

# Better than Deep Learning: Gradient Boosting Machines (GBM) - 2019 edition

Szilard Pafka, PhD  
Chief Scientist, Epoch (USA)

Berlin Buzzwords  
June 2019



Edit profile

## Szilard [Deeper than Deep Learning]

@DataScienceLA

physics PhD, chief (data) scientist, meetup organizer, [datascience.la](https://datascience.la), (visiting) professor, machine learning benchmarks 🇺🇸 🇭🇺 🇪🇺

📍 Santa Monica, California 🔗 [linkedin.com/in/szilard](https://linkedin.com/in/szilard)

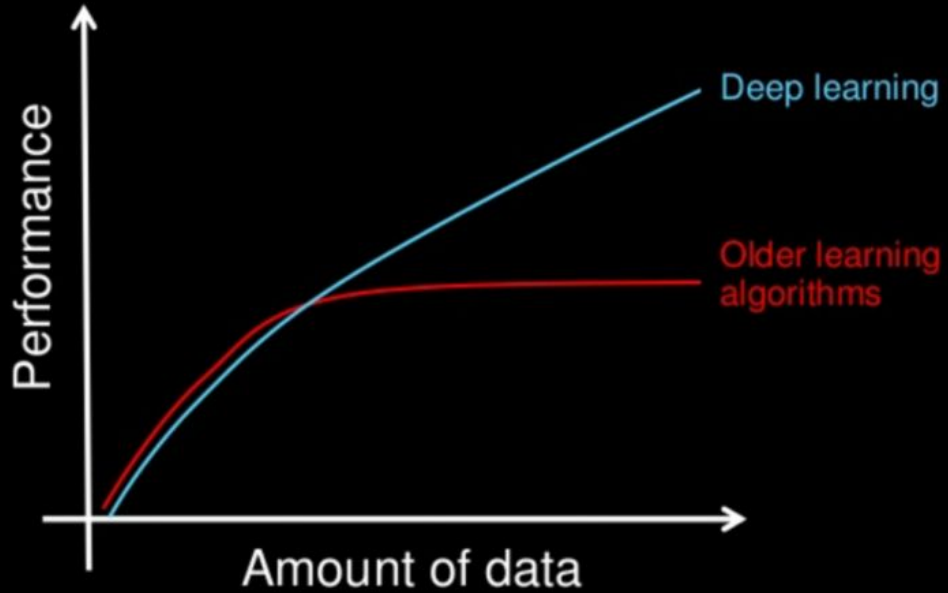
211 Following 2,428 Followers

Disclaimer:

I am not representing my employer (Epoch) in this talk

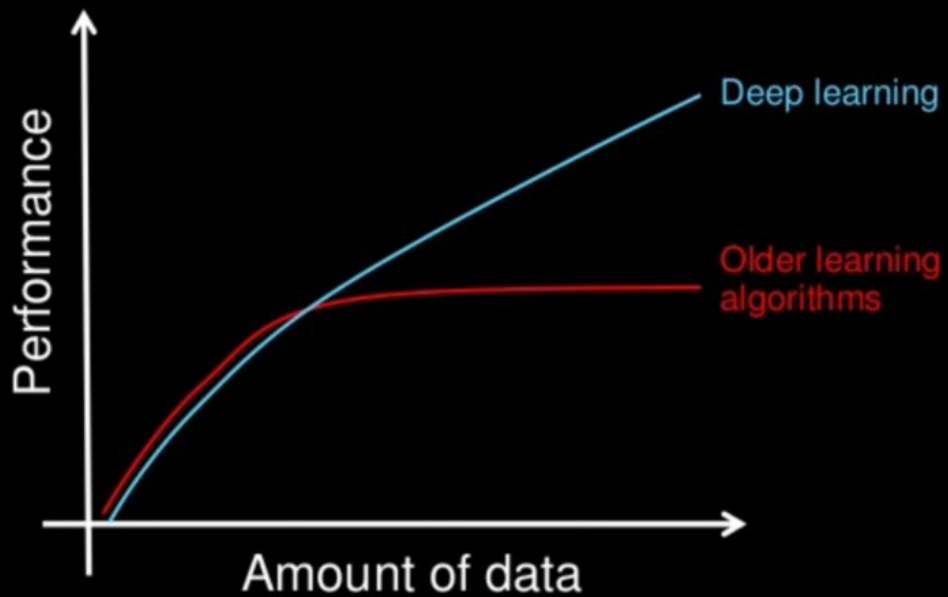
I cannot confirm nor deny if Epoch is using any of the methods, tools, results etc. mentioned in this talk

# Why deep learning



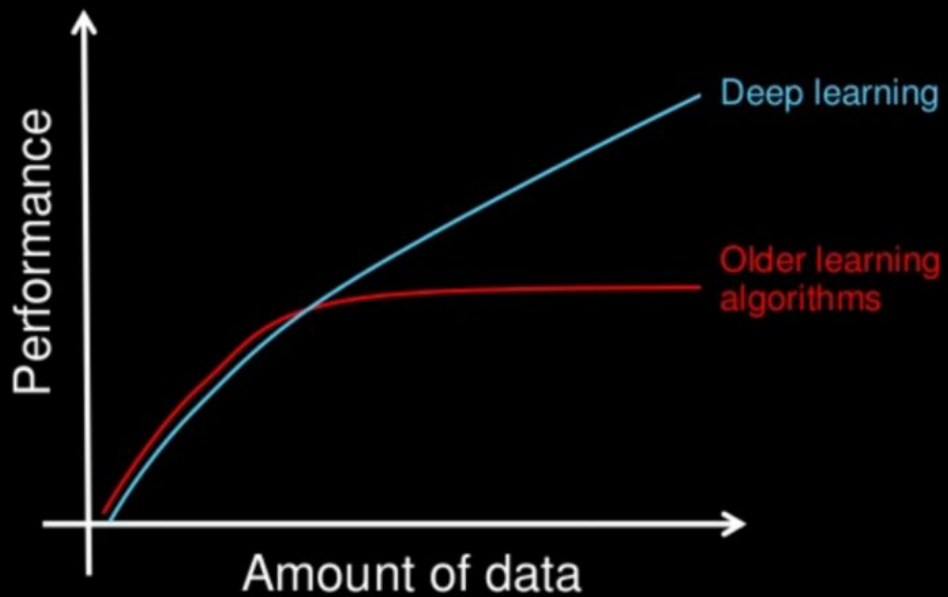
Source: Andrew Ng

# Why deep learning

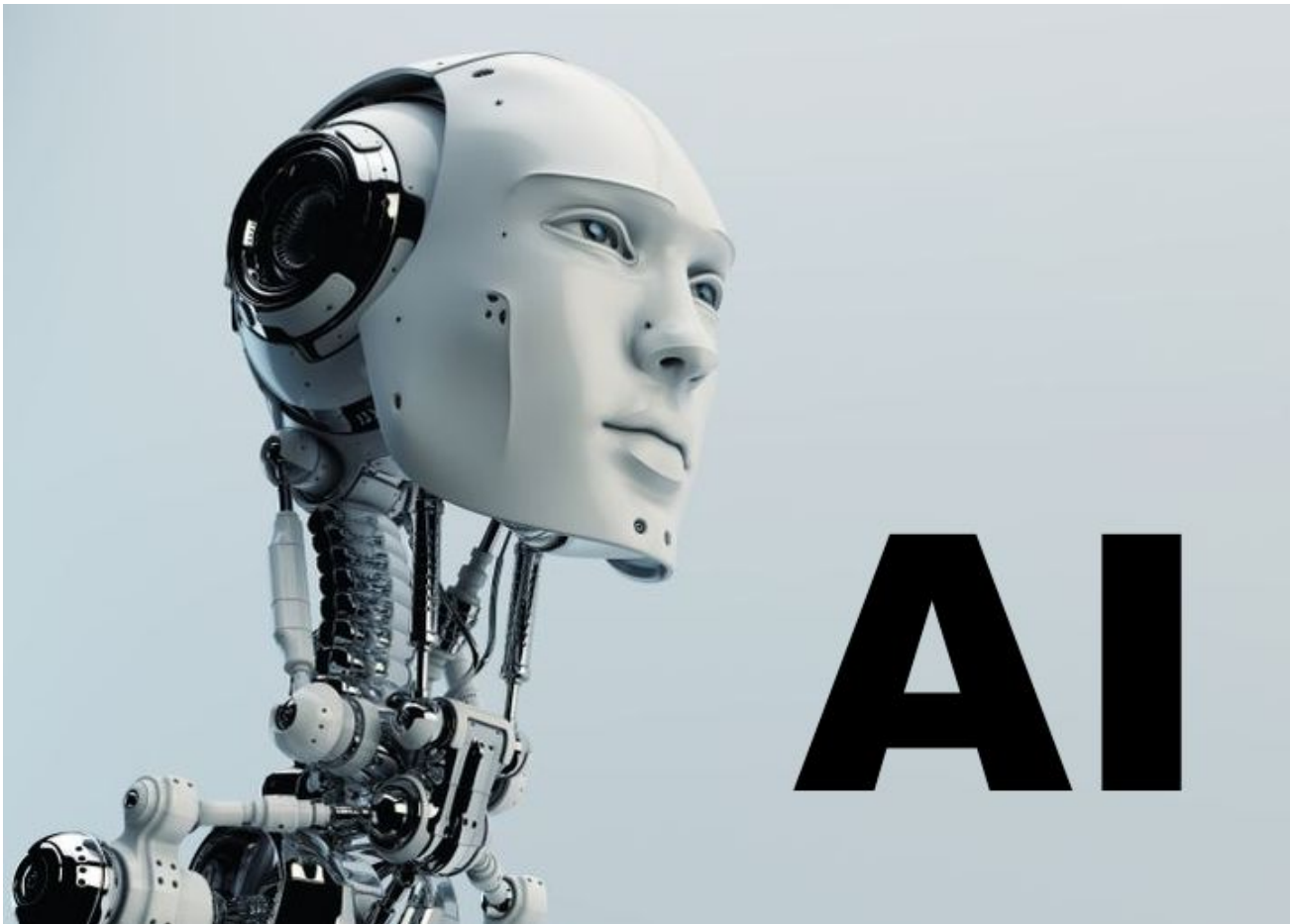


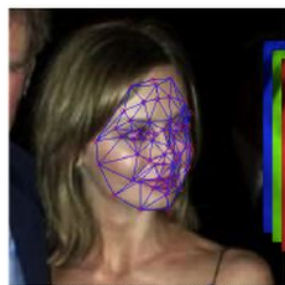
Source: Andrew Ng

# Why deep learning

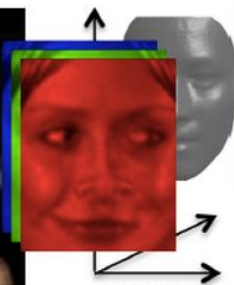


Source: Andrew Ng

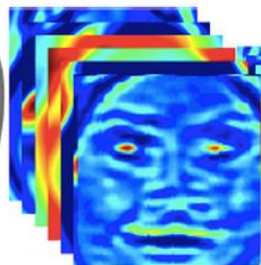




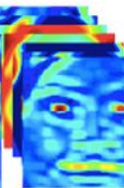
Calista\_Flockhart\_0002.jpg  
Detection & Localization



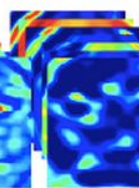
Frontalization:  
@152X152x3



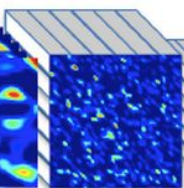
C1:  
32x11x11x3  
@142x142



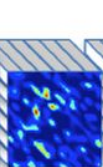
M2:  
32x3x3x32  
@71x71



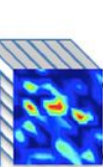
C3:  
16x9x9x32  
@63x63



L4:  
16x9x9x16  
@55x55



L5:  
16x7x7x16  
@25x25



L6:  
16x5x5x16  
@21X21

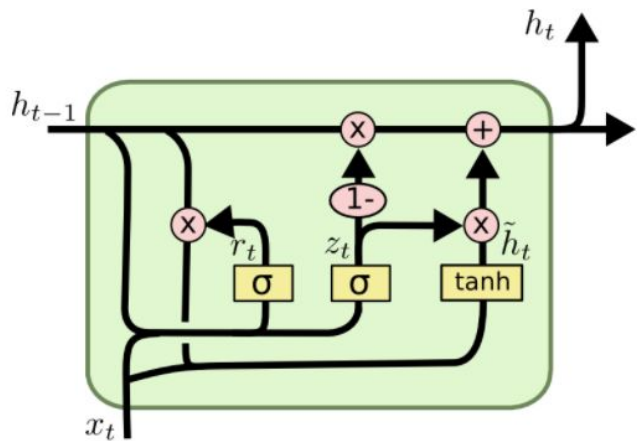
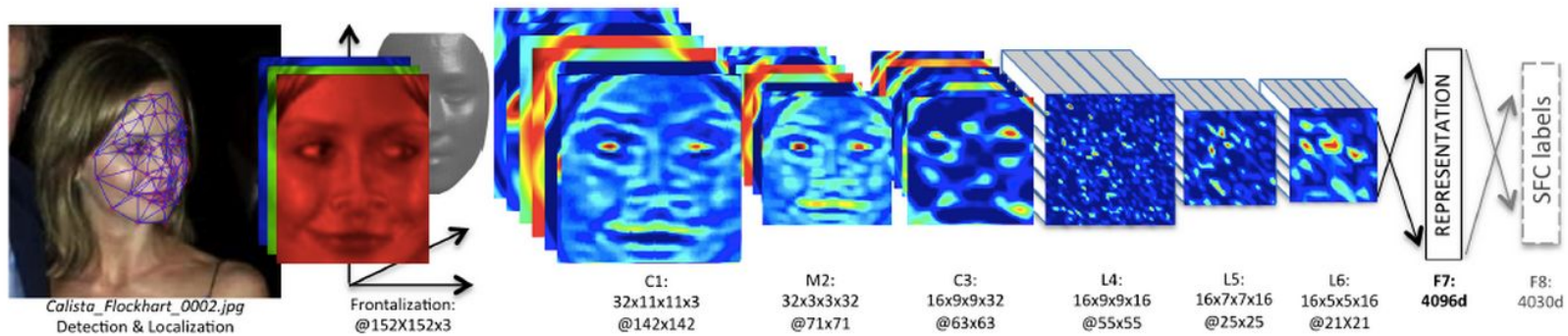


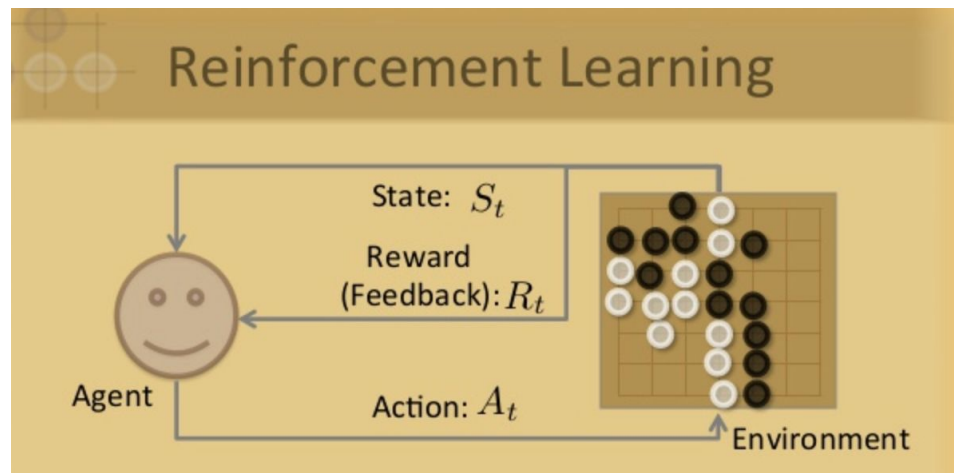
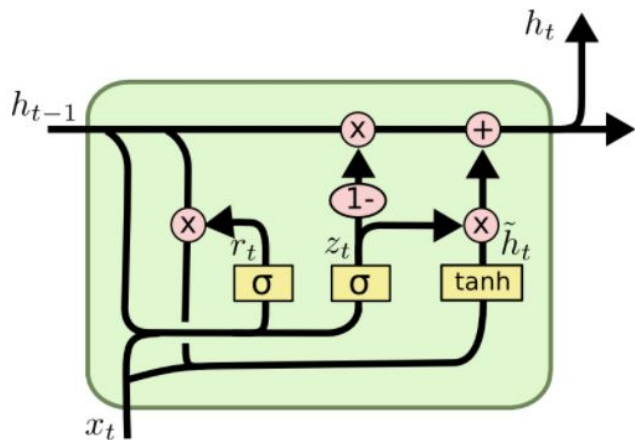
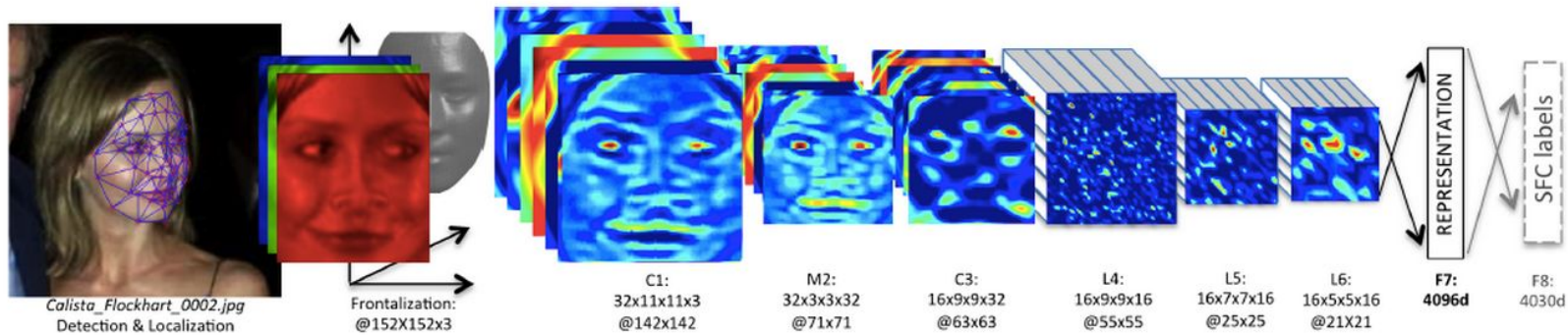
F7:  
4096d



F8:  
4030d









Source: <https://twitter.com/iamdeveloper/>





Params	AUC	Time (s)	Epochs
default: activation = "Rectifier", hidden = c(200,200)	73.1	270	1.8
hidden = c(50,50,50,50), input_dropout_ratio = 0.2	73.2	140	2.7
hidden = c(50,50,50,50)	73.2	110	1.9
hidden = c(20,20)			
hidden = c(20)			



szilard commented Nov 27, 2015

Trying to see if DL can match RF/GBM in accuracy on the airline dataset (wh years 2005-2006, while validation and test sets sampled disjunctly from 20 kept categorical artificially and are intentionally not encoded as ordinal varia structure of business datasets).

RectifierWithDropout, c(200,200,200)			
ADADELTA rho = 0.95, epsilon = 1e-6			
rho = 0.999, epsilon = 1e-08	73.3	270	1.9
adaptive = FALSE default: rate = 0.005, decay = 1, momentum = 0	73.0	340	1.1
rate = 0.001, momentum = 0.5 / 1e5 / 0.99	73.2	410	0.7
rate = 0.01, momentum = 0.5 / 1e5 / 0.99	73.3	280	0.9
rate = 0.01, rate_annealing = 1e-05, momentum = 0.5 / 1e5 / 0.99	73.5	360	1
rate = 0.01, rate_annealing = 1e-04, momentum = 0.5 / 1e5 / 0.99	72.7	3700	8.7
rate = 0.01, rate_annealing = 1e-05, momentum = 0.5 / 1e5 / 0.9	73.4	350	0.9

Machine Learning Challenge Winning Solutions

- The most frequently used tool by data science competition winners
  - 17 out of 29 winning solutions in kaggle last year used XGBoost
  - Solve wide range of problems. *store sales prediction; high seas ship detection; fraud detection; motion detection; ad click through rate prediction; malware classification; classification; customer behavior prediction; hazard risk prediction; massive online course dropout rate prediction; product categorization;*
- Present and Future of KDDCup. Ron Bekkerman (KDDCup 2015 chair): "Something dramatic happened in Machine Learning over the past couple of years. It is called XGBoost - a package implementing Gradient Boosted Decision Trees that works wonders in data classification. Apparently, every winning team used XGBoost, mostly in ensembles with other classifiers. Most surprisingly, the winning teams report very minor improvements that ensembles bring over a single well-configured XGBoost."
- A lot contributions from the kaggle community

## XGBoost A Scalable Tree Boosting System June 02, 2016

26,599 views

👍 212 🗨️ 1 ➦ SHARE ≡ SAVE ...



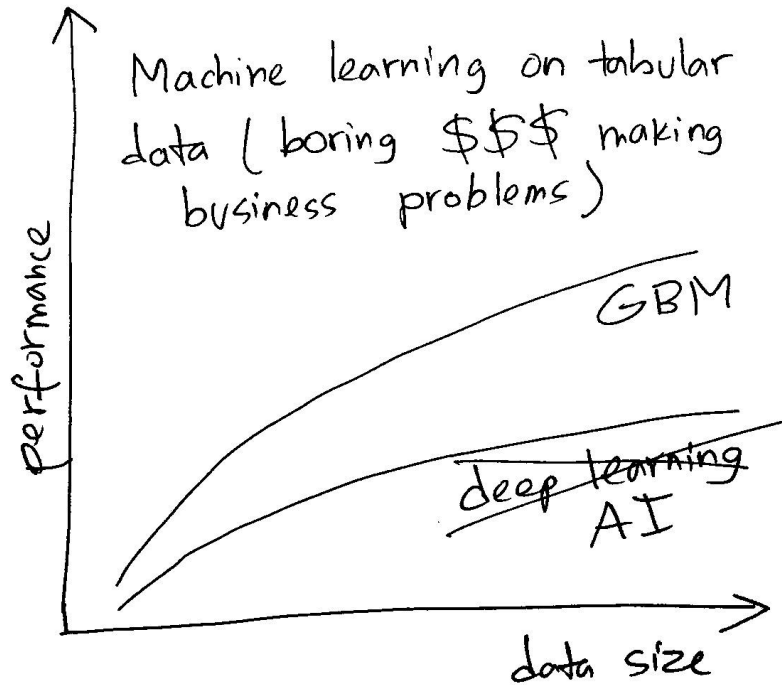
**DataScience.LA**  
Published on Jun 3, 2016

SUBSCRIBE 3.4K



Szilard [Deeper than Deep Learning] @DataScienceLA · 2 Nov 2016

Can anyone **beat** GBMs with deep learning (ahem, AI) on the airline dataset (or generally tabular/business data)? [github.com/szilard/benchm...](https://github.com/szilard/benchm...)

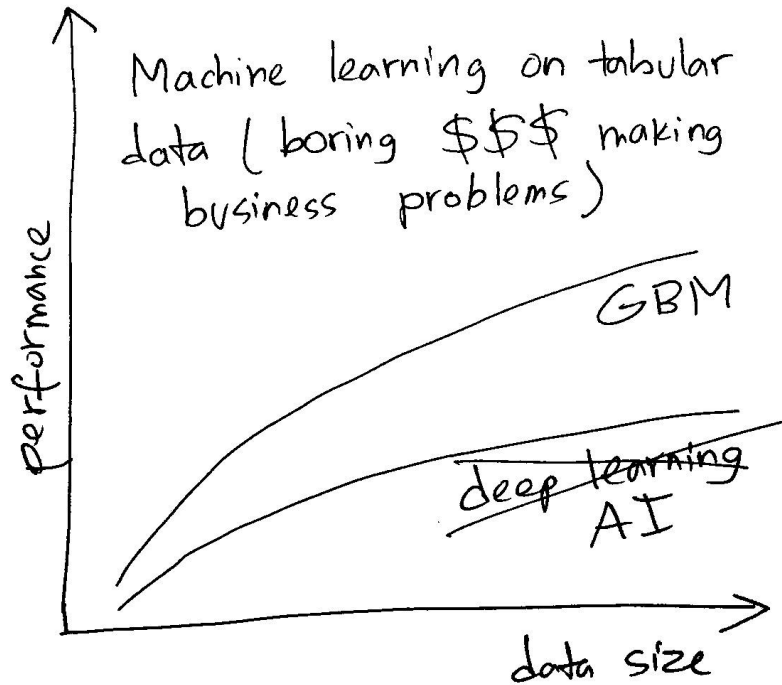






Szilard [Deeper than Deep Learning] @DataScienceLA · 2 Nov 2016

Can anyone **beat** GBMs with deep learning (ahem, AI) on the airline dataset (or generally tabular/business data)? [github.com/szilard/benchm...](https://github.com/szilard/benchm...)



### 3. Parameter tuning and ensembling

```
# train xgboost
xgb <- xgboost(data = data.matrix(tr
  label = train$destinc
  eta = 0.001,
  max_depth = 15,
  nround=25,
  subsample = 0.5,
  colsample_bytree = 0.
  seed = 1,
  eval_metric = "merror
  objective = "multi:sc
  num_class = 12,
  nthread = 4
)
```

2:58 / 4:06

What Kaggle has learned from almost a million data scientists - Anthony Goldbloom

18,153 views



O'Reilly

Published on May 25, 2017

MODEL	1ST	2ND	AVG	1ST	2ND
BST-DT	0.580	0.228	RF	0.727	0.207
RF	0.390	0.525	ANN	0.053	0.172
BAG-DT	0.030	0.232	BSTDT	0.059	0.228
SVM	0.000	0.008	SVM	0.043	0.195
ANN	0.000	0.007	LR	0.089	0.132
KNN	0.000	0.000	BAGDT	0.002	0.012
BST-STMP	0.000	0.000	KNN	0.023	0.045
DT	0.000	0.000	BSTST	0.004	0.009
LOGREG	0.000	0.000	PRC	0	0
NB	0.000	0.000	NB	0	0

An Empirical Comparison of Supervised Learning Algorithms

<http://www.cs.cornell.edu/~alex/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

 <a href="#">gbm_1.5-3.tar.gz</a>	2005-10-07 22:49 249K
 <a href="#">gbm_1.5-5.tar.gz</a>	2006-01-21 12:58 249K
 <a href="#">gbm_1.5-7.tar.gz</a>	2006-04-18 11:58 254K
 <a href="#">gbm_1.5.tar.gz</a>	2005-05-09 22:56 250K
 <a href="#">gbm_1.6-1.tar.gz</a>	2007-06-14 08:29 257K

An Empirical Comparison of Supervised Learning Algorithms

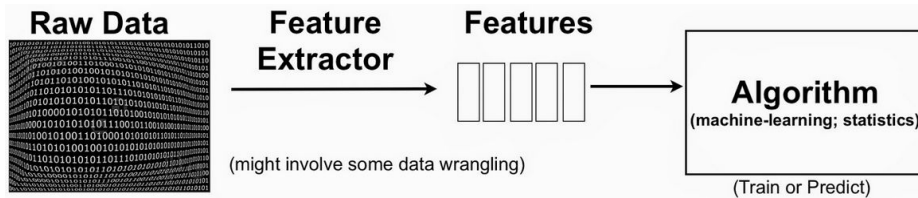
<http://www.cs.cornell.edu/~alexn/papers/empirical.icml06.pdf>

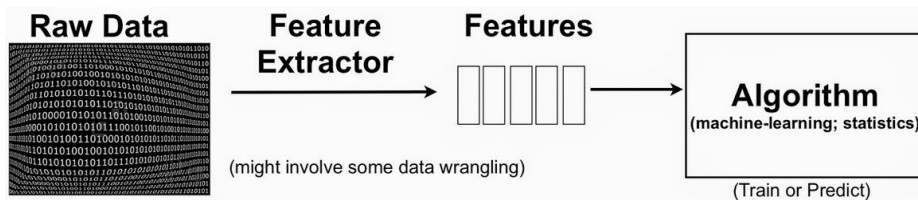
An Empirical Evaluation of Supervised Learning in High Dimensions

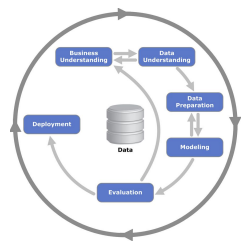
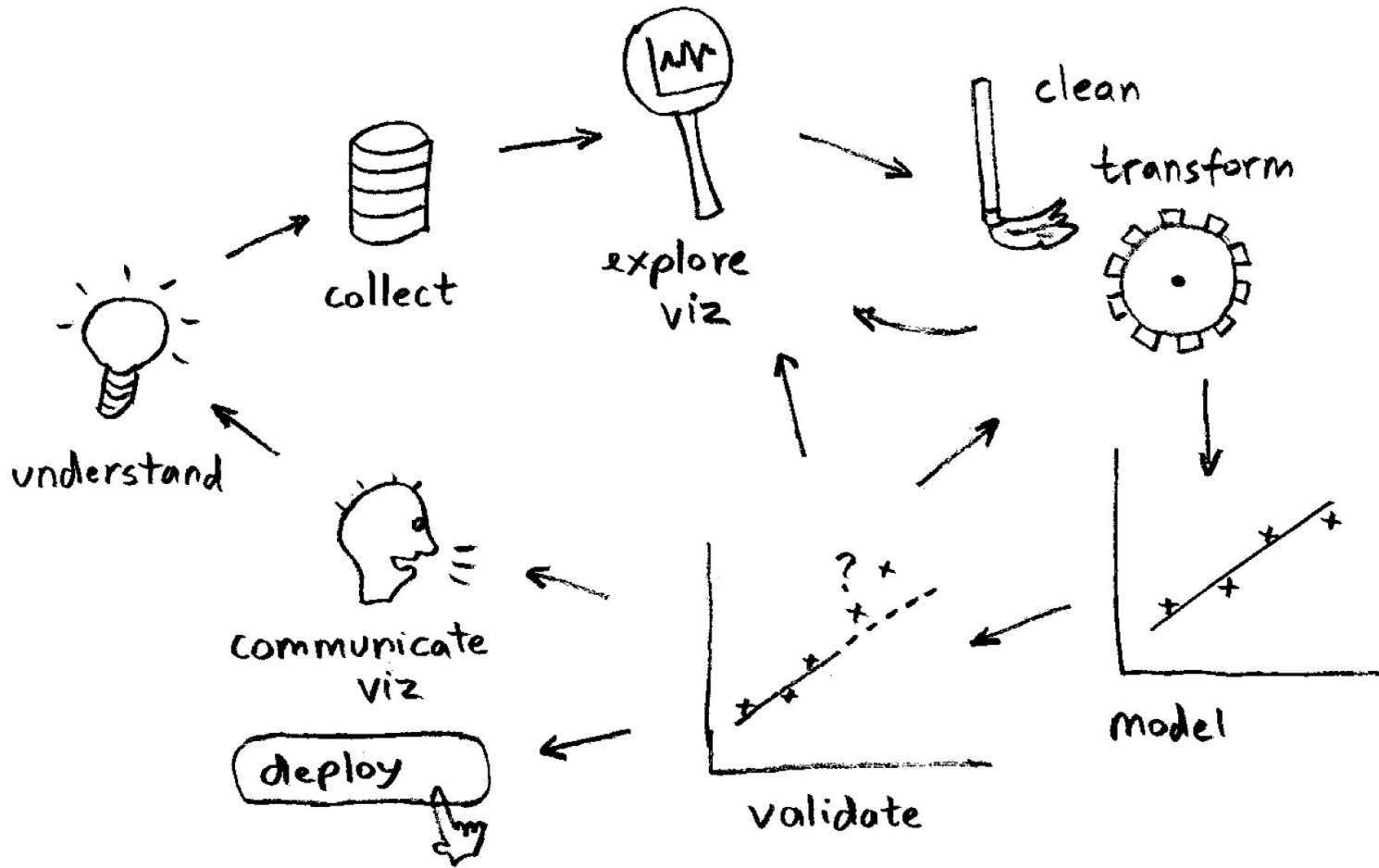
<http://lowrank.net/nikos/pubs/empirical.pdf>

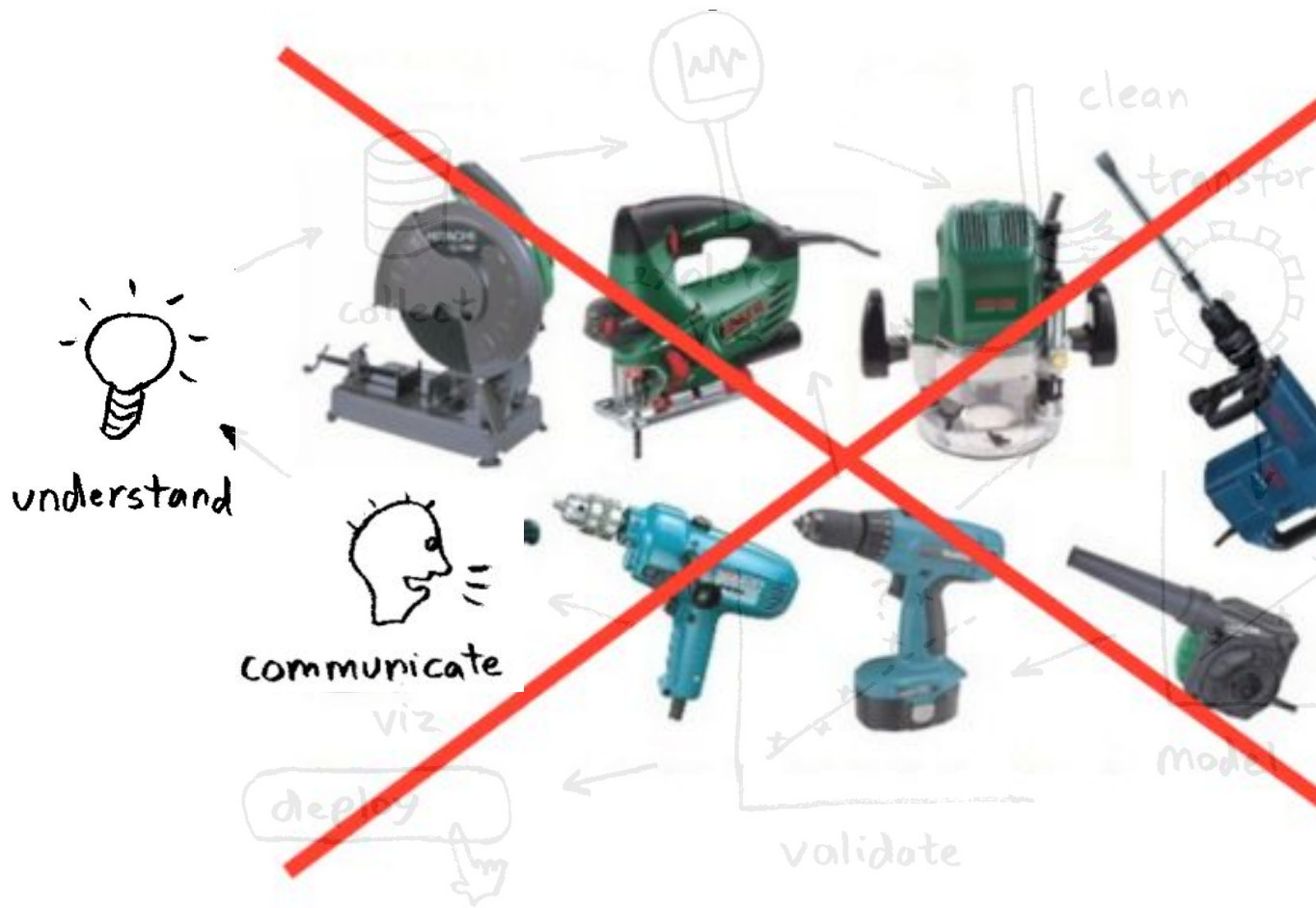
 <a href="#">randomForest_4.5-12.tar.gz</a>	2005-06-21 09:36 80K
 <a href="#">randomForest_4.5-15.tar.gz</a>	2005-09-22 19:35 81K
 <a href="#">randomForest_4.5-16.tar.gz</a>	2006-01-24 10:21 81K
 <a href="#">randomForest_4.5-18.tar.gz</a>	2006-12-10 16:07 67K
 <a href="#">randomForest_4.5-19.tar.gz</a>	2007-10-16 20:38 67K











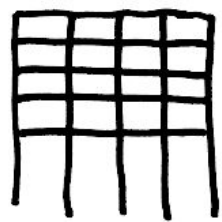
Trevor Hastie  
Robert Tibshirani  
Jerome Friedman

## The Elements of Statistical Learning

Data Mining, Inference, and Prediction

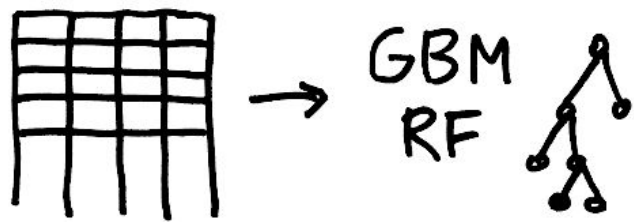
Second Edition

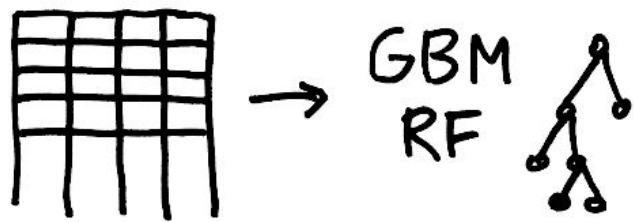


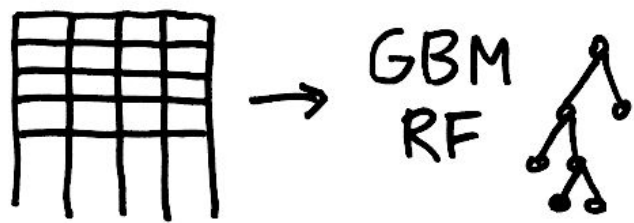


GBM  
RF

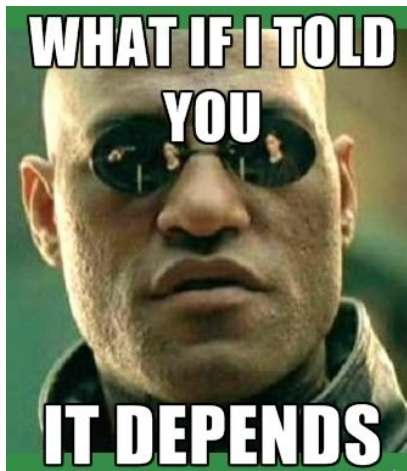














Hyperparameter  
tuning





Hyperparameter  
tuning







Hyperparameter  
tuning



Features





Hyperparameter tuning

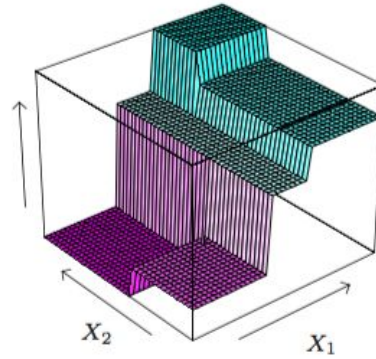
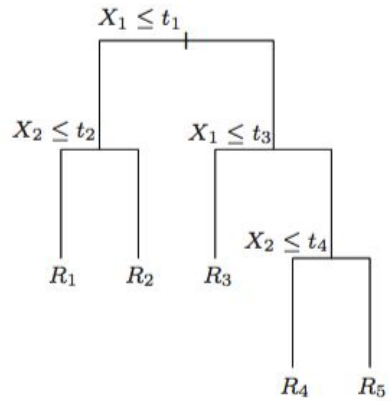
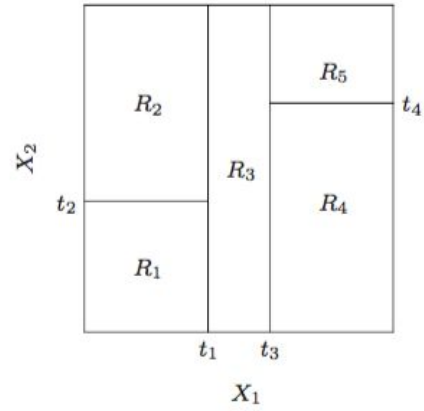
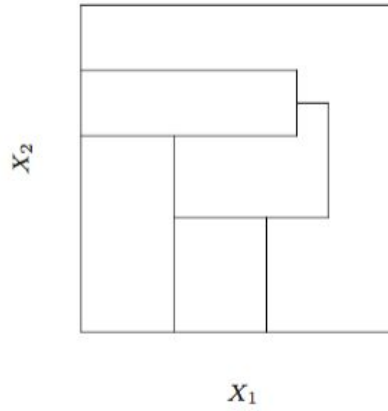


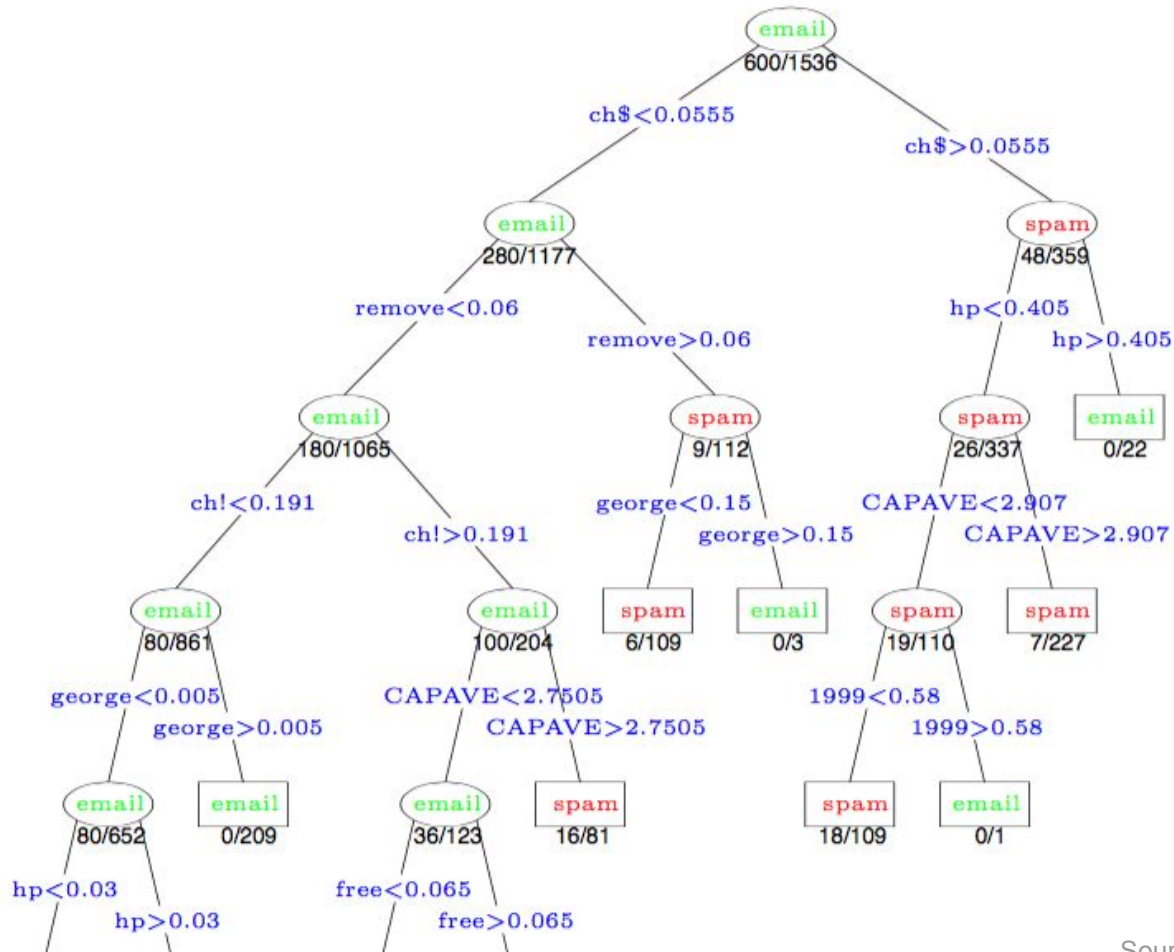
Features



**GBM?**

**TELL ME MORE**





---

**Algorithm 10.1** *AdaBoost.M1*.

---

1. Initialize the observation weights  $w_i = 1/N$ ,  $i = 1, 2, \dots, N$ .
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  $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))]$ ,  $i = 1, 2, \dots, N$ .
3. Output  $G(x) = \text{sign} \left[ \sum_{m=1}^M \alpha_m G_m(x) \right]$ .

---

**Algorithm 10.3** *Gradient Tree Boosting Algorithm.*

---

1. Initialize  $f_0(x) = \arg \min_{\gamma} \sum_{i=1}^N L(y_i, \gamma)$ .

2. For  $m = 1$  to  $M$ :

(a) For  $i = 1, 2, \dots, N$  compute

$$r_{im} = - \left[ \frac{\partial L(y_i, f(x_i))}{\partial f(x_i)} \right]_{f=f_{m-1}}.$$

(b) Fit a regression tree to the targets  $r_{im}$  giving terminal regions  $R_{jm}$ ,  $j = 1, 2, \dots, J_m$ .

(c) For  $j = 1, 2, \dots, J_m$  compute

$$\gamma_{jm} = \arg \min_{\gamma} \sum_{x_i \in R_{jm}} L(y_i, f_{m-1}(x_i) + \gamma).$$

(d) Update  $f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$ .

3. Output  $\hat{f}(x) = f_M(x)$ .

Trevor Hastie  
Robert Tibshirani  
Jerome Friedman

## The Elements of Statistical Learning

Data Mining, Inference, and Prediction

Second Edition



open source

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





open source

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

**100% FREE**



open source

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



**100% FREE**





open source

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



**100% FREE**





open source

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



**100% FREE**





open source

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

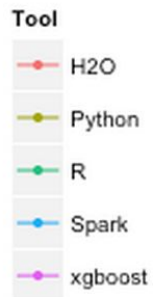
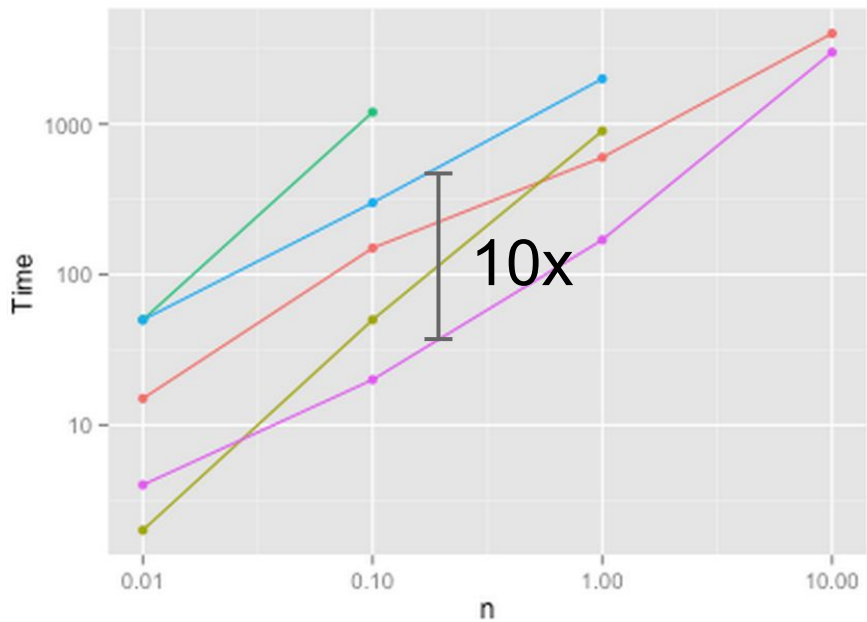


szilard / [benchm-ml](#)

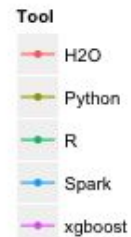
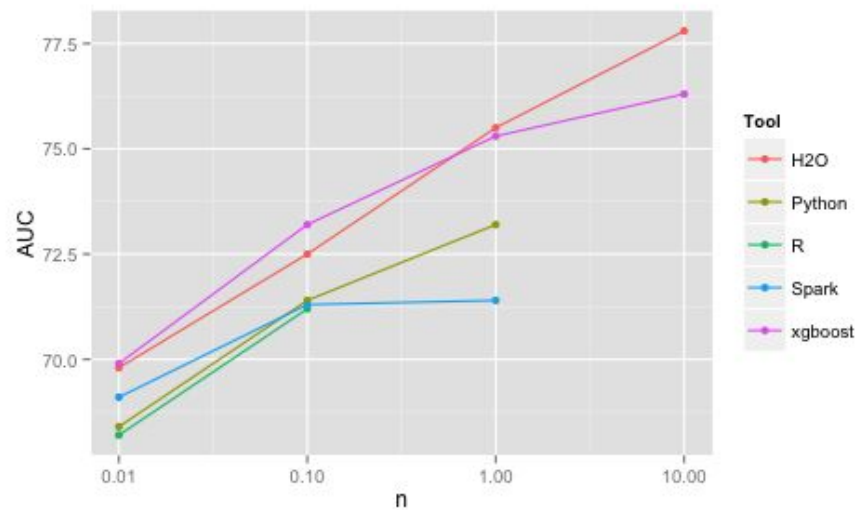
★ Star

1,203

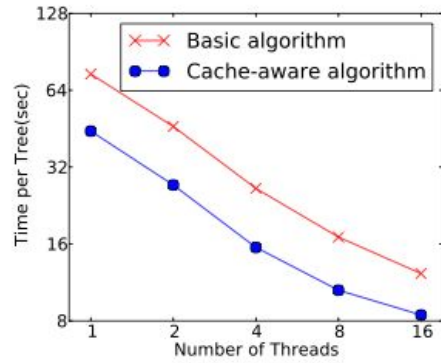
**Simple/limited/incomplete benchmark**



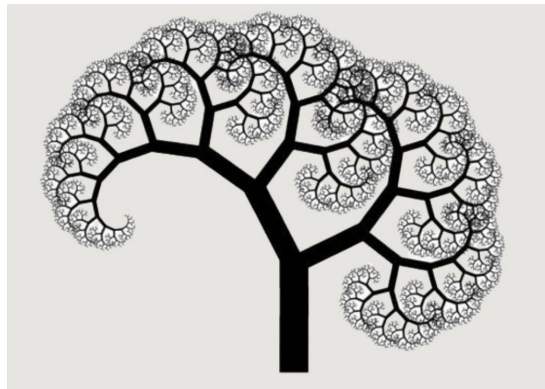
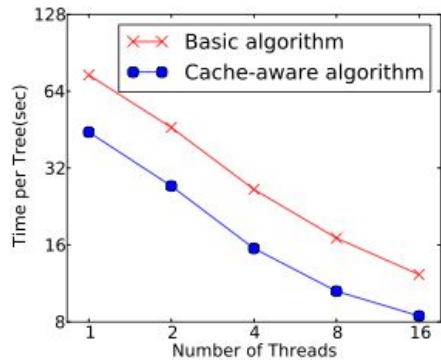
 szilard / benchm-ml



## XGBoost: A Scalable Tree Boosting System



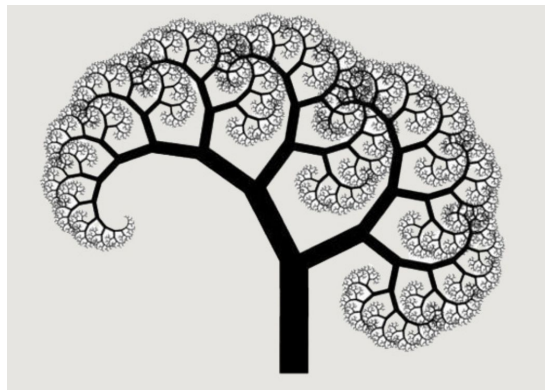
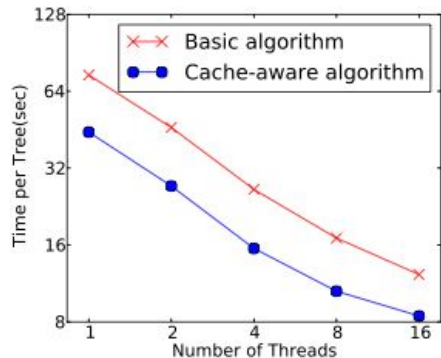
## XGBoost: A Scalable Tree Boosting System



 Microsoft / LightGBM



## XGBoost: A Scalable Tree Boosting System



Microsoft / LightGBM



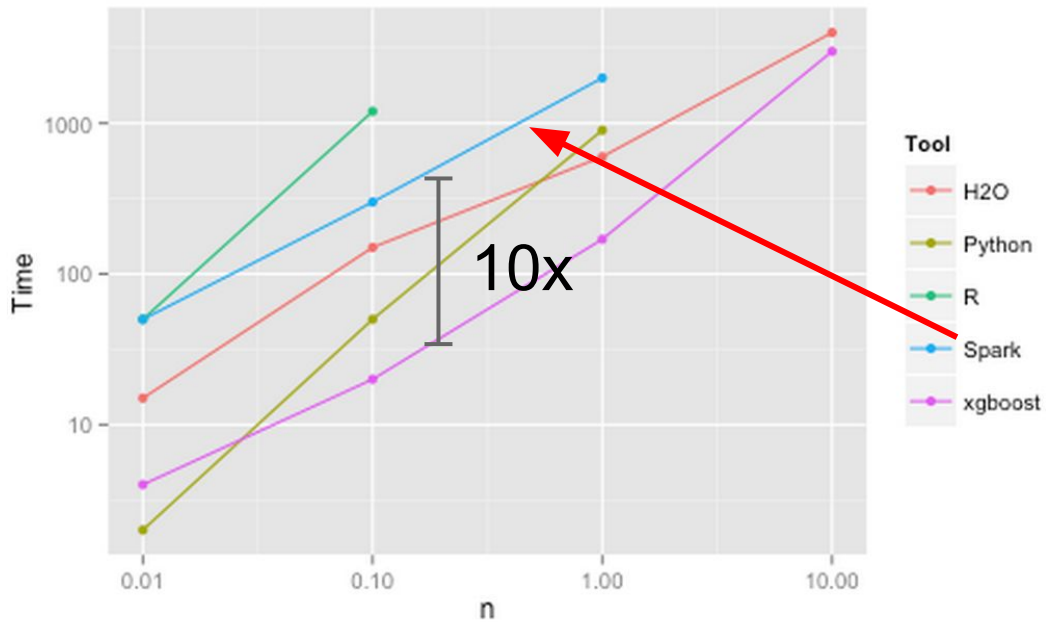
<https://cran.r-project.org/web/packages/xgboost/>

xgboost: Extreme Gradient Boosting

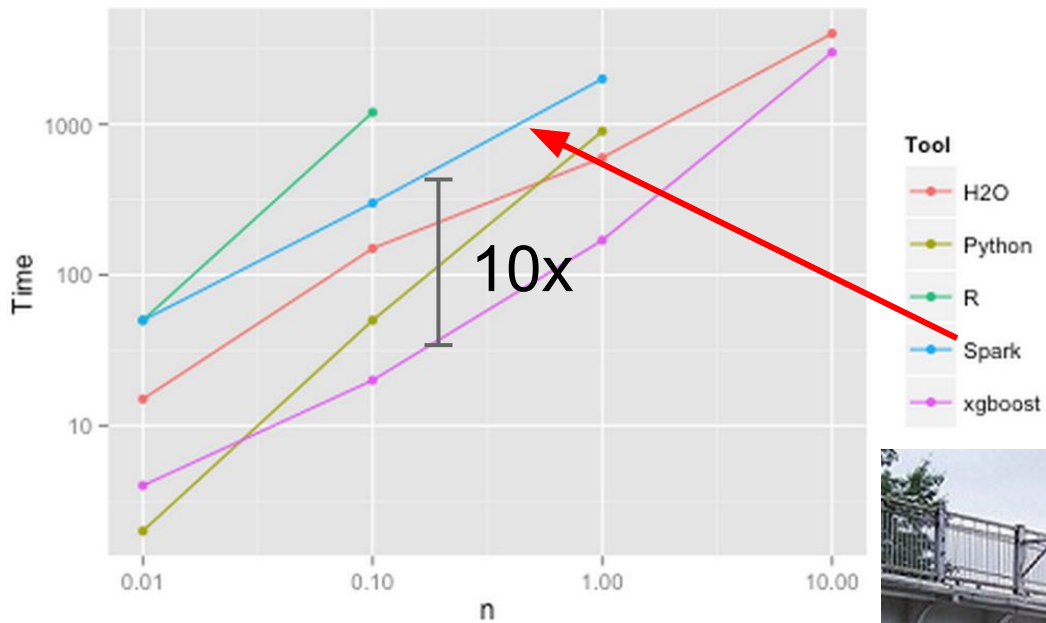
<https://cran.r-project.org/packages/h2o/>

h2o: R Interface for H2O





 [szilard / benchm-ml](#)

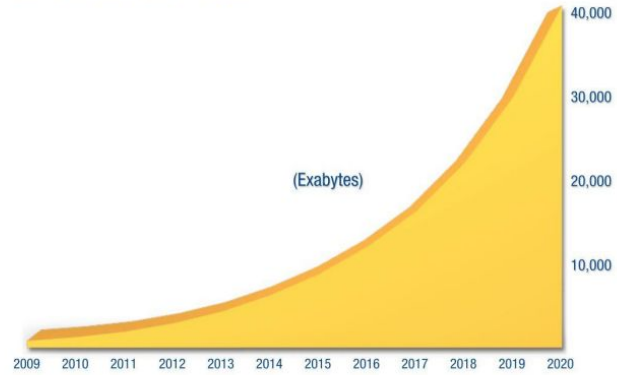


 [szilard / benchm-ml](#)



Figure 1

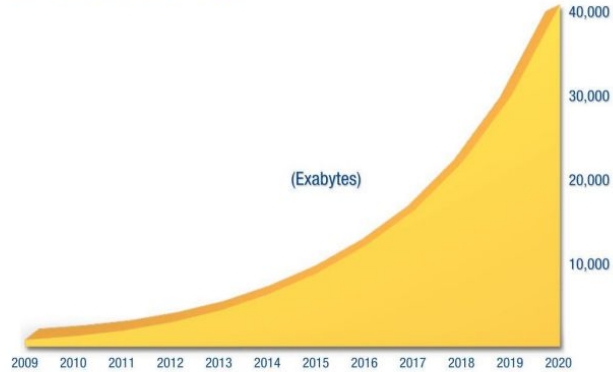
## The Digital Universe: 50-fold Growth from the Beginning of 2010 to the End of 2020



Source: IDC's Digital Universe Study, sponsored by EMC, December 2012

Figure 1

The Digital Universe: 50-fold Growth from the Beginning of 2010 to the End of 2020



Source: IDC's Digital Universe Study, sponsored by EMC, December 2012



**Hadley Wickham**

@hadleywickham

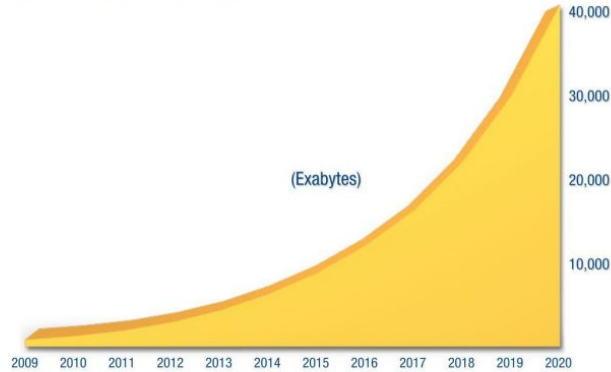


Following

"It takes a big man to admit his data is small" —  
[@jcheng](#)

Figure 1

## The Digital Universe: 50-fold Growth from the Beginning of 2010 to the End of 2020



Source: IDC's Digital Universe Study, sponsored by EMC, December 2012



**Hadley Wickham**

@hadleywickham



Following

"It takes a big man to admit his data is small" —  
[@jcheng](#)

## TYPICAL SIZE OF DATASETS

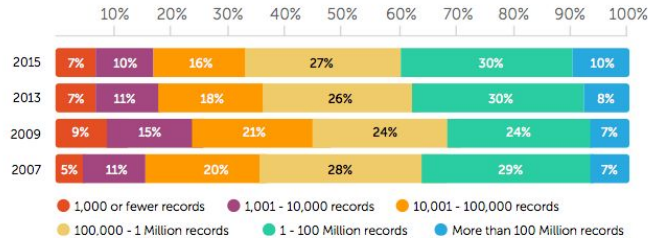
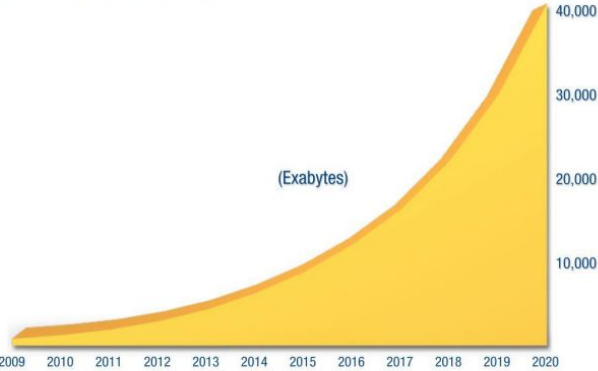


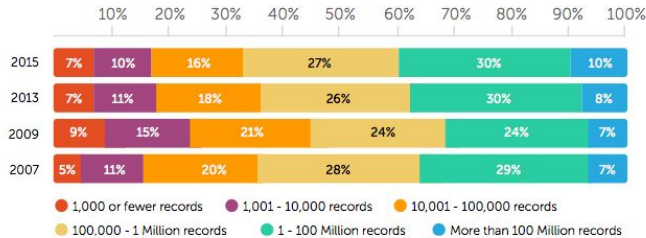
Figure 1

### The Digital Universe: 50-fold Growth from the Beginning of 2010 to the End of 2020



Source: IDC's Digital Universe Study, sponsored by EMC, December 2012

### TYPICAL SIZE OF DATASETS



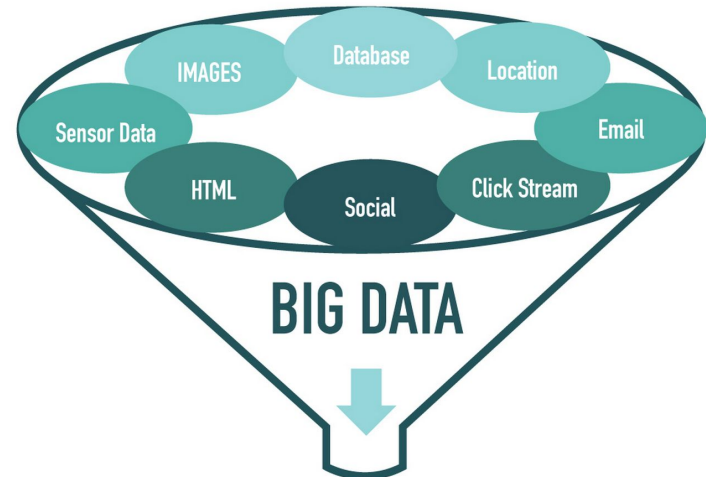
Hadley Wickham

@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](#)

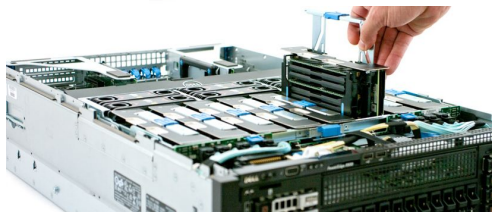
[Be the first to review this item](#)

Was: \$743.99

Price: **\$743.96** & **FREE Shipping**. [Details](#)



Model	vCPU	Mem (GiB)
r3.8xlarge	32	244
x1e.32xlarge	128	3,904
u-12tb1.metal	448	12 <b>(TiB)</b>



## 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](#)



Model	vCPU	Mem (GiB)
r3.8xlarge	32	244
x1e.32xlarge	128	3,904
u-12tb1.metal	448	12 <b>(TiB)</b>



**Szilard** @DataScienceLA · 18 Nov 2015

Big RAM is eating [#bigdata](#): datasets for analytics grew 20% /yr (last decade [@kdnuggets](#)), RAM EC2 grew 50% /yr



**Szilard** @DataScienceLA · Aug 3

I wish my [#machinelearning](#) worked... ("both" is not a choice 😊) [#bigdata](#)  
[#datascience](#) [#rstats](#) [#pydata](#) cc [@h2o](#) [@databricks](#) [@cloudera](#) [@kaggle](#)

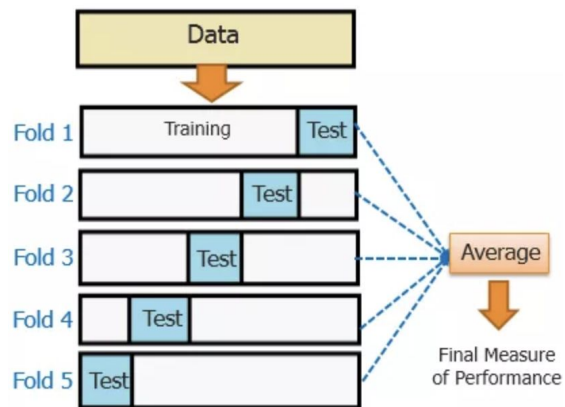
**10%** on 10x bigger data

**70%** 10x faster

**20%** I don't care about either

104 votes • Final results





Hyperparameter tuning



## szilard / GBM-perf

```
git clone https://github.com/szilard/GBM-perf
cd GBM-perf/cpu
sudo docker build -t gbmpref_cpu
sudo docker run --rm gbmpref_cpu
```

r4.8xlarge (32 cores, but run on physical cores only/no hyperthreading) with software as of

Tool	Time[s] 100K	Time[s] 1M	Time[s] 10M	AUC 1M	AUC 10M
h2o	16	20	100	0.762	0.776
xgboost	3.8	12	78	0.749	0.755
lightgbm	2.4	5.2	42	0.764	0.774
catboost	5.4	50	490	0.740	0.744

## szilard / GBM-perf

```
git clone https://github.com/szilard/GBM-perf
cd GBM-perf/cpu
sudo docker build -t gbmp perf_cpu
sudo docker run --rm gbmp perf_cpu
```

r4.8xlarge (32 cores, but run on physical cores only/no hyperthreading) with software as of

Tool	Time[s] 100K	Time[s] 1M	Time[s] 10M	AUC 1M	AUC 10M
h2o	16	20	100	0.762	0.776
xgboost	3.8	12	78	0.749	0.755
lightgbm	2.4	5.2	42	0.764	0.774
catboost	5.4	50	490	0.740	0.744

p3.2xlarge (1 GPU, Tesla V100) with software as of 2019-04-29:

Tool	Time[s] 100K	Time[s] 1M	Time[s] 10M	AUC 1M	AUC 10M
h2o xgboost	9	14	60	0.749	0.756
xgboost	2.4	4.8	13	0.750	0.756
lightgbm	10	16	67	0.766	0.774
catboost	3.9	10	135	0.742	0.750



## 100M records and RAM usage

CPU (m5.12xlarge):

Tool	time [s]	AUC	RAM train [GB]
h2o	520	0.775	8
xgboost	510	0.751	15
<b>lightgbm</b>	<b>310</b>	<b>0.774</b>	<b>5</b>
catboost	3360	0.723 ?!	140



## 100M records and RAM usage

CPU (m5.12xlarge):

Tool	time [s]	AUC	RAM train [GB]
h2o	520	0.775	8
xgboost	510	0.751	15
<b>lightgbm</b>	<b>310</b>	<b>0.774</b>	<b>5</b>
catboost	3360	0.723 ?!	140

GPU (Tesla V100):

Tool	time [s]	AUC	GPU mem [GB]	extra RAM [GB]
h2o xgboost	270	0.755	4	30
<b>xgboost</b>	<b>80</b>	<b>0.756</b>	<b>6</b>	<b>0</b>
lightgbm	400	0.774	3	6
catboost	crash (OOM)		>16	14

```
## exporting model for scoring
```

```
h2o.download_mojo(md_rf, path = "./h2o")
```

---

```
## building prediction service
```

```
# (need jetty-runner.jar ROOT.war from Steam)
```

```
java -jar jetty-runner.jar ROOT.war
```

```
curl -X POST --form mojo=@h2o_RF.zip --form jar=@h2o-genmodel.jar \  
localhost:8080/makewar > h2o_RF_M0J0.war
```

GitHub Gist

Search...



szilard / [h2o\\_scoring.R](#)

**H<sub>2</sub>O**.ai

```
## exporting model for scoring
```

```
h2o.download_mojo(md_rf, path = "./h2o")
```

---

```
## building prediction service
```

```
# (need jetty-runner.jar ROOT.war from Steam)
```

```
java -jar jetty-runner.jar ROOT.war
```

```
curl -X POST --form mojo=@h2o_RF.zip --form jar=@h2o-genmodel.jar \  
localhost:8080/makewar > h2o_RF_MOJO.war
```

---

```
## run prediction service
```

```
java -jar jetty-runner.jar --port 20000 h2o_RF_MOJO.war
```

---

```
## score via REST API
```

```
time curl "http://localhost:20000/predict?Month=c-8&DayofMonth=c-21&Day  
# (fast scoring needs JVM to warm up with a few requests)
```

GitHub Gist

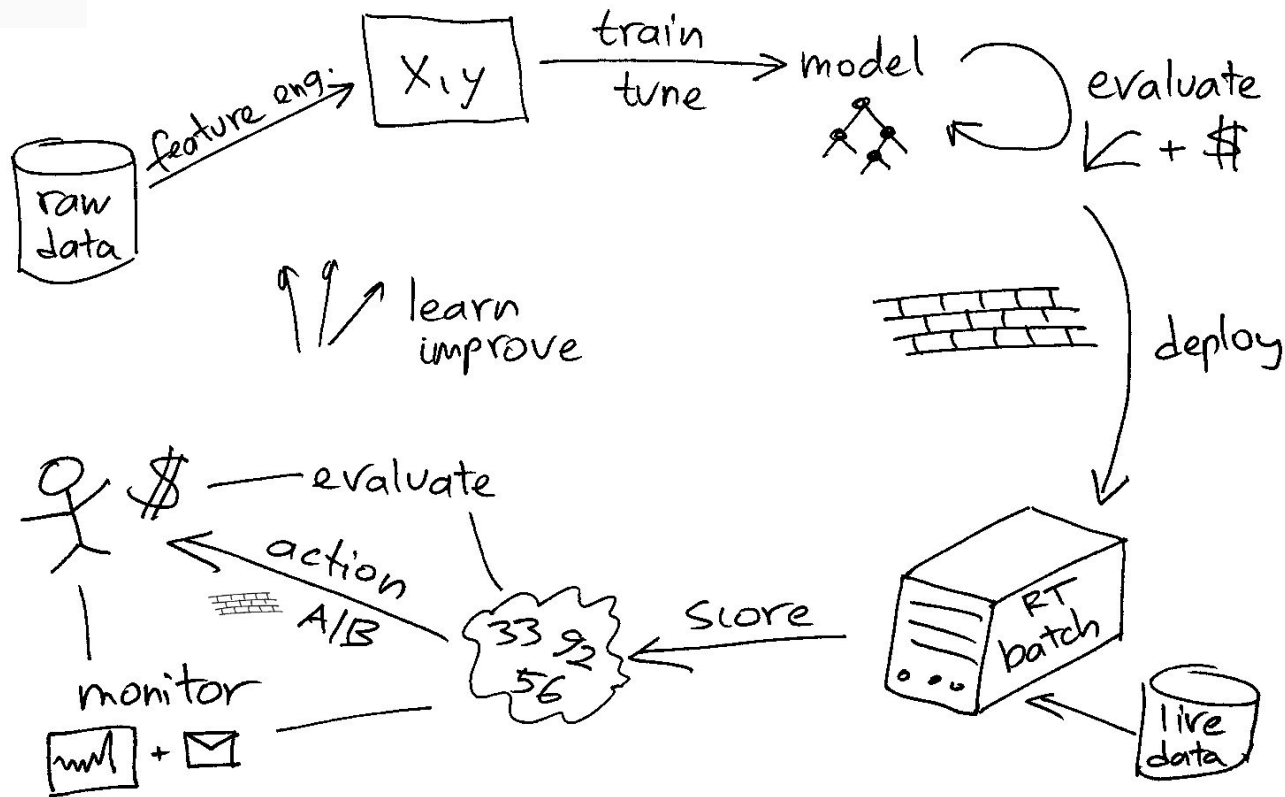
Search...



szilard / h2o\_scoring.R



H<sub>2</sub>O.ai





size	time lgbm [s]	time spark [s]	ratio
100K	2.4	1020	425
1M	5.2	1380	265
10M	42	8390	200



size	time lgbm [s]	time spark [s]	ratio
100K	2.4	1020	425
1M	5.2	1380	265
10M	42	8390	200

		100M			10M		
trees	depth	time [s]	AUC	RAM [GB]	time [s]	AUC	RAM [GB]
1	1	1150	0.634	620	70	0.635	110
1	10	1350	0.712	620	90	0.712	112
10	10	7850	0.731	780	830	0.731	125
100	10	crash OOM		>960 (OOM)	8390	0.755	230



size	time lgbm [s]	time spark [s]	ratio
100K	2.4	1020	425
1M	5.2	1380	265
10M	42	8390	200

		100M			10M		
trees	depth	time [s]	AUC	RAM [GB]	time [s]	AUC	RAM [GB]
1	1	1150	0.634	620	70	0.635	110
1	10	1350	0.712	620	90	0.712	112
10	10	7850	0.731	780	830	0.731	125
100	10	crash OOM		>960 (OOM)	8390	0.755	230

lightgbm 5GB RAM usage



size	time lgbm [s]	time spark [s]	ratio
100K	2.4	1020	425
1M			
10M			

		100M		
trees	depth	time [s]	AUC	RAM [GB]
1	1	1150	0.634	620
1	10	1350	0.712	620
10	10	7850	0.731	780
100	10	crash OOM		>960 (OOM)

lightgbm 5GB RAM usage







Spark 10 trees, depth 10:

size	system	nodes	cores	partitions	time [s]	AUC	RAM [GB]	total RAM [GB]
10M	local	r4.8xl	32	32	830	0.731	125	240
10M	Cluster_1	r4.8xl	32	64	1180	0.731	73	240
10M	Cluster_10	r4.8xl	320	320 (m)	330	0.73		2400
100M	local	x1e.8xl	32		7850	0.731	780	960
100M	Cluster_10	r4.8xl	320	585	1825	0.731	10*72	2400





Spark 10 trees, depth 10:

size	system	nodes	cores	partitions	time [s]	AUC	RAM [GB]	total RAM [GB]
10M	local	r4.8xl	32	32	830	0.731	125	240
10M	Cluster_1	r4.8xl	32	64	1180	0.731	73	240
10M	Cluster_10	r4.8xl	320	320 (m)	330	0.73		2400
100M	local	x1e.8xl	32		7850	0.731	780	960
100M	Cluster_10	r4.8xl	320	585	1825	0.731	10*72	2400



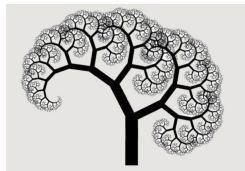
Spark 10 trees, depth 10:

size	system	nodes	cores	partitions	time [s]	AUC	RAM [GB]	total RAM [GB]
10M	local	r4.8xl	32	32	830	0.731	125	240
10M	Cluster_1	r4.8xl	32	64	1180	0.731	73	240
10M	Cluster_10	r4.8xl	320	320 (m)	330	0.73		2400
100M	local	x1e.8xl	32		7850	0.731	780	960
100M	Cluster_10	r4.8xl	320	585	1825	0.731	10*72	2400



Spark 10 trees, depth 10:

size	system	nodes	cores	partitions	time [s]	AUC	RAM [GB]	total RAM [GB]
10M	local	r4.8xl	32	32	830	0.731	125	240
10M	Cluster_1	r4.8xl	32	64	1180	0.731	73	240
10M	Cluster_10	r4.8xl	320	320 (m)	330	0.73		2400
100M	local	x1e.8xl	32		7850	0.731	780	960
100M	Cluster_10	r4.8xl	320	585	1825	0.731	10*72	2400



dmlc  
**XGBoost**

lib	size	hw	cores	time [s]	AUC	RAM [GB]	total RAM [GB]
lightgbm	10M	c5.2xl	1 (m)	29	0.743		16
xgboost	10M	c5.2xl	1 (m)	38	0.726		16
xgboost	10M	i5-5200U	1 (m)	71	0.726	6	8
lightgbm	10M	r4.8xl	16 (m)	7	0.743	4	240
lightgbm	100M	r4.8xl	16 (m)	60	0.743	13(d)+5	240

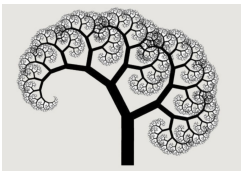
xgboost/lightgbm: 100 trees runtime is <10x 10 trees, RAM = 10 trees

Spark: 100 trees runtime = 10x 10 trees, RAM > 10 trees



Spark 10 trees, depth 10:

size	system	nodes	cores	partitions	time [s]	AUC	RAM [GB]	total RAM [GB]
10M	local	r4.8xl	32	32	830	0.731	125	240
10M	Cluster_1	r4.8xl	32	64	1180	0.731	73	240
10M	Cluster_10	r4.8xl	320	320 (m)	330	0.73		2400
100M	local	x1e.8xl	32		7850	0.731	780	960
100M	Cluster_10	r4.8xl	320	585	1825	0.731	10*72	2400

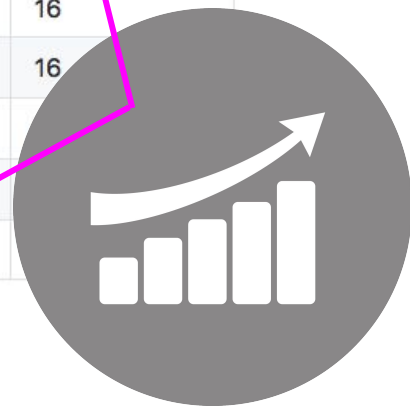


dmlc  
**XGBoost**

lib	size	hw	cores	time [s]	AUC	RAM [GB]	total RAM [GB]
lightgbm	10M	c5.2xl	1 (m)	29	0.743		16
xgboost	10M	c5.2xl	1 (m)	38	0.726		16
xgboost	10M	i5-5200U	1 (m)	71	0.726	6	
lightgbm	10M	r4.8xl	16 (m)	7	0.743	4	
lightgbm	100M	r4.8xl	16 (m)	60	0.743	13(d)+5	

xgboost/lightgbm: 100 trees runtime is <10x 10 trees, RAM = 10 trees

Spark: 100 trees runtime = 10x 10 trees, RAM > 10 trees





Spark 10 trees, depth 10:

size	system	nodes	cores	partitions	time [s]	AUC	RAM [GB]	total RAM [GB]
10M	local	r4.8xl	32	32	830	0.731	125	240
10M	Cluster_1	r4.8xl	32	64	1180	0.731	73	240
10M	Cluster_10	r4.8xl	320	320 (m)	330	0.73		2400
100M	local	x1e.8xl	32		7850	0.731	780	960
100M	Cluster_10	r4.8xl	320	585	1825	0.731	10*72	2400

dmlc  
**XGBoost**



lib	size	hw	cores	time [s]	AUC	RAM [GB]	total RAM [GB]
lightgbm	10M	c5.2xl	1 (m)	29	0.743		16
xgboost	10M	c5.2xl	1 (m)	38	0.726		16
xgboost	10M	i5-5200U	1 (m)	71	0.726	6	8
lightgbm	10M	r4.8xl	16 (m)	7	0.743	4	240
lightgbm	100M	r4.8xl	16 (m)	60	0.743	13(d)+5	240

xgboost/lightgbm: 100 trees runtime is <10x 10 trees, RAM = 10 trees

Spark: 100 trees runtime = 10x 10 trees, RAM > 10 trees



Spark 10 trees, depth 10:

size	system	nodes	cores	partitions	time [s]	AUC	RAM [GB]	total RAM [GB]
10M	local	r4.8xl	32	32	830	0.731	125	240
10M	Cluster_1	r4.8xl	32	64	1180	0.731	73	240
10M	Cluster_10	r4.8xl	320	320 (m)	330	0.73		2400
100M	local	x1e.8xl	32		7850			
100M	Cluster_10	r4.8xl	320	585	1825			

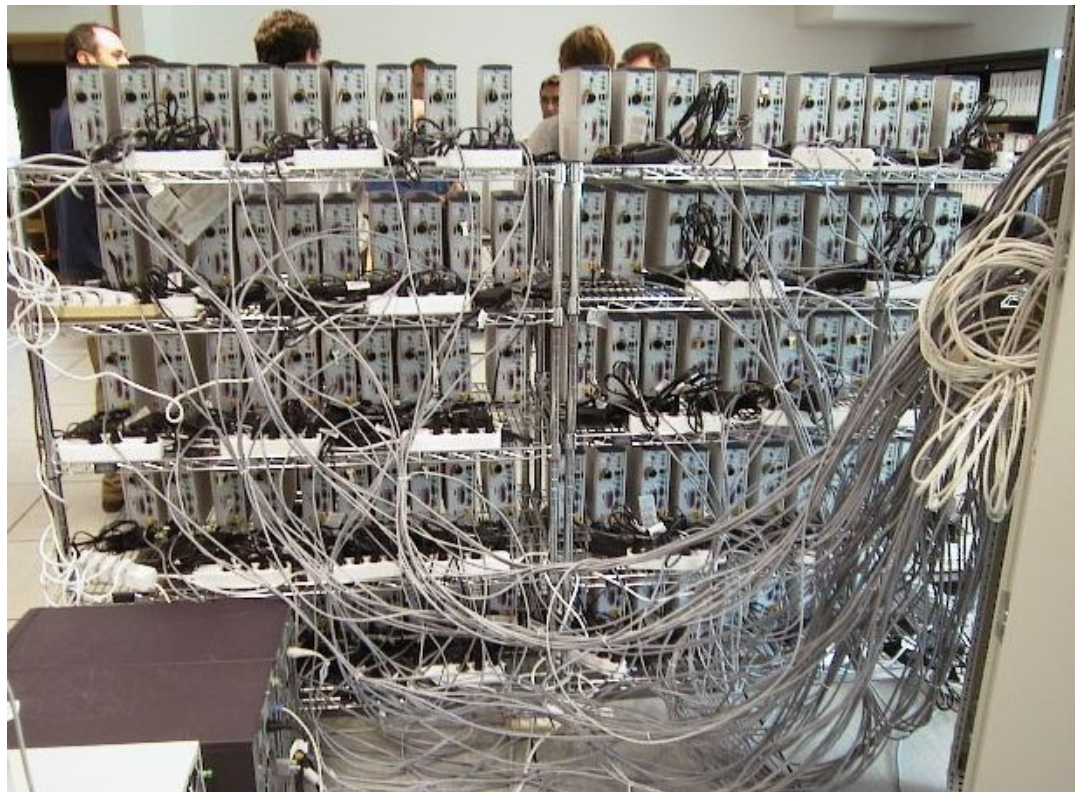
dmlc  
**XGBoost**



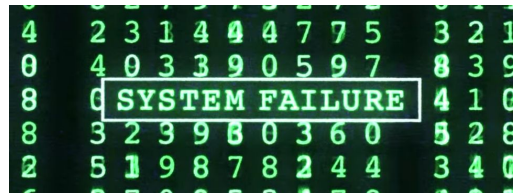
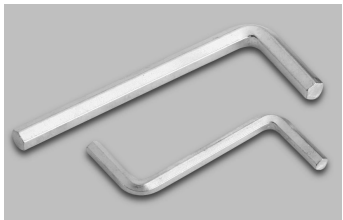
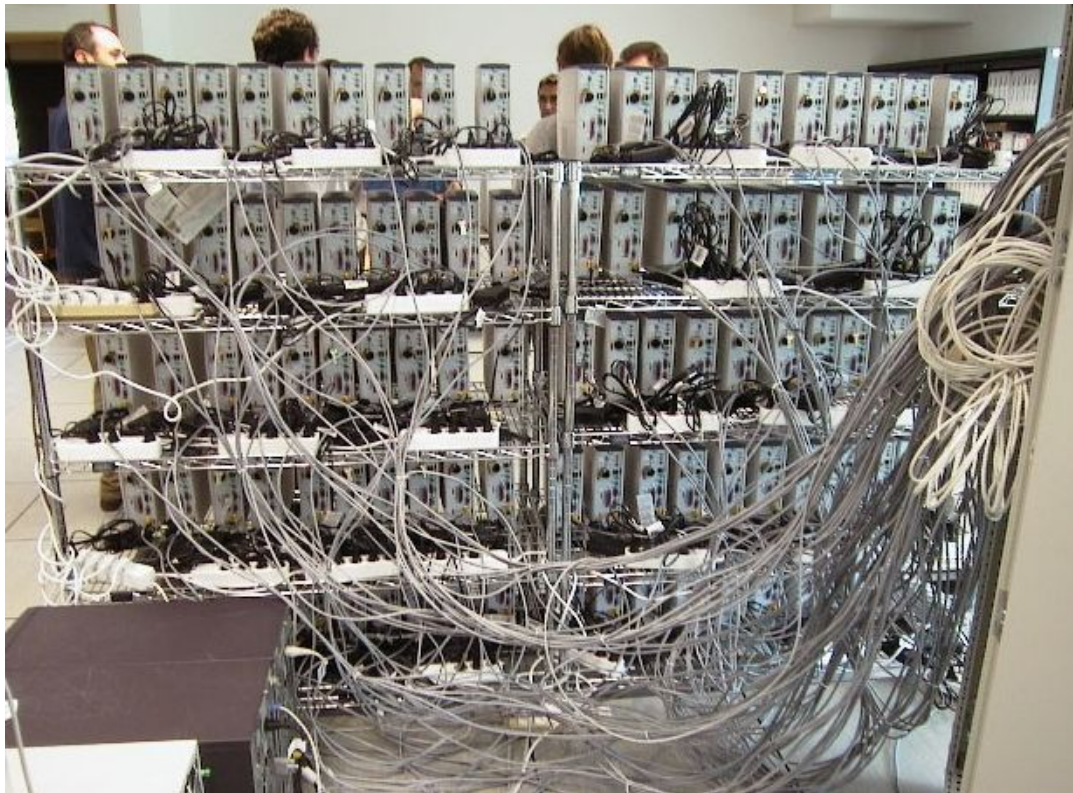
lib	size	hw	cores	time [s]	AUC
lightgbm	10M	c5.2xl	1 (m)	29	0.743
xgboost	10M	c5.2xl	1 (m)	38	0.726
xgboost	10M	i5-5200U	1 (m)	71	0.726
lightgbm	10M	r4.8xl	16 (m)	7	0.743
lightgbm	100M	r4.8xl	16 (m)	60	0.743

xgboost/lightgbm: 100 trees runtime is <10x 10 trees, RAM = 10 trees  
Spark: 100 trees runtime = 10x 10 trees, RAM > 10 trees







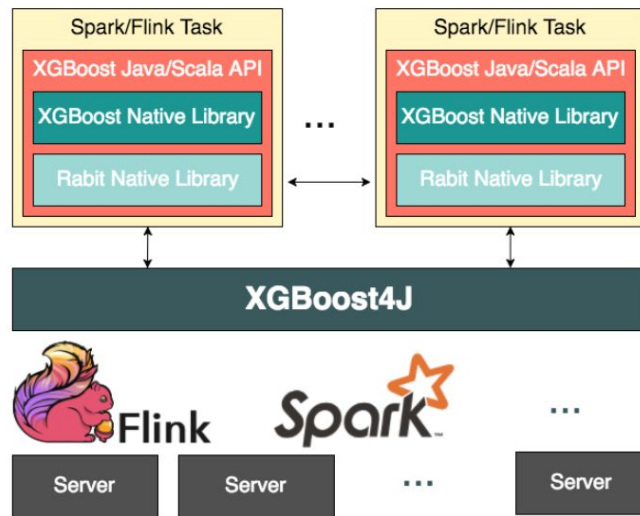


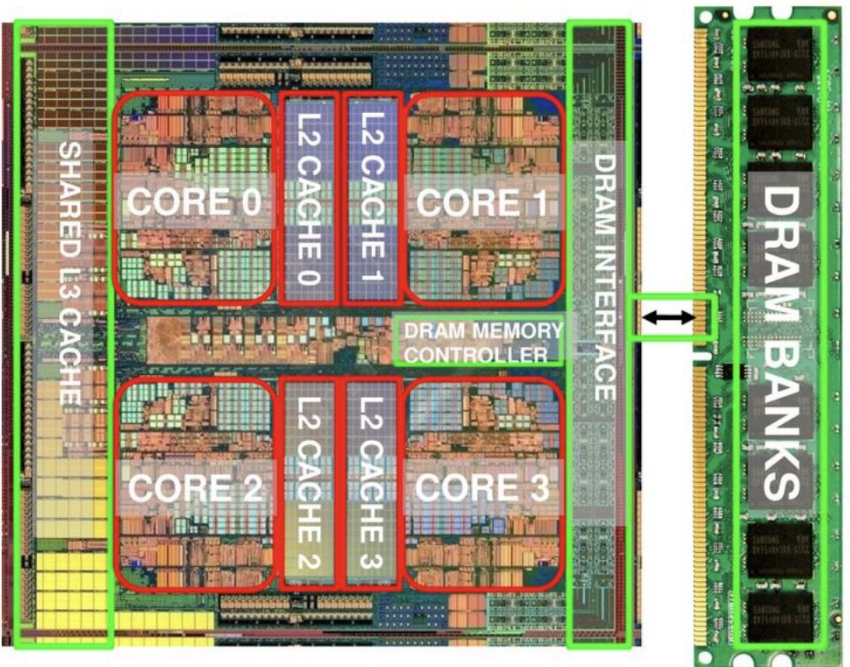


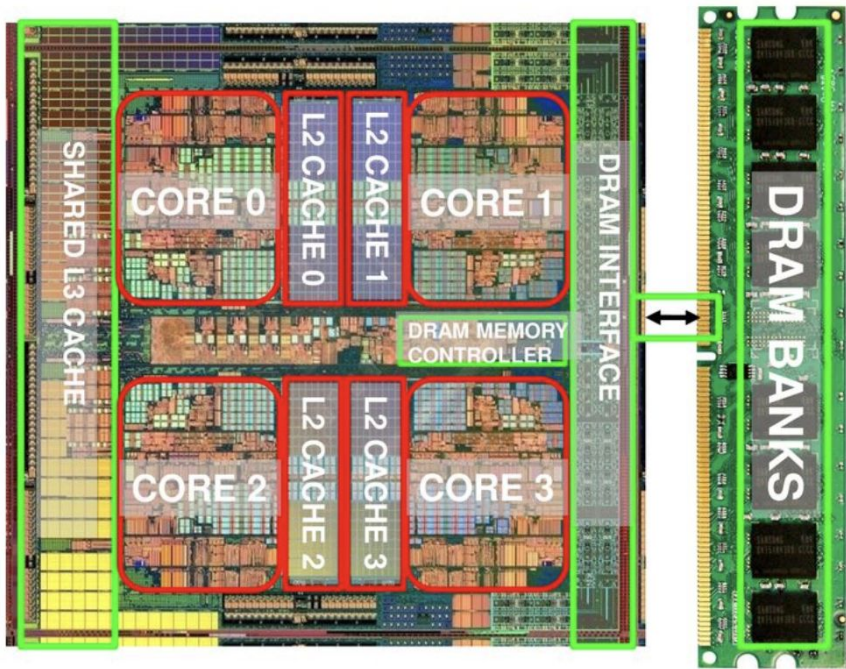


Spark + H<sub>2</sub>O

SPARKLING  
WATER







```
## read CSV (e.g. data.table::fread) or get data from database (SQL connector)
## do one-hot-encoding of categorical variables (e.g. Matrix::sparse.model.matrix)

## special optimized data structure
dxgb_train <- xgb.DMatrix(data = X_train, label = y_train)

## TRAIN
md <- xgb.train(data = dxgb_train,
                objective = "binary:logistic",
                nround = 100, max_depth = 10, eta = 0.1,
                tree_method = "hist")

## SCORE
yhat <- predict(md, newdata = X_test)

## evaluation (score distribution, ROC curve, AUC etc.)
```



```
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"))
```

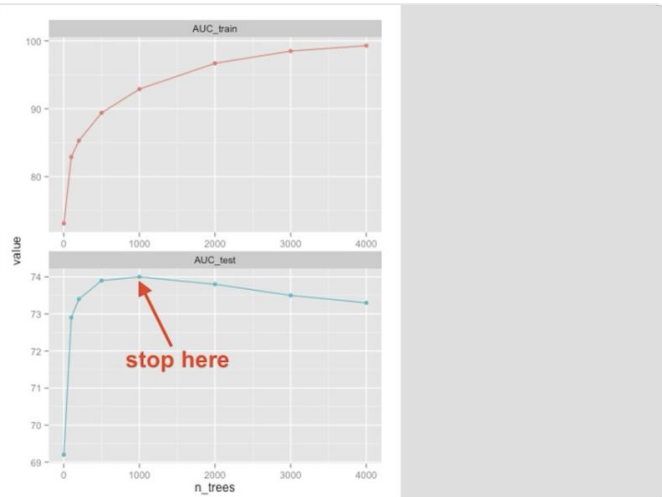


Szilard [Deeper than Deep Learning]

@DataScienceLA



GBM/GBDT tip of the day: If it's not using **early stopping**, it's crap. 🤨🤨🤨 (early stopping: checking accuracy metric on a validation set and stopping when it starts deteriorating by X) If not, you are either overfitting or wasting CPU/training time (or both). Don't do either!



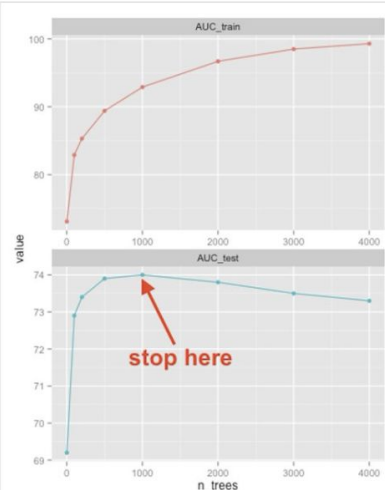




Szilard [Deeper than Deep Learning]

@DataScienceLA

GBM/GBDT tip of the day: If it's not using **early stopping**, it's crap. 😬😬😬 (early stopping: checking accuracy metric on a validation set and stopping when it starts deteriorating by X) If not, you are either overfitting or wasting CPU/training time (or both). Don't do either!



Szilard [Deeper than Deep Learning]

@DataScienceLA



in all 3 top GBM implementations (`#xgboost`, `#lightgbm`, `#h2oai`) all you have to do (for early stopping) is to set a few params in the training function

```
md <- xgb.train(data = dxgb_train,
+               objective = "binary:logistic",
+               max_depth = 10, eta = 0.1,
+               round = 10000, early_stopping_rounds = 10, watchlist = list(valid=dxgb_valid),
+               tree_method = "hist")

[1]   valid-error:0.207010
Will train until valid_error hasn't improved in 10 rounds.

[2]   valid-error:0.203230
[3]   valid-error:0.202950
[4]   valid-error:0.202810
[5]   valid-error:0.202550
[6]   valid-error:0.202380
[7]   valid-error:0.202410
[8]   valid-error:0.202380
.....
[113] valid-error:0.197940
[114] valid-error:0.197960
[115] valid-error:0.197940
[116] valid-error:0.197950
[117] valid-error:0.197820
Stopping. Best iteration:
[107] valid-error:0.197750
```

Arno Candell 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 can handle datasets of hundreds of GBs large and H2O clusters with dozens of compute nodes.

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

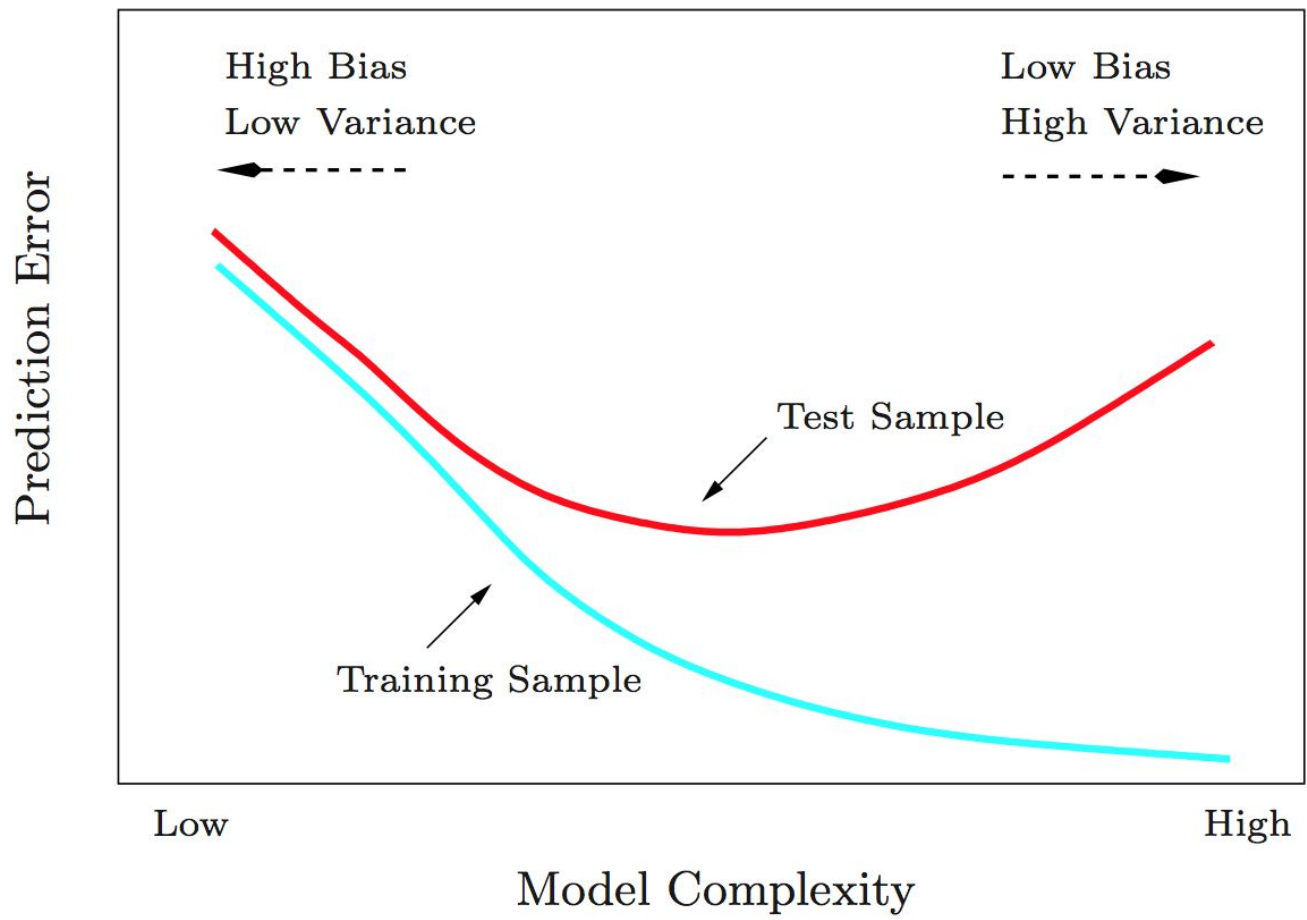


[Start Here](#)

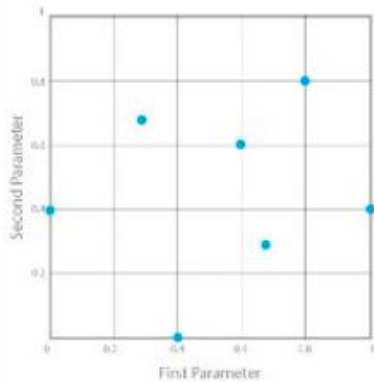
Search...

## How to Configure the Gradient Boosting Algorithm

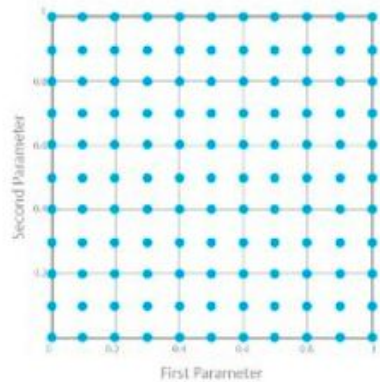
by Jason Brownlee on September 12, 2016 in XGBoost



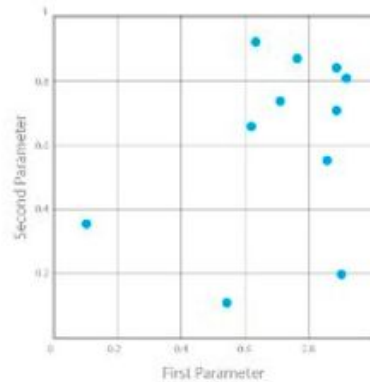
Manual Search



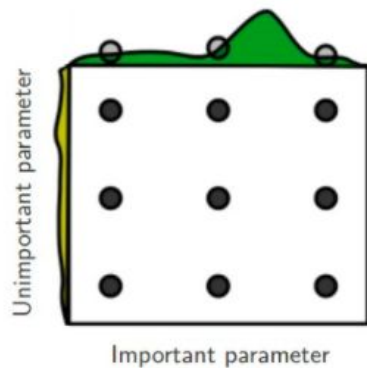
Grid Search



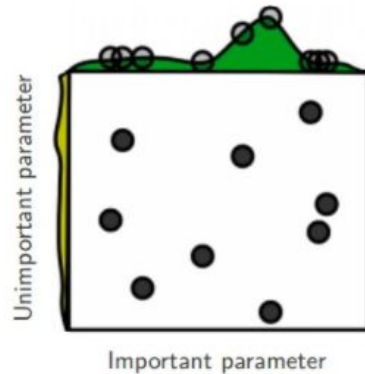
Random Search

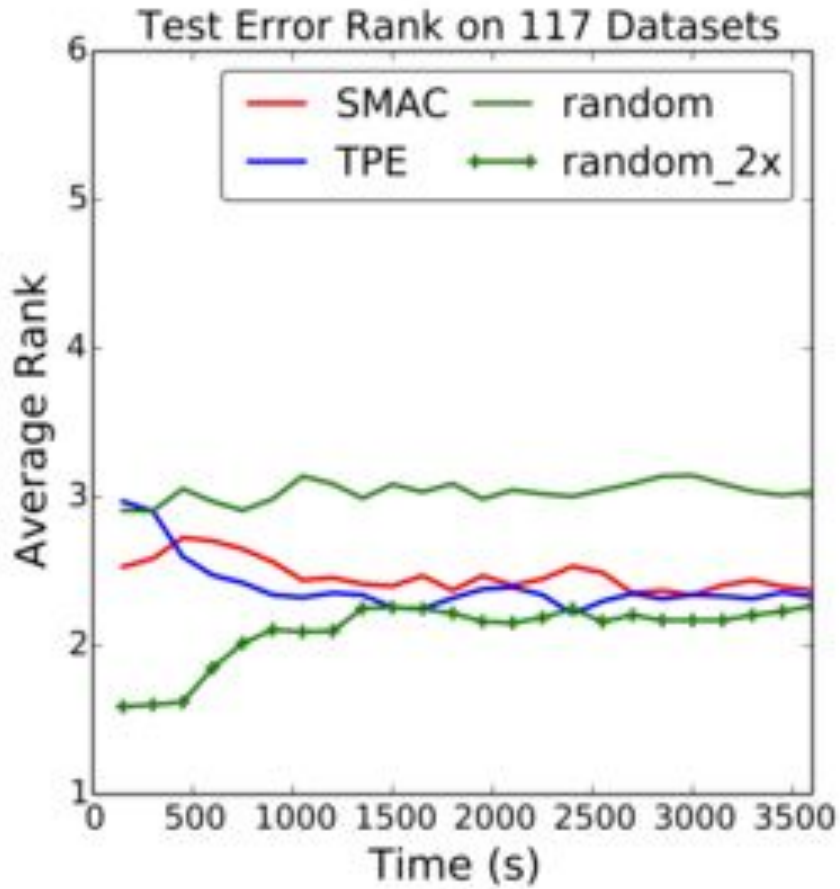


Grid Layout



Random Layout



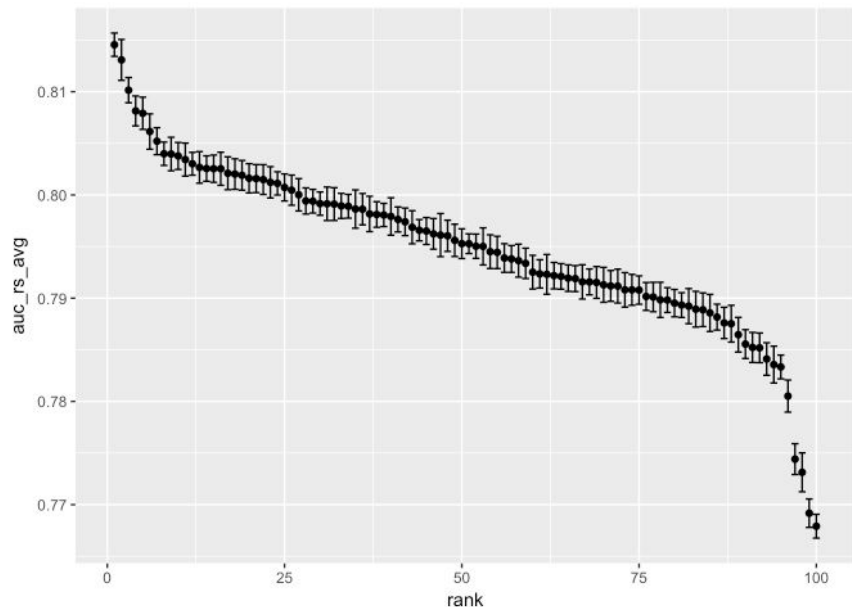


- Gaussian Processes (GP)
- Tree of Parzen Estimators (TPE)
- Sequential Model-based Algorithm Configuration (SMAC)

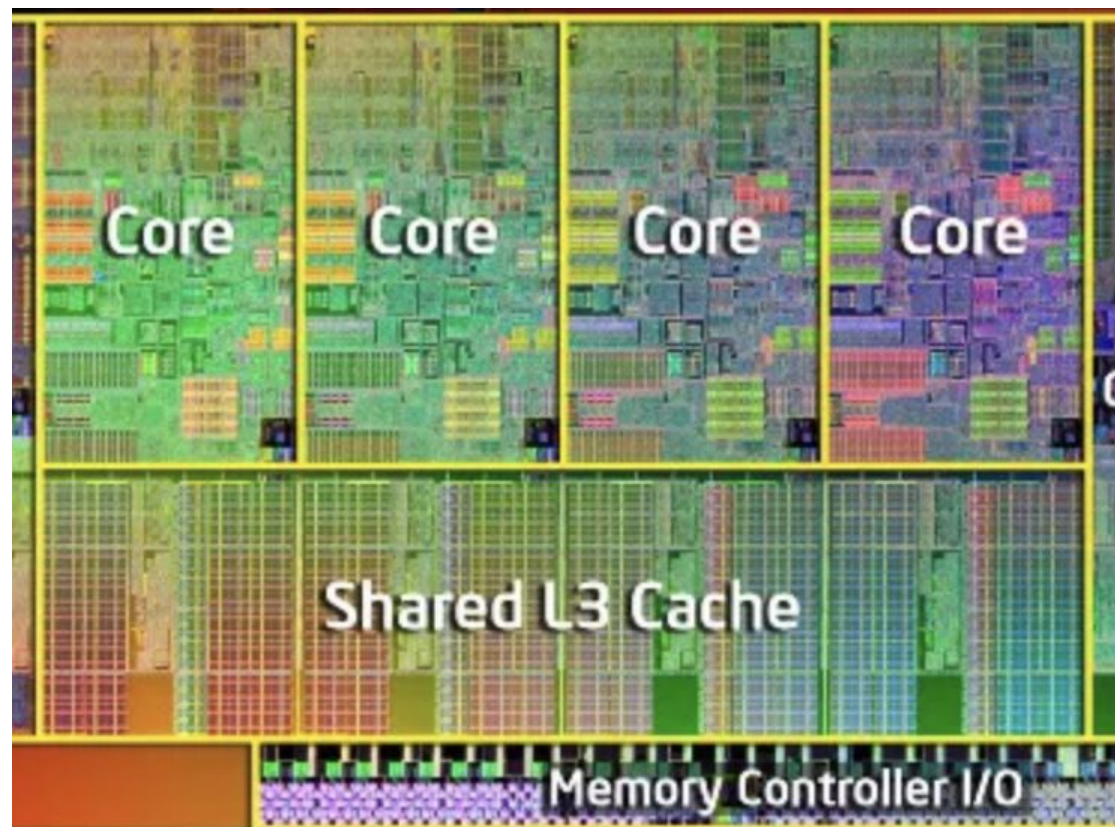
# szilard / GBM-tune

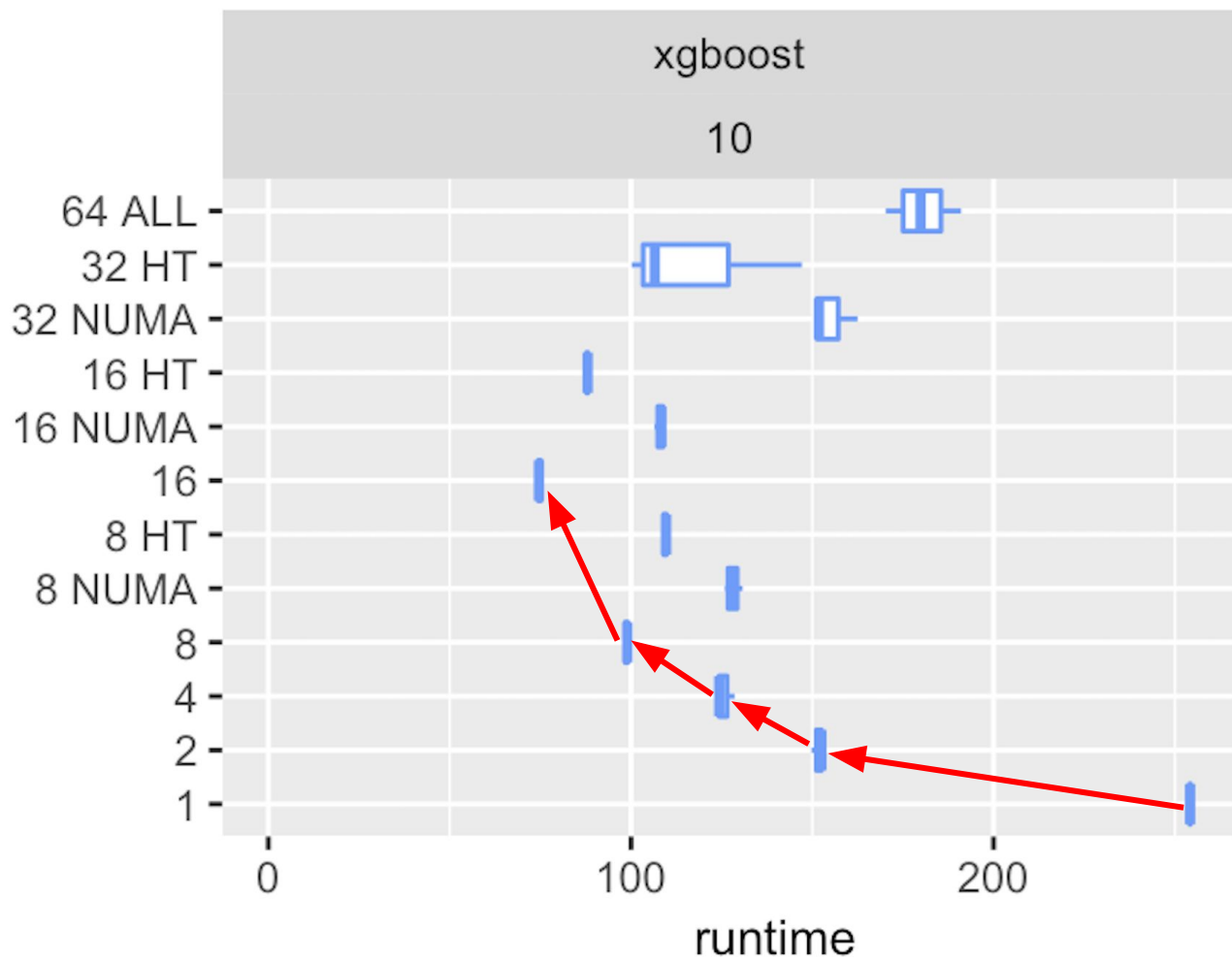
```
n_random <- 100      ## TODO: 1000?
```

```
params_grid <- expand.grid(
  num_leaves = c(100,200,500,1000,2000,5000), # default
  learning_rate = c(0.01,0.03,0.1),           # 31 (was 127)
  min_data_in_leaf = c(5,10,20,50),           # 0.1
  feature_fraction = c(0.6,0.8,1),            # 20 (was 100)
  bagging_fraction = c(0.4,0.6,0.8,1),       # 1
  lambda_l1 = c(0,0,0,0, 0.01, 0.1, 0.3),    # 1
  lambda_l2 = c(0,0,0,0, 0.01, 0.1, 0.3)     # 1
## TODO:
## min_sum_hessian_in_leaf
## min_gain_to_split
## max_bin
## min_data_in_bin
)
```

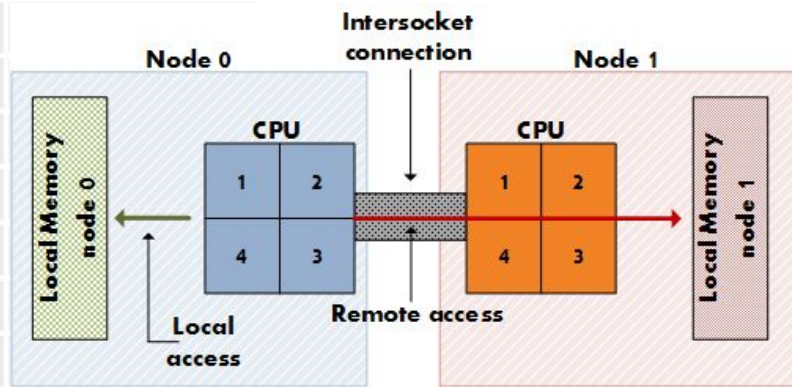
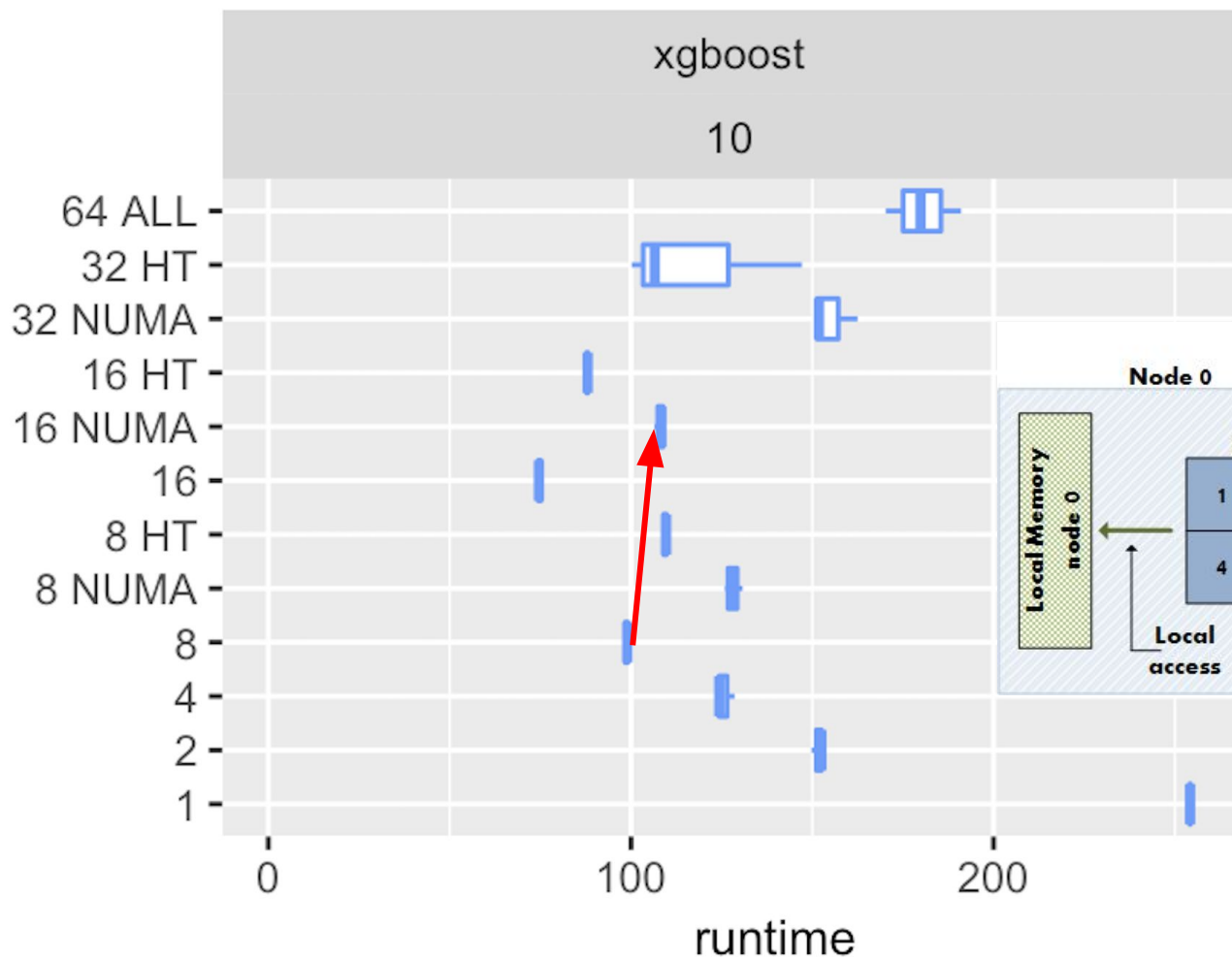


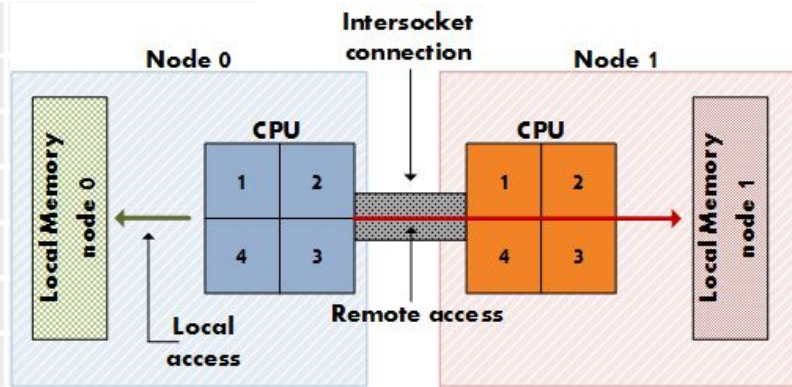
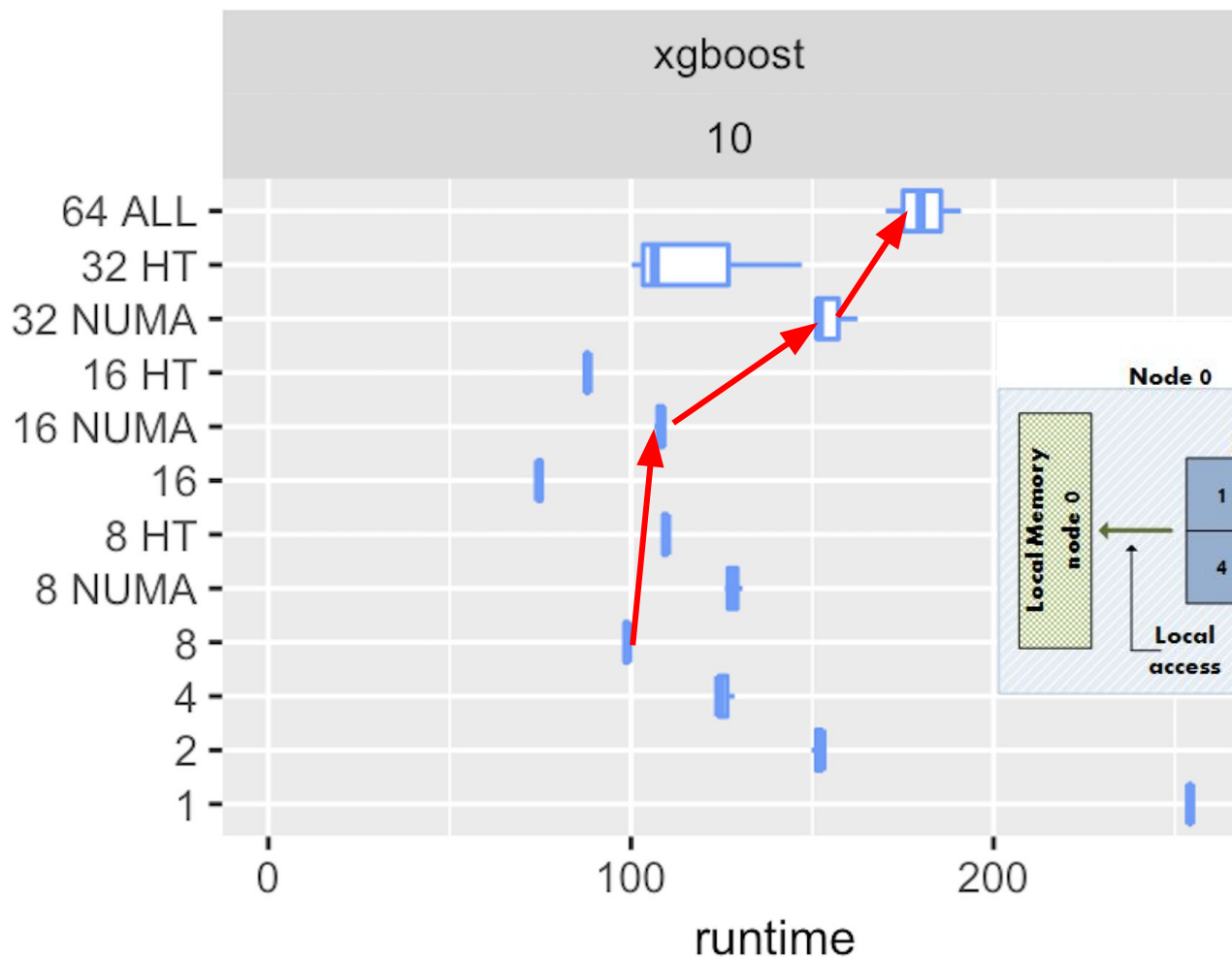
	num_leaves	learning_rate	min_data_in_leaf	feature_fraction
1	1000	0.03	5	0.8
2	1000	0.03	5	0.8
3	500	0.01	10	0.8
4	5000	0.01	5	0.8
5	500	0.01	10	0.8
6	5000	0.01	10	0.8
7	1000	0.10	20	0.8
8	5000	0.01	10	0.8
9	500	0.10	20	0.8
10	500	0.01	5	0.6

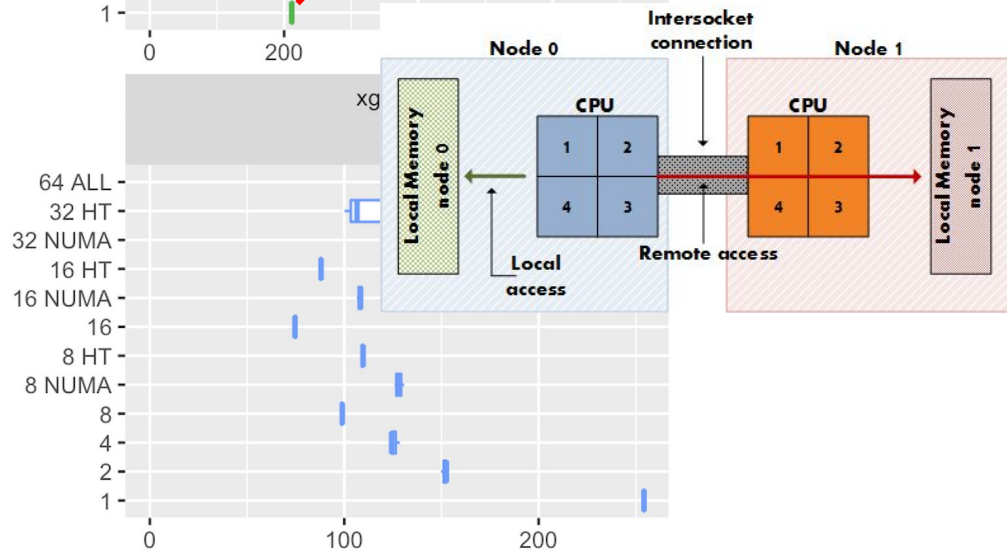
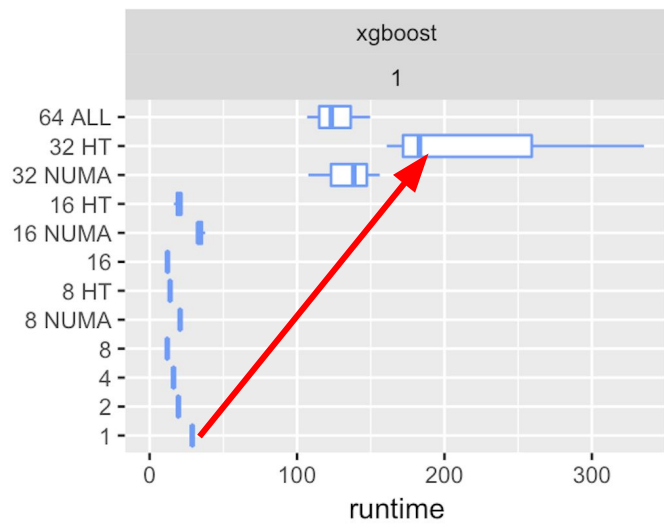
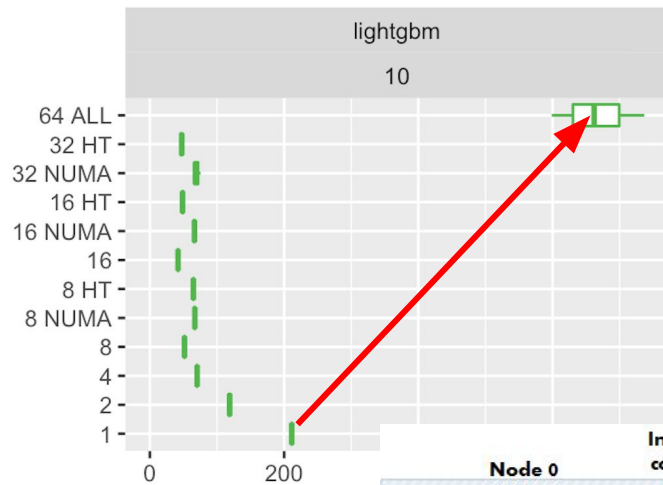
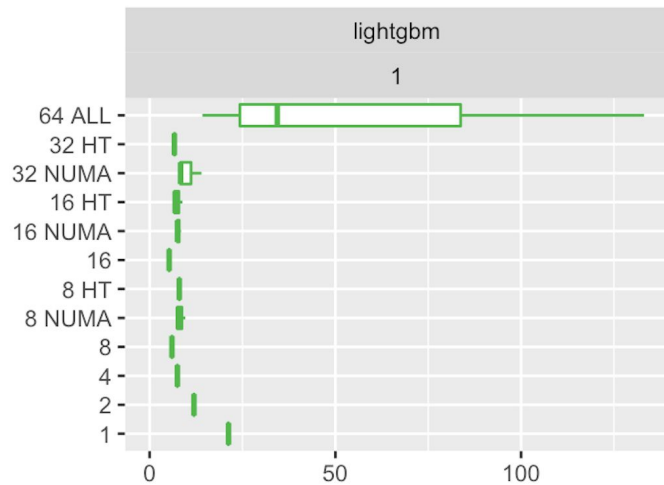


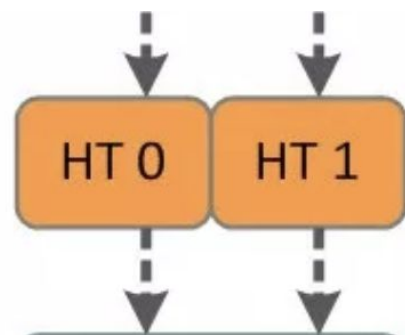
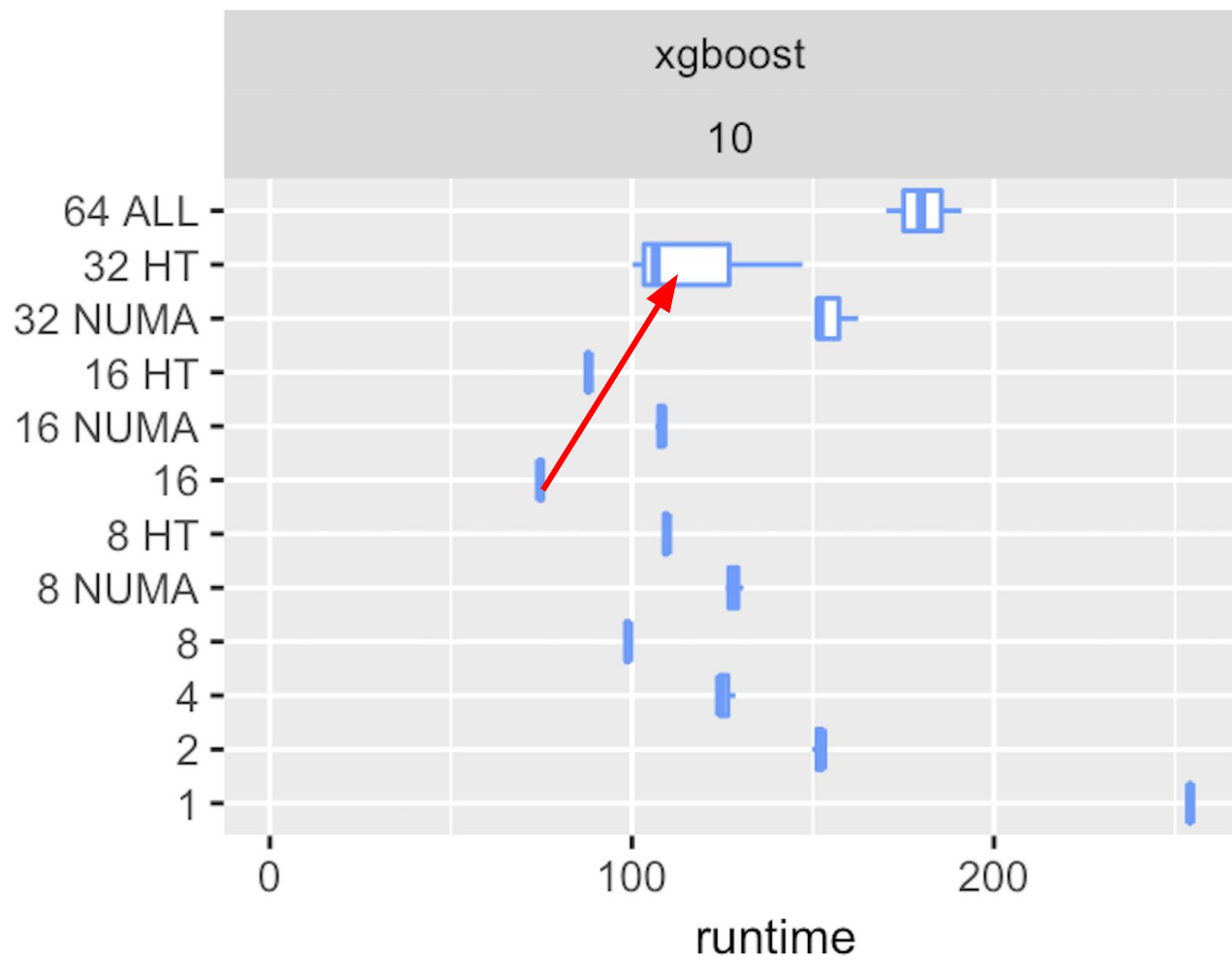












# xgboost CPU % usage patterns #1

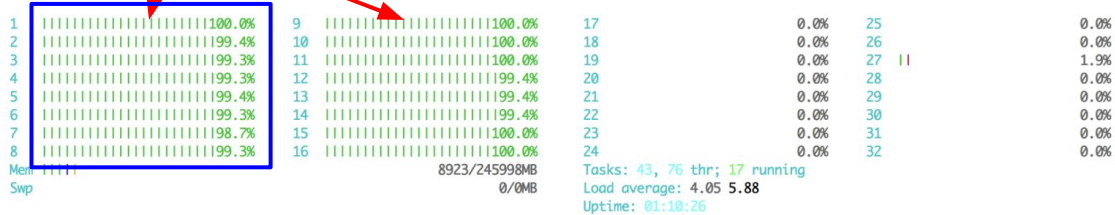
**Open** szilard opened this issue on Nov 6, 2016 · 4 comments



szilard commented on Nov 6, 2016 • edited ▾

r3.8xlarge: CPU1 0-7 (and 16-23 hyperthread pairs), CPU2 8-15

```
taskset -c 0-7 Rscript xgb.R 8 &
taskset -c 8-15 Rscript xgb.R 8
```



# xgboost CPU % usage patterns #1

**Open** szilard opened this issue on Nov 6, 2016 · 4 comments



szilard commented on Nov 6, 2016 • edited ▾

r3.8xlarge: CPU1 0-7 (and 16-23 hyperthread pairs), CPU2 8-15

szilard commented on Nov 6, 2016 • edited ▾

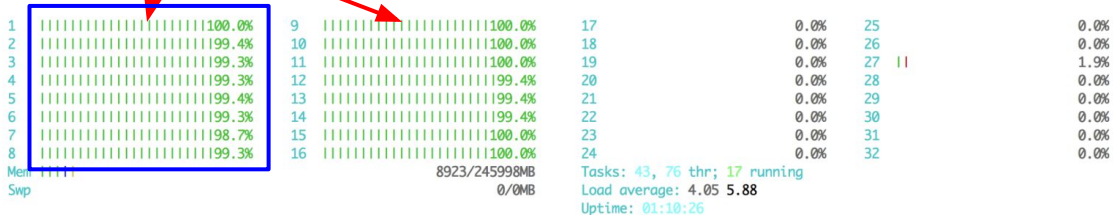
CV/hyperparam search:

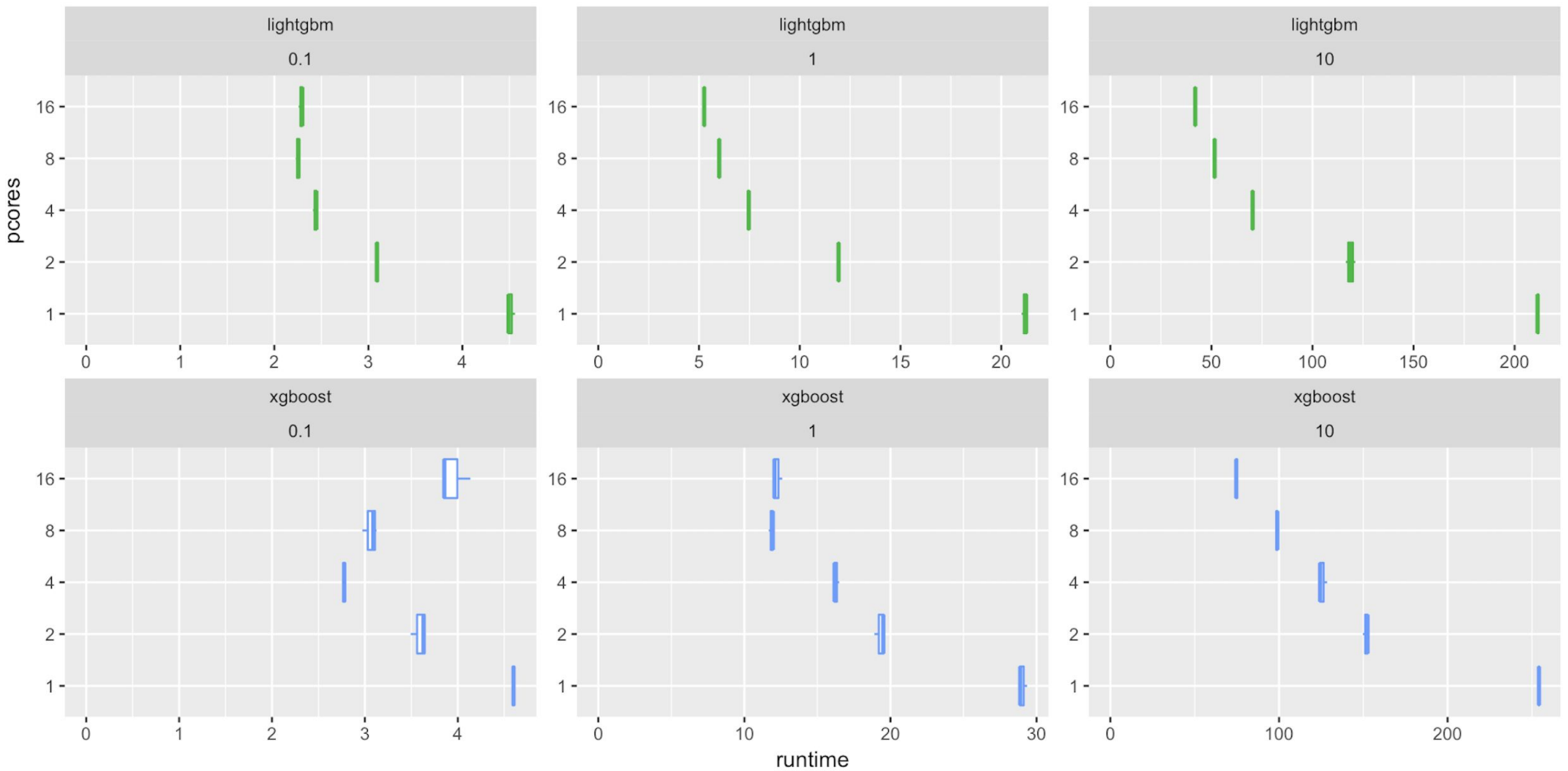
Option 1: run sequentially on 32 cores, takes 64sec/fold

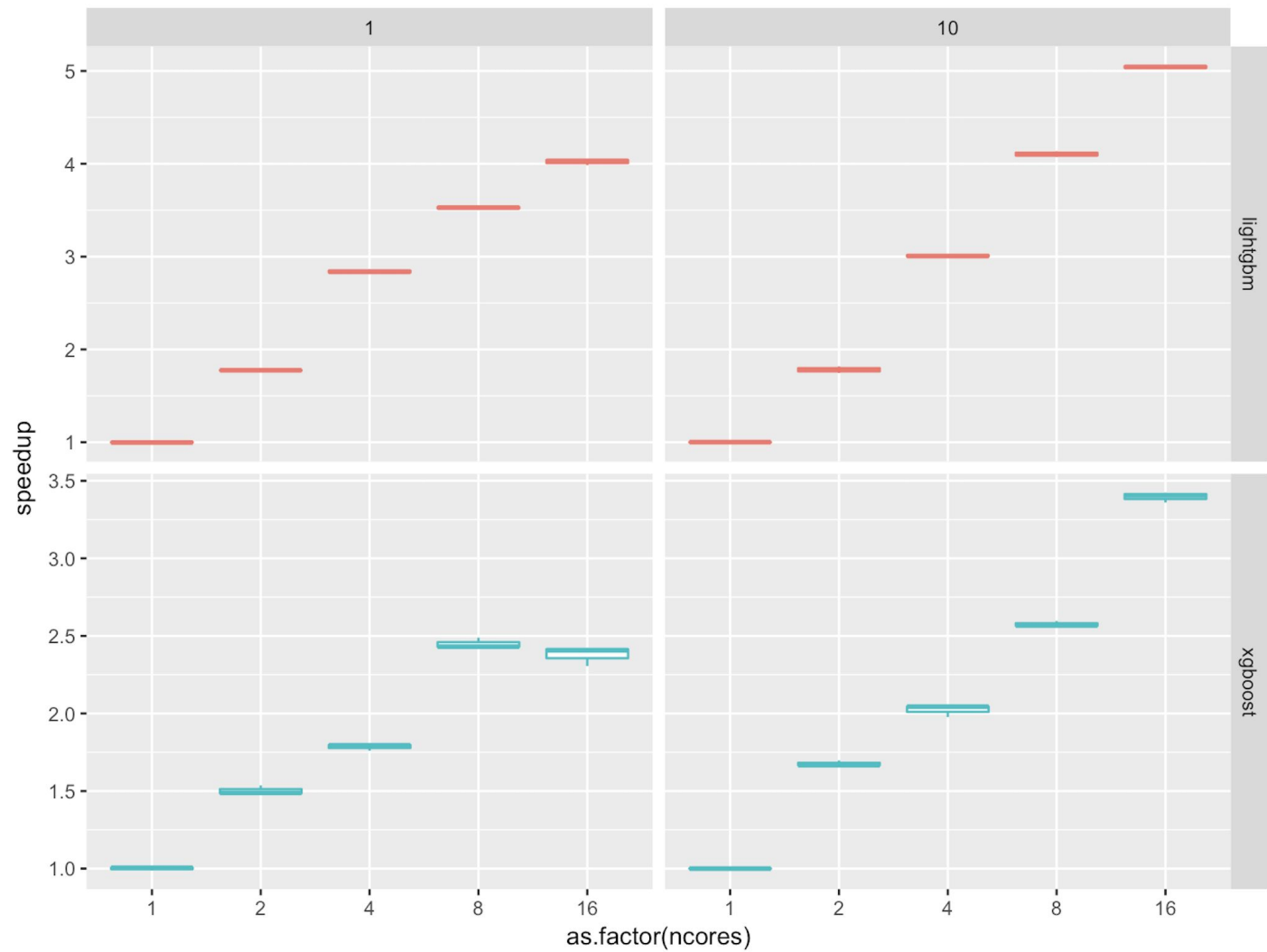
Option2: run 2 folds at a time pinned to cores 0-7 and 8-15, takes 35sec

So Opt2 is 3.6x faster than Opt1.

```
taskset -c 0-7 Rscript xgb.R 8 &
taskset -c 8-15 Rscript xgb.R 8
```









 **Laurae2 / ml-perf**



szilard commented 7 hours ago • edited ▾

I will include this (simplified results for xgboost CPU) in my talks (with credit to @Laurae2):

Parallel Threads	Model Threads	Models	Seconds / Model
1	1	25	11.39
9	1	50	1.46
18	1	100	0.78
35	1	250	0.49
70	1	500	0.43
<hr/>			
1	1	50	11.4
1	9	50	6.6
1	18	50	6.6
1	35	50	25
1	70	50	165

(for easy ref: 2 socket system with 18+18HT cores each socket, total 72 cores; 0.1m dataset; 500 trees, depth 6, learn rate 0.05)

 [Laurae2 / ml-perf](#)



**Szilard [Deeper than Deep Learning]**

@DataScienceLA

Replying to @rquintino

If you have lots of models to train (or hyperparam values to try etc.) and fixed hardware resources the best (least wall time/most throughput) is actually running 1 model/CPU core (quite unsexy, huh?) see here [github.com/szilard/GBM-pe...](https://github.com/szilard/GBM-pe...) (joint work but mostly by @Laurae\_Cht )



szilard commented 7 hours ago • edited ▾

I will include this (simplified results for xgboost CPU) in my talks (with credit to @Laurae2):

Parallel Threads	Model Threads	Models	Seconds / Model
1	1	25	11.39
9	1	50	1.46
18	1	100	0.78
35	1	250	0.49
70	1	500	0.43
<hr/>			
1	1	50	11.4
1	9	50	6.6
1	18	50	6.6
1	35	50	25
1	70	50	165

(for easy ref: 2 socket system with 18+18HT cores each socket, total 72 cores; 0.1m dataset; 500 trees, depth 6, learn rate 0.05)

	<b>gbm (R pkg)</b>	<b>xgboost</b>	<b>lightgbm</b>	<b>h2o</b>
easy R install	cran	cran	linux OK	java+cran
maintained	retired	yes	yes	yes
preprocessing	not needed	1-hot	1-hot/categ int	not needed
new cats scoring	yes	no	no	yes
early stopping	no	yes	yes	yes
speed (CPU)	1 core	ok	fastest	slow (small data)
GPU supported	no	yes	yes	via xgboost
speed GPU	NA	fastest	ok/slow	indirectly/slower
REST scoring	no	no	no	yes
other algos	no	RF	RF	RF/GLM/NN
best for	teaching/historic	Kaggle	Kaggle	prod/real-time

## Recommendations

---

If you **don't have a GPU**, **lightgbm** (CPU) trains the fastest.

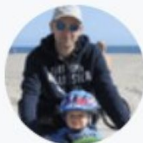
If you **have a GPU**, **xgboost** (GPU) is also very fast (and depending on the data, your hardware etc. often faster than the above mentioned lightgbm on CPU).

If you consider deployment, **h2o has the best ways to deploy** as a real-time (fast scoring) application.

Note, however, there are a lot more other criteria to consider when you choose which tool to use.



szilard / **GBM-perf**



**Szilard** @DataScienceLA · Apr 23

What algo do you use the most for supervised learning?

33% lin/logistic regression

56% random forest/GBM

4% shallow neural nets

7% deep learning

57 votes · Final results



**Szilard** @DataScienceLA · Apr 23

What algo do you use the most for supervised learning?

33% lin/logistic regression

56% random forest/GBM \*

4% shallow neural nets

7% deep learning

57 votes · Final results



**Szilard**

@DataScienceLA

Friday fun: what's your favorite gradient boosting machine (GBM) library?

58% xgboost

16% lightgbm

24% h2o

2% spark mllib

127 votes • Final results

3:21 PM - 11 May 2018



**Szilard**

@DataScienceLA

Friday fun: what's your favorite gradient boosting machine (GBM) library?

58% xgboost

16% lightgbm

24% h2o

2% spark mllib



**no-one is using this crap**

127 votes • Final results

3:21 PM - 11 May 2018



**Szilard** @DataScienceLA · May 16 (2018)

If you are using gradient boosting machines(GBM), are you running it (training) on GPUs or CPUs?

7% GPU

93% CPU

A horizontal bar chart with two bars. The top bar is light gray and represents '7% GPU'. The bottom bar is blue and represents '93% CPU'. The bars are positioned to the left of their respective percentage labels.

55 votes · Final results

**Szilard** @DataScienceLA · May 16 (2018)

If you are using gradient boosting machines(GBM), are you running it (training) on GPUs or CPUs?

7% GPU

93% CPU

55 votes · Final results



**Szilard [Deeper than Deep Learning]**

@DataScienceLA

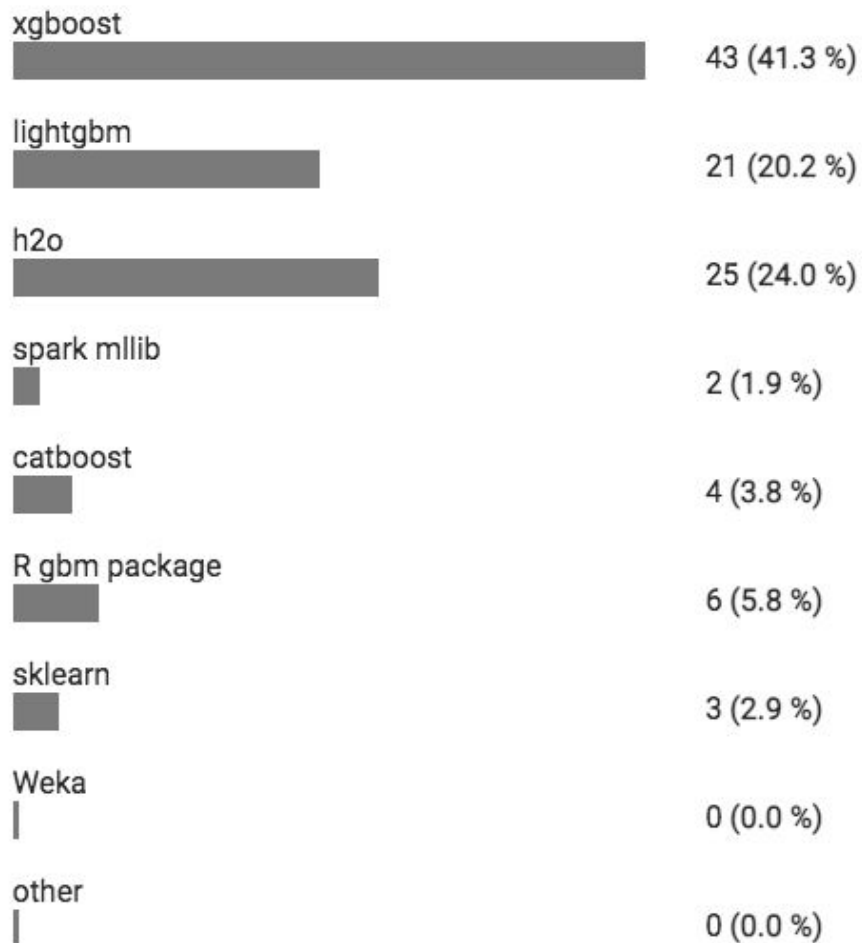
If you are using gradient boosting machines (GBM)/boosted trees (GBDT) are you using (training) them most often on the CPU or a GPU? [#xgboost](#) [#lightgbm](#) [#h2oai](#) [#catboost](#) [#apachespark](#) [#mllib](#) [#sklearn](#)

86% CPU

14% GPU

69 votes · Final results

11:39 AM - 4 May 2019



104 people have already voted  
You have already answered this poll

[i show the configuration of this Ferendum](#)

# Winning Solution of KaggleDays 2019 Competition in San Francisco

# Winning Solution of KaggleDays 2019 Competition in San Francisco

During the first two to three hours of the competition, we focused on exploratory data analysis (EDA). We first analyzed the distributions of raw features between train and test

# Winning Solution of KaggleDays 2019 Competition in San Francisco

During the first two to three hours of the competition, we focused on exploratory data analysis (EDA). We first analyzed the distributions of raw features between train and test

For model training, LightGBM is chosen as the baseline model for doing quick experiments on raw and engineered features. It turned out that our engineered features improved AUC in

# Winning Solution of KaggleDays 2019 Competition in San Francisco

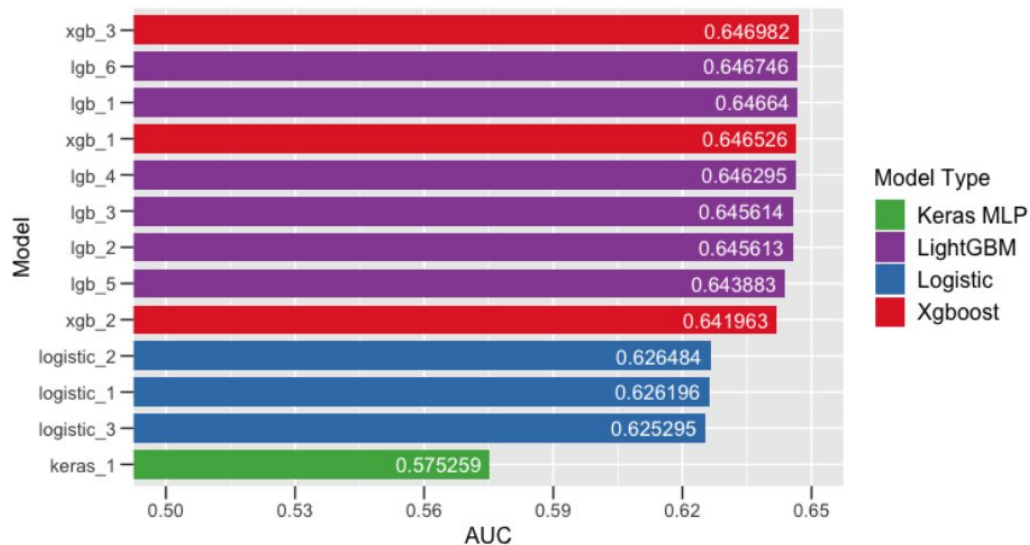
During the first two to three hours of the competition, we focused on exploratory data analysis (EDA). We first analyzed the distributions of raw features between train and test

For model training, LightGBM is chosen as the baseline model for doing quick experiments on raw and engineered features. It turned out that our engineered features improved AUC in

Based on the LightGBM baseline notebook, I created other notebooks for Xgboost, Logistic Regression, Random Forests and Extra Trees models. To save some time, I ran Random

# Winning Solution of KaggleDays 2019 Competition in San Francisco

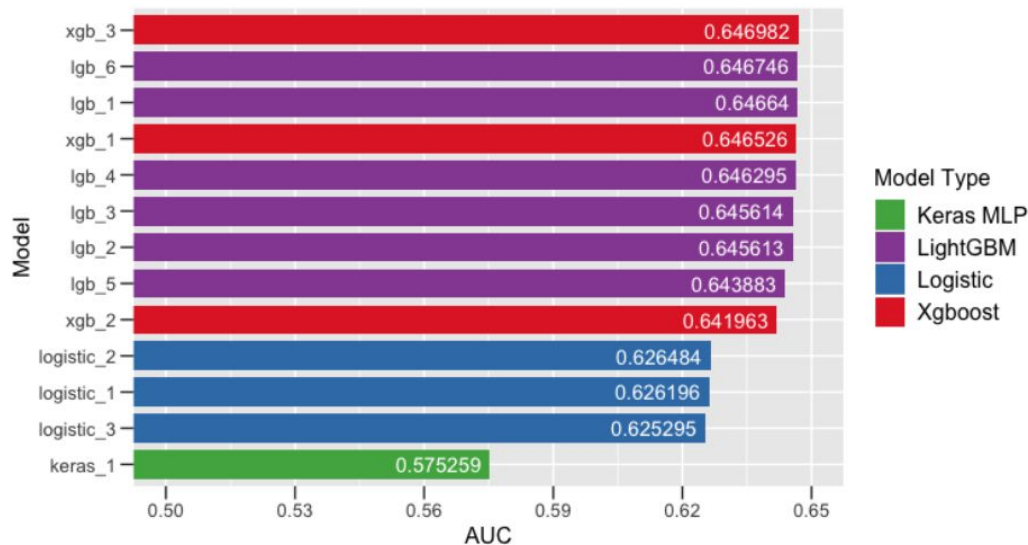
During the first two to three hours of the competition, we focused on exploratory data analysis (EDA). We first analyzed the distributions of raw features between train and test





# Winning Solution of KaggleDays 2019 Competition in San Francisco

During the first two to three hours of the competition, we focused on exploratory data analysis (EDA). We first analyzed the distributions of raw features between train and test



For blending and stacking, we looked for diverse models that perform relatively well, but are not highly correlated with each other. Xgboost and LightGBM had very similar

**TABULAR DATA YOU HAVE**



**GBM USE YOU MUST**

# More:



szilard / **benchm-ml**

★ Star

1,203



szilard / **teach-data-science-UCLA-master-appl-stats**



szilard / **teach-ML-CEU-master-bizanalytics**



szilard / **GBM-perf**



szilard / **ML-scoring**



szilard / **GBM-tune**



szilard / **GBM-multicore**



szilard / **GBM-workshop**

GitHubGist

Search...



szilard / **h2o\_scoring.R**



szilard / **ML\_with\_H2O.R**



✉ [spafka@gmail.com](mailto:spafka@gmail.com)

🐦 [@DataScienceLA](https://twitter.com/DataScienceLA)

in [linkedin.com/in/szilard](https://www.linkedin.com/in/szilard)

🐙 [github.com/szilard](https://github.com/szilard)