No-Bullshit Data Science

Szilárd Pafka, PhD
Chief Scientist, Epoch

R/Finance Conference
Chicago, May 2017

Szilard

@ DataScienceLA

Santa Monica, California

physics PhD, chief (data) scientist, meetup organizer, datascience.la, visiting professor

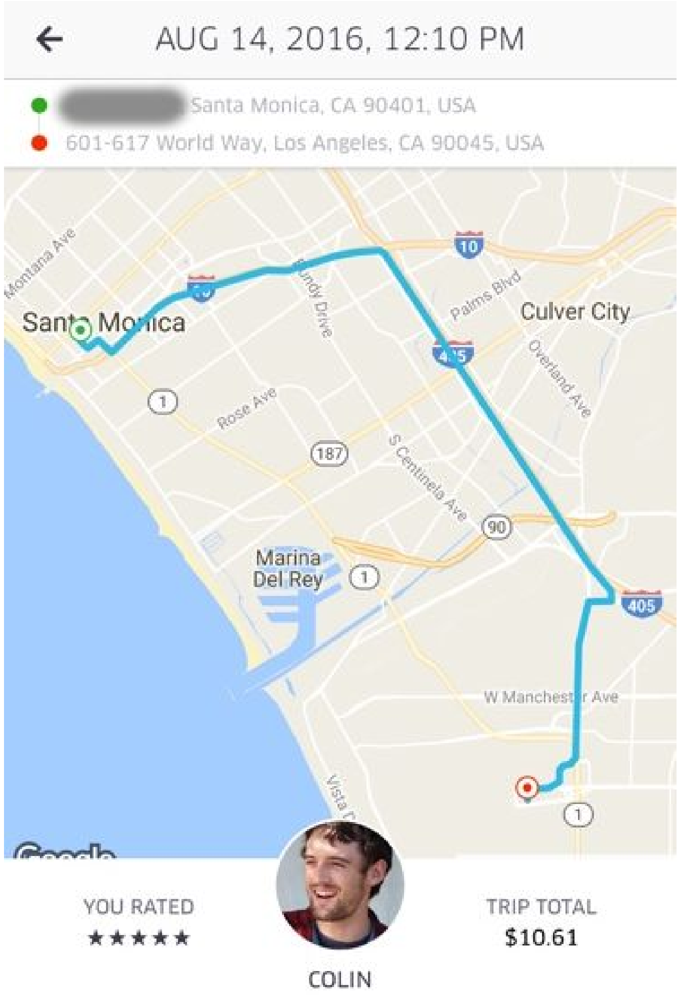
Disclaimer:

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


PLANET HYPE


PLANET HYPE

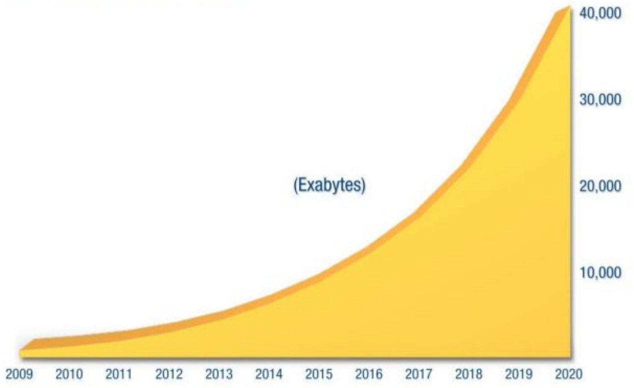

AUG 14, 2016, 12:10 PM
Santa Monica, CA 90401, USA
601-617 World Way, Los Angeles, CA 90045, USA
Montana Ave
Santa Monica
Bundy Drive
Palms Blvd
Culver City
10
405
Overland Ave
Rose Ave
1
187
S Centinela Ave
90
Marina Del Rey
W Manchester Ave
YOU RATED
★★★★★
COLIN
TRIP TOTAL
$10.61


REALITY
CHECK
AHEAD

Example #1

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

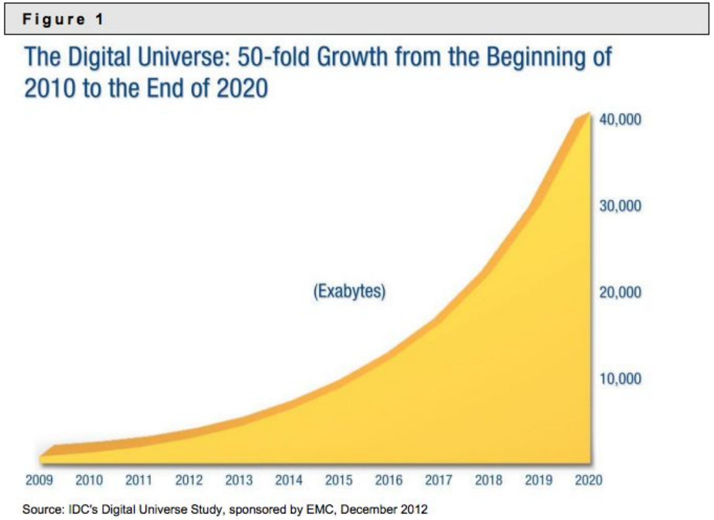
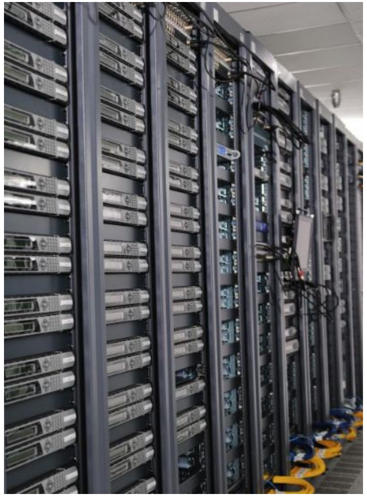
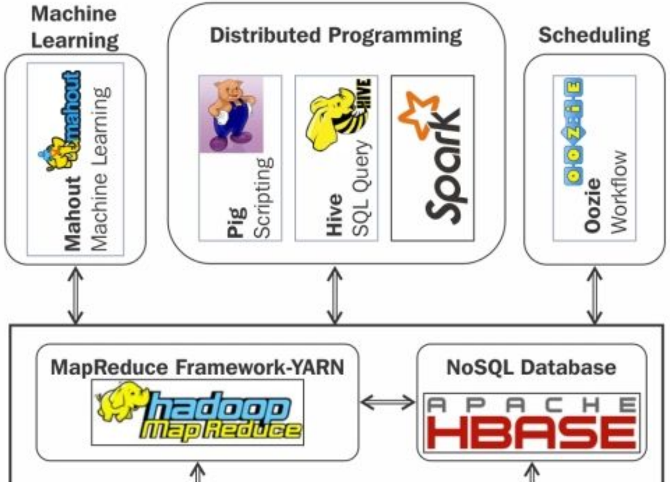


Figure 1
The Digital Universe: 50-fold Growth from the Beginning of 2010 to the End of 2020
40,000
30,000
20,000
10,000
(Exabytes)
2009 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019 2020
Source: IDC's Digital Universe Study, sponsored by EMC, December 2012




"BIG" DATA
imgflip.com


Machine Learning
Mahout
Machine Learning
Distributed Programming
Pig
Scripting
Hive
SQL Query
Spark
Scheduling
Oozie
Workflow
MapReduce Framework-YARN
hadoop
Map Reduce
NoSQL Database
APACHE
HBASE

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 997 LIKES 960

3:03 PM - 19 May 2015

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

x1.32xlarge	128	349	1952	2 x 1920 SSD	$13.338 per Hour

```
  1  [||||||||||||||||||||||||89.6%]   33 [||||||||||||||||||||||||88.5%]   65 [||||||||||||||||||||||||82.8%]   97 [||||||||||||||||||||||||88.5%]
  2  [||||||||||||||||||||||||85.8%]   34 [||||||||||||||||||||||||88.5%]   66 [||||||||||||||||||||||||85.3%]   98 [||||||||||||||||||||||||89.7%]
  3  [||||||||||||||||||||||||91.1%]   35 [||||||||||||||||||||||||84.0%]   67 [||||||||||||||||||||||||86.5%]   99 [||||||||||||||||||||||||86.5%]
  4  [||||||||||||||||||||||||85.3%]   36 [||||||||||||||||||||||||85.9%]   68 [||||||||||||||||||||||||85.4%]   100[||||||||||||||||||||||||88.5%]
  5  [||||||||||||||||||||||||84.6%]   37 [||||||||||||||||||||||||89.2%]   69 [||||||||||||||||||||||||85.4%]   101[||||||||||||||||||||||||84.7%]
  6  [||||||||||||||||||||||||84.6%]   38 [||||||||||||||||||||||||89.1%]   70 [||||||||||||||||||||||||86.5%]   102[||||||||||||||||||||||||88.5%]
  7  [||||||||||||||||||||||||84.1%]   39 [||||||||||||||||||||||||87.3%]   71 [||||||||||||||||||||||||86.5%]   103[||||||||||||||||||||||||89.2%]
  8  [||||||||||||||||||||||||85.2%]   40 [||||||||||||||||||||||||87.9%]   72 [||||||||||||||||||||||||87.9%]   104[||||||||||||||||||||||||86.5%]
  9  [||||||||||||||||||||||||84.7%]   41 [||||||||||||||||||||||||83.4%]   73 [||||||||||||||||||||||||89.1%]   105[||||||||||||||||||||||||87.1%]
  10 [||||||||||||||||||||||||85.4%]   42 [||||||||||||||||||||||||85.4%]   74 [||||||||||||||||||||||||84.7%]   106[||||||||||||||||||||||||84.6%]
  11 [||||||||||||||||||||||||85.3%]   43 [||||||||||||||||||||||||85.8%]   75 [||||||||||||||||||||||||86.0%]   107[||||||||||||||||||||||||84.6%]
  12 [||||||||||||||||||||||||86.0%]   44 [||||||||||||||||||||||||91.7%]   76 [||||||||||||||||||||||||86.5%]   108[||||||||||||||||||||||||90.4%]
  13 [||||||||||||||||||||||||85.9%]   45 [||||||||||||||||||||||||89.2%]   77 [||||||||||||||||||||||||85.4%]   109[||||||||||||||||||||||||90.4%]
  14 [||||||||||||||||||||||||83.3%]   46 [||||||||||||||||||||||||87.2%]   78 [||||||||||||||||||||||||86.0%]   110[||||||||||||||||||||||||87.8%]
  15 [||||||||||||||||||||||||84.6%]   47 [||||||||||||||||||||||||87.1%]   79 [||||||||||||||||||||||||86.6%]   111[||||||||||||||||||||||||85.9%]
  16 [||||||||||||||||||||||||85.4%]   48 [||||||||||||||||||||||||87.1%]   80 [||||||||||||||||||||||||86.7%]   112[||||||||||||||||||||||||87.8%]
  17 [||||||||||||||||||||||||83.3%]   49 [||||||||||||||||||||||||89.7%]   81 [||||||||||||||||||||||||85.8%]   113[||||||||||||||||||||||||88.5%]
  18 [||||||||||||||||||||||||85.4%]   50 [||||||||||||||||||||||||94.3%]   82 [||||||||||||||||||||||||86.6%]   114[||||||||||||||||||||||||91.1%]
  19 [||||||||||||||||||||||||82.7%]   51 [||||||||||||||||||||||||91.7%]   83 [||||||||||||||||||||||||89.1%]   115[||||||||||||||||||||||||91.1%]
  20 [||||||||||||||||||||||||81.5%]   52 [||||||||||||||||||||||||96.2%]   84 [||||||||||||||||||||||||85.9%]   116[||||||||||||||||||||||||90.4%]
  21 [||||||||||||||||||||||||81.9%]   53 [||||||||||||||||||||||||91.0%]   85 [||||||||||||||||||||||||85.3%]   117[||||||||||||||||||||||||89.2%]
  22 [||||||||||||||||||||||||84.1%]   54 [||||||||||||||||||||||||90.4%]   86 [||||||||||||||||||||||||85.9%]   118[||||||||||||||||||||||||92.3%]
  23 [||||||||||||||||||||||||83.2%]   55 [||||||||||||||||||||||||90.5%]   87 [||||||||||||||||||||||||87.2%]   119[||||||||||||||||||||||||93.6%]
  24 [||||||||||||||||||||||||85.4%]   56 [||||||||||||||||||||||||89.1%]   88 [||||||||||||||||||||||||85.9%]   120[||||||||||||||||||||||||91.7%]
  25 [||||||||||||||||||||||||86.0%]   57 [||||||||||||||||||||||||88.5%]   89 [||||||||||||||||||||||||88.5%]   121[||||||||||||||||||||||||91.1%]
  26 [||||||||||||||||||||||||87.3%]   58 [||||||||||||||||||||||||91.7%]   90 [||||||||||||||||||||||||84.2%]   122[||||||||||||||||||||||||91.1%]
  27 [||||||||||||||||||||||||83.3%]   59 [||||||||||||||||||||||||96.2%]   91 [||||||||||||||||||||||||84.8%]   123[||||||||||||||||||||||||89.8%]
  28 [||||||||||||||||||||||||84.0%]   60 [||||||||||||||||||||||||91.7%]   92 [||||||||||||||||||||||||84.6%]   124[||||||||||||||||||||||||90.4%]
  29 [||||||||||||||||||||||||87.3%]   61 [||||||||||||||||||||||||88.5%]   93 [||||||||||||||||||||||||86.5%]   125[||||||||||||||||||||||||93.6%]
  30 [||||||||||||||||||||||||84.6%]   62 [||||||||||||||||||||||||90.4%]   94 [||||||||||||||||||||||||87.3%]   126[||||||||||||||||||||||||92.9%]
  31 [||||||||||||||||||||||||86.5%]   63 [||||||||||||||||||||||||92.3%]   95 [||||||||||||||||||||||||86.6%]   127[||||||||||||||||||||||||90.4%]
  32 [||||||||||||||||||||||||86.5%]   64 [||||||||||||||||||||||||94.2%]   96 [||||||||||||||||||||||||86.5%]   128[||||||||||||||||||||||||91.0%]
  Mem[|||||||||||||||||||||||||||||||||||||||           1062263/1967522MB]   Tasks: 43, 190 thr, 1075 kthr; 129 running
  Swp[                                                            0/0MB]   Load average: 58.14 21.01 11.40
                                                                           Uptime: 18:32:29

    PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
  34941 ubuntu     20   0 1025G 1024G  1380 S 11195 53.3  1h54:07 ./pmbw -f ScanRead64PtrSimpleLoop -p 128 -P 128
  34999 ubuntu     20   0 1025G 1024G  1380 R 95.3 53.3  0:50.06 ./pmbw -f ScanRead64PtrSimpleLoop -p 128 -P 128
  35066 ubuntu     20   0 1025G 1024G  1380 R 93.4 53.3  0:50.02 ./pmbw -f ScanRead64PtrSimpleLoop -p 128 -P 128
  34963 ubuntu     20   0 1025G 1024G  1380 R 92.7 53.3  0:51.13 ./pmbw -f ScanRead64PtrSimpleLoop -p 128 -P 128
```

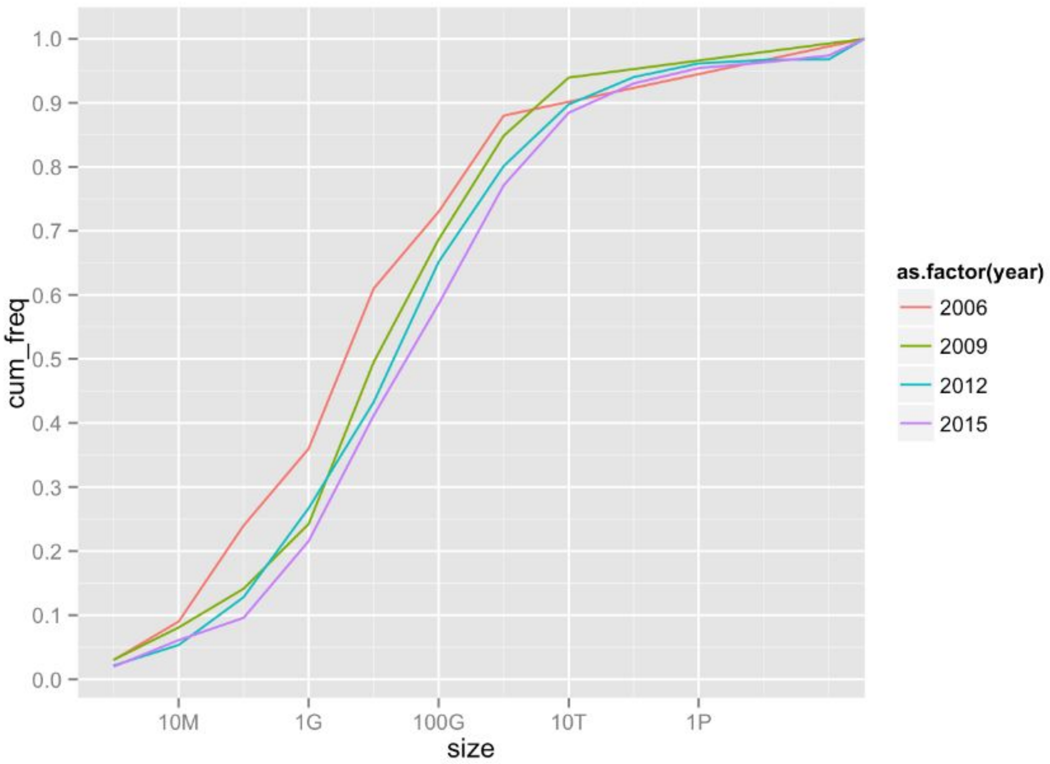
Largest Dataset Analyzed / Data Mined

Tweet

What was the largest dataset you analyzed / data mined? [392 votes]

2014 votes 2013 votes

szilard / **dataset-sizes-kdnuggets**



Szilard @DataScienceLA · Jun 17

What's the typical size of datasets you are analyzing?

18% <100MB

48% 100MB-10GB

18% 10GB-1TB

16% >1TB

151 votes • Final results

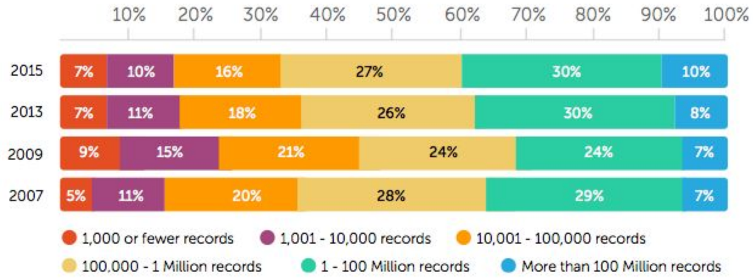
Szilard @DataScienceLA · Jun 17

What's the typical size of datasets you are analyzing?

18%	<100MB
48%	100MB-10GB
18%	10GB-1TB
16%	>1TB

151 votes • Final results

TYPICAL SIZE OF DATASETS



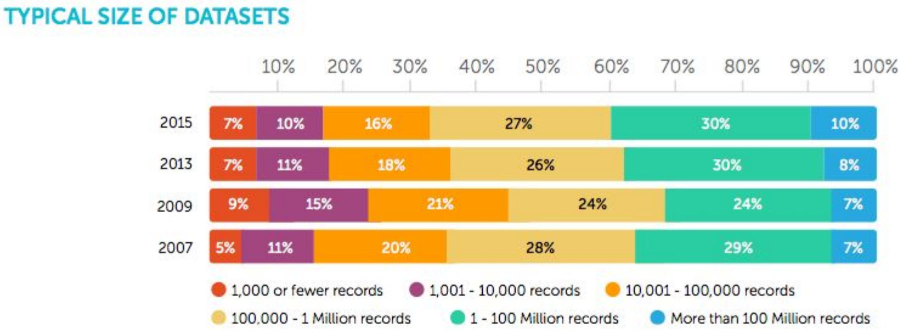
17 Rexer Analytics

Szilard @DataScienceLA · Jun 17

What's the typical size of datasets you are analyzing?

18%	<100MB
48%	100MB-10GB
18%	10GB-1TB
16%	>1TB

151 votes • Final results



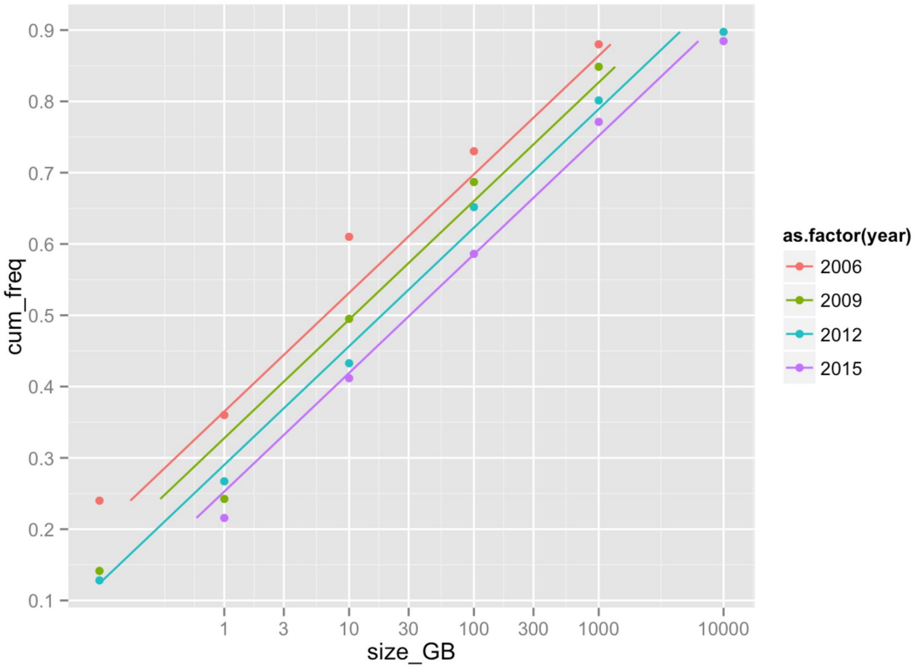
17 Rexer Analytics

Hadley Wickham
@hadleywickham

Following

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

szilard / **dataset-sizes-kdnuggets**



annual rate of increase of datasets of `10^0.075 ~ 1.2` that is 20%.

The size of EC2 instances with largest RAM:

year	type	RAM (GB)
2007	m1.xlarge	15
2009	m2.4xlarge	68
2012	hs1.8xlarge	117
2014	r3.8xlarge	244
2016*	x1	2 TB

The size of EC2 instances with largest RAM:

year	type	RAM (GB)
2007	m1.xlarge	15
2009	m2.4xlarge	68
2012	hs1.8xlarge	117
2014	r3.8xlarge	244
2016*	x1	2 TB

Szilard @DataScienceLA · 18 Nov 2015
Big RAM is eating #bigdata: datasets for analytics grew 20% /yr (last decade @kdnuggets), RAM EC2 grew 50% /yr


GB

R

PyData

GB

R

sas

PyData

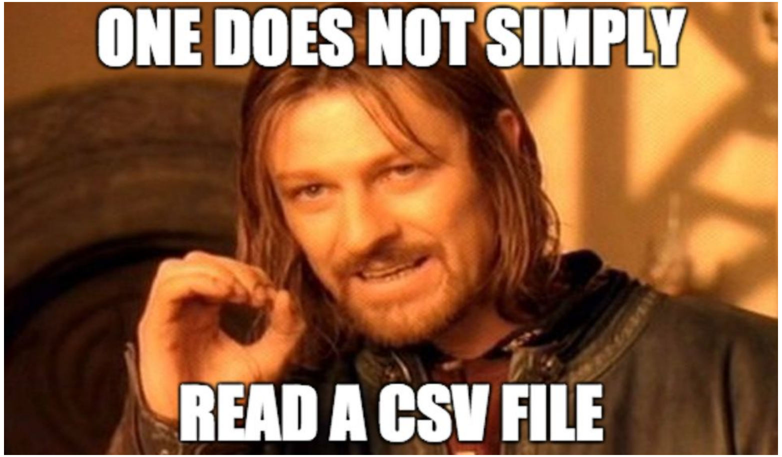
MATLAB

data frame. Yesterday I wrote a blog post about using `sqldf()` to import the data into SQLite as a staging area, and then sucking it from SQLite into R. This works really well for me. I was able to pull in 2GB (3 columns, 40mm rows) of data in < 5 minutes. By contrast, the `read.csv` command ran all night and never completed.

answered Nov 30 '09 at 15:48

JD Long

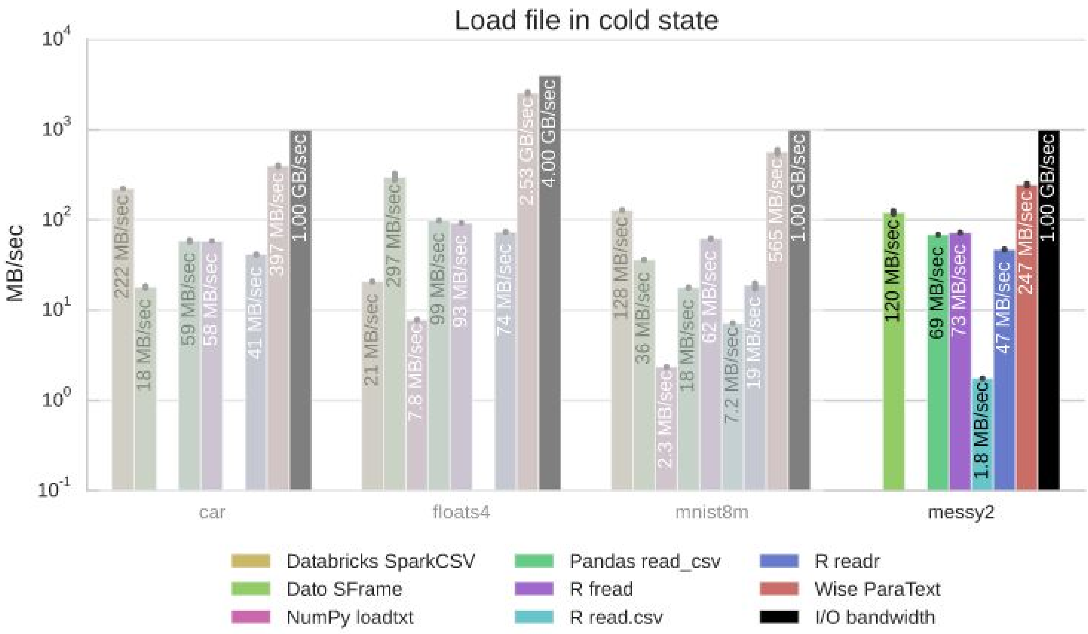
27.5k • 31 • 135 • 216



data frame. Yesterday I wrote a blog post about using `sqldf()` to import the data into SQLite as a staging area, and then sucking it from SQLite into R. This works really well for me. I was able to pull in 2GB (3 columns, 40mm rows) of data in < 5 minutes. By contrast, the `read.csv` command ran all night and never completed.

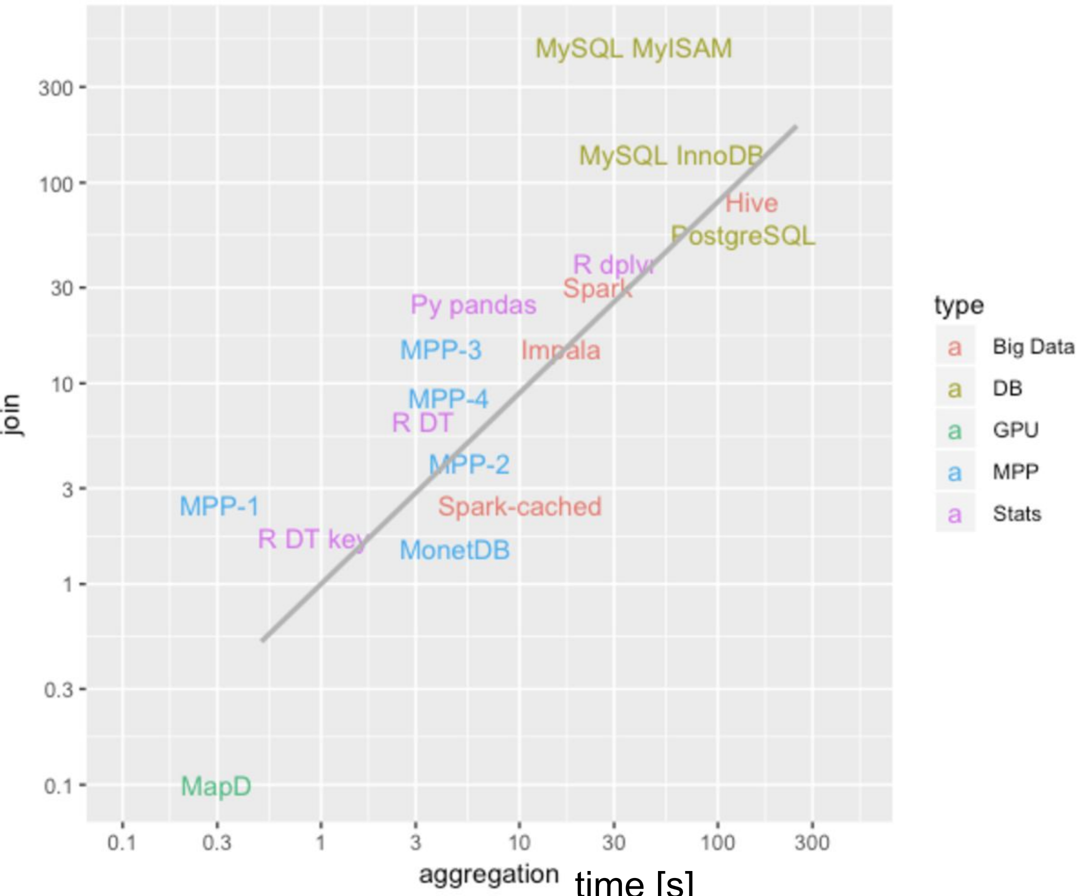
answered Nov 30 '09 at 15:48

JD Long
27.5k • 31 • 135 • 216



https://deads.gitbooks.io/paratext-bench/content/teaser.html June 2016 szilard / **benchm-databases**

Aggregation 100M rows 1M groups
Join 100M rows x 1M rows



time [s]

time [s]

szilard / **dataset-sizes-kdnuggets**

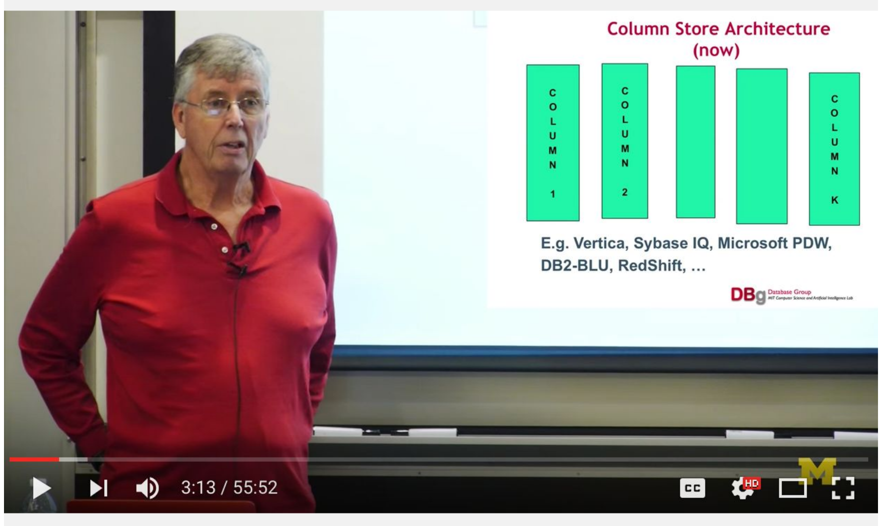
quantile	value
50%	30 GB
60%	120 GB
70%	0.5 TB
80%	2 TB
90%	8 TB

(largest data analyzed)

szilard / dataset-sizes-kdnuggets

quantile	value
50%	30 GB
60%	120 GB
70%	0.5 TB
80%	2 TB
90%	8 TB

(largest data analyzed)

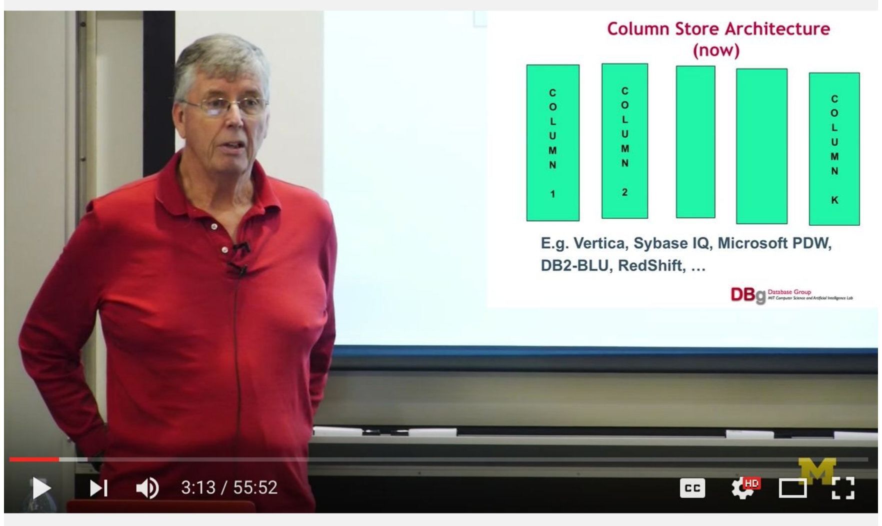


Michael Stonebraker | Big Data is (at least) Four Different Problems

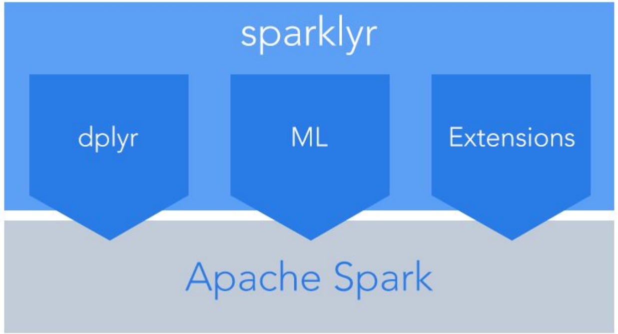
szilard / dataset-sizes-kdnuggets

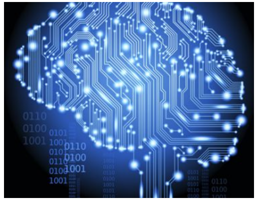
quantile	value
50%	30 GB
60%	120 GB
70%	0.5 TB
80%	2 TB
90%	8 TB

(largest data analyzed)

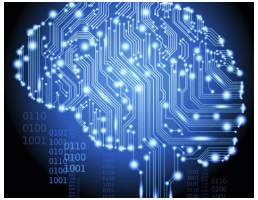


Michael Stonebraker | Big Data is (at least) Four Different Problems



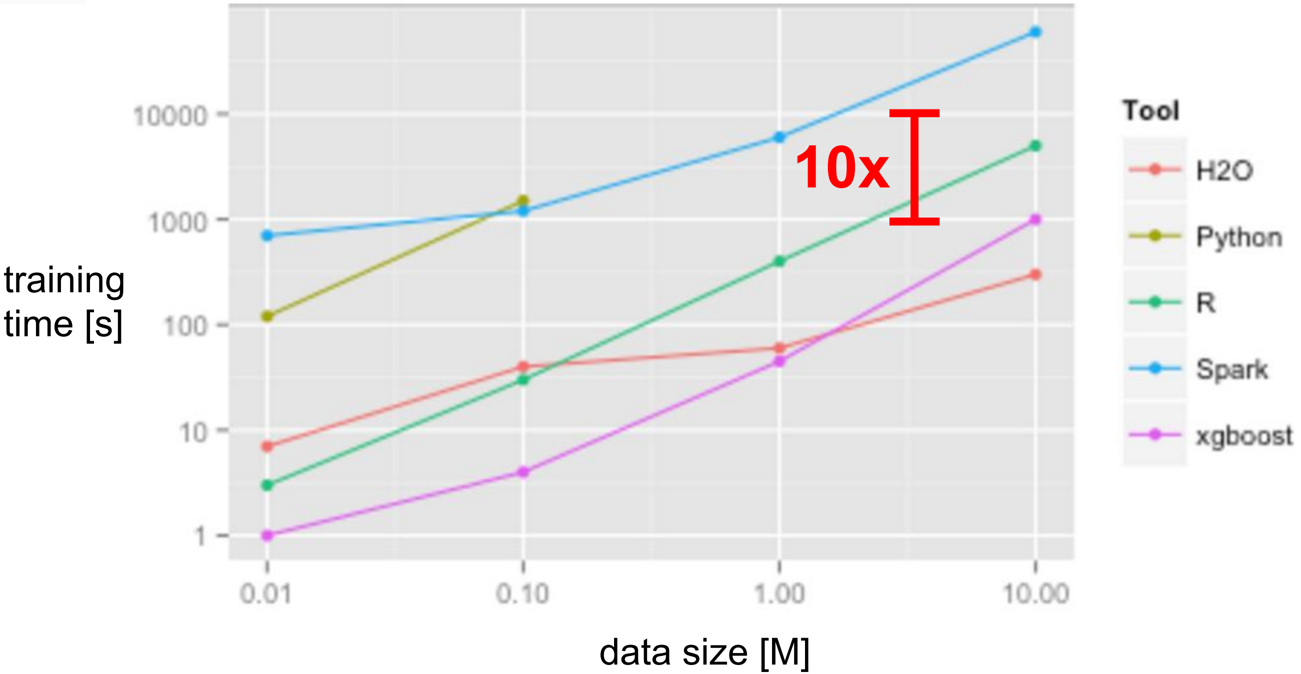


szilard / **benchm-ml**



szilard / benchm-ml

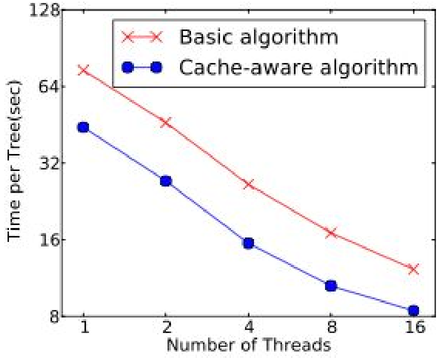
Gradient Boosting Machines



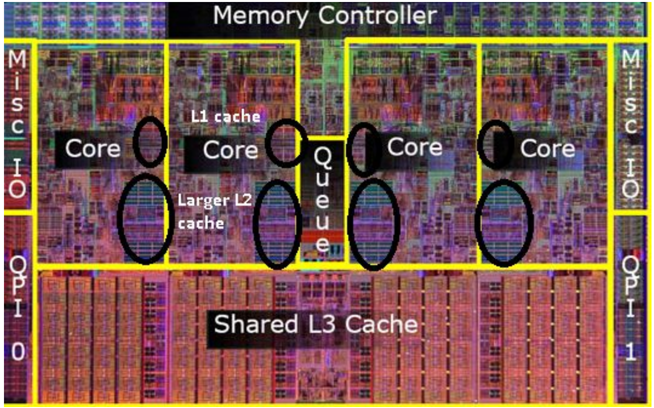
arXiv.org > cs > arXiv:1603.02754

Computer Science > Learning

XGBoost: A Scalable Tree Boosting System



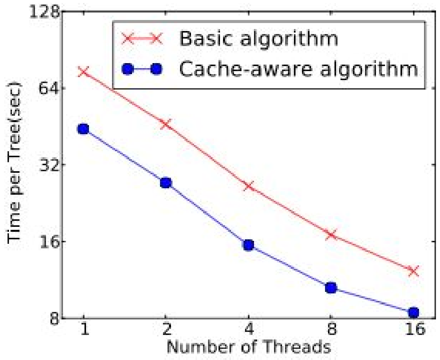
H_2O.ai

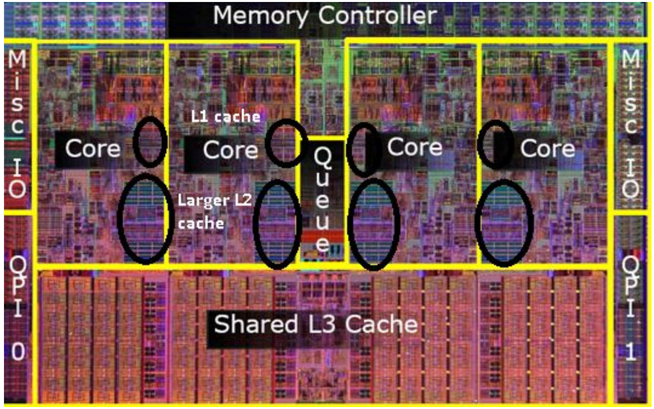


arXiv.org > cs > arXiv:1603.02754

Computer Science > Learning

XGBoost: A Scalable Tree Boosting System





H2O.ai

R

https://cran.r-project.org/web/pa

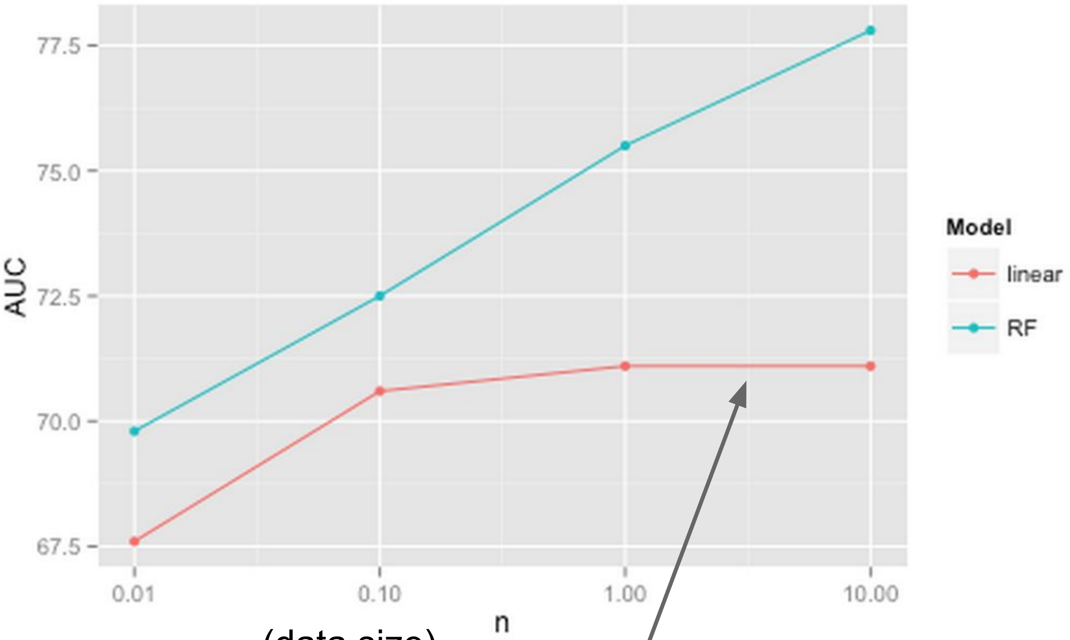
xgboost: Extreme Gradient Boosting

https://cran.r-project.org/

h2o: R Interface for H2O

PyData

(accuracy)

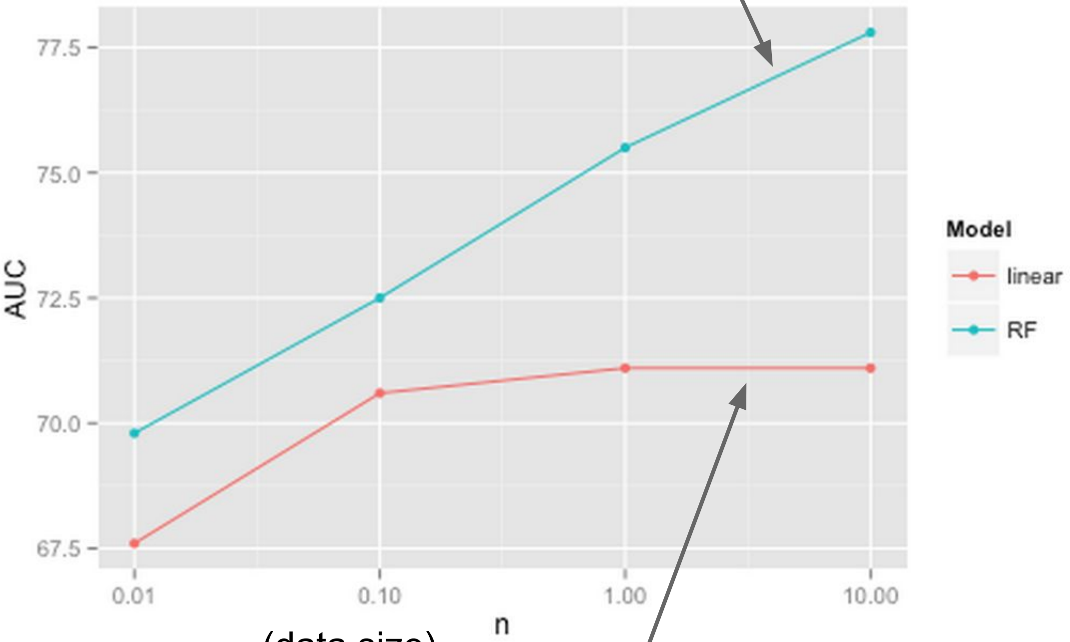


(data size)

linear tops off

more data & better algo

(accuracy)

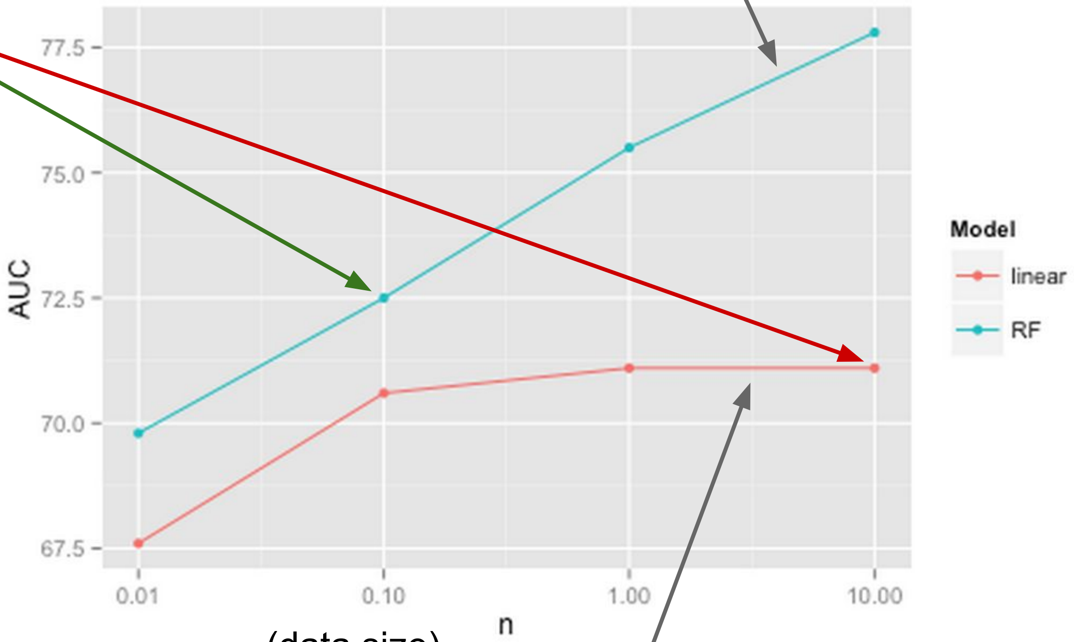


(data size)

linear tops off

random forest on 1% of data
beats linear on all data

more data & better algo



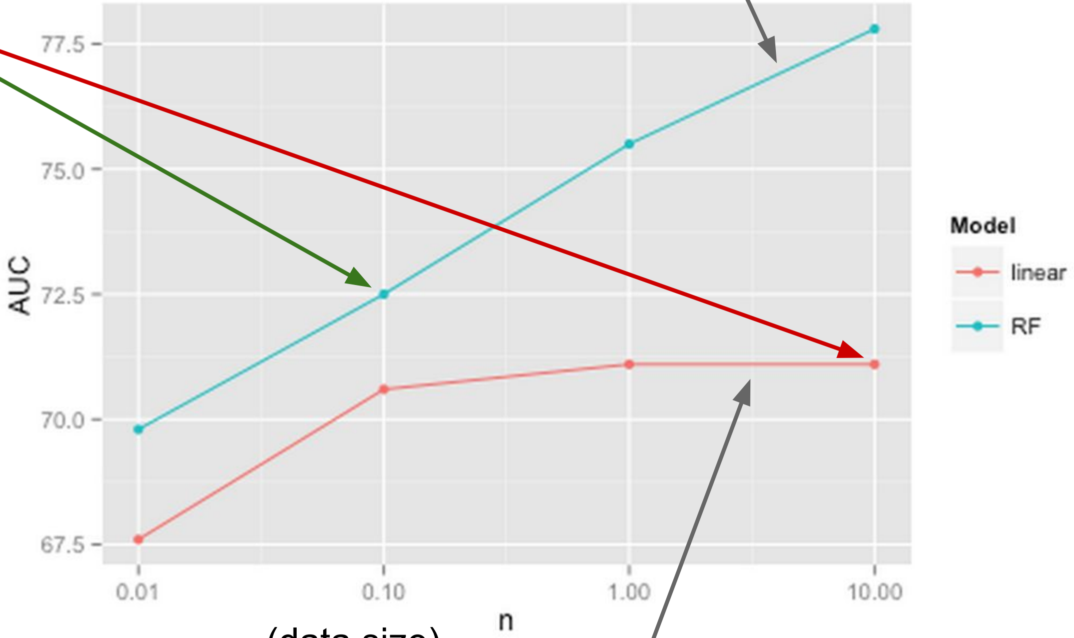
(accuracy)

(data size)

linear tops off

random forest on 1% of data
beats linear on all data

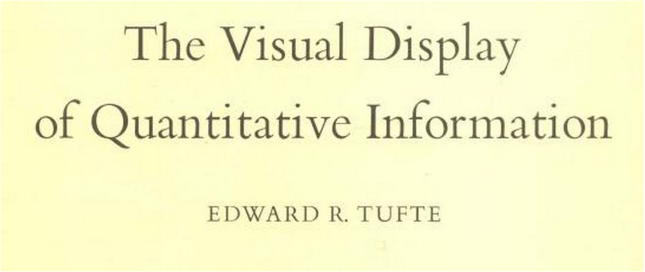
more data & better algo

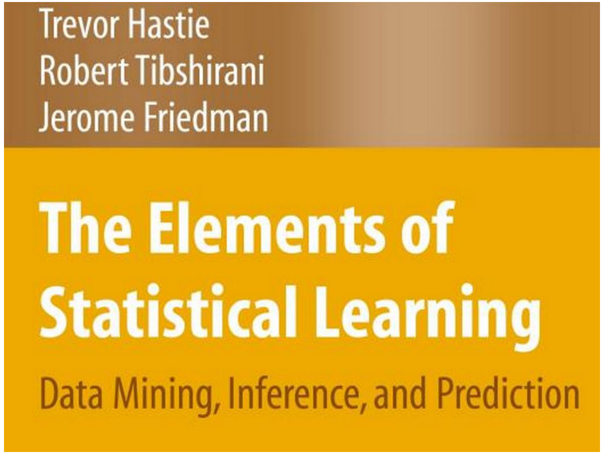


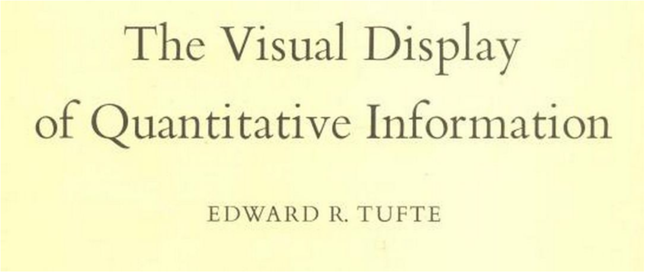
(data size)

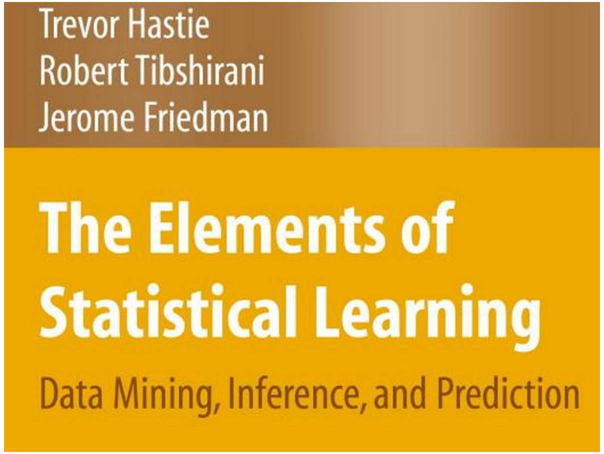
linear tops off

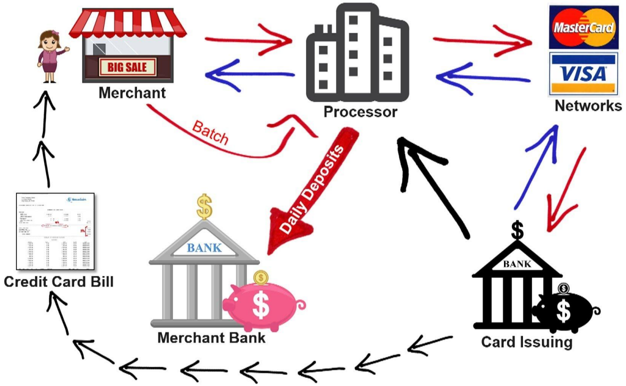

HITACHI


The Visual Display
of Quantitative Information
EDWARD R. TUFTE

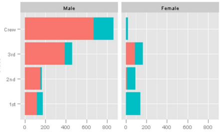

Trevor Hastie
Robert Tibshirani
Jerome Friedman
The Elements of
Statistical Learning
Data Mining, Inference, and Prediction


The Visual Display
of Quantitative Information
EDWARD R. TUFTE

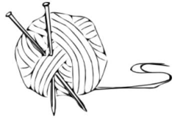

Trevor Hastie
Robert Tibshirani
Jerome Friedman
The Elements of
Statistical Learning
Data Mining, Inference, and Prediction


BIG SALE
Merchant
Processor
MasterCard
VISA
Networks
Batch
Daily Deposits
Credit Card Bill
BANK
Merchant Bank
BANK
Card Issuing

{ api }



R Studio



8,576
active packages

Meetup

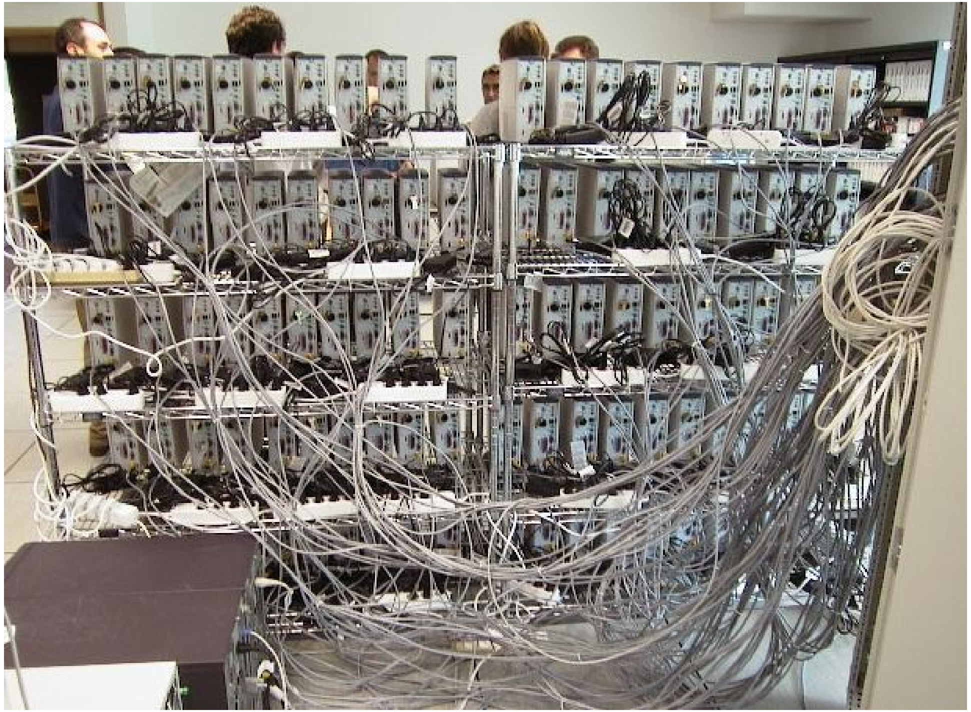
data.table

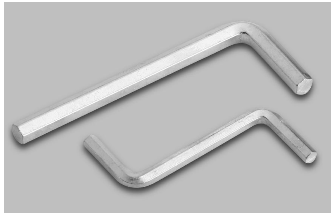
stackoverflow

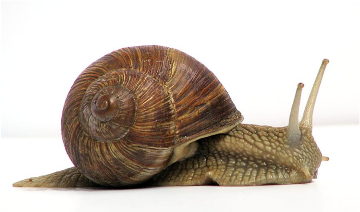
Seamless R and C++ Integration with Rcpp

useR!

R consortium

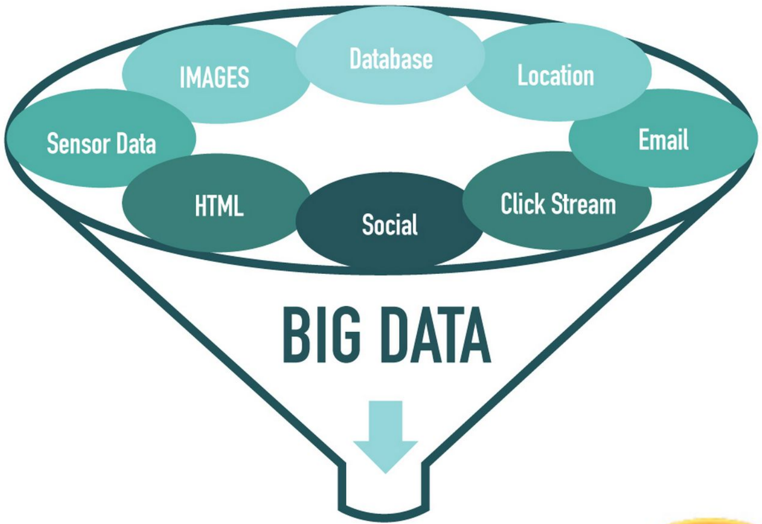




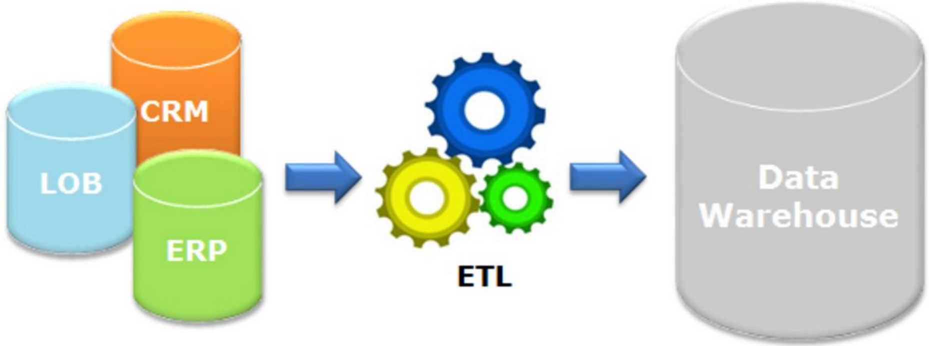



SYSTEM FAILURE

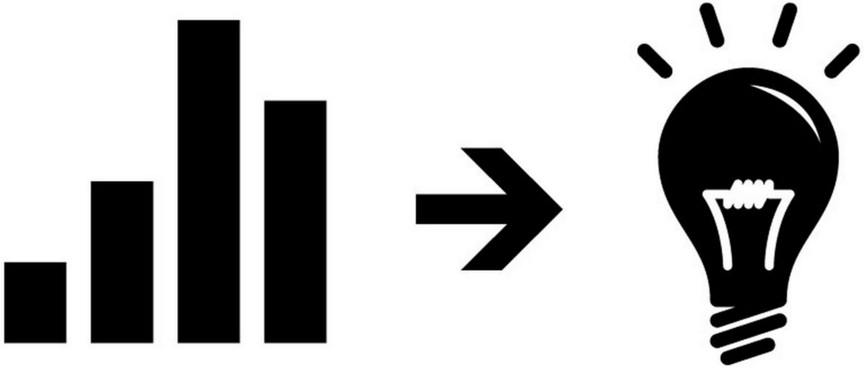

I DON'T ALWAYS ANALYZE MY DATA, BUT WHEN I DO
I SPEND ALL MY TIME FIGHTING THE TOOLS
imgflip.com

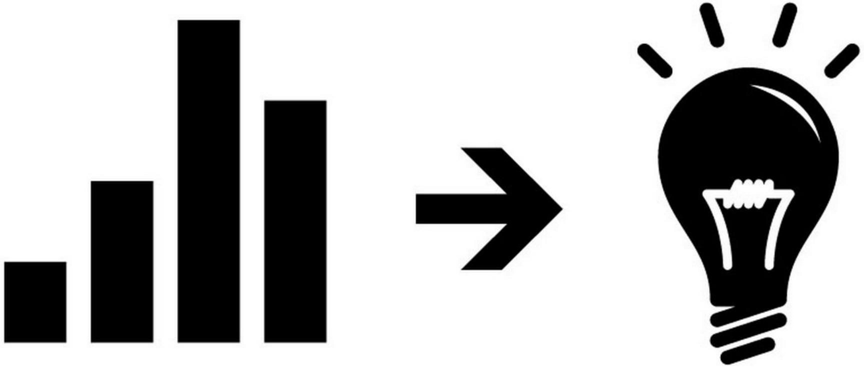


Database
IMAGES
Location
Sensor Data
Email
HTML
Social
Click Stream
BIG DATA



CRM
LOB
ERP
ETL
Data Warehouse





BE PRODUCTIVE.

Summary / Tips for analyzing “big” data:

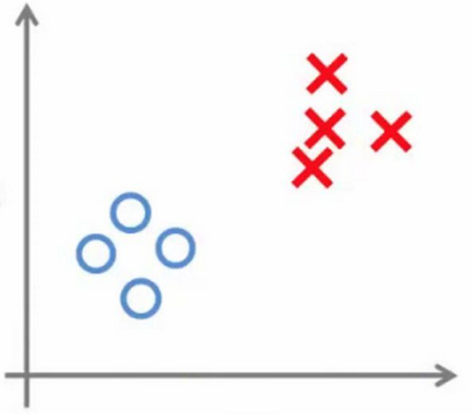
- Get lots of RAM (physical/ cloud)
- Use R/Python and high performance packages (e.g. data.table, xgboost)
- Do data reduction in database (analytical db/ big data system)
- (Only) distribute embarrassingly parallel tasks (e.g. hyperparameter search for machine learning)
- Let engineers (store and) ETL the data (“scalable”)
- Use statistics/ domain knowledge/ thinking
- Use “big data tools” only if the above tips not enough

Example #2

?

open source

Off the Shelf

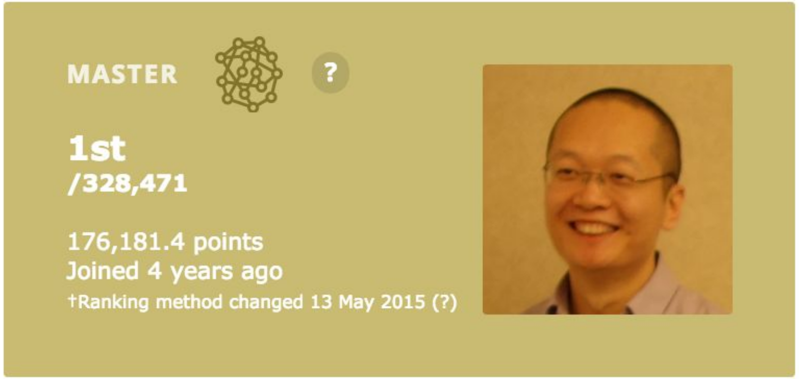


(10,000,000 rows)

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



GitHub This repository Search

szilard / **benchm-ml**

binary classification, 10M records

numeric & categorical features, non-sparse

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

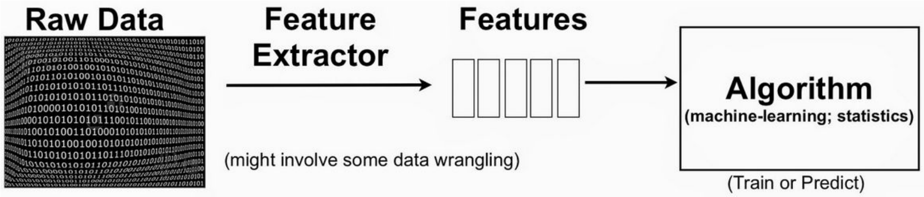
AVG	1ST	2ND
RF	0.727	0.207
ANN	0.053	0.172
BSTDT	0.059	0.228
~~SVM~~	0.043	0.195
~~LR~~	0.089	0.132
BAGDT	0.002	0.012
KNN	0.023	0.045
BSTST	0.004	0.009
PRC	0	0
NB	0	0

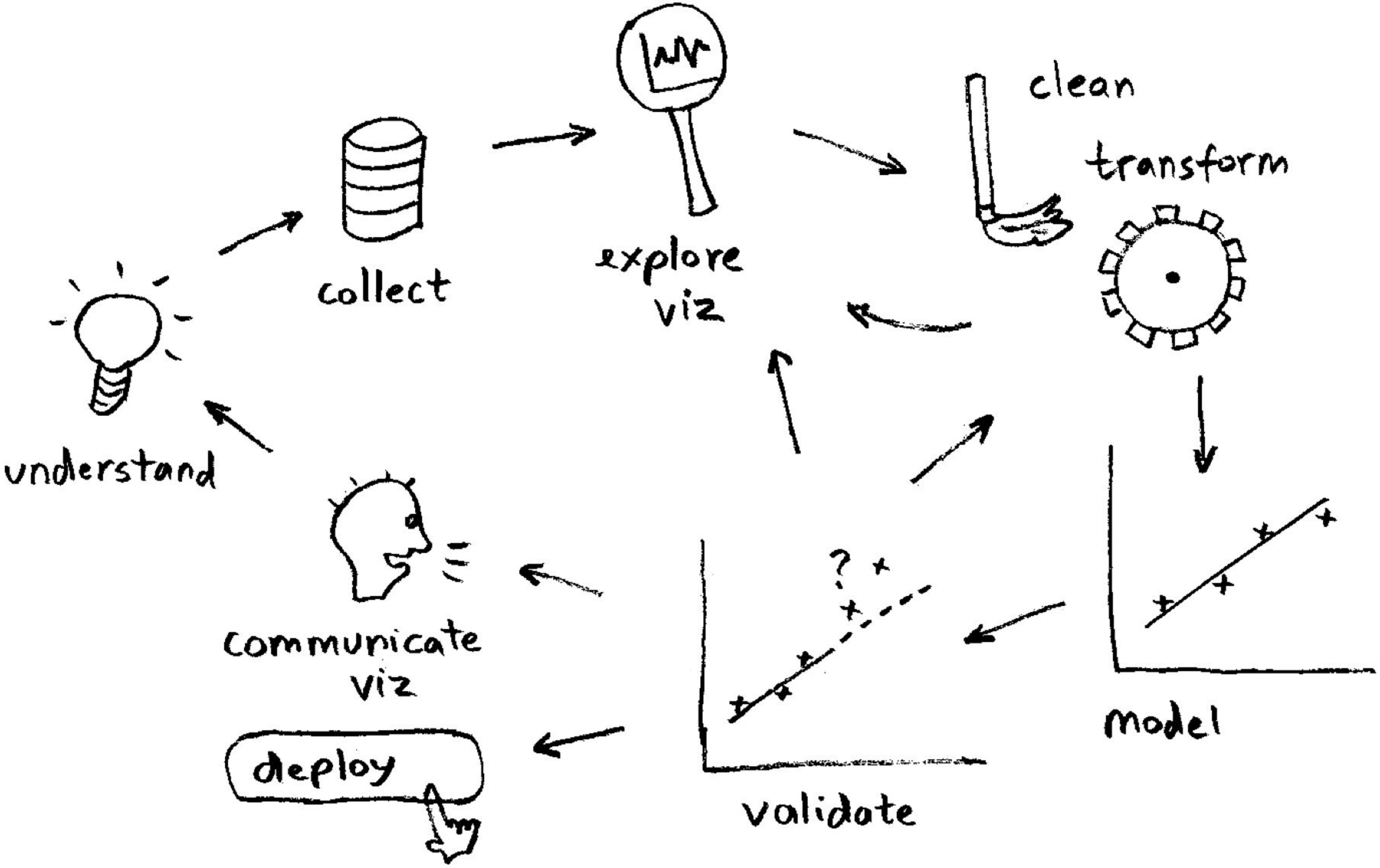
An Empirical Comparison of Supervised Learning Algorithms

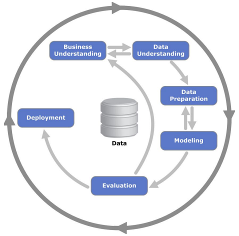
http://www.cs.cornell.edu/~alexn/papers/empirical.icml06.pdf

An Empirical Evaluation of Supervised Learning in High Dimensions

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


Raw Data
Feature
Extractor
Features
Algorithm
(machine-learning; statistics)
(might involve some data wrangling)
(Train or Predict)


understand
collect
explore
viz
clean
transform
model
validate
communicate
viz
deploy


Business Understanding
Data Understanding
Data Preparation
Modeling
Evaluation
Deployment
Data

open source

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

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

open source

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

Lightning-fast

Big Data

Large-scale

optimized

Distributed

Yes, it is possible to apply [redacted] to **big data**!

Scalable and fast

parallelized

scalable performant machine learning

built for scale


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


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

EC2

n = 10K, 100K, 1M, 10M, 100M

Training time

RAM usage

AUC

CPU % by core

read data, pre-process, score test data

n = 10K, 100K, 1M, 10M, 100M

Training time

RAM usage

AUC

CPU % by core

read data, pre-process, score test data

szilard / **benchm-ml**

branch: **master**

benchm-ml / **2-rf** / +

xgboost improve

szilard authored 19 days ago

..

1.R

2.py

4-h2o-v3.R

4-h2o.R


20%
of the
effort
80%
of the
results

szilard / benchm-ml

Star 1,117

Simple/limited/incomplete benchmark

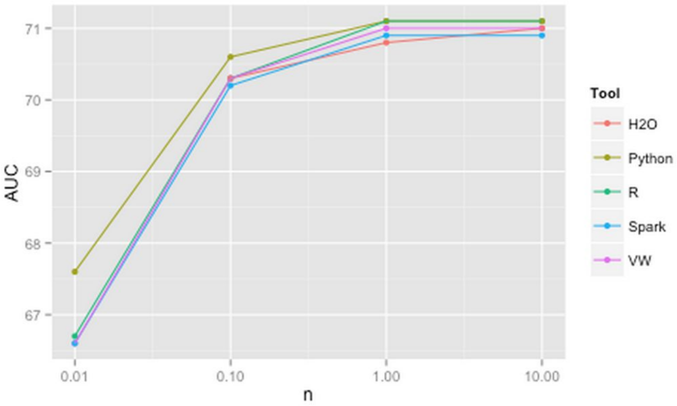
szilard / **benchm-ml** ★ Star 1,117

Simple/limited/incomplete benchmark

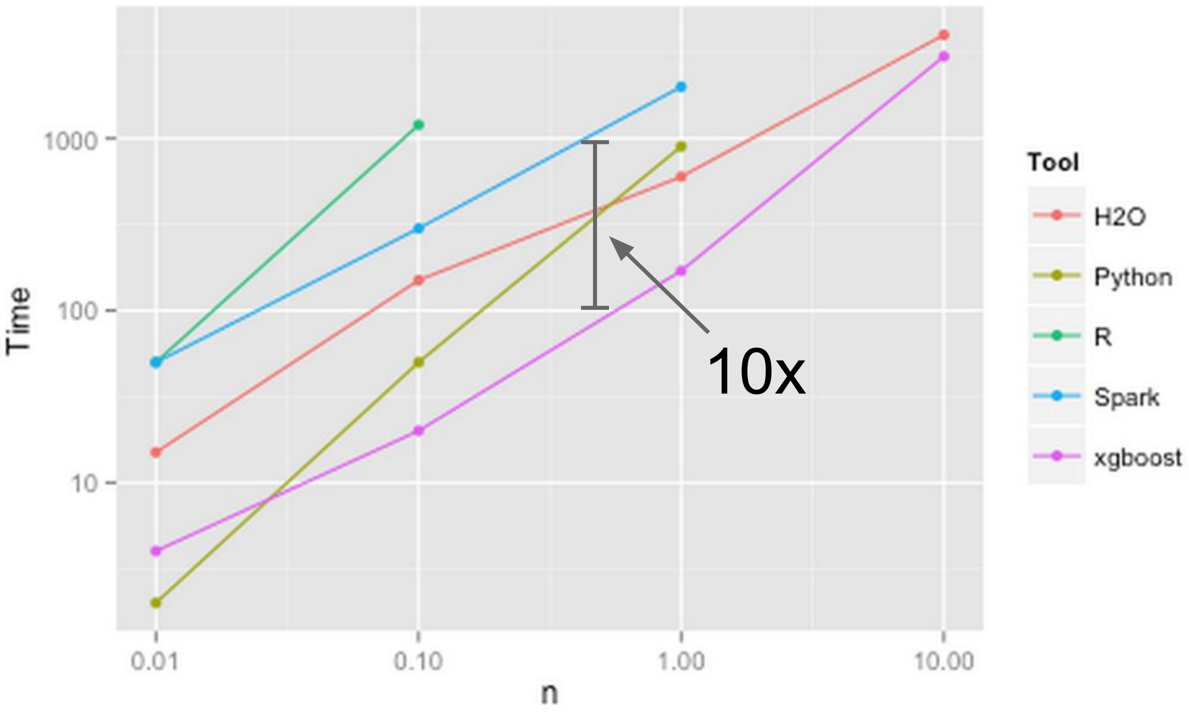
All benchmarks are wrong, but some are useful


LINEAR

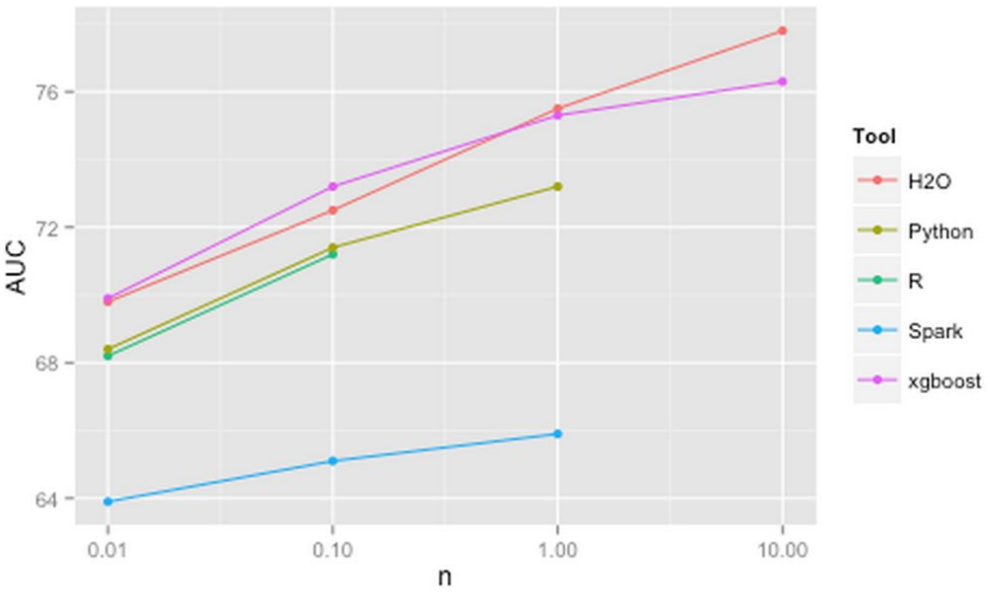
Tool	*n*	Time (sec)	RAM (GB)
R	10K	0.1	1
	100K	0.5	1
	1M	5	1
	10M	90	5

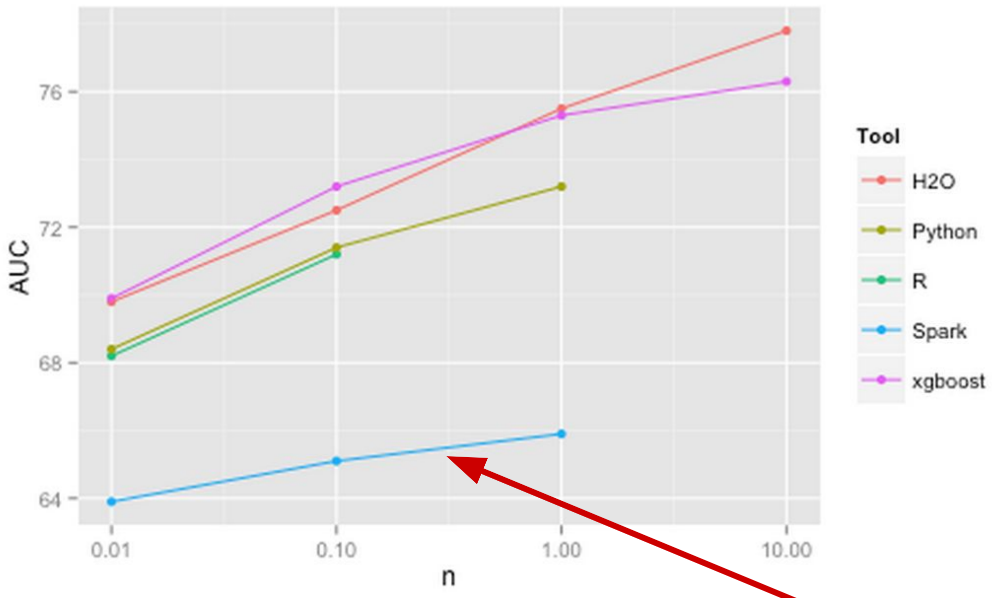


The main conclusion here is that it is trivial to train linear models even for *n* = 10M rows virtually in any of these tools on a single machine in a matter of seconds. H2O and VW are the most memory efficient



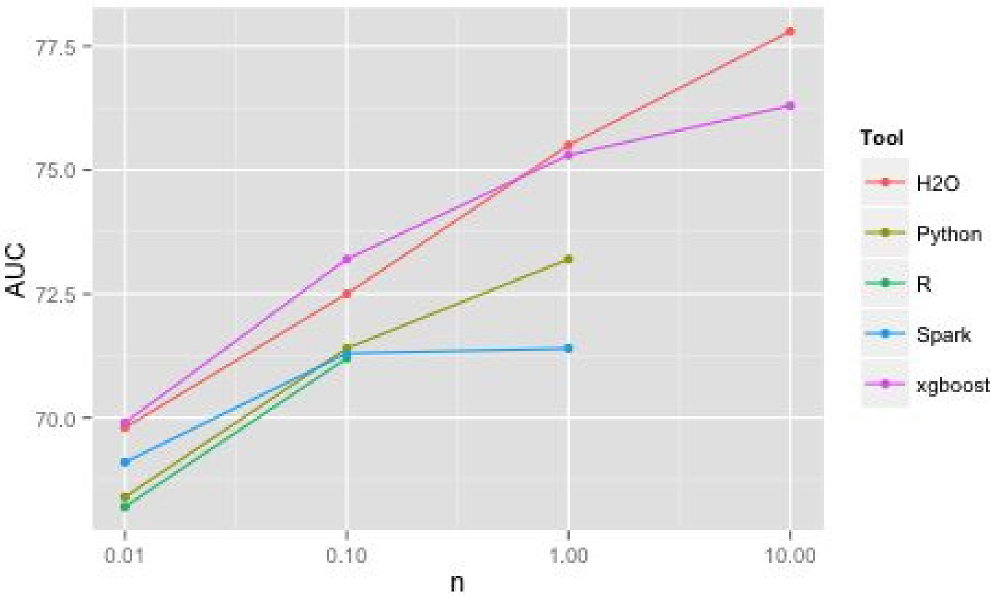
Time
1000
100
10
0.01
0.10
1.00
10.00
n
10x
Tool
H2O
Python
R
Spark
xgboost


76
72
68
64
AUC
0.01
0.10
1.00
10.00
n
Tool
H2O
Python
R
Spark
xgboost



Joseph Bradley

AUC/accuracy: The AUC issue appears to be caused by MLlib tree ensembles aggregating votes, rather than class probabilities, as you suggested. I re-ran your test using class probabilities (which can be aggregated by hand), and then got the same AUC as other libraries. We're planning on including this fix in Spark 1.5 (and thanks for providing some evidence of its importance!).


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

Timing: I didn't try to reproduce your results yet, but have a few thoughts. The main issue with MLlib's tree implementation is that it is optimized for training shallow trees, following the PLANET project. We're working on an alternative implementation geared towards training deep trees, hopefully aimed at Spark 1.5 or 1.6. One big benefit of running on top of Spark is that there is constant work on improving the underlying system, which MLlib will benefit from. In particular, the JVM memory management issues will improve as project Tungsten (which can be Googled) progresses.

http://datascience.la/benchmarking-random-forest-implementations/#comment-53599

APACHE Spark Lightning-fast

Download Libraries Documentati

Spark Release 2.0.0

Lightning-fast

Download Libraries Documentati

Spark Release 2.0.0

DataFrame-based API is primary API

APACHE Spark™ *Lightning-fast*

Download Libraries ▾ Documentati

Spark Release 2.0.0

DataFrame-based API is primary API

Spark random forest issues #19

Open **szilard** opened this issue on Jul 23, 2015 · 15 comments

szilard commented 4 days ago

It is slower than before

```
MLlib 1.5 - 250 sec
ML    2.0 - 400 sec
```

APACHE Spark™ Lightning-fast

Download Libraries ▾ Documentati

Spark Release 2.0.0

DataFrame-based API is primary API

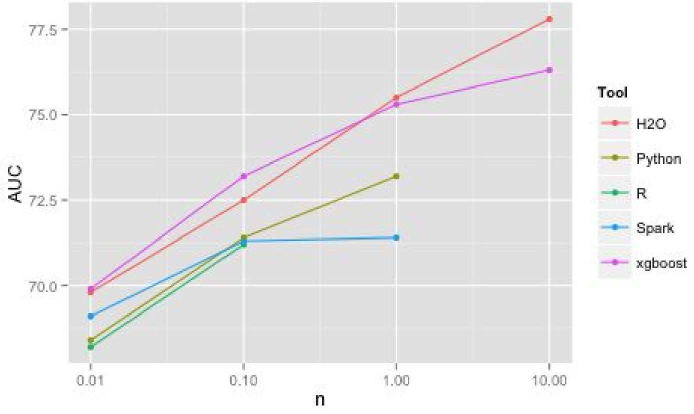
Spark random forest issues #19

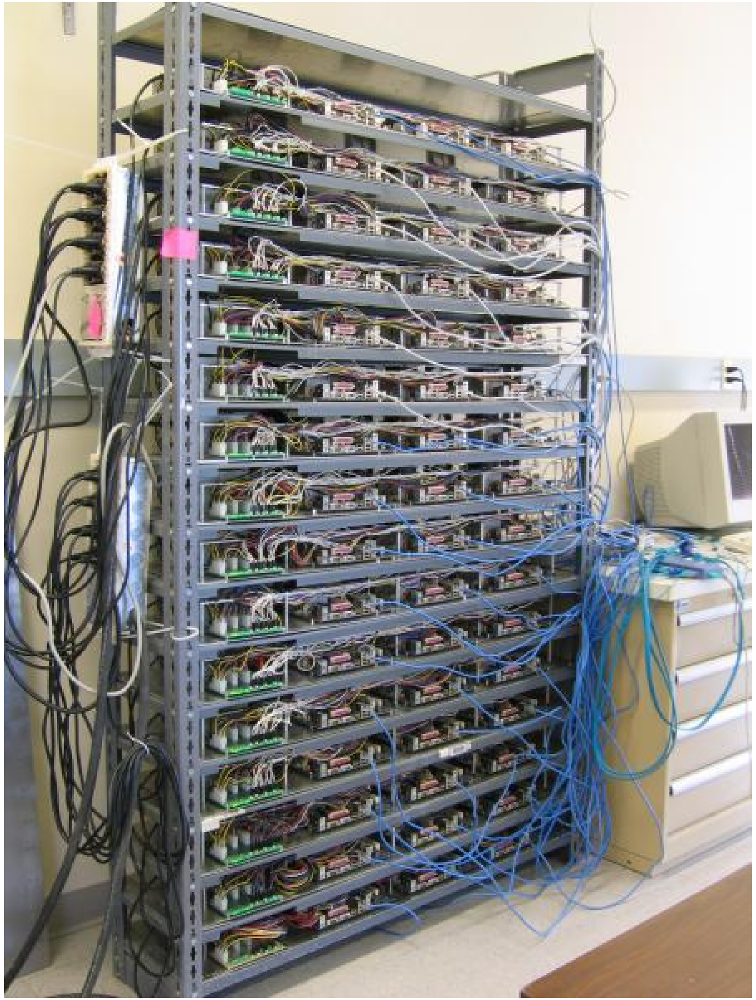
Open **szilard** opened this issue on Jul 23, 2015 · 15 comments

szilard commented 4 days ago

It is slower than before

```
MLlib 1.5 - 250 sec
ML    2.0 - 400 sec
```





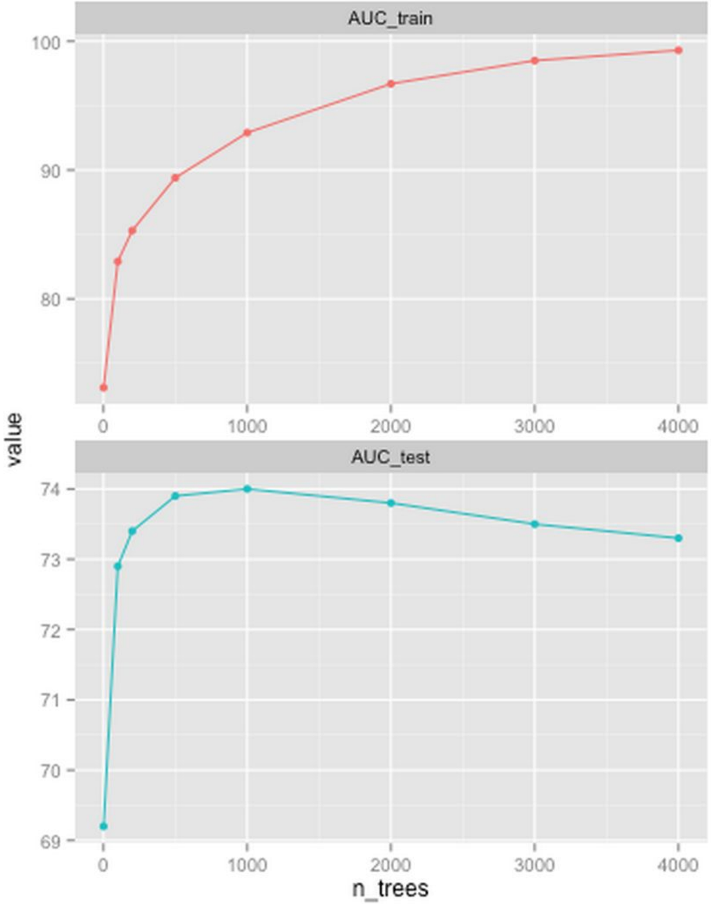
ntree	depth	nbins	mtries	Time (hrs)	AUC
500	20	20	-1 (2)	1.2	77.8
500	50	200	-1 (2)	4.5	78.9
500	50	200	3	5.5	78.9
5000	50	200	-1 (2)	45	79.0
500	100	1000	-1 (2)	8.3	80.1

ntree	depth	nbins	mtries	Time (hrs)	AUC
500	20	20	-1 (2)	1.2	77.8
500	50	200	-1 (2)	4.5	78.9
500	50	200	3	5.5	78.9
5000	50	200	-1 (2)	45	79.0
500	100	1000	-1 (2)	8.3	80.1

Best linear: 71.1

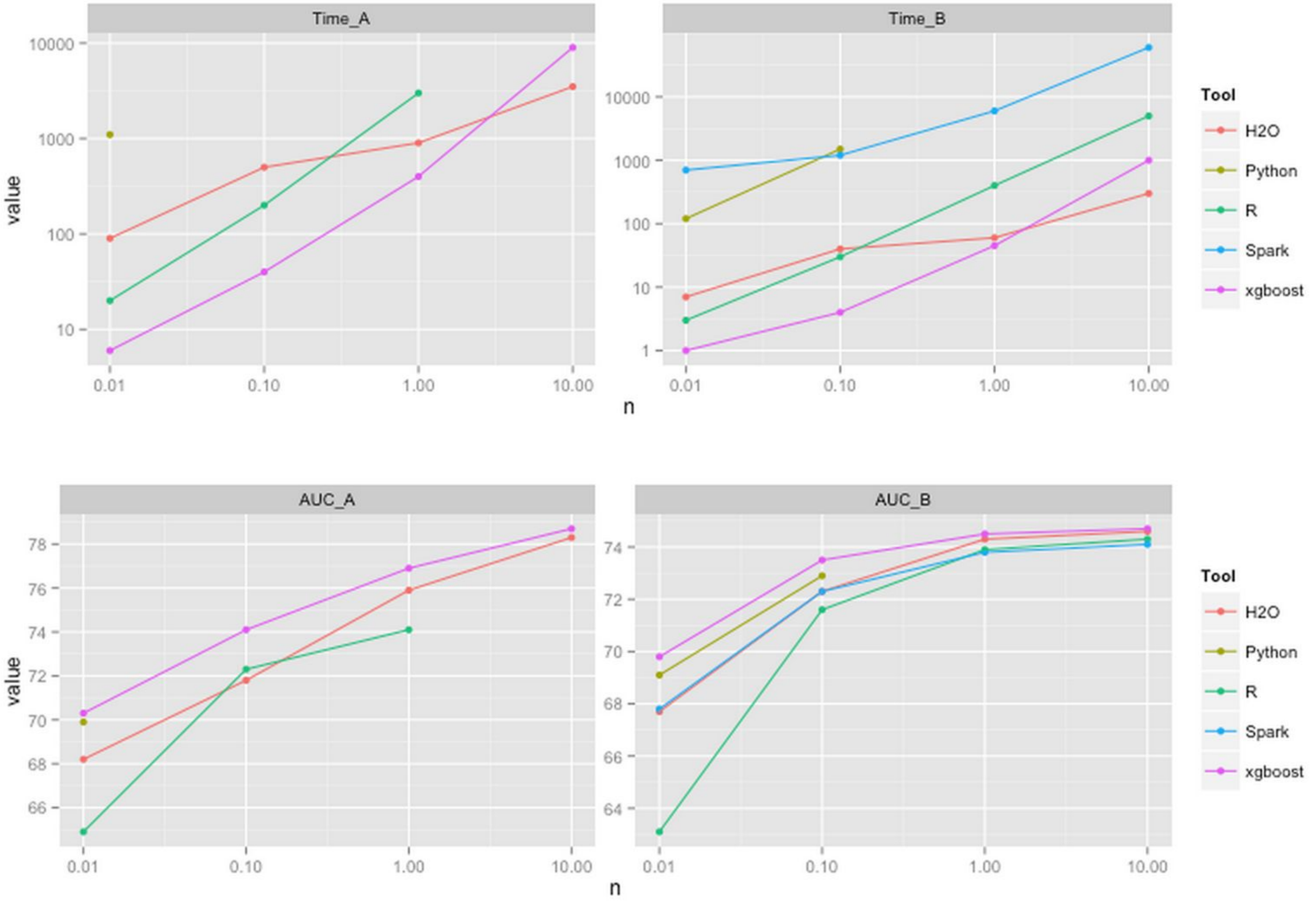

time for a boost


time for a boost

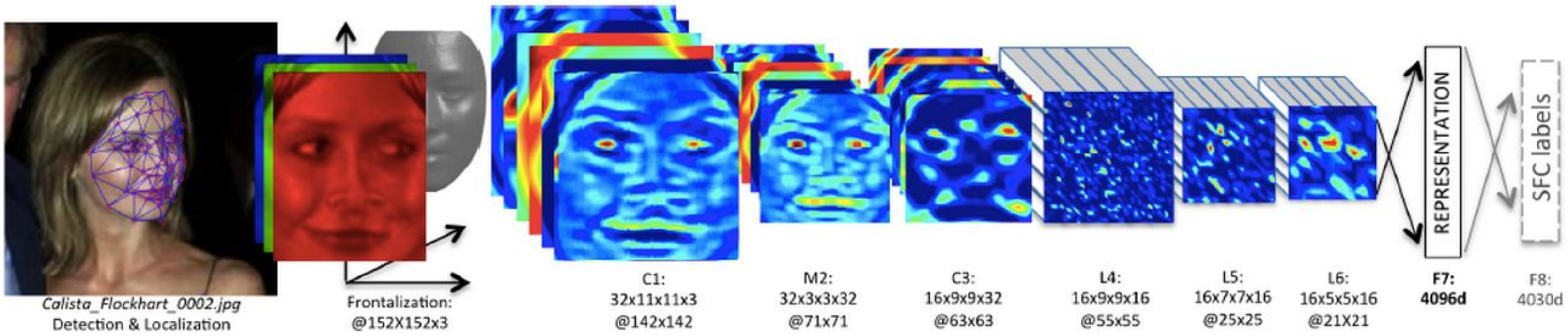

AUC_train
AUC_test
value
n_trees
100
90
80
74
73
72
71
70
69
0
1000
2000
3000
4000

learn_rate = 0.01, max_depth = 16, n_trees = 1000

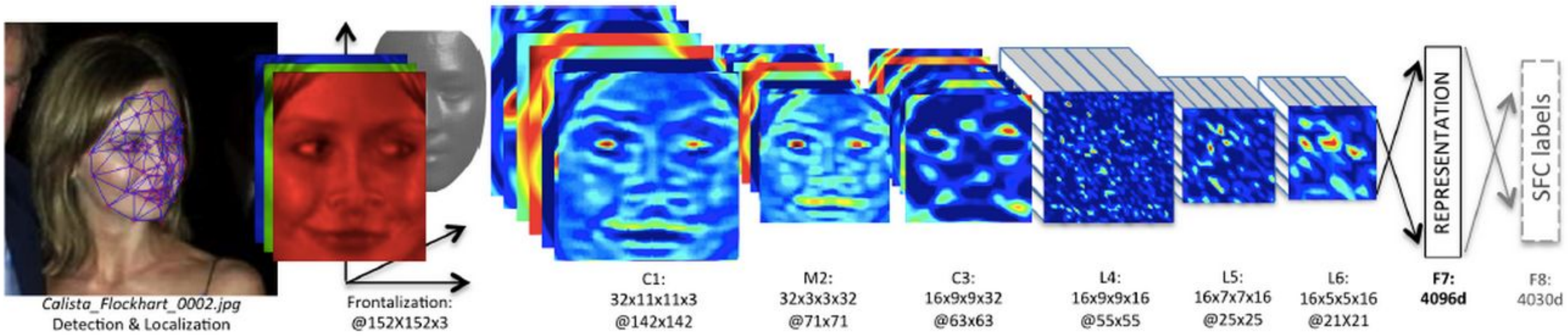
learn_rate = 0.1, max_depth = 6, n_trees = 300



Tool	Time (hr)	AUC
H2O	7.5	79.8
H2O-3	9.5	81.2
xgboost	14	81.1



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
REPRESENTATION
F7:
4096d
SFC labels
F8:
4030d



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

...

`ADADELTA rho = 0.95, epsilon = 1e-06`	71.1	240	1.7
`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


MY DATA
IS BIGGER THAN YOURS

Larger Data Sizes (on a Single Server)

Big Data Borat @BigDataBorat · 22 Nov 2013

cat data data data data data > bigdata

Larger Data Sizes (on a Single Server)

Big Data Borat @BigDataBorat · 22 Nov 2013

cat data data data data data > bigdata

Linear models, 100M rows:

Tool	Time[s]	RAM[GB]
R	1000	60
Spark	160	120
H2O	40	20
VW	150	

Larger Data Sizes (on a Single Server)

Big Data Borat @BigDataBorat · 22 Nov 2013

cat data data data data data > bigdata

Linear models, 100M rows:

Tool	Time[s]	RAM[GB]
R	1000	60
Spark	160	120
H2O	40	20
VW	150	

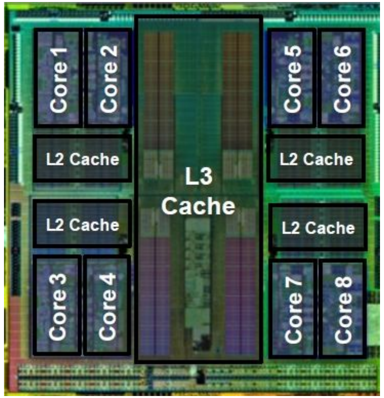
Linear models, 1B rows:

Tool	Time[s]	RAM[GB]
H2O	500	100
VW	1400	

Larger Data Sizes (on a Single Server)

RF/GBM, 100M rows:

Algo	Tool	Time[s]	Time[hr]	RAM[GB]
RF	H2O	40000	11	80
	xgboost	36000	10	60
GBM	H2O	35000	10	100
	xgboost	110000	30	50



Distributed Systems

H2O logistic runtime (sec):

	1 node	5 nodes
100M	42	9.9
1B	480	101

H2O RF runtime (sec) (5 trees):

	1 node	5 nodes
10M	42	41
100M	405	122

Summary

STANLEY
FatMax
BladeArmor
10m


Business Optimization


Business Optimization


BS


HYPE

spafka@gmail.com

@DataScienceLA

linkedin.com/in/szilard

github.com/szilard