

# Industrial IoT Data Science with H2O



Jo-fai (Joe) Chow  
Data Scientist  
[joe@h2o.ai](mailto:joe@h2o.ai)  
@matlabulous

H2O Hands-on Workshop  
La Fosse Associates  
27<sup>th</sup> & 28<sup>th</sup> September, 2016

# About Me: Civil Engineer → Data Scientist

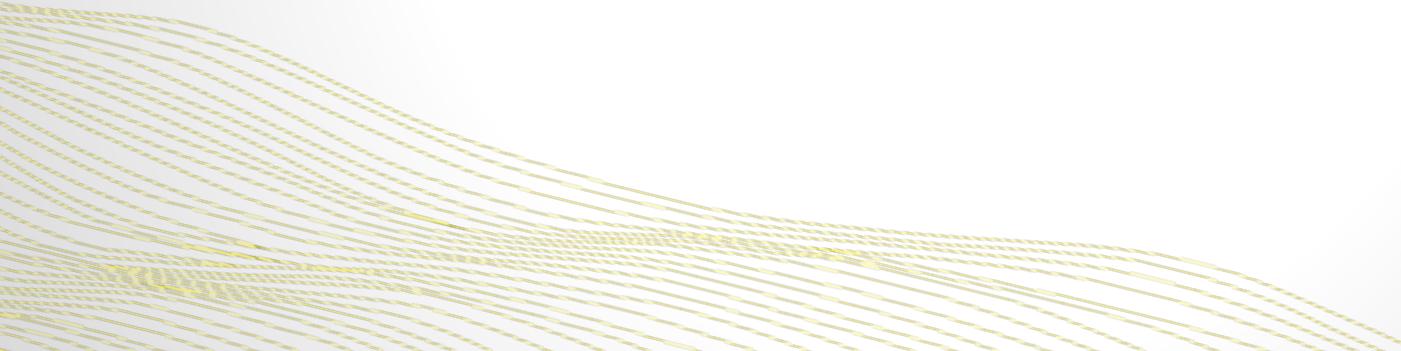
- 2005 - 2015
- Water Engineer
  - Consultant for Utilities
  - EngD Research
    - Machine learning + Water Engineering
  - ***Discovered H2O in 2014!***
- 2015 - Present
- Data Scientist
  - Virgin Media (UK)
  - Domino Data Lab (US)
  - H2O.ai (US)

Why? Long story – see [bit.ly/joe\\_h2o\\_talk2](http://bit.ly/joe_h2o_talk2)

# Agenda

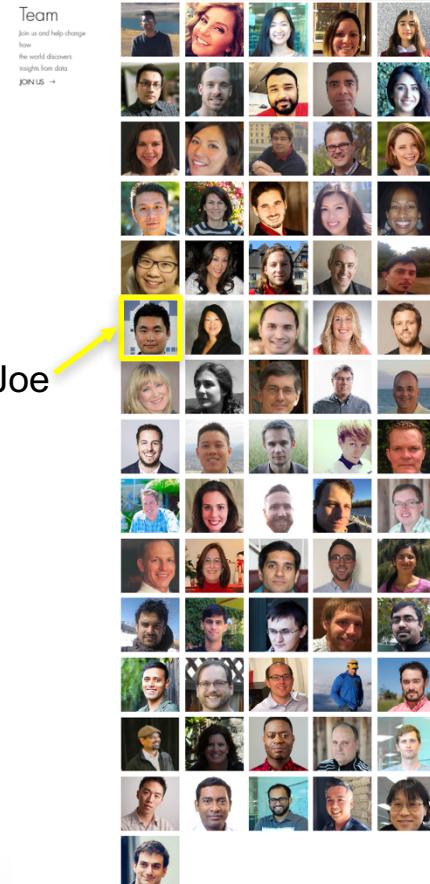
- **Workshop #1 (Today)**
  - About H2O
  - Data Munging in H2O
  - Case 1 (Basics)
    - Classification
    - Importing data
    - Exploring data
    - Training models
    - Evaluating models
    - Tuning models (manual)
- **Workshop #2 (Tmr.)**
  - Case 1 (Advanced)
    - Early stopping
    - Tuning models (auto)
    - Stacking models
    - Saving/loading models
  - Case 2
    - Anomaly detection
  - Other H2O features

# About H2O



# About H2O.ai

- **H2O.ai, the Company**
  - Team: 80 (71 shown)
  - Founded in 2012,
  - HQ: Mountain View, California
- **H2O, the Platform**
  - Open Source (Apache 2.0)
  - R, Python, Scala, Java and Web Interfaces
  - Distributed Algorithms that Scale to Big Data
  - Works with Laptop, Cloud, Hadoop & Spark



# H2O Platform Overview

- Core algorithms written in high performance Java.
- Fast, distributed and scalable.
- APIs available in R, Python, Scala, REST/JSON and web.
- Works with laptop, cloud, Hadoop and Spark



# Current Algorithm Overview

## Statistical Analysis

- Linear Models (GLM)
- Naïve Bayes

## Ensembles

- Random Forest
- Distributed Trees
- Gradient Boosting Machine
- R Package - Stacking / Super Learner

Workshop

## Deep Neural Networks

- Multi-layer Feed-Forward Neural Network
- Auto-encoder
- Anomaly Detection
- Deep Features

## Clustering

- K-Means

## Dimension Reduction

- Principal Component Analysis
- Generalized Low Rank Models

Joe's Strata Hadoop  
London Talk  
[bit.ly/joe\\_h2o\\_talk4](http://bit.ly/joe_h2o_talk4)

## Solvers & Optimization

- Generalized ADMM Solver
- L-BFGS (Quasi Newton Method)
- Ordinary least-Square Solver
- Stochastic Gradient Descent

## Data Munging

- Scalable Data Frames
- Sort, Slice, Log Transform

Joe's LondonR Talk  
[bit.ly/joe\\_h2o\\_talk3](http://bit.ly/joe_h2o_talk3)

# H2O Overview

Computer Science (CS)

Artificial Intelligence (A.I.)

Machine Learning (ML)

Deep Learning (DL)

hot hot hot hot hot



# New Stuff!

## H2O + RStudio RSparkling

sparklyr Home dplyr ML ▾ Extensions Deployment Examples Reference

**Overview**

- Algorithms
- Transformers
- Examples
- K-Means Clustering
- Logistic Regression
- PCA
- Random Forest
- Gradient Boosting Machine
- Deep Learning
- Grid Search
- Exporting Models
- Additional Resources

### Sparkling Water (H2O) Machine Learning Overview

The **rsparkling** extension package provides bindings to H2O's distributed [machine learning](#) algorithms via **sparklyr**. In particular, rsparkling allows you to access the machine learning routines provided by the [Sparkling Water](#) Spark package.

Together with sparklyr's **dplyr** interface, you can easily create and tune H2O machine learning workflows on Spark, orchestrated entirely within R.

**rsparkling** provides a few simple conversion functions that allow the user to transfer data between Spark DataFrames and H2O Frames. Once the Spark DataFrames are available as H2O Frames, the **h2o** R interface can be used to train H2O machine learning algorithms on the data.

A typical machine learning pipeline with rsparkling might be composed of the following stages. To fit a model, you might need to:

1. Perform SQL queries through the sparklyr **dplyr** interface,
2. Use the `asf_*` and `ft_*` family of functions to generate new columns, or partition your data set,
3. Convert your training, validation and/or test data frames into H2O Frames using the `as_h2o_frame` function,
4. Choose an appropriate H2O machine learning algorithm to model your data,
5. Inspect the quality of your model fit, and use it to make predictions with new data.

### Installation

The **rsparkling** R package requires the **h2o** and **sparklyr** R packages to run. The sparklyr package can be installed from CRAN and instructions for how to install the h2o and rsparkling packages follow below.



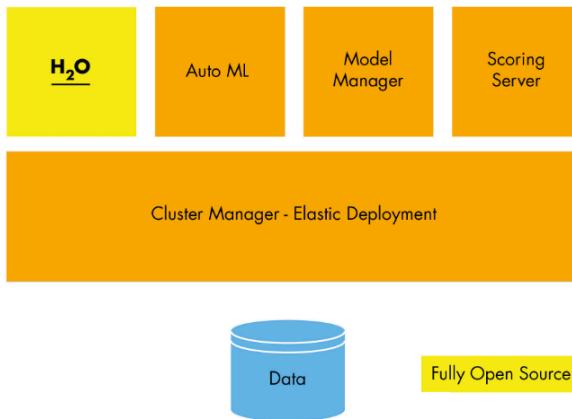
# Deep Water: Next-Gen Deep Learning in H2O

# New Stuff!



## STEAM

The Steam AI engine is an end-to-end platform that streamlines the entire process of building and deploying smart applications. Now data scientists and developers can launch turnkey compute environments for collaboratively training and deploying predictive models and integrate those models into real-time smart applications.



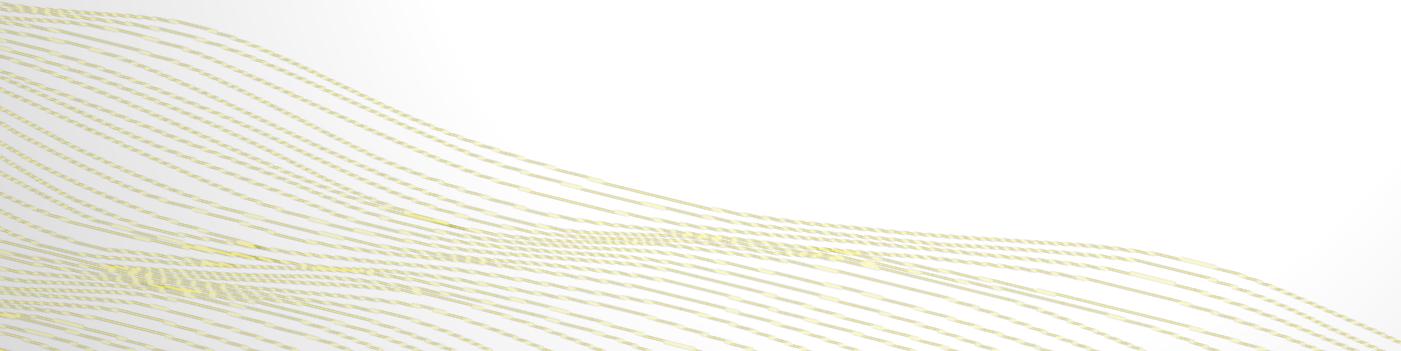
# Data Munging in H2O

by Jan Gorecki & Matt Dowle

[\(link to slides\)](#)

# Workshop #1

## Use Case 1: Classification



# Download Data & Code for Workshop

- Please go to

[bit.ly/h2o\\_iot\\_workshop1](https://github.com/woobe/H2O_London_Workshop)

No description or website provided. — Edit

5 commits 1 branch 0 releases 1 contributor Apache-2.0

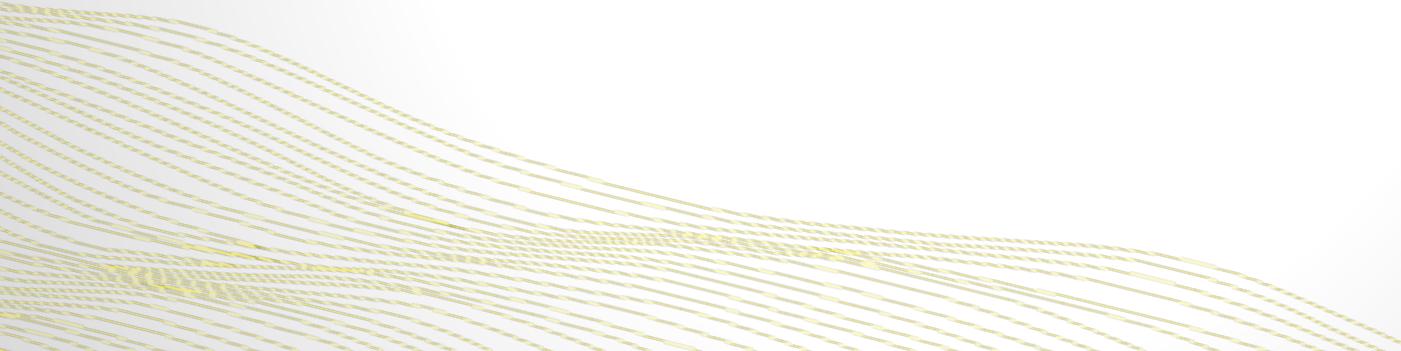
Branch: master ▾ New pull request Create new file Upload files Find file Clone or download ▾

File	Description	Time
code_day1	initial commit	13 hours ago
data	data for workshop	14 hours ago
slides	Slides from Sibjan Das	13 hours ago
.gitignore	Initial commit	14 hours ago
LICENSE	Initial commit	14 hours ago
README.md	Create README.md	13 hours ago

Workshop #1

Use Case 1: Classification

# Step 1: R Packages



# step\_01\_install\_packages.R

## Package ‘h2o’

```
1 # 
2 # Step 1: Install R Packages for Workshop
3 #
4 #
5 # -----
6 # Install "h2o" for machine learning
7 # Reference: http://www.h2o.ai/download/h2o/r
8 #
9 #
10 # The following two commands remove any previously installed H2O packages for R.
11 if ("package:h2o" %in% search()) { detach("package:h2o", unload=TRUE) }
12 if ("h2o" %in% rownames(installed.packages())) { remove.packages("h2o") }
13 #
14 # Next, we download packages that H2O depends on.
15 pkgs <- c("methods", "statmod", "stats", "graphics", "RCurl", "jsonlite", "tools", "utils")
16 for (pkg in pkgs) {
17   if (! (pkg %in% rownames(installed.packages()))) { install.packages(pkg) }
18 }
19 #
20 # Now we download, install and initialize the H2O package for R.
21 install.packages("h2o",
22                  type="source",
23                  repos=c("http://h2o-release.s3.amazonaws.com/h2o/rel-turing/7/R")))
24 #
25 # Quick test
26 suppressPackageStartupMessages(library(h2o))
27 h2o.init(nthreads = -1)
```

```
> suppressPackageStartupMessages(library(h2o))
> h2o.init(nthreads = -1)
Connection successful!
```

R is connected to the H2O cluster:

H2O cluster uptime:	1 minutes 891 milliseconds
H2O cluster version:	3.10.0.7
H2O cluster version age:	6 days
H2O cluster name:	H2O_started_from_R_jofaichow_ayn543
H2O cluster total nodes:	1
H2O cluster total memory:	3.28 GB
H2O cluster total cores:	8
H2O cluster allowed cores:	8
H2O cluster healthy:	TRUE
H2O Connection ip:	localhost
H2O Connection port:	54321
H2O Connection proxy:	NA
R Version:	R version 3.3.0 (2016-05-03)

# step\_01\_install\_packages.R

## Package ‘h2oEnsemble’

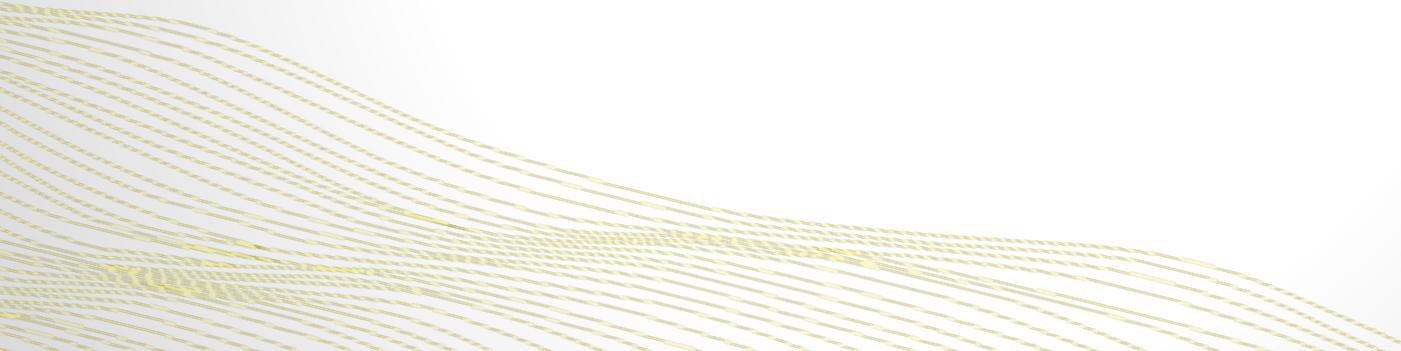
```
30 # -----
31 # Install "h2oEnsemble" for model stacking
32 # Reference: https://github.com/h2oai/h2o-3/tree/master/h2o-r/ensemble
33 #
34
35 # Install stable version (1.8)
36 install.packages(
37   "https://h2o-release.s3.amazonaws.com/h2o-ensemble/R/h2oEnsemble_0.1.8.tar.gz",
38   repos = NULL)
39
40 # Quick test
41 library(h2oEnsemble)
```

```
> library(h2oEnsemble)
h2oEnsemble (beta) for H2O >=3.0
Version: 0.1.8
Package created on 2016-03-29
```

Workshop #1

Use Case 1: Classification

# Step 2: Exploratory Analysis



# Data for Use Case 1: SECOM

**UCI** 

**Machine Learning Repository**  
Center for Machine Learning and Intelligent Systems

## SECOM Data Set

*Download:* [Data Folder](#), [Data Set Description](#)

**Abstract:** Data from a semi-conductor manufacturing process



<b>Data Set Characteristics:</b>	Multivariate	<b>Number of Instances:</b>	1567	<b>Area:</b>	Computer
<b>Attribute Characteristics:</b>	Real	<b>Number of Attributes:</b>	591	<b>Date Donated</b>	2008-11-19
<b>Associated Tasks:</b>	Classification, Causal-Discovery	<b>Missing Values?</b>	Yes	<b>Number of Web Hits:</b>	37895

**Source:**

Authors: Michael McCann, Adrian Johnston

# step\_02\_exploratory\_analysis.R

## Importing SECOM data

```
1 # -----  
2 # Step 2: Data Exploration  
3 # -----  
4  
5 # Start and connect to a local H2O cluster  
6 library(h2o)  
7 h2o.init(nthreads = -1)  
8  
9 # Import data from a local CSV file  
10 # Source: https://archive.ics.uci.edu/ml/machine-learning-databases/secom/  
11 secom <- h2o.importFile(path = "./data/secom.csv", destination_frame = "secom")  
12  
13 # (Optional) Demo - Importing files using URLs  
14 secom <- h2o.importFile(  
15   path = "https://github.com/woobie/H2O_London_Workshop/raw/master/data/secom.csv",  
16   destination_frame = "secom")  
17  
18 # (Optional) Demo - Converting R data frame into H2O data frame  
19 hdf_iris <- as.h2o(iris)  
20  
21 # (Optional) Turning off progress bar in R  
22 h2o.no_progress()
```

```
java version "1.8.0_72"  
Java(TM) SE Runtime Environment (build 1.8.0_72-b15)  
Java HotSpot(TM) 64-Bit Server VM (build 25.72-b15, mixed mode)  
  
Starting H2O JVM and connecting: ... Connection successful!  
  
R is connected to the H2O cluster:  
  H2O cluster uptime:      2 seconds 721 milliseconds  
  H2O cluster version:    3.10.0.7  
  H2O cluster version age: 7 days, 10 hours and 4 minutes  
  H2O cluster name:       H2O_started_from_R_jofaichow_cow128  
  H2O cluster total nodes: 1  
  H2O cluster total memory: 3.56 GB  
  H2O cluster total cores: 8  
  H2O cluster allowed cores: 8  
  H2O cluster healthy:    TRUE  
  H2O Connection ip:      localhost  
  H2O Connection port:    54321  
  H2O Connection proxy:   NA  
  R Version:              R version 3.3.0 (2016-05-03)  
  
>  
> # Import data from a local CSV file  
> # Source: https://archive.ics.uci.edu/ml/machine-learning-databases/secom/  
> secom <- h2o.importFile(path = "./data/secom.csv", destination_frame = "secom")  
|=====| 100%
```

Optional (different ways to import data)

# step\_02\_exploratory\_analysis.R

## Basic exploratory analysis

```
25 # Basic exploratory analysis
26 print(dim(secom)) # 1567 x 599
27 print(summary(secom$Classification))
28 # alternatively, use H2O flow to look at data (localhost:54321)
29
30 # Convert Classification to factor
31 secom$Classification <- as.factor(secom$Classification)
32 print(summary(secom$Classification))
```

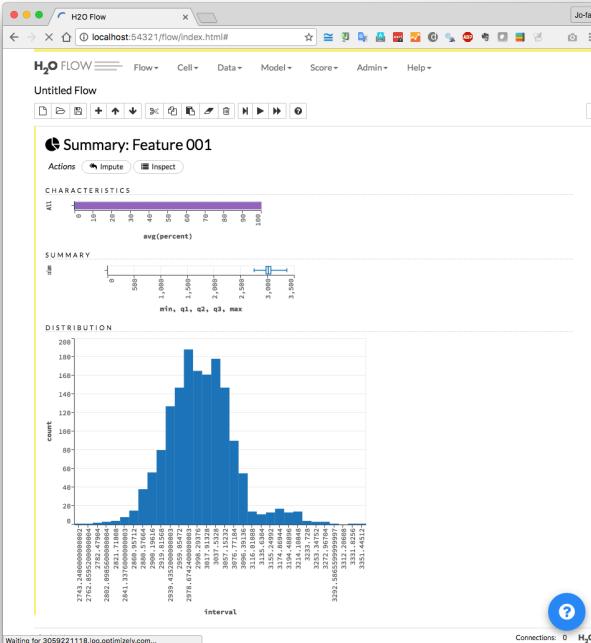
Convert -1 and 1 to categorical value

```
> # Basic exploratory analysis
> print(dim(secom)) # 1567 x 599
[1] 1567 599
> print(summary(secom$Classification))
Classification
Min.   :-1.0000
1st Qu.:-1.0000
Median :-1.0000
Mean   :-0.8673
3rd Qu.:-1.0000
Max.    : 1.0000
```

```
> # Convert Classification to factor
> secom$Classification <- as.factor(secom$Classification)
> print(summary(secom$Classification))
Classification
-1:1463
1 : 104
```

# Use H2O Flow (localhost:54321)

The screenshot shows the H2O Flow interface with the 'secom' dataset loaded. The top navigation bar includes 'Flow', 'Cell', 'Data', 'Model', 'Score', 'Admin', and 'Help'. Below the title 'Untitled Flow' is a toolbar with various icons for data manipulation. The main area displays the 'secom' dataset summary: Rows 1567, Columns 599, and Compressed Size 3MB. A detailed table of column summaries follows, listing features like ID, Feature 001, Feature 002, etc., with their respective types (e.g., int, real), missing values, and statistical details (min, max, mean, std). A 'Actions' section at the bottom right includes 'View Data', 'Split...', 'Build Model...', 'Predict', 'Download', and 'Export'. A question mark icon is in the bottom right corner.



Workshop #1

Use Case 1: Classification

# Step 3: Building & Evaluating Models



# step\_03\_basic\_models.R

## Define features & target

```
5 # -----
6 # Loading data (same as previous steps)
7 #
8
9 # Start and connect to a local H2O cluster
10 library(h2o)
11 h2o.init(nthreads = -1)
12
13 # Import data from a local CSV file
14 # Source: https://archive.ics.uci.edu/ml/machine-learning-databases/secom/
15 secom <- h2o.importFile(path = "./data/secom.csv", destination_frame = "secom")
16
17 # Convert Classification to factor
18 secom$Classification <- as.factor(secom$Classification)
19
20
21 # -----
22 # Define Targets and Features
23 #
24
25 target <- "Classification"
26 features <- setdiff(colnames(secom), c("ID", "Classification"))
27
28 print(target)
29 print(features)
```

```
> print(target)
[1] "Classification"
> print(features)
[1] "Feature 001"      "Feature 002"      "Feature 003"      "Feature 004"
[5] "Feature 005"      "Feature 006"      "Feature 007"      "Feature 008"
[9] "Feature 009"      "Feature 010"      "Feature 011"      "Feature 012"
[13] "Feature 013"     "Feature 014"     "Feature 015"     "Feature 016"
[17] "Feature 017"     "Feature 018"     "Feature 019"     "Feature 020"
[21] "Feature 021"     "Feature 022"     "Feature 023"     "Feature 024"
[25] "Feature 025"     "Feature 026"     "Feature 027"     "Feature 028"
[29] "Feature 029"     "Feature 030"     "Feature 031"     "Feature 032"
[33] "Feature 033"     "Feature 034"     "Feature 035"     "Feature 036"
[37] "Feature 037"     "Feature 038"     "Feature 039"     "Feature 040"
[41] "Feature 041"     "Feature 042"     "Feature 043"     "Feature 044"
[45] "Feature 045"     "Feature 046"     "Feature 047"     "Feature 048"
[49] "Feature 049"     "Feature 050"     "Feature 051"     "Feature 052"
[53] "Feature 053"     "Feature 054"     "Feature 055"     "Feature 056"
[57] "Feature 057"     "Feature 058"     "Feature 059"     "Feature 060"
[61] "Feature 061"     "Feature 062"     "Feature 063"     "Feature 064"
[65] "Feature 065"     "Feature 066"     "Feature 067"     "Feature 068"
[69] "Feature 069"     "Feature 070"     "Feature 071"     "Feature 072"
[73] "Feature 073"     "Feature 074"     "Feature 075"     "Feature 076"
[77] "Feature 077"     "Feature 078"     "Feature 079"     "Feature 080"
[81] "Feature 081"     "Feature 082"     "Feature 083"     "Feature 084"
[85] "Feature 085"     "Feature 086"     "Feature 087"     "Feature 088"
[89] "Feature 089"     "Feature 090"     "Feature 091"     "Feature 092"
[93] "Feature 093"     "Feature 094"     "Feature 095"     "Feature 096"
[97] "Feature 097"     "Feature 098"     "Feature 099"     "Feature 100"
[101] "Feature 101"    "Feature 102"    "Feature 103"    "Feature 104"
[105] "Feature 105"    "Feature 106"    "Feature 107"    "Feature 108"
```

# step\_03\_basic\_models.R

## Split data with a random seed

```
32 # -----
33 # Split data into training / test
34 #
35
36 # Split
37 # i.e. using 60% of data for training and 40% for test
38 secom_splits <- h2o.splitFrame(data = secom, ratios = 0.6, seed = 1234)
39 secom_train <- secom_splits[[1]]
40 secom_test <- secom_splits[[2]]
41
42 # Check
43 summary(secom_train$Classification) # 882 : 62 ... % of 1 = 0.07029478
44 summary(secom_test$Classification) # 581 : 42 ... % of 1 = 0.07228916
```

```
> summary(secom_train$Classification)
Classification
-1:882
1 : 62
```

```
> summary(secom_test$Classification)
Classification
-1:581
1 : 42
```

Classification 1 samples  $\approx$  7%

# step\_03\_basic\_models.R

## Train H2O models with default values

```
47 # -----
48 # Train H2O models with default value
49 #
50
51 # Turn off progress bar (if you want to ...)
52 # h2o.no_progress()
53
54 # GBM
55 model_gbm <- h2o.gbm(x = features, y = target,
56                         training_frame = secom_train)
57
58 # Random Forest
59 model_drf <- h2o.randomForest(x = features, y = target,
60                               training_frame = secom_train)
61
62 # Deep Neural Network
63 model_dnn <- h2o.deeplearning(x = features, y = target,
64                               training_frame = secom_train)
```

```
> # GBM
> model_gbm <- h2o.gbm(x = features, y = target,
+                         training_frame = secom_train)
|=====| 100%
Warning message:
In .h2o.startModelJob(algo, params, h2oRestApiVersion) :
  Dropping constant columns: [Feature 516, Feature 234, Feature 233, Feature 236, Feature 235, Feature 510, Feature 238, Feature 513, Feature 237, Feature 479, Feature 515, Feature 514, Feature 193, Feature 192, Feature 195, Feature 194, Feature 075, Feature 230, Feature 232, Feature 231, Feature 529, Feature 244, Feature 365, Feature 401, Feature 400, Feature 006, Feature 403, Feature 402, Feature 405, Feature 404, Feature 241, Feature 482, Feature 243, Feature 242, Feature 180, Feature 179, Feature 459, Feature 050, Feature 053, Feature 450, Feature 210, Feature 331, Feature 452, Feature 330, Feature 451, Feature 191, Feature 070, Feature 190, Feature 506, Feature 505, Feature 508, Feature 507, Feature 509, Feature 465, Feature 343, Feature 464, Feature 467, Feature 466, Feature 227, Feature 348, Feature 502, Feature 504, Feature 503, Feature 463, Feature 187, Feature 462, Feature 399, Feature 277, Feature 398, Feature 315, Feature 314, Feature 316, Feature 150, Feature 395, Feature 39 [...] truncated]
```

H2O automatically ignores  
Columns with constant values

# step\_03\_basic\_models.R

## summary(model\_xxx)

```
> print(summary(model_gbm))
Model Details:
=====
H2OBinomialModel: gbm
Model Key: GBM_model_R_1474971910804_1
Model Summary:
  number_of_trees number_of_internal_trees model_size_in_bytes min_depth max_depth mean_depth
1           50                  50             10880          5      5     5.00000
  min_leaves max_leaves mean_leaves
1       6        21    12.38000

H2OBinomialMetrics: gbm
** Reported on training data. **

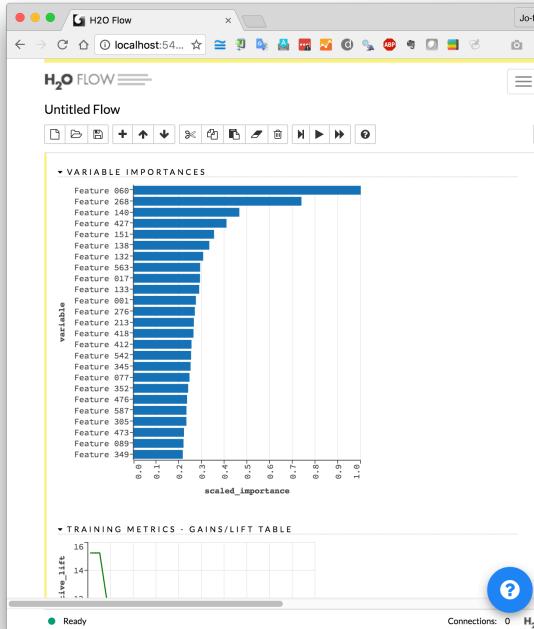
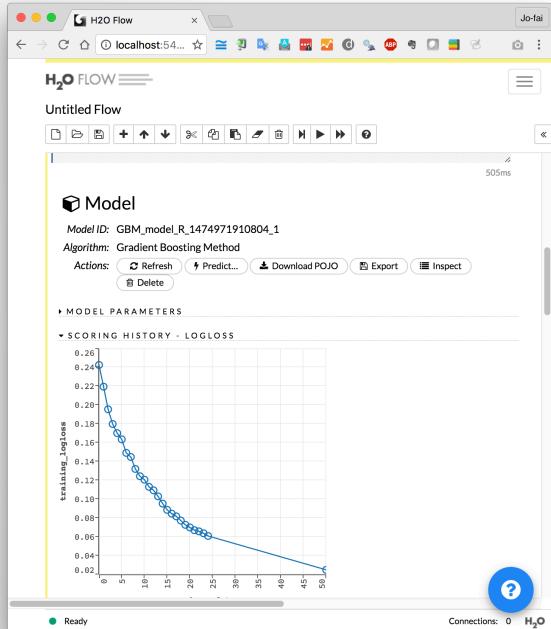
MSE: 0.002598174
RMSE: 0.05097229
LogLoss: 0.0248643
Mean Per-Class Error: 0
AUC: 1
Gini: 1

Confusion Matrix for F1-optimal threshold:
 -1 1 Error Rate
-1 882 0 0.000000 =0/882
1   0 62 0.000000 =0/62
Totals 882 62 0.000000 =0/944

Maximum Metrics: Maximum metrics at their respective thresholds
  metric threshold value idx
1  max f1 0.628627 1.000000 61
2  max f2 0.628627 1.000000 61
3  max f0point5 0.628627 1.000000 61
4  max accuracy 0.628627 1.000000 61
5  max precision 0.956650 1.000000 0
6  max recall 0.628627 1.000000 61
7  max specificity 0.956650 1.000000 0
8  max absolute_mcc 0.628627 1.000000 61
9  max min_per_class_accuracy 0.628627 1.000000 61
10 max mean_per_class_accuracy 0.628627 1.000000 61
```

```
Scoring History:
  timestamp duration number_of_trees training_rmse training_logloss training_auc
1 2016-09-27 11:25:31 0.021 sec      0  0.24772  0.24231  0.50000
2 2016-09-27 11:25:32 0.722 sec      1  0.24010  0.21926  0.83663
3 2016-09-27 11:25:32 1.001 sec      2  0.22980  0.19523  0.92679
4 2016-09-27 11:25:32 1.178 sec      3  0.22041  0.17966  0.94964
5 2016-09-27 11:25:32 1.320 sec      4  0.21468  0.16997  0.96445
  training_lift training_classification_error
1 1.000000            0.93432
2 9.13548              0.08581
3 12.18065             0.04449
4 13.70323             0.03496
5 15.22581             0.03708
  ---
  timestamp duration number_of_trees training_rmse training_logloss training_auc
21 2016-09-27 11:25:34 3.500 sec     20  0.12174  0.06986  0.99996
22 2016-09-27 11:25:34 3.631 sec     21  0.11797  0.06696  1.00000
23 2016-09-27 11:25:35 3.726 sec     22  0.11645  0.06571  1.00000
24 2016-09-27 11:25:35 3.841 sec     23  0.11373  0.06369  1.00000
25 2016-09-27 11:25:35 3.967 sec     24  0.10979  0.06082  1.00000
26 2016-09-27 11:25:37 6.596 sec     50  0.05097  0.02486  1.00000
  training_lift training_classification_error
21 15.22581            0.00106
22 15.22581            0.00000
23 15.22581            0.00000
24 15.22581            0.00000
25 15.22581            0.00000
26 15.22581            0.00000
Variable Importances: (Extract with `h2o.varimp`)
=====
Variable Importances:
  variable relative_importance scaled_importance percentage
1 Feature 060  9.594806  1.000000  0.050880
2 Feature 268  7.088909  0.738847  0.037592
3 Feature 140  4.466067  0.465467  0.023683
4 Feature 427  3.924058  0.408977  0.020809
5 Feature 151  3.397748  0.354124  0.018018
```

# Use H2O Flow (localhost:54321)



# step\_03\_basic\_models.R

## Evaluate models with test data

```
72 # -----
73 # Evaluate model performance on unseen data
74 # -----
75
76 h2o.performance(model_gbm, newdata = secom_test)
77 h2o.performance(model_drf, newdata = secom_test)
78 h2o.performance(model_dnn, newdata = secom_test)
```

```
> h2o.performance(model_gbm, newdata = secom_test)
H2OBinomialMetrics: gbm

MSE:  0.06408139
RMSE: 0.253143
LogLoss: 0.2678549
Mean Per-Class Error: 0.2928858
AUC: 0.7041841
Gini: 0.4083682

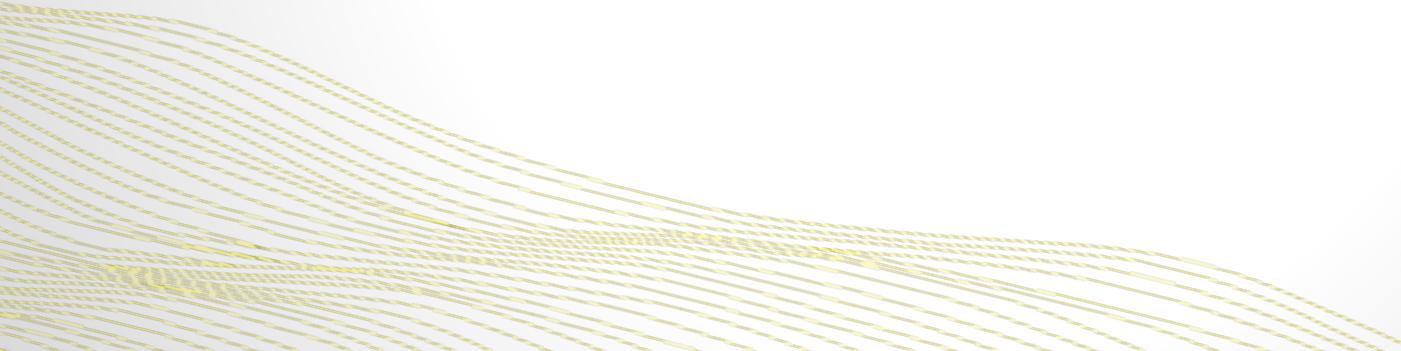
Confusion Matrix for F1-optimal threshold:
      -1    1   Error    Rate
-1    379 202 0.347676  =202/581
1      10  32 0.238095  =10/42
Totals 389 234 0.340289  =212/623

Maximum Metrics: Maximum metrics at their respective thresholds
                metric threshold  value idx
1                  max f1  0.016032 0.231884 201
2                  max f2  0.016032 0.398010 201
3                  max f0point5 0.053221 0.196078 63
4                  max accuracy 0.492634 0.930979  0
5                  max precision 0.244674 0.333333  5
6                  max recall  0.005088 1.000000 378
7                  max specificity 0.492634 0.998279  0
8                  max absolute_mcc 0.016032 0.214472 201
9  max min_per_class_accuracy 0.016481 0.659208 196
10 max mean_per_class_accuracy 0.016032 0.707114 201
```

Workshop #1

Use Case 1: Classification

# Step 4: Tuning Models



# step\_04\_manual\_tuning.R

## ?h2o.gbm ?h2o.deeplearning ...

h2o.gbm {h2o}

R Documentation

### Gradient Boosting Machine

#### Description

Builds gradient boosted classification trees and gradient boosted regression trees on a parsed data set.

#### Usage

```
h2o.gbm(x, y, training_frame, model_id, checkpoint, ignore_const_cols = TRUE,
distribution = c("AUTO", "gaussian", "bernoulli", "multinomial", "poisson",
"gamma", "tweedie", "laplace", "quantile", "huber"), quantile_alpha = 0.5,
tweedie_power = 1.5, huber_alpha = 0.9, ntrees = 50, max_depth = 5,
min_rows = 10, learn_rate = 0.1, learn_rate_annealing = 1,
sample_rate = 1, sample_rate_per_class, col_sample_rate = 1,
col_sample_rate_change_per_level = 1, col_sample_rate_per_tree = 1,
nbins = 20, nbins_top_level = 1024, nbins_cats = 1024,
validation_frame = NULL, balance_classes = FALSE, class_sampling_factors,
max_after_balance_size = 5, seed, build_tree_one_node = FALSE,
nfolds = 0, fold_column = NULL, fold_assignment = c("AUTO", "Random",
"Modulo", "Stratified"), keep_cross_validation_predictions = FALSE,
keep_cross_validation_fold_assignment = FALSE,
score_each_iteration = FALSE, score_tree_interval = 0,
stopping_rounds = 0, stopping_metric = c("AUTO", "deviance", "logloss",
"MSE", "AUC", "misclassification", "mean_per_class_error"),
stopping_tolerance = 0.001, max_runtime_secs = 0, offset_column = NULL,
weights_column = NULL, min_split_improvement = 1e-05,
histogram_type = c("AUTO", "UniformAdaptive", "Random", "QuantilesGlobal",
"RoundRobin"), max_abs_leafnode_pred, pred_noise_bandwidth = 0)
```

#### Arguments

x	A vector containing the names or indices of the predictor variables to use in building the GBM model. If x is missing, then all columns except y are used.
y	The name or index of the response variable. If the data does not contain a header, this is the column index number starting at 0, and increasing from left to right. (The response must be either an integer or a categorical variable).
training_frame	An H2OFrame object containing the variables in the model.
model_id	(Optional) The unique id assigned to the resulting model. If none is given, an id will automatically be generated.
checkpoint	"Model checkpoint" (provide the model_id) to resume training with.
ignore_const_cols	A logical value indicating whether or not to ignore all the constant columns in the training frame.
distribution	A character string. The distribution function of the response. Must be "AUTO", "bernoulli", "multinomial", "poisson", "gamma", "tweedie", "laplace", "quantile", "huber" or "gaussian".
quantile_alpha	Desired quantile for Quantile regression, must be between 0 and 1. Default is 0.5.
tweedie_power	Tweedie power for Tweedie regression, must be between 1 and 2. Default is 1.5.
huber_alpha	Desired quantile for Huber/M-regression (threshold between quadratic and linear loss, must be between 0 and 1). Default is 0.9.
ntrees	A nonnegative integer that determines the number of trees to grow. Default is 50.
max_depth	Maximum depth to grow the tree. Default is 5.
min_rows	Minimum number of rows to assign to terminal nodes. Default is 10.
learn_rate	Learning rate (from 0.0 to 1.0). Default is 0.1.
learn_rate_annealing	Scale the learning rate by this factor after each tree (e.g., 0.99 or 0.999). Default is 1.
sample_rate	Row sample rate per tree (from 0.0 to 1.0). Default is 1.
sample_rate_per_class	Row sample rate per tree per class (one per class, from 0.0 to 1.0).
col_sample_rate	Column sample rate per split (from 0.0 to 1.0). Default is 1.
col_sample_rate_change_per_level	Relative change of the column sampling rate for every level (from 0.0 to 2.0). Default is 1.
col_sample_rate_per_tree	Column sample rate per tree (from 0.0 to 1.0). Default is 1.
nbins	For numerical columns (real/int), build a histogram of (at least) this many bins, then split at the best point. Default is 20.
nbins_top_level	For numerical columns (real/int), build a histogram of (at most) this many bins at the root level, then decrease by factor of two per level. Default is 1024.
nbins_cats	For categorical columns (factors), build a histogram of this many bins, then split at the best point. Higher values can lead to more overfitting. Default is 1024.
validation_frame	An H2OFrame object indicating the validation dataset used to construct the confusion matrix. Defaults to NULL. If left as NULL, this defaults to the training data when nfolds = 0.

# step\_04\_manual\_tuning.R

```
29 # -----
30 # Train H2O models with default / manual settings
31 # -----
32
33 # Check out all parameters
34 # ?h2o.gbm
35 # ?h2o.deeplearning
36 # ?h2o.randomForest
37
38 # GBM with default settings
39 model_gbm1 <- h2o.gbm(x = features, y = target,
40                         training_frame = secom_train)
41
42 # GBM with manual settings
43 model_gbm2 <- h2o.gbm(x = features, y = target,
44                         training_frame = secom_train,
45                         model_id = "gbm_tuning", # define an ID
46                         ntrees = 100,          # default = 50
47                         max_depth = 7,         # default = 5
48                         learn_rate = 0.05)    # default = 0.1
49
50 # Use R / Flow to look at models
51 print(summary(model_gbm1))
52 print(summary(model_gbm2))
```

# step\_04\_manual\_tuning.R

## Default

```
> h2o.performance(model_gbm1, newdata = secom_test)  
H2OBinomialMetrics: gbm
```

MSE: 0.06408139  
RMSE: 0.253143  
LogLoss: 0.2678549  
Mean Per-Class Error: 0.2928858  
AUC: 0.7041841  
Gini: 0.4083682

Confusion Matrix for F1-optimal threshold:  

-1	1	Error	Rate
379	202	0.347676	=202/581
10	32	0.238095	=10/42
Totals	389 234	0.340289	=212/623

Maximum Metrics: Maximum metrics at their respective thresholds  

	metric	threshold	value	idx
1	max f1	0.016032	0.231884	201
2	max f2	0.016032	0.398010	201
3	max f0point5	0.053221	0.196078	63
4	max accuracy	0.492634	0.930979	0
5	max precision	0.244674	0.333333	5
6	max recall	0.005088	1.000000	378
7	max specificity	0.492634	0.998279	0
8	max absolute_mcc	0.016032	0.214472	201
9	max min_per_class_accuracy	0.016481	0.659208	196
10	max mean_per_class_accuracy	0.016032	0.707114	201

## Manual settings

```
> h2o.performance(model_gbm2, newdata = secom_test)  
H2OBinomialMetrics: gbm
```

MSE: 0.06372674  
RMSE: 0.2524416  
LogLoss: 0.2885224  
Mean Per-Class Error: 0.2729489  
AUC: 0.7367224  
Gini: 0.4734448

Confusion Matrix for F1-optimal threshold:  

-1	1	Error	Rate
416	165	0.283993	=165/581
11	31	0.261905	=11/42
Totals	427 196	0.282504	=176/623

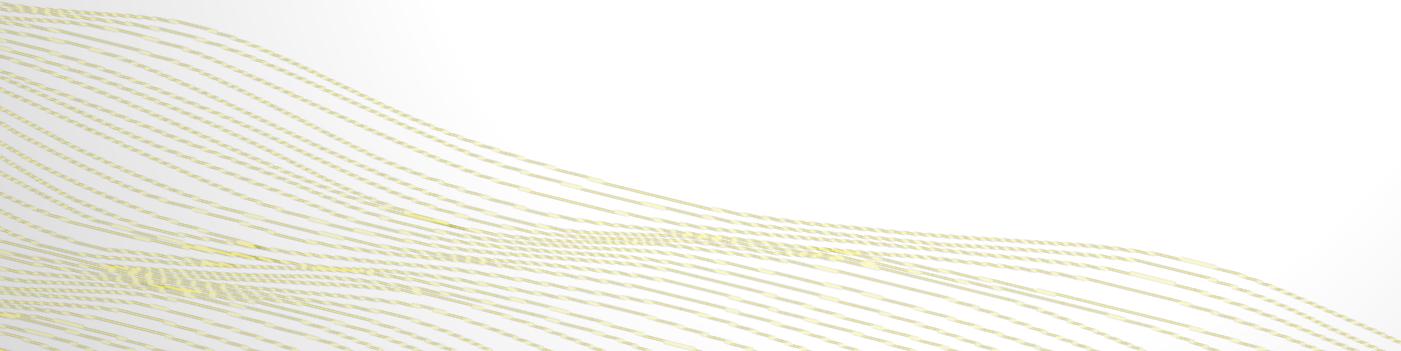
Maximum Metrics: Maximum metrics at their respective thresholds  

	metric	threshold	value	idx
1	max f1	0.010341	0.260504	183
2	max f2	0.009570	0.426667	192
3	max f0point5	0.162720	0.256410	8
4	max accuracy	0.249499	0.932584	3
5	max precision	0.249499	0.500000	3
6	max recall	0.001961	1.000000	388
7	max specificity	0.343655	0.998279	0
8	max absolute_mcc	0.009570	0.245244	192
9	max min_per_class_accuracy	0.010341	0.716007	183
10	max mean_per_class_accuracy	0.009570	0.730350	192

Better performance with manual settings

# Workshop #2

## Use Case 1: Classification



Workshop #2

Use Case 1: Classification

# Step 6: Tuning Models with Early Stopping



Workshop #2

Use Case 1: Classification

# Step 7: Tuning Models with Grid Search

Workshop #2

Use Case 1: Classification

# Step 8: Stacking Models with h2oEnsemble

Workshop #2

Use Case 1: Classification

# Step 9: Saving / Loading Models

# Workshop #2

## Use Case 2: Anomaly Detection

# Workshop #2

# Other H2O Features

