

**UNIVERSIDAD PERUANA DE CIENCIAS APLICADAS**

FACULTAD DE INGENIERIA

CARRERA DE INGENIERÍA DE SISTEMAS EPE

INTELIGENCIA ARTIFICIAL

“**Desarrollo de inteligencia artificial para el reconocimiento de comportamientos sospechosos en cámaras de seguridad del distrito de Chancay**”

**AUTOR(ES)**

Reyna Melendez, Marco Antonio (U20231B132)

Fuentes Flores, Diego Sebastian (U202423074)

Rojas Camones, Guillermo Martin (U202319685)

Taipe Reyes, Anderson Juan (U20241C750)

**ASESOR(ES)**

Alarcon Delgado, Carlos Adrian

**Lima – Perú**

**2025**

# Índice

[Índice ii](#_Toc211266235)

[Objetivo del Estudiante (Student Outcome) 1](#_Toc211266236)

[Fuentes Flores, Diego Sebastian (U202423074) 1](#_Toc211266237)

[Sustento de cumplimiento de los principios (DEI) 1](#_Toc211266238)

[1. Introducción 2](#_Toc211266239)

[1.1. Descripción del problema 2](#_Toc211266240)

[1.2. Problema 2](#_Toc211266241)

[1.3. Importancia 2](#_Toc211266242)

[1.4. Técnicas y justificación 3](#_Toc211266243)

[1.5. Objetivo 6](#_Toc211266244)

[1.6. Artículos científicos de referencia 6](#_Toc211266245)

[1.7. Propuesta de aplicación de algoritmos predictivos 7](#_Toc211266246)

[2. Estado del Arte 11](#_Toc211266247)

[2.1. Preguntas de investigación 11](#_Toc211266248)

[Respuesta a las preguntas de investigación a través del análisis de papers 11](#_Toc211266249)

[**"Performance Comparison of Convolutional Neural Networks (CNN) and Support Vector Machine (SVM) Algorithms in Human Face Classification”** 11](#_Toc211266250)

[**"Comparison of deep learning and traditional machine learning techniques for classification of Pap smear images”** 12](#_Toc211266251)

[**"Anomaly Detection in Video Surveillance Using YOLOv8”** 13](#_Toc211266252)

[3. Implementación 13](#_Toc211266253)

[3.1 - Implementación de algoritmo supervisado 14](#_Toc211266254)

[3.1.1. Resumen del Dataset 17](#_Toc211266255)

[3.1.2. Creación del Dataset 18](#_Toc211266256)

[3.1.3. Entrenamiento y Validación de Modelos de Aprendizaje Supervisado 31](#_Toc211266257)

[3.1.3.1 Random Forest 31](#_Toc211266258)

[3.1.3.2 Regresión Logística 34](#_Toc211266259)

[3.1.3.3 KNN (K-nearest-neighbors) 36](#_Toc211266260)

[3.1.3.4 SVM (Support Vector Machine) 39](#_Toc211266261)

[3.1.3.5 XGBoost 42](#_Toc211266262)

[3.1.3.6 CatBoost 45](#_Toc211266263)

[3.1.3.7 LightGBM 48](#_Toc211266264)

[3.1.4. Conclusiones del aprendizaje supervisado 51](#_Toc211266265)

[3.2 - Implementación de algoritmo no supervisado 52](#_Toc211266266)

[3.2.1. Isolation Forest 53](#_Toc211266267)

[3.2.2. DBSCAN (Density-Based Spatial Clustering of Applications with Noise) 59](#_Toc211266268)

[3.2.3. Conclusiones del algoritmo no supervisado 63](#_Toc211266269)

[3.3. Caso de estudio de algoritmo de aprendizaje profundo 64](#_Toc211266270)

[3.3.1. Metodología y Preparación de Datos 64](#_Toc211266271)

[3.3.2. Dataset y Pre-Procesamiento 65](#_Toc211266272)

[3.3.3. Modelos de Deep Learning Evaluados 66](#_Toc211266273)

[3.3.4. Entrenamiento y Evaluación del Modelo 67](#_Toc211266274)

[3.3.5. Resultados y Comparativa 69](#_Toc211266275)

[3.3.6. Conclusiones del Caso de Estudio 71](#_Toc211266276)

# Objetivo del Estudiante (Student Outcome)

## Fuentes Flores, Diego Sebastian (U202423074)

Durante este curso pude entender y aplicar diferentes tipos de modelos de machine learning para la solución de problemas reales, tratando de comprender las diferencias entre el aprendizaje supervisado y no supervisado, así como sus ventajas y limitaciones en distintos contextos. También, fortalecí mis habilidades en el uso de Python para la exploración, limpieza, transformación de datos, así como la implementación de modelos mediante la conocida librería scikit-learn. Por ende, también fortalecí mi capacidad para comunicar resultados de manera efectiva, elaborando informes y visualizaciones interactivas que facilitan la toma de decisiones estratégicas basadas en datos y algoritmos. Gracias al profesor por incentivar la lectura de papers para aplicar las técnicas dichas ahí para poder emplearlas en este trabajo.

# Sustento de cumplimiento de los principios (DEI)

Durante el desarrollo de este proyecto se procuró cumplir con los principios de Diversidad, Equidad e Inclusión (DEI) en cada etapa del proceso. Se utilizaron las detecciones de una base datos de videovigilancia y con la ayuda del modelo YOLO que se usó para extraer características de personas y algunos objetos, procurando siempre respetar la privacidad y el anonimato de los individuos presentes en el dataset. Además, se buscó que el modelo no incurriera en sesgos hacia determinados comportamientos o contextos, promoviendo una detección equitativa de anomalías. Se priorizó la transparencia en cada etapa del proceso, desde la exploración de datos hasta la validación de resultados, con el objetivo de construir una herramienta que pueda apoyar la seguridad sin comprometer los principios de diversidad, equidad e inclusión. La inclusión también se reflejó en el planteamiento de estrategias preventivas, diseñadas para ser accesibles y aplicables en contextos variados. Este compromiso con los principios de DEI subraya la relevancia de la ética en el uso de la inteligencia artificial, reforzando el objetivo de utilizar la tecnología como un medio para construir sociedades más justas y equitativas.

# 1. Introducción

### 1.1. Descripción del problema

La seguridad ciudadana es una de las principales preocupaciones de los vecinos de Chancay. A pesar de que el distrito cuenta con cámaras de videovigilancia, la falta de automatización en el análisis de las imágenes limita su efectividad, lo que impide una detección temprana de eventos como robos, peleas, merodeo sospechoso o actos vandálicos y retrasa la respuesta de las autoridades.

Para demostrar esta problemática, existen indicadores claros. Según datos del Sistema Informático de Registro de Denuncias Policiales (SIDPOL), en Chancay se registraron 215 denuncias por delitos contra el patrimonio en 2020, cifra que aumentó a 464 en 2023, representando más del doble en solo tres años. Además, entre 2020 y 2023, se reportaron 39 homicidios en el distrito.

Este incremento en la criminalidad coincide con el desarrollo del megapuerto de Chancay, un proyecto de gran envergadura que ha atraído a diversas organizaciones, incluyendo mafias que se disputan el control de la zona, lo que ha derivado en un aumento de la violencia y la inseguridad

### 1.2. Problema

¿De qué manera distintos algoritmos de inteligencia artificial, como CNN y otros enfoques comparativos, pueden predecir comportamientos normales y anormales en las cámaras de seguridad del distrito de Chancay, contribuyendo a mejorar la eficacia de la respuesta de las autoridades locales?

### 1.3. Importancia

Estudiar este problema es fundamental porque la aplicación de distintos enfoques de aprendizaje supervisado, no supervisado y profundo servirán para el análisis de características de imágenes en tiempo real permitirá evaluar cuál de ellos ofrece mejores resultados en la detección de patrones sospechosos. Este proceso contribuirá a mejorar la eficiencia operativa de los sistemas de videovigilancia, reducir la dependencia del ojo humano y potenciar el uso de los recursos tecnológicos existentes.

Desde el punto de vista económico, la inversión en estas tecnologías es crucial. Según informes del Fondo Monetario Internacional y del Banco Interamericano de Desarrollo, la delincuencia en América Latina y el Caribe representa un costo del 3,4% del PIB, afectando negativamente la inversión, la productividad y el crecimiento económico.

Además, en el contexto del desarrollo industrial y la urbanización impulsados por el megapuerto, la seguridad ciudadana es un componente clave para una gestión integral y sostenible del distrito. Por ello, la comparación de algoritmos de machine learning aplicados a la videovigilancia en Chancay es un paso necesario para enfrentar los desafíos actuales de seguridad, proteger a la población y garantizar un desarrollo sostenible.

### 1.4. Técnicas y justificación

La solución técnica se basa en la aplicación de Computer Vision y Machine Learning/Deep Learning para predecir, a partir de características de frames de cámaras de videovigilancia, si una escena corresponde a un comportamiento normal o anómalo. El objetivo es evaluar y comparar diferentes enfoques de aprendizaje supervisado, no supervisado y profundo, con el fin de identificar cuál ofrece un mejor rendimiento en la detección temprana de comportamientos sospechosos.

1. **YOLO (You Look Only One)**

Como primer paso para la extracción de características, se utilizará el modelo YOLOv8 (You Only Look Once) es una arquitectura de Deep Learning ampliamente reconocida por su capacidad de detección de objetos en tiempo real. Otros modelos iguales dividen el proceso en etapas separadas de generación de regiones y clasificación, lo que hace YOLO es abordar la detección como un único problema de regresión lo que le permite alcanzar altas velocidades de procesamiento sin comprometer significativamente la precisión

Técnicamente, lo que hace YOLO es divide la imagen en una cuadricula y para cada celda, predice simultáneamente las coordenadas de los cuadros delimitadores (bounding boxes), las confianzas a dichas predicciones y las probabilidades de clase correspondientes a los objetos presentes. Este enfoque de detección end-to-end permite procesar múltiples objetos en una sola pasada de la red neuronal, optimizando el tiempo de ejecución.

En ente trabajo, se usará YOLO para pasar por todos los frames extraídos del dataset CamNuvem con el propósito de identificar objetos relevantes como personas, vehículos o comportamientos atípicos. Las características resultantes de las detecciones (coordenadas, tamaños, frecuencia de personas, entre otros) serán luego puestas dentro de un dataframe, el cual servirá para el entrenamiento, validación y comparación de los diferentes modelos de Machine Learning y Deep Learning.

1. **Modelos de Machine Learning Tradicional**

Los algoritmos tradicionales han demostrado ser efectivos en tareas de clasificación binaria cuando se cuenta con un conjunto de características preprocesadas. Diversos estudios comparativos evidencian que, aunque los modelos de Deep Learning como las Convolutional Neural Networks (CNN) suelen superar en precisión a los algoritmos clásicos, los modelos de Machine Learning Tradicional ofrecen ventajas en términos de rapidez y menor consumo de recursos computacionales, lo que los hace relevantes en sistemas de videovigilancia con restricciones de hardware o necesidad de respuestas inmediatas (Royan et al., 2025).

1. **Modelos de Machine Learning Tradicional (no-supervisados)**

En este trabajo se usarán también distintos modelos de aprendizaje no supervisado, el cual nos va a ayudar en identificar comportamientos o patrones anómalos en las características de frames de cámaras de videovigilancia esto sin la necesidad de contar con etiquetas previas que clasifiquen los datos como normales o anormales. Estos métodos permiten descubrir irregularidades en los flujos de video a partir de desviaciones estadísticas o estructurales, siendo una herramienta clave en entornos donde los eventos inusuales son escasos o difíciles de etiquetar manualmente.

1. **Modelos de Deep Learning (CNN)**

Por otro lado, los métodos de Convolutional Neural Networks (CNN) son capaces de extraer automáticamente características espaciales directamente de los píxeles, eliminando la dependencia de la ingeniería manual de atributos. En la clasificación de imágenes médicas, se ha demostrado que CNN supera ampliamente a algoritmos tradicionales en métricas de precisión y recall, como señalan los hallazgos de la comparación entre ML clásico y Deep Learning en imágenes de citología (Pap smear).

1. **Justificación**

Este trabajo se justifica con la necesidad de evaluar y comparar distintos enfoques de aprendizaje automático aplicados a la videovigilancia inteligente, con el fin de poder identificar comportamientos anómalos o sospechosos correctamente.

En primer lugar, se busca validar si los modelos tradicionales, como SVM, pueden servir como modelo comparativo inicial para la detección de anomalías, considerando que tiene menor demanda computacional y rapidez en entornos con recursos limitados. En segundo lugar, se propone comparar su rendimiento con arquitecturas de Deep Learning, como las CNN, reconocidas por su mayor capacidad para identificar patrones complejos y generalizar en contextos dinámicos

De esta manera, el proyecto se alinea con investigaciones recientes que evidencian que, aunque las CNN suelen superar a los algoritmos clásicos en métricas de precisión y recall, los enfoques tradicionales continúan siendo útiles como referencia comparativa y como alternativa en sistemas con restricciones de hardware o conectividad (Royan et al., 2025).

### 1.5. Objetivo

Desarrollar y comparar diferentes algoritmos de machine learning aplicados a la detección de comportamientos sospechosos en videovigilancia urbana, con el fin de determinar el modelo más adecuado para el problema en el distrito de Chancay.

### 1.6. Artículos científicos de referencia

Cada integrante del grupo seleccionó un artículo científico de relevancia. En total se consideran cuatro:

1. **Royan, M. J., Suresha, M., Arun, M., & Prabhu, A. V. (2025).** *Performance Comparison of Convolutional Neural Networks (CNN) and Support Vector Machine (SVM) Algorithms in Human Face Classification.*
2. **Jahan, S., Islam, M. R., Paul, B. K., et al. (2020).** *Comparison of Deep Learning and Traditional Machine Learning Techniques for Classification of Pap Smear Images.*
3. Praveen, M. N., & Sandeep, D. (2025). Anomaly Detection in Video Surveillance Using YOLOv8. International Journal for Research in Applied Science & Engineering Technology (IJRASET), 13(5).

* **Performance Comparison of Convolutional Neural Networks (CNN) and Support Vector Machine (SVM) Algorithms in Human Face Classification**

Este artículo realiza una comparación entre técnicas de machine learning tradicional y deep learning en el ámbito de la clasificación de imágenes faciales. Los autores implementan un modelo de SVM como línea base y lo contrastan con una CNN, encontrando que la CNN logra una precisión del 96.12%, mientras que el SVM alcanza un 76.53%. El estudio concluye que las CNN superan en exactitud y capacidad de generalización a los métodos tradicionales, especialmente en problemas visuales complejos.

Referencia: Royan, M. J., Suresha, M., Arun, M., & Prabhu, A. V. (2025). Performance Comparison of Convolutional Neural Networks (CNN) and Support Vector Machine (SVM) Algorithms in Human Face Classification. International Journal of Advanced Computer Science and Applications (IJACSA), 16(1). <https://doi.org/10.70609/g-tech.v9i3.7384>.

* **Comparison of Deep Learning and Traditional Machine Learning Techniques for Classification of Pap Smear Images**

Este trabajo presenta un análisis comparativo entre modelos de aprendizaje profundo (CNN) y algoritmos tradicionales de machine learning como Random Forest, K-Nearest Neighbors y SVM, aplicados a la clasificación de imágenes de citología cervical (Pap smear). Los resultados demuestran que, aunque los modelos tradicionales pueden alcanzar desempeños aceptables, las CNN ofrecen una precisión significativamente mayor y una mejor capacidad de detección en imágenes con alta variabilidad.

**Referencia:** Jahan, S., Islam, M. R., Paul, B. K., et al. (2020). Comparison of Deep Learning and Traditional Machine Learning Techniques for Classification of Pap Smear Images. Applied Sciences, 10(17), 5752. <https://doi.org/10.48550/arXiv.2009.06366>.

### 1.7. Propuesta de aplicación de algoritmos predictivos

a) Pregunta a responder con un algoritmo supervisado

¿Un frame de videovigilancia corresponde a un comportamiento normal o a un comportamiento sospechoso?

Para responder esta pregunta se propone la aplicación de un algoritmo supervisado, entrenado con un dataset previamente etiquetado. Se compararán dos enfoques:

* **Machine Learning tradicional (SVM, Random Forest):** utilizados como línea base para clasificar las imágenes en dos categorías. Este tipo de modelos requiere que las características (features) relevantes sean previamente extraídas de los frames del video.
* **Deep Learning (CNN):** implementado como modelo principal, ya que permite aprender automáticamente representaciones espaciales de los frames y secuencias de video. Estudios comparativos han demostrado que las CNN ofrecen una mayor precisión frente a los algoritmos tradicionales en tareas de clasificación de imágenes, como en reconocimiento facial y clasificación médica.

De este modo, el modelo supervisado no solo permitirá predecir en tiempo real si la conducta observada es normal o anómala, sino también justificar, mediante evidencia experimental, la superioridad de CNN frente a los algoritmos clásicos de machine learning en el contexto de videovigilancia urbana.

c) Caso de uso de un algoritmo de aprendizaje profundo

**Caso de estudio: Suspicious Behavior Detection with Temporal Feature Extraction and Time-Series Classification for Shoplifting Crime Prevention**

1. **Descripción del caso de estudio:**

El caso de estudio se fundamenta en el artículo de Nazir et al. (2023), donde se presenta un enfoque innovador de dos etapas para la detección de comportamientos sospechosos, específicamente orientado a la prevención de hurtos en tiendas (*shoplifting*).

A diferencia de los métodos tradicionales que dependen del análisis de características espaciales extraídas directamente de los píxeles mediante redes convolucionales (CNN) pesadas (como I3D), este método propone una estrategia más eficiente:

* **Primera Etapa (Extracción de Características Temporales):** Se utiliza un modelo de detección y seguimiento de objetos en tiempo real, **YOLOv5 con Deep Sort**, para identificar y rastrear a las personas a través de los fotogramas del video. En lugar de procesar la imagen completa, el sistema extrae únicamente las coordenadas de los cuadros delimitadores (*bounding boxes*) de cada individuo (posición izquierda, superior, ancho y alto) en cada fotograma.
* **Segunda Etapa (Clasificación de Series Temporales):** La secuencia de coordenadas de cada persona se estructura como un conjunto de datos tabular, transformando el problema de análisis de video en un problema de **clasificación de series temporales**. Este nuevo dataset se utiliza para entrenar modelos de aprendizaje profundo especializados en secuencias, como **XceptionTime** y **MiniRocket**, que aprenden a distinguir entre patrones de movimiento normales y anómalos.

El estudio se llevó a cabo utilizando el conocido dataset **UCF Crime**, centrándose en los videos etiquetados como "shoplifting" para entrenar y validar el modelo.

1. **Objetivos:**

* Desarrollar un modelo de aprendizaje profundo **eficiente y rápido** para la detección de comportamientos sospechosos, evitando el alto costo computacional de los métodos basados en extracción de características espaciales de video.
* Demostrar la viabilidad de **transformar un problema de visión por computadora en uno de clasificación de series temporales** mediante el uso de coordenadas de seguimiento de objetos como características principales.
* Comparar el rendimiento del enfoque propuesto con un modelo de última generación (SOTA) como **RTFM (Robust Temporal Feature Magnitude)**, que utiliza características I3D, tanto en precisión como en velocidad de inferencia.
* Evaluar y determinar cuál de los modernos algoritmos de clasificación de series temporales (InceptionTime, XceptionTime, XCM, MiniRocket) ofrece los mejores resultados para esta tarea específica.

1. **Entradas de datos:**

* **Frames de video: Secuencias de imágenes extraídas de las cámaras de vigilancia del dataset UCF Crime**.
* **Etiquetas de comportamiento: Cada video está clasificado como Normal o Abnormal (anómalo) para el entrenamiento y la evaluación supervisada**.
* **Características temporales (entradas al modelo de Deep Learning):**
* Coordenadas del *bounding box*: Left, Top, Width, Height.
* Identificadores de secuencia: Clip, Frame, Person ID para construir la serie temporal de cada individuo rastreado.

1. **Resultados destacados:**

El enfoque propuesto demostró un rendimiento superior al de la línea base (RTFM), consolidándose como una alternativa robusta y eficiente.

* F1-score: Los mejores modelos, XceptionTime y MiniRocket, alcanzaron un F1-score mediano del 92% en la validación cruzada.
* Superioridad sobre el SOTA: Este resultado superó en un 3% al modelo de referencia RTFM, que obtuvo un 89% de F1-score.
* Velocidad de Inferencia: El método fue 8.45 veces más rápido que RTFM, gracias a que evita el costoso preprocesamiento de extracción de características I3D.
* Precisión: El modelo XceptionTime logró una alta precisión (media de 0.96 en validación cruzada), lo cual es crucial para minimizar la cantidad de falsas alarmas en un entorno real.

Estos valores demuestran la eficacia del modelo para identificar comportamientos sospechosos o violentos en entornos de videovigilancia.

1. **Conclusión:**

El estudio demuestra de manera concluyente que la estrategia de convertir el análisis de video en una clasificación de series temporales, utilizando las coordenadas de seguimiento de personas, es un método **altamente eficaz y computacionalmente eficiente** para la detección de comportamientos anómalos.

Para el **proyecto de seguridad en el distrito de Chancay**, este enfoque es sumamente relevante y aplicable. Su principal ventaja radica en su eficiencia: al no requerir el procesamiento intensivo de cada píxel, puede ser implementado en sistemas con recursos de hardware más limitados, como los que podrían existir en una central de monitoreo municipal. La alta velocidad de inferencia asegura una **detección en tiempo real**, permitiendo una respuesta rápida de las autoridades ante un posible incidente. Finalmente, al centrarse en los patrones de movimiento de las personas, el modelo puede generalizar mejor ante distintas condiciones de iluminación y entornos urbanos variados presentes en Chancay.

**Referencia:**  
Dhruba, A., Kumar, A., & Jangra, A. (2025). *Deep BiLSTM Attention Model for Spatial and Temporal Anomaly Detection in Video Surveillance.* *Sensors, 25*(2), 251. <https://doi.org/10.3390/s25010251>

# 2. Estado del Arte

### 2.1. Preguntas de investigación

## 

1. ¿Qué modelos de inteligencia artificial son más utilizados en la detección de comportamientos sospechosos o violentos en sistemas de videovigilancia?
2. ¿Cuáles son las técnicas de detección de objetos más efectivas para la vigilancia en tiempo real?
3. ¿Qué métricas y enfoques de validación son utilizados para evaluar la eficacia de sistemas de videovigilancia basados en IA en contextos urbanos reales?
4. ¿Qué retos técnicos y oportunidades de mejora se identifican al implementar sistemas de detección inteligente en entornos reales como el distrito de Chancay?

## Respuesta a las preguntas de investigación a través del análisis de papers

### **"Performance Comparison of Convolutional Neural Networks (CNN) and Support Vector Machine (SVM) Algorithms in Human Face Classification”**

1. En este estudio se comparan dos enfoques: Support Vector Machine (SVM), como ejemplo de algoritmo de machine learning tradicional, y Convolutional Neural Networks (CNN), como técnica de deep learning. Aunque el contexto es reconocimiento facial y no videovigilancia urbana, los resultados muestran que CNN obtiene mayor precisión (96.12%) que SVM (76.53%), reforzando la tendencia de que las CNN superan a los modelos tradicionales en problemas visuales complejos.
2. El artículo no aborda detección de objetos directamente, sino clasificación de imágenes. Sin embargo, demuestra que CNN es más efectiva que SVM para la identificación visual, lo que respalda su aplicabilidad en la detección de objetos y comportamientos en videovigilancia.

Se utilizan métricas clásicas de clasificación: accuracy (precisión global). CNN logra un 96.12% frente a un 76.53% de SVM. Aunque el trabajo no aplica en videovigilancia directamente, sí valida el criterio de que CNN logra mejor rendimiento medido en precisión frente a algoritmos tradicionales

1. Kim et al. (2018) proponen que para ambos enfoques realizados en su investigación se aplique validación cruzada con k=5 (*5-fold cross-validation*), con el fin de evaluar la validez del algoritmo de clasificación. Esto prevendría el problema de sobreajuste y aseguraría que el modelo de predicción tiene un performance satisfactorio en datos nuevos no vistos.
2. El estudio evidencia que algoritmos tradicionales como SVM requieren menor capacidad computacional, siendo útiles en escenarios con recursos limitados. Sin embargo, las CNN, aunque más demandantes en cómputo, alcanzan mejores resultados, lo que sugiere que en Chancay podría implementarse un enfoque híbrido: usar CNN para máxima precisión en entornos críticos y SVM como respaldo en situaciones de bajo hardware.

### **"Comparison of deep learning and traditional machine learning techniques for classification of Pap smear images”**

1. Aunque el artículo trabaja con imágenes médicas, se comparan **modelos tradicionales** (Random Forest, K-Nearest Neighbors, SVM) frente a **CNN**. Los resultados muestran que CNN supera ampliamente a los algoritmos tradicionales en precisión y recall, lo que también aplica a problemas de videovigilancia, donde la clasificación binaria (normal vs sospechoso) es fundamental.
2. El estudio demuestra que los modelos de ML clásico requieren extracción manual de características, mientras que CNN aprende representaciones de forma automática. Este hallazgo es extrapolable a videovigilancia: las CNN resultan más efectivas para detección y clasificación de patrones visuales en tiempo real que los métodos tradicionales.
3. Los autores aplicaron métricas como **accuracy**, **recall** y **F1-score**, concluyendo que CNN obtiene valores más altos en todas ellas en comparación con RF, SVM y KNN. Estas métricas son equivalentes a las usadas en videovigilancia, reforzando la aplicabilidad del enfoque.
4. El artículo destaca que los modelos tradicionales son menos costosos computacionalmente, pero presentan limitaciones en escenarios con alta variabilidad visual. Por el contrario, las CNN requieren mayor capacidad de cómputo, pero ofrecen mayor precisión y robustez. Esto sugiere que en un caso práctico como Chancay debe buscarse un equilibrio entre **precisión (CNN)** y **eficiencia computacional (modelos ligeros)**, por ejemplo, mediante optimización de arquitecturas profundas o uso de hardware especializado.

### **"Anomaly Detection in Video Surveillance Using YOLOv8”**

1. Los autores presentan un enfoque híbrido usando YOLOv8 para detección por frame de personas/objetos y combinan esa información con análisis temporal/motion (features derivados de movimiento y conteos) y un componente de detección de anomalías (reglas/detector sobre las features extraídas). El paper recomienda este tipo de pipeline híbrido porque aprovecha la robustez de detectores de objetos modernos y la información temporal para distinguir comportamiento atípico.
2. Se usa YOLOv8 por su capacidad para hacer detección en tiempo real con alta precisión (rapidez en Edge/RT) y por su facilidad de integrar clases. Los autores filtrar mediante confidence thresholds y agregan conteos/flags por frame para la etapa de anomalía.
3. Los autores reportan medidas clásicas de clasificación (Accuracy, Precision, Recall, F1, ROC-AUC) a nivel frame y/o video clip, y además discute latencia y fps. Declaran métricas globales (ej. 92.3% accuracy en su conjunto) pero no siempre detallan balance de clases ni evaluación por escenario.
4. Los autores identifican varios retos técnicos al implementar sistemas de detección inteligente en entornos reales. Uno de los principales desafíos es mantener un equilibrio entre precisión y la velocidad, ya que la detección en tiempo real exige un procesamiento rápido sin comprometer la exactitud en la identificación de comportamientos anómalos.

# 3. Implementación

La implmentación del presente proyecto tiene como objetivo identificar comportamientos anómalos o sospechosos en tiempo real a partir de características de frames de un dataset de videovigilancia llamado CamNuvem.

El proceso para lograrlo integra diferentes etapas, que incluyen la detección de personas u objetos mediante YOLO, la extracción y estructuración de características relevantes y la aplicación de múltiples algoritmos de Machine Learning y Deep Learning para el análisis y clasificación de los comportamientos observados.

Asimismo, se detallan las metodologías empleadas, los modelos comparados entre ellos tenemos SVM, Random Forest, XGBoost, CatBoost, LightGBM y CNN, el diseño y preprocesamiento del dataset generado a partir de frames extraídos de frames de cámaras de seguridad, así como las reglase y estrategias de entrenamiento y validación cruzada aplicadas para optimizar la detección de anomalías.

Esta implementación busca encontrar al mejor modelo para poder automatizar el proceso de monitoreo y análisis visual, reduciendo la dependencia del operador humano y mejorando la capacidad de respuesta ante incidentes en tiempo real. Dado el contexto del crecimiento urbano, esta propuesta tecnológica representa un paso esencial hacia un sistema de videovigilancia inteligente, orientado a fortalecer la seguridad ciudadana en Chancay.

## 3.1 - Implementación de algoritmo supervisado

Algoritmos Seleccionados y su justificación:

En este trabajo se emplean diversos modelos de aprendizaje automático con el propósito de identificar comportamientos anómalos en características de frames de cámaras de seguridad. Los algoritmos seleccionados se dividen en dos grupos: modelos supervisados, que requieren datos etiquetados, y modelos no supervisados, que permiten detectar patrones irregulares sin etiquetas explícitas.

**Modelos Supervisados**

Los modelos supervisados se entrenan a partir de ejemplos previamente clasificados como “normales” o “anómalos”. En este grupo se consideran algoritmos clásicos de clasificación, esto debido al problema que afrontamos, reconocidos por su eficacia a la hora de generalizar:

1. Regresión Logística: Uno de los modelos más utilizados en clasificación binaria. Su funcionamiento se basa en estimar la probabilidad de que una observación pertenezca a una clase determinada mediante una función sigmoide. Lo que la hace tan especial es su interpretabilidad, ya que permite analizar el peso o influencia de cada variable con la función principal.
2. K-Nearest Neighbors (KNN): Este modelo clasifica basándose en la similitud con sus k vecinos más cercanos. Utiliza las distancias de cada dato para saber dónde ubicar nuevos puntos en la evaluación. Es un algoritmo muy sencillo de usar y es fácil de calibrar ya que se basa su rendimiento en su hiperparametro principal “K”, si este se calibra bien puede dar excelentes resultados. Este modelo necesita que los datos sean escalados debido a que trabaja con distancias.
3. Support Vector Machine (SVM): El SVM es un modelo que busca encontrar un vector que separe las clases maximizando lo mejor posible el margen entre los datos. Es también útil cuando las clases no son linealmente separables, ya que permite el uso de kernels que pueden proyectar los datos a espacios de mayor dimensión. Es útil tanto en clasificación binaria como en multiclase. Como el KNN, al trabajar con distancias necesita que se escalen los datos.
4. Random Forest: Este modelo pertenece a los métodos de ensembles basados en arboles de decisión. Lo que hace es construir múltiples árboles entrenados sobre subconjuntos aleatorios de datos y características, y lo que hace es combinar sus predicciones mediante votación. Es un modelo que es robusto contra el sobreajuste y su capacidad para manejar relaciones no lineales y variables con diferentes escalas.
5. XGBoost: El Extreme Gradient Boosting es una implementación optimizada del método de boosting, la cual su trabajo es construir árboles de decisión secuencialmente para corregir los errores del modelo anterior. Destaca por su alta eficiencia computacional y su excelente rendimiento en tareas de clasificación con grandes volúmenes de datos.
6. CatBoost: Es un algoritmo basado en gradient boosting, su especialidad es su manera eficiente de trabajar con variables categóricas sin la necesidad de aplicar técnicas de conversión a números como el Label Encoder. Utiliza un enfoque de ordered boosting que mejora la estabilidad del modelo. También radica en su precisión y rapidez sin necesidad de preprocesamientos extensivos, lo que lo hace ideal para conjuntos de datos con diferentes tipos de variables.
7. LightGBM: Este modelo desarrollado por Microsoft, también perteneciente a la familia del gradient boosting, pero este está optimizado para el entrenamiento en grandes volúmenes de datos con bajo consumo de memoria. Utiliza el método de leaf-wise growth que prioriza las hojas con mayor ganancia, logrando así una mayor eficiencia en los resultados. Útil en cualquier tipo de problema de machine learning.

**Modelos No Supervisados**

Para escenarios donde no se dispone de etiquetas precisas, se implementan modelos no supervisados de detección de anomalías, orientados a identificar comportamientos inusuales a partir de patrones de datos históricos:

1. Isolation Forest: Se basa en la idea de que las anomalías son observaciones que pueden aislarse fácilmente del resto de los datos. En lugar de modelar explícitamente los comportamientos normales, este algoritmo construye múltiples árboles de decisión aleatorios que dividen el espacio de características en subconjuntos, buscando aislar los puntos individuales. De hecho, el trabajo de Praveen M.N. y Sandeep (2025) propone una arquitectura híbrida que integra YOLOv8 para la detección de características de frames y el modelo Isolation Fores como mecanismo de decisión no supervisado para la detección de anomalías.
2. DBSCAN (Density-Based Spatial Clustering of Applications with Noise): El algoritmo DBSCAN se basa en la densidad de los datos para formar grupos, identificando regiones de alta concentración de puntos como clústeres y considerando como ruido o anomalía aquellos puntos que no pertenecen a ninguna región densa. Este enfoque no requiere especificar el número de clústeres, lo que lo hace especialmente adecuado para conjuntos de datos dinámicos.

### 3.1.1. Resumen del Dataset

1. Estructura del Dataset:

En este trabajo se implementará un enfoque de aprendizaje supervisado para construir un modelo capaz de predecir, a partir de las características de personas y la presencia de ciertos objetos, si una imagen corresponde a un frame anómalo o normal. Dado que el propósito del sistema es detectar comportamientos o situaciones fuera de lo común en entornos de video vigilancia, la métrica más relevante será el recall (sensibilidad), que indica cuántos frames anómalos el modelo logra identificar correctamente. Un valor alto de recall implica que pocos eventos anómalos pasan desapercibidos. No obstante, también se buscará mantener una precisión elevada, para evitar falsas alarmas en exceso.

 Para ello, se utilizó el dataset CamNuvem (https://www.kaggle.com/datasets/louisamakye/camnuvem/data), el cual contiene diferentes imágenes extraídas de videos de cámaras de seguridad, organizadas en carpetas de entrenamiento y testeo, y clasificadas como normal o anómalo.

A partir de este dataset, se empleó un tracker llamado YOLOv8 para extraer diferentes características de las personas detectadas en cada imagen y el conteo de algunas clases aparte. Estas características se guardaron en dos archivos .csv:

\* Uno con información de cada frame como observación individual.

\* Otro con un resumen de cada video (conjunto de frames) como observación, incluyendo diferentes flags binarias que indican eventos relevantes.

### 3.1.2. Creación del Dataset

El siguiente script implementa un proceso automatizado de análisis de video mediante visión por computadora, cuyo objetivo es extraer características relevantes de frames capturados por cámaras de videovigilancia. En él se emplea el modelo YOLOv8 el cual se centra en identificar personas dentro de los frames calcular atributos espaciales y temporales (como área del bounding box, confianza, movimiento, número de personas por frame, etc) y transformar dicha información en un conjunto de datos estructurado. Este dataset servirá para entrenar y validar modelos de Machine Learning y Deep Learning orientados a la detección de comportamientos anómalos.

El flujo general del código es el siguiente:

1. Carga del modelo YOLOv8 preen trenado (yolov8n.pt).
2. Procesamiento de imágenes organizadas en carpetas según su tipo (normal o anormal).
3. Detección y extracción de atributos de cada persona detectada (posición, tamaño, confianza, desplazamiento, etc.).
4. Cálculo de estadísticas agregadas por video, como promedio de confianza, varianza del movimiento o tasa de detección.
5. Generación de archivos CSV finales, donde se consolidan las características que luego se emplearán para la fase de modelado.

Cabe recalcar que se usaron las clases del modelo YOLO ya preentrenado y se eligieron las más relevantes para este caso de uso las cuales son: Persona, Carro, Motocicleta, Bus, Mochila, Bolsa, Maleta, Cuchillo, Laptop, Celular.

from ultralytics import YOLO

import os, shutil

import pandas as pd

import re

import numpy as np

model = YOLO("yolov8n.pt")

# Lista de clases relevantes

relevant\_objects = {

0: "person",

2: "car",

3: "motorcycle",

5: "bus",

24: "backpack",

26: "handbag",

28: "suitcase",

43: "knife",

63: "laptop",

67: "cell phone",

}

def natural\_sort\_key(filename):

match = re.search(r'frame\_(\d+)', filename)

return int(match.group(1)) if match else float('inf')

def process\_split(split, base\_path, output\_path):

all\_dfs = []

for cls in ["anomaly", "normal"]:

folder = os.path.join(base\_path, split, cls)

for video\_folder in os.listdir(folder):

input\_folder = os.path.join(folder, video\_folder)

if os.path.isdir(input\_folder):

print(f"Procesando: {input\_folder}")

image\_files = [f for f in os.listdir(input\_folder) if f.endswith(('.jpg', '.png', '.jpeg'))]

if not image\_files:

continue

image\_files.sort(key=natural\_sort\_key)

all\_rows = []

previous\_frame\_data = {}

for image\_file in image\_files:

image\_path = os.path.join(input\_folder, image\_file)

frame\_match = re.search(r'frame\_(\d+)', image\_file)

frame\_number = int(frame\_match.group(1)) if frame\_match else 0

# Detectar todas las clases relevantes

results = model.predict(

source=image\_path,

save=False,

save\_txt=False,

classes=list(relevant\_objects.keys()),

imgsz=640,

conf=0.3

)

# Inicializar diccionario para columnas binarias de objetos

object\_presence = {obj\_name: 0 for obj\_name in relevant\_objects.values() if obj\_name != "person"}

person\_detections = []

other\_objects\_detected = []

for result in results:

if result.boxes is not None and len(result.boxes) > 0:

boxes = result.boxes.xyxy.cpu().numpy()

confidences = result.boxes.conf.cpu().numpy()

class\_ids = result.boxes.cls.cpu().numpy().astype(int)

for i, (box, conf, cls\_id) in enumerate(zip(boxes, confidences, class\_ids)):

if cls\_id in relevant\_objects:

obj\_name = relevant\_objects[cls\_id]

if obj\_name == "person":

# Procesar personas normalmente

person\_detections.append({

'box': box,

'conf': conf,

'cls\_id': cls\_id,

'index': i

})

else:

# Marcar presencia de otros objetos

object\_presence[obj\_name] = 1

other\_objects\_detected.append(obj\_name)

# Procesar detecciones de personas (cálculos de tracking, etc.)

if person\_detections:

for person\_data in person\_detections:

box = person\_data['box']

conf = person\_data['conf']

cls\_id = person\_data['cls\_id']

i = person\_data['index']

width = box[2] - box[0]

height = box[3] - box[1]

center\_x = (box[0] + box[2]) / 2

center\_y = (box[1] + box[3]) / 2

movement\_since\_last\_frame = 0

acceleration = 0

if frame\_number > 0 and f"person\_{i}" in previous\_frame\_data:

prev\_center\_x, prev\_center\_y = previous\_frame\_data[f"person\_{i}"]

movement\_since\_last\_frame = np.sqrt(

(center\_x - prev\_center\_x)\*\*2 +

(center\_y - prev\_center\_y)\*\*2

)

previous\_frame\_data[f"person\_{i}"] = (center\_x, center\_y)

df\_row = pd.DataFrame({

"frame\_number": [frame\_number],

"frame\_filename": [image\_file],

"left": [box[0]],

"top": [box[1]],

"right": [box[2]],

"bottom": [box[3]],

"confidence": [conf],

"class\_id": [cls\_id],

"width": [width],

"height": [height],

"bbox\_area": [width \* height],

"aspect\_ratio": [height / width],

"center\_x": [center\_x],

"center\_y": [center\_y],

"movement\_since\_last\_frame": [movement\_since\_last\_frame],

"distance\_from\_center": [np.sqrt((center\_x - 320)\*\*2 + (center\_y - 240)\*\*2)],

"is\_large\_person": [1 if (width \* height) > 10000 else 0],

"is\_small\_person": [1 if (width \* height) < 5000 else 0],

"is\_high\_confidence": [1 if conf > 0.8 else 0],

"label": [cls],

"video\_folder": [video\_folder],

"split": [split],

"person\_id": [i],

"total\_persons\_in\_frame": [len(person\_detections)],

# Añadir columnas binarias para otros objetos

\*\*{f"has\_{obj\_name}": [object\_presence[obj\_name]] for obj\_name in object\_presence}

})

all\_rows.append(df\_row)

else:

# Frame sin personas, pero con información de otros objetos

df\_row = pd.DataFrame({

"frame\_number": [frame\_number],

"frame\_filename": [image\_file],

"left": [None], "top": [None], "right": [None], "bottom": [None],

"confidence": [None], "class\_id": [None],

"width": [None], "height": [None], "bbox\_area": [None],

"aspect\_ratio": [None], "center\_x": [None], "center\_y": [None],

"movement\_since\_last\_frame": [None], "distance\_from\_center": [None],

"is\_large\_person": [0], "is\_small\_person": [0], "is\_high\_confidence": [0],

"label": [cls], "video\_folder": [video\_folder], "split": [split],

"person\_id": [None], "total\_persons\_in\_frame": [0],

# Añadir columnas binarias para otros objetos

\*\*{f"has\_{obj\_name}": [object\_presence[obj\_name]] for obj\_name in object\_presence}

})

all\_rows.append(df\_row)

previous\_frame\_data = {}

if all\_rows:

df\_folder = pd.concat(all\_rows, ignore\_index=True)

df\_folder = df\_folder.sort\_values("frame\_number").reset\_index(drop=True)

df\_folder = calculate\_additional\_features(df\_folder)

all\_dfs.append(df\_folder)

print(f"{video\_folder}: {len(image\_files)} frames procesados")

if all\_dfs:

final\_df = pd.concat(all\_dfs, ignore\_index=True)

final\_df = final\_df.sort\_values(["video\_folder", "frame\_number"])

output\_csv = os.path.join(output\_path, f"{split}\_dataset\_with\_features.csv")

final\_df.to\_csv(output\_csv, index=False)

create\_ml\_features(final\_df, split, output\_path)

return final\_df

else:

print(f"No se generaron datos para {split}")

return None

def calculate\_additional\_features(df):

if len(df) == 0:

return df

video\_groups = df.groupby('video\_folder')

for video\_name, group in video\_groups:

video\_indices = group.index

if len(group) > 1:

detection\_rate = len(group[group['class\_id'].notna()]) / len(group)

avg\_persons = group['total\_persons\_in\_frame'].mean()

df.loc[video\_indices, 'video\_detection\_rate'] = detection\_rate

df.loc[video\_indices, 'video\_avg\_persons'] = avg\_persons

return df

def create\_ml\_features(df, split, output\_path):

ml\_features = []

# Obtener todas las columnas binarias de objetos

object\_columns = [col for col in df.columns if col.startswith('has\_')]

for video\_name, group in df.groupby('video\_folder'):

detected\_frames = group[group['class\_id'].notna()]

if len(detected\_frames) == 0:

continue

# Solo añadir las columnas binarias (1 si aparece al menos una vez en el video)

object\_binary\_features = {}

for obj\_col in object\_columns:

obj\_name = obj\_col.replace('has\_', '')

object\_binary\_features[obj\_col] = 1 if group[obj\_col].max() > 0 else 0

features = {

'video\_name': video\_name,

'label': group['label'].iloc[0],

'frames\_with\_detection': len(detected\_frames),

'detection\_rate': len(detected\_frames) / len(group),

\*\*object\_binary\_features # Solo añadir las columnas binarias

}

if len(detected\_frames) > 0:

features.update({

'mean\_confidence': detected\_frames['confidence'].mean(),

'mean\_bbox\_area': detected\_frames['bbox\_area'].mean(),

'mean\_movement': detected\_frames['movement\_since\_last\_frame'].mean(),

'max\_movement': detected\_frames['movement\_since\_last\_frame'].max(),

'mean\_distance\_from\_center': detected\_frames['distance\_from\_center'].mean(),

'large\_person\_ratio': detected\_frames['is\_large\_person'].mean(),

'high\_confidence\_ratio': detected\_frames['is\_high\_confidence'].mean(),

'avg\_persons\_per\_frame': detected\_frames['total\_persons\_in\_frame'].mean(),

'var\_movement': detected\_frames['movement\_since\_last\_frame'].var(),

'var\_bbox\_area': detected\_frames['bbox\_area'].var(),

'var\_confidence': detected\_frames['confidence'].var(),

'var\_persons\_per\_frame': detected\_frames['total\_persons\_in\_frame'].var(),

'max\_persons\_per\_frame': detected\_frames['total\_persons\_in\_frame'].max(),

'max\_bbox\_area': detected\_frames['bbox\_area'].max(),

'min\_bbox\_area': detected\_frames['bbox\_area'].min(),

'max\_confidence': detected\_frames['confidence'].max(),

'min\_confidence': detected\_frames['confidence'].min(),

'extreme\_movement\_count': (detected\_frames['movement\_since\_last\_frame'] > 50).sum(),

'crowd\_frames\_count': (detected\_frames['total\_persons\_in\_frame'] >= 5).sum()

})

ml\_features.append(features)

if ml\_features:

ml\_df = pd.DataFrame(ml\_features)

ml\_csv = os.path.join(output\_path, f"{split}\_ml\_ready.csv")

ml\_df.to\_csv(ml\_csv, index=False)

print(f"ML features guardados en: {ml\_csv}")

print(f"Columnas incluidas: {list(ml\_df.columns)}")

base\_path = "images"

output\_path = "features"

os.makedirs(output\_path, exist\_ok=True)

#df\_train = process\_split("training", base\_path, output\_path)

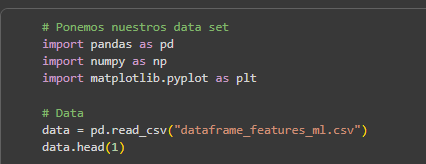
#df\_test = process\_split("test", base\_path, output\_path)

El archivo .csv de resumen por video se utilizará para el aprendizaje supervisado, con un total de 957 observaciones.

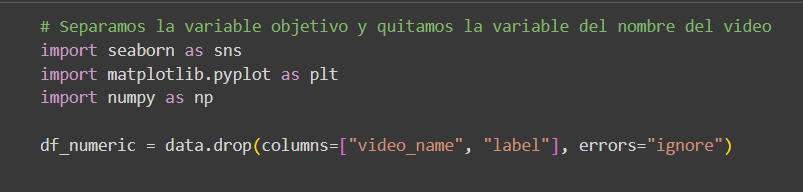
**Estructura del dataset de resumen de videos:**

|  |  |
| --- | --- |
| Nombre de la columna | Descripción |
| video\_name | Nombre del video extraído del dataset CamNuvem |
| label | Variable objetivo clasifica si un video es anomaly y normal. |
| frames\_with\_detection | Número de frames que detectaron al menos un objeto. |
| detection\_rate | Proporción de frames con detecciones respecto al total de frames del video. |
| mean\_confidence | Promedio de las probabilidades de detección de todas las personas identificadas por YOLO. |
| mean\_bbox\_area | Promedio del área de los bounding boxes de las personas detectadas. Presenta el tamaño promedio de las personas observadas. |
| mean\_movement | Promedio del desplazamiento entre frames consecustivos, calculado con la distancia euclidiana entre los centros de las cajas de una persona en dos frames sucesivos. |
| max\_movement | Mayor desplazamiento registrado para una persona entre dos frames consecutivos. |
| mean\_distance\_from\_center | Promedio de la distancia del centro del bounding box respecto al centro del frame. ¿Dónde se encuentran las personas? |
| large\_person\_ratio | Porcentaje de detecciones con área de caja mayor a un umbral (10,000 píxeles) |
| high\_confidence\_ratio | Porcentaje de detecciones con confianza superior a 0.8. Representa la estabilidad del modelo en su identificación de personas. |
| avg\_persons\_per\_frame | Promedio del número de personas detectadas por frame. |
| var\_movement | Varianza de los desplazamientos de las personas entre frames. Cuantifica la irregularidad de los movimientos. |
| var\_bbox\_area | Varianza de las áreas de las cajas de detección. |
| var\_confidence | Varianza de las puntuaciones de confianza del modelo YOLO. |
| var\_persons\_per\_frame | Varianza en el número de personas detectadas por frame. |
| max\_persons\_per\_frame | Máximo número de personas detectadas en un solo frame. |
| max\_bbox\_area | Área más grande de detección registrada. |
| min\_bbox\_area | Área más pequeña detectada. |
| max\_confidence | Valor máximo de confianza alcanzado por YOLO en cualquier detección del video. |
| min\_confidence | Valor mínimo de confianza observado. |
| extreme\_movement\_count | Número de frames donde el movimiento entre frames supera un umbral (50 píxeles) |
| crowd\_frames\_count | Cantidad de frames con cinco o más personas detectadas simultáneamente. |
| has\_car | Estas variables indican (1 o 0) si el modelo YOLO detectó al menos una vez dicho objeto (clase) dentro del video. |
| has\_motorcycle |
| has\_bus |
| has\_backpack |
| has\_handbag |
| has\_suitcase |
| has\_knife |
| has\_laptop |
| has\_cell phone |

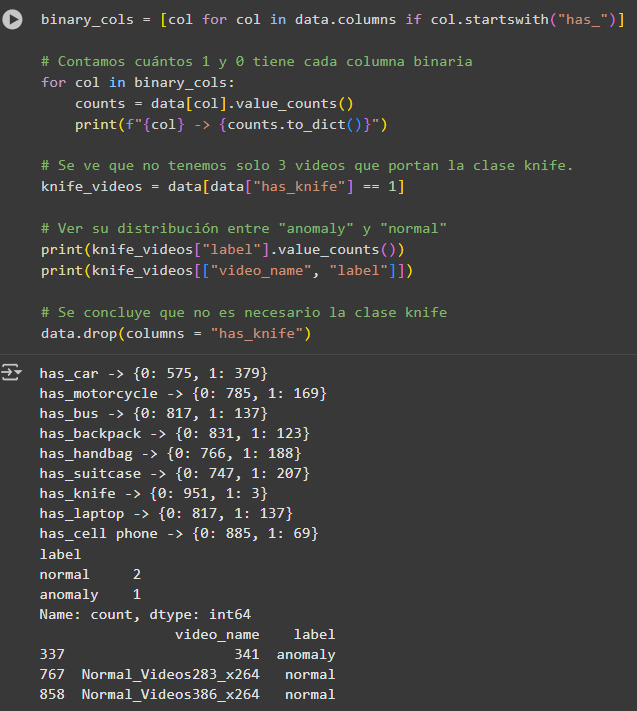
Llamamos al dataset



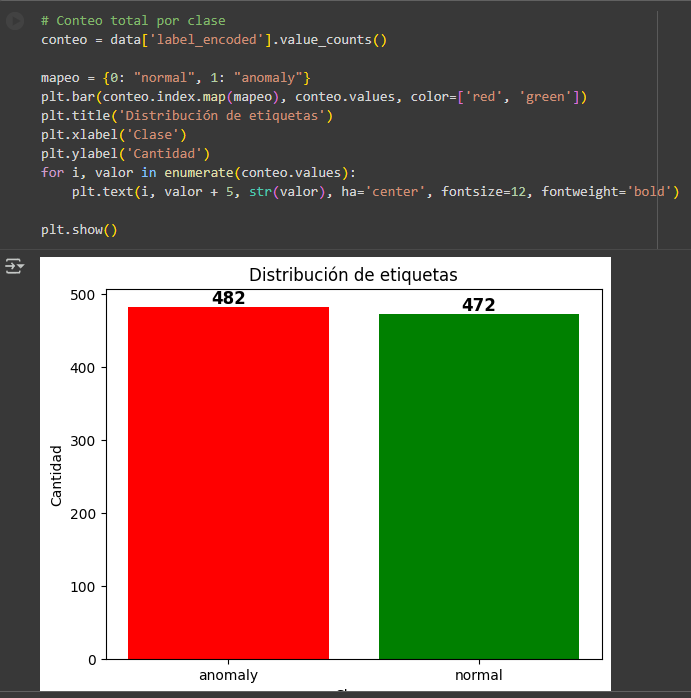
Separamos nuestro dataset de la variable objetivo y quitamos la variable del nombre del video



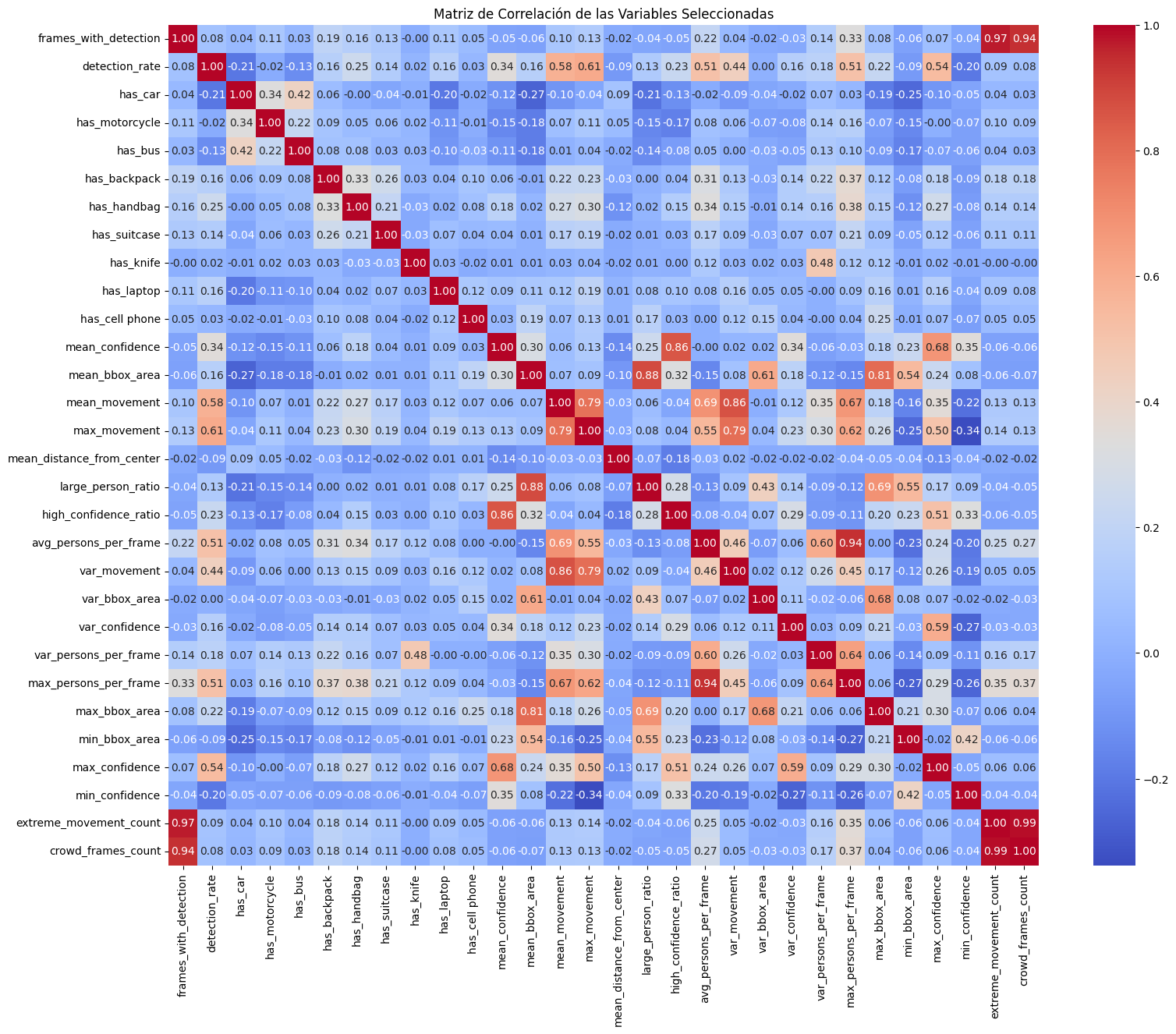
Se ve cuantos videos obtienen al menos un objeto de las clases que queríamos. El resultado nos boto que hay pocos objetos claves y que no aportan casi nada a la clasificación. Uno de ellos es la clase knife la cual solo aparece 3 veces en los 957 videos y los cuales se pudo revisar que eran errores de YOLO al clasificarlo así. Por ende, se decide eliminar del dataset la columna has\_knife.



Se hace el conteo total por clase y tenemos las clases casi balanceadas es por eso que se hace este modelado a nivel de video ya que las clases están muy bien balanceadas, pero luego veremos el caso del nivel de frame donde tendremos las clases muy desbalanceadas.



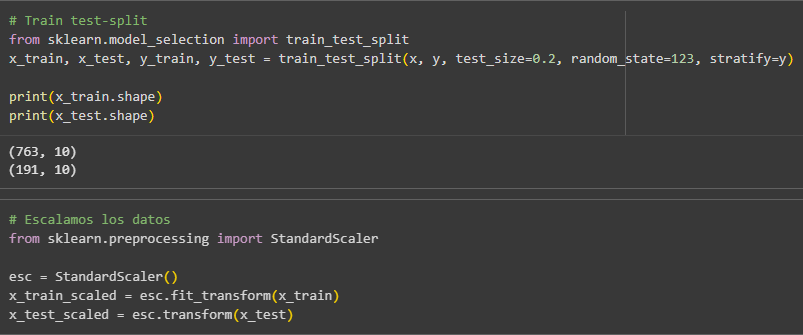
Para no entrenar los modelos con todas las variables, ya que esto generaría ruido o redundancia, lo que se hizo fue un análisis de correlación para ver qué variables estaban muy relacionadas entre sí. Luego se probó distintos subconjuntos de variables y se comparó el recall de cada modelo. Se hicieron varios subconjuntos hasta que se pudo llegar al mejor de estos. Los resultados están en la sección de conclusiones.



Se analizó bien esta matriz de correlación y se puedo ver que algunas variables estaban demasiado relacionadas entre sí. Por ejemplo, “extreme\_movement\_count” y “crowd\_frames\_count” tienen una correlación de 0.98, lo que dice que prácticamente representan lo mismo. Por eso se probó quedarse con solo una de ellas para simplificar el modelo y evitar redundar. También al ver que las variables binarias tienen correlaciones menores a 0.1 o incluso negativas y al tener también pocas detecciones en los videos se decidió quedarse con las dos clases que más aparecían en este caso “has\_car” y “has\_motorcycle”.



Este fue el mejor conjunto de variables que dieron los mejores resultados. Se omitieron varias variables binarias debido a que no aportaban casi nada en la predicción.



Dividimos con un train test-split de 80-20 y escalamos los datos con StandarScaler()

### 3.1.3. Entrenamiento y Validación de Modelos de Aprendizaje Supervisado

La estructura de la evaluación es de la siguiente manera:

Modelo X:

1.   Validación cruzada k-fold (5 splits) solo con los datos de entrenamiento (80%).

2.   Calibración de los hiperparámetros más relevantes según la métrica principal (recall)

3.   Evaluación final en el conjunto de test con los hiperparámetros seleccionados.

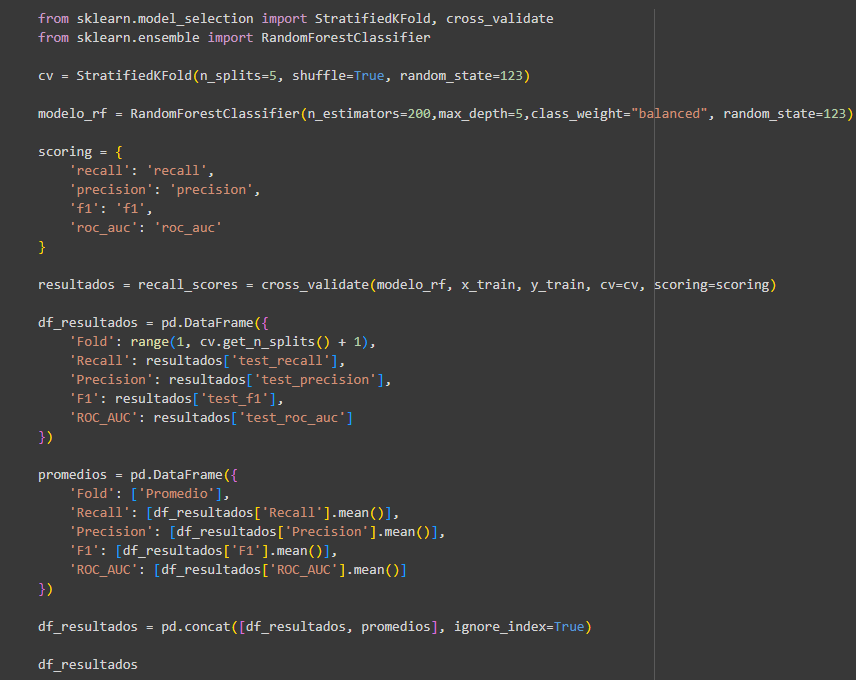
4.   Análisis mediante matriz de confusión y curva ROC-AUC.

5.   Comparación entre los resultados promedio de la validación cruzada y los del test.

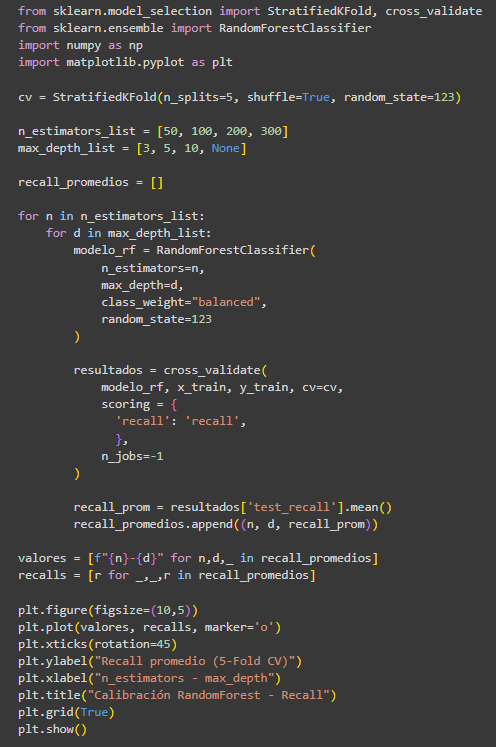
Cada uno de los modelos se evaluará de la manera anteriormente dicha al final se compararán todos los resultados de todos los modelos y se elegirá uno que maneje mejor el caso de uso. Cabe resaltar ya que las clases están bien balanceadas (482 anomaly : 472 normal) el umbral será de 0.5 y el random state para todas las evaluaciones se fijarán en "123".Esto asegura que los resultados sean reproducibles y que la métrica de recall no esté sesgada por un desbalance de clases.

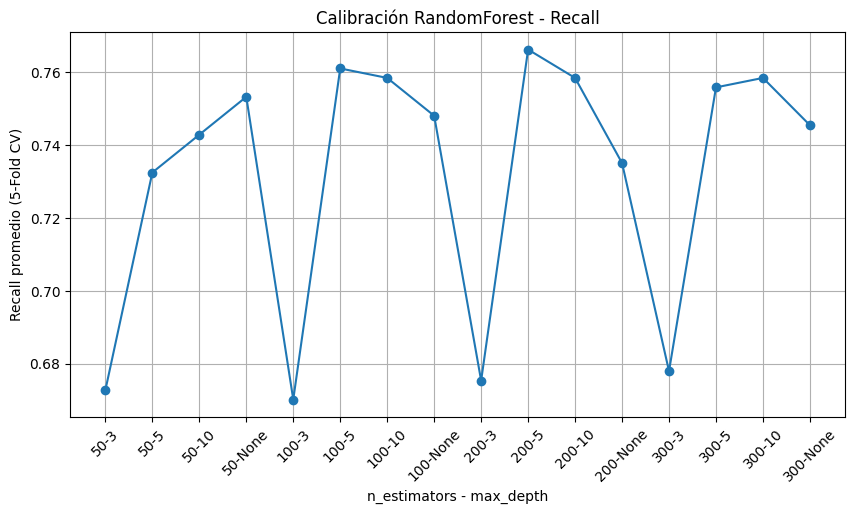
#### 3.1.3.1 Random Forest

Se uso primero Random Forest se validó mediante validación cruzada.

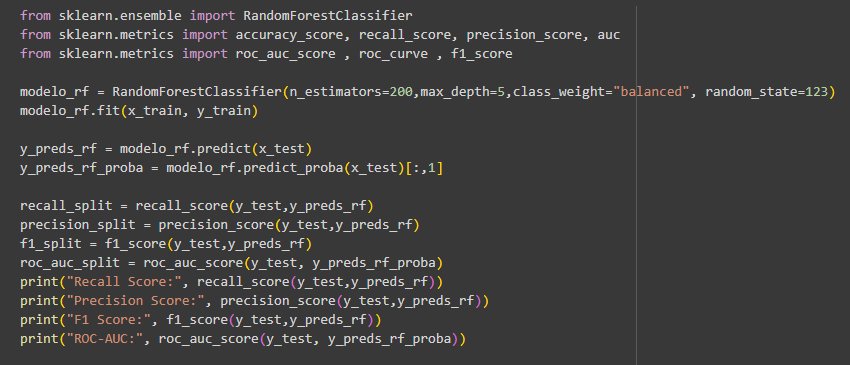


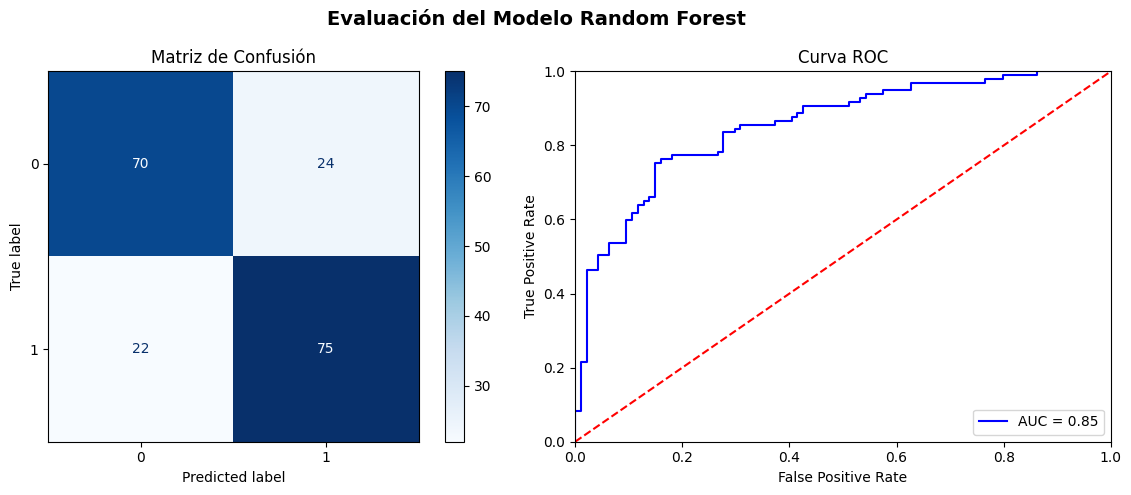
Luego se calibró los mejores hiperparametros en este modelo serian el número de árboles (n\_estimators) y la profundidad (max\_depth). En este caso fue 200 y 5.





Ahora validamos el entrenamiento con el testeo.





Comparamos lo resultados:

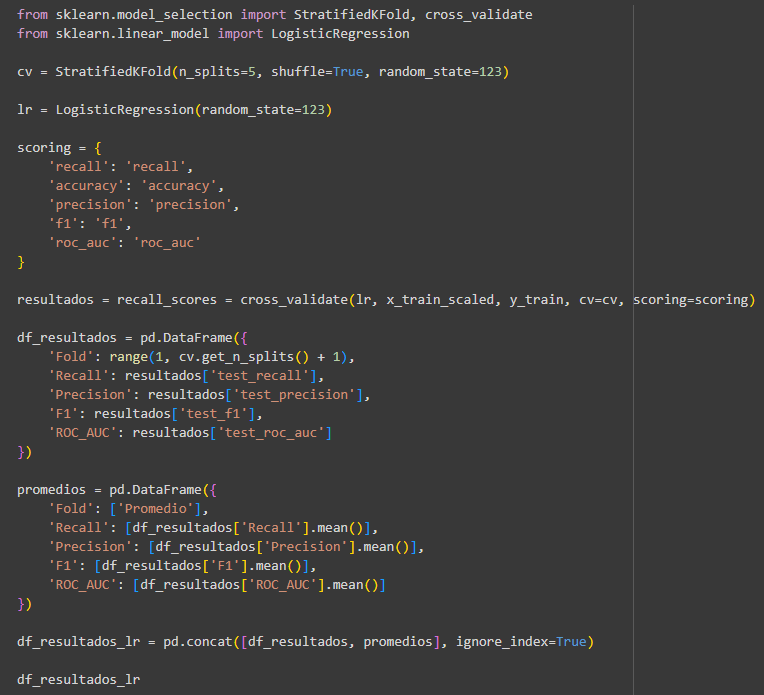
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Método | Recall | Precision | F1 | ROC\_AUC |
| Train/Test Split | 0.7731 | 0.7575 | 0.7653 | 0.8544 |
| K-Fold (5) | 0.7662 | 0.7678 | 0.7652 | 0.8406 |

Análisis comparativo en el modelo Random Forest:

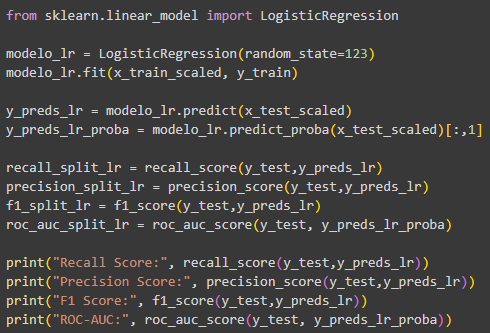
Se puede observar de los resultados que el modelo tiene rendimientos parecidos en los dos métodos de validación aplicados. Aunque se destaca que este modelo trabaja mejor al ser probado en varios subconjuntos que con solo una partición aleatoria del dataset. Esto se puede observar al ver un aumento en la métrica recall y F1 lo que indica mejor capacidad de generalización. La ligera disminución en ROC-AUC indica que en algunos folds el modelo tuvo dificultad para separar correctamente las clases. Pero en general presenta un equilibrio entre precisión y sensibilidad.

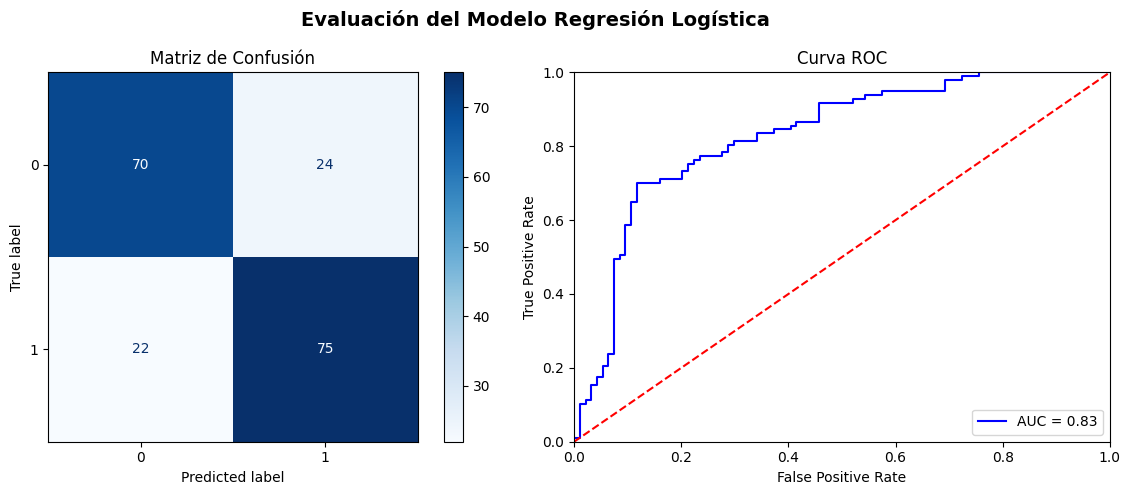
#### 3.1.3.2 Regresión Logística

Ahora se utilizó Regresión Logística se validó mediante validación cruzada.



Se evaluó el modelo con el test (20%)





Comparamos lo resultados:

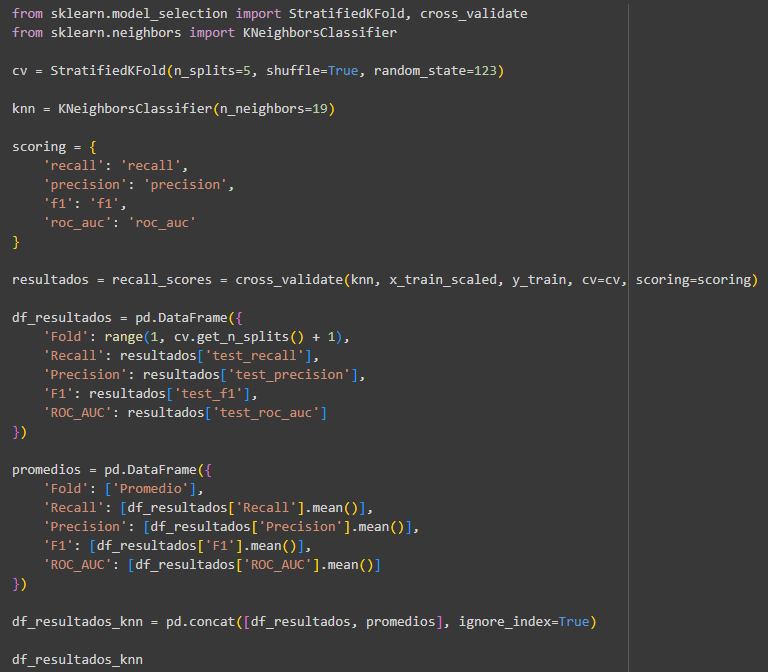
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Método | Recall | Precision | F1 | ROC\_AUC |
| Train/Test Split | 0.7732 | 0.7575 | 0.7653 | 0.8229 |
| K-Fold (5) | 0.7584 | 0.7588 | 0.7579 | 0.8218 |

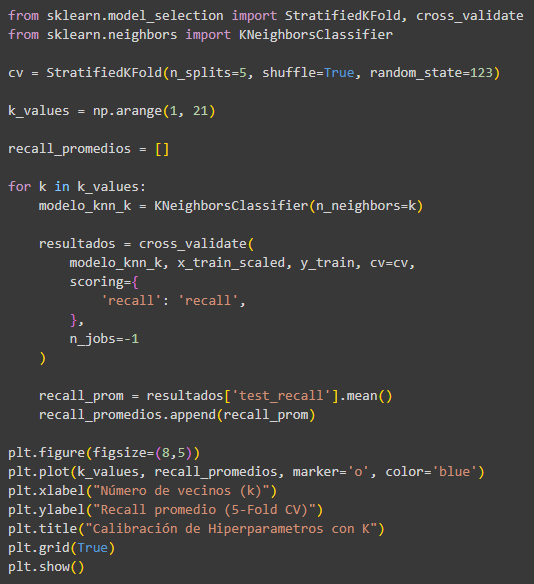
Análisis comparativo en el modelo Regresión Logística:

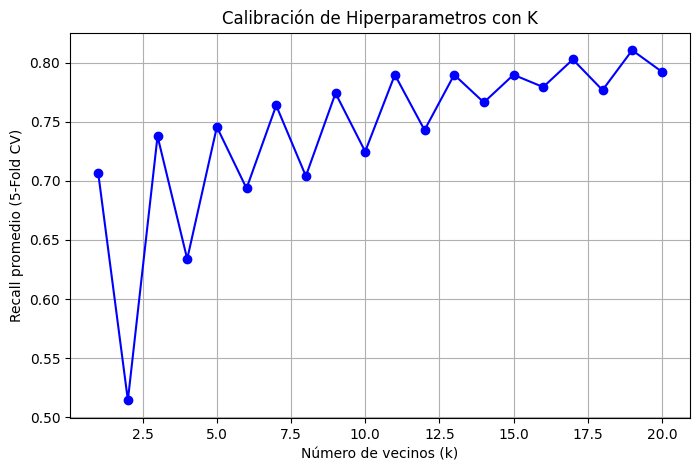
Ahora con los resultados del modelo de Regresión Logística podemos observar una ligera disminución de capacidad discriminatoria entre las clases ya que al ser un modelo lineal no tiene la misma capacidad que con un modelo más complejo. También podemos observar que el modelo detecta un poco menos de casos positivos cuando se entrena en distintos subconjuntos. Esto puede ser que en algunos folds las variables no tienen una relación no lineal entre ellas como si las tienen en otras.

#### 3.1.3.3 KNN (K-nearest-neighbors)

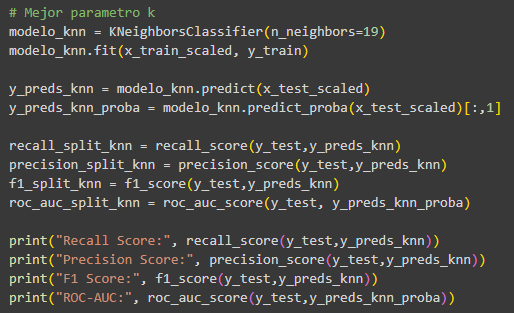
Se continuó con KNN y se calibró su parámetro K dando el mejor recall con un valor de 19.

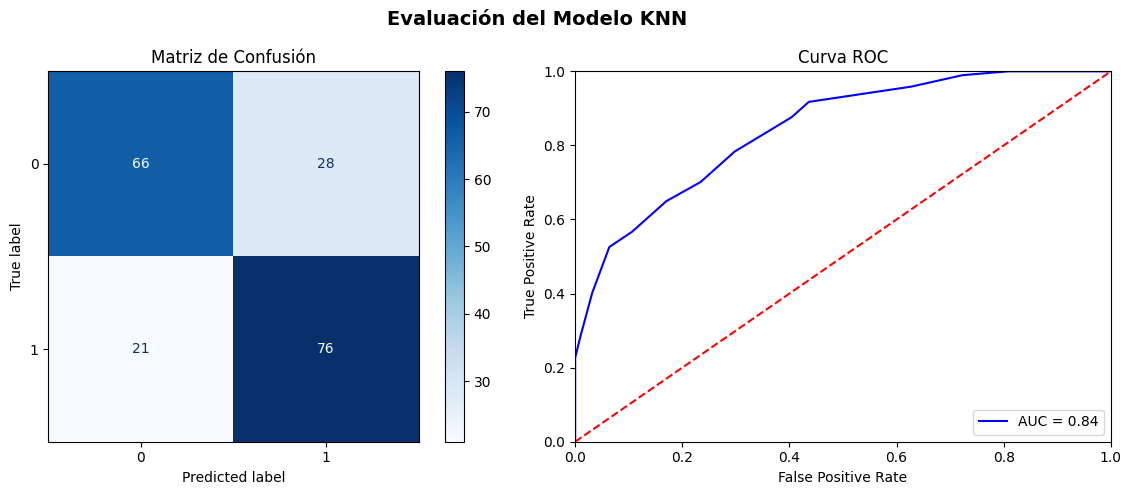






Se validó con el conjunto test.





Comparamos los resultados:

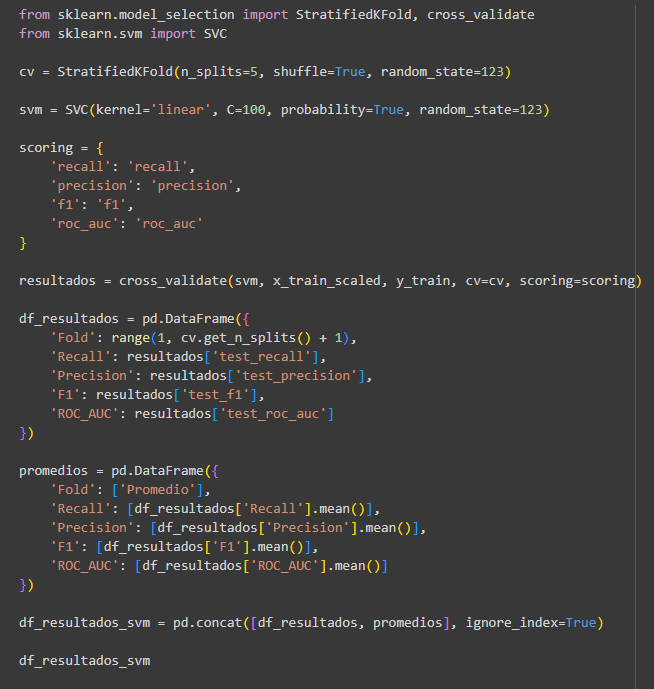
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Método | Recall | Precision | F1 | ROC\_AUC |
| Train/Test Split | 0.7835 | 0.7307 | 0.7562 | 0.8434 |
| K-Fold (5) | 0.8104 | 0.7035 | 0.7523 | 0.7998 |

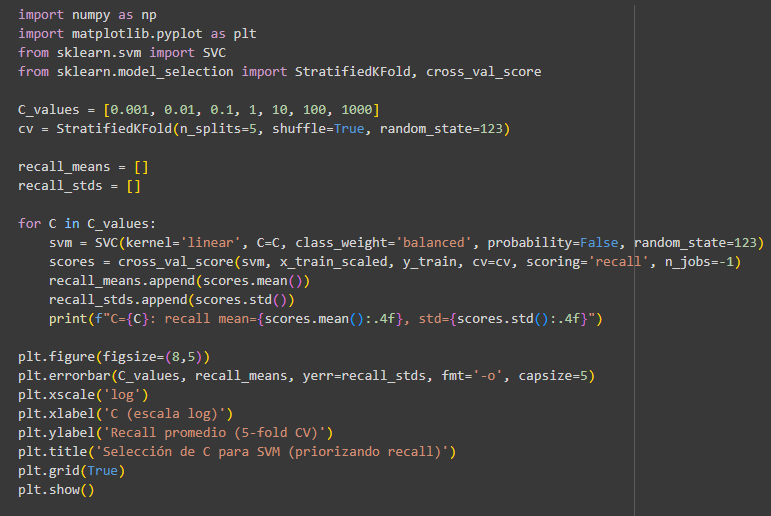
Análisis comparativo en el modelo KNN

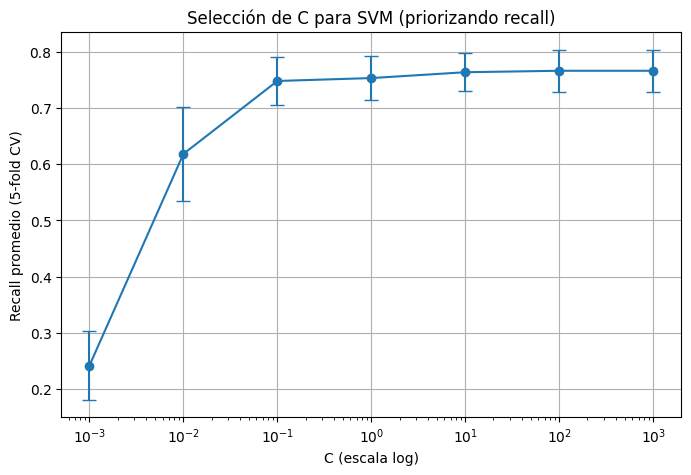
Para el modelo KNN ya con su hiperparametro calibrado en K=19, se puede observar un alto recall, especialmente en la validación cruzada, lo que indica que el modelo es el mejor por ahora en detectar la mayoría de los casos positivos. Pero este incremento en la sensibilidad viene acompañado de una ligera disminución en la precisión, en la parte de los folds, esto puede deberse a que el modelo KNN prioriza la detección cercanos tratando de aumentar la probabilidad de clasificar correctamente los casos positivos, pero eso conlleva a que etiquete por error algunos normales como anormales. Aún asi el modelo mantiene un ROC-AUC estable esto demuestra que posee una buena capacidad discriminatoria global entre clases. Aunque cabe recalcar si K fuera más pequeño al estar el modelo muy dependiente de los datos específicos de cada fold, podría tender a sobreajustar.

#### 3.1.3.4 SVM (Support Vector Machine)

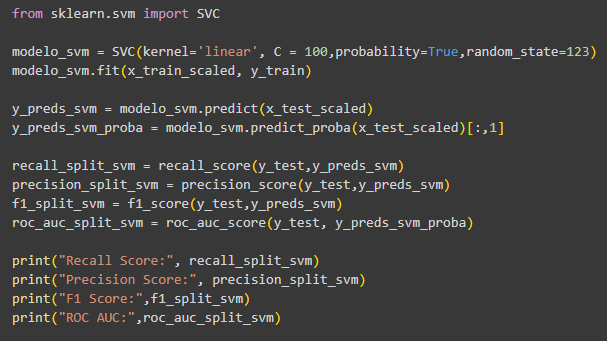
Se continuó con el modelo SVM y se calibró el parámetro C donde penalizaremos más los errores para obtener un mejor recall sin importar las falsas alarmas que vayamos a conseguir.

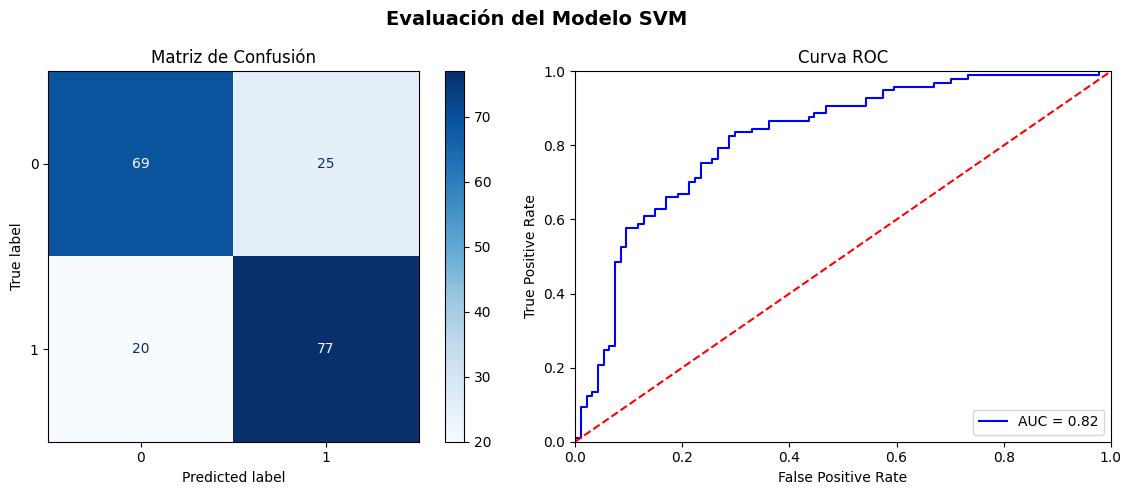






Se valido con el conjunto de test.





Comparamos los resultados.

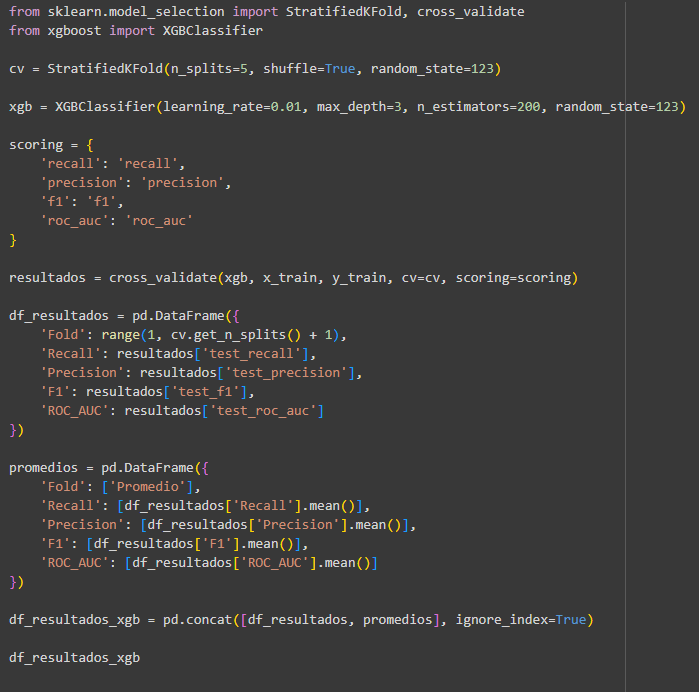
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Método | Recall | Precision | F1 | ROC\_AUC |
| Train/Test Split | 0.7835 | 0.7307 | 0.7562 | 0.8434 |
| K-Fold (5) | 0.8104 | 0.7035 | 0.7523 | 0.7998 |

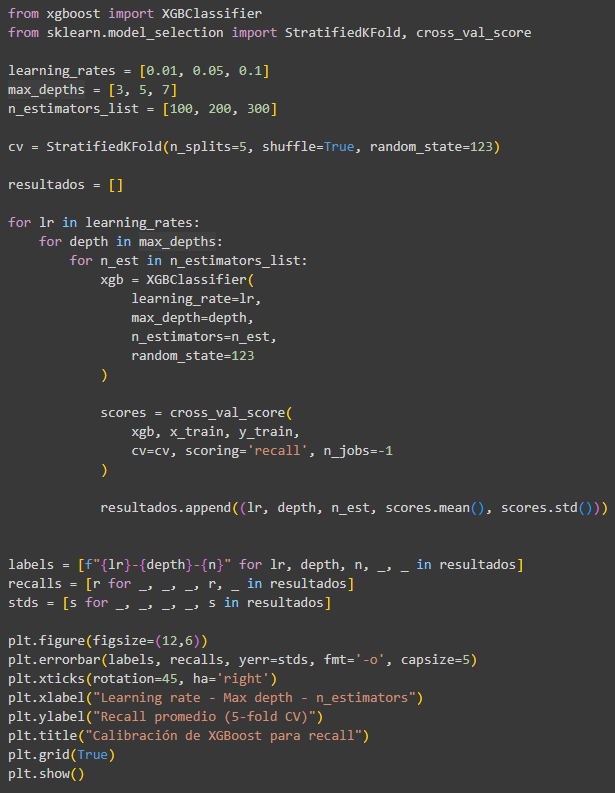
Análisis comparativo en el modelo SVM

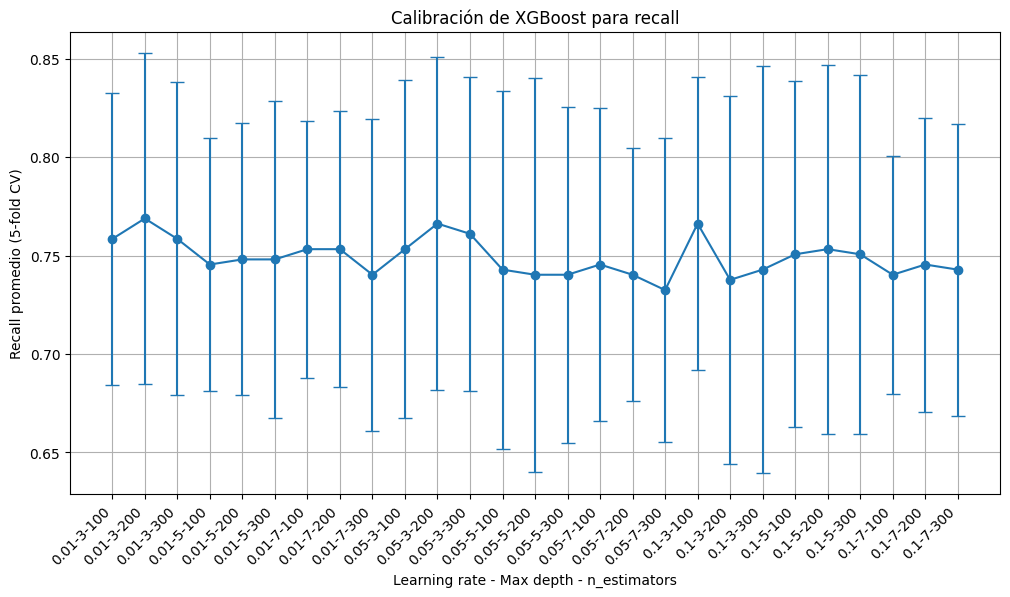
Para el modelo SVM (C=100) se observa que el recall sigue siendo alto, especialmente en el split de train/test. Además, la precisión se mantiene muy alta, aunque al poner que el modelo sea más estricto empieza a generar algunos falsos positivos al tratar de capturar todos los positivos posibles. Por otro lado, al compararlo con la evaluación con K-FOLD se nota una caída en recall, lo que indica que en algunos folds o particiones del dataset no representan perfectamente los patrones de anomalías y el modelo no puede generalizar bien. Aún así se esperaba un recall más elevado con el modelo SVM ya que este tiende a priorizar la detección de positivos frente a minimizar falsos positivos.

#### 3.1.3.5 XGBoost

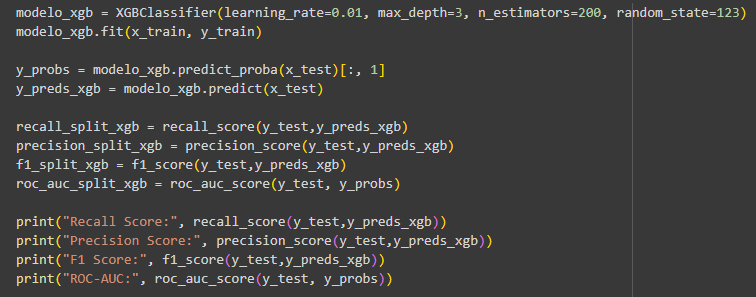
El quinto modelo es XGBoost se calibró tres de sus principales parámetros (learning\_rate, max\_depth, n\_estimators), el mejor conjunto fue: 0.01, 3, 200.

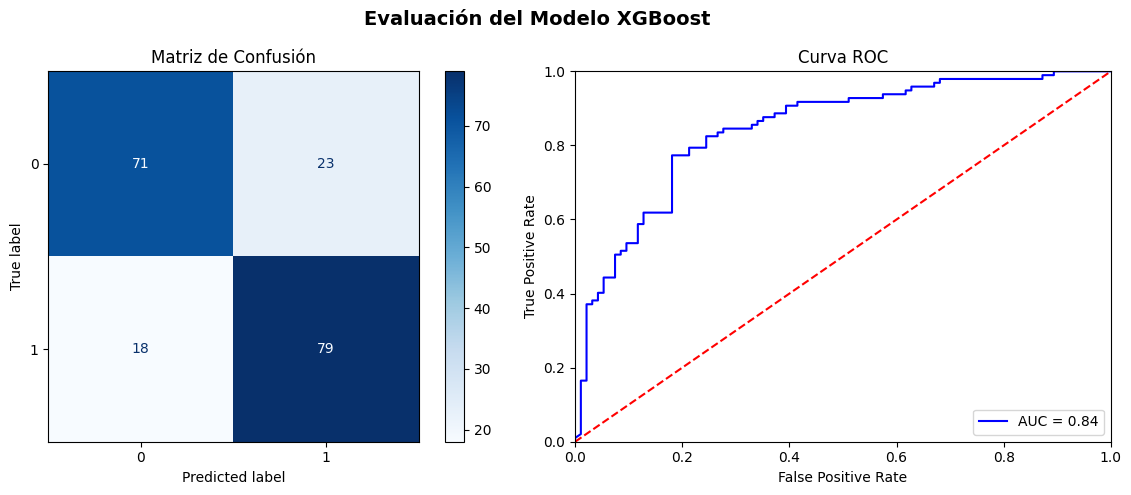






Se validó con el conjunto test.





Comparamos los resultados.

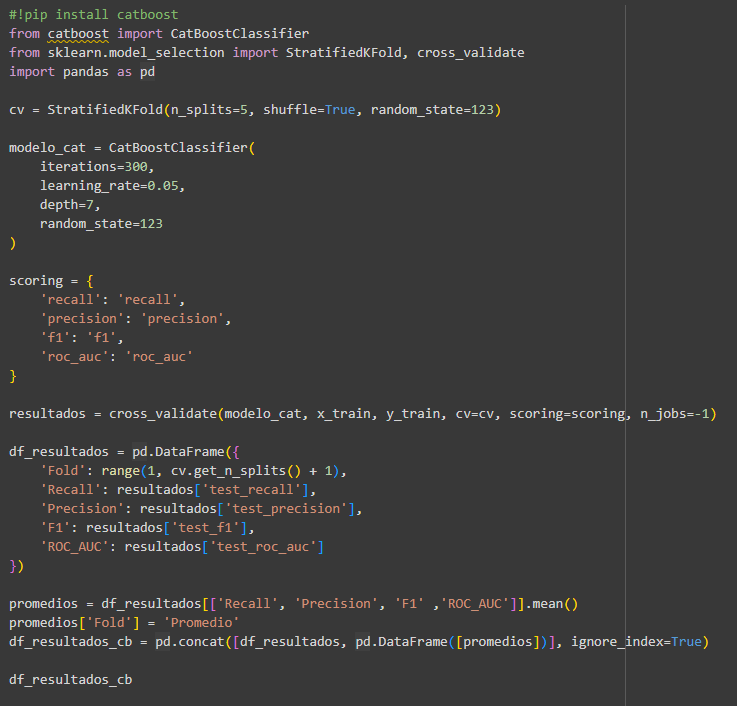
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Método | Recall | Precision | F1 | ROC\_AUC |
| Train/Test Split | 0.8144 | 0.7745 | 0.7940 | 0.8448 |
| K-Fold (5) | 0.7688 | 0.7550 | 0.7601 | 0.8230 |

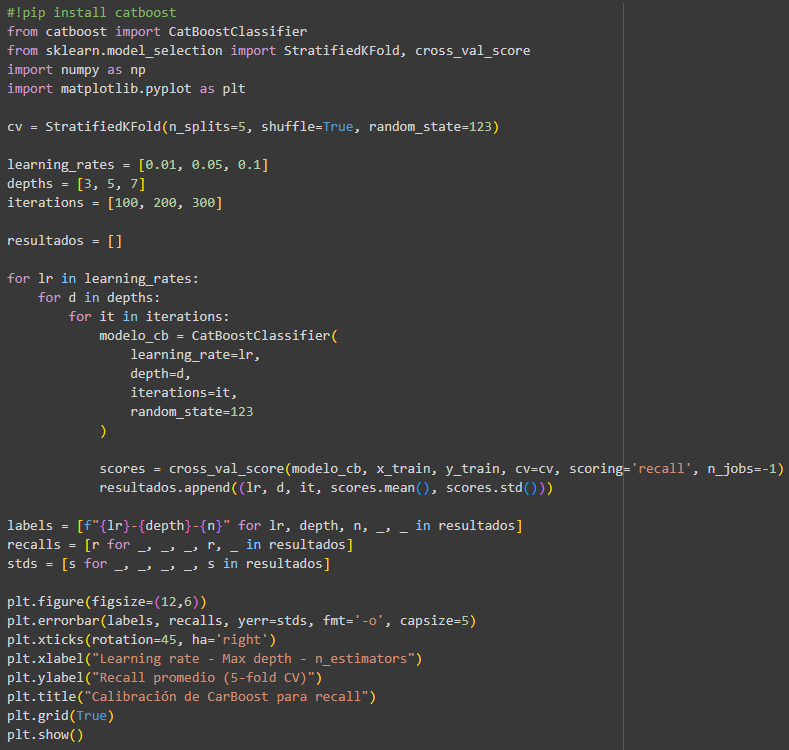
Análisis comparativo en el modelo XGBoost

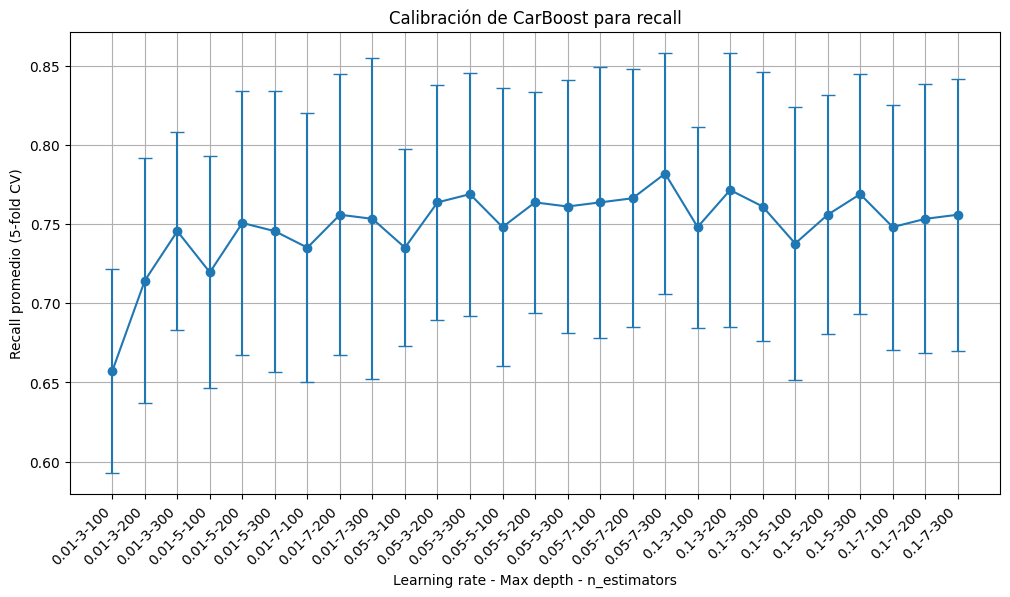
Al ver los resultados del modelo XGBoost,al evaluar mediante validación cruzada K-Fold (5 splits) observamos un recall de 0.7688, ligeramente inferior al obtenido en el train/test split (81.44%) teniendo los dos los mejores hiperparametros. Esto indica que el modelo tiene un desempeño peor en distintos subconjuntos de entrenamiento. Comparandolo con otros modelos este tiene una peor variación entre los resultados de la validación cruzada que con el train/test comparandolos con Random Forest y SVM lo que indica que tiene una tendencia a ajustar más los datos de entrenamiento debido a su naturaleza de ensamble con boosting. Su recall sigue siendo competitivo y superior al de Regresión Logística y KNN, mientras que la precisión y F1 se mantienen equilibradas. Entonces XGBoost ofrece una buena combinación entre detección de anomalías y estabilidad general, pero hay que tener en cuenta que su rendimiento en K-Fold revela que podría estar capturando algunas relaciones muy específicas del conjunto de entrenamiento.

#### 3.1.3.6 CatBoost

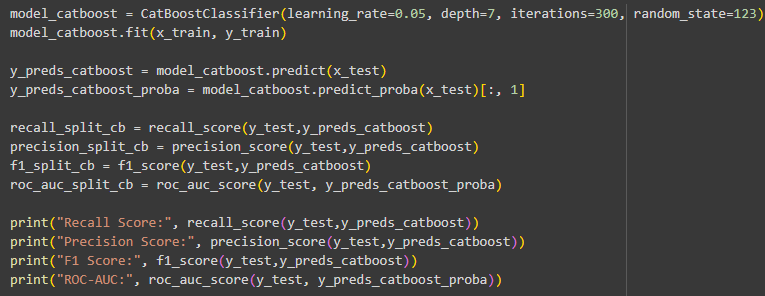
El siguiente modelo es CatBoost también se calibro de la misma manera, pero en vez del número de arboles se calibro el número de iteraciones.

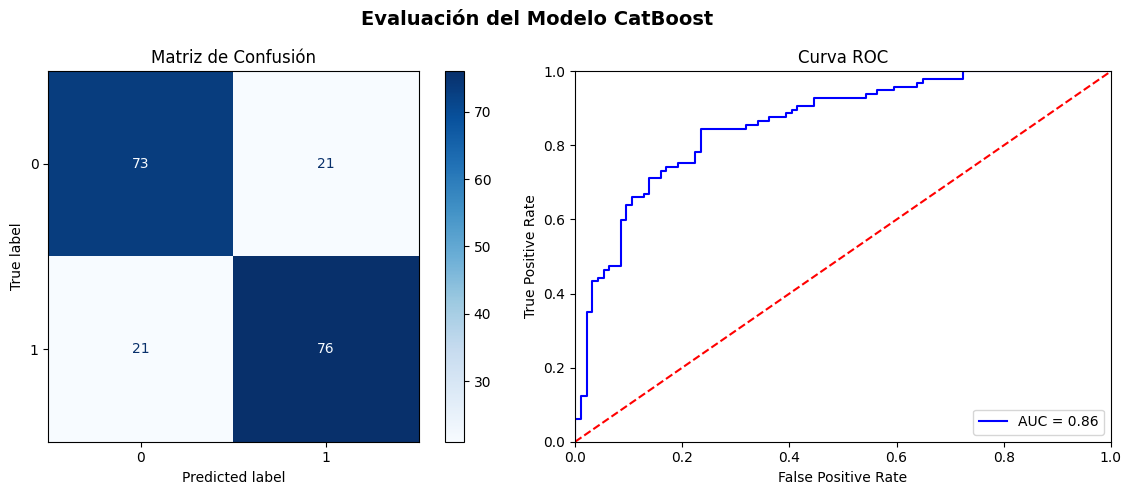






Evaluamos con el conjunto de test.





Comparamos los resultados.

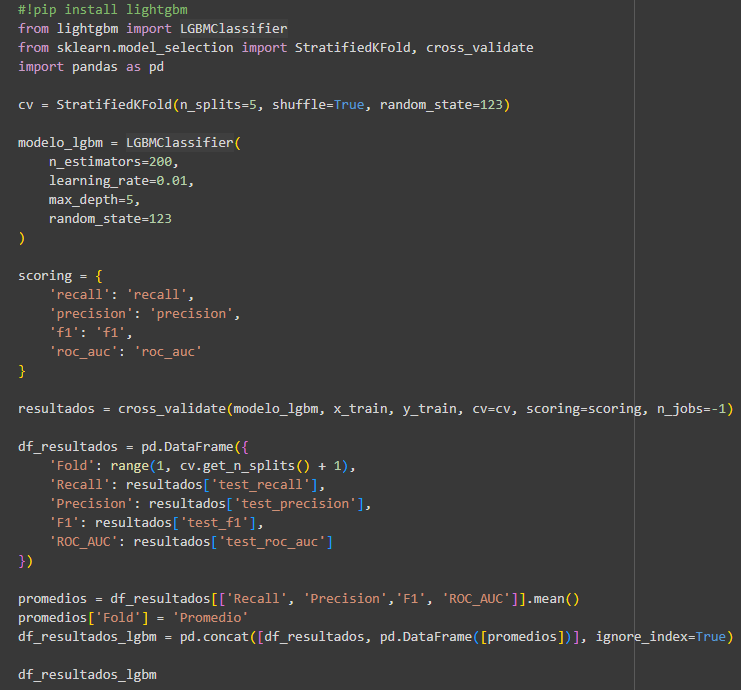
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Método | Recall | Precision | F1 | ROC\_AUC |
| Train/Test Split | 0.7835 | 0.7835 | 0.7835 | 0.8596 |
| K-Fold (5) | 0.7818 | 0.7543 | 0.7661 | 0.8323 |

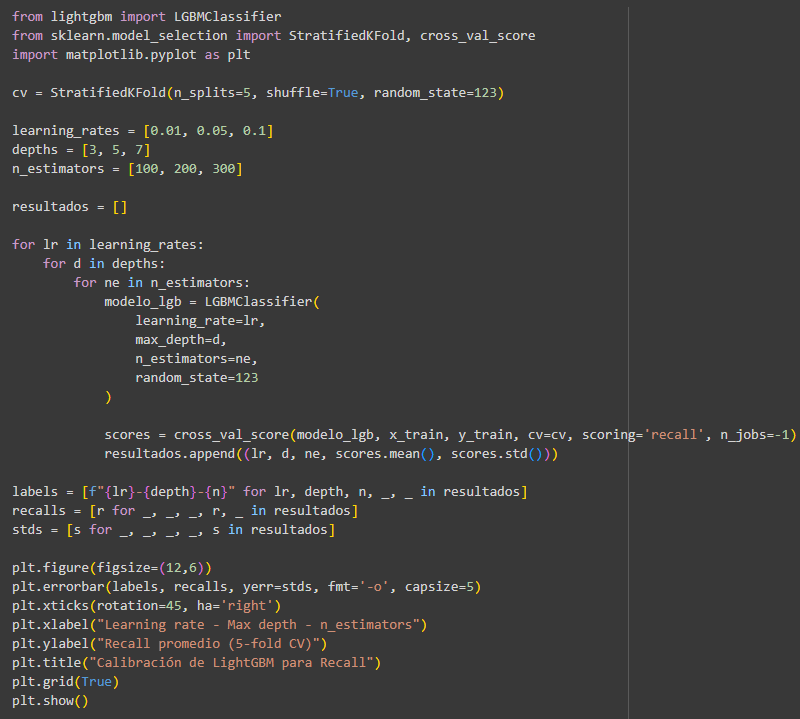
Análisis comparativo en el modelo CatBoost

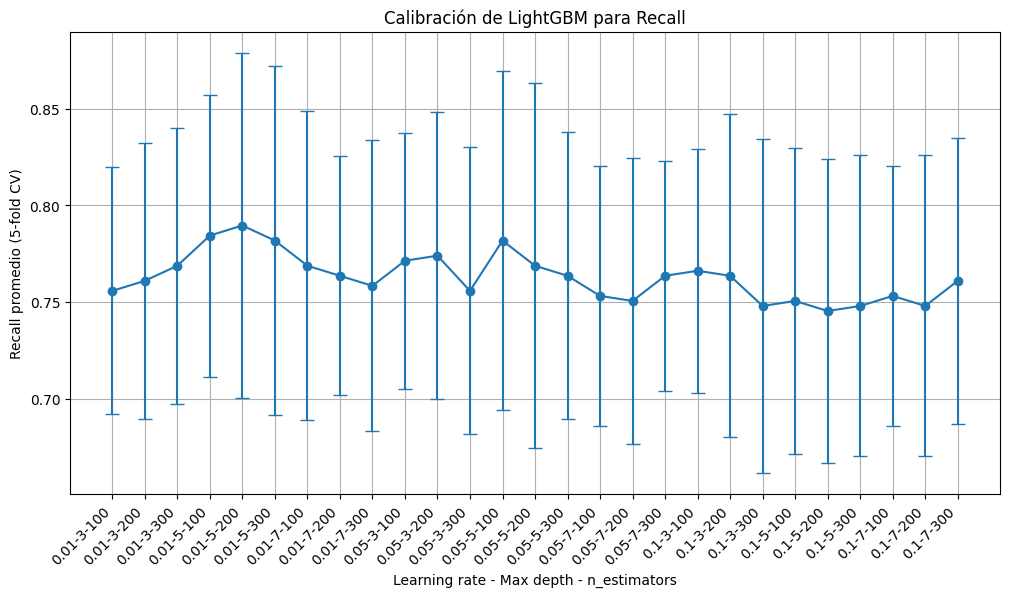
En el modelo CatBoost, las métricas del K-Fold son ligeramente menores que las del Train/Test Split, lo que indica que la validación cruzada ofrece una estimación más realista de la generalización. Aunque el Recall se mantiene alto (0.7818), la Precision baja un poco (0.7543), lo que refleja que el modelo sigue detectando correctamente la mayoría de anomalías, pero con un leve aumento de falsos positivos. El F1 y el ROC-AUC muestran que CatBoost logra un buen equilibrio entre sensibilidad y capacidad de discriminación, consolidándolo como el modelo más sólido comparándolo con todos los demás modelos anteriores.

#### 3.1.3.7 LightGBM

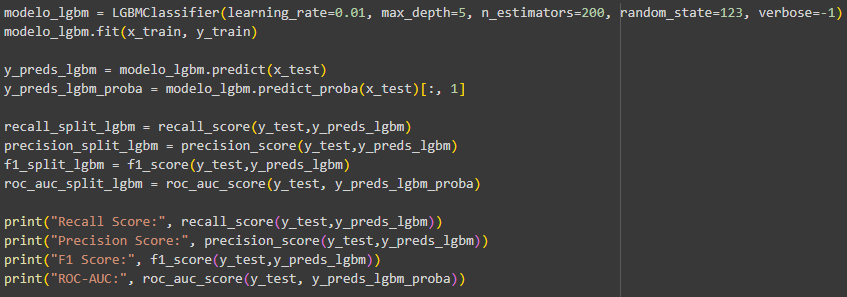
El último modelo fue LightGBM que fue elegido más por su rapidez y eficacia para este tipo de problemas de clasificación binaria. Calibrado de la misma manera que los modelos anteriores.

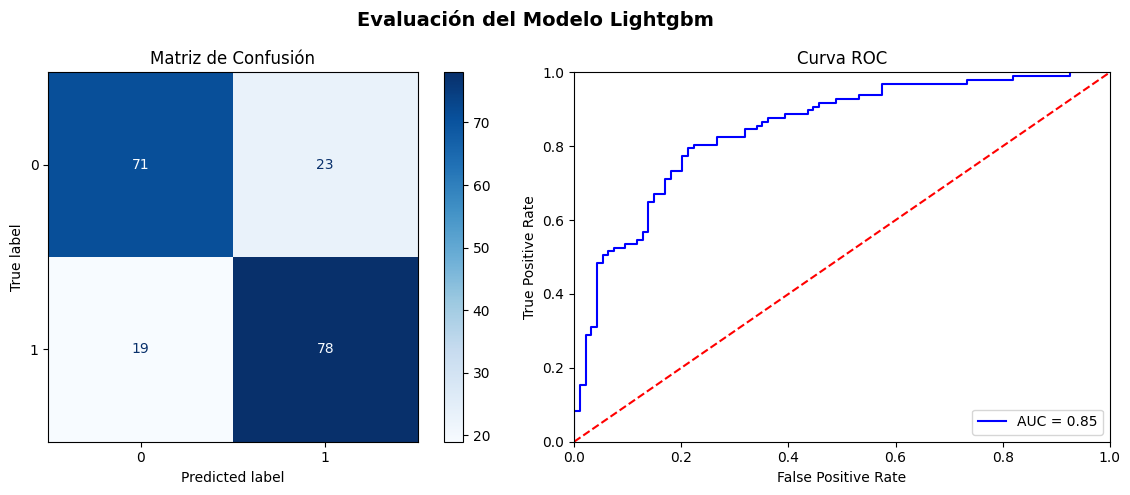






Evaluamos con el conjunto test.





Comparamos los resultados.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Método | Recall | Precision | F1 | ROC\_AUC |
| Train/Test Split | 0.8041 | 0.7723 | 0.7879 | 0.8460 |
| K-Fold (5) | 0.7896 | 0.7608 | 0.7735 | 0.8426 |

Análisis comparativo en el modelo LightGBM

En LightGBM, se observa que las métricas del K-Fold son ligeramente menores que las del Train/Test Split, indicando que la validación cruzada proporciona una estimación más conservadora de la generalización. El Recall alto (0.7896) muestra que el modelo sigue detectando la mayoría de las anomalías, mientras que la Precision (0.7608) refleja un pequeño aumento de falsos positivos. El F1 y ROC-AUC sugieren que LightGBM mantiene un buen equilibrio entre sensibilidad y capacidad de discriminación, siendo un modelo sólido incluso con el anteriormente evaluado CatBoost.

### 3.1.4. Conclusiones del aprendizaje supervisado

Se colocan todos los resultados de todos los modelos de cada validación.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Modelo | Recall K-Fold | Precision K-Fold | F1 K-Fold | ROC-AUC K-Fold | Recall Test | Precision Test | F1 Test | ROC-AUC Test |
| Random Forest | 0.7662 | 0.7678 | 0.7652 | 0.8407 | 0.7732 | 0.7575 | 0.7653 | 0.8545 |
| Regresión Logística | 0.7584 | 0.7589 | 0.7579 | 0.8219 | 0.7732 | 0.7576 | 0.7653 | 0.8292 |
| KNN | 0.8104 | 0.7035 | 0.7523 | 0.7997 | 0.7835 | 0.7308 | 0.7562 | 0.8434 |
| SVM | 0.7688 | 0.7570 | 0.7626 | 0.8226 | 0.7938 | 0.7549 | 0.7739 | 0.8208 |
| XGBoost | 0.7688 | 0.7550 | 0.7600 | 0.8231 | 0.8144 | 0.7745 | 0.7940 | 0.8448 |
| CatBoost | **0.7818** | **0.7543** | **0.7662** | **0.8324** | **0.7835** | **0.7835** | **0.7835** | **0.8596** |
| LightGBM | 0.7896 | 0.7608 | 0.7735 | 0.8426 | 0.8041 | 0.7723 | 0.7879 | 0.8460 |

**¿Cuál es el mejor modelo?**

Después de haber evaluado los siete modelos anteriormente propuestos bajo la validación cruzada y pruebas finales sobre el conjunto de test, se puede observar que el modelo con mejor desempeño en general fue el de K-Nearest Neighbors (KNN) el cual obtuvo el mayor recall promedio con 0.8104, seguido de Random Forest y LightGBM. Lo malo de KNN es que tiene un ROC-AUC demasiado bajo comparado con los demás modelos lo que se interpreta que no clasifica bien las clases comparado con otros algoritmos.

Por otro lado, LightGBM y Random Forest tienen un mejor equilibrio global: LightGBM alcanzó valores altos en recall (0.7896) como en ROC-AUC (0.8426), lo mismo sucede con Random Forest que obtuvo metricas similares (recall = 0.7662, ROC-AUC = 0.8407).

Cabe aclarar que comparamos los modelos con el promedio de las metricas obtenidos de la validación cruzada ya que estos datos me permiten evaluar al modelo de manera más estable. Ya que a diferencia de un único split de entrenamiento y prueba, la validación cruzada reduce la influencia al azar en la partición de los datos y proporciona una estimación más confiable del desempeño real del modelo. La validación final con el test es más que nada para ver si el modelo no generaliza o si hubo sobreajuste con los datos del entrenamiento.

Luego en la validación con el conjunto de datos de test, LightGBM tuvo un excelente rendimiento general (recall = 0.8041, F1 = 0.7879, ROC-AUC = 0.8460), esto confirmando su capacidad de generalización sin una caída significativa de desempeño. Por su parte, aunque el modelo CatBoost obtuvo el mayor ROC-AUC en la evaluación final con el conjunto test (0.8596), su recall promedio en la validación cruzada fue menor (0.7818), lo que lo hace menos consistente con la métrica principal.

Por ende, LightGBM con sus hiperparametros calibrados lo elegimos como el mejor modelo, ya que ofrece un equilibrio óptimo entre todas las métricas usadas, con un rendimiento estable tanto en la validación cruzada como en la evaluación final. Estos datos lo convierten en la opción más adecuada para nuestro problema de detección de frames anómalos, donde la clave es detectar la mayor cantidad posible de eventos inusuales manteniendo un leve control con las falsas alarmas.

## 3.2 - Implementación de algoritmo no supervisado

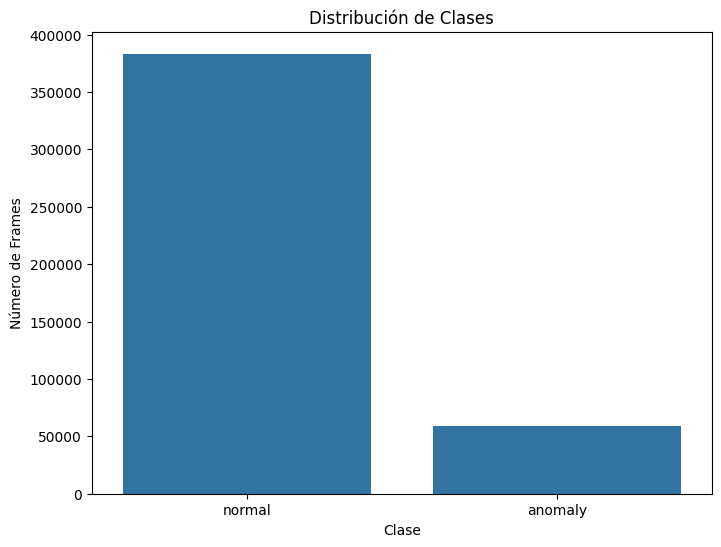
Durante la primera fase del trabajo se evaluó el desempeño de distintos modelos sobre el dataset resumido a nivel de video con un total de 957 observaciones, en el cual cada registro representaba características agregadas del conjunto de frames pertenecientes a un mismo clip. Si bien este enfoque permitió obtener una visión general del comportamiento promedio de cada video, se identificaron limitaciones importantes para la detección precisa de anomalías.

Inicialmente, se realizaron pruebas usando el dataset a nivel de video, pero los resultados obtenidos mostraron un recall demasiado bajo, incluso tras ajustar los principales hiperparametros del modelo. Esto debido a que el dataset de videos contenía pocas observaciones y a que la agregación de los valores por clip diluía la presencia de comportamientos anómalos locales dentro de la información general.

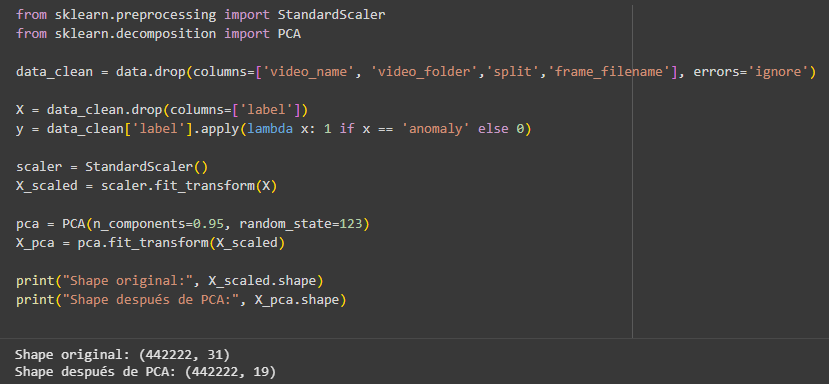
Entonces se decidió migrar el análisis al nivel de frame individual, de este modo cada frame se considera una observación independiente, lo que permite preservar el detalle de cada uno de las escenas. Este cambio de enfoque aumenta significativamente la cantidad de datos disponibles (de solo tener menos de mil videos a más de 400 000 frames), dando al modelo una mayor base para aprender patrones normales y detectar outliers con más precisión.

### 3.2.1. Isolation Forest

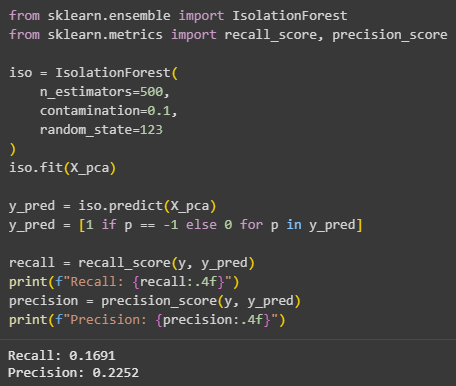
Se uso el dataset de frames y la distribución de clases en este dataset esta muy desbalanceado y se probó si esto afectaba en algo a los resultados que podríamos a obtener. Tenemos 383325 frames normales y 58897 anormales. Esto debido a que el dataset usa más videos normales que anormales para trabajar a nivel de video y aparte que los videos anómalos no contienen tantos números de frames.



Se procedió a seguir con el procesamiento de los datos, primero eliminamos las columnas que no aportan nada a la predicción que en este caso sería: “video\_name”, “video\_folder”, “split”, “frame\_filename”. Variables que solo permiten identificar al frame si es que se esta buscando en el dataset. Luego eliminamos nuestra variable objetivo ya que no la vamos a usar para entrenar debido a que estamos usando aprendizaje no supervisado. Después escalamos los datos con StandarScaler para luego usar PCA con esos datos escalados para la reducción de la dimensionalidad, esto para reducir el número de variables que puedan mostrar la mejor información. De las 31 variables que teníamos PCA las redujo a solo 19 y con esto teníamos los datos procesados para el entrenamiento de los algoritmos no supervisados.

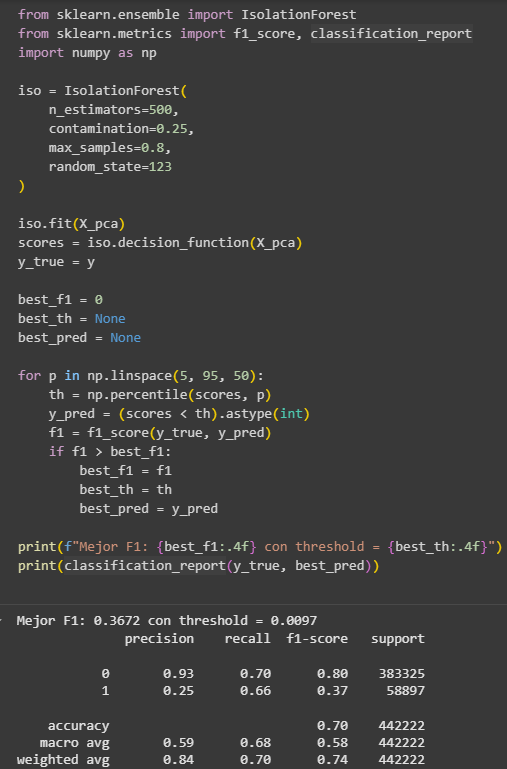


Se entreno con Isolation Forest tratando de maximizar el recall que es el propósito del caso de uso de videovigilancia.

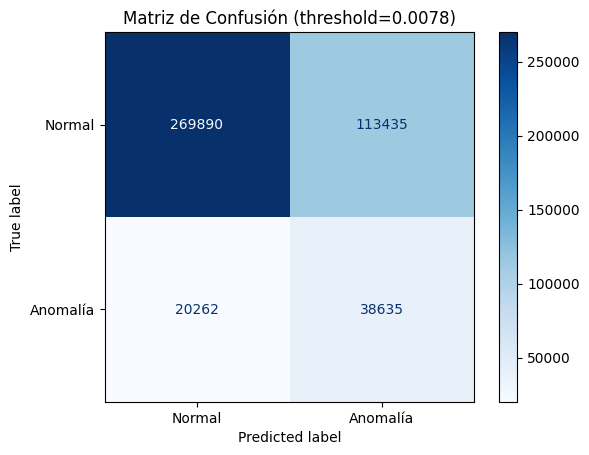


Como se puede observar el recall de anomalías se pudo llegar a 17% con el hiperparametro “contamination” en un 10% eso quiere decir que le estamos diciendo que el 10% de los datos son anómalos aproximadante.

Al ver que el recall no era muy alto se tuvo que calibrar los hiperparametros, pero esta vez para la métrica F1 la cual es un balance entre recall y precision ya que por lo que se puede observar la precision también es muy baja lo que quiere decir que hay demasiadas alertas por el coste de poder predecir algunos anómalos. También se busco el mejor treshold para este mismo objetivo con la ayuda de los percentiles.



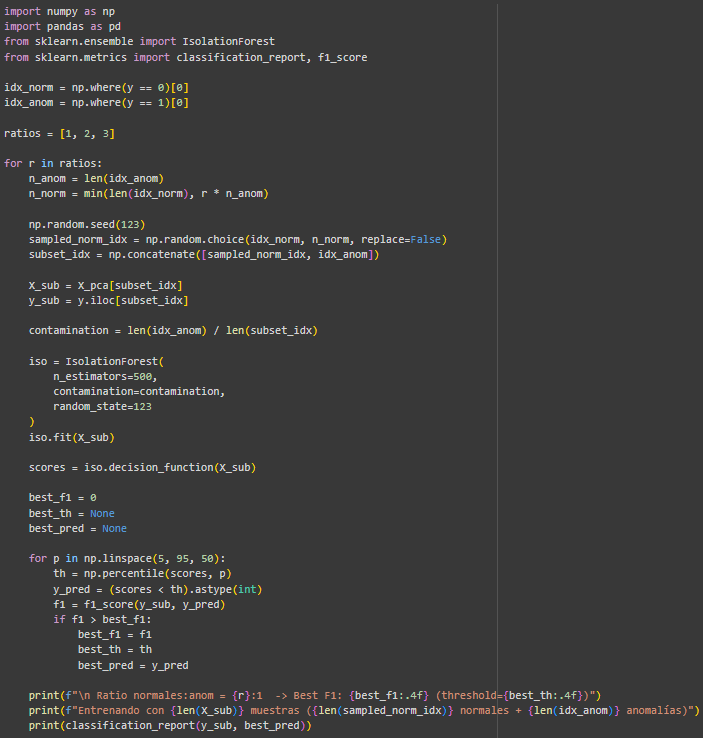
El mejor treshold encontrado para maximizar el F1 fue 0.0097 eso quiere decir que el frame con menor score que 0.0097 será considerado anomalía (1), el resto será normal (0). Se saco la matriz de confusión para este resultado con este mejor treshold y estos fueron los resultados.



Se puede observar que hay demasiadas falsas alarmas y también muchos frames anómalos son detectados como normales. Esto en un sistema de videovigilancia no sería un modelo óptimo ya que la confiabilidad del sistema se vería comprometida. Se sabe que en estos casos de uso se debe priorizar una alta sensibilidad (recall) para asegurar que la mayoría de comportamientos anómalos sean identificados, incluso si eso implica aceptar un mayor número de falsos positivos. Aún así el resultado es muy alto por lo que se probo el dataset de otra manera.

Sabemos que el dataset tiene un fuerte desbalanceo de clases, donde la cantidad de frames normales es mucho mayor que la de frames anómalos. Se considera que esta desproporción podría estar afectando el rendimiento del modelo, ya que el Isolation Forest tiende a ajustarse más hacia la clase mayoritaria. Es por ello que se implementó una función experimental cuyo objetivo es evaluar cómo cambia el desempeño del modelo cuando se reduce la cantidad de frames normales. Esta función divide el dataset y realiza pruebas bajo diferentes proporciones entre normales y anómalos (por ejemplo, 1:1, 2:1 y 3:1). Tambien encuentra al mejor treshold de cada uno de los desempeños.

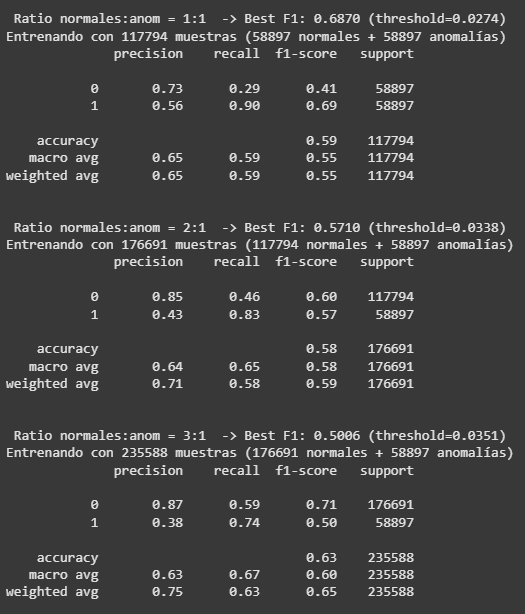
En cada caso, se entrena el modelo únicamente con una muestra balanceada de los datos y luego se evalúa su desempeño sobre el conjunto completo.



A continuación, se muestran los resultados obtenidos al aplicar la función antes mostrada para diferentes proporciones entre frames.

Como se observa, cuando el dataset se balancea completamente (ratio 1:1), el modelo alcanza su mayor recall (90%), lo cual indica que logra detectar la mayoría de los frames anómalos. Sin embargo, esto ocurre a costa de un descenso de precision, generando una cantidad de falsas alarmas como era de esperarse.

Luego al incrementar la proporción de frames normales (ratios 2:1 y 3:1), la precisión mejora gradualmente, pero el recall disminuye. Esto confirma que el balance de clases influye directamente en el desempeño del Isolation Forest, afectando la sensibilidad del modelo frente a eventos anómalos.



Para esta parte del trabajo se implementó un enfoque no supervisado para la detección de anomalías en video utilizando el algoritmo Isolation Forest trabajado con los features que devuelve YOLO, tomando como referencia el estudio “Anomaly Detection in Video Surveillance Using YOLOv8”. Si bien en dicho reporte se consigue un desempeño sobresaliente sobre un conjunto de clips preprocesados y equilibrados, en este caso el dataset empleado presentó un fuerte desbalanceo de clases, con una porporción aproximada de 6.5:1 entre frames normales y anómalos.

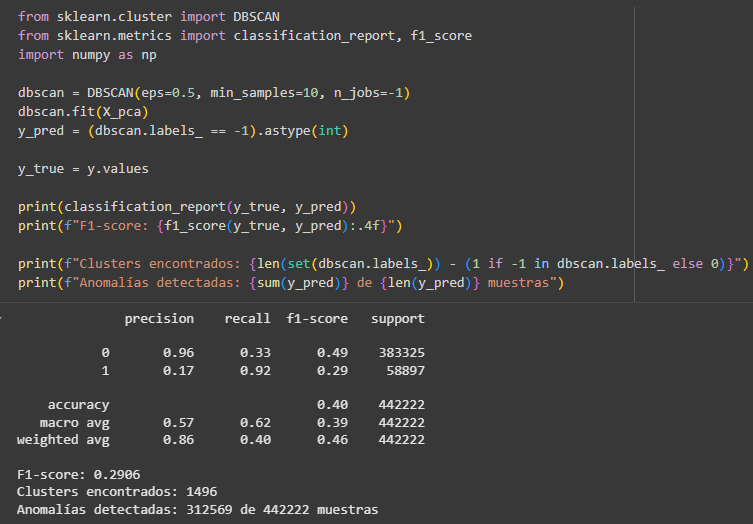
Con estas pruebas se pudo demostrar que, aunque Isolation Forest mostró cierto potencial bajo condiciones balanceadas, su rendimiento en escenarios reales (donde predominan los eventos normales) resulta limitado. Es por ello que en el paper de referencia recomienda estrategias complementarias como reajuste de umbrales, muestreo balanceado o el uso de modelo híbridos supervisados.

### 3.2.2. DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

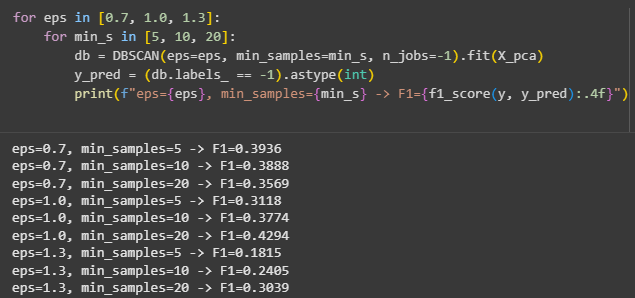
Se empleará el algoritmo DBSCAN con el propósito de detectar patrones inusuales en las características obtenidas por YOLO. A diferencia de Isolation Forest, DBSCAN permite identificar grupos de frames similares según su densidad en el espacio de características, esto sin requerir especificar previamente el número de grupos. Este enfoque se centra en utilizar DBSCAN como una herramienta no supervisada para separar los frames que presentan comportamientos normales de los posibles frames anómalos, que se ubican en zonas menos densas y son tratados como ruido por el modelo.

Al aplicar DBSCAN en este contexto es evaluar si este método logra una mejor identificación de frames anómalos en comparación con modelos previos como Isolation Forest, especialmente considerando el fuerte desbalanceo de clases presente en el dataset y la naturaleza no lineal de los datos extraídos mediante PCA.

Se intento primero con un conjunto de parámetros iniciales (eps = 0.5 y min\_samples = 10) como punto de partida para observar el comportamiento del algoritmo sobre el dataset de frames. Estos valores permiten evaluar si el método logra identificar correctamente grupos densos de frames normales y distinguir aquellos considerados anómalos. A partir de estos resultados iniciales, se puede ajustar gradualmente los parámetros para mejorar la detección de anomalías.

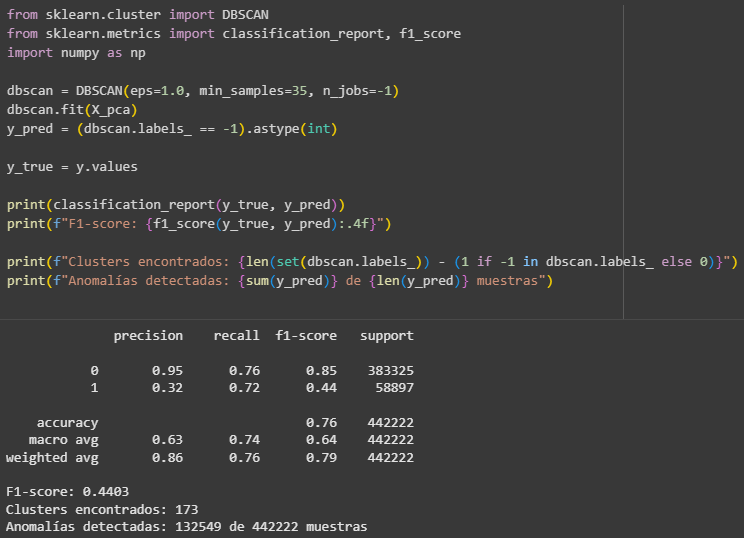


Como se puede observar con estos primeros parámetros iniciales, se observa un bajo desempeño general del modelo, con un F1-score de 0.29. Por lo que se puede ver el algoritmo logra detectar una gran cantidad de anomalías (más de 300 000), pero la mayoría corresponden a falsas alarmas, por lo que se ve con una alta precision para la clase normal pero un recall muy alto en la clase anómala.



Se realizó una calibración de hiperparametros del algoritmo DBSCAN, variando los valores de eps y min\_samples para analizar cómo afectan el desempeño del modelo en la detección de frames anómalos. En los resultados se puede ver que el rendimiento varía significativamente según la configuración: valores de eps muy bajos o muy altos tiende a empeorar el desempeño, caso contrario con configuraciones intermedias logran un mejor equilibrio.

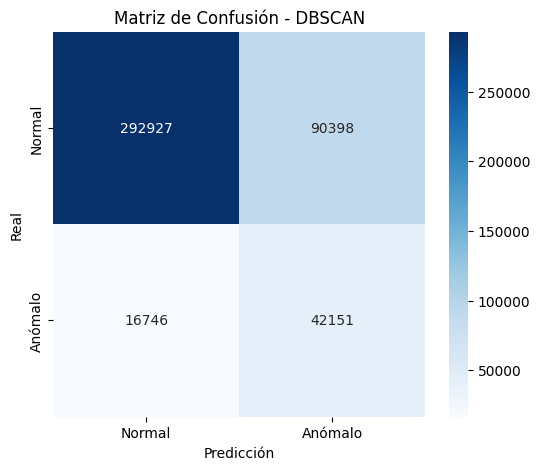
El mejor resultado se obtuvo con eps = 1.0 y min\_samples = 20, alcanzando un F1-score de 0,43, lo que indica una mejora para distinguir entre frames normales o anómalos.



A partir de los resultados anteriores se empezó fijando el valor de eps = 1.0 y variando el parámetro min\_samples con el objetivo de mejorar el rendimiento del modelo. Este parámetro controla cuántos puntos se necesitan para formar un clúster, por lo que ajustar su valor permite modificar la sensibilidad del algoritmo frente a las regiones densas o dispersas de los datos.

El mejor resultado se obtuvo con min\_samples = 35, alcanzando un F1-score de 0,44 y un recall de 0,72, luego de haber probado desde el valor 20 hasta el 50 pero por la falta de memoria su tuvo que realizar manualmente.

Esto sugiere que el algoritmo DBSCAN es capaz de adaptarse de manera más efectiva cuando se ajusta cuidadosamente la densidad mínima requerida, aunque aún enfrenta limitaciones debido al fuerte desbalance y la complejidad intrínseca del dataset de videovigilancia.



Esta es la matriz de confusión con los mejores hiperparametros obtenidos, ahí se observa la gran cantidad de frames anómalos identificados mediante las características que da YOLO. Sin embargo, también hay una gran cantidad de falsos positivos

### 3.2.3. Conclusiones del algoritmo no supervisado

Luego de haber aplicado dos de los mejores algoritmos no supervisados centrados en la detección de anomalías, Isolation Forest y DBSCAN. Ambos permitieron evaluar el comportamiento del dataset sin necesidad de etiquetas en el entrenamiento, lo que es útil en escenarios donde no se dispone de datos previamente clasificados.

Ahora hablando de cada uno de los modelos, Isolation Forest mostró un desempeño moderado, logrando identificar una cantidad razonable de frames anómalos, aunque con un recall bajo, lo que indica que muchos eventos pasaron desapercibidos. Lo que sugiere que el algoritmo no logró aislar correctamente los puntos fuera del comportamiento normal y se pudo comprobar que fue más por el desbalance de clases, ya que al equilibrar el dataset se pudo comprobar una mejora considerable en la detección de anomalías.

En el caso de DBSCAN se pudo obtener un mejor F1-score (0,44) y un recall mucho más alto siendo capaz de identificar una mayor proporción de anomalías reales, aunque con el costo de contar con un número de falsas alarmas. Aun así, este comportamiento puede considerarse aceptable en un sistema de vigilancia, donde es preferible recibir algunas alertas de más antes que dejar pasar una anomalía crítica sin detectar.

Por ende, el mejor modelo que se pudo adaptar fue DBSCAN, pero si el dataset creado mediante YOLO hubiera habido un balance de clases decente (2:1) los dos modelos hubieran sido decentes para este caso de uso.

## 3.3. Caso de estudio de algoritmo de aprendizaje profundo

Para esta sección, se implementa un enfoque de aprendizaje profundo basado en el estudio de Nazir et al. (2023), **“Suspicious Behavior Detection with Temporal Feature Extraction and Time-Series Classification for Shoplifting Crime Prevention”**. La principal innovación de este método es su capacidad para transformar un problema complejo de análisis de video en un problema de clasificación de series temporales, logrando resultados superiores a los métodos de última generación (SOTA) con una eficiencia computacional mucho mayor.

Este enfoque es directamente aplicable al contexto de Chancay, ya que su eficiencia lo hace ideal para sistemas de vigilancia en tiempo real que pueden operar con recursos de hardware limitados.

### 3.3.1. Metodología y Preparación de Datos

El método propuesto consta de dos etapas principales, diseñadas para evitar el procesamiento intensivo de píxeles que caracteriza a las redes convolucionales 3D (I3D).

1. **Etapa 1: Extracción de Características Temporales con YOLOv5 y Deep Sort**

* En lugar de analizar cada frame en su totalidad, se utiliza el modelo YOLOv5 con Deep Sort para detectar y rastrear a cada persona a través de la secuencia de video.
* Para cada persona identificada en cada frame, el sistema extrae las coordenadas de su cuadro delimitador (*bounding box*): Left, Top, Width, y Height.
* Esta información se almacena en un formato tabular, donde cada fila corresponde a la detección de una persona en un frame específico, convirtiendo efectivamente el video en un conjunto de datos de series temporales.

1. **Etapa 2: Clasificación con Modelos de Deep Learning para Series Temporales**

* El dataset tabular generado se utiliza para entrenar modelos de aprendizaje profundo especializados en la clasificación de secuencias temporales.
* Estos modelos aprenden a identificar patrones en los movimientos y posiciones de las personas (representados por las coordenadas de los *bounding boxes*) que se correlacionan con comportamientos "normales" o "anómalos".

### 3.3.2. Dataset y Pre-Procesamiento

En esta sección se detalla la creación y preparación práctica del conjunto de datos. **Sí se utiliza código** para ejecutar dos tareas clave: la generación del dataset a partir de los videos y su posterior transformación a un formato compatible con los modelos de series temporales.

Dado que ya contamos con el archivo dataframe\_frames.csv, el primer paso conceptual de extracción de características de los videos ya se considera completado. El código relevante se enfoca en la preparación de este archivo para el modelado.

**Transformación a Formato de Series Temporales (3D)**

Este es el primer paso práctico con nuestro dataset dataframe\_frames.csv. El siguiente código lo carga y lo transforma en el array 3D requerido por los modelos de Deep Learning, utilizando las 9 características seleccionadas.

Texto

El contenido generado por IA puede ser incorrecto.

### 3.3.3. Modelos de Deep Learning Evaluados

El estudio evaluó cuatro arquitecturas de aprendizaje profundo de última generación para la clasificación de series temporales, disponibles en la librería **TSAI**.

* **InceptionTime:** Inspirado en la arquitectura Inception de Google para imágenes, utiliza múltiples filtros convolucionales de diferentes tamaños en paralelo para capturar patrones temporales a diversas escalas.
* **XceptionTime:** Una evolución de InceptionTime que emplea convoluciones separables en profundidad (*depthwise separable convolutions*) para reducir el número de parámetros y mejorar la eficiencia del entrenamiento sin sacrificar el rendimiento.
* **MiniRocket:** Un modelo extremadamente rápido que utiliza una gran cantidad de núcleos convolucionales (*kernels*) generados de forma casi determinista para transformar las series temporales en un nuevo conjunto de características, que luego se utilizan para entrenar un clasificador lineal.
* **XCM (Explainable Convolutional Network):** Una arquitectura que aplica convoluciones 1D y 2D simultáneamente para extraer características tanto temporales como entre las diferentes variables (en este caso, las coordenadas del *bounding box*), diseñada también para ser interpretable.

### 3.3.4. Entrenamiento y Evaluación del Modelo

En esta sección se detalla la fase de entrenamiento y evaluación del modelo XceptionTime. Este proceso no consiste simplemente en ejecutar un comando, sino en aplicar una serie de técnicas de vanguardia en el campo del Deep Learning, encapsuladas eficientemente por la librería tsai, para garantizar un entrenamiento robusto y obtener resultados fiables.

El concepto fundamental de esta fase se puede resumir en tres componentes clave que el código implementa:

1. **El Proceso de Entrenamiento: Mini-Lotes y Descenso de Gradiente**

Entrenar un modelo de aprendizaje profundo con un dataset completo de una sola vez es computacionalmente inviable. Por ello, se utiliza una técnica fundamental llamada **Descenso de Gradiente por Mini-Lotes (Mini-Batch Gradient Descent)**. Los **DataLoaders** son los encargados de implementar esta técnica: dividen el conjunto de datos en pequeños lotes o *batches* (en nuestro caso, de 32 secuencias cada uno) y los alimentan al modelo de forma secuencial. En cada lote, el modelo calcula su error y ajusta sus parámetros (pesos) ligeramente para mejorar. Repetir este proceso miles de veces permite al modelo aprender los patrones subyacentes en los datos.

1. **El Ecosistema tsai/fastai: El Objeto Learner**

El código utiliza un objeto Learner, una poderosa abstracción de la librería fastai sobre la que se construye tsai. El Learner actúa como un "director de orquesta" que empaqueta y gestiona todos los componentes necesarios para el entrenamiento:

* **Los Datos:** A través de los DataLoaders (dls).
* **La Arquitectura:** El modelo XceptionTime que hemos elegido.
* **La Función de Pérdida:** Un mecanismo interno que calcula cuán "equivocadas" están las predicciones del modelo.
* **El Optimizador:** El algoritmo que actualiza los pesos del modelo para minimizar el error.

El uso del Learner simplifica enormemente el código y, lo que es más importante, incorpora automáticamente las mejores prácticas de la industria para el entrenamiento de redes neuronales.

1. **Optimización del Aprendizaje: La Política fit\_one\_cycle**

En lugar de un entrenamiento estándar con una tasa de aprendizaje fija, se utiliza la política "One Cycle" (fit\_one\_cycle), una técnica de vanguardia propuesta por el investigador Leslie N. Smith. Este método ajusta dinámicamente la tasa de aprendizaje durante el entrenamiento en dos fases:

* **Fase de Calentamiento:** La tasa de aprendizaje aumenta gradualmente desde un valor bajo hasta un máximo definido. Esto permite al modelo explorar el "paisaje de la pérdida" de forma más amplia al principio, evitando quedar atrapado en soluciones subóptimas.
* **Fase de Enfriamiento:** La tasa de aprendizaje disminuye de forma constante. Esto permite que el modelo converja de manera suave y precisa hacia un mínimo profundo y estable, afinando sus pesos para lograr una mayor precisión.

Esta política a menudo resulta en una convergencia más rápida y un mejor rendimiento final del modelo.

1. **Código de Implementación**

El siguiente script ejecuta el ciclo completo de división de datos, entrenamiento y evaluación, aplicando los conceptos mencionados.

Texto

El contenido generado por IA puede ser incorrecto.

### 3.3.5. Resultados y Comparativa

El entrenamiento del modelo XceptionTime se completó exitosamente a lo largo de 15 épocas. El progreso del entrenamiento mostró una tendencia estable, donde la pérdida en el conjunto de validación (valid\_loss) disminuyó consistentemente, mientras que la exactitud (accuracy) y el F1-Score aumentaron, confirmando un proceso de aprendizaje correcto.

La evaluación final del modelo sobre el conjunto de prueba (580 secuencias no vistas previamente) arrojó los siguientes resultados, que se resumen en el reporte de clasificación:

**Métricas de Rendimiento Final del Modelo XceptionTime**

**Pantalla de computadora con letras

El contenido generado por IA puede ser incorrecto.**

**Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.**

**Análisis de Resultados**

El modelo logró una exactitud (accuracy) general del 77%, un resultado muy sólido que valida la efectividad del enfoque de series temporales para esta tarea. El análisis detallado por clase revela un comportamiento interesante:

* **Excelente Rendimiento en la Clase 'Normal':** El modelo es particularmente bueno identificando comportamientos normales, alcanzando un **F1-Score de 0.81**. Esto es muy valioso en un sistema de vigilancia real, ya que significa que el sistema puede filtrar de manera fiable la gran mayoría de las actividades no sospechosas, reduciendo el ruido y permitiendo que los operadores se centren en las alertas relevantes.
* **Rendimiento Equilibrado en la Clase 'Anomaly':** Para la detección de anomalías, el modelo obtuvo una **precisión de 0.72** y un **recall de 0.71**. Esto indica un rendimiento bien equilibrado:
* Cuando el sistema genera una alerta de anomalía, es correcta el 72% de las veces.
* El sistema es capaz de detectar el 71% de todas las anomalías reales que ocurren.
* **Comparativa con el Paper de Referencia:** Si bien los resultados del estudio original fueron superiores (>90% F1-Score), nuestro modelo con un 77% de exactitud demuestra que la metodología es robusta y generalizable a diferentes conjuntos de datos y características. La diferencia puede atribuirse a la especificidad y complejidad de nuestro dataframe\_frames.csv, así como a los hiperparámetros de la implementación.

### 3.3.6. Conclusiones del Caso de Estudio

La implementación de este caso de estudio ha sido un éxito y proporciona conclusiones fundamentales para el proyecto de seguridad en el distrito de Chancay.

1. Validación Exitosa de la Metodología: Se ha demostrado de forma práctica y con resultados cuantitativos que la estrategia de transformar datos de video en series temporales para su clasificación con redes neuronales es altamente efectiva. El modelo fue capaz de aprender patrones complejos a partir de las trayectorias de los individuos, logrando una capacidad predictiva significativa.
2. Viabilidad para una Aplicación Real: Un sistema con un 77% de exactitud y, sobre todo, una gran fiabilidad para descartar comportamientos normales (81% F1-Score), es una herramienta de gran valor operacional. Permitiría a un equipo de seguridad enfocar su atención de manera mucho más eficiente, investigando alertas que tienen una alta probabilidad de ser relevantes.
3. Potencial de Optimización: El rendimiento equilibrado en la detección de anomalías (72% de precisión y 71% de recall) ofrece una base sólida. Para una implementación en Chancay, se podría ajustar el umbral de decisión del modelo para priorizar una de las métricas según las necesidades operativas:

* **Aumentar el Recall:** Para asegurar que se detecte el mayor número posible de incidentes, incluso a costa de algunas falsas alarmas adicionales.
* **Aumentar la Precisión:** Para minimizar al máximo las falsas alarmas y asegurar que cada alerta sea de muy alta confianza.

1. **Confirmación de la Relevancia del Estudio:** Este caso de estudio valida que las técnicas de aprendizaje profundo, aplicadas de forma eficiente como en este enfoque de series temporales, son una solución moderna, potente y práctica para los desafíos de la videovigilancia. El modelo no solo es eficaz, sino también computacionalmente más rápido que los métodos tradicionales, lo que lo alinea perfectamente con los objetivos de innovación y optimización de la seguridad en el distrito.