



UNIVERSIDAD DE CONCEPCIÓN  
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS

# Aplicaciones de Machine Learning en el campo de variabilidad estelar.

*Desarrollo de métodos para la identificación de señales planetarias.*

**Por: Felipe Ignacio Burgos Rubilar**

Tesis presentada a la Facultad de Ciencias Físicas y Matemáticas de la  
Universidad de Concepción para optar al grado académico de Magíster en  
Ciencias con Mención en Astronomía

Noviembre 2022

Concepción, Chile

**Profesores Guía:**  
**Ronald Menickent.**  
**Nicola Astudillo.**  
**Pierluigi Cerulo.**



© 2022, Felipe Burgos

Se autoriza la reproducción total o parcial, con fines académicos, por cualquier medio o procedimiento, incluyendo la cita bibliográfica del documento



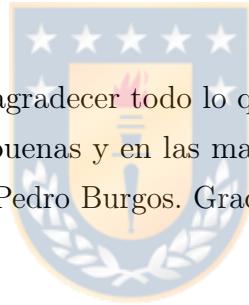
Pedro, Silvia, Matías, las estrellas brillan por ustedes.

## AGRADECIMIENTOS

Quiero agradecer en primer lugar a mi grupo cercano. Primero a mis amigos María Javiera Parot, Eduardo Medel, Yocelyn Machuca, Ricardo Meza y Brian del Valle. A mis colegas astrónomos, Javiera Soto, Daniel Gaete y Camila Huerta. Contar con ustedes ha sido una experiencia increíble. A mis supervisores de tesis, los profesores Ronald Menickent, Nicola Astudillo Defru y principalmente al profesor Pierluigi Cerulo. Su apoyo y sobre todo su paciencia han hecho posible llevar a cabo este proyecto.

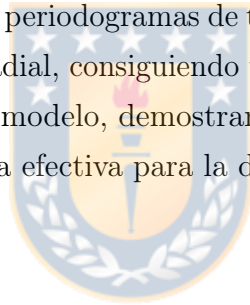
Mis padrinos, Alejandro Ruiz y Karina Ruiz, junto con mi Abuela Rosa Cortes. Han hecho posible para mí el vivir y estudiar en la ciudad de Concepción. Gracias por todo su apoyo.

No existen palabras para agradecer todo lo que mi familia ha hecho por mí, han estado junto a mí en las buenas y en las malas, mi hermano Matías, mi madre Silvia Rubilar y mi padre Pedro Burgos. Gracias a ellos vivo este sueño, por ellos sigo soñando.



## Resumen

La detección y caracterización de planetas extrasolares (exoplanetas) representa uno de los mayores desafíos en la astrofísica moderna y en el análisis de datos astronómicos. Espectroscopios como el High Accuracy Radial velocity Planet Searcher (HARPS) han recolectado observaciones desde el año 2003 y uno de los resultados más interesantes a aparecido en las estrellas de clase M. En particular, las enanas M son excelentes candidatos para encontrar planetas rocosos en la zona habitable. Usando observaciones de velocidad radial telescopio HARPS para estrellas de clase M con detecciones de exoplanetas confirmadas, hemos entrenado modelos de aprendizaje de máquinas de dos tipos (Máquinas de Vector de Soporte y Árboles Aleatorios) con el fin de crear herramientas automatizadas para detectar la presencia de señales planetarias con un alto grado de confianza. Hemos entrenado estos modelos con periodogramas de tipo Lomb-Scargle derivados de las observaciones de velocidad radial, consiguiendo una exactitud del 85 % y un Recall del 94 % con nuestro mejor modelo, demostrando que se puede utilizar el aprendizaje de máquinas de manera efectiva para la detección planetaria a partir de series de velocidad radial.



**Keywords** – Exoplanetas, Metodos de Deteccion, Metodos Estadisticos, Suport Vector Machine, Random Forest, Machine Learning

# Índice general

<b>AGRADECIMIENTOS</b>	<b>I</b>
<b>Resumen</b>	<b>II</b>
<b>1. Introducción</b>	<b>1</b>
1.1. Nuestro Universo . . . . .	1
<b>2. Marco Teórico</b>	<b>4</b>
2.1. Sistemas Exoplanetarios . . . . .	4
2.2. Técnicas de Detección . . . . .	5
2.2.1. Orbitas . . . . .	5
2.2.1.1. Anomalías Orbitales . . . . .	5
2.2.1.2. Elementos orbitales . . . . .	7
2.2.1.3. Leyes de Kepler . . . . .	8
2.2.2. Método de Velocidades Radiales . . . . .	10
2.3. Análisis de Series Temporales . . . . .	12
2.3.0.1. Transformada de Fourier . . . . .	13
2.3.0.2. Transformada Discreta de Fourier . . . . .	14
2.3.0.3. Periodograma de Lomb-Scargle . . . . .	15
2.4. Modelos de Clasificación . . . . .	17
2.4.1. Clasificadores Lineales . . . . .	17
2.4.1.1. Máquinas de vector de soporte . . . . .	20
2.4.1.2. Kernels . . . . .	22
2.4.2. Árboles de Decisión . . . . .	25
2.4.3. Random Forest . . . . .	25
<b>3. Metodología</b>	<b>28</b>
3.1. Datos utilizados . . . . .	28
3.2. Periodogramas . . . . .	29
3.3. Pre-Procesamiento . . . . .	29
3.4. Búsqueda de Rejilla . . . . .	30
3.5. Entrenamiento . . . . .	32
<b>4. Análisis</b>	<b>35</b>
4.1. Resultados . . . . .	35
4.1.1. Exactitud . . . . .	36

---

4.1.2. Precisión . . . . .	36
4.1.3. Recall . . . . .	37
4.1.4. Puntaje F1 . . . . .	38
4.1.5. Tablas . . . . .	39
<b>5. Discusión</b>	<b>41</b>
5.1. Importancia de características . . . . .	41
5.2. Número de Observaciones para las estrellas en nuestro estudio. . .	44
5.3. Comparación con otros resultados . . . . .	45
<b>6. Conclusión</b>	<b>47</b>
6.1. Conclusión . . . . .	47
<b>Referencias</b>	<b>49</b>
<b>Apéndices</b>	<b>52</b>
<b>A. Tablas y Códigos utilizados en este trabajo</b>	<b>52</b>
A1. Tabla Parámetros Principales . . . . .	52
A2. Principales códigos desarrollados para este trabajo . . . . .	55
A2.1. PeriodogramScript.py . . . . .	55
A2.2. Equalizer.py . . . . .	55
A2.3. Classifier.py . . . . .	58



# Índice de cuadros

4.1.1.Métricas de desempeño para los modelos de tipo Random Forest.	40
4.1.2.Métricas de desempeño para los modelos de tipo Support Vector Machine. . . . .	40
A1.1.Parámetros Principales de las estrellas del grupo PC. . . . .	54





# Índice de figuras

2.2.1. Diagrama ilustrando los principales parámetros utilizados para caracterizar un movimiento orbital. Podemos agrupar estos parámetros en 2 grupos. El primero son los parámetros que definen el tamaño y forma de la órbita ( $a$ y $e$ ), mientras que los parámetros ( $\Omega, \omega, i$ ) representan la proyección de la órbita real respecto a nosotros como observadores. Creditos: <a href="#">Shin et al. (2015)</a> . . . . .	8
2.2.2. Figura mostrando la forma de diversas curvas de velocidad radial que comparte un mismo valor de $K, P$ y $t_0$ . Cada columna comparte un mismo valor para la excentricidad, mientras que cada fila comparte el mismo valor de $\omega$ . Podemos notar como los valores de $\omega$ y $e$ condicionan la forma de la curva de velocidad radial. Créditos: <a href="#">Wright and Gaudi (2012)</a> . . . . .	11
2.4.1. Ejemplo bidimensional de un caso en el cual no es posible aplicar un clasificador lineal para separar ambas clases (+ y -). Como se puede apreciar, no existe línea recta capaz de separar a ambos grupos, por lo que una curva polinomial se adecua mejor a esta tarea. Figura Obtenida de "Introduction to Machine Learning" de Miroslav Kubat (2017) . . . . .	18
2.4.2. Ejemplo del uso de una máquina de vector de soporte (Modelo SVM). En esta técnica, se busca utilizar el hiperplano que este a la mayor distancia (margen) posible de todos los elementos del conjunto de entrenamiento. Figura Obtenida de "Understanding Random Forest: From Theory to Practice" de Gilles Louppe (2014)	21
2.4.3. Árbol de decisiones generado como parte de este trabajo. El nodo raíz ha seleccionado la característica número 48 y ha elegido como condicional que el valor de esta característica supere un cierto umbral. Cada nodo subsiguiente selecciona nuevas características hasta que los nodos hoja contienen exclusivamente objetos de una misma clase. El color del nodo es indicativo de su contenido, naranja para la clase <b>PC</b> y azul para la clase <b>NP</b> . . . . .	26

3.3.1. Proceso de extracción de valores de peak aplicado a la estrella LHS 1140. Este procedimiento consiste en identificar el valor máximo global en el periodograma, para acto seguido buscar los mínimos locales más cercanos a este valor máximo. De esta manera se consigue identificar el peak de mayor valor presente en el periodograma. El siguiente paso es guardar el valor máximo identificado, para después remover del periodograma los valores presentes entre ambos mínimos locales. Cumplido esto, se considera la extracción de peak como finalizada. Se pueden seguir extrayendo nuevos peaks por medio de trabajar con los residuales de un proceso de extracción anterior. . . . .	30
3.3.2. Ilustración del proceso realizado sobre las estrellas de nuestro set de entrenamiento. Primero calculamos curvas de velocidad radial, utilizando el software especializado Naira (Astudillo-Defru et al., 2017), a partir de las observaciones del espectrógrafo HARPS (Mayor et al., 2003), para luego calcular periodogramas de tipo Lomb-Scargle (VanderPlas et al., 2012) utilizando la implementación provista por Zechmeister and Kürster (2009). El último paso es representar la información del periodograma en una manera reconocible por un algoritmo de aprendizaje de máquinas, para ellos creamos una lista de características utilizando el proceso de <b>extracción de valores de peak</b> . . . . .	31
3.5.1. Comparación de la metodología aplicada sobre las estrellas de ambos grupos. En la izquierda podemos ver la curva de velocidad radial, periodograma y lista de características para la estrella del grupo NP GJ 298, en el lado derecho podemos ver los gráficos respectivos para la estrella del grupo PC LHS 1140. . . . .	34
4.1.1. Comparación de la exactitud entre los modelos SVM y RF . . . . .	36
4.1.2. Comparación de la precisión entre los modelos SVM y RF . . . . .	37
4.1.3. Comparación del Recall entre los modelos SVM y RF . . . . .	38
4.1.4. Comparación del puntaje F1 entre los modelos SVM y RF . . . . .	39
5.1.1. Importancia de características para un modelo Random Forest entrenado con 100 peaks, 500 estimadores y utilizando el criterio de impureza Gini. . . . .	43
5.1.2. Potencia normalizada para cada peak en orden descendente para todos los periodogramas de nuestra muestra. Cada línea corresponde a una lista de características para una estrella del grupo PC (izquierda) o una estrella del grupo NP (derecha). La potencia de cada peak es normalizada al valor de potencia máximo dentro de su respectivo periodograma, de esta manera $N_{peak} = 1$ tiene un valor de uno para cada estrella. . . . .	44
5.2.1. Histograma (Izquierda) e Histograma Acumulado (Derecha) de las observaciones para cada estrella en los grupos PC (arriba) y NP (Abajo) en nuestra base de datos. . . . .	45

# Capítulo 1

## Introducción

### 1.1. Nuestro Universo

En nuestra galaxia existen más estrellas que granos de arena en una playa, algunas de estas similares a nuestro Sol, pero nuestro sistema solar presenta una cualidad única, es el único sistema en el que sabemos con certeza que se ha originado vida.

Nuestra visión del universo siempre ha estado ligada a nuestra capacidad tecnológica. De esta manera, un universo sostenido por una tortuga gigante consistía en una explicación satisfactoria cuando éramos solamente grupos aislados de cazadores y recolectores. Más adelante, las primeras civilizaciones se vieron en la necesidad de medir el tiempo y el paso de las estaciones, es de aquí donde surge nuestra ciencia. El estudio del cielo se vuelve una necesidad para estos pueblos, puesto que al observar los movimientos y fases de la luna y las constelaciones, somos capaces de determinar el paso del tiempo y con ello saber cuando son las épocas de siembra, de cosecha, de lluvias y tormentas. Es la antigua Grecia la que formaliza el estudio del universo, creando variadas cosmovisiones dependiendo de la escuela filosófica. De esta manera tenemos un cosmos que se movía cuál mecanismo de relojería, esferas concéntricas moviendo los astros en órbitas epicíclicas. Los griegos jerarquizan al cielo, el Sol y la Luna poseen su propia esfera, orbitando a la Tierra. Las estrellas conforman la esfera exterior, un fondo de cuerpos inmóviles que envuelve a todo el universo. Entre medio encontramos las capas de unas estrellas especiales. Errantes, vagabundos, *planetes* fue el nombre otorgado a aquellos que poseían la capacidad de moverse de manera independiente de las

estrellas del fondo, siguiendo orbitas que a primera vista daban la impresión de ser erráticas, pero que eran podían ser explicadas por medio de modelos matemáticos ideados por los primeros astrónomos.

Hoy en día, tenemos el privilegio de vivir en una era de desarrollo científico y tecnológico incomparable. La ciencia ha expandido nuestros horizontes, acabado con enfermedades, dividido el átomo y nos ha llevado a pisar el espacio. En la actualidad, discusiones sobre la terraformación de otros mundos o la exploración de otros sistemas solares han salido del terreno de la ciencia ficción para transformarse en metas a conseguir antes del fin de siglo. En conjunto a estos logros, nuestra cosmovisión ha evolucionado también. Hoy en día somos conscientes de lo vasto que es nuestro universo. Hasta hace 30 años no teníamos la certeza de que existieran otros planetas aparte de los del sistema solar. El descubrimiento de Draugr y Poltergeist (PSR B1257+12 b y c?) los primeros exoplanetas observados alrededor de un pulsar, seguidos del descubrimiento de Pegasi 51 b ?, el primer exoplaneta descubierto alrededor de una estrella de secuencia principal cambio nuestra perspectiva, mostrándonos que existían mundos más allá del nuestro.

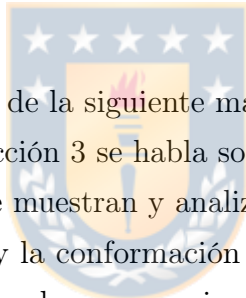
Hoy en día la búsqueda de exoplanetas está entrando en nueva era. Observaciones a gran escala, como TESS (Ricker et al., 2014) y anteriormente Kepler (Borucki et al., 2010) nos han provisto de una gran cantidad de información sobre como se distribuyen las poblaciones de exoplanetas en nuestra galaxia. El despliegue de nuevos instrumentos como el E-ELT y el telescopio James Webb nos permitirán la observación directa de mundos extrasolares y con ello poder estimar detalles atmosféricos, su posible habitabilidad e incluso la presencia de vida en estos mundos.

Una de las ventajas con las que contamos hoy en día, es la existencia de un vasto catálogo de observaciones estelares, tanto en velocidad radial como en brillo. Esto nos abre la posibilidad de aplicar técnicas estadísticas modernas para beneficio de observaciones futuras. Al momento de estudiar un nuevo sistema estelar, podríamos comparar las observaciones de la estrella con las de otros astros en los cuales es sabida la existencia de un exoplaneta. Si estas observaciones resultan similares, podríamos estar frente a un posible candidato planetario. Algo así, usualmente no vale la pena, el esfuerzo que requiere realizar esta comparación por un ser humano, podría ser mejor aplicado en estudiar derechamente las observaciones y decidir si es que estas contienen o no una señal planetaria. Pero, la ventaja de este método

consta en que no es un ser humano el que trabaja realizando este análisis, es una computadora. Mejor aún, no es solo una estrella la que se está analizando, una máquina podría trabajar con un catálogo completo, estudiando cientos de miles de estrellas, filtrando solo los candidatos más probables, realizando en horas una labor que podría tardar semanas. De esta manera, el astrónomo puede dedicar su tiempo a estudiar solo los objetos que sean dignos de interés, sin perder tiempo en aquellas estrellas que no tengan señal alguna.

Estas son las intenciones detrás de la aplicación del machine learning para este trabajo, la motivación de esta tesis es el desarrollo de métodos computacionales capaces de identificar de manera confiable candidatos planetarios dentro una base de datos. Específicamente, buscamos trabajar con datos de velocidades radiales de estrellas de tipo M, para ello contamos con un grupo de estrellas que hemos seleccionado de antemano, nuestra selección basada en la existencia de exoplanetas en las estrellas escogidas.

Este trabajo se estructura de la siguiente manera, en la sección 2 se introducen las bases teóricas, en la sección 3 se habla sobre la metodología utilizada en este proyecto, en la sección 4 se muestran y analizan los resultados, en la sección 5 se discuten estos resultados y la conformación de la base de datos y se comparan con proyectos similares, para luego resumir y concluir en la sección 6.



# Capítulo 2

## Marco Teórico

### 2.1. Sistemas Exoplanetarios

La mayoría de la información de esta sección fue obtenida del libro "The Exoplanet Handbook" de Michael Perryman (2011).

Planeta es como clasificamos a un grupo de objetos celestes en nuestro sistema solar, según la decisión de la Unión Astronómica Internacional, planeta es aquel cuerpo celeste que cumple con los siguientes requisitos:

1. Debe estar en órbita alrededor del Sol.
2. Posee suficiente masa para que su forma sea esférica.
3. Su gravedad ha limpiado la órbita de otros cuerpos de tamaño similar.

El motivo detrás de estos criterios es claro, el establecer una diferencia entre los planetas y cuerpos menores, como planetas enanos, lunas o asteroides. De esta forma, un objeto como Plutón, a pesar de tener un tamaño relativamente grande y una forma esférica, es descalificado debido a la presencia de otros cuerpos en su órbita y a su interacción con su luna Caronte.

Un exoplaneta entonces es aquel objeto que obedece estos mismos requisitos, salvo que en lugar de orbitar al Sol, orbita una estrella distinta. Se hace necesario también añadir un requisito adicional no mencionado para los planetas del sistema solar, y este es el límite de la fusión de hidrógeno. Si el llamado exoplaneta es capaz de fusionar hidrogeno en su núcleo, se le considera como estrella, por lo que

estaríamos ante un sistema binario. Por el contrario, si el objeto no es capaz de realizar la fusión, en la mayoría de casos por tener una masa menor a  $13 M_J$ , se le considera como gigante gaseoso.

## 2.2. Técnicas de Detección

Detectar un exoplaneta es un gran desafío, puesto que se está tratando de detectar la presencia de un objeto frío, opaco y relativamente poco masivo mientras este orbita a un objeto caliente, luminoso y masivo como lo es una estrella. Por esta razón, la mayoría de métodos de detección se enfocan en una detección indirecta del exoplaneta, en lugar de observarlo directamente, observan las perturbaciones que este causa sobre la estrella. Los métodos de detección más utilizados, como lo son el método de tránsito y el método de velocidades radiales, son de esta naturaleza.

### 2.2.1. Órbitas

La tercera ley de Newton establece que cuando dos cuerpos interactúan, ejercen fuerzas uno sobre el otro iguales en magnitud pero con sentido opuesto. De la misma manera, en la formulación de su ley gravitatoria, establece que ambos cuerpos interactuando ejercen atracción de gravedad uno sobre el otro.

Debido a esto, se puede decir que un planeta no gira alrededor de su estrella, en realidad ambos cuerpos giran alrededor del centro de masa del sistema (baricentro) y la ubicación del centro de masa es más próxima al cuerpo más masivo.

Esto resulta en que conforme el planeta realiza su órbita, la estrella experimenta un movimiento reflejo, generando en una perturbación periódica de propiedades observables de la estrella, como lo son su velocidad radial o su posición astrométrica en el cielo.

#### 2.2.1.1. Anomalías Orbitales

Según las leyes de Kepler, la órbita de un exoplaneta y la de su estrella es elíptica, con el centro de masa del sistema en uno de los focos de la elipse. Esta elipse puede ser descrita en coordenadas polares (con respecto a uno de los focos) de la

siguiente forma:

$$r = \frac{a(1 - e^2)}{1 + e * \cos(v)} \quad (2.2.1)$$

Donde  $a$  es el semi-eje mayor,  $v$  es la anomalía verdadera y  $e$  es la excentricidad de la elipse. O en coordenadas cartesianas con respecto al centro de la elipse:

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1 \quad (2.2.2)$$

Donde el semi-eje mayor  $a$  y el semi-eje menor  $b$  se relacionan con la excentricidad  $e$  de la manera:

$$b^2 = a^2(1 - e^2) \quad (2.2.3)$$

Las distancias de pericentro  $q$  y apocentro  $Q$  son descritas de la forma:

$$q = a(1 - e) \quad (2.2.4)$$

$$Q = a(1 + e) \quad (2.2.5)$$

Varios ángulos en el plano de la órbita son referidos como anomalías y son usados para describir la posición del planeta a lo largo de su órbita para un tiempo específico. La anomalía verdadera  $v(t)$  (a veces referida como  $f(t)$ ) es el ángulo entre la dirección del pericentro y la posición actual del planeta, medidas desde el foco ubicado en el baricentro del sistema.

La anomalía excéntrica  $E(t)$  es el ángulo correspondiente, pero referente al círculo auxiliar de la elipse. Tanto la anomalía verdadera como la excéntrica se relacionan geoméricamente de la forma:

$$\cos(v(t)) = \frac{\cos(E(t)) - e}{1 - e * \cos(E(t))} \quad (2.2.6)$$

Equivalentemente:

$$\tan \frac{v(t)}{2} = \left( \frac{1 + e}{1 - e} \right)^{\frac{1}{2}} * \tan \frac{E(t)}{2} \quad (2.2.7)$$

La anomalía media ( $M(t)$ ) corresponde a la fracción de desplazamiento transcurrido en el movimiento orbital desde que el cuerpo orbitando paso el pericentro. Un cuerpo orbitando no se mueve a una velocidad fija, pero se puede



especificar su órbita en términos del movimiento medio:

$$n = \frac{2\pi}{P} \quad (2.2.8)$$

Donde  $P$  es el periodo orbital. La anomalía media para el tiempo  $t - t_p$  después del paso del pericentro se define como:

$$M(t) = \frac{2\pi}{P}(t - t_p) = n(t - t_p) \quad (2.2.9)$$

Podemos relacionar la anomalía media con la anomalía excéntrica de la forma:

$$M(t) = E(t) - e \sin(E(t)) \quad (2.2.10)$$

La posición de un objeto a lo largo de su órbita para un tiempo determinado puede ser obtenida por medio de calcular la anomalía media para ese tiempo utilizando la ecuación 2.2.9, obteniendo el correspondiente valor para la anomalía excéntrica  $E(t)$  usando 2.2.10 y luego usando la identidad geométrica 2.2.6 para obtener  $v(t)$ .

### 2.2.1.2. Elementos orbitales

Una órbita Kepleriana en tres dimensiones (ver figura 2.2.1) puede ser descrita utilizando 7 parámetros:  $a, e, P, t_p, i, \Omega, \omega$ . Los primeros dos  $a, e$  corresponden al largo del semieje mayor y a la excentricidad orbital, por tanto, describen el tamaño y forma de la órbita. El periodo  $P$  está relacionado con  $a$  y las masas de los cuerpos involucrados, por medio de la tercera ley de Kepler, mientras que  $t_p$  corresponde a la posición del objeto a lo largo de la órbita para un tiempo particular de referencia (comúnmente se usa como referencia el paso por el pericentro).

Los tres ángulos  $i, \Omega, \omega$  corresponden a la proyección de la órbita real a la órbita aparente, son exclusivamente dependientes de la orientación del observador con respecto al movimiento orbital.

$i$  corresponde a la inclinación del plano de la órbita con respecto al plano de observación. Obtiene valores entre  $0^\circ$  y  $180^\circ$ , con  $i=0^\circ$  correspondiendo a una órbita perpendicular al observador.

$\Omega$  corresponde a la longitud del nodo ascendente, medido de acuerdo al plano de referencia. El nodo ascendente es aquel nodo (puntos donde la órbita y el plano



al cubo del semieje mayor de la elipse.

La primera y la tercera ley de Kepler son productos de la Ley de Gravedad, mientras que la segunda ley de Kepler es resultado de la conservación del momento angular (y se cumple para cualquier fuerza de atracción radial). Las leyes de Kepler fueron formuladas originalmente con respecto al Sol, pero existen formulaciones universales para cada tipo de órbita respecto al baricentro. Para un problema de dos cuerpos, donde la masa del segundo cuerpo **no** es despreciable, ambas órbitas son elipses compartiendo al baricentro como foco. La tercera ley de Kepler toma la siguiente forma en esta situación:

$$P^2 = \frac{4\pi^2}{GM} a^3 \quad (2.2.11)$$

Donde  $M$  y  $a$  toman valores distintos dependiendo al tipo de órbita que se esté midiendo.

**Orbita relativa:** El movimiento del planeta, con respecto a la estrella, en lugar de al baricentro. Toma la forma:

$$P^2 = \frac{4\pi^2}{G(M_* + M_p)} a_{rel}^3 \quad (2.2.12)$$

La coordenada de origen ahora es la estrella, no el baricentro y  $a_{rel}$  corresponde a la longitud del semieje de la órbita relativa, la del planeta alrededor de la estrella. Para un  $M_p \ll M_*$  y utilizando como unidad de distancia la UA (distancia promedio entre la Tierra y el Sol) se cumple que:

$$P = 1 \left( \frac{a_{rel}}{UA} \right)^{\frac{3}{2}} \left( \frac{M_*}{M_\odot} \right)^{-\frac{1}{2}} [years] \quad (2.2.13)$$

**Orbita Absoluta:** La órbita de la estrella alrededor del baricentro está dada por:

$$P^2 = \frac{4\pi^2}{GM'} a_*^3 \quad (2.2.14)$$

Donde  $M'$  corresponde a:

$$M' = \frac{M_p^3}{(M_p + M_*)^2} \quad (2.2.15)$$

Y  $a_*$  corresponde al semieje mayor de la órbita con respecto al baricentro. Una expresión equivalente a 2.2.14 nos daría la distancia para el semieje de  $a_p$ , la órbita

del planeta con respecto al baricentro.

Los tamaños de las tres órbitas están en proporción  $a_* : a_p : a_{rel} = M_p : M_* : (M_p + M_*)$  con  $a_{rel} = a_* + a_p$ . Más aún,  $e_p = e_{rel} = e_*$  y  $P_{rel} = P_* = P_p$ , las tres órbitas son coplanares y la orientación de las dos órbitas baricéntricas difiere por  $180^\circ$ . Debido a que el planeta por lo general no es visible, el movimiento orbital de la estrella es la única manera en que podemos determinar los parámetros de la órbita. Cabe aclarar que solo la observación por medio de astrometría es capaz de determinar todos los parámetros orbitales, las observaciones por métodos indirectos, como Velocidades radiales o tránsito, no son capaces de resolver todos los parámetros orbitales.

### 2.2.2. Método de Velocidades Radiales

El método de velocidades radiales describe el movimiento aparente en la línea de visión de la estrella mientras esta se mueve alrededor del baricentro. La posición de la estrella en la coordenada  $z$  en la línea de visión puede ser derivada trigonométricamente:

$$z = r(t) \text{sen}(i) \text{sen}(\omega + v) \quad (2.2.16)$$

Donde  $r(t)$  es la distancia al baricentro,  $\omega$  es el argumento del pericentro,  $i$  la inclinación de la órbita y  $v$  la anomalía real.

Podemos calcular la velocidad por medio de derivar la expresión 2.2.16 a través del tiempo:

$$v_r = \dot{z} = \text{sen}(i) [\dot{r} \text{sen}(\omega + v) + r \dot{v} \cos(\omega + v)] \quad (2.2.17)$$

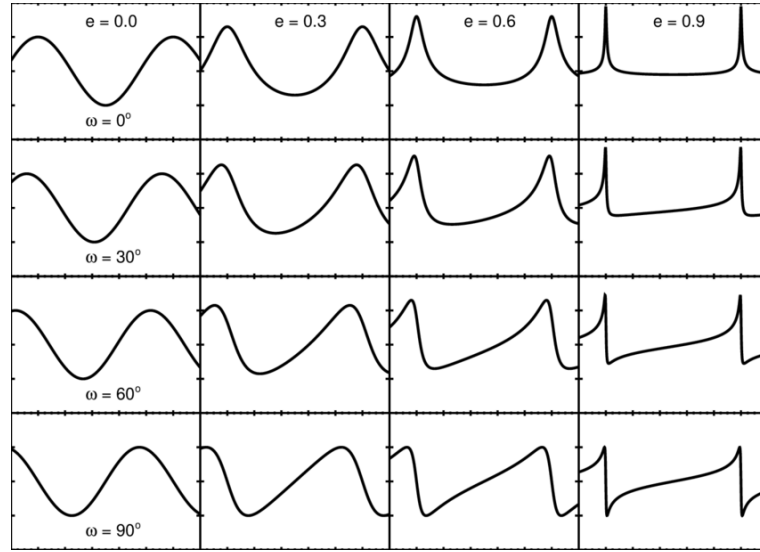
Introduciendo sustituciones para  $r$  y  $\dot{r}$  obtenemos:

$$v_r = K [\cos(\omega + v) + e * \cos(\omega)] \quad (2.2.18)$$

Donde la **Semi-Amplitud de Velocidad Radial** está dada por:

$$K = \frac{2\pi a_* \text{sen}(i)}{P (1 - e^2)^{\frac{1}{2}}} \quad (2.2.19)$$

Desde la ecuación 2.2.18 podemos ver que la velocidad radial  $v_r$  fluctúa entre los límites  $K(1 + e * \cos(\omega))$  y  $K(-1 + e * \cos(\omega))$ . La forma de la curva de velocidad radial está determinada por los valores de  $e$  y  $\omega$  (Ver figura 2.2.2). Cabe



**Figura 2.2.2:** Figura mostrando la forma de diversas curvas de velocidad radial que comparte un mismo valor de  $K, P$  y  $t_0$ . Cada columna comparte un mismo valor para la excentricidad, mientras que cada fila comparte el mismo valor de  $\omega$ . Podemos notar como los valores de  $\omega$  y  $e$  condicionan la forma de la curva de velocidad radial. Créditos: [Wright and Gaudi \(2012\)](#)

aclarar que el sistema estelar puede poseer un movimiento propio, lo que suma un componente  $\gamma$  adicional a la velocidad radial:

$$v_r = \gamma + K [\cos(\omega + v) + e * \cos(\omega)] \quad (2.2.20)$$

Podemos encontrar una expresión alternativa para  $K$  por medio de substituir las ecuaciones 2.2.14 y 2.2.15 en 2.2.19:

$$K^2 = \frac{G}{(1 - e^2)} \frac{1}{a_* \sin(i)} \frac{M_p^3 \sin^3(i)}{(M_* + M_p)^2} \quad (2.2.21)$$

Podemos combinar las ecuaciones 2.2.14, 2.2.15 y 2.2.19 para obtener un valor de  $K$  sin la apariencia explícita de  $a_*$ :

$$K = \left( \frac{2\pi G}{P} \right)^{\frac{1}{3}} \frac{M_p \sin(i)}{(M_* + M_p)^{\frac{2}{3}}} \frac{1}{(1 - e^2)^{\frac{1}{2}}} \quad (2.2.22)$$

El segundo factor de esta ecuación se simplifica cuando  $M_p \ll M_*$  lo que nos otorga:

$$K = \left( \frac{2\pi G}{P} \right)^{\frac{1}{3}} \frac{M_p \sin(i)}{(M_*)^{\frac{2}{3}}} \frac{1}{(1 - e^2)^{\frac{1}{2}}} \quad (2.2.23)$$

Lo que significa que el valor de la semiamplitud  $K$  es directamente dependiente del segundo factor de la ecuación. Debido a que la masa estelar puede ser estimada por otros medios (como el uso del tipo espectral, astrometría o similares), podemos utilizar las mediciones de velocidad radial para estimar el valor de  $M_p \sin(i)$ . Es necesario aclarar que utilizando el método de velocidades radiales por si solo es imposible estimar individualmente la masa estelar o la inclinación orbital, solo la combinación de ambas. Un valor de  $K$  pequeño, puede significar tanto un planeta de baja masa como la presencia de un objeto de gran masa, pero con una inclinación orbital pequeña, con el plano de la órbita de cara a la línea de observación.

## 2.3. Análisis de Series Temporales

Muchas veces la identificación de un exoplaneta se logra por métodos indirectos, buscando su rastro en el comportamiento de la estrella. Este rastro se suele manifestar como perturbaciones periódicas en las observaciones de flujo lumínico o velocidad radial. Por tanto, a menudo es útil representar esta información por medio de una serie temporal.

Una serie temporal es una relación que establece como se comporta una variable (en este caso concreto la velocidad) a lo largo del tiempo. Podemos definir cada medición de velocidad como  $V_N$  y su tiempo correspondiente como  $t_N$ , ambas se relacionan de la forma:

$$g(t_N) = v_N \quad (2.3.1)$$

Nos interesa en especial el caso de que esta serie temporal sea periódica, que al cabo de un periodo  $T$  el valor de velocidad sea el mismo. De esta forma, podemos modelar el comportamiento esperado en el caso de existir un exoplaneta orbitando la estrella.

$$g(t_N + T) = g(t_N) = v_N \quad (2.3.2)$$

### 2.3.0.1. Transformada de Fourier

Una herramienta muy poderosa para analizar series temporales es el uso de la **Transformada de Fourier**, una herramienta matemática capaz de descomponer una serie temporal, entregando una función dependiente de las frecuencias presentes en la serie original. La nueva función toma la forma:

$$\hat{g}(\omega) = \int_{-\infty}^{\infty} g(t)e^{-i\omega t} dt \quad (2.3.3)$$

Donde  $g(t)$  es la función temporal original,  $\hat{g}(\omega)$  su transformada,  $\omega = 2\pi f$  y  $f$  corresponde a la frecuencia. También podemos describir la relación inversa:

$$g(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \hat{g}(\omega)e^{i\omega t} d\omega \quad (2.3.4)$$

Resulta conveniente definir el operador matemático de la transformada  $\mathcal{F}$ :

$$\mathcal{F}\{g(t)\} = \hat{g}(\omega) \quad (2.3.5)$$

$$\mathcal{F}^{-1}\{\hat{g}(\omega)\} = g(t) \quad (2.3.6)$$

La transformada también posee varias propiedades útiles, una de ellas es que es una operación lineal:

$$\mathcal{F}\{g(t) + f(t)\} = \mathcal{F}\{g(t)\} + \mathcal{F}\{f(t)\} \quad (2.3.7)$$

$$\mathcal{F}\{Ag(t)\} = A\mathcal{F}\{g(t)\} \quad (2.3.8)$$

Otra propiedad útil es que la transformada de Fourier de una función sinusoidal de frecuencia  $f_0$  es una suma de funciones de delta evaluadas alrededor de  $\pm f_0$ :

$$\mathcal{F}\{e^{2\pi f_0 t}\} = \delta(f - f_0) \quad (2.3.9)$$

$$\mathcal{F}\{\cos(2\pi f_0 t)\} = \frac{1}{2}[\delta(f + f_0) + \delta(f - f_0)] \quad (2.3.10)$$

$$\mathcal{F}\{\sin(2\pi f_0 t)\} = \frac{1}{2i}[\delta(f + f_0) - \delta(f - f_0)] \quad (2.3.11)$$

Por último, un desplazamiento temporal en la serie de tiempo causa un cambio

de fase en la transformada de la forma:

$$\mathcal{F}\{g(t - t_0)\} = \mathcal{F}\{g(t)\} e^{-2\pi i f_0 t} \quad (2.3.12)$$

Al calcular la amplitud al cuadrado de la transformada, podemos eliminar los componentes complejos y la fase, obteniendo una función denominada **Espectro de Densidad de Potencias** o más comúnmente **Espectro de Potencias**.

$$\mathcal{P}_f \equiv |\mathcal{F}\{g(t)\}|^2 \quad (2.3.13)$$

Todas estas propiedades son la razón detrás de la utilidad de la transformada, puesto que debido a la linealidad del operador, toda señal compuesta por sinusoidales tendrá una transformación conformada por una suma de funciones de delta, cada una marcando la frecuencia de cada senoide, por tanto, la transformada de Fourier mide de manera directa los componentes periódicos en una función continua.

### 2.3.0.2. Transformada Discreta de Fourier

La transformada de Fourier y el espectro de potencias son herramientas esenciales para el análisis de series temporales, pero existen limitaciones para su aplicación. Hasta ahora hemos trabajado con el caso de funciones bien definidas y continuas, pero las mediciones por lo general son en intervalos de tiempo finito, realizadas con ratios de muestreo finitos. Por tanto, se necesita una manera de adaptar el análisis de Fourier para los casos reales.

Si suponemos que existe una función real y continua  $g(t)$  pero la observamos de manera regular entre intervalos de tiempo de duración  $\Delta t$ , por lo que nuestras observaciones son de la forma:

$$g_{obs} = g(t) III_{\Delta t}(t) \quad (2.3.14)$$

Donde  $III_{\Delta t}$  es un **peine de Dirac** una sumatoria de deltas de Dirac, útil para representar observaciones. Mientras se está observando, el valor de la función es igual a 1, en todo otro momento es igual a 0. Se expresa matemáticamente como:  $III_T = \sum_{k=-\infty}^{\infty} \delta(t - kT)$ .



La transformada de Fourier de nuestras observaciones toma la forma:

$$\hat{g}_{obs}(f) = \sum_{n=-\infty}^{\infty} g(n\Delta t)e^{-2\pi ifn\Delta t} \quad (2.3.15)$$

Pero, por lo general, el número de observaciones realizadas no es infinito, en realidad contamos con un límite de  $N$  observaciones. Si hacemos la sustitución  $g_n = g(n\Delta t)$  podemos escribir:

$$\hat{g}_{obs}(f) = \sum_{n=0}^N g_n e^{-2\pi ifn\Delta t} \quad (2.3.16)$$

Según el teorema elaborado por Nyquist <sup>1</sup>, las únicas frecuencias relevantes son aquellas que se encuentran en el intervalo  $0 \leq f \leq \frac{1}{2\Delta t}$  por lo que podemos definir  $N$  frecuencias con un espacio de separación  $\Delta f = \frac{1}{N\Delta t}$ . Definimos  $\hat{g}_k = \hat{g}_{obs}(k\Delta f)$ :

$$\hat{g}_k = \sum_{n=0}^N g_n e^{-\frac{2\pi i k n}{N}} \quad (2.3.17)$$

La cual es la formulación estándar de la transformada discreta de Fourier.

### 2.3.0.3. Periodograma de Lomb-Scargle

Si aplicamos la definición del **Espectro de Potencias** ?? sobre la transformada discreta, podemos obtener la formulación del periodograma clásico:

$$P_S(f) = \frac{1}{N} \left| \sum_{n=0}^N g_n e^{-2\pi ift_n} \right|^2 \quad (2.3.18)$$

Podemos reescribir la ecuación anterior descomponiendo el exponencial  $e$  en una suma de cosenos y senos:

$$P_S(f) = \frac{1}{N} \left| \left( \sum_{n=0}^N g_n \cos(2\pi f t_n) \right)^2 + \left( \sum_{n=0}^N g_n \sen(2\pi f t_n) \right)^2 \right|$$

(2.3.19)

<sup>1</sup>También llamado "Teorema de Nyquist-Shannon." "Límite de Muestreo de Nyquist", este teorema establece que para poder representar de manera certera cualquier señal existente en los datos, la frecuencia de muestreo debe ser el doble de la frecuencia que queremos estudiar.

Esta forma puede ser utilizada para identificar señales periódicas dentro de una serie temporal, pero tiene carencias. Cuando el muestreo es uniforme, el ruido, en forma de una función gaussiana, presenta una distribución  $\chi^2$  dentro del periodograma. Pero si el muestreo no es uniforme, estas propiedades ya no se mantienen, y el periodograma en general no puede ser expresado de manera analítica. Scargle (Scargle, 1982) propone una solución a este problema por medio de una versión generalizada del periodograma:

$$P(f) = \frac{A^2}{2} \left( \sum_{n=0} g_n \cos(2\pi f(t_n - \tau)) \right)^2 + \frac{B^2}{2} \left( \sum_{n=0} g_n \sen(2\pi f(t_n - \tau)) \right)^2 \quad (2.3.20)$$

Donde  $A, B$  y  $\tau$  son funciones arbitrarias dependientes de la frecuencia  $f$  y el tiempo de observación  $t_i$ . Scargle demostró que se pueden elegir funciones  $A, B$  y  $\tau$  de forma que el periodograma se reduce a la forma clásica en el caso de un muestreo uniforme, el periodograma es computable de forma analítica y que el periodograma es insensible a cambios globales en el tiempo.<sup>2</sup> Los valores de  $A$  y  $B$  que permiten estas propiedades nos entregan la formulación estándar del periodograma de Lomb-Scargle:

$$P_{LS}(f) = \frac{1}{2} \left\{ \frac{(\sum_n g_n \cos(2\pi f[t_n - \tau]))^2}{\sum_n \cos^2(2\pi f[t_n - \tau])} + \frac{(\sum_n g_n \sen(2\pi f[t_n - \tau]))^2}{\sum_n \sen^2(2\pi f[t_n - \tau])} \right\} \quad (2.3.21)$$

Donde  $\tau$  está definido de la forma:

$$\tau = \frac{1}{4\pi f} \tan^{-1} \left( \frac{\sum_n \sen(4\pi f t_n)}{\sum_n \cos(4\pi f t_n)} \right) \quad (2.3.22)$$

Una de las características del periodograma de Lomb-Scargle es que es idéntico al resultado de ajustar un modelo consistente de una senoide de frecuencia  $f$  a los datos y construir un "periodograma.<sup>a</sup> a partir del ajuste  $\chi^2$  resultante para cada frecuencia.

<sup>2</sup>Esto significa que el periodograma no es dependiente de en que momento comienzan y/o terminan las observaciones, solo de los valores observados y del tiempo transcurrido entre observaciones.

## 2.4. Modelos de Clasificación

### 2.4.1. Clasificadores Lineales

La información utilizada en esta sección ha sido recopilada principalmente del libro "An introduction to Machine Learning" de Miroslav Kubat (2017).

La información utilizada en esta sección ha sido recopilada principalmente del libro "An introduction to Machine Learning" de Miroslav Kubat (2017).

A menudo, durante el entrenamiento de un modelo de machine learning, representamos los objetos del entrenamiento como puntos en un espacio  $N$ -dimensional. Al hacer esto, podemos notar que objetos de una misma clase tienden a estar agrupados en regiones específicas de este espacio. Esta observación, motiva este tipo de clasificación, la búsqueda de una *superficie de decisión* que separa ambas clases. Esta superficie puede estar representada por una función, que arroje un resultado distinto al interactuar con elemento de un grupo u de otro.

En el caso de estar trabajando con un vector de características  $X_i = x_1, x_2, x_3 \dots x_n$ , este vector puede pertenecer tanto al grupo que nos interesa identificar, al cual denominaremos como grupo 0, como representar a un elemento del grupo no objetivo, denominado de aquí en adelante como grupo 1. Podemos definir una ecuación de la forma:

$$w_0 + w_1x_1 + \dots + w_nx_n = 0 \quad (2.4.1)$$

Si sustituimos los valores de  $X_i$ , esta ecuación debería dar un valor positivo para objetos de la clase 0, mientras que para objetos de la clase 1, el valor será negativo.

Si introducimos un valor  $x_0 = 1$  para todos los elementos en nuestro conjunto de entrenamiento, podemos escribir esta ecuación en una manera más compacta:

$$\sum_{i=0}^n w_i x_i = 0 \quad (2.4.2)$$

Notar que con  $n = 2$  esta ecuación describe una línea, con  $n = 3$  se describe un plano y con  $n > 3$  un hiperplano. Por este motivo, este tipo de clasificadores son denominados como clasificadores lineales.

El comportamiento y desempeño de este clasificador es totalmente dependiente

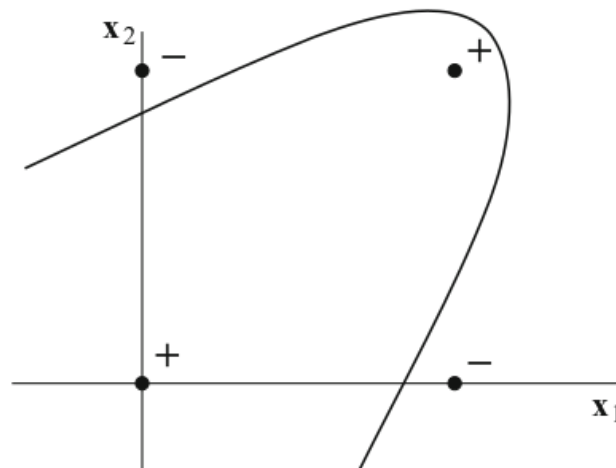
de los coeficientes:  $w = w_0, w_1, w_2 \dots w_n$ . Estos usualmente son denominados como pesos y el trabajo real del entrenamiento de machine learning es su correcta identificación. No todos los pesos representan lo mismo. Geométricamente, los coeficientes  $w_1, \dots, w_n$  representan el ángulo del hiperplano con respecto al sistema de coordenadas, mientras que el coeficiente  $w_0$  es llamado *bias* y representa la distancia del hiperplano con respecto al origen del sistema de coordenadas.

Podemos reescribir la ecuación anterior como:

$$w_1 + x_1 + \dots w_n x_n = \theta \quad (2.4.3)$$

Donde  $\theta = -w_0$  representa el **umbral**, el valor mínimo que la suman con pesos debe dar para que un elemento  $X_i$  sea considerado como parte de la clase objetivo.

Este tipo de modelos, debido a su simplicidad, presenta un impedimento, si los grupos no son linealmente separables, el modelo es incapaz de encontrar un valor adecuado para los coeficientes  $w_i$ . Esta es una situación se da de manera común, no solo el ruido puede arruinar la separación entre ambos grupos, la forma de las regiones que ocupan en el espacio N-dimensional puede causar que sea imposible trazar un hiperplano separando ambos grupos (Ver figura 2.4.2).



**Figura 2.4.1:** Ejemplo bidimensional de un caso en el cual no es posible aplicar un clasificador lineal para separar ambas clases (+ y -). Como se puede apreciar, no existe línea recta capaz de separar a ambos grupos, por lo que una curva polinomial se adecua mejor a esta tarea. Figura Obtenida de "Introduction to Machine Learning" de Miroslav Kubat (2017)

Para este tipo de casos, se pueden utilizar modelos polinomiales, modelos que, en

lugar de ajustar una línea en el espacio de características, ajustan un polinomio cuyo grado  $r$  puede ser fijado de antemano. La forma generalizada para un clasificador polinomial de orden  $r$  para un grupo con  $n$  características es:

$$\sum_{\substack{m=0 \\ (l_1+l_2+\dots+l_n=m)}}^r w_{\lambda,m} x_1^{l_1} x_2^{l_2} \dots x_n^{l_n} = 0 \quad (2.4.4)$$

Para un caso bidimensional ( $n = 2$ ) un clasificador polinomial de segundo orden ( $r = 2$ ) tiene la forma:

$$w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1^2 + w_4 x_2^2 + w_5 x_1 x_2 + w_6 x_1^2 x_2^2 = 0 \quad (2.4.5)$$

Si introducimos una lista  $z_i = z_1, z_2, \dots, z_k$  en la cual cada elemento  $z$  corresponde a una combinación posible de los elementos  $X_i$ , podemos representar nuestro clasificador polinomial como un clasificador lineal:

$$w_0 z_0 + w_1 z_1 + \dots + w_k z_k = \sum_{i=0}^k w_i z_i = 0 \quad (2.4.6)$$

En el caso bidimensional ( $n = 2$ ) con polinomios de segundo orden ( $r = 2$ )

$$\sum_{i=0}^k w_i z_i = w_0 z_0 + w_1 z_1 + w_2 z_2 + w_3 z_3 + w_4 z_4 + w_5 z_5 + w_6 z_6 = 0 \quad (2.4.7)$$

Donde  $z = 1, x_1, x_2, x_1^2, x_2^2, x_1 x_2, x_1^2 x_2^2$

Importante notar que al introducir un polinomio de segundo orden, pasamos de trabajar con un problema bidimensional a trabajar con un problema de 6 dimensiones.

La idea entonces es que el modelo polinomial es equivalente a un modelo lineal entrenado en más dimensiones. Nótese que esto aumenta notablemente el costo computacional, puesto que el número de coeficientes que hay que calcular  $N_w$  crece de la forma:

$$N_w = \binom{n+r}{r} \quad (2.4.8)$$

Como ejemplo, si quisiéramos utilizar polinomios de tercer orden ( $r = 3$ ) para

un grupo donde existen 100 características distintas ( $n = 100$ ), necesitaríamos encontrar los valores de 176,851 coeficientes, en comparación con el caso lineal, donde solo tendríamos que calcular 100 coeficientes.

Se debe notar que los modelos polinomiales ofrecen mejores resultados durante el entrenamiento, a costa de ser más propensos al sobre-ajuste.

#### 2.4.1.1. Máquinas de vector de soporte

Como hemos visto anteriormente, un clasificador lineal o polinomial permite usar varios valores para los coeficientes  $w$ . Esto se entiende como que existen varios hiperplanos capaces de separar entre los grupos a clasificar. Pero, esto no significa que estos hiperplanos sean igual de útiles para realizar predicciones sobre elementos nuevos. Nuevos elementos pueden ocupar regiones del espacio distintas a las usadas durante el entrenamiento. Por tanto, surge la necesidad de decidir cuál es el mejor hiperplano para el caso general.

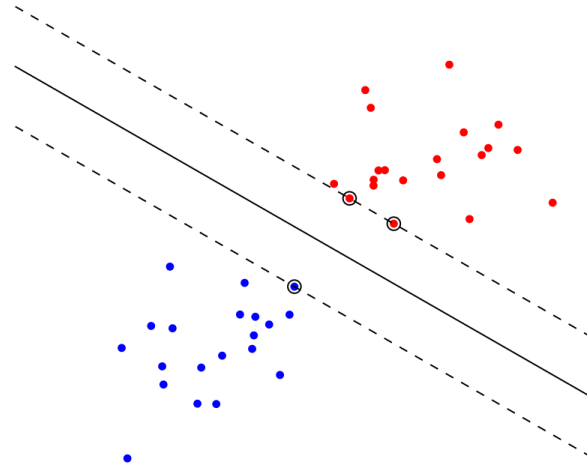
Las máquinas de vector de soporte (en inglés *Support Vector Machine* o comúnmente *SVM*) son modelos de machine learning que buscan generalizar lo aprendido por un modelo lineal o polinomial. Para ello utilizan el concepto de margen, la distancia existente entre el hiperplano y los elementos de ambos grupos. La razón detrás de esto radica en que el mejor hiperplano es aquel que maximiza las distancias entre ambos grupos, de esta manera se reduce el riesgo de clasificar erróneamente a futuros elementos.

Un modelo lineal cualquiera (recordemos que un modelo polinomial puede ser representado como un modelo lineal expandido a dimensiones superiores) tiene la forma:

$$\varphi(x_i) = \begin{cases} -1 : & w_0 + \sum_{i=1}^n x_i w_i > 0 \\ 1 : & w_0 + \sum_{i=1}^n x_i w_i < 0 \end{cases} \quad (2.4.9)$$

Es útil trabajar con el vector de pesos  $\vec{w} = (w_1, w_2, \dots, w_n)$  que corresponde al vector normal al hiperplano, y definir vectores de características  $\vec{x}_i = (x_1, x_2, \dots, x_n)$ . Cada vector  $\vec{x}_i$  tiene asignada de antemano una clase  $y_i$ . De esta manera el clasificador es de la forma:

$$\varphi(x_i) = \begin{cases} -1 : & \vec{w}^T \vec{x}_i - w_0 > 0 \\ 1 : & \vec{w}^T \vec{x}_i - w_0 < 0 \end{cases} \quad (2.4.10)$$



**Figura 2.4.2:** Ejemplo del uso de una máquina de vector de soporte (Modelo SVM). En esta técnica, se busca utilizar el hiperplano que este a la mayor distancia (margen) posible de todos los elementos del conjunto de entrenamiento. Figura Obtenida de "Understanding Random Forest: From Theory to Practice" de Gilles Louppe (2014)

Si los datos son linealmente separables, podemos elegir dos hiperplanos paralelos tal que

$$\vec{w}^T \vec{x} - w_0 = 1 \quad (2.4.11)$$

$$\vec{w}^T \vec{x} - w_0 = -1 \quad (2.4.12)$$

La distancia entre ambos hiperplanos es igual a  $\frac{2}{\|\vec{w}\|}$  por lo que para seleccionar al mejor hiperplano, debemos resolver el siguiente problema de optimización primario:

$$\min_{\xi, w} \left\{ \frac{1}{2} \|\vec{w}\|^2 + C \sum_{i=1}^N \xi_i \right\} \quad (2.4.13)$$

Donde C es un hiperparametro escogido para controlar que tan sensible es este margen en caso de que los objetos del grupo de entrenamiento no sean linealmente separables.  $\xi_i$  está sujeto a

$$y_i(\vec{w}^T \vec{x}_i) \geq 1 - \xi_i, \xi_i \geq 0 \quad (2.4.14)$$

La forma dual de este problema de optimización es:

$$\max_{\alpha} \left\{ \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i \cdot x_j \right\} \quad (2.4.15)$$

Sujeto a  $0 \leq \alpha_i \leq C$ .

El vector que soluciona este problema de optimización dual tiene la forma:

$$\vec{w} = \sum_{i=1}^N \alpha_i y_i x_i \quad (2.4.16)$$

Una máquina de vector de soporte puede extenderse para clasificar problemas no separables linealmente por medio de proyectarse en un espacio de mayores dimensiones utilizando una función kernel. En este caso el problema de optimización dual es el mismo, solo que reemplazando el producto punto  $x_i \cdot x_j$  por un kernel  $K(x_i, x_j)$

#### 2.4.1.2. Kernels

Una parte importante detrás del funcionamiento de una máquina de vector de soporte es el uso de una función Kernel. Como hemos visto antes, para resolver el problema de separación cuando los grupos no son linealmente separables, se pueden utilizar clasificadores polinomiales (Ver figura 2.4.2). Pero, estos presentan el inconveniente de aumentar en gran medida la carga computacional. Una solución es el uso de una función Kernel, una función encargada de mapear el posicionamiento de cada objeto en un espacio de mayores dimensiones. Como ejemplo podemos tener el caso de aplicar un polinomio de segundo grado a un vector de 3 dimensiones. Los vectores  $\vec{X}_i$  tienen la forma:

$$\vec{X}_i = (X_{i1}, X_{i2}, X_{i3}) \quad (2.4.17)$$

Mientras que el vector de pesos  $\vec{W}$  es de la forma:

$$\vec{W} = (w_1, w_2, w_3) \quad (2.4.18)$$

Si aplicamos un polinomio de segundo grado para resolver el problema de



clasificación, tenemos que evaluar:

$$\sum_{k=1}^Z w_k z_k = W_0 + w_1 X_{i1} + w_2 X_{i2} + w_3 X_{i3} + w_4 X_{i1} X_{i2} + w_5 X_{i1} X_{i3} + w_6 X_{i2} X_{i3} + w_7 X_{i1}^2 + w_8 X_{i2}^2 + w_9 X_{i3}^2 \quad (2.4.19)$$

Pero podemos utilizar una función Kernel definida como:

$$K(\vec{x}, \vec{y}) = (\langle \vec{x}, \vec{y} \rangle)^2 \quad (2.4.20)$$

Aplicándola sobre nuestros vectores  $\vec{W}, \vec{X}_i$ :

$$K(\vec{W}, \vec{X}_i) = (\langle \vec{W}, \vec{X}_i \rangle)^2 = (W_0 + w_1 X_{i1} + w_2 X_{i2} + w_3 X_{i3})^2 \quad (2.4.21)$$

Dado que podemos elegir los valores para los factores  $w_4, \dots, w_9$ , podemos definirlos de manera que el producto de la función Kernel es equivalente al valor obtenido de  $\sum_{k=1}^Z w_k z_k$ , con la ventaja de que al trabajar con la función Kernel solo debemos encontrar los factores  $w_1, w_2, w_3$ , puesto que los factores de mayores dimensiones nacen de la combinación de estos. De esta forma, la función Kernel actúa de la misma forma que trabajar en mayores dimensiones, sin la necesidad de proyectar directamente nuestros datos en estas dimensiones.

Durante este trabajo estudiamos la aplicación de 2 funciones de kernel.

**Kernel lineal**, en este caso se trabaja sin ninguna proyección en mayores dimensiones, por lo que se fuerza el uso de un modelo lineal para resolver el problema de clasificación. Matemáticamente, la forma de este Kernel es:

$$K(\vec{x}, \vec{y}) = (\langle \vec{x}, \vec{y} \rangle) \quad (2.4.22)$$

**Kernel Función de Base Radial**, este Kernel proyecta la similitud entre los dos vectores entregados, las cuales proyecta como una medida de distancia entre ambos. Matemáticamente, la forma de este kernel es:

$$K(\vec{x}, \vec{y}) = \exp\left(-\gamma \|\vec{x} - \vec{y}\|^2\right) \quad (2.4.23)$$

Donde  $\gamma$  es un hiperparametro utilizado para regular la influencia que un objeto

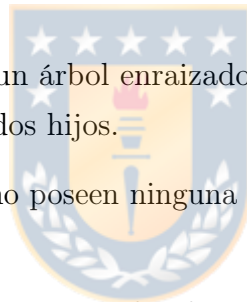
en particular pueda ejercer sobre el modelo.



### 2.4.2. Árboles de Decisión

Un árbol de decisión es un modelo predictivo que utiliza un gráfico en forma de árbol, en el cual se proyectan diversas condicionales, con el fin de organizar los datos. Un árbol de decisión se define de la siguiente manera:

- Un árbol es un gráfico en el cual todo par de vértices (llamados **Nodos**) son conectados por una única línea o **rama**.
- Un árbol enraizado (Rooted Tree), es un tipo de árbol en el cual uno de los nodos ha sido designado como el nodo raíz. Todas las ramas del árbol dan comienzo en este nodo raíz.
- Si existe una rama entre los nodos,  $t_1$  a  $t_2$  entonces se dice que el nodo  $t_1$  es el padre del nodo  $t_2$ , mientras que el nodo  $t_2$  es llamado el nodo hijo del nodo  $t_1$ .
- Un árbol binario es un árbol enraizado en el cual todos los nodos internos tienen exactamente dos hijos.
- Aquellos nodos que no poseen ninguna rama saliendo de ellos, son llamados los nodos hoja.

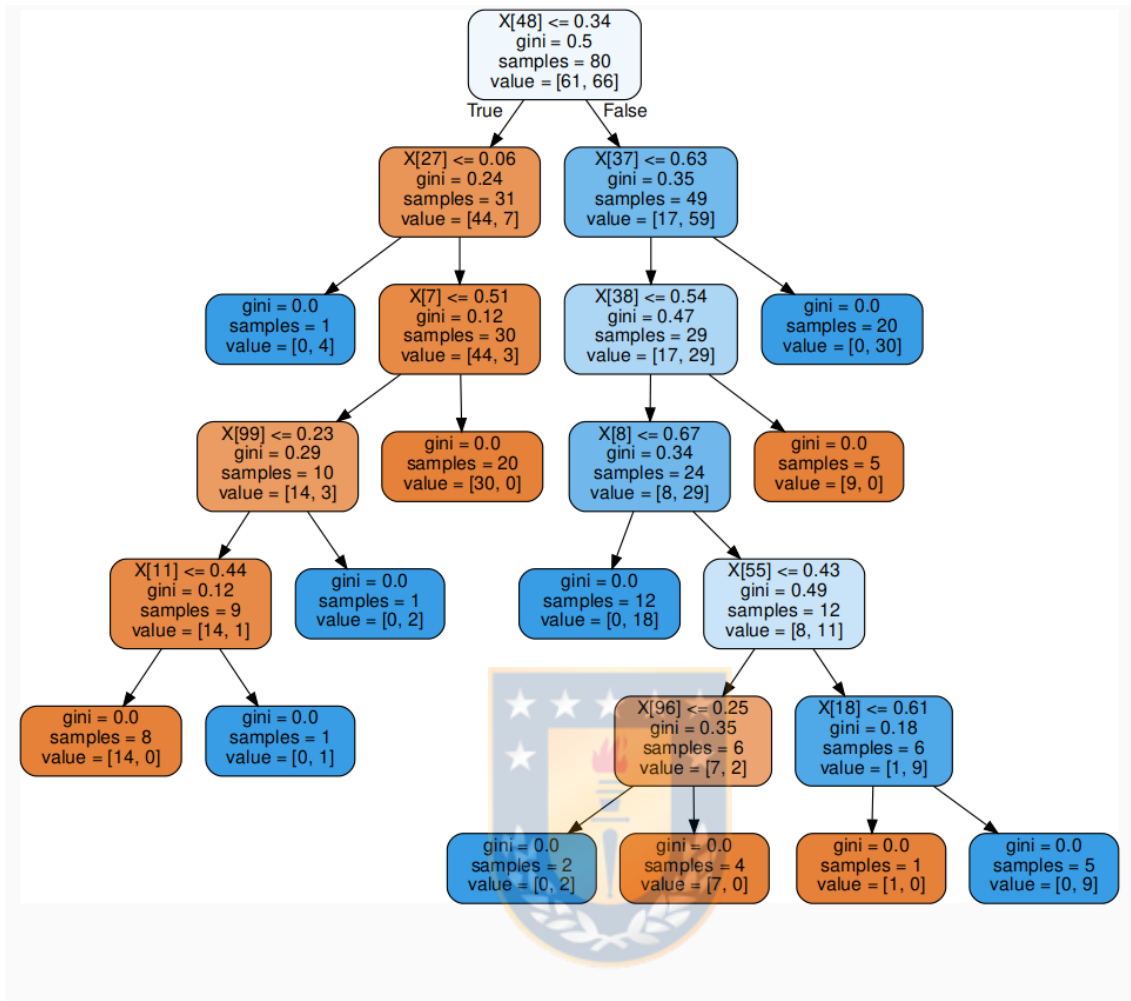


Un árbol de decisión actúa por medio de establecer condicionales, el nodo raíz elige una característica y aplica una condicional, los objetos que cumplen la condicional son enviados por una rama a uno de los nodos hijo, mientras que los que la incumplen son enviados al nodo opuesto. Durante este proceso se eligen características dando prioridad a formar nodos que sean los más distintos entre sí, mientras que al mismo tiempo cada nodo debe contener la mayor cantidad de objetos similares. Siguiendo este criterio, cada nodo padre se divide nuevamente, buscando un estado similar para sus nodos hijos, hasta que se consigue contener solo objetos de una misma clase en cada nodo hoja.

En la figura 2.4.3 se puede observar un árbol de decisión utilizado durante esta investigación.

### 2.4.3. Random Forest

Un Árbol de Decisión es una herramienta de alta complejidad utilizada para ordenar las observaciones y a partir de estas predecir observaciones futuras, pero



**Figura 2.4.3:** Árbol de decisiones generado como parte de este trabajo. El nodo raíz ha seleccionado la característica número 48 y ha elegido como condicional que el valor de esta característica supere un cierto umbral. Cada nodo subsiguiente selecciona nuevas características hasta que los nodos hoja contienen exclusivamente objetos de una misma clase. El color del nodo es indicativo de su contenido, naranja para la clase **PC** y azul para la clase **NP**.

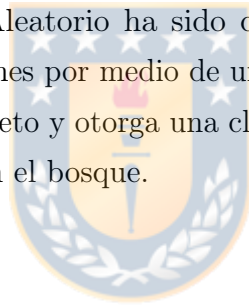
este tipo de modelo es extremadamente susceptible a la presencia de valores atípicos. Este tipo de clasificadores se denomina de alta varianza, puesto que pequeñas fluctuaciones en el grupo de entrenamiento puede cambiar enormemente la forma del clasificador. Si eligiéramos un set de datos distinto durante el entrenamiento, el árbol de decisiones generado sería distinto también.

Como corrección, nace el concepto de **Bosque Aleatorio**. En lugar de generar un solo árbol a partir del set de entrenamiento, generamos varios árboles siguiendo dos principios.

- **Re-Muestreo:** Un árbol de decisiones normalmente trabaja con todo el grupo de entrenamiento, pero en el caso de árboles pertenecientes a un bosque aleatorio, estos solo pueden trabajar con subgrupos elegidos al azar del grupo de entrenamiento original.
- **Características Aleatorias:** Un Árbol de Decisiones elige entre todas las características disponibles para generar la condicional en cada nodo. En un modelo de Bosque Aleatorio, cada árbol solo tiene disponible para elegir un número acotado de características escogidas al azar.

De esta forma, podemos asegurar que existe poca correlación entre cada árbol generado en el Bosque Aleatorio, asegurando que el conjunto de árboles trabaje en detectar los patrones de fondo presentes en las observaciones en lugar de prestar atención a fluctuaciones estocásticas en las mediciones.

Una vez que el Bosque Aleatorio ha sido conformado, se puede utilizar para predecir nuevas observaciones por medio de un voto mayoritario. Cada árbol en el bosque evalúa al nuevo objeto y otorga una clasificación. La clase final es decidida por la clase mayoritaria en el bosque.



# Capítulo 3

## Metodología

### 3.1. Datos utilizados

Para realizar este trabajo seleccionamos un grupo de estrellas de clase M observadas con el espectrógrafo HARPS (Mayor et al., 2003). El grupo principal, de aquí en adelante denominado PC (Planetary Candidates) consta de 39 estrellas de clase M alrededor de las cuales se han detectado 56 exoplanetas. Además de estas, se seleccionaron 155 estrellas de clase M, siguiendo dos criterios principales, que no existieran exoplanetas observados alrededor de estas estrellas y que contaran con al menos 10 observaciones realizadas por el espectrógrafo HARPS. Este es el denominado grupo NP (No Planet). En total, nuestro set de entrenamiento está compuesto de 194 estrellas distintas.

Desde el repositorio de observaciones de HARPS (Delmotte et al., 2006) descargamos todos los espectros disponibles de estas estrellas, para luego derivar estimaciones de velocidad radial. El software de reducción de datos de HARPS (HARPS DRS Baranne et al. (1979)) cuenta con funciones para el cálculo de velocidades radiales (Pepe et al., 2002) y es comúnmente utilizado para estas tareas, pero nosotros escogimos utilizar Naira (Astudillo-Defru et al., 2017) un programa especializado para el cálculo de velocidades radiales alrededor de estrellas de clase M. La ventaja de utilizar Naira es que permite realizar un cálculo de mayor precisión, por lo general obteniendo un ruido entre 20% a 30% menor comparado al estimado por HARPS DRS.

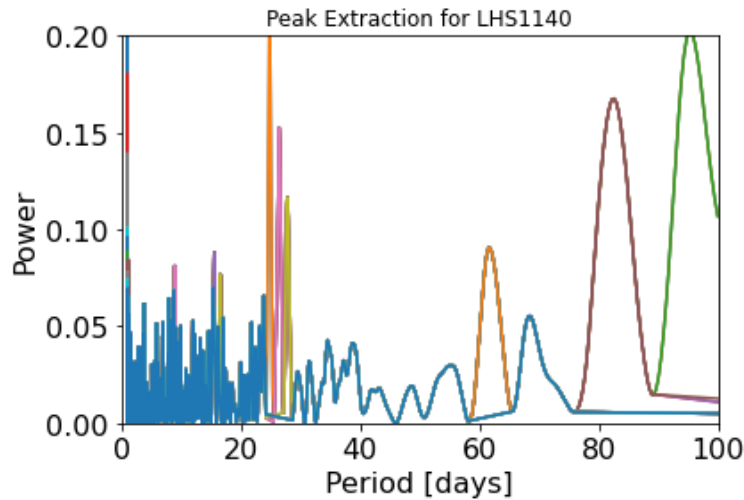
## 3.2. Periodogramas

Obtenidos ya las curvas de velocidad radial, se hace necesario extraer la información relevante para realizar una clasificación. Dado que estamos buscando señales de un fenómeno regular, utilizar periodogramas para buscar posibles periodicidades es nuestra mejor opción. El siguiente paso es calcular periodogramas para todos los exoplanetas en nuestro set de entrenamiento, además de un periodograma para todas las estrellas del grupo NP. Para esto hacemos uso de periodogramas Lomb-Scargle generalizados (Zechmeister and Kürster, 2009). técnica que nos permite obtener de manera confiable un mapeo de las frecuencias contenidas en los datos. Calculamos los periodogramas, buscando periodos entre 1.2 y 1000 días con un muestreo lineal, de esta manera evitamos un problema común para este tipo de datos, debido al muestreo de observaciones propio del instrumento, suele aparecer una frecuencia con periodo de un día en el periodograma, frecuencia que puede ser confundida con un periodo real por nuestros métodos de machine learning, de esta forma contaminando por completo los resultados. Al calcular solo frecuencias por sobre este periodo nos evitamos tener que lidiar con esta interferencia.

## 3.3. Pre-Procesamiento

Una vez obtenidos los periodogramas, necesitamos adecuar los datos a una forma que el modelo sea capaz de utilizar. Es necesario crear por cada estrella un vector de características que sea representable para el modelo de machine learning. Nuestra decisión entonces es crear listas de características para todos los periodogramas, cada elemento de esta lista representando una dimensión distinta del vector de características y porta información sobre el muestreo de potencia de un peak. Estas listas están ordenadas de manera que el primer elemento porta la información del peak más potente del periodograma, el segundo porta sobre el segundo más potente y así sucesivamente. El proceso para generar cada lista de características está ilustrado en la imagen 3.3.1. Durante esta etapa también normalizamos los valores dentro de cada vector, de manera que podamos comparar adecuadamente entre cada objeto de la base de datos.

Además, debido a que nuestro set de datos está desbalanceado en favor de los objetos de tipo NP, hacemos uso de la técnica de sobre-muestreo al aplicar el algoritmo SMOTE (Synthetic Minority Over-sampling Technique) (Chawla et al.,



**Figura 3.3.1:** Proceso de extracción de valores de peak aplicado a la estrella LHS 1140. Este procedimiento consiste en identificar el valor máximo global en el periodograma, para acto seguido buscar los mínimos locales más cercanos a este valor máximo. De esta manera se consigue identificar el peak de mayor valor presente en el periodograma. El siguiente paso es guardar el valor máximo identificado, para después remover del periodograma los valores presentes entre ambos mínimos locales. Cumplido esto, se considera la extracción de peak como finalizada. Se pueden seguir extrayendo nuevos peaks por medio de trabajar con los residuales de un proceso de extracción anterior.

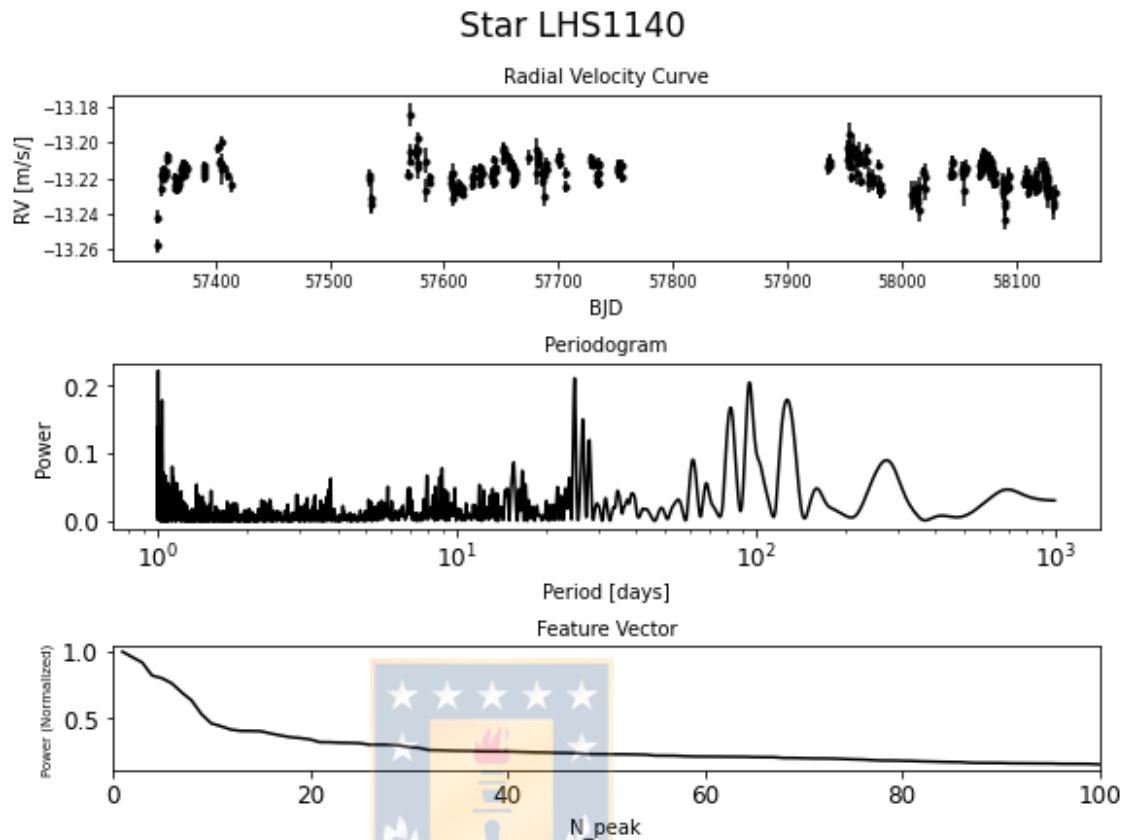
2011) este algoritmo nos permite generar nuevos objetos similares a los objetos PC ya existentes, de manera que nuestro set de entrenamiento queda balanceado al contar con 155 PC (39 reales y 116 sintéticos) y 155 NP (todas estrellas reales).

### 3.4. Búsqueda de Rejilla

Un modelo de machine learning puede utilizar diversas configuraciones, con el fin de adaptar la implementación del modelo conforme a las necesidades del entrenamiento. Estas configuraciones están dictadas por parámetros que regulan el comportamiento del modelo, llamados hiperparámetros. Para obtener una clasificación óptima, es necesario encontrar cuáles son los hiperparámetros adecuados, para esto se realizó una técnica llamada búsqueda de rejilla.

Durante la búsqueda de rejilla, se toma un subconjunto de los datos y se entrenan los modelos de machine learning variando distintas configuraciones de hiperparámetros. Terminado este proceso, se eligen aquellos que muestren mejores resultados.





**Figura 3.3.2:** Ilustración del proceso realizado sobre las estrellas de nuestro set de entrenamiento. Primero calculamos curvas de velocidad radial, utilizando el software especializado Naira (Astudillo-Defru et al., 2017), a partir de las observaciones del espectrógrafo HARPS (Mayor et al., 2003), para luego calcular periodogramas de tipo Lomb-Scargle (VanderPlas et al., 2012) utilizando la implementación provista por Zechmeister and Kürster (2009). El último paso es representar la información del periodograma en una manera reconocible por un algoritmo de aprendizaje de máquinas, para ellos creamos una lista de características utilizando el proceso de **extracción de valores de peak**

La búsqueda de rejilla para los modelos tipo Support Vector Machine consistió en:

- Hiperparámetro  $C$ , que regula la sensibilidad del margen, mientras más grande es este parámetro, más sensible es el hiperplano a la posición de los elementos del conjunto de entrenamiento. Se estudiaron posibles valores desde  $C = 0,1$  a  $C = 10,000$ . El valor escogido fue  $C = 100$
- Kernel, la función utilizada para resolver un problema de separación de regiones en el caso de que este no sea lineal. El kernel mapea las posiciones de los datos a un espacio de mayores dimensiones en el cual estos sean linealmente separables. Se estudió el uso de funciones kernel de tipo lineal y

RBF. El kernel con mejor desempeño resulto ser el de tipo RBF.

- Hiperparámetro  $\gamma$ , este valor cuantifica que tan importante es clasificar correctamente cada elemento durante el entrenamiento. Se probaron valores entre 0,00001 y 100, el valor final escogido fue de 10.

Para los modelos tipo Random Forest, la búsqueda de rejilla consistió en probar los siguientes hiperparámetros:

- Número de estimadores, la cantidad de árboles de decisión que se generaría para cada modelo, se probaron valores entre 10 árboles y 1000 árboles. Se estimó que la cantidad óptima era de 100 árboles.
- Bagging, para reducir la varianza dentro de estos modelos, se puede utilizar una técnica tipo bootstrap. En ella cada árbol es entrenado con una submuestra del conjunto de entrenamiento en lugar de que todos los árboles compartan los mismos datos. Se probaron modelos que podían utilizar o no utilizar esta técnica y se determinó que los mejores resultados se obtenían al usarla.
- Criterio de división. Al momento de crear una división dentro del árbol se trata de buscar la división que reduzca lo más posible la impureza de cada rama, por tanto, buscamos la división que más relevante para la clasificación. Los criterios de impureza sirven para evaluar que tan ordenada está la base de datos después de realizar la división. Para ello se probó el uso del criterio Gini y el criterio de entropía de Shannon. El criterio Gini resulto ser el de mejor desempeño.

## 3.5. Entrenamiento

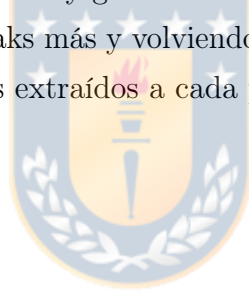
Con los vectores de características ya listos, y cada uno debidamente etiquetado (marcando a qué grupo pertenece PC o NP) podemos comenzar a entrenar. Utilizamos el paquete de Python `sklearn` para entrenar modelos tipo Random Forest (RF) y Support Vector Machine (SVM).

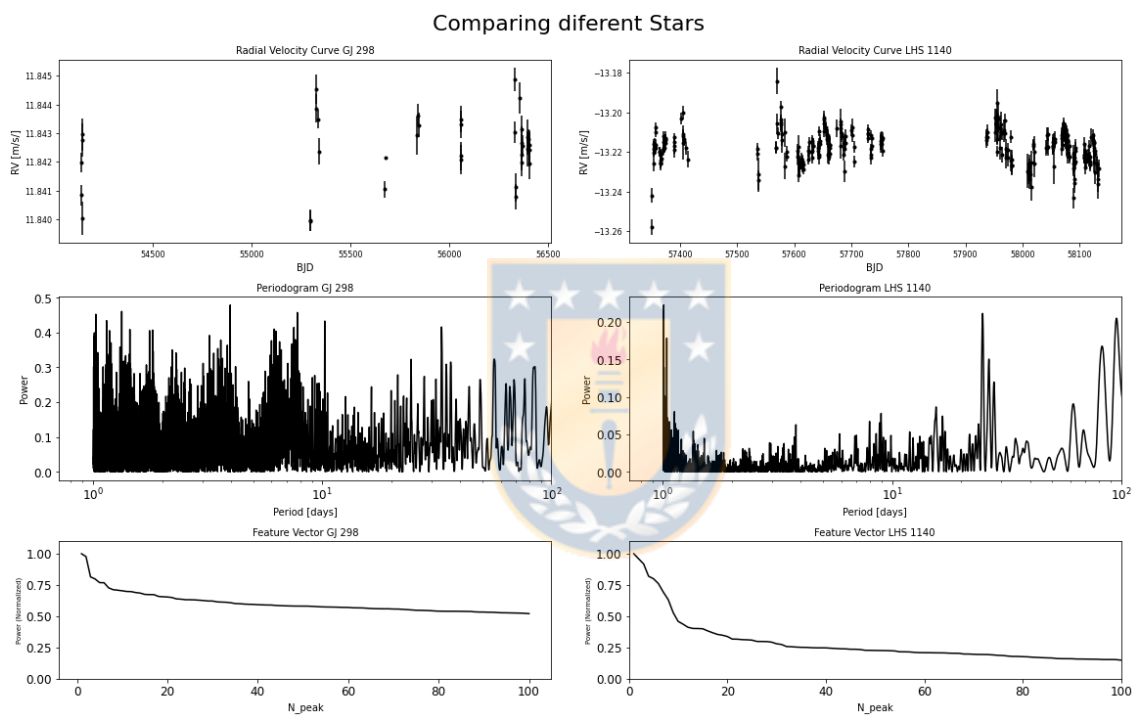
Probando distintas combinaciones de hiperparámetros para estos modelos, llegamos a la conclusión de que los mejores parámetros para los modelos de tipo Random Forest era el utilizar 100 estimadores (árboles), utilizar muestreo bootstrap del

mismo tamaño del conjunto de entrenamiento (`max_sample= 0`) y el criterio Gini para medir la impureza de cada rama.

Para los modelos tipo SVM, los hiperparámetros que dieron mejores resultados consistían en usar un kernel del tipo función de base radial (rbf), un valor de 100 para el parámetro de sensibilidad de margen  $C$  y un valor de 10 para el parámetro Gamma, que asigna la importancia en la correcta clasificación de cada objeto al momento de situar al hiperplano.

El proceso de entrenamiento consistía en partir extrayendo los valores de los 5 peaks más grandes de cada periodograma, una vez completada la extracción, se generaba una lista de características usando estos valores para luego entrenar ambos tipos de modelo utilizando las listas de características generadas, repitiendo este proceso 100 veces a fin de estimar el desempeño medio de cada modelo. Una vez finalizado el entrenamiento y guardado las métricas de desempeño, se repetía el proceso extrayendo 5 peaks más y volviendo a entrenar y comparar, hasta llegar a un máximo de 100 peaks extraídos a cada periodograma.





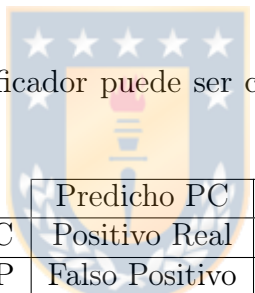
**Figura 3.5.1:** Comparación de la metodología aplicada sobre las estrellas de ambos grupos. En la izquierda podemos ver la curva de velocidad radial, periodograma y lista de características para la estrella del grupo NP GJ 298, en el lado derecho podemos ver los gráficos respectivos para la estrella del grupo PC LHS 1140.

# Capítulo 4

## Análisis

### 4.1. Resultados

El desempeño de un clasificador puede ser cuantificado utilizando la siguiente tabla:



	Predicho PC	Predicho NP
Real PC	Positivo Real	Falso Negativo
Real NP	Falso Positivo	Negativo Real

Sabemos de antemano si es que cada periodograma de nuestra base de datos contiene una señal planetaria. Si nuestros modelos de machine learning identifican una señal planetaria en un objeto que tiene en realidad una señal planetaria, estaríamos ante un **Positivo Real (TP)**, en caso contrario, si no identifican una señal planetaria cuando en la realidad existe una, estaríamos ante un **Falso Negativo (FN)**.

Por otro lado, si se da el caso de que nuestros modelos de machine learning identifican una señal planetaria, cuando en la realidad no existe ninguna, estaríamos tratando con un **Falso Positivo (FP)**. Por último podemos tener el caso de que no se detecte nada cuando no existe nada que detectar, este sería el caso de un **Negativo Real (TN)**. A partir de estas definiciones, podemos definir distintas métricas para cuantificar el desempeño de nuestros clasificadores.

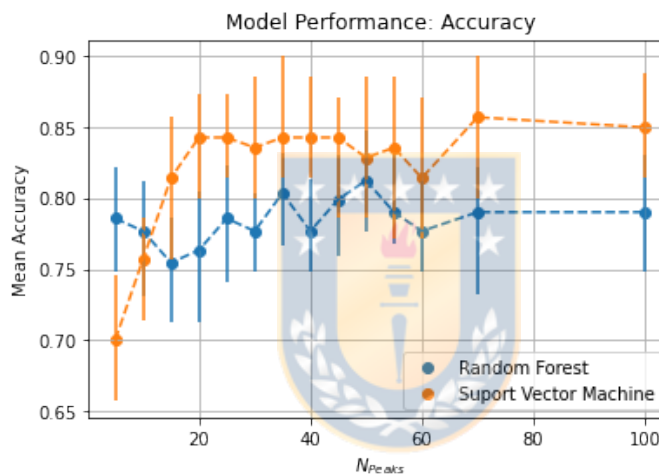
### 4.1.1. Exactitud

La exactitud se refiere a la cantidad de veces que nuestro clasificador acierta en la clasificación. Matemáticamente, la podemos definir como:

$$Exactitud = \frac{TP + TN}{TP + TN + FP + FN} \quad (4.1.1)$$

Lo que equivale al número de clasificaciones acertadas, dividido por el total de elementos clasificados.

Los resultados para nuestros clasificadores son los siguientes:



**Figura 4.1.1:** Comparación de la exactitud entre los modelos SVM y RF

Los modelos de tipo Random Forest muestran un desempeño superior a los modelos de tipo Support Vector Machine cuando el número de peaks utilizado es bajo, pero esta ventaja se disipa conforme se utilizan más y más peaks. Los modelos de tipo Support Vector Machine muestran un desempeño marcadamente mejor que los modelos Random Forest a partir de los 15 peaks, obteniendo consistentemente una exactitud cercana al 85% , mientras que el desempeño de los modelos Random Forest ronda el 77% para esta métrica.

### 4.1.2. Precisión

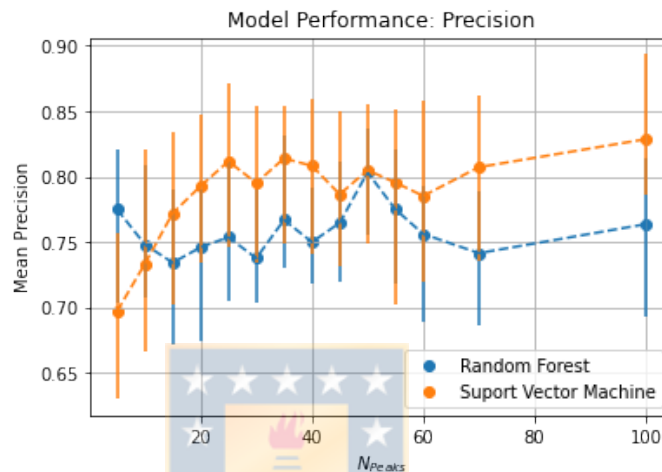
La precisión es la evaluación de que tan certero es nuestro clasificador al momento de marcar a un objeto como parte de nuestra clase objetivo. Matemáticamente se

define como:

$$Precision = \frac{TP}{TP + FP} \quad (4.1.2)$$

De esta manera estamos cuantificando que cantidad de falsos positivos estamos obteniendo al clasificar.

Los resultados para nuestros modelos son los siguientes:



**Figura 4.1.2:** Comparación de la precisión entre los modelos SVM y RF

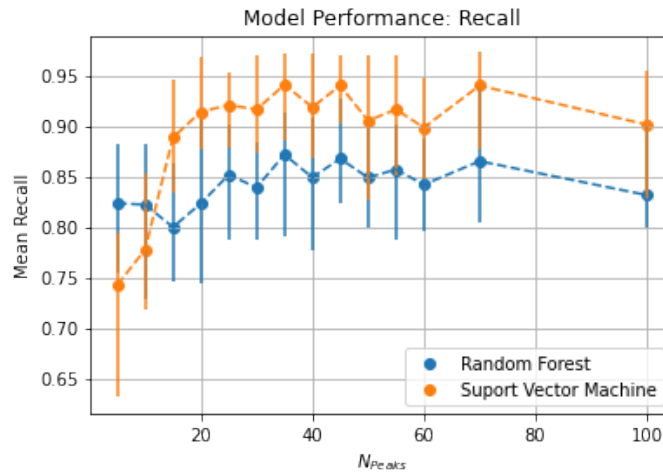
Observamos nuevamente que los modelos Random Forest tienen una ligera ventaja sobre los modelos tipo Support Vector Machine, ventaja que se reduce conforme aumentamos el número de peaks representando cada periodograma. Los modelos tipo Support Vector Machine muestran un desempeño ligeramente mayor (alrededor de un 5%), rondando una precisión sobre el 80%. Podemos notar también, que al llegar a 50 peaks, ambos modelos tienen un desempeño similar, superior al 80%, pero ambos vuelven a sus anteriores tendencias conforme avanzamos hacia los 100 peaks.

### 4.1.3. Recall

El recall es una métrica dedicada a cuantificar la cantidad de elementos de nuestra clase objetivo, que perdemos al clasificarlos erróneamente. Matemáticamente, lo definimos como:

$$Recall = \frac{TP}{TP + FN} \quad (4.1.3)$$

Para nuestros modelos, los resultados son los siguientes:



**Figura 4.1.3:** Comparación del Recall entre los modelos SVM y RF

Podemos observar que el desempeño de ambos modelos es superior a 80 % en la mayoría de casos. Nuevamente, observamos una ligera ventaja de los modelos Random Forest sobre los Support Vector Machine que se va reduciendo conforme aumentan los peaks utilizados. Los modelos tipo Support Vector Machine muestran un desempeño notable, llegando a obtener un recall del 94 %, lo que significa que podemos detectar casi la totalidad de elementos existentes en nuestra base de datos.

#### 4.1.4. Puntaje F1

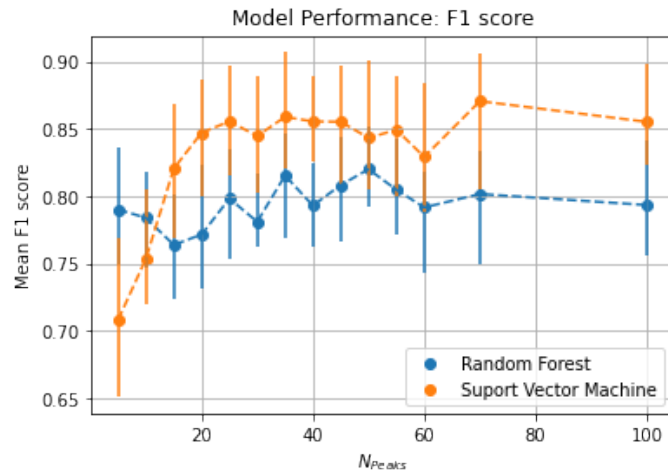
El puntaje F1 es una métrica que nos permite resumir el desempeño de nuestros modelos de machine learning al englobar los valores de precisión y recall. Se define como la media armónica entre ambos de la forma:

$$F1 = 2 \frac{Precision * Recall}{Precision + Recall} = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \quad (4.1.4)$$

Los resultados de nuestros modelos son los siguientes:

Las métricas de puntaje F1 son similares a nuestras métricas de exactitud, observando una ligera ventaja a bajos peaks para los modelos tipo Random Forest, que se va perdiendo, favoreciendo a los modelos Support Vector Machine, conforme aumentamos el número de peaks. Esta métrica nos permite observar una notoria ventaja de los modelos Support Vector Machine, que es mantenida





**Figura 4.1.4:** Comparación del puntaje F1 entre los modelos SVM y RF

durante el resto del entrenamiento, también nos indica que el balance óptimo de peaks para obtener tanto una buena precisión como un buen recall, se obtiene al usar 70 peaks para los modelos Support Vector Machine y 50 peaks para los modelos Random Forest.

#### 4.1.5. Tablas

Las métricas de desempeño de nuestros modelos durante el entrenamiento están expresadas en las siguientes tablas:

Numero de Peaks	Exactitud	Precisión	Recall	Puntaje F1
5.0	$0,78 \pm_{0,04}^{0,04}$	$0,77 \pm_{0,07}^{0,05}$	$0,82 \pm_{0,07}^{0,06}$	$0,79 \pm_{0,04}^{0,05}$
10.0	$0,77 \pm_{0,05}^{0,04}$	$0,74 \pm_{0,04}^{0,06}$	$0,82 \pm_{0,09}^{0,06}$	$0,78 \pm_{0,04}^{0,03}$
15.0	$0,75 \pm_{0,04}^{0,03}$	$0,73 \pm_{0,06}^{0,06}$	$0,8 \pm_{0,05}^{0,06}$	$0,76 \pm_{0,04}^{0,04}$
20.0	$0,76 \pm_{0,05}^{0,04}$	$0,74 \pm_{0,07}^{0,04}$	$0,82 \pm_{0,08}^{0,08}$	$0,77 \pm_{0,04}^{0,05}$
25.0	$0,78 \pm_{0,04}^{0,04}$	$0,75 \pm_{0,05}^{0,06}$	$0,85 \pm_{0,06}^{0,05}$	$0,79 \pm_{0,04}^{0,04}$
30.0	$0,77 \pm_{0,03}^{0,03}$	$0,73 \pm_{0,03}^{0,05}$	$0,83 \pm_{0,05}^{0,05}$	$0,78 \pm_{0,02}^{0,04}$
35.0	$0,80 \pm_{0,04}^{0,03}$	$0,76 \pm_{0,04}^{0,06}$	$0,87 \pm_{0,08}^{0,04}$	$0,81 \pm_{0,05}^{0,03}$
40.0	$0,77 \pm_{0,03}^{0,04}$	$0,75 \pm_{0,03}^{0,04}$	$0,84 \pm_{0,07}^{0,06}$	$0,79 \pm_{0,03}^{0,03}$
45.0	$0,79 \pm_{0,04}^{0,03}$	$0,76 \pm_{0,05}^{0,05}$	$0,86 \pm_{0,04}^{0,06}$	$0,80 \pm_{0,04}^{0,04}$
50.0	$0,81 \pm_{0,04}^{0,04}$	$0,80 \pm_{0,05}^{0,03}$	$0,84 \pm_{0,05}^{0,05}$	$0,82 \pm_{0,03}^{0,03}$
55.0	$0,79 \pm_{0,02}^{0,04}$	$0,77 \pm_{0,05}^{0,04}$	$0,85 \pm_{0,07}^{0,04}$	$0,80 \pm_{0,03}^{0,04}$
60.0	$0,77 \pm_{0,03}^{0,04}$	$0,75 \pm_{0,07}^{0,04}$	$0,84 \pm_{0,05}^{0,05}$	$0,79 \pm_{0,05}^{0,03}$
70.0	$0,79 \pm_{0,06}^{0,03}$	$0,74 \pm_{0,06}^{0,05}$	$0,86 \pm_{0,06}^{0,07}$	$0,80 \pm_{0,05}^{0,03}$
100.0	$0,79 \pm_{0,04}^{0,04}$	$0,76 \pm_{0,07}^{0,05}$	$0,83 \pm_{0,03}^{0,07}$	$0,79 \pm_{0,04}^{0,05}$

**Cuadro 4.1.1:** Métricas de desempeño para los modelos de tipo Random Forest.

Numero de Peaks	Exactitud	Precisión	Recall	Puntaje F1
5.0	$0,70 \pm_{0,04}^{0,05}$	$0,69 \pm_{0,07}^{0,06}$	$0,74 \pm_{0,11}^{0,05}$	$0,70 \pm_{0,06}^{0,06}$
10.0	$0,75 \pm_{0,04}^{0,03}$	$0,73 \pm_{0,07}^{0,09}$	$0,77 \pm_{0,06}^{0,08}$	$0,75 \pm_{0,03}^{0,05}$
15.0	$0,81 \pm_{0,06}^{0,04}$	$0,77 \pm_{0,07}^{0,06}$	$0,88 \pm_{0,06}^{0,06}$	$0,82 \pm_{0,05}^{0,05}$
20.0	$0,84 \pm_{0,04}^{0,03}$	$0,79 \pm_{0,06}^{0,05}$	$0,91 \pm_{0,04}^{0,05}$	$0,84 \pm_{0,05}^{0,04}$
25.0	$0,84 \pm_{0,03}^{0,03}$	$0,81 \pm_{0,06}^{0,06}$	$0,92 \pm_{0,04}^{0,03}$	$0,85 \pm_{0,04}^{0,04}$
30.0	$0,83 \pm_{0,04}^{0,05}$	$0,79 \pm_{0,06}^{0,06}$	$0,91 \pm_{0,04}^{0,05}$	$0,84 \pm_{0,04}^{0,04}$
35.0	$0,84 \pm_{0,03}^{0,06}$	$0,81 \pm_{0,07}^{0,04}$	$0,94 \pm_{0,06}^{0,03}$	$0,85 \pm_{0,04}^{0,05}$
40.0	$0,84 \pm_{0,03}^{0,04}$	$0,80 \pm_{0,07}^{0,05}$	$0,91 \pm_{0,05}^{0,05}$	$0,85 \pm_{0,03}^{0,03}$
45.0	$0,84 \pm_{0,06}^{0,03}$	$0,78 \pm_{0,06}^{0,06}$	$0,94 \pm_{0,04}^{0,03}$	$0,85 \pm_{0,05}^{0,04}$
50.0	$0,82 \pm_{0,04}^{0,06}$	$0,80 \pm_{0,06}^{0,05}$	$0,90 \pm_{0,08}^{0,06}$	$0,84 \pm_{0,04}^{0,06}$
55.0	$0,83 \pm_{0,06}^{0,05}$	$0,79 \pm_{0,09}^{0,06}$	$0,91 \pm_{0,06}^{0,05}$	$0,84 \pm_{0,05}^{0,04}$
60.0	$0,81 \pm_{0,04}^{0,06}$	$0,78 \pm_{0,07}^{0,07}$	$0,89 \pm_{0,05}^{0,05}$	$0,82 \pm_{0,04}^{0,06}$
70.0	$0,85 \pm_{0,06}^{0,04}$	$0,80 \pm_{0,07}^{0,06}$	$0,94 \pm_{0,06}^{0,03}$	$0,87 \pm_{0,07}^{0,04}$
100.0	$0,85 \pm_{0,04}^{0,04}$	$0,82 \pm_{0,04}^{0,06}$	$0,90 \pm_{0,07}^{0,05}$	$0,85 \pm_{0,03}^{0,04}$

**Cuadro 4.1.2:** Métricas de desempeño para los modelos de tipo Support Vector Machine.

# Capítulo 5

## Discusión

### 5.1. Importancia de características

Podemos ver que al aumentar el número de peaks utilizados para representar los periodogramas no existe mejora significativa a partir de cierto punto. Podemos argumentar que esto se debe a que a partir de cierto número de peaks, la información provista no es relevante para la clasificación, en lugar se está representando información referente al ruido del periodograma. Como forma de comprobarlo es útil revisar que tan importante es cada característica al momento de clasificar.

La importancia de características es una propiedad que se puede calcular sobre los modelos de tipo Random Forest. Al hacerlo estimamos que tan relevante es cada característica (en nuestro caso cada peak) al momento de clasificar. Un modelo Random Forest está compuesto por varios estimadores individuales o *Árboles de decisión*. Cada árbol de decisión funciona por medio de dividir su set de entrenamiento en subgrupos más pequeños o *ramas* del árbol, cada división es denominada como *Nodo*. Podemos asignar un valor de importancia a cada Nodo por medio de medir que tan ordenadas son las ramas que salen de él, dando un mayor puntaje a aquellos nodos que nos otorguen las ramas con composición más homogénea (solo elementos de una misma clase). La fórmula para asignar estos valores es llamada criterio de impureza, el utilizado en este trabajo es el **Criterio Gini**:

$$C_{Gini} = \sum_{i=0}^1 f_i(1 - f_i) \quad (5.1.1)$$

Donde  $f_i$  se refiere a que tan frecuente es encontrar la etiqueta  $i$  en el nodo ( $i = 0$  para los objetos PC,  $i = 1$  para los objetos NP). Por ejemplo, un nodo con 3 estrellas del grupo PC y 2 estrellas del grupo NP ( $f_0 = 3/5, f_1 = 2/5$ ) tiene una impureza del 0.48, mientras que un nodo "puro.<sup>en</sup> el cual hay 5 elementos del mismo tipo, tiene una impureza de 0.

La importancia de cada nodo individual para la clasificación es definida de la forma:

$$n = w * C - w_{izq}C_{izq} - w_{der}C_{der} \quad (5.1.2)$$

$w$  se refiere al peso del nodo, equivalente a la proporción de objetos del total, que alcanzan a llegar a este nodo individual,  $C$  es el valor de impureza del nodo y los subíndices  $der, izq$  se refieren a los nodos de las ramas subsiguientes.

Por último, podemos estimar que tan importante es cada característica, por medio de calcular cuantos nodos están utilizando esta característica y que tan importante son estos nodos:

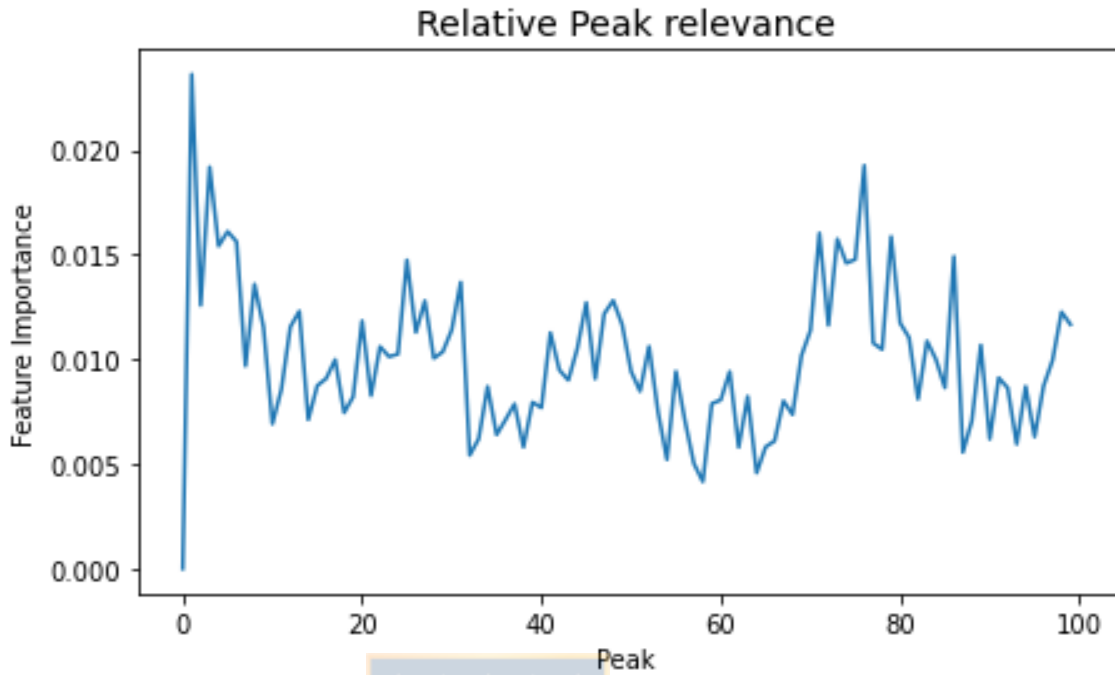
$$FI_i = \frac{\sum_j n_j}{T \sum_k n_k} \quad (5.1.3)$$

Donde  $j$  son todos los nodos que utilizan la característica  $i$ ,  $k$  son todos los nodos en cada árbol y  $T$  es el número de árboles generados para el modelo Random Forest.

La figura 5.1.1 representa la importancia de características para cada peak en nuestro modelo. Como los peaks están ordenados de mayor a menor, podemos notar que los más relevantes son aquellos cercanos al máximo valor de potencia junto a aquellos peaks cercanos a la cola del vector. En particular, los peaks más importantes dentro del periodograma son aquellos en el rango 2 a 20 y aquellos en el rango 70 a 80. Como estaba previsto, la importancia asignada al primer peak es 0, lo cual es entendible si recordamos que estamos trabajando con la potencia normalizada de cada periodograma, por lo que este valor es igual a uno para todos los objetos en nuestra base de datos. Como este valor no varía de objeto a objeto, no puede ser utilizado para distinguir entre clases distintas.

Este comportamiento se vuelve más claro una vez analizamos la forma de las listas de características en nuestra base de datos.

La figura 5.1.2 grafica la potencia normalizada al peak máximo de cada periodograma para cada estrella del grupo de entrenamiento. Podemos ver que las

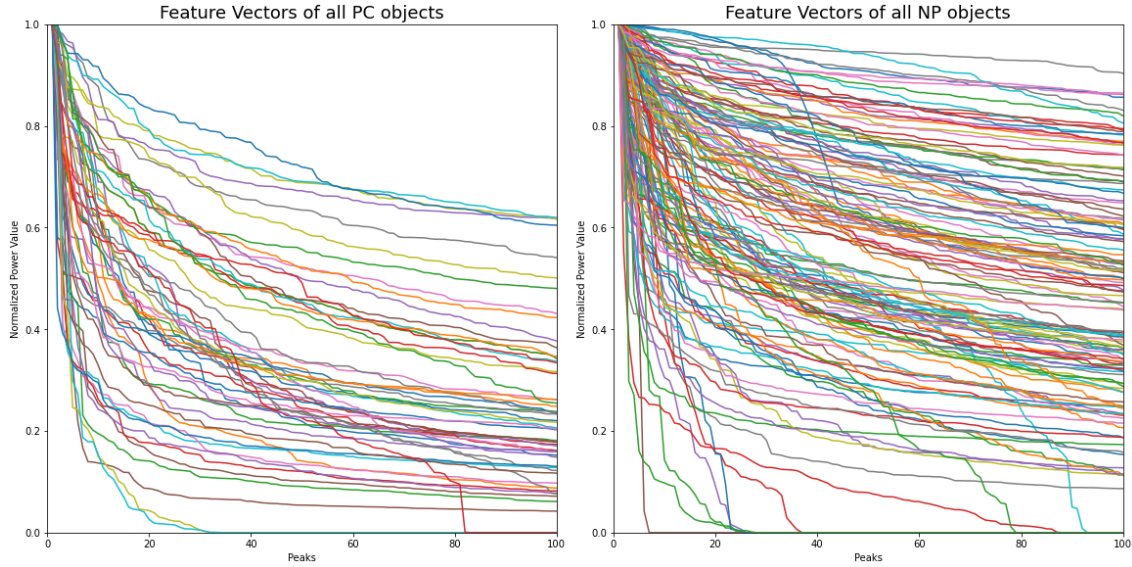


**Figura 5.1.1:** Importancia de características para un modelo Random Forest entrenado con 100 peaks, 500 estimadores y utilizando el criterio de impureza Gini.

estrellas de la clase PC (panel izquierdo) exhiben una forma similar en la cual se puede observar una caída súbita de potencia a partir del primer peak. A partir del peak número 40 podemos notar que el valor de potencia es menor a la mitad que el valor de potencia máximo para el 80 % de estrellas en la clase PC.

Por otro lado, podemos observar una gran variedad de curvas distintas para las estrellas del grupo NP. En particular, varias estrellas presentan una distribución plana, en la cual la caída en la potencia es bastante lenta, mientras que otras exhiben un descenso gradual. Interesantemente, también notamos que 37 estrellas de la clase NP poseen una forma similar a estrellas de la clase PC.

En definitiva, podemos concluir que la mejora en el desempeño de los algoritmos está directamente ligada a la información provista por los peaks en el rango 1 a 20, mientras que los peaks 70 a 80 son responsables del segundo incremento percibido a  $N_{peaks} > 70$

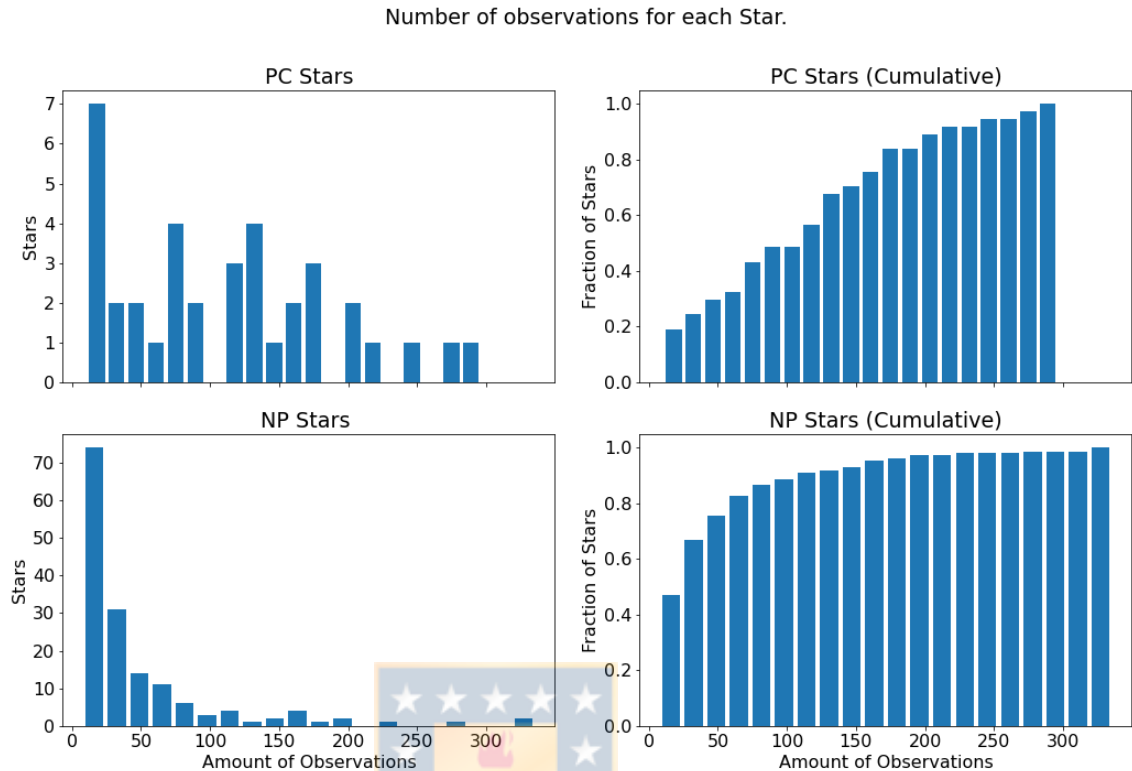


**Figura 5.1.2:** Potencia normalizada para cada peak en orden descendente para todos los periodogramas de nuestra muestra. Cada línea corresponde a una lista de características para una estrella del grupo PC (izquierda) o una estrella del grupo NP (derecha). La potencia de cada peak es normalizada al valor de potencia máximo dentro de su respectivo periodograma, de esta manera  $N_{peak} = 1$  tiene un valor de uno para cada estrella.

## 5.2. Número de Observaciones para las estrellas en nuestro estudio.

Se estima que la mayoría de estrellas en nuestra galaxia poseen exoplanetas. Por ejemplo, observaciones de eventos de microlensings sugieren que cada estrella posee al menos un planeta (Cassan et al., 2012). Observaciones realizadas con HARPS nos sugieren que planetas similares a los presentes en nuestra base de datos (orbitando estrellas de clase M, masas cercanas a la Tierra, ubicados en la zona habitable) existen en una frecuencia  $\eta_{\oplus} = 0,41^{+0,54}_{-0,17}$  planetas por estrella (Bonfils et al., 2013a). Esto no se cumple para las estrellas en nuestra base de datos, donde el grupo NP es 5 veces más grande que el grupo PC. Las estimaciones anteriores nos llevarían a suponer que deberían existir muchos más exoplanetas en nuestra muestra, pero podemos explicar esta diferencia al analizar el número de observaciones para cada objeto.

Como muestra la figura 5.2.1, las estrellas del grupo PC poseen en promedio un mayor número de observaciones ( $> 100$  observaciones para el grupo PC,  $< 50$



**Figura 5.2.1:** Histograma (Izquierda) e Histograma Acumulado (Derecha) de las observaciones para cada estrella en los grupos PC (arriba) y NP (Abajo) en nuestra base de datos.

observaciones para el grupo NP). Esto puede servir como explicación a la diferencia entre ambos grupos, debido a que existen pocas observaciones sobre el grupo NP, no se han podido encontrar exoplanetas orbitando alrededor de estas estrellas, por lo que no existe señal planetaria que nuestros modelos puedan detectar.

### 5.3. Comparación con otros resultados

Las técnicas de aprendizaje supervisado se han utilizado anteriormente en astronomía. En el área de exoplanetas, han existido estudios que buscaban entrenar modelos para la detección de planetas extrasolares, pero utilizando otras técnicas de detección. En el estudio de [Thompson et al. \(2015\)](#) se definieron las bases para el algoritmo que más tarde filtraría los candidatos planetarios del Robovetter ([Coughlin et al., 2016](#)), un catálogo completamente automatizado de señales planetarias en observaciones del telescopio espacial Kepler. El algoritmo del Robovetter adopta una métrica construida utilizando Proyecciones Preservantes

de Localidad (Métrica LPP, por Locality Preserving Projections), una técnica de reducción de dimensionalidad, que agrupa aquellos objetos que poseen señales planetarias en nuevo espacio vectorial. Para el entrenamiento contaron con el catálogo de objetos de interés del telescopio Kepler DR24 y consiguieron una performance de  $F1 = 0,957$

De manera similar, [Shallue and Vanderburg \(2018\)](#) entrenarían Redes Neurales de varios tipos (Lineares, Completamente conectadas y Convolucionares), en estos modelos se hace uso de una red neural, una agrupación de neuronas organizadas en distintas capas, las neuronas interactúan entre ellas hasta dar con una solución, la cual en este caso corresponde a la clase a la que pertenecería la estrella. Utilizando el catálogo Q1-Q16 DR24 del telescopio Kepler para su entrenamiento, se consiguió un desempeño máximo con el mejor modelo (una Red Neural Convolutacional) de  $F1 = 0,92$ .

Más tarde, la Métrica LPP sería utilizada para identificar 1402 nuevos objetos de interés en el catálogo del Robovetter ([Coughlin et al., 2016](#)), de los cuales 402 fueron marcados como candidatos planetarios. Mientras tanto, la Red Neural Convolutacional entrenada por [Shallue and Vanderburg \(2018\)](#) sería utilizada para identificar dos nuevos exoplanetas, orbitando los sistemas Kepler-80 y Kepler-90, transformando a Kepler-90 en una de las estrellas con mayor número de planetas orbitando (8 planetas en órbita, empatado con el Sol).

En comparación, los modelos entrenados durante este estudio tienen un desempeño ligeramente inferior, con una media de  $F1 = 0,8$  fluctuando  $< 0,05$  en el rango  $20 \leq N_{peaks} \leq 100$  para los modelos Random Forest, mientras que los modelos Support Vector Machine exhiben una media de  $F1 = 0,85$  con fluctuaciones  $\leq 0,05$  alrededor del mismo rango. Podemos argumentar que el motivo principal detrás de esta diferencia en desempeño radica en el tamaño de los grupos de entrenamiento. Nuestra base de datos consiste principalmente en la muestra de estrellas de clase M observada por [Bonfils et al. \(2013b\)](#), la cual ha sido expandida con observaciones adicionales de estrellas de clase M realizadas hasta el año 2020, pero aun así es 10 veces más pequeño que el catálogo Q1-Q17 DR24 de Kepler. Futuras observaciones nos ayudarían a expandir nuestra base de datos y junto a ello mejorar el desempeño futuro de estos modelos.



# Capítulo 6

## Conclusión

### 6.1. Conclusión

A lo largo de este trabajo hemos explorado la posibilidad de usar técnicas de aprendizaje de máquinas para la detección de señales planetarias. Los modelos que hemos entrenado han mostrado un muy buen desempeño en esta fase de entrenamiento, pudiendo identificar gran cantidad de las señales planetarias existentes en nuestra base de datos con una baja cantidad de falsos positivos.

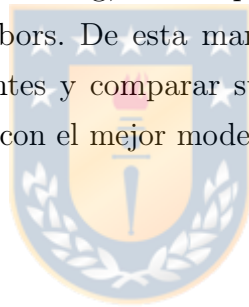
Hemos observado también que los modelos tipo Support Vector Machine presentan un desempeño marcadamente superior a los modelos tipo Random Forest, tanto en precisión como en recall. Las métricas de desempeño también nos muestran que los mejores resultados para ambos modelos se obtienen en un rango entre 30 a 70 peaks.

Hemos estudiado como afecta la presencia de cada peak individual a la calidad del modelo, y hemos determinado que los mejores modelos son aquellos entrenados con 70 peaks para ambos tipos de modelo. Por tanto, el mejor modelo es uno de tipo SVM, utilizando un kernel tipo RBF, un valor de C igual a 100, un valor de  $\gamma = 10$  y una lista de características de 70 peaks. Este modelo presenta una Exactitud del  $85\% \pm_4^6$ , una precisión del  $80\% \pm_6^7$  y un recall del  $94\% \pm_6^3$ .

En definitiva, hemos demostrado que es posible detectar señales planetarias en datos de velocidad radial, utilizando modelos de machine learning relativamente simples. Cabe acotar que estos modelos son solo capaces de indicar la presencia

de una señal planetaria, más no pueden ser usados para obtener las propiedades físicas de cada exoplaneta, como serían la masa, la distancia a la estrella o el periodo de la órbita. Estas propiedades todavía tienen que ser obtenidas por medio de un análisis manual de las observaciones existentes. La contribución de estos modelos radica en que son una alternativa factible a la estimación de falsa alarma (False Alarm Probability o FAP) que se utiliza para corroborar la presencia de una señal planetaria. Esta estimación es usualmente realizada por medio de una simulación numérica que demanda tanto tiempo como recursos computacionales. Nuestros modelos son capaces de servir como una alternativa rápida y eficiente a esta estimación.

Como proyectos futuros sugerimos tanto la ampliación del set de entrenamiento, incorporando nuevas observaciones del telescopio HARPS, como la implementación de otros modelos de machine learning, como pueden ser redes neurales convolucionales o k-nearest neighbors. De esta manera podríamos mejorar el desempeño de los modelos existentes y comparar sus capacidades con nuevos modelos, todo con el fin de contar con el mejor modelo posible para la detección de planetas extrasolares.



## Bibliografía

- Astudillo-Defru, N., Forveille, T., Bonfils, X., Ségransan, D., Bouchy, F., Delfosse, X., Lovis, C., Mayor, M., Murgas, F., Pepe, F., Santos, N. C., Udry, S., and Wünsche, A. (2017). The HARPS search for southern extra-solar planets. XLI. A dozen planets around the M dwarfs GJ 3138, GJ 3323, GJ 273, GJ 628, and GJ 3293. , 602:A88.
- Baranne, A., Mayor, M., and Poncet, J. L. (1979). Coravel— A new tool for radial velocity measurements. *Vistas in Astronomy*, 23(4):279–316.
- Bonfils, X., Delfosse, X., Udry, S., Forveille, T., Mayor, M., Perrier, C., Bouchy, F., Gillon, M., Lovis, C., Pepe, F., Queloz, D., Santos, N. C., Ségransan, D., and Bertaux, J. L. (2013a). The HARPS search for southern extra-solar planets. XXXI. The M-dwarf sample. , 549:A109.
- Bonfils, X., Delfosse, X., Udry, S., Forveille, T., Mayor, M., Perrier, C., Bouchy, F., Gillon, M., Lovis, C., Pepe, F., Queloz, D., Santos, N. C., Ségransan, D., and Bertaux, J. L. (2013b). The HARPS search for southern extra-solar planets. XXXI. The M-dwarf sample. , 549:A109.
- Borucki, W. J., Koch, D., Basri, G., Batalha, N., Brown, T., Caldwell, D., Caldwell, J., Christensen-Dalsgaard, J., Cochran, W. D., DeVore, E., Dunham, E. W., Dupree, A. K., Gautier, T. N., Geary, J. C., Gilliland, R., Gould, A., Howell, S. B., Jenkins, J. M., Kondo, Y., Latham, D. W., Marcy, G. W., Meibom, S., Kjeldsen, H., Lissauer, J. J., Monet, D. G., Morrison, D., Sasselov, D., Tarter, J., Boss, A., Brownlee, D., Owen, T., Buzasi, D., Charbonneau, D., Doyle, L., Fortney, J., Ford, E. B., Holman, M. J., Seager, S., Steffen, J. H., Welsh, W. F., Rowe, J., Anderson, H., Buchhave, L., Ciardi, D., Walkowicz, L., Sherry, W., Horch, E., Isaacson, H., Everett, M. E., Fischer, D., Torres, G., Johnson, J. A., Endl, M., MacQueen, P., Bryson, S. T., Dotson, J., Haas, M., Kolodziejczak, J., Van Cleve, J., Chandrasekaran, H., Twicken, J. D., Quintana, E. V., Clarke, B. D., Allen, C., Li, J., Wu, H., Tenenbaum, P., Verner, E., Bruhweiler, F., Barnes, J., and Prsa, A. (2010). Kepler Planet-Detection Mission: Introduction and First Results. *Science*, 327(5968):977.
- Cassan, A., Kubas, D., Beaulieu, J. P., Dominik, M., Horne, K., Greenhill, J., Wambsganss, J., Menzies, J., Williams, A., Jørgensen, U. G., Udalski, A., Bennett, D. P., Albrow, M. D., Batista, V., Brilliant, S., Caldwell, J. A. R., Cole, A., Coutures, C., Cook, K. H., Dieters, S., Dominis Prester, D., Donatowicz,

- J., Fouqué, P., Hill, K., Kains, N., Kane, S., Marquette, J. B., Martin, R., Pollard, K. R., Sahu, K. C., Vinter, C., Warren, D., Watson, B., Zub, M., Sumi, T., Szymański, M. K., Kubiak, M., Poleski, R., Soszynski, I., Ulaczyk, K., Pietrzyński, G., and Wyrzykowski, Ł. (2012). One or more bound planets per Milky Way star from microlensing observations. , 481(7380):167–169.
- Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2011). SMOTE: Synthetic Minority Over-sampling Technique. *arXiv e-prints*, page arXiv:1106.1813.
- Coughlin, J. L., Mullally, F., Thompson, S. E., Rowe, J. F., Burke, C. J., Latham, D. W., Batalha, N. M., Ofir, A., Quarles, B. L., Henze, C. E., Wolfgang, A., Caldwell, D. A., Bryson, S. T., Shporer, A., Catanzarite, J., Akeson, R., Barclay, T., Borucki, W. J., Boyajian, T. S., Campbell, J. R., Christiansen, J. L., Girouard, F. R., Haas, M. R., Howell, S. B., Huber, D., Jenkins, J. M., Li, J., Patil-Sabale, A., Quintana, E. V., Ramirez, S., Seader, S., Smith, J. C., Tenenbaum, P., Twicken, J. D., and Zamudio, K. A. (2016). Planetary Candidates Observed by Kepler. VII. The First Fully Uniform Catalog Based on the Entire 48-month Data Set (Q1-Q17 DR24). , 224(1):12.
- Delmotte, N., Dolensky, M., Padovani, P., Retzlaff, J., Rit e, C., Rosati, P., Slijkhuis, R., Wicencec, A., Fernique, P., and Micol, A. (2006). ESO Science Archive Interfaces. In Gabriel, C., Arviset, C., Ponz, D., and Enrique, S., editors, *Astronomical Data Analysis Software and Systems XV*, volume 351 of *Astronomical Society of the Pacific Conference Series*, page 690.
- Mayor, M., Pepe, F., Queloz, D., Bouchy, F., Rupprecht, G., Lo Curto, G., Avila, G., Benz, W., Bertaux, J. L., Bonfils, X., Dall, T., Dekker, H., Delabre, B., Eckert, W., Fleury, M., Gilliotte, A., Gojak, D., Guzman, J. C., Kohler, D., Lizon, J. L., Longinotti, A., Lovis, C., Megevand, D., Pasquini, L., Reyes, J., Sivan, J. P., Sosnowska, D., Soto, R., Udry, S., van Kesteren, A., Weber, L., and Weilenmann, U. (2003). Setting New Standards with HARPS. *The Messenger*, 114:20–24.
- Pepe, F., Mayor, M., Galland, F., Naef, D., Queloz, D., Santos, N. C., Udry, S., and Burnet, M. (2002). The CORALIE survey for southern extra-solar planets VII. Two short-period Saturnian companions to <ASTROBJ>HD 108147</ASTROBJ>and <ASTROBJ>HD 168746</ASTROBJ> . , 388:632–638.
- Ricker, G. R., Winn, J. N., Vanderspek, R., Latham, D. W., Bakos, G.  . A., Bean, J. L., Berta-Thompson, Z. K., Brown, T. M., Buchhave, L., Butler, N. R., Butler, R. P., Chaplin, W. J., Charbonneau, D., Christensen-Dalsgaard, J., Clampin, M., Deming, D., Doty, J., De Lee, N., Dressing, C., Dunham, E. W., Endl, M., Fressin, F., Ge, J., Henning, T., Holman, M. J., Howard, A. W., Ida, S., Jenkins, J., Jernigan, G., Johnson, J. A., Kaltenegger, L., Kawai, N., Kjeldsen, H., Laughlin, G., Levine, A. M., Lin, D., Lissauer, J. J., MacQueen, P., Marcy, G., McCullough, P. R., Morton, T. D., Narita, N., Paegert, M.,

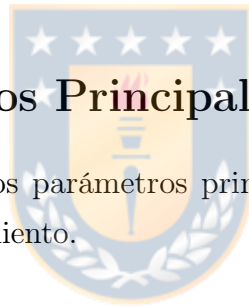
- Palle, E., Pepe, F., Pepper, J., Quirrenbach, A., Rinehart, S. A., Sasselov, D., Sato, B., Seager, S., Sozzetti, A., Stassun, K. G., Sullivan, P., Szentgyorgyi, A., Torres, G., Udry, S., and Villaseñor, J. (2014). Transiting Exoplanet Survey Satellite (TESS). In Oschmann, Jacobus M., J., Clampin, M., Fazio, G. G., and MacEwen, H. A., editors, *Space Telescopes and Instrumentation 2014: Optical, Infrared, and Millimeter Wave*, volume 9143 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, page 914320.
- Scargle, J. D. (1982). Studies in astronomical time series analysis. II. Statistical aspects of spectral analysis of unevenly spaced data. , 263:835–853.
- Shallue, C. J. and Vanderburg, A. (2018). Identifying exoplanets with deep learning: A five-planet resonant chain around kepler-80 and an eighth planet around kepler-90. *The Astronomical Journal*, 155(2):94.
- Shin, Y., Yoon, S., Seo, Y., Jin, H., and Seon, J. (2015). Radiation effect for a cubesat in slow transition from the earth to the moon. *Advances in Space Research*, 31.
- Thompson, S. E., Mullally, F., Coughlin, J., Christiansen, J. L., Henze, C. E., Haas, M. R., and Burke, C. J. (2015). A Machine Learning Technique to Identify Transit Shaped Signals. , 812(1):46.
- VanderPlas, J., Connolly, A. J., Ivezić, Z., and Gray, A. (2012). Introduction to astroML: Machine learning for astrophysics. In *Proceedings of Conference on Intelligent Data Understanding (CIDU)*, pages 47–54.
- Wright, J. and Gaudi, B. (2012). Exoplanet detection methods. *Planets, Stars and Stellar Systems. Volume 3: Solar and Stellar Planetary Systems*.
- Zechmeister, M. and Kürster, M. (2009). The generalised Lomb-Scargle periodogram. A new formalism for the floating-mean and Keplerian periodograms. , 496(2):577–584.

# Apéndice A

## Tablas y Códigos utilizados en este trabajo

### A1. Tabla Parámetros Principales

En esta sección hemos incluido los parámetros principales de los objetos que componen nuestro set de entrenamiento.



Elegir este grupo significa varias ventajas, entre ellas:

- Todas las estrellas son estrellas de tipo M. Las estrellas de tipo M son menos masivas y luminosas, por la cual las perturbaciones causadas por posibles exoplanetas son más notorias.
- Existen abundantes observaciones realizadas con diversos instrumentos para gran parte de las estrellas en este grupo, lo que significa una abundancia de material para el entrenamiento.
- La mayoría de los planetas en la muestra de entrenamiento corresponde a planetas en órbitas cercanas a la estrella, aparte gran parte de los planetas escogidos tienen masas aparentes similares a la de nuestro planeta, lo cual nos permite considerarlos como mundos aparentemente rocosos.
- Planetas rocosos en órbitas cercanas a la estrella, son comúnmente considerados como posibles planetas habitables. Por supuesto, estos no son los únicos indicadores de sí un planeta es habitable, existen una multitud

de factores geológicos y atmosféricos, pero estos dos factores astronómicos (que la mayoría de planetas en nuestro set de entrenamiento cumplen) son vitales para la presencia de vida en la superficie de un planeta.

En síntesis, el grupo escogido ofrece una abundancia de información disponible como también representa a objetos de gran interés para la astronomía presente y futura.



Nombre de la Estrella	Nombre del Planeta	Periodo Orbital [días]	Semieje Mayor [UA]	Masa Mínima [ $M_{\oplus}$ ]	Metodo de Deteccion
GJ 1132	GJ 1132 b	1.6289	0.0153	1.66	Transit
GJ 1132	GJ 1132 c	8.929	0.0476	2.64	Radial Velocity
GJ 1214	GJ 1214 b	1.5804	0.0149	8.17	Transit
GJ 163	GJ 163 b	8.6312	0.06	9.8869	Radial Velocity
GJ 163	GJ 163 c	25.6305	0.1254	6.8	Radial Velocity
GJ 163	GJ 163 d	603.9511	1.0304	29.4	Radial Velocity
GJ 180	GJ 180 b	17.133	0.092	6.49	Radial Velocity
GJ 180	GJ 180 c	24.329	0.129	6.4	Radial Velocity
GJ 180	GJ 180 d	106.3	0.309	7.56	Radial Velocity
GJ 3082	GJ 3082 b	11.95	0.079	8.2	Radial Velocity
GJ 3138	GJ 3138 b	1.2200	0.0197	1.78	Radial Velocity
GJ 3138	GJ 3138 c	5.974	0.057	4.18	Radial Velocity
GJ 3138	GJ 3138 d	257.8	0.698	10.5	Radial Velocity
GJ 317	GJ 317 b	695.66	1.151	557.1	Radial Velocity
GJ 3293	GJ 3293 b	30.5987	0.1433	23.54	Radial Velocity
GJ 3293	GJ 3293 c	122.6196	0.3617	21.09	Radial Velocity
GJ 3293	GJ 3293 d	48.1345	0.1939	7.6	Radial Velocity
GJ 3293	GJ 3293 e	13.2543	0.0820	3.28	Radial Velocity
GJ 3341	GJ 3341 b	14.207	0.0890	6.6	Radial Velocity
GJ 3470	GJ 3470 b	3.3366	0.0355	13.9	Radial Velocity
GJ 3634	GJ 3634 b	2.6456	0.0287	8.4	Radial Velocity
GJ 480	GJ 480 b	9.567	0.068	13.2	Radial Velocity
GJ 676 A	GJ 676 A d	3.6005	0.0413	4.4	Radial Velocity
GJ 676 A	GJ 676 A e	35.39	0.187	8.1	Radial Velocity
GJ 229	GJ 229 A b	526.115	0.898	8.478	Radial Velocity
GJ 229	GJ 229 A c	121.995	0.3389	7.268	Radial Velocity
GJ 433	GJ 433 b	7.3705	0.062	6.0429	Radial Velocity
GJ 433	GJ 433 d	36.059	0.1780	5.223	Radial Velocity
GJ 436	GJ 436 b	2.6438	0.0291	22.1	Radial Velocity
GJ 536	GJ 536 b	8.7076	0.0666	5.36	Radial Velocity
GJ 581	GJ 581 b	5.3686	0.0406	15.8	Radial Velocity
GJ 581	GJ 581 c	12.9140	0.0721	5.5	Radial Velocity
GJ 581	GJ 581 e	3.1489	0.0281	1.7	Radial Velocity
GJ 682	GJ 682 b	17.4779	0.08	4.4	Radial Velocity
GJ 682	GJ 682 c	57.32	0.1760	8.7	Radial Velocity
Gl 686	Gl 686 b	15.53	0.091	6.624	Radial Velocity
GJ 832	GJ 832 c	35.68	0.163	5.4	Radial Velocity
GJ 849	GJ 849 b	1915.0	-	286.0455	Radial Velocity
GJ 876	GJ 876 b	61.1166	0.2083	723.2235	Radial Velocity
GJ 876	GJ 876 c	30.0881	0.1295	226.9846	Radial Velocity
GJ 876	GJ 876 d	1.9377	0.0208	6.83	Radial Velocity
GJ 876	GJ 876 e	124.26	0.3343	14.6	Radial Velocity
HATS-6	HATS-6 b	3.3252	0.0362	101.3877	Transit
HD 110113	HD 110113 b	2.541	0.035	4.55	Transit
HD 110113	HD 110113 c	6.744	0.068	10.5	Radial Velocity
HD 137388	HD 137388 b	330.0	0.89	63.566	Radial Velocity
HD 208487	HD 208487 b	130.08	0.524	165.265	Radial Velocity
K2-122	K2-122 b	2.2193	0.0373	-	Transit
K2-89	K2-89 b	1.0960	0.0146	-	Transit
LHS 1140	LHS 1140 b	24.7369	0.0957	6.38	Transit
LHS 1140	LHS 1140 c	3.7779	0.0273	1.76	Transit
GJ 176	GJ 176 b	8.7748	0.066	7.9476	Radial Velocity
GJ 176	GJ 176 c	28.586	0.146	7.4072	Radial Velocity
Gl 514	Gl 514 b	140.43	5.2	5.2136	Radial Velocity
Gl 205	Gl 205 b	16.938	0.109	10.3	Radial Velocity
Gl 205	Gl 205 c	270.8	0.689	13.8	Radial Velocity

Cuadro A1.1: Parámetros Principales de las estrellas del grupo PC.



## A2. Principales códigos desarrollados para este trabajo

### A2.1. PeriodogramScript.py

PeriodogramScript.py es el código utilizado para generar los periodogramas a partir de las curvas de velocidad radial. Su funcionamiento es bastante simple, se le entrega la ubicación de las curvas junto con listados de todos los objetos de cada clase y el código genera una carpeta aparte en la que guarda todos los periodogramas generados.

### A2.2. Equalizer.py

Equalizer.py es un código ideado para la etapa del pre-procesado de los datos. A partir de los periodogramas genera los vectores de características que más tarde serán utilizados para clasificar.

Created on Mon Mar 8 16:24:16 2021

```
@author: pipe
"""
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import os

def NewRowMakerNoFAP(df, IntervalMax):
    data=df.head(IntervalMax)
    power= data['Periodogram_Value']
    powerlist=[]
    namelist=[]
    fulllist=[]
    columnlist=[]
    DataMax= len(power)
```

```

if DataMax < IntervalMax:
    StepN=1
    print('Periodogram Too Short, Max numbers=',DataMax)
    for i in range(0,IntervalMax):
        if StepN<DataMax:
            smoldf=data.iloc[i]
            power= smoldf['Periodogram_Value']
            pwr= power
            pwrInterval= str(i+1)+'th Peak'
            powerlist.append(pwr)
            namelist.append(pwrInterval)
            fulllist.append(pwr)
            columnlist.append(pwrInterval)
            StepN=StepN+1
        else:
            pwr= -99
            pwrInterval= str(i+1)+'th Peak'
            powerlist.append(pwr)
            namelist.append(pwrInterval)
            fulllist.append(pwr)
            columnlist.append(pwrInterval)
    else:
        for i in range(0,IntervalMax):
            smoldf=data.iloc[i]
            power= smoldf['Periodogram_Value']

            pwr= power
            pwrInterval= str(i+1)+'th Peak'
            powerlist.append(pwr)
            namelist.append(pwrInterval)
            fulllist.append(pwr)
            columnlist.append(pwrInterval)
    return powerlist,namelist,fulllist,columnlist

min1=np.log10(0.0000000000000001)

```

```
max1=np.log10(2000)
min2=1/2000
max2=10
def Main(Interval,Path):
    print('Number of Peaks = ',Interval)
    # Step=(max1-min1)/Interval
    data=np.linspace(min1,max1,Interval)
    # data=np.linspace(min1,max1,Interval*2)

    STARID= []
    LABEL= []
    for i in starfiles:
        StarID=str(i)

        starname= i[:-4]

        if starname[0]=='P':
            starlabel = 0
        elif starname[0] == 'N':
            starlabel = 1
        else:
            starlabel=3

        STARID.append(starname)
        LABEL.append(starlabel)
        df= pd.read_csv(Path+i)
        df=df.set_index('Periodogram_Value_for_Sorting')
        df= df.sort_index(ascending=False)
        print('working with file: ',i)
        a,b,c,d=NewRowMakerNoFAP(df,Interval)
        print(len(c),len(data))
        data=np.vstack((data,c))
    # print(df)
    print(len(data))
```

```

newdf= pd.DataFrame(data=data, columns=d)
newdf= newdf.loc[1:]
newdf.insert(0,'STARID',STARID)

newdf.insert(1,'LABEL',LABEL)
newdf.to_csv(Path+'Data/Data_Npeaks='+str(Interval)+'.csv',index=False)

#%

allstars= pd.read_csv('plist.csv')
starfiles=allstars['StarID']

Path= '/home/pipe/Clases/Tesis/Periodogramas/Corrected_Try/AllPer/'
os.system('mkdir '+Path+'Data/')
Intervals= [15,20,25,30,35,40,45,50,60,70,80,90,95,100,120,140,150,200,400,500,600,700]
for i in Intervals:
    Main(i,Path)

# %%
error=[0.001,0.005,0.01,0.05,0.1,0.5,1,2,3,5,10,100]
for j in error:
    Path= '/home/pipe/Clases/Tesis/Periodogramas/Corrected_Try/Test_Noise/Gaussian_sig'
    os.system('mkdir '+Path+'Data/')
# Main(2000,1)
for i in Intervals:
    Main(i,Path)

```

### A2.3. Classifier.py

`Classifier.py` es el código principal de este trabajo. Su función principal es la de generar un modelo de machine learning (pudiendo ser este un bosque aleatorio o una máquina de vector de soporte) entrenarlo y entregar sus resultados en forma de matriz de confusión. Pero esto no es la única función, pudiendo ser utilizado

para realizar múltiples entrenamientos con distintos parámetros y modelos a fin de comparar su desempeño.

Created on Thu Mar 25 19:00:52 2021

```
@author: pipe
"""
import pandas as pd
import pylab as pl
import numpy as np
import matplotlib.pyplot as plt
from sklearn import preprocessing
from imblearn.over_sampling import SMOTE
from sklearn.preprocessing import normalize, StandardScaler
from scipy import ndimage
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_validate
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import f1_score, accuracy_score, confusion_matrix, precision_score
import seaborn as sns
import os
from sklearn.model_selection import cross_val_score
from keras.wrappers.scikit_learn import KerasClassifier
from keras.models import Sequential
from keras.layers import Dense
from scipy import stats

#Importamos los paquetes.
def Preprocessing(df_location):
    # print("Working now with: ",df_location)
    df = pd.read_csv(df_location)

    X = df.loc[:, (df.columns != 'Unnamed') & (df.columns != 'LABEL') & (df.columns != 'p_el') & (df.columns != 'p_cs')].values
```

```
y = df.loc[:, (df.columns == 'LABEL')].values.flatten()

un_y, counts = np.unique(y, return_counts = True)
# print(X)
X_train=np.nan_to_num(X)

# pl.clf()
# pl.bar(un_y, counts)
# pl.show()

X_train = normalized = normalize(X_train)

X_train = filtered = ndimage.filters.gaussian_filter(X_train, sigma=10)
# print(X_train)
#Convertimos todo nan a -99, las tablas en si no deberian tener nan, pero es mejor t

#Feature scaling
std_scaler = StandardScaler()
X_train = scaled = std_scaler.fit_transform(X_train)
# print('AFTER FEATURE SCALING')
# print(X_train)
#Dimentionality reduction
# pca = PCA()
# X_train = pca.fit_transform(X_train)
# total=sum(pca.explained_variance_)
# k=0
# current_variance=0
# while current_variance/total < 0.9:
#     current_variance += pca.explained_variance_[k]
#     k=k+1
# print(k)
k=1
# print(X_train.shape)
```

```
#Apply PCA with n_componenets
# pca = PCA(n_components=k)
# X_train = pca.fit_transform(X_train)
# plt.figure()
# plt.plot(np.cumsum(pca.explained_variance_ratio_))
# plt.xlabel('Number of Components')
# plt.ylabel('Variance (%)') #for each component
# plt.title('Exoplanet Dataset Explained Variance')
# plt.show()

#Resampling
# print("Before OverSampling, counts of label '1': {}".format(sum(y==1)))
# print("Before OverSampling, counts of label '0': {} \n".format(sum(y==0)))
sm = SMOTE()
X_train_res, y_train_res = sm.fit_resample(X_train, y.ravel())
# print("After OverSampling, counts of label '1': {}".format(sum(y_train_res==1)))
# print("After OverSampling, counts of label '0': {}".format(sum(y_train_res==0)))
return X_train_res, y_train_res,k

def Preprocessing2(df_location,N_components):
    # print("Working now with: ",df_location)
    df = pd.read_csv(df_location)

    X = df.loc[:, (df.columns != 'Unnamed') & (df.columns != 'LABEL') & (df.columns != 'p_el') & (df.columns != 'p_cs')].values
    y = df.loc[:, (df.columns == 'LABEL')].values.flatten()

    un_y, counts = np.unique(y, return_counts = True)

    X_train=np.nan_to_num(X)
    # pl.clf()
    # pl.bar(un_y, counts)
    # pl.show()

    X_train = normalized = normalize(X_train)
```

```

X_train = filtered = ndimage.filters.gaussian_filter(X_train, sigma=10)
#Feature scaling
std_scaler = StandardScaler()
X_train = scaled = std_scaler.fit_transform(X_train)
#Dimentionality reduction
# pca = PCA()
# X_train = pca.fit_transform(X_train)
# total=sum(pca.explained_variance_)

#Apply PCA with n_componenets
# pca = PCA(n_components=N_components)
# X_train = pca.fit_transform(X_train)
# plt.figure()
# plt.plot(np.cumsum(pca.explained_variance_ratio_))
# plt.xlabel('Number of Components')
# plt.ylabel('Variance (%)') #for each component
# plt.title('Exoplanet Dataset Explained Variance')
# plt.show()

#Resampling
# print("Before OverSampling, counts of label '1': {}".format(sum(y==1)))
# print("Before OverSampling, counts of label '0': {} \n".format(sum(y==0)))
# sm = SMOTE()
# X_train_res, y_train_res = sm.fit_resample(X_train, y.ravel())
# print("After OverSampling, counts of label '1': {}".format(sum(y_train_res==1)))
# print("After OverSampling, counts of label '0': {}".format(sum(y_train_res==0)))
return X_train, y

# THE FLAG ALLOWS THE USER TO CHOOSE BETWEEN SYMMETRIC AND ASYMMETRIC CONFIDENCE INTERVALS
def calc_result(value_array, flag):

    value_median = stats.scoreatpercentile(value_array, 50)
    value_low = stats.scoreatpercentile(value_array, 16)
    value_high = stats.scoreatpercentile(value_array, 84)

```



```
delta_value = 0.5*(value_high - value_low)
delta_value_low = (value_median - value_low)
delta_value_high = (value_high - value_median)

result_value = np.array([value_median, delta_value_low, delta_value_high])
result_value_symmetric = np.array([value_median, delta_value])

if flag=="symmetric":
    return result_value_symmetric
if flag=="asymmetric":
    return result_value

def RFClassifier(X_send,Y_send,Interval):
    X_train, X_test, y_train, y_test = train_test_split (X_send, Y_send, test_size=Interval)
    from sklearn.ensemble import RandomForestClassifier
    clf = RandomForestClassifier(n_estimators=1000)
    clf.fit (X_train, y_train)

    y_pred = clf.predict (X_test)

    #Metrics
    accuracy= accuracy_score (y_test, y_pred)
    precision= precision_score (y_test, y_pred,average = 'macro')
    recall= recall_score (y_test, y_pred, average = 'macro')
    f1= f1_score (y_test, y_pred, average = 'macro')
    # print("accuracy: ", accuracy_score (y_test, y_pred))
    # print( "precision: ", precision_score (y_test, y_pred,average = 'macro'))
    # print ("recall: ", recall_score (y_test, y_pred, average = 'macro'))
    # print( "f1: ", f1_score (y_test, y_pred, average = 'macro'))
    # cm = confusion_matrix (y_test, y_pred) # columns: predicted, rows: true
    # print (cm)

    # Confusion Matrix
```

```

# plt.figure(figsize=(13,10))
# plt.subplot(221)
# sns.heatmap(confusion_matrix(y_test,y_pred),annot=True,cmap="viridis",fmt = "d",
# plt.title("Confusion Matrix Random Forest Classifier, I="+Interval+',fontsize=15)
return (accuracy,precision,recall,f1)

def SVMClassifier(X,Y,Interval):
    X_train, X_test, y_train, y_test = train_test_split (X, Y, test_size=0.4)
    from sklearn.pipeline import make_pipeline
    from sklearn.svm import SVC
    clf = make_pipeline(StandardScaler(), SVC(gamma='auto'))
    clf.fit(X_train,y_train)
    y_pred = clf.predict (X_test)

#Metrics
accuracy= accuracy_score (y_test, y_pred)
precision= precision_score (y_test, y_pred,average = 'macro')
recall= recall_score (y_test, y_pred, average = 'macro')
f1= f1_score (y_test, y_pred, average = 'macro')
# print("accuracy: ", accuracy_score (y_test, y_pred))
# print( "precision: ", precision_score (y_test, y_pred,average = 'macro'))
# print ("recall: ", recall_score (y_test, y_pred, average = 'macro'))
# print( "f1: ", f1_score (y_test, y_pred, average = 'macro'))
# cm = confusion_matrix (y_test, y_pred) # columns: predicted, rows: true
# print (cm)

# #Confusion Matrix
# plt.figure(figsize=(13,10))
# plt.subplot(221)
# sns.heatmap(confusion_matrix(y_test,y_pred),annot=True,cmap="viridis",fmt = "d",15)
# plt.title("Confusion Matrix Suport Vector Machine, I="+Interval+',fontsize=15)
return (accuracy,precision,recall,f1)

#%%
Intervals= [15,20,25,30,35,40,45,50,60,70,80,90,95,100,120,140,150,200,400,500,600,700]

```

```
path='/home/pipe/Clases/Tesis/Periodogramas/Corrected_Try/AllPer/Data/'
Results_path= path+'Results/'
os.system('mkdir '+Results_path)

###
Data_path= path
RFf1_scores= []
RFf1_scores_up= []
RFf1_scores_down=[]

RFprecision_scores=[]
RFprecision_scores_up=[]
RFprecision_scores_down=[]

RFrecall_scores=[]
RFrecall_scores_up=[]
RFrecall_scores_down=[]

RFacuracies=[]
RFacuracies_up=[]
RFacuracies_down=[]

SVMf1_scores= []
SVMf1_scores_up= []
SVMf1_scores_down=[]

SVMprecision_scores=[]
SVMprecision_scores_up=[]
SVMprecision_scores_down=[]

SVMrecall_scores=[]
SVMrecall_scores_up=[]
SVMrecall_scores_down=[]

SVMacuracies=[]
```



```
SVMacuracies_up=[]
SVMacuracies_down=[]

for i in Intervals:
    Interval= str(i)
    X,y,k = Preprocessing(Data_path+"Data_Npeaks="+Interval+".csv")

        #####RANDOM FOREST
    ACC= []
    PRE=[]
    REC= []
    Fe=[]
    for f in range(0,100):
        accuracy,precision,recall,f1=RFClassifier(X,y,Interval)
        ACC.append(accuracy)
        PRE.append(precision)
        REC.append(recall)
        Fe.append(f1)
    accuracy,accuracy_up,accuracy_down= calc_result(ACC,"asymmetric")
    RFacuracies.append(accuracy)
    RFacuracies_up.append(accuracy_up)
    RFacuracies_down.append(accuracy_down)

    precision,precision_up,precision_down= calc_result(PRE,"asymmetric")
    RFprecision_scores.append(precision)
    RFprecision_scores_up.append(precision_up)
    RFprecision_scores_down.append(precision_down)

    recall,recall_up,recall_down= calc_result(REC,"asymmetric")
    RFrecall_scores.append(recall)
    RFrecall_scores_up.append(recall_up)
    RFrecall_scores_down.append(recall_down)

    f1,f1_up,f1_down= calc_result(Fe,"asymmetric")
    RFf1_scores.append(f1)
```

```

RFf1_scores_down.append(f1_down)
RFf1_scores_up.append(f1_up)
#####SVM
ACC= []
PRE=[]
REC= []
Fe=[]
for f in range(0,100):
    accuracy,precision,recall,f1=SVMClassifier(X,y,Interval)
    ACC.append(accuracy)
    PRE.append(precision)
    REC.append(recall)
    Fe.append(f1)

accuracy,accuracy_up,accuracy_down= calc_result(ACC,"asymmetric")
SVMacuracies.append(accuracy)
SVMacuracies_up.append(accuracy_up)
SVMacuracies_down.append(accuracy_down)

precision,precision_up,precision_down= calc_result(PRE,"asymmetric")
SVMprecision_scores.append(precision)
SVMprecision_scores_up.append(precision_up)
SVMprecision_scores_down.append(precision_down)

recall,recall_up,recall_down= calc_result(REC,"asymmetric")
SVMrecall_scores.append(recall)
SVMrecall_scores_up.append(recall_up)
SVMrecall_scores_down.append(recall_down)

f1,f1_up,f1_down= calc_result(Fe,"asymmetric")
SVMf1_scores.append(f1)
SVMf1_scores_down.append(f1_down)
SVMf1_scores_up.append(f1_up)

dataRF=[Intervals,RFf1_scores,RFprecision_scores,RFacuracies,RFrecall_scores,RF

```

```
dataRF=np.transpose(dataRF)
```

```
dataSVM=[Intervals,SVMf1_scores,SVMprecision_scores,SVMacuracies,SVMrecall_scores,SVM
```

```
dataSVM=np.transpose(dataSVM)
```

```
d=['Number of Peaks','F1 score',"Precision","Accuracy","Recall",'F1 score UP',"Precis
```

```
RFdf= pd.DataFrame(data=dataRF, columns=d)
```

```
RFdf.to_csv(Results_path+'RF0.csv',index=False)
```

```
SVMdf= pd.DataFrame(data=dataSVM, columns=d)
```

```
SVMdf.to_csv(Results_path+'SVM0.csv',index=False)
```

```
print('FINISH')
```

