

h2o




Plataforma para machine learning

Machine Learning As a Service (MLaaS)

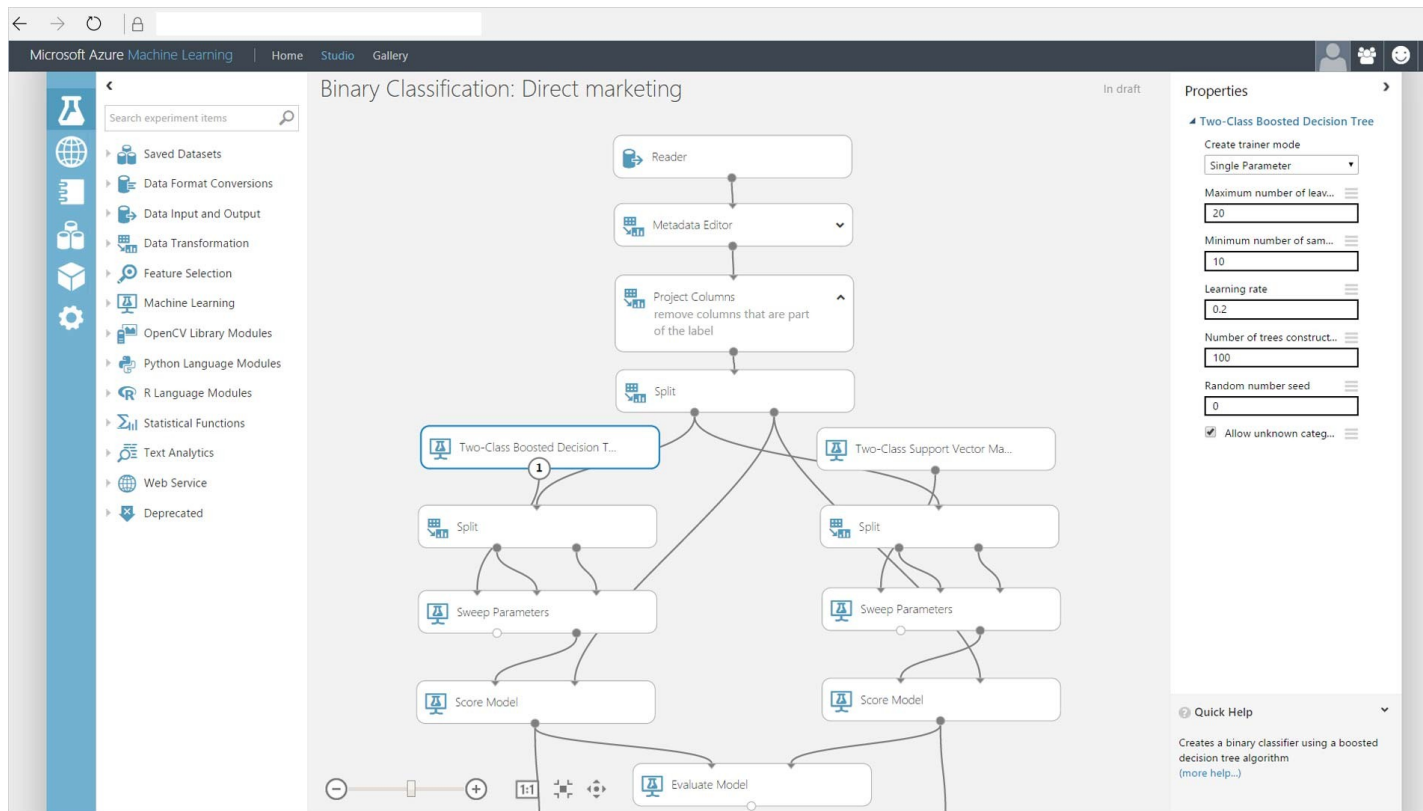
Es un concepto relativo a plataformas que ofrecen ordenadores potentes e interfaz para hacer los cálculos necesarios para machine learning reduciendo el tiempo de cómputo. Habitualmente permiten utilizar Scripts en R y Python.

En Abril de 2019 hay muchas, las más conocidas son:

Google Cloud. Para trabajar con R permite Rstudio Server, hay que crear una máquina virtual (VM) pero utiliza demasiados créditos \$ de memoria. Mejor posiblemente usar Linux y Python. En su versión Free (300\$ de uso) no merece la pena.

<div>  Por qué elegir Google Soluciones Productos Precios Empezar </div> <div>  Documentos Asistencia Consola  </div>						
<div> AI & Machine Learning Products <div> Contactar con ventas Probar gratis </div> </div>						
<div> EE. UU. </div>						
Preparación: niveles de escalabilidad predefinidos (precio por hora)		Preparación: tipos de máquinas de Cloud ML Engine (precio por hora)		Preparación: tipos de máquinas de Compute Engine (precio por hora)		Preparación: aceleración por hora)
BASIC	0,1900 USD	standard	0,1900 USD	n1-standard-4	0,1900 USD	NVIDIA_TESLA_K80
STANDARD_1	1,9880 USD	large_model	0,4736 USD	n1-standard-8	0,3800 USD	NVIDIA_TESLA_P4 (beta)
PREMIUM_1	16,5536 USD	complex_model_s	0,2836 USD	n1-standard-16	0,7600 USD	NVIDIA_TESLA_P4
BASIC_GPU	0,8300 USD	complex_model_m	0,5672 USD	n1-standard-32	1,5200 USD	NVIDIA_TESLA_V100
BASIC_TPU	4,6900 USD	complex_model_l	1,1344 USD	n1-standard-64	3,0400 USD	Con ocho núcleos TPU_V2*
CUSTOM	Consulta las tablas de tipos de máquinas.	standard_gpu	0,8300 USD	n1-standard-96	4,5600 USD	Predicción por lote de nodo)
		complex_model_m_gpu	2,5600 USD	n1-highmem-2	0,1184 USD	0,0791 USD

Microsoft Azure. Tiene un interfaz para Machine Learning tipo Eminer, llamado Azure Machine Learning Studio. Se pueden incorporar scripts de R o Python. En su versión Free es más lento que ejecutar los programas en nuestro ordenador local en Rstudio.



Amazon Web Service

El primero y más utilizado. En su versión libre SageMaker permite usar R y Python (no lo he probado todavía).

<https://aws.amazon.com/es/blogs/machine-learning/using-r-with-amazon-sagemaker/>

APRENDIZAJE AUTOMÁTICO **NOVEDADES**

Capa gratuita **PRUEBA GRATUITA**

Amazon SageMaker

250 horas

al mes de uso del cuaderno de notas t2.medium durante los primeros dos meses

Plataforma completamente administrada para crear, entrenar e implementar modelos de aprendizaje automático

250 horas al mes de uso del cuaderno de notas t2.medium durante los dos primeros meses

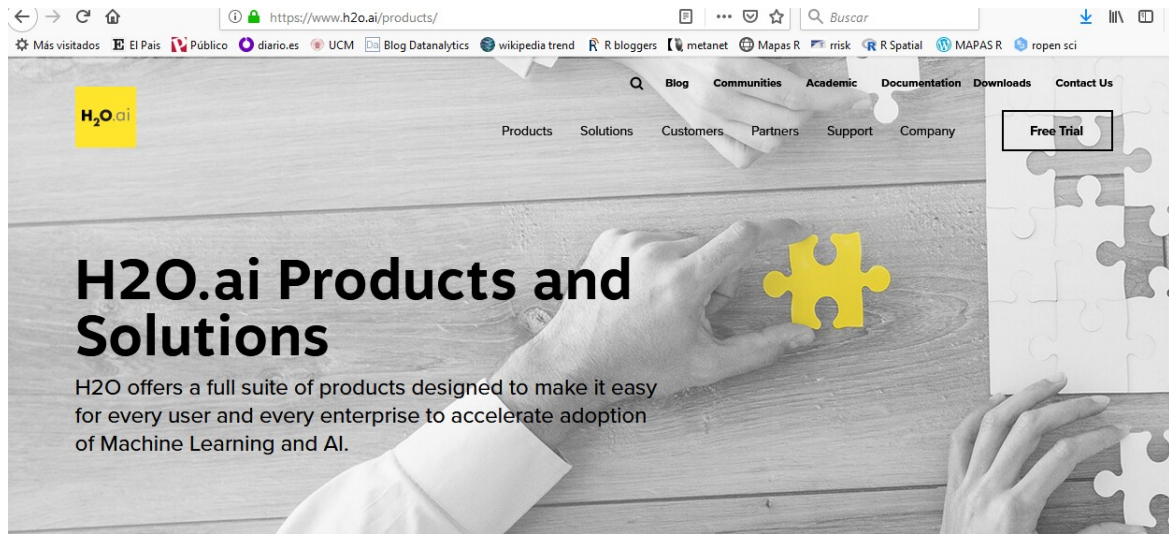
50 horas al mes de m4.xlarge para entrenamiento durante los dos primeros meses

125 horas al mes de m4.xlarge para alojamiento durante los dos primeros meses

^

H2o

Es una plataforma de Machine Learning. Permite integración con los MLaaS anteriores, pero además con Spark, o a través de librerías en R o Python de manera directa. Parte de los algoritmos se ejecutan en sus servidores de manera gratuita. A día de hoy (04-2019), es un servicio gratuito directo y eficaz respecto a los anteriores.



H2O.ai Products and Solutions

H2O offers a full suite of products designed to make it easy for every user and every enterprise to accelerate adoption of Machine Learning and AI.

Our Products

H2O.ai offers a range of AI and data science platforms from industry leading open source, H2O platform, to integrations for Apache Spark with Sparkling water, acceleration for NVIDIA GPU with H2O4GPU and the award-winning H2O Driverless AI platform that delivers an

Instalación

Para descargar, una opción seguir las instrucciones:

<http://h2o-release.s3.amazonaws.com/h2o/rel-yates/2/index.html>

Pero en su lugar recomiendo directamente ejecutar este script en R, que instala el paquete desde la web en lugar de meterlo en el disco duro primero.

```
# The following two commands remove any previously installed H2O packages for R.
if ("package:h2o" %in% search()) { detach("package:h2o", unload=TRUE) }
if ("h2o" %in% rownames(installed.packages())) { remove.packages("h2o") }

# Next, we download packages that H2O depends on.
pkgs <- c("RCurl","jsonlite")
for (pkg in pkgs) {
  if (! (pkg %in% rownames(installed.packages()))) { install.packages(pkg) }
}

# Now we download, install and initialize the H2O package for R.
install.packages("h2o", type="source", repos="http://h2o-release.s3.amazonaws.com/h2o/rel-yates/2/R")
```

Lo más importante es el Java. Lo normal es que solicite una instalación del Java más reciente para que funcione. Si nos pide java, hay que instalarlo y repetir los comandos anteriores.

Una vez instalado, se carga la librería y se inicia la conexión con Java.

```
# Finally, let's load H2O and start up an H2O cluster
library(h2o)
h2o.init()
```

Algunas cuestiones generales respecto al uso de h2o en R:

1) Se permite utilizar hasta 8 threads (cores) lo que hace mucho más rápido trabajar con h2o que con nuestro ordenador local. Si queremos resultados reproducibles, ponemos en `h2o.init` `nthreads=1` pero si queremos velocidad, `nthreads=8`.

2) Siempre hay que traducir los data frames de R a data frames tipo h2o con la función `as.h2o`. Para pasar a R se usa `as.data.frame` o trucos que se pueden ver aquí:

<https://stackoverflow.com/questions/42865609/how-to-convert-my-h2o-prediction-to-a-data-frame-in-a-fast-way>

3) No se suelen “nombrar variables”. Lo más cómodo es poner la variable dependiente en la primera columna, y contar las columnas restantes para incluir el número de columna en el algoritmo. Ver ejemplos.

4) Aunque es muy rápido en la computación, el Java abierto con `h2o.init()` se queda en la memoria RAM. El ordenador se puede ralentizar en general (es decir, navegar por internet a la vez que se ejecuta Rstudio+h2o puede ser muy lento).

Ver ejemplos de ejecución básica de redes en el código “h2o instalar.R”.

Opciones redes (lo llama deep learning pero no es exactamente):

```
h2o.deeplearning(x, y, training_frame, model_id = NULL,
  validation_frame = NULL, nfolds = 0,
  keep_cross_validation_predictions = FALSE,
  keep_cross_validation_fold_assignment = FALSE, fold_assignment = c("AUTO",
  "Random", "Modulo", "Stratified"), fold_column = NULL,
  ignore_const_cols = TRUE, score_each_iteration = FALSE,
  weights_column = NULL, offset_column = NULL, balance_classes = FALSE,
  class_sampling_factors = NULL, max_after_balance_size = 5,
  max_hit_ratio_k = 0, checkpoint = NULL, pretrained_autoencoder = NULL,
  overwrite_with_best_model = TRUE, use_all_factor_levels = TRUE,
  standardize = TRUE, activation = c("Tanh", "TanhWithDropout", "Rectifier",
  "RectifierWithDropout", "Maxout", "MaxoutWithDropout"), hidden = c(200,
  200), epochs = 10, train_samples_per_iteration = -2,
  target_ratio_comm_to_comp = 0.05, seed = -1, adaptive_rate = TRUE,
  rho = 0.99, epsilon = 1e-08, rate = 0.005, rate_annealing = 1e-06,
  rate_decay = 1, momentum_start = 0, momentum_ramp = 1e+06,
  momentum_stable = 0, nesterov_accelerated_gradient = TRUE,
  input_dropout_ratio = 0, hidden_dropout_ratios = NULL, l1 = 0, l2 = 0,
  max_w2 = 3.4028235e+38, initial_weight_distribution = c("UniformAdaptive",
  "Uniform", "Normal"), initial_weight_scale = 1, initial_weights = NULL,
  initial_biases = NULL, loss = c("Automatic", "CrossEntropy", "Quadratic",
  "Huber", "Absolute", "Quantile"), distribution = c("AUTO", "bernoulli",
  "multinomial", "gaussian", "poisson", "gamma", "tweedie", "laplace",
  "quantile", "huber"), quantile_alpha = 0.5, tweedie_power = 1.5,
  huber_alpha = 0.9, score_interval = 5, score_training_samples = 10000,
  score_validation_samples = 0, score_duty_cycle = 0.1,
  classification_stop = 0, regression_stop = 1e-06, stopping_rounds = 5,
  stopping_metric = c("AUTO", "deviance", "logloss", "MSE", "RMSE", "MAE",
  "RMSLE", "AUC", "lift_top_group", "misclassification",
  "mean_per_class_error"), stopping_tolerance = 0, max_runtime_secs = 0,
  score_validation_sampling = c("Uniform", "Stratified"),
  diagnostics = TRUE, fast_mode = TRUE, force_load_balance = TRUE,
  variable_importances = FALSE, replicate_training_data = TRUE,
  single_node_mode = FALSE, shuffle_training_data = FALSE,
  missing_values_handling = c("MeanImputation", "Skip"), quiet_mode = FALSE,
  autoencoder = FALSE, sparse = FALSE, col_major = FALSE,
  average_activation = 0, sparsity_beta = 0,
  max_categorical_features = 2147483647, reproducible = FALSE,
  export_weights_and_biases = FALSE, mini_batch_size = 1,
  categorical_encoding = c("AUTO", "Enum", "OneHotInternal", "OneHotExplicit",
  "Binary", "Eigen"), elastic_averaging = FALSE,
  elastic_averaging_moving_rate = 0.9,
  elastic_averaging_regularization = 0.001)
```

h2o permite **grid search** con cv como el caret. Ver archivo **grid search h2o.R**.

Por otra parte, h2o tiene el **paquete AutoML** que prueba diferentes algoritmos y ensamblado (stacking) con cv.

- Redes (h2o.deeplearning)
- Gbm (h2o.gbm)
- Random forest (h2o.drf)
- Logística o regresión, ambos LASSO (h2o.glm)

Ver ejemplos en el archivo **grid search h2o.R** y **h2o clasificación.R**.

Aunque funciona muy bien sin selección de variables, es bueno seleccionar antes como se ve aquí: **pruebas h2o con y sin.R**.