A Python Library for Contextual SVM Explanations

Juan Pisco-Jordán
Dept. of Electrical and Computer Engineering
ESPOL Polytechnic University
Guayaquil, Ecuador
jpisco@espol.edu.ec

Luis Ramos-Pozo
Dept. of Electrical and Computer Engineering
ESPOL Polytechnic University
Guayaquil, Ecuador
lurapozo@espol.edu.ec

Ana Tapia-Rosero
Dept. of Electrical and Computer Engineering
ESPOL Polytechnic University
Guayaquil, Ecuador
atapia@espol.edu.ec

***Abstract*—Support Vector Machines (SVMs) are powerful classification models, but their black box nature often limits their adoption in high-stakes domains where interpretability is crucial. The contextualized SVM framework provides a path towards explainability by training specialized models for specific data contexts, but a practical software implementation has been missing. This paper introduces XSVM@ctx-Lib as an open-source Python library that implements the contextualized SVM architecture. The library is built to be computationally efficient by parallelizing both the model training and evaluation processes. XSVM@ctx-Lib was validated against standard SVM implementations using multiple datasets, demonstrating significant performance advantages. The library achieved reductions in training and classification times. Furthermore, the contextual approach improved classification precision and accuracy by up to 9%. XSVM@ctx-Lib provides researchers and practitioners with a powerful tool that enhances SVMs by simultaneously improving transparency, computational speed, and predictive accuracy.**

***Index Terms*—open-source library, Python, software, explainable AI, contextual machine learning, fuzzy sets**

I. Introduction

Machine Learning (ML) models have become integral to decision-making processes in multiple areas including agriculture, finance, and medicine. However, their effectiveness can be harmed by a lack of transparency. Many models are perceived as "black boxes" due to their complex internal workings, which are difficult for users to comprehend [1]. Support Vector Machines (SVMs) use advanced mathematics and high-dimensional data structures that are notoriously difficult to interpret [1]. In fields that demand high reliability, like medicine, this opacity generates mistrust and hinders the adoption of AI [2]. This gap has created a critical need for Explainable AI (XAI) techniques to build systems that are not only powerful, but also exhibit transparency and accountability.

Several approaches that provide explainability and transparency to machine learning models have emerged. For instance, *SHAP (SHapley Additive exPlanations)* is a unified framework for interpreting predictions [3]. Other approaches are based on rule extraction in which the SVM outputs are trained on more transparent models (e.g., decision trees) [1].

Although interesting studies provide the methodology, equations, and a variety of examples, it can be difficult to apply them to real-world scenarios when the tools are not provided. So, this study implements a specific contextual explanation framework proposed in [4]. The key mechanism involves partitioning the entire dataset based on predefined contexts, e.g., patient history or image location. A separate specialized SVM model is then trained for each context in parallel. When a new item is evaluated, the user can select the relevant context, and the system provides a classification using only that specific model. This is useful because it offers explanations using examples from the same context, while enhancing the classification accuracy in certain cases [4].

The main contribution of this study is an open-source Python library named xSVM@ctx-Lib which is based on XSVMC-Lib [5] and SciKit-Learn [6]. xSVM@ctx-Lib provides a parallelized implementation of the contextual SVM architecture. Its source code has been released under Apache License 2.01 [1], and can be found in [7]. The library was built using Python and leverages Joblib [2] to manage parallel training and evaluation of the model.

The remainder of this paper is structured as follows. Section II introduces the theoretical background of the contextualized SVM framework. Section III outlines the architecture, API, and implementation of the xSVM@ctx-Lib. Section IV presents three practical use cases of the Python library. Finally, the conclusions and future work are presented in Section V.

II. Preliminaries

SVMs are powerful classification models used in a variety of fields, from agriculture [8] to healthcare [9]. However, SVMs' high complexity often renders these models as *black boxes* making their internal decision-making process difficult to understand for human users [1].

SVM classifiers evaluate objects to determine or predict the classes to which those objects should belong. In the context of fuzzy set theory the degree to which an object belongs to a class can be formally represented by a membership function $\mu(x) \in [0, 1]$ where 0 and 1 correspond, respectively, to the lowest and highest membership levels of element x [10]. Intuitionistic fuzzy set (IFS) [11] expands this idea by including a nonmembership grade $\nu(x) \in [0, 1]$ for the same element x. When using an IFS, $0 \leq \mu(x) + \nu(x) \leq 1$ must hold, which allows for a degree of hesitation $h(x) = 1 - \mu(x) - \nu(x)$.

[1] https://www.apache.org/licenses/LICENSE-2.0
[2] https://joblib.readthedocs.io/en/stable/

Building on the aforementioned concepts of membership, nonmembership and explainability, the foundation of the proposed Python library is the *explainable SVM classification* XSVMC model [5] that extends traditional SVMs by using Augmented Appraisal Degrees (AADs) to provide contextual evaluations using contextless SVM models [4]. An AAD is a generalization of a membership (or non-membership) grade, that not only captures the level of class membership (or non-membership) of an evaluated object, but also the specific aspects or features of the data that lead to that class membership level, according to the knowledge K.In the particular case of SVM classification, K is compromised from the training set the model used during the learning process.

In the XSVMC framework, these influential aspects are identified as the Most Influential Support Vectors (MISVs). For any given prediction, the model identifies two key MISVs:

- **Positive MISV (MISV+):** The support vector from the predicted class that is most similar to the object being evaluated. It serves as an example to justify why the object belongs to that class.
- **Negative MISV (MISV-):** The support vector from a different class that is also highly similar to the object. It serves as a counterexample to explain why the object was not assigned to another similar class.

The information presented by the MISVs is provided by the knowledge the SVM acquired during the learning process, providing context to the evaluation. By presenting the MISV+ and MISV-, the model can generate example-based explanations , which provides a justification that is more understandable than the raw mathematical output of the SVM.

The interpretability of SVMs provided by the inclusion of AADs can be increased by using contextualized SVM models as presented in [4]. The contribution of this paper is an implementation of the theoretical proposal provided in [4], i.e., an ensemble model that makes use of contextualized data to create a contextualized SVM model for each context provided during the learning process. The creation and training of each contextualized SVM model can be done in parallel. The separation of data in multiple contexts and the parallelization of the training process can reduce computational costs and training time.

During the evaluation process, the data context is selected, and the SVM trained with that context is used for classification. Thus, this contextualization process of SVM models results in better interpretability as the MISVs obtained from the evaluation process share the same context as the evaluated object [4]. Contextualization has also been shown to increase performance, as each model is trained with data tailored to a specific context [12].

The context of the object should be assigned prior to the evaluation, although there are cases when the context of a dataset is unknown. For these cases, the Automatic Context Selection method presented in [13] can be used. This method detect the most suitable context for a dataset by using a subset of labeled samples within the dataset. The subset is then evaluated by each contextual SVM and the resulting predictions are compared with the original labels. The context of the model that yields the highest number of agreements is selected as the context for the full dataset, considering the principle that similar contexts tend to yield similar predictions [13]. At minimum, the labeled subset should include one sample per class, though this number should be increased if two contexts produce a tied agreement count.

III. XSVM@CTX-LIB

xSVM@ctx-Lib is a Python library designed to provide a practical implementation of the contextualized SVM model presented by Loor et al. [4]. It is built to be computationally efficient and easy to integrate into existing machine learning workflows. Its primary purpose is to simplify the process of interpreting SVMs by training a set of contextualized SVM models. The library's workflow involves partitioning a dataset by defined contexts, training a separate SVM for each partition in parallel, and then using this set of models to make predictions that are accompanied by context-aware explanations. The base SVM model used for contextualization is the XSVMC implementation [5]. This implementation was selected as its use of AADs during the training process provides MISVs for example-based contrastive explanations, and membership and non-membership degrees to increase interpretability and assess uncertainty during classification.

The library consists of three modules detailed in Table I. The *parallel_xsvm* module contains the *contextualized_xSVMC* class with the implementation of the contextualized SVM classifier. The *utils* module provides functions useful for decontextualizing, contextualizing and grouping data by defined contexts, processes commonly used when using the contextualized classifier. The *explanation* module contains helper functions that provide visual explanations for comparing the evaluated objects with the MISV+ and the MISV-. Each explanation type has three modes that can be selected with the *explanation_type* parameter: *favorable* that gives a comparison between the evaluated object and the MISV+; *unfavorable* that provides a contrastive explanation between the evaluated object and the MISV-; and *complete* that provides both the *favorable* and *unfavorable* explanations together.

TABLE I
SUMMARY OF THE MAIN LIBRARY COMPONENTS.

Module	Description
parallel_xsvm	The core implementation of the contextualized SVM classifier
utils	Provides helper functions for common tasks used for the training and classification process of the contextualized SVM classifier
explanation	Various functions that generate visual explanations of model predictions on image and tabular data.

The contextualized_xSVMC class exposes attributes that are essential for understanding the trained models. Key attributes are detailed in Table II. The attributes *classes_*, *support_vectors_* and *dual_coef_* return a dictionary where the key is a context and the value is the respective attribute of the contextualized SVM in that context.

TABLE II
RELEVANT ATTRIBUTES OF THE CONTEXTUALIZED_XSVMC CLASS

Attributes	Description
k	The number of the best predicted classes to be returned. This attribute is inherited from *xsvmlib.xsvmc.xSVMC*.
*classes_*	Labels of the classes identified during training, grouped by context.
*support_vectors_*	The support vectors from the resulting SVM models, grouped by context.
*dual_coef_*	Dual coefficients of the support vectors, grouped by context.
*contexts_*	The contexts in which the model has been trained on.

TABLE III
DESCRIPTION OF SVM METHODS

Methods	Description
fit(X, y)	Performs a parallel learning process to train a separate SVM model for each context provided in *X*.
predict_with_context(X)	Performs a parallel prediction for the input samples. Each sample is classified using the SVM model corresponding to its context.
predict_with_unknown_context(X, labeled_X, y)	Performs the Automatic Context Selection process and predicts with the assigned context.

The primary methods for interacting with the model are focused on training and prediction, as described in Table III. The methods *fit(X, y)* and *predict_with_context(X)* receive a dictionary X whose key corresponds to a context and its value is a list of the context's data. *fit(X, y)* also receives a dictionary y, whose key corresponds to a context in this case, and its value corresponds to a list of labels for each instance of x. The method *predict_with_unknown_context(X, labeled_X, y)* receives two dictionaries: *labeled_X* and y. *labeled_X*'s key corresponds to a context and its value to a list of data of that context. y's key also corresponds to a context, but its value corresponds to labels of each data; X is a list of data from unknown context. In all of these methods, the output is a dictionary where each key corresponds to an evaluated context and the values correspond to the labels of the data in that context.

The library is designed with a straightforward API that mirrors the conventions of popular machine learning libraries like Scikit-Learn. Fig. 1 presents the interactions of these methods during training, context selection and testing of the contextualized model. To ensure modularity, these methods were designed to be independent from the base model used for contextualization, which facilitates the integration with other ML models outside SVM. The *contextualized_xSVMC* class first instantiates each SVM model in parallel using the *fit* method. To classify data with the trained model the class uses either the *predict_with_context* method or the *predict_with_unknown_context* method. The aforementioned uses *automatic_context_selection* and *predict_with_context* methods to select a context and predict using it. Both cases results in contextualized predictions with their respective MISVs. Listing 1 demonstrates a typical workflow using the contextualized model with the library.

```
# Import and initialize the model
from xSVM_ctx.parallel_xsvm import
    contextualized_xSVMC
model = contextualized_xSVMC(kernel='linear')
# Train the model in parallel
model.fit(contextualized_X_train,
    contextualized_y_train)
# Contextualize the test data
from xSVM_ctx.utils import group_data_by_context
grouped_data_test = group_data_by_context(X_test,
    context_test)
# Make predictions in parallel
predictions = model.predict_with_context(
    grouped_data_test)
```

Listing 1. Usage Example of the Library

IV. USE CASES

To demonstrate its versatility and practicability, XSVM@ctx-Lib was validated across three use cases: one involving image data, another focusing on natural language processing (NLP), and a third using tabular data. These examples show how researchers and practitioners can obtain coherent explanations for different data types.

To validate the performance of the library, an experimental comparison against the standard Scikit-Learn SVM classifier (SVC) [3] was conducted. All comparisons were done using the same hyperparamenters for both models, as well as using the same training and testing dataset for each model. For all use cases, the reported metrics are computed over the entire test set, where each test sample is evaluated by its corresponding contextual SVM, with all predictions consolidated into a single result set for evaluation.

A. Image Data

For this example, the HAM10000 (Dermatoscopic Dataset) [14] was used. The dataset presents multiple photos of different pigmented skin lesions with information of its location on the body.

Herein, the body location was used as the context for the model, namely, *head, torso, back, lower limbs, upper limbs, and unknown*. Only the lesions confirmed by *expert consensus* and by *in-vivo confocal microscopy* were considered. The data was split in 70% for training and 30% for testing with each context. A RBF kernel with hyperparameters $C = 40$, $\gamma = 10$ and a balanced class weight were used.

Table IV shows the performance of the contextualized model compared with the SVC model, indicating a clear improvement of approximately 4.5% across all metrics for the contextualized_xSVMC model. This shows that, since lesions exhibit different characteristics depending on body location, contextualization produced more homogeneous subsets, reducing conflicting cases and improving the model's predictions.

The *show_object_comparison* function provides a direct visual justification. It displays the input image alongside its

[3]https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

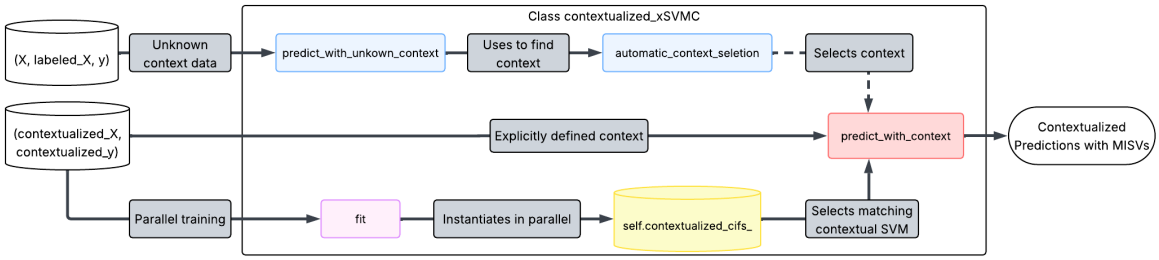

Fig. 1. Execution pipeline of the contextualized_xSVMC class during training, context selection and prediction.

TABLE IV
CLASSIFICATION METRICS (DERMATOSCOPIC DATASET)

Classification Metric	SVC	contextualized_xSVMC
Accuracy	79.8%	84.2%
Precision	79.5%	84.2%
Recall	80.0%	84.5%
F1-Score	79.8%	84.2%

MISV+, that corresponds to the most similar image from the training set that belongs to the predicted class. This allows the user to understand the prediction through a visual analogy.

Another feature is the option to ask for a counterexample that displays the MISV-, the most similar image from a different class. As shown in Fig. 2, the *complete* explanation type option adds to the interpretability of the answers by showing not only why an image was classified as a particular class, but also presenting a visually similar case from the same context that is not in the same class. This offers a contrastive explanation that helps delineate the model's decision boundaries in a understandable way.

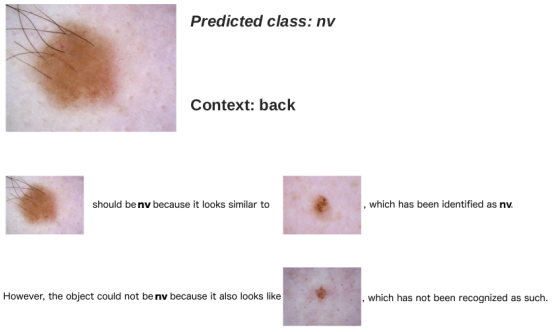

Fig. 2. Explanation of predictions by visual reference.

The user can ask the library to generate an influence map with the *show_influence_map* function, as seen in Fig. 3. This function highlights the regions of the input image most significant for the classification. This type of explanations can reveal what the model is basing its diagnosis of. For example, in Fig. 3 the diagnosis of a skin lesion is relying on the presence of hairs or specific texture patterns. This use case is vital for identifying potential model biases or false correlations. By visualizing the reasoning of the SVM, an expert can assess whether the model's focus aligns with their knowledge, which helps to build trust or to flag the need for model retraining.

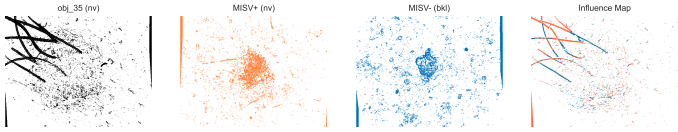

Fig. 3. Explanation of predictions by influence map.

B. NLP Data

The AG's corpus of news articles (News Dataset) [15], presents multiple news articles from different news outlets, with four classes denoting the type of news of the article: World, Sports, Business, Sci/Tech. Only the headlines were used for classification.

The dataset was divided in three contexts corresponding to the outlets with at least 160 recorded articles on each type of news. Each headline was converted to lowercase and tokenized using the NLTK word tokenizer [16]. Then, stopwords, terms referencing the news outlets, tokens shorter than three characters, and non-alphabetic tokens were removed. The remaining tokens were stemmed using the Porter Stemming Algorithm [17]. Each headline was then vectorized using TF-IDF. The resulting vectors were normalized using standardization without mean. A linear kernel, with balanced class weights, were used for classification.

As shown in Table V, both models achieve a similar performance on this dataset. This shows that, when contexts are highly similar, contextualization does not lead to noticeable improvements and may lead to marginal variations in performance instead.

TABLE V
CLASSIFICATION METRICS (NEWS DATASET)

Classification Metric	SVC	contextualized_xSVMC
Accuracy	89.15%	89.07%
Precision	89.07%	88.93%
Recall	89.22%	89.11%
F1-Score	89.15%	89.07%

Fig. 4 shows a comparison produced by the function *show_word_comparison*. This function plots the spatial distances between word embeddings from the evaluated object

and its MISVs. Dimensionality reduction was performed using PCA, but T-SNE can also be used with the *method* parameter. The visualization distinguishes words unique to each sentence from those shared across sentences. In this example, the words from the MISV+ are closer in space to those of the evaluated object than those from the MISV-. Fig. 4 also shows that the evaluated object shares more words in common with the MISV+ than with the MISV-.

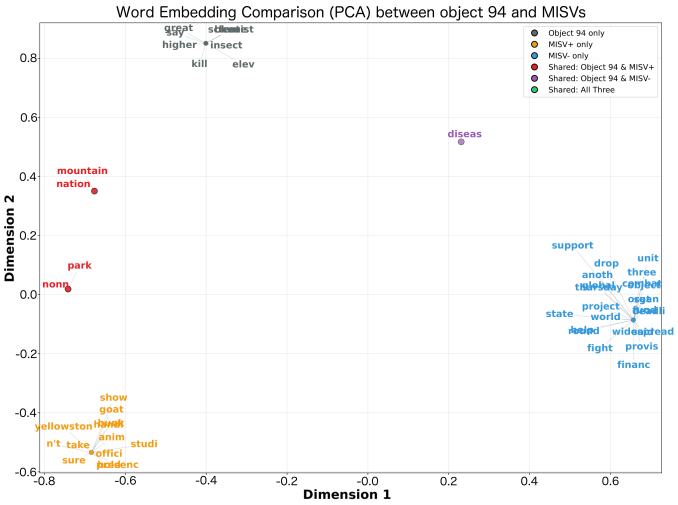

Fig. 4. Explanation of predictions with word embeddings.

Fig. 5 presents a comparison generated by the function *show_hierarchical_clustering*, which plots hierarchical clusters of words across sentences, where smaller distances indicate greater similarity. The results show that all words in the evaluated object are clustered closer to the MISV+ than to the MISV-.

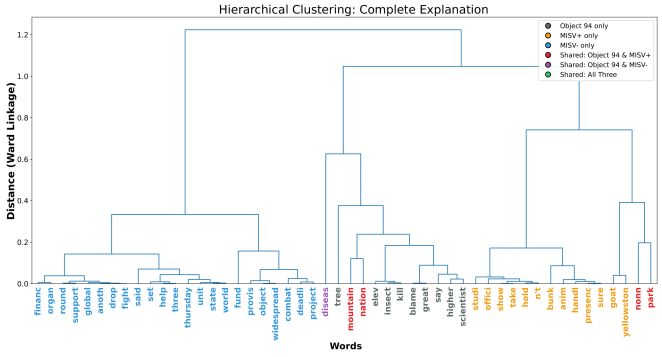

Fig. 5. Explanation of predictions with hierarchical clustering.

C. Tabular Data

The EMG Data for Gestures (Hand Movement Dataset) [18] consists of EMG recordings from multiple subjects, classified into seven types of hand gesture. The dataset was divided in 36 contexts using each individual in the dataset as a context. Records containing null values were discarded. The remaining data were normalized using L2 normalization. The hyperparameters used were an RBF kernel, $C = 100$ and $\gamma = 20$.

As shown in Table VI, when comparing the models on the Hand Movement Dataset, there is a 9% improvement in all evaluated metrics for the contextualized_xSVMC model.

TABLE VI
CLASSIFICATION METRICS (HAND MOVEMENT DATASET)

Classification Metric	SVC	contextualized_xSVMC
Accuracy	88.9%	97.9%
Precision	88.9%	97.9%
Recall	88.9%	97.9%
F1-Score	88.9%	97.9%

The *show_heatmap_difference* function was used to provide a local explanation of a classification. This function creates a heatmap, presented in Fig. 6, that color-codes the differences in feature values between the input, the MISV+, and the MISV-. The plot encodes these differences with color intensity where higher intensities indicating larger changes in feature values, with red indicating positive differences and blue indicating negative differences. This allows for a fast identification of the key attributes that drove the model to the classification.

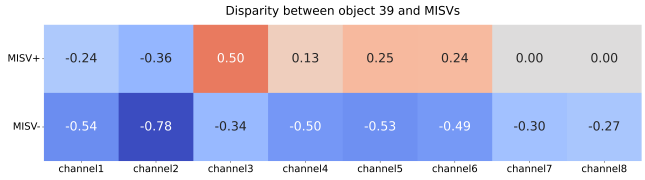

Fig. 6. Explanation of predictions by heatmap.

Additionally, the library provides other functions like *show_line_chart_comparison* and *show_kde_comparison* that present alternative visualizations to compare the input with their MISVs. These functions illustrate how each feature of the evaluated object aligns with, or deviates from, the corresponding features of the MISVs, providing further insight into the model's decision.

D. Execution Time Evaluation

As the training and testing of the contextualized_xSVMC can be parallelized, a comparison with the standard SVC was made to measure the impact of the library's parallelized and contextualized architecture on computational speed. All experiments were executed in the same device: Ryzen 9 5900X CPU and with 32 GB of RAM.

The reduction of processing time was dependent on the number of contexts defined by the dataset, as evidenced when comparing the reduction on training time between the Hand Movement Dataset and the other datasets. The Hand Movement Dataset was trained with 36 contexts, resulting in a training time of only 18 seconds for the contextualized_xSVMC, a significant reduction from 1 hour 22 minutes obtained on the standard SVC, as shown in Table VII. The other two datasets also showed a reduction in training times, albeit not as drastic as the Hand Movement Dataset.

TABLE VII
COMPARISON OF TRAINING TIMES

Data Set	Training time	
	SVC	contextualized_xSVMC
Hand Movement Dataset	1h 22m 53s	0h 0m 18s
Dermatoscopic Dataset	0h 0m 42s	0h 0m 17s
News Dataset	0h 40m 36s	0h 18m 20s

A similar reduction was seen in the evaluation times indicated in Table VIII, with the biggest reduction being from the News Dataset, followed by the Hand Movement Dataset. Overall, the contextualized_xSVMC provided evaluating times of less than four minutes across all datasets used in this study. These reductions in both training and evaluation times are useful for both researchers and practitioners when iterating on multiple hyperparameters to find optimal configurations. Additionally, this increased computational efficiency makes the framework more accessible to users with limited hardware resources, as high-performance results can be achieved without the need for high-end computing infrastructure.

TABLE VIII
COMPARISON OF EVALUATION TIMES

Data Set	Evaluating time	
	SVC	contextualized_xSVMC
Hand Movement Dataset	0h 20m 9s	0h 3m 50s
Dermatoscopic Dataset	0h 1m 13s	0h 0m 3s
News Dataset	0h 9m 34s	0h 0m 12s

V. CONCLUSIONS

In this paper, the Python library XSVM@ctx-Lib was introduced for developing and explaining contextualized Support Vector Machine models. The library provides a practical implementation of the parallel contextual SVM architecture proposed in [4]. This open-source library is available to the community in [7] to promote explainable machine learning techniques and enhance its current capabilities.

XSVM@ctx-Lib provides two significant advantages over the standard SVC implementation. First, by leveraging parallel processing, contextualized_xSVMC improved computational efficiency, decreasing both training and classification times. Second, the contextualization approach improved the classification accuracy by up to 9%, with the magnitude of improvement varying according to how distinct the contexts are, as partitioning data into more homogeneous subsets can reduce conflicting cases. In cases where the contexts were similar (e.g., the News dataset), performance did not improve, as the decision boundaries learned by each contextual SVM were also similar.

The main goal of XSVM@ctx-Lib is to generate explainability. It integrates multiple explanation methods and extends them for the contextual framework, allowing users to understand why a classification was made using intuitive visualizations like influence maps, feature-comparison heatmaps, and example/counterexample references.

A. Future Work

Bearing in mind that the core parallel architecture is model-agnostic its application could be explored for other classifiers beyond SVMs, such as Linear Regression and Naive Bayes. Additionally, future evaluations will incorporate more modern and advanced classification methods to better showcase the potential of this contextualized approach. Finally, future work should adopt a Human Computer Interaction (HCI) approach for comparing the explanations based on MISVs against established post-hoc surrogate XAI techniques like *SHAP*, *LIME*, or *NICE* in order to determine whether they provide higher faithfulness and interpretability.

REFERENCES

[1] A. Chatchinarat, K. W. Wong, and C. C. Fung, "Rule extraction from electroencephalogram signals using support vector machine," *2017 9th International Conference on Knowledge and Smart Technology: Crunching Information of Everything, KST 2017*, pp. 106–110, 3 2017.

[2] K. Witkowski, R. Okhai, and S. R. Neely, "Public perceptions of artificial intelligence in healthcare: ethical concerns and opportunities for patient-centered care," *BMC Medical Ethics 2024 25:1*, vol. 25, pp. 1–11, 2024.

[3] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[4] M. Loor, A. Tapia-Rosero, and G. De Tré, "Contextual boosting to explainable svm classification," in *Conference of the European Society for Fuzzy Logic and Technology*. Springer, 2023, pp. 480–491.

[5] M. Loor, A. Tapia-Rosero, and G. D. Tré, "An open-source software library for explainable support vector machine classification," *IEEE International Conference on Fuzzy Systems*, vol. 2022-July, 2022.

[6] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[7] J. Pisco-Jordán, L. Ramos-Pozo, and A. Tapia-Rosero, "xsvm@ctx-lib source code," https://github.com/interpretapple-lab/xsvm_ctx-lib, 2026.

[8] D. Banerjee, V. Kukreja, S. Hariharan, and V. Jain, "Enhancing mango fruit disease severity assessment with cnn and svm-based classification," *2023 IEEE 8th International Conference for Convergence in Technology, I2CT 2023*, 2023.

[9] D. Keerthana, V. Venugopal, M. K. Nath, and M. Mishra, "Hybrid convolutional neural networks with svm classifier for classification of skin cancer," *Biomedical Engineering Advances*, vol. 5, p. 100069, 2023.

[10] L. A. Zadeh, "Fuzzy sets," *Information and control*, vol. 8, no. 3, pp. 338–353, 1965.

[11] K. T. Atanassov, *On intuitionistic fuzzy sets theory*. Springer, 2012, vol. 283.

[12] D. Malowany and H. Guterman, "Biologically inspired visual system architecture for object recognition in autonomous systems," *Algorithms 2020, Vol. 13, Page 167*, vol. 13, p. 167, 7 2020.

[13] M. Loor and G. D. Tré, "Automatic context selection in explainable support vector machine classification," *IEEE International Conference on Fuzzy Systems*, 2024.

[14] P. Tschandl, C. Rosendahl, and H. Kittler, "The ham10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions," *Scientific Data 2018 5:1*, vol. 5, pp. 1–9, 2018.

[15] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Proceedings of the 29th International Conference on Neural Information Processing Systems - Volume 1*, ser. NIPS'15. Cambridge, MA, USA: MIT Press, 2015, p. 649–657.

[16] S. Bird, "Nltk: the natural language toolkit," in *Proceedings of the COLING/ACL 2006 interactive presentation sessions*, 2006, pp. 69–72.

[17] M. F. Porter, "An algorithm for suffix stripping," *Program*, vol. 14, no. 3, pp. 130–137, 1980.

[18] N. Krilova, I. Kastalskiy, V. Kazantsev, V. A. Makarov, and S. Lobov, "EMG Data for Gestures," UCI Machine Learning Repository, 2018.