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

Francisco Mateus Rocha Filho[0009−0002−2261−3767], Thiago Alves Rocha[0000−0001−7037−9683], Reginaldo Pereira Fernandes Ribeiro[0009−0000−0250−5713], and Ajalmar Rêgo da Rocha Neto[0000−0002−4512−5531]

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

Abstract. Support Vector Classifier (SVC) is a well-known Machine Learning (ML) model for linear classification problems. It can be used in conjunction with a reject option strategy to reject instances that are hard to correctly classify and delegate them to a specialist. This further increases the confidence of the model. Given this, obtaining an explanation of the cause of rejection is important to not blindly trust the obtained results. While most of the related work has developed means to give such explanations for machine learning models, to the best of our knowledge none have done so for when reject option is present. We propose a logic-based approach with formal guarantees on the correctness and minimality of explanations for linear SVCs with reject option. We evaluate our approach by comparing it to Anchors, which is a heuristic algorithm for generating explanations. Obtained results show that our proposed method gives shorter explanations with reduced time cost. Furthermore, although our approach is demonstrated with linear SVCs, it can be easily adapted to other classifiers with reject option, such as neural networks and random forests.

Keywords: Logic-based explainable AI · Support vector machines · Classification with reject option.

1 Introduction

It is undeniable that Artificial Intelligence is increasingly being inserted into the daily lives of people [14], influencing the most complex decision taking tasks [22]. Consequently, the most varied classification models in machine learning may come across instances that are difficult to correctly classify, be it due to lack of good data, to feature bias (color, size, gender) [7] or to noise present on the given instances [22].

The SVC is one of the most well-known ML models. The linear Support Vector Machine (SVM), specifically, has been used in a variety of classification problems [2,20,23]. 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 [3] can be used to remedy such cases. Classification with

RO is a paradigm that aims to improve reliability in decision support systems by handling the 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 [17]. 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 [6]. Such analysis, however, is not capable of giving decisive answers on more specific cases [12]. An example of this is depicted and more thoroughly explored in Section 3. This gives a margin for questionable explanations, especially when RO is present due to the added level of complexity. Therefore the need for a more thorough, instance-based explanation method.

Then, in this work, we consider instance-based explanations [7]. Specifically, the objective of instance-based explanations is to provide interpretable insights by highlighting relevant features that influenced the output of an ML model on a given instance. By linking the output to specific instances and their features, users can gain a better understanding of the reasons behind the predictions of ML models.

The popularization of concepts of Explainable Artificial Intelligence (XAI) increased efforts to explain most complex models [10]. These have mostly been done through heuristic methods, the more prevalent approach, such as LIME [18], SHAP [15], and Anchors [19]. These tend to be model-agnostic and not able to guarantee a trustworthy explanation with regard to the characteristics of the model or the correctness of the answers [10]. Then, their explanations can be often proven wrong through counterexamples that expose their contradictions, leading to even more doubts regarding how much the ML model can be trusted.

Thus, the importance of trustworthy ML models increases the need for logic-based approaches to explain the decisions made by these models [9,10]. 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 [7]. Moreover, logic-based approaches can guarantee minimality of explanations. Minimality is important since succinct explanations seem easier to be interpreted by humans. Recent years have seen a surge in research dedicated to investigating logic-based XAI for ML models, such as neural networks, naive Bayes, random forests, decision trees, and boosted trees [1,8,9,11,16].

Due to the importance of reject option and logic-based explainability for trustworthy ML models, this work proposes 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 [9]. We compute explanations for linear SVCs with RO by solving a set of logical constraints, specifically, boolean combinations of linear constraints. Despite the fact that we consider a linear SVC with RO in this work, our approach can be easily adapted for other ML models with RO, such as neural networks.

We conducted experiments to compare our approach against Anchors, a heuristic method that generates explanations by locally exploring the feature subspace close to a given instance [19], through six different datasets. 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 results of our experiments show that our approach is capable of generating succinct explanations up to 286 times faster than Anchors, in scenarios with and without the presence of reject option.

2 Background

2.1 Machine Learning and Binary Classification Problems

In machine learning, binary classification problems are defined over a set of features $\mathcal{F} = \{f_1, ..., f_n\}$ and a set of two classes $\mathcal{K} = \{-1, +1\}$. In this paper, 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 $l_i \leq x_i \leq u_i$. Then, each feature f_i has domain $[l_i, u_i]$. We represent this as a set of domain constraints $D = \{l_1 \leq f_1 \leq u_1, l_2 \leq f_2 \leq u_2, ..., l_n \leq f_n \leq u_n\}$. Besides, the notation $\mathbf{x} = \{f_1 = x_1, f_2 = x_2, ..., f_n = x_n\}$ represents a specific point or instance such that each x_i is in the domain of f_i.

A binary 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 $\{f_1 = x_1, f_2 = x_2, ..., f_n = x_n\}$ to class $+1$. Usually, the classifier is obtained by a training process given as input a training set $\{\mathbf{x}_i, y_i\}_{i=1}^{l}$, where $\mathbf{x}_i \in \mathbb{R}^n$ is an input vector or instance and $y_i \in \{-1, +1\}$ is the respective class label. Then, for each input vector $\mathbf{x}_i$, its input values $x_{i,1}$, $x_{i,2}$, ..., $x_{i,n}$ are in the domain of corresponding features f_1, ..., f_n. A well-known classifier and its training process are presented in Subsection 2.2.

2.2 Support Vector Machine

The SVM [4] is a supervised machine learning model often used for classification problems. It uses the concept of an optimal separating hyperplane [4] to separate the data. On a $\mathbb{R}^n$ space, such a hyperplane is defined by a set of points $\mathbf{x}$ that satisfies $\mathbf{w}_o \cdot \mathbf{x} + b = 0$, where $\mathbf{w}_o \in \mathbb{R}^n$ is the optimal 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 $\{\mathbf{x}_i, y_i\}_{i=1}^{k}$ is said to be linearly separable if there is $\mathbf{w}_o \in \mathbb{R}^n$ and $b \in \mathbb{R}$ that guarantees the separation between positive and negative class instances without error. In other words, the following inequalities

$$\text{for } i \in \{1, ..., k\}, \begin{cases} \mathbf{w}_o \cdot \mathbf{x}_i + b \geq +1, & \text{if } y_i = +1, \\ \mathbf{w}_o \cdot \mathbf{x}_i + b \leq -1, & \text{if } y_i = -1, \end{cases} \tag{1}$$

must be satisfied to obtain the optimal hyperplane $h_o = \{\mathbf{x} \mid \mathbf{w}_o \cdot \mathbf{x} + b = 0\}$.

A Hard Margin SVM (SVM-HM) [21] 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. 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:

$$\begin{aligned} \min_{\mathbf{w},b} \quad & \frac{1}{2}||\mathbf{w}||^2 \\ \text{s.t.} \quad & y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1, \quad \text{for } i \in \{1, ..., k\}. \end{aligned} \tag{2}$$

Thus, the decision function used for classifying input instances $\mathbf{x}$ is defined as $d(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b$, while the predicted label $\hat{y} \in \mathcal{K}$ of an input $\mathbf{x}$ is given by

$$\hat{y} = \begin{cases} +1, & \text{if } d(\mathbf{x}) \geq 0, \\ -1, & \text{if } d(\mathbf{x}) < 0. \end{cases} \tag{3}$$

However, in real-world problems, the training instances from the two classes can not be linearly separated by a hyperplane due to data overlapping. In order 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 [4,21]. Given this, the optimization problem for Soft-Margin SVMs is described as

$$\begin{aligned} \min_{\mathbf{w},b,\xi} \quad & \frac{1}{2}||\mathbf{w}||^2 - C\sum_{i=1}^{k} \xi_i \\ \text{s.t.} \quad & y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i, \qquad \text{for } i \in \{1, ..., k\} \\ & \xi_i \geq 0, \qquad\qquad\qquad\quad \text{for } i \in \{1, ..., k\} \end{aligned} \tag{4}$$

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 Reject Option Classification

Reject option for classification problems, as depicted in [3], is a set of techniques that aim to improve reliability in decision support systems. In the case of a binary problem, it consists in 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 [17].

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 [5]. According to [3], the optimal classifiers that best handle such a relation can be achieved by the minimization of $\hat{R}$ through

$$\hat{R} = E + w_r R \tag{5}$$

where R is the ratio of the number of rejected training instances to the number of instances in the entire training dataset; E is the ratio of the number of misclassified instances to the number of all the training instances without including those ones 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 [17] 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:

$$\hat{y} = \begin{cases} +1, & \text{if } f(\mathbf{x}) > t_+, \\ -1, & \text{if } f(\mathbf{x}) < t_-, \\ 0, & \text{otherwise}, \end{cases} \tag{6}$$

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.

2.4 First-Order Logic

In order to give explanations with guarantees of correctness, we use first-order logic (FOL) [13]. 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:

$$\begin{aligned} \varphi, \psi &:= s \mid (\varphi \wedge \psi) \mid (\varphi \vee \psi) \mid (\neg\varphi) \mid (\varphi \rightarrow \psi), \\ s &:= \sum_{i=1}^{n} a_i z_i \leq e \mid \sum_{i=1}^{n} a_i z_i < e, \end{aligned} \tag{7}$$

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,

$$\begin{aligned} \Gamma \cup \{\varphi \vee \psi\} \text{ is unsatisfiable iff } & \Gamma \cup \{\varphi\} \text{ is unsatisfiable and} \\ & \Gamma \cup \{\psi\} \text{ is unsatisfiable.} \end{aligned} \tag{8}$$

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,

$$\Gamma \models \varphi \text{ iff } \Gamma \cup \{\neg\varphi\} \text{ is unsatisfiable.} \tag{9}$$

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 $((z_1 + z_2 > 2) \wedge z_1 < 1)$.

2.5 XAI and Related Work

Some ML models are able to be interpreted by nature, such as decision trees [18]. 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 [10]. 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 [7,10].

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 [15,18,19]. 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 [10]. 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 [19]. 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 $I = \{sepal\_length = 5.0,\ sepal\_width = 2.3,\ petal\_length = 3.3,\ petal\_width = 1.0\}$ 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:

$$\textbf{IF } sepal\_width \leq 2.8 \textbf{ AND } petal\_length > 1.6 \textbf{ THEN } versicolor.$$

However, due to the heuristic characteristic of Anchors, there is still room for wrong explanations, which can lead to cases where, for the same set of given rules, different classes are predicted [10]. Therefore, both explanation correctness and size can be set as of questionable utility if they can not be fully relied upon.

Since heuristic-based explanations are unreliable, logic-based approaches have been proposed recently for a variety of ML models [1,8,9,11,16]. 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 $I = \{f_1 = x_1, f_2 = x_2, ..., f_n = x_n\}$ classified as class $c \in \{-1, +1, 0\}$, an *explanation* E is a subset of the instance $E \subseteq I$ 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 $I \setminus E$ 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.

3 Linear SVCs 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 [6]. 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 [12].

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 1. 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.

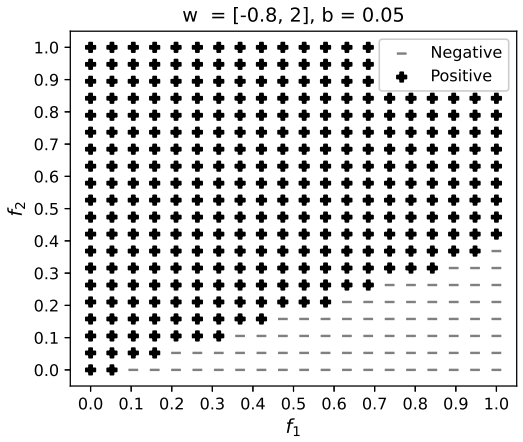

Fig. 1. Classification through features f_1 and f_2.

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 compute explanations with guarantees of correctness for linear SVCs with reject option.

4 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 [9] for traditional neural networks, i.e. without RO. Here 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 6, and an instance I classified as class $c = 0$, i.e. in the rejection class, we define the first-order formula P:

$$P = (\sum_{i=1}^{n} w_i f_i + b \leq t_+) \wedge (\sum_{i=1}^{n} w_i f_i + b \geq t_-). \tag{10}$$

Therefore, given an instance $I = \{f_1 = x_1, f_2 = x_2, ..., f_n = x_n\}$ such that its features are defined by $D = \{l_1 \leq f_1 \leq u_1, l_2 \leq f_2 \leq u_2, ..., l_n \leq f_n \leq u_n\}$, an explanation is a subset $E \subseteq I$ such that $E \cup D \models P$. Therefore, each assignment that satisfies all the formulas in $E \cup D$ must also satisfy P. In other words, prediction is guaranteed to be in the rejection class. Moreover, since our goal is to obtain minimal explanations, E must be such that, for all subset $E' \subset E$, it follows that $E' \cup D \not\models P$.

Instances predicted in other classes are treated in a similar way. For example, if instance I is classified as class $c = +1$, 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 2.4. Moreover, each first-order variable f_i represents a feature.

Earlier work [9] outlined Algorithm 1 for computing minimal explanations for traditional neural networks, i.e. without RO. By leveraging the insights from this earlier work, we can effectively employ Algorithm 1 to find a minimal explanation for linear SVCs with RO given I, D and P. If instance I is predicted as a class c by the linear SVC with RO, then P is defined accordingly to reflect c. The minimal explanation E of I is calculated by setting E to I and then removing feature by feature from E. For example, given a feature $f_i = x_i$ in E, if $E \backslash \{f_i = x_i\}, D \models P$, then the value x_i of feature f_i is not necessary to ensure the prediction, and then it is removed from E. Otherwise, if $E \setminus \{f_i = x_i\}, D \not\models P$, then $f_i = x_i$ is kept in E since it is necessary for the prediction. This process is performed for all features as described in Algorithm 1. Then, at the end of Algorithm 1, for the values of features in E, the prediction is the same and invariant with respect to the values of the remaining features.

By the result in 9, verifying entailments of the form $(E \setminus \{f_i = x_i\}) \cup D \models P$ can be done by testing whether the set of first-order formulas $(E \setminus \{f_i = x_i\}) \cup D \cup \{\neg P\}$ is unsatisfiable. If P is a conjunction $P_1 \wedge P_2$ as in 10, 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 8.

Algorithm 1 Computing a minimal explanation

Input: instance I predicted as c, domain constraints D, formula P according to c
Output: minimal explanation E
$E \leftarrow I$
for $f_i = x_i \in E$ **do**
  if $(E \backslash \{f_i = x_i\}) \cup D \models P$ **then**
    $E \leftarrow E \backslash \{f_i = x_i\}$
  end if
end for
return E

Moreover, since $(E \setminus \{f_i = x_i\}) \cup D$ is a set of linear constraints and $\neg P_1$ and $\neg P_2$ are linear constraints, the unsatisfiability checkings 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 reject option can also be solved in polynomial time. This is achieved by a linear number of calls to an LP solver, which further contributes to the efficiency and feasibility of our approach.

5 Experiments

In this paper, a total of 6 well known datasets are used. The Vertebral Column and the Sonar datasets are available on the UCI machine learning repository[1]. The Pima Indians Diabetes dataset is available on Kaggle[2]. The Iris, the Breast Cancer Wisconsin, and the Wine datasets are available through the scikit-learn package[3]. 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.

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

[1] https://archive.ics.uci.edu/ml/datasets.php
[2] https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database
[3] https://github.com/scikit-learn/scikit-learn/tree/main/sklearn/datasets/data

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

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 Subsection 2.3.

For defining the values of t_+ and t_-, we determine a range of thresholds $T = \{(t_+^1, t_-^1), ..., (t_+^p, t_-^p)\}$ 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 ambigous classification results for both known and unknown data. Thus, the attainable thresholds are achieved through $T = \{(i \cdot 0.01 \cdot upper\_limit,\ i \cdot 0.01 \cdot lower\_limit) \mid i \in \{1, ..., 100\}\}$. Hence, the selected values of t_+ and t_- are the pair that 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

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 approach[4] is written in Python and follows Algorithm 1. As an LP solver to check the unsatisfiability of sets of first-order sentences, we used Coin-or Branch-and-Cut (CBC)[5]. Moreover, the solver is accessed via the Python API PuLP[6].

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.

5.1 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

[4] https://github.com/franciscomateus0119/Logic-based-Explanations-for-Linear-Support-Vector-Classifiers-with-Reject-Option

[5] https://github.com/coin-or/Cbc

[6] https://github.com/coin-or/pulp

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

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

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 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 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 possibly 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 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.

6 Conclusions

In this paper, we propose 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 against 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 as the linear SVC. Nonetheless, we found that not only our approach takes considerably less time than Anchors, but it also has reduced size of explanations for many instances. Our approach achieved astonishing results in terms of efficiency, surpassing Anchors by an impressive factor of up to approximately 286 times.

However, our approach has limitations. 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, it is possible that there are explanations with fewer features that our method does not guarantee to find out.

Our approach can be further improved in future work. For example, it can be easily adapted to other classifiers with RO, such as neural networks and random forests. Another improvement is to provide more general explanations. For example, its possible that 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. This may enable a more comprehensive understanding of the model.

Acknowledgments. The authors thank FUNCAP and CNPq for partially supporting our research work.

References

1. Audemard, G., Bellart, S., Bounia, L., Koriche, F., Lagniez, J.M., Marquis, P.: On preferred abductive explanations for decision trees and random forests. In: 31st IJCAI. pp. 643–650 (2022)

2. Chlaoua, R., Meraoumia, A., Aiadi, K.E., Korichi, M.: Deep learning for finger-knuckle-print identification system based on PCANet and SVM classifier. Evolving Systems **10**(2), 261–272 (2019)
3. Chow, C.: On optimum recognition error and reject tradeoff. IEEE Transactions on Information Theory **16**(1), 41–46 (1970)
4. Cortes, C., Vapnik, V.: Support-vector networks. Machine learning **20**, 273–297 (1995)
5. De Oliveira, A.C., Gomes, J.P.P., Neto, A.R.R., de Souza, A.H.: Efficient minimal learning machines with reject option. In: 5th BRACIS. pp. 397–402. IEEE (2016)
6. Guyon, I., Weston, J., Barnhill, S., Vapnik, V.: Gene selection for cancer classification using support vector machines. Machine learning **46**, 389–422 (2002)
7. Ignatiev, A.: Towards trustable explainable AI. In: IJCAI. pp. 5154–5158 (2020)
8. Ignatiev, A., Izza, Y., Stuckey, P.J., Marques-Silva, J.: Using MaxSAT for efficient explanations of tree ensembles. In: AAAI. vol. 36, pp. 3776–3785 (2022)
9. Ignatiev, A., Narodytska, N., Marques-Silva, J.: Abduction-based explanations for machine learning models. In: AAAI. vol. 33, pp. 1511–1519 (2019)
10. Ignatiev, A., Narodytska, N., Marques-Silva, J.: On formal reasoning about explanations. In: RCRA (2020)
11. Izza, Y., Ignatiev, A., Marques-Silva, J.: On explaining decision trees. arXiv preprint arXiv:2010.11034 (2020)
12. Krause, J., Dasgupta, A., Swartz, J., Aphinyanaphongs, Y., Bertini, E.: A workflow for visual diagnostics of binary classifiers using instance-level explanations (2017)
13. Kroening, D., Strichman, O.: Decision procedures. Springer (2016)
14. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. nature **521**(7553), 436–444 (2015)
15. Lundberg, S.M., Lee, S.I.: A unified approach to interpreting model predictions. Advances in neural information processing systems **30** (2017)
16. Marques-Silva, J., Gerspacher, T., Cooper, M., Ignatiev, A., Narodytska, N.: Explaining naive bayes and other linear classifiers with polynomial time and delay. NeurIPS **33**, 20590–20600 (2020)
17. Mesquita, D.P., Rocha, L.S., Gomes, J.P.P., Rocha Neto, A.R.: Classification with reject option for software defect prediction. Applied Soft Computing **49**, 1085–1093 (2016)
18. Ribeiro, M.T., Singh, S., Guestrin, C.: "Why should I trust you?" explaining the predictions of any classifier. In: 22nd ACM SIGKDD. pp. 1135–1144 (2016)
19. Ribeiro, M.T., Singh, S., Guestrin, C.: Anchors: High-precision model-agnostic explanations. In: AAAI. vol. 32 (2018)
20. Richhariya, B., Tanveer, M., Rashid, A.H., Initiative, A.D.N., et al.: Diagnosis of Alzheimer's disease using universum support vector machine based recursive feature elimination (USVM-RFE). Biomedical Signal Processing and Control **59**, 101903 (2020)
21. Vapnik, V.: Statistical learning theory. Wiley (1998)
22. Weld, D.S., Bansal, G.: Intelligible artificial intelligence. CoRR **abs/1803.04263** (2018)
23. Yang, W., Si, Y., Wang, D., Guo, B.: Automatic recognition of arrhythmia based on principal component analysis network and linear support vector machine. Computers in biology and medicine **101**, 22–32 (2018)