Tecnologías emergentes y datos abiertos: INTELIGENCIA ARTIFICIAL

ACTION

Contenido elaborado por Alejandro Alija, experto en Transformación Digital y datos abiertos

Enero 2020

ACTION

En la sección Metodología, introdujimos al lector sobre la forma en la que se estructura este informe. El *recorrido AIA* (*Awareness*, *Inspire*, *Action*) nos permite adentrarnos de forma gradual en el tema de la inteligencia artificial, desde los conceptos más básicos hasta el desarrollo de un caso práctico indicado para aquellos lectores que quieran pasar a la *Action*. En esta sección hemos decidido utilizar el **superpoder** que la inteligencia artificial nos da para mejorar nuestra capacidad de interpretar imágenes. La IA verá por nuestros ojos y analizará por nuestro cerebro el contenido de unas fotografías. Quizás la IA todavía no sea muy superior a los humanos identificando objetos cotidianos en las imágenes pero sí es mucho más eficiente si tratamos de clasificar cientos de miles de imágenes en unos pocos segundos.

El conjunto de datos

En este caso de uso utilizaremos un conjunto de imágenes disponible en el catálogo de datos de datos.gob.es. En particular utilizaremos el Archivo fotográfico del Gobierno Vasco: imágenes sobre Euskadi y la actividad de Gobierno. Este archivo fotográfico contiene miles de imágenes en formato *jpg* sobre naturaleza, vida cotidiana, objetos comunes, etc. Con el desarrollo de este caso de uso, ejemplificamos como la IA ayuda a los humanos a clasificar las imágenes y anotar su descripción. Sin la ayuda de una IA, la clasificación de imágenes puede ser una tarea larga y aburrida para un humano. Con la ayuda de la IA, los humanos podemos liberar tiempo para realizar actividades propias de los humanos como desarrollar tareas creativas o artísticas todavía fuera del alcance real de la IA. Hecha ya la introducción a nuestro ejercicio. ¡Comencemos!

El archivo fotográfico está disponible para su descarga en el siguiente enlace. Los contenidos del portal Irekia están sujetos a una licencia Creative Commons Reconocimiento

España salvo que se indique lo contrario. Un ejemplo de las imágenes que nos podemos encontrar se ilustra en la figura 6.

***Figura 6**. Ejemplo de conjunto de fotografías disponibles en el sitio web https://argazki.irekia.euskadi.eus/es/photos*

Como comentamos, la IA nos puede ayudar en la clasificación de las imágenes. Tal y como observamos cuando hacemos click en una imagen en particular (como en la figura 7), **la propia web nos solicita ayuda para mejorar la clasificación**. En este caso en particular, vemos cómo la imagen está etiquetada con la descripción *pájaros,* además de las etiquetas adicionales *Animales*, *Fauna*, *Aves*, etc. Podemos imaginarnos cómo el trabajo de abrir imagen por imagen y añadir la descripción y las etiquetas de forma manual es una tarea que puede dejar exhausto a cualquier humano. Sin embargo, veamos ahora cómo la IA realiza esta tarea sin esfuerzo tantas veces como nosotros queramos.

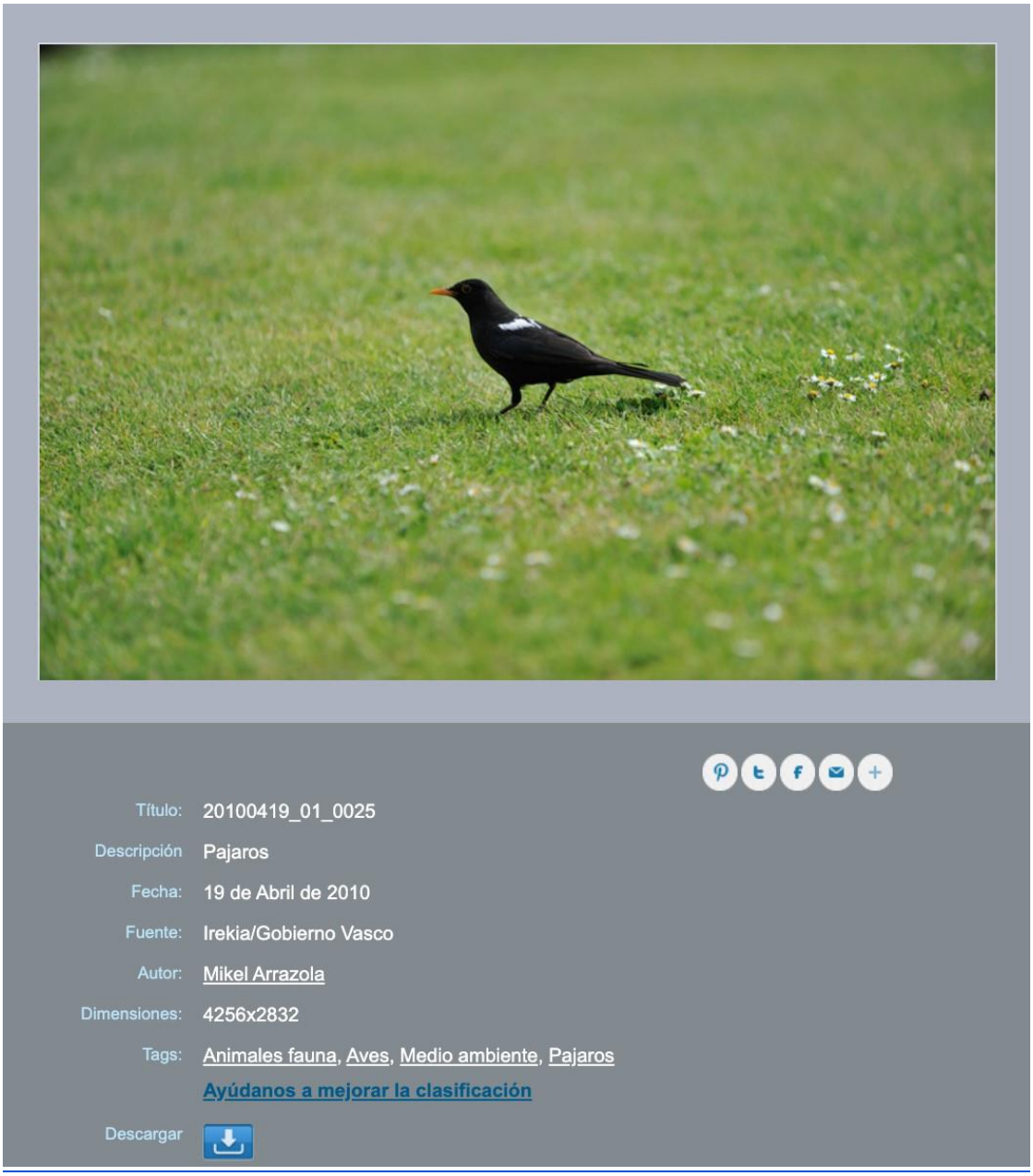

Figura 7. *Imagen particular del repositorio de imágenes https://argazki.irekia.euskadi.eus/es/photos*

La tecnología

Hasta ahora hemos hablado de inteligencia artificial más como concepto que como tecnología. Como hemos introducido anteriormente, la disciplina técnica que hace posible la IA dentro del mundo de los algoritmos se conoce como Deep Learning. Sin embargo, a continuación, describimos muy brevemente las herramientas tecnológicas que nos van a permitir cargar y clasificar imágenes de forma automática.

Para desarrollar nuestro ejemplo de clasificación automática de imágenes vamos a hacer uso de algunas herramientas que nos facilitarán considerablemente el esfuerzo necesario. La algoritmia de Deep Learning para clasificar imágenes es muy compleja y no podríamos dedicar esta sección a crear un algoritmo de clasificación desde cero. Sin embargo, podemos utilizar modelos pre-entrenados, frameworks de programación y desarrollo de Deep Learning que nos faciliten el trabajo. En particular, en este caso de uso, utilizaremos R como lenguaje de programación, R Studio como entorno o IDE (*Integrated Development Environment*) de programación y Keras como API de redes neuronales de alto nivel, escrita en Python y capaz de ejecutarse sobre TensorFlow. El propio paquete Keras incorpora varios **modelos pre-entrenados** que pueden ser utilizados para este y otros muchos casos de uso. La selección del modelo más adecuado (scoring de modelos) para cada tipo de problema es una de las labores más importantes del científico de datos.

En cualquier proceso de ciencia de datos que involucre aprendizaje automático, bien sea machine learning o deep learning, el conjunto de datos iniciales se separa en varias partes. Una de las partes, se utiliza para decirle al algoritmo de entrenamiento que busque las *relaciones ocultas* en los datos que *vinculan* algunas variables (*features*) con aquella variable que queremos predecir (*target*). En este caso, la variable que queremos predecir es la etiqueta que clasifica el contenido de la imagen. Las variables (*features*) involucradas en la predicción, están relacionadas con la información contenida en los píxeles de la imagen. Este proceso, conocido como **entrenamiento del modelo**, es el más exigente (computacionalmente hablando) y consume extensos recursos de computación. Habitualmente, estos recursos de computación son máquinas con la capacidad de realizar

cálculos matemáticos muy complejos (álgebra tensorial) en procesadores gráficos o GPUs (Graphic Processor Unit).

Dependiendo de la complejidad del algoritmo de entrenamiento y del volumen del conjunto de datos utilizados en este entrenamiento, **los cálculos pueden demorar, horas o incluso días.** Por este motivo, en este ejemplo, vamos a utilizar un algoritmo ya entrenado, que tan solo ha de examinar los nuevos datos (nuevas imágenes) e invocar al algoritmo entrenado para extraer la clasificación de la imagen. Al utilizar esta aproximación, podemos clasificar imágenes desde el principio sin dedicar tiempo y esfuerzo al entrenamiento. Como contrapartida, la exactitud de la clasificación de imágenes no será nunca tan alta como si utilizáramos parte del conjunto de imágenes del repositorio de datos abiertos para entrenar el algoritmo.

La solución al problema

Los pasos que tenemos que dar para construir nuestro sencillo sistema de clasificación automática de imágenes son:

Figura 8. *Diagrama de los pasos para construir nuestro sencillo sistema de clasificación.*

Una vez que tenemos listos los entornos de programación y las librerías necesarias instaladas[1] comenzamos por cargar la librería keras

```
library(keras)
```

Limpiamos el entorno previo para eliminar ejecuciones anteriores y comenzar en un espacio de trabajo limpio de variables y objetos almacenados con anterioridad. Esto asegura una ejecución correcta de las siguientes líneas de código.

```
# Clear out the session
k_clear_session()
```

Cargamos el modelo de clasificación de imágenes que ha sido previamente entrenado con la base de imágenes de Imagenet. Imagenet es un banco de imágenes organizado en torno a una jerarquía de nombres. Las imágenes de Imagenet incluyen etiquetas en inglés. De ahí que los resultados de nuestro modelo de clasificación sean descripciones de las imágenes en inglés.

```
model <- application_vgg16(weights = "imagenet")
```

Una vez cargado el modelo entrenado que vamos a utilizar para clasificar nuestras imágenes es el momento de cargar las primeras imágenes y realizar los primeros tests.

[1] En el anexo I se detallan las instrucciones de instalación del entorno requerido.

```
# The local path to our target image
img_path <-
"https://argazki.irekia.euskadi.eus/photos/p740/20100522_01_0203.jpg"
```

Hemos cargado la siguiente imagen (figura 9) del repositorio de datos abiertos:

Figura 9. *Imagen titulada 20100522_01_0203. CC BY-3.0-ES 2012/EJ-GV/Irekia-Gobierno Vasco/Mikel Arrazola'*

Las etiquetas introducidas de forma manual para esta imagen son:

Abejaruco, Animales fauna, Aves, Medio ambiente, Pajaros

El siguiente fragmento de código adapta y prepara la imagen para ser entendida por el algoritmo que se encargará de clasificarla. Es recomendable que el modelo trabaje con imágenes más pequeñas para asegurar tiempos de ejecución razonables. Además, el modelo necesita la imagen en forma de array con los 3 colores básicos separados por canales.

```
# Start witih image of size 224 × 224
img <- image_load(img_path, target_size = c(224,
224)) %>%
# Array of shape (224, 224, 3)
image_to_array() %>%
# Adds a dimension to transform the array into a
batch of size (1, 224, 224, 3)
array_reshape(dim = c(1, 224, 224, 3)) %>%
# Preprocesses the batch (this does channel-wise
color normalization)
imagenet_preprocess_input()
```

Tras estos pasos previos, hemos llegado al momento crucial en el que el algoritmo tratará de clasificar la imagen utilizando para ello el modelo de clasificación vgg16 entrenado con el banco de imágenes de Imagenet.

```
preds <- model %>% predict(img)
imagenet_decode_predictions(preds, top = 3)[[1]]
```

Las dos líneas de código anteriores, invocan la predicción del modelo y filtran los resultados de la clasificación, restringiendo la salida del algoritmo a los 3 resultados (etiquetas) con mayor relevancia. Los modelos de clasificación de imágenes asignan una probabilidad a los

resultados de la clasificación. Es decir, el modelo devuelve la etiqueta que creé que corresponde con la imagen junto con la probabilidad de que esta etiqueta sea la correcta.

En nuestro caso concreto, el modelo devuelve la siguiente tabla con los resultados de la clasificación:

class_name	class_description	score
<chr>	<chr>	<dbl>
n01828970	bee_eater	0.9995731711
n01530575	brambling	0.0001940502
n02011460	bittern	0.0001244307

El análisis de la tabla de salida tras la ejecución del modelo (en menos de 5 segundos) nos indica que, este modelo predice que la imagen de entrada es un *comedor de abejas* (traducción literal) con un 99.95% de probabilidad. El modelo ha calculado también que la imagen podría corresponder con un *pinzón* al 0.019% de probabilidad y un *avetoro* al 0.012% de probabilidad. Es decir, el modelo clasifica esta imagen como un *comedor de abejas* con casi total seguridad. Desgraciadamente, no somos unos expertos en aves, así que para comprobar si el modelo ha acertado, realizamos una búsqueda de imágenes en Google por la palabra bee_eater. Esto es lo que nos encontramos (figura 10).

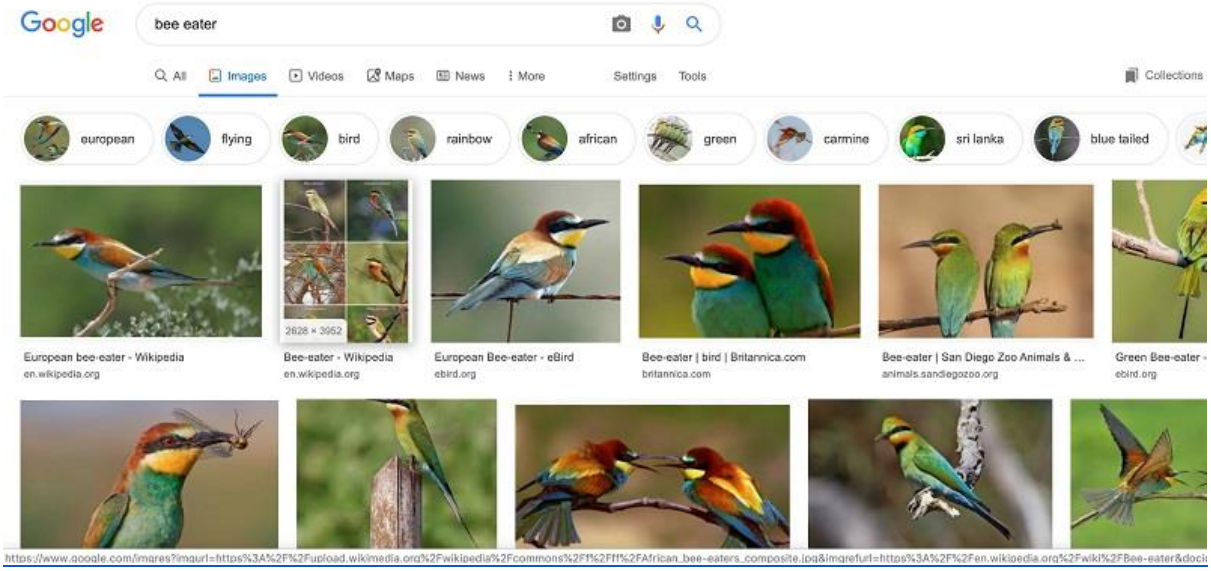

***Figura 10**. Resultados de la búsqueda de imágenes de Bee Eater en Google.*

Podemos ver cómo, efectivamente, la búsqueda en Google devuelve cientos de imágenes similares a la que hemos utilizado como entrada desconocida del modelo.

En este caso, el modelo predice la etiqueta de la imágen y acierta con contundencia el contenido de la misma. Veamos otro ejemplo en el que el resultado no es tan exacto.

Tomemos la siguiente imagen (figura 11) del mismo repositorio cuyas etiquetas manuales son:

Animales fauna, Caza, Deporte, Digital, Diurna, Epagneul breton, Exterior, Formato, Mamíferos, Medio ambiente, Perros, Plano medio, Tomas fotográficas.

Figura 11. Imágen titulada 20030930_01_0026. CC BY-3.0-ES 2012/EJ-GV/Irekia-Gobierno Vasco/Mikel Arrazola'

Ejecutemos el modelo de nuevo para obtener los resultados.

```
# The local path to our target image
img_path <- "https://argazki.irekia.euskadi.eus/photos/p740/20030930_01_0026.jpg"
```

```
# Start witih image of size 224 x 224
img <- image_load(img_path, target_size = c(224, 224)) %>%
  # Array of shape (224, 224, 3)
```

```
  image_to_array()                                                %>%
  # Adds a dimension to transform the array into a batch of size (1,
224,                              224,                              3)
  array_reshape(dim     =     c(1,     224,     224,     3))     %>%
  #   Preprocesses   the   batch   (this   does   channel-wise   color
normalization)
  imagenet_preprocess_input()

preds          <-          model          %>%          predict(img)
imagenet_decode_predictions(preds, top = 3)[[1]]
```

class_name	class_description	score
<chr>	<chr>	<dbl>
n02100735	English_setter	0.2331140
n02091244	Ibizan_hound	0.1794144
n02101388	Brittany_spaniel	0.1744492

En este caso vemos como los tres primeros resultados del modelo de clasificación tan solo suman el 58.7% de la probabilidad total. El mejor resultado del modelo (con una probabilidad del 23.3%) corresponde con la etiqueta de la imagen Setter Inglés. Si realizamos la misma búsqueda en Google (figura 12) que antes constatamos que los resultados son más dispares, aunque todavía parecen bastante correctos.

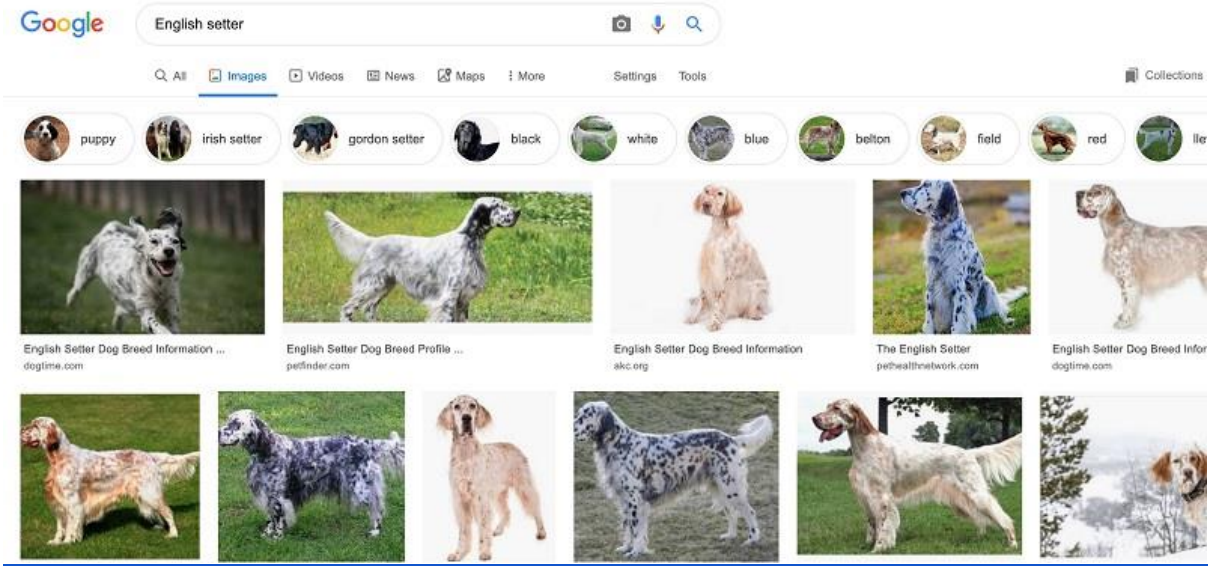

Figura 12. *Resultados de la búsqueda de imágenes de English Setter en Google.*

En este ejemplo, hemos visto cómo implementar de forma muy sencilla un pequeño código en lenguaje R que nos permite clasificar imágenes comunes disponibles en un repositorio de datos abiertos. Los resultados parecen muy razonables incluso cuando el modelo utilizado ha sido entrenado con un conjunto de imágenes que no guarda ninguna relación con el empleado aquí. Animamos a los lectores a tratar de reproducir el ejemplo con imágenes disponibles en el repositorio de datos abiertos o con cualquier otra imagen que consideren interesante.

Puedes seguir aprendiendo sobre Inteligencia Artificial en las secciones Awareness e Inspire

ANEXO I. INSTRUCCIONES DETALLADAS PARA REPLICAR EL EJEMPLO DE LA SECCIÓN ACTION

Para poder probar el ejemplo que hemos ilustrado en la sección Action es necesario instalar los paquetes de software necesarios.

Como en la mayoría de los casos, el software para análisis de datos y Deep Learning en particular, depende de la plataforma base que utilicemos (habitualmente entendemos por plataforma el sistema operativo que utilicemos como Windows, Linux, macOS, etc.)

Sin duda, la plataforma más conveniente para trabajar en ciencia de datos es un sistema Linux. La mayoría de herramientas que hemos utilizado en la sección Action son nativas en Linux o bien disponen de abundante documentación para su instalación en entornos Linux. Esto no quiere decir que no sea posible reproducir el mismo ejemplo sobre plataforma Windows, macOS u otro sistema, pero puede llegar a ser relativamente más costoso en caso de encontrarnos con problemas durante la instalación o configuración.

Dicho esto, en nuestro caso concreto vamos a utilizar una aproximación diferente a tener que escoger una plataforma en concreto para este ejemplo. Nosotros hemos decidido utilizar **una aproximación de contenedores** para descargarnos una imagen del software requerido en un sistema macOS Catalina 10.15.1. Para aquellos usuarios que no tengan experiencia previa con los sistemas de contenedores recomendamos los siguientes recursos web. Es necesario tener unas nociones básicas de contenedores para poder realizar los pasos que vienen a continuación.

¿Qué son los contenedores?

Terminología básica de contenedores

¿Por qué Docker para la ciencia de datos?

Una vez entendido lo básico sobre la tecnología de contenedores y el concepto de entorno aislado para ejecutar software procedamos con el proceso para reproducir Action.

Paso 1. Descargar e instalar Docker (en nuestro caso para MacOS)

https://docs.docker.com/docker-for-mac/install/

Paso 2. Descargar e Instalar Kitematic.

Kitematic es una interfaz de usuario visual que nos permite localizar y descargar imágenes de Docker disponibles en el Docker Hub.

Paso 3. Descargar una imagen de RStudio disponible.

Abrimos Kitematic y localizamos la imagen **rocker/tidyverse** y pulsamos **create.** Una vez terminado el proceso de descarga de la imagen ejecutamos el siguiente comando en nuestro terminal:

```
sudo docker run -d -p 8787:8787 -e PASSWORD=<your_password> --name deeplearning rocker/tidyverse
```

Paso 4. Ejecutamos nuestro contenedor.

Una vez aquí tenemos ejecutando en nuestro ordenador un contenedor que ejecuta una versión aislada de RStudio en nuestro sistema.

Paso 5. Accedemos al entorno de RStudio desde nuestro navegador.

Abrimos una nueva pestaña en nuestro navegador y escribimos en la barra de direcciones https://localhost:8787. Nos aparece la pantalla de login donde introducimos el usuario (rstudio) y el password que hayamos escogido en el paso previo.

Paso 6. Asociar un volumen de almacenamiento al contenedor.

Como hemos comentado en la introducción de este anexo los contenedores son entornos aislados que, a priori, no tienen una interfaz con nuestro sistema. De forma práctica significa que un contendor no puede ver nuestro disco duro si no se lo permitimos. Para poder guardar ficheros en nuestro ordenador y que nuestro Rstudio sea capaz de encontrarlos hemos de configurar el contenedor para que pueda *ver* una parte de nuestro disco duro. Esto nos va a permitir descargar las imágenes del repositorio de imágenes abiertas para luego ejecutar el algoritmo de clasificación.

Para asociar un volumen de almacenamiento a nuestro contenedor (así se llama la forma de que nuestro ordenador vea nuestro disco) simplemente vamos a la sección de Volúmenes de Kitematic y configuramos la ruta del directorio que queramos ver desde nuestro contenedor.

Paso 7. Instalación de Keras

Para instalar Keras en nuestro entorno aislado de Rstudio ejecutamos los siguientes comandos en la consola de Rstudio:

```
install.packages("keras")
require(keras)
install_keras()
```

Paso 8. Ejecutamos el ejemplo de la sección Action a partir de la carga del modelo pre-entrenado