Escuela Superior Politécnica del Litoral

Facultad de Ingeniería en Electricidad y Computación

Desarrollo de una librería para la explicación de resultados de SVM en contexto

TECH-380

Proyecto Integrador

Previo la obtención del Título de:

Ingeniero en Ciencias de la Computación

Presentado por:

Juan Carlos Pisco Jordán

Luis Ramos Pozo

Guayaquil - Ecuador

Año: 2024

Declaración Expresa

Nosotros Juan Carlos Pisco Jordán y Luis Enrique Ramos Pozo acordamos y reconocemos que:

La titularidad de los derechos patrimoniales de autor (derechos de autor) del proyecto de graduación corresponderá al autor o autores, sin perjuicio de lo cual la ESPOL recibe en este acto una licencia gratuita de plazo indefinido para el uso no comercial y comercial de la obra con facultad de sublicenciar, incluyendo la autorización para su divulgación, así como para la creación y uso de obras derivadas. En el caso de usos comerciales se respetará el porcentaje de participación en beneficios que corresponda a favor del autor o autores.

La titularidad total y exclusiva sobre los derechos patrimoniales de patente de invención, modelo de utilidad, diseño industrial, secreto industrial, software o información no divulgada que corresponda o pueda corresponder respecto de cualquier investigación, desarrollo tecnológico o invención realizada por mí/nosotros durante el desarrollo del proyecto de graduación, pertenecerán de forma total, exclusiva e indivisible a la ESPOL, sin perjuicio del porcentaje que me/nos corresponda de los beneficios económicos que la ESPOL reciba por la explotación de mi/nuestra innovación, de ser el caso.

En los casos donde la Oficina de Transferencia de Resultados de Investigación (OTRI) de la ESPOL comunique a los autores que existe una innovación potencialmente patentable sobre los resultados del proyecto de graduación, no se realizará publicación o divulgación alguna, sin la autorización expresa y previa de la ESPOL.

Guayaquil, 21 de mayo del 2024.

Juan Carlos Pisco Jordán

Luis Enrique Ramos Pozo

Evaluadores

Firmado electrónicamente por:
BORIS XAVIER VINTIMILLA BURGOS

Boris Xavier Vintimilla Burgos

Profesor de Materia

Firmado electrónicamente por:
ANA TERESA TAPIA ROSERO

Ana Teresa Tapia Rosero

Tutor de proyecto

Resumen

Las SVM son modelos de inteligencia artificial usados para resolver problemas de clasificación de datos. Estos modelos pueden ser complejos de interpretar, lo que afecta su usabilidad en ámbitos donde se requiere conocer el porqué de la clasificación. El modelo SVM contextualizado provee explicaciones por contexto, pero no existe una implementación de código que facilite su integración en proyectos e investigaciones. Por esto se decidió desarrollar una librería de código abierto que implemente las funcionalidades de SVMs contextualizados para facilitar el acceso a explicaciones de las clasificaciones realizadas por estos modelos. La librería se desarrolló en Python, se utilizó Joblib para paralelizar las etapas de entrenamiento y evaluación del modelo. El uso de esta librería permitió reducciones en los tiempos de entrenamiento de más de una hora, y reducciones de hasta el 80% en el tiempo de clasificación. También mejoró la precisión y exactitud hasta el 10% en múltiples conjuntos de datos en comparación al SVM no contextualizado. La librería desarrollada resulta ser una herramienta muy útil para generar transparencia en clasificaciones de modelos SVM a la vez que mejora su exactitud y facilita el uso de un mayor volumen de datos en el entrenamiento y evaluación del modelo.

Palabras Clave: IA explicable, software, código abierto, aprendizaje de máquina

Abstract

SVMs are artificial intelligence models widely used to solve classification problems. These models can be complex to interpret, which affects their usability in domains where it is necessary to know the reasoning behind their classifications. The contextualized SVM model provides explanations by context, but there is no code implementation that facilitates its integration in projects and research. For this reason, the objective of this project was to develop an open-source library that implements the functionalities of contextualized SVMs to provide tools for explanations of the classifications made by these models. The library was developed in Python, and Joblib was used to parallelize the training and evaluation stages of the model. The use of this library allowed reductions in training times of more than one hour, and reductions of up to 80% in classification time. It also improved precision and accuracy by up to 10% on multiple datasets compared to the non-contextualized SVM. The developed library proves to be a very useful tool to generate transparency in SVM model classifications while improving its accuracy and facilitating the use of a larger volume of data in model training and evaluation.

Keywords: explainable AI, software, open-source, machine learning

Índice General

Abreviaturas

IA	Inteligencia Artificial
SVM	Support Vector Machine
API	Application Programming Interface
MISV	Most Influential Support Vector
EMG	Electromiografía
KDE	Kernel Density Estimation

Índice de Figuras

Índice de Tablas

Capítulo 1

1.1 Introducción

En la actualidad, los modelos de inteligencia artificial (IA) se utilizan para realizar tareas de clasificación en diversos campos como la agricultura [10], la medicina [11] y las finanzas [12]. Existen modelos que enfrentan un desafío significativo: la falta de explicabilidad. A menudo, estos modelos operan como "cajas negras", es decir, sistemas cuyos procesos internos son tan complejos que resultan incomprensibles para los usuarios [1], lo que dificulta entender qué factores influyen en los procesos de toma de decisiones de los modelos o en cómo se llega a las clasificaciones. Existen casos donde para apoyar a la toma de decisiones basta con la clasificación que otorga el modelo, pero existen otros casos donde también es necesario algún tipo de justificación. Por ejemplo, en un sistema de diagnóstico médico que implemente modelos de IA, no basta solo con saber si un paciente padece una enfermedad o no [1], sin una comprensión clara del proceso para clasificar a los pacientes se genera desconfianza en el sistema, ralentizando los veredictos. Existe miedo en la comunidad médica de que, en caso de implementar modelos IA en los diagnósticos los médicos no puedan dar una correcta explicación a los pacientes sobre lo que padecen debido a que ellos mismos no lo comprenden completamente [9].

Superar este problema de falta de transparencia es crucial. Proporcionar una forma de entender mejor el proceso de clasificación permitirá a investigadores y profesionales identificar patrones y tomar decisiones más informadas. Por ejemplo, en un sistema de diagnóstico médico que implemente modelos de IA, si los clínicos pudieran ver y entender los factores específicos que llevaron a una clasificación, como síntomas clave o resultados de pruebas específicas, aumenta la confianza en el modelo IA y permite ajustar los tratamientos a conciencia. Ahondando un poco en el ámbito jurídico, la interpretabilidad en los modelos de IA es esencial para cumplir con regulaciones y estándares éticos que demandan transparencia en el proceso de toma de decisiones automatizadas [1, 2]. Organizaciones como la UNESCO indican que, la

transparencia y la explicabilidad son esenciales para asegurar que se respeten, protejan y promuevan los derechos humanos, libertades fundamentales y principios éticos [8]. De esta manera, se promueve una mayor responsabilidad y rendición de cuentas.

Lograr la transparencia en los modelos IA es un desafío complejo debido a varias razones. Primero, muchos algoritmos de IA, como las Máquinas de Vectores de Soporte (SVM, por sus siglas en inglés), utilizan matemáticas avanzadas y estructuras de datos multidimensionales que son difíciles de interpretar [1]. Por ejemplo, en un sistema de diagnóstico médico que implemente modelos de IA como las SVM, estas pueden considerar una vasta cantidad de características y relaciones entre datos médicos que no son evidentes a simple vista, interacciones complejas entre síntomas, antecedentes médicos y resultados de pruebas. Por ejemplo, una interacción compleja podría ser la relación entre un síntoma específico, como la fiebre, y antecedentes médicos como hipertensión, que, junto con ciertos resultados de pruebas sanguíneas, podrían influir en la clasificación del diagnóstico de una enfermedad específica. Debido a esta complejidad, que abarca múltiples variables y sus interrelaciones no lineales, es esencial desarrollar métodos que desglosen las capas de procesamiento del modelo, tales como la selección de características relevantes, la ponderación de dichas características y los procesos de toma de decisiones llevados por el modelo. Presentar esta lógica de manera comprensible para los profesionales de la salud es fundamental para aumentar la transparencia y confianza en los sistemas de IA.

Cuando se habla de interpretabilidad de un modelo IA, se refiere a que la lógica detrás de los procesos de toma de decisión de estos pueda ser interpretada por expertos y pueda ser explicadas a usuarios en un lenguaje accesible [8]. Han existido varias formas que buscan mejorar la interpretabilidad en modelos de caja negra, como es la contextualización de los modelos [2, 3, 4]. En este caso, el contexto se refiere a aquella información adicional relevante que puede ser tomado en cuenta en los procesos de toma de decisiones del modelo [3]. Conocer

el entorno en el que opera un modelo, entre otras cosas, puede ayudar a entender mejor lo procesos de toma de decisiones del modelo, otorgando un poco más de transparencia a lo que en otras circunstancias sería una caja negra. Por ejemplo, en el diagnóstico médico, el entorno podría incluir datos como el historial médico del paciente, estilo de vida y condiciones actuales, además de resultados de pruebas recientes. Al incorporar estos datos específicos, el modelo puede proporcionar clasificaciones más precisas y explicaciones más comprensibles. Por ejemplo, si un paciente tiene antecedentes de alergias severas y presenta síntomas específicos durante una consulta, el modelo contextualizado puede utilizar estos datos para priorizar ciertas condiciones o tratamientos en su diagnóstico, ofreciendo así explicaciones relevantes para el médico y el paciente. Esto aumenta la confianza en el sistema y mejora la calidad de la atención médica al personalizar las recomendaciones basadas en el contexto de cada paciente.

1.2 Descripción del Problema

Las SVMs son modelos muy utilizados en un amplio rango de campos, útiles para resolver problemas de clasificación, pero su funcionamiento interno resulta difícil de interpretar [1]. Existen casos donde se busca una mayor transparencia en el razonamiento detrás de sus clasificaciones [1] para desentrañar la lógica interna de estos modelos y poder explicarla de forma comprensible. Para abordar este desafío, se han creado librerías como [5], que proporciona explicaciones en los procesos de toma de decisiones de los modelos SVM.

Este proyecto de grado consiste en desarrollar una librería que implemente el clasificador SVM explicable propuesto en [2] el cual hace uso de modelos SVM contextualizados. Entre las restricciones a considerar se encuentra que la solución debe ser compatible con varios sistemas operativos, debe ser eficiente computacionalmente y soportar grandes cantidades de datos. Además, debe incluir una interfaz intuitiva para facilitar el uso de la librería, permitiendo la visualización de los resultados. Las variables de interés para la evaluación de la solución son la precisión de las clasificaciones, el tiempo de procesamiento y el nivel de detalle de las

explicaciones proporcionadas. Como objetivo de este proyecto se busca que la librería permita configurar SVMs para funcionar con diferentes contextos para proporcionar una explicación de los resultados obtenidos por este modelo.

1.3 Justificación del Problema

Muchos de estos modelos de caja negra han demostrado ser de gran utilidad al momento de encontrar patrones para la correcta clasificación de diferentes objetos y escenarios. En particular, los SVM se destacan por ser efectivos en múltiples aplicaciones como la clasificación de texto, reconocimiento de imágenes, entre otros.

Existen situaciones donde no solo es necesario obtener una clasificación, sino que también es necesario que estas clasificaciones estén acompañadas de una justificación [1, 2]. Es en estos casos donde muchos de los modelos SVM se ven limitados al no poder proveer la motivación de sus clasificaciones. Existen alternativas para otorgar interpretabilidad a estos modelos como [5] que permiten utilizar modelos SVM en situaciones donde se requiere que la clasificación esté acompañada de una explicación.

Mejorar la interpretabilidad de los SVMs genera transparencia en sus procesos de toma de decisiones utilizando estos modelos. Esto permite que investigadores y profesionales puedan tomar decisiones más informadas y, de ser el caso, cumplir con distintos estándares de transparencia necesarios en ciertos escenarios, como el médico o el de seguridad [1, 8].

Por estos motivos, en este trabajo se planea desarrollar una librería llamada XSVM_ctx que implementa la arquitectura propuesta en [2] para desarrollar un modelo SVM contextualizado, que provee explicaciones por contexto. Al lograr obtener una justificación de las clasificaciones realizadas por los modelos SVM, se espera ampliar la cantidad de escenarios donde estos modelos pueden ser usados, además de aumentar la confianza de los resultados obtenidos por los SVM. Además, la librería de código abierto permitirá que un grupo más amplio

de investigadores y entusiastas de la inteligencia artificial exploren formas de brindar transparencia en sistemas inteligentes.

1.4 Objetivos

Este trabajo tiene un objetivo general y tres objetivos específicos los cuales se indican a continuación:

1.4.1 Objetivo General

Desarrollar una librería de código abierto que implemente las funcionalidades de SVMs contextualizados para facilitar el acceso a explicaciones de las clasificaciones realizadas por estos modelos.

1.4.2 Objetivos Específicos

1. Desarrollar una librería computacionalmente eficiente que explique los resultados obtenidos en modelos SVM contextualizados.
2. Implementar métodos eficientes para clasificar con modelos SVM en diversos contextos.
3. Desarrollar una interfaz que permita a investigadores y profesionales utilizar la librería XSVM_ctx de forma intuitiva.

1.5 Marco Teórico

Los modelos SVM son considerados modelos de caja negra porque poseen procesos internos demasiado complejos como para intentar entenderlos [1]. Se requiere de procesos externos para dar sentido a las clasificaciones de estos modelos. Varios autores han propuesto diferentes técnicas para dar una explicación a las clasificaciones obtenidas por los modelos SVM. Entre estas técnicas se encuentran la extracción de reglas [1, 6], la construcción de vecindarios sintéticos [7], y la contextualización de los modelos [2, 3, 4].

La primera técnica se enfoca en la extracción de reglas mediante árboles de decisión [1, 6]. En este caso el modelo SVM, después de ser entrenado, es utilizado para la clasificación de un conjunto de datos, después el árbol de decisión es entrenado con el conjunto de datos

clasificados por el SVM [1, 6]. Al entrenar con los datos clasificados por el SVM, el árbol extrae las reglas que este modelo SVM definió para cada categoría y esto es utilizado para explicar el comportamiento del SVM. El árbol de decisión es utilizado por ser considerado fácil de entender y permite interpretar con facilidad la relación entre las variables de entrada y salida [1].

La segunda técnica consiste en que, cada vez que se hace una clasificación con un modelo de caja negra, se utiliza un programa secundario que toma el objeto que se da como entrada al modelo y, mediante un algoritmo, crea numerosas nuevas instancias a partir de hacer cambios menores menores al objeto de entrada original, estas nuevas instancias se conocen como vecindario sintético [7]. Una vez que el vecindario sintético está terminado, estas instancias pasan por el modelo de caja negra y con los resultados se crea un árbol de decisión. El objetivo de esta práctica es que el árbol resultante sirva como una forma de explicar qué está detrás de la clasificación, y también qué cambios debería sufrir la instancia original para provocar una clasificación distinta [7]. Este proceso implica una complejidad computacional alta debido a la necesidad de generar múltiples instancias nuevas y evaluarlas en el modelo de caja negra. A pesar de esta complejidad, los resultados son exhaustivos y facilitan la comprensión del funcionamiento de cualquier modelo de caja negra con el que se lo aplique [7].

La contextualización de modelos es una técnica que consiste en definir y utilizar contextos durante la etapa donde el modelo aprende patrones para clasificar [2, 3, 4]. Durante el proceso de clasificación del SVM se identifican qué aspectos del objeto evaluado son los más relevantes para la clasificación [3]. Esto puede ayudar en la interpretabilidad de modelos de caja negra ya que, con la información que provee, se puede identificar el contexto para poder explicar la razón de la clasificación del objeto por el modelo SVM.

En [2] se propone un clasificador SVM explicable que hace uso de múltiples modelos SVM contextualizados para mejorar la interpretabilidad de los resultados. Una vez se ha contextualizado el conjunto de datos este se divide y, de forma paralela, se entrena varios SVMs,

cada uno con los datos de cada contexto. El usuario, antes de ingresar la instancia para llevar a cabo la clasificación, debe seleccionar el contexto que se desea utilizar. Esto permite dar la explicación del por qué el modelo llega a la clasificación a la que llega con ejemplos pertenecientes al mismo contexto. Además, se ha demostrado que la contextualización puede llevar a una mejora en la precisión del modelo [2, 4].

1.6 Propuesta de Solución

Basándonos en las técnicas revisadas en el marco teórico, proponemos una implementación de software para mejorar la interpretabilidad de los modelos SVM. Mostrado de forma modular:

1. Funcionalidad Principal
2. Transparencia e Interpretabilidad
3. Evaluación y Medición de Desempeño
4. Interfaz de Programador (API, por sus siglas en inglés)

Capítulo 2

2.1 Metodología

Para desarrollar la metodología, nos reunimos con el cliente, con quien discutimos varios acercamientos para el desarrollo de la solución. Durante la reunión se discutieron los distintos requerimientos que se deseaban en el resultado final.

El cliente indicó varias directrices clave, una de ellas fue que la librería iba a ser de código abierto, permitiendo que cualquier interesado pueda utilizarla, modificarla y mejorarla sin restricciones. También pidió que acompañemos la librería con una buena documentación detallando su funcionamiento, permitiendo así que sea más sencillo integrarla en trabajos futuros.

El cliente, durante las reuniones, nos aconsejó utilizar la librería [5], esta provee herramientas para explicar clasificaciones realizadas por modelos SVM, la cual es una funcionalidad clave para el desarrollo del proyecto.

Otra funcionalidad que el cliente indicó que se esperaba del proyecto era la inclusión de interfaces, en concreto, una interfaz de programación de aplicaciones (API) que permita a los usuarios utilizar la librería con llamadas a esta, haciendo que el manejo sea más sencillo. Se llegó a discutir la inclusión de una interfaz gráfica, la cual podría hacer el trabajo con la librería más accesible, pero la idea se descartó pues añadiría un nivel de complejidad adicional al proyecto al tener que incluir aspectos como la experiencia de usuario, comprometiendo el tiempo de desarrollo y pruebas. Sin embargo, la API debería proveer las herramientas suficientes para poder integrar una interfaz gráfica en el futuro.

2.2 Usuarios

La librería propuesta se diseñó principalmente para tres tipos de usuarios: investigadores, profesionales y tomadores de decisiones. A continuación, se detalla el perfil y uso esperado de cada grupo:

- Investigadores: los investigadores que trabajan en campos relacionados con la inteligencia artificial, el aprendizaje automático y la minería de datos se espera que utilicen la librería como una herramienta entender mejor los resultados obtenidos mediante las SVMs. Esto les permite profundizar en sus estudios, validar hipótesis y mejorar la interpretabilidad de sus modelos.
- Profesionales: profesionales de la industria que aplican técnicas de IA en sus actividades. Esto incluye, pero no se limita a, analistas de datos, ingenieros de aprendizaje automático y consultores técnicos. Se espera que utilicen la librería tanto en contextos de aprendizaje como en proyectos personales y profesionales. Aunque la librería no está específicamente orientada al público general, se espera que profesionales y académicos la adopten debido a su capacidad para mejorar la interpretabilidad y transparencia de los modelos SVM.
- Tomadores de Decisiones: individuos que se benefician de modelos de IA entrenados como apoyo en procesos de toma de decisiones en vez de llevar a cabo todo el proceso de entrenamiento. Similar al perfil de los profesionales, se espera que utilicen herramientas que se han implementado en este proyecto en el desarrollo de sus actividades personales y profesionales.

2.3 Requerimientos

Como resultado de las conversaciones con el cliente, trazamos los requerimientos en cuatro secciones: funcionalidades básicas, requerimientos que aseguran la implementación de la arquitectura de modelos SVM contextualizados dentro de la librería; transparencia e interpretabilidad, requerimientos para asegurar la implementación de mecanismos de explicación de clasificaciones hechas por modelos SVM contextualizados; evaluación y medición de desempeño, requisitos de rendimiento para asegurar que el software sea eficiente y de calidad; interfaz de programación, requisitos que facilitan la interacción entre los usuarios y la librería.

2.3.1 Funcionalidades Básicas

- Proceso de Aprendizaje Paralelo: Dado un conjunto de datos contextualizados, el proceso de aprendizaje debe producir un conjunto de modelos contextualizados a los que se puedan acceder de forma paralela. Además de las etiquetas que identifican las diferentes categorías, esta funcionalidad debe permitir la especificación del campo (o campos) en el conjunto de datos que identifican los diferentes contextos. El proceso de aprendizaje debe realizarse de forma paralela teniendo en consideración los contextos especificados.
- Proceso de Evaluación Paralelo: Dado uno o varios contextos y un conjunto de datos, este proceso debe usar los modelos de conocimiento asociados a dichos contextos para evaluar hasta qué punto los datos pertenecen a cada una de las categorías descubiertas durante el proceso de aprendizaje. El proceso de evaluación debe realizarse de forma paralela considerando los contextos (primarios) requeridos. En adición al contexto primario, los resultados deben contextualizarse de acuerdo a los aspectos relevantes de los datos encontrados durante el proceso de evaluación.
- Proceso Predictivo: Dado uno o varios contextos y un conjunto de datos, este proceso debe usar los modelos de conocimiento asociados a dichos contextos para predecir las mejores categorías para los datos en dicho conjunto. El proceso predictivo debe realizarse de forma paralela considerando los contextos requeridos.

2.3.2 Transparencia e Interpretabilidad

- Interpretabilidad de Resultados: Implementar mecanismos para proporcionar explicaciones comprensibles sobre las decisiones de los modelos SVM al momento de clasificar imágenes o datos tabulados. Estos mecanismos deben dar a los usuarios una muestra de cómo se tomaron las decisiones en base a los datos proporcionados. Además, se debe incluir la capacidad de mostrar las características más relevantes que influyeron en cada decisión del modelo. Las explicaciones deben generarse y presentarse de manera

accesible, permitiendo a los usuarios explorar y comprender fácilmente los procesos de toma de decisión detrás de las clasificaciones.

2.3.3 Evaluación y Medición de Desempeño

- Comparación del Tiempo de Procesamiento: Dado uno o varios contextos, la librería debe separar el conjunto de datos de entrenamiento en varios subconjuntos, uno por cada contexto. Una vez realizada esta división, se lleva a cabo el entrenamiento con cada subconjunto en paralelo. Esta optimización está orientada a reducir el tiempo total del entrenamiento y de evaluación sin comprometer la precisión y calidad de las clasificaciones. Se planea entrenar un modelo contextualizado y uno sin contextualizar con el mismo conjunto de datos. Se espera que, al tomar el tiempo de entrenamiento y de evaluación de ambos modelos, el tiempo del modelo contextualizado sea menor que el tiempo del modelo sin contextualizar.
- Comparación de Métricas de Clasificación: Al momento de realizar el entrenamiento del modelo, la librería debe entrenar un SVM por cada contexto. Esta contextualización del modelo debería resultar en una mejora en la clasificación de datos. Cabe resaltar que esta mejora depende de la diferencia entre contextos [15]. Se espera que, al comparar diversas métricas de clasificación, como la exactitud y precisión, del modelo contextualizado con un modelo SVM sin contextualizar, las métricas de clasificación del modelo contextualizado sean mejores que las del modelo sin contextualizar, esto en casos donde los contextos son diferentes.
- Nivel de Detalle de las Explicaciones: Proporcionar explicaciones con un nivel de detalle adecuado para que los usuarios puedan entender y confiar en los resultados del modelo. Dichas explicaciones han de incluir información relevante sobre las características del conjunto de datos que influyeron en las decisiones del modelo. Las explicaciones deben

incluir la relación de los resultados con el contexto. Se espera tener dos formas de explicar resultados por cada tipo de entrada permitido: imágenes y datos tabulados.

2.3.4 Interfaz de Programación

- Interfaz Intuitiva: Diseñar una interfaz de programación (API) que facilite a investigadores y profesionales el uso de la librería y la visualización de los resultados. La interfaz debe brindar facilidades al ingresar datos y también debe mostrar los resultados de las predicciones con su respectiva explicación. El código de la interfaz debe ser fácil de entender de cara a futuros usuarios de la librería. La API debe estar bien documentada, cada función debe tener una breve explicación de lo que hace, acompañada de una descripción de sus entradas y salidas.

2.4 Alcance de la Solución

El alcance preliminar para este proyecto a nivel macro, está dado por los siguientes puntos donde la solución debe ser una librería capaz de:

Aceptar conjuntos de datos de imágenes o datos tabulados preprocesados que servirán como valores de entrada para el modelo. Los conjuntos de datos ingresados deben ser divisibles por contexto. La librería se enfocaría en la partición del conjunto de datos en múltiples subconjuntos, cada subconjunto perteneciendo a un contexto.

La librería debe constar de un proceso de entrenamiento, proceso por el cual el modelo aprende a partir de un conjunto de datos, contextualizado y paralelizado. Un entrenamiento contextualizado consiste en entrenar un modelo por cada contexto, y como se necesita que estos entrenamientos se lleven a cabo a la vez, se los ejecuta en paralelo.

Con el modelo entrenado, el usuario puede definir un contexto para la categorización de un objeto. Si definió un contexto, se realiza la clasificación con el modelo SVM entrando en ese contexto. En el caso en el que el usuario no defina un contexto, se realizará la clasificación del objeto en paralelo con todos los modelos SVM, considerando todos los contextos disponibles.

El resultado obtenido debe estar acompañado de una explicación que incluye la información del contexto bajo el cual fue realizada la clasificación.

2.5 Herramientas

Como lenguaje de programación se escogió Python, debido a que, a la fecha, es muy usado en el área de la inteligencia artificial, además de que existen varias librerías que facilitan el preprocesamiento de datos y el entrenamiento de modelos de IA.

Python tiene un *Global Interpreter Lock* (GIL), lo que evita ejecutar varios procesos en paralelo, esta es una medida que se ha tomado para evitar que varios procesos accedan a la misma variable al mismo tiempo, lo que puede ocacionar errores impredecibles [13]. Afortunadamente, existen librerías como *Joblib* que logran sobrepasar esta limitante y permiten ejecutar varios procesos en paralelo en Python [14]. A partir de esta funcionalidad se planeó incluir esta librería al proyecto.

Otra librería que se decidió usar en este proyecto es *xsvmc-lib*, la cual permite generar explicaciones de los resultados obtenidos en las categorizaciones de clases realizadas por SVMs [5].

Además, también se decidió incluir librerías que apoyen en el preprocesamiento de los datos de entrenamiento y en la generación de gráficos, las cuales son necesarios para implementar las explicaciones.

2.6 Limitaciones y Beneficios

La implementación de la solución propuesta presenta ciertas limitaciones, pero también conlleva importantes beneficios. A continuación, se detallan estos aspectos clave:

2.6.1 Limitaciones

- La explicación de los resultados estará limitada a datos tabulados e imágenes
- La explicación de los resultados solo estará disponible en inglés
- El modelo es de categorización, no de regresión.

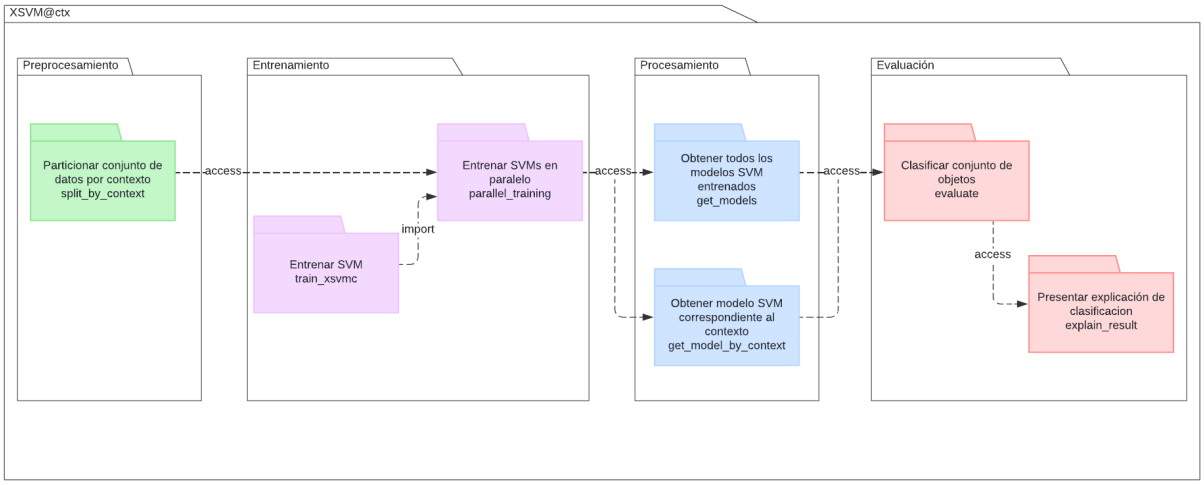
2.6.2 Beneficios

- Proveerá un mayor entendimiento de los procesos de clasificación de los modelos SVM.
- El modelo generado será más eficiente computacionalmente que otros modelos SVM, esto se debe a que requerirá menos tiempo de entrenamiento y menos tiempo de evaluación.

2.7 Diseño de la Librería

Figura 1

Diagrama de paquetes - Arquitectura de la librería

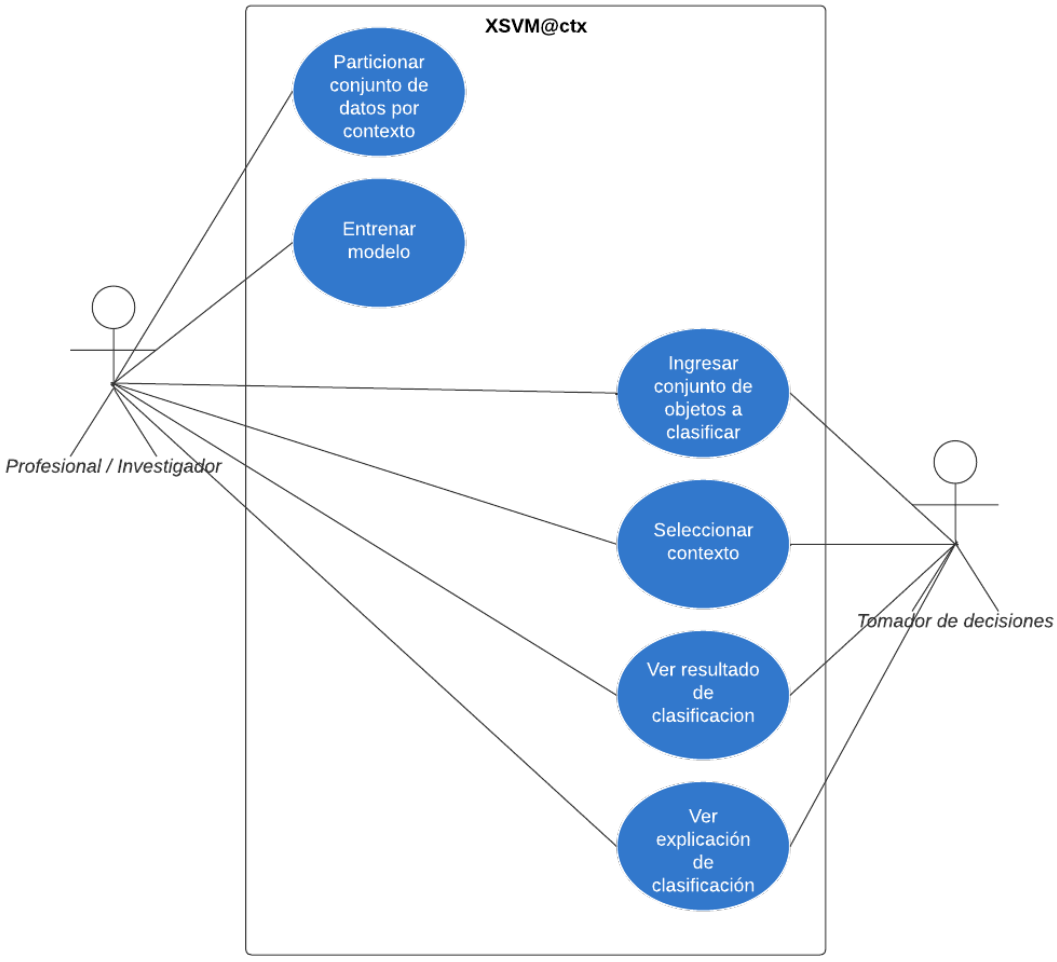


Para explicar la arquitectura del proyecto, se diseñó la Figura 1, donde se separan las funcionalidades en cuatro grupos, la primera es *Preprocesamiento* y contiene el proceso de dividir un conjunto de datos por contexto. Luego se pasa al *Entrenamiento*, donde los datos preprocesados son usados para entrenar SVMs, se utiliza una librería que permite entrenar cada SVM individualmente, pero los entrenamientos se llevan a cabo al mismo tiempo en paralelo. La siguiente sección se llama *Procesamiento*, esta se compone de dos procesos que funcionan de forma independiente, pero dependen de las secciones anteriores. Uno retorna todos los modelos SVM del sistema, el otro retorna solo el SVM correspondiente a un contexto dado. Finalmente, en la etapa de *Evaluación*, se clasifican los dados usando los modelos SVM obtenidos en la

sección anterior, retorna el resultado de la clasificación, y explica los resultados obtenidos, estas tareas se llevan a cabo por sus procesos correspondientes.

Figura 2

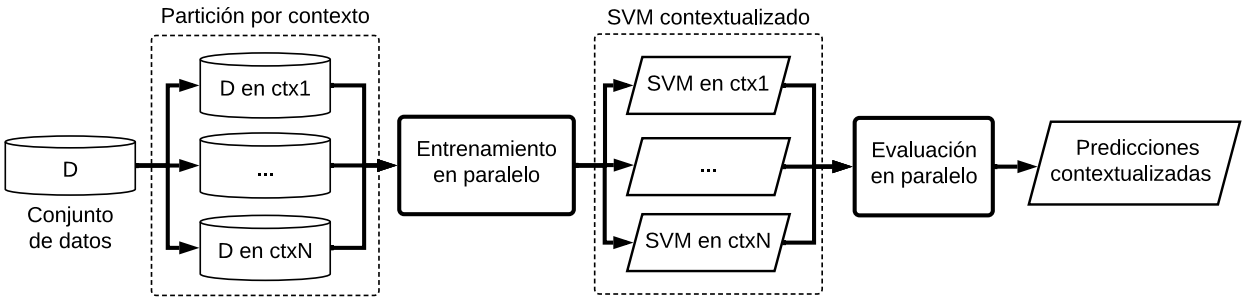
Diagrama de caso de uso – Interacción con la librería



Con el fin de explicar cómo cada usuario interactúa con cada elemento de la arquitectura, se muestra la Figura 2. Tanto el usuario *profesional* como el *investigador* realizan los mismos procesos. Deben ofrecer un conjunto de datos, particionarlo y usarlo para entrenar el modelo. Con el modelo entrenado el *profesional*, el *investigador* y el *tomador de decisiones* pueden utilizar el resto de las funcionalidades de la librería: ingresar un conjunto de objetos a clasificar, seleccionar hasta un contexto, y al finalizar la clasificación ver el resultado y su explicación.

Figura 3

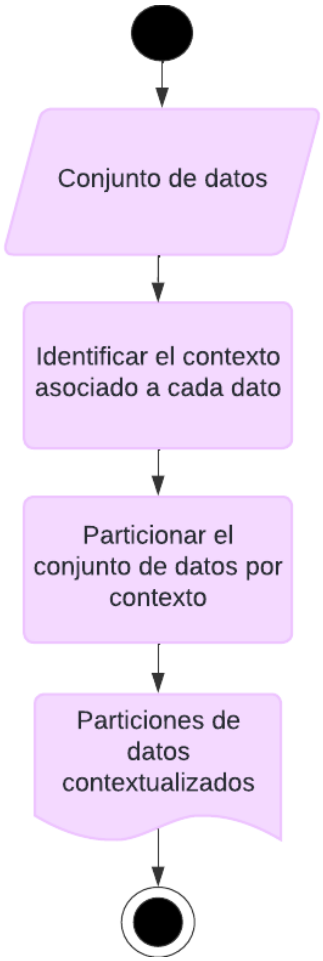
Diagrama de actividades – Funcionamiento del modelo propuesto en [2]



La estructura base del modelo clasificador que se va a utilizar fue propuesta en [2]. Como se indica en la Figura 3, se empieza particionando un conjunto de datos donde cada partición pertenece a un contexto específico. Estas particiones se usan para entrenar en paralelo varios modelos SVM contextualizados. Una vez que este proceso ha terminado, el modelo está listo para su uso, un usuario puede ingresar un conjunto de datos para clasificar, escoger hasta uno de los contextos y mediante un proceso de evaluación en paralelo se obtendrá como resultado una clasificación contextualizada.

Figura 4

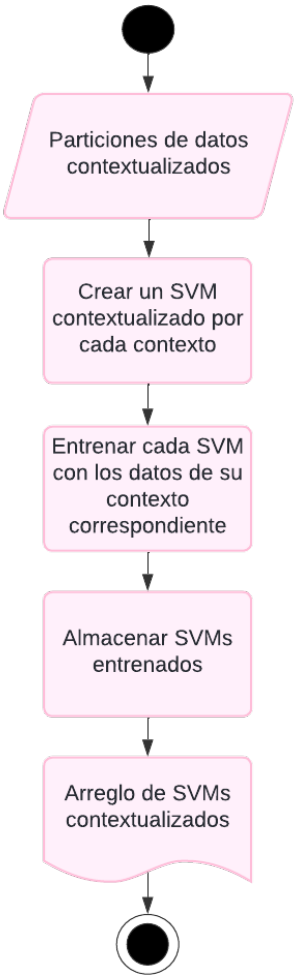
Diagrama de flujo - Proceso de contextualización de datos



Para entender mejor el funcionamiento del modelo, se lo dividió en varios procesos. El primero de estos es el *proceso de contextualización de datos*, que se detalla en la Figura 4. El usuario ingresa un conjunto de datos donde cada dato debe tener un contexto asociado. La librería identifica los contextos de cada dato y agrupa a aquellos que compartan contextos en particiones. Finalmente, se crea un arreglo de las particiones de datos.

Figura 5

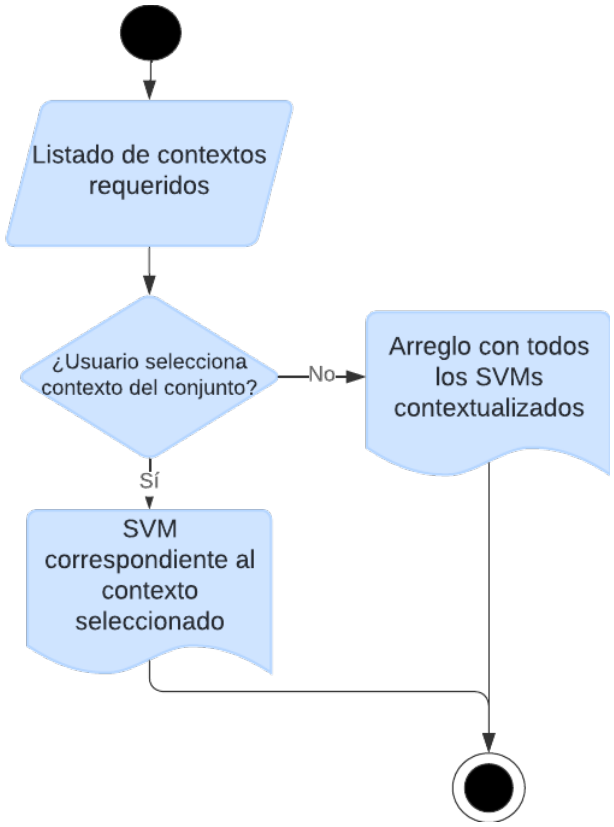
Diagrama de flujo - Proceso de aprendizaje (entrenamiento)



El siguiente proceso es el de *aprendizaje*, el cual se detalla en la Figura 5. Durante este proceso se entrena el modelo SVM de clasificación. El proceso recibe el arreglo de particiones de datos contextualizados del proceso anterior. Por cada partición, se crea un modelo SVM, cada uno de estos SVMs tiene como contexto el contexto de la particón correspondiente, y es con esos datos que se lleva a cabo el *entrenamiento*. Los modelos son entrenados en paralelo y, una vez concluido el entrenamiento, se almacenan los SVMs en un arreglo.

Figura 6

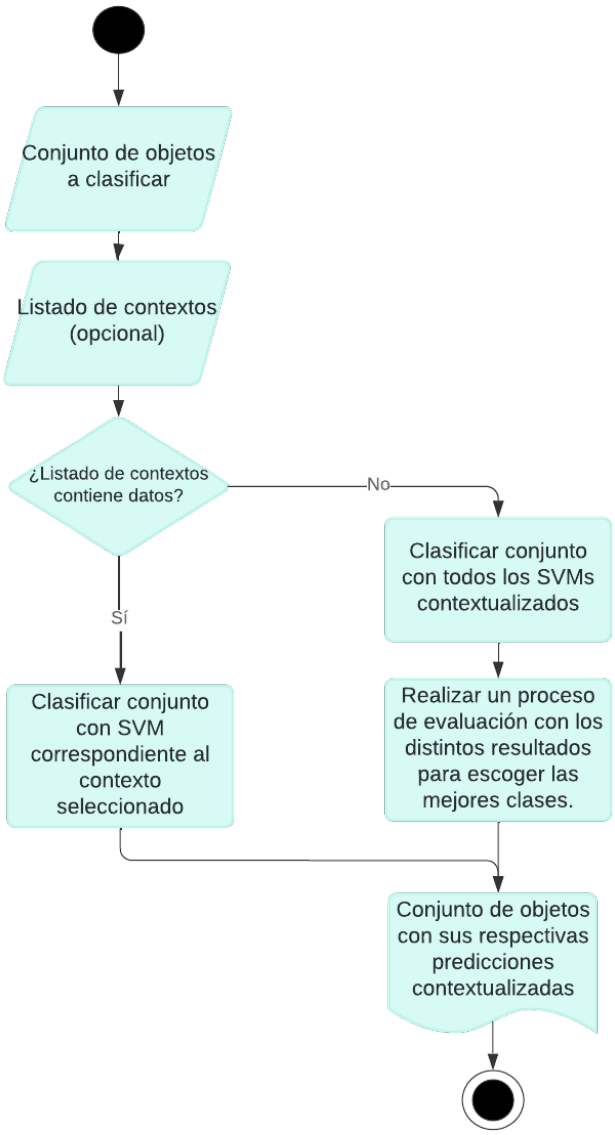
Diagrama de flujo - Proceso de selección de contexto



Antes de que se lleve a cabo una clasificación con los modelos entrenados, el usuario tiene la opción de seleccionar un contexto. La Figura 6 muestra el proceso de *selección de contexto*. Si el usuario escoge un contexto, entonces se retorna el modelo SVM correspondiente a dicho contexto. En el caso que no se especificque el contexto se retorna el arreglo con todos los modelos SVM.

Figura 7

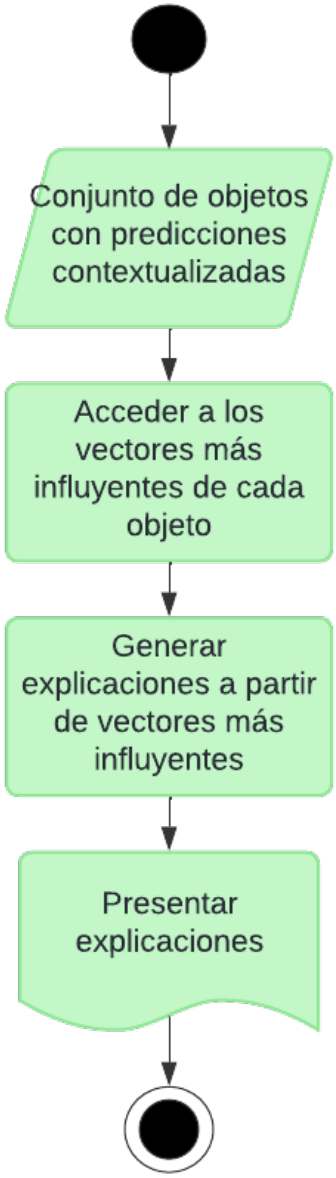
Diagrama de flujo - Proceso de clasificación



La Figura 7 muestra el proceso de *clasificación*. Empieza recibiendo por parte del usuario el conjunto de objetos a clasificar y el contexto seleccionado. Si en el proceso anterior se seleccionó un contexto, la clasificación se lleva a cabo con el SVM correspondiente a dicho contexto. Caso contrario todos los SVM realizan el proceso de clasificación de forma independiente sobre el conjunto de objetos y, mediante un *proceso de evaluación*, se escogen las mejores clases para la clasificación de cada objeto del conjunto de datos. El proceso termina con todos los objetos del conjunto clasificados.

Figura 8

Diagrama de flujo - Proceso de explicación



El último proceso, el *proceso de explicación*, se detalla en la Figura 8. Con el conjunto de objetos con sus clasificaciones contextualizadas se accede a los vectores con características más similares al objeto clasificado, a estos vectores se los conoce como los vectores más influyentes (MISV) [5]. Se utilizan los vectores más influyentes para generar una explicación del proceso de toma de decisiones que lleva a cada objeto a ser clasificado de determinada manera. Al final las clasificaciones junto a sus explicaciones son presentadas al usuario.

2.8 Prototipo

Como prototipo de la solución se desarrolló el pseudocódigo mostrado en el Apéndice A. Se tomó como ejemplo un caso de clasificación de imágenes para mostrar algunas funciones que se planea incluir en la librería.

La primera función fue *load_images_and_labels*, recibiría el directorio donde residen las imágenes y un archivo donde se encuentra la información de cada imagen. Con estos datos la función retorna tres valores: las imágenes, su etiqueta y su contexto.

La siguiente función, *train_test_split_with_context*, permitiría separar las imágenes, etiquetas y contextos en un conjunto de entrenamiento y otro de prueba, esta partición tomaría en consideración el contexto de las imágenes.

La función *group_by_context* agruparía los datos por contexto y crearía un diccionario donde la llave es el contexto y el valor corresponde a los de datos pertenecientes a ese contexto.

Para inicializar el modelo se debe crear una instancia de la clase con *contextualized_xSVMC()*. Con el modelo inicializado se lo puede entrenar con el método *fit*, encargado en entrenar el modelo. Este método recibiría el diccionario de particiones y se encargaría de generar un SVM por cada contexto, y de entrenar cada SVM en el contexto correspondiente.

Para la evaluación se planeó utilizar *predict_with_context*, este método recibe el conjunto de datos a ser evaluados. Los datos son evaluados y clasificados por el SVM respectivo a su contexto. Este método retorna las predicciones de los datos.

Por último, se planeó incluir funciones que proveen distintas formas de explicaciones a partir de los resultados de las clasificaciones. Estas explicaciones podrían ser mostradas al usuario.

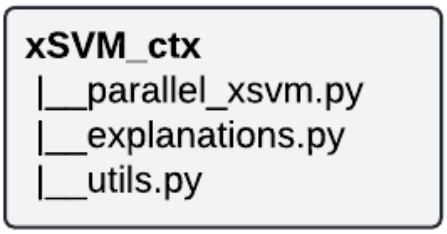
Capítulo 3

3.1 Resultados y Análisis

La librería *xSVM_ctx*, cuyo nombre proviene de la arquitectura propuesta en [2] que se utilizó como base para desarrollar el modelo SVM contextualizado, está formada por tres módulos: *explanations*, *parallel_xsvm*, y *utils*, como se muestra en la Figura 9. El módulo *parallel_xsvm* contiene el modelo SVM contextualizado llamado *contextualized_xSVMC*; el módulo *explanations* contiene las funciones que muestran las explicaciones detrás de las clasificaciones realizadas por los modelos entrenados con esta librería; y, por último, el módulo *utils* incluye varias funciones que facilitan el uso del modelo.

Figura 9

Estructura de la librería



Para determinar si la librería cumple con los requerimientos del proyecto, se llevó a cabo un proceso de validación donde se comparó los resultados obtenidos por el modelo *contextualized_xSVMC* con el modelo *SVC* de Scikit-Learn [16] en diferentes conjuntos de datos. Los conjuntos de datos que se utilizaron fueron los siguientes:

- Dataset_1: Movimientos de las manos a partir de electromiografías (EMG) [17]. Este conjunto de datos tabulados consiste en lecturas de señales EMG de manos. Se utilizaron 1,512,750 datos del conjunto de datos, donde el 50% de los datos fueron utilizados para el entrenamiento, y el resto de los datos para la evaluación.
- Dataset_2: Cancelaciones de reservas en hoteles [18]. Este conjunto de datos tabulados contiene información sobre reservas realizadas en hoteles. Se utilizaron 119,386 datos del

conjunto de datos, donde el 70% de los datos fueron utilizados para el entrenamiento y el 30% restantes para la evaluación.

- Dataset_3: Afecciones de la piel a partir de imágenes [19]. Se trata de un conjunto de imágenes de distintas afecciones de la piel, cada una etiquetada con su respectiva condición. Se utilizaron 675 imágenes del conjunto de datos donde el 80% de los datos fueron utilizados para el entrenamiento, y el resto de los datos para la evaluación.

Para la comparación de los modelos en cada uno de estos conjuntos de datos, se utilizó los siguientes indicadores de rendimiento:

- Tiempos de ejecución: tiempo que tardan los modelos en entrenar y evaluar los datos.
- Métricas de clasificación: precisión, exactitud, sensibilidad y el valor F1.

3.1.1 Tiempos del Ejecución

Tabla 1

Tiempos de ejecución de los modelos SVC y contextualized_xSVMC en los diferentes conjuntos de datos

Conjunto de datos	Tiempo de entrenamiento		Tiempo de evaluación	
	SVC	contextualized_xSVMC	SVC	contextualized_xSVMC
Dataset_1	1h 22m 53s	0h 0m 18s	0h 20m 9s	0h 3m 50s
Dataset_2	0h 6m 16s	0h 1m 50s	0h 1m 9s	0h 0m 28s
Dataset_3	0h 0m 42s	0h 0m 17s	0h 1m 13s	0h 0m 3s

En la Tabla 1 se puede apreciar cómo la paralelización en *contextualized_xSVMC* genera una reducción significativa en los tiempos de ejecución en los tres conjuntos de datos. La mayor diferencia la podemos encontrar cuando se utilizó el *Dataset_1*, donde hay una reducción de más de una hora en el tiempo de entrenamiento y de más de 15 minutos en el tiempo de evaluación. Esta gran diferencia puede ser atribuida a que se utilizaron 18 hilos para llevar a cabo el

entrenamiento. Esto significa que el proceso de entrenamiento se dividió en 18 tareas que se ejecutaban simultáneamente. Esto fue posible porque el *Dataset_1* pudo dividirse en 36 contextos.

El *Dataset_2* pudo dividirse en dos contextos, por lo que se utilizó dos hilos durante la etapa de entrenamiento y de evaluación. Por otro lado, el *Dataset_3* pudo dividirse en seis contextos, permitiendo al modelo *contextualized_xSVMC* utilizar seis hilos durante el entrenamiento y evaluación de los datos. Aunque no se utilizaron tantos hilos comparados con la prueba realizada con el *Dataset_1*, aún se puede observar una mejora considerable en los tiempos de ejecución utilizando el *Dataset_2* y el *Dataset_3*.

3.1.2 Métricas de Clasificación

Al comparar dos modelos, no basta solamente con contrastar la velocidad de sus funciones. También es importante acompañar el análisis de tiempo de ejecución con distintas métricas de clasificación para, de esta forma, determinar cuál de los dos presenta mejores resultados en sus clasificaciones. Así, se puede conocer si el modelo propuesto es igual o mejor que su contraparte, o si se está sacrificando la calidad del modelo a favor de mayor velocidad.

Como se indicó anteriormente, se utilizaron las siguientes cuatro métricas para el análisis: la *exactitud* es el promedio de predicciones correctas [21]; la *precisión* es el promedio de las veces que se predijo correctamente cada clase [21]; la *sensibilidad* es el promedio de veces que cada clase fue predicha correctamente [21]; el *valor F1* es la combinación de la *precisión* y la *sensibilidad* para tener una métrica más balanceada [21].

3.1.2.1 Dataset_1

Tabla 2

Métricas de clasificación de los modelos en el Dataset_1

Métrica de Clasificación	SVC	contextualized_xSVMC
Exactitud	88,9%	97,9%

Precisión	88,9%	97,9%
Sensibilidad	88,9%	97,9%
Valor F1	88,9%	97,9%

Al comparar el modelo *SVC* con el modelo *contextualized_xSVMC*, como muestra la Tabla 2, se observa una mejora del 9% en todas las métricas evaluadas. Esto indica que, con el conjunto de datos evaluado, el modelo contextualizado tiene un mejor rendimiento que el *SVC*.

3.1.2.2 Dataset_2

Tabla 3

Métricas de clasificación de los modelos en el Dataset_2

Métrica de Clasificación	SVC	contextualized_xSVMC
Exactitud	79,6%	79,3%
Precisión	79,4%	79,3%
Sensibilidad	79,6%	79,3%
Valor F1	79,4%	79,3%

En la Tabla 3 se observa que el modelo paralelizado obtiene resultados similares al lineal. Esta similitud en los resultados podría atribuirse a que no existe mucha diferencia entre los dos contextos [15], por lo que, en vez de aprovechar la contextualización para generar mejores resultados, el modelo paralelizado simplemente se compone por dos modelos similares que han sido entrenados con la mitad de los datos que el modelo *SVC*.

Figura 10

Matriz de confusión de SVC en el Dataset_2

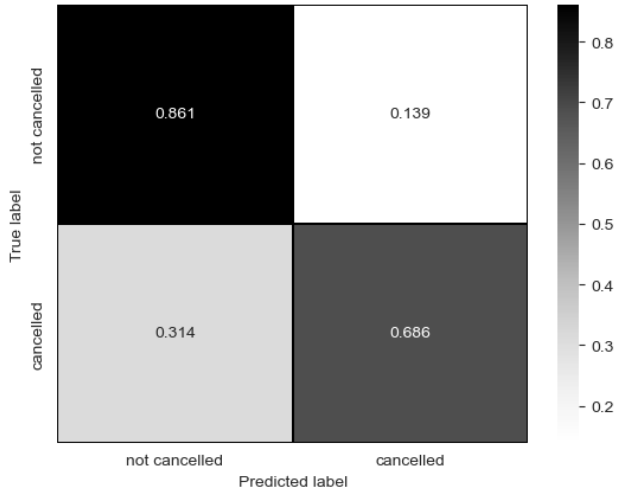
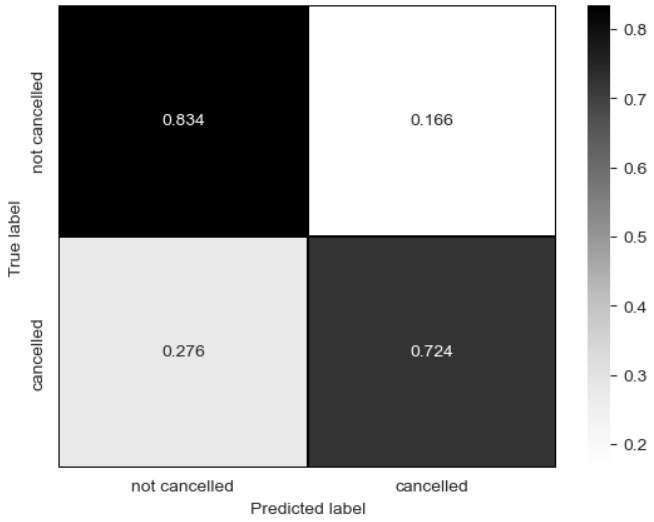


Figura 11

Matriz de confusión de contextualized_xSVMC en el Dataset_2



En la Figura 10 se ve la matriz de confusión de las predicciones del modelo *SVC*. Se observa que tiene un 86% de acierto en los casos cuando la clasificación era "no cancelada" y en un 69% cuando la clasificación era "cancelada". Por el contrario, en la Figura 11, que muestra los resultados del modelo *contextualized_xSVMC*, se aprecia que, aunque hay una disminución cercana al 3% de los casos para las reservas "no canceladas", hay un aumento del 3.8% para las

"canceladas". Aunque las predicciones suelen ser menos precisas, existe un mayor equilibrio en los resultados, disminuyendo en cierta medida el sesgo que el modelo *SVC* tenía con la categoría “no cancelada”.

3.1.2.3 Dataset_3

Tabla 4

Métricas de clasificación de los modelos en el Dataset_3

Métrica de Clasificación	SVC	contextualized_xSVMC
Exactitud	75,6%	85,2%
Precisión	75,8%	85,2%
Sensibilidad	75,6%	85,2%
Valor F1	74,5%	84,6%

Al analizar las métricas de rendimiento presentes en la Tabla 4, se puede notar una mejora de aproximadamente el 10% en todas las métricas evaluadas. Esto sugiere que, en este caso, la contextualización del modelo mejora la precisión de sus predicciones.

3.2 Explicación

Uno de los objetivos principales del proyecto era la explicación de los resultados de modelos SVM. La librería cumple con ese objetivo al dar la opción de acompañar los resultados de las clasificaciones con explicaciones. A continuación, se detallan los tipos de explicación desarrollados para imágenes y para datos tabulados.

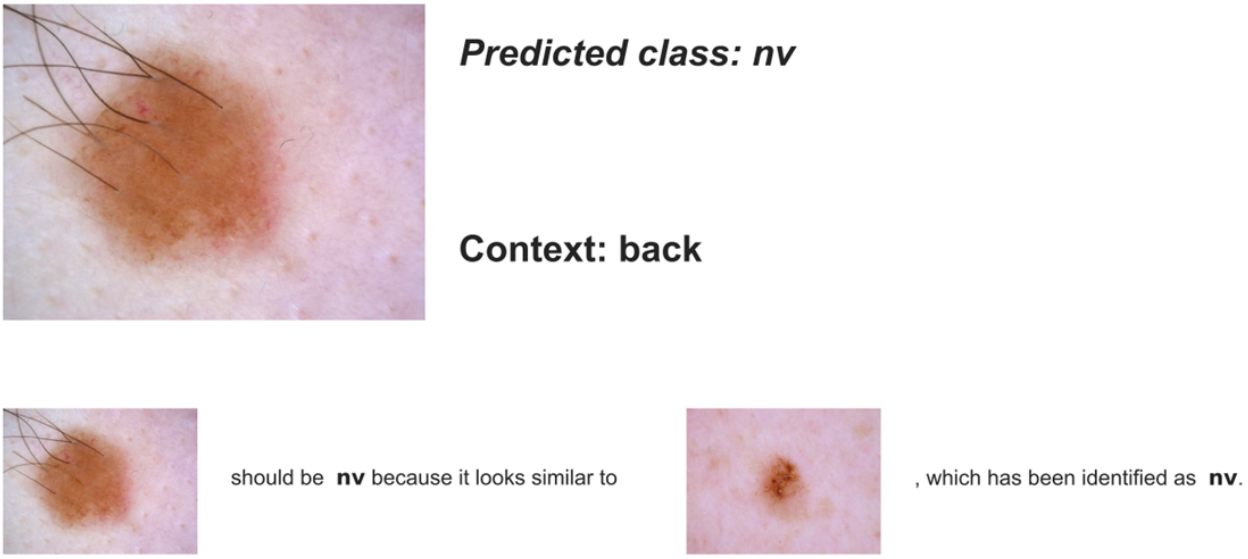
3.2.1 Explicación para Imágenes

Para las clasificaciones basadas en imágenes, la librería proporciona varias formas de explicación utilizando las imágenes más similares a la imagen evaluada en el conjunto de entrenamiento.

3.2.1.1 Explicación por Referencia Visual

Figura 12

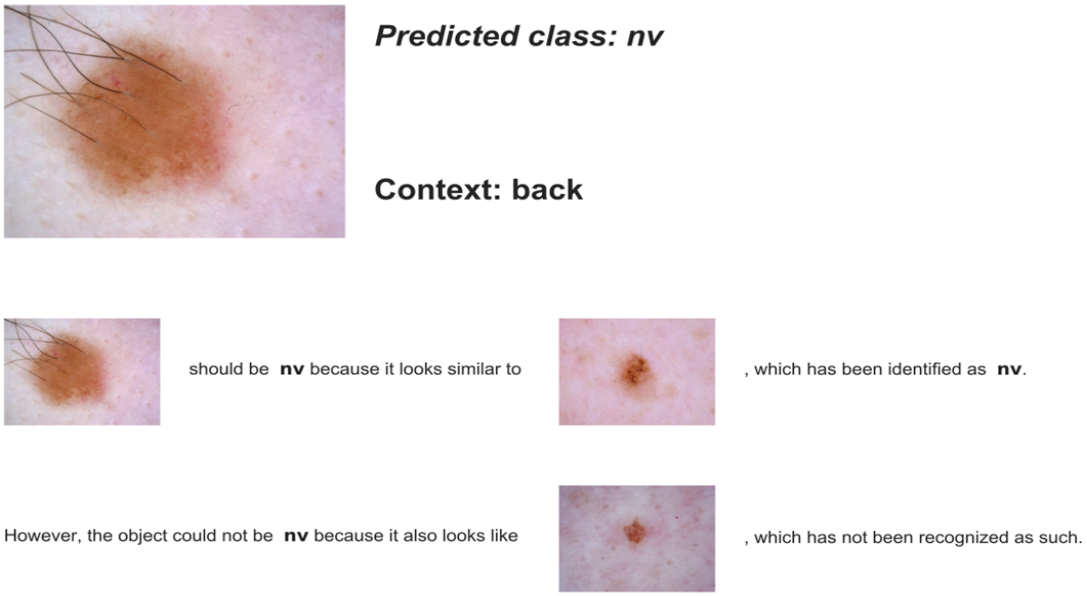
Explicación de predicciones por referencia visual



Como se puede apreciar en la Figura 12, esta explicación consiste en mostrar la imagen clasificada en el contexto bajo el cual se hizo el análisis, y la imagen del conjunto de entrenamiento con mayor similitud a la imagen clasificada. Esto permite entender que la imagen fue clasificada como miembro de una determinada clase, en este caso de ejemplo podemos ver que el modelo llegó a la clasificación nv, debido a que se parece a esta otra imagen, que también es *nv*.

Figura 13

Explicación de predicciones por referencia visual con contraejemplo

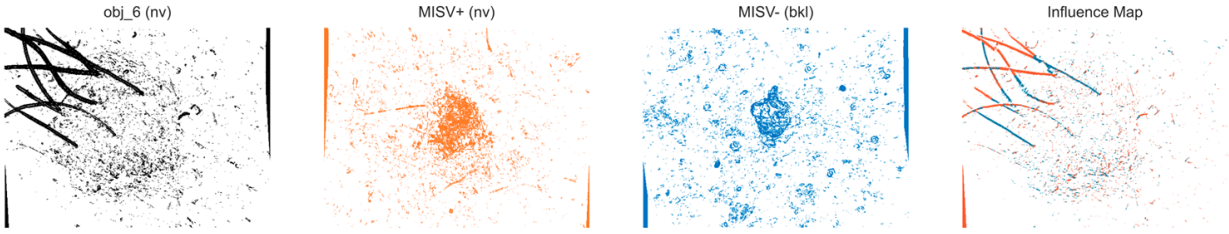


La librería permite también acompañar la explicación con contraejemplos, como se ve en la Figura 13. Un contraejemplo en este caso se refiere a la imagen de los datos de entrenamiento con mayor similitud a la imagen evaluada, pero que pertenece a una clase distinta. Por ejemplo, en el caso de la Figura 13, el contraejemplo sería la imagen que más se parece a la evaluada que no corresponde a la clase *nv*. Utilizar un contraejemplo en la explicación es opcional y permite conocer otra clase a la que podría pertenecer el objeto.

3.2.1.2 Explicación por Mapas de Influencias

Figura 14

Explicación de predicciones por mapa de influencia con contraejemplo



Otra forma que la librería ofrece para dar explicaciones de datos ingresados como imágenes es mediante un *mapa de influencias*. Similar al método anterior, la librería otorga como explicación la imagen del conjunto de entrenamiento con mayor similitud a la imagen clasificada como ejemplo de objeto similar. En el caso de los *mapas de influencias* estos pueden proporcionar las áreas que más influyeron y que provocaron que el modelo la clasifique de cierta forma. Esto es especialmente útil para saber si el modelo utilizado está tomando en consideración características que no son relevantes desde el punto de vista del usuario al momento de realizar las clasificaciones.

En la Figura 14 se aprecia como los vellos presentes en la imagen proporcionada juegan un rol importante al momento de clasificarla de una forma u otra, queda a criterio del investigador, profesional o tomador de decisión si ese es el comportamiento deseado del modelo o si se requiere de ajustes que permita ignorar esta información y así mejorar las clasificaciones. Esta forma de explicar resultados también puede estar acompañada de un contraejemplo. Esto permite no solo saber a qué otras clases podría pertenecer el objeto clasificado, sino que también permite conocer, gracias al mapa de influencias, donde se encuentran aquellas características de la imagen que lleva a que sea clasificada de una forma o de otra.

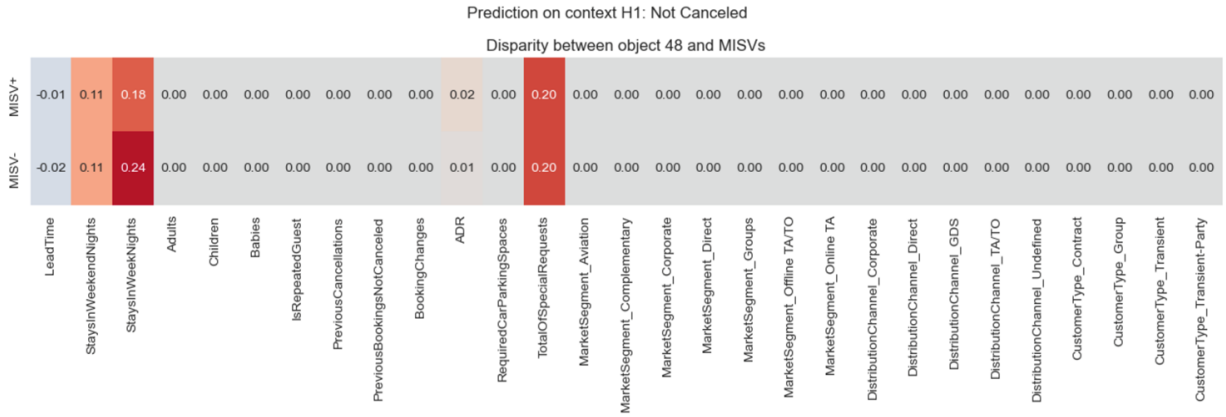
3.2.2 Clasificación y Explicación por Datos Tabulados

Para clasificaciones basadas en datos tabulados, la librería muestra la relación del objeto clasificado con el objeto más similar del conjunto de entrenamiento de distintas formas.

3.2.2.1 Explicación por Mapas de Calor

Figura 15

Explicación de predicciones por mapa de calor con contraejemplo

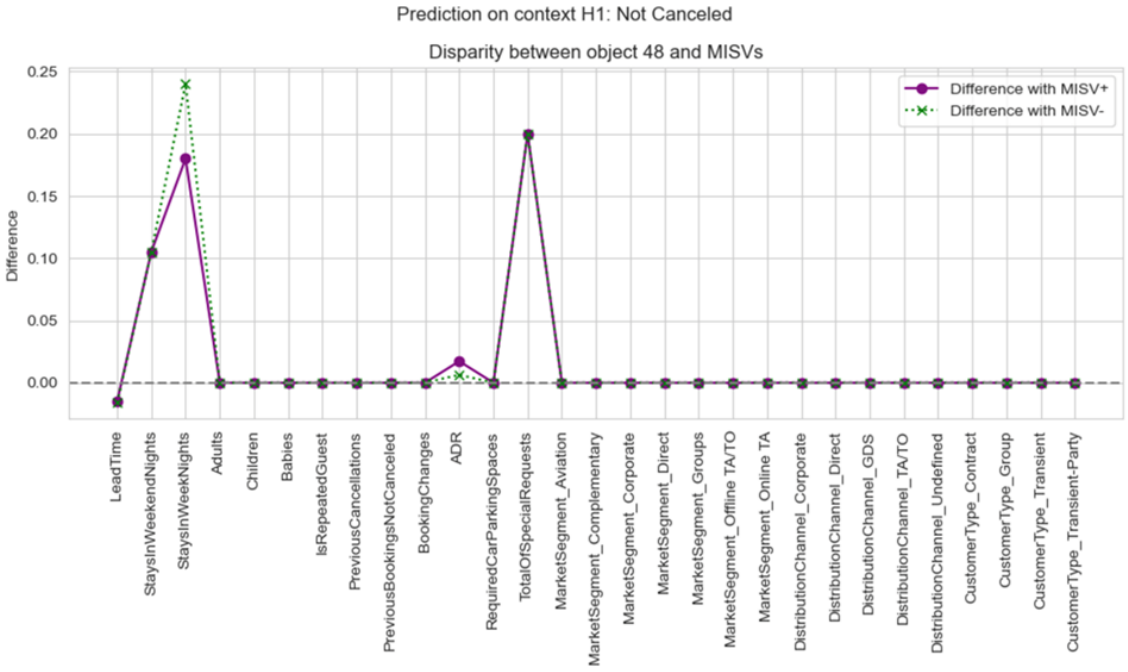


Las explicaciones por *mapa de calor* muestran las diferencias entre el objeto analizado, el objeto de los datos de entrenamiento con mayor similitud al objeto evaluado, y, en caso de solicitarlo, el objeto de los datos de entrenamiento perteneciente a otra clase con mayor similitud al objeto evaluado. Esta vez, como se muestra en la Figura 15, las diferencias están dadas por los valores de sus atributos, y se acentúa con color aquellas características que presentan mayores diferencias. Este enfoque permite identificar rápidamente las áreas donde las características del objeto evaluado coinciden o difieren, facilitando una comprensión de cómo estos atributos influyen en la clasificación.

3.2.2.2 Explicación por Gráficos de Líneas

Figura 16

Explicación de predicciones por gráfico de líneas con contraejemplo

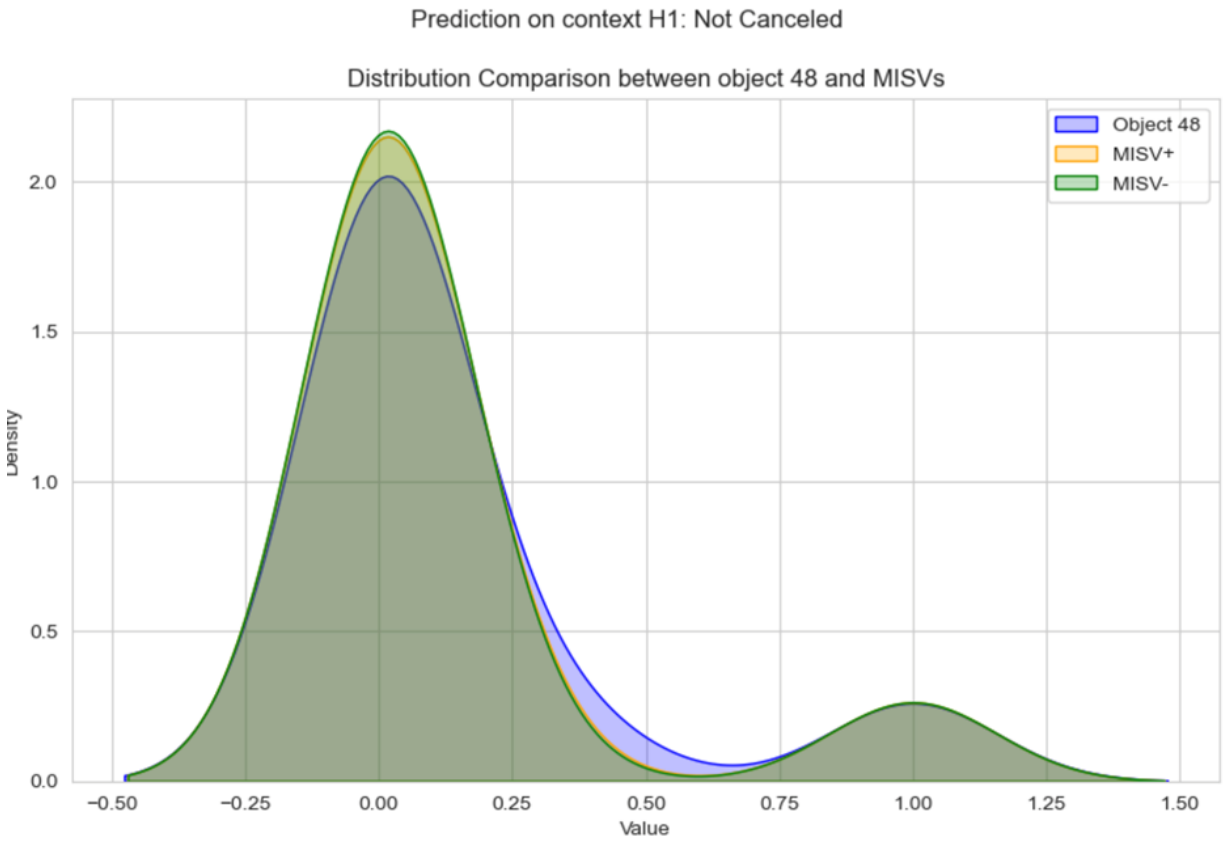


Otra forma de representar la similitud entre el objeto que se está evaluando y aquellos pertenecientes al conjunto de entrenamiento con los que comparte mayor similitud es utilizando un *gráfico de líneas*, como en la Figura 16. En el ejemplo se puede apreciar cómo las diferencias entre el objeto evaluado, el ejemplo y contraejemplo han sido graficadas en vez de ser mostradas como valores. Este acercamiento permite comparar los objetos y conocer, de una forma más visual, qué características influyen en que el objeto analizado sea clasificado de un modo y no de otro.

3.2.2.3 Explicación por Estimación de Densidad de Kernel

Figura 17

Explicación de predicciones por estimación de densidad de kernel con contraejemplo



Una forma de representar la similitud entre el objeto que se está evaluando y los objetos del conjunto de entrenamiento con mayor similitud a este, tanto de la misma clase como otras, es mediante un gráfico de *estimación de densidad de kernel* (KDE por sus siglas en inglés) como el de la Figura 17. Este tipo de gráfico permite visualizar la densidad de las características y cómo estas varían entre los dos objetos, facilitando así la comprensión de las similitudes y diferencias que llevan a una determinada clasificación.

Capítulo 4

4.1 Conclusiones y Recomendaciones

En este proyecto se desarrolló la librería xSVM_ctx, una librería de código abierto que permite explicar resultados de las clasificaciones realizadas con modelos SVM. La librería cuenta con su propio modelo SVM contextualizado, el cual fue propuesto en [2]. Esta implementación acepta tanto datos en imagen como datos tabulados para realizar las clasificaciones y explicaciones, e incluye distintas formas de presentar dichas explicaciones. La librería, acompañada por su documentación, se puede encontrar en [20].

4.1.1 Conclusiones

Tras llevar a cabo las fases planificadas en este proyecto, se han obtenido las siguientes conclusiones clave:

- La librería desarrollada permite explicar de manera efectiva las clasificaciones realizadas por modelos SVM, lo que mejora significativamente la transparencia y la comprensión de estos modelos. Este objetivo general fue alcanzado al integrar datos contextualizados en el proceso de explicación, lo que proporciona un mejor y más comprensible análisis.
- La eficiencia computacional de la librería fue lograda a través del desarrollo de algoritmos paralelizados para el entrenamiento y uso de modelos SVM contextualizados. Este objetivo específico asegura que la librería no solo sea funcional, sino también capaz de lograr tiempos de procesamiento menores, lo que la hace apta para ser utilizada en escenarios con grandes cantidades de datos o con recursos limitados.
- La implementación de funciones que facilitan la explicación de las clasificaciones realizadas por el modelo en diversos contextos amplía el alcance de la librería, aumentando los dominios donde los SVM pueden ser utilizados. Esto también fortalece el uso práctico de la librería en investigaciones y aplicaciones profesionales.

- La interfaz desarrollada facilita el uso de la librería *xSVM_ctx* y su integración en diversos proyectos que requieran de desarrollar modelos SVM explicables por parte de investigadores, profesionales y entusiastas de la IA.

4.1.2 Recomendaciones

Tras culminar lo planificado en la propuesta, se obtienen las siguientes recomendaciones:

- Explorar la posibilidad de implementar la arquitectura a otros modelos de clasificación adicional a los SVM, para desarrollar librerías similares.
- Expandir la librería para que pueda manejar otros tipos de datos más allá de las imágenes y datos tabulados, como el lenguaje natural.
- Se sugiere investigar la adaptabilidad de la librería en contextos multilingües o culturales diversos, asegurando que las explicaciones sean comprensibles y relevantes en distintos idiomas y culturas.
- Dar a conocer esta librería a estudiantes de grado de diversas disciplinas para que exploren los beneficios que los modelos de IA contextualizados y explicables pueden proveer.

Referencias

[1] A. Chatchinarat, K. W. Wong, y C. C. Fung, “Rule extraction from electroencephalogram signals using support vector machine”, en 2017 9th International Conference on Knowledge and Smart Technology (KST), 2017, pp. 106–110.

[2] M. Loor, A. Tapia-Rosero, y G. De Tré, “Contextual Boosting to Explainable SVM Classification”, en Fuzzy Logic and Technology, and Aggregation Operators, 2023, pp. 480–491.

[3] M. Loor y G. De Tré, “Explaining Computer Predictions with Augmented Appraisal Degrees”, en Proceedings of the 11th Conference of the European Society for Fuzzy Logic and Technology (EUSFLAT 2019), 2019, pp. 158–165

[4] D. Malowany y H. Guterman, “Biologically Inspired Visual System Architecture for Object Recognition in Autonomous Systems”, Algorithms, vol. 13, no. 7, 2020.

[5] M. Loor, A. Tapia-Rosero, y G. De Tré, “An Open-Source Software Library for Explainable Support Vector Machine Classification”, en 2022 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), 2022, pp. 1–7.

[6] Y. Zhang, P. Wang, K. Liang, Y. He, y S. Ma, “An Alarm and Fault Association Rule Extraction Method for Power Equipment Based on Explainable Decision Tree”, en 2021 11th International Conference on Power and Energy Systems (ICPES), 2021, pp. 442–446.

[7] R. Guidotti, A. Monreale, F. Giannotti, D. Pedreschi, S. Ruggieri, y F. Turini, “Factual and Counterfactual Explanations for Black Box Decision Making”, IEEE Intelligent Systems, vol. 34, no. 6, pp. 14–23, 2019.

[8] UNESCO. (2024). Recommendation on the ethics of artificial intelligence [Online]. Disponible en: https://unesdoc.unesco.org/ark:/48223/pf0000381137.locale=en

[9] K. Witkowski, R. Okhai, y S. R. Neely, “Public perceptions of artificial intelligence in healthcare: ethical concerns and opportunities for patient-centered care”, BMC Medical Ethics, vol. 25, no. 1, p. 74, Jun. 2024.

[10] D. Banerjee, V. Kukreja, S. Hariharan, y V. Jain, “Enhancing Mango Fruit Disease Severity Assessment with CNN and SVM-Based Classification”, en 2023 IEEE 8th International Conference for Convergence in Technology (I2CT), 2023, pp. 1–6.

[11] D. Keerthana, V. Venugopal, M. K. Nath, y M. Mishra, “Hybrid convolutional neural networks with SVM classifier for classification of skin cancer”, Biomedical Engineering Advances, vol. 5, p. 100069, 2023.

[12] Y. Wang y Z. Wang, “Stock Price Volatility Prediction in Financial Big Data on XGBoost and ARIMA Models”, en 2023 International Conference on Ambient Intelligence, Knowledge Informatics and Industrial Electronics (AIKIIE), 2023, pp. 1–5.

[13] Python Software Foundation. (2024). Initialization, Finalization & Threads [Online]. Disponible en: https://docs.python.org/3/c-api/init.html

[14] Joblib Development Team. (2021). Joblib: running Python functions as pipeline jobs [Online]. Disponible en: https://joblib.readthedocs.io/en/latest/

[15] M. Loor y G. De Tré, “Automatic Context Selection in Explainable Support Vector Machine Classification” en 2024 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), Yokohama, Japan, 2024, pp. 1-8, doi: 10.1109/FUZZ-IEEE60900.2024.10611823.

[16] Scikit-learn developers. (2024). SVC [Online]. Disponible en: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

[17] S. Lobov, N. Krilova, I. Kastalskiy, V. Kazantsev y V. Makarov. (2019). EMG data for gestures data set - UCI machine learning repository [Online]. Disponible en: https://archive.ics.uci.edu/dataset/481/emg+data+for+gestures

[18] N. Antonio, A. de Almeida y L. Nunes, “Hotel booking demand datasets,” *Data in Brief*, vol. 22, pp. 41–49, 2019. doi: 10.1016/j.dib.2018.11.126

[19] P. Tschandl. (2018). The HAM10000 dataset, a large collection of multi- source dermatoscopic images of common pigmented skin lesions [Online]. Disponible en: https://doi.org/10.7910/DVN/DBW86T

[20] J. Pisco-Jordán y L. Ramos-Pozo. (2024). xsvm_ctx-lib [Online]. Disponible en: https://github.com/interpretapple-lab/xsvm_ctx-lib

[21] Scikit-learn developers. (2024). Metrics and scoring: quantifying the quality of predictions [Online]. Disponible en: https://scikit-learn.org/stable/modules/model_evaluation.html

Apéndice A

Pseudocódigo de la Solución

```
# Importar las funciones de la librería
from xsvm_ctx import *

directory = './path/to/images/'
metadata_file = './path/to/metadata.csv'

# Cargar imágenes, etiquetas y contexto
images, labels, context = load_images_and_labels(directory, metadata_file)

# Dividir datos de entrenamiento y de prueba
X_train, y_train, context_train, X_test, y_test, context_test = train_test_split_with_context(images, labels, context)

# Agrupar datos  por contexto
grouped_data_train = group_by_context(X_train, y_train, context_train)
grouped_data_test = group_by_context(X_test, y_test, context_test)

# Creación del modelo
model = contextualized_xSVMC()

# Entrenamiento en paralelo
model.train(grouped_data_train)

# Evaluación en paralelo
evaluations = model.predict_with_context(grouped_data_test)

# Mostrar resultados de las evaluaciones
for evaluation in evaluations:
    print(
        evaluation.predicted_class_name,
        explanation_with_image(evaluation),
        explanation_with_influence(evaluation)
    )

```