each feature can assume, and therefore guaranteeing correctness. This, however, comes with the possibility of smaller increments being too costly to compute, and bigger ones leading to loss of coverage, where the attributed value range limits do not correspond to the actual maximum and minimum values the feature can assume. Even though we consider a linear SVC with RO in this work, our approach can be adapted for other ML models with RO, such as neural networks. Moreover, FOL can represent the classifiers considered in this work and can provide guarantees of correctness and minimality for explanations.

Given the importance of reject option and logic-based explainability for trustworthy ML models, this work has 2 main proposals. The first is a logic-based approach to explain linear SVCs with reject option. Given that rejected instances may be further analyzed by specialists, explanations regarding the causes of rejections can reduce the workload of these professionals. Our proposal builds on work from the literature of logic-based explainability for traditional ML models, i.e. without reject option (IGNATIEV; NARODYTSKA; MARQUES-SILVA, 2019). We compute explanations for linear SVCs with RO by solving a set of logical constraints, specifically, boolean combinations of linear constraints. This is

achieved through first-order logic (FOL) since the resultant first-order formulas can be represented as a Mixed-Integer Linear Programming (MILP) model, which can be solved with MILP solvers.

We conducted experiments to compare our approach with Anchors, a heuristic method that generates explanations by locally exploring the feature subspace close to a given instance (RIBEIRO; SINGH; GUESTRIN, 2018). The explanations generated by Anchors are expressed as rules designed to highlight important features. Therefore, the resulting explanations of Anchors are similar to the ones computed by our approach.

The second proposal further improves the previous explanation method, with both SVC and MLP as target models in a classification problem scenario without RO. Given the importance of generalization, we expand explanations by attributing a range of values to each of the features. This is done in two different approaches: fully expanding the feature ranges to the allowed limits (Onestep), up to the lower and upper bounds, and partially expanding each feature limit, with a more cadenced distribution, followed by a full expansion (Twostep). Explanations are computed by solving a set of logical constraints. We compare both approaches with each other.

1.2 General and Specific Objectives

The main objective of this work is to *deliver explanations with guarantees of correctness and minimality for machine learning models in a classification problem scenario, with and without the presence of reject option.*

In its most particular sense, the specific objectives of this dissertation are:

- Developing a logic-based method that generates explanations for classification problems.
- Building an iteration of the base method that takes the presence of reject option into account.
- Improving the base method to be able to output more general explanations.
- Evaluating, through experimental results, our proposals and compare them with existing state-of-the-art explanation methods.

1.3 Scientific Production

In the course of this master's project, the following scientific works were approved and made available for publication:

- **Rocha Filho, F. M.**, Rocha, T. A., Ribeiro, R. P. F., Rocha Neto, A. R. **Logic-based explanations for linear support vector classifiers with reject**

option. In: NALDI, M. C.; BIANCHI, R. A. C. (Ed.). *Intelligent Systems*. Cham: Springer Nature Switzerland, 2023. p. 144–159. ISBN 978-3-031-45368-7.

1.4 Dissertation Structure

The remaining parts of this work are organized as follows. Chapter 2 introduces the used notations and definitions, setting the theoretical foundation regarding classification problems, SVC and MLP machine learning models, reject option strategy, first-order logic, and related work in the XAI field. It also advocates for the significance of correctness and explanation expansion. Chapter 3 details the proposed logic-based approach for generating explanations. An SVC is compared against the heuristic method Anchors, in classification scenarios with and without reject option. It also includes the obtained results, as well as discussions regarding possible improvements. Chapter 4 presents two algorithms for generalizing explanations, Onestep and Twostep. A coverage metric is used to compare one against the other. It contains results regarding each algorithm generalization capability, through SVC and MLP ML models. Conclusions and future work related to this work are shown in Chapter 5.

2 Background

In this chapter, we introduce important concepts used throughout this work. Section 2.1 presents important notations regarding classification problems. Section 2.2 and Section 2.3 denote the classification process of the support vector classifier and multilayer perceptron, respectively, while Section 2.4 addresses the reject option strategy. Section 2.5 covers related work in the field of eXplainable AI. Section 2.6 presents first-order logic, which will be used to give explanations with guarantees of correctness.

2.1 Classification Problems

In machine learning, classification problems are defined over a set of n features $\mathcal{F} = \{f_1, \dots, f_n\}$ and a set of $\mathcal{N}$ classes $\mathcal{K} = \{c_1, c_2, \dots, c_{\mathcal{N}}\}$. In this work, we consider that each feature $f_i \in \mathcal{F}$ takes its values x_i from the domain of real numbers. Moreover, each feature f_i has an upper bound u_i and a lower bound l_i such that $l_i \leq x_i \leq u_i$, and its domain is the closed interval $[l_i, u_i]$. This is represented as a set of domain constraints $D = \{l_1 \leq f_1 \leq u_1, l_2 \leq f_2 \leq u_2, \dots, l_n \leq f_n \leq u_n\}$. Furthermore, the notation $\mathbf{x} = \{f_1 = x_1, f_2 = x_2, \dots, f_n = x_n\}$ represents a specific point or instance such that each x_i is in the domain of f_i .

A classifier $\mathcal{C}$ is a function that maps elements in the feature space into the set of classes $\mathcal{K}$. For example, $\mathcal{C}$ can map instance $\{f_1 = x_1, f_2 = x_2, \dots, f_n = x_n\}$ to class c_1 . A classifier is usually obtained by a training process, given as input a training set $\{\mathbf{x}_i, y_i\}_{i=1}^{\mathcal{L}}$ where $\mathbf{x}_i \in \mathbb{R}^n$ is an input vector or instance, $y_i \in \mathcal{K}$ is the respective class label, and $\mathcal{L}$ is the number of training patterns. Then, for each input vector $\mathbf{x}_i$, its input values $x_{i,1}, x_{i,2}, \dots, x_{i,n}$ are in the domain of corresponding features $f_1, \dots, f_n$, respectively. Throughout the work, we consider the following well-known classifiers: support vector machines and multilayer perceptrons with *rectified linear unit* (*ReLU*) (NAIR; HINTON, 2010) as activation functions. In what follows, we give some details on these classifiers.

2.2 Support Vector Machines

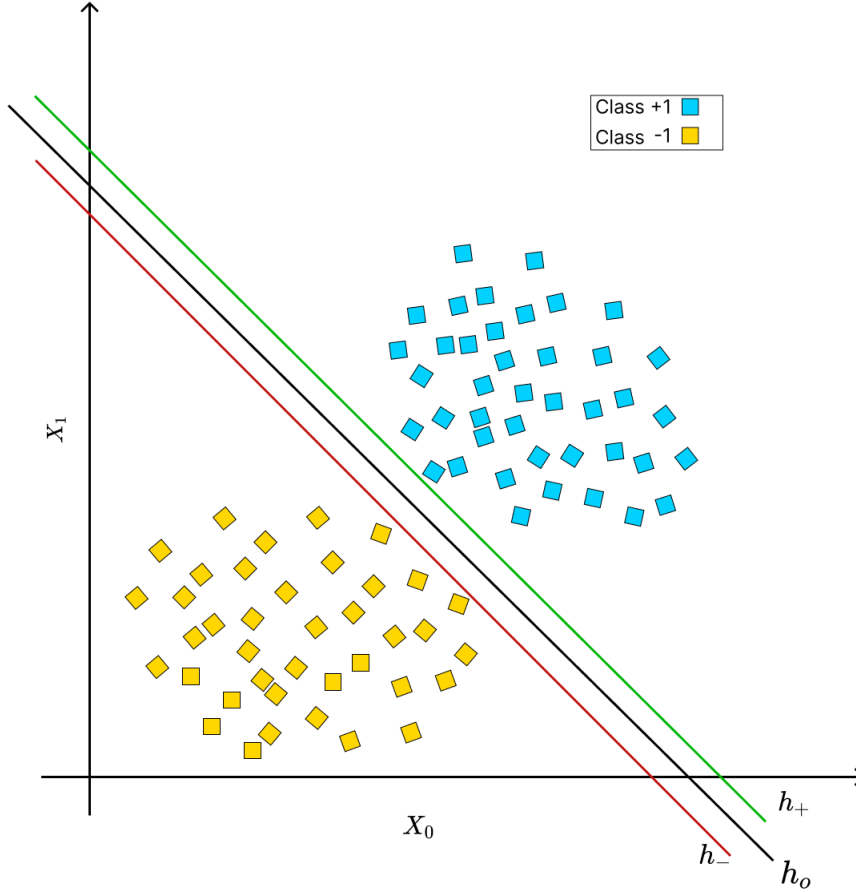
The support vector machine (SVM) (CORTES; VAPNIK, 1995) is a supervised machine learning model, often used for classification problems as support vector classifier (SVC). It separates the data by using the concept of an optimal separating hyperplane defined in Cortes and Vapnik (1995). On a $\mathbb{R}^n$ space, a hyperplane is defined by a set of points $\mathbf{x}$ that satisfies $\mathbf{w} \cdot \mathbf{x} + b = 0$, where $\mathbf{w} \in \mathbb{R}^n$ is the weight vector, $\mathbf{x} \in \mathbb{R}^n$ is a feature vector with n features and an intercept (bias) $b \in \mathbb{R}$.

A given training set $\{\mathbf{x}_i, y_i\}_{i=1}^{\mathcal{L}}$ is said to be linearly separable if there is $\mathbf{w} \in \mathbb{R}^n$ and $b \in \mathbb{R}$ that guarantees the separation between positive and negative class instances without error. In other words, the following inequalities

$$\text{for } i \in \{1, \dots, k\}, \begin{cases} \mathbf{w} \cdot \mathbf{x}_i + b \geq 0, & \text{if } y_i = +1, \\ \mathbf{w} \cdot \mathbf{x}_i + b \leq 0, & \text{if } y_i = -1, \end{cases} \quad (2.1)$$

must be satisfied to obtain a hyperplane. Among all possible hyperplanes, the optimal one, which has the maximum distance from the nearest patterns in the training set, is chosen. This optimal hyperplane h_o is defined as $\mathbf{w}_o \cdot \mathbf{x} + b_o = 0$, where w_o is the optimal weight vector and b_o the optimal bias. Thus, the decision function used for classifying input instances $\mathbf{x}$ is defined as $f(\mathbf{x}) = \mathbf{w}_o \cdot \mathbf{x} + b_o$.

Figure 1 – Support vector classifier hyperplanes.



Source: Author.

A Hard-Margin SVM (SVM-HM) (VAPNIK, 1998) can be used when the data is linearly separable and misclassifications are not allowed, maximizing the margin between two hyperplanes, h_+ and h_- , parallel to h_o , as shown in Figure 1. There can be no training instances between h_+ and h_- . Once maximizing the margin is similar to minimizing

$\frac{1}{2}||\mathbf{w}'||^2$, the optimization problem for obtaining the optimal parameters for the SVM can be described as follows:

$$\begin{aligned} \min_{\mathbf{w}, b} \quad & \frac{1}{2}||\mathbf{w}'||^2 \\ \text{s.t.} \quad & y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1, \quad \text{for } i \in \{1, \dots, k\}. \end{aligned} \quad (2.2)$$

Therefore, assuming the predicted label $\hat{y}$ and $\mathcal{K} = \{c_1, c_2\}$, such that $c_1 = +1$, $c_2 = -1$, and $\hat{y} \in \mathcal{K}$, the prediction result for an input instance $\mathbf{x}$ is computed as follows:

$$\hat{y} = \begin{cases} +1, & \text{if } f(\mathbf{x}) \geq 0, \\ -1, & \text{if } f(\mathbf{x}) < 0, \end{cases} \quad (2.3)$$

where

$$f(\mathbf{x}) = \text{sign}(\mathbf{w}_o \cdot \mathbf{x} + b_o). \quad (2.4)$$

However, in real-world problems, the training instances from the two classes can not be linearly separated by a hyperplane due to how the instances are distributed in the instance space. To overcome this situation, one must use Soft-Margin SVMs (SVM-SM) in which misclassifications are allowed to happen. To do so, slack variables can be used to relax the constraints of the SVM-HM (CORTES; VAPNIK, 1995; VAPNIK, 1998). Given this, the optimization problem for Soft-Margin SVMs is described as

$$\begin{aligned} \min_{\mathbf{w}, b, \xi} \quad & \frac{1}{2}||\mathbf{w}'||^2 - C \sum_{i=1}^k \xi_i \\ \text{s.t.} \quad & y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i, \quad \text{for } i \in \{1, \dots, k\} \\ & \xi_i \geq 0, \quad \text{for } i \in \{1, \dots, k\} \end{aligned} \quad (2.5)$$

where C is a trade-off between $\frac{1}{2}||\mathbf{w}'||^2$ and $\sum_i^n \xi_i$. Thus, for a high enough value of C , minimizing the sum of errors while maximizing the separation margin leads toward the optimal hyperplane. This can be the same as the one found through SVM-HM if the data is linearly separable. Moreover, the decision function and predicted label for SVM-HM and SVM-SM are the same.

2.3 Multilayer Perceptron

The multilayer perceptron (MLP) (BISHOP, 1995) is a feedforward neural network (NN) that can be used for dealing with classification problems. An NN is composed of $K + 1$ layers of neurons. Each layer $k \in \{0, 1, \dots, K\}$ is composed of n_k neurons, numbered from 1 to n_k . Let $\mathbf{x}^k \in \mathbb{R}^{n_k}$ be the output vector of the k -th layer and x_j^k be the output of the j -th neuron of that layer, with $j \in \{1, \dots, n_k\}$. The output values $\mathbf{x}^k = \{x_1^k, x_2^k, \dots, x_{n_k}^k\}$ of the neurons in a given layer k are computed through the output values $\mathbf{x}^{k-1}$ of the

previous layer. Assuming layer 0 as the input layer and $\mathbf{x}^0$ the input vector, such a process is repeated for each following layer, up to K layers.

Between each pair of consecutive layers, there is a weight matrix $W^{k-1} = \{\mathbf{w}_1^{k-1}, \mathbf{w}_2^{k-1}, \dots, \mathbf{w}_{n_k}^{k-1}\}$ that connects layers $k-1$ and k . Moreover, an intermediate value vector $\mathbf{y}^k = \{y_1^k, y_2^k, \dots, y_j^k\}$ is computed by the neurons in layer k through output values $\mathbf{x}^{k-1}$, the weights matrix W^{k-1} , and the respective bias vector $\mathbf{b}^{k-1}$. The computation of the output value of each neuron is computed by applying a non-linear activation function σ , such as the *ReLU* (NAIR; HINTON, 2010), on $\mathbf{y}^k$, pointwise. Moreover, the last layer K is composed of $\mathcal{N}$ neurons, one for each class. Therefore, assuming that σ is the *ReLU* activation function, the following is obtained:

$$\begin{aligned}\mathbf{y}^k &= W^{k-1} \mathbf{x}^{k-1} + \mathbf{b}^{k-1} \\ \mathbf{x}^k &= \text{ReLU}(\mathbf{y}^k).\end{aligned}\tag{2.6}$$

where $x_j^k = \max(0, y_j^k)$, for $j \in \{1, \dots, n_k\}$. As such, each scalar element x_j^k can be computed through

$$\begin{aligned}y_j^k &= \mathbf{w}_j^{k-1\top} \mathbf{x}^{k-1} + b_j^{k-1} \\ x_j^k &= \text{ReLU}(y_j^k),\end{aligned}\tag{2.7}$$

where $\top$ is the transpose.

The predicted output $\hat{y}$ is the *argmax* result over the neuron values in the last layer K , computed as follows:

$$\hat{y} = \text{argmax}(\sigma(W^{K-1} \mathbf{x}^{K-1} + \mathbf{b}^{K-1})),\tag{2.8}$$

where σ can also be other non-linear functions, such as the softmax function.

2.4 Reject Option Classification

Reject option for classification problems, as depicted in Chow (1970), is a set of techniques that aim to improve reliability in decision support systems. In the case of a binary problem, it consists of withholding and rejecting a classification result that is ambiguous enough, i.e. when the instance is too close to the decision boundary of the classifier. For the linear SVC, it is when the instance is too close to the separating hyperplane. Then, these rejected instances are analyzed through another classification method or even by a human specialist (MESQUITA et al., 2016).

In applications where a high degree of reliability is needed and misclassifications can be too costly, rejecting to classify an instance can be more beneficial than the risk of a higher error rate due to wrong classifications (OLIVEIRA et al., 2016). According to Chow (1970), the optimal classifiers that best handle such a relation can be achieved

by the minimization of $\hat{R}$ through

$$\hat{R} = Er + w_r R \quad (2.9)$$

where R is the ratio of the number of rejected training instances to the number of instances in the entire training dataset; Er is the ratio of the number of misclassified instances to the number of all the training instances without including those rejected; and w_r is a weight denoting the cost of rejection. A lower w_r gives room for a decreasing error rate at the cost of a higher quantity of rejected instances, with the opposite happening for a higher w_r .

A method is presented in Mesquita et al. (2016) for single, standard binary classifiers that do not provide probabilistic outputs. For SVCs, the proposed rejection techniques are based on the distance of instances to the optimal separating hyperplane. If the distance value is lower than a predefined threshold, then the instance is rejected. As such, a rejection region is determined after the training step of the classifier, with a threshold containing appropriate values being applied to the output of the classifier. Therefore, applying this strategy to a standard binary SVC leads to the following prediction cases:

$$\hat{y} = \begin{cases} +1, & \text{if } f(\mathbf{x}) > t_+, \\ -1, & \text{if } f(\mathbf{x}) < t_-, \\ 0, & \text{otherwise,} \end{cases} \quad (2.10)$$

where t_+ and t_- are the thresholds for the positive class and negative class, respectively, and 0 is the rejection class. Furthermore, these thresholds are chosen to generate the optimal reject region, which corresponds to the region that minimizes $\hat{R}$ by producing both the ratio of misclassified instances E and the ratio of rejected instances R .

2.5 XAI and Related Work

Some ML models can be interpreted by nature, such as decision trees (RIBEIRO; SINGH; GUESTIN, 2016). Others, such as neural networks and boosted trees, have harder-to-explain outputs, leading to the use of specific methods to get some degree of explanation (IGNATIEV; NARODYTSKA; MARQUES-SILVA, 2020). One of the most predominant ways to achieve this is through the use of heuristic methods for generating instance-dependent explanations, which can be defined as a local approach. These analyze and explore the sub-space close to the given instance (IGNATIEV, 2020; IGNATIEV; NARODYTSKA; MARQUES-SILVA, 2020).

Some of the most well-known heuristic methods are LIME, SHAP, and Anchors. These approaches are model-agnostic, generating local explanations while not taking into account the instance space as a whole (LUNDBERG; LEE, 2017; RIBEIRO; SINGH;

GUESTRIN, 2016; RIBEIRO; SINGH; GUESTRIN, 2018). This, in turn, allows the explanations to fail when applied, since they can be consistent with instances predicted in different classes. Moreover, they can include irrelevant elements which could be otherwise removed while still maintaining the correctness of the answer (IGNATIEV; NARODYTSKA; MARQUES-SILVA, 2020). Explanations with irrelevant elements may be harder to understand.

Anchors have been shown as a superior version to LIME, having a better accuracy with the resultant explanations (RIBEIRO; SINGH; GUESTRIN, 2018). The explanations obtained by Anchors are decision rules designed to highlight which parts (features) of a given instance are sufficient for a classifier to make a certain prediction while being intuitive and easy to understand. For example, consider an instance $\mathbf{x} = \{sepal_length = 5.0, sepal_width = 2.3, petal_length = 3.3, petal_width = 1.0\}$ of the well known Iris dataset predicted as class *versicolor* by a classifier $\mathcal{C}$. An example of an explanation obtained by Anchors is the decision rule:

IF *sepal_width* ≤ 2.8 **AND** *petal_length* > 1.6 **THEN** *versicolor*.

Such an explanation has some degree of generalization since it can be applied over a range of values instead of a specific point in the instance space. In this case, it is applied to all instances that follow the described rule. However, the lack of guarantees of correctness and minimality of such method reduces the extent of how such the generated explanations can be trusted.

Since heuristic-based explanations are unreliable, logic-based approaches have been proposed recently for a variety of ML models (AUDEMARD et al., 2022; IGNATIEV et al., 2022; IGNATIEV; NARODYTSKA; MARQUES-SILVA, 2019; IZZA; IGNATIEV; MARQUES-SILVA, 2020; MARQUES-SILVA et al., 2020). Logic-based approaches can guarantee the correctness of explanations, which makes them trustworthy. Although most of the related work on logic-based explanations focused on traditional ML models, there appears to be a gap in the literature regarding the provision of such explanations specifically in the presence of RO.

Then, in this work we are considering instance-based explanations for linear SVCs with RO, i.e. given a trained linear SVC with RO and an instance $\mathbf{x} = \{f_1 = x_1, f_2 = x_2, \dots, f_n = x_n\}$ classified as class $c \in \{-1, +1, 0\}$, an *explanation* E is a subset of the instance $E \subseteq \mathbf{x}$ sufficient for the prediction c . Therefore, by fixing the values of the features in E , the prediction is guaranteed to be c , regardless of the values of the other features. In other words, features in $\mathbf{x} \setminus E$ may take any value in their respective domains without changing the prediction. Moreover, since succinct explanations seem easier to understand, one would like explanations E to be *minimal*, that is, for all subset $E' \subset E$, it follows that E' does not guarantee the same prediction. Informally, a minimal explanation provides an irreducible explanation that is sufficient for guaranteeing the prediction.

2.6 First-Order Logic

To give explanations with guarantees of correctness, we use first-order logic (FOL) (KROENING; STRICHMAN, 2016). We use quantifier-free first-order formulas over the theory of linear real arithmetic. Then, first-order variables are allowed to take values from the real numbers. Therefore, we consider formulas as defined below:

$$\begin{aligned} \varphi, \psi &:= s \mid (\varphi \wedge \psi) \mid (\varphi \vee \psi) \mid (\neg \varphi) \mid (\varphi \rightarrow \psi), \\ s &:= \sum_{i=1}^n a_i z_i \leq e \mid \sum_{i=1}^n a_i z_i < e, \end{aligned} \quad (2.11)$$

such that φ and ψ are quantifier-free first-order formulas over the theory of linear real arithmetic. Moreover, s represents the atomic formulas such that $n \geq 1$, each a_i and e are concrete real numbers, and each z_i is a first-order variable. For example, $(2.5z_1 + 3.1z_2 \geq 6) \wedge (z_1 = 1 \vee z_1 = 2) \wedge (z_1 = 2 \rightarrow z_2 \leq 1.1)$ is a formula by this definition. Observe that we allow standard abbreviations as $\neg(2.5z_1 + 3.1z_2 < 6)$ for $2.5z_1 + 3.1z_2 \geq 6$.

Since we are assuming the semantics of formulas over the domain of real numbers, an *assignment* $\mathcal{A}$ for a formula φ is a mapping from the first-order variables of φ to elements in the domain of real numbers. For instance, $\{z_1 \mapsto 2.3, z_2 \mapsto 1\}$ is an assignment for $(2.5z_1 + 3.1z_2 \geq 6) \wedge (z_1 = 1 \vee z_1 = 2) \wedge (z_1 = 2 \rightarrow z_2 \leq 1.1)$. An assignment $\mathcal{A}$ *satisfies* a formula φ if φ is true under this assignment. For example, $\{z_1 \mapsto 2, z_2 \mapsto 1.05\}$ satisfies the formula in the above example, whereas $\{z_1 \mapsto 2.3, z_2 \mapsto 1\}$ does not satisfy it.

A formula φ is *satisfiable* if there exists a satisfying assignment for φ . Then, the formula in the above example is satisfiable since $\{z_1 \mapsto 2, z_2 \mapsto 1.05\}$ satisfies it. As another example, the formula $(z_1 \geq 2) \wedge (z_1 < 1)$ is unsatisfiable since no assignment satisfies it. The notion of satisfiability can be extended to sets of formulas Γ . A set of first-order formulas is satisfiable if there exists an assignment of values to the variables that makes all the formulas in Γ true simultaneously. For example, $\{(2.5z_1 + 3.1z_2 \geq 6), (z_1 = 1 \vee z_1 = 2), (z_1 = 2 \rightarrow z_2 \leq 1.1)\}$ is satisfiable given that $\{z_1 \mapsto 2, z_2 \mapsto 1.05\}$ jointly satisfies each one of the formulas in the set. It is well known that, for all sets of formulas Γ and all formulas φ and ψ ,

$$\begin{aligned} \Gamma \cup \{\varphi \vee \psi\} \text{ is unsatisfiable iff } \Gamma \cup \{\varphi\} \text{ is unsatisfiable and} \\ \Gamma \cup \{\psi\} \text{ is unsatisfiable.} \end{aligned} \quad (2.12)$$

Given a set Γ of formulas and a formula φ , the notation $\Gamma \models \varphi$ is used to denote *logical consequence*, i.e., each assignment that satisfies all formulas in Γ also satisfies φ . As an illustrative example, let Γ be $\{z_1 = 2, z_2 \geq 1\}$ and φ be $(2.5z_1 + z_2 \geq 5) \wedge (z_1 = 1 \vee z_1 = 2)$. Then, $\Gamma \models \varphi$ since each satisfying assignment for all formulas in Γ is also a satisfying assignment for φ . Moreover, it is widely known that, for all sets of formulas Γ and all formulas φ ,

$$\Gamma \models \varphi \text{ iff } \Gamma \cup \{\neg \varphi\} \text{ is unsatisfiable.} \quad (2.13)$$

For instance, $\{z_1 = 2, z_2 \geq 1, \neg((2.5z_1 + z_2 \geq 5) \wedge (z_1 = 1 \vee z_1 = 2))\}$ has no satisfying assignment since an assignment that satisfies $(z_1 = 2 \wedge z_2 \geq 1)$ also satisfies $(2.5z_1 + z_2 \geq 5) \wedge (z_1 = 1 \vee z_1 = 2)$ and, therefore, does not satisfy $\neg((2.5z_1 + z_2 \geq 5) \wedge (z_1 = 1 \vee z_1 = 2))$.

Finally, we say that two first-order formulas φ and ψ are equivalent if, for each assignment $\mathcal{A}$, both φ and ψ are true under $\mathcal{A}$ or both are false under $\mathcal{A}$. We use the notation $\varphi \equiv \psi$ to represent that φ and ψ are equivalent. For example, $\neg((z_1 + z_2 \leq 2) \wedge z_1 \geq 1)$ is equivalent to $(\neg(z_1 + z_2 \leq 2) \vee \neg(z_1 \geq 1))$. Besides, these formulas are equivalent to $((z_1 + z_2 > 2) \vee z_1 < 1)$.

2.7 Conclusions

This chapter presents an introduction to important concepts and definitions regarding classification problems, machine learning models, and eXplainable AI. The support vector machine and multilayer perceptron classifiers are presented, with explanations about how each model outputs a classification result, followed by the reject option strategy. Previous work in the field of XAI is addressed, covering the characteristics of heuristic-based explanations, decision rules, instance-based explanations, and logic-based approaches. Furthermore, it is shown important definitions and results on first-order logic regarding our work. These concepts will be used to generate explanations with guarantees of correctness.

Given the importance of logic-based explainability for machine learning model reliability and how there seems to be a lack of works tackling explanations in the presence of reject option, a method for computing logic-based explanations for linear support vector classifiers with reject option is presented in the next chapter

3 Logic-based Explanations for Linear Support Vector Classifiers with Reject Option

In this chapter, we will present one of the main contributions of this research: a method for computing logic-based explanations for linear support vector classifiers with reject option. Within this presentation, we address scenarios both with and without the presence of a rejection option. In the course of this chapter, compelling reasons for the development of this method will be offered. Additionally, the empirical results presented demonstrate the efficacy of the approach compared to competing methods.

In Section 3.1 we justify the need for a thorough interpretation method for linear SVCs, despite already existing other methods that can return interpretations of the model. Then, we define in Section 3.2 the encoding for linear SVCs when RO is applied over a binary classification context and the algorithm for computing minimal explanations. Experiments and results are depicted in Section 3.3 and Section 3.4, respectively.

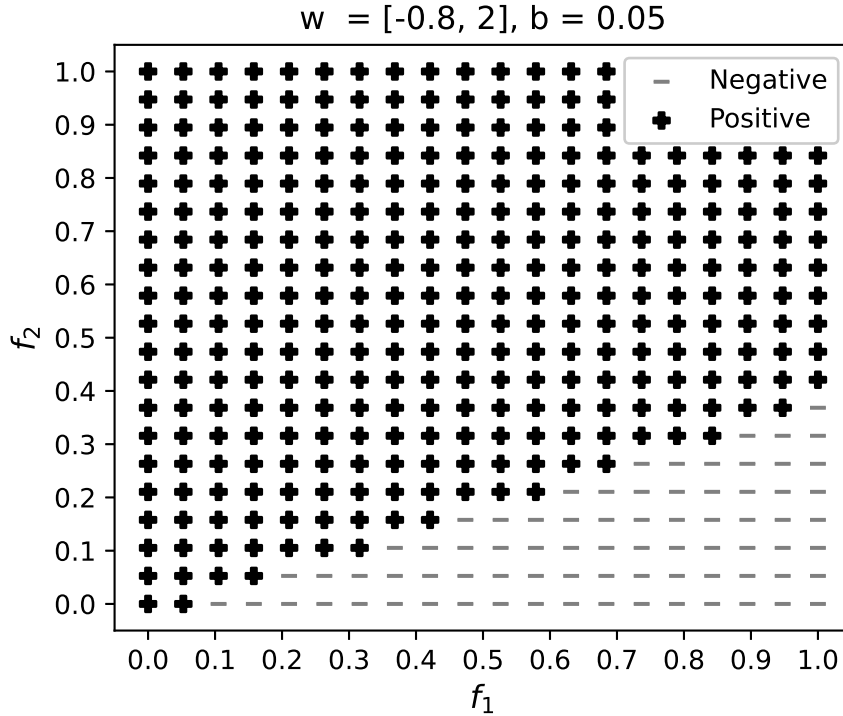
3.1 Linear Support Vector Classifiers may not be Instance-based Interpretable

Efforts have been made to bring explanations to linear SVCs prediction outputs. These have often been done through the analysis of the weights that compose the decision function, where the most important features are associated with the highest weights (GUYON et al., 2002). However, this may not be enough to enable correct interpretations. Although feature weights can give a rough indication of the overall relevance of features, they do not offer insight into the local decision-making process for a specific set of instances (KRAUSE et al., 2017).

Assume a binary classification problem where $\mathcal{F} = \{f_1, f_2\}$, and a linear SVC where $\mathbf{w} = \{w_1 = -0.8, w_2 = 2\}$ and $b = 0.05$. The features f_i can take values in range $[0, 1]$. A visual representation is depicted in Figure 2. Analyzing solely through the values of the weights, it could be assumed that the feature f_2 is determinant and f_1 is not, since $|w_2| > |w_1|$. This would mean that, for any instance, feature f_2 is more important than feature f_1 . However, for the instance $\{f_1 = 0.0526, f_2 = 0.3\}$ predicted as class +1, feature f_2 is not necessary for the prediction, since for any value of f_2 the class will not change. Therefore, feature f_1 is sufficient for the prediction of this instance. Moreover, feature f_2 is not sufficient for the prediction of this instance, since for $f_1 = 1.0$ and $f_2 = 0.3$ the prediction would change to -1 . Therefore, for instance $\{f_1 = 0.0526, f_2 = 0.3\}$, feature f_1

would be determinant and f_2 would not.

Figure 2 – Classification through features f_1 and f_2 .



Source: Author.

While it is trivial to observe if feature f_1 and f_2 can be determinant, due to the simplicity of the given decision function, it becomes harder to do so as the number of features increases. Furthermore, the presence of RO adds another layer of complexity to the problem, turning the interpretation task very non-trivial. Hence, explanations given by weights evaluation may give room for incorrect interpretations. This raises the need for an approach to computing explanations with guarantees of correctness for linear SVCs, even more so when RO is present.

3.2 Explanations For Linear SVMs with Reject Option

As depicted before, most methods that generate explanations for ML models are heuristic, which can bring issues about how correct and trustworthy the generated answers are. Added that they tend not to take into account whether RO is present, these problems are further aggravated. As such, an approach that guarantees the correctness of explanations for ML models with RO is due.

Our approach is achieved by encoding the linear SVC with RO as a first-order formula, following the ideas of Ignatiev, Narodytska and Marques-Silva (2019) for traditional neural networks, i.e. without RO. First, we will show how to define a first-order formula

for a trained linear SVC without RO. Then, we will show how to handle the reject option. Given a trained linear SVC in a binary classification context, defined by a weight vector $\mathbf{w}$, a bias b , and an instance $\mathbf{x}$ classified as class $c = +1$, we define a first-order formula P as:

$$P = (\sum_{i=1}^n w_i f_i + b \geq 0). \quad (3.1)$$

Therefore, this formula holds for all points $\mathbf{x}$ such that the linear SVC classifies as class $c = +1$. For an instance $\mathbf{x} = \{f_1 = x_1, f_2 = x_2, \dots, f_n = x_n\}$ where each of its features are defined by $D = \{l_1 \leq f_1 \leq u_1, l_2 \leq f_2 \leq u_2, \dots, l_n \leq f_n \leq u_n\}$, an explanation is a subset $E \subseteq \mathbf{x}$ such that $E \cup D \models P$. Every assignment that satisfies all the formulas in $E \cup D$ must also satisfy P . As a result, the prediction is assured to be in the positive class. Moreover, since we aim to obtain minimal explanations, E must be such that, for all subset $E' \subset E$, it follows that $E' \cup D \not\models P$. It is important to note that the values in E are the original values of instance $\mathbf{x}$, and those in $\mathbf{x} - E$ can take on any value according to D .

Instances predicted in other classes are treated similarly. For example, a first-order formula P for an instance $\mathbf{x}$ classified as class $c = -1$ is defined as

$$P = (\sum_{i=1}^n w_i f_i + b < 0). \quad (3.2)$$

Then, we need to take into account the reject option and its encoding as a first-order formula. For example, given a trained linear SVC with RO defined by w , b , t_+ and t_- as in Equation (2.10), and an instance $\mathbf{x}$ classified as class $c = 0$, i.e. in the rejection class, we define the first-order formula P :

$$P = (\sum_{i=1}^n w_i f_i + b \leq t_+) \wedge (\sum_{i=1}^n w_i f_i + b \geq t_-). \quad (3.3)$$

As such, if an instance $\mathbf{x}$ is classified as class $c = +1$, P is defined as $\sum_{i=1}^n w_i f_i + b > t_+$, while if it is classified as class $c = -1$ then P is defined as $\sum_{i=1}^n w_i f_i + b < t_-$. It is worth noting that, in these formulas, each f_i is a first-order variable, and b , t_+ , t_- and each w_i are concrete real numbers. Note that each first-order variable f_i is analogous to the z_i in Section 2.6. Moreover, each first-order variable f_i represents a feature.

Following the ideas outlined in Ignatiev, Narodytska and Marques-Silva (2019) for computing minimal explanations, we can employ Algorithm 1 to find a minimal explanation for a given $\mathbf{x}$, D , and P . For a predicted class c by the classifier, P is defined accordingly to reflect c . Therefore, the minimal explanation E for the classification of an instance $\mathbf{x}$ is computed in Algorithm 1 by setting E to $\mathbf{x}$ and removing feature by feature from E . As an example, for a feature $f_i = x_i \in E$ where $(E \setminus \{f_i = x_i\}) \cup D \models P$, then the value x_i of feature f_i is unnecessary for ensuring the prediction, therefore being removed from E . Otherwise, if $(E \setminus \{f_i = x_i\}) \cup D \not\models P$, then $f_i = x_i$ is necessary for the predictions, thus

being kept in E . This process is performed for all features. At the end of Algorithm 1, the prediction is secured and is not susceptible to variance regarding the values of the features removed from E .

By the result in (2.13), verifying entailments of the form $(E \setminus \{f_i = x_i\}) \cup D \models P$ in Algorithm 1 can be verified through the testing of whether the set of first-order formulas $(E \setminus \{f_i = x_i\}) \cup D \cup \{\neg P\}$ is satisfiable. If P is a conjunction $P_1 \wedge P_2$ as in (3.3), then $\neg P$ is equivalent to a disjunction $\neg P_1 \vee \neg P_2$. Therefore, we must check whether $(E \setminus \{f_i = x_i\}) \cup D \cup \{\neg P_1\}$ is unsatisfiable and $(E \setminus \{f_i = x_i\}) \cup D \cup \{\neg P_2\}$ is unsatisfiable, by the result in (2.12). Moreover, since $(E \setminus \{f_i = x_i\}) \cup D$ is a set of linear constraints, $\neg P_1$ is a linear constraint and $\neg P_2$ is a linear constraint, the unsatisfiability checking can be achieved by two queries answered by a linear programming (LP) solver. Therefore, if, for example, the set of linear constraints $(E \setminus \{f_i = x_i\}) \cup D \cup \{\neg P_1\}$ has a solution, then $(E \setminus \{f_i = x_i\}) \cup D \not\models P$. From a computational complexity viewpoint, the linear programming problem is solvable in polynomial time. Then, our approach for computing minimal explanations for linear SVCs, with and without RO, can also be solved in polynomial time. This is achieved by a linear number of calls to an LP solver in the number of features, which further contributes to the efficiency and feasibility of our approach.

It is important to highlight the difference in the explanations generated by our method and Anchors. Consider an instance $\mathbf{x} = \{sepal_length = 5.0, sepal_width = 2.3, petal_length = 3.3, petal_width = 1.0\}$ of the Iris dataset predicted as class *versicolor* by a classifier $\mathcal{C}$. While a possible Anchors explanation is the decision rule

IF *sepal_width* ≤ 2.8 **AND** *petal_length* > 1.6 **THEN** *versicolor*,

which contains value intervals, explanations generated by our method is a decision rule that has the original instance value for each feature in the explanation, without intervals:

IF *sepal_width* $= 2.3$ **AND** *petal_length* $= 3.3$ **THEN** *versicolor*.

For both of these explanations, features not explicitly in the explanation can assume any value within the respective feature domain.

Algorithm 1 Computing a minimal explanation

Input: instance $\mathbf{x}$ predicted as c , domain constraints D , formula P according to c

Output: minimal explanation E

$E \leftarrow \mathbf{x}$

for $f_i = x_i \in E$ **do**

if $(E \setminus \{f_i = x_i\}) \cup D \models P$ **then**

$E \leftarrow E \setminus \{f_i = x_i\}$

return E

3.3 Experiments

A total of 6 well-known datasets are used for conducting the experiments. The Vertebral Column and the Sonar datasets are available on the UCI machine learning repository¹. The Pima Indians Diabetes dataset is available on Kaggle². The Iris, the Breast Cancer Wisconsin, and the Wine datasets are available through the scikit-learn package³. As a standard procedure, all features were scaled to range $[0, 1]$.

To maintain the $\{-1, +1\}$ class domain in our context of binary classification, the Iris and the Wine dataset (each with three classes) were binarized, changed to *setosa-versus-all* and *class_0-versus-all*, respectively. The other datasets are already binary. The classes were changed to follow values in $\{-1, +1\}$. A summary of the datasets is presented in Table 1.

Table 1 – Datasets details.

Dataset	Acronym	$ \mathcal{F} $	Negative Instances	Positive Instances
Iris	IRIS	4	50	100
Vertebral Column	VRTC	6	100	210
Pima	PIMA	8	500	268
Wine	WINE	13	59	119
Breast Cancer Wisconsin	BRCW	30	212	357
Sonar	SONR	60	111	97

Source: Author.

The classifiers. For each dataset, a linear SVC was trained based on 70% of the original data, to simulate how a model would be trained for usage in real-world problems, with a regularization parameter $C = 1$ and stratified sampling. For finding the rejection region, a value of $w_r = 0.24$ was used together with the decision function outputs based on training data. Although the value itself is not too important for our purposes, it was chosen to ensure that rejections are neither penalized too softly nor too harshly, resulting in some rejected outputs. The selected rejection thresholds were obtained by minimizing $\hat{R}$, as described in Section 2.4.

For defining the values of t_+ and t_- , we determine a range of thresholds $T = \{(t_+^1, t_-^1), \dots, (t_+^p, t_-^p)\}$ containing the respective possible candidates. The maximum absolute value for t_+ and t_- in T , respectively, is the highest and lowest output value of the decision function, i.e. the *upper_limit* and the *lower_limit*, based on the training instances. We use decision function output values to achieve a reasonable reject region, that is able to properly reject ambiguous classification results for both known and unknown data. Thus, the attainable thresholds are achieved through $T = \{(i \cdot 0.01 \cdot \text{upper_limit}, i \cdot 0.01 \cdot \text{lower_limit}) \mid i \in \{1, \dots, 100\}\}$. Hence, the selected values of t_+ and t_- are the pair that

¹ <https://archive.ics.uci.edu/ml/datasets>

² <https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database>

³ <https://github.com/scikit-learn/scikit-learn/tree/main/sklearn/datasets/data>

minimizes $\hat{R}$. For each dataset, we obtained the best reject region defined by t_+ and t_- , the test accuracy of the classifier with RO, and the rejection ratio based on test data. The test accuracy of a classifier with RO is the standard test accuracy applied only to non-rejected instances. Afterward, we determine the number of instances per class from both training and test data, i.e. the entire datasets. Table 2 details these results and also the test accuracy of the classifier without reject option.

Table 2 – Reject region thresholds using training data. Classifier accuracy without reject option, accuracy with reject option, and rejection ratio for test data. Instances by class for each entire dataset.

Dataset	t_-	t_+	Accuracy w/o RO	Accuracy w/ RO	Rejection	Negative	Rejected	Positive
IRIS	-0.0157	0.0352	100.0%	100.00%	00.00%	50	0	100
VRTC	-0.3334	0.8396	76.34%	89.65%	47.92%	22	139	149
PIMA	-1.1585	0.8312	76.62%	92.20%	59.59%	232	474	62
WINE	-0.0259	0.0243	96.29%	96.29%	00.56%	56	1	121
BRCW	-0.4914	0.2370	97.66%	98.76%	04.02%	190	25	354
SONR	-0.3290	0.2039	74.60%	79.59%	20.00%	76	43	89

Source: Author.

Observe that the reject region for the IRIS dataset did not return any rejected instances in the dataset. This is likely due to the problem being linearly separable. Therefore, since our experiments rely on explaining the instances present in the dataset, this case of the IRIS dataset is treated as if reject option is not present.

The reject region obtained for the WINE dataset did not reject any of the test instances, therefore having no change in accuracy, likely due to the fact that the value chosen for w_r penalizes rejections too harshly. For the other datasets, the presence of reject option lead to a higher accuracy in all cases. This is expected since the rejected instances are not taken into account for evaluation. In addition, there was a substantial increase for both the VRTC and the PIMA dataset at the cost of rejecting roughly 48% and 60% of all the test instances, respectively. Different values for w_r can be used to achieve desirable results, depending on how much a rejection must be penalized in each dataset.

Anchors. We compared our approach against the heuristic method Anchors for computing explanations. Anchors was designed to work with traditional multiclass classification problems, i.e. classifiers without reject option. Since it was proposed as a model-agnostic method for explaining complex models, it should be more than enough for simpler models such as linear SVCs. Moreover, since we are explaining predictions of a linear SVC with RO, we used the Anchors explanation algorithm to treat the classifier with reject option as if it were a traditional classifier with classes in $\{-1, 0, +1\}$.

Our approach. The prototype implementation of our approach⁴ is written in Python and follows Algorithm 1. As an LP solver to check the unsatisfiability of sets of

⁴ Code available at: <https://github.com/franciscomateus0119/Logic-based-Explanations-for-Linear-Support-Vector-Classifiers-with-Reject-Option>

first-order sentences, we used Coin-or Branch-and-Cut (CBC)⁵. Moreover, the solver is accessed via the Python API PuLP⁶.

Evaluation. All instances were used for generating explanations, including both the ones used for training and testing. This was done to simulate scenarios closer to real-world problems, where explanations for both previously known (training) and unknown (test) data may prove valuable. Moreover, the main focus is on explaining classification results rather than model evaluation.

Since a per-instance explanation is done for each dataset, both mean elapsed time (in seconds) and the mean of the number of features in explanations are used as the basis for comparison. The results were separated by class to better evaluate and distinguish the rejected class from the others. Moreover, time to compute and size of explanations were chosen as metrics to better reflect the importance of fast and concise explanations when used to assist specialists in better analyzing rejected instances.

3.4 Results

The results based on the running time and the size of explanations are presented in Table 3 and Table 4, respectively. Following the IRIS dataset description in Table 2, there are no results for the rejected class.

Our approach has shown to be up to, surprisingly, roughly 286 times faster than Anchors. This happened for the SONR dataset, where the mean elapsed time of our method is 0.86 seconds against 246.14 seconds for Anchors. These results show how much harder it can be for Anchors to give an explanation as the number of features increases. Furthermore, it is worth noting that our approach needs much less time to explain positive, negative, and rejected classes overall.

Table 3 – Time comparison between our approach and Anchors (seconds). Less is better.

Dataset	Negative		Rejected		Positive	
	Anchors	Ours	Anchors	Ours	Anchors	Ours
IRIS	0.13 ± 0.045	0.04 ± 0.004	-	-	0.05 ± 0.024	0.04 ± 0.005
VRTC	0.27 ± 0.049	0.07 ± 0.004	0.33 ± 0.147	0.12 ± 0.018	0.28 ± 0.156	0.07 ± 0.002
PIMA	0.85 ± 0.162	0.10 ± 0.001	0.27 ± 0.302	0.16 ± 0.021	0.61 ± 0.108	0.10 ± 0.002
WINE	3.01 ± 0.335	0.17 ± 0.002	1.60 ± 0.000	0.21 ± 0.000	0.38 ± 0.532	0.17 ± 0.002
BRCW	26.58 ± 4.268	0.42 ± 0.015	20.66 ± 3.506	0.55 ± 0.072	5.84 ± 9.686	0.41 ± 0.013
SONR	233.26 ± 70.550	0.86 ± 0.011	257.51 ± 16.267	1.27 ± 0.258	246.14 ± 51.82	0.86 ± 0.019

Source: Author.

While our approach obtained more extensive explanations in certain cases, it also demonstrated the ability to be more succinct, generating explanations that were up to 43% smaller in other cases. In addition, our method is able to maintain the correctness of

⁵ <https://github.com/coin-or/Cbc>

⁶ <https://github.com/coin-or/pulp>

Table 4 – Explanation size comparison between our approach and Anchors. Less is better.

Dataset	Negative		Rejected		Positive	
	Anchors	Ours	Anchors	Ours	Anchors	Ours
IRIS	3.52 ± 0.854	3.00 ± 0.000	-	-	1.89 ± 0.733	2.2 ± 0.400
VRTC	4.95 ± 1.580	5.18 ± 0.574	4.56 ± 1.941	4.99 ± 0.515	4.24 ± 2.077	4.20 ± 1.118
PIMA	7.50 ± 1.534	5.13 ± 0.944	3.02 ± 2.249	4.71 ± 0.897	6.37 ± 2.541	4.85 ± 1.119
WINE	12.98 ± 0.132	7.44 ± 1.252	13.00 ± 0.000	12.00 ± 0.000	2.80 ± 2.803	7.04 ± 1.605
BRCW	29.17 ± 4.372	22.29 ± 3.739	29.64 ± 1.763	27.08 ± 1.016	7.70 ± 11.024	23.08 ± 2.208
SONR	51.10 ± 20.075	50.39 ± 4.283	57.95 ± 9.233	56.16 ± 1.413	54.86 ± 14.625	51.78 ± 1.413

Source: Author.

explanations. This highlights an expressive advantage of our approach over the heuristic nature of Anchors. While Anchors may provide explanations with fewer features in some cases, this is achieved due to the lack of formal guarantees on correctness.

It is important to note that cases similar to what we discussed in Section 3.1 occurred for the IRIS dataset. Through weights $\mathbf{w} = \{w_1 = 0.8664049, w_2 = -1.42027753, w_3 = 2.18870793, w_4 = 1.7984087\}$ and $b = -0.77064659$, obtained from the trained SVC, it could be assumed that feature f_1 might be not determinant for the classification. However, for the instance $\{f_1 = 0.05555556, f_2 = 0.05833333, f_3 = 0.05084746, f_4 = 0.08333333\}$ in class -1 for example, feature f_1 was in the explanation, while feature f_2 was not present. Furthermore, similar cases happened where f_2 was in the explanation while f_4 was not, even though $|w_4| > |w_2|$.

Similar occurrences are present in other datasets, reinforcing that the points discussed in Section 3.1 are not uncommon. As one more illustrative example, consider the linear SVC with reject option trained on the VRTC dataset such that $\mathbf{w} = \{w_1 = 0.72863148, w_2 = 1.97781269, w_3 = 0.85680605, w_4 = -0.32466632, w_5 = -3.42937211, w_6 = 2.43522629\}$, $b = 1.10008469$, $t_- = -0.3334$ and $t_+ = 0.8396$. For the instance $\{f_1 = 0.25125386, f_2 = 0.4244373, f_3 = 0.7214483, f_4 = 0.20007403, f_5 = 0.71932466, f_6 = 0.15363128\}$ in the reject class, feature f_3 was not in the explanation, while f_4 is present in the explanation, despite the fact that $|w_3| > |w_4|$. This is substantial for our approach since it demonstrates its capability of finding such cases.

Finally, we will present a brief case study based on the real application of the explanations generated by Anchor and our method. Let an instance from the IRIS dataset be $\mathbf{x} = \{sepal_length = 5.1, sepal_width = 3.5, petal_length = 1.59, petal_width = 0.292\}$ predicted as class *setosa*. The Anchors explanation for this is that

IF *sepal_length* ≤ 5.09 **AND** *sepal_width* > 3.30 **AND**
petal_length ≤ 1.59 **AND** *petal_width* ≤ 0.30 **THEN** *setosa*.

In comparison, the explanation given by our method is:

IF *sepal_width* $= 3.30$ **AND** *petal_length* $= 1.40$
petal_width $= 2.10$ **AND** **THEN** *setosa*.

Notice that the *sepal_length* feature was not explicitly included in this case, meaning that it can assume any value within the respective feature domain $4.30 \leq \text{sepal_length} \leq 7.90$.

Furthermore, let an instance from the VRTC dataset be $\mathbf{x} = \{\text{pelvic_incidence} = 34.76, \text{pelvic_tilt} = 2.63, \text{lumbar_lordosis_angle} = 29.50, \text{sacral_slope} = 32.12, \text{pelvic_radius} = 127.14, \text{degree_spondylolisthesis} = -0.46\}$ predicted as class *NO*. The Anchors explanation is that an instance $\mathbf{x}$ will be classified as *NO* whenever the following decision rule applies:

IF $\text{pelvic_incidence} \leq 46.88$ **AND** $\text{pelvic_tilt} \leq 16.40$ **AND**
 $\text{lumbar_lordosis_angle} \leq 37.46$ **AND** $\text{sacral_slope} \leq 32.82$ **AND**
 $\text{pelvic_radius} > 125.88$ **AND** $\text{degree_spondylolisthesis} \leq 10.42$ **THEN** *NO*.

In comparison, our method guarantees the predicted class with the explanation:

IF $\text{pelvic_incidence} = 34.76$ **AND** $\text{pelvic_tilt} = 2.63$ **AND**
 $\text{lumbar_lordosis_angle} = 29.50$ **AND**
 $\text{pelvic_radius} = 127.14$ **AND** $\text{degree_spondylolisthesis} = -0.46$ **THEN** *NO*.

The *sacral_slope* feature was not explicitly included in the explanation generated by our method, thus able to assume any value within the domain $13.37 \leq \text{sacral_slope} \leq 32.82$.

Such cases are substantial for our approach since they show it can give minimal explanations while guaranteeing the classification result. Comparatively, the correctness of explanations generated by Anchors is not guaranteed, even when considering more features, thus making them unreliable.

3.5 Conclusions

In this chapter, we proposed a logic-based approach to generate minimal explanations for a classifier with RO while guaranteeing correctness. A trained linear SVC with RO is used as a target model for explainability. Our approach is rooted in earlier work on computing minimal explanations for standard machine learning models without RO. Therefore, we encode the task of computing explanations for linear SVCs with RO as a logical entailment problem. Moreover, we use an LP solver for checking the entailment since all first-order sentences are linear constraints with real variables, and at most one disjunction occurs.

Our method is compared with Anchors, one of the most well-known heuristic methods, through six different datasets. Anchors was originally proposed as a model-agnostic method, and therefore expected to perform well for simpler models such as the linear SVC. Nonetheless, we found that not only does our approach take considerably less time than Anchors, but it also has reduced the size of explanations for many instances.

Our approach can be further improved. For example, it can be adapted to other classifiers with and without RO, such as neural networks and random forests. Moreover, some features in an explanation may change values and still lead to the same prediction. Then, a more general explanation may consider a range of values for each feature, such that the prediction does not change for any instances whose feature values fall within such ranges, which may enable a more comprehensive understanding of the model. This is further explored for standard classifiers without reject option in the next chapter.

4 Generalizing Logic-Based Explanations

In this chapter, we will present two approaches for expanding logic-based explanations and achieving higher generalization capability. Generated explanations are a set of value ranges that each feature can assume. The first method is called **Onestep**. It builds from previous work in the literature and is, in a certain way, equivalent to what is depicted in Izza et al. (2023). It has the core difference in the use of optimization to find the maximum and minimum values of the ranges. The second method is called **Twostep** and follows most of the steps of the Onestep approach. It differs in how the explanation ranges are expanded, focusing on having a higher quantity of feature ranges successfully expanded instead of few, but wide ranges.

Before explaining how the Onestep and the Twostep methods work in a detailed manner, we briefly address in Section 4.1 the concept of inflated explanations depicted in Izza et al. (2023), which we will refer to as expanded explanations and generalized explanations. In Section 4.2 we define the encoding of the MLP model. Then, we define the algorithms for generalizing explanations in Section 4.3. Experiments and results are depicted in Section 4.4 and Section 4.5, respectively.

4.1 Inflated explanations

The approach depicted in Izza et al. (2023) uses the concept of *abductive explanation* (SHIH; CHOI; DARWICHE, 2018; IGNATIEV; NARODYTSKA; MARQUES-SILVA, 2019) for generating explanations. The algorithm for computing *inflated* (generalized) explanations first picks some order for the features in a given explanation. Then, for every feature f in the chosen order, the algorithm sets the initial lower and upper bound values f that can be assumed to be equal to the original feature value. The algorithm seeks to inflate the bound by adding a value ϵ iteratively, checking in each iteration if the prediction holds with the updated value. If it does not, the last value is discarded, with the previous value being set as the upper limit of f . This process is done similarly for the lower bound, decrementing ϵ until the prediction does not hold.

This approach outputs explanations with guarantees of correctness, keeping only the feature value ranges that make the prediction hold. However, it has problems regarding the time cost of computing the explanations. Small increments can lead to a large quantity of iterations, therefore increasing the cost of computing the explanations. Furthermore, large increments may lead the algorithm to precociously find that the prediction may not hold. In other words, it could result in explanations with narrow feature ranges. Consequently, this may lead to a loss of coverage, where the instance space where the

explanation holds becomes too restricted.

In this work, we will present two methods to circumvent these issues. Just as in the work of Izza et al. (2023), we start from a minimal explanation and expand the range of values the features can assume. We will illustrate these two methods with the SVC and MLP models. Therefore, we will first show how to compute minimal explanations for MLPs in Section 4.2.

4.2 Multilayer Perceptron Encoding

As forementioned, most methods for explaining ML models are heuristic, resulting in explanations that can not be fully trusted due to a lack of guarantees of correctness. Our proposed approach is achieved by encoding the classifiers as first-order formulas, following what is depicted in Ignatiev, Narodytska and Marques-Silva (2019). For SVCs, this has been addressed in Section 3.2, which focuses on generating minimal explanations for linear SVCs, with and without RO. We now further detail the encoding of MLP in a classification context without RO.

We follow the concepts of representing a neural network model as a 0-1 Mixed Integer Linear Program (0-1 MILP) model depicted in Fischetti and Jo (2018). Given an MLP made up of $K + 1$ layers, numbered from 0 to K , with each layer $k \in \{0, 1, \dots, K\}$. For each layer $k \geq 1$, $\mathbf{x}^k$ is computed through the Equation (2.6), from which surges the scalar equation for each x_j^k in Equation (2.7).

The positive and negative part of a *ReLU* input are decoupled through the following linear conditions:

$$\mathbf{w}_j^{k-1\top} \mathbf{x}^{k-1} + b_j^{k-1} = x_j^k - s_j^k, \quad x_j^k \geq 0, \quad s_j^k \geq 0. \quad (4.1)$$

Since we must impose that at least one the two terms x_j^k and s_j^k must be zero, we introduce a binary activation variable $z_j^k \in \{0, 1\}$, imposing the logical implications

$$\begin{aligned} z_j^k = 1 &\rightarrow x_j^k \leq 0 \\ z_j^k = 0 &\rightarrow s_j^k \leq 0. \end{aligned}$$

The resultant 0-1 MILP formulation of the MLP is:

$$\left. \begin{aligned} \sum_{i=1}^{n_{k-1}} w_{ij}^{k-1} x_i^{k-1} + b_j^{k-1} &= x_j^k - s_j^k \\ x_j^k &\geq 0 \\ s_j^k &\geq 0 \\ z_j^k &\in \{0, 1\} \\ z_j^k = 1 &\rightarrow x_j^k \leq 0 \\ z_j^k = 0 &\rightarrow s_j^k \leq 0 \end{aligned} \right\} k = 1, \dots, K, \quad j = 1, \dots, n_k \quad (4.2)$$

$$l_j \leq x_j \leq u_j, \quad j = 1, \dots, n_0 \quad (4.3)$$

In the formulation, weights w_{ij}^{k-1} and biases b_j^k are given parameters. The *ReLU* output for each neuron is defined in (4.2).

We assume a 0-1 MILP formulation G as described above for a given MLP classifier, with $K + 1$ layers, $\mathbf{o}$ being the output layer, and an instance $\mathbf{x}$ classified as class $c_{j'}$ in a multiclass classification context. We define a first-order formula P as:

$$P = \bigwedge_{j \neq j'} o_{j'} > o_j. \quad (4.4)$$

This formula P represents that the value of the output neuron $o_{j'}$ associated with class $c_{j'}$ is higher than the values of the other output neurons. Similarly to what is proposed in Ignatiev, Narodytska and Marques-Silva (2019), and following our SVM strategy, the generated explanation for $\mathbf{x}$ is a subset $E \subseteq \mathbf{x}$ such that $E \cup D \cup G \models P$, with every assignment that satisfies all the formulas in $E \cup D \cup G$ also having to satisfy P . Again, E must be minimal, i.e., for a subset $E' \subset E$, it follows that $E' \cup D \cup G \not\models P$. Therefore, we must further elaborate P while taking into account these requirements.

Let the output layer $\mathbf{o}$ have $\mathcal{N}$ neurons, such that each o_j is an output neuron, for $j \in \{1, \dots, \mathcal{N}\}$. If instance $\mathbf{x}$ is classified as class $c_{j'}$, then $o_{j'} > o_j$ for all $j \in \{1, \dots, n_k\}$ such that $j \neq j'$, that is, the output of the neuron $o_{j'}$ is higher than the output of any other neuron. Through (2.13), $E \cup D \cup G \models P$ iff $E \cup D \cup G \cup \{\neg P\}$ is unsatisfiable. The constraints in $E \cup D \cup G$ are accepted by modern MILP solvers. We represent $\neg P$ as:

$$\begin{aligned} \neg(o_{j'} > o_1 \wedge \dots \wedge o_{j'} > o_{\mathcal{N}}), \quad \text{which is equivalent to} \\ (o_{j'} \leq o_1 \vee \dots \vee o_{j'} \leq o_{\mathcal{N}}), \end{aligned} \quad (4.5)$$

Therefore, in order to represent this disjunction as constraints accepted by modern MILP solvers, we add a binary activation variable r_j for each output neuron o_j such that $j \neq j'$, defined as follows:

$$\left. \begin{aligned} r_j = 1 &\rightarrow o_{j'} \leq o_j \\ r_j &\in \{0, 1\} \\ \sum_{j \neq j'} r_j &\geq 1 \end{aligned} \right\} j \in \{1, \dots, n_k\}, \quad j \neq j'. \quad (4.6)$$

Therefore, we utilize G for generating explanations. We also reuse Algorithm 1 for computing the minimal explanations for the MLP.

4.3 Generalizing Explanations

A minimal explanation $E \subseteq \mathbf{x}$, obtained as in Section 3.2 and Section 4.2, contains features and their corresponding values that will lead to the same class, independently of

the remaining features in $\mathbf{x} - E$. However, it is potentially the case that variations of values in some features still result in the same decision. Therefore, the features present in E have a meaningful impact on the classification result, thus being the targets for expansion. We define the explanation expansion task as an optimization problem. It involves fully expanding the feature ranges to the allowed limits, without changing the prediction, up to the lower and upper bounds of each feature $f_i \in E$, as in $D = \{l_1 \leq f_1 \leq u_1, l_2 \leq f_2 \leq u_2, \dots, l_n \leq f_n \leq u_n\}$. The following is the description of the optimization model for finding the lowest value t_i^* above the original value x_i that the feature f_i can assume changing the prediction.

$$\begin{aligned}
& \min && f_i \\
& \text{s.t.} && x_i \leq f_i \leq u_i \\
& && f_j = x_j, \quad \text{for all } f_j = x_j \in E \text{ with } j \neq i \\
& && l_g \leq f_g \leq u_g, \quad \text{for all } f_g \in \mathbf{x} - E \\
& && \mathcal{R}
\end{aligned} \tag{4.7}$$

whereas x_i and x_j are the values of features f_i, f_j , respectively, f_j such that $f_j = x_j \in E$ are the features in the minimal explanation whose ranges have not been found yet, such that $f_g = x_g \notin E$ in the optimization model (4.7) are the features outside of the minimal explanation, and $\mathcal{R}$ is the restriction for the prediction not holding, modified according to the classifier. For the SVC, $\mathcal{R}$ is equivalent to equation (2.3) for the non-predicted class, i.e., if the prediction result is $\hat{y} = +1$, then $\mathcal{R}$ becomes $f(\mathbf{x}) < 0$. Otherwise, if $\hat{y} = -1$ then $\mathcal{R}$ becomes $f(\mathbf{x}) \geq 0$. A similar process is done for the MLP, in which case $\mathcal{R}$ is equivalent to equation (4.6).

If the problem is satisfiable, the optimal solution value t_i^* for the feature f_i is found. Thus, it is the lowest value above the original value, i.e., $x_i < t_i^* \leq u_i$ that changes the prediction result. Therefore, we set the value of the new upper bound $u_i' = t_i^* - \varepsilon$, with ε assuming a small positive real value. Moreover, if $u_i' < x_i$ then we set $u_i' = x_i$, guaranteeing that $x_i \leq u_i'$. The value obtained from $t_i^* - \varepsilon$ is the highest value (up to an imprecision of ε) that f_i can assume that makes the prediction hold. Otherwise, if the problem is unsatisfiable, then there is no value for $x_i < f_i \leq u_i$ that changes the prediction. Therefore, we set $u_i' = u_i$.

A similar process is done for the lower bound l_i' :

$$\begin{aligned}
& \max && f_i \\
& \text{s.t.} && l_i \leq f_i \leq x_i \\
& && f_j = x_j, \quad \text{for all } f_j = x_j \in E \text{ with } j \neq i \\
& && l_g \leq f_g \leq u_g, \quad \text{for all } f_g \in \mathbf{x} - E \\
& && \mathcal{R}
\end{aligned} \tag{4.8}$$

If the problem is satisfiable, the optimal value t_i^* for f_i is found. Therefore, t_i^* , which is the highest value below the original value, i.e., $l_i \leq t_i^* < x_i$ that makes the prediction change. In this case, we set $l'_i = t_i^* + \varepsilon$. Then, l'_i is the lowest value that makes the prediction hold. Moreover, if $l'_i > x_i$ then we set $l'_i = x_i$, guaranteeing that $l'_i \leq x_i$. Otherwise, if the problem is unsatisfiable, we set $l'_i = l_i$. Therefore, the prediction holds if the feature assumes any value within the range $l'_i \leq f_i \leq u'_i$, with every other feature f_j such that $f_j = x_j \in E$ with their values still set to their minimal explanation value $f_j = x_j$.

Then, for every other feature, we find the new maximum and lowest values of the range through

$$\begin{aligned}
& \min \quad f_i \\
& \text{s.t.} \quad l'_h \leq f_h \leq u'_h, \quad \text{for all } f_h \text{ such that } f_h = x_j \in E \text{ and } h < i \\
& \quad \quad x_i \leq f_i \leq u_i, \\
& \quad \quad f_j = x_j, \quad \text{for all } f_j \text{ such that } f_j = x_j \in E \text{ and } i < j \leq n \\
& \quad \quad l_g \leq f_g \leq u_g, \quad \text{for all } f_g \text{ such that } f_g = x_g \in \mathbf{x} - E \\
& \quad \quad \mathcal{R}
\end{aligned} \tag{4.9}$$

and

$$\begin{aligned}
& \max \quad f_i \\
& \text{s.t.} \quad l'_h \leq f_h \leq u'_h, \quad \text{for all } f_h \text{ such that } f_h = x_j \in E \text{ and } h < i \\
& \quad \quad l_i \leq f_i \leq x_i, \\
& \quad \quad f_j = x_j, \quad \text{for all } f_j \text{ such that } f_j = x_j \in E \text{ and } i < j \leq n \\
& \quad \quad l_g \leq f_g \leq u_g, \quad \text{for all } f_g \text{ such that } f_g = x_g \in \mathbf{x} - E \\
& \quad \quad \mathcal{R}
\end{aligned} \tag{4.10}$$

respectively. Note that l'_h and u'_h , for all $f_h \in E$ such that $h < i$, are the previously found ranges that make the prediction hold.

4.3.1 Onestep

Our first proposal, based on the previous optimization problems, is, in a certain way, equivalent to the approach detailed in Izza et al. (2023), for ordinal features. Our approach, named **Onestep**, has the core difference in the use of optimization to find the maximum and minimum values of the ranges *in one step* instead of an incremental approach, fully expanding the feature ranges to the allowed limits, up to the lower and upper bounds.

Our tactic enables minimizing two important issues derived from the size of the increments used in Izza et al. (2023). Increments that are too small can cause a significant increase in computation costs since it can lead to computing considerably more iterations before finding the explanation ranges, which can make the approach impractical to use.

On the other hand, increments that are too big can also be detrimental. These can lead to the occurrence of range overshoot, i.e., the increments bypass the lower and upper limits of the instance features, returning very suboptimal explanation ranges. The discovered range bounds do not correspond to the minimum and maximum value the feature can assume, leading to a loss of instance coverage due to the smaller generated instance space of the explanation.

The Onestep algorithm for computing expanded explanations is depicted in Algorithm 2. We employ (4.7-4.8) to obtain u'_i and l'_i for the first feature. Then, we utilize (4.9-4.10) to find u'_i and l'_i for every subsequent feature. The algorithm outputs a generalized explanation E_{xp} , containing the updated range of values each feature can assume without changing the prediction.

Algorithm 2 Computing an Onestep expanded explanation

Input: instance $\mathbf{x}$ predicted as c , domain constraints D , restriction $\mathcal{R}$ according to c , formulation G , minimal explanation E

Output: expanded explanation E_{xp}

$h \leftarrow \{\}$

$E_{xp} \leftarrow \{\}$

let $f_i = x_i \in E$

$\text{min_opt_model} \leftarrow \text{create_first_min_model}(\mathbf{x}, D, \mathcal{R}, G, E, f_i = x_i)$ ▷ 4.7

$\text{max_opt_model} \leftarrow \text{create_first_max_model}(\mathbf{x}, D, \mathcal{R}, G, E, f_i = x_i)$ ▷ 4.8

$u'_i \leftarrow \text{solve}(\text{min_opt_model})$

$l'_i \leftarrow \text{solve}(\text{max_opt_model})$

$h \leftarrow h \cup \{f_i = x_i\}$

$E_{xp} \leftarrow E_{xp} \cup \{l'_i \leq f_i \leq u'_i\}$

for $f_i = x_i \in E$ such that $f_i = x_i \notin h$ **do**

$\text{min_opt_model} \leftarrow \text{create_min_model}(\mathbf{x}, D, \mathcal{R}, G, E_{xp}, f_i = x_i, E)$ ▷ 4.9

$\text{max_opt_model} \leftarrow \text{create_max_model}(\mathbf{x}, D, \mathcal{R}, G, E_{xp}, f_i = x_i, E)$ ▷ 4.10

$u'_i \leftarrow \text{solve}(\text{min_opt_model})$

$l'_i \leftarrow \text{solve}(\text{max_opt_model})$

$h \leftarrow h \cup \{f_i = x_i\}$

$E_{xp} \leftarrow E_{xp} \cup \{l'_i \leq f_i \leq u'_i\}$

return E_{xp}

4.3.2 Twostep

Our second method, named **Twostep**, follows almost the same steps as the Onestep method, the main change being to how the range of values are set for each feature f_i . While Onestep can find ranges that guarantee the prediction, it could be that some of the found feature ranges are too narrow, where the features can assume values within a tiny value space. Moreover, it is possible that $l'_i = u'_i = x_i$, that is, the range could not be expanded, due to any other value making the prediction change since other features have already been expanded too much.

Our Twostep method is a strategy for reducing the rate of such occurrences. This involves reducing the range of some features to allow the expansion of other ranges. We define l_i'' and u_i'' as the lower and upper limits of a subrange within $[l_i', u_i']$, i.e., $l_i' \leq l_i''$, $u_i'' \leq u_i'$, and $l_i'' \leq u_i''$. Moreover, we set $l_i'' \leq f_i \leq u_i''$, referring to all values the feature f_i can assume within the subrange.

We calculate l_i' and u_i' as depicted for the Onestep approach. Then, we set $l_i'' = x_i - (x_i - l_i') \cdot p$ and $u_i'' = x_i + (u_i' - x_i) \cdot p$, where p is a given parameter with value $0 < p \leq 1$ that controls the size of the subrange. Note that $p = 1$ is equivalent to Onestep. For example, if $p = 0.5$, then $l_i'' = x_i - (x_i - l_i') \cdot 0.5$ and $u_i'' = x_i + (u_i' - x_i) \cdot 0.5$, that is, the lower and upper limits of the subrange are equivalent to 50% of the respective original range limits.

Therefore, we can employ the following optimization problems for expanding the ranges while considering found u_i'' and l_i'' :

$$\begin{aligned}
\min \quad & f_i \\
\text{s.t.} \quad & l_h' \leq f_h \leq u_h', \quad \text{for all } f_h \text{ such that } f_h = x_j \in E \text{ and } h < i \\
& u_i'' \leq f_i \leq u_i, \\
& l_j'' \leq f_j \leq u_j'', \quad \text{for all } f_j \text{ such that } f_j = x_j \in E \text{ and } i < j \leq n \\
& l_g \leq f_g \leq u_g, \quad \text{for all } f_g \text{ such that } f_g = x_g \in \mathbf{x} - E \\
& \mathcal{R}
\end{aligned} \tag{4.11}$$

$$\begin{aligned}
\max \quad & f_i \\
\text{s.t.} \quad & l_h' \leq f_h \leq u_h', \quad \text{for all } f_h \text{ such that } f_h = x_j \in E \text{ and } h < i \\
& l_i \leq f_i \leq l_i'', \\
& l_j'' \leq f_j \leq u_j'', \quad \text{for all } f_j \text{ such that } f_j = x_j \in E \text{ and } i < j \leq n \\
& l_g \leq f_g \leq u_g, \quad \text{for all } f_g \text{ such that } f_g = x_g \in \mathbf{x} - E \\
& \mathcal{R}.
\end{aligned} \tag{4.12}$$

The Twostep algorithm for computing expanded explanations is depicted in Algorithm 3. The first step of the algorithm involves finding u_i' and l_i' of the first feature through (4.7) and (4.8). Then, we obtain the intermediate bounds u_i'' and l_i'' by reducing u_i' and l_i' according to p . We employ (4.9) and (4.10) to find u_i' and l_i' , and therefore u_i'' and l_i'' , for each subsequent features. The second step of the algorithm comprises expanding found ranges up to their limits through (4.11-4.12) for each feature, without reducing. The algorithm outputs a generalized explanation E_{xp} , containing the updated range of values each feature can assume without changing the prediction.

Algorithm 3 Computing a Twostep expanded explanation

Input: instance $\mathbf{x}$ predicted as c , domain constraints D , restriction $\mathcal{R}$ according to c , formulation G , minimal explanation E , parameter p

Output: expanded explanation E_{xp}

```

 $h \leftarrow \{\}$  ▷ Step 1
 $E_{in} \leftarrow \{\}$ 
let  $f_i = x_i \in E$ 
min_opt_model  $\leftarrow$  create_first_min_model( $\mathbf{x}, D, \mathcal{R}, G, E, f_i = x_i$ ) ▷ 4.7
max_opt_model  $\leftarrow$  create_first_max_model( $\mathbf{x}, D, \mathcal{R}, G, E, f_i = x_i$ ) ▷ 4.8
 $u'_i, l'_i \leftarrow$  solve(min_opt_model), solve(max_opt_model)
 $u''_i, l''_i \leftarrow x_i + (u'_i - x_i) \cdot p, \quad x_i - (x_i - l'_i) \cdot p$ 
 $h \leftarrow h \cup \{f_i = x_i\}$ 
 $E_{in} \leftarrow E_{in} \cup \{l''_i \leq f_i \leq u''_i\}$ 
for  $f_i = x_i \in E$  such that  $f_i = x_i \notin h$  do
    min_opt_model  $\leftarrow$  create_min_model( $\mathbf{x}, D, \mathcal{R}, G, E_{in}, f_i = x_i$ ) ▷ 4.9
    max_opt_model  $\leftarrow$  create_max_model( $\mathbf{x}, D, \mathcal{R}, G, E_{in}, f_i = x_i$ ) ▷ 4.10
     $u'_i, l'_i \leftarrow$  solve(min_opt_model), solve(max_opt_model)
     $u''_i, l''_i \leftarrow x_i + (u'_i - x_i) \cdot p, \quad x_i - (x_i - l'_i) \cdot p$ 
     $h \leftarrow h \cup \{f_i = x_i\}$ 
     $E_{in} \leftarrow E_{in} \cup \{l''_i \leq f_i \leq u''_i\}$ 
 $h \leftarrow \{\}$  ▷ Step 2
 $E_{xp} \leftarrow \{\}$ 
for  $l''_i \leq f_i \leq u''_i \in E_{in}$  such that  $l''_i \leq f_i \leq u''_i \notin h$  do
    min_opt_model  $\leftarrow$  create_min_model( $\mathbf{x}, D, \mathcal{R}, G, E_{xp}, l''_i \leq f_i \leq u''_i, E_{in}$ ) ▷ 4.11
    max_opt_model  $\leftarrow$  create_max_model( $\mathbf{x}, D, \mathcal{R}, G, E_{xp}, l''_i \leq f_i \leq u''_i, E_{in}$ ) ▷ 4.12
     $u'_i, l'_i \leftarrow$  solve(min_opt_model), solve(max_opt_model)
     $h \leftarrow h \cup \{l''_i \leq f_i \leq u''_i\}$ 
     $E_{xp} \leftarrow E_{xp} \cup \{l'_i \leq f_i \leq u'_i\}$ 
return  $E_{xp}$ 

```

4.4 Experiments

In this work, we used a total of 8 well-known datasets in our experiments¹. The Blood Transfusion, the Vertebral Column, the Parkinsons, and the Ionosphere datasets are available at the UCI machine learning repository². The Pima Indians Diabetes dataset is available on Kaggle³. The Iris, the Breast Cancer Wisconsin, and the Wine datasets are available through the scikit-learn package⁴. All features were scaled to range $[0, 1]$, as a standard procedure.

To maintain the class domain in our context of binary classification for the linear SVC, the Iris and the Wine dataset (each with three classes) were binarized, changed to *setosa-versus-all* and *class_0-versus-all*, respectively. For the MLP, the original version of

¹ Code available at <https://github.com/franciscomateus0119/GExp>

² <https://archive.ics.uci.edu/ml/datasets>

³ <https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database>

⁴ <https://github.com/scikit-learn/scikit-learn/tree/main/sklearn/datasets/data>

these datasets was used, i.e., with three classes. The other datasets are already binary. Any categorical data has been removed since we are assuming features have continuous values. A summary of the datasets, with the said modifications, is presented in Table 5.

Table 5 – Datasets details.

Dataset	Acronym	Quantity of Features	Instances
Iris	IRIS	4	150
Blood Transfusion	BLDT	4	748
Vertebral Column	VRTC	6	310
Pima	PIMA	8	768
Wine	WINE	13	178
Parkinsons	PARK	22	195
Breast Cancer Wisconsin	BRCW	30	569
Ionosphere	IONS	34	351

Source: Author.

The SVM classifier. For each dataset, a linear SVC was trained based on 70% of the original data. A regularization parameter $C = 1$ was used, along with stratified sampling. Other parameters were set to their default value, as stated within the scikit-learn package.

The MLP classifier. For each dataset, an MLP was trained based on 70% of the original data. A single hidden layer with the number of neurons equal to the input layer size was used, along with ReLU as activation function. On the output layer, a softmax activation was used. Models were compiled with Adam Optimizer. The training was done with stratified sampling, with class training and test data transformed into their categorical equivalent. Each model was trained with 400 epochs, a batch size of 32, and an early stopping patience of 40, saving only the best model.

Our approaches. Both Onestep and Twostep prototype implementations were written in Python, following Algorithms 2 and 3. Comparing Twostep with Onestep is equivalent to comparing Twostep with the method shown in Izza et al. (2023). The CPLEX MILP solver was chosen to check the unsatisfiability of sets of first-order sentences and to solve the optimization problems in (4.7)-(4.10). The modeling of the optimization problems in (4.7)-(4.10) was done with the DOcplex Python Modeling API.

Evaluation. With 70% of each dataset being used for training, the remaining 30% was used for validating the models and as a target for generating explanations. For the basis of comparison between each approach, we used: the mean of elapsed time for computing explanations; the mean of the explanation ranges, where the explanation ranges refer to the set of value ranges each feature $l'_i \leq f_i \leq u'_i \in E_{xp}$ can assume, which can be

computed through

$$\frac{\sum_{i=1} u'_i - l'_i}{|E_{xp}|}, \quad \text{for all } l'_i \leq f_i \leq u'_i \in E_{xp}; \quad (4.13)$$

and the coverage of explanations based on the dataset instances, which entails verifying how many instances are covered by the explanation, that is, whose original values are inside the explanation ranges. We used a parameter value of $p \in \{0.25, 0.50, 0.75\}$ for computing the Twostep approach results.

4.5 Results

The results based on running time are present in Table 6 and Table 7, for the SVM and MLP, respectively. Coverage results are depicted in Table 8. Results regarding the explanation ranges can be found in Table 9.

The Twostep tends to be slower than the Onestep, up to roughly 193% more (2.14 seconds vs 6.28 seconds) as depicted in the results of the IONS dataset with the MLP as a classifier in Table 7. This is expected since Twostep needs to compute more steps to generate the explanations. For the SVM results in Table 6, it is noticeable that the times for both methods are close for every dataset. This is probably due to being a simpler linear programming problem to solve, thus allowing for faster computations of the explanations.

Table 6 – Time results for SVM.

Dataset	Onestep	Twostep		
		T-Step 0.25	T-Step 0.50	T-Step 0.75
IRIS	0.02 ± 0.007	0.02 ± 0.005	0.02 ± 0.004	0.03 ± 0.009
BLDT	0.02 ± 0.005	0.03 ± 0.005	0.03 ± 0.007	0.03 ± 0.005
VRTC	0.02 ± 0.004	0.04 ± 0.011	0.03 ± 0.007	0.04 ± 0.010
PIMA	0.04 ± 0.007	0.06 ± 0.010	0.06 ± 0.008	0.06 ± 0.012
WINE	0.05 ± 0.010	0.08 ± 0.014	0.08 ± 0.017	0.08 ± 0.016
PARK	0.07 ± 0.015	0.10 ± 0.030	0.10 ± 0.031	0.10 ± 0.030
BRCW	0.12 ± 0.013	0.19 ± 0.022	0.19 ± 0.023	0.19 ± 0.021
IONS	0.13 ± 0.013	0.21 ± 0.021	0.21 ± 0.023	0.21 ± 0.023

Source: Author.

Regarding the coverage results in Table 8, the Twostep approach has demonstrated that for every dataset and both SVM and MLP models, it can achieve equal or higher coverage of instances than the Onestep method. The results of the PIMA dataset with the SVM classifier show that the Twostep can cover up to 26% more instances than the Onestep. Moreover, the results of the IRIS dataset with the MLP classifier show that the Twostep can cover up to 29.5% more instances than the Onestep. This could mean

Table 7 – Time results for MLP.

Dataset	Onestep	Twostep		
		T-Step 0.25	T-Step 0.50	T-Step 0.75
IRIS	0.18 ± 0.044	0.22 ± 0.052	0.26 ± 0.115	0.20 ± 0.024
BLDT	0.14 ± 0.021	0.18 ± 0.027	0.19 ± 0.031	0.18 ± 0.029
VRTC	0.20 ± 0.025	0.33 ± 0.122	0.30 ± 0.052	0.30 ± 0.050
PIMA	0.27 ± 0.049	0.41 ± 0.073	0.41 ± 0.103	0.42 ± 0.084
WINE	0.46 ± 0.045	0.77 ± 0.086	0.78 ± 0.097	0.80 ± 0.104
PARK	0.83 ± 0.099	1.43 ± 0.238	1.52 ± 0.328	1.49 ± 0.268
BRCW	1.67 ± 0.395	2.82 ± 0.656	3.00 ± 0.883	3.08 ± 1.107
IONS	2.14 ± 0.754	4.73 ± 1.755	6.01 ± 2.521	6.28 ± 2.730

Source: Author.

having a more conservative expansion of the feature ranges instead of expanding only a few features leads to a better generalization of the explanations, and therefore they can be applied to a higher space of instances.

Table 8 – Coverage results between Onestep and Twostep.

Dataset	SVM				MLP			
	O-Step	T-Step 0.25	T-Step 0.50	T-Step 0.75	O-Step	T-Step 0.25	T-Step 0.50	T-Step 0.75
IRIS	11.77 ± 7.996	13.77 ± 6.511	13.97 ± 5.968	13.15 ± 6.902	4.51 ± 3.493	5.84 ± 4.47	5.82 ± 4.202	5.48 ± 4.102
BLDT	225.00 ± 0.000	225.00 ± 0.000	225.00 ± 0.000	225.00 ± 0.000	62.65 ± 40.271	71.29 ± 38.301	76.40 ± 40.404	70.54 ± 38.873
VRTC	3.66 ± 3.260	4.21 ± 3.684	4.31 ± 3.815	4.10 ± 3.626	1.81 ± 1.367	2.24 ± 1.708	2.17 ± 1.663	2.00 ± 1.586
PIMA	6.57 ± 7.317	8.28 ± 8.131	8.06 ± 7.968	7.14 ± 7.557	3.33 ± 4.119	4.23 ± 5.361	3.97 ± 4.948	3.58 ± 4.431
WINE	1.11 ± 0.566	1.12 ± 0.578	1.12 ± 0.578	1.11 ± 0.566	1.00 ± 0.000	1.00 ± 0.000	1.00 ± 0.000	1.00 ± 0.000
PARK	1.11 ± 0.372	1.16 ± 0.492	1.16 ± 0.492	1.15 ± 0.480	1.00 ± 0.000	1.00 ± 0.000	1.00 ± 0.000	1.00 ± 0.000
BRCW	1.00 ± 0.000	1.00 ± 0.000	1.00 ± 0.000	1.00 ± 0.000	1.00 ± 0.000	1.00 ± 0.000	1.00 ± 0.000	1.00 ± 0.000
IONS	1.00 ± 0.000	1.00 ± 0.000	1.00 ± 0.000	1.00 ± 0.000	1.00 ± 0.000	1.00 ± 0.000	1.00 ± 0.000	1.00 ± 0.000

Source: Author.

Notably, the BLDT datasets results for the SVM were equal for both Onestep and Twostep approaches in Table 8. This was likely due to the linear SVM model not being suitable for this dataset. The trained model was biased towards one class, leading to every prediction returning the same result. This resulted in every explanation having full coverage of the feature space. Such a scenario did not repeat itself for the MLP classifier, which reached a higher generalization and, thus, has a better distinction between the classes.

It is important to point out that as the number of features increased for each dataset, the coverage capability of the explanations on the test data was reduced. This is probably due to the higher instance space complexity, where the task of outputting explanations with some degree of coverage on the test data becomes very difficult. It is also noteworthy that since coverage results are based on dataset instances, while some explanations cover only one instance they could cover other instance sets that are not present in the datasets.

The results related to the explanation ranges depicted in Table 9 show that the

Twostep approach was able to increase the overall feature ranges, achieving results similar to or better than the Onestep for various datasets, including those with many features, albeit not by much.

Table 9 – Explanation ranges results between Onestep and Twostep.

Dataset	SVM				MLP			
	O-Step	T-Step 0.25	T-Step 0.50	T-Step 0.75	O-Step	T-Step 0.25	T-Step 0.50	T-Step 0.75
IRIS	2.57 ± 0.449	2.55 ± 0.469	2.55 ± 0.472	2.56 ± 0.464	2.00 ± 0.149	1.95 ± 0.166	1.94 ± 0.178	1.96 ± 0.169
BLDT	4.00 ± 0.000	4.00 ± 0.000	4.00 ± 0.000	4.00 ± 0.000	2.06 ± 0.131	2.07 ± 0.124	2.07 ± 0.125	2.07 ± 0.129
VRTC	4.07 ± 0.945	4.06 ± 0.966	4.07 ± 0.963	4.08 ± 0.956	3.59 ± 1.019	3.55 ± 1.016	3.56 ± 1.009	3.58 ± 1.011
PIMA	5.14 ± 0.652	5.17 ± 0.658	5.17 ± 0.657	5.15 ± 0.652	4.56 ± 0.641	4.57 ± 0.644	4.57 ± 0.644	4.56 ± 0.643
WINE	8.06 ± 0.947	8.10 ± 0.933	8.09 ± 0.939	8.08 ± 0.944	6.19 ± 0.733	6.24 ± 0.668	6.23 ± 0.652	6.23 ± 0.657
PARK	15.53 ± 2.423	15.56 ± 2.405	15.56 ± 2.414	15.55 ± 2.421	12.34 ± 1.483	12.49 ± 1.420	12.56 ± 1.475	12.65 ± 1.485
BRCW	15.72 ± 3.350	15.74 ± 3.348	15.73 ± 3.349	15.73 ± 3.349	16.54 ± 2.109	16.59 ± 2.035	16.66 ± 1.935	16.69 ± 1.874
IONS	18.00 ± 2.454	18.02 ± 2.457	18.02 ± 2.457	18.01 ± 2.456	13.32 ± 4.270	13.85 ± 3.805	14.03 ± 3.654	14.13 ± 3.599

Source: Author.

This is interesting since Table 8 clearly shows that the Twostep method achieved equal or better coverage in every scenario. This further evidences that a more strategic expansion of the ranges can lead to a better generalization, even if there is a reduction of the instance space.

We will present a brief case study based on the real application of the explanations generated by Onestep and Twostep. Let an instance from the IRIS dataset be $\mathbf{x} = \{sepal_length = 4.6, sepal_width = 3.4, petal_length = 1.4, petal_width = 0.3\}$ predicted as class *setosa* by a SVC. The Onestep explanation that guarantees the prediction is:

$$\begin{aligned} \text{IF } 2.51 \leq sepal_width \leq 4.40 \text{ AND } 1.00 \leq petal_length \leq 1.40 \\ 0.10 \leq petal_width \leq 0.30 \text{ AND THEN } setosa. \end{aligned}$$

Comparatively, the Twostep explanation is:

$$\begin{aligned} \text{IF } 2.84 \leq sepal_width \leq 4.40 \text{ AND } 1.00 \leq petal_length \leq 1.73 \\ 0.10 \leq petal_width \leq 0.38 \text{ AND THEN } setosa. \end{aligned}$$

In this case, the Onestep and the Twostep explanations have the *sepal_length* feature omitted since it can assume any value within the respective domain without changing the classification result. In comparison with the Onestep explanation, the Twostep explanation has a more narrow value interval for the *sepal_width* feature, which allows a broader interval for both *petal_length* and *petal_width* features.

As a result, the Twostep explanation achieved coverage of 13 IRIS dataset instances against the coverage of 8 instances of the Onestep explanation. Therefore, Twostep was able to cover 62.5% more than Onestep.

Moreover, let an instance from the VRTC dataset be $\mathbf{x} = \{pelvic_incidence = 63.03, pelvic_tilt = 22.55, lumbar_lordosis_angle = 39.61, sacral_slope = 40.48, pelvic_radius$

$= 98.67$, $degree_spondylolisthesis = -0.25$ } predicted as class AB for the SVC. The Onestep explanation for such a prediction is

IF $27.43 \leq lumbar_lordosis_angle \leq 125.74$ **AND** $13.37 \leq sacral_slope \leq 42.38$
 $70.08 \leq pelvic_radius \leq 98.67$ **THEN** AB .

In comparison, the Twostep explanation is

IF $31.24 \leq lumbar_lordosis_angle \leq 125.74$ **AND** $13.37 \leq sacral_slope \leq 51.47$
 $70.08 \leq pelvic_radius \leq 98.79$ **THEN** AB .

For this case, the Onestep and the Twostep explanations have the *pelvic_incidence*, *pelvic_tilt*, and *degree_spondylolisthesis* features omitted since they can assume any value within their respective domain without changing the classification result. The Twostep shortened the value interval of the *lumbar_lordosis_angle* feature in favor of broadening the intervals of the *sacral_slope* and *pelvic_radius* features. This change enabled the Twostep explanation to achieve coverage of 6 VRTC dataset instances against the coverage of 4 instances of the Onestep explanation, covering 50% more instances.

These cases make more evident how the Twostep differs from the Onestep, often shortening the ranges of the first features and widening the ranges of the following features whenever possible. It is also significant for Twostep, as it highlights the potential it has to achieve greater instance coverage compared to Onestep.

5 Conclusions and Future Works

This final chapter provides the conclusions, final remarks, and identifies future work regarding the topics discussed in this dissertation.

5.1 Conclusions

In this work, we introduced a logic-based approach to generate minimal explanations for a classifier while considering the presence of reject option. The approach can ensure correctness in the generated explanations. It was compared with Anchors, a well-known heuristic method, through six different datasets, with the SVC being the target model used for evaluation and comparison. The obtained results showed that our method achieved considerable improvements regarding the computing time and the size of explanations. Our approach achieved astonishing results in terms of efficiency, surpassing Anchors by an impressive factor of up to approximately 286 times.

Such an approach has some limitations and could be further improved. For example, it can be adapted to other classifiers with and without RO, such as neural networks and random forests. Since we guarantee correctness, the size of explanations may considerably increase, as observed for some cases in Table 4. Moreover, since we only assure minimal explanations, i.e., irreducible explanations that are sufficient for guaranteeing the prediction, there may be explanations with fewer features that our method does not guarantee to find out. Furthermore, some generated explanations could be more general. For example, some features in an explanation may change values and still lead to the same prediction. Then, a more general explanation may consider a range of values for each feature, such that the prediction does not change for any instances whose feature values fall within such ranges. Such an increase in the generalization capability of explanations could enable them to hold over more instances.

Improvements were made with the other two proposed methods, presented in this work for standard classifiers without RO. We proposed two logic-based approaches with guarantees of correctness and minimality while focusing on improving their generalization capability. These methods, called Onestep and Twostep, aim to achieve a higher generalization of the feature ranges, with Onestep being, in a certain way, similar to the method shown in Izza et al. (2023).

Our methods Onestep and Twostep were compared with each other, through eight different datasets. Comparing Twostep with Onestep is equivalent to comparing Twostep with the method shown in Izza et al. (2023). Both methods were applied over the MLP

and linear SVC models, the target classifiers. The obtained results show that the more strategic expansion used by Twostep, albeit non-insignificantly slower than Onestep, was able to achieve higher coverage while maintaining a similar or better explanation range.

There are some clear limitations to both of our methods Onestep and Twostep. The loss of coverage (for test data) of explanations due to factors like higher data dimensionality demonstrates that the approaches, while able to achieve considerable degrees of generalization, could still be insufficient for more complex datasets. This could lead to results as shown in Table 8 for some cases, where the coverage of explanations is lost. Moreover, since both methods start from a previous minimal explanation, there may be explanations with fewer features that our approach does not have guarantees to discover. These smaller explanations in turn could lead to explanations with higher coverage, even more so when considering that we are only evaluating through data in the datasets.

5.2 Future Work

Suggested future work includes adapting the logic-based approach for other classifiers with reject option, such as neural networks and random forests. Moreover, both of our methods Onestep and Twostep could be adapted for when reject option is present, which could lead to important observations regarding such a scenario.

Both generalization methods can be further improved. Previously analyzing the dataset instance distribution, for example, could lead to a base representation of how the features are organized in the instance space. This could be useful as a means to improve generalization by focusing on feature ranges that lead to higher coverage.

Fixing the size of the explanations may make them easier to understand, reducing the number of features needed to be considered. Such reduced explanations may not apply to all instances, but such a strategy could lead to a better understanding of the model and how impactful each feature really is to the predictions.

Another improvement would be to transform the range expansion task into a cost function, where a solver would analyze and decide the lower and upper value limits each feature can assume. This has the potential to achieve better range distributions between the features. These improvements might bring forth even more insightful explanations, therefore delivering an enhanced comprehension of the models.

Bibliography

- AUDEMARD, G. et al. On preferred abductive explanations for decision trees and random forests. In: *31st IJCAI*. [S.l.: s.n.], 2022. p. 643–650. Cited 3 times in pages 32, 33, and 42.
- BISHOP, C. M. *Neural networks for pattern recognition*. [S.l.]: Oxford university press, 1995. Cited in page 39.
- BISWAS, S. S. Role of chat gpt in public health. *Annals of biomedical engineering*, Springer, v. 51, n. 5, p. 868–869, 2023. Cited in page 31.
- CHLAOUA, R. et al. Deep learning for finger-knuckle-print identification system based on PCANet and SVM classifier. *Evolving Systems*, Springer, v. 10, n. 2, p. 261–272, 2019. Cited in page 31.
- CHOW, C. On optimum recognition error and reject tradeoff. *IEEE Transactions on Information Theory*, v. 16, n. 1, p. 41–46, 1970. Cited 2 times in pages 31 and 40.
- CORTES, C.; VAPNIK, V. Support-vector networks. *Machine learning*, Springer, v. 20, p. 273–297, 1995. Cited 2 times in pages 37 and 39.
- DIMANOV, B. et al. You shouldn't trust me: Learning models which conceal unfairness from multiple explanation methods. IOS Press, 2020. Cited in page 32.
- FISCHETTI, M.; JO, J. Deep neural networks and mixed integer linear optimization. *Constraints*, Springer, v. 23, n. 3, p. 296–309, 2018. Cited in page 56.
- GUYON, I. et al. Gene selection for cancer classification using support vector machines. *Machine learning*, Springer, v. 46, p. 389–422, 2002. Cited 2 times in pages 32 and 45.
- IGNATIEV, A. Towards trustable explainable AI. In: *IJCAI*. [S.l.: s.n.], 2020. p. 5154–5158. Cited 3 times in pages 31, 32, and 41.
- IGNATIEV, A. et al. Using MaxSAT for efficient explanations of tree ensembles. In: *AAAI*. [S.l.: s.n.], 2022. v. 36, p. 3776–3785. Cited 3 times in pages 32, 33, and 42.
- IGNATIEV, A.; NARODYTSKA, N.; MARQUES-SILVA, J. Abduction-based explanations for machine learning models. In: *AAAI*. [S.l.: s.n.], 2019. v. 33, p. 1511–1519. Cited 8 times in pages 32, 33, 42, 46, 47, 55, 56, and 57.
- IGNATIEV, A.; NARODYTSKA, N.; MARQUES-SILVA, J. On validating, repairing and refining heuristic ML explanations. *CoRR*, abs/1907.02509, 2019. Cited 2 times in pages 31 and 32.
- IGNATIEV, A.; NARODYTSKA, N.; MARQUES-SILVA, J. On formal reasoning about explanations. In: *RCRA*. [S.l.: s.n.], 2020. Cited 3 times in pages 32, 41, and 42.
- IZZA, Y.; IGNATIEV, A.; MARQUES-SILVA, J. On explaining decision trees. *arXiv preprint arXiv:2010.11034*, 2020. Cited 3 times in pages 32, 33, and 42.

IZZA, Y. et al. *Delivering Inflated Explanations*. 2023. Cited 6 times in pages 33, 55, 56, 59, 63, and 69.

KRAUSE, J. et al. *A Workflow for Visual Diagnostics of Binary Classifiers using Instance-Level Explanations*. 2017. Cited 2 times in pages 32 and 45.

KROENING, D.; STRICHMAN, O. *Decision procedures*. [S.l.]: Springer, 2016. Cited in page 43.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. *nature*, Nature Publishing Group UK London, v. 521, n. 7553, p. 436–444, 2015. Cited in page 31.

LUNDBERG, S. M.; LEE, S.-I. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, v. 30, 2017. Cited 3 times in pages 32, 41, and 42.

MARQUES-SILVA, J. et al. Explaining naive bayes and other linear classifiers with polynomial time and delay. *NeurIPS*, v. 33, p. 20590–20600, 2020. Cited 3 times in pages 32, 33, and 42.

MARQUES-SILVA, J.; IGNATIEV, A. Delivering trustworthy ai through formal xai. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. [S.l.: s.n.], 2022. v. 36, n. 11, p. 12342–12350. Cited in page 31.

MESQUITA, D. P. et al. Classification with reject option for software defect prediction. *Applied Soft Computing*, v. 49, p. 1085–1093, 2016. ISSN 1568-4946. Cited 3 times in pages 31, 40, and 41.

MILLER, G. A. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological review*, American Psychological Association, v. 63, n. 2, p. 81, 1956. Cited in page 32.

NAIR, V.; HINTON, G. E. Rectified linear units improve restricted boltzmann machines. In: *Proceedings of the 27th international conference on machine learning (ICML-10)*. [S.l.: s.n.], 2010. p. 807–814. Cited 2 times in pages 37 and 40.

OLIVEIRA, A. C. D. et al. Efficient minimal learning machines with reject option. In: IEEE. *5th BRACIS*. [S.l.], 2016. p. 397–402. Cited in page 40.

RAMKUMAR, M. et al. Ecg cardiac arrhythmias classification using dwt, ica and mlp neural networks. In: IOP PUBLISHING. *Journal of Physics: Conference Series*. [S.l.], 2021. v. 1831, n. 1, p. 012015. Cited in page 31.

RIBEIRO, M. T.; SINGH, S.; GUESTIN, C. “Why should I trust you?” explaining the predictions of any classifier. In: *22nd ACM SIGKDD*. [S.l.: s.n.], 2016. p. 1135–1144. Cited 3 times in pages 32, 41, and 42.

RIBEIRO, M. T.; SINGH, S.; GUESTIN, C. Anchors: High-precision model-agnostic explanations. In: *AAAI*. [S.l.: s.n.], 2018. v. 32. Cited 5 times in pages 32, 33, 34, 41, and 42.

RICHHARIYA, B. et al. Diagnosis of Alzheimer’s disease using universum support vector machine based recursive feature elimination (USVM-RFE). *Biomedical Signal Processing and Control*, Elsevier, v. 59, p. 101903, 2020. Cited in page 31.

Rocha Filho, F. M. et al. Logic-based explanations for linear support vector classifiers with reject option. In: NALDI, M. C.; BIANCHI, R. A. C. (Ed.). *Intelligent Systems*. Cham: Springer Nature Switzerland, 2023. p. 144–159. ISBN 978-3-031-45368-7. Cited in page 33.

SAMEK, W.; MÜLLER, K.-R. Towards explainable artificial intelligence. *Explainable AI: interpreting, explaining and visualizing deep learning*, Springer, p. 5–22, 2019. Cited in page 31.

SHIH, A.; CHOI, A.; DARWICHE, A. *A Symbolic Approach to Explaining Bayesian Network Classifiers*. 2018. Cited 2 times in pages 33 and 55.

VAPNIK, V. *Statistical learning theory*. [S.l.]: Wiley, 1998. ISBN 978-0-471-03003-4. Cited 2 times in pages 38 and 39.

WELD, D. S.; BANSAL, G. Intelligible artificial intelligence. *CoRR*, abs/1803.04263, 2018. Cited in page 31.

YANG, W. et al. Automatic recognition of arrhythmia based on principal component analysis network and linear support vector machine. *Computers in biology and medicine*, Elsevier, v. 101, p. 22–32, 2018. Cited in page 31.