Aprendizaje Automático con Tensorflow y R

Edgar Ruiz

edgararuiz

theotheredgar

edgararuiz

RStudio

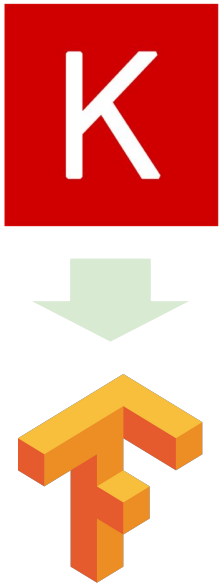
Tensorflow

- Procesa **aprendizaje automático y profundo**.
- Computación de alto desempeño
- Librería de código abierto
- Corre en una variedad de plataformas (CPUs, GPUs)
- Corre en una variedad de dispositivos (Desktops, servidores y teléfonos móvil)

Fuente: *https://www.tensorflow.org/*

Keras

- API para redes neuronales
- El mismo código funciona en CPU or GPU
- Modelos de aprendizaje profundo fácilmente
- Redes convolucionales y recurrentes
- Procesa arquitecturas de redes arbitrarias
- Varios ambientes: Tensorflow, CNTK o Theano



Fuente: *https://github.com/rstudio/keras*

RStudio

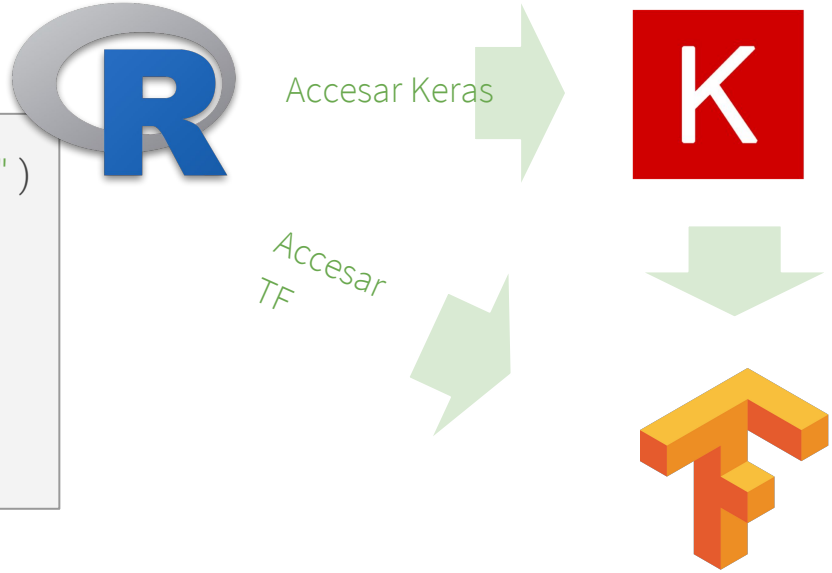
Integrando con R

Las tres tecnologías se pueden integrar mediante paquetes de R

```
install.packages("tensorflow")
install.packages("keras")

library(keras)

install_tensorflow()
install_keras()
```



RStudio

¿Por que Tensorflow, y también R?

Varios pasos conforman el Aprendizaje Automático (ML).
Todos los pasos pueden ser corridos con R

RStudio

Ademas...

R tiene excelentes paquetes que se especializan en el desarrollo de proyectos de Aprendizaje Automático.

Leér datos

Pre-procesar datos

Entrenar modelo

Validar modelo

Compartir resultados

RStudio

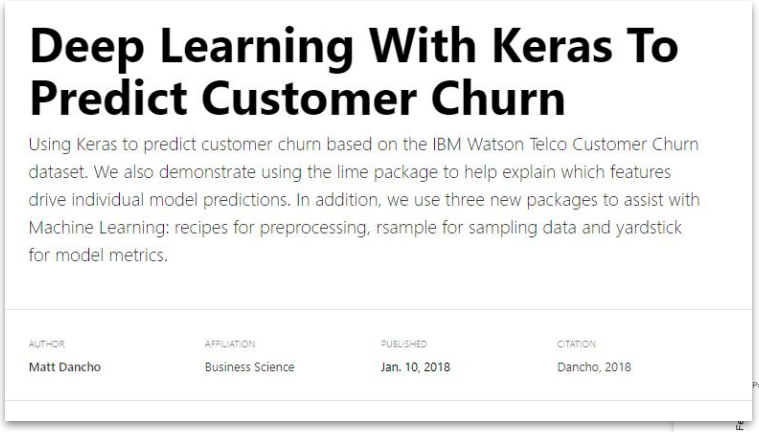
Prediciendo pérdida de clientes usando Aprendizaje Profundo en Keras

Demostración

RStudio

Análisis original

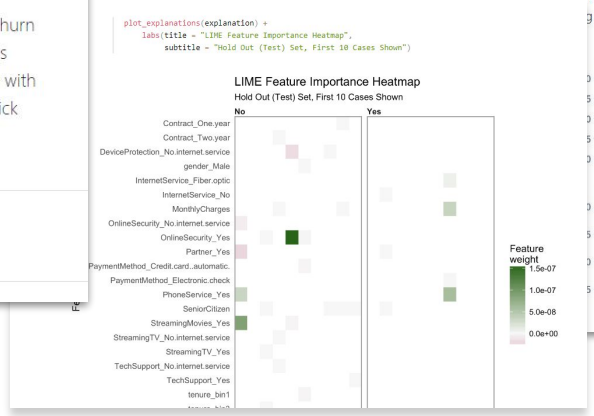
Ejemplo basado en el artículo: “**Deep Learning With Keras To Predict Customer Churn**”, publicado por **Matt Dancho**.

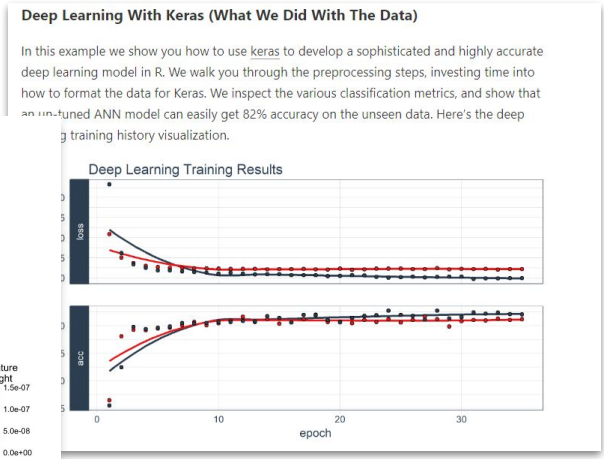


Deep Learning With Keras To Predict Customer Churn

Using Keras to predict customer churn based on the IBM Watson Telco Customer Churn dataset. We also demonstrate using the lime package to help explain which features drive individual model predictions. In addition, we use three new packages to assist with Machine Learning: recipes for preprocessing, rsample for sampling data and yardstick for model metrics.

AUTHOR	AFFILIATION	PUBLISHED	CITATION
Matt Dancho	Business Science	Jan. 10, 2018	Dancho, 2018



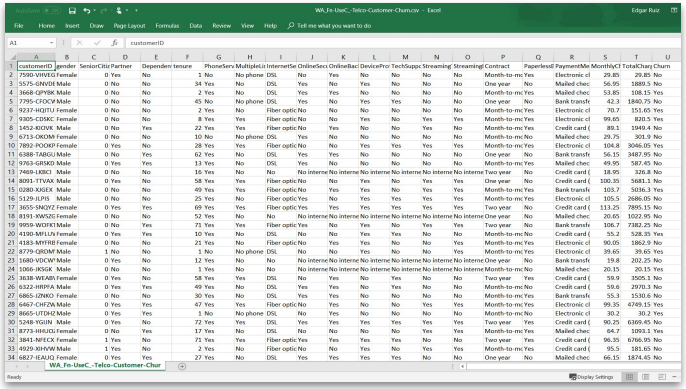


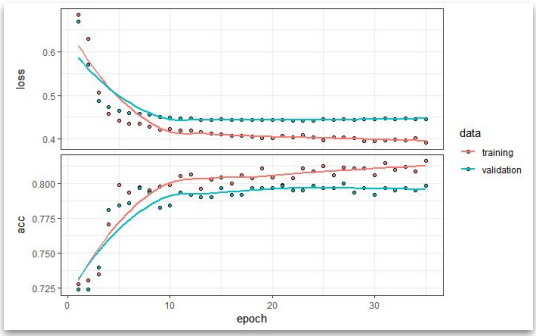
https://blogs.rstudio.com/tensorflow/posts/2018-01-11-keras-customer-churn/

RStudio®

Objetivo

Analizar datos de pérdida de clientes de una compañía de telecomunicaciones con R, Keras y Tensorflow.





RStudio®

Paquetes de R

Demostrar cuándo y cómo estos paquetes estos paquetes se pueden usar en un proyecto the auto-aprendizaje.

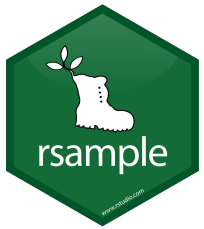
RStudio

Leér datos

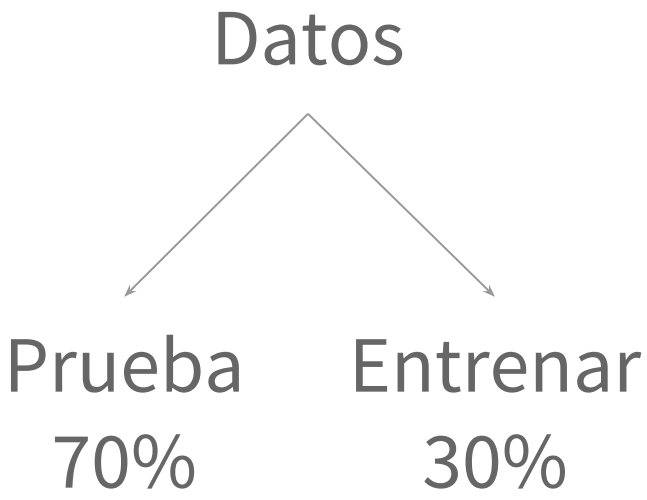
```
datos_perdimiento <- read_csv("customer_churn.csv")
```

```
# A tibble: 7,043 x 21
   customerID gender SeniorCitizen Partner Dependents tenure PhoneService
   <chr>      <chr>          <dbl> <chr>   <chr>       <dbl> <chr>
 1 7590-VHVEG Female             0 Yes     No              1 No
 2 5575-GNVDE Male               0 No      No             34 Yes
 3 3668-QPYBK Male               0 No      No              2 Yes
 4 7795-CFOCW Male               0 No      No             45 No
 5 9237-HQITU Female             0 No      No              2 Yes
 6 9305-CDSKC Female             0 No      No              8 Yes
 7 1452-KIOVK Male               0 No      Yes            22 Yes
 8 6713-OKOMC Female             0 No      No             10 No
 9 7892-POOKP Female             0 Yes     No             28 Yes
10 6388-TABGU Male               0 No      Yes            62 Yes
# ... with 7,033 more rows, and 14 more variables: MultipleLines <chr>
```

RStudio



Pre-procesar - *Muestra de datos*




```
separa_datos <- initial_split(
  datos_perdimiento,
  prop = 0.3
 )

tbl_entrenar <- training(separa_datos)

tbl_prueba  <- testing(separa_datos)
```

RStudio

Pre-procesar - *La receta!*

- Remueve columnas
- Omitir NAs
- Partir en categorías
- Conversión logarítmica
- Conversión lógica
- Variables ficticias
- Centrar valores
- Escalar valores

```
receta <- tbl_entrenar %>%
  recipe(Churn ~ .) %>%
  step_rm(customerID) %>%
  step_naomit(all_outcomes(), all_predictors()) %>%
  step_discretize(tenure,options = list(cuts = 6)) %>%
  step_log(TotalCharges) %>%
  step_mutate(Churn = ifelse(Churn=="Yes", 1, 0)) %>%
  step_dummy(all_nominal(), -all_outcomes()) %>%
  step_center(all_predictors(), -all_outcomes()) %>%
  step_scale(all_predictors(), -all_outcomes()) %>%
  prep()
```

Pre-procesar - *"Cocinar" receta*

```
x_tbl_entrenar <- receta %>%
  juice(all_predictors(),
        composition = "matrix")

y_vec_entrenar <- receta %>%
  juice(all_outcomes()) %>%
  pull()
```

La muestra de **entrenamiento** es extraída

```
baked_test <- bake(receta, tbl_prueba)

x_tbl_prueba <- baked_test %>%
  select(-Churn) %>%
  as.matrix()

y_vec_prueba <- baked_test %>%
  select(Churn) %>%
  pull()
```

La porción para **prueba** is procesada

RStudio

Entrenar el modelo - *Preparación*

Capa densa

Capa de pérdida

Capa densa

Capa de pérdida

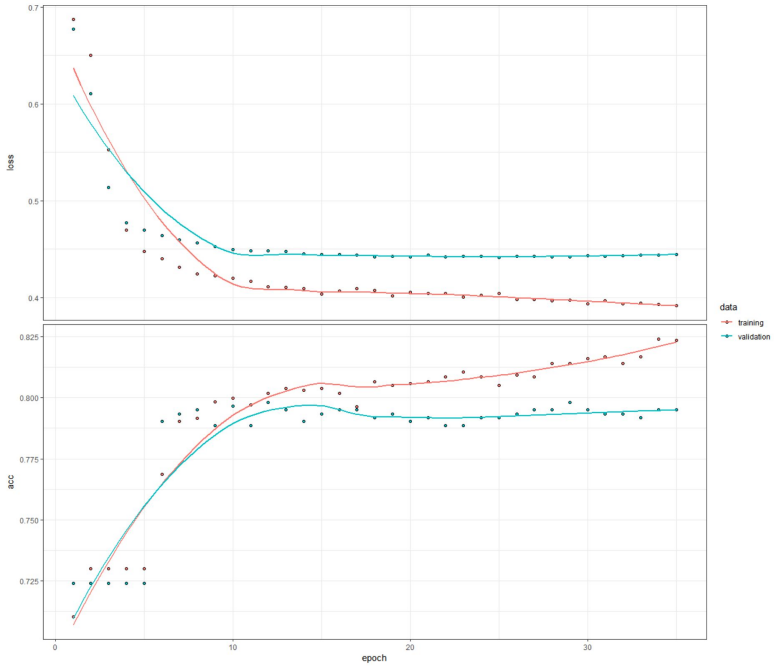
Capa densa

Compilar

```
model_keras <- keras_model_sequential() %>%
  layer_dense(units = 16,
              kernel_initializer = "uniform",
              activation = "relu",
              input_shape = ncol(x_tbl_entrenar)) %>%
  layer_dropout(rate = 0.1) %>%
  layer_dense(units = 16,
              kernel_initializer = "
              uniform", activation = "relu") %>%
  layer_dropout(rate = 0.1) %>%
  layer_dense(units = 1,
              kernel_initializer = "uniform",
              activation = "sigmoid") %>%
  compile(optimizer = 'adam',
          loss = 'binary_crossentropy',
          metrics = c('accuracy'))
```

RStudio

Entrenar el modelo - *Entrenar el modelo*




```
history <- fit(
  object = model_keras,
  x = x_tbl_entrenar,
  y = y_vec_entrenar,
  batch_size = 50,
  epochs = 35,
  validation_split = 0.30,
  verbose = 0
)
```

RStudio

Entrenar el modelo - *Predicciones*

```
yhat_keras_class_vec <- model_keras %>%
  predict_classes(x_tbl_prueba) %>%
  as.factor() %>%
  fct_recode(yes = "1", no = "0")
yhat_keras_prob_vec  <- model_keras %>%
  predict_proba(x_tbl_prueba) %>%
  as.vector()
test_truth <- y_vec_prueba %>%
  as.factor() %>%
  fct_recode(yes = "1", no = "0")
estimates_keras_tbl <- tibble(
  truth      = test_truth,
  estimate   = yhat_keras_class_vec,
  class_prob = yhat_keras_prob_vec)
estimates_keras_tbl
```

```
# A tibble: 4,920 x 3
   truth estimate class_prob
   <fct> <fct>          <dbl>
 1 no    yes          0.765
 2 yes   no           0.368
 3 yes   yes          0.749
 4 yes   yes          0.782
 5 no    yes          0.565
 6 no    no           0.106
 7 yes   yes          0.579
 8 no    no           0.00366
 9 no    no           0.221
10 no    no           0.0118
# ... with 4,910 more rows
```

RStudio

Validar modelo - *Mediciones*

```
           Truth
Prediction   no  yes
       no  3205  547
       yes  420  748
```

```
# A tibble: 2 x 3
  .metric  .estimator .estimate
  <chr>    <chr>          <dbl>
1 accuracy binary         0.803
2 kap      binary         0.477
```

```
# A tibble: 1 x 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 roc_auc binary         0.843
```

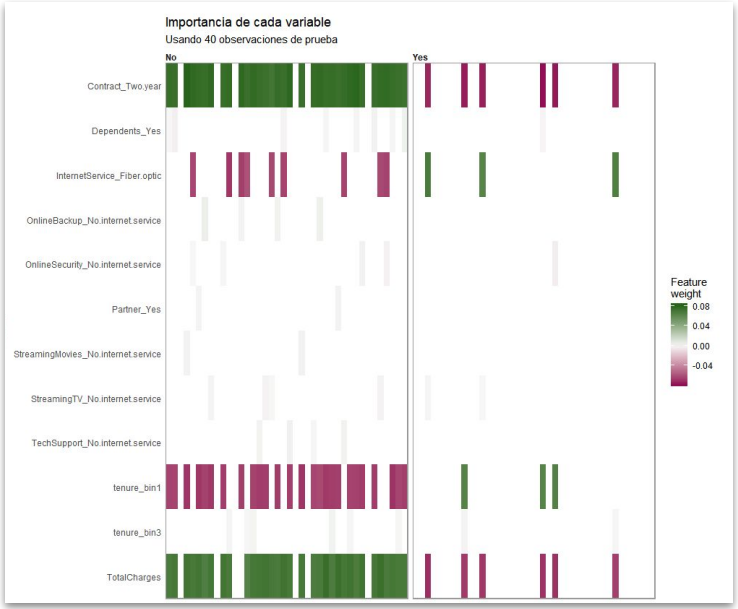
```
# A tibble: 2 x 3
  .metric   .estimator .estimate
  <chr>     <chr>          <dbl>
1 precision binary         0.640
2 recall    binary         0.578
```

```
# A tibble: 1 x 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 f_meas  binary         0.607
```

```
conf_mat(estimates_keras_tbl,
         truth, estimate)
metrics(estimates_keras_tbl,
        truth, estimate)
roc_auc(estimates_keras_tbl,
        truth, class_prob)
estimates_keras_tbl %>%
  precision(truth, estimate) %>%
  bind_rows(estimates_keras_tbl %>%
              recall(truth, estimate))
f_meas(estimates_keras_tbl,
       truth, estimate, beta = 1)
```

RStudio

Validar modelo - *Valores significantes*




```
explainer <- x_tbl_entrenar %>%
  as_tibble() %>%
  lime(model_keras, bin_continuous = FALSE)

explanation <-  x_tbl_entrenar %>%
  as.data.frame() %>%
  head(40) %>%
  lime::explain(
    explainer = explainer, n_labels = 1,
    N_features = 4, kernel_width = 0.5)
```

RStudio

Compartir resultados - *Reportes*

Aprendizaje Automatico con Tensorflow y R

Code

Install necessary packages

Hide

```
pkgs <- c("keras", "lime", "rsample", "recipes", "yardstick", "corrr")
install.packages(pkgs)
```

tidyverse

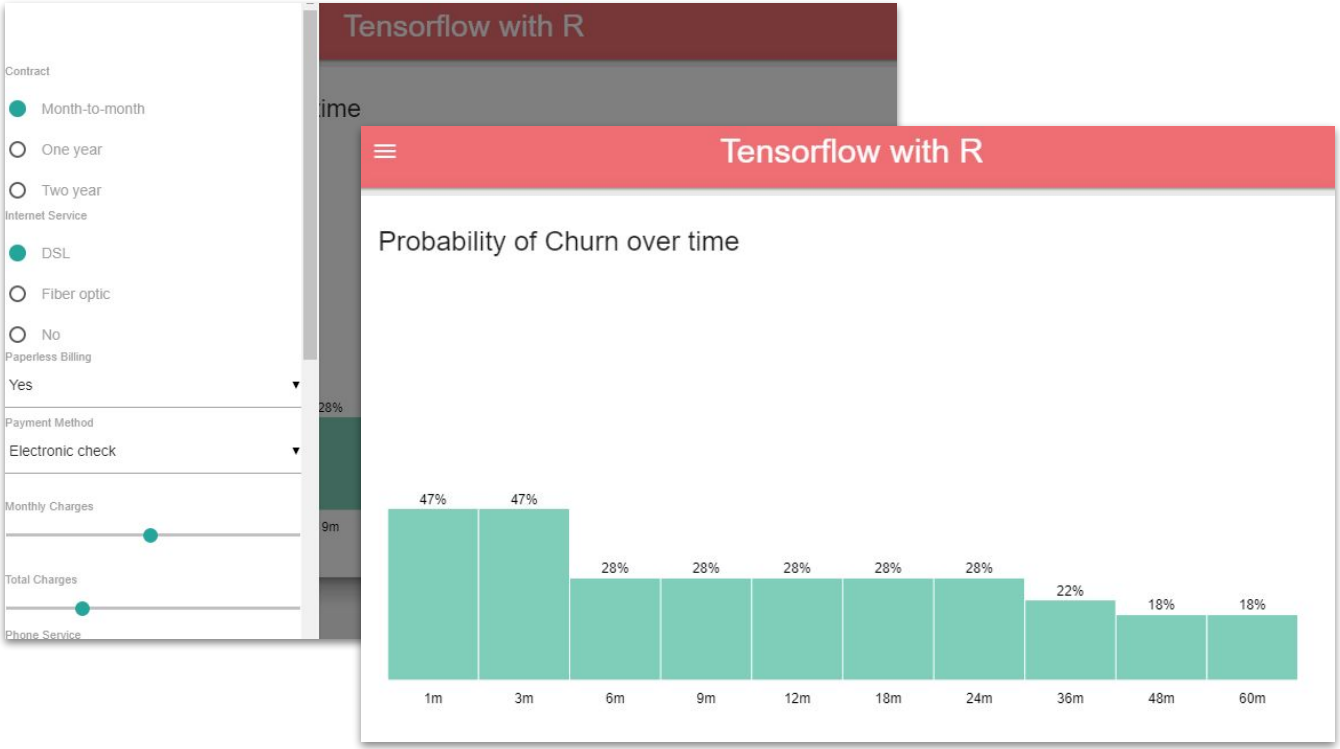
http://tidyverse.org/

The `tidyverse` packages provide an easy way to **import**, **tidy**, **transform** and **visualize** the data. Some of it's component R packages are:

- `dplyr`
- `tidyr`
- `readr`
- `ggplot2`

RStudio

Compartir resultados - *Aplicación*



Demostración

¡Pruébalo en tu teléfono!

rstd.io/churn

RStudio