Shellfish Toxicity Forecasting with Deep Learning

Ocean Hack Week 2021

Johnathan Evanilla

Configuración del entorno

In [1]:
```
# Establecer la variable de entorno para python
Sys.setenv("RETICULATE_PYTHON"="C:/Users/EQUIPO/.ai-navigator/conda/envs/OHW")
```

In [2]:
```
# Cargar los bibliotecas
library(keras)
library(dplyr)
library(keras3)
```

```
Registered S3 methods overwritten by 'keras':
  method                                from
  as.data.frame.keras_training_history keras3
  plot.keras_training_history          keras3
  print.keras_training_history         keras3
  r_to_py.R6ClassGenerator             keras3


Attaching package: 'dplyr'


The following objects are masked from 'package:stats':

    filter, lag


The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union



Attaching package: 'keras3'


The following objects are masked from 'package:keras':

    %<-active%, %py_class%, activation_elu, activation_exponential,
    activation_gelu, activation_hard_sigmoid, activation_linear,
    activation_relu, activation_selu, activation_sigmoid,
    activation_softmax, activation_softplus, activation_softsign,
    activation_tanh, adapt, application_densenet121,
    application_densenet169, application_densenet201,
    application_efficientnet_b0, application_efficientnet_b1,
    application_efficientnet_b2, application_efficientnet_b3,
    application_efficientnet_b4, application_efficientnet_b5,
    application_efficientnet_b6, application_efficientnet_b7,
    application_inception_resnet_v2, application_inception_v3,
    application_mobilenet, application_mobilenet_v2,
    application_mobilenet_v3_large, application_mobilenet_v3_small,
    application_nasnetlarge, application_nasnetmobile,
    application_resnet101, application_resnet101_v2,
    application_resnet152, application_resnet152_v2,
    application_resnet50, application_resnet50_v2, application_vgg16,
    application_vgg19, application_xception, bidirectional,
    callback_backup_and_restore, callback_csv_logger,
    callback_early_stopping, callback_lambda,
    callback_learning_rate_scheduler, callback_model_checkpoint,
    callback_reduce_lr_on_plateau, callback_remote_monitor,
    callback_tensorboard, clone_model, constraint_maxnorm,
    constraint_minmaxnorm, constraint_nonneg, constraint_unitnorm,
    count_params, custom_metric, dataset_boston_housing,
    dataset_cifar10, dataset_cifar100, dataset_fashion_mnist,
    dataset_imdb, dataset_imdb_word_index, dataset_mnist,
    dataset_reuters, dataset_reuters_word_index, freeze_weights,
    from_config, get_config, get_file, get_layer, get_vocabulary,
    get_weights, image_array_save, image_dataset_from_directory,
    image_load, image_to_array, imagenet_decode_predictions,
    imagenet_preprocess_input, initializer_constant,
    initializer_glorot_normal, initializer_glorot_uniform,
```

```
    initializer_he_normal, initializer_he_uniform,
    initializer_identity, initializer_lecun_normal,
    initializer_lecun_uniform, initializer_ones,
    initializer_orthogonal, initializer_random_normal,
    initializer_random_uniform, initializer_truncated_normal,
    initializer_variance_scaling, initializer_zeros, install_keras,
    keras, keras_model, keras_model_sequential, Layer,
    layer_activation, layer_activation_elu,
    layer_activation_leaky_relu, layer_activation_parametric_relu,
    layer_activation_relu, layer_activation_softmax,
    layer_activity_regularization, layer_add, layer_additive_attention,
    layer_alpha_dropout, layer_attention, layer_average,
    layer_average_pooling_1d, layer_average_pooling_2d,
    layer_average_pooling_3d, layer_batch_normalization,
    layer_category_encoding, layer_center_crop, layer_concatenate,
    layer_conv_1d, layer_conv_1d_transpose, layer_conv_2d,
    layer_conv_2d_transpose, layer_conv_3d, layer_conv_3d_transpose,
    layer_conv_lstm_1d, layer_conv_lstm_2d, layer_conv_lstm_3d,
    layer_cropping_1d, layer_cropping_2d, layer_cropping_3d,
    layer_dense, layer_depthwise_conv_1d, layer_depthwise_conv_2d,
    layer_discretization, layer_dot, layer_dropout, layer_embedding,
    layer_flatten, layer_gaussian_dropout, layer_gaussian_noise,
    layer_global_average_pooling_1d, layer_global_average_pooling_2d,
    layer_global_average_pooling_3d, layer_global_max_pooling_1d,
    layer_global_max_pooling_2d, layer_global_max_pooling_3d,
    layer_gru, layer_hashing, layer_input, layer_integer_lookup,
    layer_lambda, layer_layer_normalization, layer_lstm, layer_masking,
    layer_max_pooling_1d, layer_max_pooling_2d, layer_max_pooling_3d,
    layer_maximum, layer_minimum, layer_multi_head_attention,
    layer_multiply, layer_normalization, layer_permute,
    layer_random_brightness, layer_random_contrast, layer_random_crop,
    layer_random_flip, layer_random_rotation, layer_random_translation,
    layer_random_zoom, layer_repeat_vector, layer_rescaling,
    layer_reshape, layer_resizing, layer_rnn, layer_separable_conv_1d,
    layer_separable_conv_2d, layer_simple_rnn,
    layer_spatial_dropout_1d, layer_spatial_dropout_2d,
    layer_spatial_dropout_3d, layer_string_lookup, layer_subtract,
    layer_text_vectorization, layer_unit_normalization,
    layer_upsampling_1d, layer_upsampling_2d, layer_upsampling_3d,
    layer_zero_padding_1d, layer_zero_padding_2d,
    layer_zero_padding_3d, learning_rate_schedule_cosine_decay,
    learning_rate_schedule_cosine_decay_restarts,
    learning_rate_schedule_exponential_decay,
    learning_rate_schedule_inverse_time_decay,
    learning_rate_schedule_piecewise_constant_decay,
    learning_rate_schedule_polynomial_decay, loss_binary_crossentropy,
    loss_categorical_crossentropy, loss_categorical_hinge,
    loss_cosine_similarity, loss_hinge, loss_huber, loss_kl_divergence,
    loss_mean_absolute_error, loss_mean_absolute_percentage_error,
    loss_mean_squared_error, loss_mean_squared_logarithmic_error,
    loss_poisson, loss_sparse_categorical_crossentropy,
    loss_squared_hinge, mark_active, metric_auc,
    metric_binary_accuracy, metric_binary_crossentropy,
    metric_categorical_accuracy, metric_categorical_crossentropy,
    metric_categorical_hinge, metric_cosine_similarity,
    metric_false_negatives, metric_false_positives, metric_hinge,
    metric_mean, metric_mean_absolute_error,
    metric_mean_absolute_percentage_error, metric_mean_iou,
    metric_mean_squared_error, metric_mean_squared_logarithmic_error,
    metric_mean_wrapper, metric_poisson, metric_precision,
```

```
    metric_precision_at_recall, metric_recall,
    metric_recall_at_precision, metric_root_mean_squared_error,
    metric_sensitivity_at_specificity,
    metric_sparse_categorical_accuracy,
    metric_sparse_categorical_crossentropy,
    metric_sparse_top_k_categorical_accuracy,
    metric_specificity_at_sensitivity, metric_squared_hinge,
    metric_sum, metric_top_k_categorical_accuracy,
    metric_true_negatives, metric_true_positives, new_callback_class,
    new_layer_class, new_learning_rate_schedule_class, new_loss_class,
    new_metric_class, new_model_class, normalize, optimizer_adadelta,
    optimizer_adagrad, optimizer_adam, optimizer_adamax,
    optimizer_ftrl, optimizer_nadam, optimizer_rmsprop, optimizer_sgd,
    pad_sequences, pop_layer, predict_on_batch, regularizer_l1,
    regularizer_l1_l2, regularizer_l2, regularizer_orthogonal,
    set_vocabulary, set_weights, shape, test_on_batch,
    text_dataset_from_directory, time_distributed,
    timeseries_dataset_from_array, to_categorical, train_on_batch,
    unfreeze_weights, use_backend, with_custom_object_scope, zip_lists
```

In [3]:
```
# Cargar la funciones auxiliares
source("tutorial_functions.R")
```

Lectura y preprocesamiento

In [4]:
```
# Leer archivo csv
raw_data <- readr::read_csv("tutorial_data_test.csv")
#head(raw_data)
```

```
Rows: 6273 Columns: 22
-- Column specification --------------------------------------------------------
------------------------------------------------------------------------------
------
Delimiter: ","
chr   (2): id, location_id
dbl  (19): gap_days, year, classification, total_toxicity, t1, t2, t3, t4, t...
date  (1): date

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

In [5]:
```
# Transformación logaritmica
raw_data <- raw_data %>%
    log_inputs(vars = c("t1", "t2", "t3", "t4", "t5", "t6", "t7", "t8", "t9", "t
```

Generación de imágenes

In [6]:
```
#Generate images from data
image_list <- make_image_list(raw_data,
                              tox_levels =     c(0,10,30,80),
                              forecast_steps = 1,
                              n_steps =        2,
                              minimum_gap =    4,
                              maximum_gap =    10,
```

```
                                toxins =         c("t1", "t2", "t3", "t4", "t5", "
                                environmentals = c("sst_cum"))
```

In [7]:
```
#Splits image_list by year for grouping into train/test data
years <- sapply(image_list, function(x) {return(x$year)})
#str(years)
```

In [8]:
```
#image_list <- split(image_list,as.factor(years))
image_list <- split(image_list, years)
#str(image_list)
```

In [9]:
```
#configuration
YEARS_TRAINING <-   c("2014", "2016", "2017")
YEARS_TESTING <-    "2015"
```

In [10]:
```
#Make a training set
train <- pool_images_and_labels(image_list[YEARS_TRAINING], num_classes = 4)
```

In [11]:
```
#Make a test set
test <- pool_images_and_labels(image_list[YEARS_TESTING], num_classes = 4)
```

In [12]:
```
#load(file = "data.Rdata")
```

In [13]:
```
#dim(raw_data)
```

In [14]:
```
#length(years)
```

In [16]:
```
model <- keras::keras_model_sequential() %>%
  keras::layer_dense(units=64,
                     activation = "relu",
                     input_shape = dim(train$image)[2],
                     name = "input_layer") %>%
  keras::layer_dropout(rate = 0.4,
                       name = "dropout_1") %>%
  keras::layer_dense(units=32,
                     activation = "relu",
                     name = "hidden_1") %>%
  keras::layer_dropout(rate=0.3,
                       name = "dropout_2") %>%
  keras::layer_dense(units=16,
                     activation = "relu",
                     name = "hidden_2") %>%
  keras::layer_dropout(rate=0.2,
                       name = "dropout_3") %>%
  keras::layer_dense(units = 4,
                     activation = "softmax",
                     name = "output")

summary(model)
```

```
Model: "sequential_1"
<U+250F><U+2501><U+2501><U+2501><U+2501><U+2501><U+2501><U+2501><U+2501><U+2501><
U+2501><U+2501><U+2501><U+2501><U+2501><U+2501><U+2501><U+2501><U+2501><U+2501><U
+2501><U+2501><U+2501><U+2501><U+2501><U+2501><U+2501><U+2501><U+2501><U+2501><U+
2501><U+2501><U+2501><U+2501><U+2501><U+2501><U+2533><U+2501><U+2501><U+2501><U+2
501><U+2501><U+2501><U+2501><U+2501><U+2501><U+2501><U+2501><U+2501><U+2501><U+25
01><U+2501><U+2501><U+2501><U+2501><U+2501><U+2501><U+2501><U+2501><U+2501><U+250
1><U+2501><U+2501><U+2533><U+2501><U+2501><U+2501><U+2501><U+2501><U+2501><U+2501
><U+2501><U+2501><U+2501><U+2501><U+2501><U+2501><U+2501><U+2501><U+2513>
<U+2503> Layer (type)                    <U+2503> Output Shape            <U+2
503>       Param # <U+2503>
<U+2521><U+2501><U+2501><U+2501><U+2501><U+2501><U+2501><U+2501><U+2501><U+2501><
U+2501><U+2501><U+2501><U+2501><U+2501><U+2501><U+2501><U+2501><U+2501><U+2501><U
+2501><U+2501><U+2501><U+2501><U+2501><U+2501><U+2501><U+2501><U+2501><U+2501><U+
2501><U+2501><U+2501><U+2501><U+2501><U+2501><U+2547><U+2501><U+2501><U+2501><U+2
501><U+2501><U+2501><U+2501><U+2501><U+2501><U+2501><U+2501><U+2501><U+2501><U+25
01><U+2501><U+2501><U+2501><U+2501><U+2501><U+2501><U+2501><U+2501><U+2501><U+250
1><U+2501><U+2501><U+2547><U+2501><U+2501><U+2501><U+2501><U+2501><U+2501><U+2501
><U+2501><U+2501><U+2501><U+2501><U+2501><U+2501><U+2501><U+2501><U+2529>
┃ input_layer (Dense)              ┃ (None, 64)              ┃         1,728 ┃
+----------------------------------+-------------------------+---------------+
┃ dropout_1 (Dropout)              ┃ (None, 64)              ┃             0 ┃
+----------------------------------+-------------------------+---------------+
┃ hidden_1 (Dense)                 ┃ (None, 32)              ┃         2,080 ┃
+----------------------------------+-------------------------+---------------+
┃ dropout_2 (Dropout)              ┃ (None, 32)              ┃             0 ┃
+----------------------------------+-------------------------+---------------+
┃ hidden_2 (Dense)                 ┃ (None, 16)              ┃           528 ┃
+----------------------------------+-------------------------+---------------+
┃ dropout_3 (Dropout)              ┃ (None, 16)              ┃             0 ┃
+----------------------------------+-------------------------+---------------+
┃ output (Dense)                   ┃ (None, 4)               ┃            68 ┃
+-------------------------------------------------------------------------------+
 Total params: 4,404 (17.20 KB)
 Trainable params: 4,404 (17.20 KB)
 Non-trainable params: 0 (0.00 B)
```

In [17]:
```
str(train)
head(train$labels)
```

```
List of 7
 $ labels         : num [1:3413, 1:4] 1 1 0 1 1 1 1 1 1 0 ...
 $ image          : num [1:3413, 1:26] 0 0 0 0.379 0 ...
 $ classifications: num [1:3413] 0 0 2 0 0 0 0 0 0 3 ...
 $ toxicity       : num [1:3413] 0 0 45.21 4.18 2.65 ...
 $ locations      : chr [1:3413] "PSP12.01" "PSP24.13" "PSP10.11" "PSP15.25" ...
 $ dates          : num [1:3413] 16964 16238 17307 17336 17314 ...
 $ scaling_factors: NULL
```

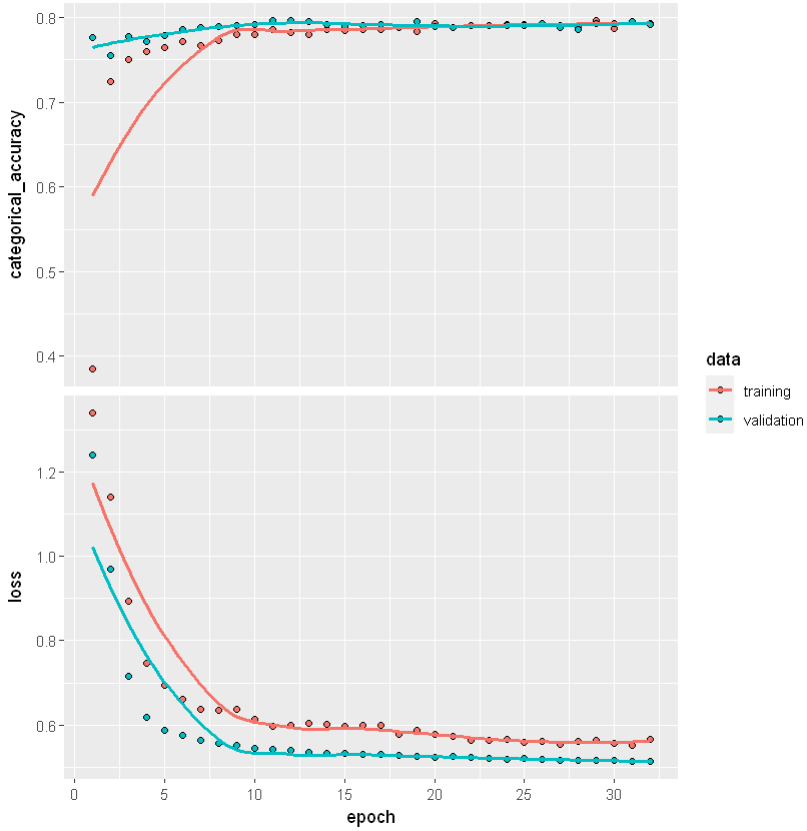
A matrix: 6 × 4 of type dbl

1	0	0	0
1	0	0	0
0	0	1	0
1	0	0	0
1	0	0	0
1	0	0	0

In [18]:
```
model %>% keras::compile(optimizer =  "adam",
                         loss =       "categorical_crossentropy",
                         metrics =    "categorical_accuracy")
```

In [19]:
```
history <- model %>%
    keras::fit(x = train$image,
               y = train$labels,
               batch_size = 128,
               epochs = 32,
               verbose=1,
               validation_split = 0.2,
               shuffle = TRUE)

plot(history)
```



In [20]:
```
metrics <- model %>%
  keras::evaluate(x = test$image,
                  y = test$labels)

predictions <- model %>% predict(test$image) %>% k_argmax() %>% as.array()
```

```
  #keras::predict_classes(test$image)

predicted_probs <- model %>%
      predict(test$image)

metrics
```

$categorical_accuracy 0.897273302078247
$loss 0.411398500204086

In [21]:
```
predictions
```

0·0·0·3·0·1·0·0·0·0·0·0·0·0·0·0·0·0·0·0·0·0·0·1·0·0·0·1·0·0·2·0·0·0·
0·0·0·0·0·0·1·0·0·0·0·0·0·0·0·0·0·0·0·0·0·2·0·1·1·0·0·0·0·0·1·0·
3·0·0·0·0·1·0·0·0·0·0·1·0·0·0·0·0·0·0·0·0·0·0·0·0·0·0·0·0·0·0·0·
0·0·0·0·0·0·0·0·0·0·0·0·0·0·0·0·0·0·0·0·0·0·0·0·0·0·0·0·0·0·0·0·
0·0·0·0·0·0·0·0·0·1·0·1·0·0·0·0·0·0·0·0·0·0·0·0·0·0·0·3·0·0·0·0·0·
0·0·0·0·0·0·0·0·0·0·0·0·0·0·0·0·0·3·0·0·0·0·0·3·1·0·0·3·0·0·0·0·0·
0·0·0·0·0·0·0·0·…·1·0·0·0·3·0·0·0·0·0·0·0·0·0·0·0·0·0·0·1·0·0·
0·0·0·0·0·0·0·0·0·1·1·0·0·0·0·0·0·0·0·0·0·0·0·0·0·0·0·0·0·0·0·0·
0·0·0·0·0·0·0·0·0·0·0·0·0·0·0·0·0·0·0·0·0·0·0·0·0·0·0·1·0·0·0·0·
3·0·0·0·0·0·0·0·0·0·0·0·0·0·0·0·0·0·0·0·0·0·0·0·0·0·0·0·0·0·0·0·
0·0·0·0·0·1·0·0·0·0·0·0·0·0·1·0·0·0·0·2·0·0·0·0·0·0·0·0·0·0·0·0·
0·0·0·1·0·0·0·0·0·0·0·0·0·3·0·0·0·0·0·0·0·0·0·0·0·0·0·0·0·1·0·0·
0·0·0·0·0·0·0·0·0·0·0·3·0·0·0·0·0

In [22]:
```
results <- dplyr::tibble(location = test$locations,
                         date = as.Date(as.numeric(test$dates), origin = as.Date
                         actual_classification = test$classifications,
                         predicted_classification = predictions) %>%
      dplyr::mutate(prob_0 = predicted_probs[,1]*100,
                    prob_1 = predicted_probs[,2]*100,
                    prob_2 = predicted_probs[,3]*100,
                    prob_3 = predicted_probs[,4]*100)

head(results)
```

A tibble: 6 × 8

location	date	actual_classification	predicted_classification	prob_0	prob_1	
<chr>	<date>	<dbl>	<dbl[1d]>	<dbl>	<dbl>	
PSP26.11	2015-08-11	0	0	94.531071	5.216153	0.
PSP10.25	2015-06-01	0	0	91.465598	7.823986	0.
PSP15.13	2015-09-03	0	0	98.496348	1.487587	0.
PSP12.002	2015-05-26	1	3	5.239873	17.535442	36.
PSP25.02	2015-07-14	0	0	96.321362	3.564764	0.
PSP27.46	2015-08-10	2	1	24.615008	52.814686	16.

In [23]:
```
metrics[2]
```

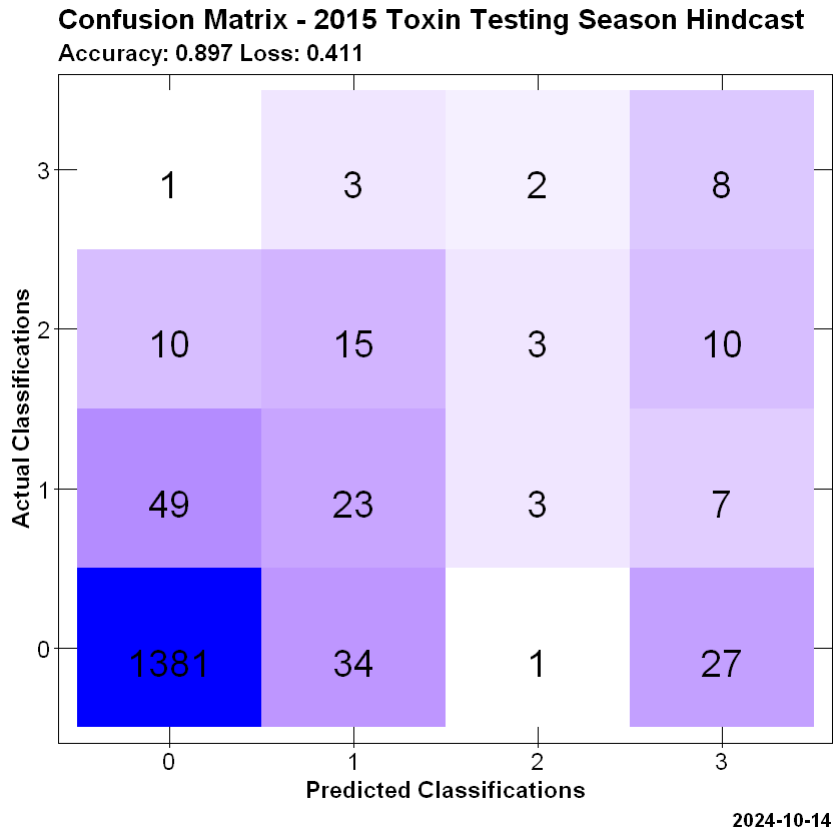
$loss = 0.411398500204086

In [24]:
```
num_levels <- 4
levels <- seq(from=0, to=(num_levels-1))

cm <- as.data.frame(table(predicted = factor(predictions, levels), actual = fact

confusion_matrix <- ggplot2::ggplot(data = cm,
                                    mapping = ggplot2::aes(x = .data$predicted,
  ggplot2::geom_tile(ggplot2::aes(fill = log(.data$Freq+1))) +
  ggplot2::geom_text(ggplot2::aes(label = sprintf("%1.0f", .data$Freq)), vjust =
  ggplot2::scale_fill_gradient(low = "white",
                               high = "blue") +
  ggplot2::labs(x = "Predicted Classifications",
                y = "Actual Classifications",
                title=paste("Confusion Matrix -", YEARS_TESTING, "Toxin Testing
                subtitle=paste("Accuracy:", round(metrics[1]$categorical_accurac
                caption=paste(Sys.Date())) +
  ggplot2::theme_linedraw() +
  ggplot2::theme(axis.text=  ggplot2::element_text(size=14),
                 axis.title= ggplot2::element_text(size=14,face="bold"),
                 title =     ggplot2::element_text(size = 14, face = "bold"),
                 legend.position = "none")

confusion_matrix
```


Confusion Matrix - 2015 Toxin Testing Season Hindcast
Accuracy: 0.897 Loss: 0.411
1
3
2
8
10
15
3
10
49
23
3
7
1381
34
1
27
Actual Classifications
Predicted Classifications
0
1
2
3
2024-10-14

In [ ]: