VNiVERSiDAD D SALAMANCA

CAMPUS DE EXCELENCIA INTERNACIONAL

UNIVERSIDAD DE SALAMANCA

Departamento de Estadística

Máster en Análisis Avanzado de Datos Multivariantes y Big Data

Trabajo Fin de Máster

Modelos de clasificación para datos astronómicos

Classification Models for Astronomical Data

Autora: Florencia Anabella Teppa Pannia

Tutor: Dr. José Luis Vicente Villardón

2022

Resumen

La aplicación de métodos de aprendizaje automático supervisado a problemas científicos ha alcanzado su auge en los últimos años como herramienta fundamental para la exploración y minería de grandes bases de datos. En particular, en el campo de la Astronomía, un tema de estudio frecuentemente abordado es el entrenamiento de modelos para la clasificación de objetos celestes a partir de imágenes y/o características físicas observables.

El objetivo general de este trabajo es investigar modelos supervisados de clasificación binaria para resolver el problema de la distinción de objetos puntuales dentro de las clases *galaxia* y *estrella*. Los objetivos particulares se detallan a continuación:

1. Presentar el marco teórico de los modelos supervisados de clasificación, con el fin de familiarizar las ventajas y desventajas que presenta cada uno, así como adquirir un dominio de las herramientas numéricas para su aplicación. En este marco, se definen también las métricas adecuadas para cuantificar y comparar las capacidades de predicción de cada modelo.

2. Entrenar los modelos presentados utilizando el catálogo astronómico ALHAMBRA, compuesto por un total de 23 filtros fotométricos, que recoge información de observaciones de más de 6×10^4 objetos celestes. Comparar las predicciones de clasificación de cada modelo para este ejemplo concreto.

Asimismo, el trabajo es llevado a cabo según la siguiente metodología:

- el marco teórico para presentar los modelos es recogido de bibliografía específica, siguiendo los lineamientos del modulo 5 (*Machine Learning*) de este máster;
- los datos utilizados son de acceso público y se presentan mediante un análisis exploratorio inicial;
- los modelos son entrenados a partir de algoritmos disponibles en librerías de `R` y los códigos se presentan detallados para la reproducibilidad de los resultados obtenidos.

Como resultado general de nuestro trabajo, encontramos que todos los modelos estudiados arrojan buenos ajustes (regresión logística, *support vector machines*, redes neuronales y árboles de decisión, entre otros), con errores de predicción bajos al ser evaluados con datos de validación. Valorando la complejidad de los modelos y aplicando el principio de simplicidad, el modelo de regresión logística resulta preferido por su buena capacidad de predicción y la simpleza en implementación e interpretación.

Los resultados obtenidos en este trabajo para la clasificación de objetos del catálogo ALHAMBRA son originales, y los modelos entrenados resultan comparables a otros estudiados con catálogos astronómicos de las mismas características.

Palabras clave: aprendizaje automático; modelos de clasificación binaria; clasificación estrella-galaxia.

Summary

Applications of supervised machine learning techniques to scientific problems have increased in the last years as an extremely powerful tool for exploring and managing big data. Particularly, in the field of Astronomy, an important application is the study of classification models to distinguish between point-like sources from images and/or observable features.

The main objective of this work is the training of binary classification models with astronomical data to deal with the problem of classifying point-like sources into *galaxy* and *star* targets. Particular goals are:

1. To present the theoretical framework for classification models, highlighting their advantages and disadvantages, and providing general criteria to use them in different situations together with the corresponding numerical tools for applications. In this context, we will also define appropriate metrics to compare models and evaluate their power of prediction.

2. To train the presented models using astronomical data from ALHAMBRA, a photometric multi-filter survey of approximately 6×10^4 celestial objects. We will compare predictions from different classifiers, as well as their efficiency and complexity.

The objectives of this work will be accomplished by means of the following methodology:

- The theoretical framework to present the models is taken from the bibliography suggested in the Master as well as specific scientific publications on the topic.

- The data, which is publicly available, will be first statistically described and prepared for training.

- All models will be trained using specific libraries in `R` and codes will be available for the reproducibility of the results.

As our main result, we find that all the studied models for binary classification (logistic regression, support vector machines, neural networks, and decision tress, among others) are in agreement with good predictions for the star/galaxy problem when validation data is tested. Taking into account the complexity of the studied models and the principle of simplicity, we find that the logistic regression is preferred due to its excellent capacity of prediction and the simplicity of its implementation and interpretation.

All the results obtained in this work for the star/galaxy classification problem using ALHAMBRA data are original, and comparable with those presented in the literature for catalogs with similar properties.

Keywords: Machine learning; models of binary classification; star-galaxy classification.

Índice general

Capítulo 1

Introducción

La Astronomía, como ciencia observacional, ha sido protagonista en las últimas décadas de un crecimiento exponencial en materia de datos y acceso a información. El avance tecnológico ha ido impulsando la creación de proyectos de observación ambiciosos en cantidad, variedad y calidad de observaciones, entre las que destacan: la espectroscopía y la fotometría, galácticas y extragalácticas (e.g., *Sloan Digital Sky Survey* (SDSS)[1], *Dark Energy Spectroscopic Instrument* (DESI)[2], *Gaia-ESO*[3]), mapas volumétricos de galaxias (e.g., *Kilo-Degree Survey* (KiDS)[4], *Dark Energy Survey* (DES)[5]), imágenes espaciales (e.g., *Hubble Space Telescope* (HST)[6], *James Webb Space Telescope* (JWST)[7]), interferometría (e.g., *Atacama Large Millimeter/submillimeter Array* (ALMA)[8], *Very Large Array* (VLA)[9]), rayos X y Gamma (e.g., *Chandra X-Ray Observatory*[10], *Fermi Gamma-ray Space Telescope*[11]), y ondas gravitacionales (e.g., *Lase Interferometer Gravitational-Wave Observatory* (LIGO)[12], Virgo[13], *KAGRA Observatory* [14]), entre otras. Este escenario conlleva naturalmente a la exploración de técnicas estadísticas dentro del paradigma de *Big Data*, capaces de abordar el tratamiento de grandes volúmenes de datos, de diferente variedad, y generados y analizados a gran velocidad.

La aplicación de los métodos de aprendizaje automático o *Machine Learning* (ML) a problemas astrofísicos ha alcanzado su auge en los últimos años como herramienta fundamental para la exploración y minería de grandes bases de datos [1, 2, 3, 4]. Entre las diversas aplicaciones, podemos destacar: la medición de parámetros estelares, la identificación y clasificación de objetos celestes, problemas de dinámica de asteroides y cuerpos pequeños, y abordaje de problemas cosmológicos y estimación de *redshifts*, entre muchas otras más. Tal es así, que incluso existe una librería específica en `Python` que recoge sus principales aplicaciones en el

[1] https://www.sdss.org/
[2] https://www.desi.lbl.gov/
[3] https://www.gaia-eso.eu/
[4] https://kids.strw.leidenuniv.nl
[5] https://www.darkenergysurvey.org/
[6] https://www.nasa.gov/mission_pages/hubble/main/index.html
[7] https://www.jwst.nasa.gov/
[8] https://www.almaobservatory.org/en/home/
[9] https://public.nrao.edu/telescopes/vla/
[10] https://chandra.harvard.edu/
[11] https://fermi.gsfc.nasa.gov/
[12] https://www.ligo.caltech.edu/
[13] https://www.virgo-gw.eu/#home
[14] https://gwcenter.icrr.u-tokyo.ac.jp/en/

campo de la Astronomía: `astroML` [5][15].

En términos generales, el aprendizaje automático puede resumirse como el campo de estudio de los mecanismos que permiten a los ordenadores *aprender* de los datos sin estar específicamente programados para ello. Entre los innumerables ejemplos de aplicaciones de ML que se desarrollan en la actualidad, podemos mencionar: la clasificación de correos como *spam* o *no spam*, la utilización general de buscadores en la web, la biometría (reconocimiento de caras, voces, huellas dactilares, etc.), y la detección de noticias falsas a través de plataformas de redes sociales, entre otras. Como definición más formal, se dice que: un ordenador aprende de la experiencia E con respecto a una serie de tareas T y una medida del rendimiento P, si su rendimiento en las tareas T, medido mediante P, mejora con la experiencia E. Los algoritmos de aprendizaje automático pueden agruparse entonces en tres categorías generales:

1. **aprendizaje supervisado**, en donde la experiencia E se aprende de un conjunto de datos llamado de entrenamiento (e.g., regresión simple y múltiple, clasificación);

2. **aprendizaje no supervisado**, en donde los métodos infieren información latente o patrones desconocidos *a priori* (e.g., análisis de componentes principales para reducción de la dimensionalidad, visualización como comprensión de la estructura de los datos, conglomerados o *Clustering*); y

3. otros **aprendizajes avanzados**, como el aprendizaje por refuerzo y los sistemas de recomendación.

En este trabajo nos centraremos en el estudio de modelos de clasificación binaria aplicados a catálogos de objetos astronómicos. En particular, nos interesa abordar el problema de la clasificación estrella/galaxia. El problema tiene especial interés en el contexto de catálogos fotométricos de objetos puntuales (es decir, sin estructura extendida observable). A diferencia de los espectros, a partir de los cuales pueden identificarse familias de objetos según las líneas de emisión/absorción que las caracteriza, la clasificación de imágenes puntuales representa un desafío mayor y es uno de los problemas actualmente abordados con interés para la construcción de catálogos astronómicos.

Los llamados cuasi-espectros, por su parte, son observaciones fotométricas multi-banda que pretenden simular la distribución espectral de la luz de un objeto a partir de una colección secuencial de observaciones en diferentes filtros dentro de los rangos visible ($400\,\text{nm} < \lambda < 750\,\text{nm}$) e infrarrojo cercano ($750\,\text{nm} < \lambda < 2{,}5\,\mu\text{m}$). El Observatorio Astrofísico de Javalambre[16] ha sido pionero en este tipo de observaciones a través de sus proyectos ALHAMBRA[17], J-PLUS[18] y J-PAS[19], con datos de acceso acceso público a través del portal del Centro de Estudios de Física del Cosmos de Aragón (CEFCA)[20]. Otro proyecto independiente de cuasi-espectros es también el *Physics of the Accelerating Universe* (PAU).[21] Este tipo de observaciones ofrece la posibilidad de abordar el problema de la clasificación de objetos a partir de métodos de ML supervisados que tomen como atributos o *inputs* las magnitudes observadas en los diferentes filtros. La ventaja de utilizar directamente las magnitudes como

[15] https://www.astroml.org/
[16] http://oajweb.cefca.es/home
[17] http://www.alhambrasurvey.com/
[18] http://www.j-plus.es/
[19] http://www.j-pas.org/
[20] http://archive.cefca.es/catalogues
[21] https://pausurvey.org/

atributos nos permite elaborar clasificaciones sin suponer modelos físicos sobre los objetos (como sucede en el caso de las líneas espectrales, en donde la caracterización se basa en conocimientos previos de características físicas de esos objetos, como temperatura, metalicidad, edad, etc.).

El objetivo general de este trabajo es estudiar modelos supervisados de clasificación a partir de su aplicación en datos astronómicos. En particular, estamos interesados en estudiar el problema de la clasificación estrellas/galaxias utilizando el catálogo ALHAMBRA. Los objetivos particulares que abordaremos son los siguientes:

1. Presentar el marco teórico de los modelos supervisados de clasificación (regresión logística, redes neuronales, *support vector machine*, y árboles de decisión, entre otros), con el fin de adquirir un dominio de las herramientas y las ventajas y desventajas que cada uno ofrece. En este marco, se definirán también las métricas adecuadas para cuantificar y comparar las capacidades predictivas de los modelos.

2. Entrenar los modelos utilizando el el catálogo ALHAMBRA [6], formado por un total de 20 filtros ópticos y 3 infrarrojos, que recoge información completa de $\sim 6 \times 10^4$ objetos. Comparar las predicciones de clasificación en cada modelo y analizar el uso de cada clasificador para este ejemplo concreto.

Asimismo, el trabajo será llevado a cabo siguiendo la siguiente metodología:

- El marco teórico para presentar los modelos será recogido de bibliografía específica, siguiendo los lineamientos del modulo 5 (Machine Learning) de este máster.

- Los datos utilizados, de acceso público, serán sometidos a un análisis exploratorio inicial, incluyendo una estadística descriptiva y el acondicionamiento de variables.

- Los modelos serán entrenados a partir de los algoritmos disponibles en librerías de `R`. Para facilitar la reproducibilidad de los resultados obtenidos, las líneas de código básicas para cada modelo serán detalladas en el Apéndice A.

Como resultado de nuestro estudio, encontramos que todos los modelos trabajados arrojan buenos resultados de ajuste, con errores de predicción bajos. Los resultados obtenidos en este trabajo para la clasificación de objetos del catálogo ALHAMBRA son originales, y los modelos entrenados resultan comparables a otros estudiados con catálogos astronómicos de las mismas características [7, 8, 9].

Capítulo 2

Algoritmos de Aprendizaje Supervisado

En términos generales, los métodos de aprendizaje supervisado trabajan con un conjunto de variables independientes X_j, también llamadas *features* o *inputs*, y una variable respuesta Y, de la que se conoce su valor correcto y a partir de la cual los modelos *aprenden*. Según el tipo de variable respuesta con la que se esté trabajando, los algoritmos supervisados se agrupan en:

- **regresión,** en el caso de respuestas numéricas (continuas o discretas); y
- **clasificación,** si las variables respuesta son categóricas o nominales.

En particular, se habla de *modelos de clasificación binaria* cuando la respuesta presenta sólo dos clases. En este caso, la respuesta también puede utilizarse para estimar la probabilidad de ocurrencia (o no ocurrencia) de un evento.

En ambos casos, regresión y clasificación, el problema general del aprendizaje supervisado consiste en diseñar un algoritmo para encontrar una función que permita predecir la variable respuesta a partir de las variables independientes. Esto es, encontrar una función h tal que

$$\hat{y} = h_\theta(x_1, \ldots, x_p)\,, \tag{2.1}$$

siendo $\hat{y}$ la predicción del valor correcto y de la variable Y para un dado individuo, dados los valores x_j de los atributos X_j correspondientes. La función h se denomina, en este contexto, *hipótesis* y dependerá de una serie de parámetros θ_k que deberán hallarse durante el proceso de aprendizaje. En lo que sigue, adoptaremos la siguiente nomenclatura única:

- n: número de ejemplos (individuos) en el conjunto de datos (tamaño de la muestra);
- $X_1, \ldots, X_p$: variables predictoras o atributos (*features* o *inputs*);
- Y: variable respuesta (*output*);
- $(x_1, \ldots, x_p, y)$: valores correspondientes a un ejemplo genérico de la muestra;
- $\left(x_1^{(i)}, \ldots, x_p^{(i)}, y^{(i)}\right)$: valores correspondientes al i-ésimo ejemplo.

El proceso de aprendizaje se denomina comúnmente *entrenamiento del modelo* (análogo al término *ajuste del modelo* en el contexto de regresión tradicional) y se basa en dividir el conjunto total de datos en dos subconjuntos:

1. un conjunto de aprendizaje o *training set* (T), que se utiliza para entrenar el algoritmo con los valores reales de la variable respuesta (aproximadamente, el 70 % del conjunto total); y

2. un conjunto de validación (V), que se utiliza para comprobar el algoritmo obtenido comparando las predicciones del modelo con los valores reales (en general, el 30 % restante de los datos).

Los datos de entrenamiento son utilizados para que el modelo *aprenda* la función h_θ. Esto es, hallar los valores de los parámetros θ_k de forma tal que $\hat{y}$ sea lo más cercano posible a y en los ejemplos (x, y) de los que se dispone el valor real. Definiremos más adelante la forma de medir esta cercanía a través de la llamada *función de coste*. El proceso es luego validado con el conjunto restante de datos. Una vez obtenido un rendimiento satisfactorio, el modelo puede ser empleado para predecir la variable respuesta en nuevos datos, para los cuales ésta se desconoce.

2.1. Regresión

Como mencionamos anteriormente, uno de los métodos básicos de ML, utilizados en innumerables campos de aplicación cuando la variable respuesta es numérica, es el de la *regresión*, simple ó múltiple. En el primer caso sólo se cuenta con una única variable independiente X, mientras que en el caso múltiple se trabaja con un conjunto de atributos. Trabajaremos bajo el supuesto de que las variables predictoras son de tipo continuo, aunque la extensión al caso de predictores cualitativos puede también implementarse a través de las llamadas variables ficticias o *dummy* (ver, por ejemplo, [10]).

Los modelos de *regresión lineal*, como su nombre lo indica, se basan en hipótesis definidas como una función lineal de p atributos X_j, esto es,

$$h_\theta(x_1, \ldots, x_p) = \theta_0 + \theta_1 x_1 + \ldots + \theta_p x_p \,, \tag{2.2}$$

con θ_j parámetros a determinar durante el entrenamiento del modelo.[1] Asimismo, la generalización para relaciones no lineales entre las variables predictoras y la variable respuesta puede implementarse, por ejemplo, a través de funciones polinómicas de la forma

$$h_\theta(x_1, x_2) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1 x_2 + \theta_4 x_1^2 + \theta_5 x_2^2 \,. \tag{2.3}$$

La forma de medir la discrepancia entre la hipótesis y los valores reales observados en la respuesta del conjunto T es a través de la llamada **función de coste**:

$$J(\theta_k) \equiv \frac{1}{2n} \sum_{i=1}^{n} \left[h_\theta \left(x_1^{(i)}, \ldots, x_p^{(i)} \right) - y^{(i)} \right]^2 = \frac{1}{2n} \sum_{i=1}^{n} \left(\hat{y}^{(i)} - y^{(i)} \right)^2 . \tag{2.4}$$

El objetivo del aprendizaje automático es entonces encontrar los parámetros θ_k que hagan mínima la función de coste. Existen varios métodos para llevar a cabo la optimización, entre los que destacamos:

[1] Notar que, por simplicidad de notación, tomamos $j = 0, 1, \ldots, p$ y $X_0 = 1$.

- el **método del descenso del gradiente**, que aplica un algoritmo iterativo de la forma

$$\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_j)\,, \quad j = 0, 1, \ldots, p\,, \tag{2.5}$$

  actualizando simultáneamente todos los parámetros hasta alcanzar la convergencia, a partir de valores iniciales $\theta_j^{(0)}$; el parámetros α se denomina *ritmo de aprendizaje* y mide el salto en cada paso del algoritmo; y

- el **método de las ecuaciones normales**, a partir del arreglo de los datos en la forma matricial

$$Y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}\,, \quad X = \begin{bmatrix} 1 & x_{11} & \ldots & x_{1p} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \ldots & x_{np} \end{bmatrix}\,,$$

  y utilizando el método de mínimos cuadrados cuadrados para hallar

$$\theta = [\theta_0, \ldots, \theta_p] = (X^t X)^{-1} X^t Y\,. \tag{2.6}$$

La ventajas de utilizar un método u otro dependen del tamaño muestral n y el número de atributos p, así como sus características particulares. Por ejemplo, trabajar con ecuaciones normales no requiere la elección del parámetro α ni el cómputo de reiteradas iteraciones, pero atributos altamente correlacionados pueden dar lugar a matrices $X^t X$ no invertibles. El método del gradiente, por su parte, requiere en general que los datos estén estandarizados para una buena convergencia, mientras que presenta una ventaja en el tiempo de cálculo si el número de atributos es grande.[2]

En los casos en los que el número de atributos es grande, puede tener lugar lo que se conoce como *sobre-ajuste* u *overfitting*. Este comportamiento consiste en un ajuste muy bueno del modelo para el conjunto de entrenamiento, pero malo cuando éste se valida con nuevos datos. Dos formas usuales de lidiar con este problema son:

1. la reducción de los atributos, mediante selección manual o automática; y
2. la **regularización**, que consiste en mantener la cantidad de atributos pero reducir la magnitud de los parámetros θ_j; esto se implementa con una penalización a la función de coste de la forma

$$J(\theta_j) \equiv \frac{1}{2n} \left[\sum_{i=1}^{n} \left(h_\theta(x_1^{(i)}, \ldots, x_p^{(i)}) - y^{(i)} \right)^2 + \lambda \sum_{j=1}^{p} \theta_j^2 \right]\,, \tag{2.7}$$

   en donde λ es el llamado *parámetro de regularización* y determina cuánto añaden los parámetros al coste.

Los conceptos estadísticos de **sesgo** y **varianza** pueden extenderse al caso de los modelos de aprendizaje supervisado. El sesgo, medida de la diferencia entre el valor estimado de una muestra respecto al valor real de la población completa, presenta en el contexto de ML una relación inversa con la *complejidad* del modelo. Esto es, si los modelos están infra-ajustados

[2] Ver, por ejemplo, https://dzone.com/articles/gradient-descent-vs-normal-equation-for-regression?ref=morioh.com&utm_source=morioh.com.

esperaremos sesgos mayores y errores que no disminuyen al aumentar el tamaño muestral (e.g., el caso de un ajuste lineal a problemas complejos, con comportamientos no lineales). La varianza, por su parte, recoge las diferencias entre distintas muestras de la población y, en el contexto de ML, se manifestará entre los conjuntos de entrenamiento y validación utilizados. Puede ser relacionada también con la *complejidad* de los modelos, esperándose varianzas grandes en los casos de sobre-ajuste (es decir, cuando el modelo se ajusta muy bien al conjunto de entrenamiento, pero no generaliza bien al conjunto de validación, incluso cuando se aumenta el tamaño muestral). Las estrategias que pueden seguirse para optimizar el modelo en presencia de sesgo o varianza altos son variadas. En el primer caso, pueden añadirse atributos adicionales, atributos polinómicos, o bien disminuir el parámetro de regularización λ. Para los casos con varianza alta, se puede trabajar con muestras mayores, con conjuntos más pequeños de atributos, o bien aumentando el valor de λ para penalizar más la función de coste.

2.2. Clasificación

Los métodos de clasificación son similares a los de regresión, con la diferencia de que la variable respuesta es ahora de tipo nominal. El problema consiste entonces de determinar qué valor de la variable categórica es más probable. Es decir, se asignan probabilidades a cada una de las categorías de la variable respuesta, y la predicción final se corresponde con el valor que presente la probabilidad más alta.

Presentaremos los lineamientos generales del caso más simple (y más utilizado) de clasificación: la *clasificación binaria*, en donde la variable respuesta posee sólo dos categorías. En el contexto de este trabajo, estaremos interesados en clasificar objetos astronómicos en las clases *estrella* o *galaxia*, aunque las aplicaciones de este tipo de algoritmos se extiende a numerosos campos (clasificación de correos electrónicos en *spam* o no), detección de transacciones bancarias fraudulentas, detección de noticias falsas, detección de tumores malignos, etc.).

El modelo más adecuado para tratar los problemas de clasificación binaria es el **modelo de regresión logística**. La variable respuesta es codificada generalmente en clase positiva ($\hat{y} = 1$) y clase negativa ($\hat{y} = 0$), mientras que la hipótesis se define como una función acotada en valores interpretables como probabilidades $p^{(i)}$ de estas clases, es decir, $0 \leq p^{(i)} \equiv h_\theta(x^{(i)}) \leq 1$. Se propone entonces una función hipótesis de la forma

$$h_\theta(x_1, \ldots, x_p) = g(\theta_0 + \theta_1 x_1 + \ldots + \theta_p x_p) = \frac{1}{1 + \exp\left[-(\theta_0 + \theta_1 x_1 + \ldots + \theta_p x_p)\right]}, \qquad (2.8)$$

en donde $g(z) \equiv 1/[1 + \exp(-z)]$ es la llamada *función logística* o *sigmoid* (Figura 2.1). De esta forma, la función hipótesis puede interpretarse como la probabilidad de respuesta buscada, es decir, $h_\theta(x_1, \ldots, x_p) = P(\hat{y} = 1 | x_j; \theta_j)$, con predicción de respuesta positiva cuando $h_\theta \geq 0$ ó, equivalentemente, $(\theta_0 + \theta_1 x_1 + \ldots + \theta_p x_p) > 0$. El argumento general $(\theta_0 + \theta_1 x_1 + \ldots + \theta_p x_p)$ se escribe en términos de las variables predictoras y los parámetros θ_j, cuyos valores se determinan mediante el aprendizaje del modelo.

El caso simple con un único atributo X y un modelo lineal, resulta $h_\theta(x) = g(\theta_0 + \theta_1 x)$. Luego, imponiendo $\theta_0 + \theta_1 x > 0$ se puede obtener de forma directa el valor x a partir del cual se predice respuesta positiva en términos de los parámetros (θ_0, θ_1). Para el caso de hipótesis dependientes de dos (o más) atributos, la función logística resulta una hiper-superficie, con probabilidades asociadas a las correspondientes curvas de nivel.

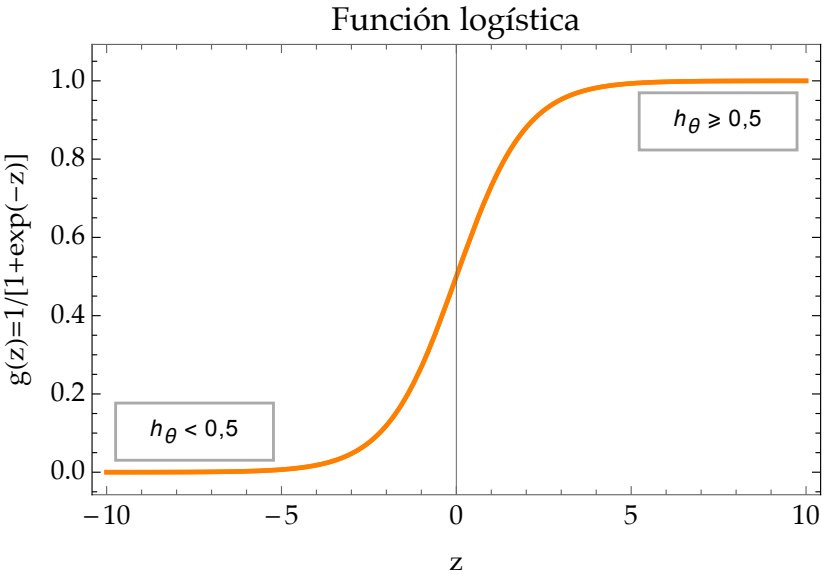

Figura 2.1: La *función logística* o *sigmoid*, $g(z)$, transforma la hipótesis lineal en una función acotada con valores interpretables como probabilidad de clases. Luego, $h_\theta(x_1, \ldots, x_p) = P(\hat{y} = 1|x_j; \theta_j)$ corresponde a la predicción de respuesta positiva cuando $h_\theta \geq 0$ ó, equivalentemente, $z = \theta_0 + \theta_1 x_1 + \ldots + \theta_p x_p > 0$.

Si bien en el contexto de este trabajo nos limitaremos a estudiar en la clasificación binaria, cabe mencionar que la generalización para variables repuesta con categorías múltiples también es posible. En estos casos, el algoritmo de clasificación puede ser adaptado a la forma "cada clase frente a todas las demás". Es decir, si la variable respuesta presenta $K + 1$ clases, con $y \in (0, 1, \ldots, K)$, se calculan k funciones hipótesis de la forma:

$$
\begin{aligned}
&h_\theta^{(0)}(x^{(i)}) = P(y = 0|x^{(i)}; \theta^{(0)})\,, \\
&h_\theta^{(1)}(x^{(i)}) = P(y = 1|x^{(i)}; \theta^{(1)})\,, \\
&\vdots \\
&h_\theta^{(K)}(x^{(i)}) = P(y = k|x^{(i)}; \theta^{(K)})\,.
\end{aligned}
$$

Luego, la predicción se calcula como: $p_i = \text{máx}\left(h_\theta^{(k)}(x^{(i)})\right)$, con $k \in (0, 1, \ldots, K)$.

Se llama *límite de decisión* a la hiper-superficie de dimensión $p - 1$ que separa aquellos valores en el espacio de las variables correspondientes a predicciones de respuesta positiva y negativa. En la Figura 2.2 presentamos dos ejemplos de límites de decisión para una hipótesis binaria con dos atributos. El de la izquierda corresponde a una recta en el plano $x_1 - x_2$, mientras que el de la derecha delimita una región circular. Este último caso corresponde a los llamados *límites de decisión no lineales*, y pueden ser recogidos correctamente por el modelo logístico si la correspondiente hipótesis es una función no lineal de los atributos. En este caso particular, por ejemplo, pueden incluirse términos cuadráticos de la forma $h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2 + \theta_5 x_1 x_2\$$; luego, la función logística resulta una superficie paraboloide y sus curvas de nivel serán círculos concéntricos en el plano $x_1 - x_2$, con centro en el punto (x_1', x_2') en donde h_θ alcanza su mínimo.

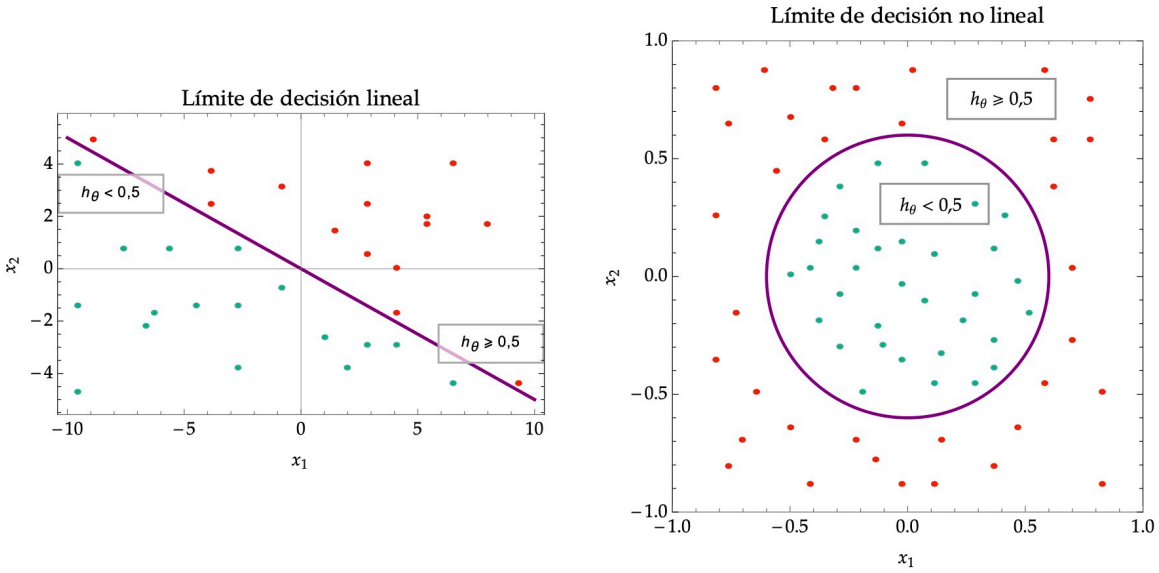

Figura 2.2: Ejemplos de límites de decisión lineal y no lineal, para una hipótesis con dos atributos.

Al igual que en el caso de la regresión, en los modelos de regresión logística se define la función de coste *Log Loss* con el fin de cuantificar la diferencia entre la hipótesis y la variable respuesta real:

$$\begin{aligned} J(\theta) &= -\frac{1}{n}\sum_{i=1}^{n}\left[y^{(i)}\log\left(h_\theta(x^{(i)})\right) + \left(1-y^{(i)}\right)\log\left(1-h_\theta(x^{(i)})\right)\right] \\ &= \frac{1}{n}\sum_{i=1}^{n}\text{Coste}\left(h_\theta(x^{(i)}), y^{(i)}\right)\,, \end{aligned} \tag{2.9}$$

en donde

$$\begin{aligned} \text{Coste}\left(h_\theta(x^{(i)}), y\right) &= -\log\left(h_\theta(x^{(i)})\right)\,, \qquad \text{si } y^{(i)} = 1\,, \\ \text{Coste}\left(h_\theta(x^{(i)}), y\right) &= -\log\left(1-h_\theta(x^{(i)})\right)\,, \qquad \text{si } y^{(i)} = 0\,. \end{aligned}$$

El comportamiento de esta función se muestra en la Figura 2.3 para la predicción de cada una de las clases. Observamos que cuando la predicción es acertada (es decir $p^{(i)} = 0, 1$), el coste es exactamente nulo, mientras que para predicciones alejadas del valor real el coste tiende a infinito. Esto es,

$$\begin{aligned} &\text{Coste}\left(h_\theta(x^{(i)}), y^{(i)}\right) = 0\,, && \text{si } h_\theta(x^{(i)}) = y^{(i)}\,; \\ &\text{Coste}\left(h_\theta(x^{(i)}), y^{(i)}\right) \to \infty\,, && \text{si } y^{(i)} = 0 \text{ y } h_\theta(x^{(i)}) \to 1\,; \\ &\text{Coste}\left(h_\theta(x^{(i)}), y^{(i)}\right) \to \infty\,, && \text{si } y^{(i)} = 1 \text{ y } h_\theta(x^{(i)}) \to 0\,. \end{aligned}$$

Uno de los métodos más utilizados para minimizar la función de coste es el *método gradiente* que, como mencionamos anteriormente, consiste en repetir hasta la convergencia el

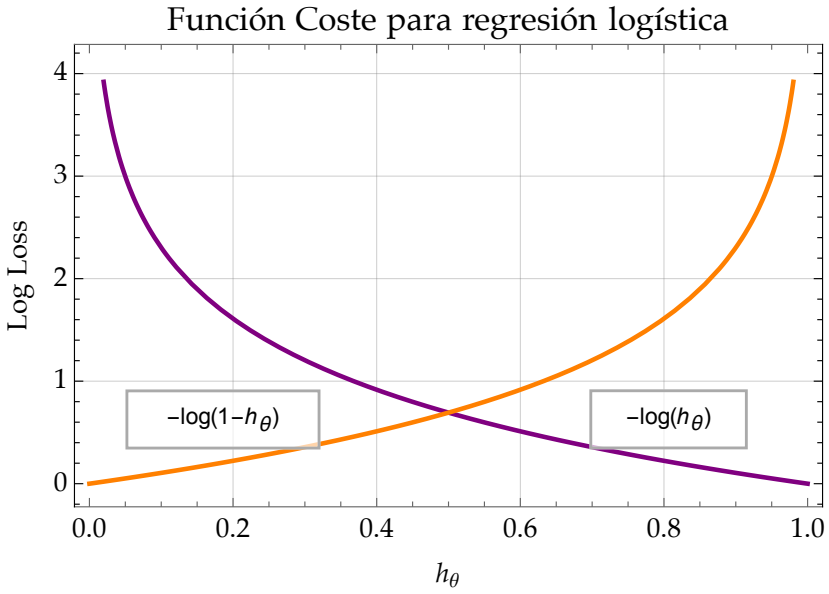

Figura 2.3: Comportamiento de la función de coste para la predicción en la clasificación binaria. Observamos que cuando la predicción es acertada (es decir $p^{(i)} = 0, 1$), el coste es exactamente nulo, mientras que para predicciones alejadas del valor real el coste tiende a infinito.

algoritmo

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1, \ldots, \theta_p)\,, \tag{2.10}$$

con p el número de atributos de la función hipótesis y α el llamado ritmo de aprendizaje. En presencia de *overfitting*, se pueden aplicar también métodos de regularización para minimizar el error generalizado del modelo logístico. Análoga al caso de la regresión, la función de coste se define entonces como:

$$J(\theta) = -\left[\frac{1}{n}\sum_{i=1}^{n} y^{(i)} \log\left(h_\theta(x^{(i)})\right) + (1 - y^{(i)}) \log\left(1 - h_\theta(x^{(i)})\right)\right] + \frac{\lambda}{2n}\sum_{j=0}^{p} \theta_j^2\,. \tag{2.11}$$

Como antes, el algoritmo de regularización resulta

$$\theta_j := \theta_j - \alpha \frac{1}{n}\left[\sum_{i=1}^{n}\left(h_\theta(x^{(i)}) - y^{(i)}\right) x_j^{(i)} + \lambda \theta_j\right]\,, \quad (j = 0, 1, \ldots, p)\,.$$

iterandose hasta la convergencia a partir de ciertos valores iniciales.

Como medida básica de la *bondad de ajuste* de la clasificación se suele utilizar la probabilidad de clasificación correcta, es decir, el número de individuos bien clasificados dividido por el número total de individuos. La probabilidad de clasificación errónea, por su parte, queda subestimada cuando esta se calcula sobre el mismo conjunto de individuos que se utilizó para estimar la función de clasificación. Es por este motivo que se utilizan dos conjuntos de individuos tomados de la muestra total: uno para estimar la función (entrenar el modelo) y el otro para valorar la clasificación obtenida (validación). El conjunto de prueba permite

↓ Obs \ Pred →	1	0	Total Marg.
1	VP	FN	Pos. Obs.
0	FP	VN	Neg. Obs.
Total Marg.	Pos. Pred.	Neg. Pred.	Total

Cuadro 2.1: Matriz de Confusión, siguiendo las siguientes referencias: VP≡Verdaderos Positivos, VN≡Verdaderos Negativos, FP≡Falsos Positivos y FN≡Falsos Negativos.

entonces estimar el error de generalización del modelo (esto es, el error real que tendrá el modelo al predecir nuevos datos). Otra forma alternativa de valoración puede llevarse a cabo clasificando cada individuo a partir de la función calculada con todos los restantes.

La llamada *matriz de confusión* es una tabla de contingencia que representa en filas las categorías verdaderas de la variable respuesta Y, y en las columnas las predicciones del modelo. Dado que se construye en base a las predicciones y los datos reales (observaciones), puede utilizarse para evaluar el funcionamiento de cualquier modelo de clasificación. Su forma más general resulta resumida en la Tabla 2.1, en donde VP≡Verdaderos Positivos, VN≡Verdaderos Negativos, FP≡Falsos Positivos y FN≡Falsos Negativos. Las métricas usuales que se calculan a partir de la matriz de confusión son la *Exactitud* (*Accuracy*) y el *Ritmo de Error* (*Error Rate*), definidas como:

$$Accuracy \equiv \frac{\text{VP} + \text{VN}}{\text{VP} + \text{VN} + \text{FP} + \text{FN}}, \tag{2.12}$$

$$Error\ Rate \equiv 1 - \text{Accuracy} = \frac{\text{FP} + \text{FN}}{\text{VP} + \text{VN} + \text{FP} + \text{FN}}. \tag{2.13}$$

En los problemas de clasificación, sin embargo, la métrica más conveniente y más utilizada para medir la exactitud de las predicciones es la llamada *curva ROC* (Receiver Operating Characteristics) ó *curva de aprendizaje*. En términos generales, consiste en una representación progresiva de los errores de entrenamiento y validación para un modelo fijo y distintos tamaños muestrales. La misma se construye a partir de un gráfico del ritmo de verdaderos positivos (*Sensitivity*) y el ritmo de falsos positivos (1-*Specificity*), definidos como:

$$Sensitivity \equiv \frac{\text{VP}}{\text{VP} + \text{FN}}. \tag{2.14}$$

$$1 - Specificity \equiv \frac{\text{FP}}{\text{VN} + \text{FP}}. \tag{2.15}$$

Luego, cada punto de una curva ROC corresponde a la ejecución de un sólo clasificador sobre una dada distribución de datos. Es muy útil porque provee una representación visual de los beneficios (VP) y costos (FP) de la clasificación de los datos, con la siguiente interpretación: cuanto mayor es el área bajo la curva ROC (AUC), más alta será la exactitud de la clasificación.

2.2.1. Clases no balanceadas

Los algoritmos de clasificación deben ser entrenados con especial atención cuando se trata predecir una variable respuesta con clases no balanceadas (esto es, distribución de individuos

proporcionalmente diferente en cada categoría). El modelo puede llevar en estos casos a predicciones sesgadas y/o pérdida de precisión en los resultados, debido a que los algoritmos no disponen de suficiente información para caracterizar a las clases minoritarias correctamente (y esto, a su vez, provoca que los clasificadores se sesguen hacia las clases mayoritarias).

El problema de las clases no balanceadas puede abordarse mediante diferentes técnicas conocidas como "métodos de muestreo". Las mismas proponen diferentes mecanismos para modificar el tamaño de la muestra original y obtener así una nueva distribución balanceada con iguales proporciones. A continuación, presentamos algunas de las más utilizadas.

1. **Submuestreo** (*Undersampling*). Este método trabaja con la clase mayoritaria. Consiste en reducir el número de observaciones de la misma hasta alcanzar una muestra balanceada. Es un método que se utiliza preferentemente cuando la base de datos original es muy grande, puesto que la reducción del número de datos de entrenamiento ayuda, por su parte, a disminuir el tiempo de cómputo y el espacio de almacenamiento.

   Existen dos tipos de submuestreos: los aleatorios (*Random*) y los informativos (*Informative*). Los métodos primeros, como su nombre lo indica, selecciona observaciones de forma aleatoria de la clase mayoritaria y estas son eliminadas hasta que la muestra alcanza proporciones balanceadas. Los métodos de submuestreo informativos, por su parte, siguen un criterio de selección pre-específicado para eliminar secuencialmente las observaciones de la clase mayoritaria. Los algoritmos conocidos como *Easy Ensemble* y *BalanceCascade* son conocidos dentro del segundo grupo por dar buenos resultados.

   El potencial problema de los métodos de submuestreo es la eliminación de observaciones que contengan información relevante para caracterizar a la clase mayoritaria.

2. **Sobremuestreo** (*Oversampling/Upsampling*). Este método trabaja con la clase minoritaria, replicando observaciones hasta alcanzar un conjunto de datos balanceado. Al igual que el caso anterior, pueden distinguirse los tipos de sobre-muestreos aleatorios y los informativos, que sobre-muestrean la clase minoritaria de forma aleatoria o con un criterio pre-seleccionado, respectivamente.

   Una ventaja de la utilización de este método es que no da lugar a pérdida de información, como en el submuestreo. LA desventaja, por su parte, es la repetición de datos dando lugar al ya mencionado sobre-ajuste (*overfitting*). Luego, la precisión de los datos de entrenamiento puede mejorar, pero no así la de los datos de validación.

3. **Generación de datos sintéticos:** (*Synthetic Data Generation*). Es un tipo particular de método de sobremuestreo, pero basado en la generación de datos sintéticos de la clase minoritaria en lugar de réplicas.

   Entre los algoritmos más utilizados destaca el llamado SMOTE (Synthetic Minority Oversamplig Technique), que crea datos artificiales a partir del espacio de los atributos similares de la muestra minoritaria (en lugar del espacio de los datos). En otras palabras, podemos decir que genera un conjunto aleatorio de datos de la clase minoritaria para mover el sesgo del clasificador hacia esta clase. La forma de generar datos artificiales es a través del *boostrapping* y del método de los k-vecinos más próximos.

4. **Aprendizaje sensible de costo:** (*Cost Sensitive Learning*). Este método evalúa el coste asociado a la clasificación errónea de las observaciones. A diferencia de los anteriores, no se basa en la creación de un nuevo conjunto de datos balanceados. Destaca el

↓ Obs \ Pred →	1	0
1	0	$C(\text{FN})= C(0,1)$
0	$C(\text{FP})= C(1,0)$	0

Cuadro 2.2: Matriz de Coste. $C(\text{FN})=C(0,1)$ y $C(\text{FP})=C(1,0)$ corresponden a los costes asociados a FN y FP, respectivamente (con $C(\text{FN}) > C(\text{FP})$).

problema del aprendizaje no balanceado a través de matrices de coste que describen el coste de la clasificación errada en un dado escenario.

La *matriz de coste* se construye de forma similar a la matriz de confusión. La diferencia está en la importancia dada a los casos FP y FN, como se muestra en el Cuadro 2.2 (comparar con el Cuadro 2.1). Utilizamos la notación $C(i|j) = 1$ para representar el costo de predicción de la clase i con valores verdaderos de la clase j, con $i \neq j$ [11]. Notemos que el coste de penalización de los casos VP y VP es cero, dado que ambos corresponden a clasificaciones correctas. El objetivo del método es elegir el clasificador con el menor costo total, definido como:

$$\text{Costo Total} = C(\text{FN}) \times \text{FN} + C(\text{FP}) \times \text{FP}\,, \tag{2.16}$$

en donde $C(\text{FN})=C(0,1)$ y $C(\text{FP})=C(0,1)$ corresponden a los costes asociados a FN y FP, respectivamente (con $C(\text{FN}) > C(\text{FP})$).

Asimismo, en los casos de clases no balanceadas la estimación el error debe ser revisada, dado que el porcentaje de clasificaciones correctas puede llevar a resultados espurios. Es por esto que otras métricas más apropiadas pueden ser derivadas de la matriz de confusión, aportando mejores resultados a la hora de evaluar el error de la clasificación. Estas son: la *Precisión* (*Precision*), medida de las clasificaciones correctas dentro de la predicción positiva; el *Recuerdo* (*Recall*) o *Sensibilidad* (*Sensitivity*) ya definida, medida de las observaciones reales que son clasificadas correctamente; y la medida F_β (F_β *Measure*), combinación de las dos anteriores como una medida de la efectividad de la clasificación. En términos de la matriz de confusión, resultan:

$$\begin{aligned} \textit{Precision} &= \frac{\text{VP}}{\text{Predicciones Positivas}} = \frac{\text{VP}}{\text{VP} + \text{FP}}\,, \\ \textit{Recall} &= \frac{\text{VP}}{\text{Positivos Observados}} = \frac{\text{VP}}{\text{VP} + \text{FN}}\,, \\ F_\beta &= [(1+\beta^2) \times \text{Recall} \times \text{Precision}]/(\beta^2 \times \text{Recall} + \text{Precision})\,. \end{aligned}$$

El coeficiente β en la última métrica determina el peso de importancia que se le asigna o bien a la Precisión o bien al Recuerdo, y se toma, en general, igual a 1 (*score* F_1).

Las diferentes métricas son utilizadas según los aspectos de la clasificación que resulten de interés para el problema. Por ejemplo, la Precisión no aporta información sobre la exactitud de las predicciones negativas, mientras que el Recuerdo es más relevante cuando se trata de conocer los casos positivos reales. En el Capítulo 4 discutiremos la estimación del error de los modelos en el contexto de la clasificación estrella/galaxia teniendo en cuenta estas definiciones.

Capítulo 3

Modelos avanzados de clasificación

Hemos estudiado en el capítulo anterior que en los casos de presencia de límites de decisión no lineales (es decir, hipótesis no lineales), es posible añadir términos polinómicos o incluso productos de variables para recoger con el modelo límites más complejos. Sin embargo, cuando el número de atributos es muy elevado, no resulta operativo introducir tantos términos de este tipo. En estos casos se propone trabajar con los llamados *modelos avanzados de aprendizaje*. Presentamos en este capítulo los principales conceptos de algunos de ellos (redes neuronales, *support vector machine* y árboles de decisión, entre otros).

3.1. Redes Neuronales

Las redes neuronales, como su nombre lo indica, son algoritmos que se desarrollan simulando el comportamiento de las neuronas (o redes de neuronas) en el cerebro. En la Figura 3.1 se muestra una representación de dicha analogía.[1] Una neurona biológica recibe estímulos (información) a través de las dendritas; estos se trasmiten al núcleo, en donde son procesados, y el resultado sale en forma de respuesta por el axón. En el caso de las neuronas artificiales, los atributos o inputs x_i constituyen la información de entrada, que es procesada a través de una *función de activación* junto con ciertos pesos w_i. El resultado del cómputo cómputo es la respuesta y_i o salida. Una red neuronal artificial ó RNA es un conjunto de estas estructuras organizadas en lo que se conoce como *capas*. Al igual que el cerebro humano, una RNA aprende a partir de ejemplos o muestras del problema por resolver (aprendizaje supervisado).

Las estructuras de redes neuronales pueden construirse de forma simple o compleja, agregando capas ocultas entre los inputs y la capa de salida. Las capas ocultas, por su parte, pueden también variar el número de nodos. La clave del modelo reside en encontrar la contribución adecuada (es decir, los pesos w_i) en una arquitectura de capas y nodos diseñada *a prori*. En este sentido, a veces es necesario ganar un poco de *intuición* probado diferentes combinaciones, aunque sin perder de vista que las capas ocultas actúan tipo *caja negra* y no suelen ser directamente interpretables. Sin embargo, la versatilidad de las diferentes funciones de activación y la libertad de combinar estructuras pueden permitirnos recoger hipótesis con límites de decisión altamente no lineales, según los requerimientos del problema dado.

La llamada propagación hacia adelante o *forward propagation* es el conjunto de procesos matemáticos que conforman funcionamiento general de una RNA, y consiste en los siguientes pasos:

[1] Créditos: https://cebebelgica.es/es_ES/blog/10/que-es-una-red-neuronal-artificial.html

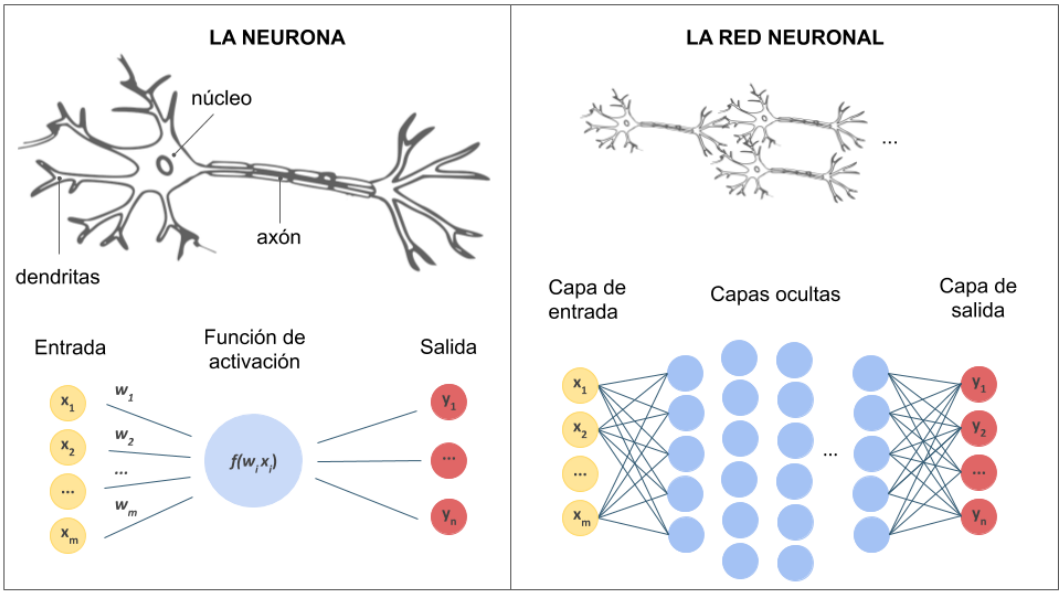

Figura 3.1: Representación de la analogía entre neuronas o redes neuronales en el cerebro y neuronas o redes neuronales artificiales. Al igual que el cerebro humano, una RNA aprende a partir de ejemplos o muestras del problema por resolver.

1. **Capa de entrada:** corresponde a los inputs (variables predictoras) de la base de datos; en esta capa tendremos tantos nodos como atributos, incluyendo $x_0 = 1$ para el cómputo del término independiente $(x_0, x_1, \ldots, x_p)$.

2. **Proceso de activación por capas**: llamaremos $g(x)$ a la función de activación elegida, $a_j^{(l)}$ a la *activación* del nodo j de la capa l ($j = 0, \ldots, s_l$, con s_l el número de nodos en la capa l), y $\theta^{(j)}$ a la matriz de pesos que controla la función de activación que pasa de la capa l a la $l+1$. La segunda capa se activa de la forma:

$$a_j^{(2)} = g\left(\theta_{20}^{(j)} x_0 + \theta_{21}^{(j)} x_1 + \ldots + \theta_{2p}^{(j)} x_p\right), \quad j = 0, 1, \ldots, s_2\,,$$

y en cada capa intermedia $(l+1)$ se calcula:

$$a_j^{(l+1)} = g\left(\theta_{(l+1)0}^{(j)} a_0^{(l)} + \theta_{(l+1)1}^{(j)} a_1^{(l)} + \ldots + \theta_{(l+1)s_l}^{(j)} a_{s_l}^{(l)}\right), \quad j = 0, 1, \ldots, s_{l+1}\,.$$

Notar que si la red posee s_l unidades en la capa l, y s_{l+1} en la capa $l+1$, entonces $\theta^{(l)}$ será de dimensión $s_{l+1} \times (s_l + 1)$.

3. **Capa de salida:** la respuesta de modelo o hipótesis se obtiene *activando* la última capa oculta. Por ejemplo, para $l+1$ capas, resulta:

$$h_\theta(x) = g\left(\theta_{10}^{(l+1)} a_0^{(l+1)} + \theta_{11}^{(l+1)} a_1^{(l+1)} + \ldots + \theta_{1s_{l+1}}^{(l+1)} a_{s_{l+1}}^{(l+1)}\right). \tag{3.1}$$

La generalización al problema de la clasificación multiclase se implementa agregando en la capa de salida tantos nodos como categorías presente la variable respuesta.

Por su parte, la propagación hacia atrás o *backward propagation* [12] es el mecanismo interno que hace posible que la RNA *aprenda* (es decir, que sea capaz de hacer cada vez mejores predicciones), y se implementa a partir de algoritmos de optimización numérica (e.g., *Stochastic Gradient Descent*,[2] *Adam* [13]). En términos generales:

1. se propaga la RNA hacia adelante y se obtiene un resultado de la capa de salida;

2. se aplica una función error o función objetivo, que coteja las diferencias entre la predicción y el valor observado; y

3. se propaga hacia atrás con el objetivo de minimizar el error, hasta hallar los pesos óptimos.

La elección de la función de activación depende de las características del problema a resolver. Entre más utilizadas podemos destacar:

- la función logística o *sigmoid*:

$$g(x) = \frac{1}{1+\exp(-x)}\,, \quad 0 \leq g(x) \leq 1\,; \tag{3.2}$$

- la función salto binario:

$$f(x) = \begin{cases} 0\,, & \text{si } x < 0 \\ 1\,, & \text{si } x \geq 0 \end{cases} \quad 0 \leq g(x) \leq 1\,; \tag{3.3}$$

- la función tangente hiperbólica:

$$f(x) = \tanh(x) = \frac{2}{1+\exp(-2x)} - 1\,, \quad -1 \leq f(x) \leq 1\,; \tag{3.4}$$

- la función ReLU (*Rectified Linear Unit*):

$$R(x) = \text{máx}(0, x)\,, \quad 0 \leq R(x) < \infty\,; \tag{3.5}$$

- la función ELU (*Exponential Linear Unit*):

$$f(x) = \begin{cases} \alpha[\exp(x) - 1]\,, & \text{si } x < 0 \\ x\,, & \text{si } x \geq 0 \end{cases} \quad -\alpha \leq g(x) < \infty\,; \tag{3.6}$$

Para los problemas de clasificación binaria la función logística es la más utilizada.

[2] https://leon.bottou.org/projects/sgd

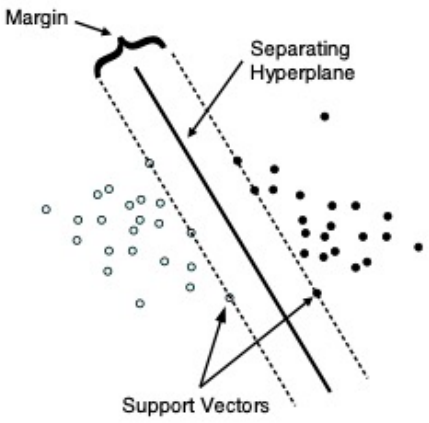

Figura 3.2: Concepto básico detrás de la técnica SVM. El margen representa el espacio entre los puntos más cercanos de cada clase, siendo su centro es el hiperplano de separación óptimo (caso separable lineal).

3.2. *Support Vector Machine* (SVM)

La llamada *Support Vector Machine* (SVM) es una técnica muy utilizada actualmente dentro de la comunidad de *Machine Learning* [14]. Representa una herramienta poderosa para abordar problemas generales de clasificación (no lineal), regresión y detección de *ouliers* con una representación intuitiva del modelo.

La técnica fue originalmente desarollada por Cortes & Vapnik (1995) [15] para clasificación binaria. Conceptualmente, lo que se busca es encontrar un hiperplano de separación óptimo entre las dos clases a partir de la maximización del llamado *margen*, definido como la distancia entre los puntos más próximos de cada clase (Figura 3.2).[3] Geométricamente, este margen corresponde a la menor distancia entre los puntos de datos y cualquier punto del hiper-plano óptimo de separación. Los puntos sobre los límites del margen son llamados vectores de soporte o *support vectors*, mientras que el centro del margen corresponde al hiperplano de separación buscado. Al igual que las técnicas ya presentadas, el método se implementa minimizando la función de coste.

Para el tratamiento de la no-linealidad (es decir, cuando no es posible encontrar un separador lineal entre las clases), los vectores *input* (espacio de los datos o *input space*) son mapeados no linealmente en hiper-espacios (espacio de los atributos o *feature space*), usualmente de dimensión superior, de forma tal que en dicho espacio pueda construirse una superficie de separación lineal entre las clases (Figura 3.3). Este procedimiento se realiza a través de lo que se conoce como técnicas de núcleos o *kernels*, que generan nuevos atributos como funciones no lineales de los datos originales a través de un mapeo. Los *kernels* más utilizados son: el radial, el lineal, el polinómico y la función *sigmoid*.

[3] Créditos: https://cran.microsoft.com/snapshot/2016-06-21/web/packages/e1071/vignettes/svmdoc.pdf

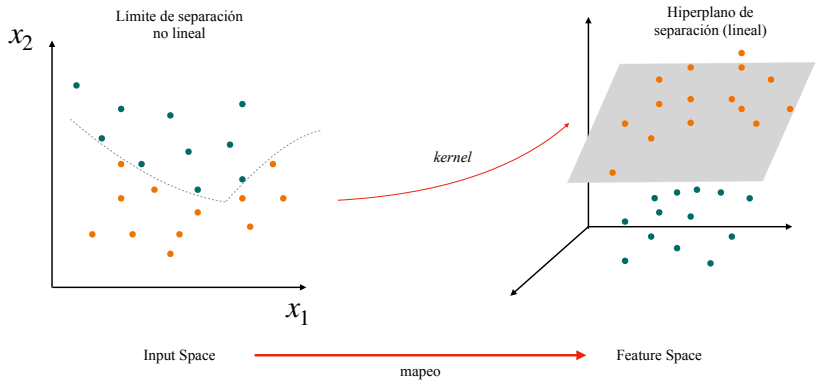

Figura 3.3: Mapeo del espacio *input* (2D) al espacio *features* (3D) a través de una función de kernel. La separación no lineal entre clases pasa a ser resulta como un problema lineal.

3.3. Árboles de regresión y clasificación

Los llamados árboles de decisión son métodos supervisados avanzados para regresión y clasificación que se basan en las divisiones binarias sucesivas del espacio de variables independientes [11].

El objetivo es crear un modelo que prediga los valores de la variable dependiente a partir de reglas de decisión simples sobre los *inputs*. Cada una de las particiones se realiza a partir del atributo diferenciador más significativo, de forma tal que en cada nodo el resultado sean grupos lo más homogéneos posibles. Las divisiones en cada nodo pueden seguir diferentes criterios, por ejemplo:

- el índice de Gini, que indica la homogeneidad de los grupos resultantes, y es definido como
$$G = 1 - \sum_k p_k \,, \tag{3.7}$$
con p_k la probabilidad de clasificación en cada una de las clases; o
- la ganacia en la información, basada en la disminución de la entropía desde las raíces del árbol (inicio) hasta las hojas (final), con
$$entropy = -\sum_k p_k \log_2 (p_k) \,, \tag{3.8}$$

entre otros. En todos los casos, las divisiones sucesivas continuan hasta que se alcanza algún criterio de corte pre-establecido.

Entre las ventajas de la utilización de los árboles de decisión podemos destacar que: (i) son modelos simples, que facilitan el entendimiento e interpretación de los usuarios, permitiendo incluso visualizar las divisiones obtenidas; (ii) requieren una preparación previa mínima de los datos, prescindiendo de variables *dummy* o normalizaciones; iii) el costo de uso de los árboles (es decir, para predecir nuevos datos) es logarítmico en el número de datos utilizados para el entrenamiento; (iv) admiten la mezcla de datos numéricos y categóricos; (v) son modelos con carácter *caja blanca*, en el sentido de presentar fácil interpretación de decisiones a partir de

condicionales booleanos (a diferencia, por ejemplo, de las redes neuronales); y (vi) posibilitan la validación de los modelos mediante test estadísticos (es decir, cuantificar la fiabilidad de los distintos modelos).

Por otra parte, también presentan algunas desventajas que caben destacar: (i) los algoritmos presentan facilidad para ramificarse y crear árboles demasiado complejos, que no se generalizan bien a otros datos (*overfitting*); (ii) son inestables cuando hay pequeñas variaciones en los datos (este problema, sin embargo, puede mitigarse utilizando los árboles en *ensembles*); (iii) las predicciones no resultan suaves ni continuas, sino constantes a trozos, por lo que no son buenos para extrapolar; y (iv) suelen crearse árboles sesgados cuando las clases nos están balanceadas. Algunas de las estrategias que pueden aplicarse para prevenir el problema del *overfitting* son:

1. limitar el tamaño del árbol manualmente, siguiendo algún criterio adecuado al problema en cuestión; o

2. el proceso de podado o *pruning*, que consiste en construir y comparar una secuencia anidada de sub-árboles mediante un recortado recursivo de las divisiones menos importantes, según la medida de costo-complejidad.

Existe varios algoritmos de clasificación basados en árboles de decisión. Entre ellos, destacamos:

- **Random Forest.** Los algoritmos *random forest* o árboles aleatorios consisten en construir muchos árboles tomando aleatoriamente partes del conjunto de datos original [16]. La predicción final se realiza por mayoría, contando las predicciones individuales de los distintos árboles.

- **Árboles con aleatorización extrema** (*Extremely Randomized Trees*). Como su nombre lo indica, este método se basan en conjuntos de árboles altamente aleatorios [17]. Esencialmente, consisten en tomar de forma aleatoria tanto los atributos como los puntos de corte en las decisiones para la creación de nodos. En el caso extremo, el método construye árboles completamente aleatorizados cuyas estructuras son independientes de los valores de salida de la muestra de entrenamiento.

Estos últimos métodos se diferencian de los *Random Forest* en considerar aleatoriedad no sólo en los subconjuntos de variables predictoras para cada uno de los árboles, sino también en el punto de corte que divide cada categoría. Como resultado, suelen obtenerse modelos con varianzas reducidas, aunque con sesgos levemente aumentados. Otra diferencia con *Random Forest* es que las muestras con las que se entrena cada árbol se escogen sin reemplazo (es decir, no son observaciones *bootstrap*).

Resumiremos a continuación algunos métodos avanzados de clasificación que pueden ser utilizados como complemento de los anteriores.

3.4. Bayes ingenuo

El método *Naive Bayes* o Bayes ingenuo se basa en la aplicación del Teorema de Bayes, pero con la suposición condicional *ingenua* de que cada pareja de atributos es independiente [18].

Dados dos eventos aleatorios A y B, el Teorema de Bayes expresa la probabilidad condicional de A, dado B, en términos de la distribución de probabilidad condicional del evento B, dado A, y la distribución de probabilidad marginal solo de A. Esto es, en el contexto de ML, que dada una variable de clasificación y y un vector de atributos $(x_1, \ldots, x_n)$, la probabilidad de y dado $(x_1, \ldots, x_n$ resulta:

$$P(y|x_1, \ldots, x_n) = \frac{P(y)P(x_1, \ldots, x_n|y)}{P(x_1, \ldots, x_n)} \,. \tag{3.9}$$

La condición "ingenua" de independencia, por su parte, establece que

$$P(x_i|y, x_1, \ldots, x_{i-1}, x_{i+i1}, \ldots, x_n) = P(x_i, |y) \,, \tag{3.10}$$

para todos los atributos i, simplificando la expresión (3.9) a la forma

$$P(y|x_1, \ldots, x_n) = \frac{P(y) \prod_{i=1}^{n} P(x_i, |y)}{P(x_1, \ldots, x_n)} \,. \tag{3.11}$$

Luego, dado que $P(x_1, \ldots, x_n)$ es constante para cada *input*, podemos establecer la siguiente regla de clasificación:

$$P(y|x_1, \ldots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i|y) \tag{3.12}$$

$$\Downarrow$$

$$\hat{y} = \arg\max_y P(y) \prod_{i=1}^{n} P(x_i|y) \,, \tag{3.13}$$

en donde se pueden estimar $P(y)$ y $P(x_i|y)$ con la probabilidad *máximo a posteriori* (MAP); en este caso, $P(y)$ es la frecuencia relativa de cada clase en el conjunto de entrenamiento T.

Luego, diferentes clasificadores pueden ser estudiados según las hipótesis que se consideren sobre la distribución $P(x_i|y)$. Se destacan los siguientes:[4]

- **Bayes ingenuo gaussiano**, en donde se considera una *likelihood* gaussiana para los atributos:

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right) \,, \tag{3.14}$$

  en donde los parámetros σ_y y μ_y son estimados utilizando la máxima probabilidad;

- **Bayes ingenuo multinomial**, en donde se toma como hipótesis que los datos están distribuidos de forma multinomial y parametrizados por vectores $\theta_y = (\theta_{y1}, \ldots, \theta_{yn})$ para cada clase, con $\theta_{y1} = P(x_i|y)$ y θ_y estimados con frecuencias relativas suavizadas:

$$\hat{\theta}_{yi} = \left(\sum_{x \in T} x_i + \alpha\right) \Big/ \left[\sum_{i=1}^{n} \left(\sum_{x \in T} x_i\right) + \alpha n\right] \,, \tag{3.15}$$

  en donde los priors suavizados ($\alpha \geq 0$) tienen en cuenta los atributos no presentes en el conjunto de aprendizaje y previenen probabilidades nulas, y $\alpha = 1$ y $\alpha < 1$ son llamados, respectivamente, suavizados de Laplace y Lidstone;

[4] https://scikit-learn.org/stable/modules/naive_bayes.html

- **Bayes ingenuo de Bernoulli**, en donde se consideran datos distribuidos según distribuciones multivariantes de Bernoulli:

$$P(x_i|y) = P(i|y)x_i + (1 - P(i|y))(1 - x_i)\,, \tag{3.16}$$

  es decir, cada atributo es tratado como una variable con valores binarios (Bernoulli, ó booleanos); y

- **Bayes ingenuo categórico**, en donde se consideran atributos con distribuciones categóricas propias.

3.5. k-vecinos más cercanos

Este algoritmo de clasificación, más conocido como *K-Nearest Neighbors* (KNN), es uno de los métodos más simples de ML [19, 20]. Consiste en calcular la distancia entre un elemento a clasificar dentro del conjunto de validación y los k vecinos más próximos pertenecientes al conjunto de entrenamiento. La distancia de proximidad puede ser computada utilizando diferentes métricas (e.g., la euclídea). El resultado del modelo puede ser o bien discreto, decidido por mayoría de votos (en donde el *voto* es la clasificación de cada vecino), o bien en términos de probabilidad de pertenencia a una clase, calculada a partir del promedio de clasificaciones de los k-vecinos más próximos ponderado con la distancia a estos, es decir:

$$f(\mathbf{x}_q) = \frac{\sum_{i=1}^{k} w_i f(\mathbf{x}_i)}{\sum_{i=1}^{k} w_i}\,, \quad \text{con} \quad w_i = \frac{1}{d(\mathbf{x}_q, \mathbf{x}_i)^2}\,, \tag{3.17}$$

en donde $\mathbf{x}_q = (x_1, \ldots, x_q)$ es el individuo a clasificar, $\mathbf{x}_i$ son los k-vecinos pertenecientes al conjunto de entrenamiento y $y_i = f(\mathbf{x}_i)$ sus respectivas clasificaciones. La elección del número k, por su parte, puede ser optimizada a través de validación cruzada usando k-particiones.

Entre las ventajas de utilizar este método podemos destacar que es simple, intuitivo y con resultados competitivos. Sin embargo, aunque suele ser rápido, el costo computacional resulta proporcional al tamaño del conjunto de entrenamiento, por lo que su complejidad aumenta para conjuntos de datos muy grandes.

Capítulo 4

Clasificación de objetos astronómicos

En este capítulo aplicaremos los métodos de clasificación ya presentados (regresión logística y algoritmos avanzados) al estudio de una base de datos particular. Se trata del catálogo de galaxias creado por el proyecto ALHAMBRA (*Advance Large Homogeneous Area Medium Band Redshift Astronomical*)[1] [21]. Dicho catálogo provee, entre otros datos, observaciones de alrededor de 100.000 galaxias en 24 filtros fotométricos, junto con un conjunto adicional de 20.000 estrellas en el halo galáctico y otros 1.000 candidatos a AGN (Núcleos Galácticos Activos). El objetivo del trabajo es aplicar las técnicas de clasificación binaria a este catálogo con el objeto de discriminar entre las categorías *galaxia* y *estrella* a partir de sus espectros fotométricos.

Comenzaremos realizando una presentación del catálogo y una estadística descriptiva de los datos. En la Sección 4.3 entrenaremos modelos de regresión logística, teniendo en cuenta el problema de las clases no balanceadas. Los resultados son comparados en la Sección 4.4 con aquellos obtenidos con algunos métodos avanzados de ML. Todos los algoritmos son implementados a través de librerías de `R` y los códigos se encuentran disponibles en el Apéndice A.

4.1. Catálogo ALHAMBRA

El objetivo principal del experimento ALHAMBRA fue construir un catálogo fotométrico multi-filtros de objetos detectables, incluyendo la determinación de sus *redshifts*[2]. ALHAMBRA observó 8 regiones diferentes del cielo (Figura 4.1), utilizando un sistema fotométrico innovador compuesto por 20 filtros contiguos (no solapados y con igual ancho ($\sim$ 300Å)), cubriendo el rango óptico completo (3500Å-9700Å), junto con los filtros estándar J, K, y Ks de la banda ancha del espectro infrarrojo cercano (*Near Infrared* o NIR) (Figura 4.2). Las observaciones fueron realizadas con el telescopio de 3,5 m del Observatorio Astronómico de Calar Alto (CAHA)[3] entre 2005 y 2012, utilizando la cámadra óptica LAICA[4] y el instrumento en el NIR Omega-2000[5].

[1] http://www.alhambrasurvey.com/
[2] Corrimiento al rojo de objetos astronómicos distantes debido a la expansión del universo.
[3] http://www.caha.es/
[4] http://www.caha.es/CAHA/Instruments/LAICA/index.html
[5] http://w3.caha.es/CAHA/Instruments/O2000/index2.html

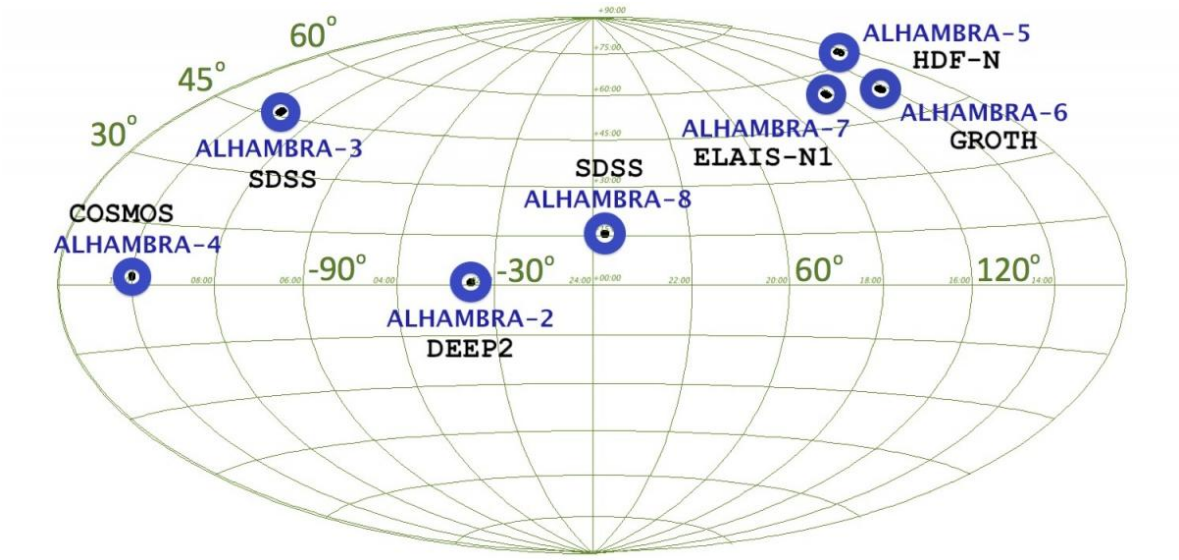

Figura 4.1: Representación en coordenadas galácticas de los sectores del cielo observados por el experimento ALHAMBRA, incluyendo las secciones coincidentes con otros experimentos (COSMOS, DEEP2, ELAIS, GOODS-N, SDSS y Groth) [6].

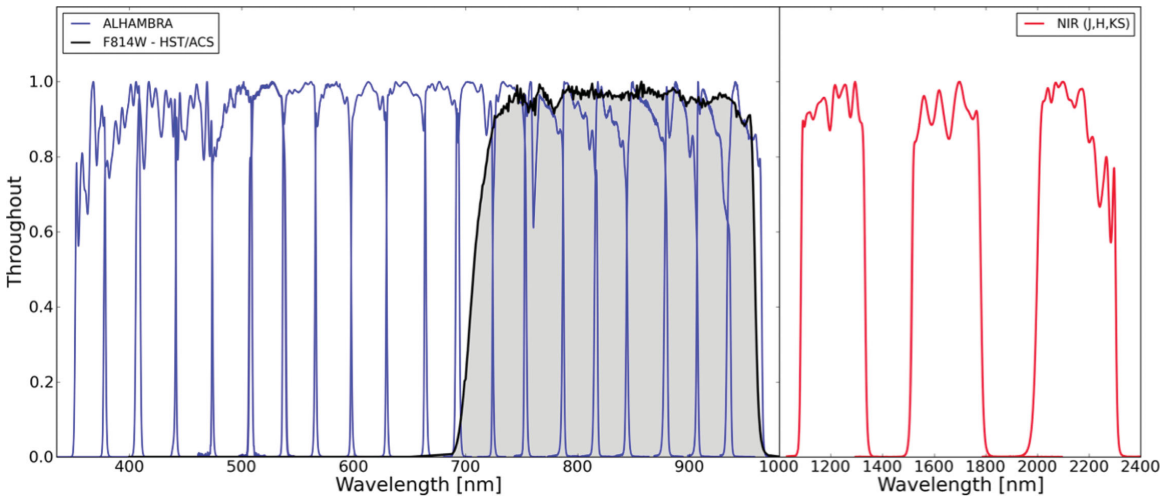

Figura 4.2: Distribución de filtros contíguos en los rangos óptico y NIR [6].

En este trabajo utilizaremos parte del catálogo ALHAMBRA descripto en Molino et. al 2014 [6]. El mismo recoge la información fotométrica de unas 438.000 galaxias, con magnitudes aparentes en el rango $17 < m_{\mathrm{F814W}} < 23$, y fotometría en 20+4 bandas, que conformarán nuestro conjuntos de variables predictoras:

- 20 filtros en el rango óptico: $F365W$, $F396W$, $F427W$, $F458W$, $F489W$, $F520W$, $F551W$, $F582W$, $F613W$, $F644W$, $F675W$, $F706W$, $F737W$, $F768W$, $F799W$, $F830W$, $F861W$, $F892W$, $F923W$, $F954W$, $F814W$;
- 3 filtros en el rango NIR: J, H, Ks;
- 1 filtro sintético $F814W$.

Esta última magnitud, $F814W$, es obtenida de una imagen sintética de detección con el propósito de definir una ventana constante y homogénea para los campos de observación de ALHAMBRA. Sin embargo, para los fines de nuestro trabajo no es relevante y optaremos por no incluirla en el conjunto de atributos.

En el catálogo se reportan, junto con las magnitudes, los respectivos errores en las medidas para cada uno de los filtros. En principio estos datos también pueden incorporarse a los modelos como atributos independientes (e.g., [7]), o bien a través de métodos para algoritmos con *atributos ruidosos* o *noisy labels* (ver, por ejemplo, [22, 23]). En este trabajo, sin embargo, nos limitaremos a trabajar sólo con las magnitudes en cada filtro como atributos sin error.

Además de la fotometría de galaxias, el catálogo incluye también la información de aproximadamente 20,000 estrellas en el halo galáctico y cerca de 1.000 candidatos a Núcleos Galácticos Activos (AGNs). La identificación *galaxia* o *estrella* se realiza siguiendo criterios estadísticos. En términos generales, a cada objeto detectado se le asigna una probabilidad dada su geometría aparente (FWHM), su magnitud en la banda $F814W$, los filtros ópticos $F489W$-$F814W$ y los colores NIR J-Ks. Para cada variable se deriva la correspondiente función de distribución de probabilidad (PDF) basada en distribuciones típicas de estrellas y galaxias. Luego, cada detección es clasificada en términos de la probabilidad de ser una estrella o una galaxia, de la siguiente forma:

$$P_{\mathrm{star}} = P_{\mathrm{star}}^{\mathrm{FWHM}} \times P_{\mathrm{star}}^{m_{F814W}} \times P_{\mathrm{star}}^{\mathrm{Opt}} \times P_{\mathrm{star}}^{\mathrm{NIR}}, \tag{4.1}$$

$$P_{\mathrm{gal}} = P_{\mathrm{gal}}^{\mathrm{FWHM}} \times P_{\mathrm{gal}}^{m_{F814W}} \times P_{\mathrm{gal}}^{\mathrm{Opt}} \times P_{\mathrm{gal}}^{\mathrm{NIR}}. \tag{4.2}$$

Las probabilidades resultantes son almacenadas en la variable estadística *Stellar_Flag* incluida en el catálogo [6].

El objetivo de este trabajo es entrenar los modelos de clasificación con los datos *crudos* (es decir, las magnitudes observadas en diferentes filtros), sin considerar hipótesis físicas de los objetos detectados. Con este propósito, la variable *Stellar_Flag* será re-codificada en una variable categórica binaria, con clases "0"(galaxias) y "1"(estrellas), y utilizada como *target* de clasificación.

4.2. Preparación de los datos y estadística descriptiva

El catálogo de galaxias ALHAMBRA puede bajarse directamente en formato CVS ("valores separados por coma") de la página web de ALHAMBRA Survey[6]. El archivo posee

[6] http://svocats.cab.inta-csic.es/alhambra/index.php?action=search

Stellar_Flag	0.0000	0.0000	0.0200	0.2395	0.2700	1.0000
F365W	12.92	22.62	23.36	23.14	24.01	27.75
F396W	12.88	22.43	23.22	22.96	23.83	28.01
F427W	12.34	22.18	23.05	22.71	23.64	27.44
F458W	12.49	21.89	22.85	22.45	23.46	27.34
F489W	12.61	21.69	22.68	22.26	23.30	27.20
F520W	12.64	21.46	22.51	22.07	23.14	26.12
F551W	12.60	21.27	22.36	21.91	23.00	26.62
F582W	12.51	21.12	22.22	21.76	22.87	25.95
F613W	12.66	20.98	22.08	21.62	22.72	27.72
F644W	12.66	20.80	21.93	21.46	22.58	24.93
F675W	12.59	20.72	21.82	21.36	22.47	25.71
F706W	12.50	20.61	21.71	21.24	22.37	24.19
F737W	12.59	20.48	21.62	21.13	22.27	23.65
F768W	12.43	20.41	21.53	21.04	22.18	23.84
F799W	12.56	20.34	21.45	20.96	22.11	24.49
F830W	12.15	20.26	21.39	20.90	22.05	23.69
F861W	12.19	20.20	21.33	20.84	21.97	23.48
F892W	12.21	20.15	21.27	20.78	21.91	23.89
F923W	11.98	20.10	21.23	20.74	21.88	23.82
F954W	11.66	20.14	21.22	20.76	21.84	24.20
J	11.43	19.77	20.89	20.46	21.59	24.63
H	11.13	19.50	20.61	20.25	21.38	24.59
Ks	11.36	19.41	20.48	20.22	21.32	25.26

Cuadro 4.1: Resumen de la estadística descriptiva del conjunto de datos seleccionados (66.346 objetos) para entrenar los modelos de clasificación.

446.343 filas (objetos detectados) y 91 columnas (atributos) que incluyen la identificación de cada objeto, sus coordenadas, información relativa a los instrumentos y características de los filtros y magintudes [6].

Para preparar la base de datos de cara al entrenamiento de los modelos, generamos un *data frame* con el subconjunto de variables que nos interesan: las magnitudes en los 20+3 filtros multi-frecuencia y la variable *Stella_Flag* con las probabilidades de pertenencia a las clases *estrella* o *galaxia* de cada objeto detectado. Del catálogo total, seleccionamos el subconjunto de individuos para los cuales disponemos de observaciones en la totalidad de los filtros. El valor *Stellar_Flag*$=0,5$, por su parte, corresponde a objetos excluidos de la clasificación en el trabajo original debido al alto nivel de incertidumbre que presentan en el cálculo de las PDFs [6]; estos objetos, por tanto, también son excluidos de nuestra muestra.[7] El conjunto de datos queda reducido entonces a 66.346 objetos observados y 24 variables, cuya estadística descriptiva se resume en el Cuadro 4.1.

[7]Estos criterios coinciden con la selección llamada *ALHAMBRA Gold Catalog*, presentada por los autores como catálogo completo de estudio [6].

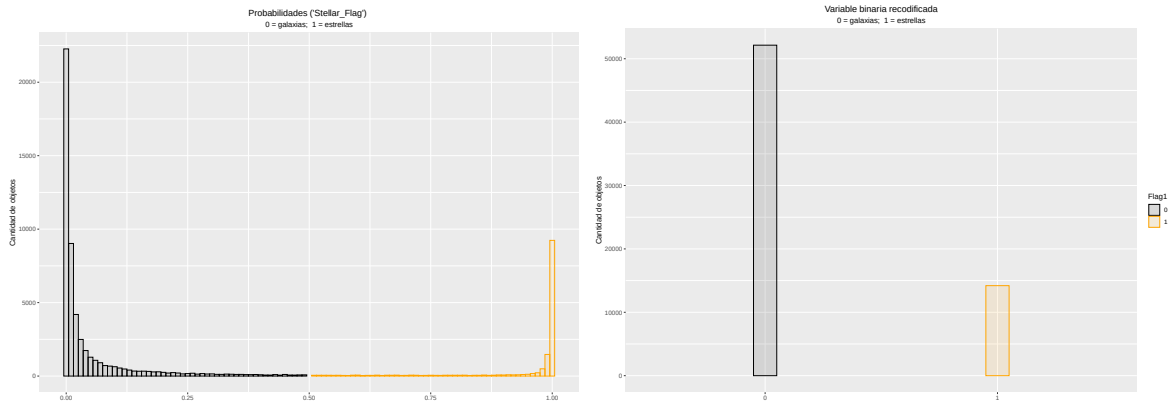

Figura 4.3: Histograma de objetos detectados según las probabilidades reportadas en la variable *Stellar_Flag* y las categorías "galaxia" ($p = 0$) y "estrella" ($p = 1$) de la recodificación en la variable binaria *Flag*.

Para estandarizar las 23 variables numéricas (magnitudes en los distintos filtros), adoptamos el criterio propuesto en [24] para la normalización de espectros galácticos. Para cada objeto i, se define la magnitud normalizada en el filtro j como

$$\hat{f}_{ij} = \frac{f_{ij}}{\sqrt{\sum_{k=1}^{23} f_{ik}^2}}, \tag{4.3}$$

de forma tal que la norma del vector magnitud de cada objeto resulta igual a la unidad. La variable *Stellar_Flag*, por su parte, es re-codificada en una variable categórica binaria con clases: "0"(galaxias) y "1"(estrellas). Definimos la variable respuesta binaria *Flag* asignando a todos los objetos con $p < 0,5$ la etiqueta "0"(galaxias) y a aquellos con $p > 0,5$ la clase "1"(estrellas). La cantidad de objetos en cada categoría resulta: 52.143 (%0,79) galaxias y 14.203 (%21) estrellas. Los respectivos histogramas según probabilidades y clases binarias para las variables *Stellar_Flag* y *Flag* se muestran en la Figura 4.3. Con el fin de explorar el comportamiento de la información recogida por los filtros según la clasificación anterior, presentamos en la Figura 4.4 histogramas y gráficos de caja para dos filtros representativos en el rango óptico ($F644W$) y el NIR (Hs), según las probabilidades agrupadas por clases. Las distribuciones en los filtros restantes presentan comportamientos similares.

Si bien los datos no representan un caso de clases altamente no balanceadas, complementaremos nuestro estudio entrenando algunos de los modelos con conjuntos de datos *exactamente* balanceados y comparando los resultados obtenidos, con el fin de ilustrar las técnicas detalladas en la Sección 2.2.1. Las referencias a los conjuntos de datos con los que trabajaremos son las siguientes:

- X: conjunto de datos completo [66.346 individuos, 23 variables predictoras (magnitudes por filtros) y una variable respuesta binaria (*Flag*)];
- $X.Train$: conjunto de entrenamiento para la clasificación de *Flag* (%70);
- $X.Val$: conjunto de validación para la clasificación de *Flag* (%30);
- $X.T.over$: conjunto de entrenamiento balanceado con *oversamplig*;

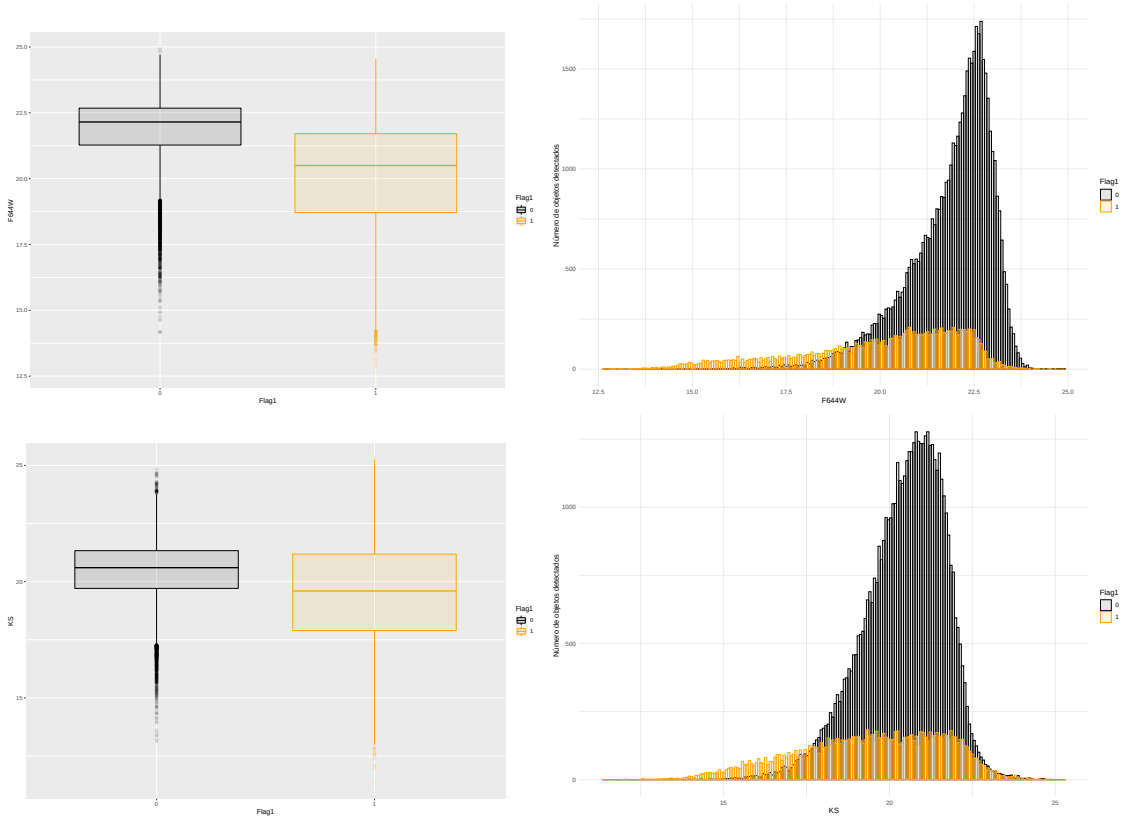

Figura 4.4: Histogramas y gráficos de caja para dos filtros representativos en el óptico ($F644W$) y el NIR (Hs), según las probabilidades agrupadas por clases. Las distribuciones son similares en los filtros restantes.

- *X.T.under*: conjunto de entrenamiento balanceado con *undersamplig*;
- *X.T.rose*: conjunto de entrenamiento balanceado con ROSE[8];

En todos los casos, los conjuntos se construyen a partir de selección sin reposición con semilla fija (condiciones iniciales), para permitir la posterior reproducción de los resultados.

4.3. Clasificación con modelos de regresión logística

El entrenamiento de los modelos de regresión logística es implementado en `R` con la función `glm` (*Generalized Linear Models*) de la librería `stats`. Utilizando las funciones `predict` (librería `stats`) y `confusionMatrix` (librería `caret`) generamos las matrices de confusión para cada modelo (es decir, para cada conjunto de entrenamiento).

Los resultados obtenidos se resumen en el Cuadro 4.2 y las correspondientes curvas ROC se muestran en la Figura 4.5. El ajuste del modelo logístico es muy bueno (Accuracy >0.9) para todos los conjuntos de entrenamiento. Como esperábamos, las técnicas de *oversampling* y *undersamplig* no muestran mejoras significativas en la predicción de clasificación dado que en nuestros datos el balanceo de clases es aceptable (36.472 (%0,79) galaxias y 14.203 (%21) estrellas en el $X.Train$). Por su parte, el entrenamiento con los datos sintéticos arroja una exactitud levemente menor, que se ve reflejada en una disminución de la correspondiente AUC ó área bajo la curva ROC.

Una forma novedosa de representar gráficamente la bondad del ajuste en los modelos de regresión logística con respuesta binaria es a través de los llamados *Separation Plots* [25], generados en `R` con la librería `separationplot`[9] (Fig. 4.6). En estas representaciones, cada una de las líneas indica un individuo de la muestra. La disposición de las mismas sigue los valores predichos para la variable respuesta, mientras que los colores son asignados según los respectivos valores observados. Luego, una separación visual apreciable se interpreta como un buen ajuste del modelo, cuya función logística se incluye representada por la curva negra central como referencia.

4.4. Clasificación con modelos avanzados

Pasaremos ahora a explorar la clasificación binaria utilizando algunos de los modelos avanzados descriptos en el Capítulo 3. En todos los casos, consideraremos $X.Train$ como único conjunto de entrenamiento.

Support Vector Machine. La clasificación binaria utilizando SVM se implementa en `R` con el paquete *texttte*1071[10], que incluye diferentes opciones de clasificación y permite la elección de distintos *kernels*.

Los resultados de entrenar modelos utilizando las funciones lineal, polinómica, radial y *sigmoid* se muestran en el Cuadro 4.3. Observamos que los modelos SVM radial, polinómico y lineal muestran métricas similares a las del modelo de regresión logística. El modelo con

[8] ROSE (*Random Over-Sampling Examples*) es un paquete disponible en `R` para generar datos artificiales sintéticos basados en métodos de muestreo y aproximación bootstrap suavizada (https://cran.r-project.org/web/packages/ROSE/ROSE.pdf).

[9] https://cran.r-project.org/web/packages/separationplot/separationplot.pdf

[10] http://127.0.0.1:37104/help/library/e1071/doc/svmdoc.pdf

	X.Train			X.T.over			X.T.under			X.T.rose		
		Obs →			Obs →			Obs →			Obs →	
Matriz	↓Pred	0	1		0	1		0	1		0	1
de	0	15313	358	0	14532	1139	0	14499	1172	0	14330	1341
Confusión	1	824	3409	1	440	3783	1	439	3794	1	546	3687
Exactitud			0.9406			0.9202			0.9191			0.9052
95 %C.I.			(0.9372, 0.9439)			(0.9163, 0.9239)			(0.9152, 0.9228)			(0.901, 0.9092)
Sensitividad			0.9489			0.9700			0.9706			0.9633
Especificidad			0.9050			0.7686			0.7640			0.7333
Precisión			0.9772			0.9273			0.9252			0.9144
F_1			0.9628			0.9482			0.9474			0.9382

Cuadro 4.2: Métricas para los modelos de regresión logística ajustados con los diferentes conjuntos de entrenamiento. En todos los casos, la clase positiva es 0.

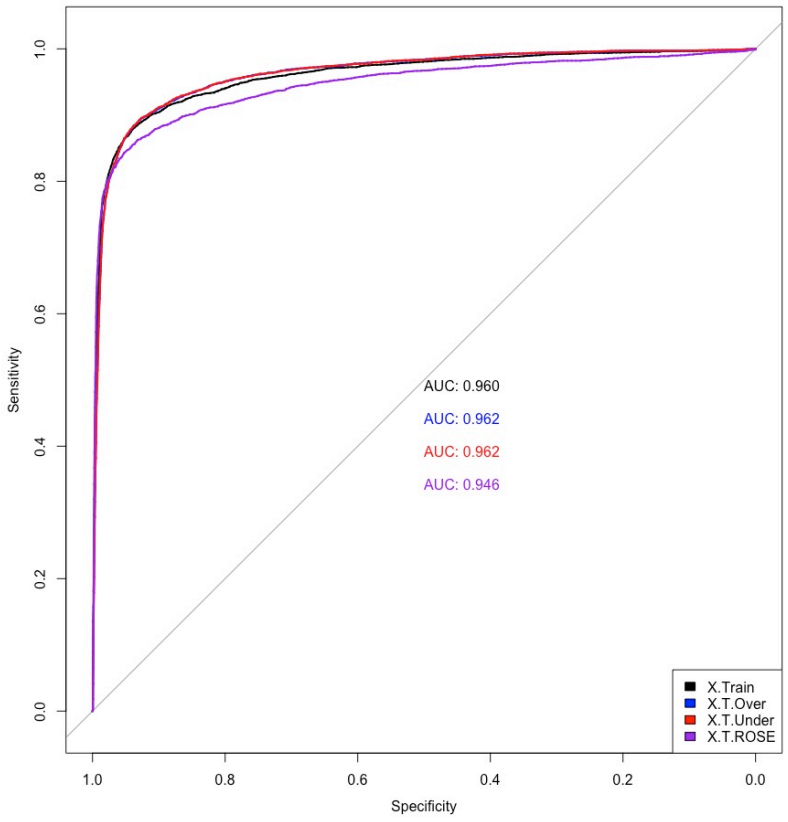

Figura 4.5: Comparación de curvas ROC para los distintos datos de entrenamiento. Las técnicas de *oversampling* y *undersamplig* no muestran mejoras significativas en los modelos de clasificación, dado que en nuestro caso el balanceo de clases es aceptable.

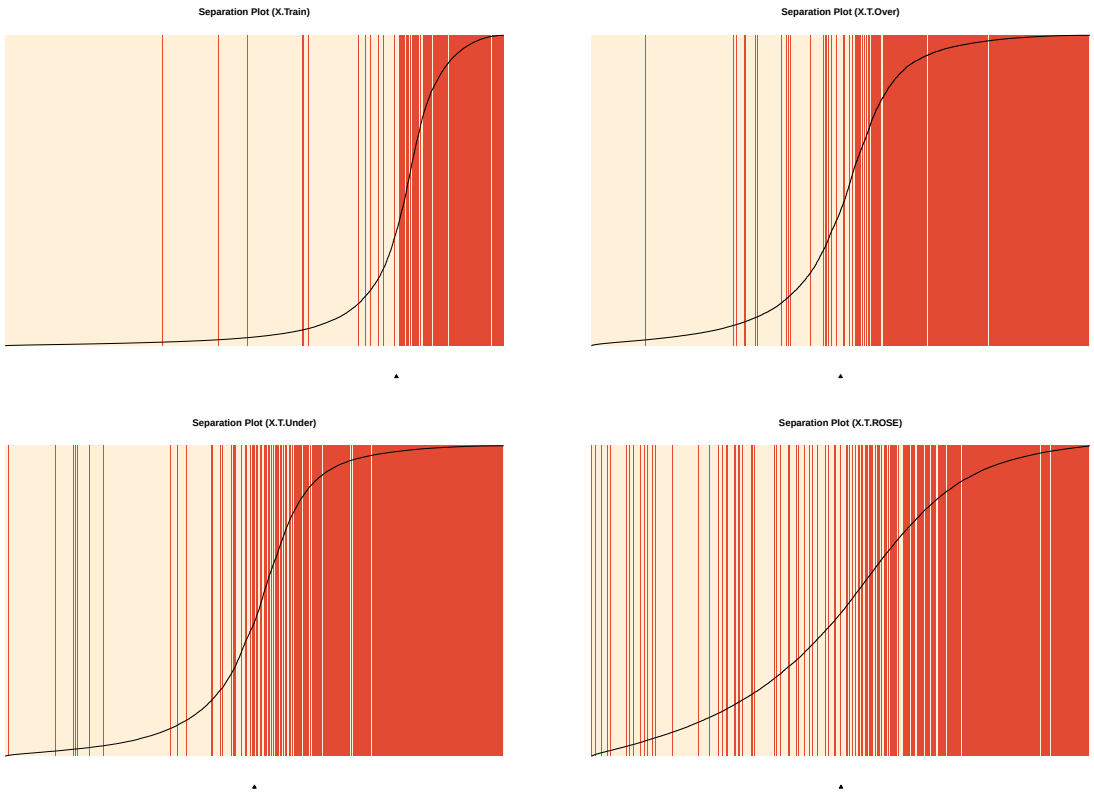

Figura 4.6: *Separation Plots*: gráficos de diagnóstico para el modelo logístico con respuesta binaria. En estas representaciones, cada una de las líneas indica un individuo de la muestra. La mismas se disponen según los valores predichos para la variable respuesta, mientras que los colores son asignados según los respectivos valores observados. Luego, una separación visual apreciable se interpreta como un buen ajuste del modelo, cuya función logística se incluye representada por la curva negra central como referencia.

	SVM-Radial			SVM-Polinomial			SVM-Sigmoid			SVM-Lineal		
		Obs →			Obs →			Obs →			Obs →	
Matriz	↓Pred	0	1		0	1		0	1		0	1
de	0	15513	158	0	15527	144	0	13560	2111	0	15324	347
Confusión	1	660	3573	1	826	3407	1	2224	2009	1	805	3428
Exactitud			0.9589			0.9513			0.7822			0.9421
95 %C.I.			(0.9561, 0.9616)			(0.9482, 0.9542)			(0.7764, 0.7879)			(0.9388, 0.9453)
Sensitividad			0.9592			0.9495			0.8591			0.9501
Especificidad			0.9577			0.9594			0.4876			0.9081
Precisión			0.9899			0.9908			0.8653			0.9779
F_1			0.9743			0.9697			0.8622			0.9638

Cuadro 4.3: Resumen de los modelos de Support Vector Machine con diferentes kernels: radial, polinómico, función sigmoide y lineal. En todos los casos, la clase positiva es 0.

	tree			c5.0			*random forest*			extraTrees		
		Obs →			Obs →			Obs →			Obs →	
Matriz	↓Pred	0	1		0	1		0	1		0	1
de	0	15234	437	0	15408	263	0	15511	160	0	15532	139
Confusión	1	1080	3153	1	761	3472	1	625	3608	1	628	3605
Exactitud			0.9238			0.9486			0.9606			0.9615
95 %C.I.			(0.92, 0.9274)			(0.9454, 0.9516)			(0.9578, 0.9632)			(0.9587, 0.9641)
Sensitividad			0.9338			0.9529			0.9613			0.9611
Especificidad			0.8783			0.9296			0.9575			0.9629
Precisión			0.9721			0.9832			0.9898			0.9911
F_1			0.9526			0.9678			0.9753			0.9354

Cuadro 4.4: Resumen de los resultados obtenidos con modelos de clasificación basados en árboles de decisión: `tree`, `c5.0`, *random forest* y *Extremely Randomized Trees*.

kernel sigmoid, por su parte, ajusta peor el conjunto de datos. En particular, la *Specificity* es significativamente más baja (0.4876), en acuerdo con el ritmo alto de FP que presenta la matriz de confusión.

Árboles de decisión. Entre las librerías disponibles en `R` para entrenar árboles se destacan: `tree`, `c50`[11], `randomForest`, y `extraTree`[12], mientras que en `Python` los algoritmos se implementan con `scikit-learn`[13].

Entrenamos diferentes modelos de árboles y comparamos los resultados en el Cuadro 4.4. Los dos primeros modelos, *tree* y 50, corresponden a árboles simples, mientras que *random forest* y *extraTrees* son modelos más complejos basados en árboles con aleatoriedad en atributos y puntos de corte. Observamos que todos los modelos clasifican muy bien los datos, obteniéndose ligeramente mejores resultados con el modelo *extraTrees*.

[11] https://cran.r-project.org/web/packages/C50/vignettes/C5.0.html
[12] https://cran.r-project.org/web/packages/extraTrees/extraTrees.pdf
[13] https://scikit-learn.org/stable/modules/tree.html#

	nn (0)			Bayes ingenuo			k-vecinos (knn5)			k-vecinos (knn20)		
		Obs →			Obs →			Obs →			Obs →	
Matriz	↓Pred	0	1		0	1		0	1		0	1
de	0	15328	343	0	13819	1852	0	15358	313	0	15422	249
Confusión	1	847	3386	1	1692	2541	1	582	3651	1	631	3602
Exactitud			0.9402			0.8219			0.955			0.9558
95 %C.I.			(0.9368, 0.9435)			(0.8166, 0.8272)			(0.9521, 0.9579)			0.9528, 0.9586)
Sensitividad			0.9476			0.8909			0.9635			0.9607
Especificidad			0.9080			0.5784			0.9210			0.9353
Precisión			0.9781			0.8818			0.98			0.9841
F_1			0.9626			0.8863			0.9717			0.9723

Cuadro 4.5: Resumen de los resultados obtenidos con los modelos avanzados de clasificación: redens neuronales (nn) Bayes ingenuo y k-vecinos más próximos (con $k = 5, 20$). En todos los casos, la clase positiva es 0.

Redes Neuronales Implementaremos modelos de clasificación utilizando algoritmos de redes neuronales en `R`, mediante la librería `neuralnet`.[14]

A diferencia de los árboles de decisión, estos algoritmos funcionan como *cajas negras*, por lo que en general es necesario realizar varias pruebas con los parámetros y el diseño de la arquitectura de capas para ganar intuición y mejorar el modelo. Nos limitaremos aquí a presentar el modelo de RNA que ajusta el conjunto completo de variables en los datos de entrenamiento y que no posee capas ocultas. Los resultados obtenidos se resumen en el Cuadro 4.5. Observamos que este caso simple presenta un ajuste muy bueno de los datos, con métricas comparables a las de los modelos anteriores.

Naive Bayes. Nos limitaremos a presentar los resultados de la clasificación utilizando el método Bayes ingenuo estándar, que supone independencia entre las variables predictoras y una distribución Gaussiana, para una clase dada. La implementación de estos clasificadores en `R` está incluida en la librería `e1071`, mientras que en `Python` es parte de `scikit-learn`[15].

Los resultados obtenidos se resumen en el Cuadro 4.5. En este caso observamos que el modelo presenta un valor de *Specificity* bajo, en acuerdo con los valores elevados de FP en su matriz de confusión. El ajuste en general es peor que el obtenido con el resto de los modelos, comparándose al del modelo SMV-sigmoid.

k-vecinos más cercanos. Implementamos el algoritmo en `R` utilizado la función `knn3` de la librería `caret`. Los resultados obtenidos para $k = 5, 10, 50, 100$ son similares, mostrando en todos los casos modelos con buena predicción, comparable a la del modelo logístico. En el cuadro 4.5 se detallan las métricas correspondientes a $k = 5$ y $k = 20$. Observamos que para $k = 20$ las métricas mejoran, resultando este el modelo con mejor *Precision* de todos los que hemos entrenado.

[14] https://www.rdocumentation.org/packages/neuralnet/versions/1.44.2/topics/neuralnet
[15] https://scikit-learn.org/stable/index.html

Capítulo 5

Conclusiones

En la era actual de acceso a grandes volúmenes de datos, las técnicas estadísticas de aprendizaje automático representan una herramienta poderosa para el tratamiento de la información. Los campos de aplicación de estos métodos son diversos y, en particular, en el área de las ciencias observacionales como la Astronomía han conllevado a un cambio de paradigma en la investigación e interpretación de los datos.

En este trabajo, nos propusimos presentar una introducción a los métodos de aprendizaje automático supervisado con una aplicación particular a la clasificación binaria de objetos astronómicos. Para tal fin, elegimos el catálogo de observaciones del proyecto ALHAMBRA, de acceso público, con el objetivo de clasificar objetos celestes puntuales en las categorías *galaxia* y *estrella*, a partir de sus magnitudes observadas en 23 filtros fotométricos contiguos en los rangos óptico y NIR.

Entrenamos primero el modelo de regresión logística, para pasar luego a algoritmos más avanzados, pensados para resolver problemas más complejos (SVM, árboles de decisión, redes neuronales, entre otros). En el primer caso, contemplamos también técnicas específicas para abordar posibles casos de bases de datos con clases no balanceadas.

La implementación fue realizada en `R` mediante librerías específicas disponibles. Los códigos se encuentran disponibles en el Apéndice A, con el fin de posibilitar la reproducibilidad de los resultados obtenidos. En todos los modelos estudiados, los resultados arrojan buenos ajustes, con errores de predicción bajos cuando son testados con datos de validación. Las métricas obtenidas son presentadas en tablas comparativas.

Cabe destacar, asimismo, que en este tipo de comparaciones es importante valorar también la complejidad de los modelos, aplicando el principio de simplicidad (es decir, si los resultados para dos modelos son similares, preferiremos trabajar siempre con el modelo más simple). Luego, en este estudio particular, el modelo de regresión logística resulta preferido, por su buena capacidad de predicción y la simpleza en implementación e interpretación.

Los resultados obtenidos en este trabajo para la clasificación de objetos del catálogo ALHAMBRA son originales, y los modelos entrenados resultan comparables a otros estudiados con catálogos astronómicos de características similares (miniJPAS [7], PAU [8] y J-PLUS[9]).

Bibliografía

[1] L.-L. Li, Y.-X. Zhang, Y.-H. Zhao and D.-W. Yang, *The application of artificial neural networks in astronomy*, *Progress in Astronomy* **24** (2006) 285.

[2] N. M. Ball and R. J. Brunner, *Data Mining and Machine Learning in Astronomy*, *International Journal of Modern Physics D* **19** (2010) 1049 [0906.2173].

[3] Y. Zhang and Y. Zhao, *Astronomy in the Big Data Era*, *Data Science Journal* **14** (2015) 11.

[4] D. Baron, *Machine Learning in Astronomy: a practical overview*, *arXiv e-prints* (2019) arXiv:1904.07248 [1904.07248].

[5] J. VanderPlas, A. J. Connolly, Z. Ivezic and A. Gray, *Introduction to astroML: Machine learning for astrophysics*, in *Proceedings of Conference on Intelligent Data Understanding (CIDU*, pp. 47–54, Oct., 2012, DOI [1411.5039].

[6] A. Molino, N. Benítez, M. Moles, A. Fernández-Soto, D. Cristóbal-Hornillos, B. Ascaso et al., *The ALHAMBRA Survey: Bayesian photometric redshifts with 23 bands for 3 deg2*, *Monthly Notices of the Royal Astronomical Society* **441** (2014) 2891.

[7] P. O. Baqui, V. Marra, L. Casarini, R. Angulo, L. A. Díaz-García, C. Hernández-Monteagudo et al., *The miniJPAS survey: star-galaxy classification using machine learning*, *Astronomy & Astrophysics* **645** (2021) A87 [2007.07622].

[8] L. Cabayol, I. Sevilla-Noarbe, E. Fernández, J. Carretero, M. Eriksen, S. Serrano et al., *The PAU survey: star-galaxy classification with multi narrow-band data*, *Monthly Notices of the Royal Astron. Society* **483** (2019) 529 [1806.08545].

[9] C. Wang, Y. Bai, C. López-Sanjuan, H. Yuan, S. Wang, J. Liu et al., *J-PLUS: Support vector machine applied to STAR-GALAXY-QSO classification*, *Astronomy & Astrophysics* **659** (2022) A144 [2106.12787].

[10] *Class variables in regression*, in *Applied Regression Analysis: A Research Tool*, J. O. Rawlings, S. G. Pantula and D. A. Dickey, eds., pp. 269–323, Springer New York, (1998).

[11] L. Breiman, J. Friedman, R. Olshen and C. Stone, *Classification and regression trees*. Chapman & Hall, New York, 1984.

[12] P. J. Werbos, *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*, Ph.D. thesis, Harvard University, 1974.

[13] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2014. 10.48550/ARXIV.1412.6980.

[14] K. P. Bennett and C. Campbell, *Support vector machines: Hype or hallelujah?*, *SIGKDD Explor. Newsl.* **2** (2000) 1–13.

[15] C. Cortes and V. Vapnik, *Support-vector networks*, *Machine Learning* **20** (1995) 273.

[16] L. Breiman, *Random forests*, *Machine learning* **45** (2001) 5.

[17] P. Geurts, D. Ernst and L. Wehenkel, *Extremely randomized trees*, *Machine Learning* **63** (2006) 3.

[18] H. Zhang, *The optimality of naïve bayes*, in *In FLAIRS2004 conference*, 2004.

[19] N. S. Altman, *An introduction to kernel and nearest-neighbor nonparametric regression*, *The American Statistician* **46** (1992) 175.

[20] T. Hastie, *The elements of statistical learning : data mining, inference, and prediction*, Springer series in statistics. Springer, New York, 2001.

[21] M. Moles, N. Benítez, J. A. L. Aguerri, E. J. Alfaro, T. Broadhurst, J. Cabrera-Caño et al., *The Alhambra Survey: a Large Area Multimedium-Band Optical and Near-Infrared Photometric Survey*, *The Astronomical Journal* **136** (2008) 1325 [0806.3021].

[22] N. Natarajan, I. S. Dhillon, P. K. Ravikumar and A. Tewari, *Learning with noisy labels*, in *Advances in Neural Information Processing Systems*, C. Burges, L. Bottou, M. Welling, Z. Ghahramani and K. Weinberger, eds., vol. 26, Curran Associates, Inc., 2013.

[23] H. Song, M. Kim, D. Park, Y. Shin and J.-G. Lee, *Learning from noisy labels with deep neural networks: A survey*, 2020. 10.48550/ARXIV.2007.08199.

[24] SDSS collaboration, *Spectral classification of quasars in the Sloan Digital Sky Survey: Eigenspectra: redshift and luminosity effects*, *Astron. J.* **128** (2004) 2603 [astro-ph/0408578].

[25] B. Greenhill, M. D. Ward and A. Sacks, *The separation plot: A new visual method for evaluating the fit of binary models*, *American Journal of Political Science* **55** (2011) 991.

Apéndice A

Códigos en R

Cómputo en paralelo:

```
library(foreach)
library(parallel)
library(iterators)
library(doParallel)
detectCores() # [1] n
registerDoParallel(cores = n-1)
```

Conjuntos de entrenamiento y validación:

```
tr=round(nrow(X1)*0.7)
set.seed(0)
muestra=sample.int(nrow(X),tr)
X.Train=X[muestra,]
X.Val=X[-muestra,]

library(ROSE)
# OVERSAMPLING:
N1 = 2*sum(X.Train$Flag==0)
X.T.Over <- ovun.sample(Flag ~ .,data=X.Train,method="over",N=N1)$data
# UNDERSAMPLIG:
N2 = 2*sum(X.Train$Flag==1)
X.T.Under <- ovun.sample(Flag ~ .,data=X.Train,method="under",N=N2,seed=1)$data
# OVERSAMPLING y UNDERSAMPLING:
N3 = dim(X.Train)[1]
X.T.Both <- ovun.sample(Flag ~ .,data=X.Train,method="both",p=0.5, N=N3,seed=1)$data
# SINTÉTICOS:
X.T.ROSE <- ROSE(Flag ~ .,data=X.Train,seed=1)$data
```

Regresión logística:

```
library(caret)
library(pROC)
library(separationplot)
```

```
gfit1=glm(Flag ~ ., data=X.Train, family=binomial)
p=predict(gfit1, X.Val, type="response")
PredFlag = as.factor( p>0.5 )
levels(PredFlag1)=c("0", "1")
matrizLogis1 <- confusionMatrix(X1.Val$Flag, PredFlag)
test_prob = predict(gfit1, newdata = X1.Val, type = "response")
test_roc = roc(X.Val$Flag ~ test_prob, plot=TRUE, print.auc=TRUE)
somers2(gfit1$fitted.values, gfit1$y)
separationplot(pred=gfit1$fitted.values, actual=gfit1$y, type="rect",
   line=TRUE, show.expected=TRUE, heading="Separation Plot (X.Train)")
```

Support Vector Machine:

```
library(e1071)
fitsvm1 <- svm(Flag ~ ., data = X.Train)
predictedSVM1 = predict(fitsvm1,X.Val)
matrizSVM1<-confusionMatrix(X.Val$Flag, predictedSVM1)

fitsvm2 <- svm(Flag ~ ., data = X.Train, kernel="polynomial")
fitsvm3 <-svm(Flag ~ ., data = X.Train, kernel="sigmoid")
fitsvm4 <-svm(Flag1 ~ ., data = X.Train, kernel="linear")
```

Redes Neuronales:

```
library(neuralnet)
Train = data.frame(X.Train$Flag, model.matrix(Flag ~ ., data=X.Train)[,-1])
Validate = data.frame(X.Val$Flag, model.matrix(Flag ~ ., data=X.Val)[,-1])
nn1 <- neuralnet(Flag ~ ., data=Train, hidden = c(0), act.fct="logistic")
pred <- predict(nn1,Validate)
predictedNN1 = factor(Predict$net.result[,2]>0.5, labels=c("0","1"))
matrizNN1 <- confusionMatrix(Val.Xmodel$Flag, predictedNN1)
```

k-vecinos más próximos:

```
library(caret)
knn <- knn3(Flag1 ~ ., data = X.Train, k=10)
pred.knn <- predict(knn, X.Val)
p = factor(pred.knn[,2]>0.5)
levels(p) <- c("0", "1")
matriz.knn <- confusionMatrix(X.Val$Flag, p)
```

Árboles de decisión:

```
library(tree)
tree1 = tree(Flag ~ ., data=X.Train)
predicedtree1 = predict(tree1, X.Val, type="class")
matriztree1<-confusionMatrix(X.Val$Flag, predicedtree1)

library(C50)
c50 = C5.0(Flag ~ ., data= X.Train)
```

```
predictc50 = predict(c50, X.Val, type = çlass")
matrizc50 <- confusionMatrix(X.Val$Flag, predictc50)

library(randomForest)
fitRF <- randomForest(Flag ~ ., data = X.Train, ntree=500)
predictedRF = predict(fitRF,X.Val)
matrizRF1<-confusionMatrix(X.Val$Flag, predictedRF)

library(rJava)
library(extraTrees)
et <- extraTrees(x=X.Train[,-c(25)], y=X.Train$Flag)
yhat <- predict(et, X.Val[,-c(25)])
matrizET1<-confusionMatrix(X.Val$Flag, yhat)
```

Bayes ingenuo:

```
library(e1071)
fitbayes <- naiveBayes(Flag1  ., data = X1.Train)
predictedBayes= predict(fitbayes,X1.Val)
matrizNB <- confusionMatrix(X1.Val$Flag1, predictedBayes)
```