CATS vs DOGS

Autores: Kevin Cifuentes Salas
Luis Carlos Fernández San Martín

Índice

1. Resumen

El objetivo del presente proyecto es la aplicación de algoritmos de aprendizaje automático para resolver un problema de modelización predictiva planteado en la plataforma de competiciones analiticas Kaggle. En concreto, se trata de la competición CATS vs DOGS, que plantea diseñar un algoritmo que aprenda a clasificar el contenido de imágenes no etiquetadas de perros y gatos tomadas directamente de dicha plataforma, para determinar a cuál de estos dos animales pertenece cada una de ellas.
En los puntos que componen el proyecto, se explica brevemente en qué consiste la plataforma web Kaggle y su funcionamiento, se describe el algoritmo de aprendizaje automático (red neuronal) aplicado, así como los resultados finales que se han obtenido tras su implementación.

2. Introducción a Kaggle

Kaggle es una plataforma de competiciones de análisis y modelización predictiva de datos proporcionados por empresas e investigadores. Fue fundada en 2010 por el economista australiano Anthony Goldbloom, inspirándose en la competición organizada por la plataforma Netflix para mejorar su software de recomendación de películas. A día de hoy han tenido lugar más de 200 competiciones, con premios de hasta 3 millones de dólares. El funcionamiento de la plataforma es muy sencillo, un promotor prepara un conjunto de datos de su negocio o investigación. Una parte de estos datos son publicados en la web de Kaggle para que los concursantes desarrollen sus modelos predictivos, mientras que la otra parte se reserva de manera privada para permitir la clasificación de los concursantes en función del comportamiento de sus modelos y, por tanto, determinar el ganador de la competición. Los datos públicos contienen la variable que se necesita modelar (aprendizaje supervisado), y para valorar la bondad de las predicciones se publica también una métrica, es decir, una fórmula para medir el error. Finalmente, se comparan las predicciones sobre los datos privados, obteniendo así el poder predictivo de los modelos y, por tanto, estableciendo una tabla de posiciones en función del valor obtenido.

3. Descripción del problema

El hilo principal de la competición CATS vs DOGS se basa en diseñar un algoritmo que clasifique si una imagen contiene un perro o un gato. Por tanto, nos enfrentamos a un problema de clasificación binaria (sólo se permiten dos posibles clases) basado en aprendizaje supervisado. Es decir, nuestro objetivo es definir una función que establezca

una correspondencia entre las entradas y las salidas deseadas del sistema, donde la base de conocimientos del sistema está formada por ejemplos etiquetados, o dicho de otra forma, ejemplos de los que sabemos su clasificación correcta. Estos ejemplo etiquetados consisten en una base de 25000 imagenes de perros y gatos facilitada por la plataforma Kaggle. Disponer únicamente de imágenes, ha supuesto una dificultad añadida ya que hemos tenido que generar nuestro propio dataset a partir de los cálculos obtenidos como resultado de aplicar distintas técnicas de procesamiento de imágenes.

4. Solución planteada

Como ya se ha comentado anteriormente, la competición CATS vs DOGS plantea un problema de clasificación binaria y aunque podríamos haberlo abordado con otros métodos de clasificación como Naive Bayes o Random Forest, nos hemos decantado por utilizar una red neuronal para poder obtener el valor de pertenencia a cada tipo de animal. Es decir si los valores arrojados por el algoritmo se sitúan entre 0 y 0,5 indican que se trata de un gato, por el contrario, valores entre 0,5 y 1 indican que se trata de un perro. El resultado final obtenido después de ejecutar el test sobre la red neural, es una lista que contiene un componente animalType (valor de pertenencia) con la clasificación predicha para cada imagen analizada.

Para poder entrenar nuestra red neuronal hemos tenido que generar un dataset con los datos obtenidos como resultado de aplicar diferentes técnicas (Escalado, Edge detection, Dithering, Histéresis, Canny, etc..) de procesamiento de imágenes. A continuación, se muestra un listado de las características que hemos considerado relevantes para tratar de determinar si la imagen analizada pertenece a un perro o a un gato.

CARACTERÍSTICA	**DESCRIPCIÓN**
feature_mean_color	Valor resultante de calcular la media de los tres colores (red, green, blue) que componen la imagen.
feature_mean_red	Valor resultante de calcular la media del color rojo que tiene la imagen.
feature_mean_green	Valor resultante de calcular la media del color verde que tiene la imagen.
feature_mean_blue	Valor resultante de calcular la media del color azul que tiene la imagen.

feature_mean_gradient	Valor resultante de calcular la media del gradiente de la imagen.
feature_mean_gradient_channel1	Valor resultante de calcular la media del gradiente del canal 1 (red) de la imagen.
feature_mean_gradient_channel2	Valor resultante de calcular la media del gradiente del canal 2 (green) de la imagen.
feature_mean_gradient_channel3	Valor resultante de calcular la media del gradiente del canal 3 (blue) de la imagen.
feature_numberOfLittleObjects	Valor resultante de aplicar el determinante de Hessian a la imagen, para obtener el número de objetos pequeños en la misma.
feature_numberOfMediumObjects	Valor resultante de aplicar el determinante de Hessian a la imagen, para obtener el número de objetos de tamaño mediano en la misma.
feature_detectedStrongEdges	Valor resultante del proceso de Histéresis sobre la imagen, para obtener el número de bordes prominentes.
feature_detectedWeakEdges	Valor resultante del proceso de Histéresis sobre la imagen, para obtener el número de bordes suaves.
feature_numberOfStringPixelsForEdges	Valor que representa el número de píxeles que forman parte de un borde.

5. Interfaz gráfica

Con el objetivo de poder darle un uso más amigable a la posible solución que estamos implementando, hemos desarrollado un *frontend* que permita ejecutar la red neuronal sobre una imagen que el usuario seleccione a través de una interfaz web. Para este cometido se evaluaron distintas soluciones que permitieran utilizar el modelo, el cual está implementado en *R*: lenguaje de programación Python con GUI, Java con Swing… Finalmente, promovido en parte por recomendación del profesor, se ha implementado la parte visual utilizando una Shiny App, la cual también se implementa en el lenguaje de programación *R*. Esta manera de proceder era completamente desconocida para nosotros, con lo que llevó tiempo entender y programar este apartado (nótese que no tiene ninguna similitud aparente a otra lenguaje de programación utilizado previamente). A continuación, pueden verse una serie de imágenes de la solución visual implementada:

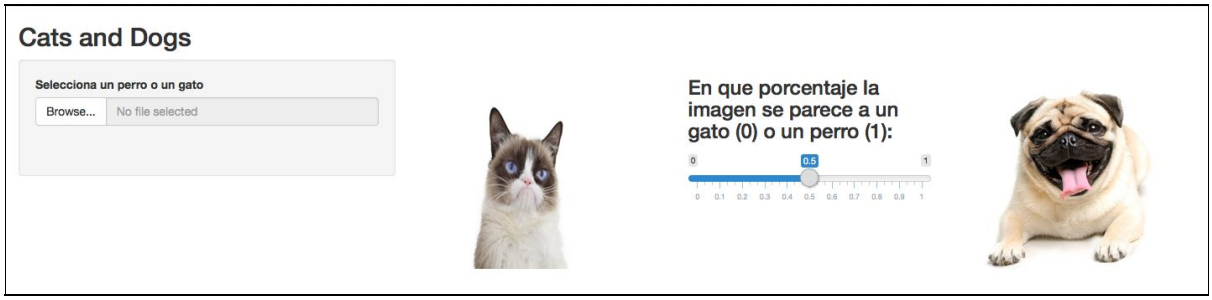

Figura 1: Interfaz visual por defecto

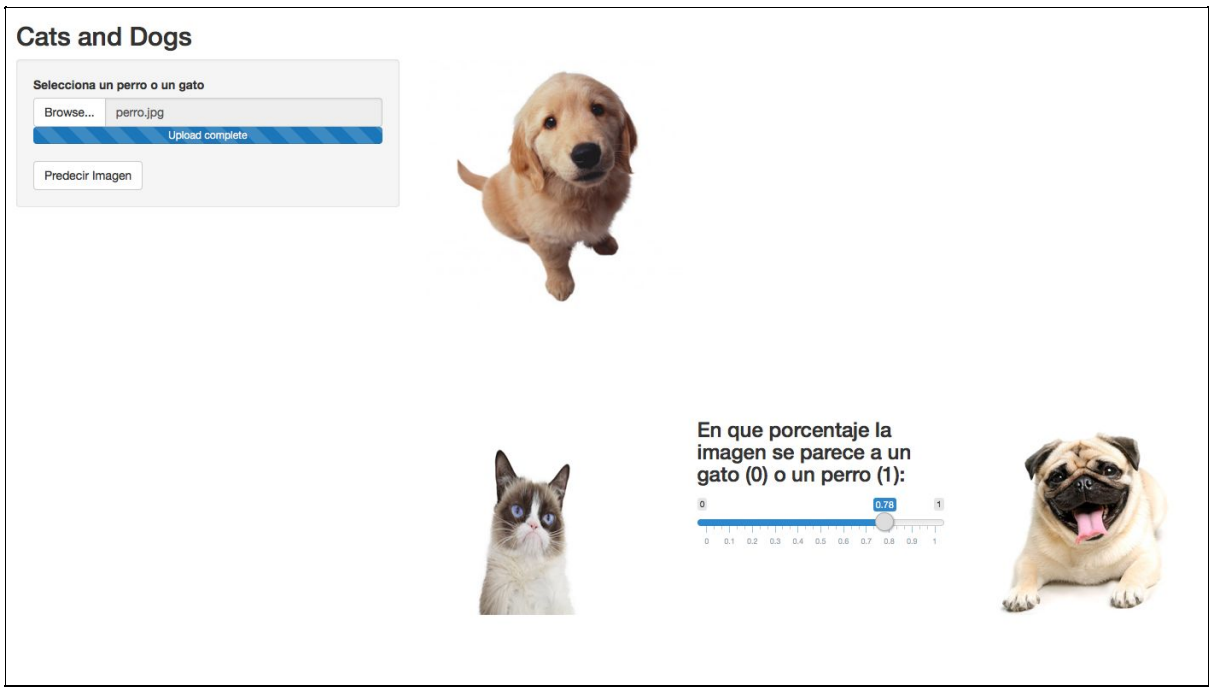

Figura 2: Interfaz visual después de predecir

El código de la parte visual (accesible desde la siguiente web, https://github.com/Kevincifuentes/CatsAndDogs/tree/master/CatsAndDogsShiny) consta de dos partes clave: la parte servidora (o *server)* y la parte visual (o *ui*). Cada parte realiza la función para la que ha sido creada, lógica de negocio o visualización respectivamente. Sin embargo, es necesario enlazar las dos partes para poder realizar ciertas funcionalidades. En estos casos se utilizan observadores (*observe* en el código) que comprueban tanto las variables de entrada (*inputs*) como las de salida (*outputs*).

Teniendo esto en cuenta, podemos pasar a comentar cuales son los pasos seguidos para dotar de funcionalidad a nuestra aplicación. Al ejecutar la Shiny app, lo primero que esta inicializa es la parte servidora, la cual va a contener nuestra red neuronal. Es por esto que cada vez que se inicia el servidor, lo primero que preparamos o ejecutamos es la red

neuronal, aunque no la ejecutamos normalmente, debido a la complejidad de las operaciones y el tiempo que supondría realizar esto. Simplemente cargamos desde disco el modelo que vamos a utilizar, el cual ya ha sido entrenado adecuadamente con el dataset de gatos y perros. Una vez dispuesto esto, simplemente se espera a que el usuario suba una foto para que pueda predecir la imagen. Además, se mostrará la imagen subida, a modo de ejemplo, en la página web. Al final, el resultado, es decir, el porcentaje de probabilidad de que sea un gato o un perro según la predicción de la red neuronal, se mostrará en el *slider* que aparece en el panel principal.

Figura 3: Interfaz visual al subir una foto

6. Conclusiones

El proceso de pruebas se ha llevado a cabo analizando 5000 imágenes (2500 perros y 2500 gatos) con diferentes configuraciones de la red neuronal. En la siguiente tabla se recogen los valores obtenidos como resultado de aplicar dichas configuraciones.

Número de capas (nº neuronas)	**Imagen**	**Porcentaje acierto gatos**	**Porcentaje aciertos perros**	**Porcentaje acierto total**	**Time**
1 capa (13)	Figura 4	68.3%	55.48%	62.8%	1.5 hours
2 capas (6,3)	Figura 5	70.1%	58.16%	64.38%	3.9 hours

2 capas (13,6)	Figura 6	70.3%	60.36%	65.28%	4.13 hours
3 capas (20,10,5)	Figura 7	70.72%	60.86%	65.61%	5.37 hours

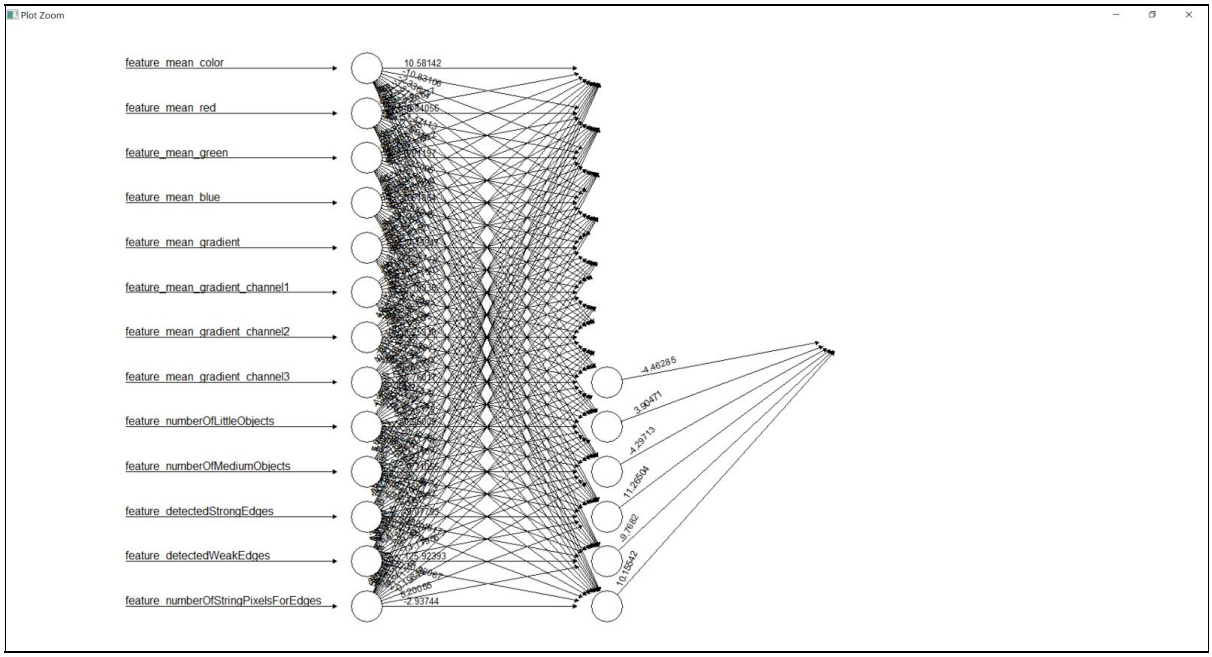

Figura 4: Plot de la red neuronal utilizando una capa (13)

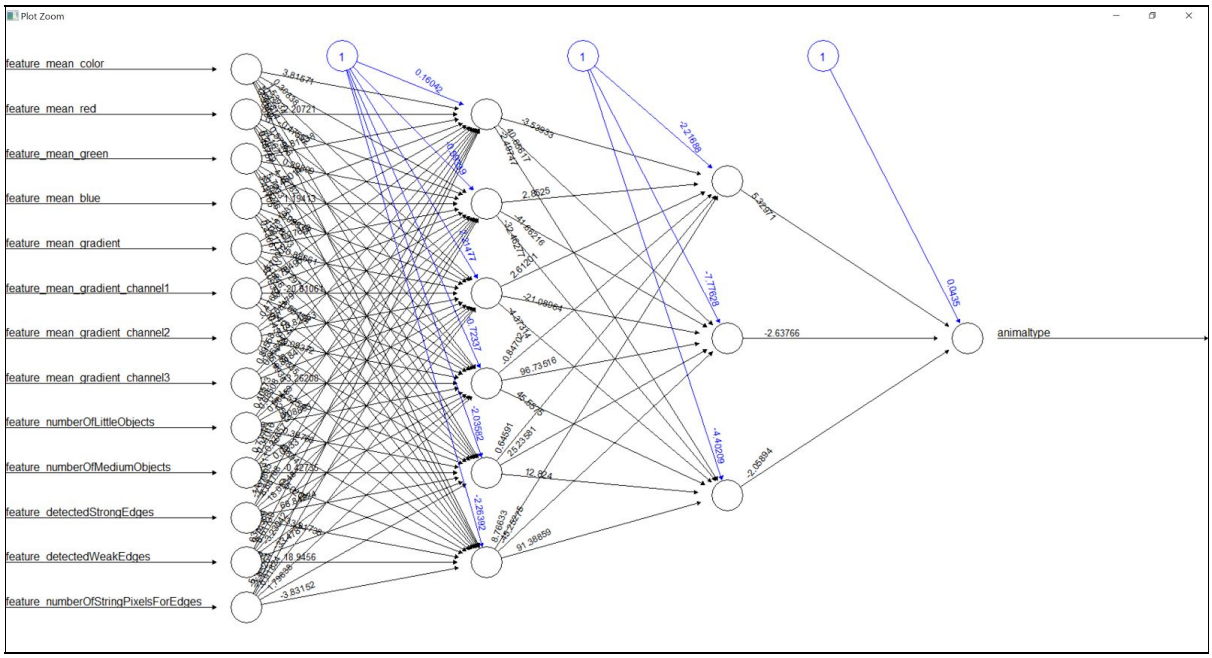

Figura 5: Plot de la red neuronal utilizando dos capas (6 , 3)

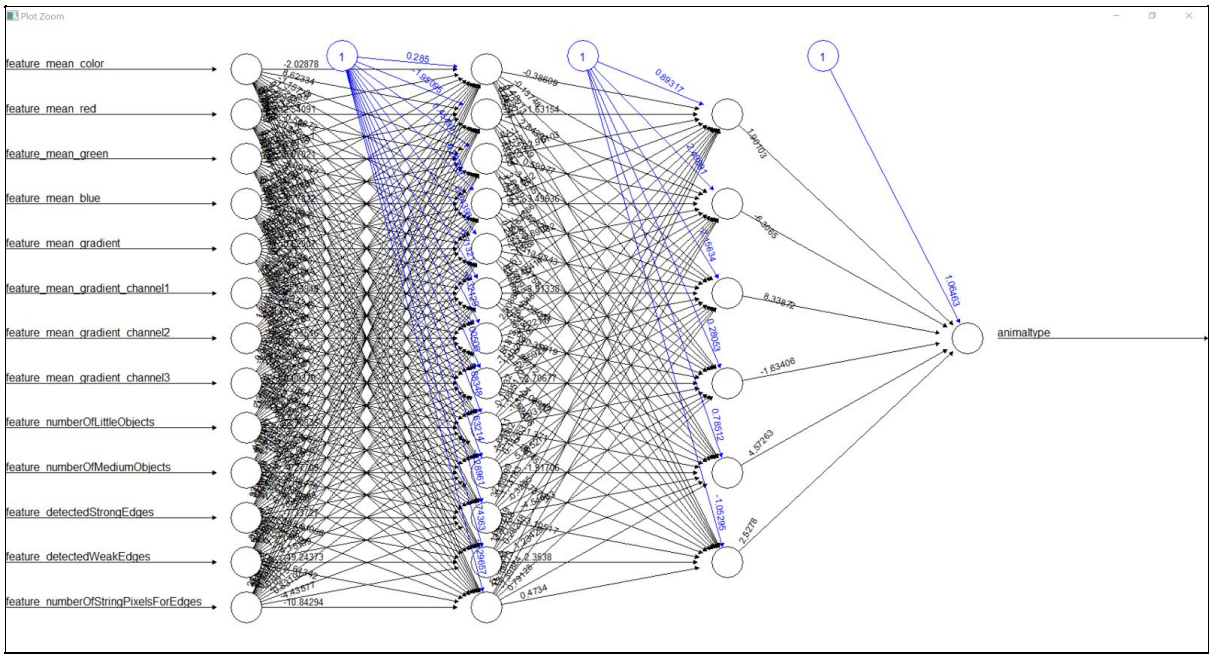

Figura 6: Plot de la red neuronal utilizando dos capas (13 , 6)

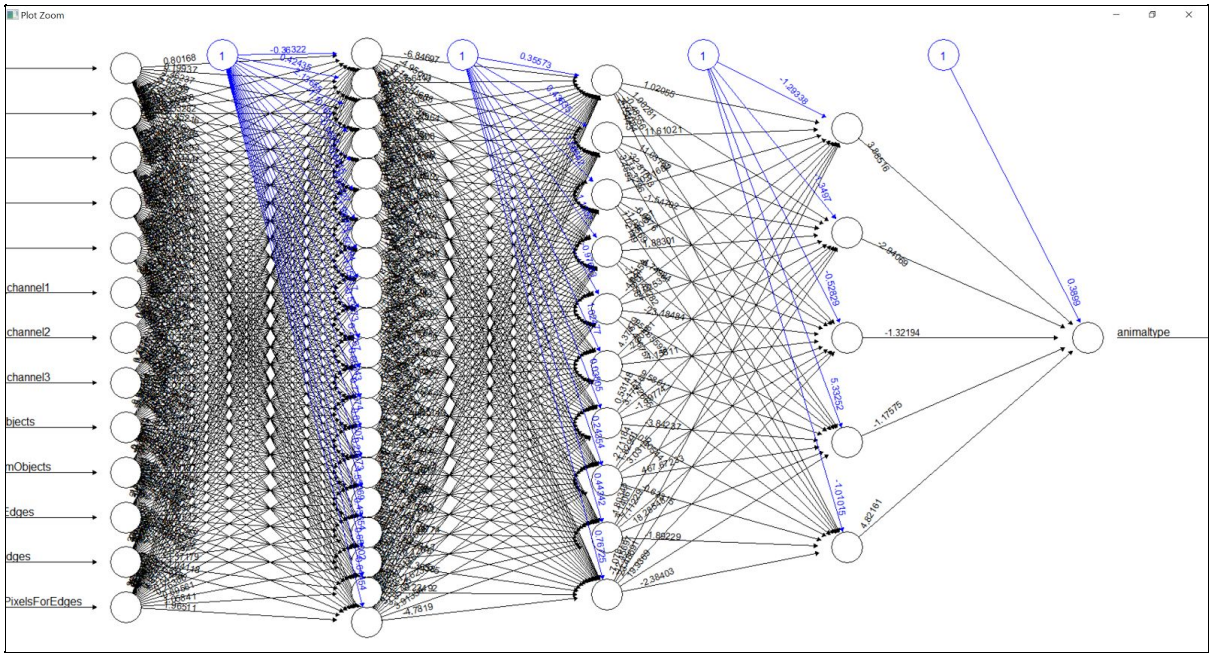

Figura 7: Plot de la red neuronal utilizando tres capas (20,10,5)

El resultado final que hemos obtenido es una red neuronal que aprende a clasificar el contenido de imágenes no etiquetadas consiguiendo valores por encima del 65% en la detección conjunta de perros y gatos, y más del 60% (perros) y 70% (gatos) de acierto en la detección individual.

7. Bibliografía

"Dogs vs. Cats", Kaggle, https://www.kaggle.com/c/dogs-vs-cats, accedido el 17 de Enero del 2017

"Deep Learning Wins Dogs vs Cats competition on Kaggle", KDnuggets, http://www.kdnuggets.com/2014/02/deep-learning-wins-dogs-vs-cats-competition.html, accedido el 17 de Enero del 2017

"ConvNetJS, Deep Learning in your browser", ConvNetJS, http://cs.stanford.edu/people/karpathy/convnetjs/, accedido el 17 de Enero del 2017

"Deep Learning, el resurgir de las redes neuronales", Beeva, https://www.beeva.com/beeva-view/innovacion/deep-learning, accedido el 17 de Enero del 2017

"Redes Neuronales", Wikispaces, https://rpro.wikispaces.com/Redes+neuronales, accedido el 17 de Enero del 2017

"Data Mining con R: Regresion con Red Neuronal", Data Mining con R, http://apuntes-r.blogspot.com.es/2015/12/regresion-con-red-neuronal.html, accedido el 17 de Enero del 2017

"Imager: an R package for image processing", Imager, http://dahtah.github.io/imager/, accedido el 17 de Enero del 2017

"The magick package: Advanced Image-Processing in R", Magick, https://cran.r-project.org/web/packages/magick/vignettes/intro.html, accedido el 17 de Enero del 2017

"Color Quantization in R", R-Bloggers, https://www.r-bloggers.com/color-quantization-in-r/, accedido el 17 de Enero del 2017

"Image recognition tutorial in R using deep convolutional neural networks (MXNet package)", R-Bloggers,https://www.r-bloggers.com/image-recognition-tutorial-in-r-using-deep-convolutional-neural-networks-mxnet-package/, accedido el 17 de Enero del 2017

"A loop-free Canny edge detector", Imager, http://dahtah.github.io/imager/canny.html, accedido el 17 de Enero del 2017