Universidad Internacional de La Rioja
Escuela Superior de Ingeniería y Tecnología

Máster Universitario en Astrofísica y Técnicas de Observación en Astronomía

Modelo de identificación de tránsitos en curvas de luz de estrellas intrínsecamente variables

Trabajo fin de estudio presentado por:	Alberto Argilés / Fabio Cepeda
Tipo de trabajo:	Iniciación a la investigación en Astronomía y Astrofísica
Línea de trabajo:	Evolución estelar y dinámica galáctica Sistemas planetarios y astrobiología
Director/a:	Guillermo Torralba Elipe
Fecha:	17/07/2024

Resumen

En este proyecto de investigación se muestra el desarrollo de un modelo supervisado de machine learning a partir de los datos de los catálogos Gaia DR3, Kepler y TESS de curvas de luz de estrellas intrínsecamente variables, con el objetivo de identificar eventos de tránsito de exoplanetas ocultos en la variabilidad intrínseca de este tipo de estrellas.

Se ha trabajado con los modelos clasificadores XGBoost y Random Forest, sobre los cuales se ha evaluado su desempeño usando conjuntos de datos de las curvas de luz disponibles del catálogo Kepler. Como parte del desarrollo de estos modelos, se han realizado técnicas de preprocesamiento de datos para reducir el ruido en las señales y la variabilidad del flujo de cada curva de luz, así como un análisis de sus componentes principales. Adicionalmente, se ha evaluado cada modelo mediante la optimización de hiperparámetros y ejecución de procesos iterativos para maximizar las métricas principales de desempeño.

Los resultados para los modelos muestran que para el conjunto de validación se ha obtenido un F1-score en la clase positiva (Tránsito) de 0.83 y un accuracy de 0.81 en XGBoost. Respecto a Random Forest, se ha obtenido un F1-score de 0.82 en la misma clase y un accuracy de 0.82.

El modelo con mejores resultados para el conjunto de prueba ha sido XGBoost con un F1-score de 0.79 y un accuracy de 0.80 (sección 5.2.1). Por su parte, con Random Forest hemos obtenido un F1-score de 0.78 y un accuracy de 0.80 (sección 5.2.2).

A partir de estos resultados, se han ejecutado los mejores modelos de XGBoost y Random Forest con una nueva muestra de 1792 curvas de luz que no han sido clasificadas como objetos de interés en la misión Kepler para evaluar la capacidad de generalización de ambos modelos. Se ha obtenido una predicción de Tránsito de 11.16% y No Tránsito de 88,84% para XGBoost versus una predicción de Tránsito de 15,12% y No Tránsito de 84,88% en Random Forest.

Finalmente, se plantean los siguientes pasos, así como estrategias adicionales que sirvan como punto de partida para trabajos futuros.

Palabras clave: Gaia DR3, Kepler, TESS, exoplanetas, curvas de luz, machine learning, variabilidad estelar, XGBoost, Random Forest.

Abstract

In this research project, the development of a supervised machine learning model is presented, based on the data from the Gaia DR3, Kepler, and TESS catalogs of light curves of intrinsically variable stars. The objective is to identify exoplanet transit events hidden within the intrinsic variability of these types of stars.

The classifiers XGBoost and Random Forest have been used, and their performance has been evaluated using datasets from the available light curves in the Kepler catalog. As part of the development of these models, data preprocessing techniques have been applied to reduce noise in the signals and the variability of the flux of each light curve, as well as a principal component analysis. Additionally, each model has been evaluated through hyperparameter optimization and iterative processes to maximize the main performance metrics.

The results for the models show that for the validation set, an F1-score of 0.83 in the positive class (Transit) and an accuracy of 0.81 were obtained with XGBoost. For Random Forest, an F1-score of 0.82 in the same class and an accuracy of 0.82 were obtained.
The model with the best results for the test set was XGBoost, with an F1-score of 0.79 and an accuracy of 0.80 (section 5.2.1). On the other hand, Random Forest achieved an F1-score of 0.78 and an accuracy of 0.80 (section 5.2.2).

Based on these results, the best XGBoost and Random Forest models were run on a new sample of 1792 light curves that had not been classified as objects of interest in the Kepler mission to evaluate the generalization capability of both models. A prediction of 11.16% Transit and 88.84% No Transit was obtained with XGBoost, compared to a prediction of 15.12% Transit and 84.88% No Transit with Random Forest.

Finally, the following steps are proposed, along with additional strategies that can serve as a starting point for future work.

Keywords: Gaia DR3, Kepler, TESS, exoplanets, light curves, machine learning, stellar variability, XGBoost, Random Forest.

Organización del trabajo en grupo

El siguiente trabajo de investigación se ha realizado de manera conjunta siguiendo una metodología de trabajo coordinada en la que se han distribuido tareas de tal forma que la carga de trabajo sea equilibrada. Así mismo, las componentes relacionadas con el análisis, desarrollo y ejecución del modelo de machine learning objetivo de esta investigación se ha realizado en conjunto siguiendo una metodología agile en la cual se han llevado a cabo tareas de colaboración, iteración y retroalimentación y sobre alguna de ellas se han realizado una revisión en conjunto según necesidad, para comprobar la reproducibilidad de los resultados e identificar posibles errores.

La siguiente tabla muestra las tareas generales que componen esta investigación, si bien en las tareas que se han asignado de manera individual también ha habido participación el otro integrante del equipo no mencionado en la tabla.

Apartado de la memoria y/o Tarea asignada	Responsables
Introducción	Alberto Argilés, Fabio Cepeda
Contexto y estado del arte	Alberto Argilés, Fabio Cepeda
Objetivos de la investigación	Alberto Argilés, Fabio Cepeda
Preparación de datos y análisis de curvas de luz	Alberto Argilés
Proceso de generación del modelo de machine learning	Alberto Argilés, Fabio Cepeda
Preprocesamiento de las curvas de luz	Fabio Cepeda
Modelos de clasificación utilizando curvas preprocesadas	Alberto Argilés, Fabio Cepeda
Optimización de los modelos	Fabio Cepeda
Otros experimentos	Alberto Argilés
Resultados y análisis	Alberto Argilés, Fabio Cepeda
Conclusiones y perspectivas futuras	Alberto Argilés, Fabio Cepeda

Cabe mencionar que la comunicación y gestión de documentos, entregas y versiones de código se han gestionado a través de la plataforma colaborativa Microsoft Teams de la cual se tiene licencia a través de la UNIR.

Índice de contenidos

Índice de figuras

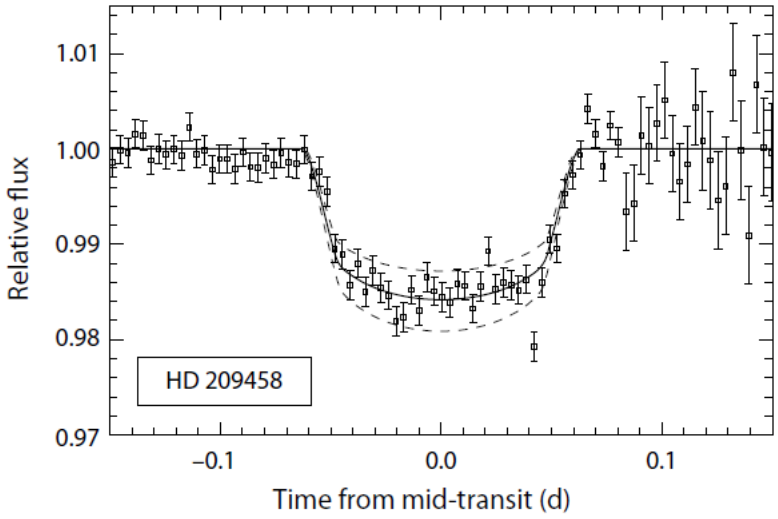
Índice de tablas

1. Introducción

El estudio de exoplanetas (planetas fuera del Sistema Solar) tiene una gran relevancia científica, ya que la observación de los planetas de nuestro Sistema Solar permite únicamente observar su estado actual, sin embargo, la investigación de exoplanetas permite estudiar estos cuerpos en distintas etapas de su evolución. En la actualidad ya se han observado exoplanetas en formación, lo cual sumado a otros estudios nos puede dar información sobre la formación de nuestro Sistema Solar (Calvet et al., 2002).

Desde el descubrimiento del primer exoplaneta, (Mayor & Queloz, 1995), y cuya curva de luz se puede ver en la Figura 1, este campo de la astronomía ha experimentado un rápido desarrollo, impulsado por avances tecnológicos y la disponibilidad de datos cada vez más abundantes y precisos.

Figura 1. *Curva de luz de tránsito del exoplaneta HD 209458 b.*



Fuente: Perryman, M. (2018). The Exoplanet Handbook. Cambridge University Press.

Entre las técnicas utilizadas para la identificación de exoplanetas, el método de tránsito se ha destacado por su eficacia, por lo que la mayor parte de los descubrimientos de exoplanetas en la actualidad se han realizado mediante esta técnica, como se muestra en la Tabla 1.

Tabla 1. *Exoplanetas confirmados al 25/06/2024*

Método de descubrimiento	Número de exoplanetas
Astrometría	3
Imagen	82
Velocidad radial	1089
Tránsito	4216
Variación de tiempo de tránsito	29
Variaciones de tiempo de eclipses	17
Microlentes	222
Variaciones de tiempo de púlsar	8
Variaciones de tiempo de pulsaciones	2
Modulaciones de brillo orbital	9
Cinemáticas de disco	1
Total	**5678**

Fuente: Adaptado de NASA Exoplanet Archive

(https://exoplanetarchive.ipac.caltech.edu/docs/counts_detail.html)

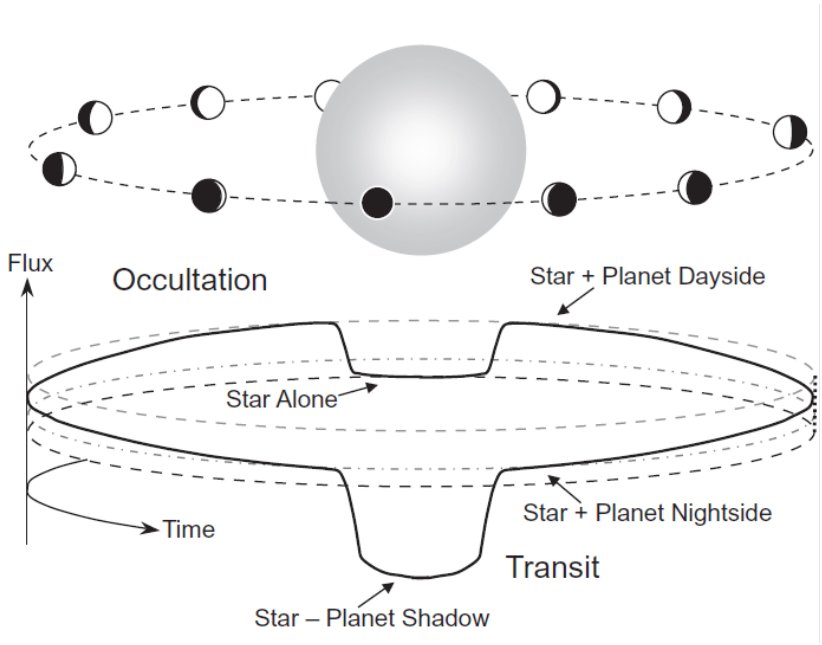
En el siguiente apartado se exponen las características principales de la técnica de descubrimiento de exoplanetas conocida como el método del tránsito.

1.1. Método del tránsito

El método de tránsito se basa en la observación de la disminución periódica en el brillo de una estrella cuando un exoplaneta pasa frente a ella, bloqueando parcialmente la luz estelar. Este fenómeno produce una curva de luz característica que puede ser registrada y analizada para inferir la presencia, tamaño, órbita y otras características del exoplaneta.

Un tránsito de un exoplaneta, también denominado eclipse primario, se presenta cuando el planeta pasa en frente de la estrella vista desde La Tierra. Por su parte, cuando el exoplaneta se oculta tras su estrella, se denomina ocultación o eclipse secundario, como se muestra en la Figura 2.

Figura 2. *Tránsito y ocultación de un exoplaneta.*



El flujo observado corresponde a la combinación de la estrella y el exoplaneta. Durante un tránsito, el flujo cae debido a que el planeta bloquea una fracción de la luz de la estrella. Así mismo, el flujo vuelve a caer cuando el planeta se oculta tras su estrella.

Fuente: Seager, S. (2011). Exoplanets. University of Arizona Press

La mayoría de los descubrimientos de exoplanetas por el método del tránsito se han realizado observando grandes números de estrellas cuyo brillo se ve disminuido periódicamente, indicando un posible tránsito de un planeta.

Un requisito fundamental del método del tránsito es que la órbita del sistema estrella-exoplaneta debe estar alineada con el observador. Dado que el tránsito de exoplanetas con periodos orbitales cortos ocurre en intervalos de tiempo pequeños, es más probable detectar este tipo de objetos en las misiones de búsqueda y descubrimiento de exoplanetas mediante el método del tránsito.

Desde la observación del primer exoplaneta por el método del tránsito realizada por Henry et al. (1999, 2000), otros sistemas descubiertos a partir de catálogos de velocidad radial también fueron monitorizados en busca de posibles tránsitos. Se establecieron rápidamente catálogos

de observaciones desde La Tierra y desde el espacio para llevar a cabo "búsquedas ciegas" de nuevos planetas a partir de sus firmas de tránsito periódico.

El método del tránsito no solamente es útil para descubrir exoplanetas. Observaciones continuas de tránsitos proveen una gran cantidad de información que no puede ser obtenida únicamente por otros métodos, como la velocidad radial. Las curvas de luz observadas pueden proporcionar una estimación del radio de un exoplaneta respecto a su estrella. Las densidades de los exoplanetas se pueden calcular a partir de su masa, lo que a su vez da una primera estimación de su composición. Además, se puede determinar la orientación de la órbita del exoplaneta respecto al eje de rotación de su estrella. Anomalías del tiempo del tránsito y otras perturbaciones detectadas durante la observación de diferentes tránsitos pueden ser indicador de otros exoplanetas o incluso lunas, además las propiedades estructurales y atmosféricas de los planetas pueden ser estudiadas más a fondo a través de fotometría y espectroscopia durante el tránsito y durante el eclipse secundario cuando el planeta pasa detrás de la estrella (Yang, 2023).

Para finales de 2010, se conocían alrededor de 70 planetas en tránsito (Díez Alonso et al., 2023), mientras que las búsquedas desde tierra y desde el espacio seguían intensificándose, y nuevas propiedades de la población planetaria, tanto individuales como estadísticas, se desplegaban rápidamente. Estas observaciones revelaron un panorama cada vez más diverso de exoplanetas, abarcando desde gigantes gaseosos hasta mundos rocosos, algunos de ellos ubicados en la denominada zona habitable (Yang, 2023).

Las observaciones desde el espacio, las cuales no están afectadas por la atmósfera terrestre, están permitiendo el descubrimiento de exoplanetas con profundidades de tránsito de tan solo unas pocas partes por millón, extendiendo las masas detectables de los exoplanetas hasta solo unas pocas masas terrestres ($M_{\oplus}$) (Batalha et al., 2013). Esta capacidad para detectar planetas de menor masa es fundamental para comprender la diversidad de sistemas planetarios, su evolución y evaluar su potencial habitabilidad.

La mayoría de los exoplanetas en tránsito se encuentran a partir de búsquedas dedicadas de ángulo amplio, que monitorean grandes números de estrellas durante largos períodos

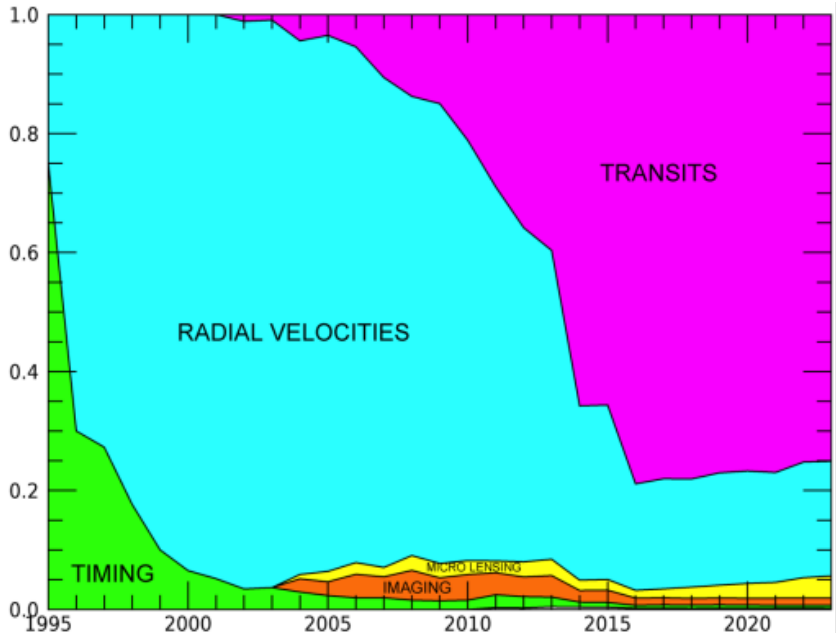
buscando las sutiles caídas periódicas en la intensidad de luz que podrían indicar la presencia de planetas en tránsito. Un ejemplo claro de este proceso son los cuadrantes de la misión Kepler que se explica más en detalle en la sección 2.1.1 de este documento.

Inicialmente, hubo un considerable optimismo de encontrar grandes números de exoplanetas a partir de búsquedas de tránsito dedicadas. Sin embargo, los rigurosos requisitos observacionales pronto se hicieron evidentes: la necesidad de telescopios dedicados, la alta precisión fotométrica, instrumentación estable con bajos niveles de ruido sistemático y procedimientos para manejar un gran número de detecciones de falsos positivos. Este creciente interés a su vez derivó en estrategias de seguimiento más perfeccionadas, con acceso a instrumentos espectroscópicos de alta precisión adecuados para confirmar los descubrimientos y estudiar las propiedades atmosféricas y estructurales de estos exoplanetas (Borucki et al., 2010)

A lo largo de los últimos años, la observación e interpretación de los tránsitos ha evolucionado rápidamente. Ahora, va más allá de la simple detección fotométrica del tránsito, incluyendo fotometría multicolor y espectroscopia durante las fases de tránsito y eclipse secundario, así como la búsqueda de estructuras fotométricas detalladas, variaciones en el tiempo de tránsito y luz reflejada durante la iluminación del lado diurno.

El método de tránsito se basa en la geometría del sistema estrella-planeta. Las ecuaciones geométricas describen las características principales de la curva de luz del tránsito, incluyendo su profundidad, duración total y forma. Además, la probabilidad de que ocurra un tránsito depende de la inclinación orbital del planeta y la relación entre los radios del planeta y la estrella. Aun así, el método de tránsito supera con creces el número de exoplanetas detectados por los otros métodos actuales (Díez Alonso et al., 2023), como ya se ha mencionado anteriormente y se muestra en la Figura 3.

Figura 3. *Distribución de descubrimientos de exoplanetas según el método.*



Desde 2016, los planetas descubiertos por tránsito representan aproximadamente el 74% del total de 5678 planetas conocidos (junio de 2024), según el Archivo Exoplanetario de la NASA.

Fuente: Deeg & Alonso, 2024

1.2. Variabilidad estelar

La variabilidad estelar se refiere a los cambios en el brillo de una estrella a lo largo del tiempo. Estos cambios pueden ser causados por varios factores y se clasifican en dos grandes grupos: intrínsecos y extrínsecos, tal y como se muestra en la clasificación incluida en la Figura 4. Las estrellas variables son aquellas cuya luminosidad varía debido a procesos internos como la pulsación, explosiones, o debido a interacciones con otras estrellas o planetas.

1.2.1. Variables Intrínsecas

Este tipo de variables se caracterizan por el cambio de su brillo debido a procesos internos en la estrella misma. Entre ellas se encuentran las Variables Pulsantes, como las Cefeidas y las RR Lyrae (Eyeret al. 2018), que se expanden y contraen periódicamente y se utilizan para determinar distancias en el universo. También están las Variables Eruptivas, que incluyen estrellas jóvenes como las T Tauri y aquellas que experimentan erupciones súbitas de brillo,

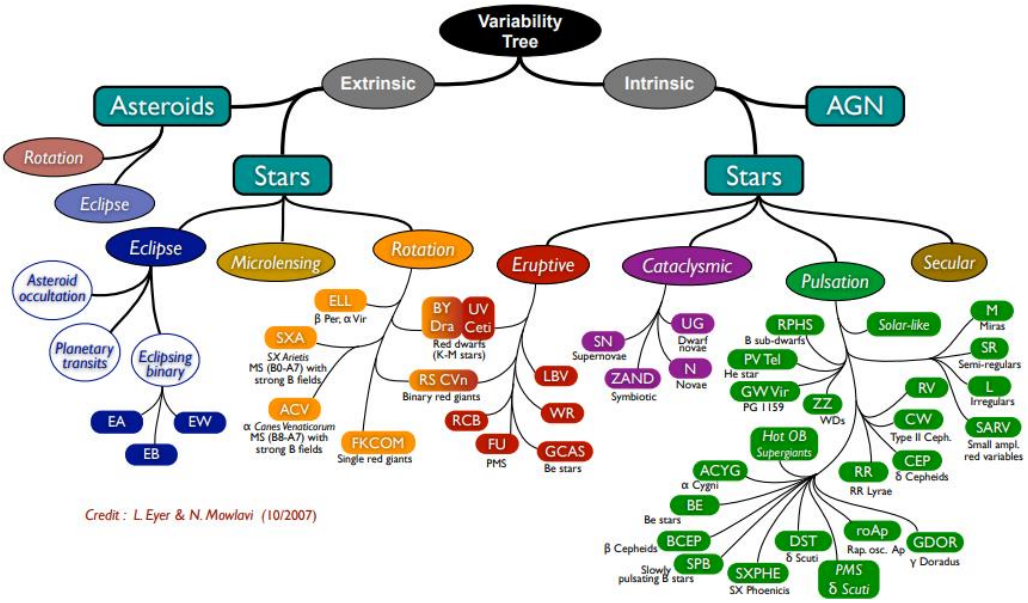
como las variables FU Orionis. Por último, las Variables Cataclísmicas son sistemas binarios donde una enana blanca acumula material de su compañera hasta que ocurre una supernova tipo Ia, que también es un método utilizado para determinar distancias en el universo.

1.2.2. Variables Extrínsecas

Por su parte, en este tipo de estrellas, la variabilidad es causada por factores externos, como la interacción con otras estrellas o planetas (Eyeret al. 2018). Ejemplos de estas son las Variables Eclipsantes, que son sistemas binarios donde una estrella pasa frente a la otra, causando eclipses periódicos y variaciones en el brillo observadas desde la Tierra. También están las Variables Rotacionales, que muestran variabilidad debido a manchas estelares rotando dentro y fuera de nuestra vista.

En este proyecto hemos tenido en cuenta esta particularidad de variabilidad estelar, enfocándonos en las curvas de luz disponibles para sistemas con estrellas intrínsecamente variables.

Figura 4. *Representación de los tipos de estrellas variables.*



Se muestra las tipologías agrupadas por fenómenos físicos que originan su variabilidad, partiendo de la base de intrínsecas o extrínsecas.

Fuente: Eyer, L., & Mowlavi, N. (2008). Variable stars across the observational HR diagram

2. Contexto y estado del arte

En este apartado se presenta en primer lugar la descripción de las misiones de detección de exoplanetas de interés en este proyecto. La mayoría de los estudios sobre exoplanetas han estado enfocados en la caracterización de sistemas individuales o en la determinación de propiedades de poblaciones específicas de exoplanetas de una misión en particular.

Existen numerosos estudios para comprender la diversidad de exoplanetas detectados por las misiones Kepler, K2 y TESS en base a la información obtenida de las mismas, incluyendo la determinación de sus radios, semiejes mayores y flujos incidentes.

En esta sección, también se realiza una breve descripción de la misión GAIA, que se ha utilizado para filtrar y tabular la información disponible de las misiones Kepler, K2 y TESS.

Seguidamente se presentan algunos estudios de interés relacionados con la detección de exoplanetas y obtenidos por la comunidad científica mediante el uso de algoritmos computacionales tradicionales, así como mediante técnicas de Machine Learning. En concreto, se presentarán los resultados de los estudios de Hey et al. (2021), Hippke y Heller (2019), Malik, et al. (2022), Shallue y Vanderburg (2018), y Jin, et al. (2022).

2.1. Misiones

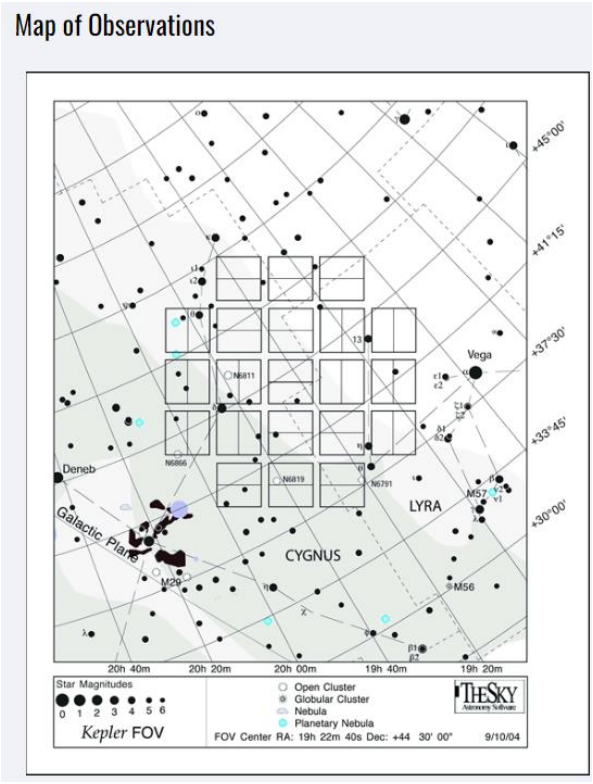
2.1.1. Misión Kepler

La misión Kepler, lanzada el 6 de marzo de 2009 por la NASA en colaboración con el ARC y el JPL desde Cabo Cañaveral, Florida, es una de las misiones fundamentales de la búsqueda de exoplanetas en la Vía Láctea. Equipado con un fotómetro en un telescopio especializado, el cual consta de 21 módulos CCD (cada uno con dos CCDs de 2200x1024 píxeles), ha detectado planetas mediante el método de tránsito, observando disminuciones en el brillo de una estrella cuando un planeta pasa frente a ella (Kepler Mission Team, 2023).

Cada módulo del telescopio espacial Kepler abarca 5 grados cuadrados del cielo y su campo de visión (FOV) completo de 116 grados cuadrados se muestra en la Figura 5.

A lo largo de su misión, Kepler ha permitido identificar numerosos sistemas planetarios, incluyendo por ejemplo el descubrimiento del planeta Kepler-22b en la zona habitable de una estrella en 2011, destacando su capacidad para hallar mundos potencialmente habitables (Incha et al., 2023).

Figura 5. *Cuadrantes de las observaciones de la misión Kepler.*



Fuente: Nasa (https://science.nasa.gov/mission/kepler/in-depth/)

A pesar de desafíos técnicos que llevaron a una misión extendida denominada K2, Kepler logró confirmar más de 2.700 exoplanetas (Incha et al., 2023). Tras agotar su combustible en 2018, Kepler fue retirado, pero sus datos continúan proporcionando valiosa información, como, por ejemplo, contribuyendo al estudio de fenómenos como las estrellas CP y las variables del tipo δ Scuti (Guzik, 2021).

2.1.2. Misión TESS

La misión TESS (Transiting Exoplanet Survey Satellite), ha contribuido significativamente a la exploración y descubrimiento de exoplanetas. Algunos de sus logros significativos son el descubrimiento de una supertierra alrededor de la estrella enana M TOI-1680 (Ghachoui et al., 2023); también este satélite ha permitido estudios detallados sobre la rotación estelar, detectando periodos de rotación de hasta 80 días en estrellas del tipo FGK y enanas M (Claytor et al., 2023). Además, TESS ha sido fundamental para entender la actividad magnética en estrellas variables, donde se han observado eventos de fulguraciones en diversas categorías de estrellas (Zhong et al., 2023). Otro logro importante ha sido la creación de catálogos de candidatos a exoplanetas, como el Triple-9 Catalog II, que presenta 999 candidatos a exoplanetas verificados uniformemente (Guirguis y Huang, 2023).

2.1.3. Misión Gaia

La misión Gaia, emprendida por la Agencia Espacial Europea (ESA), tiene como objetivo principal realizar un censo tridimensional de aproximadamente mil millones de estrellas en nuestra galaxia. Esta misión recopila datos precisos sobre la posición, brillo, distancia y movimiento propio de estas estrellas.

Gaia Data Release 3 (conocida como Gaia DR3), que abarca observaciones desde julio de 2014 hasta mayo de 2017, incluye mediciones fotométricas en las bandas G, G_{BP} y G_{RP} para cerca de 1.8 mil millones de fuentes (Gaia Collaboration et al., 2023).

Gaia DR3 ha clasificado objetos tanto galácticos como extragalácticos que muestran variabilidad, a lo largo de todo el cielo. Para clasificar y obtener parámetros de los objetos presentes en Gaia DR3 se emplearon modelos de aprendizaje automático supervisado, incluyendo *Extreme Gradient Boosting (XGBoost)* y *Random Forest*, resultando en la clasificación de aproximadamente 12.4 millones de fuentes variables en 25 clases diferentes. Este proceso incluyó la identificación de varios tipos de estrellas variables en la Vía Láctea, las Nubes de Magallanes, la galaxia Andrómeda y explosiones de supernovas en galaxias lejanas (Gaia Collaboration et al., 2023).

Los resultados del Gaia DR3 proporcionan una clasificación detallada de cerca de 9 millones de estrellas variables, aproximadamente 2.5 millones de galaxias y un millón de núcleos galácticos activos. La publicación de estas clasificaciones de estrellas variables ha facilitado el desarrollo de este proyecto, proporcionándonos los datos necesarios para clasificar de una forma sencilla la información obtenida mediante los métodos de detección de tránsito anteriormente mencionados.

2.2. Algoritmos de detección de exoplanetas.

2.2.1. Detección de exoplanetas mediante algoritmos tradicionales

2.2.1.1. Algoritmo de mínimos cuadrados de tránsito (TLS)

Recientemente se han desarrollado diversos estudios relacionados con la detección de exoplanetas mediante el método del tránsito y apoyado en algoritmos computacionales tradicionales. En este contexto, Hippke y Heller (2019) han desarrollado el algoritmo de mínimos cuadrados de tránsito (TLS) que permite la detección de tránsitos planetarios a partir fotometría de series temporales. TLS realiza búsquedas asociadas a tránsitos teniendo en cuenta la disminución de la luz estelar, así como el ingreso y salida del exoplaneta en el campo de visión del observador.

El método desarrollado en el algoritmo TLS utiliza una curva de luz de tránsito modelo optimizada para encontrar planetas pequeños, a partir de una extensa muestra de exoplanetas observados con la misión Kepler (Hippke & Heller, 2019). Adicionalmente Hippke y Heller (2019) han compensado la alta carga computacional usando el algoritmo Mergesort (von Neumann, 1945) para el ordenamiento de datos y la restricción de las duraciones de los tránsitos de prueba a un rango más pequeño que abarcara todos los exoplanetas conocidos. Esto último permitió al algoritmo limitarse a un conjunto específico de datos más propenso a contener curvas de luz con tránsitos de exoplanetas, lo que redujo la carga computacional al evitar la exploración de rangos innecesariamente amplios. Estas técnicas de reducción de carga computacional permitieron al algoritmo TLS analizar un conjunto de datos de curvas de

luz de la misión K2, comprendiendo 80 días de observaciones con cadencia de 30 minutos, en menos de 10 segundos, tiempo comparable con el ampliamente utilizado algoritmo de mínimos cuadrados de caja (BLS) (Kovács et al. 2002, 2016).

Hippke y Heller (2019) han realizado un experimento de inyección y recuperación de tránsitos en el cual se ha simulado la presencia de exoplanetas con tamaños similares al de La Tierra orbitando estrellas similares a nuestro Sol para luego intentar detectar y recuperar estos tránsitos utilizando el algoritmo TLS. Para esto, se han utilizado curvas de luz generadas artificialmente para simular las observaciones y se les han agregado un ruido blanco de 110 partes por millón (ppm) para imitar las fluctuaciones en el brillo estelar. La cadencia de observación es de 30 minutos, lo que significa que se toma una medición del brillo estelar cada 30 minutos. Adicionalmente, se establecieron umbrales de eficiencia de detección de señales (SDE) para los algoritmos BLS y TLS, de tal forma que la tasa de falsos positivos fuera del 1%. Como resultado, se obtuvo que TLS tenía una tasa de verdaderos positivos del 93%, mientras que BLS tenía una tasa del 76% (ver Figura 6), lo que implica que las detecciones realizadas por TLS son más confiables en comparación con BLS, ya que TLS logró una tasa de verdaderos positivos más alta.

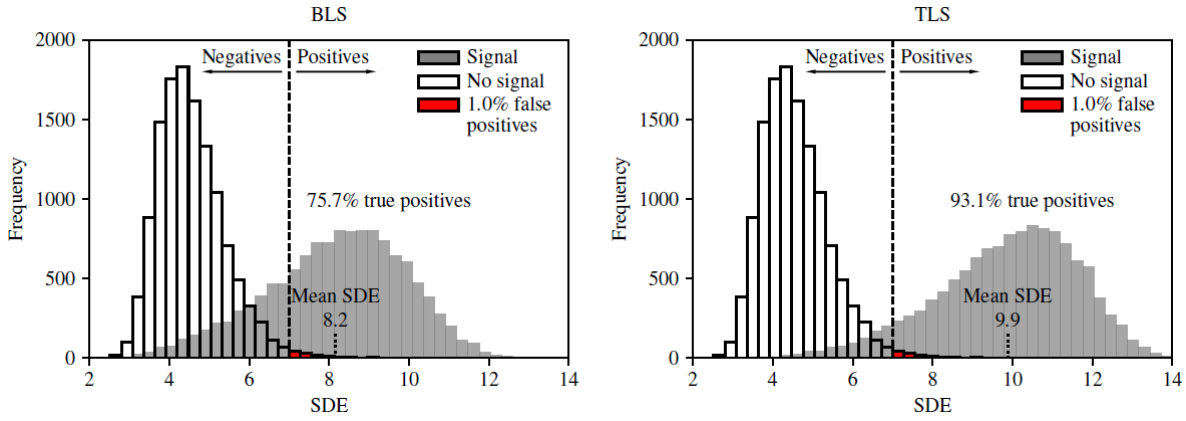
Además, Hippke y Heller (2019) realizaron una prueba adicional del algoritmo TLS utilizando la curva de luz de K2 del sistema TRAPPIST-1, detectando seis de los siete planetas del tamaño de la Tierra que se sabe que orbitan alrededor de TRAPPIST-1. Lo que significa que el algoritmo TLS fue capaz de detectar la mayoría de los planetas conocidos en este sistema. Para lograr este resultado, se realizó una búsqueda iterativa utilizando TLS, ajustando la SDE para encontrar planetas más difíciles de detectar.

Este estudio sugiere que el algoritmo TLS es más confiable que BLS para encontrar cualquier tipo de planeta en tránsito, siendo especialmente adecuado para la detección de planetas pequeños en series temporales largas de Kepler, TESS y PLATO.

A partir de los resultados de Hippke y Heller (2019) se han realizado nuevos estudios para la detección de tránsitos de exoplanetas mediante algoritmos computacionales. Hey et al. (2021) incorporan en su estudio el uso del algoritmo TLS, así como el algoritmo BLS, en el desarrollo

de un método computacional para la búsqueda de tránsitos alrededor de estrellas variables δ Sct conocidas ($6.500\ K\ < T_{eff} < 10.000\ K$) en datos de cadencia larga de Kepler, después de restar la señal de pulsación de las estrellas mediante un procedimiento automatizado (*prewhitening*). Este método resultó computacionalmente eficiente para distinguir entre pulsaciones de baja frecuencia y tránsitos en curvas de luz. Con este método se identificaron tres nuevos eventos de tránsito candidatos que no habían sido detectados previamente debido a la presencia de pulsaciones estelares en las curvas de luz, aunque es probable que estos candidatos sean falsos positivos.

Figura 6. *Valores estadísticos de la eficiencia de detección de señales para un experimento de inyección-recuperación de tránsitos de curvas de luz simuladas con ruido blanco.*



Panel izquierdo: algoritmo BLS. Panel derecho: algoritmo TLS. Se muestran los resultados de 10,000 simulaciones de curvas de luz de 3 años de duración con solo ruido blanco (histogramas abiertos) y de la misma cantidad de curvas de luz con ruido blanco y un tránsito planetario similar a La Tierra alrededor de una estrella G2V (histogramas grises). Se encuentra que los umbrales de SDE en los que las tasas de falsos positivos son del 1% son $SDE_{fp=1\%} = 7$. En este umbral de SDE, la tasa de recuperación de las señales inyectadas (la tasa de verdaderos positivos) es del 75.7% para BLS y del 93.1% para TLS.

Fuente: Hippke y Heller (2019). Optimized transit detection algorithm to search for periodic transits of small planets.

2.2.1.2. Búsqueda de tránsitos en estrellas intrínsecamente variables

La secuencia o cadena de procesamiento (*pipeline*) desarrollada por Hey et al. (2021), mostrada en la Figura 7, fue sometida a pruebas de inyección simple donde se introdujeron exoplanetas con radio 0.5 veces el radio de Júpiter ($0.5\ R_{Jup}$) o incluso radios mayores en los datos. En este mismo artículo se plantea que al extrapolar el número de estrellas δ Sct en el catálogo de Kepler, se espera que haya 12 planetas detectables con un tamaño mayor a $0.5\ R_{Jup}$ en los datos recopilados por el satélite TESS.

La muestra utilizada en el estudio de Hey et al. (2021) incluye algunos de los planetas en tránsito más calientes conocidos que orbitan alrededor de estrellas evolucionadas. Además, esta muestra es la primera en su tipo en contener tránsitos alrededor de estrellas variables de tipo δ Sct (delta scuti), siendo la primera vez que se logra una muestra completa de este tipo de tránsitos.

El diagrama mostrado en la Figura 7 representa las funciones principales del *pipeline* de Hey et al. (2021) desarrollado para buscar señales de tránsitos ocultas por las pulsaciones de las estrellas tipo δ Sct. El *pipeline* está compuesto por varios componentes que realizan una serie de operaciones descritas a continuación:

1. Correcciones de la curva de luz: se realizaron correcciones en la curva de luz para eliminar o mitigar posibles fuentes de ruido o interferencia que puedan afectar la detección de señales de tránsito, en la que se eliminan variaciones lentas en el brillo de cada estrella. Para hacer esto, se eliminan los valores extremos de las curvas de luz y se ajusta una curva suave a los datos para eliminar cualquier variación persistente que no estuviera relacionada con pulsaciones estelares para luego combinar los datos en una sola curva de luz.
2. Ajuste e iteración para sustraer pulsaciones: el procedimiento de limpieza es un procedimiento iterativo ampliamente utilizado para estrellas pulsantes, también conocido como *prewhitening* (Lenz y Breger 2004). El método ajusta y sustrae señales en el dominio del tiempo de la forma:

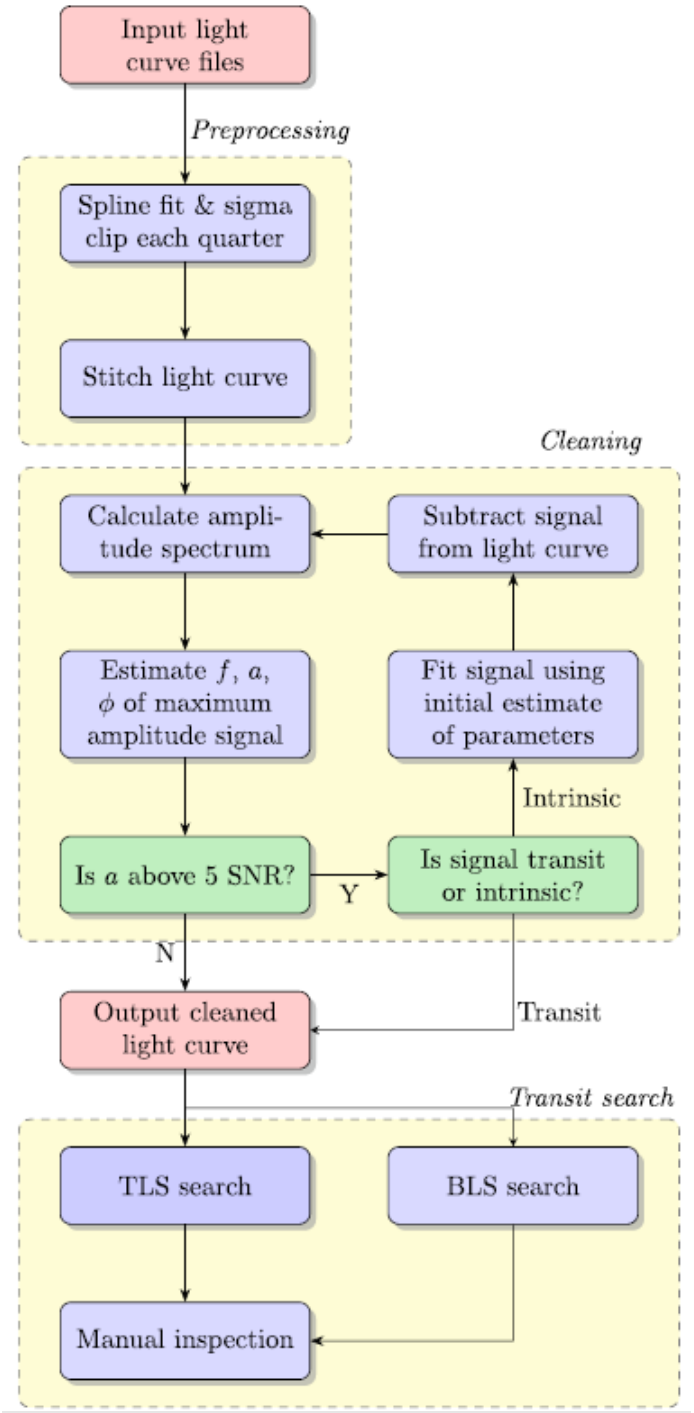
$$y(t) = A\cos(\omega t + \phi) \quad (1)$$

donde A, ω y ϕ son la amplitud, frecuencia angular y fase del modo de oscilación respectivamente. El procedimiento selecciona el pico de frecuencia en el espectro de amplitud con la amplitud más alta en cada paso, calcula una estimación inicial de la amplitud y la fase, y luego ajusta la Ecuación (1) a la curva de luz. El ajuste resultante se resta del flujo total para iteraciones posteriores. La iteración continúa hasta que se cumpla una de dos condiciones: no queda ninguna señal por encima de una relación señal-ruido (SNR) de 5, o se han eliminado más de 100 señales.

3. Tratamiento de variabilidad de baja frecuencia: una dificultad importante al intentar eliminar las pulsaciones de las curvas de luz de las estrellas tipo δ Sct es que comúnmente presentan variaciones de baja frecuencia, causadas ya sea por la modulación rotacional basada en manchas o por modos de gravedad de baja frecuencia (Breger 2011; Van Reeth et al. 2015, 2018). Dado que realizar la limpieza de la señal en frecuencias bajas requiere decidir si la señal observada es intrínseca a la estrella, causada por su variabilidad, o debida a un tránsito, los investigadores diseñaron un método para diferenciar entre tránsitos y variabilidad estelar intrínseca mediante el uso del Criterio de Información Bayesiano (BIC) (Schwarz 1978; Neath & Cavanaugh 2012). Este criterio proporciona una medida de la calidad del ajuste de un modelo a los datos, teniendo en cuenta la complejidad del modelo. Un valor más bajo de BIC indica un mejor ajuste del modelo a los datos. Siguiendo este criterio, se modeló la variabilidad intrínseca estelar, causada por la pulsación o rotación, y modelando los tránsitos usando el algoritmo BLS. Para probar la eficiencia del criterio BIC como clasificador entre tránsitos y variabilidad intrínseca, se aplicó a 1000 curvas aleatorias de Kepler. Para cada una, se inyectó un tránsito calculado utilizando el paquete Python BATMAN (Kreidberg 2015), con un período orbital aleatorio entre 0.5 y 50 días, y un radio planetario entre 0.01 $R_{\odot}$ y 0.5 $R_{\odot}$, a partir de

distribuciones uniformes. Los resultados de la prueba de inyección se presentan como una matriz de confusión en la Figura 8.

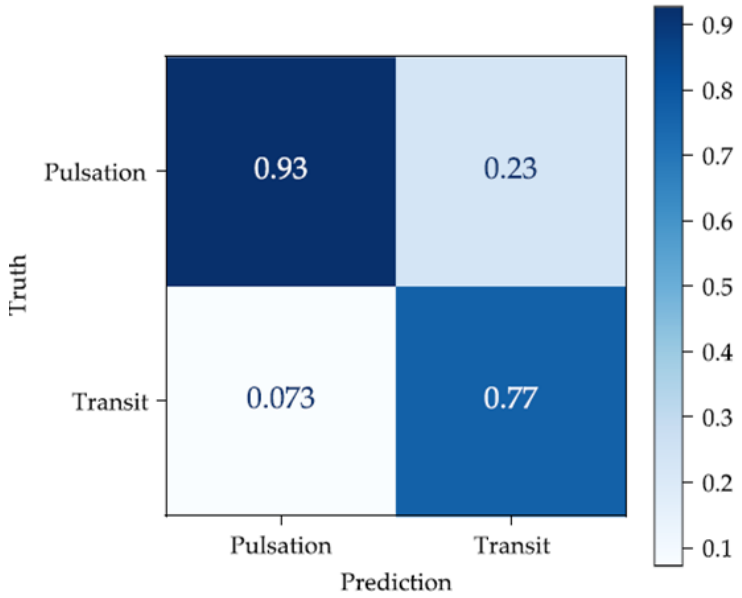
Figura 7. *Diagrama de flujo de las funciones principales del Pipeline de Hey et al. (2021)*



Fuente: Hey et al. (2021). A Search for Transits among the Delta Scuti Variables in Kepler

4. Búsqueda de tránsito: Una vez eliminadas las pulsaciones y la variabilidad intrínseca, se realiza la búsqueda de tránsito de exoplanetas en cada curva de luz. Para cada conjunto de datos de Kepler, se eliminan la variabilidad en escalas de tiempo significativamente más largas que las duraciones de tránsito utilizando el filtro Savitzky-Golay implementado en el paquete *lightkurve* (Barentsen et al., 2019) y se buscan los tránsitos usando el algoritmo TLS, limitando la búsqueda a períodos entre 0.5 y 50.0 días[1].

Figura 8. *Matriz de confusión de las pruebas de inyección utilizando el criterio BIC.*



Se diferencia entre tránsitos y variabilidad intrínseca en una curva de luz. El criterio BIC se ha normalizado a 1.

Fuente: Hey et al. (2021). A Search for Transits among the Delta Scuti Variables in Kepler

Como resultado, al analizar las curvas de luz en el *pipeline* definido, Hey et al. (2021) obtuvieron el 100% de los exoplanetas con un tamaño mayor a aproximadamente $0.5\ R_{Jup}$, alrededor del 80% con $0.36\ R_{Jup}$ y el 50% con $0.25\ R_{Jup}$. De acuerdo con Hey et al. (2021), estas pruebas dan confianza en que el método desarrollado es sensible, como mínimo, a exoplanetas de $0.5\ R_{Jup}$.

[1] El 77% de todos los objetos de interés de Kepler se encuentran en ese rango de período Hey et al. (2021).

2.2.2. Detección de exoplanetas mediante técnicas de machine learning

El desarrollo y evolución de técnicas de Machine Learning ha permitido optimizar el análisis de grandes volúmenes de datos, así como el estudio y clasificación de éstos a partir de nuevos enfoques para identificar patrones complejos en el estudio de curvas de luz estelares, permitiendo el descubrimiento de señales que indican la presencia de tránsito de exoplanetas.

Uno de los enfoques ampliamente usados son los algoritmos de clasificación, como Support Vector Machines (SVM), Random Forest, o redes neuronales, que facilitan la distinción entre curvas de luz que muestran tránsitos de exoplanetas y aquellas que no los tienen. Estos algoritmos aprenden de manera controlada a partir de conjuntos de datos etiquetados, donde se les enseña a reconocer características específicas que son indicativas de tránsitos.

El éxito de estudios a partir de estos enfoques de Machine Learning depende en gran medida de la calidad y la cantidad de los datos de entrada, así como de la selección adecuada de características y la optimización de los parámetros del modelo. Además, la validación cuidadosa de los resultados es crucial para garantizar la fiabilidad de los descubrimientos realizados con estos algoritmos.

2.2.2.1. Modelos clasificadores

Uno de estos estudios basado en el aprendizaje automático para detectar exoplanetas utilizando el método de tránsito es presentado por Malik, et al. (2022), en el que utilizando la biblioteca de análisis de series temporales TSFRESH para analizar curvas de luz, se extraen 789 características (*features*) de cada curva, las cuales capturan la información sobre las características de una curva de luz.

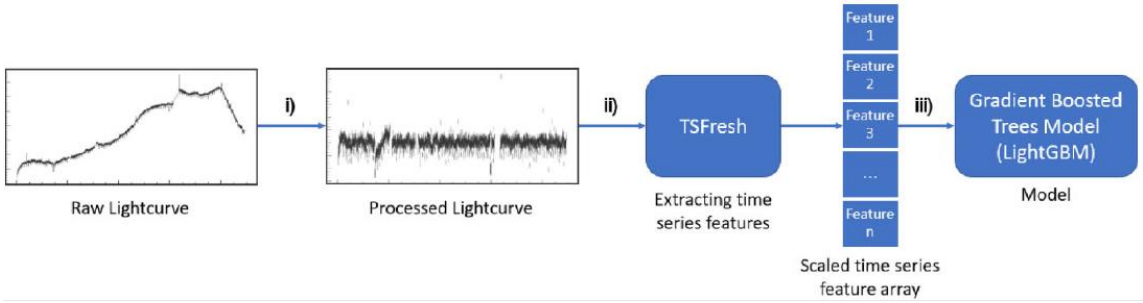
Estas características son utilizadas para entrenar un modelo clasificador de aumento de gradiente utilizando la herramienta de aprendizaje automático LIGHTGBM. Este modelo clasifica cada curva de luz en planeta candidato o no, indicando si un tránsito es observado. Para probar este modelo se usan curvas de luz del conjunto de datos procedentes de las

misiones Kepler y TESS. En la Figura 9 se muestra el diagrama de flujo de las funciones principales del *pipeline* usado por Malik, et al. (2022).

Para los datos de Kepler, el método usado por Malik, et al. (2022) fue capaz de predecir un planeta con un AUC de 0.948 (probabilidad de que una curva de luz elegida al azar con un tránsito verdadero tenga una clasificación más alta que una curva de luz elegida al azar sin tránsito), de modo que el 94.8% de las señales tránsitos de exoplanetas reales se clasifican más alto que las señales que no son de exoplanetas. La tasa de verdaderos positivos resultante es de 0.96, por lo que el 96% de los tránsitos reales son clasificados como tránsitos de exoplanetas.

Respecto a los datos de TESS, el método de Malik, et al. (2022) logró clasificar las curvas de luz con una precisión del 0.98, y fue capaz de identificar exoplanetas con una tasa de verdaderos positivos de 0.82 a una precisión de 0.63.

Figura 9. *Diagrama de flujo de las funciones principales del pipeline de Malik, et al. (2022)*



En el primer paso (i), las curvas de luz en bruto se procesan para eliminar la variabilidad de baja frecuencia y el ruido, y para muestrear la curva de manera uniforme en el tiempo. En el segundo paso (ii), se extraen características de las curvas de luz y se organizan en un arreglo de características. En el tercer y último paso (iii), los arreglos de características se utilizan como entrada para un algoritmo de clasificación y se entrena el modelo.

Fuente: Malik, et al. (2022). Exoplanet detection using machine learning.

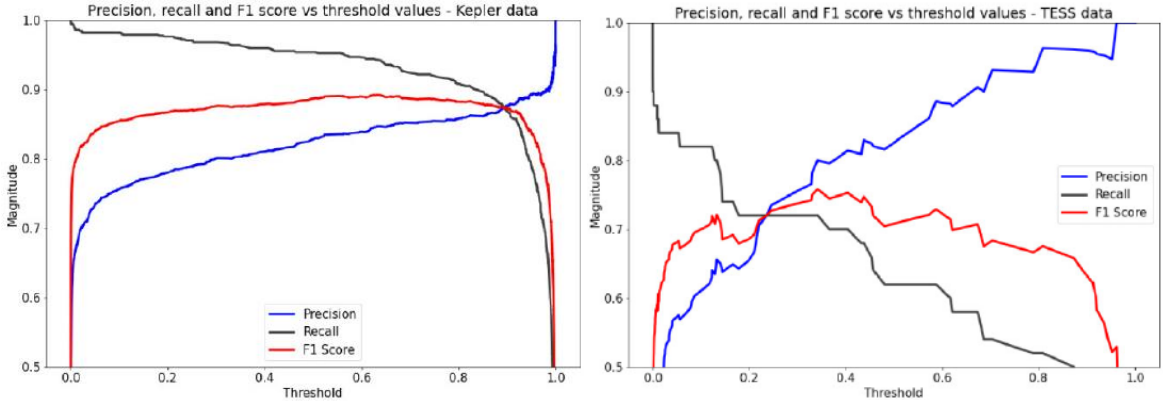
Las principales métricas obtenidas de este estudio se presentan en la Figura 10 y Tabla 2.

Tabla 2. *Resultados del modelo con datos de Kepler y TESS de Malik, et al. (2022).*

Conjunto de datos	AUC	*Recall*	*Precision*
Kepler	0.948	0.96	0 .82
TESS	0.81	0.82	0 .63

Fuente: Adaptado de Malik, et al. (2022). Exoplanet detection using machine learning.

Figura 10. *Precision, recall y F1-score del estudio de Malik, et al. (2022)*



El threshold típico de 0.5 para problemas de clasificación puede adaptarse para aumentar o disminuir la sensibilidad del modelo. Panel izquierdo: Conjunto de datos de Kepler con un threshold de 0.46. Panel derecho: Conjunto de datos de TESS con un threshold de 0.12.

Fuente: Malik, et al. (2022). Exoplanet detection using machine learning.

2.2.2.2. Redes neuronales convolucionales

Shallue y Vanderburg (2018) presentan un modelo para clasificar señales potenciales de exoplanetas utilizando *deep learning,* en donde se entrena una red neuronal convolucional para distinguir entre señales que corresponden a exoplanetas en tránsito y falsos positivos. Este tipo de redes se alimenta con una gran cantidad de datos de curvas de luz, donde cada

señal está etiquetada como exoplaneta o falso positivo. Durante el entrenamiento, la red ajusta automáticamente sus parámetros internos para optimizar su capacidad de discriminar entre estas clases.

Las redes neuronales completamente conectadas (*fully connected neural networks*) son un tipo de arquitectura de redes neuronales en el campo del *deep learning*. En estas redes, cada neurona de una capa está conectada a todas las neuronas de la capa siguiente, por lo que todas las entradas de una capa se conectan a cada neurona de la capa siguiente (Haykin, 1999).

En una red neuronal completamente conectada, las conexiones entre las neuronas tienen asociados pesos que se ajustan durante el entrenamiento del modelo para optimizar su rendimiento, como en la clasificación de imágenes o la predicción de series temporales. Esto permite que la red aprenda y capture relaciones complejas entre las características de los datos de entrada y las salidas deseadas.

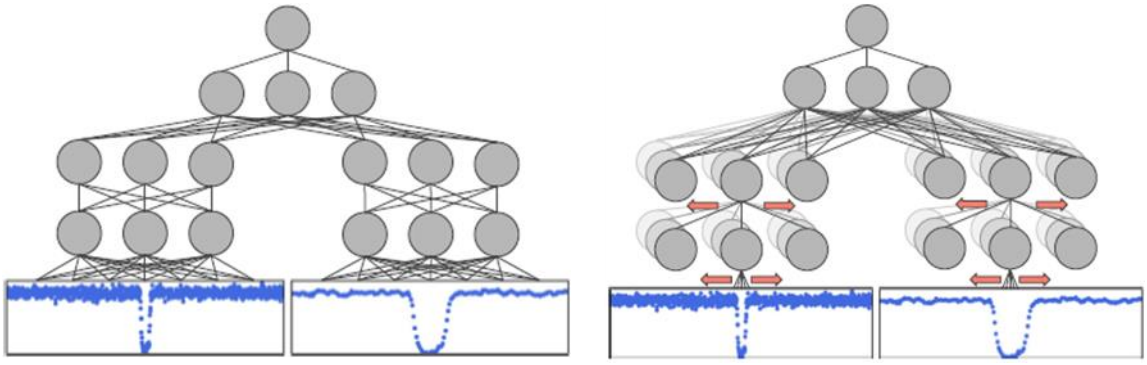
Por otro lado, las redes neuronales convolucionales (CNNs), son un tipo especializado de arquitectura de redes neuronales diseñadas específicamente para el procesamiento eficiente de datos que tienen una estructura de malla, como imágenes o datos de series temporales. A diferencia de las redes neuronales completamente conectadas, donde todas las neuronas de una capa están conectadas a todas las neuronas de la capa siguiente, en las CNNs las neuronas están organizadas en capas de convolución, seguidas generalmente por capas de submuestreo (*pooling*) y capas completamente conectadas al final (Haykin, 1999).

La operación clave en una CNN es la convolución, donde se aplican filtros a regiones locales de los datos de entrada para extraer características relevantes, como bordes, texturas o patrones distintivos. Estos filtros son aprendidos durante el entrenamiento de la red y se desplazan a lo largo de la entrada para producir mapas de características (*feature maps*) que representan las características detectadas en diferentes ubicaciones de la imagen de entrada. Las capas de submuestreo (*pooling*) se utilizan para reducir la dimensionalidad de los mapas de características, disminuyendo así la cantidad de parámetros y cálculos en la red. Esto ayuda a evitar el sobreajuste y a mejorar la eficiencia computacional.

Shallue y Vanderburg (2018) consideran tres tipos de redes neuronales para clasificar TCEs (*Threshold Crossing Event*) de Kepler como "planetas" o "no planetas". Para cada tipo, se tomaron tres opciones de entrada diferentes: vista global, vista local y vistas global y local (ver Figura 11).

1. Arquitectura lineal. El modelo de referencia de Shallue y Vanderburg (2018) es una red neuronal con cero capas ocultas (modelo de regresión logística lineal). Con este modelo se realiza la suposición de que los datos de entrada son linealmente separables; es decir, planetas y no planetas están separados por una superficie de decisión lineal en el espacio de entrada.
2. Arquitectura completamente conectada. Si tanto las vistas globales como locales están presentes, se pasan los dos vectores a través de columnas separadas de capas completamente conectadas antes de combinarlos en capas completamente conectadas compartidas.
3. Arquitectura convolucional. Si están presentes tanto las vistas globales como las locales, se pasan los dos vectores a través de columnas convolucionales disjuntas antes de combinarlos en capas completamente conectadas compartidas.

Figura 11. *Tipos de redes neuronales para clasificar TCEs de Kepler*

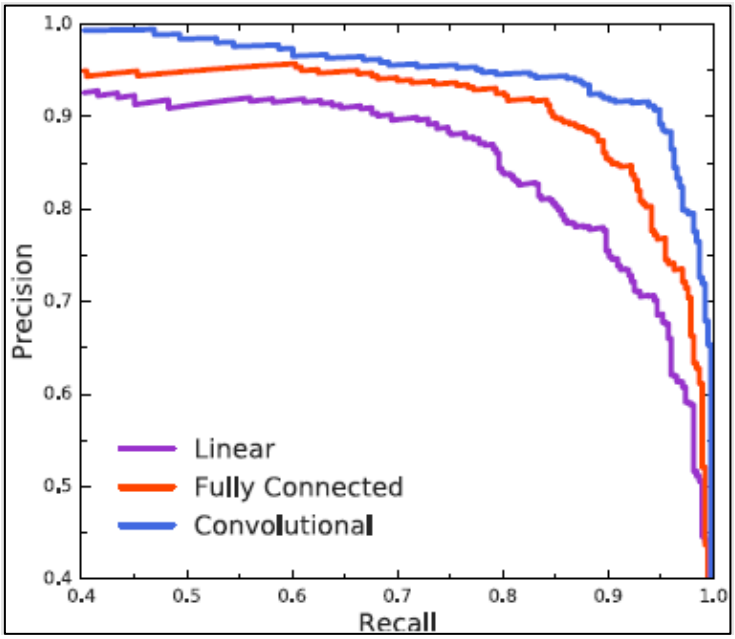


Panel izquierdo: Arquitectura de red neuronal completamente conectada para clasificar curvas de luz, con vistas de entrada tanto globales como locales. Panel derecho: Arquitectura de red neuronal convolucional con vistas de entrada tanto globales como locales.

Fuente: Shallue y Vanderburg 2018). Identifying Exoplanets with Deep Learning: A Five-planet Resonant Chain around Kepler-80 and an Eighth Planet around Kepler-90

El modelo de Shallue y Vanderburg (2018) es altamente efectivo para clasificar candidatos individuales según la probabilidad de que realmente sean exoplanetas: el 98.8% de las veces clasifica las señales de planetas por encima de las señales falsos positivos en el conjunto de prueba. Las métricas obtenidas se muestran en la Figura 12.

Figura 12. *Precisión vs. recall en el conjunto de prueba para las tres arquitecturas de redes neuronales usadas por Shallue y Vanderburg (2018)*



Los tres modelos utilizan vistas de entrada tanto globales como locales. Todos los TCEs predichos como planetas con una probabilidad mayor que el umbral se consideran clasificados como planetas, y todos los TCEs con una probabilidad por debajo del umbral se consideran clasificados como falsos positivos. Se observa que la arquitectura convolucional tiene el mejor rendimiento general. Por ejemplo, con una elección adecuada del umbral, tiene un recall de 0.95 (es decir, el 95% de los planetas reales se clasifican como planetas).

Fuente: Shallue y Vanderburg 2018.

3. Objetivos de la investigación

El objetivo de este trabajo es desarrollar un modelo de machine learning robusto y efectivo para la detección de tránsitos de exoplanetas en estrellas intrínsecamente variables, utilizando curvas de luz extraídas de la misión de búsqueda de exoplanetas Kepler.

Se buscará generar un modelo que sea capaz de identificar posibles patrones de variabilidad causados por tránsitos de exoplanetas ocultos en la variabilidad intrínseca de las estrellas, basándonos en estudios previos como los de Hey et al 2021 y Hippke & Heller 2019.

El proceso (*pipeline*) incluye desde la obtención de la información hasta la generación de uno o varios modelos que puedan clasificar las curvas de luz como candidatas a posibles tránsitos.

Los modelos clasificadores obtenidos, se probarán con una muestra externa para asegurar su generalización. Esperamos que estos modelos se puedan extrapolar a otros catálogos de manera similar a lo expuesto por (A. Berger et 2023).

Para conseguir el objetivo principal de este trabajo, los objetivos específicos que deben cumplirse son los siguientes:

1. **Obtener un modelo clasificador suficientemente robusto:**
    - Desarrollar y entrenar modelos de machine learning, como *XGBoost* y *Random Forest*, utilizando curvas de luz de la misión Kepler.
    - Implementar técnicas de preprocesamiento, como el *prewhitening*, para mejorar la calidad de los datos y la precisión futura del modelo.
    - Maximizar la métrica F1-score del conjunto de validación de la clase 1.0 (tránsitos). Adicionalmente, se buscará maximizar la métrica accuracy de dicha clase.
    - Optimizar los hiperparámetros de los modelos desarrollados para maximizar las métricas objetivo que se han planteado en este estudio.
    - Desarrollar diferentes experimentos con los modelos seleccionados que permitan encontrar los modelos que mejor se ajustan a nuestra necesidad.

- Seleccionar el mejor o mejores modelos a partir de los resultados de los procesos de optimización y experimentos realizados. Estos mejores modelos se usarán para probar la precisión del modelo con el conjunto de datos de prueba.

2. **Evaluar los resultados con una muestra externa:**
   - Aplicar el modelo entrenado a una muestra externa de datos no utilizados en el entrenamiento inicial para validar su correcto funcionamiento. Este objetivo es clave, ya que el modelo clasificador debe procesar los datos de las curvas de luz existentes y futuras de nuevos objetos de misiones futuras de una manera generalizada.
3. **Correlacionar los datos obtenidos para identificar nuevos candidatos a exoplanetas:**
   - Recopilar los datos obtenidos al procesar las muestras y realizar un ejemplo de búsqueda y obtención de sus datos en Gaia DR3.
   - Generar un ejemplo de los siguientes pasos o procesos a seguir una vez determinado un candidato a estudio, para futuras investigaciones.

Es importante destacar que se ha elegido como la métrica principal a optimizar el F1-score debido a que se considera más importante la precisión del modelo en los casos con tránsitos (positivos). La razón de esta maximización es evitar falsos positivos como resultado del modelo que generen trabajo innecesario en evaluaciones posteriores, ya que la principal ventaja de un modelo clasificador es filtrar la información para poder ser evaluada después mediante otros métodos de análisis de tránsitos u otros métodos de detección de exoplanetas como velocidades radiales.

Cumpliendo estos objetivos, se pretende mejorar la capacidad de detección de exoplanetas en estrellas variables, e identificar posibles candidatos que no hayan sido considerados en las distintas misiones.

4. Metodología

En esta sección vamos a introducir la metodología utilizada para realizar este proyecto, desde el proceso de evaluación de la información disponible, pasando por el desarrollo de los modelos de machine learning y finalizando con los modelos clasificadores que serán probados con una muestra de estrellas en la sección de resultados.

Para comenzar el proyecto necesitamos definir la fuente de los datos a utilizar, las misiones en las que vamos a basar la información de entrada será la disponible en las misiones Kepler y Gaia. Se realizará un análisis en esta sección de los datos de la misión Kepler, así como los de la misión TESS con la que se espera obtener los datos de curva de luz (flujo frente al tiempo), mientras que los datos de la misión Gaia los utilizaremos para clasificar los datos iniciales.

Se va a describir el proceso de preparación del modelo de machine learning, que incluye la evaluación y el filtrado de las fuentes de datos. Este proceso es esencial para asegurar que los datos utilizados son de alta calidad y relevancia para el objetivo de detección de tránsitos. La herramienta TOPCAT (Taylor, M. 2005) será fundamental en esta etapa para la manipulación y visualización de datos tabulares, así como de la obtención de los datos para su posterior ejecución y manipulación con los distintos algoritmos utilizados.

Todo el desarrollo y análisis de los datos se ha llevado a cabo utilizando el lenguaje Python (Jupyter Notebook), lo que permite una documentación clara y reproducible del proceso.

Además, se detallarán los algoritmos clasificadores utilizados, XGBoost y Random Forest, y se explicará cómo estos algoritmos se aplican a las curvas de luz de entrada para identificar tránsitos de exoplanetas. Para obtener unos resultados satisfactorios, se describen las distintas fases de la implementación del preprocesamiento de datos, incluyendo técnicas como el *prewhitening*, con el fin de obtener un modelo más robusto.

4.1. Preparación de datos y análisis de curvas de luz

4.1.1. Preparación de datos

Antes de llevar a cabo el desarrollo del modelo destinado a la detección de tránsitos en estrellas intrínsecamente variables, hemos seguido un proceso sistemático de evaluación de las fuentes de datos y análisis de las curvas de luz correspondientes. En esta sección, se detalla el procedimiento seguido para la preparación y una breve evaluación de la información disponible.

Para realizar esta evaluación, se emplea la herramienta TOPCAT, un software de visualización y edición de datos para tablas y catálogos astronómicos. Esta herramienta es esencial para acceder al análisis y manipulación de datos tabulares, facilitando una exploración eficaz y profunda de los conjuntos de datos astronómicos.

El conjunto de datos utilizado en esta fase del proyecto proviene de la misión Kepler, específicamente el conjunto de datos Q1-Q17 DR25. Este conjunto se correlaciona con los datos de la misión Gaia DR3 para determinar el tipo de estrella correspondiente a cada curva de luz y para identificar las estrellas variables que son objeto de este estudio (Gaia Collaboration et al., 2023). Este cruce de datos es crucial para identificar adecuadamente las estrellas variables y las características específicas de sus tránsitos.

Tras este cruce, se incluyen 196,617 objetivos de Kepler con Gaia DR3. Dentro de estos objetivos, la misión Kepler ya ha identificado una serie de objetos, denominados "Objetos de Interés" (KOI, Kepler Objects of Interest), que son esenciales para trabajar con los distintos métodos que permiten validar o descartar los posibles exoplanetas encontrados en las curvas de luz. En este caso, una vez cruzado con Gaia DR3, tenemos un total de 7,928 objetos (Kepler Mission Team, 2023).

La información obtenida se filtra utilizando TOPCAT, mediante la definición de estrella variable de Gaia DR3. Este proceso de filtrado permite distinguir entre candidatos positivos y aquellos que se tratan de falsos negativos, como se muestra en la Figura 13.

Figura 13. *Detalle del filtro en TOPCAT para determinar la muestra inicial.*

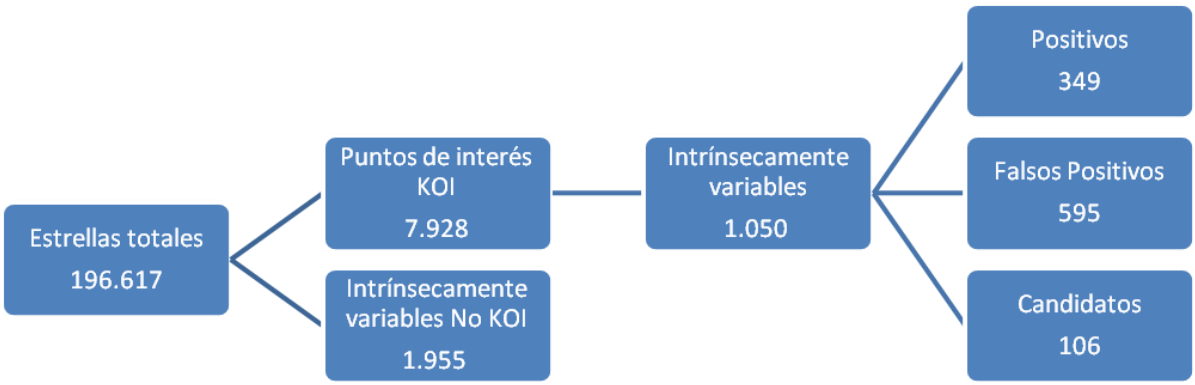
Row Subsets for 4: 4xGaia DR3 (Epoch 2016)

ID	Name	Size	Fraction	Expression	Col $ID
_1	All	7928	100%		
_2	Activated	1	0%		
_3	APF	732	9%		$192
_4	estrellas_variables	1050	13%	VarFlag == "VARIABLE"	
_5	Planetas_confirmados	349	4%	estrellas_variables && koi_disposition == "CONFIRMED"	
_6	Falsos_positivos	595	8%	estrellas_variables && koi_disposition == "FALSE POSITIVE"	

Fuente: Elaboración propia

Estos serían los puntos de partida o muestras utilizadas en la preparación del modelo. En la Figura 14, se muestran de manera esquemática la cantidad de objetos obtenidos a partir de los filtros configurados.

Figura 14. *Resultados de la selección de la muestra inicial.*



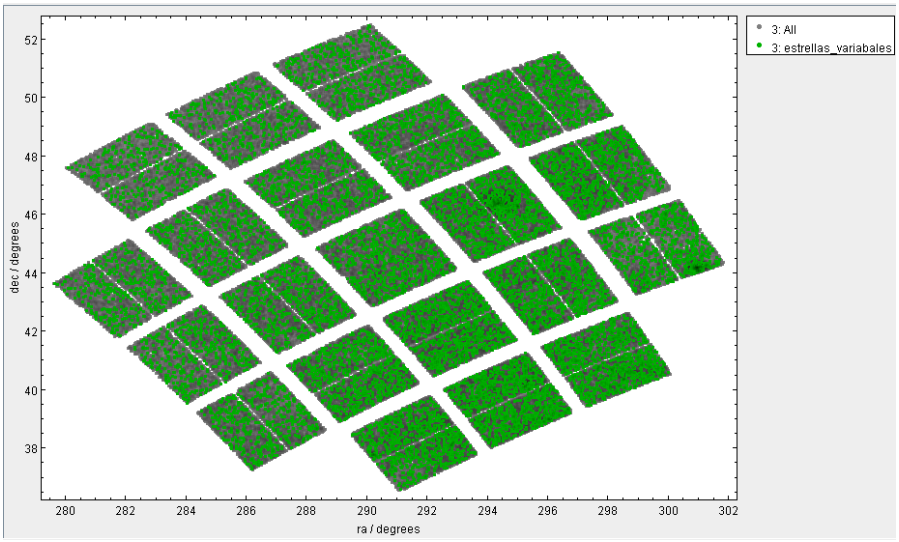
Fuente: Elaboración propia

Los falsos positivos (595) y los positivos (349) serán utilizados para la alimentación del modelo inicial. Una vez determinado un modelo con un nivel de exactitud aceptable, se utilizará en el grupo de estrellas intrínsecamente variables de acuerdo con Gaia DR3 pero que no están en el grupo de las no determinadas como KOI (1.955), este grupo es candidato más plausible para

encontrar alguna curva de luz que pueda haber sido pasada por alto y en el que ejecutar el algoritmo. Los resultados se analizarán en el capítulo de este documento.

En las Figuras 15, 16 y 17, se presentan gráficamente, en función de la ascensión recta y declinación, las distintas agrupaciones en función del número de estrellas totales de la misión Kepler.

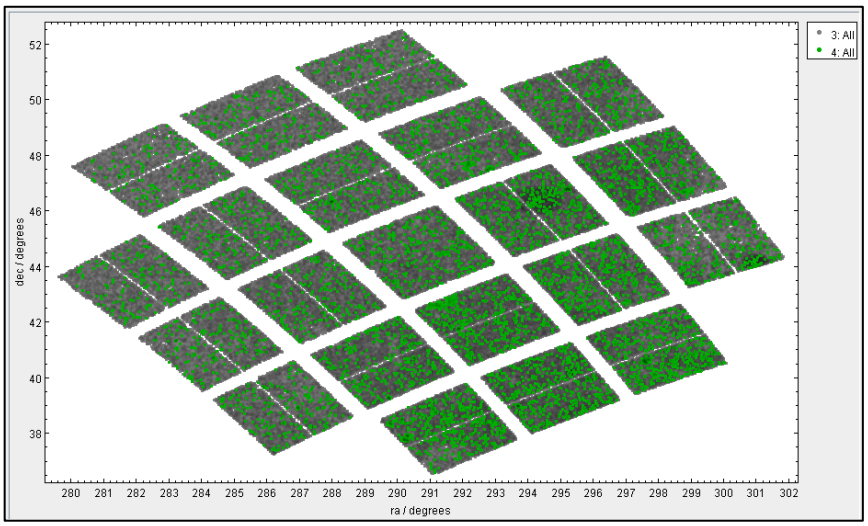
Figura 15. *Zonas de exploración de Kepler en DR25 (1)*



Se muestra en gris las zonas de exploración de Kepler en DR25 y en verde las estrellas clasificadas como estrellas variables en Gaia DR3 de dicha exploración.
Fuente: Elaboración propia

La figura anterior coincide con la información disponible de la misión Kepler anteriormente mencionada en la introducción de este proyecto.

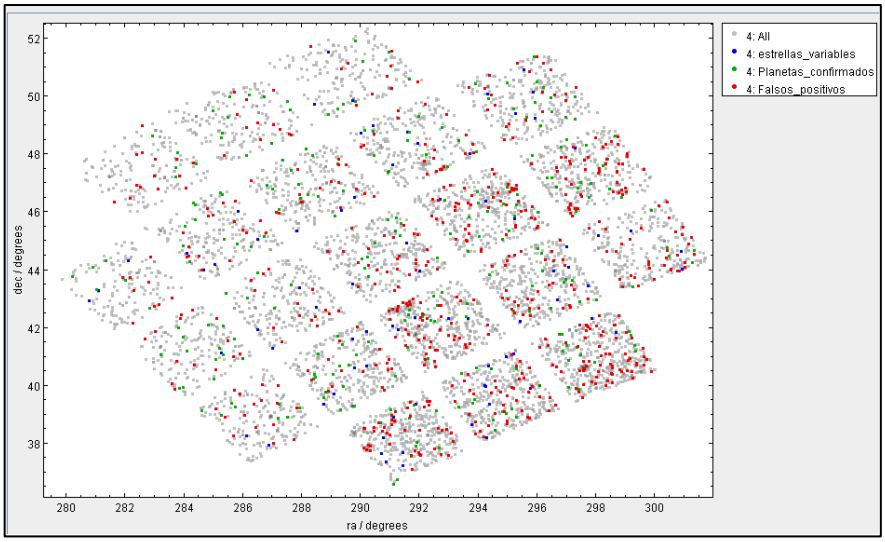
Figura 16. *Zonas de exploración de Kepler en DR25 (2).*



Se muestra en gris las zonas de exploración de Kepler en DR25 y en verde las estrellas clasificadas como puntos de interés en Kepler.

Fuente: Elaboración propia

Figura 17. *Zonas de exploración de Kepler en DR25 (3).*



Se muestra la superposición de los puntos de interés de Kepler (En gris) de los cuales se han separado las estrellas variables, y dentro de las mismas aquellas que están clasificadas como falso positivo (rojo) y planetas confirmados (verde), las estrellas variables que no han obtenido ninguna de estas clasificaciones han quedado marcadas en azul.

Fuente: Elaboración propia

Una vez obtenidos ambos listados, se utilizará el número de identificación del objetivo en el Catálogo de Entrada de Kepler (KIC) que se basa en un estudio previo al lanzamiento para seleccionar estrellas para la encuesta de exoplanetas. Este ID único se puede encontrar en el archivo MAST y se utiliza para identificar cada objetivo en los datos de Kepler.

Mediante los KIC, podemos extraer las curvas de luz de ambas muestras (tránsitos confirmados y falsos positivos). Para ello desarrollamos un algoritmo en Python que descargará las curvas a partir de un archivo .txt generado mediante la lista de KIC extraída de TOPCAT. (Ver Anexo A: Código A1)

4.1.2. Análisis de fuentes y formato de los datos

Una vez descargadas las curvas de luz, se analiza la forma en la que se encuentra la información. Las curvas de luz se encuentran en formato .FITS, existiendo varios archivos que habrá que concatenar para cada identificador. Este formato es un estándar en astronomía para el almacenamiento, análisis y archivo de datos astronómicos (Marmo et al., 2018), De esta forma las curvas de luz pueden ser trabajadas y evaluadas de forma manual mediante TOPCAT.

Cada curva de luz de Kepler almacena varios tipos de valores de flujo, así como los errores asociados y los tiempos medidos. En este proyecto nos vamos a centrar en dos tipos de valores de flujo fundamentales para analizar los datos recogidos por la misión. Estos flujos son definidos y almacenados como sigue:

1. **Flujo de Fotometría de Apertura Simple (SAP):** este flujo se calcula sumando la luminosidad de los píxeles dentro de una apertura predefinida. El flujo SAP se utiliza para estudios iniciales de la luminosidad estelar, pero puede contener errores derivados del proceso de observación, lo cual puede complicar su interpretación directa (Cui et al., 2019).
1. **Flujo SAP de Prebúsqueda de Acondicionamiento de Datos (PDCSAP):** a diferencia del flujo SAP, el flujo PDCSAP ha sido corregido para eliminar los efectos sistemáticos que pueden afectar la calidad de los datos. Esta corrección se lleva a cabo mediante la

Pipeline de Procesamiento de Datos de Kepler, diseñados específicamente para preservar los tránsitos planetarios reales mientras se eliminan otras variaciones no deseadas. (Stumpe et al., 2012).

El análisis de las figuras de la misión Kepler en el siguiente capitulo, muestra que el flujo SAP presenta un cambio a largo plazo en la luminosidad que ha sido eliminado en el flujo PDCSAP, mientras se conservan los tránsitos a la misma profundidad. En este proyecto se usará el flujo PDCSAP como variable de entrada para el modelo.

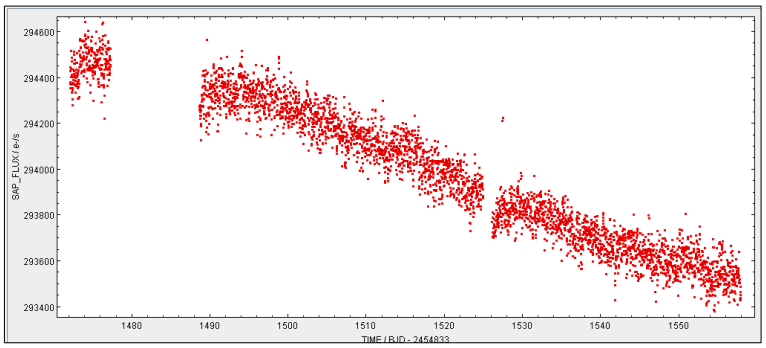
4.1.3. Muestra de curvas de luz de la misión Kepler.

Para evaluar en detalle el estado y el contenido de dicha información, así como las formas de las curvas de luz, se ha realizado manualmente una muestra de cuatro curvas de luz de la misión Kepler (Kid 000757076, Kid 006223256, Kid 00757450 y Kid 008891684) seleccionadas manualmente de forma aleatoria.

EL primer ejemplo de curva de luz seleccionado es Kid 000757076, ésta no presenta tránsito y se trata de una estrella que no está clasificada como intrínsecamente variable, se ha extraído para evaluar los parámetros que utiliza Kepler en su curva de luz. Se observa la diferencia claramente en las figuras 18 y 19 entre la variable SAP_FLUX frente a la variable TIME (tiempo en días) y la variable PDCSAP_FLUX.

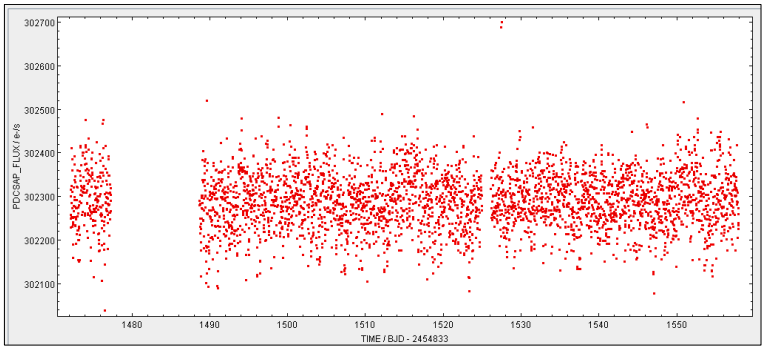
Podemos observar en las figuras 20 y 21 la existencia de ruido que tiene la señal de medición de flujo, esto es un factor importante ya que puede afectar al proceso de entrenamiento del modelo.

Figura 18. *Extracto de la variable SAP_FLUX para Kid 000757076*



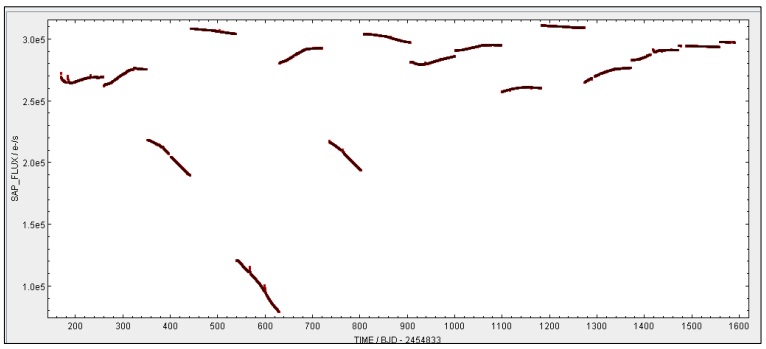
Fuente: Elaboración propia

Figura 19. *Extracto de la variable PDCSAP_FLUX para Kid 000757076*



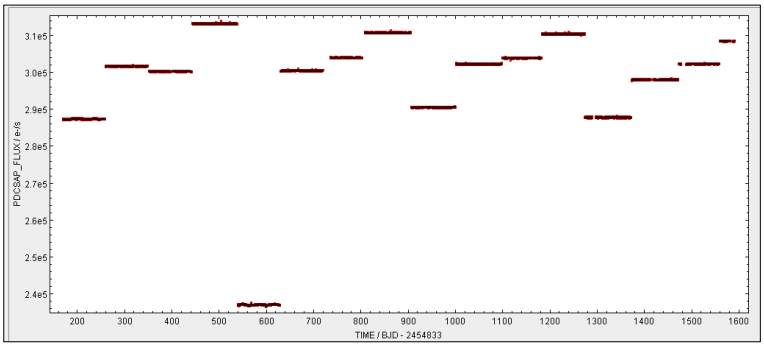
Fuente: Elaboración propia

Figura 20. *Curva total concatenada de la variable SAP_FLUX para Kid 000757076*



Fuente: Elaboración propia

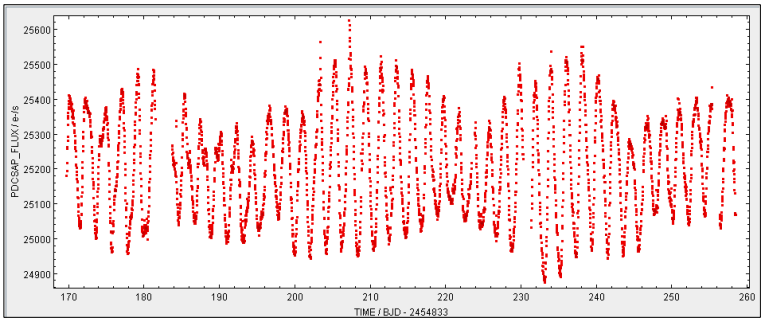
Figura 21. *Curva total concatenada de la variable PDCSAP_FLUX para Kid 000757076*



Fuente: Elaboración propia

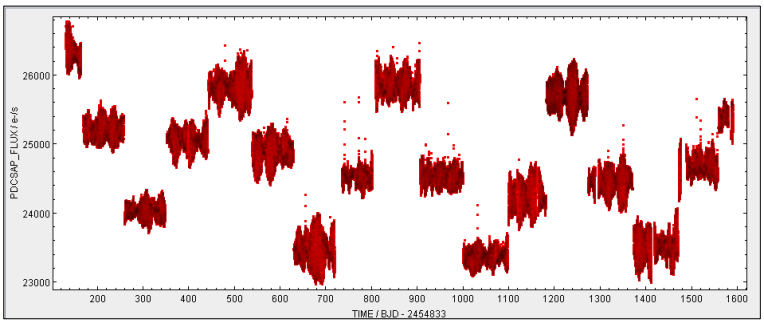
El segundo Kid seleccionado (Kid 006223256), está determinado en los datos disponibles de la misión Gaia DR3 *(Gaia DR3 2076855045430886912)* que han sido cruzados con los datos de Kepler (ver capítulo 4.1) como una estrella intrínsicamente variable. Pero según los datos de la misión Kepler no tiene tránsito confirmado. Se observa en las figuras 22 y 23 que se trata de un tipo de estrella con variabilidad periódica. Este tipo de curvas de luz sería uno de los candidatos objetivo final del algoritmo, ya que, aunque aparentemente no se detecta ningún tránsito, es posible que un análisis de machine learning de lugar a posibles detecciones que no puedan hacerse de manera manual.

Figura 22. *Extracto de la variable PDCSAP_FLUX para Kid 006223256*



Fuente: Elaboración propia

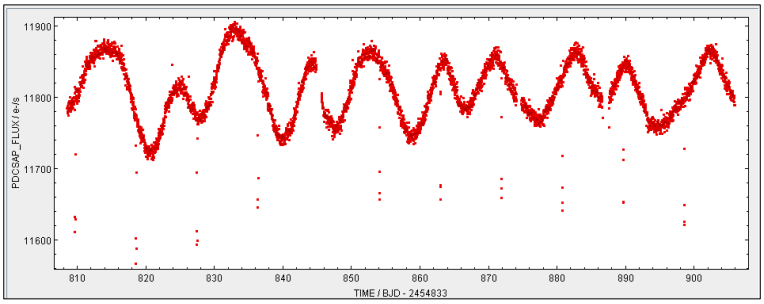
Figura 23. *Concatenado de la variable PDCSAP_FLUX para Kid 006223256*



Fuente: Elaboración propia

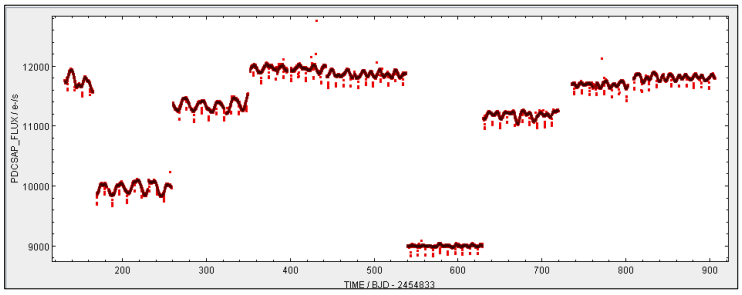
En el caso del tercer candidato seleccionado, Kid 00757450, la curva de luz presenta un tránsito con exoplaneta confirmado, y se tata de una estrella clasificada en Gaia DR3 como estrella intrínsecamente variable. Se puede ver claramente el tránsito en las figuras 24 y 25 dentro de la curva de luz de manera manual. Se trata del exoplaneta confirmado Kepler-450c con un periodo orbital de 15,41 días (NASA, n.d.).

Figura 24. *Extracto de la variable PDCSAP_FLUX para Kid 00757450*



Fuente: Elaboración propia

Figura 25. *Concatenado de la variable PDCSAP_FLUX para Kid 00757450*

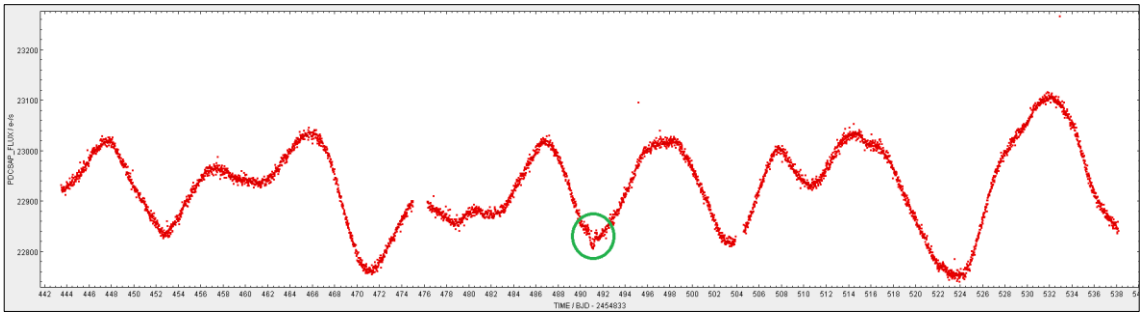


Fuente: Elaboración propia

Por último, para ilustrar la dificultad, se ha buscado un último caso (Kid 008891684) en el que se presenta una estrella clasificada como intrínsecamente variable en Gaia DR3 y con un tránsito, de un exoplaneta confirmado (KEPLER 1634 b) (NASA, n.d.). En este caso con un tránsito de 374 días lo que genera solamente 3 eventos de tránsito a lo largo de la curva de luz (figuras 26 y 27) (definidos por la variable koi_num_transits)

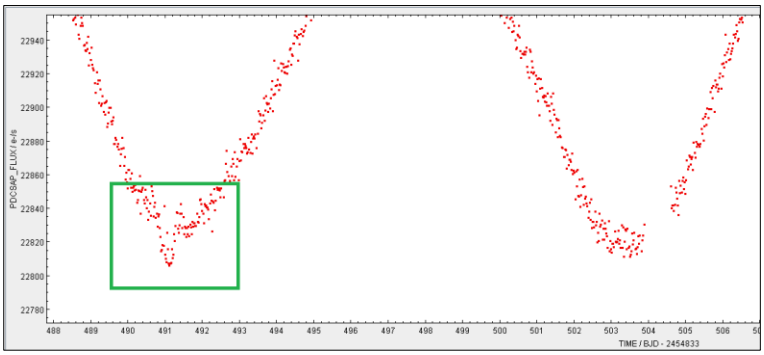
Comparadas con la curva mostrada en la figura 24, la curva de la figura 26 muestra más complejidad para detectar el tránsito incluso de manera manual. Se puede observar en el detalle ampliado de la figura 27 que apenas se puede diferenciar la profundidad del tránsito con respecto al ruido de las mediciones de flujo de la curva de luz.

Figura 26. *Concatenado de la variable PDCSAP_FLUX para Kid 008891684*



Fuente: Elaboración propia

Figura 27. *Detalle del tránsito en la variable PDCSAP_FLUX para Kid 008891684*



Fuente: Elaboración propia

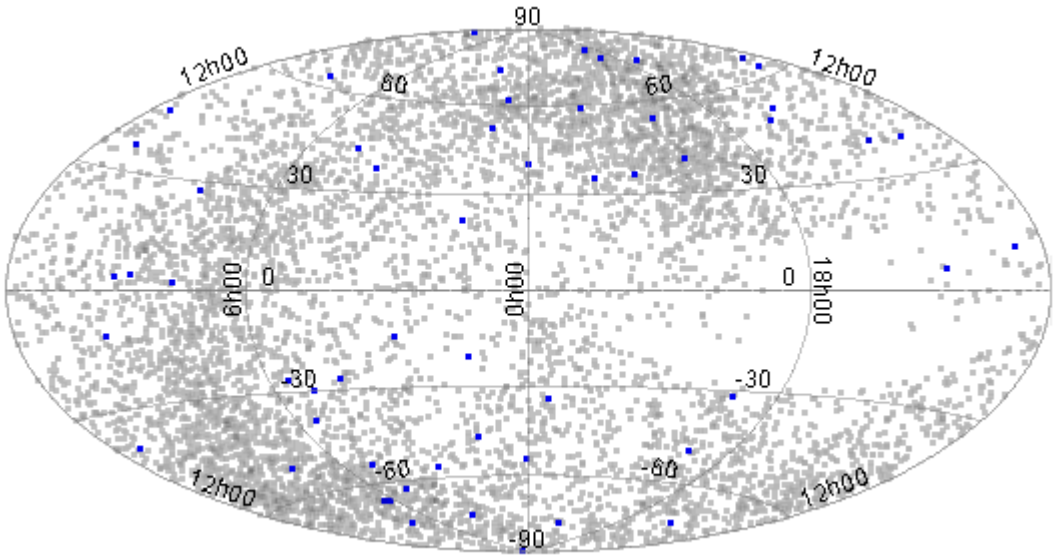
4.1.4. Muestra de curvas de luz de la misión TESS.

De la misma forma que con la misión Kepler, se ha analizado en el transcurso de este proyecto la información disponible en cuanto a curvas de luz de la misión TESS.

La diferencia fundamental frente a la misión Kepler es cómo TESS define un punto de interés. En su interior divide la información en dos catálogos: TESS Input Catalog (TIC) y Candidate Target List (CTL). El catálogo TIC es una colección de 1,73 x 10^9 fuentes estelares medidas en casi la totalidad del cielo. (Paegert et al. 2021). A partir de estos datos, la misión TESS utiliza una serie de criterios y mediante una serie de algoritmos selecciona una serie de objetivos a los que considera candidatos (CTL), de los cuales toma curvas de luz en series de tiempo de 2 minutos de intervalo. Para ello elimina entre otros los candidatos con magnitudes <13, las gigantes rojas mediante un filtro de temperatura efectiva, las enanas blancas y realiza una comparación con Gaia DR2 en cuestión de paralajes y fotometría. Aun así, la lista de candidatos (CTL) es mayor a 9 millones (Stassun et al. 2019).

Estos candidatos ya son analizados por la misión TESS e identifica los llamados objetos de interés (TOI) en aquellos que observa caída de brillo de la estrella que pudiera indicar un posible tránsito. El número total de TOI es de 7.203 candidatos repartidos alrededor del cielo en los cuales existen 68 exoplanetas confirmados en estrellas intrínsecamente variables a partir de estudios adicionales, como se muestra en azul en los datos obtenidos mediante TOPCAT en la Figura 28.

Figura 28. *Coordenadas ecuatoriales de los datos de la misión TESS extraídos mediante TAP search en TOPCAT*



Fuente: Elaboración propia

Durante el transcurso del proyecto hemos podido analizar la estructura de las curvas de luz disponibles de los mencionados objetos de interés (TOI) y podemos confirmar que los parámetros mencionados de la misión Kepler de flujo corregido frente al tiempo (SAPPDC_FLUX) también existen en esta misión. Sin embargo, para la primerea fase del proyecto se ha decidido utilizar la misión Kepler en lugar de la misión TESS principalmente debido a la poca cantidad de tránsitos confirmados en la misión TESS frente a la misión Kepler, que nos impediría un entrenamiento apropiado del modelo por datos insuficientes.

En cualquier caso y como se comentará más adelante en la sección de conclusiones y perspectivas futuras, el modelo generado podrá servir para futuros estudios con los datos en crecimiento de la misión TESS.

4.2. Proceso de generación del modelo de machine learning

En esta sección se presenta el proceso seguido a partir de las curvas de luz seleccionadas de la misión Kepler para generar el modelo de machine learning objeto de esta investigación.

El proceso principal ha seguido una consistencia en base a los siguientes pasos principales:

1. Modelos de clasificación sin preprocesar
2. Preprocesamiento de las curvas de luz
   - Prewhitening
   - Reducción de dimensionalidad de las curvas de luz, generación de nuevas características y creación de nuevos conjuntos de datos
3. Modelos de clasificación utilizando curvas preprocesados
4. Balance de clases y optimización de los modelos mediante la librería Optuna
5. Desarrollo de otros experimentos
6. Resultados, análisis y selección del mejor modelo. Uso del conjunto de datos de prueba

Como se ha mencionado anteriormente, para el desarrollo y búsqueda del mejor modelo, se ha tomado como criterio maximizar la métrica F1-score de la clase 1.0 (Tránsito) en el conjunto de datos de validación. Este criterio se sustenta en que partimos de un modelo con clases desbalanceadas (mayor cantidad de No Tránsitos que Tránsitos) por lo que la métrica de accuracy puede ser engañosa debido a que el modelo podría predecir mejor la clase mayoritaria (No Tránsitos) y aun así obtener una alta precisión. Adicionalmente, consideramos que la utilidad de nuestro modelo radica en identificar correctamente las curvas de luz con tránsito. Sin embargo, como parte de los experimentos y técnicas usadas para optimizar el modelo, se ha incluido la métrica accuracy como segunda métrica a maximizar, como se puede ver en el capítulo 4.5.

4.2.1. Modelos de clasificación de la muestra inicial sin preprocesar

Para comprender y evaluar nuestra investigación sobre la información extraída de la misión Kepler (véase sección 4.1.1), hemos empleado los modelos clasificadores de aprendizaje supervisado XGBoost y Random Forest.

En este primer enfoque, se ha utilizado la muestra de curvas de luz de la misión Kepler sin preprocesar, aprovechando la información de la variable de flujo ya normalizada PDCSAP_FLUX que incluye las curvas de luz de Kepler (véase sección 4.1.2).

Nuestro objetivo principal en este capítulo ha sido evaluar el comportamiento de estos modelos, así como su capacidad para trabajar con los datos de la curva de luz. Además, hemos buscado determinar si el preprocesamiento de las curvas de luz utilizadas es necesario y, en caso afirmativo, identificar qué tipo de preprocesamiento sería el más adecuado.

Los algoritmos clasificadores utilizados para este apartado son los siguientes:

1. **XGBoost** (Chen et al. 2016). Abreviatura de Extreme Gradient Boosting, es una implementación escalable de árboles de decisión potenciados por gradientes, efectiva en problemas de aprendizaje supervisado. Utiliza un enfoque de ensamblaje secuencial de árboles débiles, donde cada árbol se ajusta sucesivamente a los errores residuales del anterior, lo que resulta en un modelo robusto y preciso.
2. **Random Forest (**Breiman, 2001). Algoritmo de aprendizaje automático que construye múltiples árboles de decisión durante el entrenamiento y los combina para obtener una predicción más precisa y estable. Cada árbol se entrena con una muestra aleatoria del conjunto de datos y realiza su propia predicción. Luego, las predicciones de todos los árboles se ponderan. Esta ponderación puede ser mediante un promedio o, si es una clasificación, usando el voto mayoritario para obtener la predicción final.

Como se mencionó en la sección 4.1.2, cada estrella o KOI (Kepler Object of Interest) está asociada con múltiples archivos con extensión .FITS que contienen su correspondiente curva de luz en diferentes instantes de tiempo, por lo que cada parte es concatenada en un único archivo para facilitar su tratamiento.

Una vez obtenidas las curvas concatenadas, extraemos la información necesaria para analizar las curvas de luz, las cuales tenemos clasificadas según si tienen tránsito confirmado (exoplanetas confirmados) o no (curvas de luz sin tránsito).

El siguiente paso por realizar es una limpieza de datos, eliminando los valores erróneos y nulos de las curvas de luz, así como la normalización de las longitudes de los datos.

La normalización, en esta primera generación de los modelos sin procesar las curvas de luz, simplemente ha consistido en eliminar valores de flujo erróneos (valores NaN) y ajustar las longitudes a un mismo valor, completando con valores de flujo nulos.

Por último, antes de pasar a la fase de entrenamiento de los modelos, se divide la muestra aleatoriamente en un 80% para el conjunto de entrenamiento, un 10% para el conjunto de validación y el 10% restante para el conjunto de prueba, como se muestra en la Tabla 3. Como esta fase es una primera aproximación con los datos en bruto, no se ha realizado balance de clases.

Tabla 3. *Distribución inicial de los conjuntos de Entrenamiento, Validación y Prueba*

Conjunto	Total
Entrenamiento	664
Validación	84
Prueba	84

El objetivo implícito de este primer ejercicio es definir las bases de los parámetros de F1 score y eficiencia de los modelos, para definir un punto de partida con el peor resultado posible para ver las mejoras que producen cada método estudiado durante este proyecto.

Con esta muestra, procedemos a desarrollar los modelos de XGBoost y Random Forest sin optimización de parámetros.

4.2.2. Modelo XGBoost

Las tablas 4 y 5 muestran los resultados de los conjuntos de entrenamiento y validación del modelo XGBoost usando los datos de las curvas de luz de Kepler sin realizar ningún preprocesamiento a las mismas. Se puede apreciar en los resultados del entrenamiento valores cercanos a 1, a diferencia del conjunto de validación en donde el valor más alto obtenido es de 0.81 para la métrica de precisión de la clase No Tránsito. Esto indica que, a diferencia de un entrenamiento casi perfecto, existe un overfitting en los datos, es decir el modelo ha aprendido los valores que se le han pasado para entrenar. Este comportamiento implica que el modelo se ajusta excesivamente a las particularidades del conjunto de entrenamiento y no ha sido capaz de identificar las tendencias y patrones propias de los datos, lo hace que el rendimiento del modelo con los datos no vistos (validación) sea bajo, como se aprecia en la Tabla 5.

Tabla 4. *Resultados del modelo XGBoost para el conjunto de entrenamiento con datos sin preprocesar*

	precision	**recall**	**F1-score**	**support**
No tránsito	0,99	1,00	0,99	464
Tránsito	0,99	0,97	0,98	200
accuracy			0,99	664
macro avg	0,99	0,99	0,99	664
weighted avg	0,99	0,99	0,99	664

Se aprecia que hay un overfitting en los valores obtenidos.

Tabla 5. *Resultados del modelo XGBoost para el conjunto de validación con datos sin preprocesar*

	precision	**recall**	**F1-score**	**support**
No tránsito	0,81	0,73	0,73	64
Tránsito	0,35	0,45	0,39	20
Accuracy			0,67	84
macro avg	0,58	0,59	0,58	84
weighted avg	0,70	0,67	0,68	84

Se aprecia que la media armónica entre precision y recall (F1-score) para la clase 1.0 (curvas con tránsitos) es bastante baja.

4.2.3. Modelo Random Forest

Las tablas 6 y 7 muestran los resultados de los conjuntos de entrenamiento y validación del modelo RandomForest usando los datos de las curvas de luz de Kepler sin ser preprocesadas. Se puede apreciar que al igual que el modelo anterior, los resultados del entrenamiento tiene valores cercanos a 1, lo cual indica que existe un overfitting en los datos. Se puede apreciar que el rendimiento del modelo para el conjunto de validación es bajo, como se aprecia en la tabla 7.

Tabla 6. *Resultados del modelo Random Forest para el conjunto de entrenamiento con datos sin preprocesar.*

	precision	**recall**	**F1-score**	**support**
No tránsito	0,98	0,99	0,99	464
Tránsito	0,98	0,95	0,97	200
accuracy			0,98	664
macro avg	0,98	0,97	0,98	664
weighted avg	0,98	0,98	0,98	664

Se aprecia que hay un overfitting en los valores obtenidos.

Tabla 7. *Resultados del modelo Random Forest para el conjunto de validación con datos sin preprocesar.*

	precision	**recall**	**F1-score**	**support**
No tránsito	0,81	0,73	0,77	64
Tránsito	0,35	0,45	0,39	20
accuracy			0,67	84
macro avg	0,58	0,59	0,58	84
weighted avg	0,70	0,67	0,68	84

Se aprecia que el F1-score para la clase Tránsito es bastante bajo.

Para esta primera etapa de utilización de los modelos clasificadores con los datos sin preprocesar de las curvas de luz, obtenemos un modelo clasificador limitado con unos valores de F1-score para la clase Tránsito en el conjunto de validación bajos (nuestro objetivo es maximizar esta métrica como se mencionó al inicio de esta sección), por lo que consideramos que los resultados de los modelos XGBoost y RandomForest con datos sin preprocesar se pueden mejorar; para ello, se explorará un enfoque que incluya el preprocesamiento de los datos, así como un análisis de parámetros funcionales de los modelos.

4.3. Preprocesamiento de las curvas de luz

Analizando los resultados obtenidos por los modelos con las curvas de luz sin preprocesar, se ha decidido realizar un tratamiento a las mismas que permita obtener mejores resultados en el modelo. Parte de este preprocesamiento será la reducción del nivel de ruido en la señal y la variabilidad del flujo inherente de la estrella mediante el mecanismo de *prewhitening* (Lenz & Breger, 2004). Adicionalmente, se realizará un análisis de las componentes principales

funcionales (en inglés Functional Principal Component Analysis) mediante la funcionalidad FPCA[2]. A continuación, se presentan los mecanismos realizados en cada proceso.

4.3.1. Prewhitening

El mecanismo de *prewhitening* es ampliamente usado en el tratamiento de curvas de luz de estrellas variables, tal como se ha descrito en la sección 2.2. Para realizar el proceso de *prewhitening* de las señales, hemos tomado como punto de partida el algoritmo desarrollado por Hey et al. (2021) y lo hemos adaptado a las características de nuestra investigación.

A continuación, se presenta el flujo del proceso de *prewhitening*, cuyo algoritmo puede consultarse en el Anexo A: código A2.

1. Extraer datos de las curvas de luz originales. A partir de las curvas de luz originales de la misión Kepler, se ha desarrollado un algoritmo que itera en cada archivo FITS y extrae los datos de las columnas TIME, PDCSAP_FLUX y PDCSAPFLUX_ERR. Estos datos son revisados por el algoritmo para eliminar de la muestra los campos vacíos y posteriormente son llevados a un *dataframe* de Pandas. Cada *dataframe* generado se guarda en un archivo CSV que será la entrada para el proceso de *prewhitening*.
2. *Prewhitening*: el proceso utilizado se compone de los siguientes pasos.
    a. Cálculo del espectro de amplitud. A partir de una serie de tiempo y flujo, se ha calculado el espectro de amplitud para identificar las frecuencias dominantes presentes en la señal.

       El proceso calcula los tiempos máximos y mínimos de la señal, así como la resolución espectral, correspondiente a la separación máxima entre dos frecuencias, según se muestra en la Ecuación 1:

$$R = \frac{1}{\Delta t} \tag{1}$$

[2] Librería scikit-fda de Python

donde Δt se ha calculado como la diferencia entre el tiempo máximo y el mínimo.

Adicionalmente, se ha calculado la frecuencia más alta de la señal siguiendo el criterio de Nyquist, H. (1928), el cual establece que la frecuencia máxima que puede ser representada correctamente en una serie temporal está limitada por la mitad de la frecuencia de muestreo (Ecuación 2):

$$f_{max} = \frac{0.5}{\delta t} \tag{2}$$

donde δt corresponde al muestreo temporal, el cual se calcula a partir de la mediana de las diferencias entre los tiempos adyacentes en la serie temporal.

El proceso calcula el periodograma de la señal mediante la herramienta Lomb-Scargle de la librería Astropy, el cual es una herramienta estadística diseñada para analizar series temporales irregulares y no uniformemente muestreadas. A partir de esta herramienta se calcula el espectro de potencias para un arreglo de frecuencias compuesto por la frecuencia mínima y la frecuencia máxima calculada previamente mediante el criterio de Nyquist. Así mismo, se ha seleccionado el tipo de normalización *standard*, mediante el cual el periodograma se normaliza por los residuos de los datos alrededor del modelo de referencia constante:

$$P_{standard}(f) = \frac{\chi^2_{ref} - \chi^2(f)}{\chi^2_{ref}} \tag{3}$$

La potencia resultante P es una cantidad adimensional que se encuentra en el rango $0 \leq P \leq 1$.
Finalmente se calcula las amplitudes de la señal siguiendo la ecuación 4:

$$a = (\sqrt{P}) \times (fct) \quad (4)$$

donde P es el espectro de potencia y *fct* es el factor de normalización para las amplitudes en el espectro de amplitud.

b. Estimación de la frecuencia (f_0), amplitud (a_0) y fase (ϕ_0) iniciales de la señal.

Tomando como entrada las frecuencias y amplitudes de la señal, se utiliza la técnica de interpolación parabólica de tres puntos para encontrar la frecuencia (f_0) que corresponde al pico de mayor potencia en el espectro de amplitud. Mediante esta técnica se toman tres puntos alrededor del pico más alto: (f_1, y_1), (f_2, y_2) y (f_3, y_3), donde y_2 corresponde al valor máximo de la amplitud (a). Luego, se calculan los términos t_1 y t_2 de acuerdo con las siguientes ecuaciones:

$$t_1 = (y_2 - y_3) \times (f_2 - f_1)^2 - (y_2 - y_1) \times (f_2 - f_3)^2 \quad (5)$$

$$t_2 = (y_2 - y_3) \times (f_2 - f_1) - (y_2 - y_1) \times (f_2 - f_3) \quad (6)$$

A partir de estos valores se calcula el pico máximo de amplitud en el espectro de frecuencia:

$$\nu_{pico} = f_2 - \frac{0.5\, t_1}{t_2} \quad (7)$$

Con la frecuencia f_0 obtenida en el paso anterior, se calcula la amplitud mediante la herramienta Lomb-Scargle de la forma:

$$a_0 = \sqrt{P\, f(0)} \times \sqrt{\frac{4}{N}} \quad (8)$$

donde, P es la potencia de la señal en la frecuencia $f(0)$ y N es el número total de puntos en la serie temporal.

La fase se calcula mediante una Transformada de Fourier Discreta (DFT) mediante la ecuación 10:

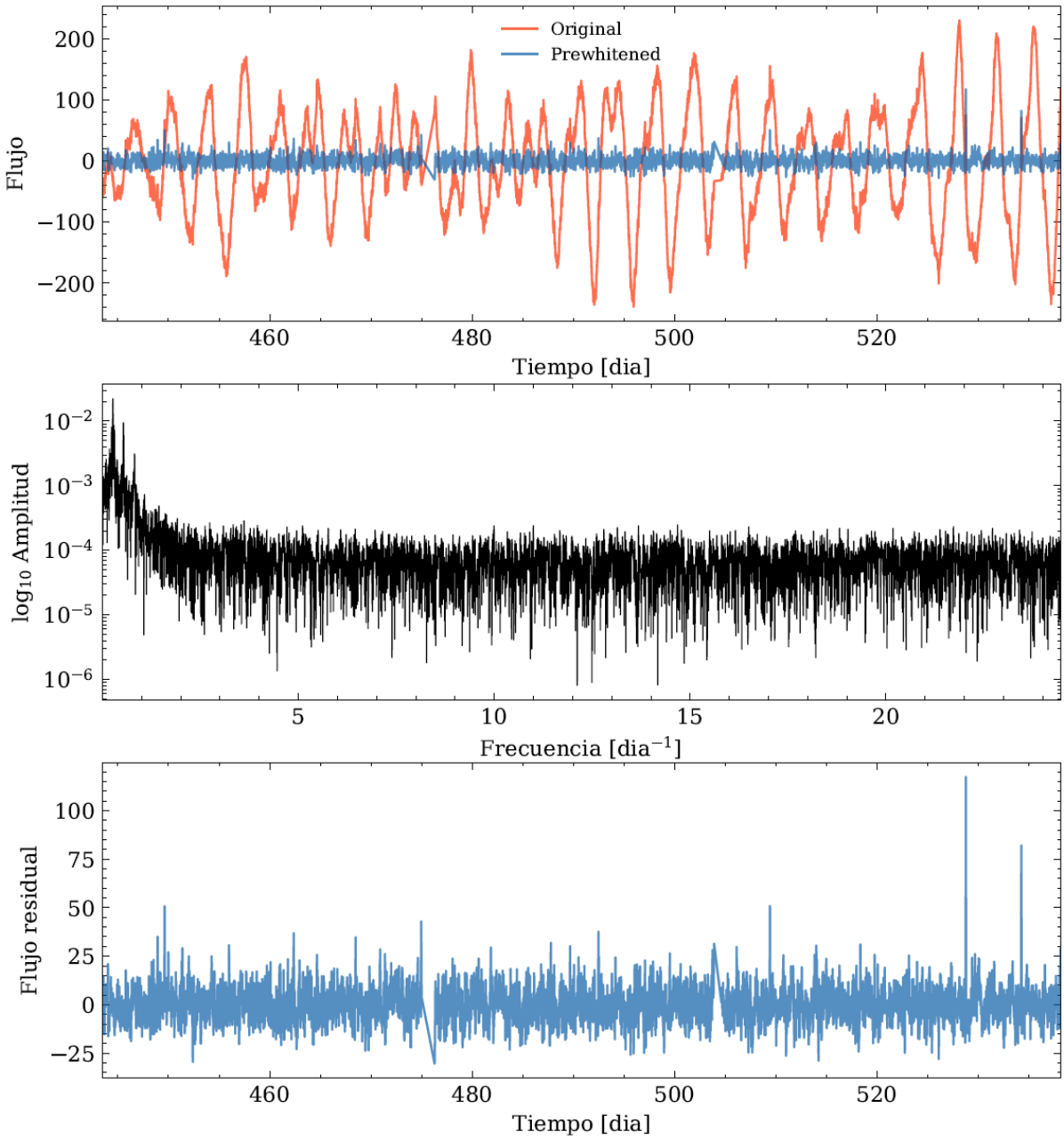
$$\phi_0 = arctan\left(\frac{\sum_{i=1}^{N} y_i \, sin(2\pi f x_i)}{\sum_{i=1}^{N} y_i \, cos(2\pi f x_i)}\right) \quad (10)$$

donde, x_i son los valores de tiempo, y_i son los valores de flujo, N es el número total de puntos en la serie temporal y f es la frecuencia para la cual se está calculando la fase.

Finalmente, obtenidas la (f_0), amplitud (a_0) y fase (ϕ_0) iniciales se realiza el proceso iterativo que busca señales periódicas en el residuo de la serie temporal y realiza el proceso de eliminación de las frecuencias significativas considerando el condicional que la amplitud sea mayor que el nivel de ruido.

En las figuras 29 y 30 se muestran los resultados del proceso de Prewhitening para el objeto kplr004144238, clasificado como estrella variable sin tránsito de exoplaneta.

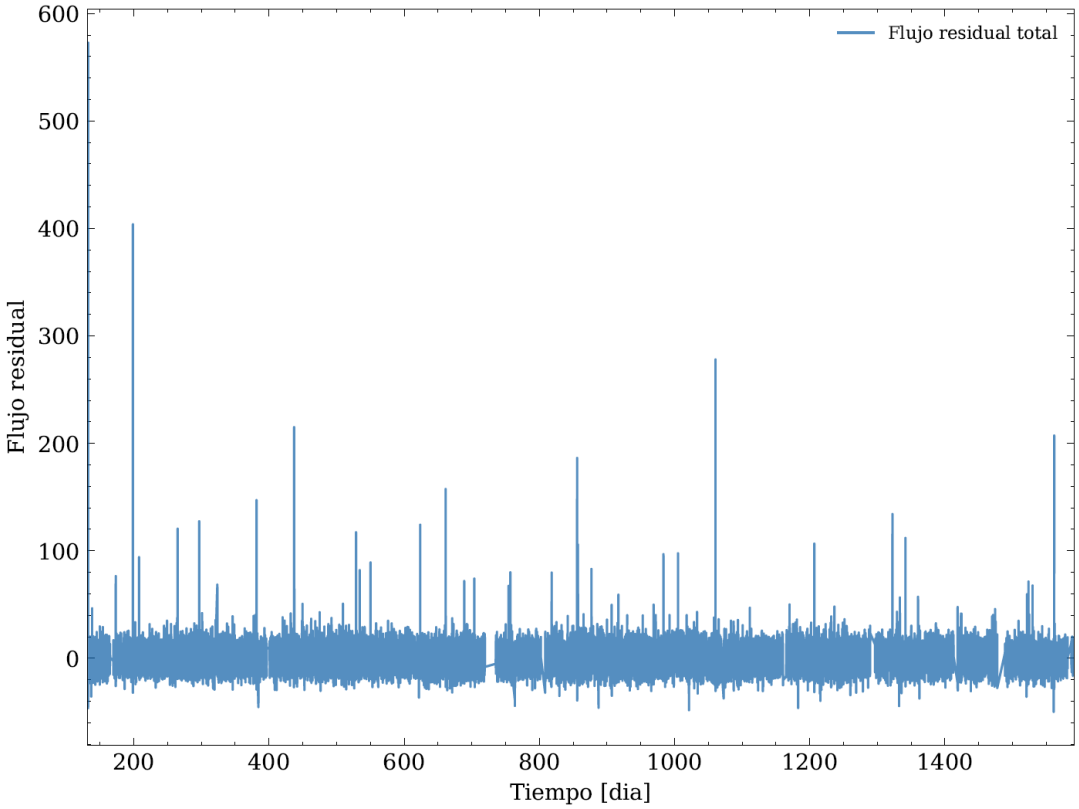
Figura 29. *Proceso de prewhitening aplicado a la curva de luz del objeto kplr004144238.*



Panel superior: Diagrama de Flujo Vs Tiempo de la curva de luz original y residual que se ha obtenido en el proceso de prewhitening. Panel medio: Espectro de amplitudes y frecuencias. Panel Inferior: curva de luz residual.

Fuente: Elaboración propia

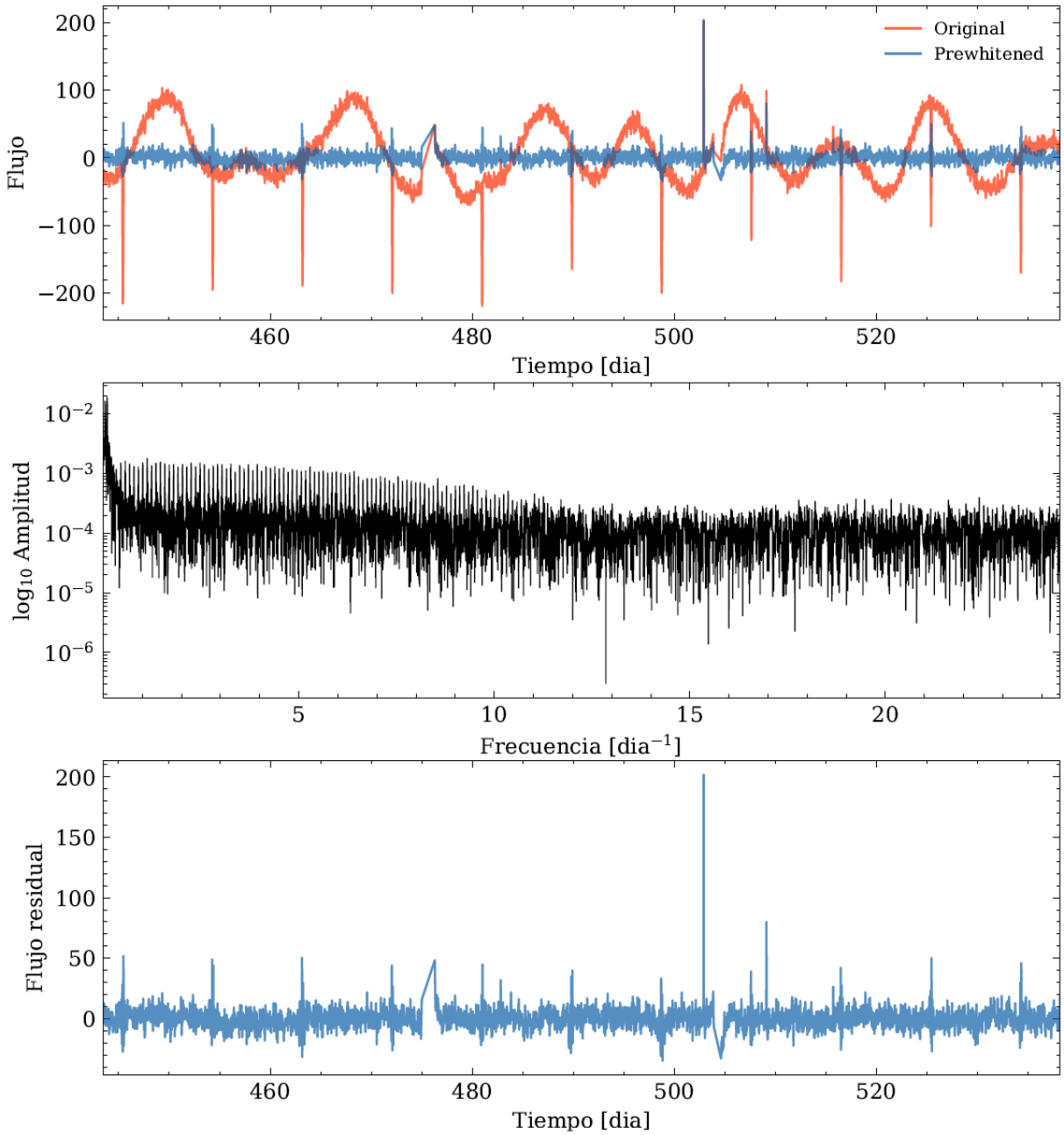
Figura 30. *Curva de luz residual del objeto kplr004144238 luego de aplicar el prewhitening*



Fuente: Elaboración propia

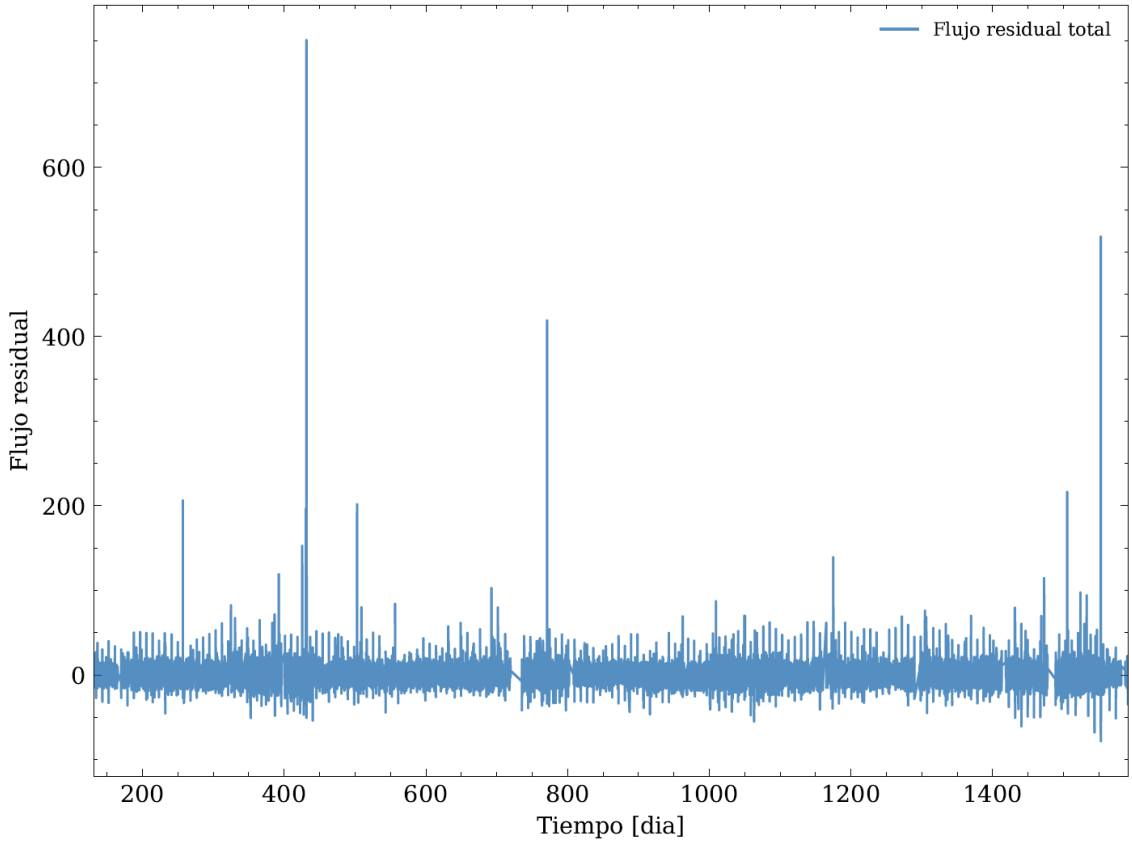
Así mismo, en las figuras 31 y 32 se muestran los resultados del proceso de Prewhitening para el objeto kplr000757450, clasificado como estrella variable con tránsito confirmado de un exoplaneta con periodo orbital de 8.8849 días (Gajdoš et al. 2019).

Figura 31. *Proceso de prewhitening aplicado a la curva de luz del objeto kplr000757450*



Panel superior: Diagrama de Flujo Vs Tiempo de la curva de luz original y residual que se ha obtenido en el proceso de prewhitening en la que se aprecian las variaciones de la curva de luz debida al evento de tránsito. Panel medio: Espectro de amplitudes y frecuencias. Panel Inferior: curva de luz residual. Fuente: Elaboración propia

Figura 32. *Curva de luz residual del objeto kplr000757450 luego de aplicar el prewhitening*



Se aprecia que se ha disminuido notablemente la variabilidad de la señal respecto a la curva de luz original, con lo cual además de eliminar el ruido, se busca facilitar el análisis de las curvas de luz.

Fuente: Elaboración propia

Adicionalmente, en el análisis realizado a las curvas de luz de la misión, hemos identificado estrellas que tienen tránsitos de exoplanetas confirmados y tránsitos catalogados como falsos positivos o candidatos sin confirmación, lo que origina curvas de luz duplicadas para un mismo objeto (positivos y negativos). Se ha revisado manualmente y confirmado esta situación en la base de datos de exoplanetas de la NASA Para remediar esta duplicidad y evitar conflictos en futuros procesos, se han eliminado estas curvas de luz del grupo de estrellas sin tránsito (negativos). La Tabla 8 muestra los objetos con esta situación.

Tabla 8. *Objetos de Kepler con curvas de luz duplicadas.*

	Conjunto de datos Tránsito	**Grupo de datos No Tránsito**	
Kepler id	**Exoplanetas confirmados**	**Candidatos no confirmados**	**Falsos positivos**
kplr005792202	3	2	0
kplr002438513	1	0	1
kplr005351250	5	0	1
kplr004947556	1	1	0
kplr002975770	1	1	0
kplr005688683	1	1	0
kplr010019708	1	0	1
kplr009785921	1	0	1

4.3.2. Reducción de dimensionalidad, generación de nuevas características y creación de nuevos conjuntos de datos

Uno de los principales problemas observados al ejecutar la totalidad de los datos en los algoritmos clasificadores, es que no se puede discernir con exactitud si los algoritmos clasificadores están teniendo en cuenta que los datos del fujo siguen un orden temporal y que el evento que queremos detectar esta correlacionado con esta temporalidad en la naturaleza del flujo, puesto que el proceso del tránsito del exoplaneta afecta de forma específica al flujo como se expuso en la sección 1.1.

Los modelos de clasificación de las curvas de luz sin preprocesar no consideran el orden de los valores frente al tiempo y clasifican los valores de estos, pero no su dependencia temporal, esto puede ser una razón por la que el modelo obtenido sin considerar la serie temporal podría no ser óptimo para nuestra investigación.

En añadido al problema de la dependencia temporal de los valores, el procesamiento de toda la información incluida en la curva de luz genera que el proceso de creación de los modelos requiera un alto coste de tiempo computacional.

Para mejorar ambos puntos, se procede a realizar un análisis de las componentes principales funcionales (en inglés Functional Principal Component Analysis) mediante la funcionalidad FPCA. (Ramsay et al. 2005).

Los métodos de análisis de componentes principales son métodos para reducir la dimensionalidad de un conjunto de datos proyectándolos en una dimensión menor, de tal manera que se pueda capturar una mayor variabilidad con menos volumen de datos. En el caso concreto del FPCA se diseña datos funcionales, que son aquellas funciones continuas cuyos valores tienen un orden concreto, como en nuestro caso, que se trata de series temporales de valores de flujo en las que el orden de los valores es importante.

Uno de los requisitos de FPCA es que todos los grupos de datos deben tener el mismo número de valores, por lo que las curvas de luz se deben ajustar para que tengan la misma cantidad de mediciones. En lugar de añadir valores no reales para equiparar los datos al número mayor, se ha realizado un análisis mediante un diagrama de caja, o boxplot para representar gráficamente la serie de datos e identificar la mediana, los cuartiles de los datos, y los valores atípicos u outliers. Las figuras 33 y 34 muestran los diagramas de caja para los conjuntos de datos de No Tránsitos y Tránsitos respectivamente.

A partir de este análisis, se considera que las curvas de luz con menos de 10000 flujos observados son datos atípicos de la muestra, por lo que se ajustan los datos para dejar en cada curva los primeros 10000 flujos observados.

Figura 33. *Diagrama de caja del conjunto de datos de curvas de luz sin tránsitos.*

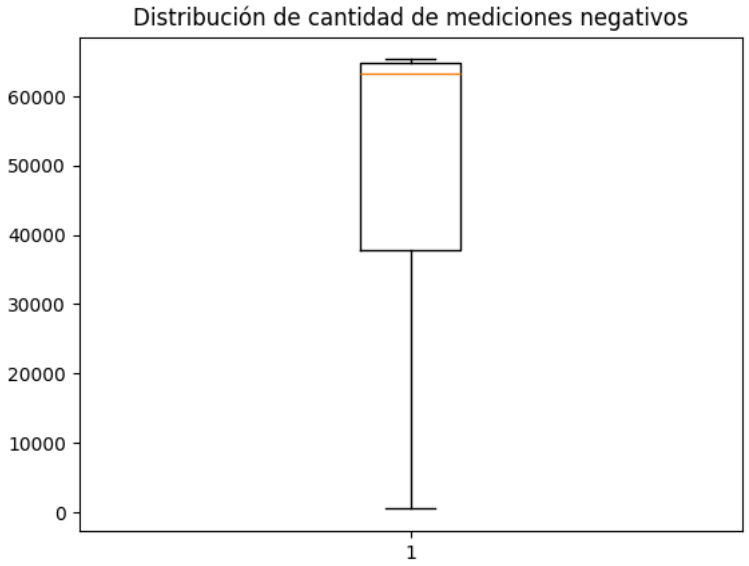
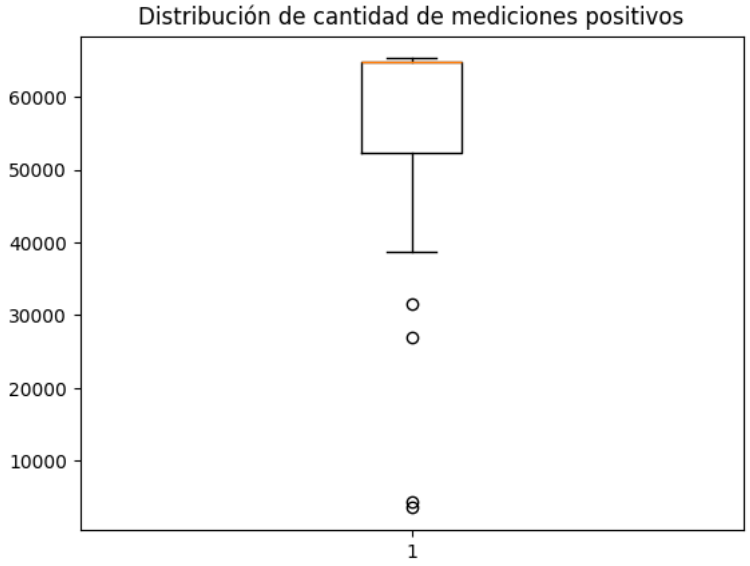


Figura 34. *Diagrama de caja del conjunto de datos de curvas de luz con tránsitos.*



El siguiente paso es el de definir e inicializar un objeto FPCA para transformar la serie de tiempo de las curvas en 10 componentes principales para capturar la mayor parte de la variabilidad en los datos. A partir de la elección de dichos componentes se hace un ajustado de los datos y se evalúa la importancia de cada uno de los componentes. Los resultados muestran que el primer componente explica aproximadamente el 96.68% de la variabilidad total, seguido por el segundo componente con un 2.34%, el resto de los componentes

contribuyen en fracciones muy pequeñas hasta llegar al 100%. Esto indica que el proceso de reducción de dimensionalidad ha sido muy eficiente al concentrar la mayor parte de la información en los primeros pocos componentes y no son necesarios más de 10 componentes o incluso se podrían haber utilizado menos componentes.

Una vez realizado el análisis de componentes, añadimos una serie de parámetros estadísticos adicionales, quedando las características (*features*) de los conjuntos de datos según se muestran en la Tabla 9.

Tabla 9. *Características para cada curva de luz de los conjuntos No Tránsito y Tránsito*

Tipología	Descripción	Cantidad
Componentes principales FPCA	variabilidad en los datos a partir del procesamiento FPCA (ver 5.2.3)	10
Características simples	Valores estadísticos de la serie de datos: suma, máximo, mínimo, media, mediana, desviación estándar	6
Características basadas en bandas de Bollinger	Medición de la volatilidad de los datos. Se calcula cuantas mediciones del flujo se encuentran a n-veces la desviación estándar tomando como valores para n: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 50, 100	24
Características basadas en percentiles	Medición de la distribución de los datos a partir de los percentiles 1, 2, 3, 4, 5, 10, 15, 20, 25, 30, 40, 45, 55, 60, 70, 80, 85, 90, 95, 96, 97, 98, 99	23

El resultado final es un conjunto de datos con 63 características para cada curva de luz de los conjuntos de datos de No Tránsito y Tránsito.

Una vez realizado el proceso de reducción de dimensionalidad mediante FPCA y de agregar las nuevas características a los datos, se generan los nuevos conjuntos de datos que serán la entrada para el modelo (Anexo A: código A3).

4.4. Modelos de clasificación utilizando curvas preprocesadas.

Utilizando los datos preprocesados a partir del *prewhitening* y del proceso de reducción de dimensionalidad FPCA y generación de nuevas características, obtenemos los resultados que se muestran en esta sección. Es importante mencionar que gracias a la reducción de dimensionalidad realizada mediante FPCA, se han podido ejecutar varias configuraciones distintas de los parámetros de los distintos algoritmos clasificadores, ya que el tiempo de computación se ha reducido drásticamente comparado con las versiones anteriores que utilizaban los datos en bruto. En la Tabla 10 se muestra la distribución de datos en los conjuntos de entrenamiento, validación y prueba luego de realizar el preprocesamiento de datos.

Tabla 10. *Distribución de los conjuntos de Entrenamiento, Validación y Prueba con datos preprocesados*

Conjunto	Total
Entrenamiento	450
Validación	150
Prueba	150

Los hiperparámetros optimizados de cada modelo usando la librería Optuna (Akiba, et al 2019) se mostrarán en la sección 4.5. A continuación, se muestran los resultados del conjunto de entrenamiento y validación de los modelos XGBoost y Random Forest usando las curvas que han resultado del preprocesamiento de estas.

4.4.1. Modelo XGBoost

Las tablas 11 y 12 muestran los resultados de los conjuntos de entrenamiento y validación del modelo XGBoost usando los datos de las curvas de luz de Kepler después de realizar el preprocesamiento a las mismas. Se aprecia que, a diferencia de los resultados obtenidos con los datos sin preprocesar, el modelo no presenta un sobreajuste (overfitting), sin embargo, se

considera que el modelo puede mejorar aún más si se optimizan los hiperparámetros y se busca maximizar las métricas de interés en nuestra investigación.

Tabla 11. *Resultados del modelo XGBoost para el conjunto de entrenamiento con datos preprocesados.*

	precision	**recall**	**F1-score**	**support**
No Tránsito	0.98	0.95	0.97	301
Tránsito	0.91	0.97	0.94	149
accuracy			0.96	450
macro avg	0.94	0.96	0.95	450
weighted avg	0.96	0.96	0.96	450

Tabla 12. *Resultados del modelo XGBoost para el conjunto de validación con datos preprocesados.*

	precision	**recall**	**F1-score**	**support**
No Tránsito	0.87	0.77	0.82	101
Tránsito	0.62	0.76	0.68	49
accuracy			0.77	150
macro avg	0.74	0.76	0.75	150
weighted avg	0.79	0.77	0.77	150

4.4.2. Modelo Random Forest

De manera similar, se ha realizado el entrenamiento del modelo Random Forest con datos preprocesados. Las tablas 13 y 14 muestran los resultados de los conjuntos de entrenamiento y validación del modelo Random Forest usando los datos de las curvas de luz de Kepler después de realizar el preprocesamiento a las mismas mediante la técnica de *prewhitening* y

FPCA. Al igual que los resultados obtenidos con XGBoost, se aprecia que el modelo no se encuentra sobreajustado.

Tabla 13*. Resultados del modelo Random Forest para el conjunto de entrenamiento con datos preprocesados*

	precision	**recall**	**F1-score**	**support**
No Tránsito	0.84	0.79	0.82	301
Tránsito	0.63	0.70	0.66	149
accuracy			0.76	450
macro avg	0.73	0.75	0.74	450
weighted avg	0.77	0.76	0.77	450

Tabla 14. *Resultados del modelo Random Forest para el conjunto de validación con datos preprocesados.*

	precision	**recall**	**F1-score**	**support**
No Tránsito	0.87	0.74	0.80	101
Tránsito	0.59	0.78	0.67	49
accuracy			0.75	150
macro avg	0.73	0.76	0.74	150
weighted avg	0.78	0.75	0.76	150

A nivel general, se observa una mejora significativa de las métricas de los modelos XGBoost y Random Forest para los conjuntos de datos preprocesados en comparación con los mismos modelos para los datos sin preprocesamiento.

Con el objetivo de mejorar los resultados de estos modelos, se realizará una optimización de los hiperparámetros de cada uno de ellos y se realizará una serie de experimentos para

encontrar la mejor solución. Lo anterior se podrá ver en la sección 4.5 Optimización de los modelos.

La comparativa completa de todos los resultados obtenidos en los distintos pasos de esta sección se puede encontrar en las Tabla 31 del capítulo 5 Resultados y análisis.

4.5.Optimización de los modelos

En esta sección se optimizarán los hiperparámetros mediante la librería Optuna, la cual ofrece funcionalidades de optimización automática de hiperparámetros de los modelos XGBoost y Random Forest diseñado para tareas de aprendizaje automático y otras tareas de optimización. Esta librería se encarga de encontrar los mejores valores de los hiperparámetros que maximicen o minimicen una función objetivo dada.

Optuna utiliza optimización bayesiana para encontrar el conjunto óptimo de valores de hiperparámetros, pero también realiza una búsqueda aleatoria que evita que el algoritmo termine la búsqueda en un máximo o mínimo local. Por otro lado, detiene temporalmente las configuraciones de hiperparámetros que encuentra candidatas a dar malos resultados, ahorrando tiempo de computación total.

Adicionalmente, se usará la funcionalidad Optuna Dashboard para ver de una forma gráfica la distribución de los experimentos que realiza Optuna y ver métricas de interés en el proceso de optimización. Las secciones 4.5.2 y 4.5.3 muestran las características y resultados principales del proceso para ambos modelos.

En las siguientes secciones se muestran los pasos y configuración realizada en Optuna para optimizar los hiperparámetros de los modelos XGBoost y Random Forest. El algoritmo utilizado para esta optimización y entrenamiento de los modelos se incluye en los anexos (Anexo A: códigos A4 y A5).

4.5.1. Balance de clases

Como se ha explicado en el apartado 1.1, nuestro modelo parte de un conjunto de datos desbalanceados, teniendo mayor cantidad de curvas de luz sin tránsitos que con tránsitos. Para corregir esta situación, hemos realizado un balance de los datos de tal forma que el número de datos en cada clase sea igual.

Para evitar que nuestro modelo tenga un sesgo hacia la clase mayoritaria, es decir la clase 0.0 (No Tránsito), hemos hecho un balance de las clases, tomando como entrada al modelo la misma cantidad de datos en cada clase, según se muestra en la Tabla 15.

Tabla 15*. Total de curvas de luz balanceadas según su clase*

Clase	Curvas de luz usadas en el modelo
No Tránsito	250
Tránsito	250

Dado que las clases se han balanceado, se ha realizado una nueva segmentación de los conjuntos de datos de Tránsito y No Tránsito en una distribución de Entrenamiento: 60%, Validación: 20% y Prueba: 20% según se muestra en la Tabla 16. Cabe destacar que el conjunto de entrenamiento se usará únicamente para entrenar el modelo, el conjunto de validación para optimizar los hiperparámetros y maximizar las métricas F1-score y accuracy y el conjunto de prueba solo se usará cuando se haya seleccionado el mejor modelo y no se le enseñará a este durante la fase de entrenamiento y optimización.

Tabla 16. *Distribución de clases balanceadas en conjuntos de Entrenamiento, Validación y Prueba*

Conjunto	Total
Entrenamiento	300
Validación	100
Prueba	100

4.5.2. Optimización de hiperparámetros en XGBoost

El primer paso en la optimización del modelo es el de definir los hiperparámetros base que serán usados en el entrenamiento del modelo y sobre los cuales Optuna realizará la búsqueda del valor máximo. Como se ha mencionado anteriormente, se ha decidido maximizar las métricas F1-score y accuracy del conjunto de datos de validación. Esta decisión es clave en la parametrización de Optuna, pues será sobre estas métricas con las que se explorará diferentes combinaciones de valores de hiperparámetros del modelo en cada iteración. La Tabla 17 muestra los hiperparámetros definidos para el entrenamiento de XGBoost en donde se puede apreciar los hiperparámetros seleccionados del modelo, el rango de valores de cada hiperparámetro que Optuna usará en la optimización y la descripción de cada uno de ellos.

Tabla 17. *Hiperparámetros iniciales para optimización de XGBoost*

Hiperparámetros del modelo (nombre en librería)	**Rango de valores**	**Descripción**
max_depth	3 - 20 log = True	Controla la profundidad máxima de cada árbol de decisión en el modelo. Se explorarán valores desde 3 hasta 20
n_estimators	100 - 1000	Número de árboles que se construyen en el modelo. Se explorarán valores desde 100 hasta 1000
learning_rate	0.01 - 0.3 log = True	Controla la contribución de cada árbol individual en el modelo final, permitiendo determinar el peso que cada árbol adicional tiene al actualizar el modelo en cada iteración. Se explorarán valores desde 0.01 hasta 0.3
min_child_weight	20 - 45	Controla la cantidad mínima de peso necesaria en cada nodo hoja. Se explorarán valores desde 20 hasta 45
lambda	1e-8 - 1.0 log = True	Controla la regularización L2 aplicada a los pesos de los árboles, ayudando a prevenir el sobreajuste penalizando los modelos que tienen coeficientes de gran magnitud. Se explorarán valores desde 1×10^{-8} hasta 1.0

Hiperparámetros del modelo (nombre en librería)	Rango de valores	Descripción
alpha	1e-8 - 1.0 log = True	Controla la regularización L1 aplicada a los pesos de los árboles, ayudando a prevenir el sobreajuste. Se explorarán valores desde 1×10^{-8} hasta 1.0
subsample	0.5 - 1.0 step = 0.1	Controla la fracción de muestras de entrenamiento utilizadas para construir cada árbol, introduciendo variabilidad y ayudando a prevenir el sobreajuste. Se explorarán valores desde 0.5 hasta 1.0
gamma	1e-8 - 1.0 log = True	Controla la reducción mínima necesaria en la función de pérdida para permitir una partición adicional en los árboles de decisión. Se explorarán valores desde 1×10^{-8} hasta 1.0
colsample_bytree	0.5 - 1.0 step = 0.1	Controla la fracción de características utilizadas para construir cada árbol, introduciendo variabilidad y ayudando a prevenir el sobreajuste. Se explorarán valores desde 0.5 hasta 1.0
scale_pos_weight	0.1 - 10.0 log = True	Permite manejar el desbalance de clases en problemas de clasificación binaria ajustando el peso de las instancias positivas en la función de pérdida. Se explorarán valores desde 0.1 hasta 10.0
max_delta_step	0 - 10	Controla el paso máximo permitido para cada árbol. Se explorarán valores desde 0 hasta 10.0
colsample_bylevel	0.5 - 1.0 step = 0.1	Permite ajustar cuántas características están disponibles para cada nivel del árbol. Se explorarán valores desde 0.5 hasta 1.0

El valor log = True indica que la búsqueda se realizará en una escala logarítmica, lo que permite explorar más eficientemente el espacio de hiperparámetros. El valor step = 0.1 indica que se explorará un rango de valores en incrementos de 0.1.

Una vez definidos los hiperparámetros del modelo y el rango de valores que Optuna deberá explorar en el proceso de optimización, se definen las características mismas del proceso, denominado estudio en Optuna. La Tabla 18 muestra la configuración realizada en el estudio.

Tabla 18. *Configuración de parámetros del proceso de Optuna para XGBoost*

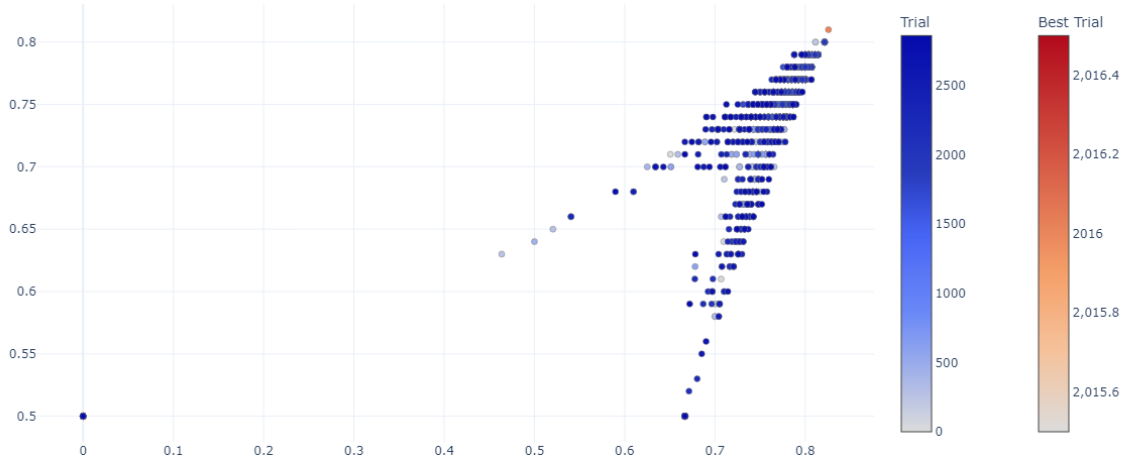
Parámetros	Valores	Descripción
Método de muestreo	TPESampler	Utiliza el algoritmo TPE (Tree-structured Parzen Estimator) para optimizar de manera eficiente los hiperparámetros, explorando y explotando el espacio de búsqueda basado en distribuciones de probabilidad.
Objetivo de la función	['maximize', 'maximize']	Objetivo que alcanzar en la función Optuna. Se ha parametrizado como métricas a maximizar f1_score_validation y accuracy_validation (conjunto de datos de validación)
Iteraciones	2865	Número de iteraciones que realizará Optuna en la función de optimización

La Figura 35, muestra la distribución de valores de cada objetivo obtenido por cada iteración del proceso de optimización del modelo XGBoost realizado en Optuna.

Como resultado, se ha obtenido que la iteración 2016 ha encontrado la configuración de los hiperparámetros que da un valor más alto para ambas las métricas f1-score y accuracy del conjunto de datos de validación, los cuales corresponden con 0.8256880733944953 y 0.81 respectivamente.

La Tabla 19 muestra los valores obtenidos para los hiperparámetros del modelo XGBoost.

Figura 35. *Distribución de valores obtenidos en cada métrica objetivo en la optimización de XGBoost*



En el eje X se muestran los valores de la métrica accuracy_validation y en el eje Y se muestran los valores obtenidos para f1_score_validation. Nótese cómo a medida que Optuna avanza en las iteraciones se obtienen valores más altos para las métricas objetivo.

Fuente: Elaboración propia mediante Optuna Dashboard

Tabla 19. *Valores obtenidos para los hiperparámetros de XGBoost en el proceso de optimización*

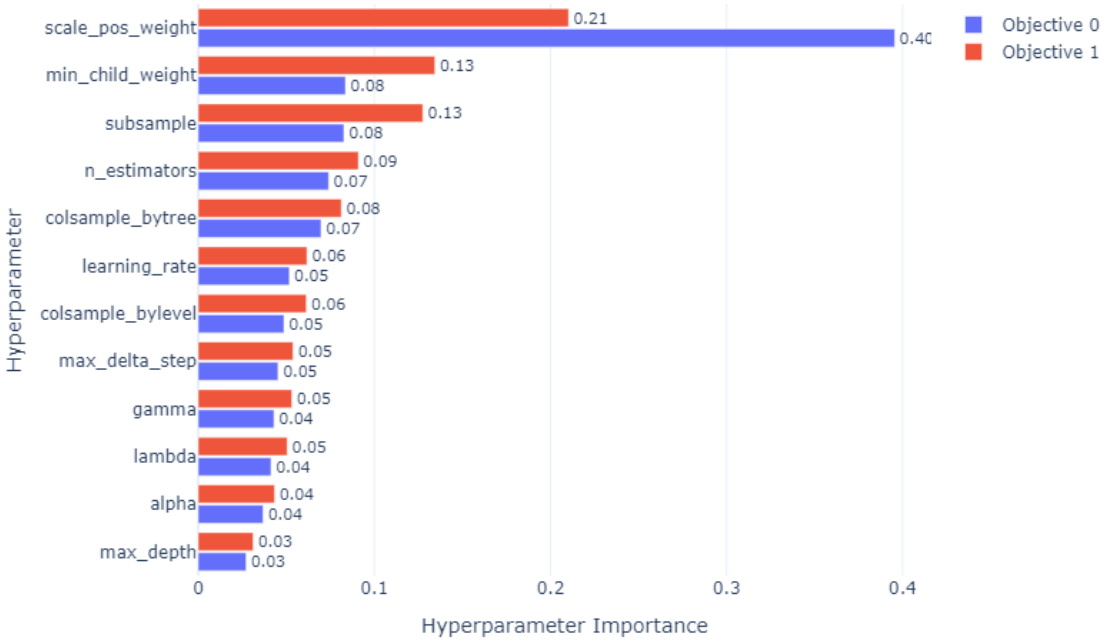
Hiperparámetros del modelo (nombre en librería)	Valores
max_depth	13
n_estimators	752
learning_rate	0.0144678555810096
min_child_weight	22
lambda	7.893735442162978e-07
alpha	0.00026942770322221877
subsample	1.0

Hiperparámetros del modelo (nombre en librería)	Valores
gamma	1.4290576007652381e-08
colsample_bytree	0.8
scale_pos_weight	1.424066040026292
max_delta_step	3
colsample_bylevel	0.9

Adicionalmente, la Figura 36 muestra la contribución de cada hiperparámetro al modelo XGBoost donde se puede apreciar cómo cada hiperparámetro ha aportado al cálculo de cada una de las métricas objetivo, es decir, f1_score_validation y accuracy_validation.

Figura 36. *Contribución de cada hiperparámetro al modelo XGBoost*

Objective 0 corresponde a la métrica f1_score_validation y Objective 1 corresponde a la métrica accuracy_validation



Fuente: Elaboración propia mediante Optuna Dashboard

Las tablas 20 y 21 muestran los resultados obtenidos en entrenamiento y validación para el modelo XGBoost a partir de los hiperparámetros optimizados mediante Optuna.

Tabla 20. *Resultados del modelo XGBoost para el conjunto de entrenamiento con hiperparámetros optimizados.*

	precision	**recall**	**F1-score**	**support**
No Tránsito	0.85	0.67	0.75	146
Tránsito	0.74	0.89	0.81	154
accuracy			0.78	300
macro avg	0.80	0.78	0.78	300
weighted avg	0.79	0.78	0.78	300

Tabla 21. *Resultados del modelo XGBoost para el conjunto de validación con hiperparámetros optimizados.*

	precision	**recall**	**F1-score**	**support**
No Tránsito	0.88	0.72	0.79	50
Tránsito	0.76	0.90	0.83	50
accuracy			0.81	100
macro avg	0.82	0.81	0.81	100
weighted avg	0.82	0.81	0.81	100

4.5.3. Optimización de hiperparámetros en Random Forest

Al igual que el proceso de optimización del modelo XGBoost descrito en la sección anterior, para optimizar el modelo Random Forest se define como primer paso la selección de los hiperparámetros a optimizar y los rangos de valores que Optuna usará para buscar maximizar los objetivos establecidos, los cuales siguen siendo las métricas F1-score y accuracy del

conjunto de datos de validación. La Tabla 22 muestra los hiperparámetros definidos para el entrenamiento de Random Forest y los valores posibles que puede tomar cada uno de ellos.

Tabla 22. *Hiperparámetros iniciales para optimización de Random Forest*

Hiperparámetros del modelo (nombre en librería)	Rango de valores	Descripción
max_depth	2 - 300 log = True	Controla la profundidad máxima de cada árbol de decisión en el modelo. Se explorarán valores desde 2 hasta 300
n_estimators	100 - 1000	Número de árboles que se construyen en el modelo. Se explorarán valores desde 100 hasta 1000
min_samples_split	2 - 45	Número mínimo de muestras necesarias para dividir un nodo. Se explorarán valores desde 2 hasta 45
min_samples_leaf	1 - 45	Número mínimo de muestras necesarias para estar en un nodo hoja. Se explorarán valores desde 1 hasta 45
min_weight_fraction_leaf	0.0 - 0.5	Fracción ponderada mínima de las sumas totales (de todas las muestras) necesarias para estar en un nodo hoja. Se explorarán valores desde 0 hasta 0.5
bootstrap	True, False	Afecta la variabilidad de los modelos y su capacidad de generalización. Si es True, utiliza muestras de arranque al construir árboles.
max_samples	Si bootstrap = True ⇒ 0.5, 1.0	Controla el tamaño del subconjunto de datos utilizado para entrenar cada árbol. Parámetro condicional que depende del valor de parámetro Bootstrap. Si Bootstrap = True, se explorarán valores desde 0.5 hasta 1.0.

Hiperparámetros del modelo (nombre en librería)	Rango de valores	Descripción
max_features	"auto", "sqrt", "log2", None	Número de características a considerar cuando se busca la mejor división. "auto": Usa todas las características. "sqrt": Usa la raíz cuadrada del número total de características. "log2": Usa el logaritmo base 2 del número total de características. None: Usa todas las características.
criterion	"gini", "entropy"	Función utilizada para medir la calidad de una división. "gini": Impureza de Gini. Mide la probabilidad de clasificación incorrecta "entropy": Ganancia de información (o entropía). Mide la cantidad de incertidumbre o aleatoriedad en los datos y cómo esta se reduce después de una división.

El valor log = True indica que la búsqueda se realizará en una escala logarítmica, lo que permite explorar más eficientemente el espacio de hiperparámetros. El valor step = 0.1 indica que se explorará un rango de valores en incrementos de 0.1.

La Tabla 23 muestra la configuración realizada en el proceso de optimización de Optuna para Random Forest, la cual es similar a la configuración del proceso de optimización usado en XGBoost.

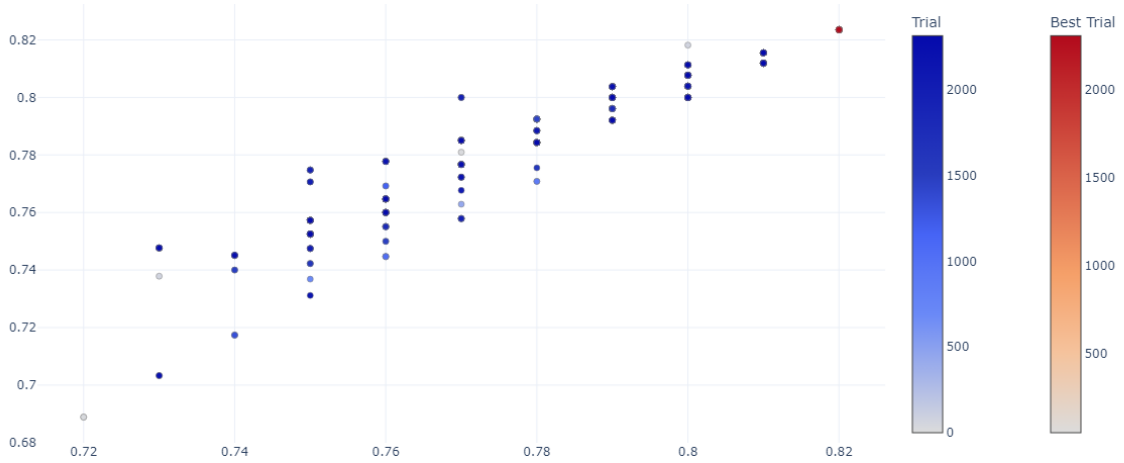
Tabla 23. *Configuración de parámetros del proceso de Optuna para Random Forest*

Parámetros	Valores	Descripción
Método de muestreo	TPESampler	Utiliza el algoritmo TPE (Tree-structured Parzen Estimator) para optimizar de manera eficiente los hiperparámetros, explorando y explotando el espacio de búsqueda basado en distribuciones de probabilidad.
Objetivo de la función	['maximize', 'maximize']	Objetivo que alcanzar en la función Optuna. Se ha parametrizado como métricas a maximizar f1_score_validation y accuracy_validation (conjunto de datos de validación)
Iteraciones	2317	Número de iteraciones que realizará Optuna en la función de optimización

La Figura 37, muestra la distribución de valores de cada objetivo obtenido por cada iteración del proceso de optimización del modelo Random Forest realizado en Optuna.

El resultado de este proceso de optimización ha arrojado que la iteración 1319 ha encontrado la configuración de los hiperparámetros que da un valor más alto para ambas las métricas f1-score y accuracy del conjunto de datos de validación, los cuales corresponden con 0.8235294117647058 y 0.82 respectivamente. La Tabla 24 muestra los valores obtenidos para los hiperparámetros del modelo Random Forest.

Figura 37. *Distribución de valores obtenidos en cada métrica objetivo en la optimización de Random Forest*



En el eje X se muestran los valores de la métrica accuracy_validation y en el eje Y se muestran los valores obtenidos para f1_score_validation. Nótese cómo a medida que Optuna avanza en las iteraciones se obtienen valores más altos para las métricas objetivo.

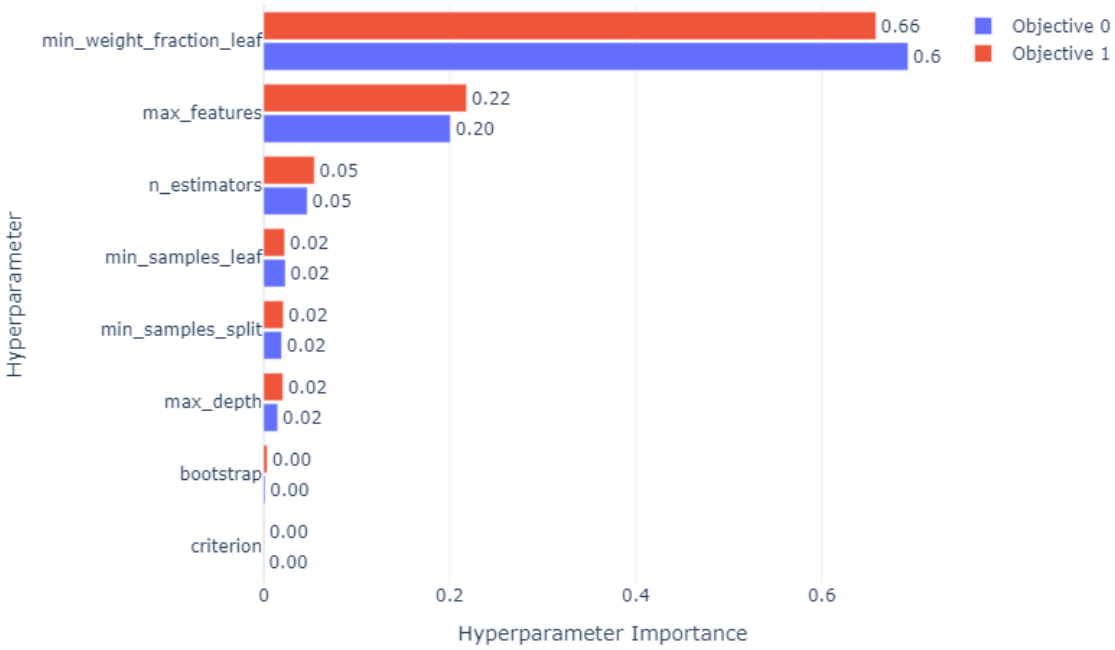
Fuente: Elaboración propia mediante Optuna Dashboard

Tabla 24. *Valores obtenidos para los hiperparámetros de RF en el proceso de optimización*

Hiperparámetros del modelo (nombre en librería)	Valores
max_depth	53
n_estimators	208
min_samples_split	33
min_samples_leaf	33
min_weight_fraction_leaf	0.34280999226261916
bootstrap	True
max_samples	0.767093742447935
max_features	None
criterion	gini

La Figura 38 muestra la contribución de cada hiperparámetro al modelo Random Forest donde se puede apreciar cómo cada hiperparámetro ha aportado al cálculo de cada una de las métricas objetivo, es decir, f1_score_validation y accuracy_validation.

Figura 38. *Contribución de cada hiperparámetro al modelo Random Forest*



Objective 0 corresponde a la métrica f1_score_validation y Objective 1 corresponde a la métrica accuracy_validation

Fuente: Elaboración propia mediante Optuna Dashboard

Las tablas 25 y 26 muestran los resultados obtenidos en entrenamiento y validación para el modelo Random Forest a partir de los hiperparámetros optimizados mediante Optuna.

Tabla 25. *Resultados del modelo Random Forest para el conjunto de entrenamiento con hiperparámetros optimizados*

	precision	**recall**	**F1-score**	**support**
No Tránsito	0.80	0.66	0.73	146
Tránsito	0.73	0.84	0.78	154
accuracy			0.76	300
macro avg	0.76	0.75	0.75	300
weighted avg	0.76	0.76	0.75	300

Tabla 26. *Resultados del modelo Random Forest para el conjunto de validación con hiperparámetros optimizados*

	precision	**recall**	**F1-score**	**support**
No Tránsito	0.83	0.80	0.82	50
Tránsito	0.81	0.84	0.82	50
accuracy			0.82	100
macro avg	0.82	0.82	0.82	100
weighted avg	0.82	0.82	0.82	100

A partir de estos resultados, se observa que el balance de clases, el proceso de Prewhitening, el FPCA y la optimización de hiperparámetros mejora el accuracy global para el conjunto de datos de validación en comparación con los modelos con conjuntos de datos sin preprocesar, pasando de 0.77 a 0.81 en XGBoost y de 0.75 a 0.82 en Random Forest. La comparativa completa de todos los resultados se puede encontrar en la Tabla 34 en el capítulo 5.

4.6. Otros experimentos

Durante la ejecución de este proyecto no solamente nos hemos centrado en el objetivo principal de la elaboración y optimización de un modelo clasificador de curvas de luz con XGBoost y Random Forest, sino que hemos explorado otras diferentes alternativas y métodos de optimización. En esta sección se recoge el trabajo realizado, que, aunque no se considera en los resultados finales, puede servir de apoyo a futuras investigaciones basadas en el proyecto.

4.6.1. Experimentos con otros modelos clasificadores

Se ha explorado otros algoritmos clasificadores para buscar alternativas a los modelos XGBoost y Random Forest. En concreto se ha explorado los algoritmos AdaBoostClassifier, GradientBoostingClassifier y LightGBM. Dado que los resultados obtenidos en la métrica de F1-score no son mejores que los obtenidos con XGBoost y Random Forest, incluso se ha obtenido overfitting en el entrenamiento del modelo GradientBoostingClassifier, no se continuará trabajando con estos modelos. La tabla 27 muestra las principales métricas obtenidas con estos experimentos.

Tabla 27. *Resultados de experimentos para AdaBoostClassifier, GradientBoostingClassifier y LightGBM*

Clasificador	Método de muestreo de Optuna	F1-score validación (Tránsito)
AdaBoostClassifier	TPESampler	0.70
GradientBoostingClassifier	TPESampler	0.64
LightGBM	TPESampler	0.61

4.6.2. Experimentos con redes neuronales

Como se comentó en la sección 2, además de los modelos clasificadores existen los modelos de redes neuronales para poder realizar clasificaciones y procesamientos complejos. En este proyecto nos hemos centrado en los algoritmos clasificadores XGBoost y Random Forest, tanto para la generación de los modelos como para su optimización.

En presentación o introducción a futuras investigaciones que partan de la base de este proyecto, hemos incluido en esta sección un esbozo de una primera aproximación del proceso utilizado con los algoritmos clasificadores mencionados pero extrapolado a redes neuronales, mediante la plataforma TensorFlow (Abadi et al., 2016).

Se ha desarrollado un algoritmo en Python (Anexo A: código A6) en el cual se ha realizado una optimización mediante Optuna de los hiperparámetros requeridos por la red neuronal de TensorFlow. El alto volumen de información para cada una de las épocas de la red neuronal genera un proceso de cómputo de varios días, por lo que se considera un proceso inviable para el corto tiempo de investigación disponible durante la creación de este proyecto, pero sirve como base o punto de partida para otros proyectos futuros. Sólo se han podido recoger datos preliminares debido a las limitaciones de tiempo para un conjunto de datos limitado.

El modelo de redes neuronales utilizado es un tipo de redes neuronales recurrentes, llamado Long Short-Term Memory (LSTM). Los modelos LSTM son una variante de las redes neuronales recurrentes diseñadas para manejar dependencias a largo plazo en datos secuenciales, superando las limitaciones de las redes recurrentes tradicionales en la captura de patrones a largo plazo (Hochreiter y Schmidhuber, 1997). Esto es útil en nuestro proyecto para el manejo de curvas de luz, que son de por sí datos secuenciales de flujo y la secuencialidad es una característica importante.

Para poder utilizar las redes LSTM, es importante realizar una previa normalización de los datos. En nuestro caso hemos ajustado los datos para que tengan media cero y varianza unitaria, por lo que preveamos que se facilitará la convergencia de los algoritmos de optimización. (Ioffe y Szegedy, 2015).

Como en la mayoría de los modelos de redes se han usado capas dropout para prevenir el sobreajuste. El dropout es una técnica en la cual se desconectan neuronas durante el entrenamiento, lo que ayuda a mejorar la capacidad de generalización del modelo al reducir la dependencia de configuraciones específicas de las neuronas (Srivastava et al., 2014).

Para la optimización de hiperparámetros se ha utilizado la misma biblioteca que para los modelos clasificadores principales de este proyecto (Optuna). En este caso ajusta parámetros los parámetros propios de la red neuronal, como son el número de unidades en las capas LSTM, la tasa de dropout, la tasa de aprendizaje del optimizador y el tamaño del lote de entrenamiento (Bergstra et al., 2011).

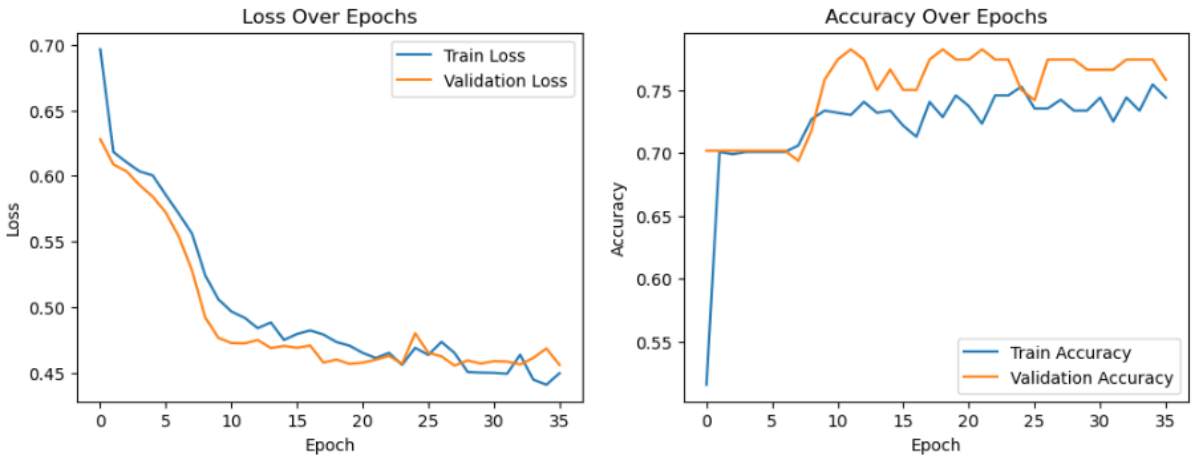
Para el entrenamiento del modelo, se ha empleado el optimizador Adam, esta técnica de optimización está basada en gradientes adaptativos que ajusta las tasas de aprendizaje individuales para cada parámetro. (Kingma y Ba, 2014).

Por último, se implementa la técnica de parada temprana para prevenir el sobreentrenamiento (Prechelt, 1998). Esto asegura que el modelo mantenga una buena capacidad de generalización sin ajustarse demasiado a los datos de entrenamiento.

Se han dividido de forma aleatoria el conjunto de datos (el mismo que se ha usado en los otros modelos), en 70% para entrenamiento, 15% para validación y 15% para prueba.

Los resultados de exactitud del conjunto de validación con este modelo de redes neuronales son los siguientes, tal como se muestra en la Figura 39: la pérdida de entrenamiento (Train Loss) es de 0.429, indicando el error promedio del modelo durante el entrenamiento. La precisión de entrenamiento es del 75.95%. Por otro lado, la pérdida de validación (Validation Loss) es de 0.456, muy similar a la perdida durante el entrenamiento; como ambas curvas decrecen y convergen para el aumento de las épocas (Epoch), se entiende que el modelo tiene buen ajuste y no tiene sobredimensionamiento (overfitting).

Figura 39. *Resultados de exactitud del conjunto de validación para un modelo de redes neuronales*



Izquierda: valor de la perdida por época para entrenamiento y validación; derecha: valor de la precisión para entrenamiento y validación por época.

Fuente: elaboración propia

La precisión de validación (Validation Accuracy) es del 75.81%, también muy similar a la precisión de entrenamiento (Train Accuracy), ambas tienden a converger en un valor y aunque la de validación presenta unos valores mayores la tendencia indica que el modelo no parece sobreajustado.

Estos resultados, si bien son inferiores a los modelos obtenidos mediante los clasificadores XGBoost y Random Forest en el capítulo anterior, son indicativos de un buen punto de partida para que este método alternativo mediante redes neuronales sea explorado en futuros proyectos.

4.6.3. Generación de curvas de luz simuladas

Uno de los problemas de los procesos de generación de modelos realizados en este proyecto es que el conjunto de datos de entrenamiento en cuanto a instancias positivas está limitado al número de exoplanetas confirmados, por lo que ni los posibles candidatos pueden usarse para alimentar el modelo si no han confirmado sus exoplanetas por otros métodos. Esta limitación nos plantea la generación de curvas de luz con tránsitos simulados como una de las posibles alternativas para aumentar el conjunto de datos de entrenamiento.

Es importante destacar que ya existen algoritmos sofisticados como el TLCM (Csizmadia et al. 2020), que permiten simular tránsitos de manera precisa. Estos algoritmos incorporan efectos de binning, beaming, reflexión y efectos elipsoidales, entre otros. Sin embargo, para nuestro proyecto, como solamente vamos a realizar un ejercicio ilustrativo del proceso de generación de curvas de luz simuladas, no creemos necesario tanta complejidad a la hora de generar una muestra de curvas de luz para entrenar el modelo y por lo tanto hemos optado por una formulación simplificada para el tránsito, considerando una función gaussiana.

En primer lugar, obtenemos los valores significativos de las curvas de luz preprocesadas, mediante un algoritmo con el que obtenemos la mediana de los valores de flujo, así como la mediana de los valores máximos y mínimos de flujo y de las distintas longitudes de las curvas de luz. Se utiliza la mediana ya que ésta es más robusta frente a desviaciones que el promedio de los valores. Este proceso, conocido como filtro de mediana, se ha utilizado en otros trabajos como tratamiento de información de imágenes para obtener valores significativos promedio sin efectos de desviaciones puntuales (Argiles et al., 2011).

Realizando los filtros de mediana para los valores mencionados, se busca la información de número de tránsitos, anchuras y profundidades promedio de la misión Kepler, para alimentar al algoritmo generador de curvas de luz simuladas con valores coherentes. A modo resumen se obtienen los valores de la Tabla 28.

Tabla 28. *Valores promedio y medianas de curvas de luz simuladas*

Elemento estudiado	Mediana	Promedio
Valores de flujo	-6	-644
Valores máximos de flujo	707	707
Valores mínimos de flujo	-715	-715
Longitud curva de luz	55581	64793

A partir de estos valores, se realiza un algoritmo de generación curvas de luz sintéticas, (Anexo A: código A7) Este algoritmo genera un número aleatorio de tránsitos para cada curva de luz simulada con una profundidad de tránsito y achura del mismo generada de manera aleatoria entre los valores obtenidos en la tabla 26 con una señal de ruido (SNR) aleatoria de un 1% del valor total de flujo.

Los tránsitos se generan mediante la siguiente fórmula:

$$F_i = D_i \cdot e^{\left(\frac{1(t_i - t_c)^2}{2\ (W_i)^2}\right)} \quad (10)$$

donde para cada iteración en un punto de tiempo de la curva de luz (t_i)

- F_i = es el punto de flujo generado
- D_i = Valor de profundidad de tránsito aleatoria
- W_i = Valor de anchura de tránsito aleatoria
- t_c = Valor del punto central del tránsito de cada tránsito

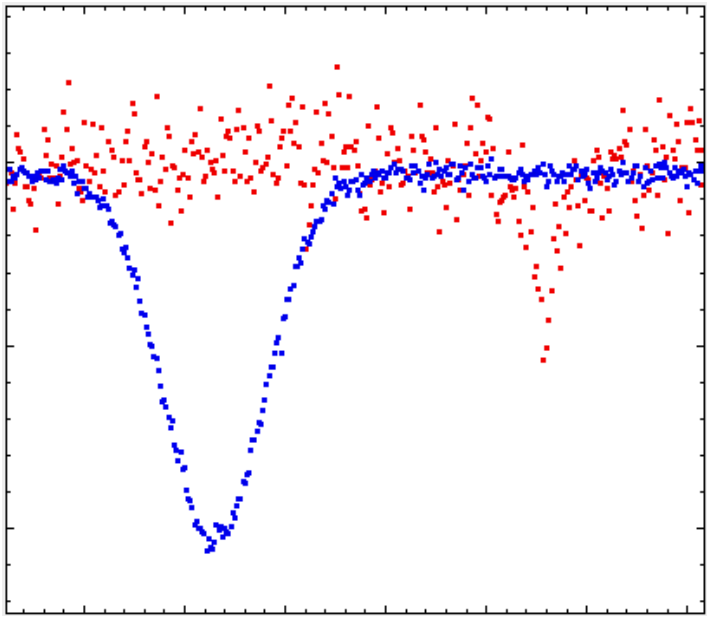
Estos parámetros pueden ser modificados en futuras iteraciones del algoritmo para ver cómo afectan a los algoritmos clasificadores. La Figura 40 muestra un ejemplo de curva de luz generadas artificialmente para alimentar el modelo. Con este algoritmo, se crean un número similar de curvas de luz artificiales con y sin tránsito que se añaden al conjunto de curvas de luz de preparación del modelo. Para el entrenamiento, se han dividido de forma aleatoria el conjunto de datos (incluyendo las curvas originales que hemos usado en los otros modelos y las artificiales) en 70% para entrenamiento, 15% para validación y 15% para prueba, según se muestra en la Tabla 29.

Tabla 29. *Distribución de los conjuntos de Entrenamiento, Validación y Prueba con curvas simuladas*

Conjunto	Total
Entrenamiento	664
Validación	84
Prueba	84

Con este nuevo conjunto de datos se repite el proceso del apartado 5.5 con la misma muestra obteniéndose los resultados recogidos en las tablas 30, 31, 32 y 33 al final de esta sección.

Figura 40. *Ejemplo de curva de luz simulada*



Se muestra en roja un detalle del tránsito curva de luz preprocesada original con tránsito y en azul una curva de luz generada.

Fuente: Elaboración propia mediante TOPCAT

Para futuras investigaciones se puede mejorar el modelo de generación de curvas aleatorias de tránsito para evitar añadirle más variabilidad y dificultad de tal manera que el modelo tenga más datos y se evite que se aprenda los patrones específicos.

Como se ha comentado al principio de esta sección, el proceso realizado en el proyecto de generación de curvas de luz simuladas es meramente informativo de un posible proceso de mejora del conjunto de datos de entrenamiento disponibles, es por ello por lo que no se han realizado pruebas adicionales con balance de clases u optimización de hiperparámetros con estas curvas de luz simuladas y se deja la puerta abierta para futuras investigaciones.

4.6.3.1. Modelo XGBoost

Tabla 30. *Resultados del modelo XGBoost para el conjunto de entrenamiento con curvas simuladas*

	precision	recall	F1-score	support
0.0	0,99	1,00	0,99	330
1.0	0,99	0,98	0,99	155
accuracy			0,99	485
macro avg	0,99	0,99	0,99	485
weighted avg	0,99	0,99	0,99	485

Tabla 31. *Resultados del modelo XGBoost para el conjunto de validación con curvas simuladas*

	precision	recall	F1-score	support
0.0	0,70	0,77	0,73	77
1.0	0,53	0,44	0,48	45
accuracy			0,65	122
macro avg	0,61	0,61	0,61	122
weighted avg	0,64	0,65	0,64	122

4.6.3.2. Modelo Random Forest

Tabla 32. *Resultados del modelo Random Forest para el conjunto de entrenamiento con curvas simuladas*

	precision	recall	F1-score	support
0.0	0,98	1,00	0,99	330
1.0	0,99	0,95	0,97	155
accuracy			0,98	485
macro avg	0,98	0,97	0,98	485
weighted avg	0,98	0,98	0,98	485

Tabla 33. *Resultados del modelo Random Forest para el conjunto de validación con curvas simuladas*

	precision	**recall**	**F1-score**	**support**
0.0	0,71	0,82	0,76	77
1.0	0,58	0,42	0,49	45
accuracy			0,67	122
macro avg	0,64	0,62	0,62	122
weighted avg	0,66	0,67	0,66	122

A partir de los resultados obtenidos para los modelos XGBoost y Random Forest con curvas simuladas, se observa que los valores de las métricas del conjunto de validación en ambos modelos son significativamente más bajos que los valores de las métricas para el conjunto de entrenamiento. Esto indica que los modelos están sobreajustados y que se han ajustado excesivamente a las particularidades del conjunto de entrenamiento, por lo tanto, no generalizan ni se adaptan a nuevos datos.

5. Resultados y análisis

5.1. Resultados generales del proceso

El objetivo principal de nuestro proyecto era conseguir desarrollar un modelo de machine learning capaz de utilizar datos de curvas de luz de estrellas intrínsicamente variables para encontrar fenómenos de tránsitos de exoplanetas. Se ha mostrado durante la sección anterior el proceso de creación del conjunto de datos para entrenar ambos modelos clasificadores utilizando XGBoost y Random Forest, así como los preprocesamientos y optimizaciones necesarios para mejorar su eficiencia.

La Tabla 34 muestra un resumen de los valores obtenidos en validación para el conjunto de datos de Tránsito (clase 1.0). Se observa que a medida que se ha avanzado en el preprocesamiento de los datos y en la optimización de los modelos se ha logrado mejorar los resultados de los modelos XBGOOST y Random Forest con respecto a ejecuciones anteriores.

Tabla 34. *Resumen de resultados de los modelos XBGOOST y Random Forest para el conjunto de datos de validación*

	XGBoost			**Random Forest**		
	Accuracy	**F1-score Tránsito**	**Recall Tránsito**	**Accuracy**	**F1-score Tránsito**	**Recall Tránsito**
Curvas de luz sin procesar	0.67	0.39	0.45	0.67	0.39	0.45
Curvas procesadas	0.77	0.68	0.76	0.75	0.67	0.78
Curvas procesadas con hiperparámetros optimizados	0.81	0.83	0.90	0.82	0.82	0.84

Se observa que existe una mejora gradual en ambos modelos conforme se realizan más pasos de preprocesamiento de los datos y de optimización, llegando a unos valores que podemos considerar aceptables y útiles para los datos disponibles de curvas de luz y la complejidad del

trabajo, Se confirma, que además del preprocesamiento de los datos y la optimización de hiperparámetros se ha logrado mejorar el resultado obtenido en experimentos anteriores.

Adicionalmente, consideramos que los modelos desarrollados pueden facilitar el trabajo de otros algoritmos o modelos más detallados para la búsqueda de exoplanetas.

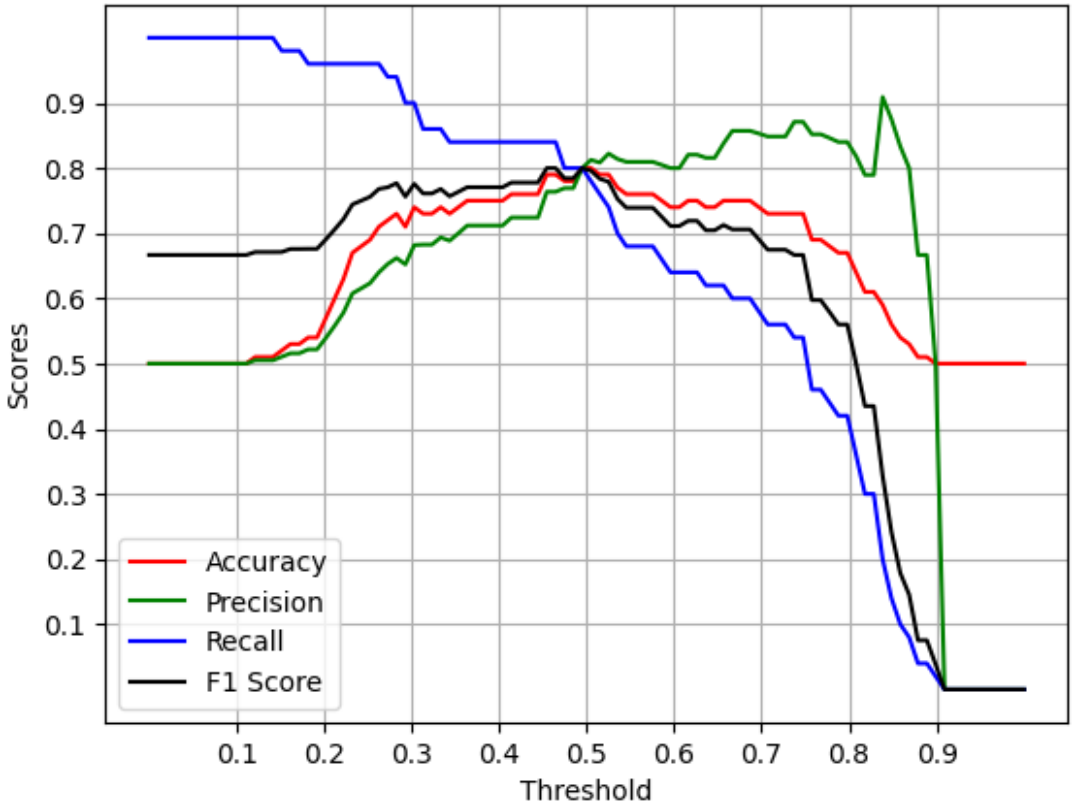
5.2.Selección del mejor modelo

Una vez analizados los resultados de los modelos desarrollados para el conjunto de datos de validación, hemos decidido seleccionar los modelos con hiperparámetros optimizados como los mejores modelos desarrollados a partir de los datos disponibles de la misión Kepler. Por esta razón se ha decidido analizar los modelos XGBoost y Random Forest desarrollados en dicha fase con el conjunto de datos de prueba.

5.2.1. Mejor modelo XGBoost con el conjunto de datos de prueba

Inicialmente, hemos analizado la relación de las curvas de accuracy, precision, recall y F1-score del conjunto de validación en función de los hiperparámetros optimizados del modelo (Figura 41). Se observa que los valores más altos de f1-score y accuracy se encuentran entre un threshold (probabilidad de que una curva sea clasificada como Tránsito) de 0.49 a 0.51. Por lo anterior, consideramos que el modelo está bien calibrado y que produce probabilidades predichas que están bien alineadas con las probabilidades reales observadas en el conjunto de datos. Adicionalmente, este resultado confirma que la distribución de las clases en el conjunto de datos es bastante equilibrada.

Figura 41. *Curvas de métricas de validación del mejor modelo XGBoost*



Fuente: Elaboración propia

La Tabla 35 muestra los resultados obtenidos para el conjunto de datos de prueba del mejor modelo XGBoost. Se aprecia que las métricas de interés para esta investigación (F1-score Tránsitos y accuracy) han obtenido el valor 0.79 y 0.80 respectivamente.

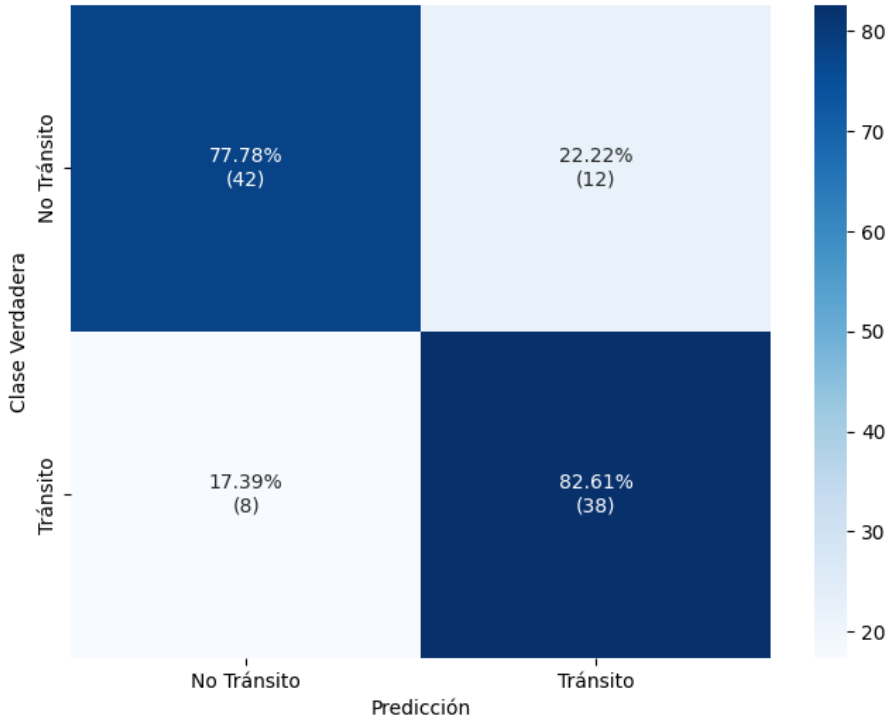
Tabla 35. *Resultados del mejor modelo XGBoost para el conjunto de prueba con hiperparámetros optimizados*

	precision	**recall**	**F1-score**	**support**
No Tránsito	0.84	0.78	0.81	54
Tránsito	0.76	0.83	0.79	46
accuracy			0.80	100
macro avg	0.80	0.80	0.80	100
weighted avg	0.80	0.80	0.80	100

La Figura 42 muestra la matriz de confusión del conjunto de prueba del mejor modelo XGBoost, luego de optimizar los hiperparámetros del modelo. Se puede apreciar que:

- el modelo predijo correctamente los No Tránsitos (clase 0) el 77.78% de las veces cuando la clase verdadera era 0 (Verdaderos Negativos).
- el modelo predijo incorrectamente los Tránsitos (clase 1) el 22.22% de las veces cuando la clase verdadera era 0 (Falsos Positivos)
- el modelo predijo incorrectamente los No Tránsitos (clase 0) el 17.39% de las veces cuando la clase verdadera era 1 (Falsos Negativos)
- el modelo predijo correctamente los Tránsitos (clase 1) el 82.61% de las veces cuando la clase verdadera era 1 (Verdaderos Positivos)

Figura 42. *Matriz de confusión del conjunto de prueba del mejor modelo XGBoost*



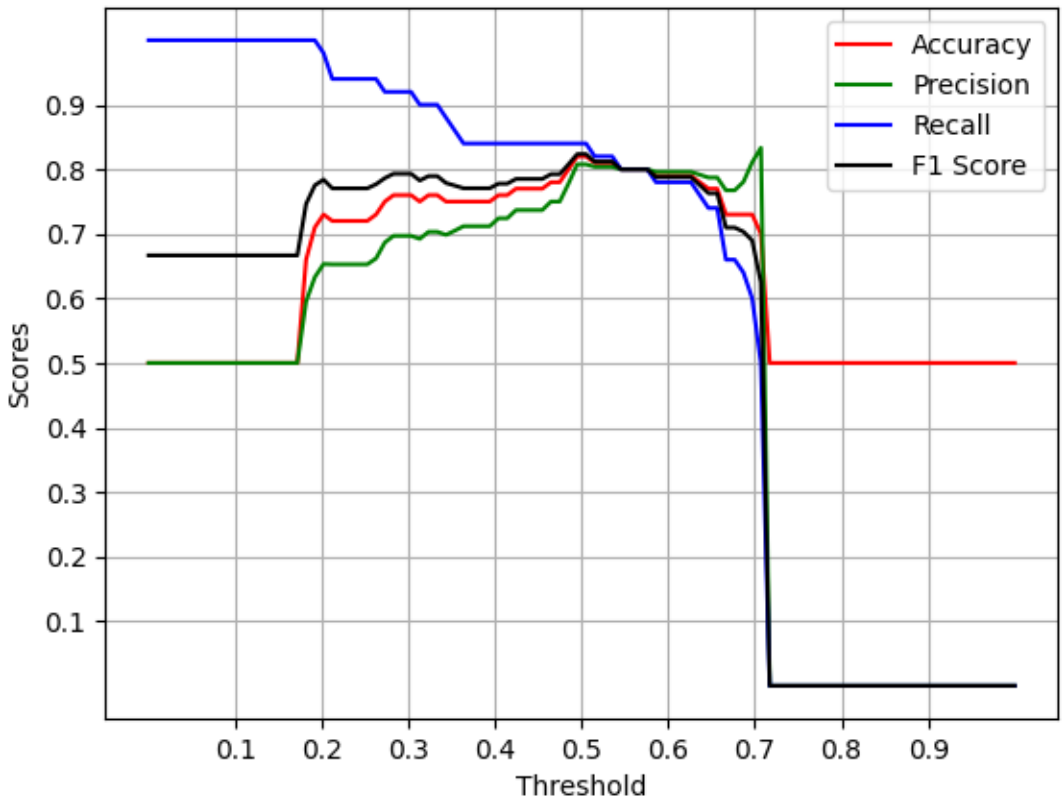
Fuente: Elaboración propia

En el Anexo A: código A8, se muestra el algoritmo desarrollado para mostrar los resultados del conjunto de prueba del mejor modelo de XGBoost.

5.2.2. Mejor modelo Random Forest con el conjunto de datos de prueba

Al igual que con el mejor modelo XGBoost, hemos analizado la relación de las curvas de accuracy, precision, recall y F1-score del conjunto de validación en función de los hiperparámetros optimizados del modelo (Figura 43). Se confirma que al igual que XGBoost los valores más altos de f1-score y accuracy se encuentran entre un threshold de 0.49 a 0.51, por lo que consideramos que el mejor modelo de Random Forest también está bien calibrado y que la distribución de las clases en el conjunto de datos es bastante equilibrada.

Figura 43. *Curvas de métricas de validación del mejor modelo Random Forest*



Fuente: Elaboración propia

La Tabla 36 muestra los resultados obtenidos para el conjunto de datos de prueba con el mejor modelo Random Forest. Las métricas de interés para esta investigación (F1-score Tránsitos y accuracy) han obtenido el valor 0.78 y 0.80 respectivamente.

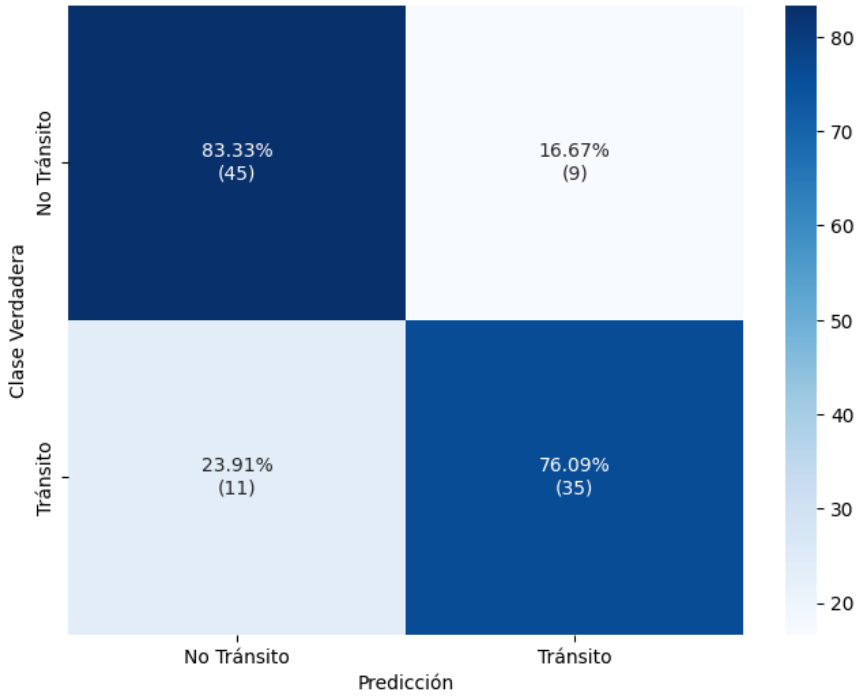
Tabla 36. *Resultados del mejor modelo Random Forest para el conjunto de prueba con hiperparámetros optimizados*

	precision	**recall**	**F1-score**	**support**
No Tránsito	0.80	0.83	0.82	54
Tránsito	0.80	0.76	0.78	46
accuracy			0.80	100
macro avg	0.80	0.80	0.80	100
weighted avg	0.80	0.80	0.80	100

La figura 44 muestra la matriz de confusión del conjunto de prueba del mejor modelo Random Forest, luego de optimizar los hiperparámetros del modelo. Se puede apreciar que:

- el modelo predijo correctamente los No Tránsitos (clase 0) el 83.33% de las veces cuando la clase verdadera era 0 (Verdaderos Negativos).
- el modelo predijo incorrectamente los Tránsitos (clase 1) el 16.67% de las veces cuando la clase verdadera era 0 (Falsos Positivos)
- el modelo predijo incorrectamente los No Tránsitos (clase 0) el 23.91% de las veces cuando la clase verdadera era 1 (Falsos Negativos)
- el modelo predijo correctamente los Tránsitos (clase 1) el 76.09% de las veces cuando la clase verdadera era 1 (Verdaderos Positivos)

Figura 44. *Matriz de confusión del conjunto de prueba del mejor modelo Random Forest*



Fuente: Elaboración propia

En el Anexo A: código A9, se muestra el algoritmo desarrollado para mostrar los resultados del conjunto de prueba del mejor modelo de RandomForest.

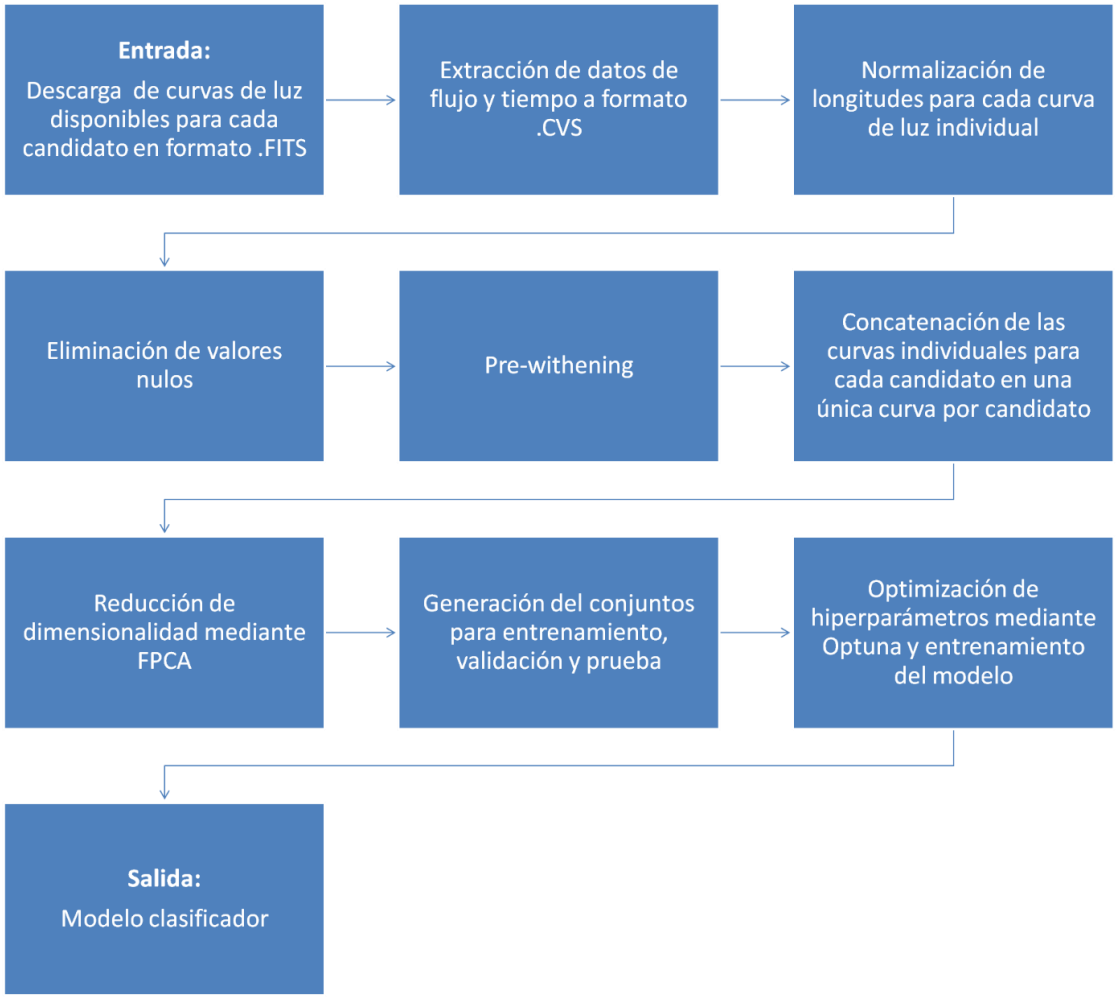
Estos dos modelos son los finalmente seleccionados como candidatos a la ejecución con datos reales, como se mencionó al comienzo de esta sección, de la misión Kepler, para evaluar la eficiencia de estos y así poder comprobar el cumplimiento del objetivo principal del proyecto, que era generar un algoritmo clasificador que pueda ser extrapolado a conjuntos de datos sin reales y con una exactitud lo más alta posible.

En añadido, se han explorado durante la realización de este proyecto alternativas a otros métodos clasificadores o incluso el uso de redes neuronales a partir de los datos en bruto.

Como resumen del resultado obtenido en este proyecto en base al objetivo principal, de desarrollar un modelo que, a partir de una entrada de curvas de luz de estrellas variables sin procesar, generase un modelo clasificador de tránsitos, se presenta en el diagrama de flujo

mostrado en la Figura 45, el flujo de trabajo realizado y las distintas etapas mencionadas de recopilación, preprocesamiento y optimización de datos.

Figura 45. *Pipeline del proyecto*



Fuente: elaboración propia

5.3. Ejecución en datos adicionales

En la sección anterior hemos obtenido dos modelos clasificadores (Random Forest y XGBoost) entrenados para determinar curvas de luz concatenadas susceptibles de tener tránsito. En esta primera sección de resultados vamos a probar estos modelos obtenidos con una muestra de 1.955 curvas de luz disponibles de la misión Kepler que no están definidas como KOI pero que están definidas como estrellas variables en Gaia DR3, como se definió al comienzo de la sección 4.

Como se comentó en el comienzo de la sección 4, el objetivo de la ejecución del algoritmo en este conjunto de datos no es en primera instancia evaluar dichos datos en búsqueda de exoplanetas, sino evaluar el comportamiento del modelo utilizado.

Para ejecutar ambos modelos, en primer lugar, procedemos a realizar el proceso de *prewhitening* para que los modelos trabajen con las curvas de luz en el mismo formato.

La muestra objetivo, es reducida de 1.792 curvas de luz frente a las 1.955 disponibles, esto es debido a que eliminamos todas aquellas que tienen valores NaN o nulos y nos pueden generar error. Estas 1.792 se procesan mediante la reducción de dimensionalidad y se añaden los parámetros adicionales de la misma forma que fueron entrenados los modelos clasificadores.

Los resultados obtenidos se recogen en la Tabla 37, se observa que ambos modelos encuentran un conjunto limitado de positivos, mucho menor al que podríamos esperar de sus matrices de confusión. Estos resultados tienen cierta lógica ya que, como se explicó en la sección 4, estos 1.792 candidatos ya han sido evaluados y descartados por la misión Kepler para la determinación de los KOI y se estima poco probable la existencia de exoplanetas, por lo que tiene sentido esperar tal como se ha obtenido, un número bajo de positivos.

Tabla 37. *Predicción de los modelos XBGOOST y Random Forest con nuevos datos*

	XGBoost			Random Forest		
	Tránsito	**No Tránsito**	**Total**	**Tránsito**	**No Tránsito**	**Total**
Resultados de la muestra	200 (11,16%)	1592 (88,84%)	1792	271 (15,12%)	1521 (84,88%)	1792

Se ha comprobado mediante una comparativa de identificadores de los candidatos, que 189 de los 200 positivos clasificados por XGBoost se encuentran en los 271 positivos obtenidos mediante Random Forest.

Para ilustrar el potencial de este proyecto, se ha analizado aleatoriamente uno de los candidatos clasificados como positivo en los resultados de ambos clasificadores (ID Kepler 009528112). Este enfoque se ha utilizado para destacar la robustez de nuestro proyecto, ya que desde el principio se integran datos cruzados con Gaia, para realizar búsquedas adicionales de información de los candidatos y tránsitos una vez ejecutados los modelos mediante otros métodos o información adicional.

La idea de este ejemplo ilustrativo con KIC 009528112 es mostrar que el proceso de generación del modelo y ejecución es un proyecto completo y funcional, capaz de iniciar con datos preliminares y terminar con el proceso evaluación de candidatos.

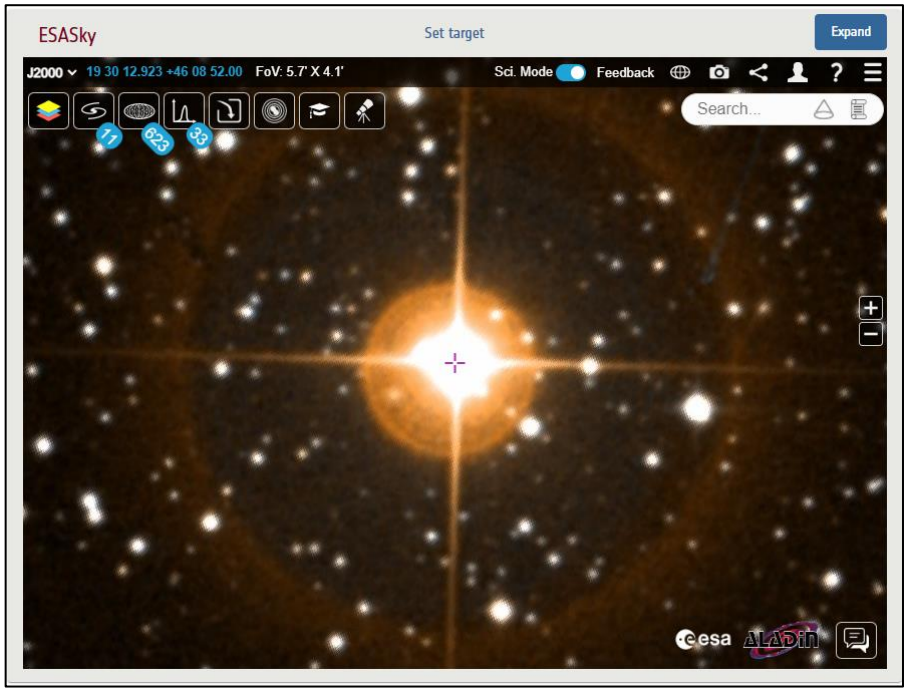
El candidato seleccionado aleatoriamente del conjunto de positivos como se mencionó es KIC 009528112 que ha sido localizado en Gaia mediante la trazabilidad de nuestro proyecto como Gaia DR3 2128253262651259904 (AF Cyg - HD 184008). El primer paso es realizar una investigación exhaustiva de la información y artículos disponibles del candidato en cuestión, antes de realizar cualquier tipo de análisis.

En el caso de ejemplo, esta estrella AF Cyg (HD 184008) es bastante conocida por ser una gigante roja variable de largo periodo, los datos extraídos de Simbad (figura 46) y de Gaia DR3 se recogen en la Tabla 38.

Tabla 38. *Características principales de KIC 009528112*

Coordenadas (J2000)	19 30 12,85 +46 08 52,08
Velocidad radial	-12.02 km/s
Paralaje	3.7384 mas
Tipo espectral	M5III
Temperatura efectiva aproximada	3000 K

Figura 46. *Imagen de KIC 009528112 / AF Cyg / HD 184008*

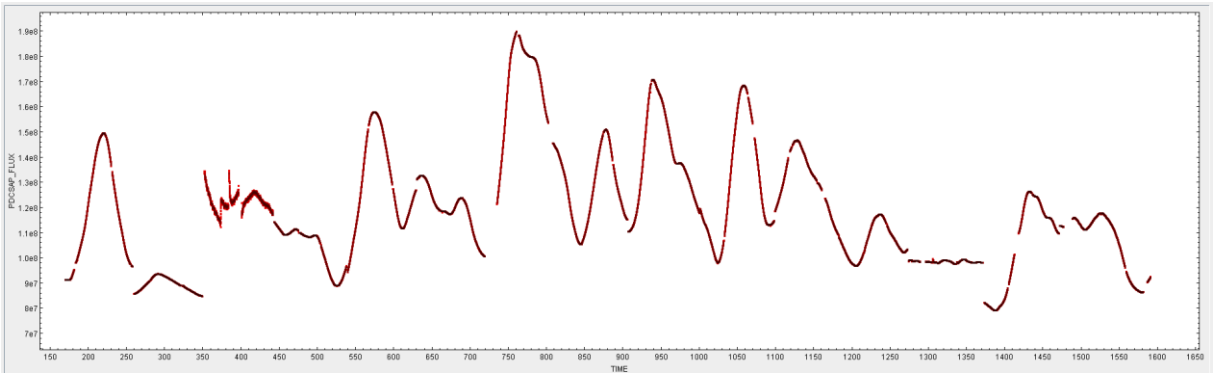


Fuente: https://gea.esac.esa.int/archive/

La estrella AF Cyg, (HD 184008, KIC 9528112), es una estrella compleja como resultado positivo del modelo de búsqueda de tránsitos, ya que se trata de una gigante roja con una variabilidad larga y complicada, como muestra su curva de luz (Figura 47). Según Hinkle et al.

(2002), AF Cyg muestra períodos de variabilidad de 93 y 163 días, con un período adicional más largo de 927.5 días que ha sido observado en sus datos de velocidad radial.

Figura 47. *Curva de luz de KIC 9528112*



Fuente: Elaboración propia

Este comportamiento de múltiples períodos de variabilidad sugiere la presencia de pulsaciones complejas que, según el artículo de Hinkle, pueden ser hasta posibles interacciones con un compañero binario, lo que puede haber sido uno de los motivos por los que el modelo ha considerado un candidato positivo.

El estudio de Hinkle es muy interesante, pues algunos de sus modelos propuestos comentan un compañero invisible, que podría causar eclipses que coinciden con las mínimas de luz observadas. Este objeto posiblemente sea una enana marrón o un planeta masivo.

Tras la investigación existen diferentes procesos posteriores para evaluar y buscar tránsitos. Esto queda fuera del alcance de nuestro proyecto, pero se podrían realizar en futuras investigaciones o con otros resultados positivos obtenidos mediante nuestro modelo clasificador. Para ello se podría utilizar la biblioteca Lightkurve (Cardoso et al 2018). Esta biblioteca está diseñada para analizar datos de curvas de luz astronómicas, e incluye funciones para detectar exoplanetas y clasificarlos, y otros fenómenos astronómicos.

6. Conclusiones y perspectivas futuras

La inteligencia artificial (IA) ha demostrado en los últimos años ser una herramienta muy efectiva para tratar grandes volúmenes de datos y es especialmente útil para distintos campos en la astrofísica y la astronomía.

En el campo concreto de la búsqueda de exoplanetas, además de una cantidad elevada de información debido a los múltiples y en constante aumento de objetos analizados continuamente por las distintas misiones, surge la dificultad adicional relacionada con una cantidad de información muy elevada en lo referido a cada curva de luz, que como hemos visto durante este proyecto, una sola curva de luz de una estrella puede llegar a tener más de 60.000 puntos de información.

En el contexto de este proyecto, hemos querido además abordar un caso particular de la información disponible de las misiones de búsqueda de exoplanetas, específicamente en sistemas con estrellas con variabilidad intrínseca, cuya curva de luz además de ser más compleja y dificultar la identificación de los posibles eventos de tránsito presentes en su flujo, puede inducir al modelo clasificador a cometer más errores.

Se presentan en este capítulo las conclusiones del proyecto, en base al proceso y los resultados obtenidos, se mencionan también las dificultades encontradas y los puntos de partida para proyectos futuros.

6.1. Conclusiones generales

Como se mencionó en la sección 3, nuestros principales objetivos en este proyecto se centraban en la generación de un modelo que pudiera trabajar con los datos de las curvas de luz y generar unas predicciones con un valor para las métricas F1-score y accuracy del conjunto de datos de validación lo más alto posible, de tal manera que la precisión en la clasificación de tránsitos permitiera optimizar el trabajo posterior de evaluación de las curvas de luz mediante otros métodos.

El objetivo de la generación por tanto parte de la base obtenida a partir de la ejecución de los algoritmos clasificadores sin procesamiento de los datos (sección 4.2). Las métricas F1-score y accuracy del conjunto de datos de validación de los modelos base son el punto de partida del contenido de este proyecto, que se ha enfocado en analizar los pasos necesarios y generar el pipeline descrito en los resultados de procesamiento de las curvas de luz y generación del modelo, de tal manera que los valores de estas métricas han subido 44% en F1-score y 14% en accuracy para XGBoost y 43% en F1-score y 15% en accuracy para Random Forest en los mejores modelos frente a los valores obtenidos en la primera generación base.

Si bien estas mejoras en los modelos son significativas, estos resultados abren la puerta a nuevas mejoras para conseguir unos modelos aún más eficaces que reduzcan el número de falsos positivos al mínimo posible, con el objetivo final de poder hacer filtrados a gran escala de múltiples curvas de luz con un proceso automatizado, dejando para el análisis final un grupo reducido de las mismas.

6.2. Dificultades encontradas

Durante la ejecución y desarrollo del proyecto, hemos encontrado diversas dificultades para la generación del modelo y la posterior ejecución de este.

La primera dificultad y posiblemente la más importante de cara al contexto del proyecto, es la cantidad de datos disponibles para entrenar el modelo de forma supervisada. Partiendo de la base de que necesitábamos curvas de luz clasificadas con tránsito positivo en estrellas intrínsecamente variables para entrenar al modelo (es decir, aquellas curvas de luz que entrenan el modelo como valor 1), estábamos limitados a los exoplanetas descubiertos en sistemas de este tipo a fecha de este proyecto. Por otro lado, y para no contaminar el modelo con curvas de luz incógnitas, también teníamos la limitante del número de curvas de luz clasificadas como tránsito negativo (es decir, aquellas curvas de luz que entrenan al modelo como valor 0). Esto limita el número de elementos disponibles a un valor, que, en vista de los resultados obtenidos, consideramos que es insuficiente para conseguir una exactitud y precisión óptima.

En base a esta dificultad, se han planteado en este proyecto alternativas adicionales como la segmentación de los datos o la generación de curvas simuladas, para aumentar el volumen de datos disponibles. Estas alternativas se sugieren para trabajos futuros con base en este proyecto.

Otra dificultad añadida es el desequilibrio entre la clase negativa y la clase positiva, con una relación aproximada de 2:1, lo que nos ha obligado a realizar un balance de clases, dejando de lado un volumen considerable de datos de la clase negativa que se disponía en el inicio del proyecto. Este balance ha mejorado los resultados obtenidos en el conjunto de validación de los modelos clasificadores realizados.

Por último, el gran volumen de información contenido en cada curva de luz, con más de 60000 valores de flujo frente al tiempo, incrementa la complejidad computacional para los distintos algoritmos de preprocesamiento y optimización previos a la generación del modelo, obligándonos a limitar el número de iteraciones de optimización y de valores en las curvas de luz a grupos más pequeños, por lo que existe el riesgo de perder información importante.

Todas estas dificultades encontradas han afectado a la exactitud final y al F1 score de positivos que se ha conseguido obtener en los modelos. Si bien hemos conseguido elevar los valores de exactitud gracias a los métodos de optimización y preprocesamiento utilizados, no se ha conseguido llegar a un mínimo de 90% de exactitud o de F1-score para los modelos, que hubieran sido los valores deseados para estos parámetros.

6.3. Perspectivas futuras

Como se ha observado durante el proyecto y en base a los resultados obtenidos, el algoritmo de clasificación tiene un amplio margen de optimización en el futuro.

Se consideraron diversas acciones en este proyecto como el uso de curvas de luz con tránsitos simulados para aumentar el volumen de la muestra disponible para entrenar el modelo. En un futuro, el aumento de la cantidad de información disponible de las continuas misiones que se dedican a la toma de datos y la detección de exoplanetas harán que el conjunto de datos

positivos y negativos para alimentar el modelo sea mayor, obteniendo mejores posibilidades de resultados óptimos. El proceso de alimentación de dichos modelos generará a su vez una búsqueda más eficiente de planetas mediante el método de tránsito de forma automatizada, como se ha comenzado a plantear en este proyecto.

Dado que una de las dificultades encontradas ha sido la complejidad computacional de algunos algoritmos que nos ha obligado a realizar restricciones en iteraciones y datos a analizar, se plantea como mejora futura el uso de sistemas computacionales más potentes.

Uno de los posibles trabajos futuros que se visualizan para explorar mejoras en los modelos es el uso de la librería de Python TSFRESH (Time Series FeatuRe Extraction on basis of Scalable Hypothesis tests) para extraer automáticamente características estadísticas y de dominio de la frecuencia de las series temporales de las curvas de luz. Esta estrategia reemplazaría en parte el preprocesamiento realizado con FPCA y generación de características adicionales descritas en la sección 4.3.2, lo que implicaría un cambio significativo en la metodología del proyecto. Además, se debe tener en cuenta que el éxito de esta estrategia está condicionado a una disponibilidad de altos recursos computacionales que permitan explotar su potencial.

Siguiendo por el camino de mejorar la información inicial, otro posible camino de mejora para contrarrestar la limitación del volumen de datos es la de realizar una segmentación de las curvas de luz a partir de información correlacionada de eventos de tránsitos, como son periodos orbitales, profundidad y duración de tránsitos. De esta manera, se podría aprovechar curvas de luz con varios eventos de tránsitos detectados para particionarlas en nuevas curvas de luz que sirvan para entrenar el modelo. Sin embargo, se debe analizar la conveniencia de esta estrategia de cara al riesgo de perder características claves en cada curva de luz.

Por otro lado, las estrategias adicionales a los obtenidas en este proyecto, como otros modelos clasificadores o redes neuronales, presenta una puerta a futuras investigaciones que completen el proceso iniciado con este proyecto buscando tránsitos para estrellas variables. Estas estrategias adicionales han incluido también posibles mejoras para aumentar el conjunto de datos disponibles para el entrenamiento, ya sea mediante la generación de curvas

de luz simuladas o mediante la partición de las curvas de luz en varios componentes individuales.

En añadido, la integración de algoritmos similares de inteligencia artificial con datos de nuevas misiones podría permitir un filtrado que ayudase a facilitar el descubrimiento de exoplanetas que previamente no eran detectables o requerían más procesamiento, debido a las limitaciones de los métodos anteriores, como por ejemplo el algoritmo TLS (Transit Least Squares).

La metodología utilizada en este proyecto puede adaptarse a otros proyectos con mucho volumen de datos, tanto en astronomía como en otros campos.

Referencias bibliográficas

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... & Zheng, X. (2016). TensorFlow: A system for large-scale machine learning. *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation (OSDI '16)*, 265-283. https://doi.org/10.5555/3026877.3026899

Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019). Optuna: A next-generation hyperparameter optimization framework. En *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '19)* (pp. 2623-2631). Association for Computing Machinery. https://doi.org/10.1145/3292500.3330701

Argiles, A., Civera, J., & Montesano, L. (2011). Dense multiplanar scene estimation from a sparse set of images. En *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2011)* (pp. 4448–4454). IEEE. https://doi.org/10.1109/IROS.2011.6094458

Barentsen, G., Hedges, C., Vinícius, Z., Saunders, N., Sergeev, E., Penna, R., ... & Gully-Santiago, M. (2019). Lightkurve: Kepler and TESS time series analysis in Python. *Astrophysics Source Code Library*. https://ascl.net/1812.013

Batalha, N. M., Rowe, J. F., Bryson, S. T., et al. (2013). Planetary Candidates Observed by Kepler. III: Analysis of the First 16 Months of Data. *The Astrophysical Journal Supplement Series, 204*(2), 24. https://doi.org/10.1088/0067-0049/204/2/24

Berger, A., Huber, D., Gaidos, E., van Saders, J. L., Weiss, L. M., Furlan, E., ... & Kitzmann, D. (2018). The Gaia-Kepler Stellar Properties Catalog. II. Planetary Radii and Stellar Types. *The Astrophysical Journal, 866*(2), 99. https://doi.org/10.3847/1538-3881/aba18a

Bergstra, J., Bardenet, R., Bengio, Y., & Kégl, B. (2011). Algorithms for hyper-parameter optimization. *Advances in Neural Information Processing Systems, 24*.

Borucki, W. J., Koch, D., Basri, G., et al. (2010). Kepler Planet-Detection Mission: Introduction and First Results. *Science, 327*(5968), 977-980. https://doi.org/10.1126/science.1185402

Breger, M. (2011). *Delta Scuti stars as probes of their environment: A review.* Astronomical Society of the Pacific Conference Series, 462, 3-14. https://arxiv.org/abs/1106.4003

Breiman, L. (2001). Random forests. *Machine Learning, 45*(1), 5-32. https://doi.org/10.1023/A:1010933404324

Calvet, N., D'Alessio, P., Hartmann, L., Wilner, D. J., Walsh, A. J., & Sitko, M. L. (2002). *Evidence for a developing gap in a 10 myr old protoplanetary disk.* The Astrophysical Journal, 568(2), 1008-1016. https://doi.org/10.1086/339061

Cardoso, J. V. de M., Hedges, C., Gully-Santiago, M. A., Barentsen, G., Vinícius, Z., Dotson, J., ... & Barclay, T. (2018). Lightkurve: Kepler and TESS time series analysis in Python. *Astrophysics Source Code Library*. http://www.ascl.net/1812.013

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785-794. https://doi.org/10.1145/2939672.2939785

Claytor, Z. R., van Saders, J., Cao, L., Pinsonneault, M. H., Teske, J., & Beaton, R. L. (2023). TESS Stellar Rotation up to 80 days in the Southern Continuous Viewing Zone. *The Astrophysical Journal*.

Csizmadia, S. (2020). The Transit and Light Curve Modeller. *Monthly Notices of the Royal Astronomical Society, 496*(4), 4442-4467. https://arxiv.org/pdf/1908.09599

Cui, K., Liu, J., Yang, S., Gao, Q., Yang, H.-Q., Soria, R., He, L., Wang, S., Bai, Y., & Yang, F. (2019). Long rotation period main-sequence stars from Kepler SAP light curves. *Monthly Notices of the Royal Astronomical Society*.

Díez Alonso, E., Sánchez Rodríguez, M. L., Sánchez Lasheras, F., & de Cos Juez, F. J. (2023). One-Dimensional Convolutional Neural Networks for Detecting Transiting Exoplanets.

Eyer, L., & Mowlavi, N. (2008). Variable stars across the observational HR diagram. arXiv. https://arxiv.org/pdf/0712.3797

Gaia Collaboration et al. (2023). Gaia Data Release 3. Catalogue Validation. *Astronomy and Astrophysics*.

Ghachoui, M., Soubkiou, A., Wells, R. D., Rackham, B. V., Triaud, A. H. M. J., Sebastian, D., ... & Schwarz, R. (2023). TESS discovery of a super-Earth orbiting the M-dwarf star TOI-1680.

Guirguis, M., & Huang, H. (2023). The TESS Triple-9 Catalog II: a new set of 999 uniformly vetted exoplanet candidates.

Guzik, J. A. (2021). Highlights of Discoveries for δ Scuti Variable Stars From the Kepler Era. *Frontiers in Astronomy and Space Sciences*.

Haykin, S. (1999). *Neural Networks: A Comprehensive Foundation (2nd ed.).* Prentice Hall.

Henry, G. W., Marcy, G. W., Butler, R. P., & Vogt, S. S. (2000). *A Transiting "51 Peg-like" Planet. The Astrophysical Journal, 529(1)*, L41-L44. https://arxiv.org/pdf/1001.2010

Hey, D. R., Montet, B. T., Pope, B. J. S., Murphy, S. J., & Bedding, T. R. (2021). A search for transits among the δ Scuti variables in Kepler. arXiv. https://arxiv.org/pdf/2108.03785

Hinkle, K. H., Lebzelter, T., & Scharlach, W. W. G. (2002). *AGB Variable Stars. The Astronomical Journal*, 123(2), 1002-1010.

Hippke, M., & Heller, R. (2019). Optimized transit detection algorithm to search for periodic transits of small planets. *Astronomy and Astrophysics, 623*, A39. https://doi.org/10.1051/0004-6361/201834672

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation, 9*(8), 1735-1780. https://doi.org/10.1162/neco.1997.9.8.1735

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*. https://arxiv.org/abs/1502.03167

Jin, Y., Yang, L., & Chiang, C.-E. (2022). Identifying exoplanets with machine learning methods: A preliminary study. arXiv. https://arxiv.org/abs/2204.00721

Kepler Mission Team. (2023). Kepler Q1-Q17 DR25 data release.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*. https://arxiv.org/abs/1412.6980

Kovács, G., Zucker, S., & Mazeh, T. (2002). A box-fitting algorithm in the search for periodic transits. *Astronomy & Astrophysics, 391*(1), 369-377. https://arxiv.org/abs/astro-ph/0206099

Kovács, G., Hartman, J. D., Bakos, G. Á., & Noyes, R. W. (2016). The detection efficiency of the box-fitting least squares (BLS) transit search algorithm. *Monthly Notices of the Royal Astronomical Society, 462*(3), 1882-1890. https://arxiv.org/abs/1607.08269

Kreidberg, L. (2015). Batman: BAsic Transit Model cAlculatioN in Python. *Publications of the Astronomical Society of the Pacific, 127*(957), 1161-1165. https://arxiv.org/abs/1507.08285

Lenz, P., & Breger, M. (2004). Period04: A software package to extract multiple frequencies from real data. *Proceedings of the International Astronomical Union, 2004*(IAUS224), 786-790.

Malik, M., et al. (2022). Exoplanet detection using machine learning. https://escience.aip.de/ag2020/AG2020-Malik.pdf

Marmo, C., Hare, T. M., Erard, S., Minin, M., Pineau, F.-X., Zinzi, A., Cecconi, B., & Rossi, A. P. (2018). FITS Format for Planetary Surfaces: Definitions, Applications, and Best Practices. *Earth and Space Science, 5*(10), 484-494. https://doi.org/10.1029/2018EA000388

Mayor, M., & Queloz, D. (1995). A Jupiter-mass companion to a solar-type star. *Nature, 378*(6555), 355-359. https://arxiv.org/pdf/2405.05115

NASA. (s.f.). Kepler-1634 b. NASA Science. Obtenido en Julio 4, 2024, de https://science.nasa.gov/exoplanet-catalog/kepler-1634-b/

NASA. (s.f.). Kepler-450 c. NASA Science. Obtenido en Julio 4, 2024, de https://science.nasa.gov/exoplanet-catalog/kepler-450-c/

Neath, A. A., & Cavanaugh, J. E. (2012). The Bayesian information criterion: Background, derivation, and applications. *Wiley Interdisciplinary Reviews: Computational Statistics, 4*(2), 199-203. https://doi.org/10.1002/wics.199

Paegert, M., Stassun, K. G., Collins, K. A., Pepper, J., Torres, G., Jenkins, J., Twicken, J. D., & Latham, D. W. (2021). TESS Input Catalog versions 8.1 and 8.2: Phantoms in the 8.0 Catalog and How to Handle Them.

Perryman, M. (2018). The Exoplanet Handbook. Cambridge University Press.

Prechelt, L. (1998). Early stopping-but when? En G. B. Orr & K. R. Müller (Eds.), *Neural Networks: Tricks of the Trade* (pp. 55-69). Springer. https://doi.org/10.1007/3-540-49430-8_3

Ramsay, J. O., & Silverman, B. W. (2005). *Functional Data Analysis* (2nd ed.). Springer. https://doi.org/10.1007/b98888

Schwarz, G. (1978). Estimating the dimension of a model. *The Annals of Statistics, 6*(2), 461-464. https://doi.org/10.1214/aos/1176344136

Seager, S. (2011). *Exoplanets.* University of Arizona Press.

Shallue, C. J., & Vanderburg, A. (2018). Identifying Exoplanets with Deep Learning: A Five-planet Resonant Chain around Kepler-80 and an Eighth Planet around Kepler-90. *The Astronomical Journal, 155*(2), 94. https://doi.org/10.3847/1538-3881/aa9e09

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research, 15*(1), 1929-1958.

Stassun, K. G., Oelkers, R. J., Paegert, M., Torres, G., Pepper, J., De Lee, N., Collins, K., Latham, D. W., Muirhead, P. S., Chittidi, J., ... & Winn, J. N. (2019). The revised TESS input catalog and candidate target list.

Stumpe, M. C., Smith, J. C., Van Cleve, J. E., Twicken, J. D., Barclay, T. S., Fanelli, M. N., Girouard, F. R., Jenkins, J. M., Kolodziejczak, J. J., McCauliff, S. D., & Morris, R. L. (2012). Kepler Presearch Data Conditioning I - Architecture and Algorithms for Error Correction in Kepler Light Curves. *Publications of the Astronomical Society of the Pacific*. https://doi.org/10.1086/667698

Taylor, M. B. (2005). TOPCAT & STIL: Starlink Table/VOTable Processing Software. *Astronomy and Geophysics, 46*(2), 26.

Vanderburg, A., Incha, E., Jacobs, T., LaCourse, D., Bieryla, A., Pass, E., ... & Mann, A. W. (2023). Kepler's Last Planet Discoveries: Two New Planets and One Single-Transit Candidate from K2 Campaign 19. *Monthly Notices of the Royal Astronomical Society*.

Van Reeth, T., Mombarg, J. S. G., Aerts, C., Pedersen, M. G., Garcia, S., Pápics, P. I., Bowman, D. M., & Johnston, C. (2018). Gravity-mode period spacings and near-core rotation rates of 611 γ Doradus stars. *Monthly Notices of the Royal Astronomical Society, 491*(3), 3586-3603. https://doi.org/10.1093/mnras/stz3358

Van Reeth, T., Tkachenko, A., Aerts, C., Pápics, P. I., Triana, S. A., Zwintz, K., & Bloemen, S. (2015). KIC 7760680: A slowly rotating hybrid B-type pulsator studying g-mode period spacings. *Astronomy & Astrophysics, 570*, A8. https://doi.org/10.1051/0004-6361/201424094

von Neumann, J. (1945). First draft of a report on the EDVAC. Institute for Advanced Study, Princeton, NJ.

Yang, G. (2023). Detection of Exoplanets based on the Transit Method. *Highlights in Science, Engineering and Technology*. https://doi.org/10.54097/hset.v31i.5140

Zhong, M., Zhang, L., Yang, Z., & Su, T. (2023). Magnetic Activity of Different Types of Variable Stars Observed by TESS Mission. *Universe, 9*(5), 227. https://doi.org/10.3390/universe9050227

Anexo A. Algoritmos principales

A.1 Descarga de curvas de luz a partir de un archivo .txt

```
import os
import urllib.request

def descargar_curvas_luz(ruta_archivo, ruta_destino):
    with open(ruta_archivo, "r") as archivo:
        urls = archivo.readlines()
        for url in urls:
        nombre_archivo = url.strip().split('/')[-1]
        # Construir una ruta de destino compleeta
        ruta_desetino_completa = os.path.join(ruta_desteino_completa)

ruta_archivo_negativos = r"D:\KOI_negativos.txt"
ruta_archivo_positivos = r"D:\KOI_positivos.txt"
ruta_destino_negativos = r"D:\Negativos"
ruta_destino_positivos = r"D:\Positivos"

descargar_curvas_luz(ruta_archivo_negativos, ruta_destino_negativos)
descargar_curvas_luz(ruta_archivo_positivos, ruta_destino_positivos)
```

A.2 Prewhitening de curvas de luz

```
from __future__ import division, print_function
import numpy as np
from astropy.timeseries import LombScargle
from scipy.optimize import curve_fit
import matplotlib.pyplot as plt
import tqdm

class Prewhitening:

    def __init__(self, time: np.ndarray, flux: np.ndarray):
        """Realiza el prewhitening

        Argumentos:
            time (np.ndarray): Valores del tiempo
            flux (np.ndarray): Flujo correspondiente al parámetro de
entrada `time`
        """
        self.time = time
        self.flux = flux - np.median(flux) #calcula la mediana del flujo y
se resta de todos los valores, centrando datos alrededor de cero
        self.residual = np.copy(self.flux) # se copia los valores del flujo
a self.residual
        self.removed = []


    def prewhiten(self, fmin=None, fmax=None, minimum_snr=5, maxiter=200,
diagnose=True):
```

```
        # Calcula el espectro de amplitud
        freq, amp = self.amplitude_spectrum(self.time, self.residual,
fmin=fmin, fmax=fmax)

        # Calcula el nivel de ruido como un múltiplo del valor mediano del
espectro de amplitud
        noise_level = minimum_snr * np.median(amp)

        # Obtiene una primera estimación de la frecuencia, la amplitud y la
fase de la señal
        f0, a0, phi0 = self.initialize_guess(fmin=fmin, fmax=fmax)

        # bucle que iterará hasta maxiter. Cada iteración busca señales
periódicas en el residuo de la serie temporal y realiza el prewhitening de
la señal si la amplitud es mayor que el nivel de ruido.
        for i in tqdm.tqdm(range(maxiter)):

            if a0 > noise_level:

                # Fit theta to lc
                popt = self.fit([f0, a0, phi0])
                self.removed.append(popt.tolist())

                # Subtract off the fitted model
                self.residual -= self.model(self.time, *popt)

                # Get new params for next iteration
                f0, a0, phi0 = self.initialize_guess(fmin=fmin, fmax=fmax)

            else:
                break

        if diagnose:
            pass



    def fit(self, theta: list) -> np.ndarray:
        """Small wrapper for curve_fit.

        Args:
            time (np.ndarray): Time values
            flux (np.ndarray): Flux values
            theta (list): Array-like of initial guesses

        Returns:
            list: Fitted parameters
        """
        popt, _ = curve_fit(
            self.model, self.time, self.residual, p0=theta,
jac=self.grad_model
        )

        if popt[1] < 0:
            popt[1] *= -1.
```

```
            popt[2] += np.pi

        return popt

    def grad_model(
        self, time: np.ndarray, freq: float, amp: float, phi: float
    ) -> np.ndarray:
        """Gradient function of our pulsation model

        Args:
            time (np.ndarray): Time values
            freq (float): Frequency
            amp (float): Amplitude
            phi (float): Phase

        Returns:
            np.ndarray: Gradient vector (dModel/d_{freq,amp,phi})
        """
        factor = 2 * np.pi * freq * time + phi
        return np.array(
            [
                -2 * np.pi * amp * time * np.sin(factor),
                np.cos(factor),
                -1 * amp * np.sin(factor),
            ]
        ).T

    def model(self, time:np.ndarray, freq:float, amp:float, phi:float) ->
np.ndarray:
        """And at the heart of it all, a tiny model function.

        Args:
            time (np.ndarray): Time values
            freq (float): Frequency
            amp (float): Amplitude
            phi (float): Phase

        Returns:
            np.ndarray: Sinusoid at the given parameters
        """
        return amp * np.cos((2 * np.pi * freq * time) + phi)

    def estimate_background(
        self, x: np.ndarray, y: np.ndarray, log_width: float = 0.01
    ) -> np.ndarray:
        """Estimates the background signal

        Args:
            x (np.ndarray): [description]
            y (np.ndarray): [description]
            log_width (float, optional): [description]. Defaults to 0.01.

        Returns:
            [type]: [description]
        """
        count = np.zeros(len(x), dtype=int)
```

```
        bkg = np.zeros_like(x)
        x0 = np.log10(x[0])
        while x0 < np.log10(x[-1]):
            m = np.abs(np.log10(x) - x0) < log_width
            bkg[m] += np.median(y[m])
            count[m] += 1
            x0 += 0.5 * log_width
        return bkg / count

    def find_highest_peak(self, f: np.ndarray, a: np.ndarray) -> float:
        """Uses three point parabolic interpolation to find the highest
peaks in the amplitude spectrum

        Args:
            f (np.ndarray): Frequency values
            a (np.ndarray): Amplitude values

        Returns:
            float: Maximum frequency
        """
        nu, p = f, a
        nu, p = np.atleast_1d(nu, p)

        # Get index of highest peak.
        imax = np.argmax(p)

        # Determine the frequency value by parabolic interpolation
        if imax == 0 or imax == p.size - 1:
            nu_peak = p[imax]
        else:
            # Get values around the maximum. This is kinda gross
            frq1 = nu[imax - 1]
            frq2 = nu[imax]
            frq3 = nu[imax + 1]
            y1 = p[imax - 1]
            y2 = p[imax]
            y3 = p[imax + 1]

            # Parabolic interpolation formula.
            t1 = (y2 - y3) * (frq2 - frq1) ** 2 - (y2 - y1) * (frq2 - frq3)
** 2
            t2 = (y2 - y3) * (frq2 - frq1) - (y2 - y1) * (frq2 - frq3)
            nu_peak = frq2 - 0.5 * t1 / t2
        return nu_peak

    def amplitude_spectrum(
        self, t, y, fmin: float = None, fmax: float = None,
oversample_factor: float = 5.0,
    ) -> tuple:
        """Calculates the amplitude spectrum at a given time and flux input

        Args:
            t (np.ndarray): Time values
            y (np.ndarray): Flux values
            fmin (float, optional): Minimum frequency. Defaults to None.
            fmax (float, optional): Maximum frequency. Defaults to None.
```

```
            oversample_factor (float, optional): Amount by which to
oversample the light curve. Defaults to 5.0.

        Returns:
            tuple: Frequency and amplitude arrays
        """
        # t, y = self.time, self.residual
        tmax = t.max()
        tmin = t.min()
        df = 1.0 / (tmax - tmin) # Resolución espectral: separación máxima
entre dos frecuencias

        if fmin is None:
            fmin = df


        # fmax representa la frecuencia máxima que se puede resolver de
manera confiable en el conjunto de datos, basada en el criterio de Nyquist,
que establece que la frecuencia máxima que se puede detectar es la mitad de
la frecuencia de muestreo. Esto asegura que los datos se puedan analizar
correctamente sin introducir efectos de aliasing.
        if fmax is None:
            fmax = 0.5 / np.median(np.diff(t))  # *nyq_mult


        # se genera el arreglo de frecuencia que comienza en fmin, termina
en fmax, y avanza en incrementos de df/oversample_factor
        freq = np.arange(fmin, fmax, df / oversample_factor)
        #print(freq)

        # Calcular el Periodograma de Lomb-Scargle
        model = LombScargle(t, y) #Se crea una instancia de Lomb-Scargle
con los datos de la serie de tiempo y flujo
        sc = model.power(freq, method="cython", normalization="standard") #
calcula el espectro de potencia para el arreglo freq

        fct = np.sqrt(4.0 / len(t)) # calcula un factor de normalización
para las amplitudes en el espectro de amplitud
        #print(sc)
        amp = np.sqrt(sc) * fct # aplica el factor de normalización
calculado a las amplitudes

        return freq, amp

    def dft_phase(self, x: np.ndarray, y: np.ndarray, f: float) -> float:
        """Calculates the phase at a single frequency using the Discrete
Fourier Transform

        Args:
            x (np.ndarray): Time values
            y (np.ndarray): Flux values
            f (float): Frequency

        Returns:
            float: Phase at given frequency
        """
```

```
        expo = 2.0 * np.pi * f * x
        return np.arctan2(np.sum(y * np.sin(expo)), np.sum(y *
np.cos(expo)))

    def initialize_guess(self, fmin: float, fmax: float):
        time, flux = self.time, self.residual
        f, a = self.amplitude_spectrum(time, flux, fmin=fmin, fmax=fmax,
oversample_factor=5.0)

        # Get freq of max power using parabolic interpolation
        f0 = self.find_highest_peak(f, a)
        # Calculate a0 at f0
        a0 = np.sqrt(
            LombScargle(time, flux).power(f0, method="cython",
normalization="standard")
        ) * np.sqrt(4.0 / len(time))
        # Calculate phi0, since ASTC needs to be negative
        phi0 = -1 * self.dft_phase(time, flux, f0)
        return f0, a0, phi0
```

A.3 Generación de nuevos conjuntos de datos

```
import skfda
from skfda.exploratory.visualization import FPCAPlot
from skfda.preprocessing.dim_reduction import FPCA
from skfda.representation.basis import (
    BSplineBasis,
    FourierBasis,
    MonomialBasis,
)


# Definimos los directorios de los archivos
directorios = {
    'Negativos': 'D:/Master/TFM/CurvasLuz/Curvas de luz procesadas/modelo/
negativos',
    'Positivos': 'D:/Master/TFM/CurvasLuz/Curvas de luz procesadas/modelo/
positivos'
}

# Preparar estructura de datos
datos = {key: [] for key in directorios}

# Procesar cada carpeta
for grupo, path in directorios.items():
    archivos = [f for f in os.listdir(path) if f.endswith('.csv')]
    for archivo in tqdm(archivos, desc=f"Procesando {grupo}",
unit='archivo'):
        archivo_ruta = os.path.join(path, archivo)
        try:
            # Cargamos el archivo CSV
            df = pd.read_csv(archivo_ruta)
            # Asumiendo que la columna de interés se llama 'RESIDUAL'
            flujo = df['Residual'].values
```

```
            tiempo = df['Time'].values
#             tiempo = df['Time'].values if 'TIME' in df.columns else
np.arange(len(flujo))
            identificador = archivo.split('_')[0]
            datos[grupo].append({'identificador': identificador, 'flujo':
flujo, 'tiempo': tiempo})
        except Exception as e:
            print(f"Error al procesar {archivo}: {str(e)}. Saltando al
siguiente archivo.")

# ruta para guardar los datasets:
path = 'D:/Master/TFM/CurvasLuz/Curvas de luz procesadas/modelo/datasets'


# Extraer los arrays de cada grupo
positivos = pd.DataFrame(datos['Positivos'])
negativos = pd.DataFrame(datos['Negativos'])
negativos.head(2)


# Cálculo de cantidad de mediciones para cada curva.
negativos['len'] = negativos['flujo'].str.len()
positivos['len'] = positivos['flujo'].str.len()
# Muestra distribución de cantidad de mediciones por negativos y positivos.
plt.title('Distribución de cantidad de mediciones negativos')
plt.boxplot(negativos['len'])
plt.show()
plt.title('Distribución de cantidad de mediciones positivos')
plt.boxplot(positivos['len'])
plt.show()
# Se considera que los flujos observados menores de 10.000 son datos
atípicos de la muestra.
# Para ser procesado con FPCA es necesario que los flujos tengan la misma
cantidad de mediciones.


# Se recortan las columnas y eliminan registros que tengan valores nulos.
negativos = pd.DataFrame([curva[1] for curva in negativos.values],
index=negativos.identificador)[np.arange(10000)]
positivos = pd.DataFrame([curva[1] for curva in positivos.values],
index=positivos.identificador)[np.arange(10000)]


print(negativos.shape, ' Negativos: Se eliminan los NaN->',
negativos.dropna().shape)
print(positivos.shape, ' Positivos: Se eliminan los NaN->',
positivos.dropna().shape)
X_negativos = negativos.dropna()
X_positivos = positivos.dropna()


def sup_inf(df, std, var_std):
    """
```

```
        Asigna un marcador = 1 cuando el valor de la serie está var_std-
desviaciones estándar por encima o debajo.
    Args:
        df (pd.DataFrame): Todos los valores de la serie de tiempo a
procesar
        std (float): Desviación estándar de la serie de tiempo.
        var_std (int): Multiplicador de la desviación estándar para
implementar criterio del límite de la banda de Bollinger
    Returns:
        df[nombre_col] (np.Array): Listado de marcadores asociados a la
serie. (1 o 0)
        nombre_col (str): Nombre de la columna con el valor del marcador
    """
    nombre_col = f'ext_{str(var_std)}_std'
    df[nombre_col] = 0
    df.loc[(df.iloc[:,0] > var_std*std) | (df.iloc[:,0] < (-var_std*std)),
nombre_col] = 1
    return df[nombre_col], nombre_col

def get_df_outliers(serie, std_limits):
    """
        Genera el dataframe con todos los marcadores asociados a los
lìmites de las bandas de bollinger asociados.
    Args:
        serie (np.Array): Todos los valores de la serie de tiempo a
procesar.
        std_limits (list): Todos los lìmites de las bandas de Bollinger
para implementar.
    Returns:
        df (pd.DataFrame): Todos los marcadores asociados a la serie. (1 o
0) de todos los límites.
    """
    std = np.std(serie)
    df = pd.DataFrame(serie)
    for i in std_limits:
        serie, nombre_col = sup_inf(df, std, i)
        df[nombre_col] = serie
    return df

def get_feature_outlier(df, label, values):
    """
        Aplica el criterio de 'conteo' o 'porcentaje' a los valores que
están fuera de las bandas de Bollinger.
    Args:
        df (pd.DataFrame):  Todos los marcadores asociados a la serie. (1 o
0) de todos los límites.
        label (str): Tipo de feature a crear, Conteo o Porcentaje de
valores fuera de las bandas de bollinger
        values (list): Todos los lìmites de las bandas de Bollinger para
implementar.
    Returns:
        df_return (pd.DataFrame): Las features derivadas de la combinacion
de label con values asociada.
    """
    columns = [f'{label}_std_{i}' for i in values]
    if label == 'count':
```

```
            df_return = pd.DataFrame(df.sum()).T
            df_return.columns = columns
            return df_return
        if label == 'percentage':
            df_return = pd.DataFrame(df.sum()/df.shape[0]).T
            df_return.columns = columns
            return df_return

def get_features(serie):
    suma = np.sum(serie)
    maximo = np.max(serie)
    minimo = np.min(serie)
    media = np.mean(serie)
    mediana = np.median(serie)
    std = np.std(serie)

    return pd.DataFrame({'suma': [suma], 'maximo':[maximo],
'minimo':[minimo], 'media':[media], 'mediana':[mediana],'std':[std]})



def get_features_adicionales(serie):
    # Generaciòn de features simples (max, min, mean, median, std)
    df_simple_feature = get_features(serie)

    # Generaciòn features basadas en bandas de Bollinger (n-veces
desviación estándar)
    std_limits = [1,2,3,4,5,6,7,8,9,10,50,100]
    df_outliers = get_df_outliers(serie, std_limits).iloc[:,1:]
    df_count = get_feature_outlier(df_outliers, 'count', std_limits)
    df_percentage = get_feature_outlier(df_outliers, 'percentage',
std_limits)
    df_bollinger_features =  pd.concat([df_count,df_percentage],  axis=1)

    # Generacion de features basadas en percentiles (1)
    list_percentil =
[1,2,3,4,5,10,15,20,25,30,40,45,55,60,70,80,85,90,95,96,97,98,99]
    perc_values = [np.percentile(serie, i) for i in list_percentil]
    perc_cols = [f'percentile{str(i)}' for i in list_percentil]
    df_percentile_features = pd.DataFrame(perc_values).T
    df_percentile_features.columns = columns=perc_cols
    df_percentile_features

    df_new_features = pd.concat([df_simple_feature,df_bollinger_features,
df_percentile_features], axis=1)
    #df_new_features.index = [label]
    return df_new_features



# Crear las etiquetas para las clases
y_positivos = np.ones(len(X_positivos))
y_negativos = np.zeros(len(X_negativos))



# Concatenar datos y etiquetas para el entrenamiento
X = pd.concat([X_positivos, X_negativos]).drop_duplicates()
```

```
y = np.concatenate((y_positivos, y_negativos))


features_adicionales = pd.concat([get_features_adicionales(serie) for serie
in X.values])
features_adicionales.index = X.index
print(features_adicionales.shape)
features_adicionales.head(2)
fd = X


grid_points = X.columns.astype(float)
data_matrix = X
fd = skfda.FDataGrid(
    data_matrix=data_matrix,
    grid_points=grid_points,
)


# Inicialización de Objeto FPCA para tranformar Serie de tiempo en 10
componentes.
fpca_discretized = FPCA(n_components=10)


# Ajustar transformador a los datos y almacenar los scores de cada
componente.
scores = fpca_discretized.fit_transform(fd)


X_original = X
X = pd.DataFrame(scores, index=X.index)


# Porcentaje de varianza explicada por cada componente principal.
fpca_discretized.explained_variance_ratio_


# Dividir los datos en entrenamiento y prueba
# Train 60%, test 20%, validacion 20%
X_80_porciento, X_validacion, y_80_porciento, y_validacion =
train_test_split(X, y, test_size=0.2, random_state=42)
#  Se divide 80_porciento entre: X_train 60% y X_test 20%
X_train, X_test, y_train, y_test = train_test_split(X_80_porciento,
y_80_porciento, test_size=y_validacion.shape[0], random_state=42)
print(X_train.shape[0], X_test.shape[0], X_validacion.shape[0])
print(y_train.shape[0], y_test.shape[0], y_validacion.shape[0])


X_train = pd.merge(X_train, features_adicionales, right_index= True,
left_index = True, how='inner')
```

```
X_test = pd.merge(X_test, features_adicionales, right_index= True,
left_index = True, how='inner')
X_validacion = pd.merge(X_validacion, features_adicionales, right_index=
True, left_index = True, how='inner')


X_train_new_features = X_train.copy()
X_train_new_features['clase'] = y_train
X_train_new_features.to_csv(f'{path}X_train_new_features_2.csv')
X_train.to_csv(f'{path}X_train.csv')
X_test.to_csv(f'{path}X_test.csv')
X_validacion.to_csv(f'{path}X_validacion.csv')
pd.DataFrame(y_train).to_csv(f'{path}y_train.csv')
pd.DataFrame(y_test).to_csv(f'{path}y_test.csv')
pd.DataFrame(y_validacion).to_csv(f'{path}y_validacion.csv')
features_adicionales.to_csv(f'{path}features_adicionales.csv')
```

A.4 Optimización hiperparámetros XGBoost

```
import os
import optuna
import pandas as pd
from optuna.samplers import RandomSampler, GPSampler, BaseSampler,
TPESampler
path = 'D:/Master/TFM/CurvasLuz/Curvas de luz
procesadas/modelo/datasets/ajuste balance clases/quitando negativos/'


X_train = pd.read_csv(f'{path}X_train.csv', index_col=0)
X_test = pd.read_csv(f'{path}X_test.csv', index_col=0)
X_validacion = pd.read_csv(f'{path}X_validacion.csv', index_col=0)
y_train = pd.read_csv(f'{path}y_train.csv', index_col=0)['0']
y_test = pd.read_csv(f'{path}y_test.csv', index_col=0)['0']
y_validacion = pd.read_csv(f'{path}y_validacion.csv', index_col=0)['0']


def objective(trial):

    X_train = pd.read_csv(f'{path}X_train.csv', index_col=0)
    X_validacion = pd.read_csv(f'{path}X_validacion.csv', index_col=0)
    y_train = pd.read_csv(f'{path}y_train.csv', index_col=0)['0']
    y_validacion = pd.read_csv(f'{path}y_validacion.csv', index_col=0)['0']

    max_depth = trial.suggest_int("max_depth", 3, 20, log=True)
    n_estimators = trial.suggest_int("n_estimators", 100, 1000)
    learning_rate = trial.suggest_float("learning_rate", 0.01, 0.3,
log=True)
    min_child_weight = trial.suggest_int("min_child_weight", 20, 45)
    lambda_param = trial.suggest_float("lambda", 1e-8, 1.0, log=True)
    alpha = trial.suggest_float("alpha", 1e-8, 1.0, log=True)
    subsample = trial.suggest_float("subsample", 0.5, 1.0, step=0.1)
    gamma = trial.suggest_float("gamma", 1e-8, 1.0, log=True)
```

```
    colsample_bytree = trial.suggest_float("colsample_bytree", 0.5, 1.0,
step=0.1)
    scale_pos_weight = trial.suggest_float("scale_pos_weight", 0.1, 10.0,
log=True)
    max_delta_step = trial.suggest_int("max_delta_step", 0, 10)
    colsample_bylevel = trial.suggest_float("colsample_bylevel", 0.5, 1.0,
step=0.1)
    random_state = 42

    params = {
        'max_depth': max_depth,
        'n_estimators': n_estimators,
        'learning_rate': learning_rate,
        'random_state': random_state,
        'min_child_weight': min_child_weight,
        'lambda': lambda_param,
        'alpha': alpha,
        'subsample': subsample,
        'gamma': gamma,
        'colsample_bytree': colsample_bytree,
        'scale_pos_weight': scale_pos_weight,
        'max_delta_step': max_delta_step,
        'colsample_bylevel': colsample_bylevel
    }

    model = xgb.XGBClassifier(**params)
    model.fit(X_train, y_train)

    y_pred_validacion = model.predict(X_validacion)

    f1_score_validacion = m.f1_score(y_validacion, y_pred_validacion)
    accuracy_validacion = m.accuracy_score(y_validacion, y_pred_validacion)

    return f1_score_validacion, accuracy_validacion



sampler = TPESampler(seed=42)
study = optuna.create_study(directions=['maximize', 'maximize'],
sampler=sampler,
                            storage="sqlite:///db.sqlite3",
                            study_name="xg_102_max_accuracy_f1score"
                           )
study.optimize(objective, n_trials=5000, show_progress_bar = True)
study.best_trial


# Utilizando un clasificador de XGBoost
xgb_model = xgb.XGBClassifier(**params)
xgb_model.fit(X_train, y_train)


# Predicciones y reporte de clasificación
print("Reporte de clasificación del conjunto de train de XGBoost:")
y_pred_train = xgb_model.predict(X_train)
```

```
print(classification_report(y_train, y_pred_train))

y_pred = xgb_model.predict(X_validacion)
print("Reporte de clasificación del conjunto de validacion de XGBoost:")
print(classification_report(y_validacion, y_pred))
cf_matrix = confusion_matrix(y_validacion, y_pred)
print(cf_matrix)
```

A.5 Optimización hiperparámetros RandomForest

```
def objective(trial):

    X_train = pd.read_csv(f'{path}X_train.csv', index_col=0)
    X_validacion = pd.read_csv(f'{path}X_validacion.csv', index_col=0)
    y_train = pd.read_csv(f'{path}y_train.csv', index_col=0)['0']
    y_validacion = pd.read_csv(f'{path}y_validacion.csv', index_col=0)['0']

    bootstrap = trial.suggest_categorical("bootstrap", [True, False])

    if bootstrap:
        max_samples = trial.suggest_float("max_samples", 0.5, 1.0)
    else:
        max_samples = None


    params = {
        'n_estimators': trial.suggest_int("n_estimators", 100, 1000),
        'max_depth': trial.suggest_int("max_depth", 2, 300, log=True),
        'min_samples_split': trial.suggest_int("min_samples_split", 2, 45),
        'min_samples_leaf': trial.suggest_int("min_samples_leaf", 1, 45),
        'min_weight_fraction_leaf':
trial.suggest_float("min_weight_fraction_leaf", 0.0, 0.5),
        'random_state': 42,  # Fixed random_state for reproducibility
        'max_samples': max_samples,
        'max_features': trial.suggest_categorical("max_features", ["auto",
"sqrt", "log2", None]),
        'bootstrap': bootstrap,
        'criterion': trial.suggest_categorical("criterion", ["gini",
"entropy"])
    }


    model = RandomForestClassifier(**params)
    model.fit(X_train, y_train)

    y_pred_train = model.predict(X_train)
    y_pred_validacion = model.predict(X_validacion)

    f1_score_validacion = m.f1_score(y_validacion, y_pred_validacion)
    accuracy_validacion = m.accuracy_score(y_validacion, y_pred_validacion)

    return f1_score_validacion, accuracy_validacion


# Utilizando un clasificador de RandomForest
```

```
rf_model = RandomForestClassifier(**params)
rf_model.fit(X_train, y_train)


# Predicciones y reporte de clasificación
print("Reporte de clasificación del conjunto de train de RandomForest:")
y_pred_train = rf_model.predict(X_train)
print(classification_report(y_train, y_pred_train))

y_pred = rf_model.predict(X_validacion)
print("Reporte de clasificación del conjunto de validacion de
RandomForest:")
print(classification_report(y_validacion, y_pred))
cf_matrix = confusion_matrix(y_validacion, y_pred)
print(cf_matrix)



sampler = TPESampler(seed=42)

study = optuna.create_study(directions=['maximize', 'maximize'],
sampler=sampler,
                            storage="sqlite:///db.sqlite3",
                            study_name="rf_101_max_accuracy_f1score"
                            )
study.optimize(objective, n_trials=5000, show_progress_bar = True)
study.best_trial
```

A.6 Optimización red neuronal TensorFlow

```
import os
import pandas as pd
import numpy as np
from tqdm import tqdm
from sklearn.metrics import confusion_matrix

# Definimos los directorios de los archivos
directorios = {
    'Muestra2': 'D:/CurvasLuz_Prewhitened_CSV/Muestra2_Prewhitened_CSV',
    'Negativos':
'D:/CurvasLuz_Prewhitened_CSV/negativos_concatenados_Prewhitened_CSV',
    'Positivos':
'D:/CurvasLuz_Prewhitened_CSV/positivos_concatenados_Prewhitened_CSV',
}
# Preparar estructura de datos
datos = {key: [] for key in directorios}

# Procesar cada carpeta
for grupo, path in directorios.items():
    archivos = [f for f in os.listdir(path) if f.endswith('.csv')]
    for archivo in tqdm(archivos, desc=f"Procesando {grupo}",
unit='archivo'):
        archivo_ruta = os.path.join(path, archivo)
```

```
        try:

            df = pd.read_csv(archivo_ruta)

            flujo = df['Residual'].values
            tiempo = df['Time'].values
            identificador = archivo.split('_')[0]
            datos[grupo].append({'identificador': identificador, 'flujo':
flujo, 'tiempo': tiempo})
        except Exception as e:
            print(f"Error al procesar {archivo}: {str(e)}. Saltando al
siguiente archivo.")



def generar_caracteristicas(curva):
    flujo = curva['flujo']
    tiempo = curva['tiempo']
    caracteristicas = {
        'flujo': flujo,
        'media_movil': pd.Series(flujo).rolling(window=5,
min_periods=1).mean().values,
        'derivada': np.gradient(flujo, tiempo)
    }
    return caracteristicas

# Generar características para todos los datos
for grupo in datos.keys():
    for curva in datos[grupo]:
        curva['caracteristicas'] = generar_caracteristicas(curva)



from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.sequence import pad_sequences

def preparar_datos_para_modelo(datos, grupos, maxlen=5000):
    X = []
    y = []
    for grupo, label in grupos.items():
        for curva in datos[grupo]:
            caracteristicas = curva['caracteristicas']
            secuencia = np.vstack([
                caracteristicas['flujo'],
                caracteristicas['media_movil'],
                caracteristicas['derivada']
            ]).T
            X.append(secuencia)
            y.append(label)
    X = pad_sequences(X, maxlen=maxlen, dtype='float32', padding='post',
truncating='post')
    return np.array(X), np.array(y)

grupos = {'Positivos': 1, 'Negativos': 0}
X, y = preparar_datos_para_modelo(datos, grupos)
```

```
# Dividir los datos en conjuntos de entrenamiento, validación y prueba
X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.3,
random_state=42)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp,
test_size=0.5, random_state=42)



from sklearn.preprocessing import StandardScaler
from tensorflow.keras.preprocessing.sequence import pad_sequences
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM, Dropout
from tensorflow.keras.callbacks import EarlyStopping
import optuna
from tensorflow.keras.optimizers import Adam

# Definir la función objetivo
def objective(trial):
    # Sugerir valores de hiperparámetros
    lstm_units = trial.suggest_int('lstm_units', 32, 128)
    dropout_rate = trial.suggest_float('dropout_rate', 0.1, 0.5)
    learning_rate = trial.suggest_float('learning_rate', 1e-5, 1e-1)
    batch_size = trial.suggest_int('batch_size', 16, 64)
    epochs = trial.suggest_int('epochs', 10, 50)

    # Construir el modelo
    model = Sequential()
    model.add(LSTM(lstm_units, input_shape=(X_train.shape[1],
X_train.shape[2]), return_sequences=True))
    model.add(Dropout(dropout_rate))
    model.add(LSTM(lstm_units, return_sequences=False))
    model.add(Dropout(dropout_rate))
    model.add(Dense(1, activation='sigmoid'))

    # Compilar el modelo
    model.compile(optimizer=Adam(learning_rate=learning_rate),
loss='binary_crossentropy', metrics=['accuracy'])

    # Early stopping
    early_stopping = EarlyStopping(monitor='val_loss', patience=5,
restore_best_weights=True)

    # Entrenar el modelo
    history = model.fit(X_train, y_train, epochs=epochs,
batch_size=batch_size,
                        validation_data=(X_val, y_val),
callbacks=[early_stopping], verbose=0)

    # Evaluar el modelo
    val_accuracy = history.history['val_accuracy'][-1]

    return val_accuracy

# Crear el estudio de Optuna y optimizar
study = optuna.create_study(
```

```
    direction='maximize',
    study_name="redes prueba 2",
    storage="sqlite:///db.sqlite3",
    load_if_exists=True
)
study.optimize(objective, n_trials=300)

# Obtener los mejores hiperparámetros
best_params = study.best_params
print('Mejores hiperparámetros: ', best_params)

# Obtener los mejores hiperparámetros del primer estudio
best_params = study.best_params

# Definir la función objetivo para el segundo estudio
def fine_tune_objective(trial):
    # Ajustar ligeramente los valores de hiperparámetros basados en los
mejores resultados anteriores
    lstm_units = trial.suggest_int('lstm_units', max(32,
best_params['lstm_units'] - 10), min(128, best_params['lstm_units'] + 10))
    dropout_rate = trial.suggest_float('dropout_rate', max(0.1,
best_params['dropout_rate'] - 0.1), min(0.5, best_params['dropout_rate'] +
0.1))
    learning_rate = trial.suggest_float('learning_rate', max(1e-5,
best_params['learning_rate'] / 2), min(1e-1, best_params['learning_rate'] *
2))
    batch_size = trial.suggest_int('batch_size', max(16,
best_params['batch_size'] - 16), min(128, best_params['batch_size'] + 16))
    epochs = trial.suggest_int('epochs', max(10, best_params['epochs'] -
10), min(100, best_params['epochs'] + 10))

    # Construir el modelo
    model = Sequential()
    model.add(LSTM(lstm_units, input_shape=(X_train.shape[1],
X_train.shape[2]), return_sequences=True))
    model.add(Dropout(dropout_rate))
    model.add(LSTM(lstm_units, return_sequences=False))
    model.add(Dropout(dropout_rate))
    model.add(Dense(1, activation='sigmoid'))

    # Compilar el modelo
    model.compile(optimizer=Adam(learning_rate=learning_rate),
loss='binary_crossentropy', metrics=['accuracy'])

    # Early stopping
    early_stopping = EarlyStopping(monitor='val_loss', patience=5,
restore_best_weights=True)

    # Entrenar el modelo
    history = model.fit(X_train, y_train, epochs=epochs,
batch_size=batch_size,
                        validation_data=(X_val, y_val),
callbacks=[early_stopping], verbose=0)

    # Evaluar el modelo
    val_accuracy = history.history['val_accuracy'][-1]
```

```
    return val_accuracy

# Crear el segundo estudio de Optuna y optimizar
fine_tune_study = optuna.create_study(
    direction='maximize',
    study_name="redes prueba 2 afinado",
    storage="sqlite:///db_finetune.sqlite3",
    load_if_exists=True
)
fine_tune_study.optimize(fine_tune_objective, n_trials=50)  # Reducido para
una optimización más fina

# Obtener los mejores hiperparámetros del segundo estudio
fine_tune_best_params = fine_tune_study.best_params
print('Mejores hiperparámetros afinados: ', fine_tune_best_params)

# Nota sobre el estudio guardado
fine_tune_study.set_user_attr("note", "Estudio de optimización LSTM afinado
guardado para su seguimiento en Optuna Dashboard")



# Construir el modelo con los mejores hiperparámetros
model = Sequential()
model.add(LSTM(best_params['lstm_units'], input_shape=(X_train.shape[1],
X_train.shape[2]), return_sequences=True))
model.add(Dropout(best_params['dropout_rate']))
model.add(LSTM(best_params['lstm_units'], return_sequences=False))
model.add(Dropout(best_params['dropout_rate']))
model.add(Dense(1, activation='sigmoid'))

# Compilar el modelo
model.compile(optimizer=Adam(learning_rate=best_params['learning_rate']),
loss='binary_crossentropy', metrics=['accuracy'])

# Entrenar el modelo
history = model.fit(X_train, y_train, epochs=best_params['epochs'],
batch_size=best_params['batch_size'], validation_data=(X_val, y_val))

# Evaluar el modelo
train_loss, train_accuracy = model.evaluate(X_train, y_train)
val_loss, val_accuracy = model.evaluate(X_val, y_val)
test_loss, test_accuracy = model.evaluate(X_test, y_test)

print(f'Train Loss: {train_loss}, Train Accuracy: {train_accuracy}')
print(f'Validation Loss: {val_loss}, Validation Accuracy: {val_accuracy}')
print(f'Test Loss: {test_loss}, Test Accuracy: {test_accuracy}')



# Evaluar el modelo en los conjuntos de prueba, validación y entrenamiento
train_loss, train_accuracy = model.evaluate(X_train, y_train)
val_loss, val_accuracy = model.evaluate(X_val, y_val)
test_loss, test_accuracy = model.evaluate(X_test, y_test)
```

```
print(f'Train Loss: {train_loss}, Train Accuracy: {train_accuracy}')
print(f'Validation Loss: {val_loss}, Validation Accuracy: {val_accuracy}')
print(f'Test Loss: {test_loss}, Test Accuracy: {test_accuracy}')

# Visualizar la historia del entrenamiento
plt.figure(figsize=(12, 4))

# Pérdida durante el entrenamiento
plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.title('Loss Over Epochs')

# Precisión durante el entrenamiento
plt.subplot(1, 2, 2)
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.title('Accuracy Over Epochs')

plt.show()



from sklearn.metrics import confusion_matrix, classification_report

# Función para evaluar y mostrar resultados
def evaluar_modelo(model, X, y, conjunto):
    y_pred = model.predict(X)
    y_pred_class = (y_pred > 0.5).astype(int)

    cm = confusion_matrix(y, y_pred_class)
    cr = classification_report(y, y_pred_class)

    print(f'Confusion Matrix - {conjunto}:')
    print(cm)
    print(f'Classification Report - {conjunto}:')
    print(cr)

# Evaluar el modelo en los tres conjuntos
evaluar_modelo(model, X_train, y_train, 'Train')
evaluar_modelo(model, X_val, y_val, 'Validation')
evaluar_modelo(model, X_test, y_test, 'Test')
```

A.7 Generación curvas de luz sintéticas

```
import numpy as np
```

```
# Inicializamos las listas para almacenar los flujos totales y las
estadísticas de cada grupo
flujos_totales = []
maximos = []
minimos = []
promedios = []
medianas = []
longitudes = []

# Calculamos estadísticas para cada grupo
for grupo, datos_grupo in datos.items():
    for dato in datos_grupo:
        flujos_totales.extend(dato['flujo'])
        maximos.append(np.max(dato['flujo']))
        minimos.append(np.min(dato['flujo']))
        promedios.append(np.mean(dato['flujo']))
        medianas.append(np.median(dato['flujo']))
        longitudes.append(len(dato['flujo']))

# Estadísticas globales de todos los grupos
maximo_total = np.max(flujos_totales)
minimo_total = np.min(flujos_totales)
maximo_mediana = np.median(maximos)
minimo_mediana = np.median(minimos)
maximo_promedio = np.median(maximos)
minimo_promedio = np.median(minimos)
promedio_total = np.mean(flujos_totales)
mediana_total = np.median(flujos_totales)
numero_valores_promedio = len(promedios)
numero_valores_mediana = len(medianas)
longitud_media = np.mean(longitudes)
longitud_mediana = np.median(longitudes)

# Imprimimos los resultados
print("Máximo total:", maximo_total)
print("Mínimo total:", minimo_total)
print("Máximo promedio:", maximo_promedio)
print("Mínimo promedio:", minimo_promedio)
print("Máximo mediana:", maximo_mediana)
print("Mínimo mediana:", minimo_mediana)
print("Promedio total de flujo:", promedio_total)
print("Mediana total de flujo:", mediana_total)
print("Número de valores promedio:", numero_valores_promedio)
print("Número de valores mediana:", numero_valores_mediana)
print("Longitud media de las curvas de luz:", longitud_media)
print("Longitud mediana de las curvas de luz:", longitud_mediana)



import numpy as np
import pandas as pd
import os
import random

# Los valores de la función se cambian en base a los valores obtenidos en
el algoritmo del apartado A7
```

```
def generate_light_curve(n_points, transit=False):
    """Generate a single light curve with or without a transit."""
    time = np.linspace(0, 64793, n_points)
    noise = np.random.normal(-6, 7, n_points)
    flux = -6.0 + noise
    '''
    if transit:
        # Simple transit model using a Gaussian dip
        depth = 700
        width = 100
        center = 30000
        transit_shape = depth * np.exp(-0.5 * ((time - center)**2 /
(width**2)))
        flux -= transit_shape
    '''

    if transit:
        n_transits = random.randint(3, 100)
        depth = np.random.uniform(7, 700)
        width = np.random.uniform(5, 30)
        interval = n_points / n_transits
        for i in range(n_transits):
            center = i * interval + interval / 2
            transit_shape = depth * np.exp(-0.5 * ((time - center)**2 /
(width**2)))
            flux -= transit_shape

    return time, flux

def create_light_curves(n_positive, n_negative, n_points=64793):
    """Create a set of light curves with specified number of positives and
negatives."""
    base_path = "D:\\CurvasLuz_Prewhitened_CSV\\Extra"
    pos_path = os.path.join(base_path, "Positivos")
    neg_path = os.path.join(base_path, "Negativos")


    os.makedirs(pos_path, exist_ok=True)
    os.makedirs(neg_path, exist_ok=True)


    for i in range(n_positive):
        time, flux = generate_light_curve(n_points, transit=True)
        df = pd.DataFrame({'Time': time, 'Residual': flux})
        df.to_csv(os.path.join(pos_path, f"positive_{i}.csv"), index=False)


    for i in range(n_negative):
        time, flux = generate_light_curve(n_points, transit=False)
        df = pd.DataFrame({'Time': time, 'Residual': flux})
        df.to_csv(os.path.join(neg_path, f"negative_{i}.csv"), index=False)


n_positive = 250
n_negative = 500
create_light_curves(n_positive, n_negative)
```

A.8 Mejor modelo XGBoost

```
y_pred_test = xgb_model.predict(X_test)
print("Reporte de clasificación del conjunto de test de XGBoost (No se usa
para ajustar/tunning parámetros):")
print(classification_report(y_test, y_pred_test))

cf_matrix = confusion_matrix(y_test, y_pred_test)
cf_matrix_pct = cf_matrix/np.sum(cf_matrix)

group_names = ['True Neg','False Pos','False Neg','True Pos']
group_counts = ["{0:0.0f}".format(value) for value in cf_matrix.flatten()]
labels = [f"{v1}\n{v2}" for v1, v2 in zip(group_names,group_counts)]
labels = np.asarray(labels).reshape(2,2)

categories = ['Zero', 'One']

print(labels)
print(cf_matrix_pct)



def a_simple_classifier(x, thres = 0.5, model=None):
    predicted = pd.DataFrame(model.predict_proba(X_validacion))[1]
    for i in range(len(x)):
        if predicted[i] > thres:
            predicted[i] = 1
        else:
            predicted[i] = 0
    return predicted

def calculate_metrics(predicted, actual, model=None):
    #print(predicted.sum())
    #print(actual.sum())
    TP, FP, TN, FN = 0, 0, 0, 0
    for i in range(len(predicted)):
        if   (predicted[i] == 1) & (actual[i] == 1):
            TP += 1
        elif (predicted[i] == 1) & (actual[i] == 0):
            FP += 1
        elif (predicted[i] == 0) & (actual[i] == 0):
            TN += 1
        else:
            FN += 1

    accuracy  = (TP + TN) / (TP + FP + TN + FN)
    precision = (TP) / ((TP + FP) if (TP + FP) >0 else 1 )
    recall    = (TP) / ((TP + FN) if (TP + FN) >0 else 1 )
    f1_score  = (2 * precision * recall) / ((precision + recall) if
(precision + recall) >0 else 1 )

    return accuracy, precision, recall, f1_score



thresh = np.linspace(0,1,100)
```

```
accuracy = np.zeros(len(thresh))
precision = np.zeros(len(thresh))
recall = np.zeros(len(thresh))
f1_score = np.zeros(len(thresh))

print('Threshold \t Accuracy \t Precision\t Recall \t  F1 Score ')

for i in range(len(thresh)):
    prediction = a_simple_classifier(X_validacion, thresh[i], xgb_model)
    accuracy[i], precision[i], recall[i],
f1_score[i]=calculate_metrics(prediction, y_validacion, xgb_model)
    print(f'{thresh[i]: .2f}\t\t {accuracy[i]: .2f}\t\t {precision[i]:
.2f}\t\t {recall[i]: .2f}\t\t {f1_score[i]: .2f}')



plt.plot(thresh, accuracy, 'r')
plt.plot(thresh, precision, 'g')
plt.plot(thresh, recall,'b')
plt.plot(thresh, f1_score,'k')

plt.legend(['Accuracy', 'Precision','Recall', 'F1 Score' ])
plt.xlabel('Threshold')
plt.ylabel('Scores')
plt.title('Change of Evaluation Metrics according to Threshold')

plt.xticks([0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9])
plt.yticks([0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9])
plt.grid(True)

plt.show()
```

A.9 Mejor modelo Random Forest

```
y_pred_test = rf_model.predict(X_test)
print("Reporte de clasificación del conjunto de test de Random Forest 1 (No
se usa para ajustar/tunning parámetros):")
print(classification_report(y_test, y_pred_test))



def a_simple_classifier(x, thres = 0.5, model=None):
    predicted = pd.DataFrame(model.predict_proba(X_validacion))[1]
    for i in range(len(x)):
        if predicted[i] > thres:
            predicted[i] = 1
        else:
            predicted[i] = 0
    return predicted

def calculate_metrics(predicted, actual, model=None):
    #print(predicted.sum())
    #print(actual.sum())
    TP, FP, TN, FN = 0, 0, 0, 0
```

```
    for i in range(len(predicted)):
        if   (predicted[i] == 1) & (actual[i] == 1):
            TP += 1
        elif (predicted[i] == 1) & (actual[i] == 0):
            FP += 1
        elif (predicted[i] == 0) & (actual[i] == 0):
            TN += 1
        else:
            FN += 1

    accuracy  = (TP + TN) / (TP + FP + TN + FN)
    precision = (TP) / ((TP + FP) if (TP + FP) >0 else 1 )
    recall    = (TP) / ((TP + FN) if (TP + FN) >0 else 1 )
    f1_score  = (2 * precision * recall) / ((precision + recall) if
(precision + recall) >0 else 1 )

    return accuracy, precision, recall, f1_score



thresh = np.linspace(0,1,100)
accuracy = np.zeros(len(thresh))
precision = np.zeros(len(thresh))
recall = np.zeros(len(thresh))
f1_score = np.zeros(len(thresh))

print('Threshold \t Accuracy \t Precision\t Recall \t  F1 Score ')

for i in range(len(thresh)):
    prediction = a_simple_classifier(X_validacion, thresh[i], rf_model)
    accuracy[i], precision[i], recall[i],
f1_score[i]=calculate_metrics(prediction, y_validacion, rf_model)
    print(f'{thresh[i]: .2f}\t\t {accuracy[i]: .2f}\t\t {precision[i]:
.2f}\t\t {recall[i]: .2f}\t\t {f1_score[i]: .2f}')



plt.plot(thresh, accuracy, 'r')
plt.plot(thresh, precision, 'g')
plt.plot(thresh, recall,'b')
plt.plot(thresh, f1_score,'k')

plt.legend(['Accuracy', 'Precision','Recall', 'F1 Score' ])
plt.xlabel('Threshold')
plt.ylabel('Scores')
plt.title('Change of Evaluation Metrics according to Threshold')

plt.xticks([0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9])
plt.yticks([0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9])
plt.grid(True)

plt.show()
```

Índice de Acrónimos

ARC: Centro de Investigación Ames de la NASA (Ames Research Center), colaborador de la misión Kepler.

AUC: Área bajo la curva (Area Under the Curve), una métrica para evaluar el rendimiento de un modelo de clasificación.

BATMAN: Paquete de Python utilizado para el modelado preciso de tránsitos exoplanetarios.

BIC: Criterio de información bayesiano (Bayesian Information Criterion), utilizado para la selección de modelos estadísticos.

BLS: Algoritmo de búsqueda de mínimos cuadrados (Box-fitting Least Squares), utilizado para la detección de tránsitos planetarios.

CCD: Dispositivo de carga acoplada (Charge-Coupled Device), utilizado en la detección de luz en telescopios.

CNNs: Redes neuronales convolucionales (Convolutional Neural Networks), utilizadas en el procesamiento y análisis de imágenes entre otros campos.

CTL: Lista de candidatos de TESS (TESS Candidate Target List), catálogo de posibles exoplanetas observados por TESS.

DFT: Transformada de Fourier discreta (Discrete Fourier Transform), técnica matemática para convertir señales de dominio del tiempo a dominio de la frecuencia.

ESA: Agencia Espacial Europea (European Space Agency).

FGK: Tipo de estrella (espectral F, G, K), que abarca desde estrellas blancas hasta amarillas y anaranjadas.

FOV: Campo de visión (Field of View), el área observable en el cielo por un telescopio.

FPCA: Análisis de componentes principales funcionales (Functional Principal Component Analysis), técnica para reducir la dimensionalidad de datos funcionales.

JPL: Laboratorio de Propulsión a Chorro (Jet Propulsion Laboratory), colaborador de la misión Kepler.

KIC: Catálogo de entrada Kepler (Kepler Input Catalog), que incluye estrellas observadas por la misión Kepler antes de ser clasificadas como KOI.

KOI: Objetos de interés de Kepler (Kepler Object of Interest), candidatos a exoplanetas identificados por la misión Kepler.

L1: Regularización L1, técnica utilizada para penalizar los coeficientes de los modelos con el fin de reducir el sobreajuste y simplificar el modelo.

L2: Regularización L2, similar a L1 pero con una penalización diferente, tiende a reducir el impacto de los coeficientes altos sin eliminarlos completamente.

LightGBM: Herramienta de aprendizaje automático (Light Gradient Boosting Machine).

LSTM: Redes neuronales recurrentes de memoria a largo plazo (Long Short-Term Memory).

MAST: Archivo de datos del telescopio espacial (Mikulski Archive for Space Telescopes).

NaN: Valor nulo (Not a Number).

NASA: Administración Nacional de Aeronáutica y del Espacio (National Aeronautics and Space Administration).

PDCSAP: Flujo de pre-búsqueda de acondicionamiento de datos (Pre-search Data Conditioning SAP).

PDCSAP_FLUX: Valor de flujo PDCSAP.

SAP: Flujo de fotometría de apertura simple (Simple Aperture Photometry).

SAP_FLUX: Valor de flujo SAP.

SDE: Valor que evalúa la eficiencia de la detección de señales en curvas de luz astronómicas (Signal Detection Efficiency).

SNR: Relación señal-ruido (Signal-to-Noise Ratio), medida de la calidad de una señal.

SVM: Máquinas de vectores de soporte (Support Vector Machines), algoritmo de clasificación supervisada.

TAP search: Protocolo de Acceso a Tablas (Table Access Protocol) utilizado para buscar y acceder a datos tabulares en archivos astronómicos. Facilita la consulta y recuperación de datos almacenados en bases de datos distribuidas.

TCEs: Eventos de cruce de umbral (Threshold Crossing Events), candidatos a exoplanetas detectados por misiones como Kepler y TESS.

TIC: Catálogo de entrada de TESS (TESS Input Catalog).

TLCM: Algoritmo de simulación de tránsitos (Transit Light Curve Modeller).

TLS: Mínimos cuadrados de tránsito (Transit Least Squares), método utilizado para la detección de exoplanetas en curvas de luz.

TOI: Objetos de interés de TESS (TESS Objects of Interest), candidatos a exoplanetas observados por TESS.

TPE: Método de optimización de hiperparámetros que utiliza modelos probabilísticos (Tree-structured Parzen Estimator).

TSFRESH: Herramienta de extracción de características para series temporales basada en pruebas de hipótesis escalables (Time Series FeatuRe Extraction on basis of Scalable Hypothesis tests).