FACULTAD DE ESTUDIOS ESTADÍSTICOS

MÁSTER EN MINERÍA DE DATOS E INTELIGENCIA DE NEGOCIOS

Curso 2022/2023

Trabajo de Fin de Máster

TÍTULO: Aplicación de técnicas de* Deep Learning *para la búsqueda e identificación de exoplanetas a través del método del tránsito astronómico

***Alumno*: Iván González Martín**

***Tutor*: Javier Álvarez Liébana**

Julio de 2023

UNIVERSIDAD COMPLUTENSE
MADRID

Resumen

Un exoplaneta, también conocido como planeta extrasolar, es un planeta que orbita una estrella más allá de nuestro sistema solar. La existencia de estos planetas se conjeturó durante siglos pero no fue hasta la década de los 90 cuando se confirmó la existencia del primero. Desde entonces, el interés de la comunidad científica por encontrar nuevos planetas dentro de la zona de habitabilidad estelar no ha dejado de aumentar, propiciando el desarrollo de técnicas que permiten detectar con mayores garantías la presencia de estos objetos astronómicos situados a años luz de la Tierra.

Este trabajo se centra en el estudio y desarrollo de diferentes herramientas de *Machine Learning* con el propósito de identificar y clasificar planetas extrasolares. Para ello se hace uso de información fotométrica y espectroscópica, datos relativos a los niveles de radiación electromagnética que emanan de la estrella orbitada por el exoplaneta una vez detectados por los filtros paso banda de los satélites artificiales. A este método se le conoce como «técnica del tránsito astronómico», y permite detectar disminuciones en la luminosidad de una estrella causadas por el posible tránsito de un planeta frente a ella, siempre desde la perspectiva del satélite artificial que la esté monitorizando. El resultado del registro continuado de la luminosidad de una estrella en el tiempo recibe el nombre de «curva de luz».

En este trabajo, los datos empleados para la construcción de los modelos son los recopilados por el telescopio estelar Kepler durante sus cuatro años de actividad (entre 2009 y 2013), con un total de 15 737 curvas de luz de más de 10 000 estrellas. Como se podrá comprobar, las curvas de luz en su estado original no son aptas para su aplicación directa a modelos de aprendizaje automático, por lo que requieren de un exhaustivo preprocesado a fin de discriminar las fluctuaciones naturales de la radiación estelar de aquellas provocadas por el tránsito del posible exoplaneta.

Palabras clave: Detección e Identificación de Exoplanetas, *Deep Learning*, Técnica del Tránsito Astronómico, Análisis de Curvas de Luz, Inteligencia Artificial en Astronomía

Abstract

An exoplanet, or extrasolar planet, is a planet that orbits a star beyond our solar system. The existence of these planets was hypothesized for centuries, but it wasn't until the 1990s that the first one was confirmed. Since then, interest in discovering new planets in the habitable zone has grown within the scientific community. This has spurred the development of techniques for more reliable detection of these distant astronomical bodies.

This study focuses on the exploration and development of various machine learning tools for the purpose of identifying and classifying extrasolar planets. For this purpose, we use photometric and spectroscopic information, which is data related to the electromagnetic radiation levels emanating from the star orbited by the exoplanet, once detected by the passband filters of artificial satellites. This method, known as the "astronomical transit technique", enables the detection of decreases in a star's brightness caused by a potential planet passing in front of it, as seen from the perspective of the monitoring artificial satellite. The continuous record of a star's brightness over time is referred to as a "light curve".

In this study, the data used to build the models were collected by the Kepler space telescope during its four years of operation (from 2009 to 2013), with a total of 15,737 light curves from over 10,000 stars. As will be demonstrated, the raw light curves are not suitable for direct application to machine learning models and require extensive preprocessing to distinguish natural stellar radiation fluctuations from those caused by the transit of a potential exoplanet.

Keywords: Exoplanet Detection and Identification, *Deep Learning*, Astronomical Transit Technique, Light Curve Analysis, Artificial Intelligence in Astronomy

Índice general

Índice de figuras

Índice de tablas

1. Introducción

Durante las últimas décadas, la investigación en la detección e identificación de exoplanetas se ha ido posicionando como una de las ramas más prolíficas dentro del campo de la astrofísica, atrayendo la atención tanto del público general como de la comunidad científica especializada. Este incipiente campo de estudio se originó en 1989, cuando se descubrió por primera vez un objeto astronómico con características similares a las de los planetas del Sistema Solar que orbitaba la estrella HD 114762 en la constelación de Coma Berenices, a unos 40 años luz de distancia de la Tierra [1]. Este objeto, con denominación HD 114762 b, fue uno de los primeros cuerpos astronómicos clasificados como planeta extrasolar y, a pesar de las controversias actuales en cuanto a su masa[1], es al que debemos la definición actual de exoplaneta. En este sentido, la denominación de exoplaneta o de planeta extrasolar abarca cualquier cuerpo astronómico con características planetarias que orbite una estrella que no sea el Sol.

Unos años después, en 1995, el campo cobró plena relevancia con el hallazgo del primer planeta extrasolar confirmado, 51 Pegasi b, hoy día conocido como Dimidio [2]. Gracias a la evolución de los métodos de detección, el astrofísico estadounidense Geoffrey Marcy pudo identificar el exoplaneta a través de la técnica de las velocidades radiales –antecesora natural de la técnica del tránsito astronómico–, la cual permite detectar pequeñas variaciones en la velocidad de una estrella causadas por la influencia gravitacional del posible planeta que la esté orbitando [3].

A mayo de 2023, la NASA Exoplanet Archive [4] ha catalogado como exoplanetas confirmados a más de 5400 cuerpos astronómicos, de los cuales 4060 se descubrieron gracias a la técnica del tránsito astronómico. Del total de exoplanetas confirmados, 2778 han sido anunciados por la misión espacial Kepler, que fue lanzada en una órbita heliocéntrica alrededor de la Tierra en marzo

[1] En la actualidad existe un debate en curso sobre si HD 114762 b debe ser clasificado como exoplaneta o como enana marrón. Su masa mínima es alrededor de 11 veces la de Júpiter, que es aproximadamente el límite superior para la clasificación de un objeto astronómico como planeta. Sin embargo, como el ángulo de inclinación de la órbita del objeto es desconocido, es posible que su masa real sea mucho mayor, lo que lo colocaría en la categoría de enana marrón.

de 2009, y cesó su actividad en agosto de 2013 a causa de fallos en las ruedas de reacción del satélite [5]. Tras el éxito de la misión espacial Kepler, la NASA no tardó en convertir la disciplina de la detección de exoplanetas en un pilar fundamental de su programa Explorers[2], diseñando y desarrollando nuevas misiones espaciales con nuevos satélites artificiales mejor preparados.

A medida que las misiones espaciales iban recopilando cada vez más información acerca de las estrellas monitorizadas y de sus candidatos a exoplaneta, el enfoque de la comunidad científica fue virando hacia el estudio generalizado de la información proveída por las misiones, en un intento por construir un gran catálogo en el que cada registro de cada estrella estuviera etiquetado de acuerdo a una serie de parámetros estandarizados. Esta es la línea que han seguido trabajos como el proyecto *Autovetter* de McCauliff et al. (2018), a través del cual se emplean modelos no supervisados para la validación de las curvas; o los estudios de Thompson et. al. (2018) y de Armstrong et al. (2016), en los que se emplean técnicas de *clustering* para agrupar curvas de datos con características y formas similares. Estas fueron las primeras aproximaciones del aprendizaje no supervisado, y del aprendizaje automático en general, al campo de la detección e identificación de exoplanetas, investigaciones que se siguen extrapolando a nuevas misiones estelares más actuales que recopilan información sobre nuevas estrellas nunca analizadas.

Gracias a estas investigaciones y a la labor de la NASA por certificar la veracidad de las clasificaciones, en el año 2016 la NASA Exoplanet Archive [6] publicó una lista etiquetada de todos los posibles tránsitos rescatados de los registros de la misión Kepler. Como tal, las listas preliminares producidas por el equipo de la misión se ensamblaron de manera heterogénea, en un esfuerzo por incluir tantos candidatos a planeta como fuera posible [7]. Las primeras revisiones de las listas de la misión se produjeron a partir de registros de los «Eventos de Cruce de Umbral» (*Threshold-Crossing Event*, TCE), disminuciones periódicas en la intensidad lumínica de la estrella analizada que pudieran ser consistentes con el tránsito de un exoplaneta [8]. Estas revisiones fueron depuradas manualmente por astrofísicos especializados para eliminar falsos positivos causados por alteraciones naturales en la intensidad lumínica de las estrellas o por errores en el propio sensor del satélite.

Con estas listas a disposición de la comunidad científica comenzaron a surgir nuevos artículos en el contexto del aprendizaje supervisado, permitiendo incrementar la eficiencia en la detección de TCE y facilitando la identificación de patrones más complejos en las curvas de luz estelares. En la actualidad, los modelos de arquitectura de red neuronal están a la vanguardia en este campo, demostrando ser herramientas extraordinariamente efectivas para el análisis de patrones más suti-

[2] El programa Explorers de la NASA es una serie continua de misiones espaciales que se centran en la ciencia y la investigación astronómica. Desde su inicio en 1958, el programa ha lanzado más de 90 misiones estelares de toda clase.

les presentes en las curvas. Las redes neuronales convolucionales (CNN), tanto unidimensionales (CNN 1D) como bidimensionales (CNN 2D), se han utilizado con éxito en numerosas ocasiones para detectar señales de tránsito de posibles candidatos a exoplaneta [9]. Cabe destacar que también se han aplicado toda clase de modelos clásicos con resultados bastante competitivos, aunque no superiores a los modelos de arquitectura neuronal [10][11].

En este sentido, el propósito de este trabajo es implementar algunas de las técnicas mencionadas sobre las curvas de luz originales de la misión Kepler, aplicando previamente sobre ellas un *pipeline* determinado. Este preprocesado preliminar pretende servir de arquetipo para los resultados de cualquier otra misión estelar que emplee la técnica del tránsito astronómico para la detección de exoplanetas, siendo el *Transiting Exoplanet Survey Satellite* (TESS) la más cercana en el tiempo.

1.1 Motivación

La búsqueda de nuevos planetas que se encuentren dentro de la zona de habitabilidad estelar y en los que pueda hallarse agua en cualquier estado es uno de los mayores retos, sino el mayor, al que se enfrenta la astrofísica contemporánea. Con cada nuevo descubrimiento, con cada misión estelar, nuestra comprensión del universo se va expandiendo progresivamente, a la par que se incrementan las posibilidades de encontrar planetas con características similares al nuestro y con nuevas formas de vida. Esta reflexión permanece latente en cualquier artículo o trabajo de este campo, y no es diferente en el caso del presente trabajo.

Con el preprocesado personalizado de los datos y con las técnicas empleadas, se espera poder alcanzar resultados competitivos dentro del estado del arte. Además, también se espera poder extrapolar todo el proceso –desde el preprocesado de los datos hasta la aplicación del modelo– a nuevas fuentes de datos de próximas misiones estelares. A día de hoy, solo es posible entrenar los modelos con datos de las misiones Kepler o K2, pues son los únicos etiquetados con el aval de la NASA. Se espera que los registros de otras misiones estelares actualmente en curso, tales como Hubble, Webb, TESS o Roman, puedan servir como datos de entrada al *pipeline* generado en este trabajo y aplicarse al modelo que mejores resultados haya obtenido.

1.2 Objetivos

El objetivo principal de este Trabajo de Fin de Máster (TFM) es generar un proceso secuencial completo con los datos recopilados por el satélite artificial Kepler, que se adecúe a la metodología

SEMMA aprendida durante el Máster (ver más detalles en Sección 1.3). El objetivo del proceso es lograr clasificar los tránsitos astronómicos de las estrellas monitorizadas por Kepler para poder discernir si existe un posible candidato a exoplaneta orbitando la estrella analizada o no.

Los objetivos parciales que se plantean en este trabajo son los siguientes:

- Analizar los datos de la misión estelar Kepler y tratarlos en la fase de preprocesado para que se adapten a los formatos requeridos por los algoritmos seleccionados en la fase de modelización.
- Aplicar el Análisis de Componentes Principales Funcionales (ACPF) a las curvas de luz originales con el objeto de reducir la complejidad de los datos de entrada.
- Estudiar los métodos de *Deep Learning* más usuales en la literatura y adaptarlos a las necesidades del presente trabajo, tales como las redes neuronales convolucionales unidimensionales (CNN 1D) o las redes neuronales recurrentes (RNN).
- Estudiar métodos alternativos a los habituales con el objeto de comprobar si la complejidad de la arquitectura neuronal es conveniente. Se aplicarán también algoritmos tradicionales menos complejos, como el *K-Nearest Neighbors* (KNN) o el *Random Forest*.
- Comparar los resultados de todos los modelos construidos a fin de encontrar el que mejor se adapta a los datos originales. Se compararán los resultados en términos de exactitud con trabajos anteriores que hayan empleado otras metodologías.
- Comprobar la escalabilidad del proceso y de los resultados a otras misiones con sectores de observación estelares distintos a los asignados para Kepler.

1.3 Metodología

Como ya se ha comentado en anteriores apartados, el presente trabajo sigue la metodología SEMMA para la depuración de los datos y la creación de los modelos. El método SEMMA es un acrónimo que denota un proceso estándar de cinco pasos para cualquier proyecto relacionado con la ciencia de datos: *Sample* o Muestreo, *Explore* o Exploración, *Modify* o Modificación, *Model* o Modelización y *Assess* o Evaluación. A continuación se expone brevemente cómo se ha ido adecuando cada una de estas fases al proyecto actual.

En primer lugar, conviene recordar cómo los datos utilizados en el presente trabajo provienen de la misión estelar Kepler de la NASA y son accesibles de manera pública. Para la descarga de las curvas de luz se recurrió a la base de datos astronómicos del Mikulski Archive for Space Telescopes (MAST) [12], mientras que para la descarga del catálogo de estrellas etiquetadas (Kepler IDs) se optó por acceder a la base de datos de la propia NASA en lo referente a exoplanetas: la NASA Exoplanet Archive [13]. Ambos *datasets* se encuentran disponibles en [https://cutt.ly/1wy8BqFN]. Además, en la Figura 1.1 se puede visualizar una parte de la estructura del *dataset*.

	kepid	flux_1	flux_2	flux_3	flux_4	flux_5
1	11709124	-6.923974e-02	-2.563927e-02	-4.750805e-02	-6.341263e-02	-2.865565e-02
2	5471217	-5.287453e-02	-1.275703e-01	-1.959715e-01	-2.731851e-01	-3.113722e-01
3	5732026	6.615532e-02	1.677852e-01	2.435283e-01	2.435283e-01	3.144775e-01
4	10122538	1.685645e-01	1.815356e-01	9.091724e-02	9.091724e-02	-3.203921e-02
5	4947556	-3.751596e-02	-3.389208e-01	-3.288634e-01	-2.830460e-01	0.000000e+00
6	5687700	-7.819831e-02	-1.567094e-01	-4.469815e-01	-3.515796e-01	-2.527369e-01
7	10338279	5.146763e-02	9.006836e-02	4.825091e-03	1.857660e-01	7.478891e-02
8	8244190	2.933563e-02	0.000000e+00	7.707794e-02	5.752085e-02	2.185792e-02
9	11968439	-8.782435e-02	7.185629e-02	5.988024e-02	5.988024e-02	5.269461e-01
10	6781548	-1.766980e-02	-2.135100e-01	-1.731088e-01	1.932634e-02	-1.185349e-01
11	5436013	0.000000e+00	-1.966690e-01	-1.348391e-01	4.061145e-02	-2.418435e-02
12	4254741	3.521573e-01	5.010828e-01	5.837082e-01	6.893220e-01	5.990338e-01
13	10616679	7.585199e-03	1.510593e-02	1.587950e-02	8.186858e-03	-6.682711e-03
14	4946627	6.047382e-01	4.912718e-01	4.788030e-01	4.638404e-01	3.965087e-01
15	4664743	-7.311072e-01	-7.803164e-01	-8.066784e-01	-7.715290e-01	-6.080844e-01

Figura 1.1: Formato de uno de los *datasets* empleado durante la elaboración del trabajo. Mientras que cada fila se corresponde con una curva de luz, cada columna es una medición en el tiempo de la intensidad lumínica de cada estrella.

Una vez construido el *dataset* completo, comenzaremos con las fases de muestreo y exploración del método SEMMA. En nuestro caso no se realizará muestreo, únicamente probaremos a reducir la dimensionalidad del *dataset* a través de las técnicas del Análisis de Componentes Principales Funcionales (ACPF). En la Sección 5 ya se explicará de manera más pormenorizada, pero las particiones serán 70 % para la fase de entrenamiento y 30 % (15-15 %) para las fases de validación y test. En lo referente a los métodos de validación y por motivos de capacidad de cómputo, se optará por validación cruzada en el caso de los modelos de arquitectura neuronal, y validación cruzada repetida para los modelos tradicionales.

La fase de modificación se correspondería en este caso con el preprocesamiento del *dataset* inicial. A modo de resumen, el preprocesado ha implicado normalizar y ajustar temporalmente las curvas de luz, eliminar sus valores *outliers*, eliminar su tendencia natural a través de *splines* polinomiales e interpolar el resto de información, aplicarles un *binneado* o agrupamiento para eliminar puntos

de datos que se encuentren demasiado cercanos entre sí, y uniformizar sus longitudes para que se puedan adaptar como *inputs* de entrada al formato de los subsiguientes modelos. En la Sección 4 se ahondará con mayor profundidad en cada una de las fases enumeradas.

En cuanto a la fase de modelización, en este TFM se ha optado por un enfoque de clasificación binaria –el habitual en la literatura especializada– en el que se clasifican las curvas de luz en dos categorías: «con exoplaneta» y «sin exoplaneta». En esta fase se propondrán diferentes modelos con distintos *inputs* de entrada. Por un lado trabajaremos con la versión original de las curvas de luz tras su preprocesado, un enfoque que requiere mayor tiempo, complejidad y capacidad de procesamiento. Por otro lado aplicaremos la técnica del Análisis de Componentes Principales Funcionales (ACPF) a las curvas de luz originales –entendidas como elementos aleatorios de un espacio de funciones regulares (en concreto, un espacio de Hilbert)–, con tal de reducir las dimensiones del *dataset* y la complejidad de los modelos. Para ambos enfoques se aplican, por un lado, modelos más tradicionales –en concreto, Regresión Logística, KNN, *Gradient Boosting*, *Random Forest* y varias modalidades de *Support Vector Machines* (SVM)–; y, por otro, los modelos más habituales en el estado del arte debido a su eficacia –principalmente, las redes neuronales convolucionales unidimensionales (CNN 1D), las redes neuronales *fully connected*, y las redes neuronales recurrentes (RNN)–. Se ha optado también por construir combinaciones de varias redes con distintas capas y convoluciones a fin de encontrar el modelo más óptimo en cuanto a *performance*.

Por último, en la fase de evaluación de los modelos seleccionados, todos los modelos se evaluaron utilizando el mismo conjunto de datos, por lo que es posible la comparación directa de su rendimiento. Además, en la fase de evaluación también se comparan los resultados con los trabajos más relevantes de la literatura especializada, a fin de comprobar cuán competitivos son los mejores modelos construidos durante el trabajo.

En cuanto al *software* manejado durante la elaboración del trabajo, se ha utilizado principalmente el lenguaje Python con el entorno Anaconda a través del editor de código Visual Studio Code. Las librerías más utilizadas han sido `lightkurve`, `astropy`, `wotan` y `pandas` para la fase de preprocesado, y `keras`, `tensorflow` y `scikit-learn` para la creación de los modelos. También se ha empleado el lenguaje R para la elaboración del ACPF (paquetes `goffda` y `fda.usc`), y para la visualización de los datos (entorno `tidyverse` y, específicamente, `ggplot2`). Todo el código requerido para la elaboración de este TFM se puede consultar de manera pública en el siguiente repositorio de GitHub: [https://github.com/Leztin/TFM].

2. Estado del arte

En esta sección se expone el estado del arte en lo referente a la detección e identificación de exoplanetas a través de la técnica del tránsito astronómico. El proceso para la construcción de un modelo de *Machine Learning* en este campo pasa habitualmente por tres etapas bien diferenciadas: (1) la obtención y el preprocesado de las curvas de luz a analizar; (2) la detección de los tránsitos o TCE en los registros de las curvas de luz; y (3) la identificación de los tránsitos detectados en función de si se corresponden o no con el paso de un exoplaneta frente a la estrella a la que pertenece la curva.

En los siguientes apartados se ahonda en profundidad en algunos de los trabajos y métodos más representativos de cada una de estas etapas, comenzando por las técnicas de detección. Los siguientes párrafos discuten los procedimientos que se han ido empleando a lo largo del tiempo para la detección de exoplanetas, más allá de la técnica del tránsito astronómico que es la que se utiliza en este trabajo.

2.1 Métodos de detección de exoplanetas

Como consecuencia de la pequeña escala y de la gran distancia a la que se encuentran estos planetas resulta complicada su observación directa incluso con los mejores telescopios astronómicos. En vez de ello, los astrónomos deben emplear métodos indirectos en los que, a través de diferentes parámetros proveídos por la estrella orbitada, se puedan detectar indicios que apunten a la presencia orbital de algún objeto astronómico con características planetarias.

Concretamente, los métodos de detección indirecta más utilizados desde 1995 –fecha en la que se descubrió el primer exoplaneta confirmado– a esta parte son los siguientes: (1) método de las variaciones temporales (*timing*); (2) método de las microlentes gravitacionales (*microlensing*); (3) método de la detección por imagen directa (*imaging*); (4) método de la velocidad radial o espectroscopia Doppler (*radial velocity*); y (5) método del tránsito astronómico (*transit photometry*).

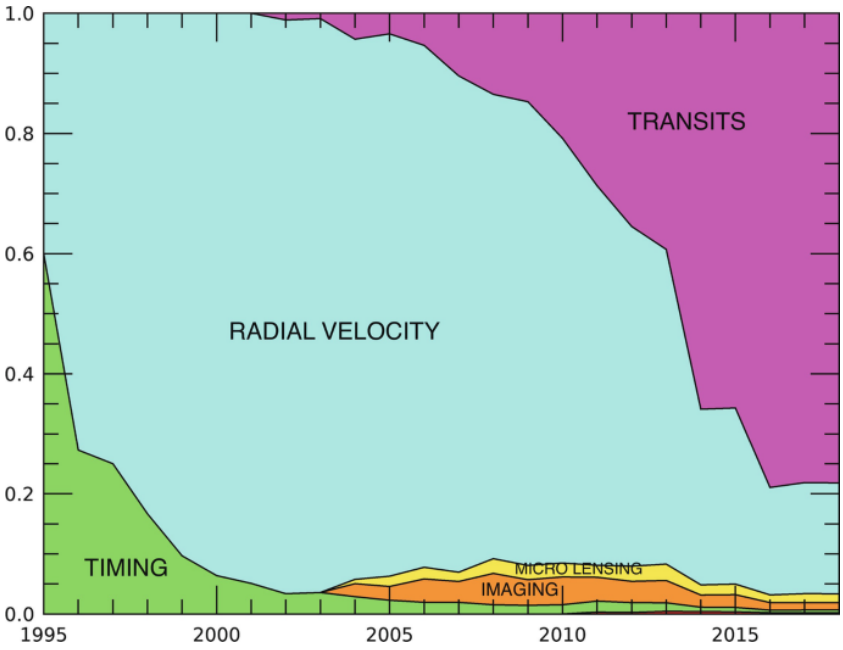


Figura 2.1: Representación de los distintos métodos de detección en función del porcentaje de su uso en el tiempo [14].

En la Figura 2.1 se puede observar la evolución en el uso de los métodos de detección de exoplanetas en artículos publicados desde 1995 hasta 2018. Como se puede observar, la técnica del tránsito astronómico, empleada en este trabajo, es la más utilizada en la actualidad debido a su marcada eficacia. A continuación comentaremos brevemente el funcionamiento de cada una de ellas.

Método de las variaciones temporales (*timing*)

El método de las variaciones temporales o *timing* consiste en observar cambios minuciosos y sistemáticos en los momentos en los que se producen determinados eventos astronómicos en el área de influencia gravitacional de la estrella analizada [15]. Estos cambios pueden indicar la presencia de objetos gravitacionalmente vinculados al astro, como un exoplaneta.

Existen distintas variantes de este método, pero la más utilizada durante la década de los 90 fue el *timing* de estrellas pulsantes. Los pulsares son estrellas de neutrones que emiten haces de radiación que pueden ser detectados desde la Tierra. Algunos pulsares rotan con una precisión increíblemente alta, lo que permite a los astrónomos utilizarlos como relojes cósmicos muy rigurosos. Si un púlsar tiene un planeta orbitándolo, su presencia causará pequeñas variaciones en los tiempos de llegada de los pulsos, que pueden ser analizados para inferir la existencia del propio planeta [15].

Debido a la complejidad que suponía discernir si el objeto astronómico que causaba la variación

temporal era o no un exoplaneta, con la llegada de los 2000 fue sustituido rápidamente por otros métodos superiores, como es el caso del método de la velocidad radial.

Método de las microlentes gravitacionales (*microlensing*)

El método de las microlentes gravitacionales es una técnica basada en los principios de la teoría de la relatividad general de Albert Einstein, que postula que la trayectoria de la luz se curva en presencia de un campo gravitacional. En el contexto de la detección de exoplanetas, esto se manifiesta como un aumento temporal y simétrico en el brillo de una estrella lejana cuando otro objeto astronómico en primer plano –la denominada «lente»– pasa directamente frente a ella desde nuestro lugar de observación en la Tierra [16].

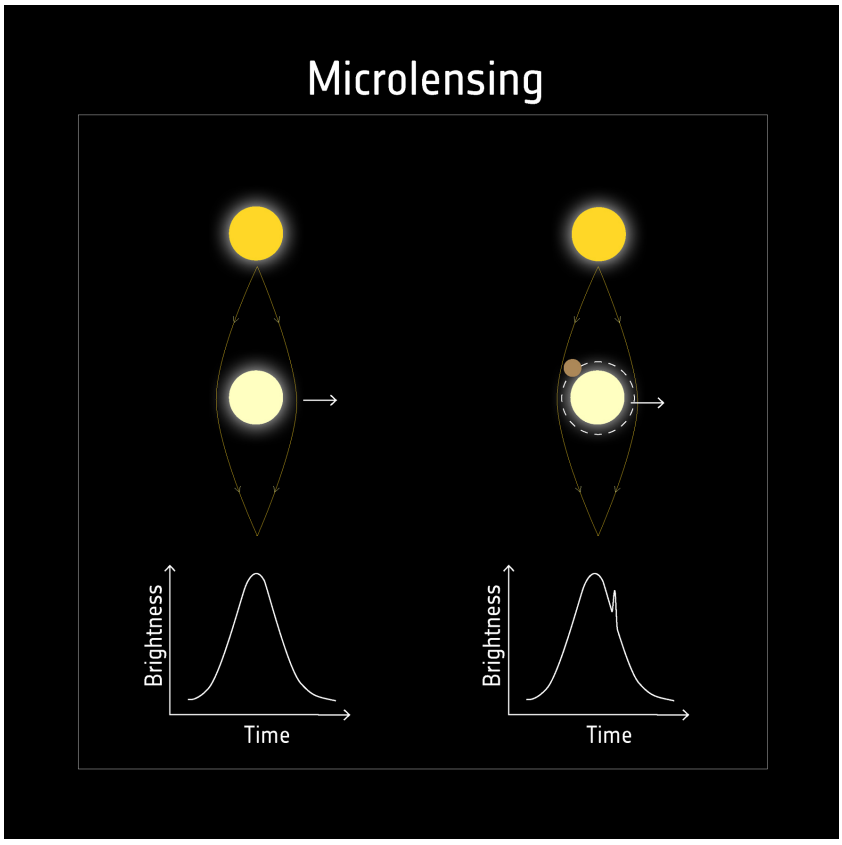


Figura 2.2: Representación visual de la técnica de las microlentes gravitacionales. En la curva de luz de la estrella de la derecha se advierte una anomalía producida por el efecto gravitacional al discernirse un planeta orbitándola [17].

En el caso de las microlentes gravitacionales, el objeto «lente» debe ser relativamente pequeño, como otra estrella o un planeta orbitante. En el caso de que un exoplaneta esté orbitando la estrella «lente», el efecto genera automáticamente una anomalía en el patrón de luz esperado. Estas

anomalías, que se manifiestan como desviaciones en la curva de luz de la estrella «lente», pueden ser analizadas por astrofísicos especializados para inferir la presencia del exoplaneta. La Figura 2.2 recoge visualmente esta explicación.

Al igual que en el anterior apartado, este método se ha visto ampliamente superado por técnicas posteriores, y en la actualidad prácticamente se ha abandonado. Los eventos en los que se produce el efecto de la microlente gravitacional son aleatorios y no repetibles dado que requieren de una alineación precisa entre la estrella de fondo, el objeto «lente» y el observador [16]. Como consecuencia, la detección de exoplanetas mediante este método requeriría de la monitorización continua de grandes áreas del cielo a fin de poder detectar con claridad el evento astronómico.

Método de la imagen directa (*imaging*)

El método de la imagen directa, o *imaging*, es la única técnica de observación directa para la detección de exoplanetas, pues implica capturar directamente una imagen del planeta en el cielo.

El desafío del *imaging* es que las estrellas son mucho más brillantes que los posibles planetas que las orbitan, lo que hace que sea muy complicado el localizarlos debido al deslumbramiento de la propia estrella. Aunque existen técnicas complementarias que permiten la visualización del planeta ignorando el brillo de su estrella, como la ocultación coronográfica[1] o el uso de ópticas adaptativas, el planeta debe de encontrarse lo suficientemente cercano al punto de observación como para poder ser captado de manera directa por los grandes telescopios astronómicos [15]. A mayo de 2023, tan solo han podido descubrirse 67 planetas extrasolares mediante este método [13].

Método de la velocidad radial (*radial velocity*)

El método de la velocidad radial, también conocido como espectroscopia Doppler o método Doppler, es una técnica indirecta que implica detectar los efectos que provocan los planetas en el movimiento de la estrella que orbitan [18].

Cuando un planeta orbita alrededor de una estrella, no sólo se mueve el planeta, sino que la estrella también realiza un pequeño movimiento en respuesta a la gravedad del objeto que la orbita. Este ligero movimiento de la estrella puede ser detectado desde la Tierra mediante la observación de cambios en el espectro lumínico de su curva de luz. La luz emitida por cualquier estrella se

[1] La ocultación coronográfica es una técnica utilizada en astronomía para observar objetos con señales muy débiles que están próximos a estrellas muy brillantes. Esta técnica se basa en el uso de un dispositivo conocido como coronógrafo, el cual tiene la capacidad de bloquear la luz de una estrella para permitir la observación de los objetos que se encuentran cerca de ella.

desplaza hacia el rojo –esto es, hacia longitudes de onda más largas– cuando se aleja del punto de observación, y se desplaza hacia el azul –hacia longitudes de onda más cortas– cuando sucede lo contrario [18]. Este fenómeno se conoce en astronomía como efecto Doppler. Al observar estos desplazamientos hacia el rojo y hacia el azul en el espectro lumínico de una estrella, los astrónomos pueden determinar su velocidad y la dirección de su propio movimiento.

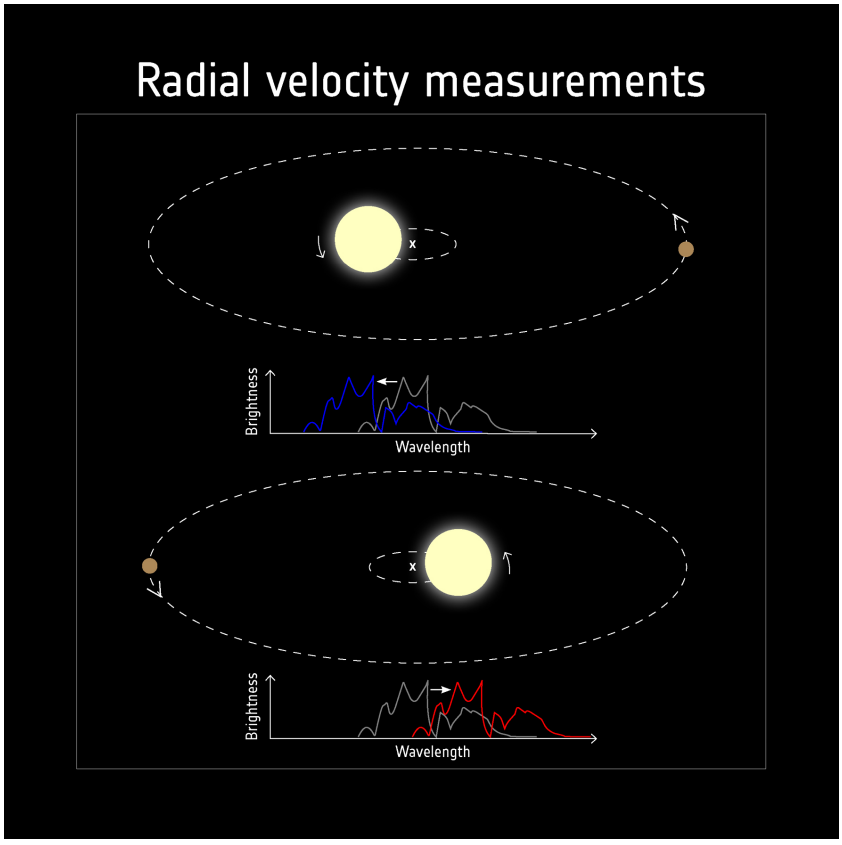


Figura 2.3: Representación visual de la técnica de la velocidad radial. Las oscilaciones de la estrella pueden advertirse en el patrón regular de desplazamientos registrados en su curva de luz [17].

Si una estrella muestra un patrón regular de desplazamientos hacia el rojo y hacia el azul, ello indica que está oscilando hacia adelante y hacia atrás en el espacio, lo cual suele ser una señal bastante evidente de que hay un planeta en órbita alrededor de la estrella [15]. A partir de estas oscilaciones, los astrónomos pueden estimar la masa del planeta y la distancia a la que orbita de su estrella. La Figura 2.3 recoge visualmente esta explicación.

Este método ha sido el más exitoso en las últimas décadas, solo superado por la técnica del tránsito astronómico. A día de hoy, gracias a las velocidades radiales se han conseguido detectar e identificar más de 1000 exoplanetas en las zonas más lejanas del espacio [13].

Método del tránsito astronómico (*transit photometry*)

En la actualidad, el método del tránsito astronómico es la técnica con la que más planetas extrasolares se han podido descubrir. Tanto es así que el telescopio espacial de la misión TESS actual está equipado con todo lo necesario para monitorizar grandes áreas del espacio únicamente a través de esta técnica.

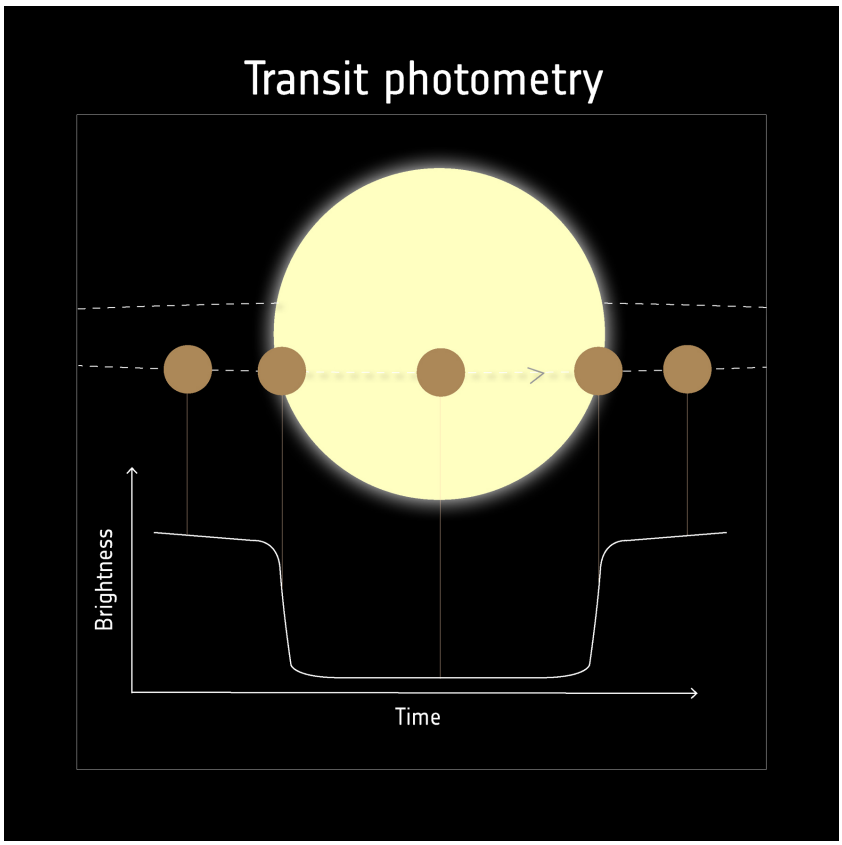


Figura 2.4: Representación visual de la técnica del tránsito astronómico. Cuando el exoplaneta eclipsa parcialmente el brillo de la estrella con su tránsito se produce una clara disminución en la intensidad lumínica habitual de la estrella, registrada en las mediciones de su curva de luz [17].

Este método se basa en observar la disminución en el brillo de una estrella cuando un exoplaneta pasa por delante de ella, es decir, cuando el exoplaneta «transita» la estrella desde la perspectiva del satélite artificial que la esté monitorizando. Durante el tránsito o TCE, el exoplaneta bloquea una pequeña fracción de la luz de la estrella, lo que resulta en una disminución medible en su brillo. Esta disminución en su intensidad lumínica es muy pequeña, a menudo menos del 1 %, pero puede ser detectada gracias a los filtros tan precisos equipados en los telescopios espaciales [14]. Al observar la curva de luz original de la estrella y sus tránsitos, los astrónomos pueden determinar el periodo orbital del exoplaneta, el tiempo que dura el tránsito, y su tamaño relativo en comparación con la estrella. La Figura 2.4 recoge visualmente esta explicación.

A través de este método se han conseguido descubrir y certificar 4106 exoplanetas de los 5438 que se encuentran catalogados en la actualidad [13]. Es por ello que lo emplearemos en este trabajo como método de clasificación. Los tránsitos o TCE aquí discutidos serán los elementos a clasificar que, en función de su forma, tamaño y profundidad, podrán corresponderse con el paso de un posible exoplaneta o con fluctuaciones naturales de la radiación de la estrella. En posteriores apartados ahondaremos en esta cuestión.

2.2 Obtención y preprocesado de las curvas de luz

Obtención de las curvas de luz

Son numerosas las misiones estelares que se han ido sucediendo a lo largo de los años con el objeto de recopilar curvas de luz de distintos sectores de observación estelares. Desde la misión Kepler y su continuación como Kepler K2, no han parado de surgir nuevas misiones por parte de la iniciativa pública y privada con telescopios estelares cada vez más refinados y adaptados a la técnica de detección del tránsito astronómico. A modo de ejemplo, por parte de la alianza colaborativa entre la NASA y SpaceX, destacan las misiones de los telescopios estelares Hubble, Roman, Webb y TESS; por parte de la Estación Espacial Europea, destacan las misiones Gaia, CHEOPS y PLATO, todas ellas destinadas a explorar grandes zonas del espacio a través de distintos métodos.

Aún con tal cantidad de misiones en curso, los actuales trabajos de detección de exoplanetas relacionados con la ciencia de datos y el *Machine Learning* emplean únicamente los resultados de la misión Kepler/Kepler K2, cuyos registros son los únicos etiquetados y en donde existe un mayor número de exoplanetas confirmados. Pese a la antigüedad de los datos y a que el satélite espacial hace años que fue sustituido, aún hoy día se siguen descubriendo nuevos TCE ocultos entre las más de 500 000 curvas de luz recopiladas entre ambas misiones durante sus respectivos años de actividad [13].

Tanto para Kepler como para Kepler K2, las curvas de luz se almacenaron en ficheros únicos con identificadores de la clase KeplerID, esto es, denominaciones auxiliares de las estrellas monitorizadas en cada misión. Para cada estrella los telescopios espaciales capturaban dos clases de curvas: las primeras, de cadencia corta, registraban una medición de la intensidad del brillo cada 58.8 segundos; las segundas, de cadencia larga, registraban la misma medición pero cada 29.4 minutos [4]. De esta manera, se conseguían dos curvas con distinta densidad de información para cada objeto

estelar analizado. En la práctica, dentro del campo de la ciencia de datos, la mayoría de trabajos emplean las curvas de cadencia larga. El reconocimiento de los TCE por parte de los algoritmos no requiere de curvas con tantos puntos de información, puesto que ello ralentizaría en exceso el proceso de detección.

Además de las curvas de larga y corta cadencia, cada curva de luz se comprimió dividida en cuatrimestres debido a la forma en la que se recolectaban los datos. La nave tenía que rotar aproximadamente 90 grados cada cuatro meses para reorientar sus paneles solares hacia el Sol y su antena de alta ganancia hacia la Tierra. De esta manera se procedía a la comunicación y a la descarga de los datos recopilados hasta el momento. Como consecuencia, la orientación del campo de estrellas almacenada en los detectores del satélite era diferente en cada reorientación. Ello provocaba que las estrellas cayeran en diferentes píxeles de los detectores en cada cuatrimestre, aunque en realidad siguieran siendo las mismas [4]. Este es el motivo por el cual, en la fase de preprocesamiento, se deben normalizar y estandarizar todos los cuatrimestres pertenecientes a una misma curva de luz, pues cada uno se encuentra en una escala diferente.

Como la misión Kepler estuvo en activo unos 5 años, para cada estrella se almacena información de unos 17 cuatrimestres. En el caso de las curvas de cadencia larga, se consiguen, por tanto, curvas de luz continuadas de unos 75 000 puntos de información –en torno a 4000 por cada cuatrimestre–.

En cuanto al formato y a la descarga de los archivos, todas las curvas son accesibles de manera pública a través de la base de datos astronómica del Mikulski Archive for Space Telescopes (MAST). El MAST sirve como repositorio para los datos recopilados por telescopios espaciales como Hubble, Kepler o TESS, y está disponible para científicos, investigadores y el público en general [12]. Por otra parte, cada curva de luz se comprime en archivos *Flexible Image Transport System* (FITS), un formato de archivo digital diseñado específicamente para almacenar y procesar datos astronómicos. Cada archivo almacena una gran cantidad de información sobre el objeto estelar analizado, en donde cada fila es una medición de su luminosidad en un instante temporal determinado.

Las dos variables que almacenan información sobre el flujo lumínico de la estrella son las variables «SAP Flux» (*Simple Aperture Photometry Flux*) y «PDC Flux» (*Pre-search Data Conditioning Flux*). Por una parte, la variable «SAP Flux» es la medida más básica de la intensidad de la luz de una estrella. Se almacena tal y como la registra el telescopio estelar, por lo que puede estar contaminada por distintos tipos de ruido, como la luz de fondo o la radiación lumínica de estrellas cercanas. Por otra parte, la variable «PDC Flux» responde al resultado de un *pipeline* elaborado por la NASA en el que se aplica un proceso de corrección a las medidas del «SAP Flux». En esencia se trata de un flujo corregido que ha sido procesado para minimizar el ruido y las fluctuaciones naturales que no están relacionadas con la variabilidad natural del brillo de la estrella en sí [19].

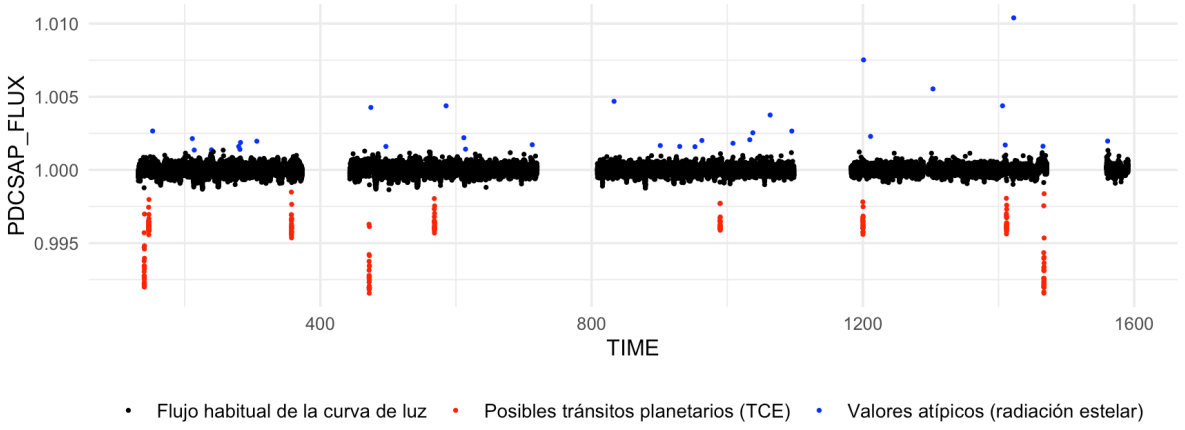


Figura 2.5: Curva de luz en formato «PDC Flux» de la estrella KIC 11442793.

La Figura 2.5 muestra a modo de ejemplo el registro lumínico en formato «PDC Flux» de la estrella KIC 11442793. Aparece marcado en color negro el registro lumínico habitual de la estrella, en color rojo los tránsitos astronómicos potencialmente consistentes con la presencia de un posible exoplaneta en órbita, y en color azul valores atípicos provocados por radiación estelar o por fallos en los sensores del propio satélite.

A pesar de que la gran mayoría de la literatura emplea las mediciones que ofrece el *pipeline* oficial de la NASA como el set de datos primigenio, las curvas todavía están lejos de poder ser introducidas como *inputs* de los modelos a aplicar. En la siguiente sección se exploran algunas de las etapas más comunes dentro de la fase de preprocesado de las curvas de luz, así como se detallan los procedimientos más habituales recogidos en los artículos más representativos de la literatura.

Preprocesado de las curvas de luz

Como ya se ha comentado en el anterior apartado, las señales sin procesar que se obtienen de los telescopios espaciales, incluso con el *pipeline* previo de la NASA, contienen ruido instrumental que interfiere y enmascara los posibles tránsitos o TCE de los exoplanetas. A continuación se exponen brevemente las fases más habituales en el preprocesado de este tipo de curvas.

Unión y centrado de las curvas de luz

Dado que las curvas de luz están divididas originalmente en 17 cuatrimestres como consecuencia de la recalibración de los sensores del satélite, cada periodo temporal se encuentra en una escala diferente. Esta cuestión se resuelve en la mayoría de trabajos dividiendo cada cuatrimestre por su mediana [10][20]. El objetivo es centrar y agrupar todos los cuatrimestres en un único eje temporal.

Eliminación de valores *outliers* por encima de la curva

Tras la unión de todos los cuatrimestres en una única curva, se suele proceder a eliminar sus valores *outliers*, aunque únicamente aquellos que superan en σ veces el valor medio de la propia curva. Estos valores por encima de la curva son atribuibles a rayos cósmicos, erupciones estelares que incrementan temporalmente el flujo lumínico, o, incluso a errores en el propio sensor del satélite. En este sentido, a la hora de identificar posibles tránsitos, lo único que es relevante a ojos del modelo son las disminuciones en la actividad lumínica de la estrella, disminuciones que puedan coincidir con el paso de un exoplaneta frente a los sensores del satélite. La mayoría de trabajos emplean el valor de 4σ para definir el límite superior de los valores *outliers* [21][22].

Eliminación de la tendencia de la curva de luz e interpolación de los TCE

En esta tercera fase se elimina la tendencia natural de las curvas de luz a fin de poder identificar más fácilmente sus descensos en el flujo luminoso. El método más habitual se conoce como filtrado Savitzky-Golay y es el que utilizan trabajos como el de Staudemeyer (2015). Este método suaviza la curva de luz ajustando subconjuntos de datos adyacentes a un polinomio de bajo orden (*spline*) por medio de un proceso de regresión lineal. Tras el suavizado se suelen volver a interpolar los TCE para que no queden aplanados como consecuencia del paso del *spline* polinomial.

Otros métodos menos populares pasan por descomponer la tendencia y la estacionalidad de las curvas a través de distintos algoritmos –como el algoritmo de suavizado LOESS–, o identificar y eliminar directamente los datos corruptos causados normalmente por partículas de alta energía o rayos cósmicos. Los puntos de datos afectados son reemplazados por valores interpolados o por la media de los puntos vecinos [23].

Creación de las versiones *phase-fold* de las curvas de luz

Tras la eliminación de la tendencia natural de las curvas de luz, se produce usualmente el *phase-fold* o plegado de fase de la curva de luz final. En concreto, en esta fase destaca el trabajo de Shallue y Vanderburg (2018), en el que aplican esta técnica sobre datos previamente clasificados.

El objetivo de la técnica del *phase-folding* es el de reforzar la visión de los tránsitos planetarios en una curva de luz. Una vez que se conoce el período orbital de un exoplaneta catalogado, se «pliega» la curva de luz de su estrella alrededor de ese período. Esto implica que se toma el tiempo de cada punto de información y se divide por el período orbital del planeta al que corresponde el TCE,

tomando sólo la fracción restante. Ello traduce el tiempo original en que estaba representada la curva de luz –usualmente en Días Julianos Baricéntricos– a una «fase temporal» que varía entre -1 y 1. Así, todos los puntos de datos que están en la misma fase del período se alinean. Con ello conseguimos centrar el TCE en el instante temporal 0 a fin de facilitar el reconocimiento de los tránsitos a los modelos posteriores [21].

Agrupamiento de los puntos de datos

Esta última fase es opcional y suele ser incluida para reducir la dimensionalidad del *dataset* final. Recordemos que cada curva de luz contiene unos 75 000 puntos de información en promedio, y los *datasets* finales pueden albergar más de 20 000 curvas. Ello implicaría manejar más de mil millones de puntos de información en total, cuestión impracticable si no se dispone de una capacidad de cómputo lo suficientemente elevada.

Con la técnica del agrupamiento o *binning* se pueden agrupar y promediar puntos de datos en intervalos de tiempo más grandes. Al agrupar ciertos conjuntos de datos sobre la curva de luz final, se busca reducir la cantidad de datos y el ruido, mejorando así la relación señal-ruido de características significativas en la curva de luz [21].

Es importante advertir que mientras el *binning* puede mejorar la relación señal-ruido y hacer que las características más repetidas durante el curso de la curva sean más fáciles de identificar, también implica una gran pérdida de información de alta resolución temporal. Ello se traduce en que las variaciones de alta frecuencia o los cambios rápidos en la intensidad de la luz pueden ser menos visibles o no detectables en la curva de luz tras el agrupamiento [22].

Aún con todo, en estos casos suele merecer la pena el sacrificar cierta cantidad de información si se consigue mejorar la visión del tránsito y se eliminan, con ello, puntos de datos no relevantes en pro de los modelos que se quieran aplicar posteriormente.

2.3 Métodos para la detección e identificación de tránsitos astronómicos (TCE)

A la hora de construir el set de datos final para comenzar con el proceso de entrenamiento, no solo serán necesarias las mediciones de la intensidad lumínica de cada estrella, sino también determinados datos sobre el propio tránsito o TCE –su periodicidad, su duración, etc.– así como su etiqueta de clasificación.

En el caso de la misión Kepler, todos los tránsitos astronómicos de todas las estrellas fueron identificados y etiquetados a través de distintos algoritmos de clasificación. En concreto, se empleó el KTSA (*Kepler Transit Search Algorithm*) en conjunto con el algoritmo BLS (*Box Least Squares*), que, en esencia, buscaban automáticamente eventos de tránsito en las curvas de luz al detectar caídas significativas y periódicas en el brillo de la estrella. Para aumentar la precisión en la clasificación, se requería que al menos tres tránsitos fueran observados con una periodicidad consistente antes de pasar a considerar el TCE como candidato a exoplaneta [4].

Este proyecto, denominado oficialmente como *pipeline Autovetter*, fue publicado por la NASA Exoplanet Archive en el año 2016 con todos los TCE detectados en las curvas de luz de Kepler por los algoritmos automáticos. El *dataset* final contenía 20 367 tránsitos o filas[2] con un gran número de variables referidas a las características del supuesto planeta, tales como su masa, su densidad, su periodo orbital, entre muchas otras. Además, también se incluía la etiqueta con la que fue clasificado cada TCE, cuestión indispensable para aplicar modelos supervisados a la clasificación de los tránsitos.

En concreto, la NASA clasificó los tránsitos astronómicos en función de cuatro etiquetas: *Planet Candidate* (PC), *Astrophysical False Positive* (AFP), *Non-Transiting Phenomenon* (NTP) y Unknown (UNK) [13].

Para que un tránsito sea clasificado como PC deben cumplirse ciertos criterios de calidad que confirmen que los descensos en la luminosidad de la estrella no se deban a otras causas astrofísicas. Por su parte, los AFP son TCE causados por fenómenos astrofísicos distintos al paso de un exoplaneta. Por ejemplo, puede ser el resultado del tránsito de una estrella binaria eclipsante, donde otra estrella pasa por delante de la analizada, o también a causa de manchas estelares que puedan imitar el efecto de un tránsito. Por último, los NTP son curvas de luz que no muestran características de tránsito consistentes porque presentan variaciones en la intensidad de la luz atribuibles a la variabilidad intrínseca de la estrella, como pulsaciones o rotación estelar [13].

Tras este repaso en lo referente al procesamiento de las curvas de luz y de la clasificación etiquetada de los TCE, en el siguiente apartado ahondaremos en cómo las herramientas del *Machine Learning* y del *Deep Learning* se sirven de ambas clases de información para crear nuevos modelos de clasificación.

[2] Conviene recordar que cada fila del *dataset* se trata de un tránsito astronómico, y no de una curva de luz de una estrella. Una curva de luz puede llegar a contener más de un tránsito astronómico, todo en función del número de planetas extrasolares que mantenga en órbita.

2.4 Aplicaciones actuales de *Machine Learning* a la detección de exoplanetas

En el año 2015 comenzaron a implementarse las primeras técnicas de aprendizaje supervisado a los incipientes conjuntos de datos que fue publicando la NASA en su repositorio de acceso libre. Los primeros trabajos empleaban modelos muy tradicionales dentro del campo de la ciencia de datos, y ni siquiera consideraban toda la información que se podía extraer de las curvas de luz. Únicamente trabajaban con características adyacentes a los propios tránsitos o TCE descubiertos hasta la fecha, como podían ser sus periodos orbitales, su duración en términos de tiempo, o su profundidad. Este es el caso de los modelos *K-Nearest Neighbors* (KNN) de los trabajos de Thompson et al. (2015), de los modelos *Random Forest* de McCauliff et al. (2015) en su proyecto *Autovetter*, o de los árboles de decisión clásicos de Coughlin et al. (2016) conocidos como *Robovetters*.

Tras esta primer generación de trabajos, un par de años después, hacia el año 2017, la disciplina fue comenzando a considerar el total de la información que había a disposición de la comunidad científica. La NASA Exoplanet Archive estaba siendo ampliada con nuevas curvas de luz pertenecientes a la extensión de la misión Kepler original –la K2–, que apuntaba a nuevos sectores de observación estelares. Durante este periodo de tiempo se comenzaron a publicar trabajos en los que se aplicaban principalmente modelos de la familia SVM directamente sobre las curvas de luz. Este es el caso del artículo de Pearson et al. (2018), en el que, a partir de distintas curvas de luz sin preprocesar, se implementan SVM lineales, polinomiales y radiales a fin de encontrar una configuración lo suficientemente óptima.

En estos últimos años, de 2019 a esta parte, todos los modelos mencionados anteriormente han sido superados ostensiblemente por otros métodos basados principalmente en las técnicas de *Deep Learning*. En esta sección, conviene hacer referencia a los trabajos de Shallue y Vanderburg (2019), probablemente los mayores exponentes de la aplicación de estas técnicas a la detección e identificación de exoplanetas.

Lo que proponen estos autores en su trabajo es la aplicación de modelos de arquitectura neuronal a la clasificación de TCE, modelos que usualmente se utilizaban para la clasificación de imágenes. En concreto emplean dos tipos de red: redes neuronales de arquitectura totalmente conectada (*fully connected*) y redes neuronales de arquitectura convolucional unidimensional (CNN 1D). Además realizan un procesado previo de las curvas de luz muy exhaustivo, en el que dividen los tránsitos astronómicos de los exoplanetas catalogados en dos versiones: una visión global con gran amplitud en sus extremos; y una segunda visión local, en la que el tránsito queda centrado y focalizado. Esta

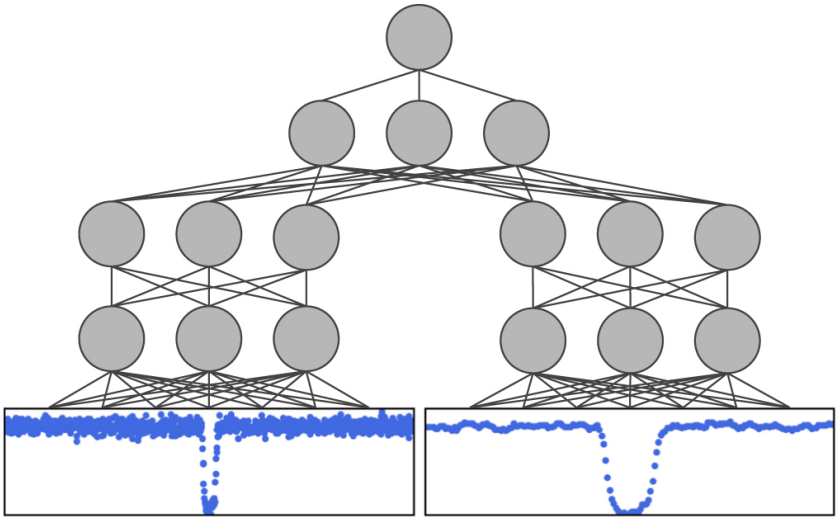


Figura 2.6: Representación de un modelo de red neuronal totalmente conectada (*fully connected*) con dos *inputs* de entrada: la visión global y local de un TCE.

visión dual puede apreciarse en los *inputs* de entrada del modelo representado en la Figura 2.6.

La Figura 2.6 también sirve como esquema para poder comprender el funcionamiento de las capas que conforman las redes neuronales totalmente conectadas del trabajo de Shallue y Vanderburg (2019). En el caso de estas redes, se emplean para su construcción fundamentalmente tres tipos de capas: la capa *Flatten*, las capas *fully connected* o *Dense* y las capas *Dropout*.[3]

- **Capa *Flatten***. La capa *Flatten* se emplea en este tipo de modelos para convertir la estructura original de los datos, que es multidimensional, en un vector unidimensional. Todas las redes neuronales creadas con el propósito de detectar exoplanetas en curvas de luz deben poder adaptarse a la estructura unidimensional de los datos, porque nuestro canal de entrada es únicamente el registro del brillo de un conjunto de estrellas.

- **Capas *fully connected* o *Dense***. Este tipo de capas están conectadas a todas las neuronas de la capa anterior y a todas las de la capa siguiente. En estas capas, las entradas se combinan mediante una suma ponderada para, posteriormente, aplicarles una función de activación no lineal (ReLU). Son conocidas como «totalmente conectadas» porque, a diferencia de las capas convolucionales, cada neurona no se restringe a tener conexiones locales, sino que mantiene conexiones con todas las características de la entrada. En el caso de las redes totalmente conectadas, este tipo de capas son las que conforman el grueso del modelo.

- **Capas *Dropout***. Por último, la capa *Dropout* es una capa que se emplea para prevenir el sobreajuste. En este sentido, evita que la red se vuelva demasiado dependiente de cualquier

[3] La exposición de las definiciones se ha extraído de Albawi, S., Mohammed, T.A., y Al-Zawi, S. *Understanding of a convolutional neural network* (ICET, 2017).

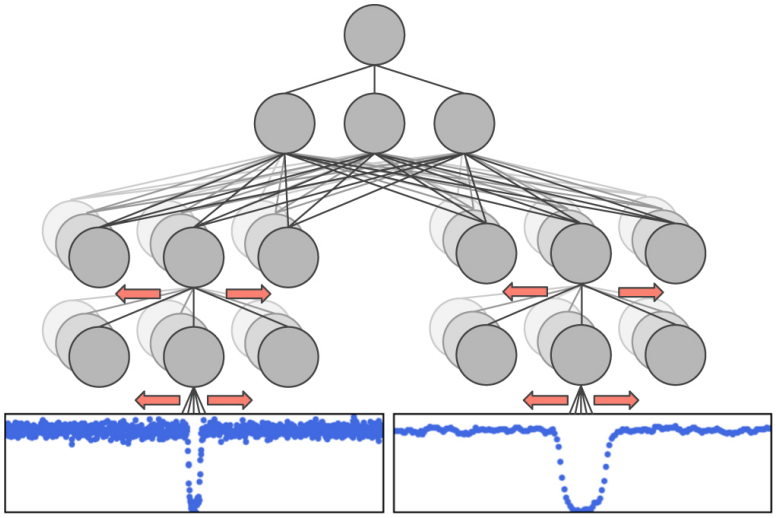


Figura 2.7: Representación de un modelo de red neuronal convolucional unidimensional con dos *inputs* de entrada: la visión global y local de un TCE.

neurona en particular y fomenta que la red aprenda características más robustas y generalizables del set de datos. Su funcionamiento es muy simple: durante la fase de entrenamiento, la capa «apaga» aleatoriamente un subconjunto de neuronas sobre el resto de capas en las que se ha aplicado, provocando que esas neuronas no contribuyan en esa iteración en concreto al proceso de aprendizaje.

Por otro lado, las redes neuronales convolucionales unidimensionales son muy similares en su construcción a las redes totalmente conectadas, ya que añaden únicamente dos nuevos tipos de capas: las capas convolucionales unidimensionales y las capas *Pooling*.[4] La Figura 2.7 recoge visualmente la forma de una red neuronal convolucional con dos *inputs* de entrada.

- **Capas convolucionales unidimensionales**. Las capas convolucionales unidimensionales (1D) son una variante de las capas convolucionales clásicas que se utilizan principalmente para analizar datos secuenciales, como series temporales. A diferencia de las capas convolucionales 2D, que se aplican sobre matrices bidimensionales –usualmente imágenes–, las capas convolucionales 1D operan sobre vectores unidimensionales. Estas capas utilizan filtros que se desplazan a lo largo de una sola dimensión de la entrada, y, en cada posición, realizan una operación de convolución sobre un segmento de la secuencia. Lo que se logra con ello es que el tamaño del vector original se vaya reduciendo mientras que el modelo va detectando características locales dentro de cada secuencia. En función de los hiperparámetros seleccionados –filtro, *kernel*, función de activación, etc.– se puede conseguir que la salida de la capa convolucional sea o no del mismo tamaño que la entrada.

[4]La exposición de las definiciones se ha extraído de Albawi, S., Mohammed, T.A., y Al-Zawi, S. *Understanding of a convolutional neural network* (ICET, 2017).

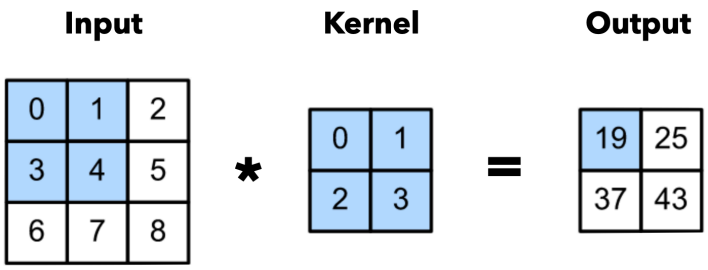


Figura 2.8: Diagrama de funcionamiento de una capa convolucional con una entrada bidimensional y un *kernel* de tamaño 2×2.

En la Figura 2.8 se muestra el funcionamiento de una capa convolucional muy simple: sobre un *input* de tamaño 3×3 se aplica una ventana de *kernel* de tamaño 2×2, consiguiendo resumir la información en un *output* del mismo tamaño que la matriz del *kernel*.

- **Capas *Pooling***. Este tipo de capas se introducen intercaladas en el modelo para tratar de reducir la dimensionalidad espacial de los mapas de características, lo que ayuda a disminuir la cantidad de parámetros y cálculos en la red. Esto se hace típicamente para reducir el posible sobreajuste y para mejorar la eficiencia computacional del modelo. Las capas de *Pooling* operan de manera similar a las capas convolucionales, pero en lugar de aprender de los filtros, realizan una operación de reducción en bloques de la entrada. Las dos formas más comunes de *Pooling* son el *Max Pooling*, que toma el valor máximo de un bloque de entrada; y el *Average Pooling*, que toma su promedio.

La salida de ambos tipos de red se produce a través de una última capa *Dense* con función de activación sigmoidal, de manera que el resultado sea una probabilidad entre 0 y 1 que indica si el TCE analizado se corresponde o no con el tránsito de un exoplaneta.

Para finalizar, otro trabajo al que conviene hacer referencia en esta sección es el elaborado por Hinners et al. (2018) [24], en el que se implementan redes neuronales de arquitectura recurrente (RNN) para la detección de exoplanetas en las curvas de luz de la misión K2. Mientras que las CNN tomaban la información de entrada como un todo para, a partir de ahí, extraer las características relevantes de las secuencias; las RNN tienen capacidad para «recordar» las características de datos ya observados y aplicar ese conocimiento a nuevos datos integrados en nuevas repeticiones. Para conseguirlo, los primeros módulos de la red reciben como *input* una secuencia de datos codificados. Tras producir el *output* de esa primera secuencia, la red memoriza su resultado y lo reutiliza para el cálculo de los siguientes elementos [24]. La Figura 2.9 representa la estructura recurrente de este tipo de redes mediante un esquema en donde se muestra desplegada.

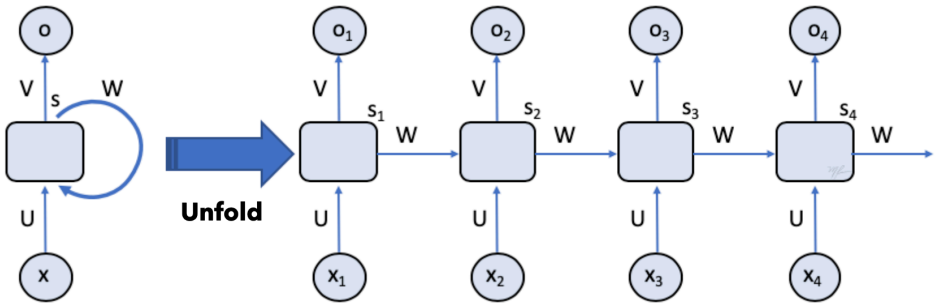


Figura 2.9: Diagrama de funcionamiento de una red neuronal recurrente (RNN) desplegada.

Concretamente, este trabajo emplea módulos *Long Short-Term Memory* (LSTM) para la construcción de su modelo final. Este tipo de módulos son una variante de las redes recurrentes clásicas que pretende solventar el problema habitual de las dependencias a largo plazo. En muchas aplicaciones de secuencias, como el procesamiento del lenguaje natural, las series temporales, o las señales de audio, es importante no solo capturar patrones locales, sino también comprender las dependencias entre elementos que están distantes en el tiempo. Los módulos LSTM abordan este problema con una estructura de memoria especializada que puede mantener, añadir, o eliminar información de su estado a lo largo del tiempo mediante puertas de entrada, olvido y salida [25]. Para detalles más específicos sobre el funcionamiento de este tipo de redes, véase el trabajo de Staudemeyer (2015).

Como se ha podido comprobar a lo largo de esta sección, el campo de la detección e identificación de exoplanetas a través de curvas de luz sigue siempre una estructura más o menos homogénea. Siempre hay una etapa previa de procesado de las curvas de luz y de detección de los TCE, a fin de adaptar el set de datos original a los canales de entrada requeridos por cada modelo. Después de esta etapa llega la fase de construcción de los modelos, en donde, en el caso específico de las redes neuronales, se apilan distintos tipos de capas con el objeto de encontrar la configuración más óptima. Por último, llegaría la interpretación de los resultados, proveídos en el caso de la clasificación binaria por la función de activación sigmoidal o, en el caso de la multiclase, por la función *SoftMax*.

En siguientes secciones se sigue este mismo esquema con los objetivos de, por un lado, comparar las características de los modelos aquí discutidos y de, por otro, encontrar el modelo más óptimo que se adapte a los datos proveídos por la misión estelar Kepler.

3. Fuentes de datos empleadas

En esta sección se exponen las distintas fases que se han ido siguiendo para la construcción del set de datos final. El *dataset* que se ha empleado en este trabajo para la elaboración de los modelos se ha construido desde cero gracias a diferentes bases de datos astronómicas de acceso público.

Siguiendo la tendencia habitual en la literatura, los datos que componen nuestro *dataset* final provienen de dos fuentes de información: (1) el catálogo de muestras etiquetadas de los TCE de la misión Kepler con información adicional de los tránsitos, obtenido de la base de datos de la NASA Exoplanet Archive [13]; y (2) las curvas de luz de larga cadencia de las estrellas a las que pertenecen los TCE previamente descargados, obtenidas de la base de datos MAST [12].

3.1 NASA Exoplanet Archive

En primer lugar, accedimos al repositorio online de la NASA dedicado exclusivamente a exoplanetas: la NASA Exoplanet Archive [13]. Dentro de la base de datos, descargamos información de todos los TCE recopilados de la misión estelar Kepler: en total 20 367 tránsitos. Del total de variables que se encontraban a disposición para su descarga, almacenamos únicamente las siguientes.

- **Identificador numérico**. Identificación numérica por filas del TCE sobre el catálogo completo (*rowid*).
- **Kepler ID**. Identificador numérico de la estrella a la que pertenece cada TCE (*kepid*). Esta será la clave de unión entre este *dataset* y el set de curvas de luz que descargaremos posteriormente a través del MAST.
- **Número de TCE**. Número del TCE en cada curva de luz (*tce_plnt_num*). Una estrella puede estar siendo orbitada por más de un exoplaneta, por lo que también puede albergar más de un TCE a lo largo de su registro lumínico.

- **Periodo orbital**. En Días Julianos Baricéntricos (BJD), el periodo orbital del TCE (*tce_period*).
- **Duración del TCE**. En horas, la duración del TCE sobre el registro lumínico de la estrella (*tce_duration*).
- **Mínimo del TCE**. En Días Julianos Baricéntricos (BJD), menos un desplazamiento constante de 2 454 833 días, el instante temporal en el que se produce el mínimo de cada TCE (*tce_time0bk*).
- **Etiqueta**. Etiqueta de cada TCE para la fase de entrenamiento (*av_training_set*). Como ya se ha comentado previamente, son cuatro etiquetas las definidas por la NASA: *Planet Candidate* (PC), *Astrophysical False Positive* (AFP), *Non-Transiting Phenomenon* (NTP) y *Unknown* (UNK).

Tras la descarga de los datos, obtendremos un *dataset* en formato CSV de dimensiones $20\,367 \times 7$ que incluye el total de TCE descubiertos en las curvas de luz de la misión Kepler. La distribución de TCE y del número total de estrellas que contiene el *dataset* se muestra en la Tabla 3.1.

Etiqueta	Número de TCE	Número de estrellas (Kepler ID)
PC	3600	2178
AFP	9596	5186
NTP	2541	2160
UNK	4630	3145
Total	**20 367**	**12 669**

Tabla 3.1: Tabla resumen del número de TCE y estrellas descargados en función de las cuatro etiquetas de entrenamiento planteadas por la NASA.

En este punto, conviene recordar que el número total de TCE y de estrellas no es coincidente porque cada estrella es capaz de albergar más de un TCE. En el caso de que una estrella contenga más de un TCE catalogado como candidato a exoplaneta nos encontraríamos ante un posible sistema planetario. Este es el caso del sistema Kepler-90 (KOI-351), descubierto en 2013 a partir de los TCE de la estrella con Kepler ID 11442793 [13]. Actualmente es el sistema planetario más grande jamás descubierto junto con nuestro sistema solar, pues cuenta con 8 planetas en órbita.

En cuanto a las etiquetas incluidas en nuestro set de datos, se han dividido las categorías entre la clase positiva y la negativa de la siguiente manera. De la clase positiva dispondríamos únicamente de la categoría PC o candidato a planeta. Pese a que los TCE de esta categoría no tienen porqué corresponderse necesariamente con el tránsito de un exoplaneta –tan solo son candidatos–, la

gran mayoría de trabajos referenciados los toman siempre como datos de la clase positiva [20]. En total dispondríamos de 3600 TCE pertenecientes a esta clase.

Por otro lado, la clase negativa la compondrían originalmente el resto de etiquetas: AFP, NTP y UNK. Los TCE catalogados como UNK representan tránsitos de origen desconocido, sobre los que ningún astrofísico ha podido determinar su origen. Como consecuencia los desecharemos directamente, pues no presentan ningún valor de cara a las fases de entrenamiento posteriores y, además, ya disponemos de suficientes datos de la clase negativa con la suma de los TCE catalogados como falsos positivos astrofísicos (AFP) y como fenómenos no transitorios (NTP). Por tanto, en total dispondríamos de 12 137 TCE pertenecientes a la clase negativa.

Como se puede observar, existe un claro desbalanceo a favor de los datos de la clase negativa. Esta cuestión se resolverá en la fase de implementación de los modelos mediante la aplicación de técnicas de *data augmentation*, en donde los TCE clasificados como candidatos a planeta serán rotados y alterados sintéticamente mediante la adición de reflejos a fin de equilibrar el número total de clases.

3.2 Mikulski Archive for Space Telescopes

En segundo lugar, accedimos a la base de datos del MAST, en donde se almacenan las curvas de luz registradas por todos los telescopios espaciales de todas las misiones estelares que se han ido lanzando en los últimos años. En concreto, necesitamos descargar las 12 669 curvas de luz de las estrellas monitorizadas por la misión Kepler. Para ello, la base de datos requiere introducir mediante distintas *queries* sus identificadores –sus Kepler ID– que ya obtuvimos de nuestra visita anterior a la NASA Exoplanet Archive.

De esta manera, podemos seleccionar únicamente las curvas de luz de las estrellas que necesitamos, además de indicar en la *query* que solo se descarguen las de larga cadencia. Este tipo de curvas, como ya se comentó anteriormente, eran aquellas tomadas por el satélite en intervalos de 29.4 minutos, un periodo de tiempo lo suficientemente amplio como para que los modelos puedan detectar las variaciones en la intensidad lumínica de cada estrella. Además, las curvas de luz están almacenadas en periodos cuatrimestrales, de manera que a cada estrella le corresponden unos 17 archivos que posteriormente habremos de unir manualmente.

Por otro lado, conviene destacar que lo que devuelve cada consulta no son los archivos FITS en donde se almacenan usualmente las curvas de luz, sino archivos TXT con URLs para su descarga. A través de un bucle se fueron recorriendo todas las URL y se descargaron los archivos FITS en

disco. En total se descargaron más de 215 000 archivos FITS, aproximadamente 17 para cada una de las 12 669 estrellas. El proceso de descarga se prolongó durante varios días y ocupa finalmente en disco ∼120 GB.

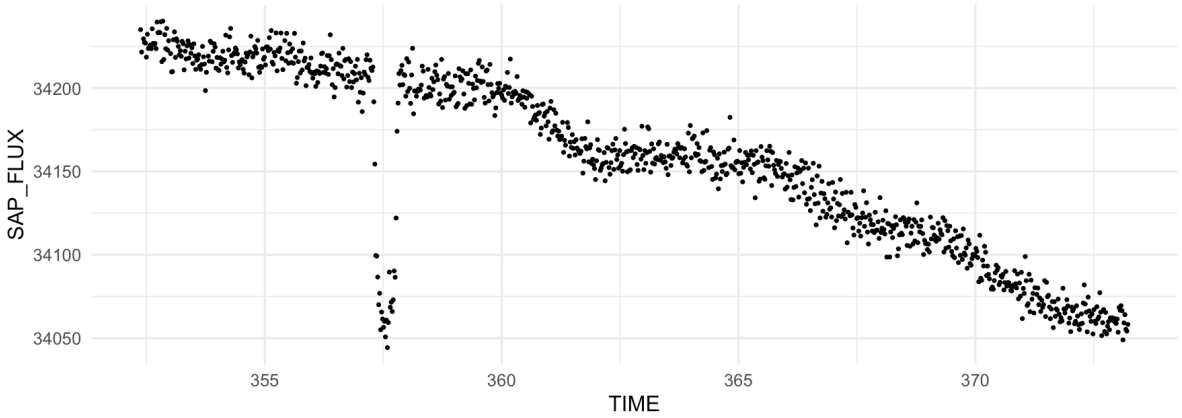


Figura 3.1: Curva de luz de la estrella 11442793 del cuarto cuatrimestre en formato SAP.

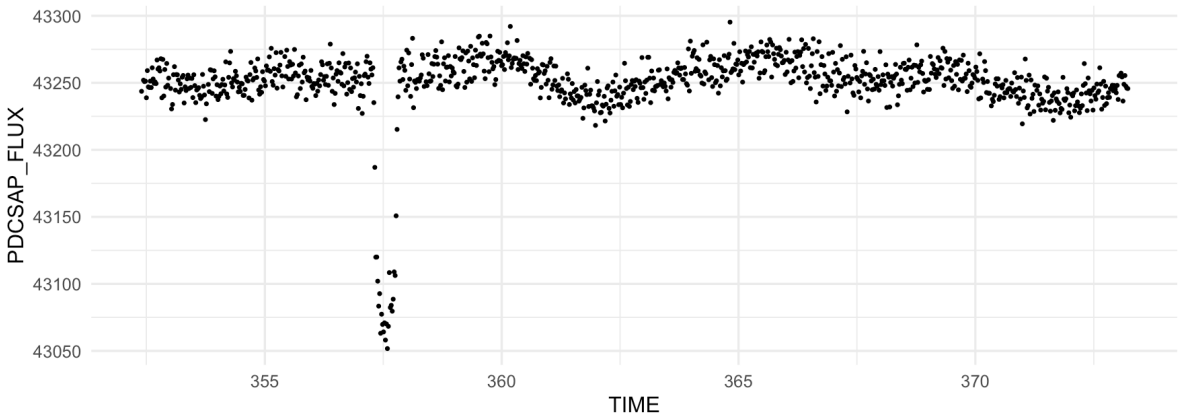


Figura 3.2: Curva de luz de la estrella 11442793 del cuarto cuatrimestre en formato PDC.

Cada uno de los ficheros FITS contiene en su interior una gran cantidad de información de la estrella a la que pertenecen, pero la necesaria para el presente trabajo es únicamente la relativa a las propias mediciones de su intensidad lumínica. En este sentido, las dos variables que almacenan información sobre el flujo lumínico de la estrella son las variables «SAP Flux» (*Simple Aperture Photometry Flux*) y «PDC Flux» (*Pre-search Data Conditioning Flux*). Para una explicación más detallada sobre estos dos tipos de medición, véase el apartado 2.2 de la Sección 2. La Figura 3.1 muestra una curva de luz en formato «SAP Flux», mientras que la Figura 3.2 muestra la misma curva pero en formato «PDC Flux». De cara a la fase de preprocesado, y para agilizar el proceso de depuración, emplearemos finalmente las mediciones de la variable «PDC Flux».

3.3 Unión de las fuentes de información

Una vez almacenados todos los archivos en formato FITS, se extrajeron y se reconvirtieron a formato CSV para proceder a su unión con el *dataset* anterior. Para ello se emplearon diversas funcionalidades de las librerías `lightkurve` y `astropy` en Python para la fusión de todos los cuatrimestres y su posterior transformación a formato CSV.

Con ambos *datasets* en el mismo formato, se procedió a su unión gracias a la variable común Kepler ID, de manera que cada fila pasase a ser un único TCE. La Figura 1.1 del Apartado 1.3 ilustra el formato final en que quedó la unión de ambos *datasets*.

4. Preprocesado y visualización de las curvas de luz

En esta sección se exponen las diferentes etapas que han sido necesarias para la adaptación de los flujos de luz al formato de entrada requerido por los modelos definidos en la fase de implementación. En el último epígrafe de esta sección se confeccionan los *datasets* finales como vectores unidimensionales de cara a los modelos presentados en el siguiente apartado.

4.1 Preprocesado de los datos

A modo de ejemplo, realizaremos todo el preprocesado con el único TCE catalogado como candidato a planeta de la curva de luz de la estrella KIC 11442793. Todo este procedimiento previo se repitió en bucle para los 15 737 TCE que componen el *dataset* actual.

Normalización y escalado de los datos

La forma original de la curva de luz, graficada en función de la variable «PDC Flux» y el tiempo en Días Julianos Baricéntricos, es la que se muestra en la Figura 4.1. En esta figura se pueden observar saltos temporales en los que no existe información acerca del flujo lumínico de la estrella. Estos periodos se consideran valores ausentes, y ocurrieron como consecuencia de la reconfiguración cuatrimestral de los filtros paso banda del satélite Kepler durante sus años de actividad. Como consecuencia de esta misma reconfiguración, también se produjeron cambios en la escala de la propia curva, por lo que cada cuatrimestre se encuentra representado en una posición diferente respecto del eje y. Para una explicación más detallada sobre las limitaciones técnicas de los sensores del satélite, véase el Apartado 2.2.

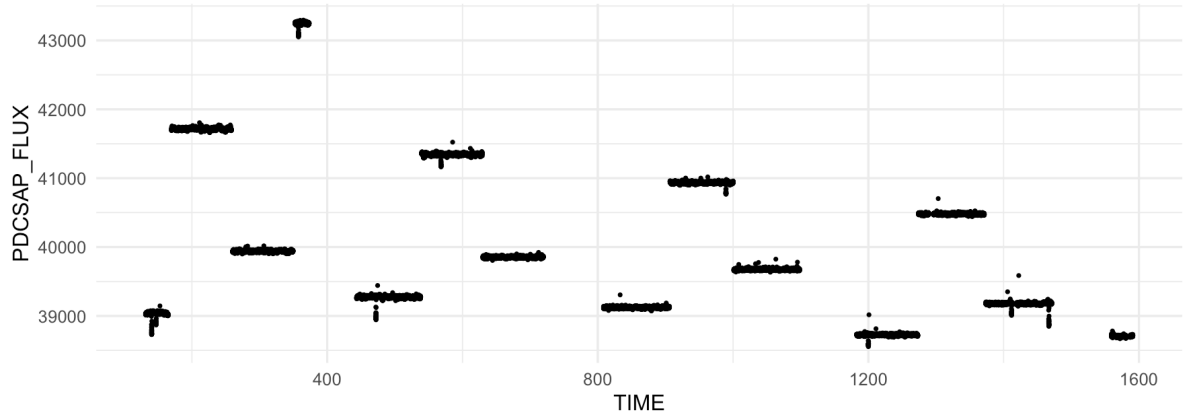


Figura 4.1: Curva de luz de la estrella KIC 11442793 en su estado original. Se pueden observar las diferencias en la escala como consecuencia de la división cuatrimestral de la curva.

Antes de llevar a cabo cualquier modificación estructural sobre los datos de la curva, deberemos normalizar y escalar los puntos de datos respecto del eje y. Nuestro objetivo es centrar todos los cuatrimestres que componen la curva de luz en torno a un mismo valor, de manera que todos los datos de la curva se encuentren contenidos en un rango similar de valores. Esto se consigue dividiendo cada cuatrimestre entre su mediana, lo que provoca que los datos dentro de ese cuatrimestre se reescalen de tal manera que su magnitud pase a estar en relación con su mediana, esto es, en torno al valor $y = 1$. Los resultados del reescalado pueden apreciarse en la Figura 4.2.

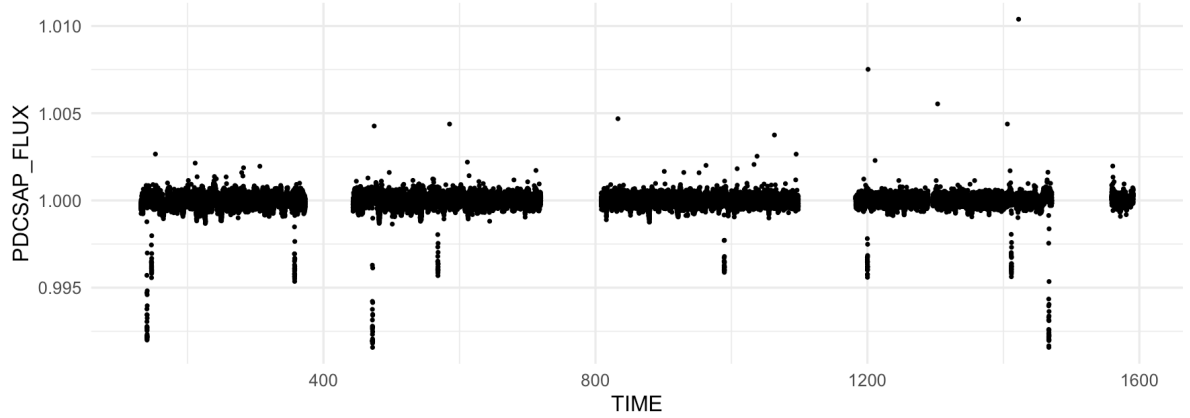


Figura 4.2: Curva de luz de la estrella KIC 11442793 tras la aplicación del proceso de normalización y reescalado de los datos.

Esta sería la configuración inicial de la curva de luz, sobre la cual comenzaría la fase de depuración y preprocesado. En el caso de la estrella que hemos tomado como ejemplo, sin necesidad de alterar su estado original, ya es posible apreciar pequeñas disminuciones en su intensidad lumínica consistentes con el paso de posibles exoplanetas frente a ella.

Tratamiento de valores *outliers*

En esta segunda fase se eliminan los posibles valores *outliers* que pueda presentar la curva de luz, pero únicamente aquellos por encima del valor medio de la propia curva.

Estos valores atípicos, que suelen mostrarse con cierta habitualidad por encima de la media de la curva, son atribuibles a rayos cósmicos, erupciones estelares que incrementan temporalmente el flujo lumínico, o, incluso a errores en el propio sensor del satélite. A efectos del presente trabajo, a la hora de identificar posibles tránsitos, lo único que merece interés son las disminuciones en la actividad lumínica de la estrella, disminuciones que puedan coincidir con el paso de un exoplaneta frente a los sensores del satélite. Es por este motivo que los valores por debajo del valor medio de la curva deben permanecer en su estado original, pues si los alterásemos podríamos malograr los tránsitos de potenciales exoplanetas.

Para la eliminación de estos valores, definimos un límite superior de 4σ. Este valor es el habitual dentro de la literatura y permite capturar y eliminar sin dificultad cualquier dato que supere en 4 veces la desviación estándar de la propia curva. Los resultados pueden apreciarse en la Figura 4.3 si se comparan con el anterior estado de la curva (Figura 4.2).

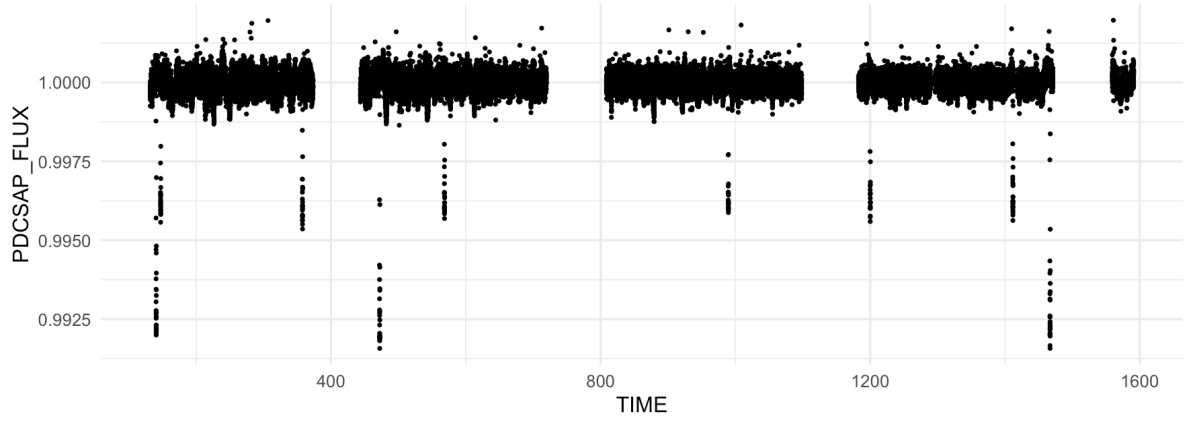


Figura 4.3: Curva de luz de la estrella KIC 11442793 tras la aplicación del proceso de eliminación de valores *outliers*. Es posible apreciar cómo el procedimiento de eliminación no ha afectado a los valores atípicos por debajo del valor medio de la curva.

Eliminación de la tendencia e interpolación de los tránsitos a través de *splines* polinomiales

En esta tercera fase se elimina la tendencia natural de las curvas de luz a fin de poder identificar con mayor facilidad las disminuciones en la luminosidad de las estrellas.

Mediante la creación de un *spline* polinomial adaptativo, lo que se propone en esta fase es eliminar la variación natural de las curvas sin que el propio *spline* se vea afectado por los descensos de los potenciales tránsitos. En este sentido, un *spline* polinomial es una función matemática suave creada para su adaptación a la tendencia de una determinada curva. Una vez ajustado el *spline* a los datos, esta nueva función puede entenderse como la representación de la tendencia de la curva original. Al restar el *spline* ajustado a los datos originales se elimina la tendencia y se obtiene una curva de luz con valores *cuasi* homogéneos, centrados en torno al valor $y = 1$.

El procedimiento para llevar a cabo este proceso es el siguiente. En primer lugar deberemos eliminar temporalmente el tránsito de la estrella que se esté analizando para que no se vea afectado por el aplanamiento provocado por el paso del *spline*. A este proceso se le conoce como «enmascaramiento», y se puede efectuar gracias a los datos que habíamos recopilado anteriormente a través del repositorio digital de la Nasa Exoplanet Archive.

Con los datos del repositorio podremos generar una matriz booleana con *True* durante el periodo del tránsito y *False* para el resto de instantes temporales. Usaremos esta matriz para enmascarar los valores *True* e interpolarlos una vez eliminada la tendencia de la curva. El cálculo de la matriz se realiza en base al periodo orbital, a la duración, y al instante temporal 0 en el que se produce el primer TCE, información que deberemos almacenar previamente como atributos. Para el ejemplo que estamos siguiendo, y en base a la información de que disponemos, la estrella KIC 11442793 presenta un único TCE con $P = 331.603000$ días, $T_0 = 140.480$ días, y una duración del tránsito de 0.60375 horas.

En segundo lugar, deberemos generar el *spline* polinomial en base a una ventana temporal determinada. Esta ventana temporal permite ir reajustando y actualizando el *spline* a las variaciones de la curva, adaptando la tendencia a conveniencia. En el caso del presente trabajo, se decidió una ventana temporal de tres veces la duración del tránsito analizado. Esta ventana, que se adapta en función de las características temporales del TCE, es la más óptima porque permite que el *spline* se ajuste lo suficientemente bien a la curva sin que las actualizaciones sean demasiado costosas en términos de tiempo.

En la Figura 4.4 se puede apreciar la curva de luz sin aplanar con la función del *spline* en color azul. Se puede observar cómo el *spline* se adapta a la variabilidad habitual de la curva sin verse afectado por los descensos provocados por los posibles tránsitos. En la Figura 4.5 se muestra la curva de luz ya aplanada, tras la aplicación del *spline*.

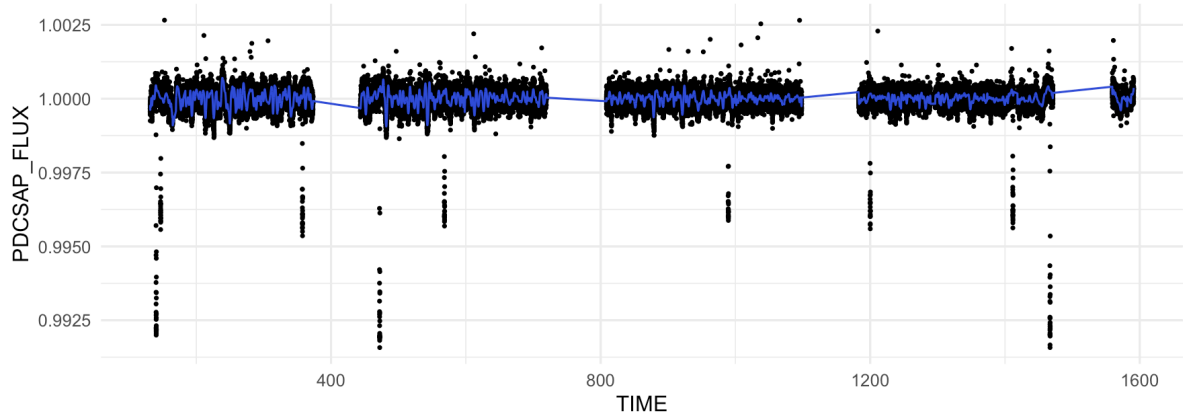


Figura 4.4: Curva de luz de la estrella KIC 11442793 antes de la aplicación del *spline* polinomial. Se puede apreciar en color azul el recorrido de la función generada a partir del *spline*.

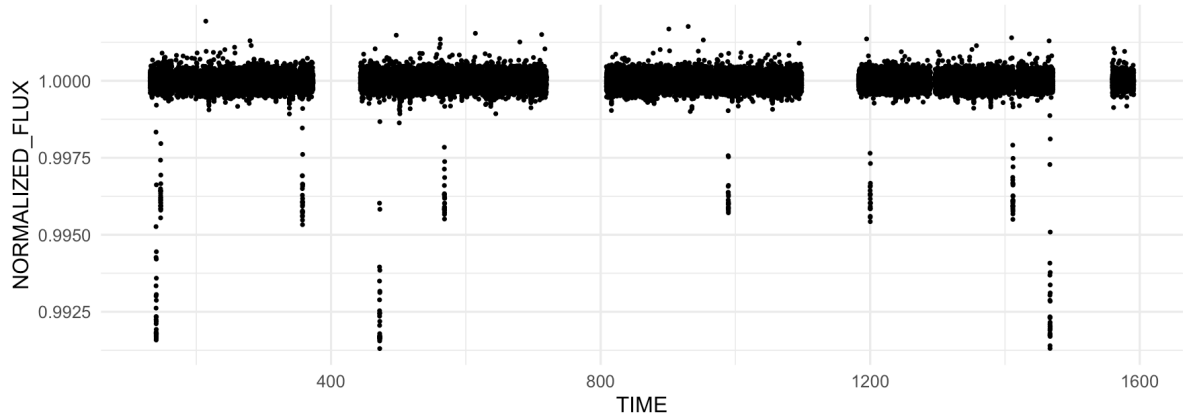


Figura 4.5: Curva de luz de la estrella KIC 11442793 después de la aplicación del *spline* polinomial.

Como el efecto del *spline* es difícil de apreciar en la curva de luz completa, a continuación se muestra su funcionamiento en detalle. En la Figura 4.6 se muestra una porción de la curva de luz antes del aplanado. Se puede observar como el *spline*, representado por la línea azul, se adapta perfectamente a la forma de la curva e ignora los dos posibles tránsitos que se encuentran al principio de la gráfica gracias al enmascarado. La Figura 4.7 muestra la misma porción de la curva tras la eliminación de su tendencia: mientras que los dos tránsitos del principio conservan su forma, el resto de la curva se ha aplanado satisfactoriamente.

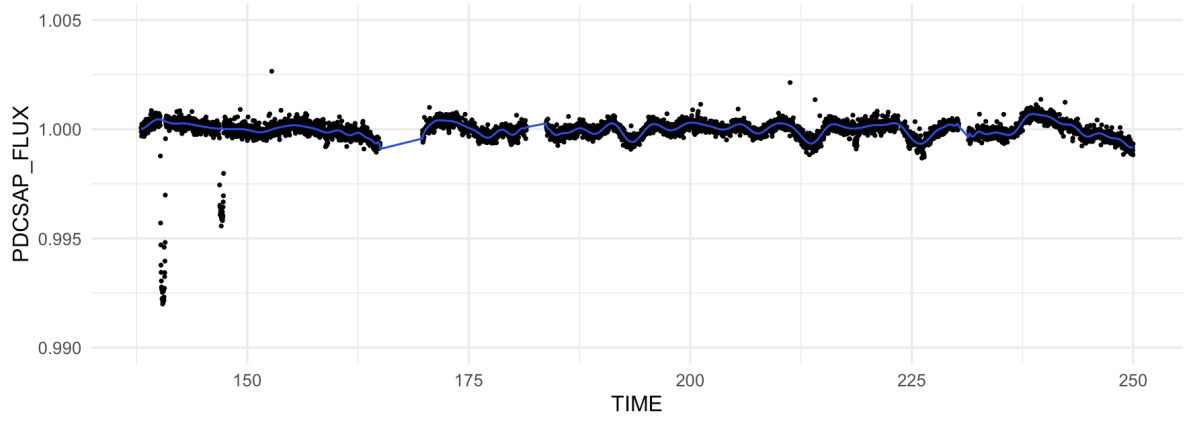


Figura 4.6: Porción de la curva de luz de la estrella KIC 11442793 entre los instantes temporales 138 y 250 antes de la aplicación del *spline* polinomial. Se puede apreciar en color azul el recorrido de la función generada a partir del *spline*.

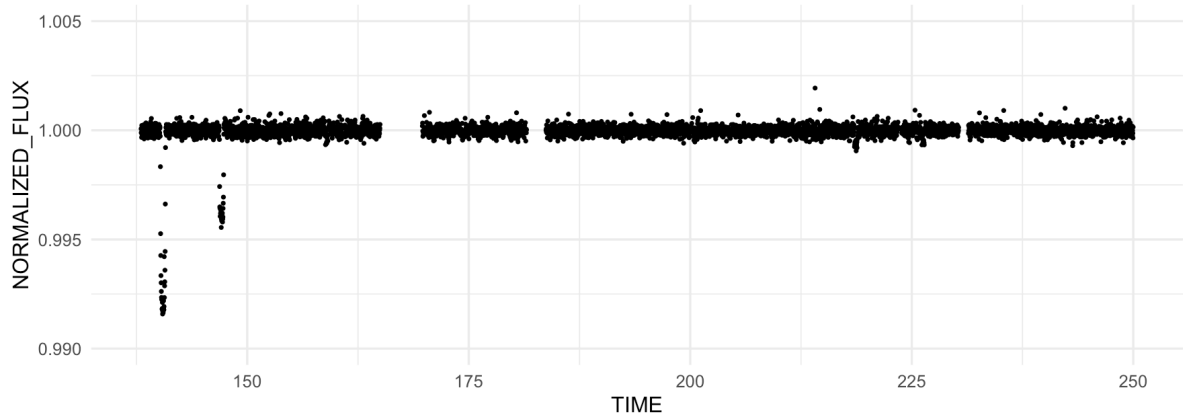


Figura 4.7: Porción de la curva de luz de la estrella KIC 11442793 entre los instantes temporales 138 y 250 tras la aplicación del *spline* polinomial.

Uniformación de las longitudes de las curvas a través de la técnica del *phase-folding*

En este cuarto apartado se generan las versiones «dobladas» o *phase-fold* de las curvas de luz. El objetivo de esta fase es lograr una visión aún más reforzada del tránsito, de manera que los algoritmos posteriores puedan aprender de sus formas con mayor facilidad.

En este sentido, la técnica del *phase-folding* consiste en tomar una curva de luz y «doblarla» en un patrón periódico para que la señal parezca repetirse a lo largo de un ciclo completo. Ello se logra dividiendo el tiempo por el período de la señal del tránsito, lo que da como resultado un valor

conocido como «fase». El *phase-folding* es útil porque permite visualizar patrones en los datos que no son evidentes cuando se representan en el tiempo. Por ejemplo, las curvas de luz de la mayoría de estrellas variables no muestran patrones claros si se grafican en función del tiempo debido a su característica variabilidad, pero cuando se realiza el *phase-folding* con el período orbital del tránsito analizado, el patrón del TCE del posible planeta se hace evidente, ya que queda centrado en el instante temporal 0 como consecuencia de su solapamiento.

Para poder realizar el *phase-folding* sobre el tránsito de una curva de luz deberemos conocer previamente el valor de dos parámetros del propio tránsito, como son su periodo orbital, y el instante temporal en el que se produce el primer TCE sobre la curva de luz. Para el ejemplo que estamos siguiendo, y en base a la información que extrajimos del repositorio digital de la NASA Exoplanet Archive, el único TCE de la estrella KIC 11442793 presenta un periodo orbital de 331.603000 días cuya primera señal se registra en el instante temporal 140.480.

En la Figura 4.8 se puede observar la versión «doblada» de la curva de luz una vez aplicada la técnica del *phase-folding*. Todos los tránsitos registrados en la curva se han solapado y centrado en el instante temporal 0. Como consecuencia, la visión del tránsito ha quedado reforzada.

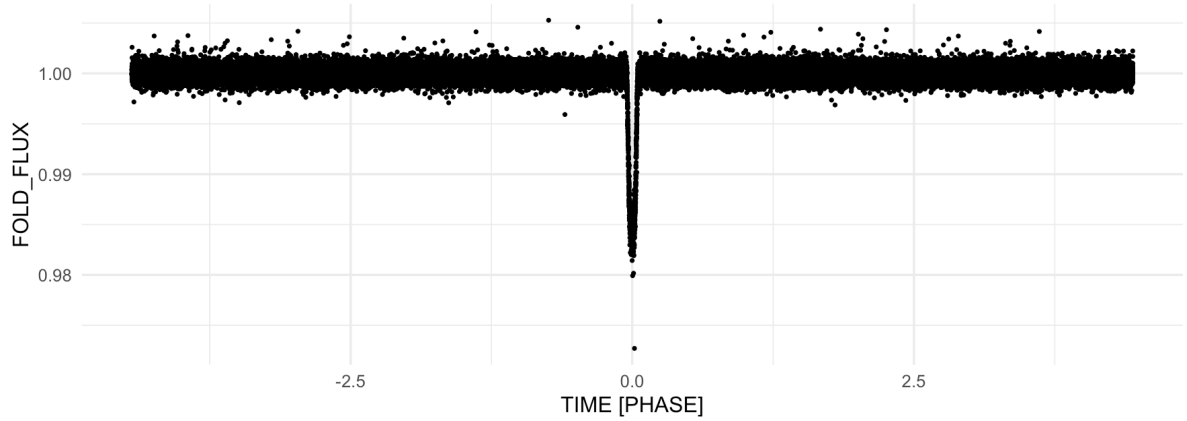


Figura 4.8: Versión *phase-fold* de la curva de luz de la estrella KIC 11442793. Se puede observar la forma del TCE centrado en $x = 0$.

Agrupamiento o *binning* de las curvas de luz

En esta última fase se generan los datos de entrada finales para la posterior implementación de los modelos. El objetivo de esta fase es obtener dos visiones de cada TCE: una visión global con gran amplitud en sus extremos, y una segunda visión local, más cercana, en la que el tránsito quede centrado y enfocado.

A partir de la versión anterior, en la que se plegaron los tránsitos y se generó una curva en la que se aglomeraron todos los puntos de datos en un rango temporal más limitado, en esta fase aplica-

remos la técnica del agrupamiento o *binning* con tal de eliminar datos irrelevantes y uniformizar el tamaño de todas las versiones finales de las curvas. En muchos de los modelos que aplicaremos posteriormente es necesario que todos los datos de entrada presenten el mismo tamaño, por lo que deberemos limitar el agrupamiento de las dos visiones a vectores de medidas específicas. Además, ambas versiones finales se normalizan en torno a $y = 0$ con mínimo en $y = -1$.

En la Figura 4.9 se puede apreciar el tránsito final en su versión global de la estrella KIC 11442793. Las versiones globales de todos los tránsitos están limitadas a un vector unidimensional de tamaño 2001×1, de manera que todos contengan 2001 puntos de datos o *bins*. Por un lado, se consigue visualizar más claramente el tránsito y, por otro, se elimina información irrelevante y reiterativa que ralentizaría en exceso la puesta en marcha de los modelos.

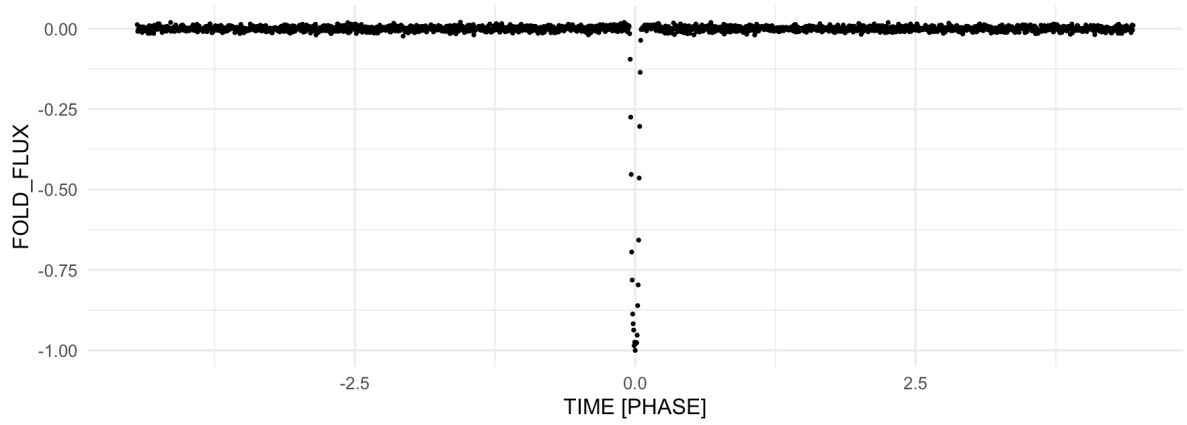


Figura 4.9: Visión global del tránsito analizado de la estrella KIC 11442793.

Por último, en la Figura 4.10 se muestra el tránsito final en su versión local de la estrella KIC 11442793. Para el caso de las versiones locales, el vector se limita a la dimensión 201×1, de manera que todos contengan 201 puntos de datos o *bins* por igual.

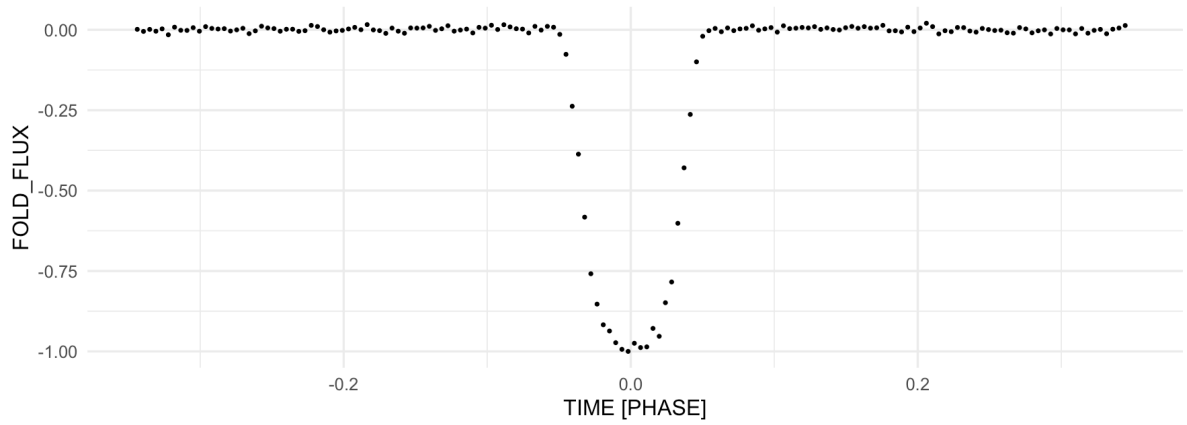


Figura 4.10: Visión local del tránsito analizado de la estrella KIC 11442793.

4.2 Configuración final del *dataset*

Tras aplicar todas estas fases a cada uno de los 15 737 TCE –proceso que tardó varias semanas en completarse–, finalmente se obtuvieron dos *datasets*: uno para las versiones globales de los TCE, y otro para las versiones locales. Los datos de entrada están almacenados de manera que cada fila se corresponde con un tránsito astronómico y cada columna con una medición de la intensidad lumínica en un instante temporal determinado. De esta manera, mientras que el *dataset* de las versiones globales tiene una dimensión de 15 737 filas y 2001 columnas –una por cada punto de información o *bin*– (15737×2001), el *dataset* de las versiones locales es de 15737×201. Todos los tránsitos astronómicos presentan la misma longitud y están preparados para introducirse como vectores unidimensionales en los modelos discutidos. La Figura 1.1 del Apartado 1.3 ilustra el formato genérico que presentan ambos *datasets*.

En la siguiente sección se muestran los detalles, la construcción y los hiperparámetros de cada uno de los modelos planteados en el Apartado 1.3, así como se definen las principales métricas para la evaluación de su rendimiento con el objetivo de poder comparar posteriormente los resultados.

5. Implementación y evaluación de los resultados

En esta sección se presentan los distintos métodos y modelos empleados para la clasificación de los tránsitos astronómicos de las curvas de luz preprocesadas.

La sección está dividida de la siguiente manera. En primer lugar se introduce un apartado de metodología en donde se definen distintas cuestiones de carácter técnico relacionadas con las métricas, las particiones y los modelos a aplicar. En segundo lugar procedemos a balancear los *datasets* y a crear los generadores de «lotes» o *batches* de cara a la construcción de los modelos de arquitectura neuronal. En tercer lugar pasamos directamente a los apartados de implementación de los modelos, en donde se presentan dos formas de modelización: una primera aplicando sobre los tránsitos preprocesados el Análisis de Componentes Principales Funcionales (ACPF); y una segunda forma empleando directamente los tránsitos astronómicos preprocesados.

Sobre estas dos formas de presentar la información se construyen distintos tipos de modelos: por un lado, se aplican modelos tradicionales –KNN, *Random Forest*, *Gradient Boosting*, etc.– que, a pesar de haberse visto sobrepasados por otra clase de modelos más complejos, consideramos importante mantenerlos y experimentar con ellos sobre nuestros propios *datasets* preprocesados. Y, por otro, se construyen los modelos de arquitectura neuronal que mejores resultados ofrecen en la actualidad. Concretamente se construyen tres tipos de red neuronal para, posteriormente, combinarse a conveniencia: las redes neuronales recurrentes (RNN), las redes neuronales totalmente conectadas o *fully connected*, y las redes neuronales convolucionales unidimensionales (CNN 1D).

5.1 Metodología

En este primer apartado se detallan algunas cuestiones de carácter técnico que conciernen a todos los modelos implementados en el trabajo. En concreto se definen las métricas con las que se eva-

luarán los resultados de los modelos, se definen los principales hiperparámetros, y se detallan las particiones y los métodos de validación de cara a la fase de entrenamiento.

Métricas

Las métricas que se emplearán para evaluar el funcionamiento de los modelos son las habituales dentro de la ciencia de datos. En concreto, emplearemos el *accuracy*, el *recall*, la *precision*, y la curva ROC y el AUC.

En primer lugar, el *accuracy* representa el porcentaje de muestras correctamente clasificadas frente al total de observaciones. Gracias a las técnicas de *data augmentation* nuestros *datasets* quedarán balanceados, por lo que puede servirnos de utilidad a la hora de comparar los resultados de nuestros modelos. Para su cálculo son necesarios los siguientes resultados proveídos por los modelos:

- ***True Positive*** **(TP)**. Muestras positivas correctamente clasificadas.
- ***True Negative*** **(TN)**. Muestras negativas correctamente clasificadas.
- ***False Positive*** **(FP)**. Muestras erróneamente clasificadas como positivas.
- ***False Negative*** **(FN)**. Muestras erróneamente clasificadas como negativas.

Siguiendo esta notación, la fórmula se desarrolla como sigue:

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.1}$$

Por su parte, el *recall* representa el porcentaje de positivos reales que fueron correctamente identificados como tales. Se calcula de la siguiente manera:

$$RC = \frac{TP}{TP + FN} \tag{5.2}$$

De manera complementaria al *recall*, la *precision* representa qué porcentaje de las observaciones que el modelo etiquetó como positivos son verdaderamente positivos. Se calcula de la siguiente manera:

$$PC = \frac{TP}{TP + FP} \tag{5.3}$$

La curva ROC (*Receiver Operating Characteristic*) es un gráfico que representa la capacidad de un modelo para discriminar entre clases positivas y negativas a diferentes umbrales de clasificación. En el eje y, se representa la tasa de verdaderos positivos (sensibilidad) y en el eje x la tasa de falsos positivos (1 - especificidad). Una curva ROC ideal se acercaría al extremo superior izquierdo, indicando una alta tasa de verdaderos positivos y una baja tasa de falsos positivos.

Por su parte, el AUC (*Area Under the Curve*) es una métrica que cuantifica el rendimiento general del modelo, representando el área bajo la curva ROC. Un AUC de 1 indica un modelo perfecto, mientras que un AUC de 0.5 sugiere que el modelo no tiene capacidad de discriminación, equivalente a clasificar al azar.

Hiperparámetros

A continuación se definen los principales hiperparámetros comunes a la mayoría de modelos neuronales:

- **Épocas (*epochs*)**. Número de iteraciones en las que se procesa todo el conjunto de datos a través de la red. En nuestro caso seleccionamos un valor habitual de 100 *epochs* para todos los modelos de arquitectura neuronal.

- **Tasa de aprendizaje (*learning rate*)**. El *learning rate* o tasa de aprendizaje controla cuánto se ajustan los pesos del modelo en respuesta al error estimado en cada iteración del entrenamiento. En nuestro caso seleccionamos un valor que varía entre 0.01 y 0.0001 para todos los modelos de arquitectura neuronal.

- **Tamaño de los lotes (*batch size*)**. Paquete de observaciones que procesará la red en cada *epoch*. En nuestro caso seleccionamos un valor habitual de 64 *batches* para todos los modelos de arquitectura neuronal.

- **Función de pérdida (*loss*)**. Métrica que cuantifica cuán lejos están las predicciones de un modelo de los valores verdaderos que intenta predecir. Se utiliza durante el proceso de entrenamiento para ajustar los parámetros del modelo con el objetivo de minimizar el valor de la propia función, lo que a su vez hace aumentar su precisión. En nuestro caso hemos seleccionado la función de pérdida *binary crossentropy*, habitual en la clasificación binaria.

- **Optimizador (*optimizer*)**. El optimizador ayuda a ajustar los parámetros internos del modelo, como los pesos y sesgos, con el objetivo de minimizar la función de pérdida. Para nuestros modelos, el optimizador empleado es *Adam*, con un *learning rate* asociado del 0.0001.

- **Función de activación (*activation function*)**. La función de activación se aplica a la salida de cada neurona y determina si la neurona debe «activarse» o no, es decir, si debe transmitir

información más adelante a través de la red. La función de activación ReLU (*Rectified Linear Unit*) es la más utilizada y la que emplearemos en nuestros modelos.

Particiones y métodos de validación

De los 15 737 TCE que disponemos en nuestros dos *datasets*, se han decidido dividir los datos en 70 % para la fase de entrenamiento, 15 % para validación, y 15 % para datos test. En total, para la fase de entrenamiento disponemos de 13 219 curvas, mientras que para las fases de validación y test disponemos de 1259 curvas de luz.

Por otro lado, del total de curvas de luz, 3500 pertenecen originalmente a la clase positiva (*Planet Candidate*), mientras que 12 237 pertenecen a la clase negativa (*Astrophysical False Positive* y *Non-Transiting Phenomenon*). En el siguiente apartado solventaremos este claro desbalanceo mediante la aplicación de técnicas de *data augmentation* a los patrones de las curvas de luz.

En lo referente a los métodos de validación y por motivos de capacidad de cómputo, se optará por seguir el método de la validación cruzada en el caso de los modelos de arquitectura neuronal, y validación cruzada repetida para los modelos tradicionales.

5.2 Preparación de las particiones: *data augmentation* y creación de *batches*

Data augmentation

Las técnicas de *data augmentation* hacen referencia al conjunto de operaciones que permiten aumentar artificialmente la muestra de nuestro *dataset* generando nuevos datos modificados de los originales. Esta práctica es muy habitual en las tareas de clasificación de imágenes, en donde la muestra se rota, se recorta o se invierte a fin de engrosar el set de datos y generar modelos más robustos. En el caso que nos ocupa, nuestros datos son secuencias temporales unidimensionales, por lo que los procedimientos aplicados han sido ligeramente distintos. En concreto hemos empleado la inversión, el desplazamiento lateral y el escalado de las secuencias de datos para ambos *datasets*.

- **Inversión**. Una de las formas más comunes de modificación artificial de los datos es emplear la misma secuencia temporal pero desplegada a la inversa. A pesar de su simplicidad, este procedimiento es muy limitado dado que solo podremos obtener una nueva secuencia

de cada una de las originales. Es por ello que debe combinarse con otros procedimientos alternativos.

- **Escalado**. El escalado de los datos es otro procedimiento simple pero efectivo que consiste en reajustar los valores originales de las curvas de manera aleatoria en un rango previamente determinado.
- **Desplazamiento lateral**. El procedimiento del desplazamiento lateral de las secuencias implica el desplazamiento de los puntos de datos sobre la curva de luz en la escala de tiempo de manera horizontal. Con ello es posible seleccionar, cortar e insertar determinados segmentos de datos en los extremos opuestos de las secuencias, generando con ello nuevas curvas con diferentes aspectos.

En los modelos de arquitectura neuronal, estas técnicas no se aplican directamente sobre los sets de datos, sino sobre cada *batch* construido durante la fase de entrenamiento. En el siguiente apartado se desarrolla esta cuestión mediante la creación de los *batch generators*.

Batch generators

En el caso de los modelos de red neuronal, para poder proceder a su compilación se requiere de una herramienta previa para que la capa de entrada pueda reconocer nuestros sets de datos. Esta herramienta es conocida como generador de lotes o *batch generator*. Un *batch generator* es un dispensador utilizado para cargar datos en lotes (o *batches*) durante la fase de entrenamiento de modelos normalmente construidos a través de redes neuronales. Tras introducir los datos, estos se reconfiguran en *batches* de un mismo tamaño pudiendo ser agrupados según diferentes criterios.

En nuestro caso, como ya se ha comentado con anterioridad, el tamaño de los *batches* será de 64 muestras. De estas 64, 32 pertenecerán a la clase positiva y las otras 32 a la clase negativa, de manera que cada *batch* quede automáticamente balanceado. Como las técnicas de *data augmentation* están configuradas dentro del propio generador, cada vez que se genera un nuevo *batch*, parte de las 32 muestras de la clase positiva –la minoritaria– serán creadas sintéticamente. Con ello lograremos, por un lado, modelos más robustos que aprendan de otros tipos de curvas menos depuradas que las habituales; y, por otro, que nuestros modelos aprendan por igual de ambas clases.

5.3 Implementación con Análisis de Componentes Principales Funcionales

En este primer apartado de implementación emplearemos como datos de entrada el resultado del Análisis de Componentes Principales Funcionales (ACPF) realizado sobre las curvas de luz preprocesadas.

El ACPF es una extensión de la técnica del Análisis de Componentes Principales (ACP) para datos funcionales, como curvas o series temporales. En lugar de tratar con datos en un número finito de dimensiones, como es el caso del ACP, el ACPF se ocupa de datos relativos a funciones continuas. Esta técnica busca identificar las principales funciones o componentes que capturan la mayor variabilidad sobre el conjunto total de datos funcionales.

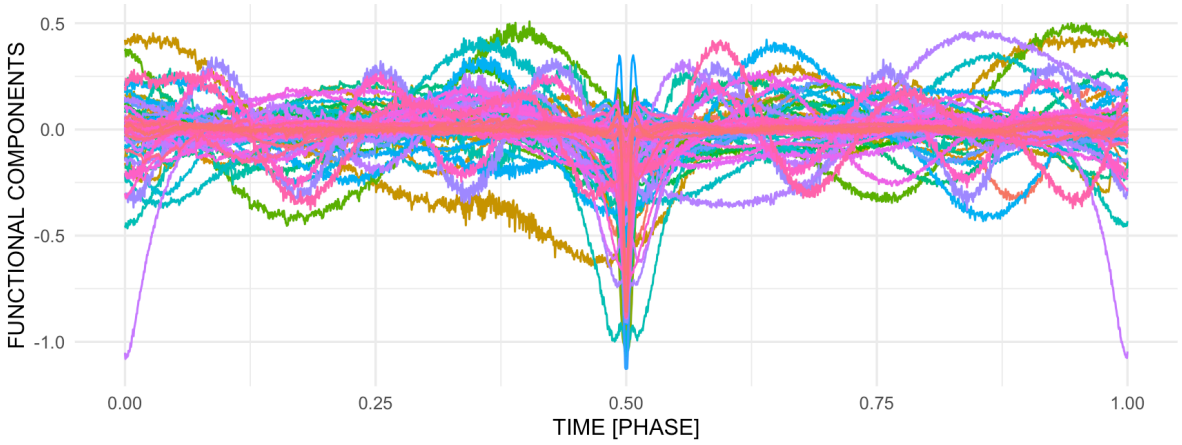


Figura 5.1: Representación de las funciones principales para el *dataset* de curvas de luz globales.

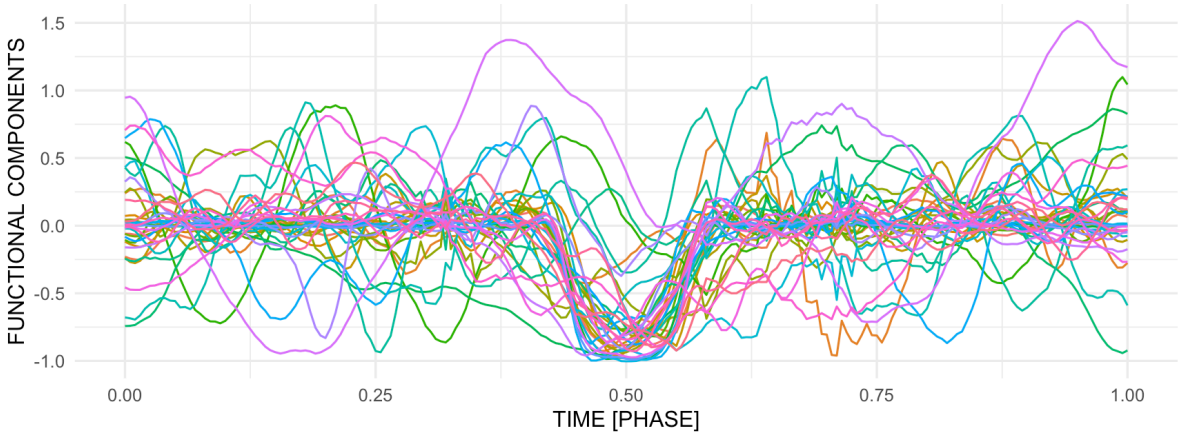


Figura 5.2: Representación de las funciones principales para el *dataset* de curvas de luz locales.

Mientras que para nuestro *dataset* de curvas de luz en su versión global seleccionamos 1456 componentes de 2001 con el 90.01 % de la variabilidad de los datos explicada, para el *dataset* en su versión local seleccionamos 36 componentes de 201, con un 90.18 % de la variabilidad explicada. Las Figuras 5.1 y 5.2 muestran la representación de las funciones principales para ambos *datasets*.

Experimentación con modelos clásicos

En esta sección se experimenta con modelos clásicos sobre los datos que ha ofrecido la técnica del ACPF. En concreto, probaremos con Regresión Logística, *K-Nearest Neighbors*, *Gradient Boosting*, *Random Forest* y distintos modelos de la familia SVM. En primer lugar se experimenta con los *datasets* de las versiones globales para, posteriormente, probar con los de las versiones locales.

Regresión Logística

Para los modelos de Regresión Logística se ha aplicado la función `LogisticRegression()` del paquete `sklearn`, que permite los hiperparámetros `C` y `max_iter` para su tuneo. Mientras que el hiperparámetro `C` representa el inverso de la fuerza de regularización, `max_iter` representa el número máximo de iteraciones a ejecutar. El *grid* definido para este modelo viene indicado en la Tabla 5.1.

Hiperparámetro	***Grid***
`C`	[0.001, 0.01, 0.1, 1, 10, 100, 1000]
`max_iter`	[50, 100, 200, 500, 1000, 2000]

Tabla 5.1: Tabla resumen de los hiperparámetros a tunear y su *grid* para Regresión Logística.

En primer lugar, lanzaremos el modelo con el *dataset* ACPF de las versiones globales de las curvas de luz. El modelo tardó 1 minuto y 13 segundos en ejecutarse, y seleccionó para `C` y `max_iter` los parámetros 10 y 100, respectivamente. Sobre validación, el *accuracy* alcanzó el 84.14 %.

En segundo lugar, volvemos a lanzar el modelo pero esta vez con el *dataset* ACPF de las versiones locales. El modelo tardó 40.8 segundos en ejecutarse, y seleccionó para `C` y `max_iter` los parámetros 100 y 1000, respectivamente. Sobre validación, el *accuracy* alcanzó el 80.43 %.

La Tabla 5.2 muestra sobre datos test los resultados de ambos modelos.

Modelo Regresión Logística	*Accuracy*	*Recall*	*Precision*	AUC
Versiones globales	82.32 %	82.46 %	81.85 %	0.8232
Versiones locales	82.32 %	81.99 %	82.59 %	0.8231

Tabla 5.2: Tabla resumen de los resultados sobre datos test para los modelos de Regresión Logística.

K-Nearest Neighbors

Para los modelos KNN se ha aplicado la función `KNeighborsClassifier()`, que permite los hiperparámetros `n_neighbors`, `weights`, `metric` y `p` para su tuneo. El *grid* definido para este modelo viene indicado en la Tabla 5.3.

Hiperparámetro	*Grid*
`n_neighbors`	[3, ... , 50]
`weights`	['uniform', 'distance']
`metric`	['euclidean', 'manhattan']
`p`	[0.5, 1, 1.5, 2]

Tabla 5.3: Tabla resumen de los hiperparámetros a tunear y su *grid* para *K-Nearest Neighbors*.

En primer lugar, lanzaremos el modelo con el *dataset* ACPF de las versiones globales de las curvas como *input*. El modelo tardó 115 minutos y 21 segundos en ejecutarse, y seleccionó para `n_neighbors` un valor de 23, para `weights` el parámetro `'distance'`, para `metric` la opción `manhattan`, y para `p` el valor 1. Sobre validación, el *accuracy* alcanzó el 85.01 %.

En segundo lugar, volvemos a lanzar el modelo pero esta vez con el *dataset* ACPF de las versiones locales. El modelo tardó 65 minutos y 17 segundos en ejecutarse, y seleccionó para `n_neighbors` el valor 25, para `weights` el parámetro `'distance'`, para `metric` la opción `euclidean`, y para `p` el valor 1. Sobre validación, el *accuracy* alcanzó el 80.99 %.

La Tabla 5.4 muestra sobre datos test los resultados de ambos modelos.

Modelo *K-Nearest Neighbors*	*Accuracy*	*Recall*	*Precision*	AUC
Versiones globales	89.78 %	86.29 %	94.44 %	0.8980
Versiones locales	80.85 %	75.62 %	90.74 %	0.8090

Tabla 5.4: Tabla resumen de los resultados sobre datos test para los modelos *K-Nearest Neighbors*.

Gradient Boosting

En el caso de los modelos *Gradient Boosting* se ha aplicado la función `GradientBoostingClassifier()`, que permite los hiperparámetros `n_estimators`, `learning_rate` y `max_depth` para su tuneo. El *grid* definido para este modelo viene indicado en la Tabla 5.5.

Hiperparámetro	***Grid***
`n_estimators`	[10, 50, 100, 200, 400, 600]
`learning_rate`	[0.0001, 0.001, 0.01, 0.1, 0.5]
`max_depth`	[5, 10, 15, 20, 25, 30]

Tabla 5.5: Tabla resumen de los hiperparámetros a tunear y su *grid* para *Gradient Boosting*.

En primer lugar, lanzaremos el modelo con el *dataset* ACPF de las versiones globales de las curvas como *input*. El modelo tardó 332 minutos y 56 segundos en ejecutarse, y seleccionó para `n_estimators` un valor de 400, para `learning_rate` un valor de 0.1, y para `max_depth` el valor 10. Sobre validación, el *accuracy* alcanzó el 89.54 %.

En segundo lugar, volvemos a lanzar el modelo pero esta vez con el *dataset* ACPF de las versiones locales. El modelo tardó 102 minutos y 23 segundos en ejecutarse, y seleccionó para `n_estimators` un valor de 200, para `learning_rate` un valor de 0.1, y para `max_depth` el valor 25. Sobre validación, el *accuracy* alcanzó el 81.95 %.

La Tabla 5.6 muestra sobre datos test los resultados de ambos modelos.

Modelo *Gradient Boosting*	***Accuracy***	***Recall***	***Precision***	**AUC**
Versiones globales	90.25 %	90.07 %	89.32 %	0.9002
Versiones locales	82.78 %	78.42 %	90.19 %	0.8301

Tabla 5.6: Tabla resumen de los resultados sobre datos test para los modelos *Gradient Boosting*.

Random Forest

La función aplicada para la ejecución de los modelos *Random Forest* es `RandomForestClassifier()`, que permite los hiperparámetros `n_estimators`, `max_depth` y `min_samples_split` para su tuneo. El *grid* definido para este modelo viene indicado en la Tabla 5.7.

Hiperparámetro	*Grid*
`n_estimators`	[10, 50, 100, 200, 400, 600]
`max_depth`	[5, 10, 15, 20, 25, 30]
`min_samples_split`	[2, 5, 10]

Tabla 5.7: Tabla resumen de los hiperparámetros a tunear y su *grid* para *Random Forest*.

En primer lugar, lanzaremos el modelo con el *dataset* ACPF de las versiones globales de las curvas como *input*. El modelo tardó 203 minutos y 31 segundos en ejecutarse, y seleccionó para `n_estimators` un valor de 400, para `max_depth` un valor de 20, y para `min_samples_split` el valor 5. Sobre validación, el *accuracy* alcanzó el 89.21 %.

En segundo lugar, volvemos a lanzar el modelo pero esta vez con el *dataset* ACPF de las versiones locales. El modelo tardó 63 minutos y 13 segundos en ejecutarse, y seleccionó para `n_estimators` un valor de 100, para `max_depth` un valor de 10, y para `min_samples_split` el valor 5. Sobre validación, el *accuracy* alcanzó el 82.39 %.

La Tabla 5.8 muestra sobre datos test los resultados de ambos modelos.

Modelo *Random Forest*	*Accuracy*	*Recall*	*Precision*	AUC
Versiones globales	90.87 %	88.97 %	92.07 %	0.9101
Versiones locales	82.97 %	78.68 %	90.19 %	0.8301

Tabla 5.8: Tabla resumen de los resultados sobre datos test para los modelos *Random Forest*.

Modelos SVM

Por último, para los modelos SVM se ha aplicado la función `SVC()`, que, de nuevo, permite los hiperparámetros `C`, `gamma` y `kernel` para su tuneo. El *grid* definido para este modelo viene indicado en la Tabla 5.9.

Hiperparámetro	*Grid*
`C`	[0.1, 0.5, 0.8, 1, 1.2, 1.5, 2]
`gamma`	[1, 0.1, 0.01, 0.001, 0.0001]
`kernel`	['linear', 'poly', 'rbf']

Tabla 5.9: Tabla resumen de los hiperparámetros a tunear y su *grid* para SVM.

En primer lugar, lanzaremos el modelo con el *dataset* ACPF de las versiones globales de las curvas de luz. El modelo tardó 98 minutos y 78 segundos en ejecutarse, y seleccionó para `C`, `gamma` y

`kernel` los parámetros 1.5, 0.1 y `rbf`, respectivamente. Sobre validación, el *accuracy* alcanzó el 88.79 %.

En segundo lugar, volvemos a lanzar el modelo pero esta vez con el *dataset* ACPF de las versiones locales. El modelo tardó 34 minutos y 21 segundos en ejecutarse, y seleccionó para `C`, `gamma` y `kernel` los parámetros 1.2, 0.1 y `rbf`, respectivamente. Sobre validación, el *accuracy* alcanzó el 78.47 %.

La Tabla 5.10 muestra sobre datos test los resultados de ambos modelos.

Modelo SVM	*Accuracy*	*Recall*	*Precision*	AUC
Versiones globales	89.78 %	87.57 %	92.59 %	0.8979
Versiones locales	77.90 %	73.45 %	87.41 %	0.7801

Tabla 5.10: Tabla resumen de los resultados sobre datos test para los modelos SVM.

Experimentación con redes neuronales recurrentes (RNN)

En esta sección se aplica sobre los mismos sets de datos un modelo de red neuronal recurrente (RNN). Estos modelos se construyen por niveles y de manera secuencial, por lo que los hiperparámetros a tunear se encuentran integrados en la configuración de las propias capas. A continuación se muestra la estructura del modelo para los *datasets* ACPF de ambas versiones concatenadas.

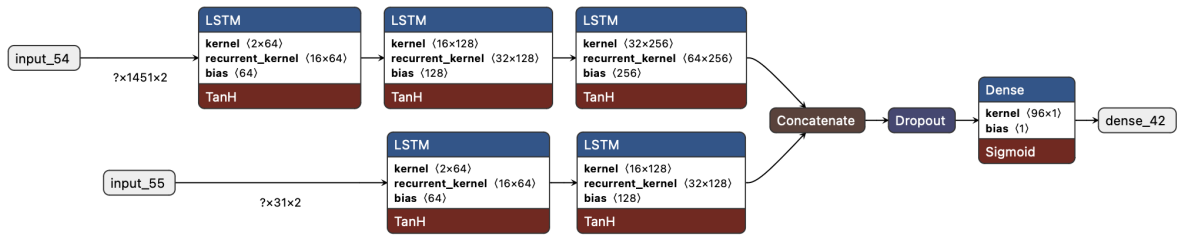


Figura 5.3: Esquema secuencial de las capas que configuran el modelo de arquitectura recurrente en su versión combinada.

Como se puede observar en la Figura 5.3, el modelo presenta en primer lugar dos capas *input* en donde se introducen los datos de entrenamiento de ambos *datasets*. Para las versiones globales se dispone un modelo recurrente de tres capas LSTM de 16, 32 y 64 unidades, mientras que para las locales se configura un modelo con una capa LSTM menos, ya que los datos no presentan tanta complejidad. Las unidades de sus capas son 16 y 32. Ambas construcciones se unen a través de una capa *Concatenate*, se aplica una capa *Dropout* de 0.25 para controlar el sobreajuste, y se concluye

con una capa *Dense* con función de activación sigmoidal que devuelve la probabilidad entre 0 y 1 de que cada una de las curvas introducidas coincida o no con el tránsito de un exoplaneta.

Los modelos tardaron 17 minutos de media en ejecutarse y alcanzaron como máximo en validación un *accuracy* del 74.1 %. La Tabla 5.11 muestra sobre datos test los resultados de los modelos para cada una de las versiones.

Modelo RNN	*Accuracy*	*Recall*	*Precision*	AUC
Versiones globales	70.62 %	69.21 %	72.54 %	0.7156
Versiones locales	71.57 %	71.42 %	71.25 %	0.7202
Versiones combinadas	73.43 %	72.23 %	73.67 %	0.7325

Tabla 5.11: Tabla resumen de los resultados sobre datos test para los modelos de arquitectura recurrente.

Experimentación con redes neuronales totalmente conectadas (*fully connected*)

Tras el modelo de arquitectura recurrente, probaremos ahora con una red neuronal totalmente conectada (*fully connected*). De nuevo, estos niveles se construyen secuencialmente y, en el caso específico de las totalmente conectadas, la gran mayoría de capas serán capas *Dense*. A continuación se muestra la estructura del modelo para los *datasets* ACPF de ambas versiones concatenadas.

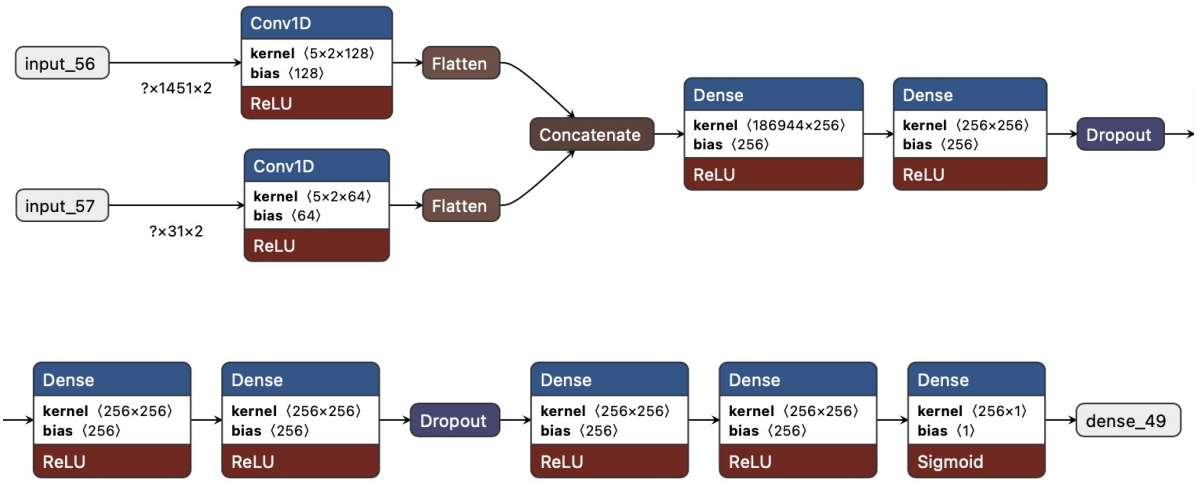


Figura 5.4: Esquema secuencial de las capas que configuran el modelo de arquitectura totalmente conectada en su versión combinada.

Como se puede apreciar en la Figura 5.4, el modelo presenta en primer lugar dos capas *input* en donde se introducen los datos de entrenamiento de ambos *datasets*. Para que la red pueda funcionar correctamente, se introducen primeramente dos capas convolucionales unidimensionales simples, de filtros 128 para las versiones globales y de 64 para las versiones locales. Tras ello se aplican don capas *Flatten* para reducir su dimensión y se combinan con una capa *Concatenate*. Una vez combinadas, se aplican las capas totalmente conectadas. En nuestro caso, aplicamos seis capas *Dense* de 256 unidades cada una sobre las que se van intercalando capas *Dropout* de 0.15 para evitar el sobreajuste. Finalmente, se concluye con una última capa *Dense* de 1 unidad con función de activación sigmoidal que, de nuevo, devuelve la probabilidad entre 0 y 1 de que cada una de las curvas introducidas coincida o no con el tránsito de un exoplaneta.

Los modelos tardaron 160 minutos de media en ejecutarse y alcanzaron como máximo en validación un *accuracy* del 87.45 %. La Tabla 5.12 muestra sobre datos test los resultados de los modelos para cada una de las versiones.

Modelo *fully connected*	***Accuracy***	***Recall***	***Precision***	**AUC**
Versiones globales	85.09 %	83.67 %	86.89 %	0.8663
Versiones locales	86.34 %	86.12 %	86.12 %	0.8701
Versiones combinadas	89.41 %	97.59 %	83.78 %	0.9005

Tabla 5.12: Tabla resumen de los resultados sobre datos test para los modelos de arquitectura totalmente conectada.

Experimentación con redes neuronales convolucionales unidimensionales (CNN 1D)

En esta última sección se aplica sobre los mismos sets de datos un modelo de red neuronal convolucional unidimensional (CNN 1D). El modelo sigue la misma estructura secuencial que en los casos anteriores, por lo que, de nuevo, se emplearán combinados los *datasets* de las versiones globales y locales resultado del ACPF.

Como se puede apreciar en la Figura 5.5, el modelo presenta en primer lugar dos capas *input* en donde se introducen los datos de entrenamiento de ambos *datasets*. Tras ello, comienzan a intercalarse capas convolucionales unidimensionales con capas *MaxPooling* para reducir la dimensionalidad de los datos. Para el caso de las capas convolucionales, todas ellas presentan un tamaño de *kernel* de 5 y filtros de 16, 32, 64 y 128 para las versiones globales; y de 16 y 32 para las locales.

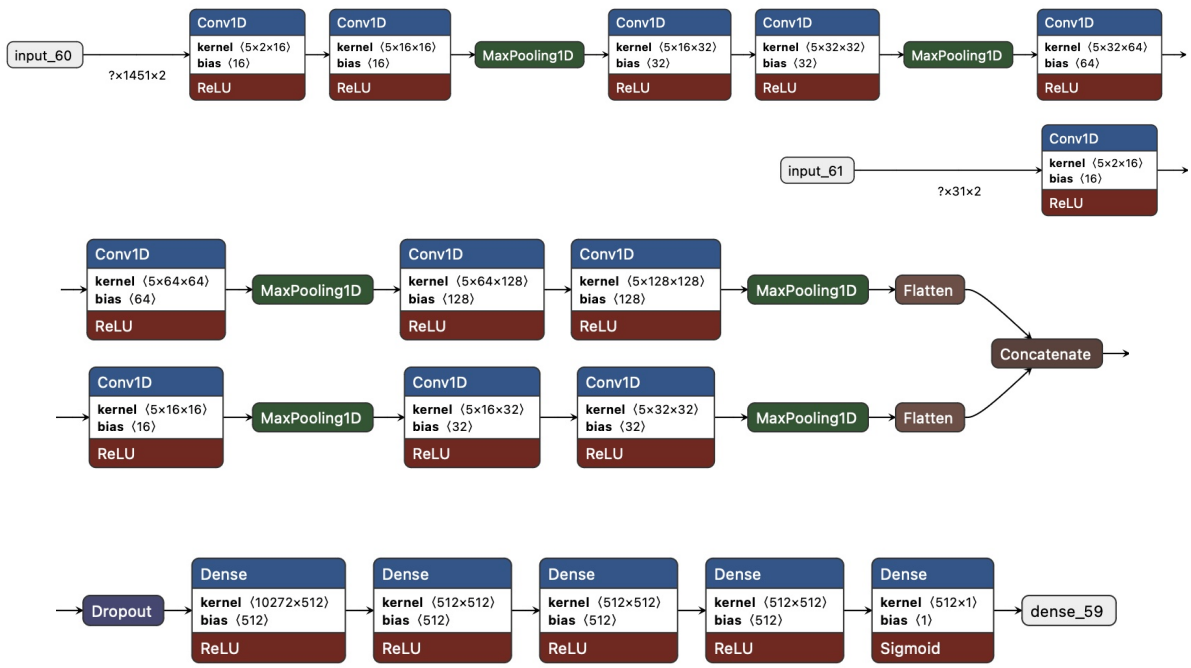


Figura 5.5: Esquema secuencial de las capas que configuran el modelo de arquitectura convolucional unidimensional en su versión combinada.

Tras las últimas capas *MaxPooling*, se aplican dos capas *Flatten* para uniformar las longitudes y se combinan con una capa *Concatenate*. Tras ello se aplica una capa *Dropout* de 0.15 y cuatro capas totalmente conectadas o *Dense* de 512 unidades.

Los modelos tardaron 20 minutos de media en ejecutarse y alcanzaron como máximo en validación un *accuracy* del 89.1 %. La Tabla 5.13 muestra sobre datos test los resultados de los modelos para cada una de las versiones.

Modelo CNN 1D	*Accuracy*	*Recall*	*Precision*	AUC
Versiones globales	88.12 %	88.43 %	86.23 %	0.8863
Versiones locales	89.24 %	87.67 %	87.43 %	0.8987
Versiones combinadas	91.56 %	89.07 %	89.32 %	0.9121

Tabla 5.13: Tabla resumen de los resultados sobre datos test para los modelos de arquitectura convolucional unidimensional.

5.4 Implementación con curvas de luz originales

En este segundo apartado de modelización emplearemos como datos de entrada el resultado del preprocesado aplicado a las curvas de luz originales expuesto en la Sección 4. Al igual que en el apartado anterior, disponemos de dos datasets: uno primero con las versiones globales de las curvas de luz (15737×2001), y uno segundo con las versiones locales (15737×201). Las Figuras 5.6 y 5.7 muestran un ejemplo del formato de los datos de entrada para ambas versiones.

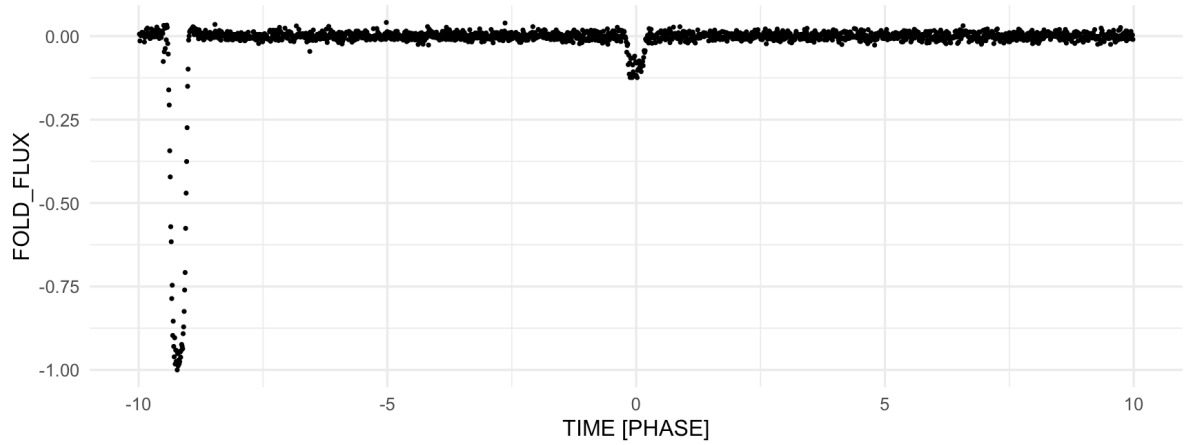


Figura 5.6: Representación de la versión global del segundo TCE de la estrella KIC 5130380 catalogado como AFP.

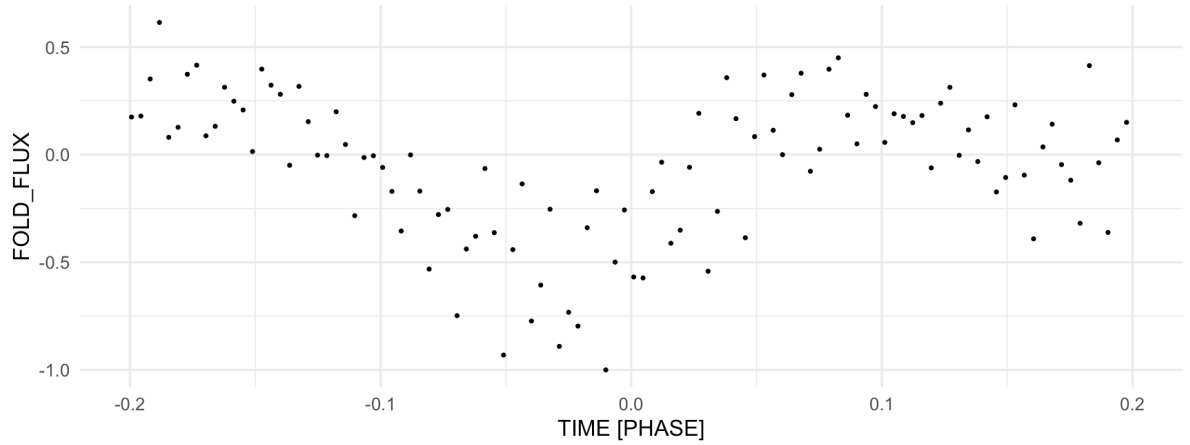


Figura 5.7: Representación de la versión local del segundo TCE de la estrella KIC 5002361 catalogado como NTP.

Experimentación con modelos clásicos

En esta sección se experimenta con modelos clásicos sobre los datos preprocesados de las curvas de luz originales. Se seguirá la misma estructura que en el apartado anterior. En primer lugar se experimenta con los *datasets* de las versiones globales para, posteriormente, probar con los de las versiones locales.

Para todos los modelos se han aplicado las mismas funciones, los mismos *grids*, y se han tuneado los mismos hiperparámetros expuestos en el Apartado 5.3. Nos remitimos a esa sección del TFM para más detalles acerca de la configuración del *grid* de cada uno de los siguientes modelos.

Regresión Logística

Con el *dataset* preprocesado de las versiones globales originales, el modelo tardó 2 minutos y 23 segundos en ejecutarse, y selección para `C` y `max_iter` los parámetros 0.01 y 100, respectivamente. Sobre validación, el *accuracy* alcanzó el 84.55 %.

En segundo lugar, con el *dataset* preprocesado de las versiones locales originales, el modelo tardó 1 minutos y 14 segundos en ejecutarse, y selección para `C` y `max_iter` los parámetros 0.1 y 50, respectivamente. Sobre validación, el *accuracy* alcanzó el 83.64 %. La Tabla 5.14 muestra sobre datos test los resultados de ambos modelos.

Modelo Regresión Logística	***Accuracy***	***Recall***	***Precision***	**AUC**
Versiones globales	85.27 %	83.69 %	87.41 %	0.8528
Versiones locales	84.81 %	83.30 %	86.85 %	0.8482

Tabla 5.14: Tabla resumen de los resultados sobre datos test para los modelos de Regresión Logística.

K-Nearest Neighbors

Con el *dataset* preprocesado de las versiones globales originales, el modelo tardó 223 minutos y 54 segundos en ejecutarse, y seleccionó para `n_neighbors` un valor de 87, para `weights` el parámetro `'uniform'`, para `metric` la opción `manhattan`, y para `p` el valor 1. Sobre validación, el *accuracy* alcanzó el 86.77 %.

En segundo lugar, con el *dataset* preprocesado de las versiones locales originales, el modelo tardó 102 minutos y 65 segundos en ejecutarse, y seleccionó para `n_neighbors` el valor 25, para

`weights` el parámetro `'distance'`, para `metric` la opción `euclidean`, y para `p` el valor 1. Sobre validación, el *accuracy* alcanzó el 85.58 %. La Tabla 5.15 muestra sobre datos test los resultados de ambos modelos.

Modelo *K-Nearest Neighbors*	*Accuracy*	*Recall*	*Precision*	AUC
Versiones globales	86.31 %	89.21 %	83.28 %	0.8656
Versiones locales	85.64 %	88.71 %	81.45 %	0.8561

Tabla 5.15: Tabla resumen de los resultados sobre datos test para los modelos *K-Nearest Neighbors*.

Gradient Boosting

Con el *dataset* preprocesado de las versiones globales originales, el modelo tardó 459 minutos y 21 segundos en ejecutarse, y seleccionó para `n_estimators` un valor de 400, para `learning_rate` un valor de 0.1, y para `max_depth` el valor 5. Sobre validación, el *accuracy* alcanzó el 91.19 %.

En segundo lugar, con el *dataset* preprocesado de las versiones locales originales, el modelo tardó 321 minutos y 67 segundos en ejecutarse, y seleccionó para `n_estimators` un valor de 200, para `learning_rate` un valor de 0.1, y para `max_depth` el valor 5. Sobre validación, el *accuracy* alcanzó el 89.92 %. La Tabla 5.16 muestra sobre datos test los resultados de ambos modelos.

Modelo *Gradient Boosting*	*Accuracy*	*Recall*	*Precision*	AUC
Versiones globales	91.53 %	89.30 %	94.26 %	0.9154
Versiones locales	90.52 %	88.54 %	92.96 %	0.9053

Tabla 5.16: Tabla resumen de los resultados sobre datos test para los modelos *Gradient Boosting*.

Random Forest

Con el *dataset* preprocesado de las versiones globales originales, el modelo tardó 376 minutos y 12 segundos en ejecutarse, y seleccionó para `n_estimators` un valor de 200, para `max_depth` un valor de 10, y para `min_samples_split` el valor 5. Sobre validación, el *accuracy* alcanzó el 89.25 %.

En segundo lugar, con el *dataset* preprocesado de las versiones locales originales, el modelo tardó 202 minutos y 13 segundos en ejecutarse, y seleccionó para `n_estimators` un valor de 400, para

`max_depth` un valor de 20, y para `min_samples_split` el valor 5. Sobre validación, el *accuracy* alcanzó el 90.15 %. La Tabla 5.17 muestra sobre datos test los resultados de ambos modelos.

Modelo *Random Forest*	***Accuracy***	***Recall***	***Precision***	**AUC**
Versiones globales	89.04 %	89.49 %	88.33 %	0.8904
Versiones locales	90.88 %	88.35 %	94.07 %	0.9090

Tabla 5.17: Tabla resumen de los resultados sobre datos test para los modelos *Random Forest*.

Modelos SVM

Con el *dataset* preprocesado de las versiones globales originales, el modelo tardó 105 minutos y 16 segundos en ejecutarse, y seleccionó para `C`, `gamma` y `kernel` los parámetros 1.2, 0.001 y `rbf`, respectivamente. Sobre validación, el *accuracy* alcanzó el 86.48 %.

En segundo lugar, con el *dataset* preprocesado de las versiones locales originales, el modelo tardó 78 minutos y 89 segundos en ejecutarse, y seleccionó para `C`, `gamma` y `kernel` los parámetros 1.5, 0.1 y `rbf`, respectivamente. Sobre validación, el *accuracy* alcanzó el 89.56 %. La Tabla 5.18 muestra sobre datos test los resultados de ambos modelos.

Modelo SVM	***Accuracy***	***Recall***	***Precision***	**AUC**
Versiones globales	87.20 %	84.16 %	91.48 %	0.8722
Versiones locales	89.87 %	88.67 %	91.30 %	0.8988

Tabla 5.18: Tabla resumen de los resultados sobre datos test para los modelos SVM.

Experimentación con redes neuronales recurrentes (RNN)

En esta sección se aplica sobre los mismos sets de datos un modelo de red neuronal recurrente (RNN). El método de construcción secuencial del modelo es el mismo que en apartados anteriores, aunque las capas hayan sufrido sutiles modificaciones debido a la mayor complejidad de estos *datasets* respecto de los anteriores. A continuación se muestra la estructura del modelo de ambas versiones concatenadas.

Como se puede observar en la Figura 5.8, el modelo presenta en primer lugar dos capas *input* en donde se introducen los datos de entrenamiento de ambos *datasets*. Para ambas versiones se dispone un modelo recurrente de tres capas LSTM de 16, 32 y 64 unidades. Ambas construcciones

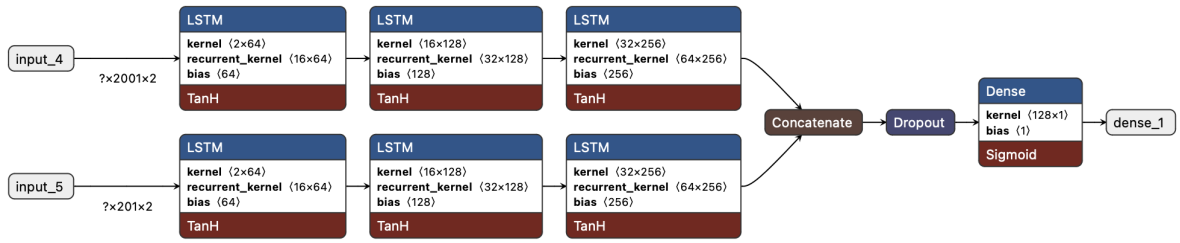


Figura 5.8: Esquema secuencial de las capas que configuran el modelo de arquitectura recurrente en su versión combinada.

se unen a través de una capa *Concatenate*, se aplica una capa *Dropout* de 0.25 para controlar el sobreajuste, y se concluye con una capa *Dense* con función de activación sigmoidal que devuelve la probabilidad entre 0 y 1 de que cada una de las curvas introducidas coincida o no con el tránsito de un exoplaneta.

Los modelos tardaron 31 minutos de media en ejecutarse y alcanzaron como máximo en validación un *accuracy* del 79.04 %. La Tabla 5.19 muestra sobre datos test los resultados de los modelos para cada una de las versiones.

Modelo RNN	*Accuracy*	*Recall*	*Precision*	AUC
Versiones globales	72.63 %	72.59 %	63.63 %	0.7203
Versiones locales	70.59 %	71.26 %	61.54 %	0.7034
Versiones combinadas	60.03 %	47.22 %	63.12 %	0.6345

Tabla 5.19: Tabla resumen de los resultados sobre datos test para los modelos de arquitectura recurrente.

Experimentación con redes neuronales totalmente conectadas (*fully connected*)

Tras el modelo de arquitectura recurrente, continuaremos con una red de arquitectura totalmente conectada (*fully connected*). A continuación se muestra la estructura del modelo, ligeramente diferente a la generada para el *dataset* del ACPF.

Como se puede apreciar en la Figura 5.9, el modelo presenta en primer lugar dos capas *input* en donde se introducen los datos de entrenamiento de ambos *datasets*. Para que la red pueda funcionar correctamente, se introducen primeramente dos capas convolucionales unidimensionales simples, de filtros 128 para las versiones globales y de 64 para las versiones locales. Tras ello se aplican

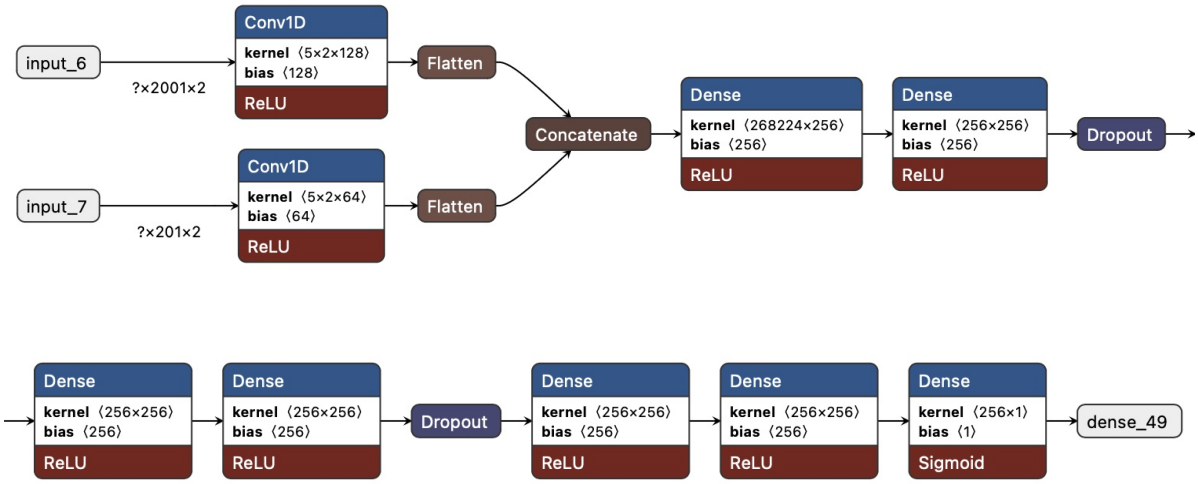


Figura 5.9: Esquema secuencial de las capas que configuran el modelo de arquitectura totalmente conectada en su versión combinada.

don capas *Flatten* para reducir su dimensión y se combinan con una capa *Concatenate*. Una vez combinadas, se aplican las capas totalmente conectadas. En nuestro caso, aplicamos seis capas *Dense* de 256 unidades cada una sobre las que se van intercalando capas *Dropout* de 0.15 para evitar el sobreajuste. Finalmente, se concluye con una última capa *Dense* de 1 unidad con función de activación sigmoidal que, de nuevo, devuelve la probabilidad entre 0 y 1 de que cada una de las curvas introducidas coincida o no con el tránsito de un exoplaneta.

Los modelos tardaron 140 minutos de media en ejecutarse y alcanzaron como máximo en validación un *accuracy* del 92.5 %. La Tabla 5.20 muestra sobre datos test los resultados de los modelos para cada una de las versiones.

Modelo *fully connected*	*Accuracy*	*Recall*	*Precision*	AUC
Versiones globales	85.09 %	83.67 %	86.89 %	0.8663
Versiones locales	86.34 %	86.12 %	86.12 %	0.8701
Versiones combinadas	91.89 %	90.34 %	91.45 %	0.9197

Tabla 5.20: Tabla resumen de los resultados sobre datos test para los modelos de arquitectura totalmente conectada.

Experimentación con redes neuronales convolucionales unidimensionales (CNN 1D)

En esta última sección se aplica sobre los sets de datos preprocesados un modelo de red neuronal convolucional unidimensional (CNN 1D). A continuación se muestra la estructura del modelo, diferente a la generada para el *dataset* del ACPF debido a la complejidad de los datos.

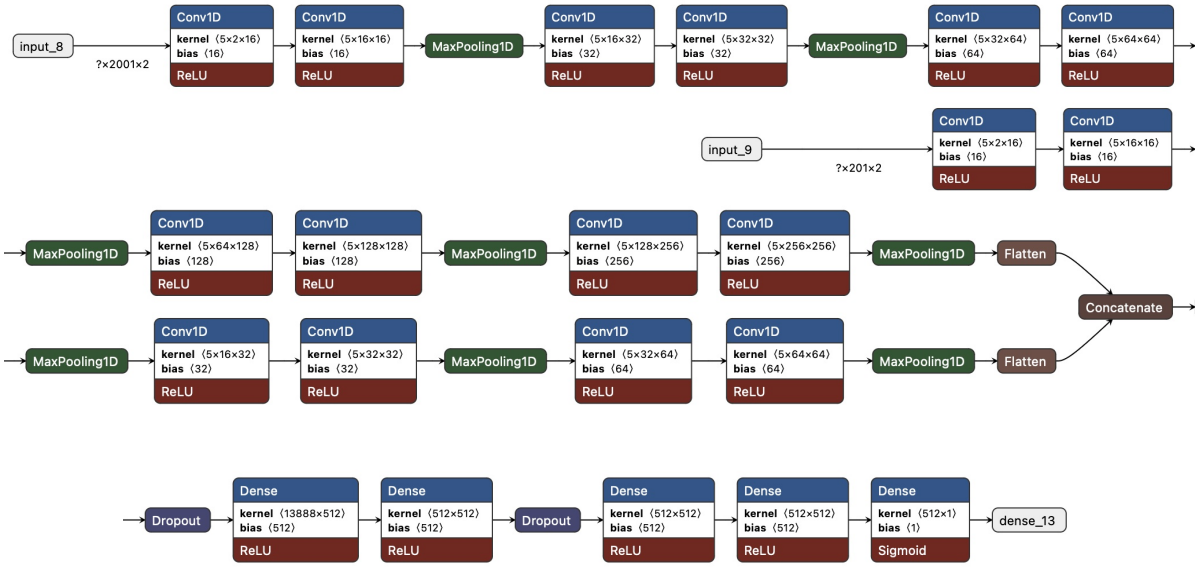


Figura 5.10: Esquema secuencial de las capas que configuran el modelo de arquitectura convolucional unidimensional en su versión combinada.

Como se puede apreciar en la Figura 5.10, el modelo presenta en primer lugar dos capas *input* en donde se introducen los datos de entrenamiento de ambos *datasets*. Tras ello, comienzan a intercalarse capas convolucionales unidimensionales con capas *MaxPooling* para reducir la dimensionalidad de los datos. Para el caso de las capas convolucionales, todas ellas presentan un tamaño de *kernel* de 5 y filtros de 16, 32, 64 y 128 para las versiones globales; y de 16, 32 y 64 para las locales. Tras las últimas capas *MaxPooling*, se aplican dos capas *Flatten* para uniformar las longitudes y se combinan con una capa *Concatenate*. Tras ello se aplican dos capas *Dropout* de 0.15 y de 0.25, respectivamente, y cuatro capas totalmente conectadas o *Dense* de 512 unidades. Se finaliza con la capa *Dense* de 1 unidad con función de activación sigmoidal al igual que en anteriores modelos.

Los modelos tardaron 25 minutos de media en ejecutarse y alcanzaron como máximo en validación un *accuracy* del 95.71 %. La Tabla 5.21 muestra sobre datos test los resultados de los modelos para cada una de las versiones.

Modelo CNN 1D	*Accuracy*	*Recall*	*Precision*	AUC
Versiones globales	89.24 %	87.67 %	87.43 %	0.8987
Versiones locales	92.01 %	91.56 %	93.71 %	0.9321
Versiones combinadas	95.12 %	93.56 %	92.65 %	0.9687

Tabla 5.21: Tabla resumen de los resultados sobre datos test para los modelos de arquitectura convolucional unidimensional.

5.5 Comparativa y selección del mejor modelo

En esta última sección se comparan los modelos expuestos y se selecciona el que mejores resultados ofrezca de acuerdo a una serie de criterios previamente especificados. En primer lugar compararemos los resultados de los modelos tradicionales para posteriormente hacer lo propio con los modelos de arquitectura neuronal.

Los criterios que tomaremos en cuenta a la hora de seleccionar el modelo final serán (1) el rendimiento, (2) la complejidad y (3) el tiempo. El rendimiento entendido como la capacidad de un modelo de hacer predicciones precisas sobre datos nuevos; la complejidad como el número de parámetros que presenta un modelo; y el tiempo como el tiempo de cómputo necesario para entrenar el modelo y efectuar predicciones.

Comparativa entre modelos tradicionales

En primer lugar haremos una comparativa entre todos los modelos tradicionales implementados en este trabajo. La comparativa se hará de acuerdo con los criterios expuestos en los párrafos anteriores, teniendo en cuenta la dicotomía sesgo-varianza. En la Tabla 5.22 se muestran los resultados generales de todos los modelos tradicionales y de todos los *datasets* expuestos en secciones anteriores ordenados en función de su AUC.

Si tomamos el AUC y las métricas de rendimiento como único criterio, la selección final quedaría disputada entre los modelos *Random Forest* y *Gradient Boosting*, con métricas de AUC que rondan el 0.91. Además, las versiones que mejores resultados ofrecen son las globales, en las que se podía apreciar una mayor amplitud en los extremos de las curvas.

Por otro lado, si tomamos en cuenta la complejidad del modelo en cuanto a estructura y datos de entrada, el modelo *Random Forest* ACPF sería el ganador. Estos modelos presentan menos parámetros que los modelos *Gradient Boosting* y, además, sus datos de entrada están reducidos en cuanto a dimensión. De 2001 un variables que presenta la versión original, la versión del ACPF cuenta únicamente con 1456 componentes.

En lo que respecta al tiempo de ejecución en la fase de entrenamiento, el modelo *Random Forest* sale ganando nuevamente, con 203 minutos y 31 segundos respecto de los 332 minutos y 56 segundos del modelo *Gradient Boosting*.

Modelo y *dataset*	Mejor versión	Tiempo *Train*	*Accuracy* Val.	*Accuracy* Test	*Recall* Test	*Precision* Test	AUC Test
Gradient Boost. Original	Global	459.21 mins.	91.20 %	91.53 %	89.30 %	94.26 %	0.9154
Random Forest ACPF	Global	203.31 mins.	89.32 %	90.87 %	88.97 %	92.07 %	0.9101
Random Forest Original	Local	202.13 mins.	89.25 %	90.88 %	88.35 %	94.07 %	0.9090
Gradient Boost. ACPF	Global	332.56 mins.	90.12 %	90.25 %	90.07 %	89.32 %	0.9002
SVM Radial Original	Local	78.89 mins.	86.48 %	89.87 %	88.67 %	91.30 %	0.8988
SVM Radial ACPF	Global	98.78 mins.	87.65 %	89.78 %	87.57 %	87.41 %	0.8979
KNN ACPF	Global	115.21 mins.	90.29 %	89.78 %	86.29 %	94.44 %	0.8980
KNN Original	Global	102.65 mins.	85.32 %	86.31 %	89.21 %	83.28 %	0.8656
Regresión Log. Original	Global	2.23 mins.	84.56 %	85.27 %	83.69 %	87.41 %	0.8528
Regresión Log. ACPF	Global	1.13 mins.	84.14 %	82.32 %	82.46 %	81.85 %	0.8232

Tabla 5.22: Tabla resumen de los modelos tradicionales con mejores resultados ordenada según AUC.

Otra cuestión adicional que conviene considerar a la hora de seleccionar el modelo final es la dicotomía clásica sesgo-varianza. En la Figura 5.11 se puede apreciar la comparativa entre modelos según su tasa de error y su variabilidad. Como se puede observar, el modelo *Random Forest* con el *dataset* ACPF se encuentra en segunda posición, muy cercana al modelo *Gradient Boosting*. Debido a su menor complejidad, tanto en datos de entrada como en tiempo de ejecución, nos decantaremos finalmente por esta versión del modelo.

Por tanto, el modelo tradicional seleccionado es el *Random Forest* con el *dataset* resultado del Análisis de Componentes Principales Funcionales con las siguientes métricas sobre datos test. *Acc.*: 90.87 %, *Recall*: 88.97 %; *Prec.*: 92.07 %; y AUC: 0.9101.

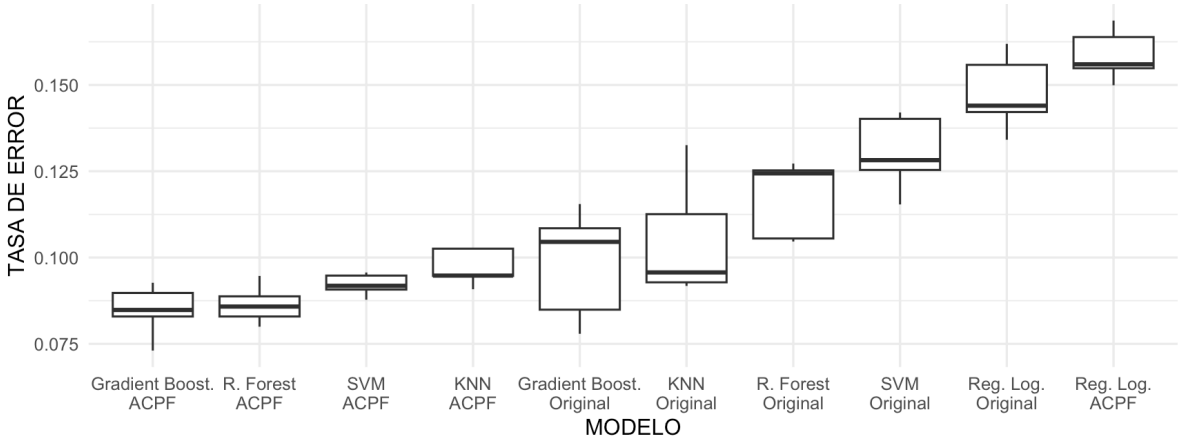


Figura 5.11: Distribución de la tasa de error según los modelos tradicionales finales.

Comparativa entre redes neuronales

En segundo lugar volveremos a replicar el mismo proceso pero con los modelos de arquitectura neuronal implementados en este trabajo. De nuevo, la comparativa se hará de acuerdo con los criterios expuestos en las primeras líneas de esta sección. En la Tabla 5.23 se muestran los resultados generales de todos los modelos de arquitectura neuronal ordenados en función de su AUC.

Modelo y *dataset*	**Mejor versión**	**Parámetros**	**Tiempo *Train***	***Accuracy* Test**	***Recall* Test**	***Precision* Test**	**AUC Test**
CNN 1D Original	Combin.	8 594 945	25.56 mins.	95.12 %	93.56 %	92.65 %	0.9687
Fully Connect. Original	Combin.	68 996 929	25.5 mins.	91.89 %	90.34 %	91.45 %	0.9197
CNN 1D ACPF	Combin.	6 220 673	16.26 mins.	91.56 %	89.07 %	89.32 %	0.9121
Fully Connect. ACPF	Combin.	48 189 249	175.43 mins.	89.41 %	97.59 %	83.78 %	0.9005
RNN ACPF	Combin.	39 905	18.56 mins.	73.43 %	72.23 %	73.67 %	0.7325
RNN Original	Global	32 385	38.56 mins.	72.63 %	72.59 %	63.63 %	0.7203

Tabla 5.23: Tabla resumen de los modelos de arquitectura neuronal con mejores resultados ordenada según AUC.

En este caso, si tomamos el AUC y las métricas de rendimiento como único criterio, la elección final sería la de la red neuronal convolucional unidimensional con el *dataset* global y en su versión

combinada. Los resultados en cuanto a rendimiento son muy positivos, alcanzando un AUC del 0.9687.

Por otro lado, si tomamos en cuenta los criterios de complejidad y tiempo, la elección ya estaría dividida entre las dos versiones de red neuronal convolucional de que disponemos. La versión con el *dataset* del ACPF es menos compleja y presenta un menor número de parámetros, pero las diferencias en cuanto a rendimiento son bastante significativas.

Por tanto, el modelo de arquitectura neuronal que seleccionaremos finalmente es el CNN 1D con el *dataset* preprocesado sin ACPF. Sus métricas sobre datos test son las siguientes. *Acc.*: 95.12 %, *Recall*: 93.56 %; *Prec.*: 92.65 %; y AUC: 0.9687.

Selección final

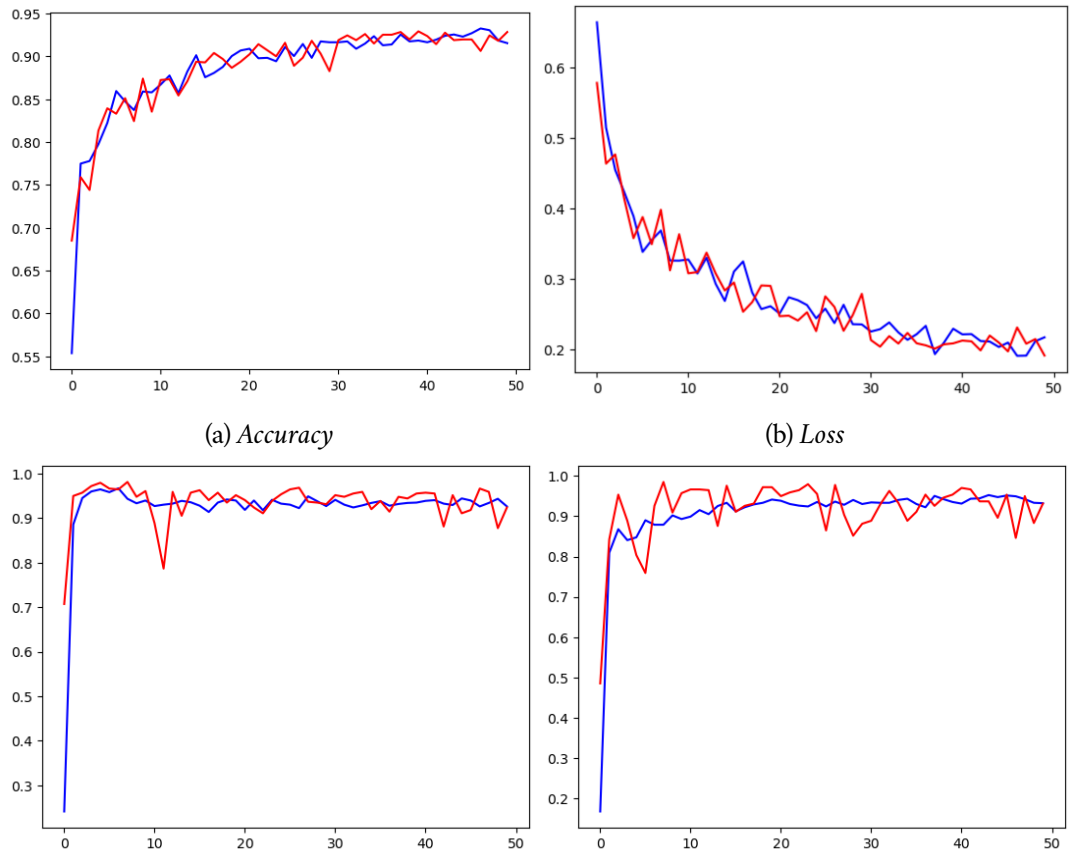
En esta última sección se muestran algunos gráficos y métricas del modelo finalmente seleccionado. Como ya se ha comentado, tenemos dos candidatos:

- ***Random Forest* - Set ACPF**. *Acc.*: 90.87 %, *Recall*: 88.97 %; *Prec.*: 92.07 %; y AUC: 0.9101.
- **CNN 1D - Set original**. *Acc.*: 95.12 %, *Recall*: 93.56 %; *Prec.*: 92.65 %; y AUC: 0.9687.

Dadas las mejoras en cuanto a rendimiento, seleccionaremos finalmente la CNN 1D sobre el set de datos preprocesado. Las mejoras en cuanto al porcentaje de positivos correctamente clasificados son bastante significativas, por lo que nos decantaremos finalmente por la red.

En la Figura 5.12 se muestran las gráficas extraídas durante la fase de entrenamiento del modelo. La línea azul grafica la mejora de los resultados durante el entrenamiento, y la línea roja durante la fase de validación. Ambas se encuentran muy igualadas, lo que indica que el modelo está bien ajustado sin indicios de sobreentrenamiento.

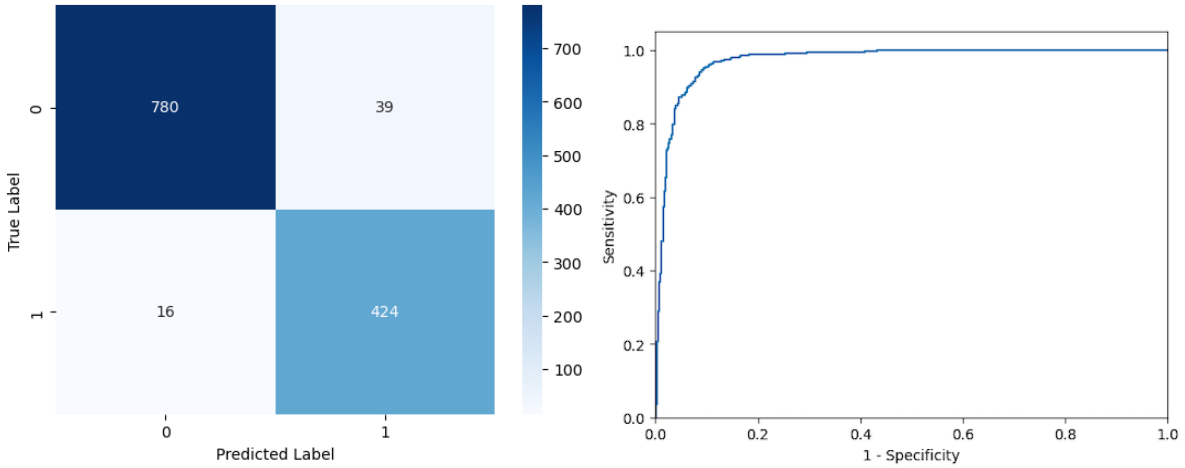
Por otra parte, en la Figura 5.13 se puede visualizar la matriz de confusión del modelo seleccionado y su curva ROC. De 440 tránsitos de exoplanetas que había en el conjunto test, el modelo ha clasificado correctamente 424. Por otro lado, de 819 tránsitos sin exoplanetas identificados, el modelo ha clasificado correctamente 780. Se ha considerado un punto de *crossover* sensibilidad-especificidad de 0.74, en donde se sacrifica especificidad a costa de incrementar el número de muestras de la clase positiva bien clasificadas.



(a) *Accuracy* (b) *Loss*

(c) *Precision* (d) *Recall*

Figura 5.12: Métricas del modelo seleccionado durante las fases de entrenamiento y validación de los datos.



(a) Matriz de confusión (b) Curva ROC

Figura 5.13: Matriz de confusión y curva ROC del modelo seleccionado.

6. Interpretabilidad

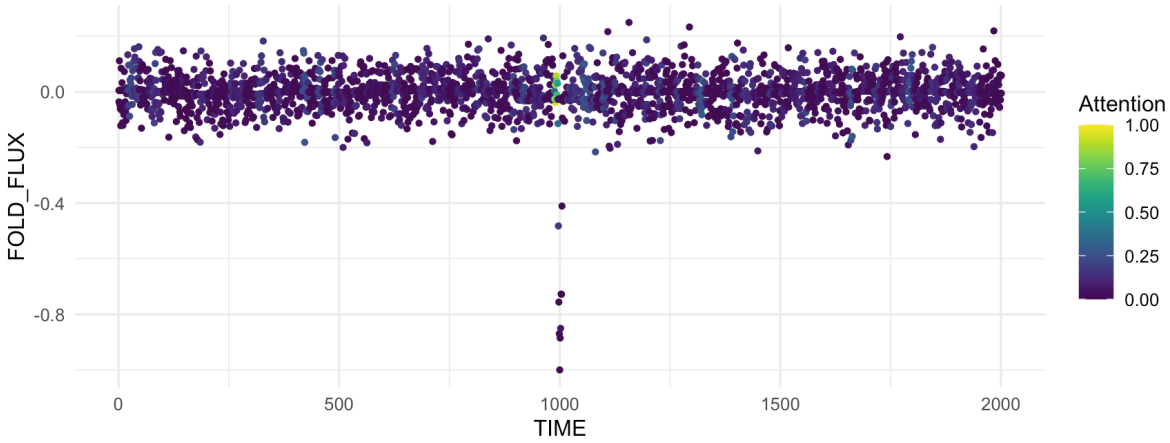
Uno de los grandes problemas de los modelos construidos a partir de redes neuronales es su opacidad, es decir, la dificultad por conocer los motivos que llevan a los nodos de una red a tomar una decisión determinada. De hecho, a estos modelos se los clasifica usualmente como algoritmos de «caja negra», en donde se puede observar claramente qué entra y qué sale de la red mientras que el proceso intermedio permanece indescifrado. En este sentido, el objetivo de esta sección es tratar de comprender a través de mapas de calor cómo las capas del algoritmo memorizan determinados patrones para poder llegar de los datos de entrada a las predicciones de salida.

Para este propósito emplearemos Grad-CAM (*Gradient-weighted Class Activation Mapping*) [26], una técnica utilizada en *Deep Learning* para comprender qué partes de una imagen son responsables de la activación de ciertos nodos en una red neuronal, es decir, para entender en qué sectores de la imagen se está fijando la red a la hora de efectuar una predicción. Para estos casos, Grad-CAM genera un mapa de calor sobre la imagen original con el propósito de poder visualizar fácilmente las áreas de la imagen más relevantes para la red.

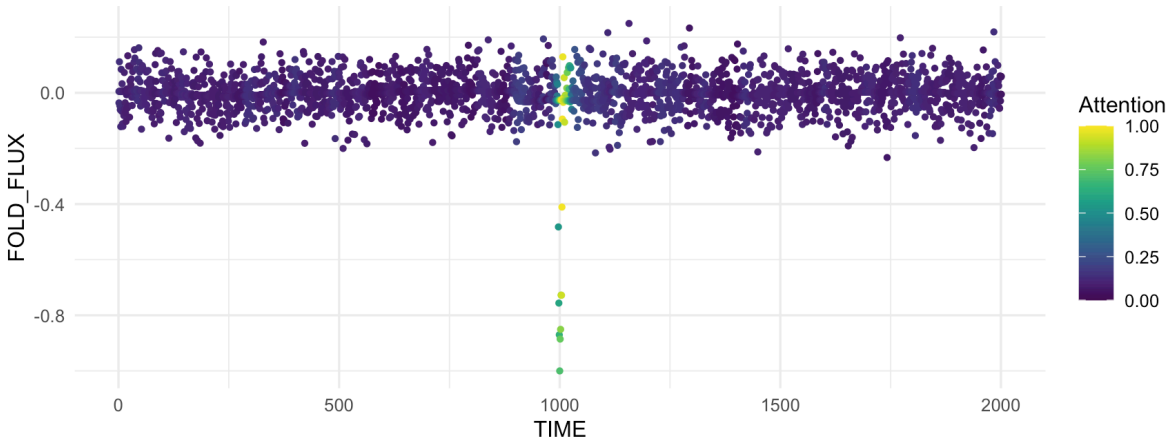
La técnica Grad-CAM también se puede adaptar para que genere mapas de calor sobre series unidimensionales –en nuestro caso, las curvas de luz preprocesadas– en lugar de sobre vectores bidimensionales propios de las imágenes. Al aplicar la técnica a la curva es posible obtener un «mapa de atención» de la red que muestre qué partes de la curva de luz contribuyen más a la decisión del algoritmo a clasificarla como exoplaneta o no. Es esperable, además, que estas partes coincidan con las zonas en donde se producen los posibles tránsitos, ya que son las más diferenciables.

En concreto, Grad-CAM se aplicará sobre las diferentes capas convolucionales de nuestra red CNN 1D para que examine la información de gradiente que fluye hacia esa misma capa. Si analizamos la importancia de los gradientes de cada una de las capas que queramos analizar, podremos compararlos en función de su tamaño y ponderar su relevancia para la red. Si los mapas de activación de la capa se han accionado en una región concreta de la curva y si sus gradientes son elevados en esa zona, ello implica que ese área de la curva será muy significativa a la hora de re-

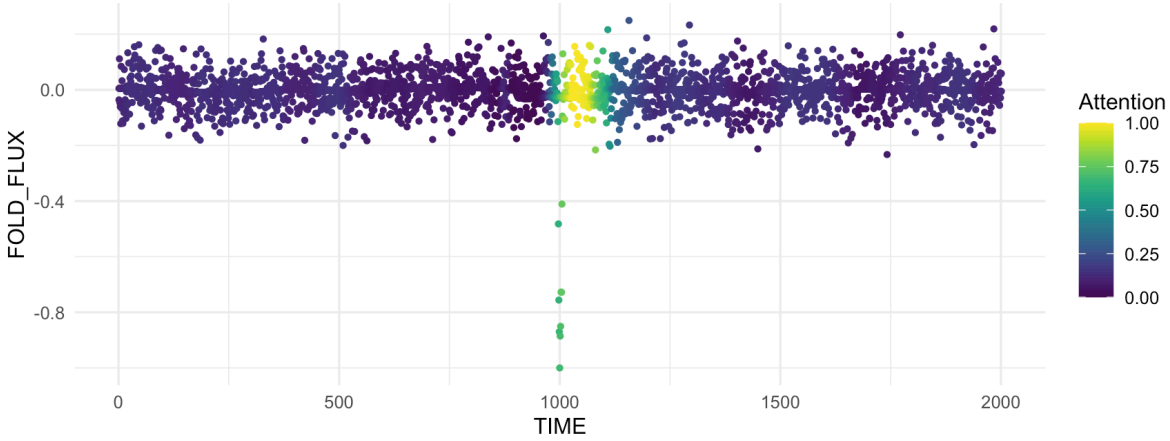
tornar la predicción. Toda esta información se puede visualizar como si de imágenes se tratara a través de la creación de mapas de calor sobre las curvas de luz originales.



(a) Mapa de calor del grado de atención para la cuarta capa convolucional de la red CNN 1D seleccionada.



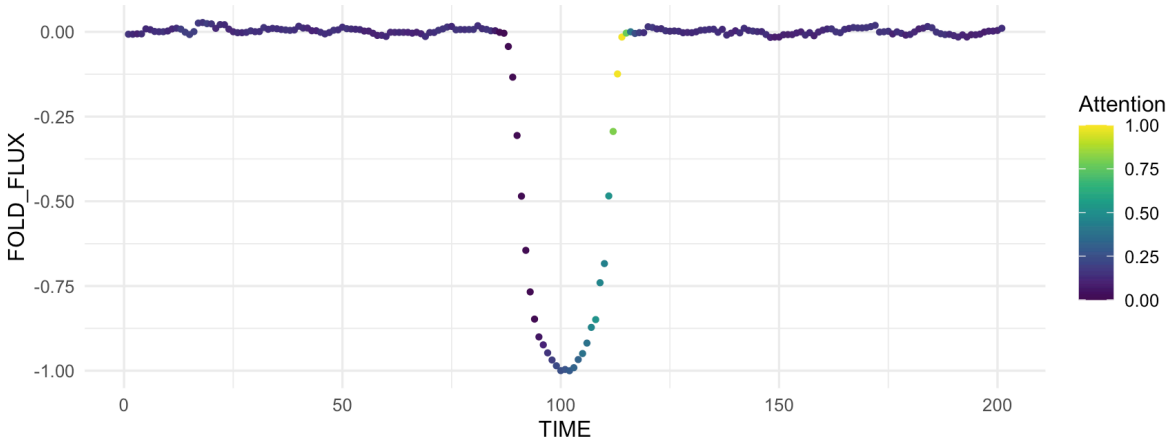
(b) Mapa de calor del grado de atención para la séptima capa convolucional de la red CNN 1D seleccionada.



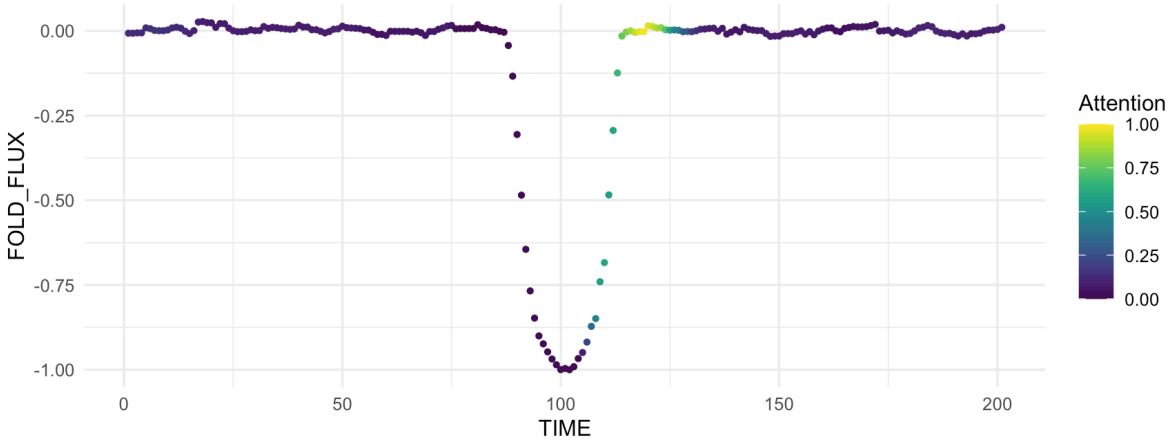
(c) Mapa de calor del grado de atención para la décima capa convolucional de la red CNN 1D seleccionada.

Figura 6.1: Aplicación gradual de Grad-CAM sobre la versión global de la curva de luz de la estrella KIC 9946525 catalogada como PC.

En la Figura 6.1 se puede observar el funcionamiento de Grad-CAM sobre la versión global de la estrella KIC 9946525, catalogada originalmente como *Planet Candidate*. Se puede apreciar cómo con el paso de las capas la atención de la red se va focalizando cada vez con mayor intensidad en el tránsito del posible exoplaneta. La Figura 6.1a representa el grado de atención de la cuarta capa, una capa convolucional unidimensional de 32 filtros. En ella ya se pueden apreciar indicios de mayor intensidad en el centro de la curva. Por su parte, las Figuras 6.1b y 6.1c se corresponden con el grado de atención de las capas séptima y décima (la última de la red). Son, respectivamente, capas convolucionales de 128 y 256 filtros, y gracias a la información de los gradientes de capas anteriores, el tránsito de la curva se encuentra plenamente localizado por la red en su última capa.



(a) Mapa de calor del grado de atención para la segunda capa convolucional de la red CNN 1D seleccionada.



(b) Mapa de calor del grado de atención para la cuarta capa convolucional de la red CNN 1D seleccionada.

Figura 6.2: Aplicación gradual de Grad-CAM sobre la versión local de la curva de luz de la estrella KIC 9946525 catalogada como PC.

En la Figura 6.2 se aplica el mismo procedimiento, pero esta vez para la versión local de la curva de luz. La estructura neuronal del modelo generado para las versiones locales es más reducida porque la red no requiere de tantas capas para identificar patrones en una versión de la curva con un menor número de puntos de información. De esta manera, las Figuras 6.2a y 6.2b representan las capas segunda (16 filtros) y cuarta de la red (32 filtros), siendo está última el nivel final que conecta con el resto de capas *Dense*. En este caso la atención de las capas no parece focalizarse tanto en las zonas cercanas al mínimo del tránsito, sino en la parte creciente de la parábola.

7. Conclusiones y líneas de trabajo futuras

A modo de cierre, en esta última sección se presenta un resumen de las principales conclusiones alcanzadas durante la elaboración del TFM. Con ellas se pretende responder a los objetivos planteados en las primeras líneas de este trabajo, así como discutir el futuro de la ciencia de datos como disciplina auxiliar del campo de la detección e identificación de exoplanetas.

7.1 Conclusiones

El principal objetivo planteado al inicio del trabajo consistía en generar todo un proceso secuencial de descarga, preparación y modelización de los datos recopilados por el satélite artificial Kepler de acuerdo a los estándares de la metodología SEMMA. Todo este proceso debía culminar en un modelo que pudiese discernir con suficiente exactitud cuándo una curva de luz podría contener uno o varios tránsitos astronómicos consistentes con el paso de un exoplaneta. Consideramos que a través de las distintas secciones que componen este TFM el objetivo se ha alcanzado de manera progresiva. A continuación, revisitaremos los resultados alcanzados en cada una de ellas.

En primer lugar se llevó a cabo una investigación acerca del estado del arte en lo relativo al campo de la detección e identificación de exoplanetas, y en cómo las diferentes técnicas de *Machine Learning* han podido entrar a formar parte de la disciplina. De esta sección hemos podido rescatar los conocimientos necesarios para crear nuestros propios modelos, así como para poder construir una base sobre la cual comparar futuras experimentaciones. Además de ello, también se pudieron conocer los distintos métodos de descarga de las curvas de luz, métodos que aplicamos en la siguiente sección gracias a la publicidad de las bases de datos de la NASA Exoplanet Archive y del Mikulski Archive for Space Telescopes.

En la tercera sección se expuso el exhaustivo preprocesado al que se debieron someter las curvas de luz antes de la fase de modelización. Las fases que configuran la sección de preprocesado del presente trabajo son fruto de la combinación de diferentes perspectivas inmersas en la literatura, por lo que constituyen un *pipeline* propio, personalizado y adaptado al formato de entrada de las curvas de luz proveídas por la misión Kepler y sus posteriores expansiones. El mayor defecto presente en esta clase de preprocesados es que para poder limpiar la curva de las fluctuaciones naturales de la radiación estelar se requieren conocer previamente datos acerca de los propios tránsitos astronómicos, tales como su periodo orbital o su duración. Ello hace que misiones estelares más recientes, cuyos resultados no han sido examinados todavía por astrofísicos profesionales, no puedan someterse al *pipeline* generado en el presente trabajo, precisamente por no disponer de datos preliminares sobre los potenciales tránsitos presentes en las curvas de luz. Aún con todo, las misiones Kepler y su expansión K2 sí que son compatibles, y se espera que en un futuro muy cercano este preprocesado personalizado pueda adaptarse con facilidad a misiones más actuales, como TESS o Hubble.

En la cuarta sección se presentaron los modelos creados para el presente trabajo. La CNN 1D sobre el *dataset* original, el mejor modelo de los casi 40 implementados a lo largo del trabajo, ha alcanzado un *accuracy* del 95.12 %, llegando a competir con los resultados de algunos artículos del área publicados en los últimos años (Tabla 7.1). Además de los modelos de arquitectura neuronal, competitivos en el actual estado del arte, también se han implementado modelos tradicionales en ciencia de datos con resultados no superiores pero también bastante positivos. Este es el caso de los modelos *Gradient Boosting* y *Random Forest* para ambos tipos de *dataset*, en donde se han llegado a alcanzar en test cotas de *accuracy* cercanas al 90 %.

Modelo	**Accuracy**	**Recall**	**Precision**	**AUC**
CNN 1D - Astronet Shallue y Vanderburg (2018)	96 %	95 %	93 %	0.988
RNN Hinners et al. (2018)	92 %	92 %	90 %	0.937
CNN 1D Creación propia	95.12 %	93.56 %	92.65 %	0.9687

Tabla 7.1: Tabla comparativa entre el modelo seleccionado en este trabajo y los modelos de Shallue y Vanderburg (2018) y de Hinners et al. (2018).

En la Tabla 7.1 se puede apreciar la comparativa entre los resultados de los trabajos de Shallue y Vanderburg (2018) y de Hinners et al. (2018), en donde se utilizan redes neuronales convolucionales unidimensionales y redes neuronales recurrentes al igual que en este trabajo. Además, ambos trabajos aplican un preprocesado y una metodología muy similares.

Como el modelo finalmente seleccionado ha sido una red neuronal convolucional, en la siguiente sección se interpretaron las predicciones del modelo a través de la técnica Grad-CAM. En este apartado pudimos observar cómo las diferentes capas de la red se van focalizando progresivamente en aquellas zonas de las curvas que permiten identificar posibles tránsitos astrónomicos consistentes con la presencia de un exoplaneta en órbita. En este sentido, los mapas de calor son útiles no solo para la interpretación de curvas de luz con tránsitos candidatos a planeta, sino también para aquellas con falsos positivos astronómicos y con fenómenos no transitorios.

A modo de cierre, consideramos que se ha podido demostrado a lo largo del TFM el cumplimiento tanto del objetivo general, como de los objetivos parciales. Se ha llevado a cabo una amplia investigación sobre el estado del arte; se ha creado una exhaustiva fase de preprocesado adaptada específicamente a las peculiaridades de los datos de entrada; se han construido una gran cantidad de modelos, tanto tradicionales como de arquitectura neuronal, a partir de dos tipos diferentes de *dataset*; y, en último lugar, los resultados obtenidos han sido muy positivos, demostrando ser competitivos en el panorama actual de las aplicaciones del *Machine Learning* a la detección e identificación de exoplanetas.

7.2 Líneas de trabajo futuras

En este último apartado se proponen futuras aplicaciones de las técnicas presentadas en este TFM con el objeto de ampliar los resultados obtenidos. En este sentido, la principal línea de trabajo a futuro consistiría en aplicar la metodología y las fases de preprocesado presentadas a nuevos *datasets* de nuevas misiones todavía en curso con sectores de observación estelares distintos a los asignados para Kepler. Actualmente, gracias al Mikulski Archive for Space Telescopes, es posible acceder a los datos de misiones como K2 o TESS con la misma facilidad con la que se ha accedido a los de Kepler en este trabajo. Como ya se ha comentado con anterioridad, el gran problema que presentan estas ampliaciones es que no se dispone de un catálogo de tránsitos clasificados con la misma exhaustividad que con Kepler, en donde distintos grupos de investigación han revisado en numerosas ocasiones las curvas de luz proveídas por el satélite. Se espera que un futuro no muy lejano esto mismo ocurra con otras misiones más actuales, permitiendo extrapolar la metodología y las técnicas empleadas en este trabajo.

Otra posible línea de trabajo a futuro podría pasar por la creación de modelos de clasificación multiclase. En este trabajo se ha optado por seguir una aproximación binaria a la problemática, principalmente porque nuestro interés recaía en tratar de clasificar correctamente los tránsitos pertenecientes a potenciales exoplanetas. A pesar de ello, los códigos creados para los modelos presentados en este TFM requerirían tan solo de algunos ajustes menores para transformar dicha aproximación binaria en una clasificación multiclase (PC / AFP / NTP).

Bibliografía

[1] P. Drineas, R. Kannan y M. W. Mahoney, «Fast Monte Carlo algorithms for matrices I: Approximating matrix multiplication», *SIAM Journal on Computing*, vol. 36, nº 1, págs. 132-157, 2006.

[2] M. Rudelson y R. Vershynin, «Sampling from large matrices: An approach through geometric functional analysis», *Journal of the ACM (JACM)*, vol. 54, nº 4, 21-es, 2007.

[3] G. W. Marcy, R. P. Butler, E. Williams et al., «The planet around 51 Pegasi», *The Astrophysical Journal*, vol. 481, nº 2, pág. 926, 1997.

[4] R. Akeson, J. Christiansen, D. R. Ciardi, S. Ramirez, J. Schlieder, J. C. Van Eyken et al., «The nasa exoplanet archive», en *American Astronomical Society Meeting Abstracts*, vol. 229, 2017.

[5] W. J. Borucki, «KEPLER Mission: development and overview», *Reports on Progress in Physics*, vol. 79, nº 3, pág. 36, 2016.

[6] W. J. Borucki, D. G. Koch, G. Basri et al., «Characteristics of planetary candidates observed by Kepler. II. Analysis of the first four months of data», *The Astrophysical Journal*, vol. 736, nº 1, pág. 19, 2011.

[7] N. M. Batalha, W. J. Borucki, S. T. Bryson et al., «Kepler's first rocky planet: Kepler-10b», *The Astrophysical Journal*, vol. 729, nº 1, pág. 27, 2011.

[8] J. M. Jenkins, D. A. Caldwell, H. Chandrasekaran et al., «Overview of the Kepler science processing pipeline», *The Astrophysical Journal Letters*, vol. 713, nº 2, pág. 87, 2010.

[9] J. A. Dittmann, J. M. Irwin, D. Charbonneau et al., «A temperate rocky super-Earth transiting a nearby cool star», *Nature*, vol. 544, nº 7650, págs. 333-336, 2017.

[10] S. Millholland y G. Laughlin, «Supervised learning detection of sixty non-transiting hot Jupiter candidates», *The Astronomical Journal*, vol. 154, nº 3, pág. 83, 2017.

[11] D. Mislis, E. Bachelet, K. Alsubai, D. Bramich y N. Parley, «SIDRA: a blind algorithm for signal detection in photometric surveys», *Monthly Notices of the Royal Astronomical Society*, vol. 455, nº 1, págs. 626-633, 2016.

[12] M. A. for Space Telescopes, *Maximizing the scientific accessibility productivity of astronomical data*, 2019. dirección: https://cutt.ly/3weBUfKB.

[13] N. Aeronautics y S. Administration, *NASA Exoplanet Archive*, 2019. dirección: https://cutt.ly/RweBQ7Up.

[14] H. J. Deeg y R. Alonso, «Transit photometry as an exoplanet discovery method», *arXiv preprint arXiv:1803.07867*, 2018.

[15] J. T. Wright y B. S. Gaudi, «Exoplanet detection methods», *arXiv preprint arXiv:1210.2471*, 2012.

[16] B. S. Gaudi, «Microlensing surveys for exoplanets», *Annual Review of Astronomy and Astrophysics*, vol. 50, págs. 411-453, 2012.

[17] E. S. Agency, *How to find an exoplanet*, 2019. dirección: https://cutt.ly/3weBQCsa.

[18] C. Lovis, D. Fischer et al., «Radial velocity techniques for exoplanets», *Exoplanets*, págs. 27-53, 2010.

[19] B. Wohler, J. M. Jenkins, J. D. Twicken et al., «The Kepler Science Data Processing Pipeline Source Code Road Map», inf. téc., 2016.

[20] A. Dattilo, A. Vanderburg, C. J. Shallue et al., «Identifying exoplanets with deep learning. ii. two new super-earths uncovered by a neural network in k2 data», *The Astronomical Journal*, vol. 157, nº 5, pág. 169, 2019.

[21] C. J. Shallue y A. Vanderburg, «Identifying exoplanets with deep learning: A five-planet resonant chain around kepler-80 and an eighth planet around kepler-90», *The Astronomical Journal*, vol. 155, nº 2, pág. 94, 2018.

[22] A. Vanderburg y J. A. Johnson, «A technique for extracting highly precise photometry for the two-wheeled Kepler mission», *Publications of the Astronomical Society of the Pacific*, vol. 126, nº 944, págs. 948-958, 2014.

[23] G. A. Caceres, E. D. Feigelson, G. J. Babu et al., «AutoRegressive planet search: methodology», *The Astronomical Journal*, vol. 158, nº 2, pág. 57, 2019.

[24] T. A. Hinners, K. Tat y R. Thorp, «Machine learning techniques for stellar light curve classification», *The Astronomical Journal*, vol. 156, nº 1, pág. 7, 2018.

[25] R. C. Staudemeyer, «Applying long short-term memory recurrent neural networks to intrusion detection», *South African Computer Journal*, vol. 56, nº 1, págs. 136-154, 2015.

[26] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh y D. Batra, «Grad-cam: Visual explanations from deep networks via gradient-based localization», en *Proceedings of the IEEE international conference on computer vision*, 2017, págs. 618-626.

[27] J. D. Twicken, J. H. Catanzarite, B. D. Clarke et al., «Kepler Data Validation I—Architecture, Diagnostic Tests, and Data Products for Vetting Transiting Planet Candidates», *Publications of the Astronomical Society of the Pacific*, vol. 130, nº 988, pág. 64, 2018.

[28] S. E. Thompson, J. L. Coughlin, K. Hoffman et al., «Planetary candidates observed by Kepler. VIII. A fully automated catalog with measured completeness and reliability based on data release 25», *The Astrophysical Journal Supplement Series*, vol. 235, nº 2, pág. 38, 2018.

[29] D. J. Armstrong, D. Pollacco y A. Santerne, «Transit shapes and self organising maps as a tool for ranking planetary candidates: application to kepler and k2», *Monthly Notices of the Royal Astronomical Society*, stw2881, 2016.

[30] S. E. Thompson, F. Mullally, J. Coughlin et al., «A machine learning technique to identify transit shaped signals», *The Astrophysical Journal*, vol. 812, nº 1, pág. 46, 2015.

[31] S. D. McCauliff, J. M. Jenkins, J. Catanzarite et al., «Automatic classification of Kepler planetary transit candidates», *The Astrophysical Journal*, vol. 806, nº 1, pág. 6, 2015.

[32] K. A. Pearson, L. Palafox y C. A. Griffith, «Searching for exoplanets using artificial intelligence», *Monthly Notices of the Royal Astronomical Society*, vol. 474, nº 1, págs. 478-491, 2018.

[33] S. Albawi, T. A. Mohammed y S. Al-Zawi, «Understanding of a convolutional neural network», en *2017 international conference on engineering and technology (ICET)*, Ieee, 2017, págs. 1-6.