

Práctica reconocimiento de emociones de la asignatura IPO del MUSI

Análisis de un modelo de reconocimiento de emociones con la herramienta LIME

Autores:

- Ramón Rotaeché Fernández de la Riva
- Lluís Bernat Ladaria

Introducción

Se nos ha proporcionado una red neuronal entrenada con un conjunto de caras que expresan las 6 emociones básicas más la neutra.

Deseamos obtener una explicación de los elementos que componen las imágenes que nos permita saber en qué se fija la red neuronal proporcionada para dirimir qué emoción se está expresando. Es decir, queremos averiguar qué partes de las caras son las que más contribuyen en cada probabilidad que calcula la red.

Requisitos

Son necesarias las siguientes librerías y sus dependencias:

- keras
- lime
- abind
- magick
- lattice
- ggplot2
- gdata
- caret
- e1071
- pillow (`pip install pillow`)
- SciPy (`pip install SciPy`)

Pruebas LIME con la red neuronal

En los siguientes apartados vamos a experimentar con la red neuronal suministrada. De las dos opciones que se nos han ofrecido, hemos decidido usar la entrenada con las *caras recortadas* por arrojar una precisión un poco mejor que la original.

Metodología

Suministraremos a la red entrenada una selección de caras del conjunto de pruebas que expresarán algunas emociones de nuestro interés. Por cada una de las caras suministradas, la red generará un conjunto de probabilidades correspondientes a cada una de las posibles emociones.

En cada caso, se usará la herramienta LIME (Local Interpretable Model-Agnostic Explanations) para conseguir el conjunto de explicaciones que se consideren más ilustrativas. Comentaremos los resultados obtenidos, destacando lo singular por encima de lo esperado.

También comentaremos los resultados de la encuesta sobre discernimiento de emociones que se hizo a los alumnos.

Carga de la red neuronal

Vamos a cargar el modelo de red neuronal ya entrenado con el *dataset* de caras recortadas.

```
library(keras)

# Trayectoria de la serialización del modelo
model_path = "../data/model/model7Emociones.h5"
# Carga del modelo HDF5
model <- load_model_hdf5(
  model_path,
  custom_objects= NULL,
  compile = TRUE
)
```

Preparación de las caras

Además de cargar la red en el apartado anterior, vamos a comprobar que todo funciona, pasando por la red todas las caras de prueba y examinando brevemente las probabilidades generadas.

Por tanto este apartado y el siguiente son opcionales, aunque los hemos introducido para guardarlos como referencia futura.

```
# Trayectoria del conjunto de imágenes de test
images_path = "../data/test_crop/"

# Preparación de las imágenes: Imágenes normalizadas en escala de grises de tamaño 128x128
## Genera lotes de imágenes con data augmentation en tiempo real
datagen <- image_data_generator(rescale= 1/255 # Normalización de las imágenes entre 0 y 1
)
## Genera lotes de imágenes a partir de un directorio
test_generator <- flow_images_from_directory(
  directory = images_path, # Debe contener un subdirectorío por clase
  generator = datagen,
  color_mode = "grayscale", # Las imágenes se convertirán para tener 1 canal de color
  target_size = c(128, 128), # Las imágenes se redimensionarán a 128 x 128
  class_mode= "categorical", # 2D one-hot encoded labels
  batch_size=1, # Lotes formados por 1 imagen
  shuffle= FALSE # Sin aleatorización
)
```

Evaluación de las predicciones

```
# Realización de las predicciones
## Generación de predicciones a partir de un generador de datos
STEP_SIZE =
  test_generator$n/test_generator$batch_size
prediction <- model %>% predict_generator(
  # Generador que produce lotes de imágenes
```

```

generator = test_generator,
# Número total de pasos (lotes de imágenes) a proporcionar por el generador antes de detenerse
steps = STEP_SIZE)
head(prediction)

##          [,1]      [,2]      [,3]      [,4]      [,5]      [,6]
## [1,] 0.9732536 1.486878e-05 2.031849e-05 1.048117e-04 2.640544e-02 2.002597e-04
## [2,] 0.9712657 4.825610e-07 2.140396e-05 1.420997e-04 2.716509e-02 1.404623e-03
## [3,] 0.9995404 1.227113e-05 1.644911e-08 2.000600e-04 2.438742e-04 2.089900e-07
## [4,] 0.9826591 7.536783e-08 6.509033e-06 2.541134e-08 9.518804e-06 3.288751e-06
## [5,] 0.9409639 3.319219e-08 5.603477e-02 1.709934e-06 2.710622e-03 2.889431e-04
## [6,] 0.9346032 1.730108e-06 9.230022e-07 2.439824e-04 6.511680e-02 3.332497e-05
##          [,7]
## [1,] 8.594183e-07
## [2,] 6.865957e-07
## [3,] 3.143650e-06
## [4,] 1.732141e-02
## [5,] 4.443893e-08
## [6,] 4.477892e-08

labels <- unlist(test_generator$class_indices)
print(labels)

##      ANGER  DISGUST     FEAR      JOY  NEUTRAL  SADNESS  SURPRISE
##          0        1        2        3        4        5        6

```

Preparativos LIME

```

library(lime)
library(abind)

# Necesitamos pasar a LIME un ítem cualquiera del mismo tipo de los que le pediremos explicaciones.
# En nuestro caso le pasamos una imagen del conjunto de test
sample_path = '../data/test_crop/JOY/Wanda_IngratiatingSmile.png'

# Definimos la función que vamos a utilizar para preparar las imágenes para el modelo.

# Transforma un vector de caracteres al formato que espera el modelo (imágenes normalizadas en escala de grises)
img_preprocess <- function(x) {
  arrays <- lapply(x, function(path) {
    img <- image_load(path, grayscale = TRUE, target_size = c(128, 128))
    x <- image_to_array(img, data_format = "channels_last")
    x <- array_reshape(x, c(1, dim(x)))
    x <- x/255.0
  })
  do.call(abind, c(arrays, list(along = 1)))
}

# Creamos un explicador con LIME a partir de nuestro modelo y con nuestras etiquetas
explainer <- lime(
  x = sample_path,
  model = as_classifier(model, labels = names(labels)),
  preprocess = img_preprocess
)

```

```
)
```

Cara número 1: alegría

Vamos a analizar una imagen con la etiqueta alegría



Figure 1: Wanda, alegre ma non troppo

```
explain_img_path = '../data/test_crop/JOY/Wanda_IngratiatingSmile.png'

res <- predict(model, img_preprocess(explain_img_path))
colnames(res) <- names(labels)
print(res)
```

##

ANGER

DISGUST

FEAR

JOY

NEUTRAL

SADNESS

```

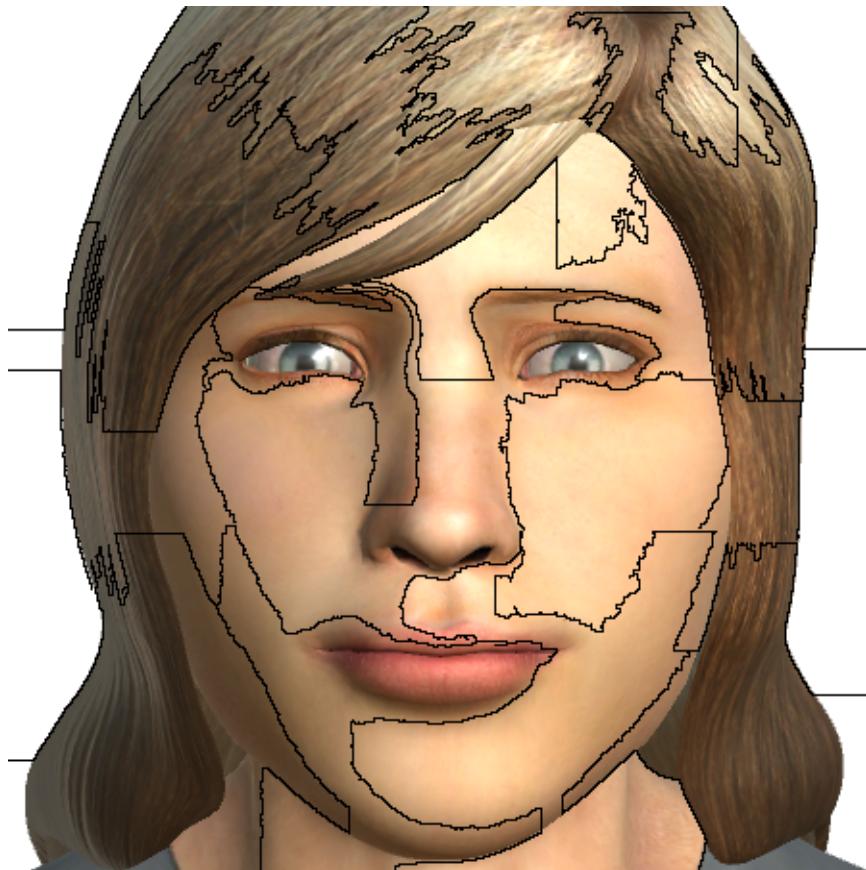
## [1,] 1.062606e-08 0.000476656 1.607817e-10 0.9956175 0.003896179 9.74334e-06
##          SURPRISE
## [1,] 1.226902e-16
library(magick)

## Linking to ImageMagick 6.9.10.23
## Enabled features: fontconfig, freetype, fftw, lcms, pango, webp, x11
## Disabled features: cairo, ghostscript, heic, raw, rsvg

## Using 4 threads
n_superpixels = 20

# Mostraremos los superpixels para poder evaluar cuáles son las zonas que se utilizan antes de aplicar
# Un superpixel es cada una de las zonas en las que se segmenta la imagen para facilitar las explicaciones
# Estos superpíxeles deberían contener patrones específicos de la imagen, por ello visualizarlos nos puede
# Ayudar a entender qué partes de la imagen son más relevantes para la predicción
# Tenemos que tener en cuenta que si las características importantes de la imagen se cortan en demasiadas
# Pequeñas zonas, la explicación no será muy útil. Por otro lado, si son muy grandes, perderemos información
# Detallada. Cuanto más grande sea el objeto que buscamos en relación al tamaño de la imagen, menos superpíxeles debemos usar
plot_superpixels(explain_img_path, n_superpixels)

```



```

# Generación de la explicación
explanation <- explain(
  explain_img_path,
  explainer,
  n_labels = 2,
  n_features = 3 * n_superpixels %/% 4, # limitamos el cardinal del conjunto de features para la explicación
  n_superpixels = n_superpixels)

```

```
# Visualización de los resultados
```

```
plot_image_explanation(as.data.frame(explanation))
```

Label: JOY
Probability: 1
Explanation Fit: 0.87



Label: NEUTRAL
Probability: 0.0039
Explanation Fit: 0.28

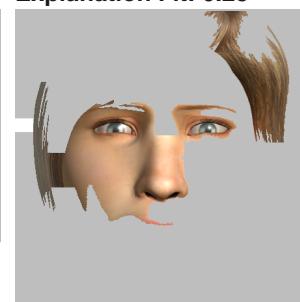


```
plot_image_explanation(as.data.frame(explanation),  
                      display = 'block',  
                      show_negative = TRUE)
```

Label: JOY
Probability: 1
Explanation Fit: 0.87

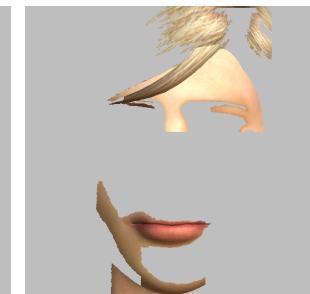


Label: NEUTRAL
Probability: 0.0039
Explanation Fit: 0.28



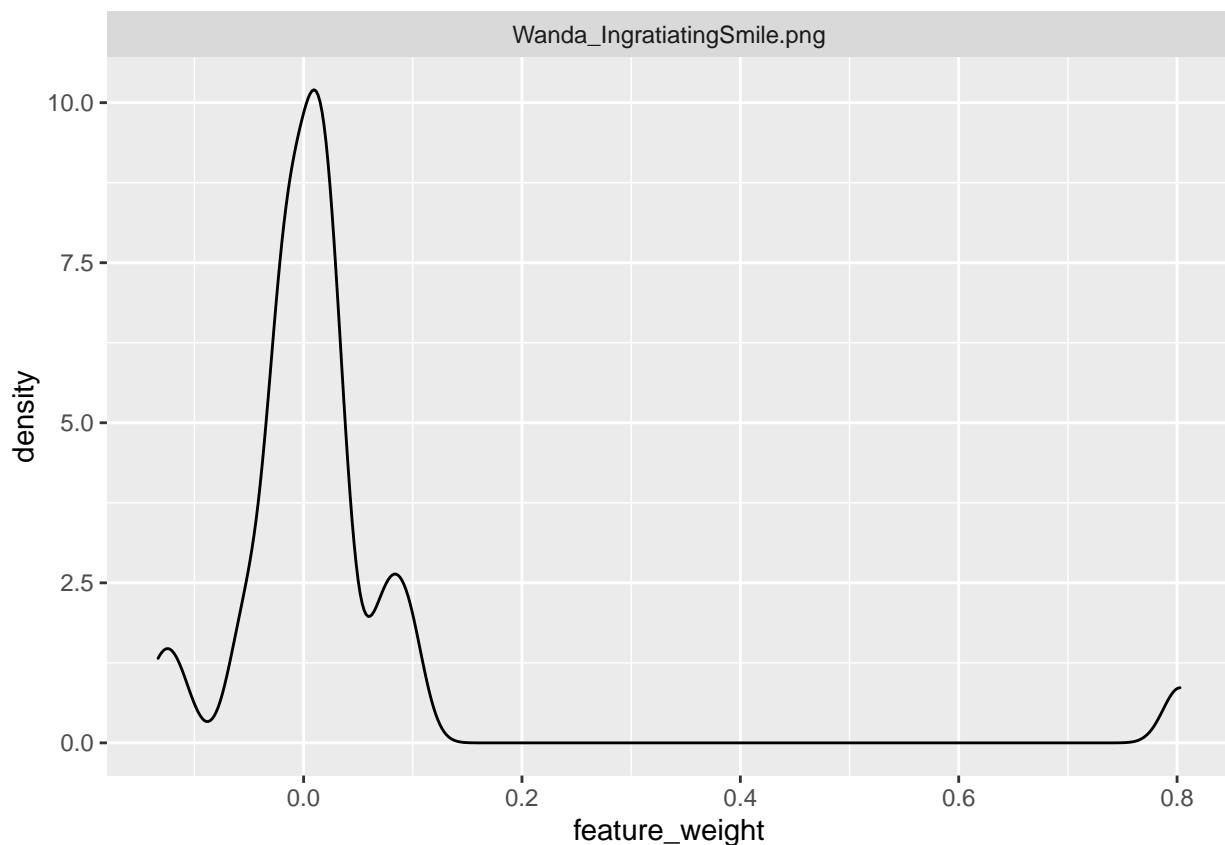
y

Type: Contradicts

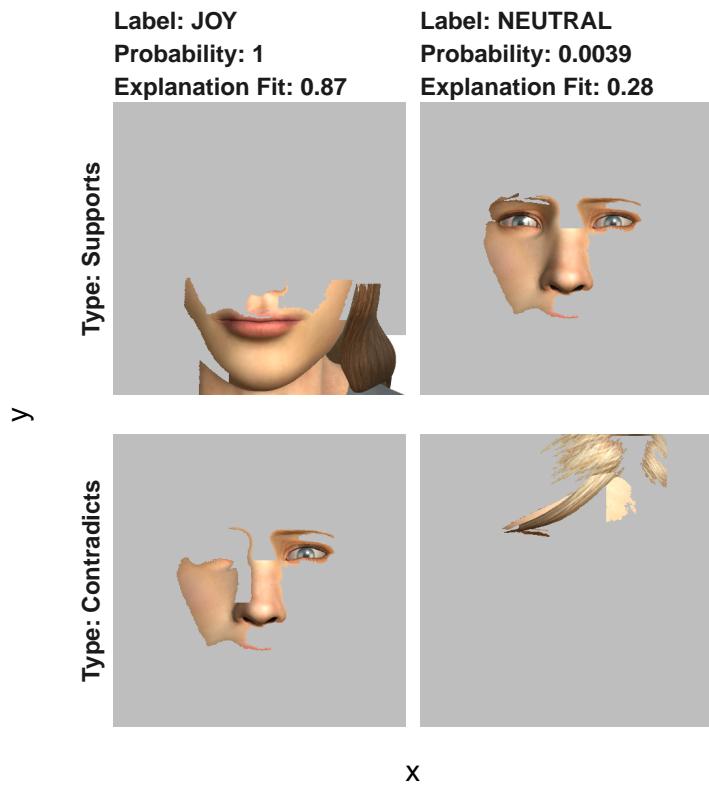


x

```
# Se puede precisar un poco más mostrando la gráfica de los pesos de las características para establecer
library(ggplot2)
explanation %>%
  ggplot(aes(x = feature_weight)) +
  facet_wrap(~ case, scales = "free") +
  geom_density()
```



```
plot_image_explanation(as.data.frame(explanation),
                      display = 'block',
                      show_negative = TRUE,
                      threshold = 0.06)
```



Conclusiones cara 1

Hemos elegido este caso porque nos ha parecido una expresión que arrastraba una cierta melancolía y queríamos saber si la red neuronal lo clasificaría con probabilidades bajas.

Si bien parece que la red lo tiene medianamente claro, pues afirma con un 100% de probabilidad que Wanda está alegre.

La puntuación de alegría se ve favorecida por la configuración de labios y barbilla (como parecería lógico) y extrañamente también por una parte importante del cuello. En el caso presentado la nariz y el ojo izquierdo puntúan en contra. Suponemos que es la forma que ha encontrado la red de darnos parte de razón en que la expresión de alegría de Wanda podría ser más expresiva.

La valoración de expresión neutral se ve favorecida por los ojos (seguimos pensando que melancólicos) y nariz. Sin embargo el flequillo parece ir en contra.

Cara número 2: sorpresa

Vamos a analizar una imagen con la expresión de sorpresa.

```
explain_img_path = '../data/test_crop/SURPRISE/Simona_Surprise.png'

res <- predict(model, img_preprocess(explain_img_path))
colnames(res) <- names(labels)
print(res)

##          ANGER      DISGUST      FEAR        JOY      NEUTRAL      SADNESS
## [1,] 4.44277e-05 2.533189e-07 0.2346969 1.244936e-06 1.594702e-05 8.391011e-06
##          SURPRISE
## [1,] 0.7652329
```



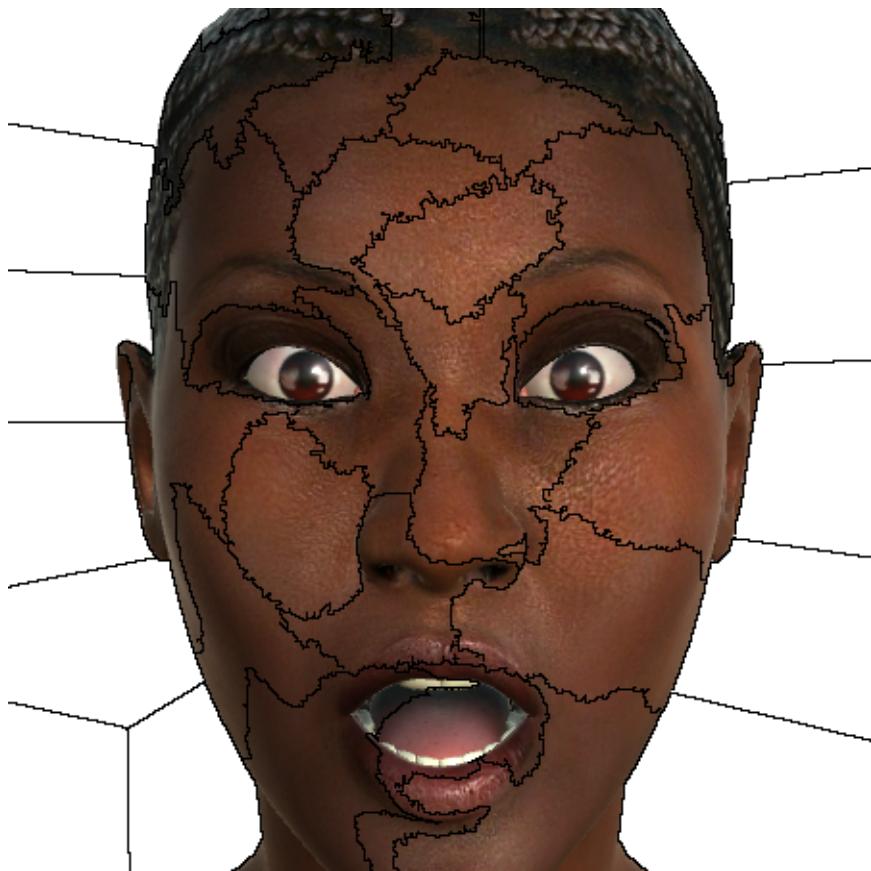
Figure 2: Simona, recibiendo la nueva factura de la luz

```

library(magick)
n_superpixels = 40

# Mostraremos los superpixels para poder evaluar cuáles son las zonas que se utilizan antes de aplicar
# Un superpixel és cada una de las zonas en las que se segmenta la imagen para facilitar las explicaciones
# Estos superpixeles deberían contener patrones específicos de la imagen, por ello visualizarlos nos permite
# Tenemos que tener en cuenta que si las características importantes de la imagen se cortan en demasiadas
# Cuanto más grande sea el objeto que buscamos en relación al tamaño de la imagen, menos superpixeles devolverá
plot_superpixels(explain_img_path, n_superpixels)

```



```

# Generación de la explicación
explanation <- explain(
  explain_img_path,
  explainer,
  n_labels = 2,
  n_features = 3 * n_superpixels %/% 4, # limitamos el cardinal del conjunto de features para la explicación
  n_superpixels = n_superpixels)
# Visualización de los resultados
plot_image_explanation(as.data.frame(explanation))

```

Label: SURPRISE
Probability: 0.77
Explanation Fit: 0.15



Label: FEAR
Probability: 0.23
Explanation Fit: 0.31

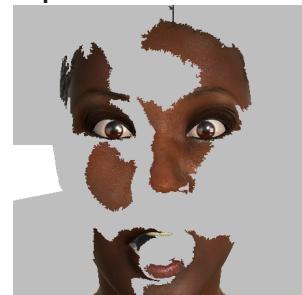


```
plot_image_explanation(as.data.frame(explanation),  
                      display = 'block',  
                      show_negative = TRUE)
```

Label: SURPRISE
Probability: 0.77
Explanation Fit: 0.15

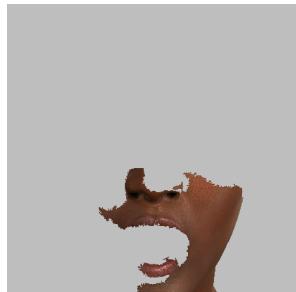


Label: FEAR
Probability: 0.23
Explanation Fit: 0.31



y

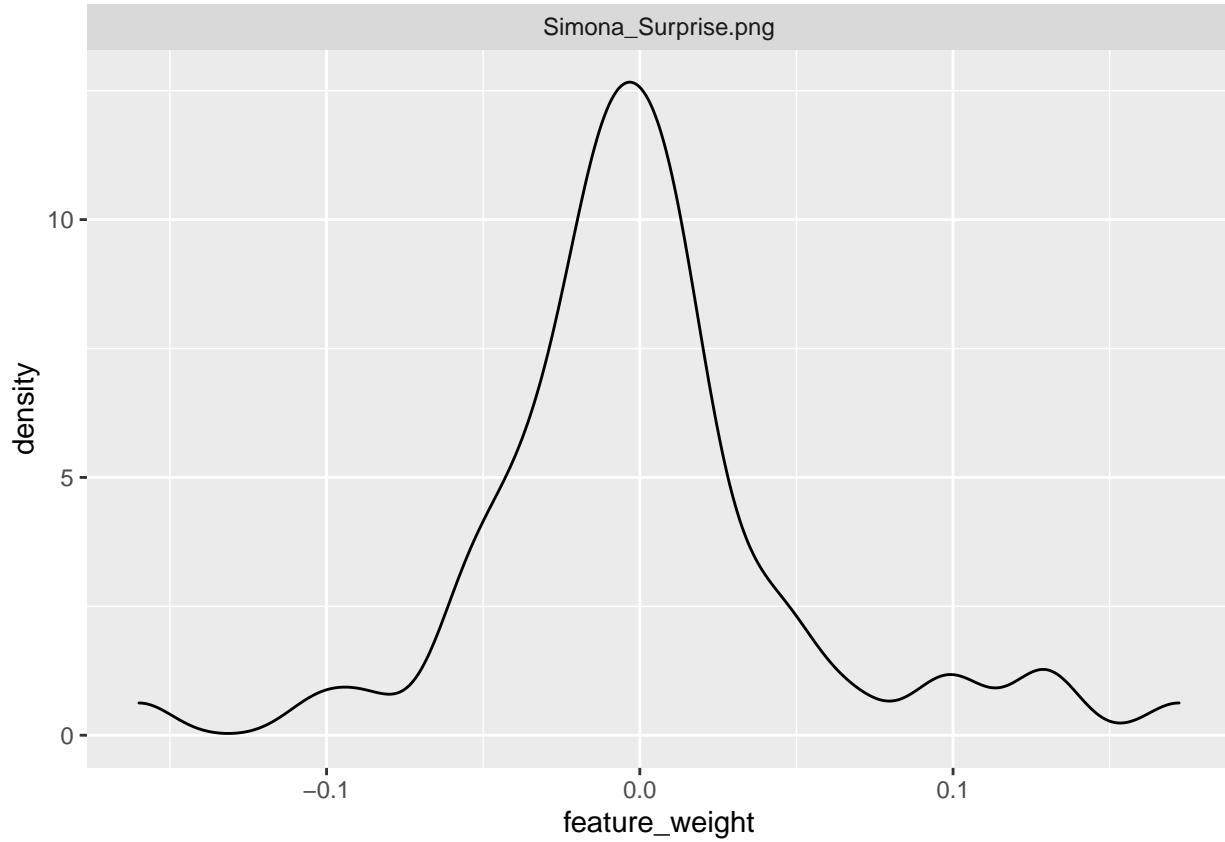
Type: Contradicts



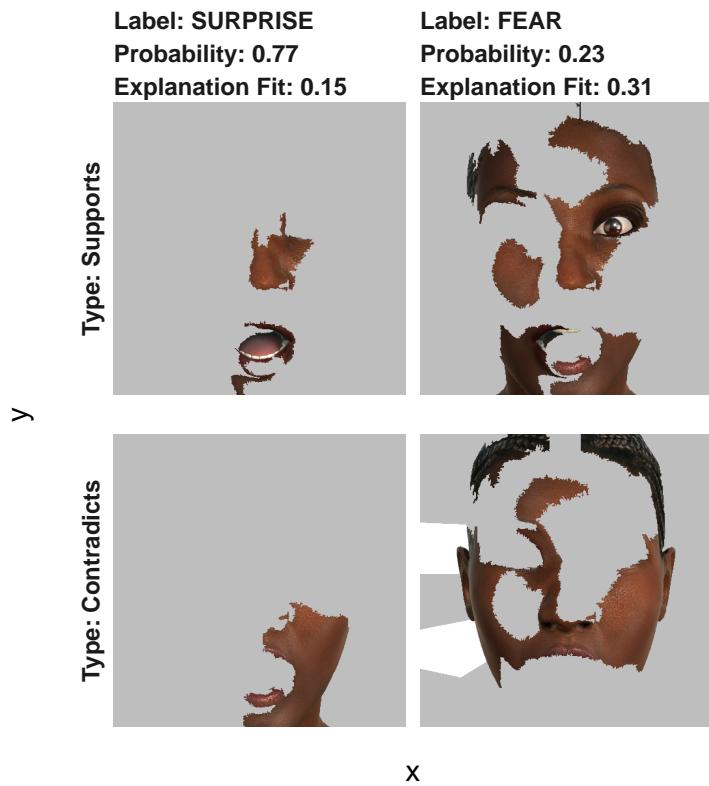
x

```
# Se puede precisar un poco más mostrando la gráfica de los pesos de las características para establecer  
library(ggplot2)  
explanation %>%
```

```
ggplot(aes(x = feature_weight)) +  
  facet_wrap(~ case, scales = "free") +  
  geom_density()
```



```
plot_image_explanation(as.data.frame(explanation),  
                      display = 'block',  
                      show_negative = TRUE,  
                      threshold = 0.03)
```



Conclusiones cara 2: sorpresa

En este caso, parece que para decidir sorpresa (77%) es suficiente fijarse en la apertura de la boca. Sin embargo la apreciación de miedo requiere revisar zonas como la barbilla, una parte del labio, ojos, cejas y algo de frente.

Llama la atención que la red se fije principalmente en la abertura de la boca para decidir sorpresa y en cambio considere casi toda la cara para decidir miedo (con un 23% de probabilidad).

Bonus: El grito (E. Munch)

En este pequeño *bonus* hemos querido experimentar con una de las obras maestras de la pintura nórdica expresionista. Evidentemente el experimento (más bien un juego) carece de rigor pues la red no ha sido entrenada para reconocer emociones en esta tesisura.

Han podido más el interés por aprender y la curiosidad que la prudencia.

```
explain_img_path = '../data/munch/the_scream.png'

res <- predict(model, img_preprocess(explain_img_path))
colnames(res) <- names(labels)
print(res)
```

```
##          ANGER      DISGUST       FEAR        JOY      NEUTRAL     SADNESS
## [1,] 0.0006293501 0.001220247 5.358278e-05 0.9772571 0.008772788 0.01206672
##          SURPRISE
## [1,] 1.592846e-07

library(magick)
n_superpixels = 46
```

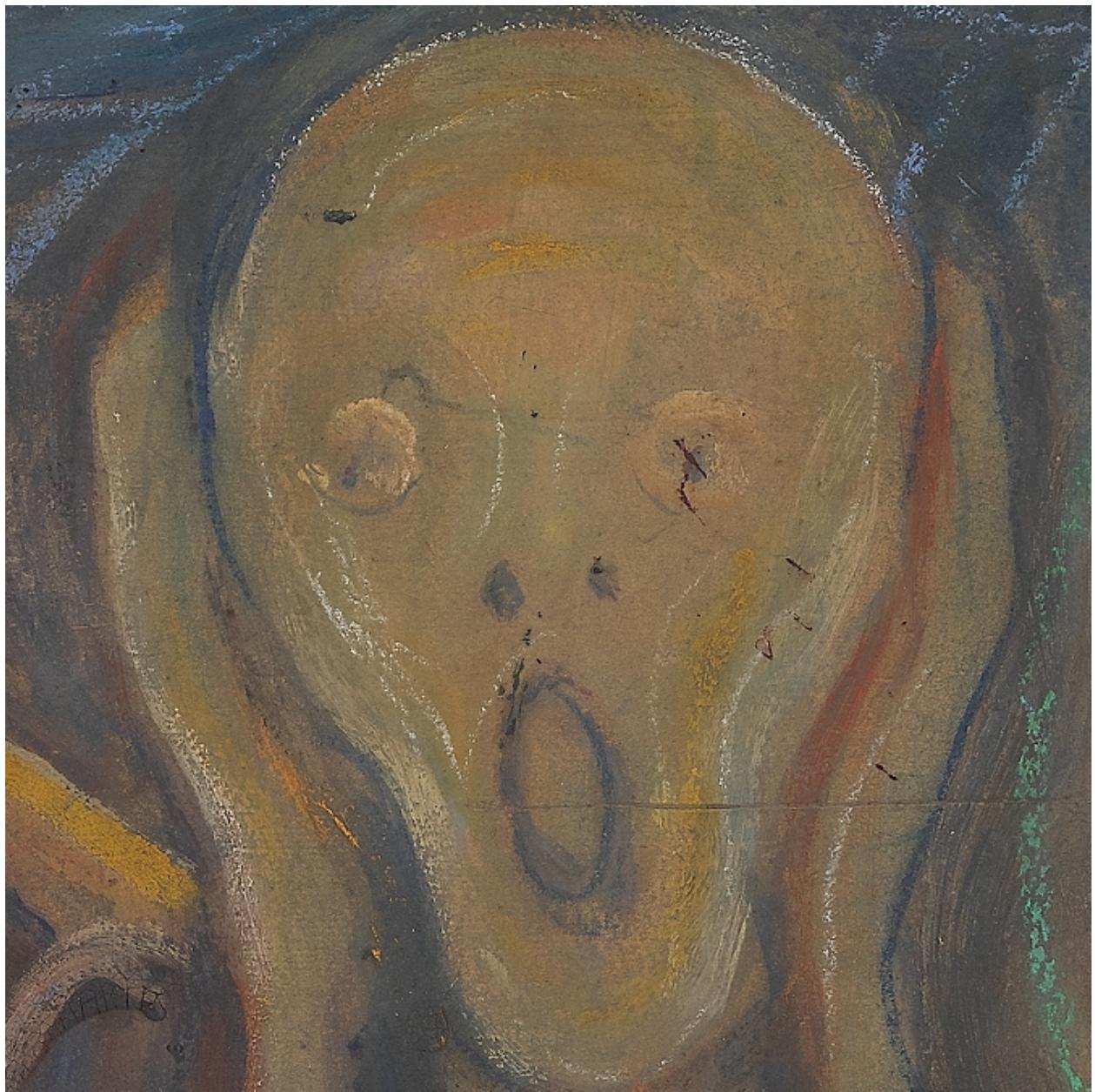
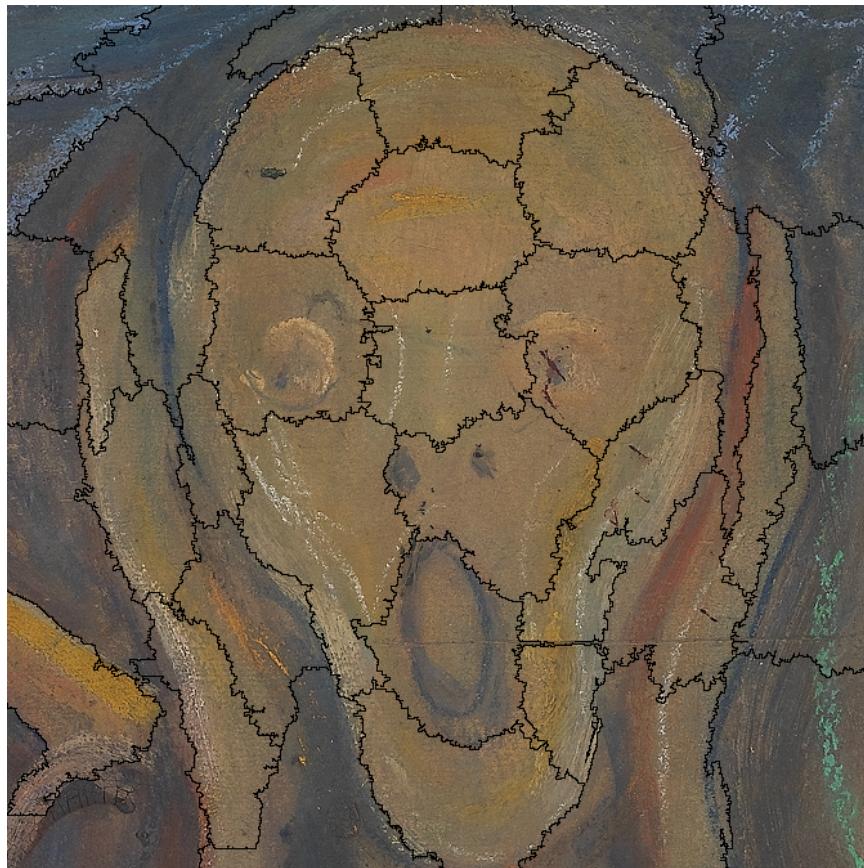


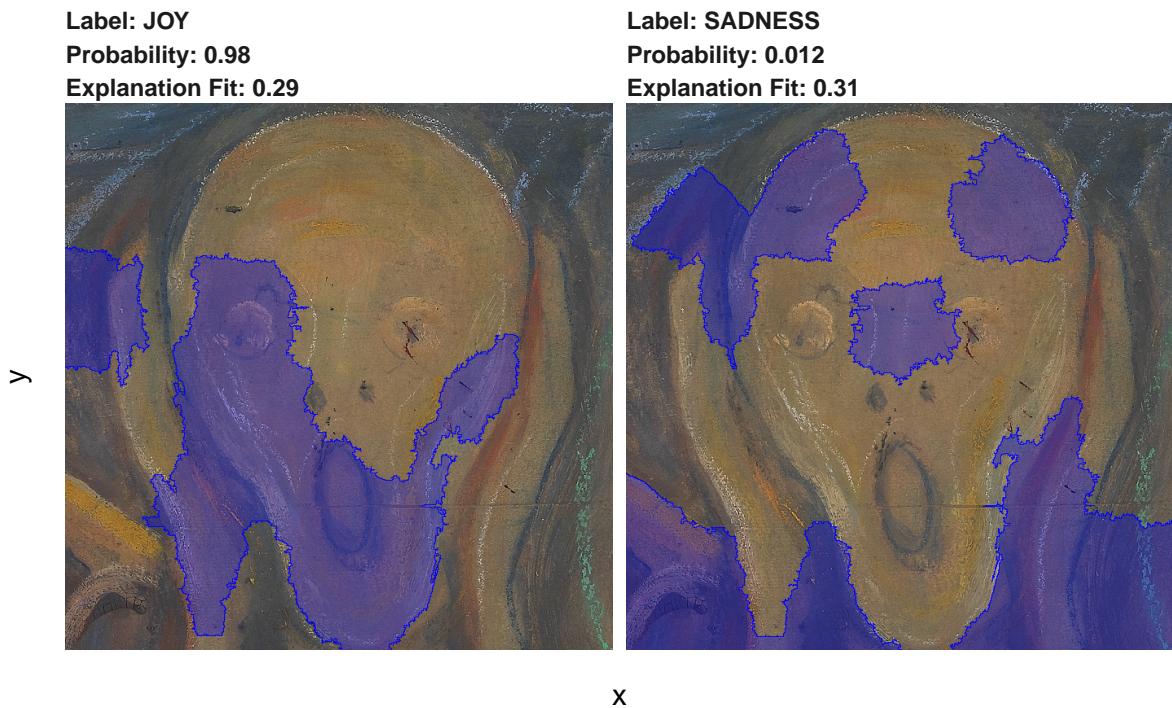
Figure 3: E. Munch, la cara más famosa del mundo

```
# Mostraremos los superpixels para poder evaluar cuáles son las zonas que se utilizan antes de aplicar
# Un superpixel es cada una de las zonas en las que se segmenta la imagen para facilitar las explicaciones
# Estos superpixels deberían contener patrones específicos de la imagen, por ello visualizarlos nos permite
# Tenemos que tener en cuenta que si las características importantes de la imagen se cortan en demasiados
# Cuanto más grande sea el objeto que buscamos en relación al tamaño de la imagen, menos superpixels devolverá
```

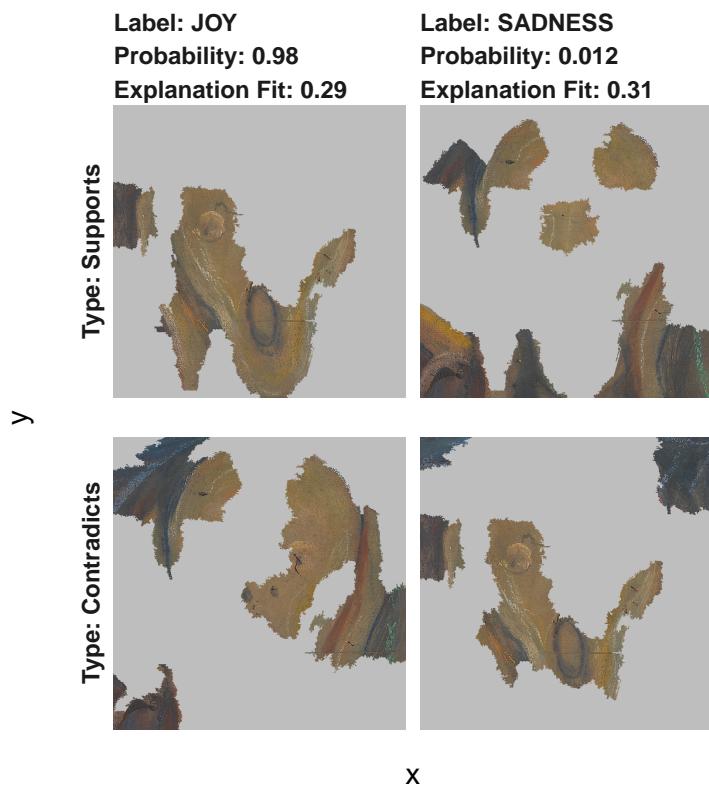
```
plot_superpixels(explain_img_path, n_superpixels)
```



```
# Generación de la explicación
explanation <- explain(
  explain_img_path,
  explainer,
  n_labels = 2,
  n_features = 3 * n_superpixels %/% 4, # limitamos el cardinal del conjunto de features para la explicación
  n_superpixels = n_superpixels)
# Visualización de los resultados
plot_image_explanation(as.data.frame(explanation))
```

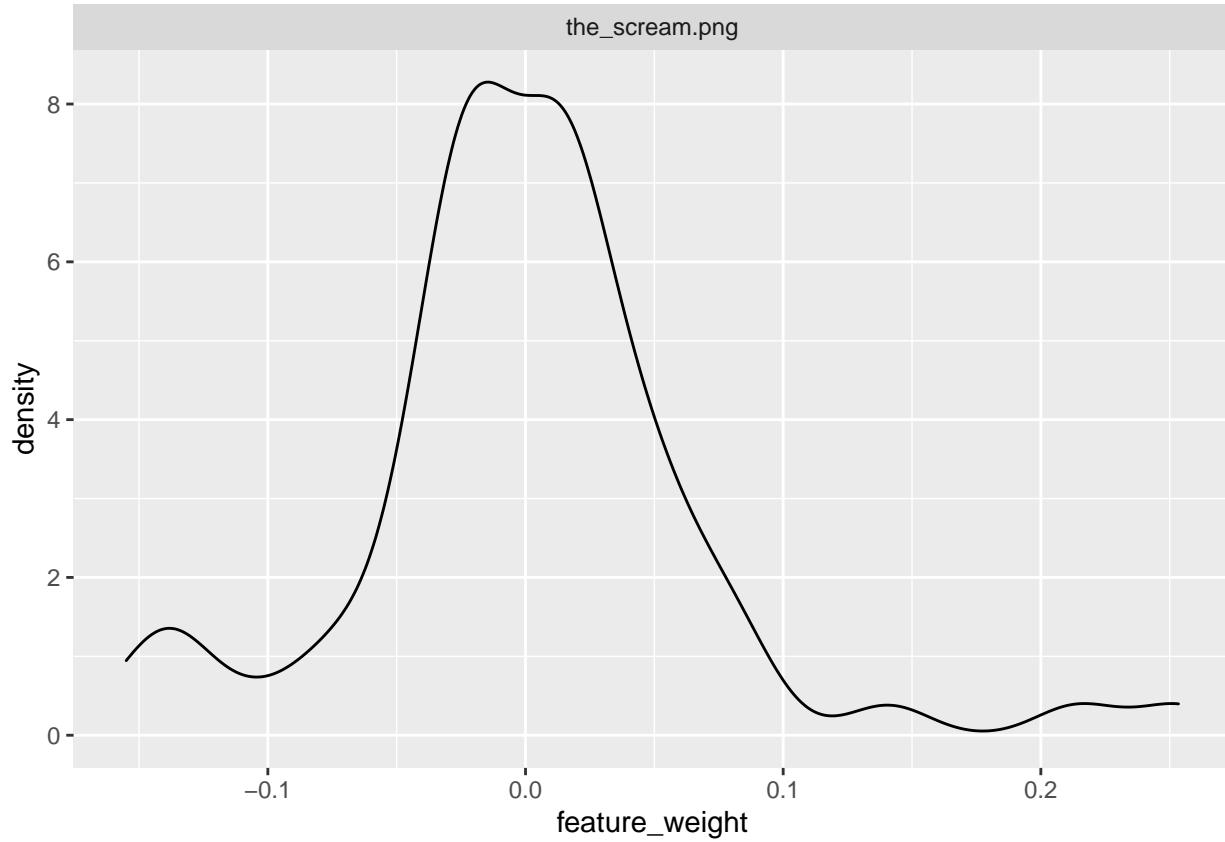


```
plot_image_explanation(as.data.frame(explanation),
                      display = 'block',
                      show_negative = TRUE)
```

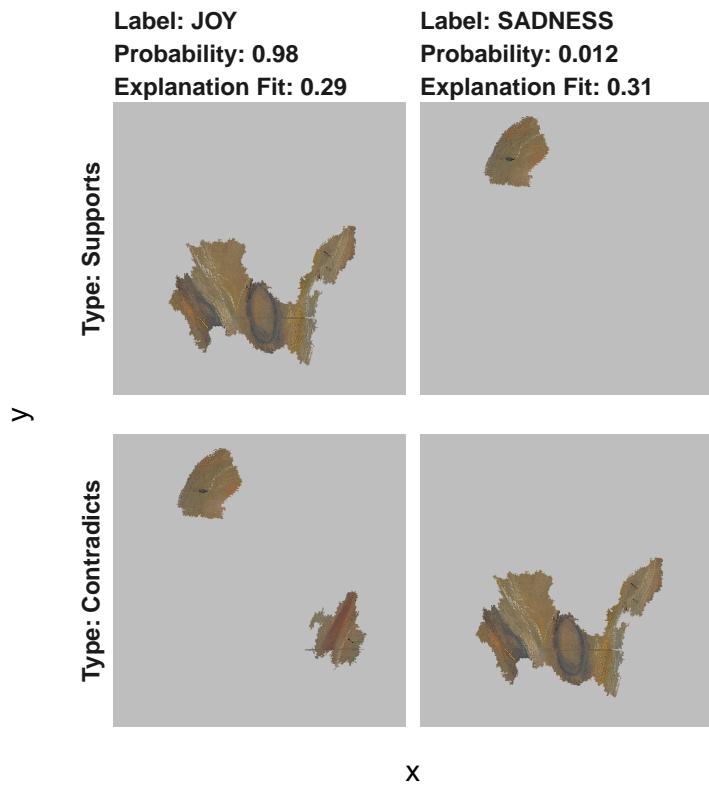


```
# Se puede precisar un poco más mostrando la gráfica de los pesos de las características para establecer
library(ggplot2)
explanation %>%
```

```
ggplot(aes(x = feature_weight)) +  
  facet_wrap(~ case, scales = "free") +  
  geom_density()
```



```
plot_image_explanation(as.data.frame(explanation),  
                      display = 'block',  
                      show_negative = TRUE,  
                      threshold = 0.07)
```



Conclusiones bonus

En este caso lo que llama poderosamente la atención es que la red esté segura al 98% de que el famoso cuadro de Munch es un grito de alegría, lo cual sería toda una bomba informativa en los círculos de arte.

A parte de esto, poco tenemos que decir, pues se trata como ya hemos mencionado de un juego más que un experimento o valoración.

Esperamos que os haya gustado y divertido.

Apéndice

Antecedentes: el experimento en clase

Durante una sesión práctica se encuestó a los alumnos para saber cuan buenos reconocedores de emociones son. En el siguiente apartado se muestra la matriz de confusión del resultado de la prueba.

Matriz de confusión de las respuestas de la clase

```
library(gdata)

## gdata: read.xls support for 'XLS' (Excel 97-2004) files ENABLED.
##
## gdata: read.xls support for 'XLSX' (Excel 2007+) files ENABLED.
##
## Attaching package: 'gdata'
```

```

## The following object is masked from 'package:stats':
##
##     nobs

## The following object is masked from 'package:utils':
##
##     object.size

## The following object is masked from 'package:base':
##
##    startsWith

# Importamos el fichero CSV
data = read.csv("../data/csv/Respuestas_emociones.csv", stringsAsFactors= FALSE)

# Muestra la estructura interna del objeto
str(data)

## 'data.frame': 880 obs. of 2 variables:
##   $ actual : chr "Anger" "Anger" "Anger" "Anger" ...
##   $ predicted: chr "Anger" "Anger" "Fear" "Joy" ...
# 'data.frame':880 obs. of 2 variables:
# $ actual : chr"Anger" "Anger" "Anger" "Anger" ...
# $ predicted: chr"Anger" "Anger" "Fear" "Joy" ...

library(caret)

## Loading required package: lattice

# creamos la matriz de confusión
confusionMatrix(
  # un factor de clases predichas
  data = as.factor(data$predicted),
  # un factor de clases que se utilizará como resultados verdaderos
  reference = as.factor(data$actual)
)

## Confusion Matrix and Statistics
##
##          Reference
## Prediction Anger Disgust Fear Joy Neutral Sadness Surprise
##   Anger      61     22    4  10      7      5      5
##   Disgust     15     69    5   7      0     27      0
##   Fear        5     34    8   0      0     10     15
##   Joy         3     04   60      3      2      0
##   Neutral     27     14   29  16     97     26      0
##   Sadness     13     24    8   4      2     61      0
##   Surprise     8     048   5      1      1    112
##
## Overall Statistics
##
##           Accuracy : 0.5614
##           95% CI : (0.5279, 0.5945)
##   No Information Rate : 0.15
##   P-Value [Acc > NIR] : < 2.2e-16
##
##           Kappa : 0.4888

```

```

## 
##  Mcnemar's Test P-Value : NA
## 
## Statistics by Class:
## 
##          Class: Anger Class: Disgust Class: Fear Class: Joy
## Sensitivity           0.46212      0.52273     0.25758    0.54545
## Specificity           0.92914      0.92781     0.94519    0.98442
## Pos Pred Value        0.53509      0.56098     0.45333    0.83333
## Neg Pred Value        0.90731      0.91678     0.87826    0.93812
## Prevalence            0.15000      0.15000     0.15000    0.12500
## Detection Rate        0.06932      0.07841     0.03864    0.06818
## Detection Prevalence  0.12955      0.13977     0.08523    0.08182
## Balanced Accuracy     0.69563      0.72527     0.60138    0.76494
##          Class: Neutral Class: Sadness Class: Surprise
## Sensitivity           0.8818       0.46212     0.8485
## Specificity           0.8545       0.93182     0.9158
## Pos Pred Value        0.4641       0.54464     0.6400
## Neg Pred Value        0.9806       0.90755     0.9716
## Prevalence            0.1250       0.15000     0.1500
## Detection Rate        0.1102       0.06932     0.1273
## Detection Prevalence  0.2375       0.12727     0.1989
## Balanced Accuracy     0.8682       0.69697     0.8821

```

Una representación más visual:

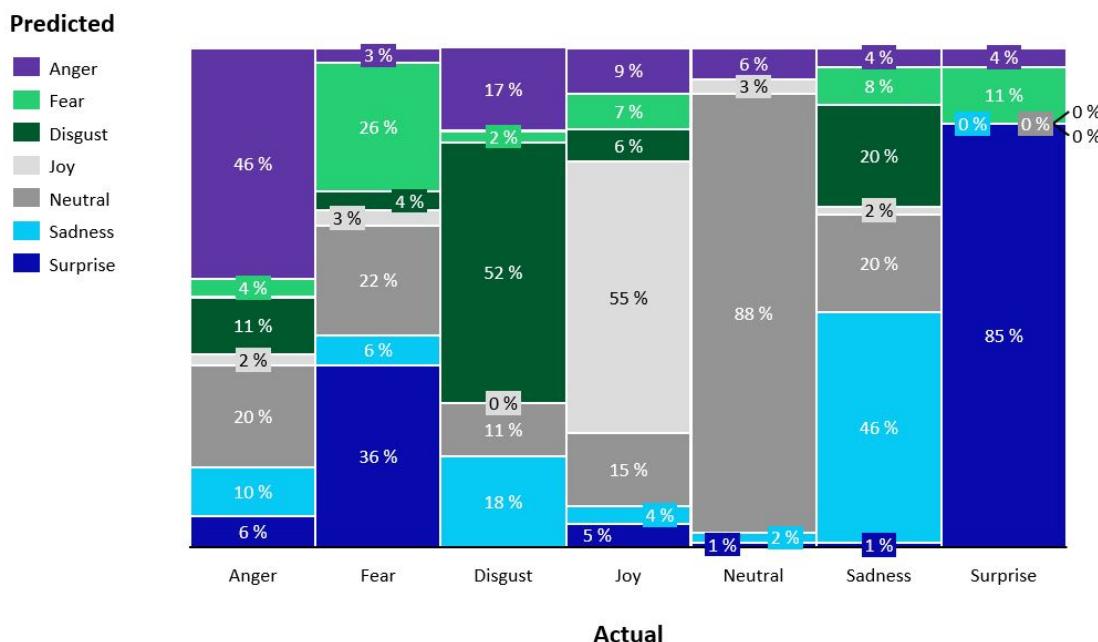


Figure 4: Distribución de las etiquetas asignadas a cada emoción (columnas suman 100%)

Observaciones del resultado de la encuesta en clase

- El accuracy medio de los alumnos es aproximadamente un 56%, creemos que esta precisión tan baja para un humano puede ser debida a dos factores:
 1. falta de contexto, pues se trata de imágenes no animadas y sin mayor información que la puramente gráfica
 2. Falta de referencias, es decir, ninguna de las personas encuestadas conocía previamente las expresiones de los avatares expuestos
- Los alumnos son “conservadores” a la hora de identificar emociones. La etiqueta más puesta por los alumnos, con diferencia, es “Neutral” (209 asignaciones, seguida de “Sorpresa” con 175 asignaciones). Entendemos que los alumnos ponen esta etiqueta cuando no ven que la cara esté transmitiendo ninguna emoción. Sería interesante comprobar si el modelo, para aquellas emociones que los alumnos clasifican erróneamente como “Neutral”, les asigna una emoción pero con una probabilidad más alejada del 100% (es decir, si las emociones en las que el modelo “duda” más, son las que los alumnos clasifican como neutral)
- “Neutral”, “Sorpresa” y “Alegría” son las emociones en las que más han acertado los alumnos (accuracy del 88%, 85% y 55%). Mientras que “Miedo”, “Enfado” y “Desgusto” son en las que más hemos fallado. A lo mejor es cierto eso que dicen de que *millenials* y *Generación Z* nos hemos criado entre algodones, hasta el punto que no estamos familiarizados ni sabemos reconocer las emociones negativas ;-).