

Your data fits in RAM: How to avoid cluster computing

PYDATA MIAMI 2019



Aaron Richter

@rikturr

DATA SCIENTIST

(DATA ENGINEER)

PHD CANDIDATE



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.

6:03 PM · 5/19/15 · GRB t app

1,320 Retweets 1,616 Likes





Cluster computing has its place

The cloud is your friend

Your laptop is stronger than you think

The cloud is your friend

Your laptop is stronger than you think



There is no cloud
it's just someone else's computer



Contact Sales Support English ▾ My Account ▾

Create an AWS Account

Products Solutions Pricing Documentation Learn Partner Network AWS Marketplace E ▾ Q

Amazon EC2

Overview Features Pricing Instance Types FAQs Getting Started Resources ▾

Memory Optimized

General Purpose

Compute Optimized

Memory Optimized

Accelerated Computing

Storage Optimized

Instance Features

Measuring Instance Performance

R5 R5a R4 X1e X1 High Memory z1d

[R5 instances](#) deliver 5% additional memory per vCPU than R4 and the largest size provides 768 GiB of memory. In addition, R5 instances deliver a 10% price per GiB improvement and a ~20% increased CPU performance over R4.

Features:

- Up to 768 GiB of memory per instance
- Intel Xeon Platinum 8000 series (Skylake-SP) processors with a sustained all core Turbo CPU clock speed of up to 3.1 GHz
- Powered by the AWS Nitro System, a combination of dedicated hardware and lightweight hypervisor
- With R5d instances, local NVMe-based SSDs are physically connected to the host server and provide block-level storage that is coupled to the lifetime of the R5 instance



Contact Sales Support English ▾ My Account ▾

Create an AWS Account

Products Solutions Pricing Documentation Learn Partner Network AWS Marketplace E ▾ Q

Amazon EC2

Overview Features Pricing Instance Types FAQs Getting Started Resources ▾

Memory Optimized

General Purpose

Compute Optimized

Memory Optimized

Accelerated Computing

Storage Optimized

Instance Features

Measuring Instance Performance

R5 R5a R4 X1e X1 High Memory z1d

R5 instances deliver 5% additional memory per vCPU than R4 and the largest size provides 768 GiB of memory. In addition, R5 instances deliver a 10% price per GiB improvement and a ~20% increased CPU performance over R4.

Features:

- Up to 768 GiB of memory per instance
- Intel Xeon Platinum 8000 series (Skylake-SP) processors with a sustained all core Turbo CPU clock speed of up to 3.1 GHz
- Powered by the AWS Nitro System, a combination of dedicated hardware and lightweight hypervisor
- With R5d instances, local NVMe-based SSDs are physically connected to the host server and provide block-level storage that is coupled to the lifetime of the R5 instance

P2

P2 instances are intended for general-purpose GPU compute applications.

Features:

- High frequency Intel Xeon E5-2686 v4 (Broadwell) processors
- High-performance NVIDIA K80 GPUs, each with 2,496 parallel processing cores and 12GiB of GPU memory
- Supports GPUDirect™ for peer-to-peer GPU communications

Model	GPUs	vCPU	Mem (GiB)	GPU Memory (GiB)
p2.xlarge	1	4	61	12
p2.8xlarge	8	32	488	96
p2.16xlarge	16	64	732	192

GPU Instances - Current Generation

p2.xlarge	4	12	61	EBS Only	\$0.9 per Hour
p2.8xlarge	32	94	488	EBS Only	\$7.2 per Hour
p2.16xlarge	64	188	732	EBS Only	\$14.4 per Hour

P2

P2 instances are intended for general-purpose GPU compute applications.

Features:

- High frequency Intel Xeon E5-2686 v4 (Broadwell) processors
- High-performance NVIDIA K80 GPUs, each with 2,496 parallel processing cores and 12GiB of GPU memory
- Supports GPUDirect™ for peer-to-peer GPU communications

Model	GPUs	vCPU	Mem (GiB)	GPU Memory (GiB)
p2.xlarge	1	4	61	12
p2.8xlarge	8	32	488	96
p2.16xlarge	16	64	732	192

GPU Instances - Current Generation

p2.xlarge	4	12	61	EBS Only	\$0.9 per Hour
p2.8xlarge	32	94	488	EBS Only	\$7.2 per Hour
p2.16xlarge	64	188	732	EBS Only	\$14.4 per Hour

 Search or jump to... / Pull requests Issues Marketplace Explore 

 [rikturr / aws-ml-experimenter](#)  2  1

[Code](#)  0  0  0   

Framework for running ML models on Amazon EC2 and analyzing results with Shiny

scikit-learn aws aws-s3 aws-ec2 boto3 keras tensorflow gpu shiny r python [Manage topics](#)

 23 commits  2 branches  0 releases  1 contributor  MI

Branch: master ▾ [New pull request](#) [Create new file](#) [Upload files](#) [Find file](#) [Clone or download](#)

 Aaron Richter update readme Latest commit d339601 on ...

File	Description	Commits
dashboard	add plot to dashboard	4
examples	update slides	11
experiments	couple tweaks to experiment code	4

<https://github.com/rikturr/aws-ml-experimenter>

<https://youtu.be/JIasG2DfDVc>

Use your libraries

Classification and regression

This page covers algorithms for Classification and Regression. It also includes specific classes of algorithms, such as linear methods, trees, and ensembles.

Table of Contents

- Classification
 - Logistic regression
 - Binomial logistic regression
 - Multinomial logistic regression
 - Decision tree classifier
 - Random forest classifier
 - Gradient-boosted tree classifier
 - Multilayer perceptron classifier
 - Linear Support Vector Machine
 - One-vs-Rest classifier (a.k.a. One-vs-All)
 - Naive Bayes
- Regression
 - Linear regression
 - Generalized linear regression
 - Available families
 - Decision tree regression
 - Random forest regression
 - Gradient-boosted tree regression
 - Survival regression
 - Isotonic regression
- Linear methods
- Decision trees
 - Inputs and Outputs
 - Input Columns
 - Output Columns
- Tree Ensembles
 - Random Forests
 - Inputs and Outputs
 - Input Columns

Use your libraries



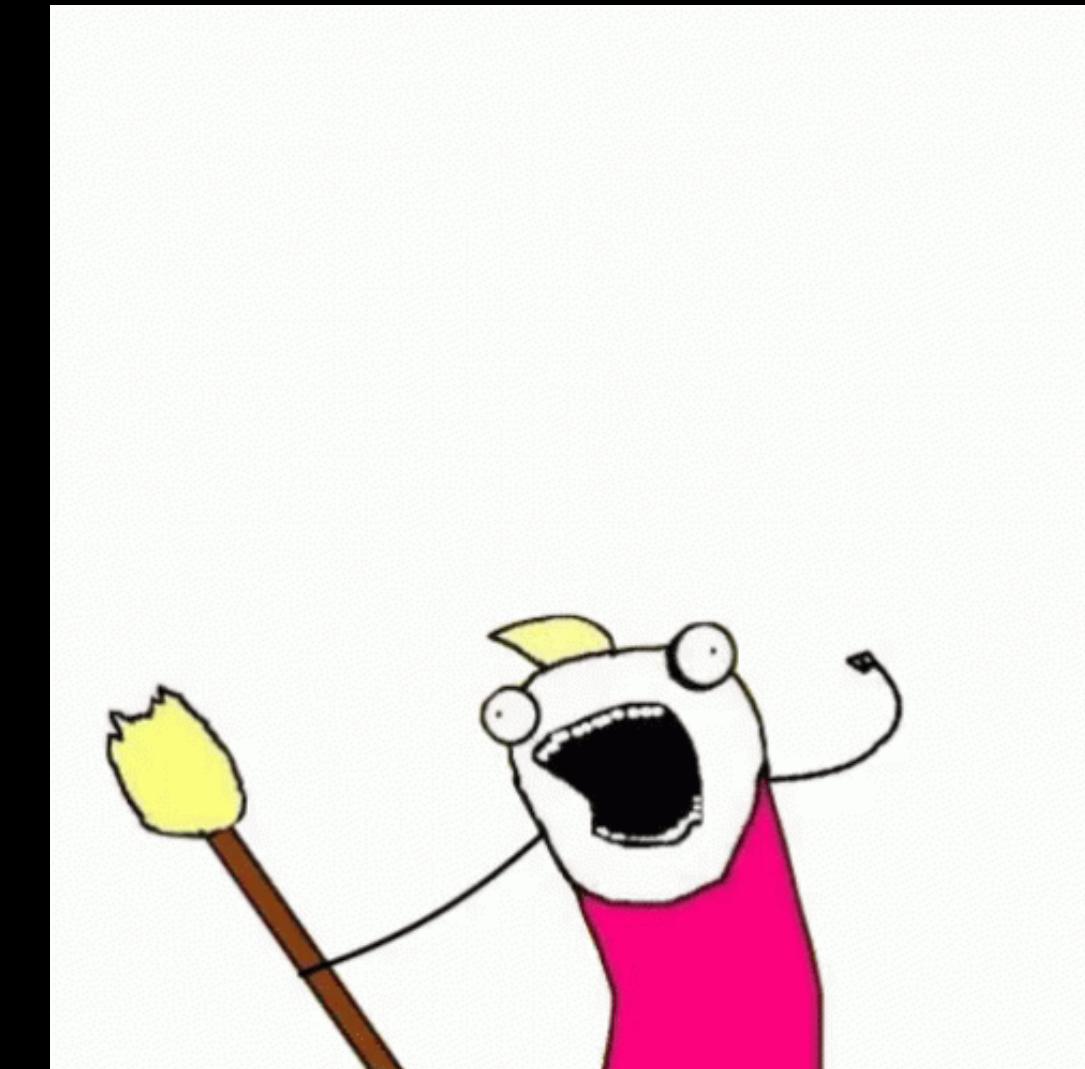
1. Supervised learning

1.1. Generalized Linear Models

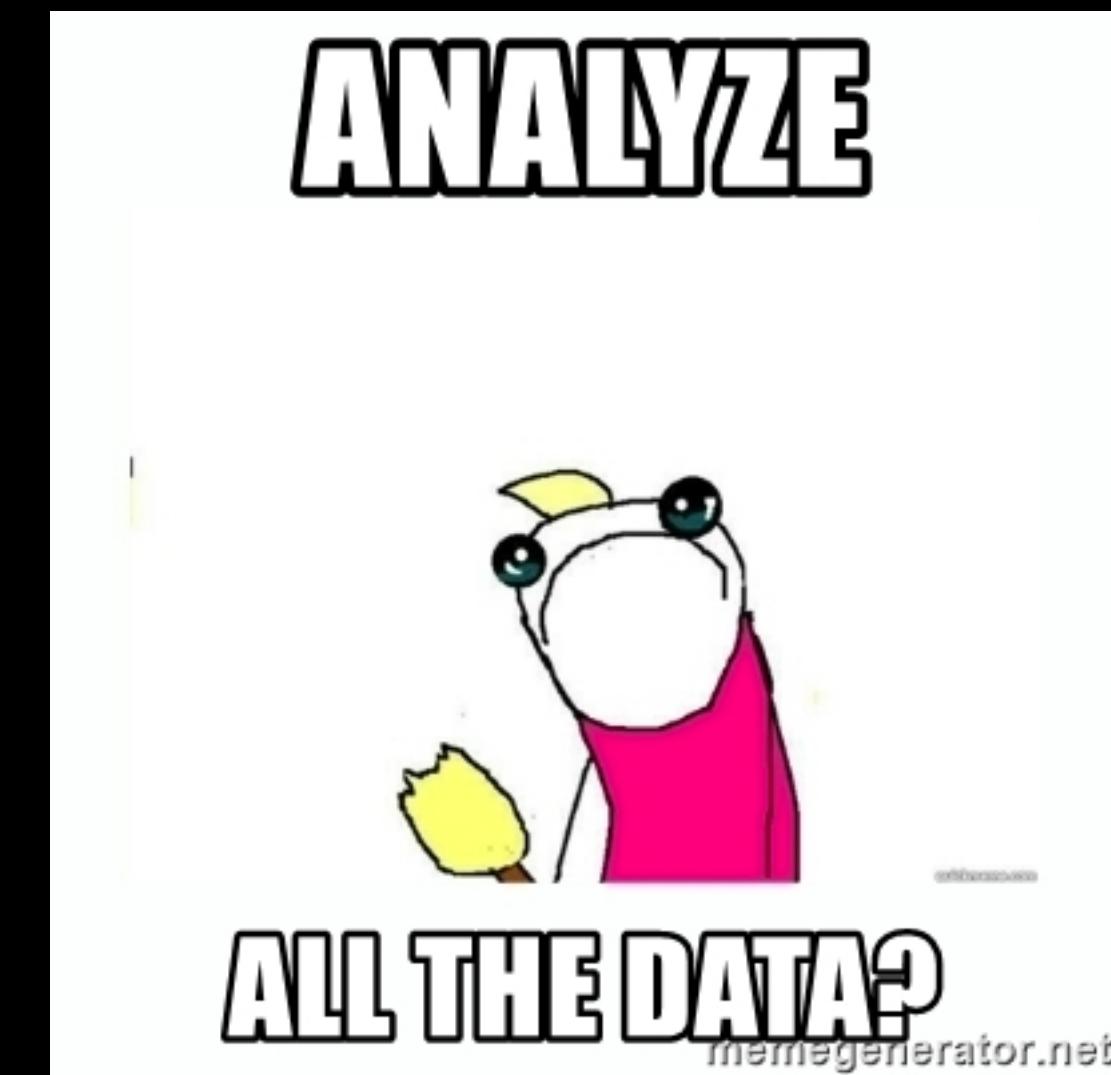
- 1.1.1. Ordinary Least Squares
 - 1.1.1.1. Ordinary Least Squares Complexity
- 1.1.2. Ridge Regression
 - 1.1.2.1. Ridge Complexity
 - 1.1.2.2. Setting the regularization parameter via generalized Cross-Validation
- 1.1.3. Lasso
 - 1.1.3.1. Setting regularization parameter
 - 1.1.3.1.1. Using cross-validation
 - 1.1.3.1.2. Information-criteria based model selection
 - 1.1.3.1.3. Comparison with the regularization parameter of SVM
- 1.1.4. Multi-task Lasso
- 1.1.5. Elastic Net
- 1.1.6. Multi-task Elastic Net
- 1.1.7. Least Angle Regression
- 1.1.8. LARS Lasso
 - 1.1.8.1. Mathematical formulation
- 1.1.9. Orthogonal Matching Pursuit (OMP)
- 1.1.10. Bayesian Regression
 - 1.1.10.1. Bayesian Ridge Regression
 - 1.1.10.2. Automatic Relevance Determination - ARD
- 1.1.11. Logistic regression
- 1.1.12. Stochastic Gradient Descent - SGD
- 1.1.13. Perceptron
- 1.1.14. Passive Aggressive Algorithms

Do you need all the data?

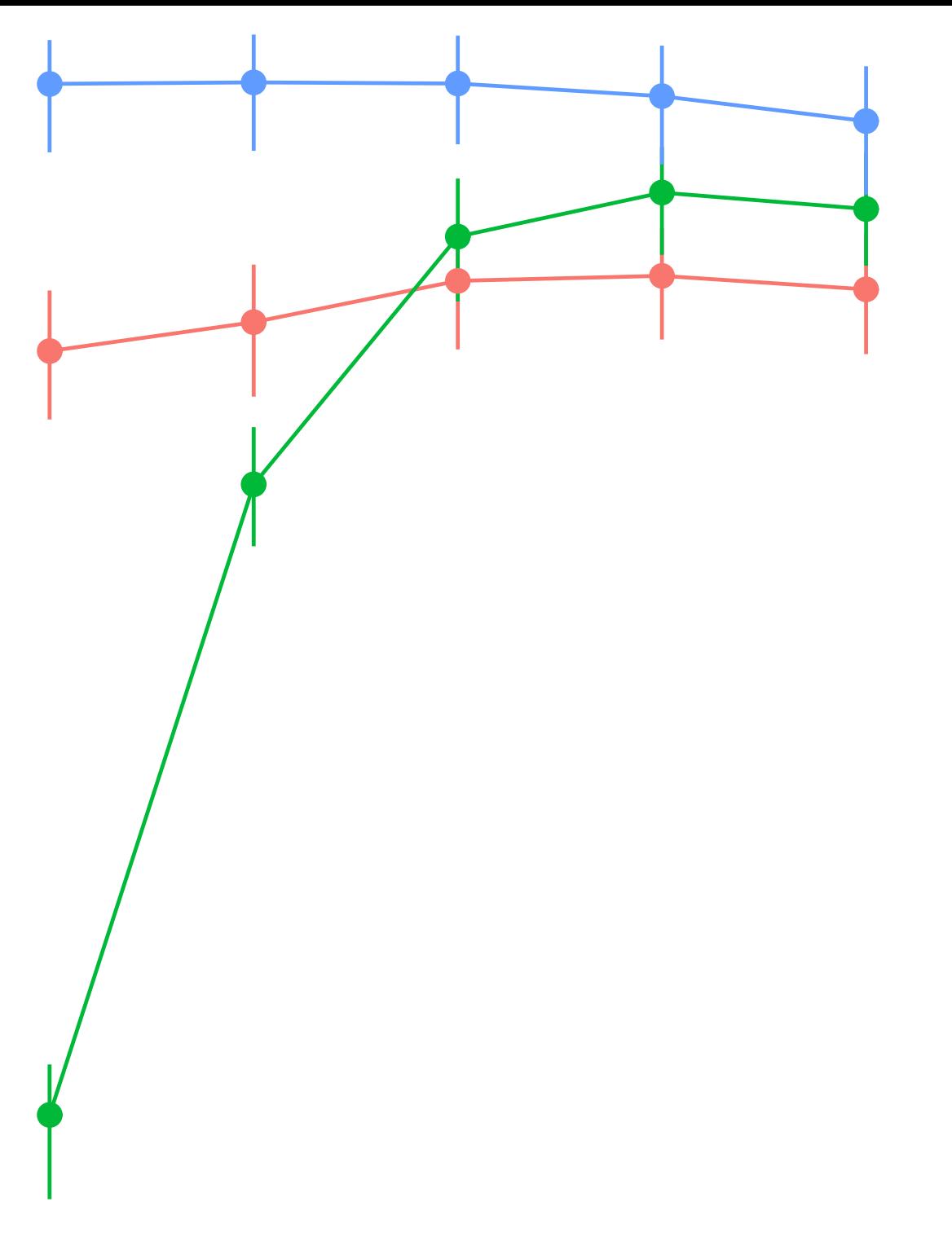
Do you need all the data?



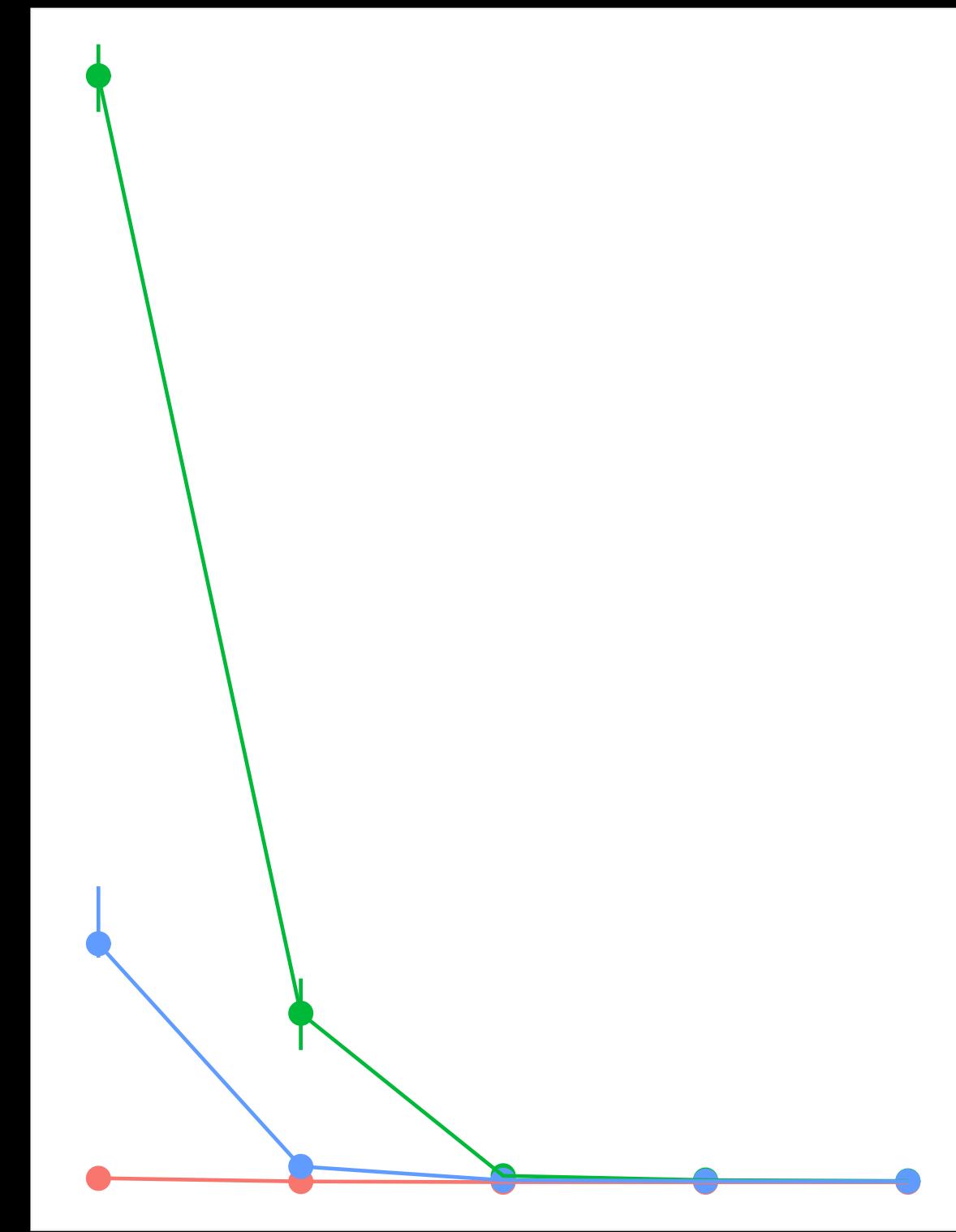
Do you need all the data?



AUC



Cost



File formats are key

File formats are key

database excel csv

parquet feather hdf5

```
In [136]: 1 !ls -alh 2015_partB_miami.txt
-rw-r--r-- 1 aaron.richter staff 11M Jan 20 2018 2015_partB_miami.txt

In [140]: 1 df = pd.read_csv('2015_partB_miami.txt', sep='\t')
2 df.to_parquet('2015_partB_miami.parquet')

In [141]: 1 !ls -alh 2015_partB_miami.parquet
-rw-r--r-- 1 aaron.richter staff 2.7M Jan 8 18:44 2015_partB_miami.parquet
```

File formats are key

database excel csv

parquet feather hdf5

<https://gist.github.com/rikturr/888c402e788e560e38380a7f6ac352d7>

Sparse matrices

Sparse matrices

scipy pandas sklearn

libsvm arff

```
6 one_hot

Out[122]: array([[ 1.00301111e+09,  0.00000000e+00,  0.00000000e+00, ...,
                   0.00000000e+00,  0.00000000e+00,  0.00000000e+00],
                   [ 1.00301711e+09,  0.00000000e+00,  0.00000000e+00, ...,
                   0.00000000e+00,  0.00000000e+00,  0.00000000e+00],
                   [ 1.00302358e+09,  0.00000000e+00,  0.00000000e+00, ...,
                   0.00000000e+00,  0.00000000e+00,  0.00000000e+00],
                   ...,
                   [ 1.99297706e+09,  0.00000000e+00,  0.00000000e+00, ...,
                   0.00000000e+00,  0.00000000e+00,  0.00000000e+00],
                   [ 1.99299449e+09,  0.00000000e+00,  0.00000000e+00, ...,
                   0.00000000e+00,  0.00000000e+00,  0.00000000e+00],
                   [ 1.99299752e+09,  0.00000000e+00,  0.00000000e+00, ...,
                   0.00000000e+00,  0.00000000e+00,  0.00000000e+00]]))

In [123]: 1 one_hot.shape, one_hot.shape[0] * one_hot.shape[1]

Out[123]: ((5740, 2185), 12541900)

In [124]: 1 np.count_nonzero(one_hot)

Out[124]: 56550
```

```
In [126]: 1 import scipy.sparse as sp  
2  
3 one_hot_sparse = sp.csc_matrix(one_hot)  
4 one_hot_sparse
```

```
Out[126]: <5740x2185 sparse matrix of type '<class 'numpy.float64'>'  
with 56550 stored elements in Compressed Sparse Column format>
```

```
In [130]: 1 np.save('dense.npy', one_hot)  
2 sp.save_npz('sparse.npz', one_hot_sparse)
```

```
In [132]: 1 !ls -alh dense.npy  
  
-rw-r--r-- 1 aaron.richter staff 96M Jan 8 18:36 dense.npy
```

```
In [133]: 1 !ls -alh sparse.npz  
  
-rw-r--r-- 1 aaron.richter staff 186K Jan 8 18:36 sparse.npz
```

The cloud is your friend

Your laptop is stronger than you think

Use your cores

Use your cores

multiprocessing pyspark dask RAPIDS

snow multidplyr sparklyr

Don't be patient

Time is \$\$

Use the internet

<https://lmgtfy.com/>

Cluster computing has its place

The cloud is your friend

Use your libraries

Do you need all the data?

File formats are key

Sparse matrices

Your laptop is stronger than you think

Use your cores

Don't be patient

Time is \$\$

Use the internet



Aaron Richter

@rikturr

Get the code!

<https://github.com/rikturr/pydata-your-data-fits-in-ram>