



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE
TELECOMUNICACIÓN

GRADO EN INGENIERÍA EN TECNOLOGÍAS DE LA
TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

CLASIFICACIÓN DE LA MORFOLOGÍA DE GALAXIA DISTANTES
UTILIZANDO REDES NEURONALES

Autor: Adrián Oter Astillero

Tutor: Óscar Barquero Pérez

Cotutor: Rebeca Goya Esteban

Curso académico 2016 / 2017

Resumen

Los astrónomos tienen más de cien mil millones de galaxias registradas en sus bases de datos. Cada galaxia tiene diferentes formas y colores. Conocer la formación, evolución o distribución de estas galaxias, puede ser clave para entender la creación del universo. Para ello, antes deben ser ordenadas y clasificadas según sus características.

Durante la última década, esta clasificación se ha intentado llevar a cabo a través de la ayuda de miles de personas, clasificando cada galaxia una a una. Pero la cantidad de galaxias registradas no deja de crecer. Por ese motivo, la clasificación de las galaxias se debe hacer de forma automática.

En los últimos años, las técnicas de análisis de datos (*machine learning*) han resurgido debido a la cantidad masiva de datos que se pueden obtener en cualquier ámbito.

En este contexto, este Trabajo Fin de Grado se enfoca en utilizar distintas técnicas de *machine learning* (principalmente redes neuronales) para obtener una clasificación sobre una base de datos relevante de galaxias. Esta base de datos estará compuesta por aproximadamente 60000 imágenes de galaxias lejanas, obtenidas por distintos telescopios de la Tierra.

Índice general

Capítulos	Página
1. Introducción	7
1.1. Motivación	7
1.2. Objetivos	8
1.3. Estructura de la memoria	8
2. Estado del Arte	10
2.1. Contexto del estudio	10
3. Descripción de datos	16
3.1. Obtención de imágenes	16
3.2. Tipos de galaxias	18
3.3. Descripción de la competición Kaggle	22
3.4. Datos del estudio	25
4. Métodos	27
4.1. Statistical Learning	27
4.2. Machine Learning	28
4.2.1. Tipos de problemas	28
4.2.1.1. Supervised Learning	29
4.2.1.2. Unsupervised Learning	31
4.3. Generalización, sobreajuste y sobregeneralización	32
4.4. Naive Bayes	33
4.5. Redes neuronales	36
4.5.1. Descenso por Gradiente	39

4.6.	Validación cruzada	40
4.6.1.	Validación cruzada de K iteraciones (<i>K-fold cross-validation</i>)	40
4.6.2.	Validación cruzada aleatoria	41
4.6.3.	Validación cruzada dejando uno fuera (<i>Leave-one-out cross-validation, LOOCV</i>)	42
5.	Análisis y resultados	43
5.1.	Funciones de <i>Sklearn</i>	43
5.2.	Configuración de los parámetros	45
5.3.	Matriz de confusión	46
6.	Conclusiones y líneas futuras	52

Índice de figuras

2.1. Paso 1 clasificación en Galaxy Zoo.[1]	12
2.2. Paso 2 clasificación en Galaxy Zoo.[1]	12
2.3. Paso 3 clasificación en Galaxy Zoo.[1]	13
2.4. Paso 4 clasificación en Galaxy Zoo.[1]	13
3.1. Nebulosa Ojo de Gato.[4]	17
3.2. Diagrama de Morfología Galáctica Hubble - de Vaucouleurs. [25]	18
3.3. NGC 1316. Ejemplo de galaxia elíptica.[4]	19
3.4. Galaxia Remolino. Ejemplo de galaxia espiral.[4]	20
3.5. Galaxia Sombrero. Ejemplo de galaxia lenticular.[4]	20
3.6. NGC 55. Ejemplo de galaxia irregular.[4]	21
3.7. Árbol de decisión sobre la clasificación de las galaxias.	23
3.8. Descripción del árbol de decisión.	24
3.9. Ejemplo de imagen que proporciona Kaggle (424x424).	26
3.10. Ejemplo de imagen procesada (64x64).	26
4.1. Clasificación vs Regresión.[21]	31
4.2. <i>Overfitting</i> y <i>Underfitting</i> .[17]	33
4.3. Ejemplo de red neuronal.[19]	36
4.4. Perceptrón.[19]	37
4.5. Función Sigmoid.[19]	38
4.6. Función ReLU.[19]	38
4.7. Ejemplo de función de coste.[20]	39
4.8. Validación cruzada de K iteraciones.[23]	41
4.9. Validación aleatoria de K iteraciones.[23]	42

ÍNDICE DE FIGURAS

4.10. Validación cruzada dejando uno fuera.[23]	42
5.1. Matriz de confusión.[17]	47
5.2. Matriz de confusión de <i>naive</i> Bayes.	48
5.3. Matriz de confusión de la red neuronal.	48
5.4. Ejemplo de soluciones de la red neuronal.	49
5.5. Ejemplo de soluciones de <i>naive</i> Bayes.	50
5.6. Visualización de los pesos de las neuronas.	51

1

Introducción

El objetivo de este Trabajo Fin de Grado (TFG) es clasificar en diferentes tipos galaxias distantes del universo según su morfología mediante imágenes de telescopio. Para ello vamos a utilizar una familia de algoritmos llamada redes neuronales. Para trabajar con redes neuronales utilizaremos el módulo de Python *sklearn*, que nos permite usar herramientas de *machine learning*.

1.1. Motivación

Una de las preguntas fundamentales del ser humano es saber cómo y por qué estamos aquí. Existen galaxias de todas las formas y colores, desde galaxias circulares a espirales. Una pieza clave para poder responder a esta pregunta es comprender cómo la distribución y ubicación de las galaxias puede variar en función del tamaño, forma o color de éstas. [2]

Actualmente se tienen registradas aproximadamente cien mil millones de galaxias, y con el paso de los días, los telescopios de la Tierra, cada vez más potentes, obtienen más imágenes de galaxias distantes. Todas estas imágenes recogidas necesitan ser ordenadas y clasificadas, para comprender mejor cómo se relacionan las diferentes formas (o morfologías) de las galaxias con las físicas que las crean. [2]

¿Por qué usar redes neuronales? Existen otros algoritmos para clasificación como pueden ser modelos lineales, árboles de decisión, naive Bayes... En general, las redes neuronales sirven para resolver cualquier tipo de problema. Una de las principales ventajas de las redes neuronales es que funcionan muy bien con muchos datos (como es nuestro caso). Consiguen construir un modelo muy complejo, que necesita un tiempo de computación largo, y que obtiene muy buenos resultados. Sin embargo, los datos necesitan un procesado cuidadoso antes de poder usarlos en la red neuronal. Además, la elección de los parámetros de la red neuronal es también complicada. [18]

1.2. Objetivos

El objetivo principal es clasificar en diferentes tipos galaxias distantes del universo según su morfología utilizando imágenes de telescopio.

Para ello, vamos a utilizar técnicas de *machine learning*, concretamente, redes neuronales. Específicamente se pretende:

1. Reunir una base de imágenes relevante.
2. Procesar las imágenes.
3. Trabajar con estas imágenes utilizando algoritmos de *machine learning*, en concreto redes neuronales y *naive Bayes*.

1.3. Estructura de la memoria

La estructura del TFG es la siguiente:

- En el Capítulo 2 se realiza una introducción sobre cómo surge la necesidad de clasificar galaxias.

- En el Capítulo 3 se presentan los datos con los que se van a trabajar en este TFG. Hablaremos sobre cómo se captura una imagen de una galaxia, qué tipos de galaxias existen, cuáles serán las imágenes que utilizaremos y el preprocesado de las mismas.
- En el Capítulo 4 se exponen conceptos básicos sobre técnicas de *statistical learning* y *machine learning*, además de métodos de *machine learning* que vamos a utilizar durante el TFG.
- En el Capítulo 5 se exponen los resultados obtenidos con redes neuronales y *naive Bayes*.
- En el Capítulo 6 se finaliza el TFG con las conclusiones acerca de los resultados obtenidos y las líneas de investigación futuras.

2

Estado del Arte

En este capítulo vamos a realizar una introducción acerca de cómo surge el interés de clasificar las galaxias por su morfología.

2.1. Contexto del estudio

Galaxy Zoo

En julio de 2007, astrónomos de la universidad de Oxford tenían en sus bases de datos aproximadamente 1 millón de imágenes de galaxias, captadas por Sloan Digital Sky Survey (SDSS¹). Estas galaxias debían ser clasificadas por su morfología para comprender mejor los procesos galácticos. [1]

En este contexto surge la idea del proyecto Galaxy Zoo. La idea principal del proyecto es que cualquier persona pudiera ayudar a clasificar las galaxias, a través de su página web. Es decir, querían que el proceso de clasificación fuese el resultado de un trabajo colaborativo, desinteresado, de personas en todo el mundo.

¹SDSS es un proyecto de investigación del espacio mediante imágenes

En un principio, los fundadores del proyecto estimaron que llevaría años clasificar todas las imágenes. Sin embargo, quedaron sorprendidos con el recibimiento que tuvo. En el primer día, recibieron casi 70.000 clasificaciones por hora. Durante el primer año, recibieron más de 50 millones de clasificaciones, gracias a más de 150.000 personas. [1]

El proyecto está dividido en varias fases. En la primera fase, la tarea que se les pedía a los voluntarios era sencilla: dividir las galaxias en elípticas, fusiones y espirales, y si la galaxia era espiral, también se pedía la dirección de los brazos. Además, durante esta fase se fueron introduciendo nuevas imágenes de otros telescopios, como el de las islas de La Palma en Canarias, Gemini en Chile, Very Large Array en Nuevo México, etc. [1]

En la segunda fase, Galaxy Zoo 2, se introdujeron nuevas características sobre las galaxias para clasificar, como el número de brazos de una espiral, el tamaño de las protuberancias de las galaxias, etc. [1]

En las siguientes fases, Galaxy Zoo: Hubble y Galaxy Zoo: CANDELS, se siguen introduciendo nuevas imágenes captadas por el proyecto de Hubbles CANDLES (Cosmic Assembly Near-infrared Deep Extragalactic Legacy Survey). CANDELS hace uso de la nueva cámara instalada durante la misión final del transbordador a Hubble, para tomar imágenes ultra-profundas del universo. [1]

A continuación, vamos a ver un ejemplo de cómo se hace esta clasificación en la página web de Galaxy Zoo. [1]

En el primer paso, vemos en la Figura 2.1, nos piden seleccionar entre tres tipos posibles de formas. En este caso, seleccionamos *smooth*.

2.1. CONTEXTO DEL ESTUDIO

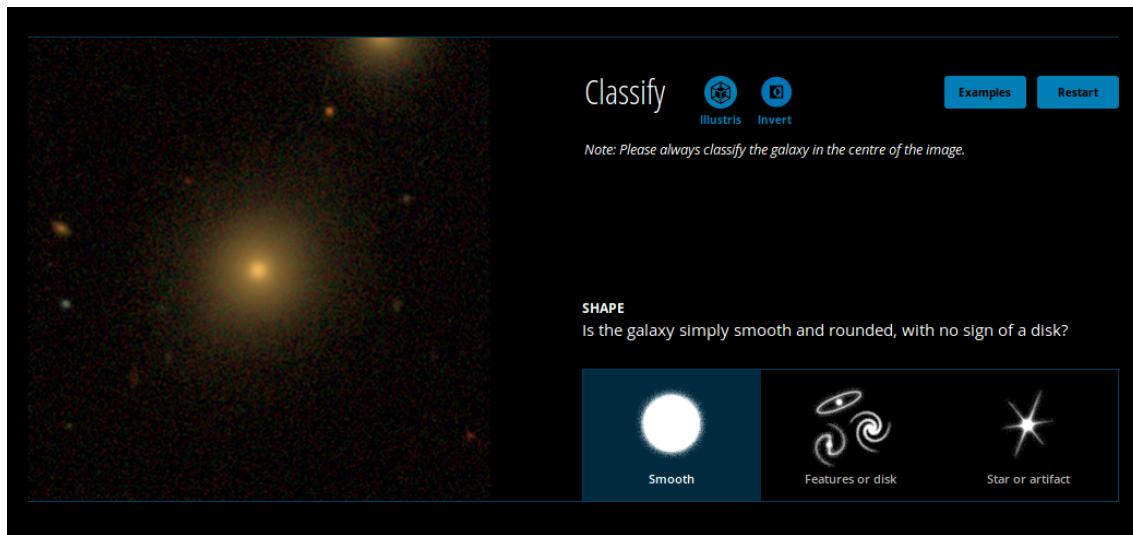


Figura 2.1: Paso 1 clasificación en Galaxy Zoo.[1]

Dependiendo de lo que seleccionemos en el primer paso, nos preguntarán sobre una característica u otra. En nuestro caso, como hemos seleccionado que la galaxia tiene una forma redondeada y suave, lo siguiente que nos preguntan es cómo de redonda es la galaxia. En la Figura 2.2, vemos que vamos a pulsar sobre “completamente redonda”.

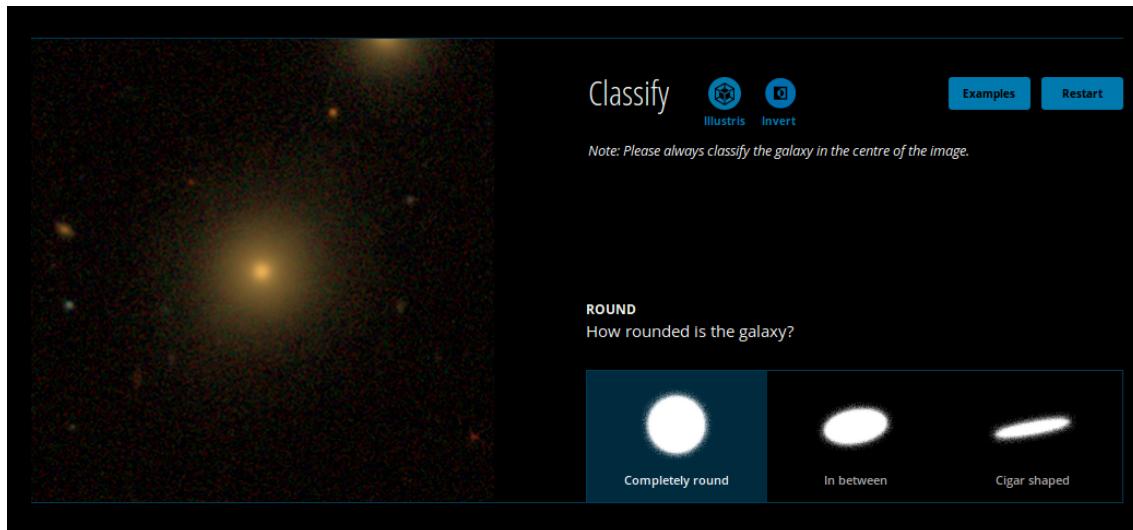


Figura 2.2: Paso 2 clasificación en Galaxy Zoo.[1]

A continuación, Figura 2.3, nos preguntan si hay algo extraño, alguna característica

que resulte llamativa. Pulsamos “no”.

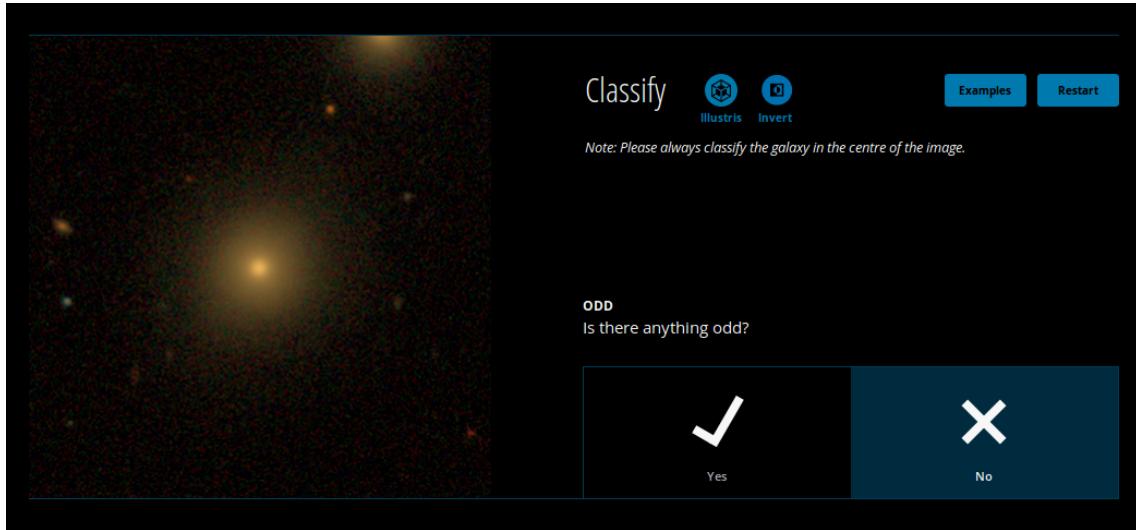


Figura 2.3: Paso 3 clasificación en Galaxy Zoo.[1]

Por último, Figura 2.4, preguntan si quieres discutir acerca de esta galaxia. Esta última pregunta la formulan para todas las galaxias. Si pulsas “sí”, se abre una especie de foro donde se discute más a fondo sobre esta galaxia.

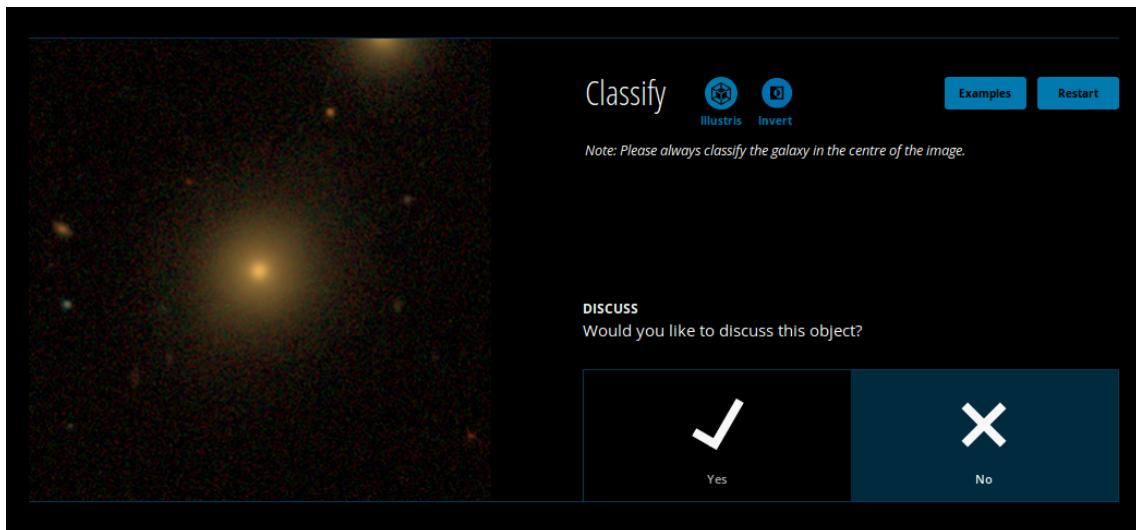


Figura 2.4: Paso 4 clasificación en Galaxy Zoo.[1]

Así termina la clasificación de esta galaxia. Si en algún momento piensas que te has

equivocado y quieres cambiar alguna de las características, puedes pulsar el botón *restart* para empezar de nuevo la clasificación de esa galaxia. También puedes invertir los colores de la imagen, que puede ayudar a ver alguna característica.

Kaggle

Kaggle es una plataforma fundada en 2010 para el aprendizaje de análisis de datos estadístico, *machine learning* y otros conceptos de *data science* con un enfoque práctico de esas habilidades en un entorno competitivo. Cualquiera puede competir por obtener el mejor modelo predictivo sobre un tema propuesto. [3]

Actualmente, Kaggle tiene registrado más de 540.000 usuarios. Es la comunidad más grande del mundo de datos. Los usuarios provienen de una gran variedad de campos: informática, biología, medicina, etc. Como hemos dicho, cualquier equipo o persona puede participar en las competiciones. Incluso han participado investigadores conocidos en el mundo, por ejemplo, miembros del equipo de IBM Watson². [28]

Las empresas pueden organizar una competición con distintos propósitos: para resolver problemas empresariales, para reclutamiento o para investigación. Las competiciones funcionan de la siguiente manera: [28]

1. La empresa que la organiza prepara una descripción del problema y los datos.
2. Los participantes compiten entre sí para conseguir el mejor modelo. El trabajo que se realiza es público, así que otros competidores pueden verlo para inspirar nuevas ideas. A medida que se entrega el trabajo, se actualizan las puntuaciones. Así, hay una clasificación en vivo de los participantes.
3. Una vez terminada la fecha límite, la empresa paga el premio a cambio de “una licencia mundial, perpetua, irrevocable y libre de royalties³ [...] para utilizar la entrada ganadora”. En caso de que la competición fuera para reclutamiento, los mejores participantes tendrán la oportunidad de hacer una entrevista con dicha empresa.

²Watson es una tecnología capaz de responder preguntas formuladas por humanos, gracias a una base de datos con información que proviene de multitud de fuentes. [https://es.wikipedia.org/wiki/Watson_\(inteligencia_artificial\)](https://es.wikipedia.org/wiki/Watson_(inteligencia_artificial))

³Derecho que hay que pagar al titular de una patente por utilizarla o explotarla comercialmente.

En diciembre de 2013, a raíz del proyecto Galaxy Zoo 2, dio comienzo una competición, en Kaggle, sobre este proyecto. El objetivo de la competición era obtener, un modelo que clasifique las galaxias de la misma manera que hemos visto anteriormente, pero que lo haga de forma automática. El método debería ser una modelo supervisado utilizando las clasificaciones humanas de las galaxias. Esta competición será la base de nuestro estudio.

3

Descripción de datos

En este capítulo vamos a definir los datos de nuestro TFG, desde cómo obtener imágenes de galaxias hasta qué procesado vamos a hacer a esas imágenes para poder utilizarlas en nuestro algoritmo.

3.1. Obtención de imágenes

Existen muchos telescopios situados alrededor del mundo. En concreto, Hubble es uno de los más famosos. Hubble orbita en el exterior de la atmósfera terrestre, a una altura de 600 km sobre el nivel del mar. De esta manera, se evita que las imágenes sean distorsionadas por la refracción atmosférica. Tampoco se ve afectado por factores meteorológicos o contaminación lumínica, como los telescopios terrestres, por lo que obtiene imágenes de mejor calidad. [32]

Hubble no utiliza filtros de color para crear las imágenes. Funciona como una cámara digital, utilizando un CCD (*charge-coupled device*) para registrar los fotones de luz entrantes. Se compone de espejos para reunir y llevar la luz a un foco donde se encuentran sus “ojos”. Tiene varios filtros para dejar entrar sólo un rango de longitud de onda específico de la luz. De esta manera, puede detectar la luz a través del espectro visible, la luz ultravioleta e infrarroja. [16, 12]

La luz detectada por los CCDs se convierte en señales digitales, que se almacenan en ordenadores de a bordo y se transmiten a la Tierra. Los científicos pueden entonces transformar los datos digitales en imágenes, por ejemplo, asignando la luz azul a los datos que entraron a través del filtro azul, la luz roja a los datos leídos a través del filtro rojo, etc. Estas imágenes son ya una versión mejorada, ya que la mayoría de los objetos celestes emiten colores muy débiles para que el ojo humano los pueda distinguir. Se necesita que la luz se acumule en el CCD a lo largo del tiempo para obtener tonos ricos. [16]

Sin embargo, el rango de longitudes de onda de la luz ultravioleta e infrarrojos no se corresponde con colores naturales. Cuando obtienen datos en estas longitudes de onda, los científicos asignan colores a los filtros de esa luz, de tal forma que esa luz se pareciera a los colores visibles por el ojo humano. [16]

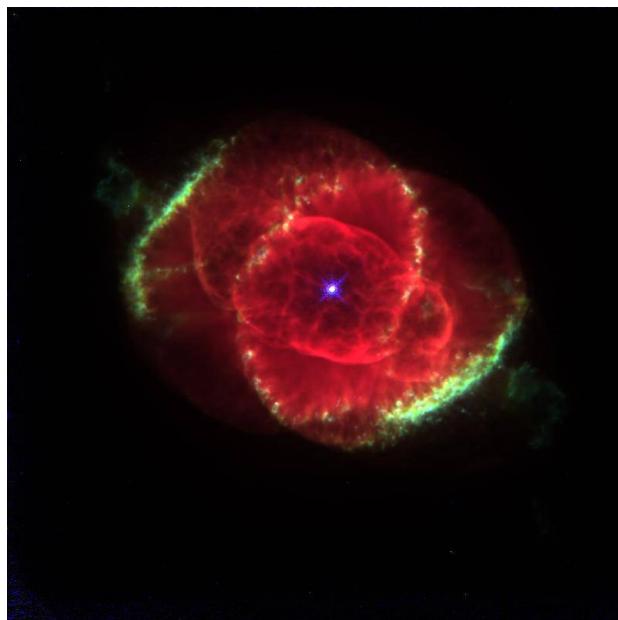


Figura 3.1: Nebulosa Ojo de Gato.[4]

Por ejemplo, como vemos en la Figura 3.1, la Nebulosa Ojo de Gato se fotografió a través de tres longitudes de onda de la luz roja muy difíciles de distinguir para los seres humanos, que corresponden con la radiación de los átomos del hidrógeno, de los átomos del oxígeno y de los iones del nitrógeno. Los científicos asignaron colores rojos, azules y verdes a los filtros y los combinaron para obtener esta imagen. [16]

3.2. Tipos de galaxias

El esquema de clasificación más utilizado para las galaxias se basa en uno diseñado por Edwin P. Hubble y perfeccionado por el astrónomo Gerard de Vaucouleurs. Este esquema divide las galaxias en cuatro clases, basadas en su morfología: elípticas, espirales, lenticulares e irregulares.

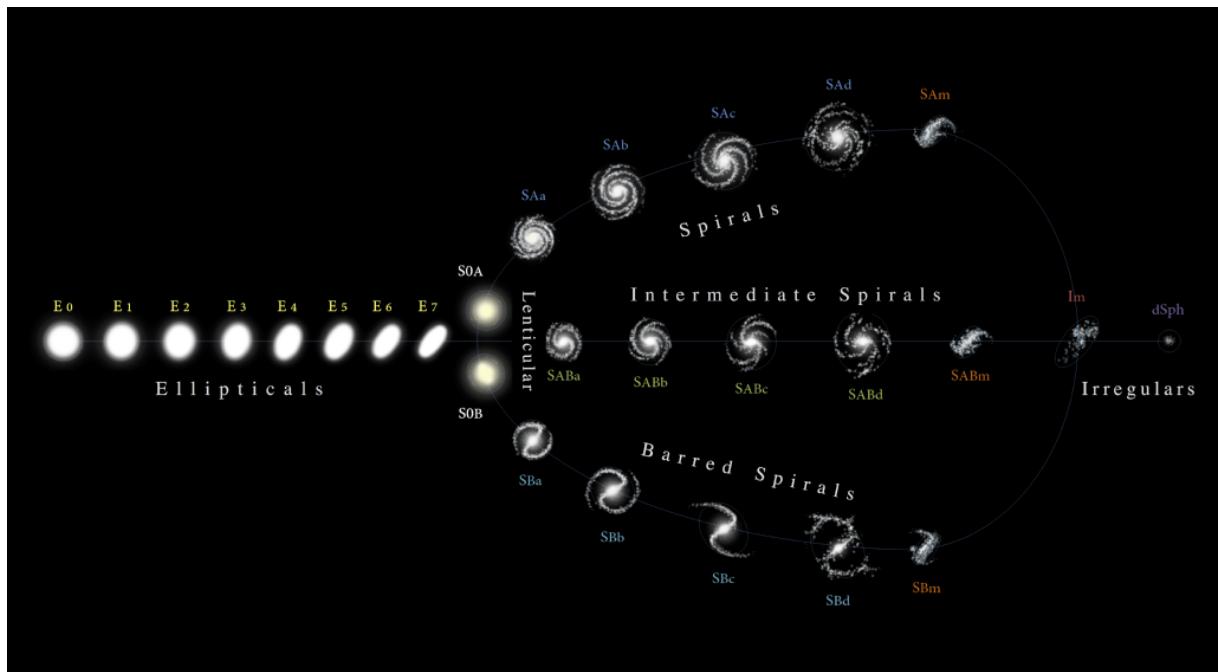


Figura 3.2: Diagrama de Morfología Galáctica Hubble - de Vaucouleurs. [25]

Elípticas

Las galaxias de esta clase tienen brillos suavemente variables, disminuyendo constantemente hacia fuera desde el centro. Tienen una forma más tridimensional y sin una estructura definida. Sus estrellas tienen órbitas aleatorias alrededor del centro. Contienen un agujero negro supermasivo en el centro. [15, 24]

Este tipo de galaxia, Figura 3.3, se crean a partir de la fusión de galaxias más pequeñas. El esquema de la Figura 3.2, sugiere que estas galaxias evolucionan en galaxias espirales. Se ha demostrado que esto es falso, ya que las estrellas que se encuentran en

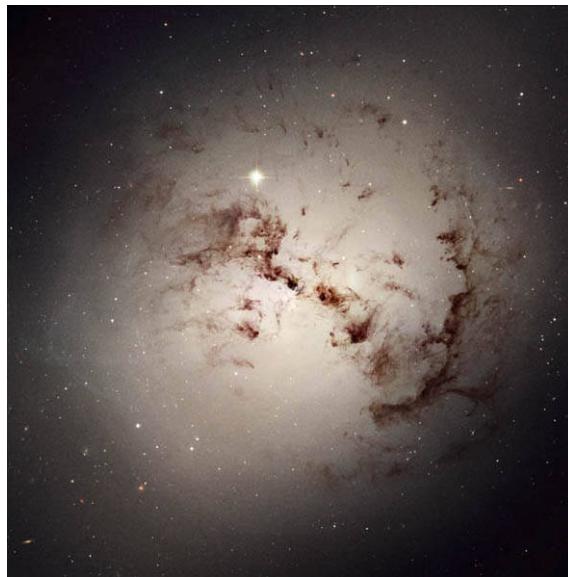


Figura 3.3: NGC 1316. Ejemplo de galaxia elíptica.[4]

galaxias elípticas son en promedio más antiguas que las que se encuentran en galaxias espirales.

Espirales

Son las más comunes del universo (como la Vía Láctea). Tienen tres componentes principales: una protuberancia esférica en el centro, que contiene las galaxias más antiguas, un disco que forma las estructuras de los brazos y contiene galaxias más jóvenes, y un halo alrededor de la protuberancia y del disco que contiene viejas estrellas. En la Figura 3.4 podemos ver un ejemplo de este tipo de galaxias. [15, 31]

El origen de este tipo de galaxias no está definido. Existen varias hipótesis acerca de ello. No vamos a entrar en detalles, ya que no es el fin de este trabajo¹.

Lenticulares

Es un tipo de galaxia intermedia entre una galaxia elíptica y una galaxia espiral (Figura 3.5). Se diferencian de las elípticas porque tienen una protuberancia en el centro,

¹https://en.wikipedia.org/wiki/Spiral_galaxy



Figura 3.4: Galaxia Remolino. Ejemplo de galaxia espiral.[4]

y son diferentes de las espirales porque no tienen una estructura en espiral. [15, 29]



Figura 3.5: Galaxia Sombrero. Ejemplo de galaxia lenticular.[4]

La formación de estas galaxias tiene también varias teorías. Unas sugieren que provienen de galaxias espirales desvanecidas, cuyos rasgos de brazos han desaparecido. Otras

proponen que la evolución de las galaxias lenticulares luminosas puede estar estrechamente vinculada a la de las galaxias elípticas, mientras que las lenticulares más débiles podrían estar más asociadas con las galaxias espirales.

Irregulares

Las galaxias de esta clase no tienen ninguna forma regular, ni algún rasgo característico o estructura. Las galaxias irregulares pueden contener abundantes cantidades de gas y polvo que bloquean la mayor parte de la luz de las estrellas. Todo este polvo hace que sea casi imposible distinguir estrellas en la galaxia (Figura 3.6). [15, 27]



Figura 3.6: NGC 55. Ejemplo de galaxia irregular.[4]

La teoría sobre la formación de este tipo de galaxias es que alguna vez estas galaxias fueron espirales o elípticas, pero fueron deformadas por una fuerza gravitatoria externa. [27]

Como hemos dicho al principio, es parte del ser humano entender la creación del universo. Clasificar las galaxias por sus tipos puede ser clave para lograrlo. Las galaxias lejanas nos dan información acerca del pasado. Pueden ayudar a los astrónomos a entender cómo la evolución de las galaxias ha cambiado a lo largo del tiempo, lo que proporciona una idea de cómo los procesos del universo han cambiado. De hecho, los astrónomos creen

3.3. DESCRIPCIÓN DE LA COMPETICIÓN KAGGLE

que el proceso que impulsa el crecimiento de las galaxias ha cambiado desde el inicio del universo. Las fusiones pueden haber sido el proceso dominante desde el principio, mientras que la acumulación de gas es mucho más común ahora, dando como resultado a menudo brazos de espirales. La comparación de galaxias de diferentes edades debería permitir a los astrónomos confirmar esta teoría de la evolución de las galaxias. [9]

Con la clasificación de las galaxias, los astrónomos podrán construir rápidamente estadísticas sobre cuántas galaxias tiene brazos, cuántas tiene protuberancias, etc., para poder constatar a todas estas preguntas. Estas estadísticas se pueden utilizar para construir un modelo de cómo las galaxias han evolucionado en el universo.

3.3. Descripción de la competición Kaggle

Como dijimos al principio del trabajo, vamos a tomar como base para el TFG la competición en Kaggle sobre Galaxy Zoo. En esta competición se ofrece un conjunto de imágenes de galaxias que ya han sido clasificadas por miles de voluntarios en el proyecto Galaxy Zoo. El problema surge cuando este conjunto de imágenes no deja de crecer, hasta contener miles de millones de galaxias.

En esta competición, se pide encontrar un método que, mediante las imágenes JPG de las galaxias, se reproduzcan las distribuciones de probabilidad derivadas de las clasificaciones humanas. Es decir, determinar para cada galaxia la probabilidad de que pertenezca a una clase en particular. [2]

La clasificación que se hace en la competición es la misma que la que sugieren en Galaxy Zoo. En la Figura 3.7, podemos ver el árbol de decisión completo.

La clasificación de cada galaxia se corresponde con una camino específico en el árbol de decisión. Todos los voluntarios clasificaron la misma galaxia, lo que significa que habrá múltiples caminos a lo largo del árbol de decisión para cada galaxia. Todas esas rutas generan para cada nodo probabilidades. Podemos ver mejor cada nodo en la Figura 3.8, que muestra una tabla con la descripción del árbol de decisión.

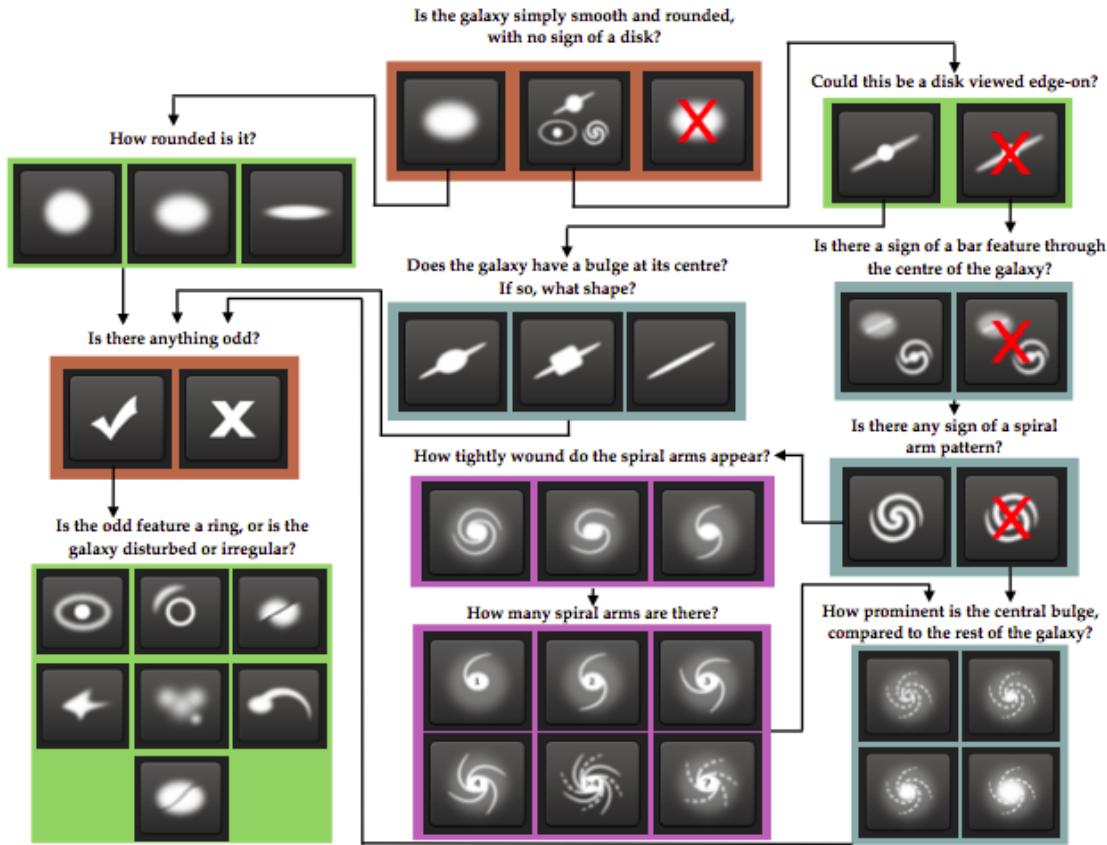


Figura 3.7: Árbol de decisión sobre la clasificación de las galaxias.

Cada clase tiene varios nodos. En la primera clase, podemos elegir redondeada, espiral o irregular, que se corresponden con las clases 1.1, 1.2 y 1.3 respectivamente (esta será la notación que se use para todo el árbol de decisión). Cada clase tendrá una probabilidad. Veámos un ejemplo: supongamos que para una determinada galaxia, el 80 % de usuarios determinó que era la clase 1.1, el 15 % la clase 1.2 y el 5 % la clase 1.3.

El 80 % de usuarios que determinó que la galaxia era elíptica, tuvieron que contestar si era completamente redondeada, elíptica o alargada (clase 7). Supongamos que los valores son 50 %, 25 % y 25 % respectivamente. Por tanto, las probabilidades para esa clase serán:

- Clase 7.1 = $80\% * 50\% = 40\%$
- Clase 7.2 = $80\% * 25\% = 20\%$
- Clase 7.3 = $80\% * 25\% = 20\%$

3.3. DESCRIPCIÓN DE LA COMPETICIÓN KAGGLE

Task	Question	Responses	Next
01	<i>Is the galaxy simply smooth and rounded, with no sign of a disk?</i>	smooth features or disk star or artifact	07 02 end
02	<i>Could this be a disk viewed edge-on?</i>	yes no	09 03
03	<i>Is there a sign of a bar feature through the centre of the galaxy?</i>	yes no	04 04
04	<i>Is there any sign of a spiral arm pattern?</i>	yes no	10 05
05	<i>How prominent is the central bulge, compared with the rest of the galaxy?</i>	no bulge just noticeable obvious dominant	06 06 06 06
06	<i>Is there anything odd?</i>	yes no	08 end
07	<i>How rounded is it?</i>	completely round in between cigar-shaped	06 06 06
08	<i>Is the odd feature a ring, or is the galaxy disturbed or irregular?</i>	ring lens or arc disturbed irregular other merger dust lane	end end end end end end end
09	<i>Does the galaxy have a bulge at its centre? If so, what shape?</i>	rounded boxy no bulge	06 06 06
10	<i>How tightly wound do the spiral arms appear?</i>	tight medium loose	11 11 11
11	<i>How many spiral arms are there?</i>	1 2 3 4 more than four can't tell	05 05 05 05 05 05

Figura 3.8: Descripción del árbol de decisión.

Este método de multiplicar todas las probabilidades se aplica durante todo el árbol de decisión. La suma de las clases 1.1-1.3 siempre será 1. Las clases 6.1 y 6.2 también se han normalizado para sumar 1.0, eliminando el efecto de elegir la clase 1.3 (irregular), ya que si se elige esta clase, termina la clasificación.

En total hay 37 clases, cada una con su probabilidad. La competición se centra en calcular esas probabilidades a partir del análisis de las imágenes, y conseguir que sean lo más parecidas a las probabilidades reproducidas por la clasificación manual de los voluntarios.

3.4. Datos del estudio

El problema original planteado en la competición de Kaggle representa un problema de clasificación supervisado de una complejidad elevada. Para poder abordar este problema en el contexto de un TFG, el primer paso fue intentar simplificarlo. El objetivo de la simplificación fue tener un modelo de datos para aplicar *machine learning*, lo suficientemente sencillo como para poder entender bien el problema que teníamos que resolver.

La simplificación propuesta fue convertir un problema de clasificación multiclasa desbalanceado (problema original propuesto) en un problema de clasificación en tres clases. De esta forma, el primer paso fue preprocessar la base datos y agrupar todas las galaxias en tres clases: clase 1 (redondeada), clase 2 (espiral), clase 3 (irregular). Esta nueva reclasificación se realizó de forma automática. Uno de los objetivos fue simplificar la tarea que tendría que resolver el algoritmo de *machine learning*, pues es más sencillo resolver un problema de clasificación en tres clases que uno de multiclassificación.

Después de realizar la reclasificación, observamos que a la clase 3, correspondían el 3% de los datos. Esto representaba una clara distorsión, planteando un problema terriblemente desbalanceado. Por lo que, en una primera aproximación, se decidió eliminar esta clase, quedando finalmente un problema de clasificación binaria con las clases: clase 1(redondeada) y clase 2(espiral).

El conjunto de imágenes ya clasificadas que nos proporciona Kaggle está compuesto por 61578 imágenes de galaxias en color, como la de la Figura 3.9. Las imágenes tienen un tamaño de 424x424 píxeles. Como hemos dicho, de este conjunto de imágenes eliminamos las que corresponden a la clase 3. De esta forma la base de datos de imágenes final tenía un total de 59619 elementos.

La siguiente parte del preprocessado fue pasar las imágenes de color a escala de grises, de esta forma se redujo el tamaño de las imágenes de 3x424x424 a 424x424. El objetivo de la reducción es reducir la dimensionalidad del espacio de entrada de características, de otra forma el algoritmo de aprendizaje tendría graves problemas para conseguir plantear modelos con alta generalización.

Después de varias pruebas con las imágenes en escalas de grises con tamaño 424x424, entrenando redes neuronales simples, se pudo observar que la carga computacional era

3.4. DATOS DEL ESTUDIO

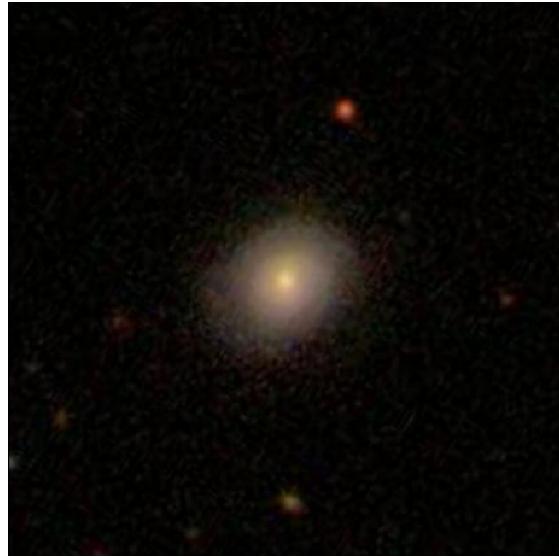


Figura 3.9: Ejemplo de imagen que proporciona Kaggle (424x424).

muy elevada, por lo que se probó a submuestrear la imagen, reduciendo su tamaño. Se probaron diferentes resoluciones, y finalmente se optó por un tamaño 64x64, de forma heurística (Figura 3.10).

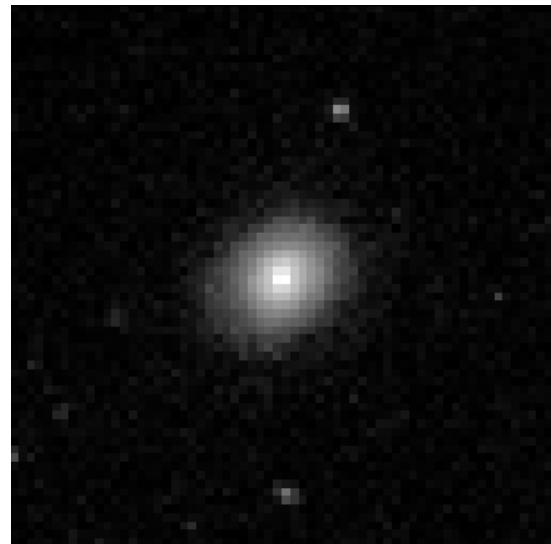


Figura 3.10: Ejemplo de imagen procesada (64x64).

4

Métodos

En este capítulo, vamos a exponer las técnicas de aprendizaje estadístico utilizadas en este TFG: redes neuronales y *naive Bayes*. Además, se hará una breve introducción sobre conceptos como *statistical learning* y *machine learning*.

4.1. Statistical Learning

El término *statistical learning* es relativamente nuevo, aunque muchos de los conceptos que abarcan este término se desarrollaron mucho antes. Podemos definirlo como “un subcampo de las matemáticas que trata de encontrar relaciones entre variables de entrada (variables independientes, características) para predecir los resultados (variable dependiente)”. [14, 22]

De manera general, supongamos que tenemos n variables independientes X_1, X_2, \dots, X_n , que tienen alguna relación con una variable dependiente Y , que la podemos escribir de la siguiente forma: [14]

$$Y = f(X) + \epsilon \quad (4.1)$$

f es una función de X_1, \dots, X_n y ϵ es un término de error. En esencia, *statistical*

learning consiste en estimar f . Existen dos razones principales por las que queremos estimar f : [14]

- Predicción: dado un conjunto de datos X , queremos estimar cuál es la salida Y .
- Inferencia: en este caso queremos estimar f no para predecir Y , sino para entender las relaciones entre X e Y , específicamente, como le afectan a Y los cambios en X_1, \dots, X_n . Podemos estar interesado en preguntas como ¿qué variables independientes X_1, \dots, X_n están asociadas con la respuesta Y ?; ¿cuál es la relación entre la respuesta Y y cada variable independiente X_1, \dots, X_n ?; etc.

4.2. Machine Learning

A raíz de conceptos como el anterior, surgen otras disciplinas como *machine learning*. Podemos definirlo como “un subcampo de la informática y de la inteligencia artificial que se ocupa de la construcción de sistemas que pueden aprender de los datos, en lugar de instrucciones explícitamente programadas”. *Machine learning* identifica patrones de los datos utilizando técnicas de *statistical learning*. [22]

En los últimos años, podemos encontrar aplicaciones de *machine learning* en cualquier ámbito, tantos como datos con los que contemos. Algunos ejemplos donde nos podemos encontrar con esta técnica en aplicaciones de detección del rostro (en nuestra cámara del móvil), sugerencias de búsqueda en el navegador, vehículos autónomos, análisis de datos económicos, etc. [13]

4.2.1. Tipos de problemas

En los inicios de la inteligencia artificial, la mayoría de las aplicaciones utilizaban reglas de decisión *if else* para procesar los datos. Este tipo de técnica puede funcionar para aplicaciones como el *spam* del correo, pero tiene principalmente dos desventajas: [18]

- La lógica requerida es única para cada aplicación. Cambiar ligeramente la tarea de la aplicación, puede significar cambiar toda la lógica.
- Diseñar la lógica requiere un estudio profundo del problema, de cómo un humano debería tomar la decisión.

Por ejemplo, con esta técnica no se pueden hacer aplicaciones como detección de la cara. Los ordenadores y los humanos perciben los píxeles que componen una imagen de forma muy diferente. Sin embargo, usando técnicas de *machine learning*, con una gran colección de imágenes podemos crear una algoritmo para determinar qué características son necesarias para detectar una cara. [18]

Possiblemente la parte más importante en el proceso de *machine learning* es entender los datos con los que se está trabajando. No todos los algoritmos funcionan de la misma manera. Son diferentes en términos de qué tipos de datos puede manejar, qué tipos de datos optimiza, etc. [18]

Antes de empear a construir el modelo es importante saber responder a preguntas como: [18]

- ¿Qué pregunta quiero responder? ¿Puedo responder a esa pregunta con este conjunto de datos?
- ¿Cuántos datos tengo? ¿Necesito más?
- ¿Cuántas características tengo? ¿Son suficientes?
- ¿Faltan datos? ¿Debo eliminar filas en las que faltan datos o las manejo de forma diferente?

Podemos dividir los problemas de *machine learning* en dos tipos: *supervised learning* y *unsupervised learning*.

4.2.1.1. Supervised Learning

Este tipo de problemas son los más comunes y los que más éxito tienen en *machine learning*. Consisten en automatizar procesos de decisión a partir de ejemplos ya conocidos.

Se aportan al algoritmo pares de entradas y salidas conocidas (se *entrena* el algoritmo), y el algoritmo encuentra de alguna manera producir salidas deseadas dada una entrada. [18, 14]

Es muy importante tener un conjunto de datos que hemos procesado y adaptado a nuestro problema específico. De esta manera, esta técnica será capaz de resolver el problema. [18]

Algunos ejemplos de este tipo de problemas pueden ser:

- **Reconocimiento de dígitos manuscritos.** Los datos de entrada serán las imágenes de los dígitos y la salida el dígito en cuestión. [18]
- **Determinar si un tumor es benigno o no mediante imágenes.** Los datos de entrada serían imágenes del tumor y la salida si el tumor es benigno o no. [18]
- **Como es en nuestro caso de estudio, determinar si una galaxia es de un tipo u otro.** Los datos de entrada son imágenes de galaxias y la salida el tipo de galaxia.

Clasificación y Regresión

Existen dos tipos principales de algoritmos supervisados de *machine learning*: clasificación y regresión.

Las variables pueden ser caracterizadas como cuantitativas o cualitativas (o categóricas). Las variables cuantitativas toman valores numéricos, como puede ser la edad, la altura, el precio de un producto, etc. Por otro lado, las variables cualitativas son aquellas que expresan distintas características o cualidades (clases). Variables de este tipo pueden ser el género de una persona (masculino o femenino), tipo de galaxia (elíptica, espiral, irregular), etc. [14]

En general, nos referimos a problemas con variables respuesta cuantitativas como problemas de regresión, y para problemas con variables respuesta cualitativa como problemas de clasificación. En la Figura 4.1 se puede ver la diferencia entre clasificación y regresión. [14]

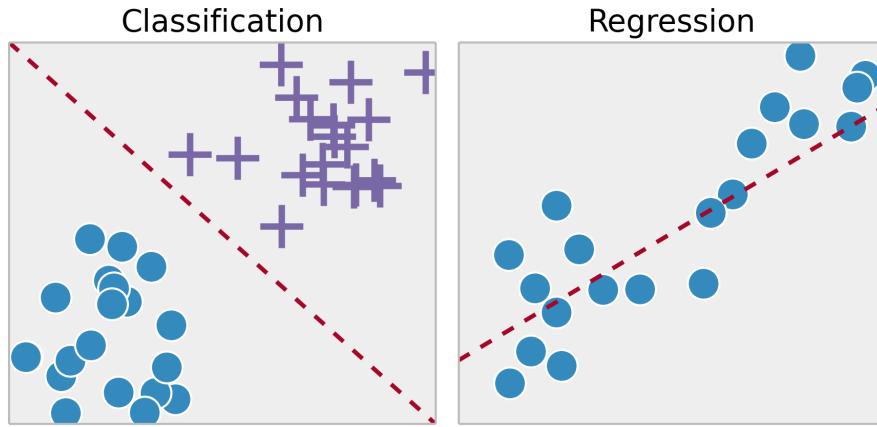


Figura 4.1: Clasificación vs Regresión.[21]

Los problemas de clasificación se suelen dividir entre clasificación binaria, cuando las variables solo tienen dos clases, o multiclasicación cuando tienen más de dos. A su vez, en la clasificación binaria se habla de una clase positiva y otra clase negativa. En este caso, positivo no representa beneficio, sino el objeto de estudio. En general, a una clase se le llama positiva de manera subjetiva, dependiendo del problema y del contexto del mismo. Por ejemplo, para el ejemplo del *spam* del correo, la clase positiva podría significar *spam*. [18]

En nuestro estudio, el problema original (de la competición de Kaggle) consistía en un problema de multiclasicación. Con las simplificaciones que hemos hecho, el problema se ha convertido en uno de clasificación binaria, donde se tendrá que decidir entre galaxias redondeadas o espirales.

4.2.1.2. Unsupervised Learning

En este otro tipo de problemas, solo conocemos el conjunto de datos de entrada. No conocemos un conjunto de datos de salida que se pueda entregar al algoritmo. Los métodos de estos problemas son en general más difíciles de entender y evaluar. Este tipo de problemas no tienen el objetivo de predecir, ya que no tenemos variables conocidas para hacerlo. Sin embargo, podemos buscar entender las relaciones entre las variables (inferencia). Una herramienta que se utiliza en este contexto es *cluster analysis*. El objetivo de *cluster analysis* es determinar si las observaciones se pueden dividir en grupos relativamente distintos. [18, 14]

Ejemplos de *unsupervised learning* pueden ser:

- **Identificar temas en un conjunto de publicaciones de blog.** Podemos tener como datos de entrada un conjunto de textos resumidos. No conocemos los temas de estos textos, por tanto no tenemos datos de salida. [18]
- **Separar clientes en grupos con preferencias similares.** Si tenemos clientes como un conjunto de datos, podemos querer identificar si hay clientes similares o grupos. No sabemos qué tipos de grupos puede haber, podrían ser “adultos”, “lectores”, “jugadores de videojuegos”. Pero no lo sabemos, así que no tenemos datos de salida. [18]

Una de las tareas en las que más tiempo hay que invertir es en la resolución de problemas con *machine learning* es el preprocesado (acondicionado) de los datos. En general, ayuda pensar que nuestros datos son como una tabla. Cada fila de la tabla representa las observaciones (cada imagen de la galaxia, cada cliente) y que cada columna representa una característica de esta observación (cada píxel de una imagen, edad, altura). [18]

4.3. Generalización, sobreajuste y sobregeneralización

A continuación vamos a definir algunos conceptos importantes de *machine learning*. Como ya hemos dicho, en *machine learning* entrenamos nuestro algoritmo con un conjunto de datos (datos de entrenamiento). El objetivo es que una vez entrenado el algoritmo, se logre obtener buenas predicciones con cualquier dato de entrada que no se haya visto nunca antes (datos de test). Generalización se refiere a la capacidad del método de *machine learning* para realizar predicciones acertadas en los datos de test (en realidad, en datos nuevos nunca vistos antes) una vez entrenado el algoritmo. Conociendo este concepto, pueden surgir dos problemas: [10, 17]

- El algoritmo obtiene muy buenas predicciones con el conjunto de datos de entrenamiento, pero no consigue buenas predicciones con el conjunto de datos de test. En este caso, diremos que el algoritmo sobreajusta (*overfitting*).

- El algoritmo no logra buenas predicciones con los datos de entrenamiento, por lo que no se espera que consiga buenos resultados con los datos de test. En este caso, diremos que el algoritmo sobregeneraliza (*underfitting*).

En la Figura 4.2, podemos ver un ejemplo tanto en regresión como en clasificación de estos dos conceptos.

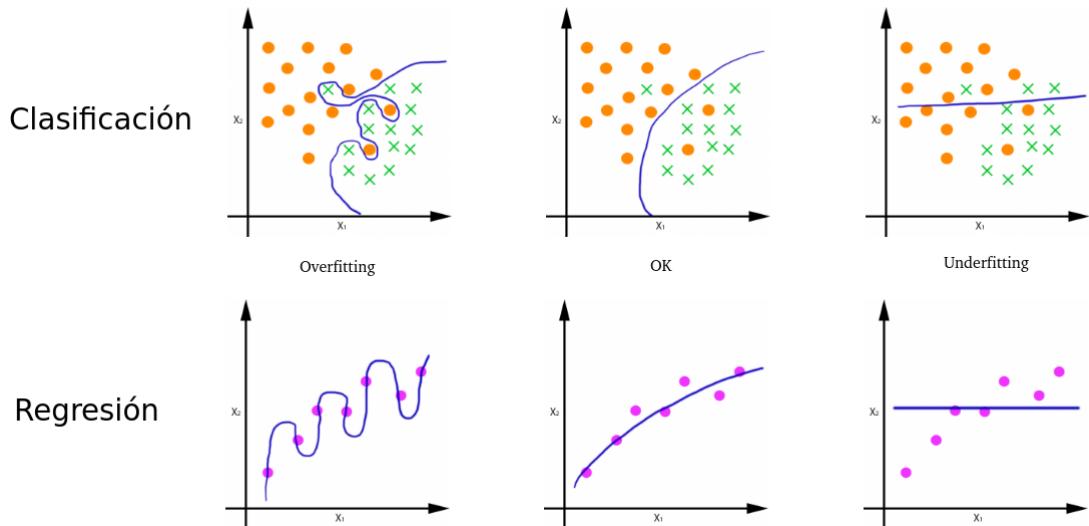


Figura 4.2: *Overfitting* y *Underfitting*.[17]

4.4. Naive Bayes

Los clasificadores *naive* Bayes son una familia de clasificadores que derivan de los modelos lineales de clasificación. [18]

Los modelos lineales de clasificación utilizan una función lineal con las características de entrada para obtener la predicción. La fórmula general para este tipo de modelo es: [18]

$$\hat{y}(x) = \underline{w}^T \underline{x} \quad (4.2)$$

En esta ecuación, \underline{x} es una matriz con las características de los datos de entrada. \underline{w}^T equivale a los pesos que el modelo “aprende” durante el entrenamiento del algoritmo. Estos pesos representan cómo de importante es cada característica en el modelo.

$$\underline{w}^T = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_n \end{bmatrix} \quad \underline{x} = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} \quad (4.3)$$

En concreto para clasificación binaria, una vez obtenidos los pesos, en realidad lo que queremos es que para cada dato de nuestro conjunto, decidir si es de una clase u otra. Para eso, le pasamos ésta fórmula como parámetro de una función.

$$\hat{y}(x) = f(\underline{w}^T \underline{x}) \quad (4.4)$$

La función f es conocida como función de activación. Imaginemos que en un problema de clasificación tenemos dos clases: la clase positiva “es *spam*”, y la clase negativa “no es *spam*”. Esta función nos devuelve, dado un valor de la entrada X_i (en este ejemplo sería un correo), cuál es la probabilidad de que pertenezca a la clase positiva, “es *spam*”. Si la probabilidad es mayor de 0.5, se decide que el correo es *spam*, y si es menor, no es *spam*. Más adelante veremos algunos tipos de funciones de activación en redes neuronales.

Los métodos *naive Bayes* se basan en el teorema de Bayes y en la suposición “*naive*” de independencia entre las características de los datos. Dada la variable dependiente de clase y y la matriz de características \underline{x} , veamos la relación que establece el teorema de Bayes: [8, 30]

$$P(y|\underline{x}) = P(y|x_1, \dots, x_n) = \frac{P(y)P(x_1, \dots, x_n|y)}{P(x_1, \dots, x_n)} \quad (4.5)$$

En la práctica, el denominador es constante, ya que conocemos los datos de entrada. El numerador podemos reescribirlo aplicando repetidamente la definición de probabilidad condicional:

$$\begin{aligned}
 P(y)P(x_1, \dots, x_n|y) &= \\
 P(y)P(x_1|y)P(x_2, \dots, x_n|y, z_1) &= \\
 P(y)P(x_1|y)P(x_2|y, x_1)P(x_3, \dots, x_n|y, x_1, x_2) \dots
 \end{aligned} \tag{4.6}$$

Ahora es cuando asumimos que cada x_i es independiente de cualquier x_j , siendo $j \neq i$. Esto significa que:

$$P(x_i|y, x_j) = P(x_i|y) \tag{4.7}$$

Por lo que podemos expresar el numerador como:

$$P(y)P(x_1|y)P(x_2|y)\dots P(x_n|y) = P(y) \prod_{i=1}^n P(x_i|y). \tag{4.8}$$

Finalmente, la expresión queda así:

$$P(y|x_1, \dots, x_n) \propto P(y) \prod_{i=1}^n P(x_i|y) \tag{4.9}$$

Podemos utilizar un estimador Maximum a Posteriori (MAP) para obtener la salida \hat{y} :

$$\hat{y} = \arg \max_y P(y) \prod_{i=1}^n P(x_i|y) \tag{4.10}$$

Existen tres tipos de clasificadores *naive* Bayes: Gaussiano, Multinomial y Bernoulli. Se diferencian principalmente por las suposiciones que hacen respecto a la distribución de $P(x_i|y)$. El clasificador Gaussiano se puede utilizar con datos continuos, en el Multinomial cada característica es un número entero y en Bernoulli asume datos binarios. En nuestro caso utilizaremos el clasificador Gaussiano. [18]

En general, los clasificadores *naive* Bayes son más rápidos en la etapa de entrenamiento que los clasificadores lineales convencionales y funcionan muy bien con grandes

cantidades de datos dispersos. Sin embargo, obtienen ligeramente peores resultados que los modelos lineales de clasificación convencionales. [18]

4.5. Redes neuronales

La idea original de las redes neuronales era imitar el funcionamiento del cerebro humano en un ordenador. Dr. Robert Hecht-Nielsen fue el inventor de la primera red neuronal y lo define como “*...a computing system made up of a number of simple, highly interconnected processing elements, which process information by their dynamic state response to external inputs.*”. [11]

La red neuronal está compuesta por varias capas de nodos o neuronas, conectadas entre sí, como vemos en la Figura 4.3. La primera capa se denomina capa de entrada. Cada neurona de esta capa se corresponde con las características de los datos de entrada del modelo (en nuestro caso, cada neurona se corresponde con un pixel de la imagen). La última capa se denomina capa de salida. Esta última capa genera la salida final de la red neuronal. Entre medias, podemos tener varias capas llamadas capas ocultas. En estas capas es donde se produce el procesamiento real de los datos. Para entenderlo mejor, veámos cómo funciona una neurona. [19]

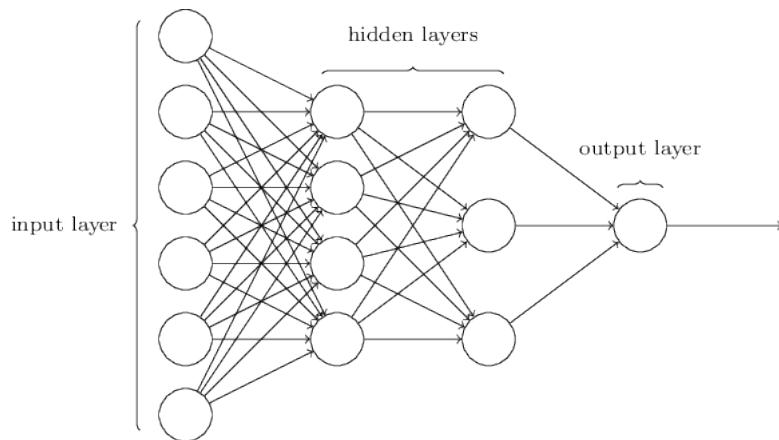


Figura 4.3: Ejemplo de red neuronal.[19]

Las neuronas o perceptrones (Figura 4.4) reciben varias entradas $x_1, x_2\dots$ y producen

una salida binaria. Cada entrada está asociada con un peso $w_1, w_2\dots$, que representa la importancia de esa entrada. [19]

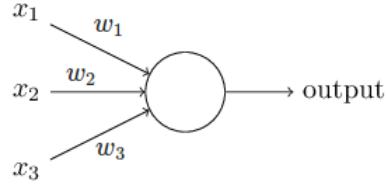


Figura 4.4: Perceptrón.[19]

La salida de cada neurona se calcula de la siguiente manera: si la suma ponderada de las entradas es mayor que un umbral, la salida será 1. En caso contrario será 0. A menudo, se suele usar otro término en lugar del umbral denominado *bias*, $b \equiv -\text{umbral}$. Podemos entender por *bias* una medida de cómo de fácil se activa (su salida es 1) una neurona. Si el valor de *bias* es muy grande, la neurona se activa muy fácilmente. Y al revés, si el valor de *bias* es muy pequeño, será difícil que la neurona se active. [19]

$$\text{salida} = \begin{cases} 1 & \text{si } \underline{w}^T \underline{x} + b > 0 \\ 0 & \text{si } \underline{w}^T \underline{x} + b \leq 0 \end{cases} \quad (4.11)$$

De la misma manera que antes, \underline{x} es una matriz con las características de los datos de entrada y \underline{w}^T equivale a los pesos del modelo.

Anteriormente, hemos hablado sobre la función de activación. Vamos a ver dos tipos de funciones de activación que serán utilizadas durante este TFG: Sigmoid y ReLU (*rectified linear unit*). Supongamos para ambos casos que $z = \underline{w}^T \underline{x} + b$. La función de activación Sigmoid (Figura 4.5) tiene la siguiente forma:

$$f(z) = \frac{1}{1 + e^{-z}} \quad (4.12)$$

Y la función ReLU (Figura 4.6):

$$f(z) = \begin{cases} 0 & \text{si } z < 0 \\ z & \text{si } z \geq 0 \end{cases} \quad (4.13)$$

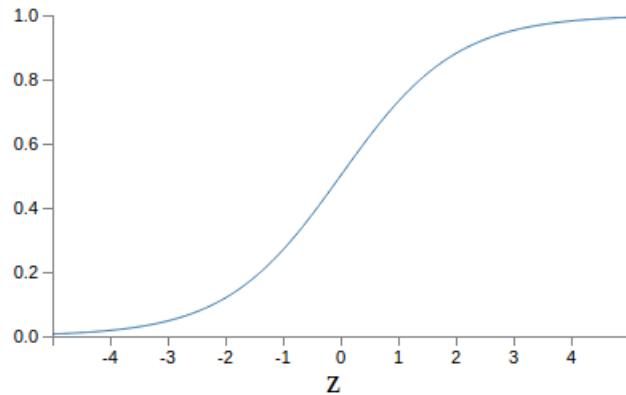


Figura 4.5: Función Sigmoid.[19]

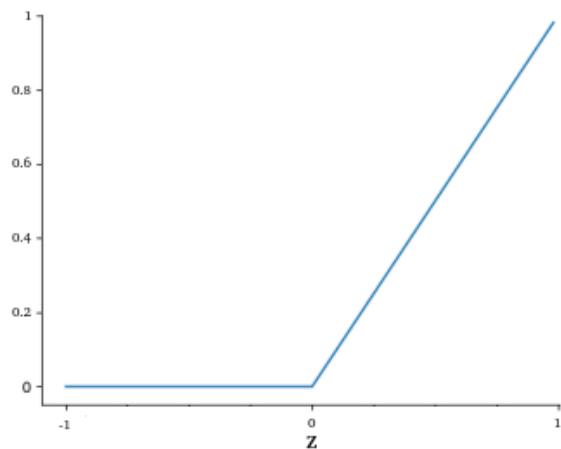


Figura 4.6: Función ReLU.[19]

En ambos casos, $f(z)$ nos devuelve la probabilidad de que la neurona se active o no. La expresión de salida queda de la siguiente forma:

$$\text{salida} = \begin{cases} 1 & \text{si } f(z) > 0,5 \\ 0 & \text{si } f(z) \leq 0,5 \end{cases}; \quad z = \underline{w}^T \underline{x} + b \quad (4.14)$$

4.5.1. Descenso por Gradiente

En esencia, lo que queremos del algoritmo es calcular los pesos \underline{w} y los *biases* para obtener la salida $y(x)$ para cualquier dato de entrada. Vamos a definir una función de coste J para cuantificar cómo de bien logramos este objetivo: [19]

$$J(w) = \|y(x) - \hat{y}(x)\|^2 = \|y - (\underline{w}^T \underline{x} + b)\|^2 \quad (4.15)$$

El propósito del algoritmo de entrenamiento será encontrar los pesos \underline{w} y *biases* para minimizar este coste $J(w)$. Para ello, se utiliza un algoritmo conocido como descenso por gradiente (*gradient descent*). [19]

Supongamos que tenemos una función de coste $J(w)$ como la de la Figura 4.7, queremos encontrar $J_{min}(w)$. Partimos de un punto cualquiera de la función, un peso cualquiera w_0 . Calculamos el gradiente¹ ($\nabla J(w_0)$) para ese punto de la función y actualizamos el siguiente peso w_1 . [20]

$$w_1 = w_0 - \eta \nabla J(w_0) \quad (4.16)$$

Como lo que queremos es el mínimo, ponemos el gradiente negativo para obtener la dirección opuesta. Este proceso se repite hasta que converger al mínimo deseado. η es lo que se denomina *learning rate*; se puede definir como la velocidad a la que se actualizan los pesos w_j . Este valor es muy importante, ya que con un *learning rate* muy pequeño puede llevar mucho tiempo en calcular el mínimo, y un *learning rate* muy grande puede provocar que nunca converja en un mínimo. [20]

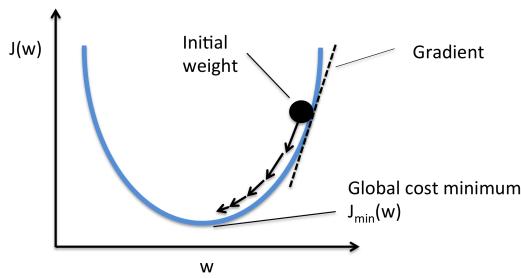


Figura 4.7: Ejemplo de función de coste.[20]

¹Recordemos que el gradiente es un vector que indica la dirección de máxima variación de una magnitud

Para poder explicar el descenso por gradiente, hemos escogido una función de coste muy simple. En general, las funciones de coste no son así de simples. Pueden tener mínimos “locales”, que puede complicar mucho el cálculo del mínimo absoluto de la función. Además, pueden ser funciones con más variable (más dimensiones) lo que complica aún más el problema. De hecho, en redes neuronales es muy habitual tener muchas dimensiones.

Para hacernos una idea de la dificultad del problema, recordemos que cada característica del problema de *machine learning* es una variable. En nuestro caso, las características serán los píxeles, que con imágenes de 64x64 son 4096 píxeles o variables. Así que nuestra función de coste tendrá 4096 dimensiones. Como se pueden imaginar, esto necesita de una carga computacional tremenda.

4.6. Validación cruzada

Uno de los mayores problemas cuando hablamos de técnicas de *machine learning* es el sobreajuste. La validación cruzada (*cross-validation, CV*) es una técnica que consiste en dividir múltiples veces los datos de entrenamiento a su vez en otro subconjunto de datos de entrenamiento y evaluar el análisis en un subconjunto de datos de test. En cada división se escoge un subconjunto de entrenamiento y de test distinto. De esta manera, tenemos mucha más información para ajustar el modelo, aunque requiere mucha carga computacional. [23]

Existen varias formas de afrontar esta técnica, que vamos a ver a continuación.

4.6.1. Validación cruzada de K iteraciones (*K-fold cross-validation*)

Este método consiste en dividir el conjunto de datos de entrenamiento de K subconjuntos de la misma proporción. En cada iteración, se elige un subconjunto de datos de entrenamiento diferente k , y se valida sobre el subconjunto de test $k-1$ (Figura 4.8). [23]

En cada iteración se calcula el error (*mean square error, MSE*) para evaluar las diferentes validaciones. El resultado final se obtiene realizando la media aritmética de cada

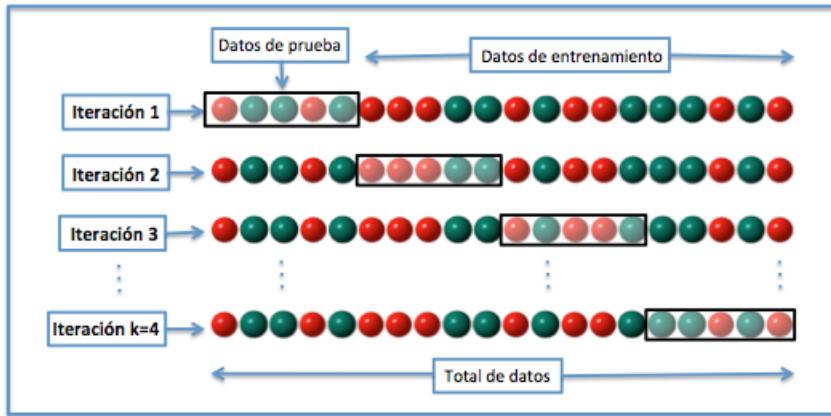


Figura 4.8: Validación cruzada de K iteraciones.[23]

MSE en las distintas iteraciones. [23]

La principal ventaja de este método es que es muy preciso, ya que se consigue evaluar el modelo a partir de K combinaciones de datos de entrenamiento y de test. Sin embargo, el número de iteraciones está limitado, depende del volumen de datos que se tenga. [23]

4.6.2. Validación cruzada aleatoria

La idea de este método es la misma que el anterior, pero la elección del subconjunto de datos de entrenamiento y de test se hace de forma aleatoria (Figura 4.9). El resultado final se obtiene de la misma manera, calculando la media aritmética del error cometido en cada iteración. [23]

La ventaja de este método es que se pueden hacer tantas iteraciones como se quieran, el número de iteraciones no depende de la cantidad de datos que se tengan. Por el contrario, con este método puede haber muestras que se quedan sin evaluar y otras que se evalúen más de una vez. [23]

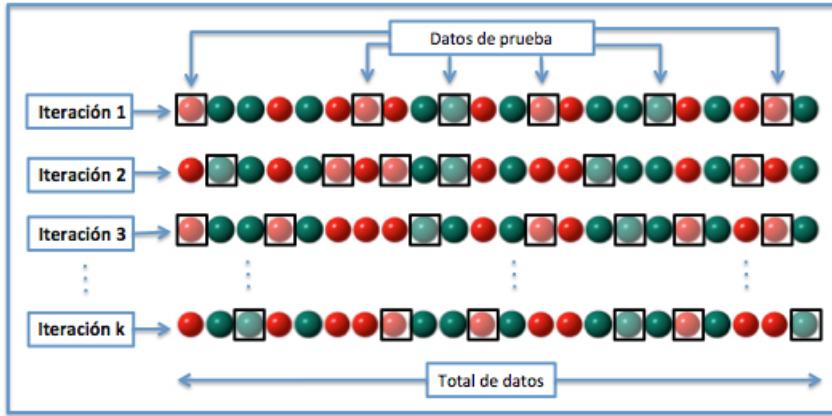


Figura 4.9: Validación aleatoria de K iteraciones.[23]

4.6.3. Validación cruzada dejando uno fuera (*Leave-one-out cross-validation, LOOCV*)

En este caso, los subconjuntos de datos de entrenamiento y de test se divide de tal manera que en cada iteración solo se tenga un dato de test y el resto datos de entrenamiento (Figura 4.10). El resultado final sigue siendo la media aritmética de cada iteración. [23]

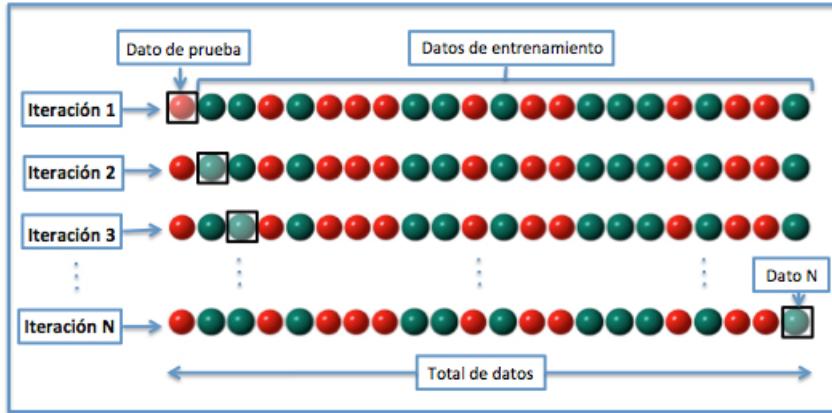


Figura 4.10: Validación cruzada dejando uno fuera.[23]

En este tipo de método, el error de la evaluación es muy bajo. A cambio, la carga computacional es muy alta, ya que tenemos que hacer tantas iteraciones como muestras tengamos. [23]

5

Análisis y resultados

En este capítulo, vamos a explicar las funciones utilizadas para poder utilizar redes neuronales y *naive* Bayes. Despues veremos los resultados obtenidos a través, principalmente, de la matriz de confusión.

5.1. Funciones de *Sklearn*

Para poder trabajar con herramientas de *machine learning*, vamos a utilizar el módulo de Python scikit-learn. Vamos a utilizar las funciones de MLPClassifier para redes neuronales y GaussianNB para *naive* Bayes.

La función GaussianNB es bastante sencilla de usar, ya que solo tiene un parámetro que se le puede modificar, *priors*: es un array con la probabilidades a priori de cada clase. En nuestro caso no vamos a modificar este parámetro. Por el preprocesado de los datos se ha terminado teniendo un conjunto de datos en los que las dos clases estaban balanceadas. De esta forma, no es necesario hacer ninguna asunción sobre las probabilidades a priori de las diferentes clases. [8]

La función para redes neuronales MLPClassifier si es más compleja, ya que tiene muchos parámetros libres, o hiperparámetros, que van a ser seleccionados utilizando

grid-search con validación cruzada. Vamos a ver los más importantes y los que vamos a modificar: [6]

- **hidden_layer_sizes:** es una tupla. La longitud de la tupla indica el número de capas ocultas de la red neuronal. El elemento i-ésimo representa el número de neuronas en la i-ésima capa oculta. Ejemplo: $hidden_layer_sizes = (4, 3)$ significa que la red neuronal tendrá 4 neuronas en la primera capa oculta y 3 neuronas en la segunda capa oculta.
- **activation:** {'identity', 'logistic', 'tanh', 'relu'}, indica la función de activación que se va a utilizar. Utilizaremos 'logistic' (Sigmoid) y 'relu'.
- **solver:** {'lbfgs', 'sgd', 'adam'}, indica el método que se usa para obtener los pesos w . Utilizaremos 'sgd', que es descenso por gradiente.
- **alpha:** es un número, término de regularización L2. Evita problemas de sobreajuste. [6]
- **learning_rate:** {'constant', 'invscaling', 'adaptive'}, indica cómo se calculan los nuevos pesos w_j durante el descenso por gradiente. Utilizaremos 'constant', cada peso se obtiene restandole una cantidad fija, determinada por **learning_rate_init**.
- **learning_rate_init:** es un número, que indica el valor del *learning rate*.

Uno de los objetivos es probar el modelo de redes neuronales con varias configuraciones para saber cuál es la mejor. *Sklearn* nos permite hacerlo con la función GridSearchCV. Esta función implementa una búsqueda de fuerza bruta de los mejores parámetros para un estimador. Para cada combinación de parámetros, utiliza además validación cruzada. Los parámetros de entrada de la función GridSearchCV se detallan a continuación: [5]

- **estimator:** es el objeto del estimador. En nuestro caso será el objeto creado por MLPClassifier.
- **param_grid:** es un diccionario, en el que se indican los hiperparámetros y los valores sobre los que se quieren hace la búsqueda.
- **n_jobs:** este tipo de búsquedas requieren de mucha carga computacional. Este parámetro permite repartir la búsqueda en diferentes núcleos del ordenador, paralelamente.
- **cv:** es un número, que indica el número de iteraciones que se harán durante la validación cruzada (se utiliza el método de validación cruzada de K iteraciones).

5.2. Configuración de los parámetros

El primer paso que tenemos que hacer es dividir nuestros datos en un conjunto datos de entrenamiento y un conjunto datos de test. Tenemos un total de 59619 imágenes de galaxias. Decidimos escoger de forma arbitraria que el 25 % de esas imágenes sean de test, de forma que terminamos con un conjunto de entrenamiento con un número suficiente de muestras para entrenar el algoritmo y un número de muestras en test suficiente para poder estimar adecuadamente el desempeño de nuestro modelos de predicción. En concreto, tenemos 44714 imágenes de entrenamiento y 14905 imágenes de test.

Una vez que ya tenemos los datos listos para usarlos, tenemos que configurar los parámetros que vamos a utilizar para GridSearchCV (solo para el caso de la red neuronal). Utilizaremos los siguientes parámetros:

- ***hidden_layer_sizes***. Vamos a probar las siguientes configuraciones: (2), (3), (3, 3), (4, 5, 3), (10, 10).
- ***activation***.. Utilizaremos las funciones de activación ‘relu’ y ‘logistic’.
- ***alpha***. Vamos a generar un vector de 70 valores equiespaciados en escala logarítmica, desde 10^{-4} hasta 10^4 .

```
alpha = np.logspace(-4, 4, 70)
```

- ***learning_rate_init***. Generamos un vector parecido al anterior, desde 10^{-2} hasta 10^3 .

```
learning_rate = np.logspace(-2, 3, 70)
```

De esta manera, nuestro diccionario de parámetros para ajustar el modelo será el siguiente:

```
tuned_parameters = { 'hidden_layer_sizes': [[2],[3],[3,3,3], [4,5,3],[10,10]],  
                    'learning_rate_init': learning_rate,  
                    'alpha': alpha,  
                    'activation': ['logistic', 'relu'] }
```

Ahora solo queda generar nuestro modelo. Un valor común de cv (iteraciones) para la validación cruzada suele ser 7 o 10, sin embargo, por razones de cómputo, decidimos utilizar un valor bajo. El tiempo de entrenamiento era muy elevado, incluso en una máquina con 32 núcleos y 126 Gb de RAM, el entrenamiento duró más de 2 días. $n_jobs = -1$ distribuye la computación entre todos los núcleos del procesador.

```
mlp = MLPClassifier(solver='sgd');
neural_network = GridSearchCV(mlp, param_grid=tuned_parameters,
                               cv=2, n_jobs = -1)
```

Generamos también el modelo de *naive Bayes*.

```
naive_bayes = GaussianNB()
```

Y por último, entrenamos ambos modelos con los datos de entrenamiento.

```
naural_network.fit(X_train, y_train)
naive_bayes.fit(X_train, y_train)
```

5.3. Matriz de confusión

Una vez que ya tenemos los modelos entrenados, pasamos a analizar los resultados. Todos los resultados que ofrecemos son calculados sobre el conjunto de test. Lo primero que podemos hacer es ver cuáles son los parámetros elegidos por GridSearchCV:

```
'activation': 'logistic',
'alpha': 5.6705239413811688,
'hidden_layer_sizes': [3],
'learning_rate_init': 0.074056846922624389
```

El parámetro más revelante es *hidden_layer_sizes*. La red neuronal elegida consiste en una sola capa oculta con 3 neuronas, no es muy compleja.

Por otro lado, la matriz de confusión es una herramienta muy útil para problemas de clasificación binaria. En una matriz de 2x2, como la de la Figura 5.1.

	TN	FP
negative class		
	FN	TP
positive class		
	predicted negative	predicted positive

Figura 5.1: Matriz de confusión.[17]

Elegimos como clase positiva que las galaxias sean espirales. Teniendo esto en cuenta, podemos definir cada celda de la matriz:

- **TN (true negative)**: se acierta en la predicción de la clase negativa, acertar que la galaxia es redondeada.
- **FP (false positive)**: se predice que es la clase positiva cuando en realidad es la clase negativa, predecir que la galaxia es espiral cuando en realidad es redondeada.
- **FN (false negative)**: se predice que es la clase negativa cuando en realidad es la clase positiva, predecir que la galaxia es redondeada cuando en realidad es espiral.
- **TP (true positive)**: se acierta en la predicción de la clase positiva, acertar que la galaxia es espiral.

Sabiendo los conceptos de la matriz de confusión, vamos a ver qué resultados nos salen. En las Figuras 5.2 y 5.3 tenemos las matrices de confusión de *naive Bayes* y de la red neuronal respectivamente.

Podemos sacar varias conclusiones:

1. Los resultados sobre acertar en la predicción de que la galaxia es redondeada (TN) son similares para ambos modelos.

5.3. MATRIZ DE CONFUSIÓN

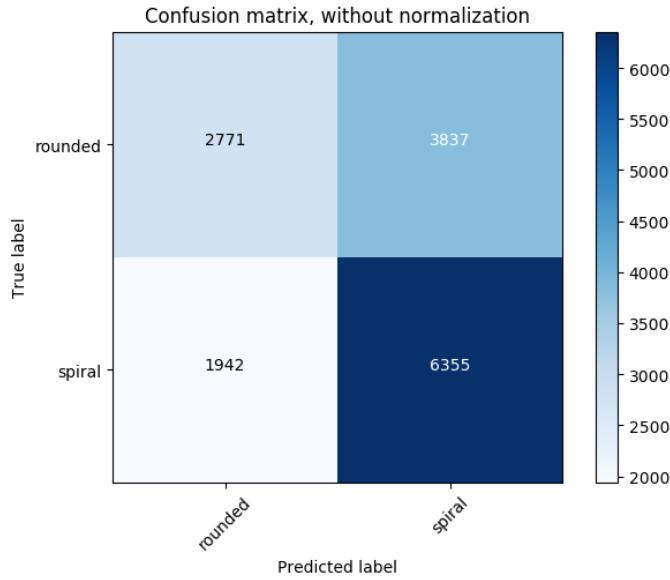


Figura 5.2: Matriz de confusión de *naive Bayes*.

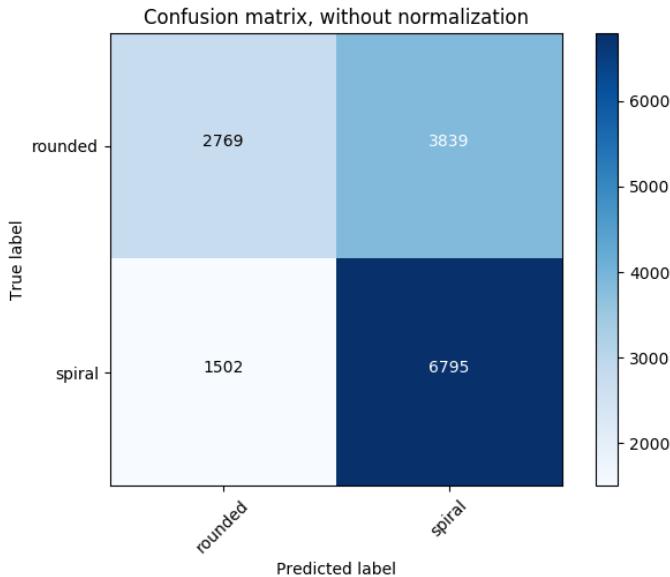


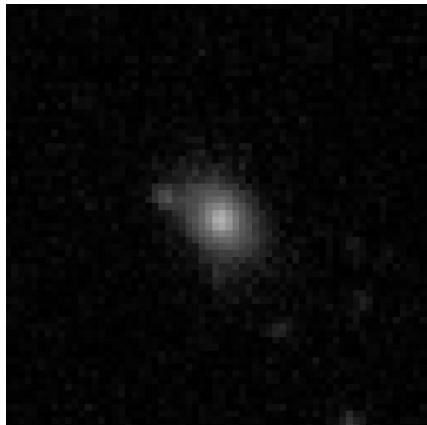
Figura 5.3: Matriz de confusión de la red neuronal.

2. Sin embargo, la red neuronal mejora los resultados respecto a *naive Bayes* de acertar en la predicción de que la galaxia es espiral (TP). Esto se debe a que *naive Bayes* es un modelo demasiado simple para detectar formas complicadas como una espiral.
3. Respecto a los fallos, ambos modelos tienen resultados parecidos cuando predicen

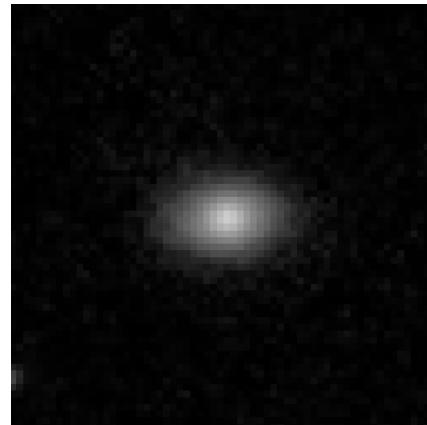
que la galaxia es espiral pero en realidad es redondeada (FP).

4. *Naive Bayes* obtiene peores resultados cuando la galaxia se trata de una espiral pero predice que es redondeada (FN), por la misma razón del punto 2.

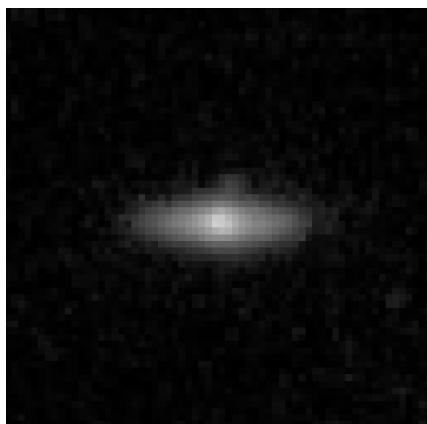
A continuación, podemos ver la solución de la red neuronal y *naive Bayes*, Figuras 5.4 y 5.5 respectivamente, sobre algunos ejemplos de imágenes. A simple vista, no podemos sacar una conclusión sobre por qué puede fallar el algoritmo.



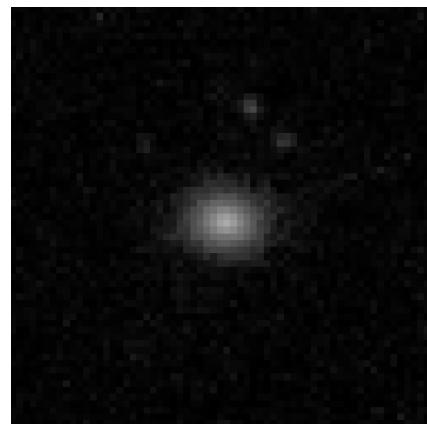
(a) Acierto espiral.



(b) Acierto redondeada.

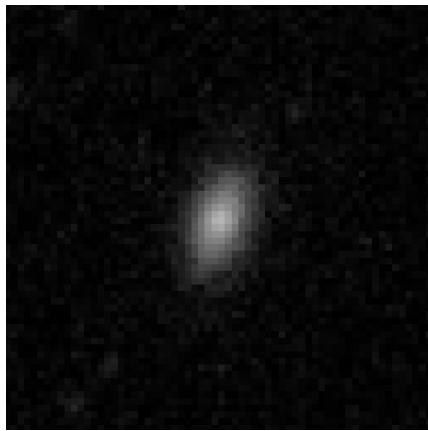


(c) Fallo espiral.

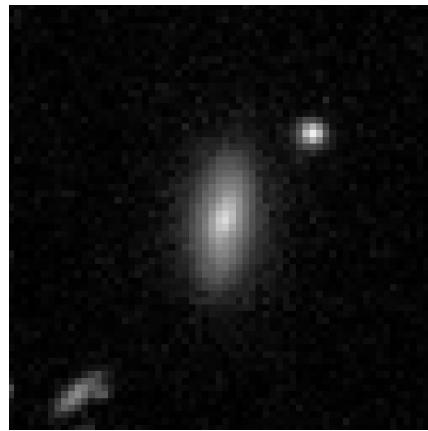


(d) Fallo redondeada.

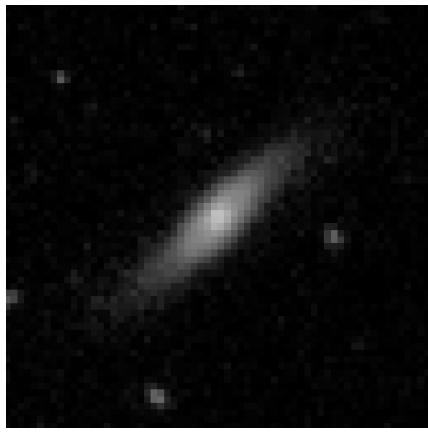
Figura 5.4: Ejemplo de soluciones de la red neuronal.



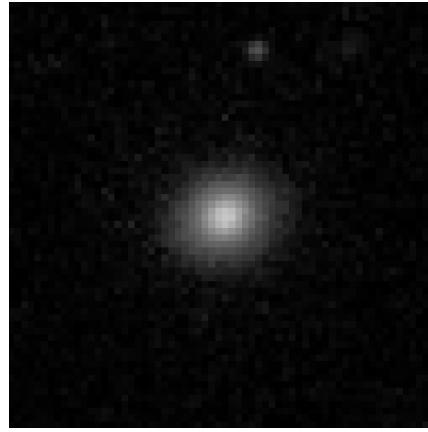
(a) Acierto espiral.



(b) Acierto redondeada.



(c) Fallo espiral.



(d) Fallo redondeada.

Figura 5.5: Ejemplo de soluciones de *naive Bayes*.

Existen varias métricas para evaluar un modelo de clasificación binaria: [7]

- ***Accuary score:*** precisión en las predicciones del modelo.
- ***Precision score:*** capacidad del clasificador de no predecir como clase positiva una muestra que es de la clase negativa. El mejor valor es 1 y el peor 0. Se calcula como:

$$\frac{TP}{TP + FP} \quad (5.1)$$

- ***Recall score:*** capacidad del clasificador para predecir todas las muestras de la clase positiva. El mejor valor es 1 y el peor 0. Se calcula como:

$$\frac{TP}{TP + FN} \quad (5.2)$$

- **F1 score:** se interpreta como una media ponderada entre *precision score* y *recall score*. El mejor valor es 1 y el peor 0. Se calcula como:

$$2 * \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}} \quad (5.3)$$

En la tabla 5.1, podemos ver estos resultados. Vemos que en general , el modelo de la red neuronal mejora a *naive Bayes*.

	Accuary	Precision	Recall	F1
Red neuronal	0.64166	0.64832	0.41903	0.50905
<i>naive Bayes</i>	0.61227	0.58794	0.41934	0.48953

Cuadro 5.1: Tabla de resultados

Por último, podemos visualizar los pesos en cada neurona, es decir, en que regiones de la imagen se fijan cada neurona. En la Figura 5.6, vemos que las dos primeras neuronas multiplican con mucha importancia el centro de la imagen (en blanco) y un poco el radio exterior (se aprecia una región más clara). Sin embargo, la tercera neurona multiplica por 0 lo que hay en el centro de la imagen y justo alrededor del centro hay una región más clara. Parece que la tercera neurona es la complementaria de las otras, e intentan mezclar esa información para dar el resultado.

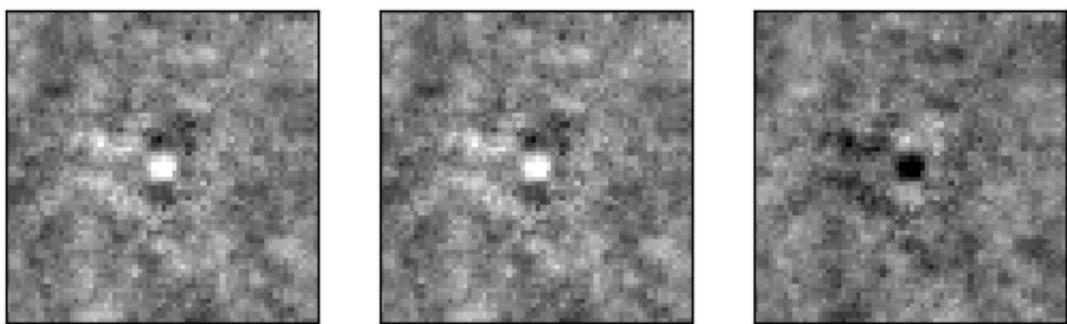


Figura 5.6: Visualización de los pesos de las neuronas.

6

Conclusiones y líneas futuras

Al principio del TFG se han expuesto una serie de objetivos específicos:

1. Reunir una base de imágenes relevante.
2. Procesar las imágenes.
3. Trabajar con estas imágenes utilizando algoritmos de *machine learning*, en concreto redes neuronales y *naive Bayes*.

Estos objetivos se han resuelto de forma satisfactoria: se ha conseguido obtener una base de datos interesante, procesarla en función de nuestros intereses y trabajar sobre los algoritmos de *machine learning*.

Sin embargo, el objetivo principal era conseguir clasificar las galaxias morfológicamente, y no se han obtenido muy buenos resultados. Con el mejor algoritmo, el de redes neuronales, la precisión de predecir bien a qué tipo de galaxia pertenece es del 64,16 %, un dato que se podría mejorar.

Anteriormente, comentamos que para poder abordar este problema en el contexto de un TFG, hemos tenido que simplificarlo mucho. La posibles líneas de trabajo futuras se enfocan en deshacer dichas simplificaciones.

En primer lugar, si nos centramos en redes neuronales, se podrían utilizar las imágenes a la resolución original (424x424) en blanco y negro. Este paso debería mejorar bastante el resultado, ya que al fin y al cabo, lo que se consigue con imágenes de más resolución, es añadir más características a la red neuronal.

En un siguiente paso, se podrían utilizar imágenes a color, lo que introduce aún más características a la red neuronal.

Por último, para abordar el problema original (de Kaggle), quedaría convertir el problema de clasificación binaria en uno de multiclasicación.

A partir de aquí, las nuevas evoluciones consistirían principalmente en encontrar nuevas características en las imágenes, para que la red neuronal tenga más información. Por ejemplo, las imágenes están centradas, por lo que se podría detectar formas circulares o elípticas. Se puede crear una circunferencia alrededor de la imagen con un determinado radio y calcular el error entre esa circunferencia y el perímetro de la galaxia. También se podrían buscar píxeles negros dentro del perímetro de la galaxia, y ver si siguen un determinado patrón (para diferenciar distintos tipos de espirales).

Todas estas evoluciones requieren de redes neuronales más complejas, con muchas más capas ocultas y neuronas (*deep learning*), lo que se traduce en mucha más carga computacional.

Todo el código utilizado en este TFG está subido en un repositorio de GitHub¹.

¹<https://github.com/adrioter94/TFG>

Bibliografía

- [1] Galaxy zoo. <https://www.galaxyzoo.org/>.
- [2] Kaggle, galaxy zoo. <https://www.kaggle.com/c/galaxy-zoo-the-galaxy-challenge>.
- [3] Kaggle, wiki. <https://www.kaggle.com/wiki/Home>.
- [4] Nasa. <https://www.nasa.gov/multimedia/imagegallery/index.html>.
- [5] Scikit-learn. gridsearchcv. http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html.
- [6] Scikit-learn. mlpclassifier. http://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html.
- [7] Scikit-learn. model evaluation. http://scikit-learn.org/stable/modules/model_evaluation.html.
- [8] Scikit-learn. naive bayes. http://scikit-learn.org/stable/modules/naive_bayes.html.
- [9] ANDREWS, K. Why classify far away galaxies? <https://www.galaxyexplorer.net.au/science/the-science>. [Online; galaxyexplorer.net.au].
- [10] BROWNLEE, J. Overfitting and underfitting with machine learning algorithms. <http://machinelearningmastery.com/>

- overfitting-and-underfitting-with-machine-learning-algorithms/. [Online; machinelearningmastery.com].
- [11] CAUDILL, M. *Neural Network Primer: Part I*. AI Expert, 1989.
- [12] CRAIG FREUDENRICH, P., AND GODDARD, S. How hubble space telescope works. <http://science.howstuffworks.com/hubble.htm>, Diciembre 2000. [Online; HowStuffWorks.com].
- [13] GARCÍA, V. F. ¿qué es machine learning y qué aplicaciones tiene en nuestro día a día? <http://www.intelygenz.es/que-es-machine-learning-y-que-aplicaciones-tiene-dia-a-dia/>. [Online; www.intelygenz.es].
- [14] JAMES, G., WITTEN, D., HASTIE, T., AND TIBSHIRANI, R. *An Introduction to Statistical Learning: with Applications in R*. Springer Science & Business Media, 2013.
- [15] McNALLY, E. Types and classification of galaxies. <http://www.astro.cornell.edu/academics/courses/astro201/galaxies/types.htm>, Abril 2000. [Online; astro.cornell.edu].
- [16] MOSKOWITZ, C. Truth behind the photos: What the hubble space telescope really sees. <https://www.space.com/8059-truth-photos-hubble-space-telescope-sees.html>, Marzo 2010. [Online; Space.com].
- [17] MOYA, R. ¿qué es el machine learning? <https://jarroba.com/que-es-el-machine-learning/>. [Online; jarroba.com].
- [18] MÜLLER, A. C., AND GUIDO, S. *Introduction to Machine Learning with Python: A Guide for Data Scientists*. O'Reilly Media, Inc., 2016.
- [19] NIELSEN, M. A. *Neural Networks and Deep Learning*. Determination Press, 2015.
- [20] RASCHKA, S. Machine learning faq. <https://sebastianraschka.com/faq/docs/closed-form-vs-gd.html>. [Online; sebastianraschka.com].
- [21] ROSSANT, C. *IPython Interactive Computing and Visualization Cookbook*. Packt Publishing Ltd, 2014.
- [22] SRIVASTAVA, T. Difference between machine learning and statistical modeling. <https://www.analyticsvidhya.com/blog/2015/07/difference-machine-learning-statistical-modeling/>. [Online; www.analyticsvidhya.com].

BIBLIOGRAFÍA

- [23] WIKIPEDIA. Validación cruzada. https://es.wikipedia.org/w/index.php?title=Validaci%C3%B3n_cruzada&oldid=94338594, Octubre 2016.
- [24] WIKIPEDIA. Elliptical galaxy. https://en.wikipedia.org/w/index.php?title=Elliptical_galaxy&oldid=782764818, Mayo 2017.
- [25] WIKIPEDIA. Galaxy morphological classification. https://en.wikipedia.org/w/index.php?title=Galaxy_morphological_classification&oldid=779567338, Junio 2017.
- [26] WIKIPEDIA. Galaxy zoo. https://en.wikipedia.org/w/index.php?title=Galaxy_Zoo&oldid=766008571, Junio 2017.
- [27] WIKIPEDIA. Irregular galaxy. https://en.wikipedia.org/w/index.php?title=Irregular_galaxy&oldid=786463308, Junio 2017.
- [28] WIKIPEDIA. Kaggle. <https://en.wikipedia.org/w/index.php?title=Kaggle&oldid=787479914>, Junio 2017.
- [29] WIKIPEDIA. Lenticular galaxy. https://en.wikipedia.org/w/index.php?title=Lenticular_galaxy&oldid=773801096, Abril 2017.
- [30] WIKIPEDIA. Naive bayes classifier. https://en.wikipedia.org/w/index.php?title=Naive_Bayes_classifier&oldid=772906859, Marzo 2017.
- [31] WIKIPEDIA. Spiral galaxy. https://en.wikipedia.org/w/index.php?title=Spiral_galaxy&oldid=784527624, Junio 2017.
- [32] WIKIPEDIA. Telescopio espacial hubble. https://es.wikipedia.org/w/index.php?title=Telescopio_espacial_Hubble&oldid=99905864, Junio 2017.