MODEL	1ST	2ND
BST-DT	0.580	0.228
RF	0.390	0.525
BAG-DT	0.030	0.232
SVM	0.000	0.008
ANN	0.000	0.007
KNN	0.000	0.000
BST-STMP	0.000	0.000
DT	0.000	0.000
LOGREG	0.000	0.000
NB	0.000	0.000

AVG	1ST	2ND
RF	0.727	0.207
ANN	0.053	0.172
BSTDT	0.059	0.228
SVM	0.043	0.195
LR	0.089	0.132
BAGDT	0.002	0.012
KNN	0.023	0.045
BSTST	0.004	0.009
PRC	0	0
NB	0	0

An Empirical Comparison of Supervised Learning Algorithms

http://www.cs.cornell.edu/~alexn/papers/empirical.icml06.pdf

An Empirical Evaluation of Supervised Learning in High Dimensions

http://lowrank.net/nikos/pubs/empirical.pdf

MODEL	1ST	2ND
BST-DT	0.580	0.228
RF	0.390	0.525
BAG-DT	0.030	0.232
SVM	0.000	0.008
ANN	0.000	0.007
KNN	0.000	0.000
BST-STMP	0.000	0.000
DT	0.000	0.000
LOGREG	0.000	0.000
NB	0.000	0.000

AVG	1ST	2ND
RF	0.727	0.207
ANN	0.053	0.172
BSTDT	0.059	0.228
SVM	0.043	0.195
LR	0.089	0.132
BAGDT	0.002	0.012
KNN	0.023	0.045
BSTST	0.004	0.009
PRC	0	0
NB	0	0

An Empirical Comparison of Supervised Learning Algorithms

http://www.cs.cornell.edu/~alexn/papers/empirical.icml06.pdf

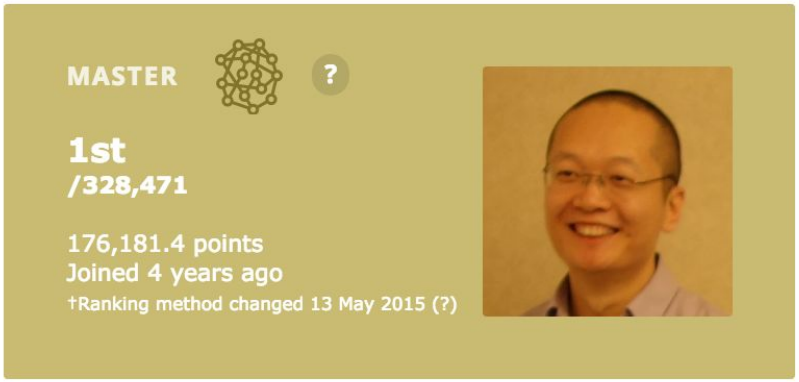
An Empirical Evaluation of Supervised Learning in High Dimensions

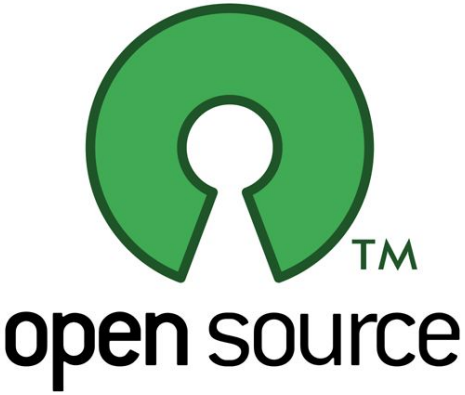
http://lowrank.net/nikos/pubs/empirical.pdf

I usually use other people's code [...] I can find open source code for what I want to do, and my time is much better spent doing research and feature engineering -- Owen Zhang

http://blog.kaggle.com/**2015/06/22**/profiling-top-kagglers-owen-zhang-currently-1-in-the-world/

kaggle




TM
open source


100% FREE

open source

100% FREE

Meetup

stackoverflow

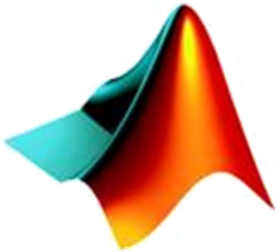
open source

100% FREE

Meetup

stackoverflow

Me in 2006



MATLAB

- cost was not an issue!
- data.frame
- 800 packages

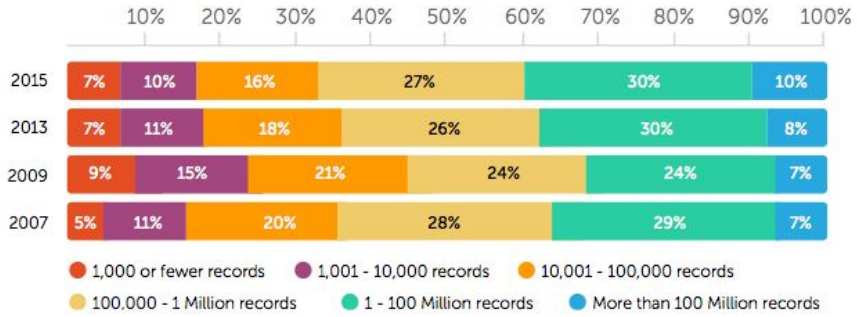
open source

- R packages
- Python scikit-learn
- Vowpal Wabbit
- H2O
- xgboost
- Spark MLlib

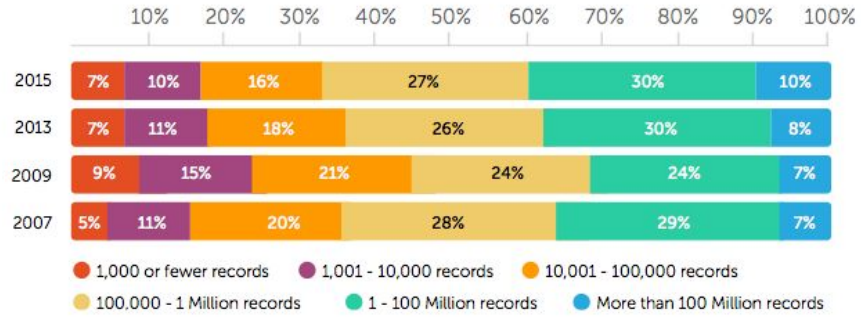
open source

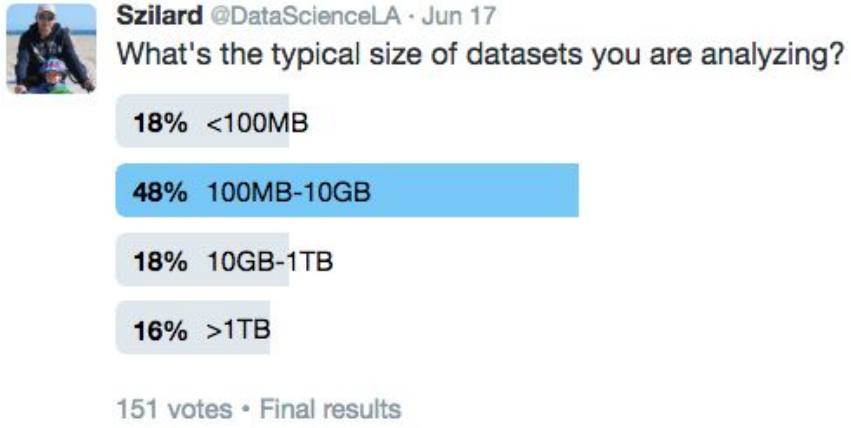
- R packages 30%
- Python scikit-learn 40%
- Vowpal Wabbit 8%
- H2O 10%
- xgboost 8%
- Spark MLlib 6%

TYPICAL SIZE OF DATASETS

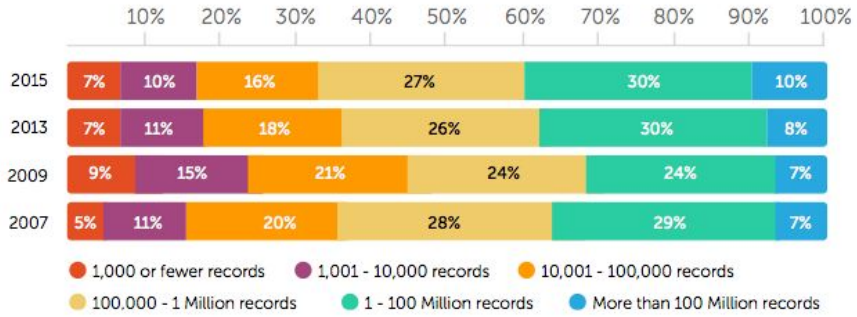


TYPICAL SIZE OF DATASETS

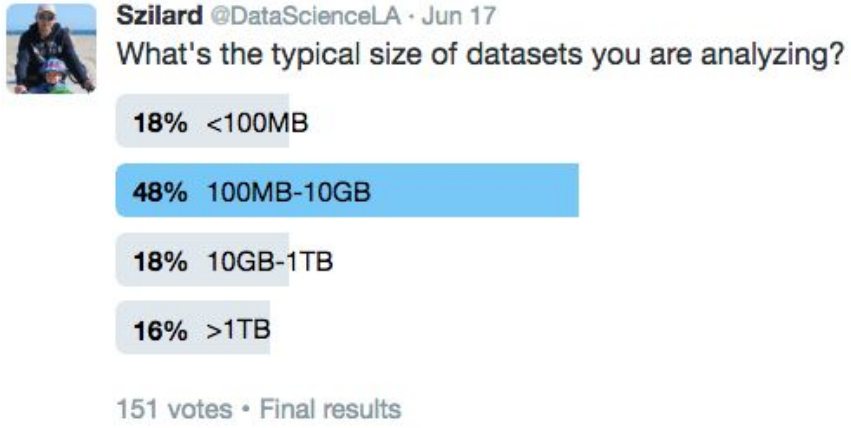




TYPICAL SIZE OF DATASETS



17 Rexer Analytics



Hadley Wickham
@hadleywickham

Following

"It takes a big man to admit his data is small" — @jcheng

Kingston Technology Value RAM 128GB Kit (4x32GB) 2133MHz DDR4 ECC Reg CL15 (KVR21R15D4K4/128)

by Kingston Technology

Be the first to review this item

Was: ~~$743.99~~

Price: $743.96 & **FREE Shipping**. Details

Kingston Technology Value RAM 128GB Kit (4x32GB) 2133MHz DDR4 ECC Reg CL15 (KVR21R15D4K4/128)

by Kingston Technology

Be the first to review this item

Was: ~~$743.99~~

Price: $743.96 & **FREE Shipping**. Details

amazon web services

Model	vCPU	Mem (GiB)
r3.8xlarge	32	244
x1.32xlarge	128	1,952

Gary Bernhardt
@garybernhardt

Consulting service: you bring your big data problems to me, I say "your data set fits in RAM", you pay me $10,000 for saving you $500,000.

RETWEETS	LIKES
997	960

3:03 PM - 19 May 2015

GitHub This repository Search

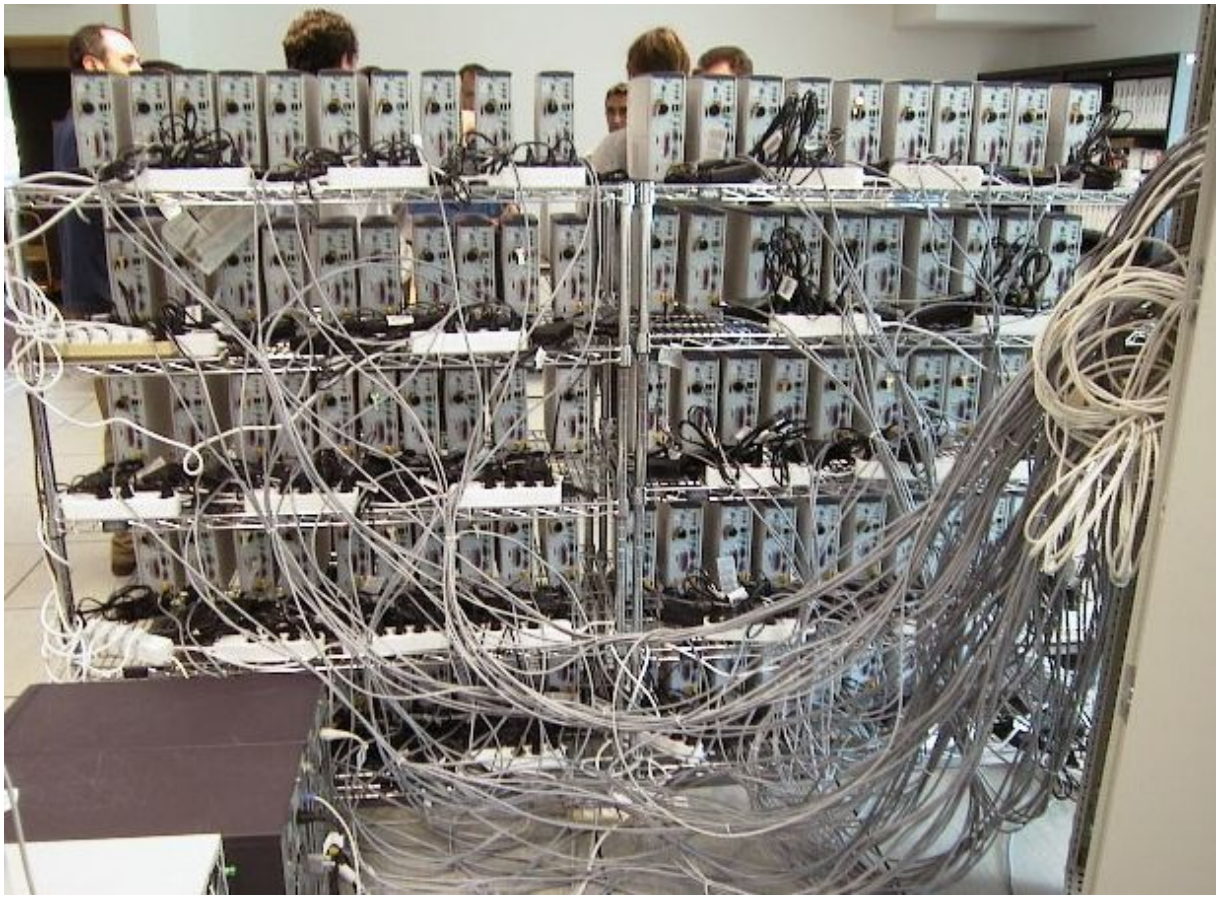
szilard / **benchm-ml**

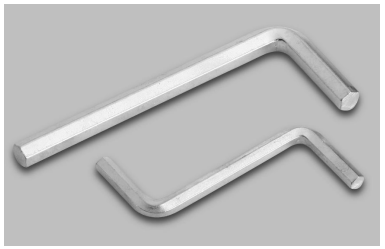
binary classification, 10M records

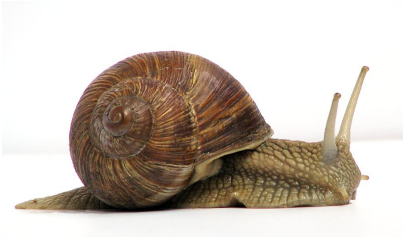
numeric & categorical features, non-sparse


Destination
Gate
PARIS A43 12:00 ON TIME
FRANKFURT A15 12:10 ON TIME
NEW YORK B08 12:25 ON TIME
BRUSSELS A21 12:30 ON TIME
ROME A30 12:30 ON TIME
BOSTON B01 12:35 ON TIME
LONDON A19 12:40 ON TIME
RIO DE JANEIRO B13 12:45 ON TIME
MADRID A26 12:45 ON TIME
ATHENS A37 12:50 ON TIME
STOCKHOLM A40 13:00 ON TIME

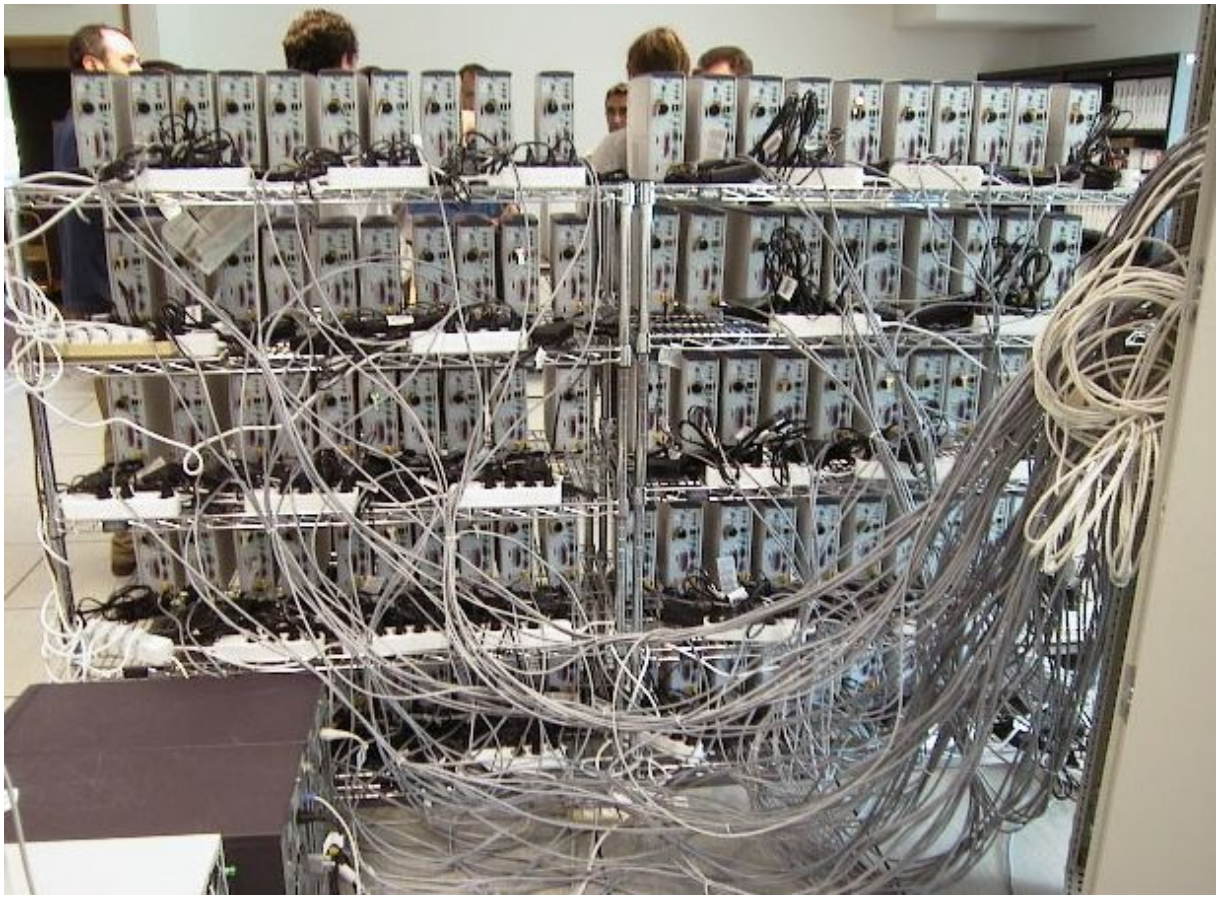
EC2

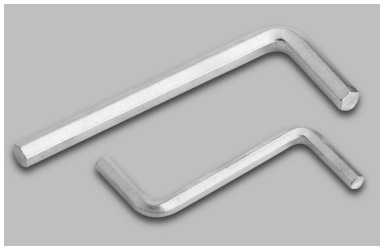


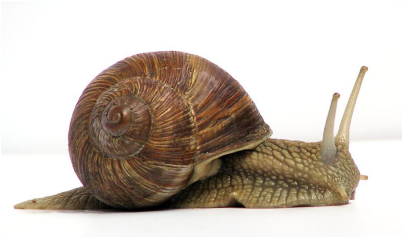




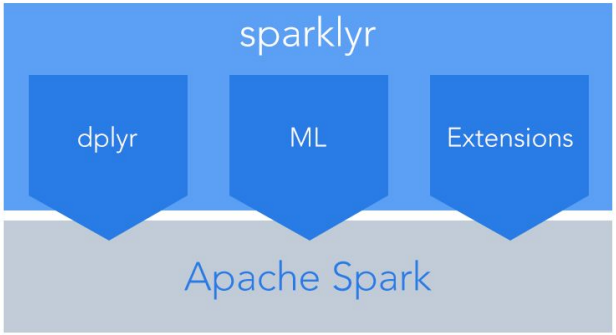

SYSTEM FAILURE





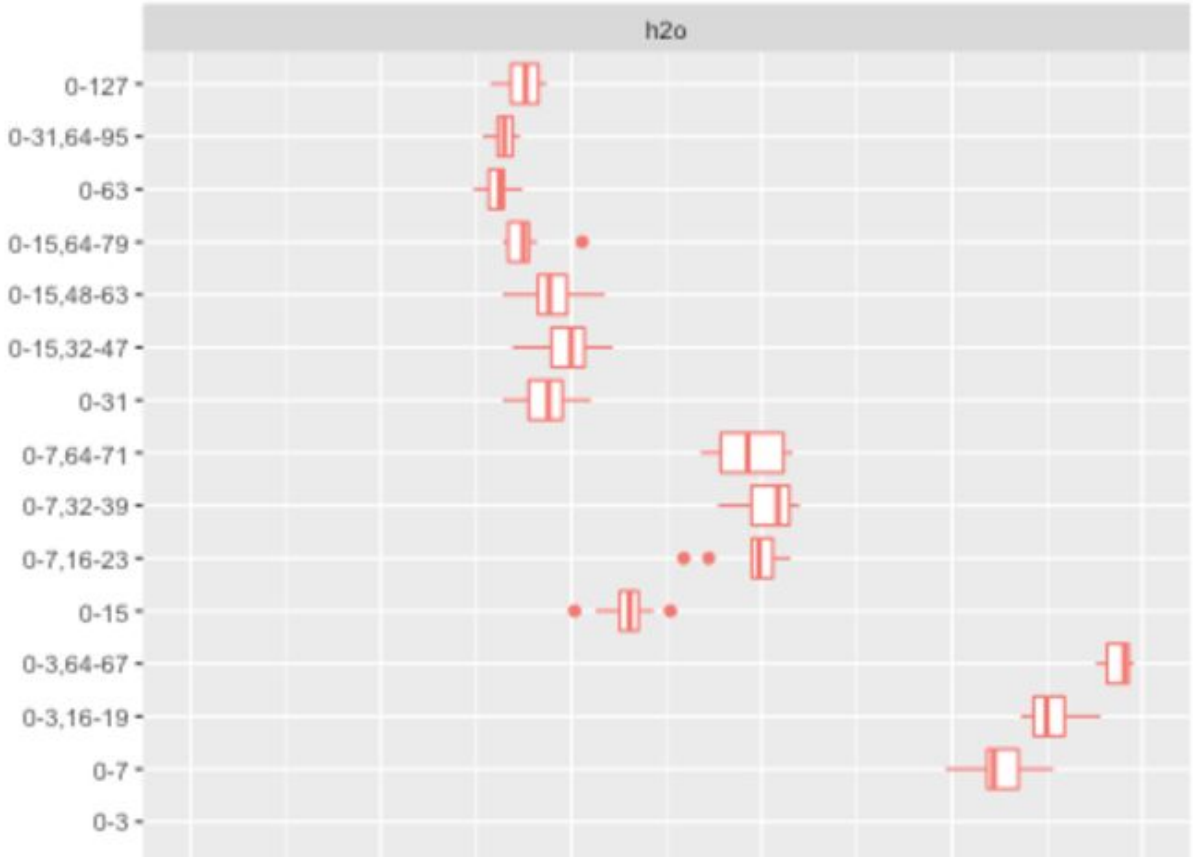



SYSTEM FAILURE


sparklyr
dplyr
ML
Extensions
Apache Spark

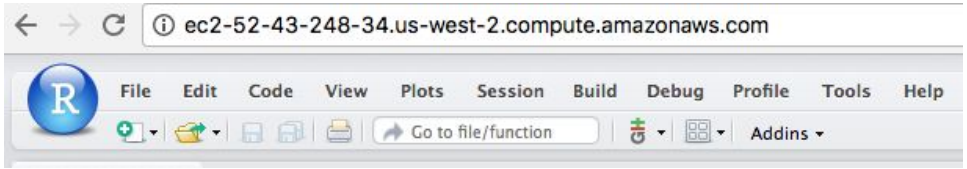
x1.32xlarge (128 cores)

x1.32xlarge has 4 CPU sockets with 16+16 hyperthreaded cores each. Cores 0-15 are on CPU1, then 64-79 are hyperthread pairs of 0-15 etc.



For Class:

	vCPU	ECU	Memory (GiB)	Instance Storage (GB)	Linux/UNIX Usage
General Purpose - Current Generation					
m4.large	2	6.5	8	EBS Only	$0.108 per Hour
m4.xlarge	4	13	16	EBS Only	$0.215 per Hour
m4.2xlarge	8	26	32	EBS Only	$0.431 per Hour
m4.4xlarge	16	53.5	64	EBS Only	$0.862 per Hour
m4.10xlarge	40	124.5	160	EBS Only	$2.155 per Hour
m4.16xlarge	64	188	256	EBS Only	$3.447 per Hour




```
"Month","DayofMonth","DayOfWeek","DepTime","UniqueCarrier","Origin","Dest","Distance","dep_delayed_15min"
"c-7","c-4","c-1",1835,"AS","SFO","SEA",679,"N"
"c-7","c-21","c-5",2341,"DL","SEA","CVG",1964,"N"
"c-10","c-9","c-7",1027,"NW","GTF","FCA",146,"N"
"c-10","c-20","c-5",1823,"XE","DCA","CLE",310,"N"
"c-11","c-15","c-3",2059,"FL","ATL","LGA",761,"Y"
```

rows	csv size
100K	<10MB
1M	<100MB
10M	<1GB

amazon web services

DOMINO

Compute

EC2

EC2 Container Service

Lightsail

Elastic Beanstalk

Lambda

Batch

Storage

S3

EFS

Glacier

Storage Gateway

Database

RDS

DynamoDB

ElastiCache

Redshift

Analytics

Athena

EMR

CloudSearch

Elasticsearch Service

Kinesis

Data Pipeline

QuickSight

Artificial Intelligence

Lex

Polly

Rekognition

~~Machine Learning~~

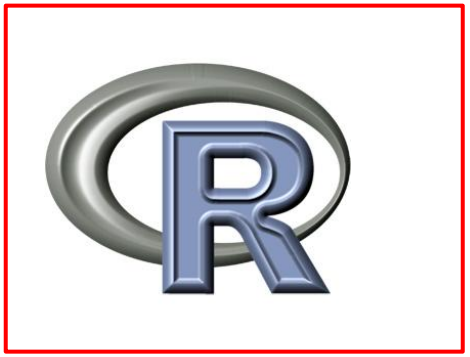
PyData

scikit learn

H2O.ai

dmlc XGBoost

Spark



PyData

scikit learn

H2O.ai

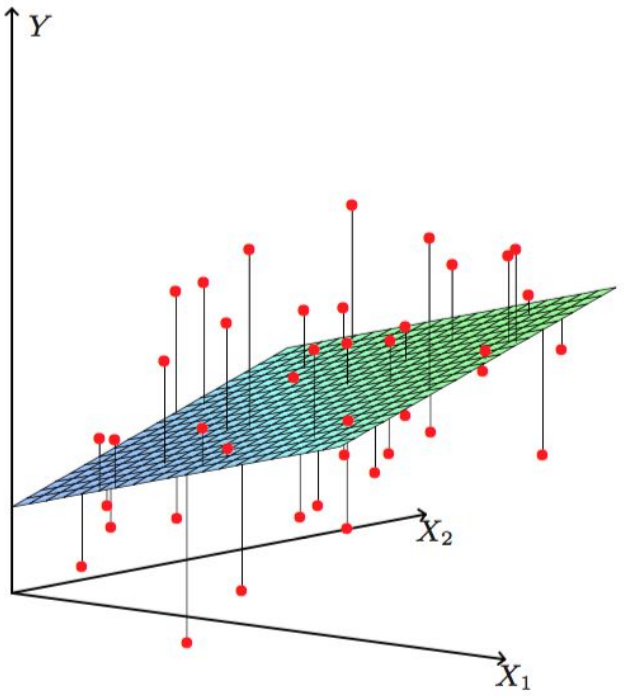
dmlc XGBoost

Spark

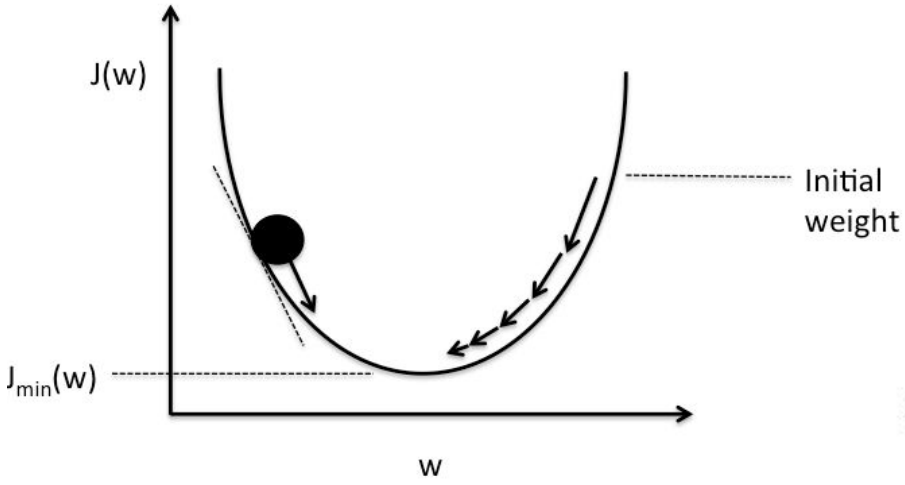
Linear Regression
Logistic Regression

$$f(X) = \beta_0 + \sum_{j=1}^{p} X_j \beta_j.$$

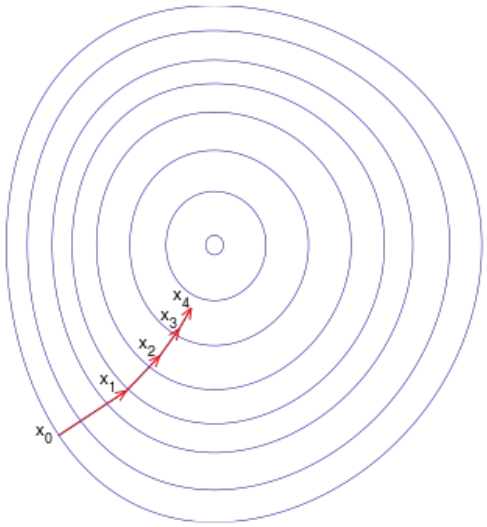
$$\begin{aligned} \mathrm{RSS}(\beta) &= \sum_{i=1}^{N} (y_i - f(x_i))^2 \\ &= \sum_{i=1}^{N} \Big(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij} \beta_j \Big)^2 \end{aligned}$$

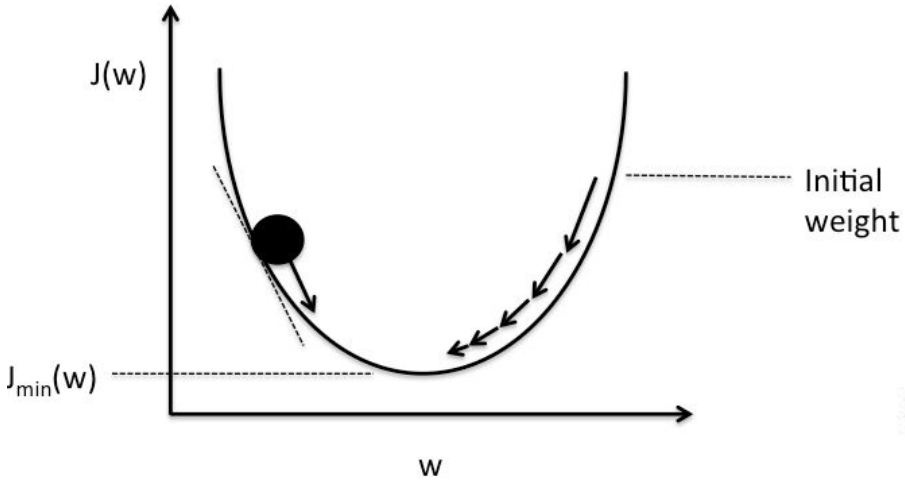


$$\hat{\beta} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}.$$

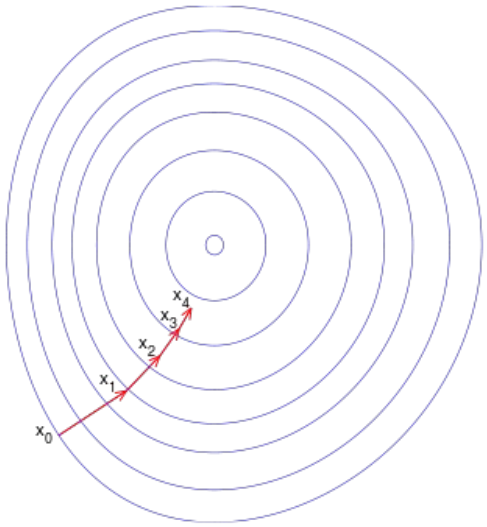


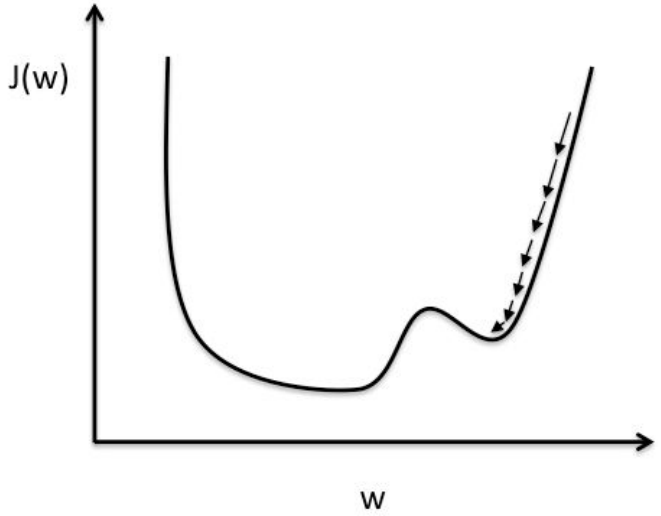
$$\mathbf{a^{n+1}} = \mathbf{a^n} - \gamma \nabla F(\mathbf{a^n})$$





$$\mathbf{a^{n+1}} = \mathbf{a^n} - \gamma \nabla F(\mathbf{a^n})$$





$$\hat{\beta}^{\text{ridge}} = \underset{\beta}{\text{argmin}} \left\{ \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij} \beta_j \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right\}$$

$$\hat{\beta}^{\text{lasso}} = \underset{\beta}{\text{argmin}} \left\{ \frac{1}{2} \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij} \beta_j \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\}$$

elastic net:

$$\lambda \sum_{j=1}^{p} \left(\alpha \beta_j^2 + (1 - \alpha) |\beta_j| \right)$$

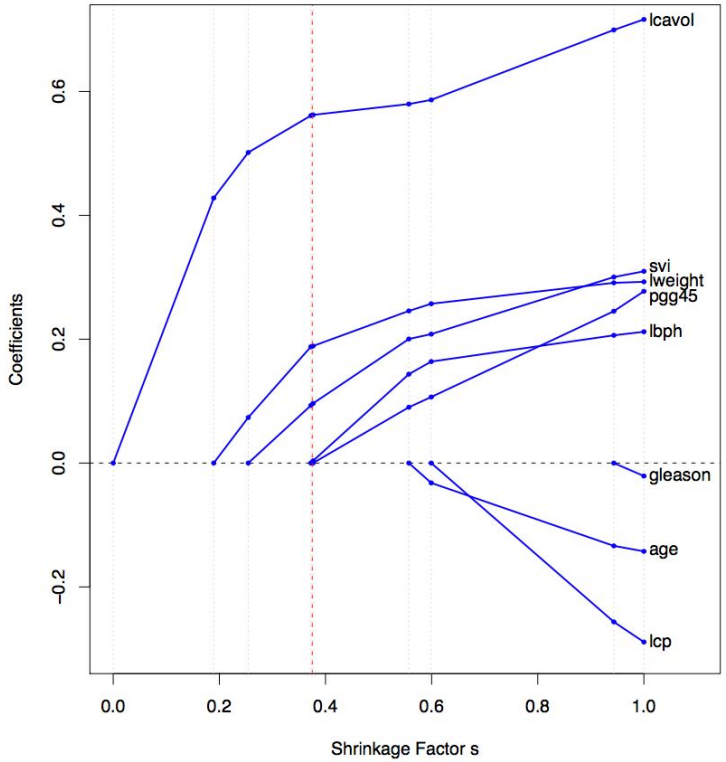


FIGURE 3.10. *Profiles of lasso coefficients, as the tuning parameter t is varied. Coefficients are plotted versus $s = t / \sum_1^p |\hat{\beta}_j|$. A vertical line is drawn at $s = 0.36$,*

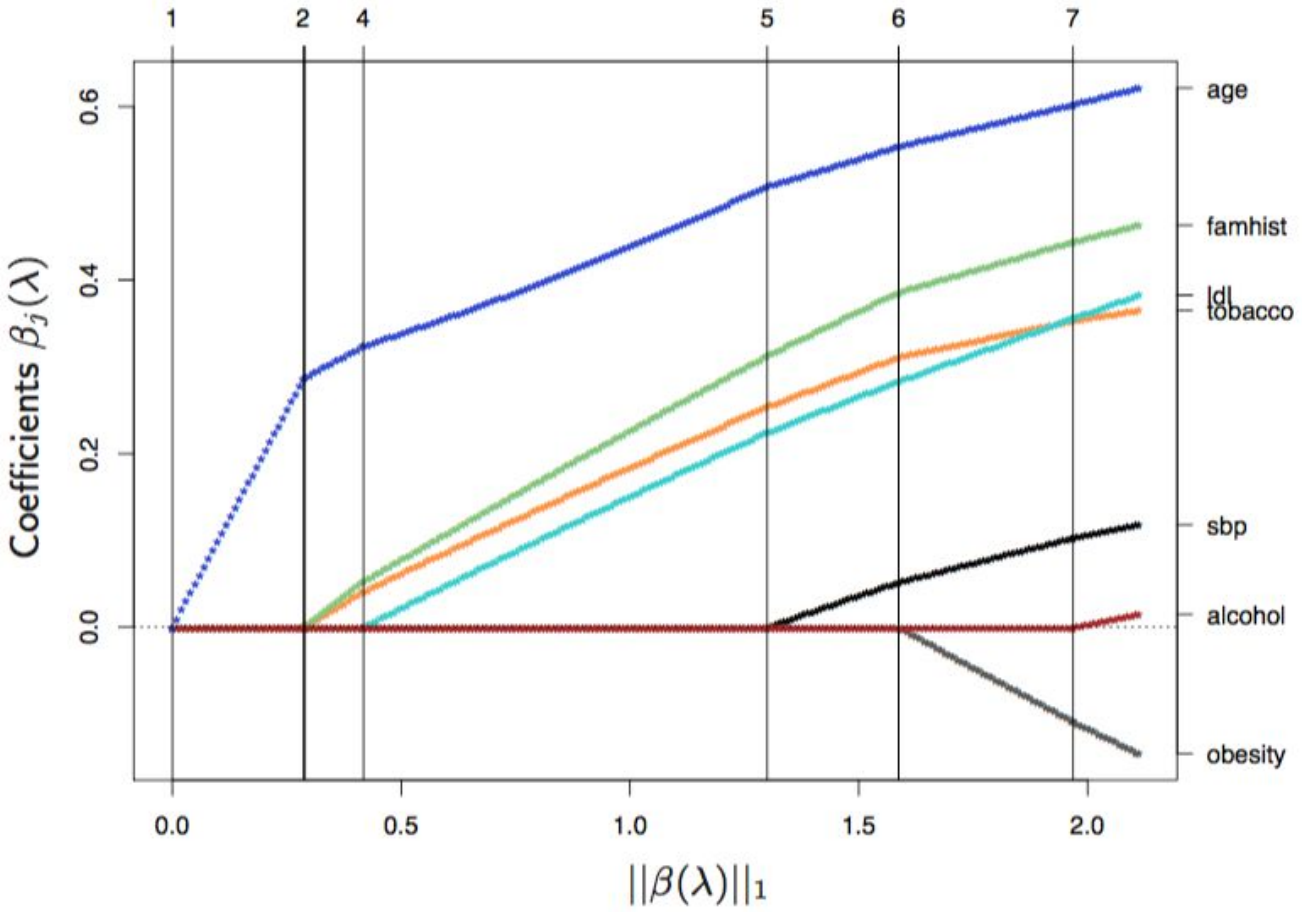
$$\begin{aligned}
\Pr(G=k|X=x) &= \frac{\exp(\beta_{k0}+\beta_k^T x)}{1+\sum_{\ell=1}^{K-1}\exp(\beta_{\ell 0}+\beta_\ell^T x)}, \; k=1,\ldots,K-1, \\
\Pr(G=K|X=x) &= \frac{1}{1+\sum_{\ell=1}^{K-1}\exp(\beta_{\ell 0}+\beta_\ell^T x)}, \qquad (4.18)
\end{aligned}$$

$$\begin{aligned}
\ell(\beta) &= \sum_{i=1}^{N}\Big\{y_i \log p(x_i;\beta)+(1-y_i)\log(1-p(x_i;\beta))\Big\} \\
&= \sum_{i=1}^{N}\Big\{y_i\beta^T x_i - \log(1+e^{\beta^T x_i})\Big\}.
\end{aligned}$$

Starting with β^{old}, a single Newton update is

$$\beta^{\text{new}} = \beta^{\text{old}} - \left(\frac{\partial^2\ell(\beta)}{\partial\beta\partial\beta^T}\right)^{-1}\frac{\partial\ell(\beta)}{\partial\beta}$$

$$\max_{\beta_0,\beta}\left\{\sum_{i=1}^{N}\left[y_i(\beta_0+\beta^T x_i)-\log(1+e^{\beta_0+\beta^T x_i})\right]-\lambda\sum_{j=1}^{p}|\beta_j|\right\}$$



```r
glmnet( X_train, d_train$dep_delayed_15min, family = "binomial", lambda = 0)
```

```
md = LogisticRegression(tol=0.00001, C=1000)
%time md.fit(X_train, y_train)
```


VOWPAL WABBIT

```
vw -h

Update options:
  -l [ --learning_rate ] arg Set learning rate
  --power_t arg              t power value
  --decay_learning_rate arg  Set Decay factor for learning_rate between passes
  --initial_t arg            initial t value

Feature options:
  -b [ --bit_precision ] arg      number of bits in the feature table
  --ngram arg                     Generate N grams. To generate N grams for a
                                  single namespace 'foo', arg should be fN.
  --skips arg                     Generate skips in N grams. This in
                                  conjunction with the ngram tag can be used to
                                  generate generalized n-skip-k-gram. To
                                  generate n-skips for a single namespace
                                  'foo', arg should be fN.
  -q [ --quadratic ] arg          Create and use quadratic features
  --q: arg                        : corresponds to a wildcard for all printable
                                  characters
  --cubic arg                     Create and use cubic features

Example options:
  -t [ --testonly ]              Ignore label information and just test
  --passes arg                   Number of Training Passes
  --loss_function arg (=squared) Specify the loss function to be used, uses
                                 squared by default. Currently available ones
                                 are squared, classic, hinge, logistic and
                                 quantile.
  --l1 arg                       l_1 lambda
  --l2 arg                       l_2 lambda

Gradient Descent options:
  --sgd                 use regular stochastic gradient descent update.
  --adaptive            use adaptive, individual learning rates.
  --invariant           use safe/importance aware updates.
  --normalized          use per feature normalized updates
  --sparse_l2 arg (=0)  use per feature normalized updates

Input options:
  -d [ --data ] arg     Example Set
  -c [ --cache ]        Use a cache.  The default is <data>.cache
```

H2O.ai

- high-performance implementation of best algos (RF, GBM, NN etc.)
- R, Python etc. interfaces, easy to use API
- open source
- advisors: Hastie, Tibshirani
- Java, but C-style memalloc, by Java gurus
- *distributed, “big data”*
- many knobs/tuning, model evaluation, cross validation, model selection (hyperparameter search)
- ensembles
- model deployment (POJO/MOJO export), fast scoring

H2O FLOW Flow Cell Data Model Score Admin Help

CS

```
getFrameSummary "train_1m.hex_sid_9329_1"
```

train_1m.hex_sid_9329_1

Actions: View Data | Split... | Build Model... | Predict | Download | Export

Rows	Columns	Compressed Size
1000000	9	12MB

▾ COLUMN SUMMARIES

label	type	Missing	Zeros	+Inf	-Inf	min	max	mean	sigma	cardinality	Actions
Month	enum	0	81458	0	0	0	11.0	·	·	12	Convert to numeric
DayofMonth	enum	0	32539	0	0	0	30.0	·	·	31	Convert to numeric
DayOfWeek	enum	0	146871	0	0	0	6.0	·	·	7	Convert to numeric
DepTime	int	0	0	0	0	1.0	2615.0	1343.1189	476.6631	·	Convert to enum
UniqueCarrier	enum	0	92424	0	0	0	21.0	·	·	22	Convert to numeric
Origin	enum	0	651	0	0	0	290.0	·	·	291	Convert to numeric
Dest	enum	0	617	0	0	0	291.0	·	·	292	Convert to numeric
Distance	int	0	0	0	0	21.0	4962.0	728.8050	574.4751	·	Convert to enum
dep_delayed_15min	enum	0	807018	0	0	0	1.0	0.1930	0.3946	2	Convert to numeric

← Previous 20 Columns | Next 20 Columns

▸ CHUNK COMPRESSION SUMMARY

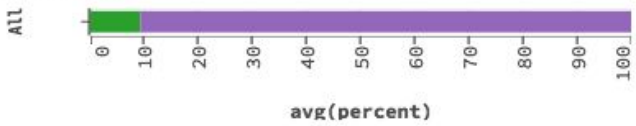
▸ FRAME DISTRIBUTION SUMMARY

```
getColumnSummary "train_1m.hex_sid_9329_1", "UniqueCarrier"
```

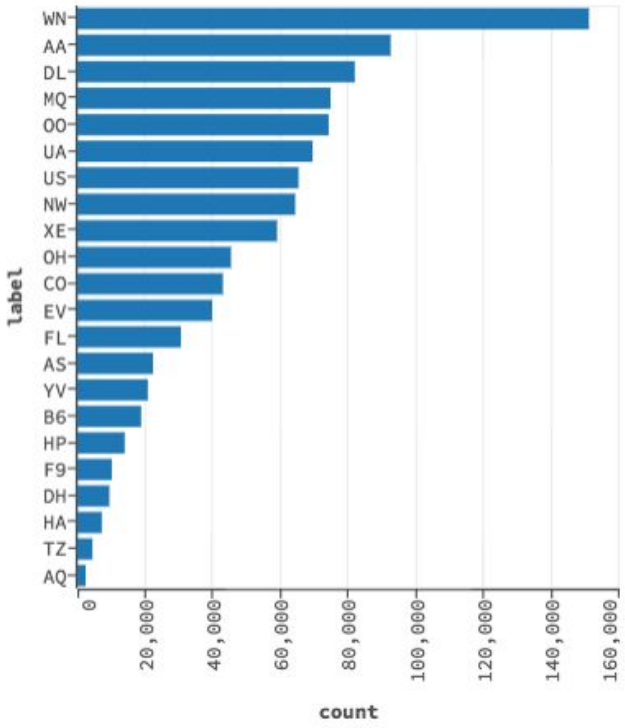
Summary: UniqueCarrier

Actions Impute Inspect

CHARACTERISTICS



DOMAIN (MAX 1000 LEVELS)



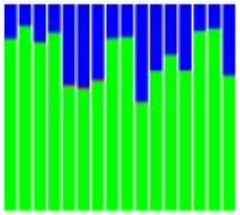
CLOUD STATUS

HEALTHY CONSENSUS LOCKED

Version	Started	Nodes (Used / All)
3.8.2.8	an hour ago	1 / 1

NODES

	Name	Ping	Cores	Load	My CPU %	Sys CPU %	GFLOPS
✓	127.0.0.1:54321	a few seconds ago	16	11.790	71	72	8.709
✓	TOTAL	-	16	11.790	-	-	8.709



Legend

Each bar represents one CPU.

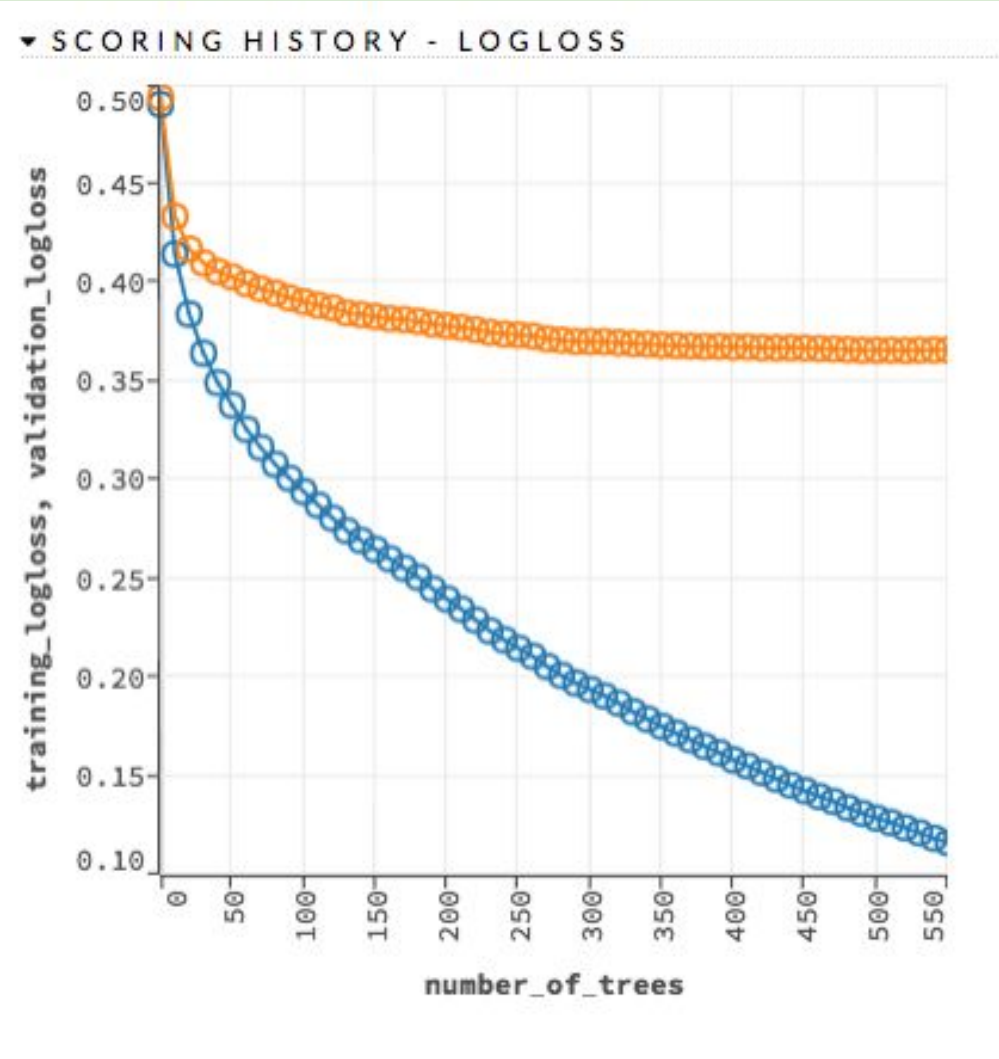
Blue: idle time
Green: user time
Red: system time
White: other time (e.g. i/o)

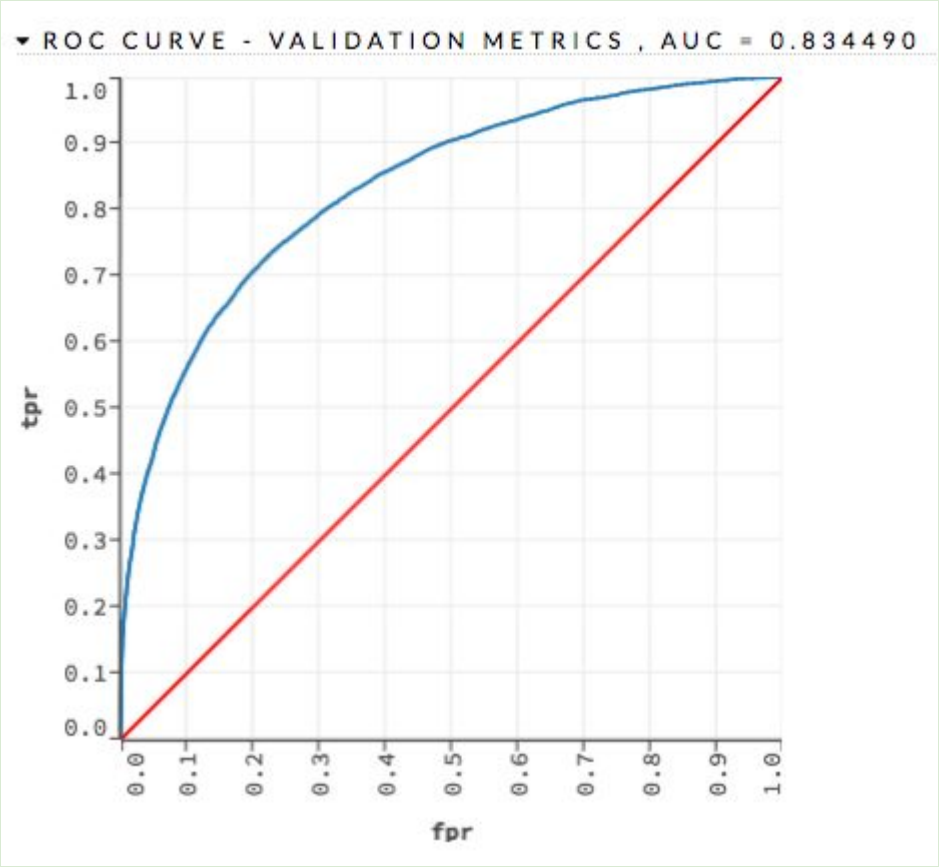
Job

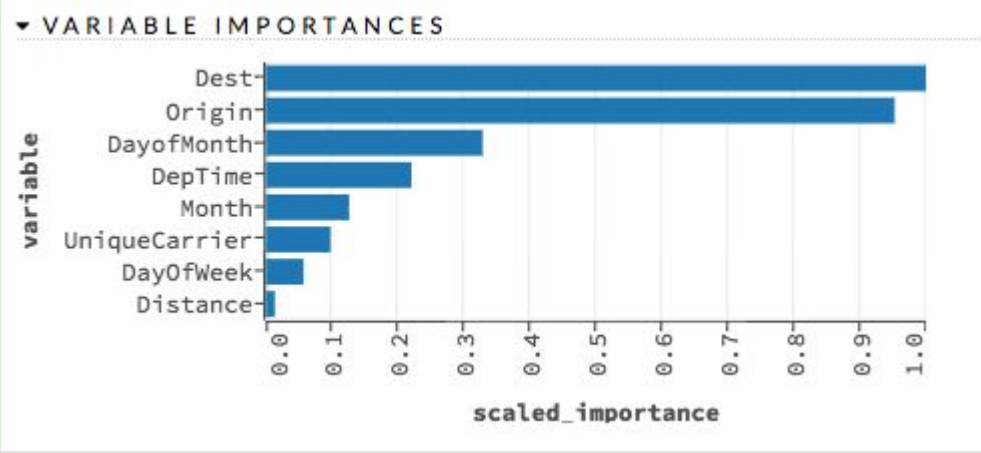
Run Time 00:14:35.11
Remaining Time 07:53:09.571
Type Model
Key GBM_model_R_1466713759449_16
Description GBM
Status RUNNING
Progress 3%

Built 299 trees so far (out of 10000).

Actions View Cancel Job


SCORING HISTORY - LOGLOSS
training_logloss, validation_logloss
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0
50
100
150
200
250
300
350
400
450
500
550
number_of_trees


ROC CURVE - VALIDATION METRICS , AUC = 0.834490
tpr
fpr
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0


VARIABLE IMPORTANCES
variable
Dest
Origin
DayofMonth
DepTime
Month
UniqueCarrier
DayOfWeek
Distance
scaled_importance
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0

```
h2o.glm(x, y, training_frame, model_id, validation_frame = NULL,
  ignore_const_cols = TRUE, max_iterations = 50, beta_epsilon = 0,
  solver = c("IRLSM", "L_BFGS"), standardize = TRUE,
  family = c("gaussian", "binomial", "poisson", "gamma", "tweedie",
  "multinomial"), link = c("family_default", "identity", "logit", "log",
  "inverse", "tweedie"), tweedie_variance_power = 0, tweedie_link_power = 1,
  alpha = 0.5, prior = NULL, lambda = 1e-05, lambda_search = FALSE,
  early_stopping = FALSE, nlambdas = -1, lambda_min_ratio = -1,
  nfolds = 0, seed = NULL, fold_column = NULL,
  fold_assignment = c("AUTO", "Random", "Modulo", "Stratified"),
  keep_cross_validation_predictions = FALSE,
  keep_cross_validation_fold_assignment = FALSE, beta_constraints = NULL,
  offset_column = NULL, weights_column = NULL, intercept = TRUE,
  max_active_predictors = -1, interactions = NULL, objective_epsilon = -1,
  gradient_epsilon = -1, non_negative = FALSE, compute_p_values = FALSE,
  remove_collinear_columns = FALSE, max_runtime_secs = 0,
  missing_values_handling = c("MeanImputation", "Skip"))
```


Spark

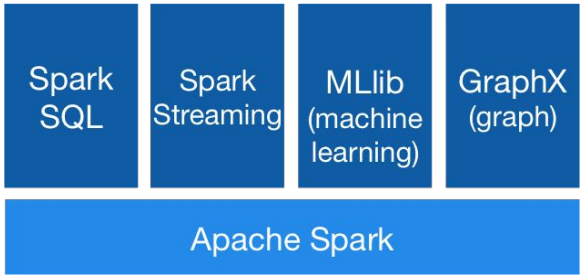
MLlib

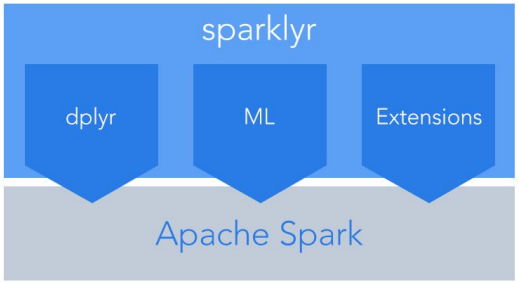

hadoop

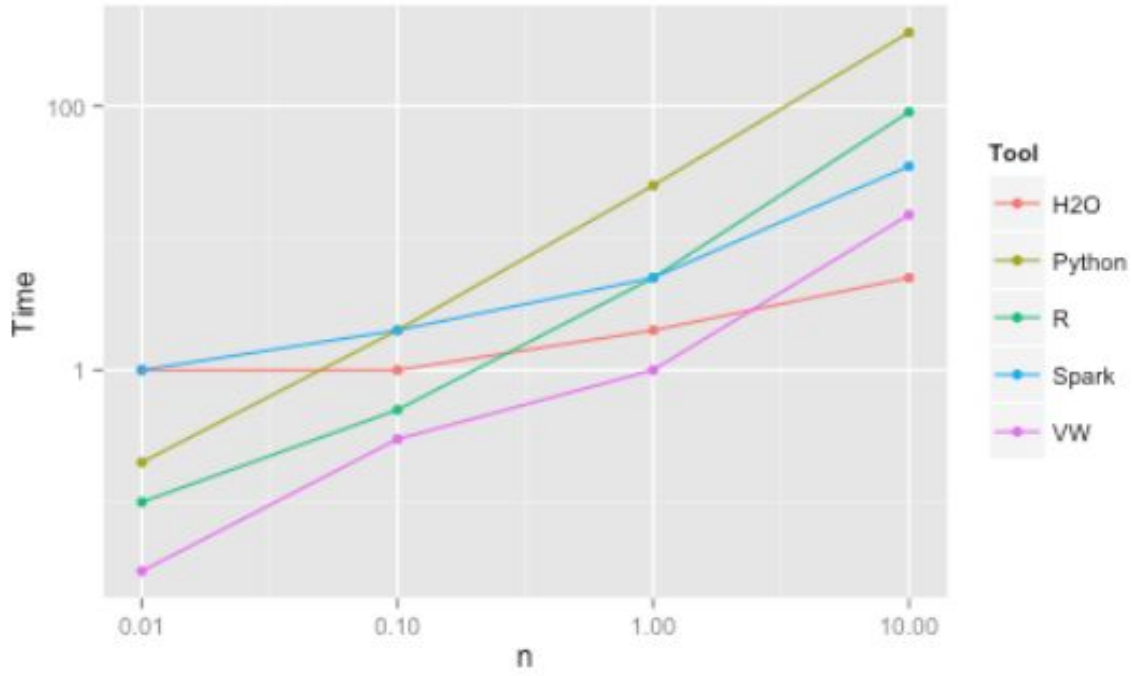

HIVE


Spark


VERTICA

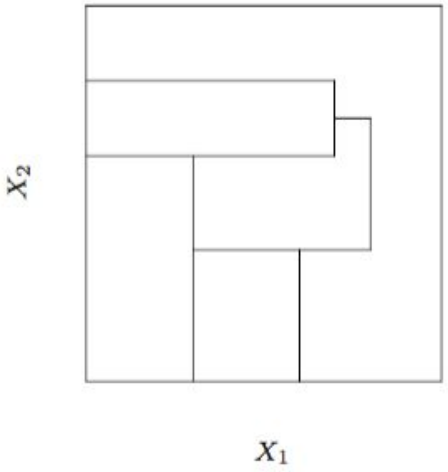

Spark SQL
Spark Streaming
MLlib (machine learning)
GraphX (graph)
Apache Spark

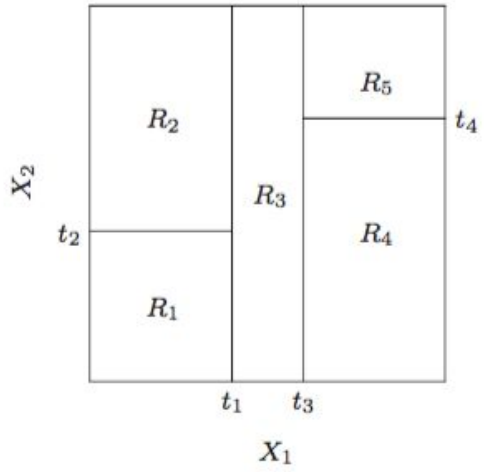

sparklyr
dplyr
ML
Extensions
Apache Spark

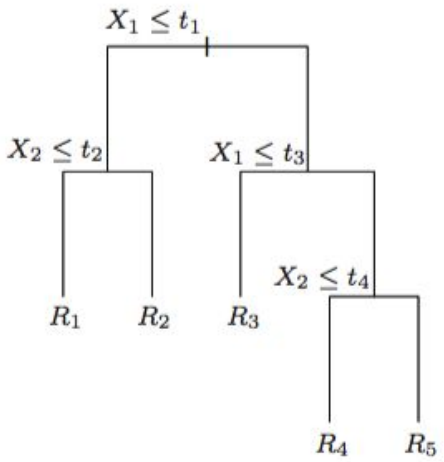


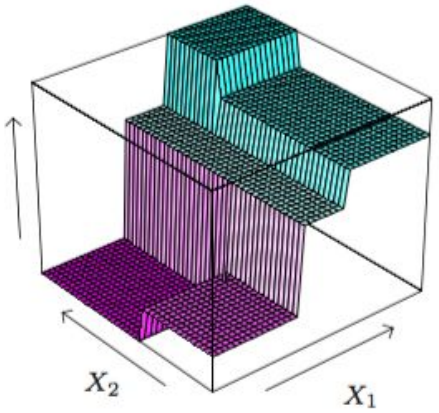
RAM usage see bench-ml github

Decision Trees


X_2
X_1


R_1
R_2
R_3
R_4
R_5
t_1
t_2
t_3
t_4
X_1
X_2


$X_1 \leq t_1$
$X_2 \leq t_2$
$X_1 \leq t_3$
$X_2 \leq t_4$
R_1
R_2
R_3
R_4
R_5


X_2
X_1

$$R_1(j,s) = \{X | X_j \leq s\} \text{ and } R_2(j,s) = \{X | X_j > s\}. \tag{9.12}$$

Then we seek the splitting variable j and split point s that solve

$$\min_{j,\, s} \left[\min_{c_1} \sum_{x_i \in R_1(j,s)} (y_i - c_1)^2 + \min_{c_2} \sum_{x_i \in R_2(j,s)} (y_i - c_2)^2 \right]. \tag{9.13}$$

For any choice j and s, the inner minimization is solved by

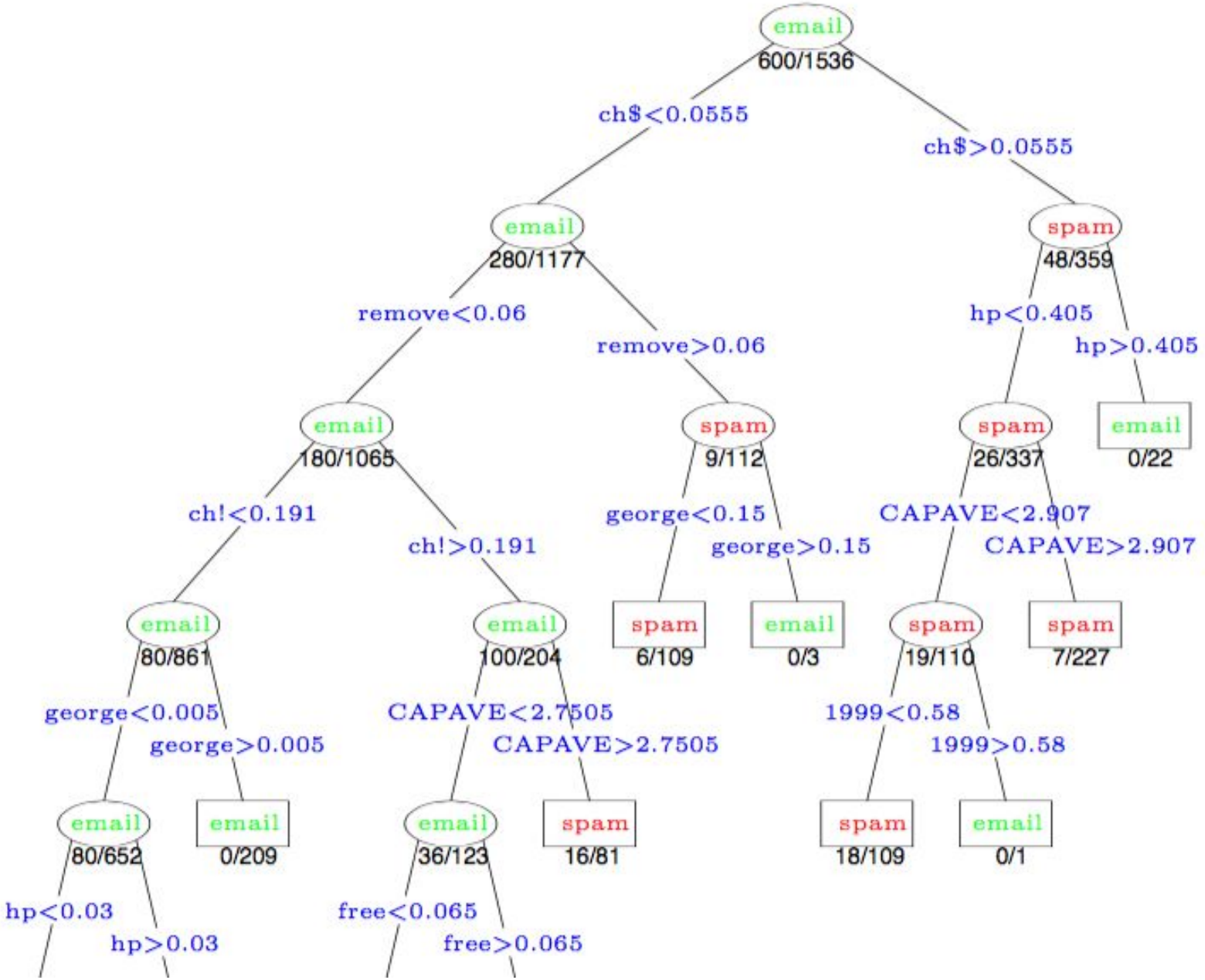
$$\hat{c}_1 = \text{ave}(y_i | x_i \in R_1(j,s)) \text{ and } \hat{c}_2 = \text{ave}(y_i | x_i \in R_2(j,s)). \tag{9.14}$$

For each splitting variable, the determination of the split point s can be done very quickly and hence by scanning through all of the inputs, determination of the best pair (j, s) is feasible.

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{x_i \in R_m} I(y_i = k),$$

the proportion of class k observations in node m. We classify the observations in node m to class $k(m) = \arg\max_k \hat{p}_{mk}$, the majority class in node m. Different measures $Q_m(T)$ of node impurity include the following:

Misclassification error:	$\frac{1}{N_m} \sum_{i \in R_m} I(y_i \neq k(m)) = 1 - \hat{p}_{mk(m)}$.
Gini index:	$\sum_{k \neq k'} \hat{p}_{mk} \hat{p}_{mk'} = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$.
Cross-entropy or deviance:	$-\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}$.


email
600/1536
ch$<0.0555
ch$>0.0555
email
280/1177
spam
48/359
remove<0.06
remove>0.06
hp<0.405
hp>0.405
email
180/1065
spam
9/112
spam
26/337
email
0/22
ch!<0.191
ch!>0.191
george<0.15
george>0.15
CAPAVE<2.907
CAPAVE>2.907
email
80/861
email
100/204
spam
6/109
email
0/3
spam
19/110
spam
7/227
george<0.005
george>0.005
CAPAVE<2.7505
CAPAVE>2.7505
1999<0.58
1999>0.58
email
80/652
email
0/209
email
36/123
spam
16/81
spam
18/109
email
0/1
hp<0.03
hp>0.03
free<0.065
free>0.065

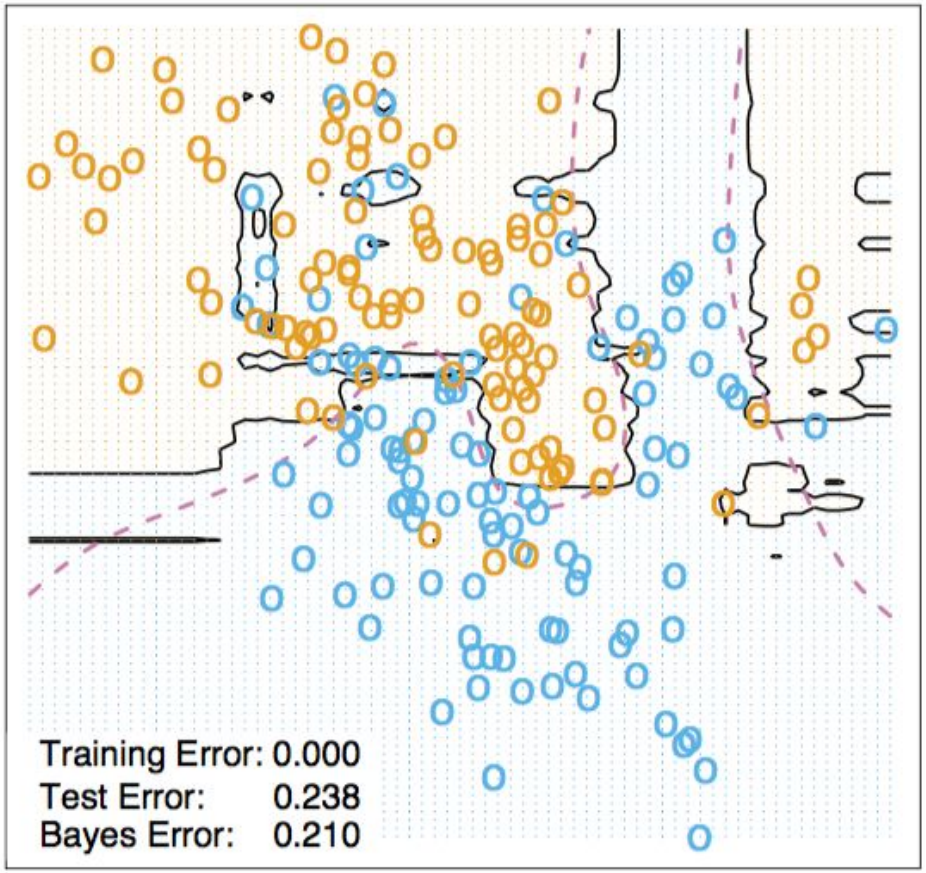
Random Forest

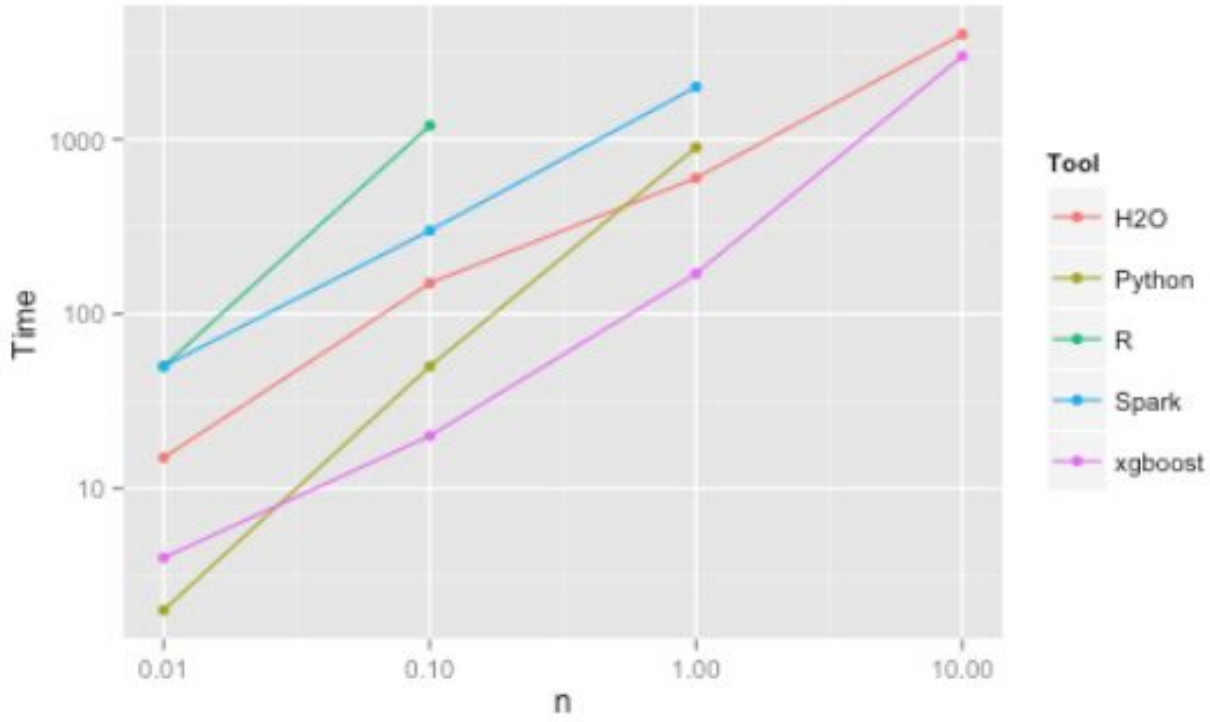
Algorithm 15.1 *Random Forest for Regression or Classification.*

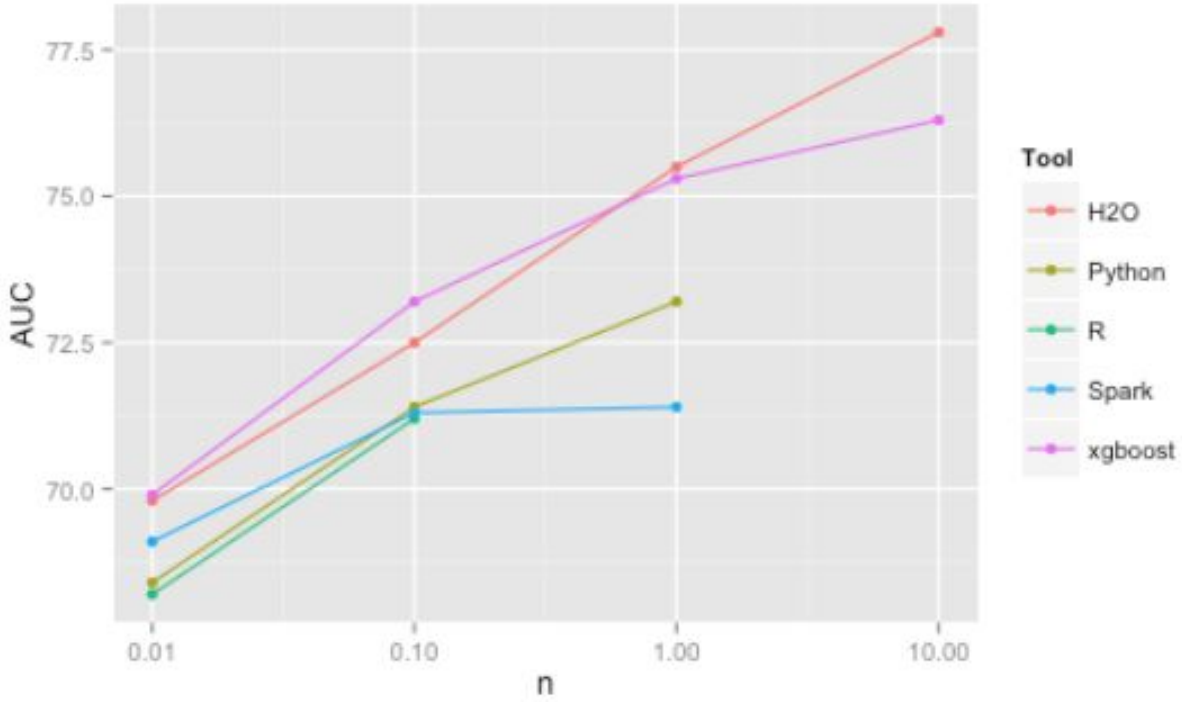
1. For $b = 1$ to B:
   (a) Draw a bootstrap sample $\mathbf{Z}^*$ of size N from the training data.
   (b) Grow a random-forest tree T_b to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached.
      i. Select m variables at random from the p variables.
      ii. Pick the best variable/split-point among the m.
      iii. Split the node into two daughter nodes.
2. Output the ensemble of trees $\{T_b\}_1^B$.

- For classification, the default value for m is $\lfloor\sqrt{p}\rfloor$ and the minimum node size is one.
- For regression, the default value for m is $\lfloor p/3\rfloor$ and the minimum node size is five.

Random Forest Classifier




Time
1000
100
10
0.01
0.10
1.00
10.00
n
Tool
H2O
Python
R
Spark
xgboost


77.5
75.0
72.5
70.0
AUC
0.01
0.10
1.00
10.00
n
Tool
H2O
Python
R
Spark
xgboost

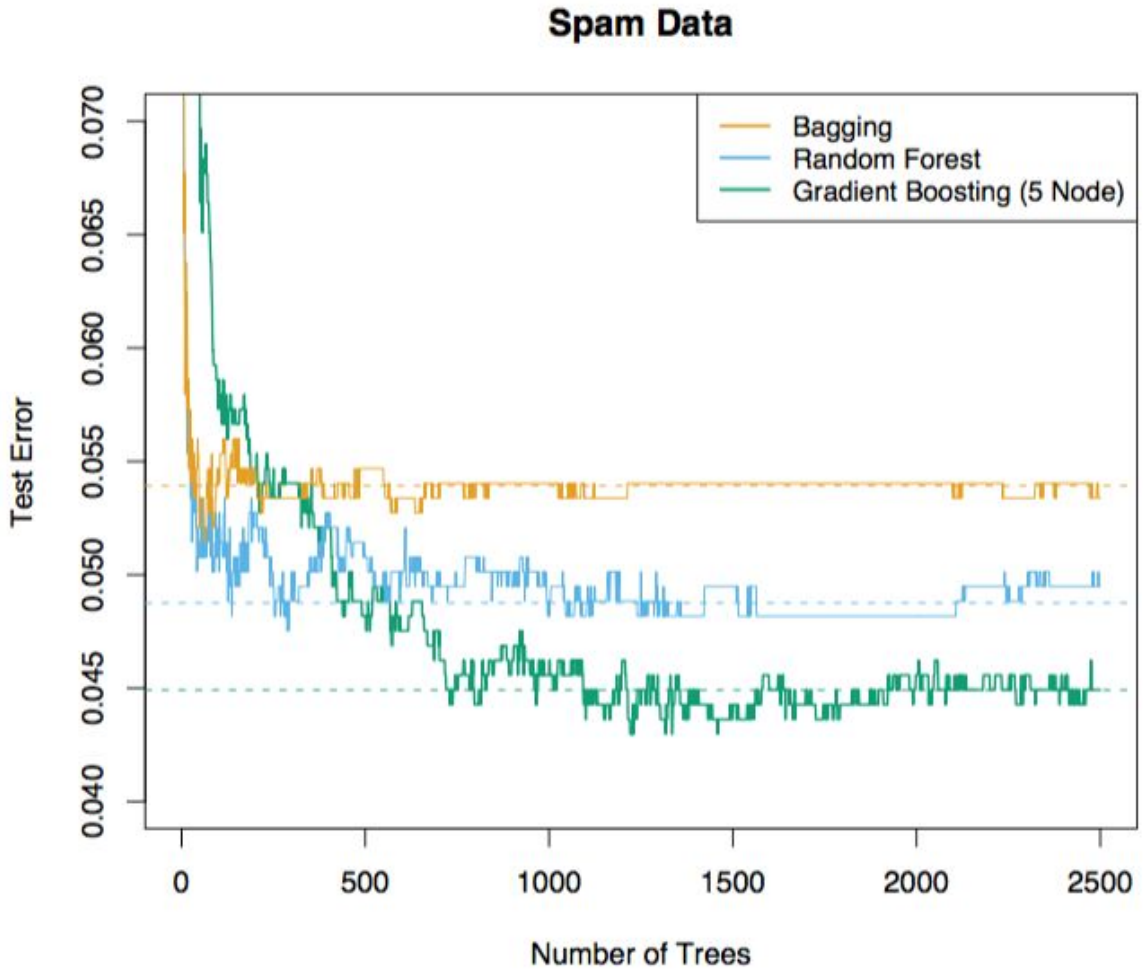
```
h2o.randomForest(x, y, training_frame, model_id, validation_frame = NULL,
  ignore_const_cols = TRUE, checkpoint, mtries = -1,
  col_sample_rate_change_per_level = 1, sample_rate = 0.632,
  sample_rate_per_class, col_sample_rate_per_tree = 1,
  build_tree_one_node = FALSE, ntrees = 50, max_depth = 20,
  min_rows = 1, nbins = 20, nbins_top_level = 1024, nbins_cats = 1024,
  binomial_double_trees = FALSE, balance_classes = FALSE,
  class_sampling_factors, max_after_balance_size = 5, seed,
  offset_column = NULL, weights_column = NULL, 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,
  min_split_improvement = 1e-05, histogram_type = c("AUTO",
  "UniformAdaptive", "Random", "QuantilesGlobal", "RoundRobin"),
  categorical_encoding = c("AUTO", "Enum", "OneHotInternal", "OneHotExplicit",
  "Binary", "Eigen"))
```

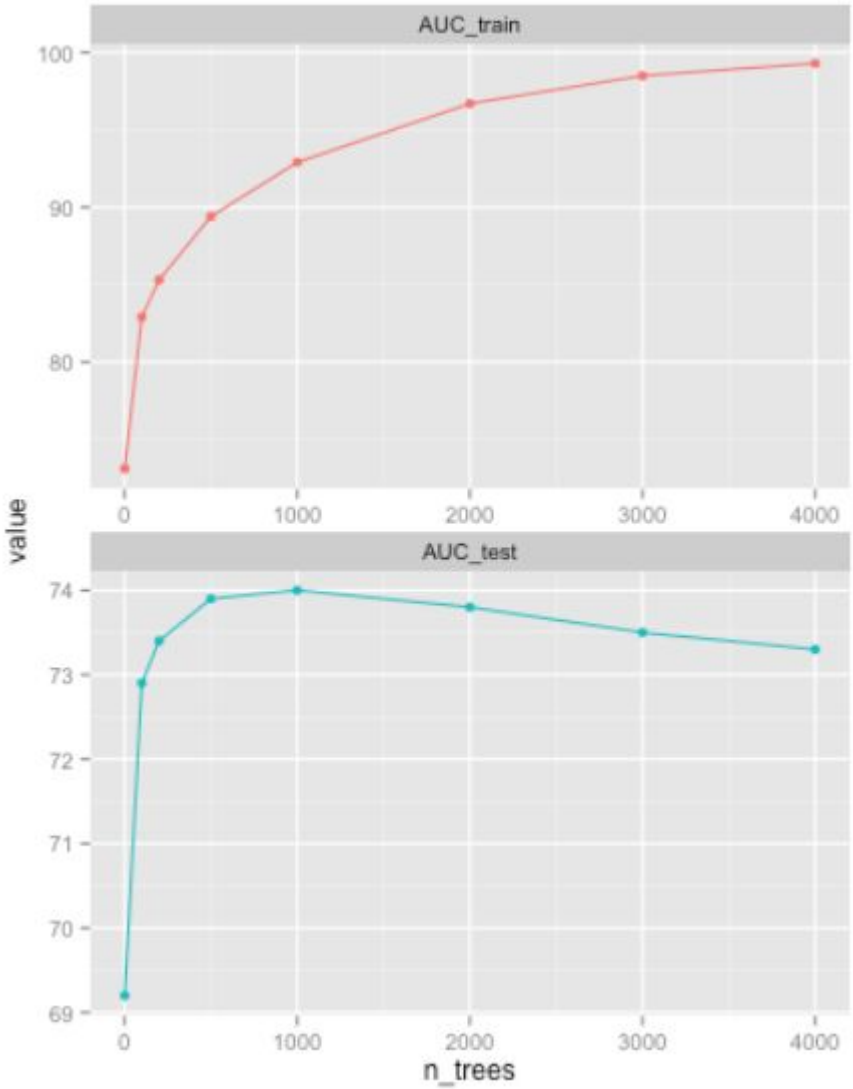
Gradient Boosted Trees

Algorithm 10.1 *AdaBoost.M1.*

1. Initialize the observation weights $w_i = 1/N,\ i = 1, 2, \ldots, N$.
2. For $m = 1$ to M:
   (a) Fit a classifier $G_m(x)$ to the training data using weights w_i.
   (b) Compute
   $$\text{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}.$$
   (c) Compute $\alpha_m = \log((1 - \text{err}_m)/\text{err}_m)$.
   (d) Set $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))],\ i = 1, 2, \ldots, N$.
3. Output $G(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$.

tions. At step m, those observations that were misclassified by the classifier $G_{m-1}(x)$ induced at the previous step have their weights increased, whereas the weights are decreased for those that were classified correctly. Thus as iterations proceed, observations that are difficult to classify correctly receive ever-increasing influence. Each successive classifier is thereby forced


Spam Data
Bagging
Random Forest
Gradient Boosting (5 Node)
Test Error
0.040
0.045
0.050
0.055
0.060
0.065
0.070
0
500
1000
1500
2000
2500
Number of Trees


AUC_train
100
90
80
0
1000
2000
3000
4000
value
AUC_test
74
73
72
71
70
69
0
1000
2000
3000
4000
n_trees

```
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,
  categorical_encoding = c("AUTO", "Enum", "OneHotInternal", "OneHotExplicit",
  "Binary", "Eigen"))
```

blog.h2o.ai/2016/06/h2o-gbm-tuning-tutorial-for-r/

Arno Candel in GBM, R, Technical, Tutorials | June 16, 2016

H2O GBM Tuning Tutorial for R

In this tutorial, we show how to build a well-tuned H2O GBM model for a supervised classification task. and use a small dataset to allow you to reproduce these results in a few minutes on a laptop. This script ca dreds of GBs large and H2O clusters with dozens of compute nodes.

machinelearningmastery.com/configure-gradient-boosting-algorithm/

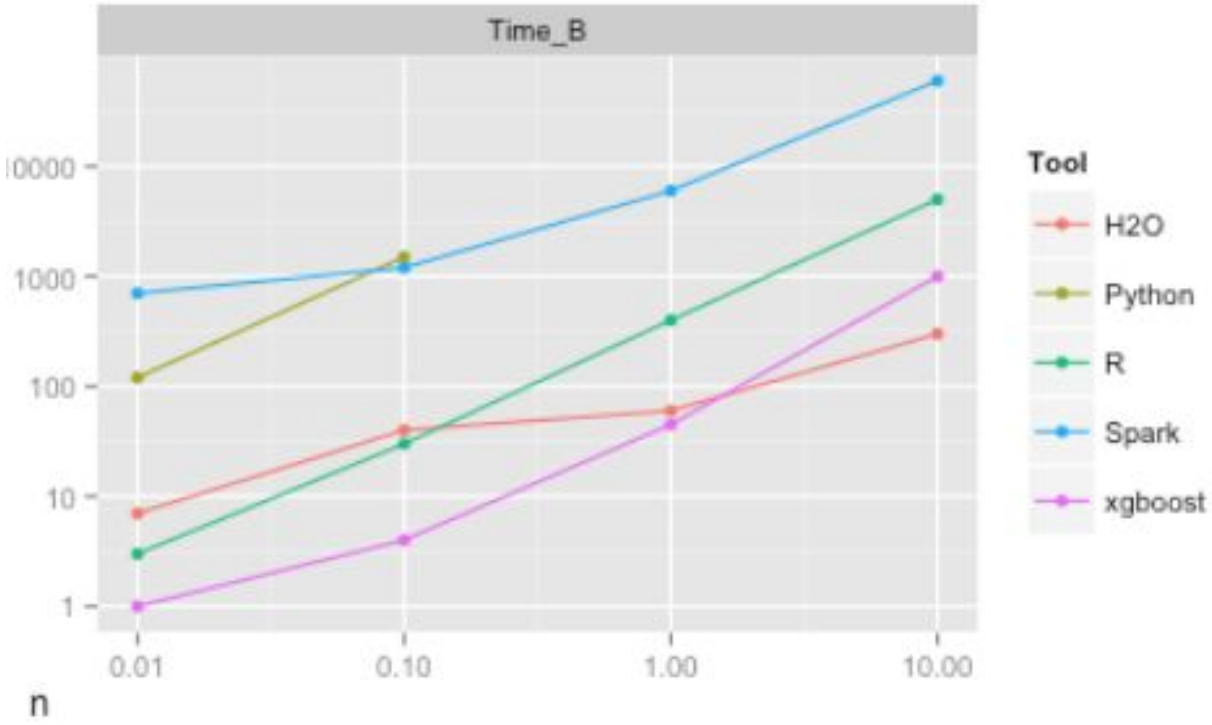
MACHINE LEARNING MASTERY

Start Here

Search...

How to Configure the Gradient Boosting Algorithm

by **Jason Brownlee** on September 12, 2016 in **XGBoost**


Time_B
10000
1000
100
10
1
0.01
0.10
1.00
10.00
n
Tool
H2O
Python
R
Spark
xgboost