Universidad Internacional de La Rioja
Escuela Superior de Tecnología e Ingeniería

Máster Universitario en Astrofísica y Técnicas de Observación en Astronomía

Análisis de curvas de luz Kepler mediante la transformada *wavelet*

Trabajo fin de estudio presentado por:	Luis Guirado Fuentes
Tipo de trabajo:	Desarrollo de las tecnologías aplicadas a la astrofísica.
Línea de trabajo:	Análisis de imágenes astronómicas.
Director/a:	Roberto Baena Gallé
Fecha:	18 de septiembre de 2022

Resumen

La misión Kepler ha sido la más exitosa hasta ahora en la búsqueda y caracterización de exoplanetas mediante la técnica del tránsito. Con este método, se mide la intensidad de la luz emitida por la estrella a intervalos regulares para detectar reducciones fotométricas que se repiten periódicamente, a partir de las cuales se puede inferir la presencia de un objeto eclipsante.

La transformada *wavelet* se ha utilizado como alternativa a la transformada de Fourier en el filtrado de ruido en datos fotométricos astronómicos, así como en la detección de tránsitos de exoplanetas. Proponemos un nuevo enfoque basado en el uso de la transformada *wavelet* como herramienta matemática para establecer criterios estadísticos para la caracterización del objeto eclipsante, con el fin de diferenciar los exoplanetas de los falsos positivos, con el objetivo de que los resultados obtenidos puedan ser utilizados para entrenar un modelo de ML (Machine Learning) para analizar automáticamente miles de curvas de luz de las misiones Kepler y K2.

Palabras clave: exoplanetas, tránsitos, curvas de luz, *wavelet*, python

Abstract

The Kepler mission has been the most successful so far in the search for and characterization of exoplanets using the transit technique. With this method, the intensity of light emitted by the star is measured at regular intervals to detect periodically recurring photometric reductions in the star, from which the presence of an eclipsing object can be inferred.

The *wavelet* transform has been used as an alternative to the Fourier transform in noise filtering in astronomical photometric data, as well as in the detection of exoplanet transits. We propose a new approach based on the use of the *wavelet* transform as a mathematical tool to establish statistical criteria for the characterization of the eclipsing object, in order to differentiate exoplanets from false positives, with the aim that the results obtained can be used to train a ML (Machine Learning) model to automatically analyze thousands of light curves Kepler and K2 missions.

Keywords: exoplanets, transits, light curves, *wavelet*, python

Índice de contenidos

Índice de figuras

aprecia claramente el decaimiento de flujo en el periodo especificado.

Índice de tablas

1. Introducción

La investigación de planetas que orbitan una estrella diferente al Sol y no pertenecen al sistema solar, denominados exoplanetas, ha experimentado un gran crecimiento los últimos años debido a la gran cantidad de datos procedentes de misiones como Kepler, K2, CoRoT o TESS. La disponibilidad de estos datos para la comunidad científica y el desarrollo de las técnicas de *big data* e inteligencia artificial, ha permitido el descubrimiento y caracterización de más de 5000 exoplanetas (National Aeronautics and Space Administration [NASA], 2022), como se puede ver en la Tabla 1.

Tabla 1: *Exoplanetas confirmados, candidatos a TESS y candidatos a Kepler.*

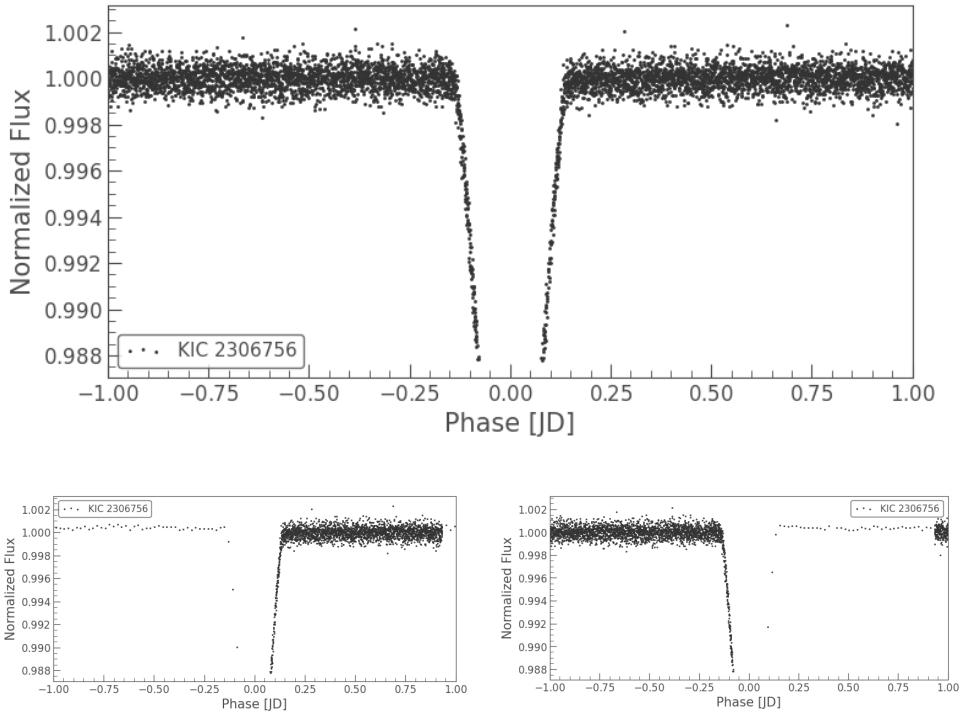
Total de exoplanetas	5014
Planetas confirmados descubiertos por Kepler	2709
Candidatos del proyecto Kepler aún por confirmar	2057
Planetas confirmados descubiertos por K2	537
Candidatos de K2 por confirmar	969
Planetas confirmados descubiertos por TESS	205
Candidatos del proyecto TESS integrados en el archivo (2022-04-21 13:00:02)	5637
Candidatos del proyecto TESS en ExoFOP	5637
Candidatos del proyecto TESS aún por confirmar	3829

Fuente: Adaptada de https://exoplanetarchive.ipac.caltech.edu/docs/counts_detail.html

En este proyecto se analiza la detección de exoplanetas mediante el procesamiento de curvas de luz, utilizando la técnica de tránsito, y se estudia la aplicación de la transformada *wavelet* como herramienta matemática que permita establecer un criterio estadístico para diferenciar exoplanetas de falsos positivos, ver Figura 1.

En los objetos observados por diferentes misiones, existen sistemas estelares cuyos tránsitos producen curvas de luz de características similares que pueden dar lugar a la detección de exoplanetas transitando estrellas, cuando en realidad se trata de sistemas binarios. También pueden confundirse con falsos positivos los tránsitos de exoplanetas tipo Júpiter calientes, que se calientan significativamente por la irradiación estelar en sistemas binarios (Gaudí et al., 2005).

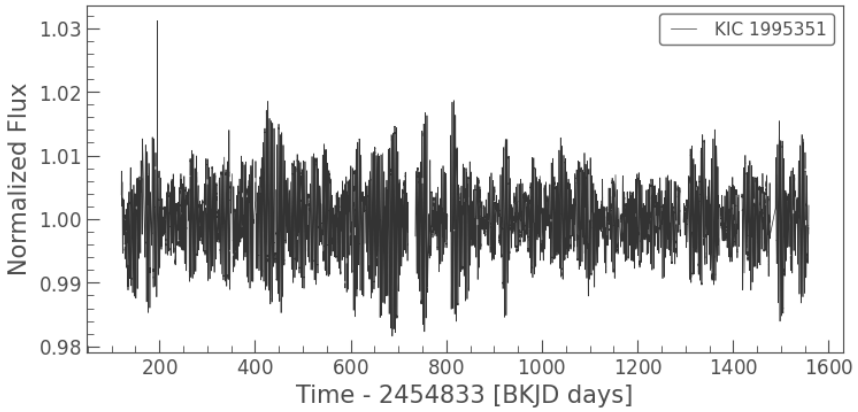
Figura 1: *Curva de Luz Kepler KIC 2306756 plegada en fase en la parte superior y plegado en fase para tránsitos impares, abajo izquierda y pares a su derecha.*



Nota. Se trata de un eclipse binario que da como resultado un falso positivo en la detección de exoplanetas[1]. Fuente: Elaboración propia usando Lightkurve[2].

Para este trabajo se utilizarán principalmente las curvas de luz procedentes de la misión Kepler, como la mostrada en la Figura 2, al ser el conjunto de datos que más planetas confirmados aporta al total de exoplanetas descubiertos, tal y como se aprecia en la Tabla 2.

Figura 2: *Curva de Luz Kepler KIC 1995351 completa, de 2009-2013*.



Fuente: Elaboración propia.

[1] https://archive.stsci.edu/kepler/data_search/search.php?action=Search&ktc_kepler_id=2306756
[2] http://docs.lightkurve.org/

Tabla 2: *Estadísticas de la misión Kepler*.

Planetas confirmados descubiertos por Kepler	2709
Candidatos y confirmados en la zona habitable (180 K < Equilibrio (T) < 310 K) o (0,25 < Insolación (flujo terrestre) < 2,2)	361
Candidatos del Proyecto Kepler	4717
Candidatos del Proyecto Kepler aún por confirmar	2057
Total de candidatos y planetas confirmados[3]	4781

Fuente: Adaptada de https://exoplanetarchive.ipac.caltech.edu/docs/counts_detail.html

1.1. Motivación

La motivación del autor de este trabajo es analizar las técnicas existentes de búsqueda de exoplanetas a través de algoritmos en lenguaje de programación Python[4], así como evaluar el uso de un nuevo método de determinación de la naturaleza del objeto eclipsante.

El desarrollo de este trabajo permite abordar el ciclo completo del tratamiento y uso de los datos, desde el análisis de los mismos para comprender la naturaleza física del fenómeno estudiado, pasando por la preparación de los conjuntos de datos necesarios, hasta la propuesta de un método estadístico de clasificación, que pueda ser utilizado para analizar de forma automática miles de curvas de luz.

Siendo Python el presente y el futuro del análisis de grandes conjuntos en observación astronómica y simulaciones astrofísicas, este trabajo es un ejercicio de aplicación práctica directa del potencial de Python en el tratamiento y análisis de grandes volúmenes de datos procedentes de misiones espaciales.

[3] El total de Candidatos y Planetas Confirmados es la unión de los conjuntos de datos de Planetas Confirmados y KOI Acumulados. Se debe tener en cuenta que algunos planetas confirmados nunca fueron designados como candidatos.
[4] https://www.python.org/

2. Objetivos

El objetivo principal del proyecto es determinar si la herramienta matemática transformada *wavelet* permite establecer una caracterización de las curvas de luz, que sea capaz de discernir la naturaleza de los objetos eclipsantes que puedan contener.

Los objetivos específicos del proyecto son los siguientes:

- Crear un *dataset* que permita determinar, mediante la aplicación de la transformada *wavelet*, parámetros estadísticos para la caracterización del objeto eclipsante, con la finalidad de diferenciar exoplanetas de falsos positivos.
- Desarrollar una metodología en Python para el procesado de los datos, la aplicación de la transformada *wavelet*, la utilización de algoritmos de *machine learning* supervisados y no supervisados.
- Desarrollar rutinas de procesamiento, clasificación y visualización.
- Establecer métricas para la evaluación de los resultados.

Para alcanzar estos objetivos se debe:

- Analizar los datos disponibles de la Misión Kepler, que se encuentran disponibles para la comunidad científica a través de las bases de datos de la NASA.
- Analizar las técnicas de detección de exoplanetas mediante el método de tránsito e investigar cómo aplicar la transformada *wavelet* para caracterizar tránsitos.
- Analizar estudios previos para establecer mecanismos de caracterización ya validados y que constituyan un punto de partida óptimo.
- Estudiar el acceso a los datos existentes en los observatorios virtuales a través de Python.

3. Estado del arte

3.1. Métodos de detección de exoplanetas

Existen múltiples técnicas para la detección de exoplanetas:

- **Método de la velocidad radial**. Un planeta que orbita alrededor de una estrella provoca un movimiento reflejo en torno al centro de masas del sistema estrella-planeta. Este movimiento produce perturbaciones periódicas en la velocidad radial, afectando por el efecto Doppler al espectro observado. Si este movimiento se produce en dirección al observador, las líneas espectrales se desplazarán hacia longitudes de onda más cortas (más azules). Si se aleja del observador se desplaza hacia longitudes de onda más largas (más rojas). La primera detección convincente de exoplanetas orbitando una estrella de secuencia principal, se realizó mediante mediciones de la velocidad radial en 1995 (Perryman, 2018), por parte de Michel Mayor y Didier Queloz, quienes descubrieron el exoplaneta de tipo júpiter caliente Dimido (51 Pegasi b), orbitando la estrella de clase G Helvetios (51 Pegasi). Para su descubrimiento se utilizó el espectrógrafo ELODIE del observatorio de Haute-Provence (Francia). Este instrumento permite medir la velocidad radial con una precisión de unos 13 $m \cdot s^{-1}$ en estrellas de hasta 9 magnitudes con un tiempo de exposición inferior a 30 minutos (Mayor y Queloz, 1995). Este método también suele emplearse para confirmar exoplanetas detectados con otras técnicas de detección, aunque mediante esta técnica se han confirmado 2056 exoplanetas en el NASA Exoplanet Archive[5].
- **Microlentes gravitacionales**. El término microlente, según Perryman (2018), se refiere a fenómenos que operan a escala de galaxia (microlente de cuásar) y a escala de masa estelar e inferior (relevante para los exoplanetas). En este último caso, la lente primaria es una masa puntual del orden de $1M_{\odot}$, y las dos imágenes de una fuente de fondo tienen una separación del orden de 1 microarcosegundo (mas), muy por debajo de la resolución instrumental típica en la superficie terrestre. El sistema de exoplanetas (estrella anfitriona y planeta) actúa como una lente múltiple, y una

[5] https://exoplanetarchive.ipac.caltech.edu/

estrella más distante dentro de la Galaxia actúa como una fuente puntual de detección. El cambio de magnitud de las sub imágenes debido a la geometría de alineación variable en el tiempo de la fuente observador-lente-fondo puede producir una variación significativa de la intensidad de la suma de las imágenes de la lente múltiple en escalas de tiempo de semanas. Este cambio de intensidad con el tiempo permite reconocer el evento como un evento de microlente. Una minuciosa monitorización de la curva de luz a medida que la alineación cambia durante varias horas permite identificar los efectos de lente adicionales de un planeta acompañante. Un seguimiento cuidadoso de la curva de luz a medida que la alineación cambia durante varias horas permite identificar los efectos de lente adicionales de un planeta acompañante. Se han detectado más de 70 planetas con este método, pero las detecciones requieren una alineación extremadamente precisa del observador, la fuente y la lente, dentro del radio angular de Einstein, o alrededor de 1 mas.

- **Astrometría**. La astrometría, en términos generales, se refiere al estudio de las posiciones y movimientos de cuerpos estelares, con el objetivo principal de determinar los paralajes estelares y los movimientos propios, de gran importancia para entender las propiedades de la estrella anfitriona. Para la detección de exoplanetas se utiliza la astrometría de alta precisión, con la que se determina la componente transversal del desplazamiento de la estrella anfitriona, ocasionado por la perturbación gravitacional producida por el planeta que la órbita. Esta manifestación dinámica de su influencia gravitacional en el plano del cielo está estrechamente relacionada con las mediciones de la velocidad radial, que son sensibles al correspondiente desplazamiento del fotocentro a lo largo de la línea de visión. Este método es extremadamente complejo, debido a la precisión necesaria para detectar los desplazamientos de las posiciones estelares debidos a planetas orbitando. Las mejores precisiones alcanzadas en la actualidad, de aproximadamente 1 microarcosegundo, conseguidas con la misión Hipparcos y el HST, no son suficientes. En la actualidad, solo existe un exoplaneta confirmado mediante esta técnica, el DENIS-P J082303.1-491201 b, descubierto en 2013 (Sahlmann et al. 2013).

También se han encontrado algunos planetas mediante otras técnicas, como la imagen directa o la sincronización de púlsares, pero su número no es demasiado relevante debido a las dificultades que implica su detección (Perryman, 2018). Combinando los resultados de las observaciones y los estudios realizados con diferentes técnicas, podemos construir una imagen representativa de la diversidad de exoplanetas y sistemas planetarios.

La mayor parte de los exoplanetas, en torno al 90% se han detectado a través del método de tránsito. En este proyecto usaremos dicho método, por lo que lo explicaremos en profundidad en el siguiente apartado.

3.1.1. Método del tránsito

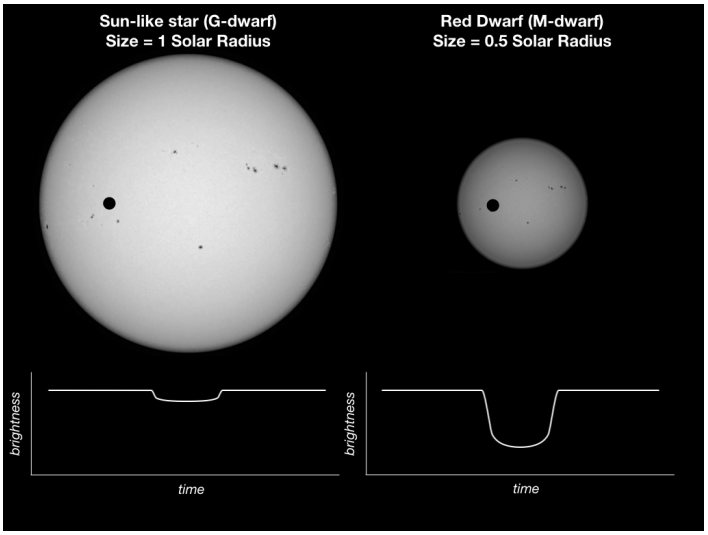
En 1999 se detectó el primer exoplaneta HD 209458 b, como resultado del seguimiento fotométrico del sistema HD 209458 a través de la misión Hipparcos (Robichon y Arenou, 2000). Otros sistemas descubiertos utilizando otros métodos, principalmente el de la velocidad radial, fueron monitorizados en busca de posibles tránsitos. Las curvas de luz de los sistemas con planetas en tránsito proporcionan una estimación de sus radios. Si se puede estimar su masa, también se pueden estimar las densidades, lo que proporciona una primera aproximación a la determinación de su composición. La fotometría y la espectroscopia durante el tránsito y durante el eclipse secundario, cuando el planeta pasa por detrás de la estrella, permiten realizar más pruebas de las propiedades estructurales y atmosféricas del planeta.

Hasta junio de 2022, se habían descubierto 3858 planetas a través del método del tránsito. La misión Kepler de la NASA, que buscó planetas mediante dicho método entre 2009 y 2013, encontró miles de posibles candidatos a exoplanetas (*5 ways to find a planet*, s. f.).

El método del tránsito consiste en detectar variaciones periódicas en el flujo de una estrella, producidas cuando un exoplaneta pasa por delante de ella. Desde la Tierra, por ejemplo, se observa el tránsito ocasional de Venus o Mercurio. Dichos eventos se observan como un punto negro que se desplaza a través del Sol, ya que Venus o Mercurio bloquean parte de la luz solar cuando el planeta se sitúa entre el Sol y nosotros. Esta técnica, aplicada a otros sistemas, encuentra exoplanetas mediante la observación de estrellas durante largos periodos de tiempo y la detección de pequeñas caídas en el brillo observado, producido cuando dicho exoplaneta transita por la estrella, como se aprecia en la Figura 3. Para poder

detectar exoplanetas mediante este método, el plano orbital del planeta tiene que estar alineado con la observación desde la Tierra o un telescopio espacial.

Figura 3: *Método del tránsito.*



Nota. Un planeta que transita por una estrella pequeña (derecha) provoca una señal de tránsito mayor que la de un planeta del mismo tamaño que transita por una estrella de mayor tamaño (izquierda). Fuente: Adaptada de https://avanderburg.github.io/tutorial/tutorial.html

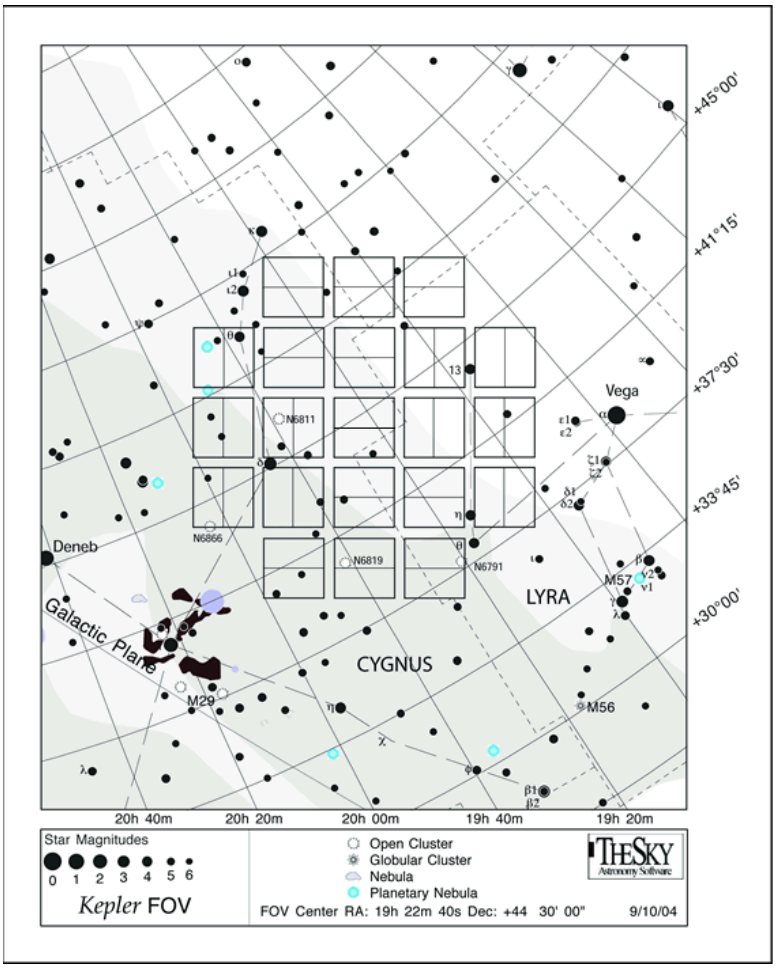
A partir de la disminución del brillo durante el tránsito, se puede estimar el radio del planeta. Determinando la masa, a partir de mediciones espectroscópicas Doppler, se puede obtener la densidad media del planeta, lo que permite conocer su composición interna.

La caracterización de la atmósfera de un exoplaneta se realiza mediante mediciones realizadas en diferentes puntos de su órbita. Estas mediciones incluyen espectros tomados durante el tránsito, variaciones fotométricas a lo largo de la órbita y datos de eclipses secundarios, producidos cuando el planeta cruza por detrás de la estrella. Las mediciones de fase y las realizadas durante el eclipse secundario, proporcionan información sobre la temperatura, el albedo planetario y las características espectrales del exoplaneta.

La misión COROT (COnvection, ROtation and planetary Transits) de la Agencia Espacial Europea (*COROT overview*, s. f.), fue la primera misión de observatorio espacial para la búsqueda de planetas extrasolares y se inició con el lanzamiento del satélite COROT el 27 de diciembre de 2006 y operó hasta 2013. Esta misión usaba el método del tránsito para la detección de exoplanetas. Tras esta misión, en 2009, se lanzó la Misión Kepler, con el

objetivo de medir las variaciones en el flujo de 200.000 estrellas, siendo su principal misión la detección de planetas transitando dichas estrellas. El telescopio espacial Kepler rotaba 90 grados cada 90 días para mantener sus paneles solares apuntando al Sol. Es por ello que los datos de Kepler se dividen en cuartos de 90 días. Kepler almacenaba la información de los píxeles alrededor de las estrellas a intervalos de entre 1 y 30 minutos. La misión produjo series de datos temporales de flujo (curvas de luz) para cada estrella observada en su campo de visión, ver Figura 4.

Figura 4: *Campo de visión de la misión Kepler.*

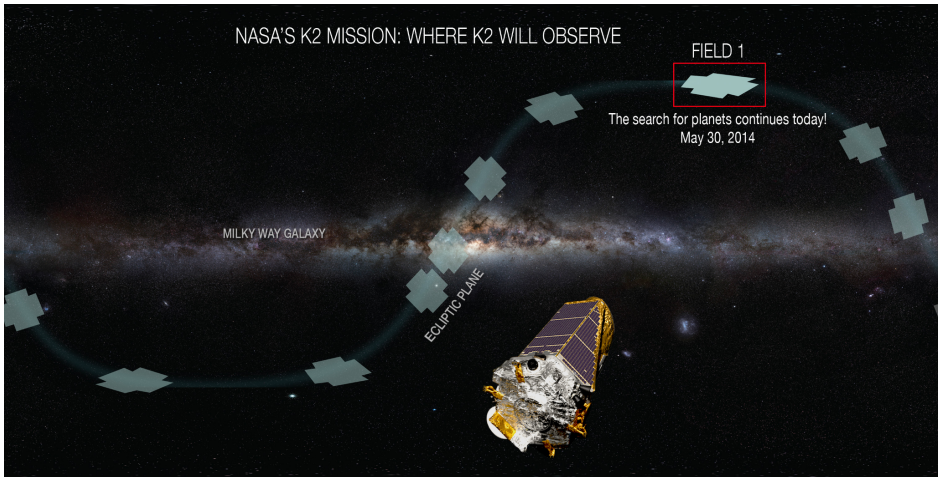


Fuente: Adaptada de https://www.nasa.gov/mission_pages/kepler/overview/index.html

La misión K2 fue la segunda misión de Kepler, diseñada como solución a un fallo de funcionamiento en uno de sus volantes de inercia, que impedía que el telescopio permaneciera inmóvil mientras observaba un objetivo, por lo que se extendió la vida de la misión pero ya no centrada en la misma región del cielo de Cygnus, sino varias zonas alineadas con el plano del sistema solar cada 80 días (*Kepler and K2 Mission overview*, s. f.),

como se aprecia en la Figura 5. Otras misiones como TESS[6] (2018) también se basan en el método del tránsito para la detección de exoplanetas.

Figura 5. *Campo de visión de la misión K2.*



Fuente: Adaptada de https://www.nasa.gov/mission_pages/kepler/overview/index.html

La Agencia Espacial Europea (ESA) lanzó 18 December 2019 el satélite de caracterización de exoplanetas CHEOPS (CHaracterising ExOPlanet Satellite)[7], que observa estrellas brillantes que albergan exoplanetas y registra los tamaños precisos de estos planetas relativamente pequeños. En combinación con las mediciones de masa ya calculadas por otros observatorios, permite determinar la densidad del planeta y realizar una primera caracterización de su naturaleza. CHEOPS ha proporcionado objetivos bien caracterizados para que el telescopio espacial internacional James Webb (JWST) realice estudios más detallados de sus atmósferas.

Además, la ESA tiene prevista otras dos misiones:

- PLATO[8] (PLAnetary Transits and Oscillations of stars), prevista para 2027, tiene por objetivo encontrar y estudiar un gran número de sistemas planetarios extrasolares, con énfasis en las propiedades de los planetas terrestres en la zona habitable alrededor de estrellas de tipo solar. También ha sido diseñado para investigar la actividad sísmica de las estrellas, lo que permitirá caracterizar con precisión la estrella que alberga al planeta, incluida su edad.

[6] https://tess.mit.edu/
[7] https://sci.esa.int/web/cheops/
[8] https://sci.esa.int/web/plato

- ARIEL[9] (Atmospheric Remote-sensing Infrared Exoplanet Large-survey), prevista para 2029, estudiará de qué están compuestos los exoplanetas, cómo se formaron y cómo evolucionan, inspeccionando una muestra diversa de unos 1000 planetas extrasolares, simultáneamente en longitudes de onda visibles e infrarrojas. Se trata de la primera misión dedicada a medir la composición química y las estructuras térmicas de cientos de exoplanetas en tránsito.

3.2. Datos de exoplanetas

Como ya se indicó en el Apartado 1, en este proyecto nos centraremos principalmente en los datos de la misión Kepler, ya que constituye el mayor repositorio de datos de curvas de luz con exoplanetas confirmados, candidatos y falsos positivos, lo que permite aplicar técnicas de aprendizaje automático y aprendizaje profundo, para la detección de exoplanetas mediante el método del tránsito.

Los datos relativos a los descubrimientos de exoplanetas se encuentran en la base de datos NASA Exoplanet Archive[10]. Y los datos de las curvas de luz se encuentran a disposición de la comunidad científica en el observatorio virtual Mikulski Archive for Space Telescopes[11] (MAST). Para cada objeto observado por Kepler (KepID), se dispone de curvas de luz (KIC) identificadas por trimestres de observación, que contienen información del flujo de dicho objeto.

Estas curvas de luz se encuentran en formato FITS (Flexible Image Transport System). Este formato de datos es el más utilizado en astronomía para almacenar y analizar archivos de datos científicos. FITS está diseñado para almacenar conjuntos de datos científicos consistentes en matrices multidimensionales (imágenes) y tablas bidimensionales organizadas en filas y columnas de información.

3.2.1. Formato FITS

Un archivo FITS se compone de segmentos denominados unidades de datos/cabecera (HDU), donde la primera HDU se denomina "HDU primaria" o "matriz de datos primaria". La matriz de datos primaria puede contener una matriz de 1 a 999 dimensiones de enteros de 1, 2 o 4

[9] https://sci.esa.int/web/ariel
[10] https://exoplanetarchive.ipac.caltech.edu/cgi-bin/TblView/nph-tblView?app=ExoTbls&config=koi
[11] https://archive.stsci.edu/

bytes o números de punto flotante de 4 u 8 bytes utilizando representaciones IEEE. Cualquier número de HDUs adicionales puede seguir al array primario. Estas HDU adicionales se denominan "extensiones" de FITS. Actualmente se definen tres tipos de extensiones estándar:

- Las extensiones de imagen contienen una matriz de píxeles de 0 a 999 dimensiones, similar a la matriz primaria. La cabecera comienza con XTENSION = 'IMAGE '.
- Las extensiones de tabla ASCII almacenan información tabular con toda la información numérica almacenada en formatos ASCII. Aunque las tablas ASCII son generalmente menos eficientes que las tablas binarias, pueden hacerse relativamente legibles para las personas y pueden almacenar información numérica con un tamaño y precisión esencialmente arbitrarios (por ejemplo, reales de 16 bytes). La cabecera comienza con XTENSION = 'TABLE '.
- Las extensiones de tablas binarias almacenan información tabular en una representación binaria. Cada celda de la tabla puede ser una matriz, pero la dimensionalidad de la matriz debe ser constante dentro de una columna. El estándar estricto sólo admite matrices unidimensionales, pero la convención para admitir matrices multidimensionales está ampliamente aceptada. La cabecera comienza con XTENSION = 'BINTABLE').

Los ficheros FITS de las curvas de luz de la Misión Kepler tienen la estructura de la Tabla 3.

Tabla 3: *Estructura FIT de una curva de luz de la Misión Kepler*.

```
Filename: kplr011904151-2009291181958_slc.fits
No.    Name      Ver    Type      Cards     Dimensions   Format
  0  PRIMARY       1 PrimaryHDU      58             ()
  1  LIGHTCURVE    1 BinTableHDU    155   44550R x 20C   [D, E, J, E, E, E, E, E, E, J,
                                                         D, E, D, E, D, E, D, E, E, E]
  2  APERTURE      1 ImageHDU        48       (11, 10)   int32
```

Fuente: Elaboración propia.

La curva de luz de Kepler contiene dos valores diferentes para el flujo medido:

- Flujo de fotometría de apertura simple (SAP_FLUX). Es el flujo en crudo.
- Flujo Calibrado (PDC_SAP_FLUX). Es el flujo eliminando las variaciones instrumentales. Este es el flujo que usaremos en este proyecto. Al estar normalizado

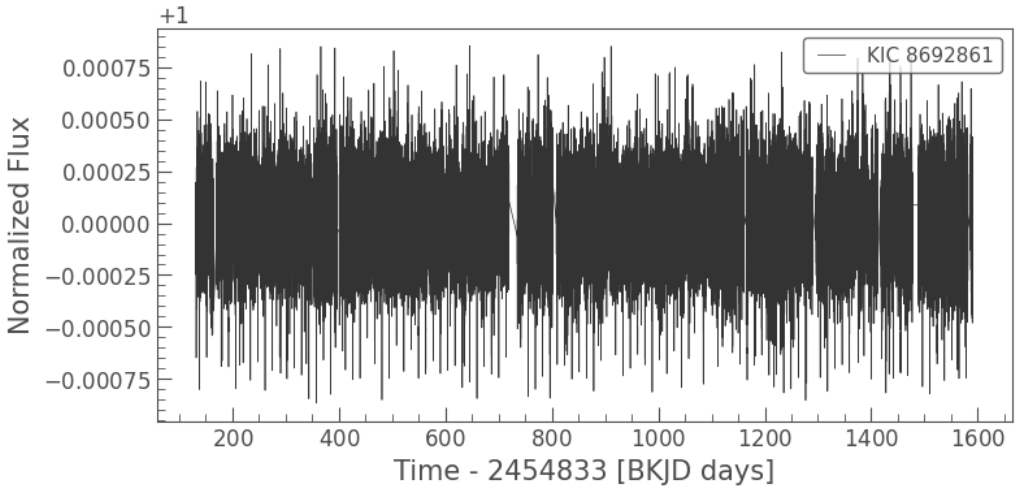
y calibrado nos permitirá establecer relaciones entre las diferentes curvas de luz procedentes de diferentes sistemas, con diferentes niveles de intensidad.

3.2.2. Series temporales Kepler

El telescopio espacial Kepler, para la búsqueda de exoplanetas transitando estrellas, realizaba un seguimiento casi continuo de un único campo de visión, desde una órbita alrededor del Sol, siguiendo a la Tierra. Al seguir a la Tierra y mantener una orientación constante, los paneles solares recibían cada vez menos luz solar. Para garantizar que los paneles solares siguieran orientados hacia el Sol, Kepler realizó rotaciones trimestrales, una cada 93 días. Es por ello que las observaciones de series temporales largas se dividen en trozos separados, llamados trimestres (*quarters*).

Construir las curvas de luz con todos los datos disponibles (varios trimestres) es necesario cuando se buscan señales pequeñas, como los tránsitos planetarios. Además, cuando se accede a los datos de Kepler a través de MAST, dichos datos se encuentran almacenados en varios ficheros FITS correspondientes a un trimestre de observaciones. Combinando y normalizando estas observaciones separadas, se puede formar una única curva de luz que incluya todos los trimestres observados (ver Figura 6).

Figura 6: *Curva de Luz Kepler KIC 8692868 completa.*



Fuente: Elaboración propia.

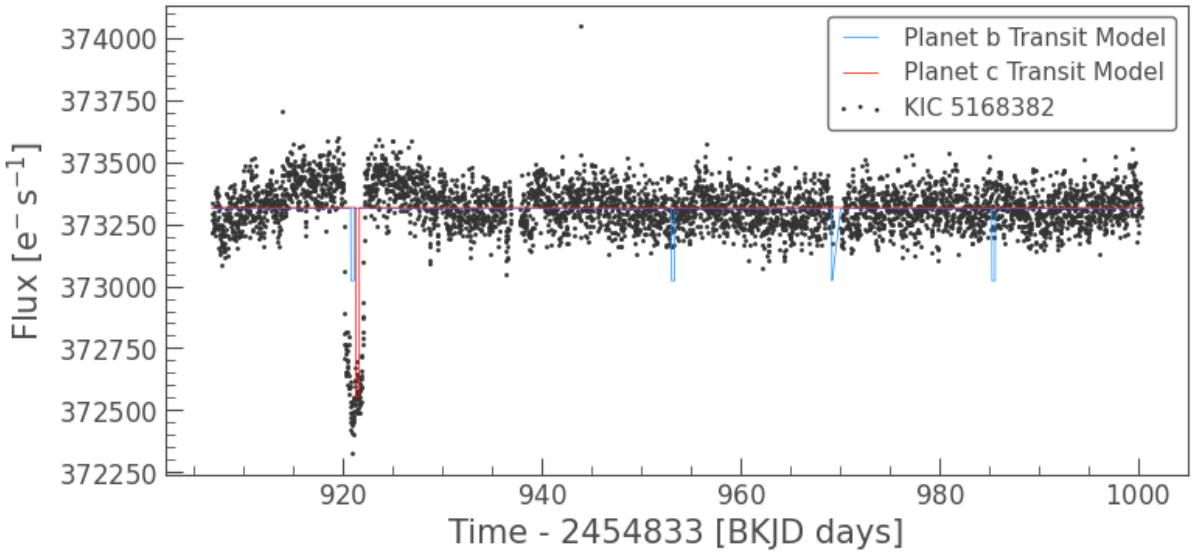
3.3. Paquetes de Python para el procesamiento de curvas Kepler

3.3.1. Lightkurve

Lightkurve[12] es un paquete de Python que tiene como objetivo apoyar el análisis de datos de series temporales sobre planetas, estrellas y galaxias obtenidos por telescopios que recogen imágenes en luz visible o infrarroja, como la Misión Kepler.

El objetivo principal de Lightkurve es simplificar el acceso a los datos y facilitar el análisis de los mismos, como la detección de tránsitos (ver Figura 7).

Figura 7. *Modelo de tránsito realizado con Lightkurve sobre la curva de luz KIC 5168382 de la misión Kepler.*



Fuente: Elaboración propia.

Se trata de una herramienta de código abierto que dispone de múltiples funcionalidades para el análisis de series temporales. Está desarrollado sobre el proyecto AstroPy[13], un paquete básico para la astronomía que utiliza el lenguaje de programación Python, desarrollado por la comunidad astronómica, que mejora la usabilidad, interoperabilidad y colaboración entre los paquetes Python de astronomía. El paquete central de AstroPy contiene funcionalidades dirigidas a astrónomos y astrofísicos profesionales, pero puede ser útil para cualquiera que desarrolle software de astronomía.

[12] http://docs.lightkurve.org/
[13] https://www.astropy.org/

El objeto LightCurve de Lightkurve extiende el objeto TimeSeries de AstroPy TimeSeries, que a su vez es una subclase del objeto Table de AstroPy. Un objeto AstroPy TimeSeries es una tabla que contiene una columna especial de tiempo (primera columna). LightCurve extiende esta clase agregando columnas especiales flux y flux_err, que son siempre la segunda y tercera columna. Estas columnas dedicadas nos permiten proporcionar métodos extra que son específicos para la manipulación de los datos de luminosidad. Por ejemplo, los métodos que actualmente son exclusivos de los objetos LightCurve incluyen plot(), normalize(), fill_gaps(), flatten(), remove_outliers(), remove_nans(), estimate_cdpp(), plot_river(), to_seismology()...

El paquete Lightkurve tiene funciones para buscar y descargar observaciones de Kepler/K2 y TESS. Estas herramientas han sido creadas para que el acceso a los datos de los telescopios espaciales sea claro y sencillo, con nombres intuitivos de métodos y palabras clave.

En este trabajo de fin de estudios, utilizaremos el paquete Lightkurve para automatizar la descarga de las curvas de luz que analizaremos posteriormente con otras herramientas. Lightkurve utiliza Astroquery[14] para realizar búsquedas basadas en el número de identificación del catálogo o el nombre.

Para buscar un objeto Kepler, utilizaremos su número de identificación del Catálogo de Entrada Kepler (KIC). Por ejemplo: podemos buscar KIC 5168382 utilizando la función search_lightcurve:

```
search_result = lk.search_lightcurve('KIC 5168382', author='Kepler')
search_result
```

Esta búsqueda proporciona el resultado mostrado en la Tabla 4. En concreto, tenemos 17 trimestres de datos a lo largo de 5 años de observación:

Tabla 4: *Resultado de la búsqueda de curvas de luz del objeto KIC 5168382 mediante Lightkurve.*

SearchResult containing 14 data products.

#	mission	year	author	exptime	target_name	distance
0	Kepler Quarter 00	2009	Kepler	1800	kplr005168382	0.0
1	Kepler Quarter 01	2009	Kepler	1800	kplr005168382	0.0
2	Kepler Quarter 02	2009	Kepler	1800	kplr005168382	0.0
3	Kepler Quarter 03	2009	Kepler	1800	kplr005168382	0.0

[14] https://astroquery.readthedocs.io/en/latest/

4	Kepler Quarter	04	2010	Kepler	1800	kplr005168382	0.0
5	Kepler Quarter	06	2010	Kepler	1800	kplr005168382	0.0
6	Kepler Quarter	07	2010	Kepler	1800	kplr005168382	0.0
7	Kepler Quarter	08	2011	Kepler	1800	kplr005168382	0.0
8	Kepler Quarter	10	2011	Kepler	1800	kplr005168382	0.0
9	Kepler Quarter	11	2012	Kepler	1800	kplr005168382	0.0
10	Kepler Quarter	12	2012	Kepler	1800	kplr005168382	0.0
11	Kepler Quarter	14	2012	Kepler	1800	kplr005168382	0.0
12	Kepler Quarter	15	2013	Kepler	1800	kplr005168382	0.0
13	Kepler Quarter	16	2013	Kepler	1800	kplr005168382	0.0

Fuente: Elaboración propia.

Estos datos pueden ser descargados y tratados de manera sencilla y ordenada mediante los métodos que implementa Lightkurve, permitiendo seleccionar en todo momento qué trimestres queremos descargar, en caso de no necesitar los 5 años de datos acumulados. Como se explica en el Apartado 4, esta funcionalidad es indispensable para el procesamiento de series temporales largas.

3.3.2. tsfresh

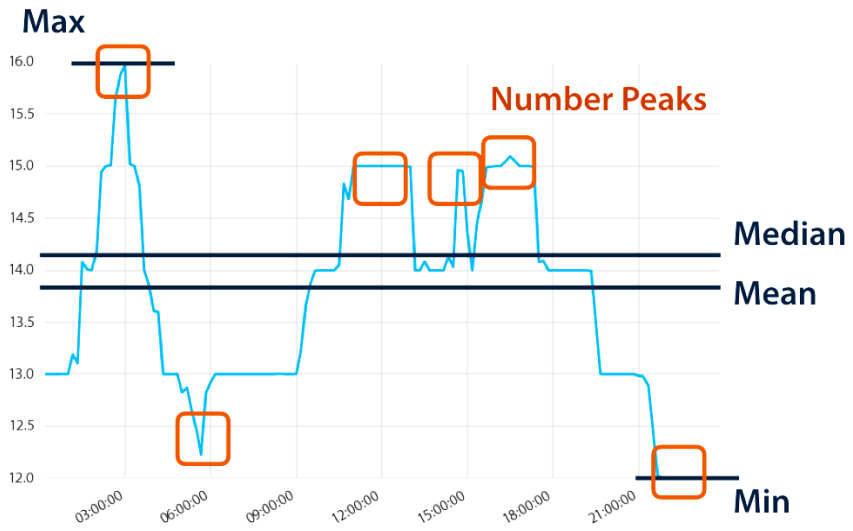
tsfresh[15] es un paquete Python de código abierto utilizado para el análisis de los datos de series temporales, como las de las curvas de luz, que son datos secuenciales generados utilizando una variable independiente (la variable temporal).

Mediante esta herramienta se pueden extraer características[16] (ver Figura 8) de las series temporales. Estas *features* pueden ser complejas, como la entropía de la densidad espectral de potencia de la serie temporal o el valor medio de una aproximación central de la segunda derivada, o más simples como el mínimo, el máximo o la media de la serie. La extracción de características es útil para hacer clusters de series temporales o realizar tareas de clasificación.

Al ser compatible con sklearn[17], una herramienta que implementa diversos modelos de aprendizaje automático, se facilita la aplicación de algoritmos supervisados (SVM, nearest neighbors, random forest...) y no supervisados (K-Means, DBSCAN, OPTICS...).

[15] https://tsfresh.readthedocs.io/
[16] https://tsfresh.readthedocs.io/en/latest/text/list_of_features.html
[17] https://scikit-learn.org/stable/

Figura 8. *Ejemplo de características extraídas de series temporales mediante tsfresh.*



Fuente: Adaptada de https://tsfresh.readthedocs.io

Con tsfresh se pueden extraer más de 1500 características de las series temporales, lo que nos permitirá analizar las curvas de luz de la misión Kepler.

3.4. La transformada *wavelet*

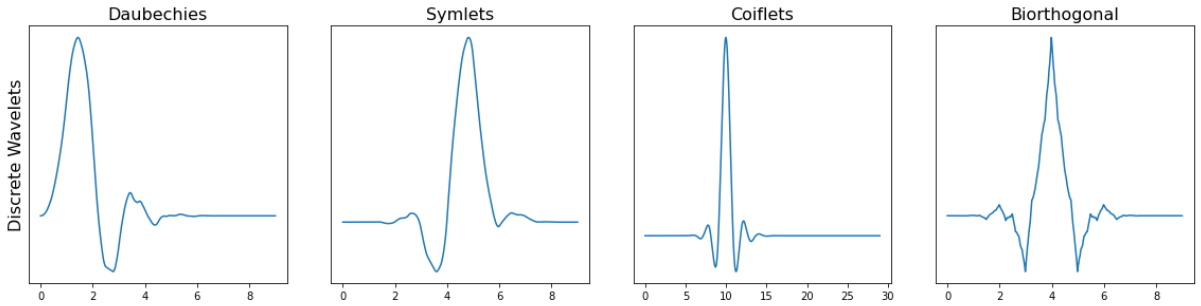
Las transformadas *wavelet* son herramientas matemáticas para analizar datos cuyas características varían en diferentes escalas (*Wavelet Transforms in MATLAB*, s. f.). En el caso de las señales, las características pueden ser frecuencias que varían en el tiempo, transitorios o tendencias que varían lentamente. En el caso de las imágenes, las características son los bordes y las texturas.

Las transformadas *wavelet* se crearon principalmente para resolver las limitaciones de la Transformada de Fourier. Una de las principales desventajas de la Transformada de Fourier es que captura información de frecuencia global, es decir, frecuencias que persisten a lo largo de toda una señal. Este tipo de descomposición de la señal no es adecuada para todas las aplicaciones, por ejemplo en aquellas donde las señales tienen intervalos cortos de oscilación característica o que son difíciles de detectar, como las transiciones en las curvas de luz.

Mientras que el análisis de Fourier consiste en descomponer una señal en ondas sinusoidales de frecuencias específicas, el análisis *wavelet* se basa en descomponer las señales en versiones desplazadas y escaladas de una *wavelet*.

Una *wavelet*, a diferencia de una onda sinusoidal, es una oscilación ondulada que decae rápidamente. Esto permite a las *wavelets* representar los datos en múltiples escalas. Se pueden utilizar diferentes familias de *wavelets* (Figura 9) dependiendo de la aplicación: Daubechies, Coiflet, Biorthogonal y Symlets.

Figura 9. *Familias de wavelets discretas del paquete PyWawelets.*



Fuente: Elaboración propia.

3.4.1. Caracterización mediante transformada *wavelet*

La idea básica de la caracterización de señales consiste en determinar qué cantidad de una *wavelet* existe en una señal para una escala y una ubicación determinadas. La señal se convoluciona con un conjunto de *wavelets* a distintas escalas (Talebi, 2020).

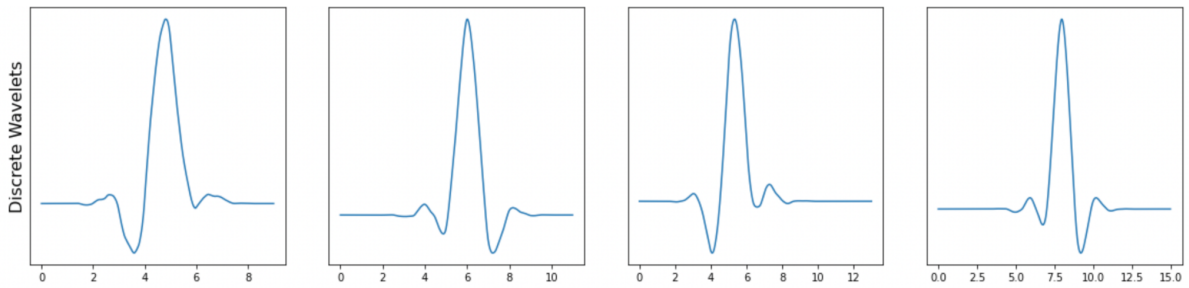
Una *wavelet* de una escala concreta se traslada por toda la señal variando su ubicación, y en cada paso de tiempo multiplicamos la *wavelet* y la señal. El producto de esta multiplicación nos da un coeficiente para esa escala de *wavelet* en ese paso de tiempo. A continuación, aumentamos la escala de la *wavelet* y se repite el proceso.

En este trabajo usaremos la transformada *wavelet* que se define a través de la ecuación (1).

$$W_{a,b} = \int_{-\infty}^{\infty} f(t)\Psi_{a,b}(t)dt \qquad (1)$$

En la *wavelet*, se multiplica *f(t)* por la función *wavelet* madre $\Psi_{a,b}(t)$, que tiene 2 componentes (a, b): el parámetro de escala y el parámetro de traslación. A diferencia de la Transformada de Fourier, en la transformada *wavelet* la función *wavelet* madre no es única, por lo que tenemos tantas transformadas *wavelet* como transformadas *wavelet* madre que podamos generar, como se aprecia en la Figura 10.

Figura 10. *De izquierda a derecha, diferentes funciones wavelet madre de la familia Symlet: Symlet 5, Symlet 6, Symlet 7 y Symlet 8*



Fuente: Elaboración propia.

Las funciones *wavelet* madre deben tener una propiedad oscilatoria y atenuada. Esta compactación es la que permite localizar el momento en el que se producen las frecuencias.

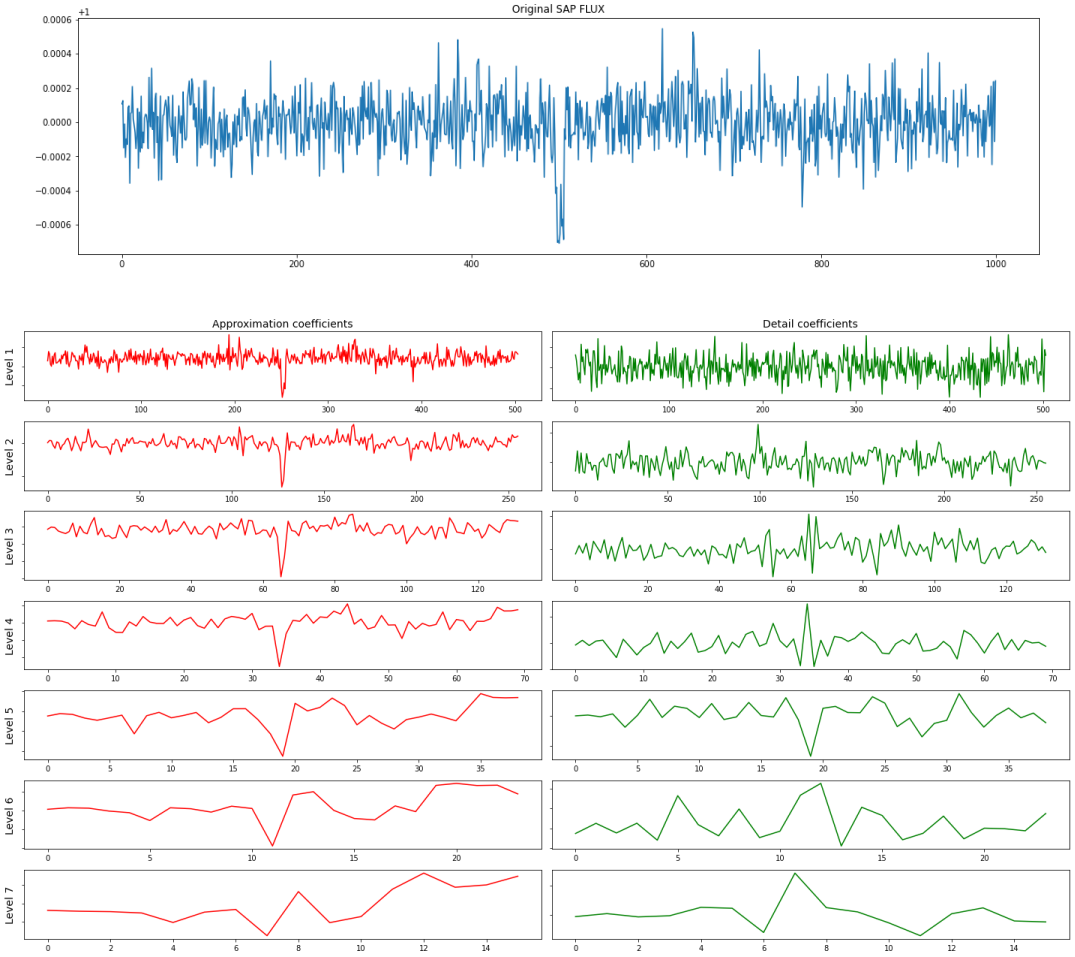
Pueden diseñarse funciones *wavelet* madre, propias, teniendo en cuenta que tienen que cumplir las siguientes propiedades:

- Su media tiene que ser cero (el área bajo la curva es cero).
- Tienen norma igual a uno.
- Se atenúan partiendo de un punto máximo.
- Tienen un soporte finito.

La transformada *wavelet* se utiliza para una gran cantidad de aplicaciones: compresión de datos, filtrado de ruido o extracción de parámetros y características. La transformada *wavelet* permite la clasificación de las diferentes aportaciones de frecuencia existentes en una señal, ya que al tratarse de una representación de la misma a diferentes resoluciones, permite ubicar dichas aportaciones en escalas espaciales y temporales.

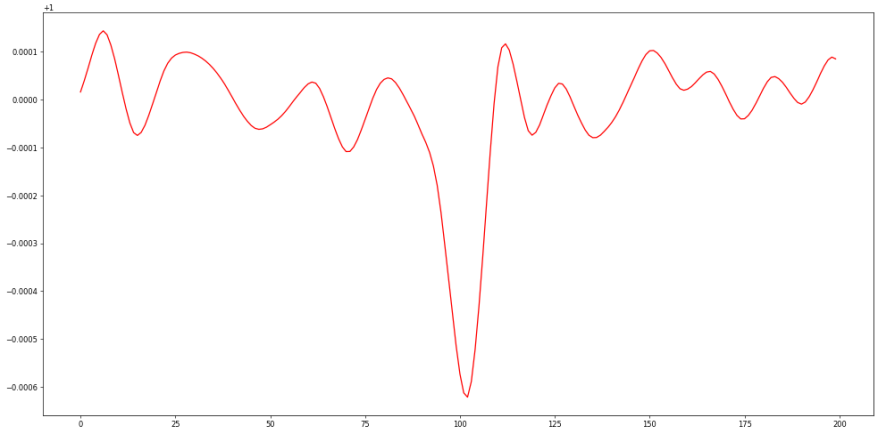
A modo de ejemplo, se descompuso la señal del flujo medido de una curva de luz plegada en fase, en diferentes escalas (Figura 11) y posteriormente se reconstruyó utilizando únicamente las componentes de baja frecuencia (Figura 12).

Figura 11. *Transformada wavelet discreta de siete niveles, aplicada sobre la curva KIC 8692861 plegada en fase (en azul).*



Nota. Los coeficientes de aproximación o de escala (en rojo), son la representación paso bajo de la señal y los coeficientes de detalles (en verde), representan las altas frecuencias de la señal. Fuente: Elaboración propia.

Figura 12. *Detalle del tránsito de la señal de la curva KIC 8692861 plegada en fase, reconstruida utilizando las 4 componentes de baja frecuencia.*



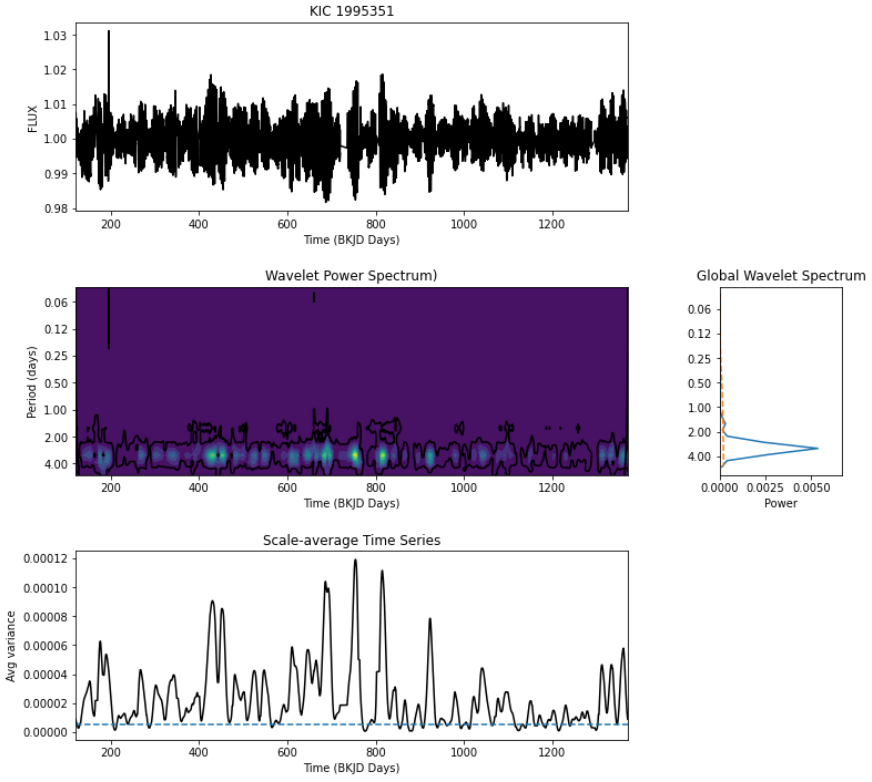
Fuente: Elaboración propia.

La transformada *wavelet* suele utilizarse como mecanismo de filtrado del ruido presente en las curvas de luz (Régulo et al., 2007). Nuestra aproximación se basa en aplicar la transformada *wavelet* y su reconstrucción sobre la curva plegada en fase y no sobre la curva completa de valores de tiempo y flujo observado.

3.4.2. Implementación de C. Torrence and G. Compo

A lo largo del proceso de investigación y aprendizaje, se testeó la implementación de la transformada *wavelet* propuesta por los doctores Christopher Torrence y Gilbert P. Compo (Torrence y Compo, 1998), desarrollada en Python por Evgeniya Predybaylo (Torrence, s. f.). Dicha implementación, con las modificaciones necesarias, fue aplicada en una curva de luz Kepler, para valorar el espectro *wavelet* como herramienta para conseguir los objetivos de este proyecto y aprender los fundamentos de la transformada *wavelet*, ver Figura 13.

Figura 13. *Implementación de C. Torrence y G. Compo aplicada a la curva de luz KIC 11995351.*



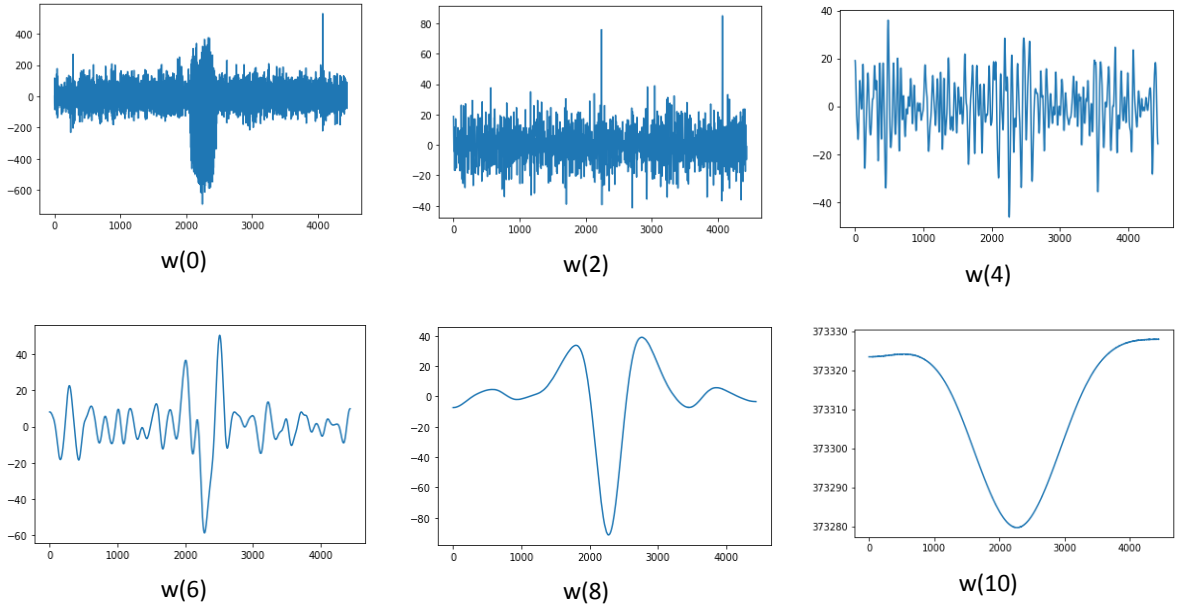
Nota. El objeto representado es una estrella variable dominada por la actividad magnética, con un periodo de 3,30 días asociado a su rotación (Bravo et al., 2014). Señal completa de flujo (arriba), espectro que indica los periodos donde se producen picos de potencia (medio) que coinciden con la serie temporal escalada (abajo). Fuente: Elaboración propia.

3.4.3. Implementación *wavelet starlet*

La implementación del apartado anterior se descartó por la complejidad de aplicar los resultados de los espectros a algoritmos de aprendizaje supervisado y no supervisado, por lo que se buscaron otras implementaciones *wavelet*, como la *wavelet starlet* (Starck et al., 1994). Ver Figura 14.

Fue adaptada para su uso en Google Colab, que usa Python 3, y testeada también para valorar su utilización en el proyecto. Se descartó la misma debido a que no formaba parte de un paquete específico de Python con clases y métodos documentados.

Figura 14. *Implementación de la transformada wavelet starlet aplicada a la curva plegada del objeto KIC 5168382.*



Nota. De izquierda a derecha y de arriba a abajo, la descomposición en las diferentes escalas frecuenciales. Fuente: Elaboración propia.

3.4.4. Implementación PyWavelets

PyWavelets[18] es un software de transformada *wavelet* de código abierto para Python. Las principales características de PyWavelets son:

- Transformada *wavelet* 1D, 2D y nD Discreta Directa e Inversa (DWT e IDWT)
- Cálculo de aproximaciones de funciones *wavelet* y de escala

[18] https://pywavelets.readthedocs.io/en/latest/

- Más de 100 filtros de *wavelets* incorporados y soporte para *wavelets* personalizadas
- Cálculos de precisión simple y doble
- Cálculos reales y complejos

PyWavelets es el paquete que se ha elegido para la aplicación de la transformada *wavelet* en este trabajo. Los motivos de dicha elección son los siguientes:

1. Permite la descomposición en componentes de manera sencilla.
2. Implementa varios métodos de reconstrucción a través de los coeficientes hallados, posibilitando el filtrado en función de los objetivos planteados.
3. Implementa una gran cantidad de *wavelets* diferentes, lo que facilita las variaciones y los test de código con diferentes *wavelets*, para determinar las que obtienen mejores resultados.

4. Creación del *dataset*

4.1. Fuentes utilizadas

La creación del *dataset* es una de las actividades críticas de este trabajo. Por un lado, debemos obtener un *dataset* que incluya el objeto observado (`kepid`), los exoplanetas confirmados, los falsos positivos y los candidatos (identificados en `koi_disposition`), así como los parámetros que nos interesan para poder procesar correctamente los datos:

- `period`: Intervalo entre 2 tránsitos consecutivos.
- `t0`: La hora correspondiente al centro del primer tránsito detectado en día juliano baricéntrico (BJD), que es la fecha juliana (JD) corregida por diferencias en la posición de la Tierra con respecto al baricentro del Sistema Solar.
- `duration_hours`: La duración de los tránsitos observados. La duración se mide desde el primer contacto entre el planeta y la estrella hasta el último.

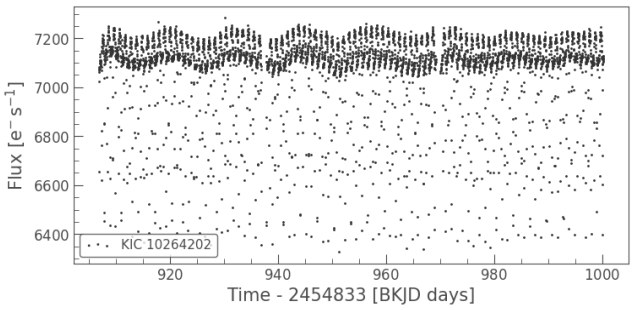
Para la creación de este *dataset* en Python se ha usado el fichero acumulativo[19] que proporciona la NASA y que presenta un total de 9564 registros.

4.2. Pre-procesado de las curvas y visualización

4.2.1. Visualización

Para cada `kepid` se puede obtener una curva de luz completa (formada por varios trimestres a lo largo de varios años) como la mostrada en la Figura 2, o una curva para uno o varios trimestres (ver Figura 15).

Figura 15. *Ejemplo de curva de luz para el objeto KIC 10264202, para un solo trimestre de observación.*



Fuente: Elaboración propia.

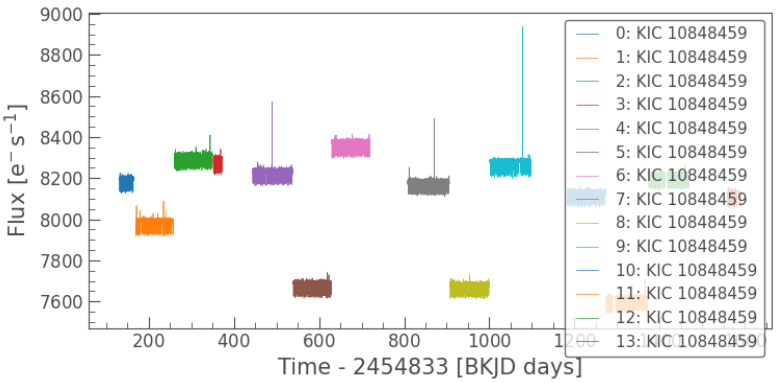
[19] https://exoplanetarchive.ipac.caltech.edu/cgi-bin/TblView/nph-tblView?app=ExoTbls&config=cumulative

4.2.2. Pre-procesado

A través del *dataset* creado se obtienen las curvas de luz para cada uno de los objetos. Tal y como se explica en el Apartado 2.2, los datos de la misión Kepler se almacenan por trimestres. Por ello, es necesario realizar varias tareas previas para el correcto análisis y tratamiento de los datos:

- Unir todas las curvas de la colección de un objeto en una única curva. Como se puede ver en la Figura 16, al descargar una colección de curvas, se deben unir los diferentes trimestres para obtener una curva única que sirva para el análisis.
- Eliminar las bajas frecuencias utilizando el filtro Savitzky-Golay (Savitzky y Golay, 1964) de SciPy[20]. Necesario para eliminar aquellos factores externos que influyen en la curva de luz, ya que las estrellas parecen más brillantes o más tenues dependiendo de dónde estén sus manchas.
- Eliminar aquellos datos cuyos valores de flujo son mayores o menores que el flujo medio en, al menos, sigma veces la desviación estándar. Usaremos un valor sigma de 20.

Figura 16. *Ejemplo la colección de curvas para el objeto KIC 10848459.*



Fuente: Elaboración propia.

4.2.3. Folding

Entre la gran variedad de curvas de luz que se obtienen al observar diferentes objetos, en ocasiones es evidente que existen periodos en los que pudiera inferirse un tránsito. Pero en

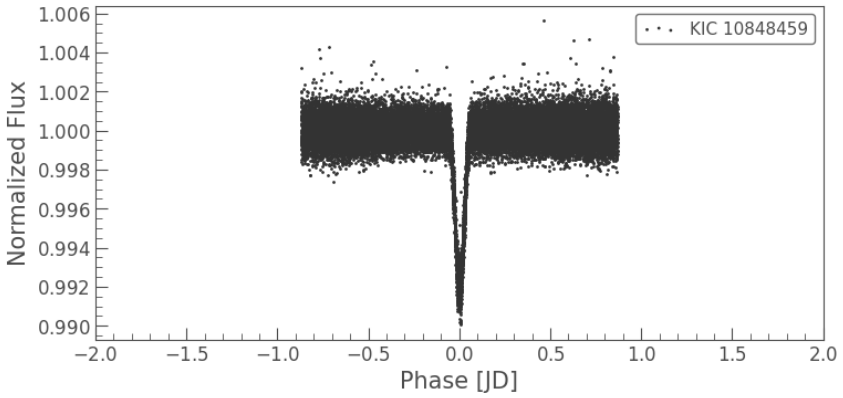
[20] https://scipy.org/

la mayor parte de las ocasiones es necesario realizar gráficos de fase, sobre todo cuando los periodos son mayores.

Los gráficos de fase contienen los datos de flujo de las curvas "plegados" para ajustarse a un período definido (en nuestro caso `period` y `t0`). A estos gráficos se les denomina curvas de luz plegadas. En un gráfico de fase, el eje x no será el tiempo como en una curva de luz de luz sino que representa la fase de 0 a 1 (en la práctica los valores pueden ser un poco más o menos e incluso es posible mostrar dos ciclos completos para mostrar todas las características del objeto). Ver Figura 17.

Se ha desarrollado en este trabajo una rutina en Python que realiza el plegado (*folding*) a partir del periodo de tránsito confirmado de las tablas de la NASA.

Figura 17. *Ejemplo plegado en fase del objeto KIC 10848459.*



Nota. En el gráfico se aprecia claramente el decaimiento de flujo en el periodo especificado. Fuente: Elaboración propia.

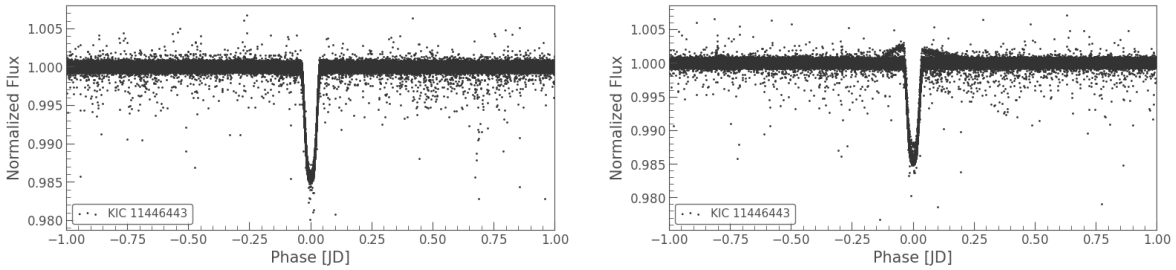
Una cuestión importante a considerar en el desarrollo del presente trabajo, es que la curva de luz plegada tiene el mismo tamaño (en puntos de datos) que la curva no plegada, lo que influye en el tiempo de procesamiento y la cantidad de recursos utilizados para la extracción de parámetros característicos de cada curva procesada.

4.2.4. Tránsitos pares e impares

Los tránsitos pares e impares suelen emplearse para diferenciar exoplanetas de falsos positivos en sistemas automatizados de detección de tránsito (Li et al., 2019). Si los eclipses alternos ("par-impar") de un objetivo muestran diferentes profundidades en forma de V,

entonces esto es suele indicar la existencia de dos estrellas en tránsito (Mallonn et al., 2022), lo que produce un falso positivo (ver Figura 18).

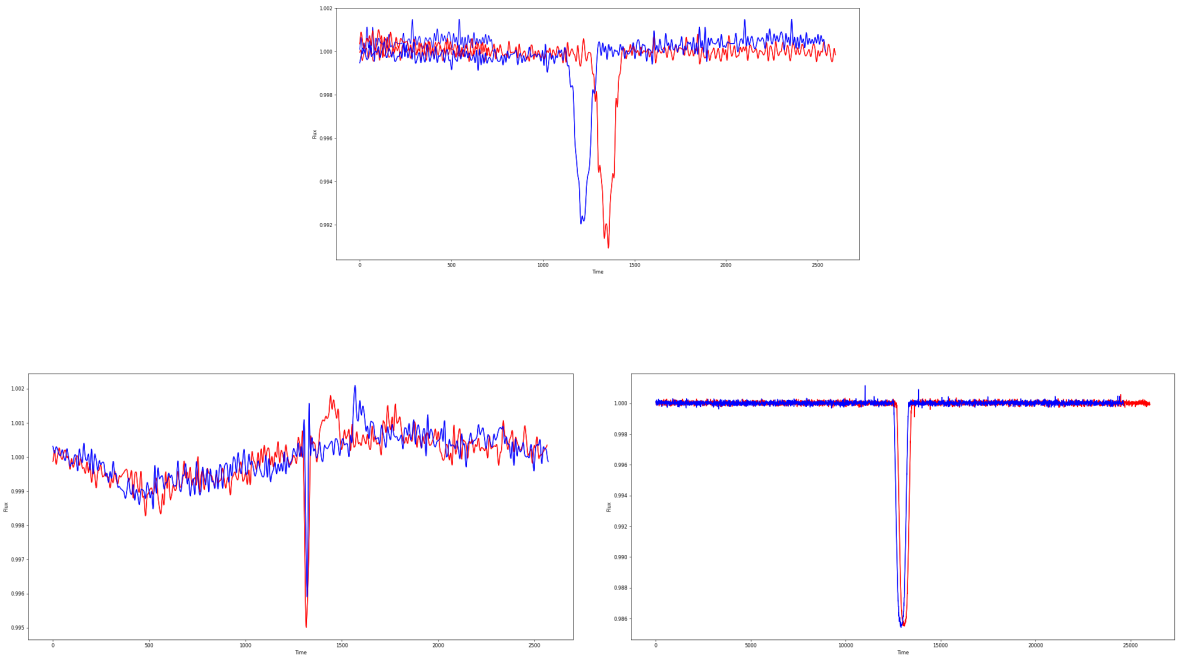
Figura 18. *Ejemplo plegado en fase par (izquierda) e impar (derecha) del objeto KIC 11446443.*



Fuente: Elaboración propia.

Nuestra aproximación a la aplicación de la transformada *wavelet* sobre los tránsitos pares e impares (ver Figura 19), se fundamenta en la hipótesis de que determinados parámetros característicos de las curvas plegadas de fase par e impar, además de la diferente profundidad, pueden ser utilizados para discernir la naturaleza del objeto eclipsante (exoplaneta o sistema binario).

Figura 19. *Representación de la reconstrucción de varias curvas de luz plegadas en fase, utilizando las componentes de baja frecuencia de la transformada wavelet "Symlet 5".*



Nota. Arriba, un sistema binario (falso positivo). Abajo, la transformada *wavelet* "Symlet 5" a las curvas de los objetos KIC 10910878 (izquierda) y KIC 11446443 (derecha) de los exoplanetas Kepler-1 b y Kepler-229 c. Tránsitos impares en rojo y pares en azul. Fuente: Elaboración propia.

4.3. *Dataframes*

4.3.1. *Dataframe* para el análisis

En primer lugar, se seleccionará un conjunto limitado de datos de la fuente utilizada, en concreto los trimestres 2, 3 y 4 (ver Tabla 4). La limitación obedece a dos restricciones del sistema:

- Capacidad de RAM para el procesamiento.
- Tiempo de procesamiento.

Las limitaciones impuestas son:

- Eliminar los candidatos a exoplanetas del *dataset* y evaluar solo los confirmados y los falsos positivos (7508 de 9564).
- Obtener solo tres trimestres de datos de cada curva para permitir el procesamiento de las curvas y extracción de parámetros en Google Colab sin consumir todos los recursos.
- Limitar la obtención de parámetros únicamente a aquellos que se calculan de manera eficiente y no a los que requieren una gran capacidad de cómputo. Para ello se utiliza la clase *EfficientFCParameters*[21] del paquete tsfresh.

Tras limitar la cantidad de datos, se realizaron las siguientes acciones:

- Obtener el plegado de las curvas (tanto el par como el impar).
- Aplicar la transformada *wavelet* (Symlet 5) de siete niveles a las curvas plegadas par e impar. La elección de la "Symlet 5" se debe a los resultados obtenidos de un dataset reducido al que se aplicó el proceso completo para diferentes *wavelets*, siendo el que obtuvo mejores métricas, en comparación con el resto de madres *wavelet* Symlets, Daubechies, Coiflet y Biorthogonal.
- Reconstruir las curvas plegadas eliminando las altas frecuencias, reconstruyendo la señal con cuatro escalas *wavelets* de baja frecuencia. La mejor manera de reconstruir

[21] https://tsfresh.readthedocs.io/en/latest/api/tsfresh.feature_extraction.html#tsfresh.feature_extraction.settings.EfficientFCParameters

una señal con Pywavelets mediante el método *waverec*[22], omitiendo determinados coeficientes *wavelet*, es establecer las matrices correspondientes a las frecuencias que queremos eliminar a cero, ya que la eliminación explícita de las entradas de la lista no es posible.

- Procesar las escalas *wavelets* con tsfresh.
- Filtrar los parámetros obtenidos con tsfresh para obtener el conjunto de características más relevantes (620 de un total de 1566 calculadas). Para el filtrado se usa el algoritmo fresh (FeatuRe Extraction based on Scalable Hypothesis tests), implementado a través del procedimiento *calculate_relevance_table*. Se trata de un algoritmo de extracción de características eficiente y escalable, que filtra las características disponibles con respecto a su importancia para la tarea de clasificación o regresión.

4.3.2. *Dataframe* de entrenamiento y test

Los parámetros obtenidos con tsfresh suelen emplearse con sistemas de clasificación, por lo que se crearon conjuntos de entrenamiento y test, con la función train_test_split[23] del paquete sklearn, que divide matrices o arrays en subconjuntos aleatorios para los datos de entrenamiento y test.

[22] https://pywavelets.readthedocs.io/en/latest/ref/idwt-inverse-discrete-wavelet-transform.html?highlight=np.zeros_like

[23] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

5. Implementación

5.1. Entorno de ejecución

Como entorno de ejecución se ha elegido Google Colab Pro+[24], que tiene un coste de 42,25 €/mes, ya que tiene unas características que lo hacen idóneo para este proyecto, al no disponer de acceso a un supercomputador:

- Ejecución en segundo plano. Permite la ejecución de los *notebooks* de Python aunque el navegador se haya cerrado.
- GPUs y TPUs más rápidas, lo que permite reducir los tiempos de ejecución para poder realizar cambios y pruebas de manera más eficiente.
- 35 GB de memoria, el máximo disponible para evitar el consumo total de la misma con el proceso de extracción de parámetros (ver Apartado 5.2.1).

5.2. Generación del *dataset*

Como se ha comentado en apartados anteriores, la generación del *dataset* de análisis es uno de los aspectos críticos de este trabajo, pues su implementación debe permitir la variación de parámetros (como la elección de *wavelets*), el número de curvas a analizar y el tipo de parámetros característicos a extraer con tsfresh. Y todo ello debe realizarse de manera que sea computacionalmente eficiente e implementable con el entorno de ejecución disponible.

5.2.1. Dataframe para la caracterización

El *dataframe* para la caracterización, comienza con la carga del fichero en formato *csv* acumulativo descargado del NASA EXOPLANET ARCHIVE[25]. Este fichero se carga en un Panda Dataframe[26] para facilitar su procesado:

```
df_koi = pd.read_csv("/cumulative_2022.08.09_10.37.12.csv", skiprows=144)
```

Es necesario omitir las primeras 144 filas de metadatos para obtener el resultado deseado, ya que las primeras 144 filas incluyen información sobre el conjunto de datos y no es hasta la fila 145 donde se obtiene el nombre de las columnas:

[24] https://colab.research.google.com/signup
[25] https://exoplanetarchive.ipac.caltech.edu/cgi-bin/TblView/nph-tblView?app=ExoTbls&config=cumulative
[26] https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html

```
     rowid      kepid       kepoi_name       kepler_name    koi_disposition
0        1   10797460       K00752.01      Kepler-227 b   CONFIRMED
1        2   10797460       K00752.02      Kepler-227 c   CONFIRMED
2        3   10811496       K00753.01      NaN            CANDIDATE
3        4   10848459       K00754.01      NaN            FALSE POSITIVE
4        5   10854555       K00755.01      Kepler-664 b   CONFIRMED
...    ...     ...     ...     ...     ...
9559  9560   10090151       K07985.01      NaN            FALSE POSITIVE
9560  9561   10128825       K07986.01      NaN            CANDIDATE
9561  9562   10147276       K07987.01      NaN            FALSE POSITIVE
9562  9563   10155286       K07988.01      NaN            CANDIDATE
9563  9564   10156110       K07989.01      NaN            FALSE POSITIVE

9564 rows × 141 columns
```

Como se puede apreciar, se obtienen un total de 9564 objetos con 141 columnas de datos. Los candidatos a exoplanetas podrían desvirtuar el análisis de los datos, por lo que se omiten del *dataset*, quedando únicamente 7508 registros de la tabla anterior.

Cada uno de los registros representa un objeto Kepler del que se deben obtener las curvas de luz a través de Lightkurve.

Para procesar dichas curvas de luz a través de tsfresh, este paquete exige que los datos estén en un formato determinado y con una estructura específica, por lo que se debe crear un *dataframe* específico para ir rellenando los datos a medida que son procesados:

```
dataset_fold_total = pd.DataFrame(columns=['id','time','flux','event'])
```

tsfresh requiere un identificador (número que identifica la curva que estamos procesando), una variable temporal (tiempo en la curva de luz), una variable que varía en función de la variable temporal (flujo captado) y un tipo de evento (evento par o impar de las transiciones).

Tabla 5: *Ejemplo del formato requerido por tsfresh.*

id	time	flux	event
1	t1	even(1, t1)	even
1	t2	even(1, t2)	even
1	t3	even(1, t3)	even
1	t1	odd(1, t1)	odd
1	t2	odd(1, t2)	odd
1	t3	odd(1, t3)	odd

2	t1	even(2, t1)	even
2	t2	even(2, t2)	even
2	t3	even(2, t3)	even
2	t1	odd(2, t1)	odd
2	t2	odd(2, t2)	odd
2	t3	y(2, t3)	odd

Fuente: Adaptada de https://tsfresh.readthedocs.io/en/latest/text/data_formats.html

Además, para el filtrado de las características halladas por tsfresh, también se requiere un Panda Dataframe Series[27] de etiquetas (que llamaremos *y*), en el que se especifique si la curva analizada corresponde a un exoplaneta confirmado o a un falso positivo:

```
data = np.array(True)
y = pd.Series(data,index=[0])
```

Así creamos una serie a la que iremos añadiendo valores a medida que vayamos analizando las curvas:

```
0    True
dtype: bool
```

Una vez completa la serie, eliminaremos la primera fila, ya que nuestro identificador siempre comenzará en 1 y no en 0.

Para la creación del *dataset* y extracción de datos se probaron 2 estrategias:

1. Creación de un *dataset* total con los valores pares e impares de flujo para todas las curvas. Una vez se tuvo el *dataset* completo, se calcularon los parámetros tsfresh sobre ese enorme conjunto de datos. Esta estrategia no resultaba eficiente, ya que, para solo 1500 curvas y 3 trimestres para cada una de ellas, se obtenía un dataset de unos 20 millones de registros, lo que computacionalmente tiene un coste demasiado grande desde el punto de vista del procesado. Se intentó la paralelización con Dask[28], pero no es soportado por Google Colab, así que hubo que buscar una segunda estrategia.

[27] https://pandas.pydata.org/docs/reference/api/pandas.Series.html
[28] https://www.dask.org/

	id	time	flux	event	
0	1	393.509231	1.000698	EVEN	
1	1	336.582443	1.000652	EVEN	
2	1	279.656488	1.000594	EVEN	
3	1	412.492070	1.000513	EVEN	
4	1	355.564672	1.000411	EVEN	
...	...	...	...	...	...
20502087	1500	388.257336	1.000091	ODD	
20502088	1500	327.224118	1.000144	ODD	
20502089	1500	266.191548	1.000191	ODD	
20502090	1500	393.345256	1.000226	ODD	
20502091	1500	332.311857	1.000261	ODD	

20502092 rows × 4 columns

2. La segunda estrategia consistió en extraer, de manera iterativa, las características con tsfresh de cada curva descargada y plegada en fase para el evento par e impar. El conjunto de características extraídas es el que se va añadiendo al *dataset*. Con esta estrategia conseguimos aumentar la eficiencia y el consumo de memoria del proceso iterativo del cálculo de parámetros.

	EVEN__sum_values	EVEN__abs_energy	EVEN__mean_abs_change	...
1	4090.029605	4090.059435	190	
2	4085.999780	4086.000580	64	
3	2542.520982	2542.046197	396	
4	4232.298624	4231.598035	309	
5	4316.920416	4316.841168	225	
...	...	...	...	...
7324	4068.094815	4074.239376	1.636	
7325	4078.019895	4078.039878	136	
7326	4078.554018	4078.108840	460	
7327	4138.079626	4138.160072	35	
7328	4025.050624	4025.101294	114	

7328 rows × 1566 columns

El *dataset* resultante de la segunda estrategia, será nuestro *dataset* de análisis. Para el cálculo de las 1566 características de cada una de las 7328 curvas de luz (solo 7328 de las 7508 pudieron descargarse sin errores, bien porque las curvas no tenían datos o no podían ser localizadas por el método de búsqueda utilizado), se han utilizado los parámetros más eficientes en tsfresh y el procesado ha tenido una duración total de 21 horas. La duración del procesado para cada pequeño cambio realizado, es una de las principales limitaciones encontradas a la hora de testear diferentes parámetros, estrategias, *wavelets* y algoritmos.

A través del *dataset* de etiquetado (*y*), se pueden filtrar las características relevantes (aquellas realmente influyentes en una clasificación) extraídas con tsfresh:

```
      ODD_agg_autocorrelation_f_agg_"median"_maxlag
1      307.195
2      685.343
3      -27.238
4      451.114
5      -23.170
...    ...
7328 rows × 620 columns
```

De esta manera se pueden reducir las características halladas (de 1566 a 620), pero como veremos en el Apartado 6, debemos analizar si los mecanismos de clasificación son más precisos utilizando el conjunto de características completo o el filtrado.

5.2.2. *Dataframe* de entrenamiento y test

Como ya se comentó anteriormente, los parámetros obtenidos con tsfresh suelen emplearse con sistemas de clasificación, por lo que se crearán conjuntos de entrenamiento y test a partir de las características extraídas por tsfresh, tanto para el total de los parámetros como para los parámetros filtrados por relevancia:

```
X_full_train, X_full_test, y_train, y_test =
             train_test_split(extracted_features_total, y, test_size=.3)
X_filtered_train, X_filtered_test =
             X_full_train[features_filtered.columns],
             X_full_test[features_filtered.columns]
```

5.3. Caracterización a través de la transformada *wavelet*

5.3.1. Proceso

El procedimiento de aplicación de la transformada *wavelet* a las curvas de luz de la misión Kepler es el siguiente:

1. Descarga de la curva del KIC objetivo a través de Lightkurve:

```
lc_search = lk.search_lightcurve(kic, mission='Kepler')
lc_collection = lc_search[2:4].download_all()
```

Como ya se comentó con anterioridad, se descargan solo los trimestres 2, 3 y 4 para reducir el tiempo de cómputo y el consumo de recursos en el procesamiento posterior.

2. Unión de los trimestres de datos, eliminación de valores anómalos y valores nulos:

```
lc = lc_collection.stitch().remove_outliers(sigma=20, sigma_upper=4)
lc_nonans=lc.remove_nans()
```

3. Cálculo de la curva plegada en fase y extracción de la máscara impar (*odd*) y par (*even*):

```
lc_fold = lc_nonans.fold(period=period, epoch_time=t0)
klc = lc_fold
klc_odd=klc[klc.odd_mask]
x_odd = klc_odd.time.value
sap_flux_odd = klc_odd.flux.value
data_odd = sap_flux_odd
klc_even=klc[klc.even_mask]
x_even = klc_even.time.value
sap_flux_even = klc_even.flux.value
data_even = sap_flux_even
```

4. Aplicación de la transformada discreta *wavelet* “Symlet 5” de 7 niveles (según nuestras pruebas, más de 7 eran innecesarios y menos de 7 no permitían un filtrado adecuado) a los datos pares e impares:

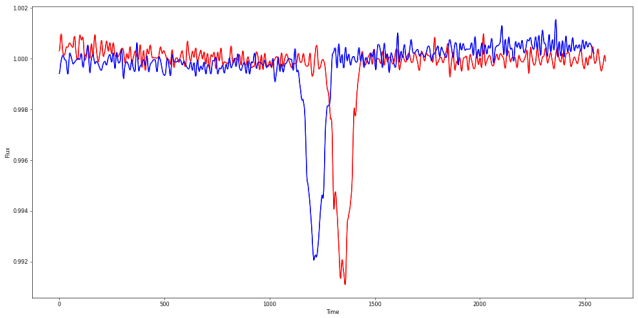
```
wl = pywt.Wavelet("sym5")
coeff_all_odd = pywt.wavedec(data_odd, wl, level=7)
cA6, cD7, cD6, cD5, cD4, cD3, cD2, cD1= coeff_all_odd


w2 = pywt.Wavelet("sym5")
coeff_all_even = pywt.wavedec(data_even, wl, level=7)
cA6, cD7, cD6, cD5, cD4, cD3, cD2, cD1= coeff_all_odd
```

5. Reconstrucción de señal eliminando las componentes de alta frecuencia (Figura 20):

```
coeff_all_odd[-1] = np.zeros_like(coeff_all_odd[-1])
coeff_all_odd[-2] = np.zeros_like(coeff_all_odd[-2])
coeff_all_odd[-3] = np.zeros_like(coeff_all_odd[-3])
recon_odd = pywt.waverec(coeff_all_odd, wavelet= wl)
coeff_all_even[-1] = np.zeros_like(coeff_all_even[-1])
coeff_all_even[-2] = np.zeros_like(coeff_all_even[-2])
coeff_all_even[-3] = np.zeros_like(coeff_all_even[-3])
recon_even = pywt.waverec(coeff_all_even, wavelet= w2)
```

Figura 20. *Reconstrucción impar (en rojo) y par (en azul) de una curva de luz de un sistema binario plegada en fase a la que se le ha aplicado la transformada wavelet discreta "Symlet 5".*



Fuente: Elaboración propia.

5.3.2. Extracción de parámetros con tsfresh

Una vez que ya se ha aplicado la transformada *wavelet* y se ha reconstruido la señal de los plegados en fase par e impar, eliminando las altas frecuencias, se aplica la extracción de parámetros con tsfresh. El proceso se realiza de manera iterativa para cada curva de luz descargada:

```
[ 2034 / 3300 ] descargando curva de luz del objeto KIC 11295426  -  Kepler-68 c
Feature Extraction: 100%|██████████| 2/2 [01:06<00:00, 33.11s/it]
```

La extracción de parámetros se realiza con el siguiente proceso:

1. En primer lugar, se construye un *dataset* para los tránsitos pares siguiendo las especificaciones de tsfresh (ver Apartado 5.2.1):

```
dataset_fold_even_aux['time'] = klc_even.time_original
dataset_fold_even_aux['flux'] = recon_even[0:len(klc_even)]
dataset_fold_even_aux['event'] = 'EVEN'
dataset_fold_even_aux.loc[:,'id'] = i+1
dataset_fold_even_aux.loc[:,'event'] = 'EVEN'
```

2. Se repite el proceso para los tránsitos impares:

```
dataset_fold_odd_aux['id'] = i+1
dataset_fold_odd_aux['time'] = klc_odd.time_original
dataset_fold_odd_aux['flux'] = recon_odd[0:len(klc_odd)]
dataset_fold_odd_aux.loc[:,'id'] = i+1
dataset_fold_odd_aux.loc[:,'event'] = 'ODD'
```

3. Se crea el *dataset* completo que contiene la información de los tránsitos pares e impares para el mismo identificador (*id*) del objeto:

```
dataset_fold_total = dataset_fold_total.append(dataset_fold_even_aux,
ignore_index=True)
dataset_fold_total = dataset_fold_total.append(dataset_fold_odd_aux,
ignore_index=True)
```

4. Se extraen los parámetros eficientes:

```
extracted_features = extract_features(dataset_fold_total,
                                      column_id="id",
                                      column_sort="time",
                                      column_kind="event",
                                      column_value="flux",
                                default_fc_parameter = EfficientFCParameters())
```

5. Se añaden los parámetros al *dataset* de parámetros de cada objeto descargado y procesado de manera iterativa:

```
if i == 0:
     extracted_features_total = extracted_features.copy()
   else:
     extracted_features_total =
extracted_features_total.append(extracted_features, ignore_index=True)
   extracted_features = extracted_features.iloc[0:0]
```

6. Se eliminan los valores nulos y se filtran los parámetros con el *dataset* de etiquetas creado:

```
impute(extracted_features_total)
features_filtered = select_features(extracted_features_total, y)
```

5.3.3. Algoritmos de machine learning

El objetivo que planteamos en este trabajo es crear un modelo capaz de predecir el valor de una variable objetivo (exoplaneta o falso positivo) mediante el aprendizaje de reglas de decisión simples inferidas a partir de las características de los datos (parámetros extraídos con tsfresh sobre curvas de luz plegadas en fase reconstruidas con las componentes de baja frecuencia mediante la transformada *wavelet*).

5.3.3.1. Árbol de decisión

Los árboles de decisión (*Decision Trees*) son métodos de aprendizaje supervisado no paramétrico que se utiliza para la clasificación y la regresión.

Las ventajas de la utilización de los árboles de decisión son las siguientes:

- Fáciles de entender e interpretar. Los árboles pueden visualizarse.
- Requiere poca preparación de los datos.
- Si una situación dada es observable en un modelo, la explicación de la condición se explica fácilmente mediante la lógica booleana.
- Es posible validar un modelo mediante pruebas estadísticas, lo que permite determinar la fiabilidad del modelo.

Y sus desventajas:

- Los árboles demasiado complejos se ajustan bien a los datos, pero no presentan buenos resultados para otros datos con los que no se ha entrenado el sistema (*overfitting*) .
- Pueden ser inestables, ya que pequeñas variaciones en los datos pueden hacer que se genere un árbol completamente diferente.
- Si existen clases dominantes se puede crear sesgo en las predicciones.

Se ha empleado la clase DecisionTreeClassifier[29] del paquete scikit-learn en este proyecto:

```
X_full_train, X_full_test, y_train, y_test = train_test_split(extracted_features,
y_fold_1500, test_size=.3)
classifier_full = DecisionTreeClassifier()
classifier_full.fit(X_full_train, y_train)
print(classification_report(y_test, classifier_full.predict(X_full_test)))
```

5.3.3.2. LightGBM

LightGBM[30] es un framework de potenciación de gradiente (*gradient boosting*) que utiliza algoritmos de aprendizaje basados en árboles. Está diseñado para ser distribuido y es eficiente. La idea fundamental del "*gradient boosting*", es elegir conjuntos de entrenamiento

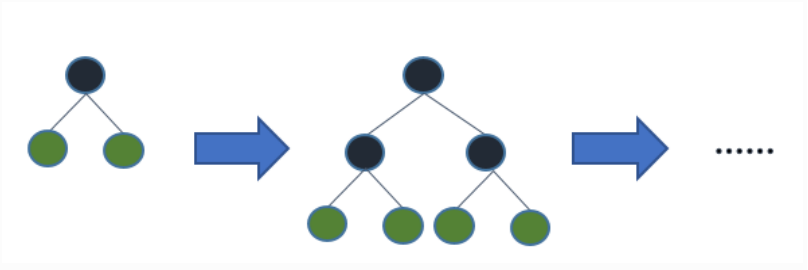
[29] https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier

[30] https://lightgbm.readthedocs.io/

para el algoritmo de aprendizaje base de tal manera que lo obligue a inferir algo nuevo sobre los datos cada vez que se lo llame (*Boosting en Machine Learning con Python*, 2017).

La mayoría de los algoritmos de aprendizaje de árboles de decisión hacen crecer los árboles por niveles (profundidad) (*Features - LightGBM 3.3.2 documentation*, s. f.) , como en la Figura 21.

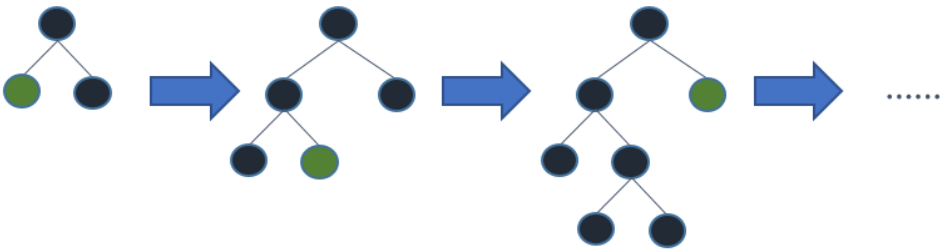
Figura 21. *Crecimiento del árbol de decisión por niveles.*



Fuente: Adaptada de https://lightgbm.readthedocs.io/en/v3.3.2/Features.html

LightGBM hace crecer los árboles en función de las hojas (*best-first*), ver Figura 22. Se elige la hoja con la máxima pérdida delta para crecer. Manteniendo fija la hoja, los algoritmos de hoja tienden a lograr una pérdida menor que los algoritmos de nivel.

Figura 22. *Crecimiento del árbol de decisión por hojas.*



Fuente: Adaptada de https://lightgbm.readthedocs.io/en/v3.3.2/Features.html

Para evitar el sobreajuste si el conjunto de datos es pequeño, se puede limitar la profundidad del árbol.

5.3.3.3. Auto-sklearn

Auto-sklearn[31] es un algoritmo automatizado de aprendizaje automático (*AutoML*). Está basado en scikit-learn y utiliza 15 clasificadores, 14 métodos de preprocesamiento de

[31] https://automl.github.io/auto-sklearn/master/index.html

características y 4 métodos de preprocesamiento de datos, dando lugar a un espacio de hipótesis estructurado con 110 hiperparámetros (Feurer et al., 2019).

Auto-sklearn nos permite determinar qué modelo obtiene mejor resultado para nuestro *dataset*:

```
cls = autosklearn.classification.AutoSklearnClassifier(
                                    ensemble_size=1,
                                    initial_configurations_via_metalearning=0)
cls.fit(X_train, y_train)
predictions = cls.predict(X_test)
```

5.3.3.4. t-SNE

t-SNE[32] (*T-distributed Stochastic Neighbor Embedding*) es un algoritmo diseñado para visualizar conjuntos de datos de alta dimensionalidad. Cuando el número de dimensiones es muy alto, como es nuestro caso, en el que tenemos 1566 dimensiones en el *dataset* de parámetros totales y 620 parámetros filtrados, se recomienda en su documentación utilizar un método de reducción de dimensionalidad previo (como PCA[33]) para reducir el conjunto de datos a un número de dimensiones razonable (por ejemplo 50), lo que reducirá el ruido y acelerará la ejecución del algoritmo (*t-SNE*, s. f.).

t-SNE se ejecuta en dos pasos:

- Primero, se construye una distribución de probabilidad sobre parejas de muestras en el espacio original, de forma tal que las muestras semejantes reciben alta probabilidad de ser escogidas, mientras que las muestras muy diferentes reciben baja probabilidad de ser escogidas. El concepto de "semejanza" se basa en la distancia entre puntos y densidad en las proximidades de un punto.
- Segundo, t-SNE lleva los puntos del espacio de alta dimensionalidad al espacio de baja dimensionalidad de forma aleatoria, define una distribución de probabilidad semejante a la vista en el espacio destino (el espacio de baja dimensionalidad), y minimiza la denominada divergencia Kullback-Leibler (Kullback y Leibler, 1951) entre las dos distribuciones con respecto a las posiciones de los puntos en el mapa (la

[32] https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html
[33] https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html

divergencia de Kullback-Leibler mide la similitud o diferencia entre dos funciones de distribución de probabilidad).

Scikit-Learn implementa este algoritmo mediante la clase sklearn.manifold.TSNE[34]. Esta clase incluye una serie de parámetros que permiten definir el comportamiento del algoritmo:

- *n_components* (2 por defecto): Dimensiones del conjunto transformado.
- *perplexity* (30 por defecto): Es recomendable utilizar valores entre 5 y 50 (mayor cuanto mayor sea el *dataset*).
- *early_exaggeration* (12 por defecto): Controla la distancia entre bloques semejantes en el espacio final.
- learning_rate (200 por defecto): Habitualmente en el rango (10-1000).
- *n_iter* (1000 por defecto): Número máximo de iteraciones para la optimización. Debe ser, al menos, 250.
- *metric*: métrica para la medición de las distancias.
- *method*: algoritmo a usar para el cálculo del gradiente.

En este trabajo hemos aplicado t-SNE al *dataset* de parámetros filtrados a los que se les ha aplicado una reducción de dimensionalidad, para la evaluación de la formación de clústers.

5.3.3.5. UMAP

Uniform Manifold Approximation and Projection (UMAP) es una técnica de reducción de dimensiones que puede utilizarse para la visualización de conjuntos de datos de alta dimensionalidad, de forma similar a t-SNE, pero también para la reducción de dimensiones no lineales. El algoritmo se basa en tres supuestos sobre los datos:

1. Los datos se distribuyen uniformemente en una variedad riemanniana ("Riemannian curvature", 2001).
2. La métrica riemanniana es localmente constante (o puede aproximarse como tal).
3. El conjunto está localmente conectado.

A partir de estos supuestos es posible modelar el conjunto con una estructura topológica difusa. La estructura se encuentra buscando una proyección de baja dimensión de los datos

[34] https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html

que tenga la estructura topológica difusa equivalente más cercana posible (McInnes et al., 2018).

Utilizaremos UMAP sobre el conjunto de parámetros característicos filtrados para visualizar los clústers modelados de manera no supervisada.

5.3.4. Métricas

Se han utilizado las siguientes métricas para evaluar los resultados obtenidos.

5.3.4.1. Precisión de la clasificación.

La función accuracy_score[35] calcula la precisión, ya sea la fracción (por defecto) o el recuento de predicciones correctas.

En la clasificación multi etiqueta, la función devuelve la precisión del subconjunto. Si todo el conjunto de etiquetas predichas para una muestra coincide estrictamente con el conjunto verdadero de etiquetas (*y*), entonces la precisión del subconjunto es 1.0; en caso contrario, es 0.0.

Si $\widehat{y}_i$ es el valor predicho de la muestra i e y_i es el valor verdadero correspondiente a dicha muestra, entonces la fracción de predicciones correctas sobre n muestras se define como (2)

$$accuracy(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} 1(\hat{y}_i = y_i) \qquad (2)$$

Donde $1(x)$ es la función característica que mapea los elementos de un subconjunto de datos a 1 y el resto a 0.

5.3.4.2. Matriz de confusión.

La función matriz_de_confusión[36] (Figura 23) evalúa la precisión de la clasificación calculando la matriz de confusión a cada fila correspondiente a la clase verdadera. La entrada

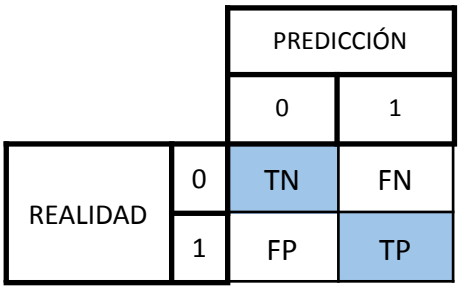
[35] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html
[36] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html

i, j en una matriz de confusión es el número de observaciones que realmente están en el grupo i , pero que se predice que están en el grupo j.

En una clasificación binaria, como es nuestro caso, se pueden dar las siguientes predicciones de los modelos aplicados:

- Número de verdaderos negativos: TN.
- Número de falsos negativos: FN.
- Verdaderos positivos: TP.
- Falsos positivos: FN.

Figura 23. *Matriz de confusión.*



Fuente: Elaboración propia.

5.3.4.3. F1 score.

La puntuación F1 (*F1 score*) puede interpretarse como una media armónica de la precisión y la sensibilidad, donde la puntuación F1 alcanza su mejor valor en 1 y la peor puntuación en 0. La contribución relativa de la precisión y la sensibilidad, a la puntuación F1 son iguales. La fórmula de la puntuación F1 es:

$F1 = 2\frac{Precisión \cdot Sensibilidad}{Precisión + Sensibilidad}$, donde $Precisión = \frac{TP}{FP+TP}$ y $Sensibilidad = \frac{TP}{FN+TP}$

Esta métrica ofrece una medida más precisa del desempeño de los algoritmos de clasificación que la precisión (*accuracy score*), sobre todo en clases desbalanceadas. Hemos usado esta métrica, debido a que las clases en nuestro *dataset* están ligeramente desbalanceadas, ya que tenemos 4681 falsos positivos y 2647 exoplanetas confirmados.

6. Resultados obtenidos

Los resultados obtenidos y mostrados en el presente apartado se han obtenido en base a 2 *datasets*:

- *Dataset* de 7328 elementos (curvas de luz), donde cada fila se corresponde con los parámetros característicos extraídos (1566 dimensiones) con tsfresh para cada curva plegada en fase y reconstruida con la transformada *wavelet*. A este *dataset* lo hemos denominado `extracted_features_total`.
- *Dataset* completo de 7328 elementos, donde cada fila se corresponde con los parámetros característicos extraídos con tsfresh (los 1566 anteriores), pero filtrados en base a su relevancia (620 dimensiones). A este *dataset* lo denominamos `features_filtered`.

Para la validación de nuestra hipótesis principal, que la transformada *wavelet* puede ser utilizada para discernir la naturaleza del objeto eclipsante, se construyeron dos conjuntos de datos diferenciados: uno con las características extraídas con tsfresh de las curvas plegadas en fase con los datos en bruto y otro con las características también de tsfresh pero obtenidas de los datos filtrados y reconstruidos mediante la transformada *wavelet*. Estos dos conjuntos de datos fueron utilizados en los algoritmos de clasificación DecisionTreeClassifier, LightGBM y Auto-sklearn, obteniéndose una mejora en los datos a los que se había aplicado la transformada *wavelet* de manera previa a la extracción de parámetros característicos. Ver la Tabla 6.

Tabla 6: *Precisión obtenida sobre los dos conjuntos de datos obtenidos.*

Algoritmo	Precisión con *dataset* sin procesamiento *wavelet*	Precisión con *dataset* con procesamiento *wavelet*	Mejora
DecisionTreeClassifier	66%	71%	5%
LightGBM	75%	80%	5%
Auto-sklearn	74%	78%	4%

Fuente: Elaboración propia.

Aunque la mejora obtenida no es lo suficientemente significativa, la aplicación de la transformada *wavelet* de manera previa a la extracción de los parámetros característicos de

las curvas plegadas en fase, ha mejorado la clasificación de curvas y contribuido de manera positiva a diferenciar exoplanetas de falsos positivos.

Además, debemos tener en cuenta:

- Que se ha empleado un conjunto de datos reducido de 7328 curvas reales con exoplanetas confirmados (generalmente mediante el método de la velocidad radial) y falsos positivos. El conjunto de datos podría aumentarse utilizando curvas simuladas en las que se introducen tránsitos de manera artificial.
- Que no se han extraído aquellos parámetros computacionalmente más costosos con tsfresh, debido a las limitaciones de recursos.
- Que no se han creado *datasets* específicos para diferentes tipologías de estrellas observadas, por ejemplo en base a su clase y tipo espectral.

Tal y como se puede apreciar en la Tabla 7, la aplicación de LightGBM ofrece los mejores resultados, tanto para el conjunto de parámetros total como para los filtrados, siendo este último conjunto el que alcanza una puntuación F1 mayor, de aproximadamente un 81%.

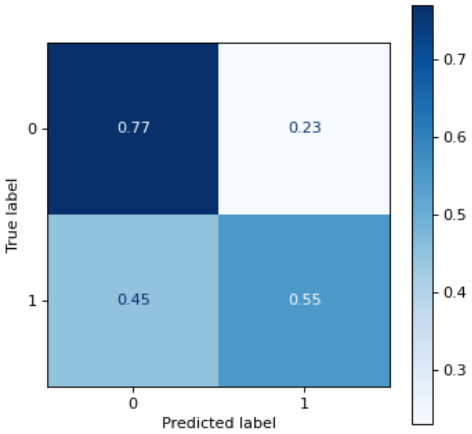
Tabla 7: *Precisión y puntuación F1 obtenidas sobre los dos conjuntos de datos obtenidos con diferentes algoritmos de clasificación.*

	extracted_features_total		features_filtered	
Algoritmo	Accuracy score	F1 score	Accuracy score	F1 score
DecisionTreeClassifier	0.6926	0.6917	0.7262	0.7266
LightGBM	0.8008	0.7981	0.8103	0.8083
Auto-sklearn	0.8008	0.8018	0.7976	0.7798

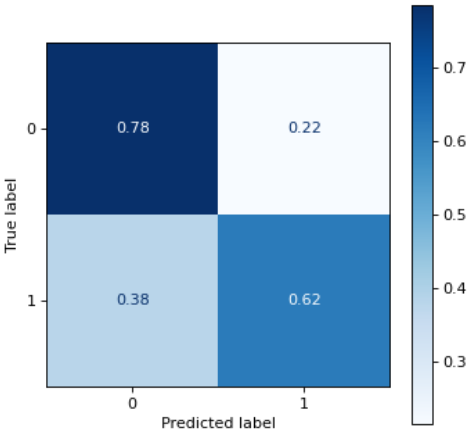
Fuente: Elaboración propia.

Las matrices de confusión se muestran en la Figura 24, para DecisionTreeClassifier, LightGBM y Auto-sklearn.

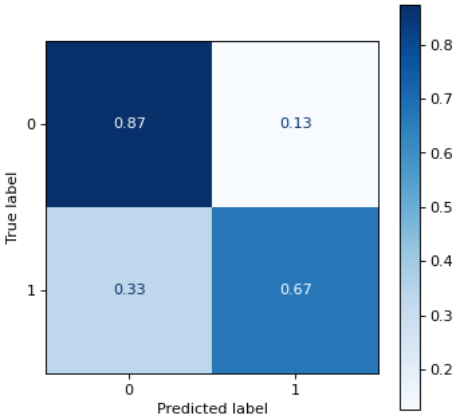
Figura 24: *Matrices de confusión obtenidas para los dos conjuntos de datos analizados (extracted_features_total y features_filtered), para los algoritmos DecisionTreeClassifier, LightGBM y Auto-sklearn.*



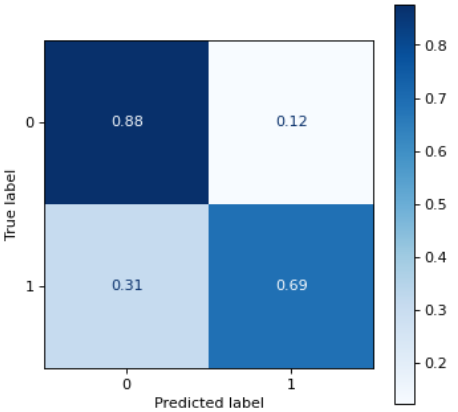
DecisionTreeClassifier (extracted_features_total)



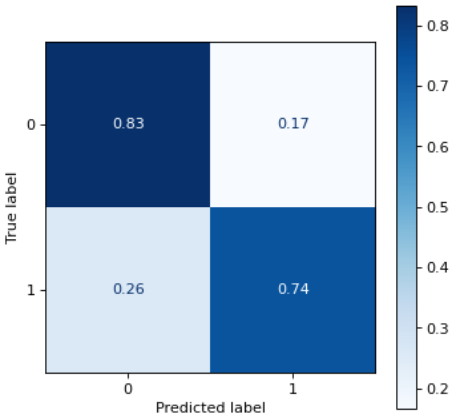
DecisionTreeClassifier (features_filtered)



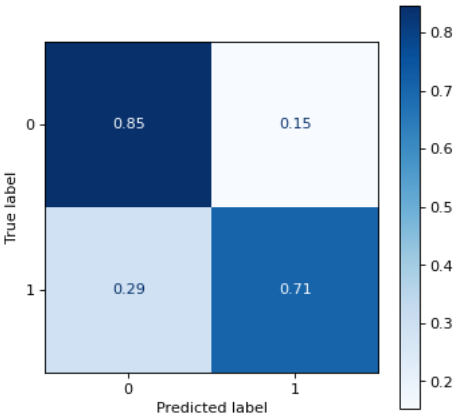
LightGBM (extracted_features_total)



LightGBM (features_filtered)



Auto-sklearn (extracted_features_total)



Auto-sklearn (features_filtered)

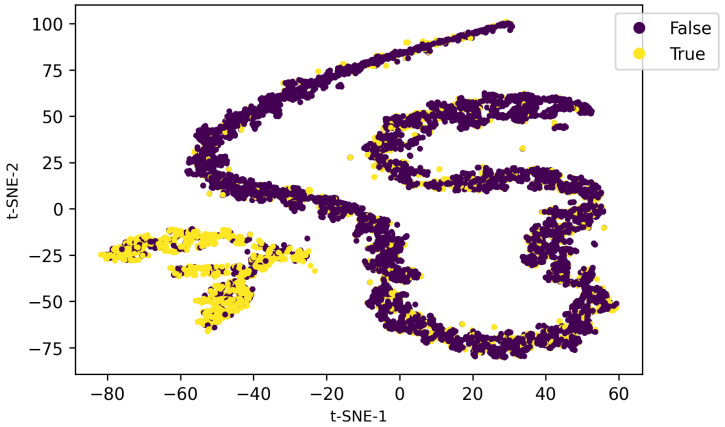
Fuente: Elaboración propia.

El resultado obtenido por el algoritmo LightBGM del 81% en el F1-score es bueno, pero no es excelente al estar por debajo del 90%. Aun así, haber obtenido un resultado superior al 80% sobre un conjunto de datos heterogéneo (observaciones de estrellas de diferente tipo espectral y distintas tipologías de exoplanetas), creado a partir de la extracción de parámetros característicos de las 7328 curvas plegadas en fase, sobre las que se ha realizado una reconstrucción mediante la transformada wavelet; sugiere que es probable una mejora en los resultados si se extraen aquellos parámetros más exigentes computacionalmente con tsfresh.

Otro aspecto abordado en este trabajo es que se ha empleado el aprendizaje no supervisado para estudiar las estructuras y relaciones presentes en nuestros datos, ya que es importante comprender los patrones y la estructura subyacente de los datos analizados y obtener información explorando los datos, ya sea trazando o buscando patrones predominantes. El aprendizaje no supervisado suele ser una parte importante de la etapa de análisis exploratorio de datos. Con el aprendizaje estadístico no supervisado, hay entradas pero no hay salida de supervisión; no obstante, podemos aprender relaciones y estructura de esos datos (Hastie et al., 2008). El análisis tiende a ser más subjetivo que en el aprendizaje supervisado, ya que no existe una forma estándar de evaluar los resultados mediante validación cruzada o conjuntos de datos independientes. Además, la mayoría de los algoritmos de aprendizaje automático supervisado no se adaptan bien al problema de la dimensionalidad (Brownlee, 2020).

En este trabajo, de la aplicación de t-SNE con reducción de dimensionalidad PCA, hemos obtenido resultados donde se pueden apreciar 2 clústers, pero con mucho ruido, lo que indica que no se pueden extraer grupos claramente definidos a partir de los conjuntos de datos utilizados (ver Figura 25).

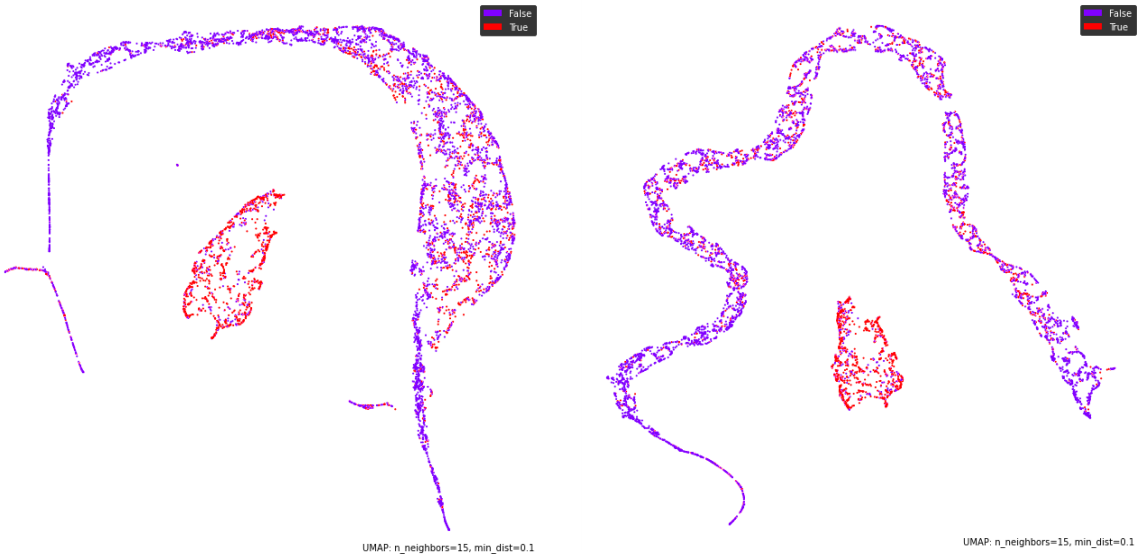
Figura 25: *Visualización de la dimensionalidad de los clusters generados por el método t-SNE sobre el conjunto de datos filtrados de características extraídas con tsfresh (features_filtered). Falsos positivos en violeta.*



Fuente: Elaboración propia

Para extender el análisis de la dimensionalidad, se ha usado también el algoritmo UMAP sobre los datos filtrados y sobre los datos filtrados con reducción de dimensionalidad (ver Figura 26). Con este algoritmo se visualizan mejor los clústers formados, pero también el ruido y los outliers de los grupos, siendo la visualización sobre el conjunto de datos con reducción de dimensionalidad, muy similares a los obtenidos por t-SNE.

Figura 26: *Visualización de la dimensionalidad de los clusters generados por el método UMAP sobre el conjunto de parámetros extraídos con tsfresh.*



Nota. A la izquierda sobre los 620 parámetros (features_filtered) y a la derecha con reducción de dimensionalidad. Falsos positivos en azul, exoplanetas confirmados en rojo. Fuente: Elaboración propia.

7. Conclusiones

Inicialmente nos planteamos si era posible utilizar la herramienta matemática transformada *wavelet* para establecer una caracterización de las curvas de luz, que permitiera discernir la naturaleza de los objetos eclipsantes que pudieran contener.

Los resultados obtenidos muestran que la caracterización implementada (ver Anexo A) permite obtener parámetros útiles para diferenciar, de manera automática, eclipses binarios que suelen ser detectados como exoplanetas (falsos positivos), con una precisión del 81%.

Se ha podido corroborar que el método desarrollado funciona de manera aceptable con un *dataset* heterogéneo de datos. Además, se ha desarrollado un *notebook* en Python para el procesado de los datos procedentes de los catálogos de la NASA, se han aplicado algoritmos de aprendizaje automático y se han establecido métricas para la evaluación de resultados.

Este trabajo es una primera aproximación a la identificación de la naturaleza de los objetos eclipsantes que están presentes en las curvas de luz de misiones como Kepler, mediante la extracción de parámetros característicos de dichas curvas de luz procesadas y filtradas a través de la transformada *wavelet*.

Aunque los resultados son buenos, no son concluyentes, por lo que se requiere la evaluación de más parámetros característicos y la ampliación del conjunto de datos a través de curvas simuladas.

Cabe destacar que las estimaciones iniciales realizadas sobre la capacidad y tiempo de procesamiento necesarias para desarrollar este proyecto, fueron bastante optimistas. El tratamiento de miles de curvas de luz y su procesado, requieren una gran cantidad de recursos y tiempo de procesamiento, lo que ha condicionado la realización de extracciones de parámetros complejos y los procesados completos (todos los trimestres observados) de las curvas de luz. Aún así, a través de este trabajo de fin de estudios se han cumplido con creces las expectativas de aprendizaje y la aplicación de los conocimientos adquiridos en el Máster Universitario en Astrofísica y Técnicas de Observación en Astronomía.

Además, se debe reseñar que con este proyecto se ha obtenido una visión completa del potencial de Python en el tratamiento y análisis de grandes volúmenes de datos procedentes de misiones espaciales y disponibles en los observatorios virtuales.

Por último, comentar que este trabajo ha sido el origen de un póster, titulado *Analysis of Kepler light curves using the wavelet transform to discriminate with machine learning the astrophysical nature of the eclipsing object*, aceptado por la Sociedad Española de Astronomía (SEA), para su publicación en la XV Reunión Científica de la SEA, celebrada en La Laguna entre el 5 y el 9 de septiembre de 2022.

Referencias bibliográficas

5 Ways to Find a Planet. (s. f.). Exoplanet Exploration: Planets Beyond our Solar System. https://exoplanets.nasa.gov/alien-worlds/ways-to-find-a-planet/#/2

Boosting en Machine Learning con Python. (2017, 10 de junio). Raul E. Lopez Briega. https://relopezbriega.github.io/blog/2017/06/10/boosting-en-machine-learning-con-python/

Bravo, J. P., Roque, S., Estrela, R., Leão, I. C. y De Medeiros, J. R. (2014). Wavelets: a powerful tool for studying rotation, activity, and pulsation inKeplerand CoRoT stellar light curves. *Astronomy & Astrophysics*, *568*, A34. https://doi.org/10.1051/0004-6361/201323032

Brownlee, J. (2020, 6 de mayo). *Introduction to Dimensionality Reduction for Machine Learning*. Machine Learning Mastery. https://machinelearningmastery.com/dimensionality-reduction-for-machine-learning/

COROT overview. (s. f.). European Space Agency. https://www.esa.int/Science_Exploration/Space_Science/COROT_overview

Features — LightGBM 3.3.2 documentation. (s. f.). Welcome to LightGBM's documentation! — LightGBM 3.3.2 documentation. https://lightgbm.readthedocs.io/en/v3.3.2/Features.html

Feurer, M., Klein, A., Eggensperger, K., Springenberg, J. T., Blum, M. y Hutter, F. (2019). Auto-sklearn: Efficient and Robust Automated Machine Learning. En Automated Machine Learning (pp. 113–134). Springer International Publishing. https://doi.org/10.1007/978-3-030-05318-5_6

Gaudi, B. S., Seager, S. y Mallen-Ornelas, G. (2005). On the Period Distribution of Close-in Extrasolar Giant Planets. The Astrophysical Journal, 623(1), 472–481. https://doi.org/10.1086/428478

Hastie, T., Tibshirani, R. y Friedman, J. (2008). Introduction. En *The Elements of Statistical Learning* (pp. 1–8). Springer New York. https://doi.org/10.1007/b94608_1

Kepler and K2 Mission overview. (s. f.). https://www.nasa.gov/mission_pages/kepler/overview/index.html

Kullback, S. y Leibler, R. A. (1951). On Information and Sufficiency. The Annals of Mathematical Statistics, 22(1), 79–86. https://doi.org/10.1214/aoms/1177729694

Li, J., Tenenbaum, P., Twicken, J. D., Burke, C. J., Jenkins, J. M., Quintana, E. V., Rowe, J. F. y Seader, S. E. (2019). Kepler Data Validation II–Transit Model Fitting and Multiple-planet Search. Publications of the Astronomical Society of the Pacific, 131(996), 024506. https://doi.org/10.1088/1538-3873/aaf44d

Mallonn, M., Poppenhaeger, K., Granzer, T., Weber, M. y Strassmeier, K. G. (2022). Detection capability of ground-based meter-sized telescopes for shallow exoplanet transits. Astronomy & Astrophysics, 657, A102. https://doi.org/10.1051/0004-6361/202140599

Mayor, M. y Queloz, D. (1995). A Jupiter-mass companion to a solar-type star. Nature, 378(6555), 355–359. https://doi.org/10.1038/378355a0

McInnes, L., Healy, J., Saul, N. y Großberger, L. (2018). UMAP: Uniform Manifold Approximation and Projection. *Journal of Open Source Software*, *3*(29), 861. https://doi.org/10.21105/joss.00861

National Aeronautics and Space Administration. (21 de marzo de 2021). *Cosmic milestone: NASA confirms 5,000 Exoplanets*. https://exoplanets.nasa.gov/news/1702/cosmic-milestone-nasa-confirms-5000-exoplanets/

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

Régulo, C., Almenara, J. M., Alonso, R., Deeg, H. y Roca Cortés, T. (2007). TRUFAS, a wavelet-based algorithm for the rapid detection of planetary transits. *Astronomy & Astrophysics*, *467*(3), 1345–1352. https://doi.org/10.1051/0004-6361:20066827

Riemannian curvature. (2001). En *Geometry of Manifolds* (pp. 161–184). American Mathematical Society. https://doi.org/10.1090/chel/344/09

Robichon, N & Arenou, Frédéric. (2000). HD 209458 planetary transits from Hipparcos photometry. Astron. Astrophys. 355. 295-298.

Sahlmann, J., Lazorenko, P. F., Ségransan, D., Martín, E. L., Queloz, D., Mayor, M. y Udry, S. (2013). Astrometric orbit of a low-mass companion to an ultracool dwarf. *Astronomy & Astrophysics*, *556*, A133. https://doi.org/10.1051/0004-6361/201321871

Savitzky, A. y Golay, M. J. E. (1964). Smoothing and Differentiation of Data by Simplified Least Squares Procedures. *Analytical Chemistry*, *36*(8), 1627–1639. https://doi.org/10.1021/ac60214a047

Starck, J.-L., Murtagh, F. D. y Bijaoui, A. (1994). Image restoration with noise suppression using a wavelet transform and a multiresolution support constraint. En T. J. Schulz y D. L. Snyder (Eds.), SPIE's 1994 International Symposium on Optics, Imaging, and Instrumentation. SPIE. https://doi.org/10.1117/12.188035

t-SNE. (s. f.). Laurens van der Maaten. https://lvdmaaten.github.io/tsne/

Talebi, S. (2020, 21 de diciembre). *The Wavelet Transform*. Medium. https://towardsdatascience.com/the-wavelet-transform-e9cfa85d7b34

Torrence, C. (s. f.). *wavelets/wave_python*. GitHub. https://github.com/chris-torrence/wavelets/tree/main/wave_python

Torrence, C. y Compo, G. P. (1998). A Practical Guide to Wavelet Analysis. *Bulletin of the American Meteorological Society*, *79*(1), 61–78. https://doi.org/10.1175/1520-0477(1998)079%3C0061:apgtwa%3E2.0.co;2

Wavelet Transforms in MATLAB. (s. f.). MathWorks - Creadores de MATLAB y Simulink - MATLAB y Simulink - MATLAB & Simulink. https://es.mathworks.com/discovery/wavelet-transforms.html

Anexo A. Enlaces a código y ficheros.

Enlace al *notebook* de creación del *dataset*:

https://colab.research.google.com/drive/1fp2GpHe3nDXewOBhFERQ9OKkUfuNjfUQ?usp=sharing

Enlace al *notebook* de clasificación:

https://colab.research.google.com/drive/1LVlSLSq7LdMjXFoR2b3OYdgfbfKfyPiP?usp=sharing

Enlace a los ficheros de datos necesarios para ejecutar el *notebook*:

https://drive.google.com/drive/folders/1gzJlmi45NA8CicxfrpVnLhgknVtnuewM?usp=sharing