Designing a Machine Learning Algorithm to Conserve the RDA's Computing Resources

Jordan DuBeau,
Ph.D. Student,
University of Colorado Boulder

Mentors: Riley Conroy and Brian Vanderwende

July 28, 2021

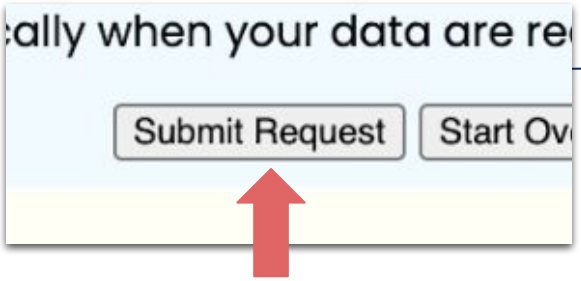
This material is based upon work supported by the National Center for Atmospheric Research, which is a major facility sponsored by the National Science Foundation under Cooperative Agreement No. 1852977.

The RDA and its website

- RDA stands for **Research Data Archive.**
- The website, rda.ucar.edu, holds a large amount of **data** for **scientific use.**
- The data are organized into a variety of **diverse datasets** (different formats, different file sizes, different layouts)
- Often users want to download just a **subset** of one dataset.
- There's a way to do this on the website!

After the request...

Request submitted



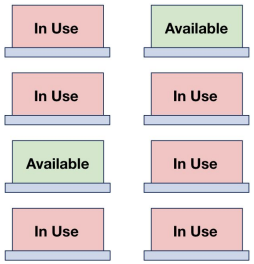
rinfo string formed

```
gui=yes;
dsnum=094.0;
startdate=2015-01-01 00:00;
enddate=2015-12-31 03:00;
parameters=3!7-0.2-1:0.1.0;
product=486;
grid_definition=5;
nlat=42;
slat=36;
wlon=-110;
elon=-102
```

Preparatory program

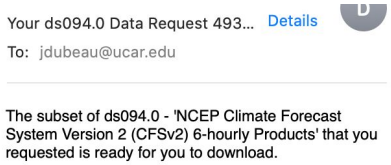
```
if dsnum == 'abc.d':
    req_mem = 2000
    req_time = 12

elif dsnum == 'wxy.z':
    req_mem = mem_function(rinfo)
    req_time = time_function(rinfo)

else:
    ...
```

Workload manager (Slurm or PBS)



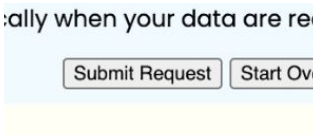
Supercomputer grabs data

Data delivered



After the request...

Request submitted



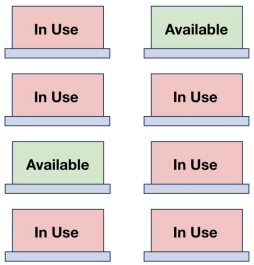
rinfo string formed

gui=yes;
dsnum=094.0;
startdate=2015-01-01 00:00;
enddate=2015-12-31 03:00;
parameters=3!7-0.2-1:0.1.0;
product=486;
grid_definition=5;
nlat=42;
slat=36;
wlon=-110;
elon=-102

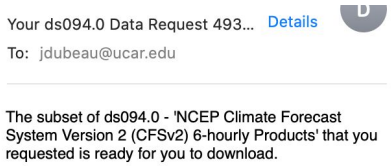
Preparatory program

```
if dsnum == 'abc.d':
    req_mem = 2000
    req_time = 12

elif dsnum == 'wxy.z':
    req_mem = mem_function(rinfo)
    req_time = time_function(rinfo)

else:
    ...
```

Workload manager (Slurm or PBS)

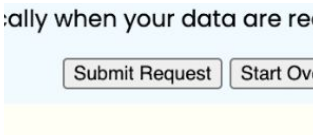


Data delivered



After the request...

Request submitted



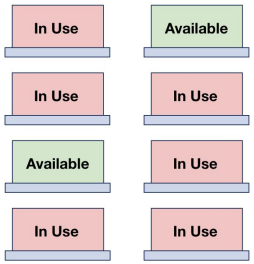
rinfo string formed

```
gui=yes;
dsnum=094.0;
startdate=2015-01-01 00:00;
enddate=2015-12-31 03:00;
parameters=3|7-0.2-1:0.1.0;
product=486;
grid_definition=5;
nlat=42;
slat=36;
wlon=-110;
elon=-102
```

Preparatory program

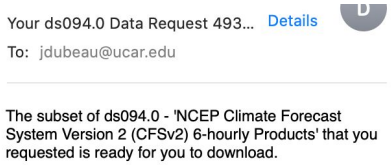
```
if dsnum == 'abc.d':
    req_mem = 2000
    req_time = 12

elif dsnum == 'wxy.z':
    req_mem = mem_function(rinfo)
    req_time = time_function(rinfo)

else:
    ...
```

Workload manager (Slurm or PBS)



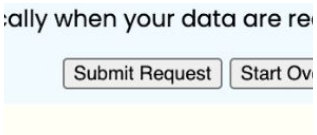
Supercomputer grabs data

Data delivered



After the request...

Request submitted



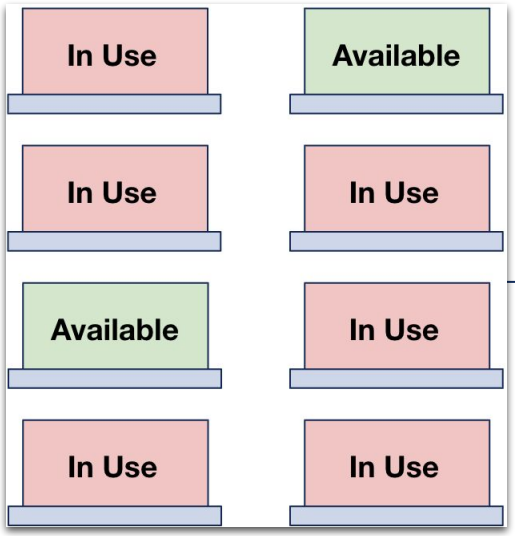
rinfo string formed

```
gui=yes;
dsnum=094.0;
startdate=2015-01-01 00:00;
enddate=2015-12-31 03:00;
parameters=3!7-0.2-1:0.1.0;
product=486;
grid_definition=5;
nlat=42;
slat=36;
wlon=-110;
elon=-102
```

Preparatory program

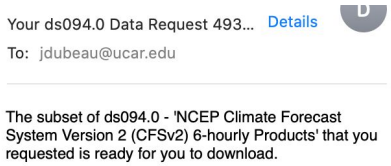
```
if dsnum == 'abc.d':
    req_mem = 2000
    req_time = 12

elif dsnum == 'wxy.z':
    req_mem = mem_function(rinfo)
    req_time = time_function(rinfo)

else:
    ...
```

Workload manager (Slurm or PBS)



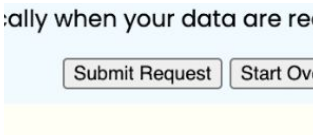
Supercomputer grabs data

Data delivered



After the request...

Request submitted



rinfo string formed

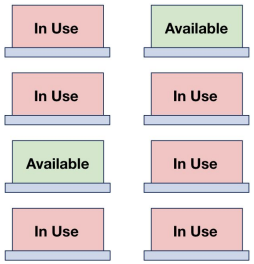
```
gui=yes;
dsnum=094.0;
startdate=2015-01-01 00:00;
enddate=2015-12-31 03:00;
parameters=3!7-0.2-1:0.1.0;
product=486;
grid_definition=5;
nlat=42;
slat=36;
wlon=-110;
elon=-102
```

Preparatory program

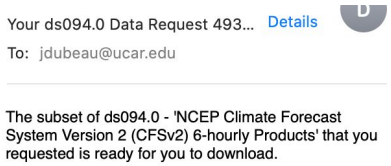
```
if dsnum == 'abc.d':
    req_mem = 2000
    req_time = 12

elif dsnum == 'wxy.z':
    req_mem = mem_function(rinfo)
    req_time = time_function(rinfo)

else:
    ...
```

Supercomputer grabs data

Workload manager (Slurm or P

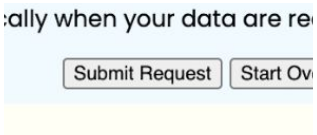


Data delivered



After the request...

Request submitted



rinfo string formed

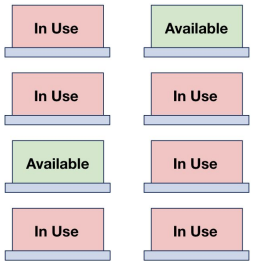
```
gui=yes;
dsnum=094.0;
startdate=2015-01-01 00:00;
enddate=2015-12-31 03:00;
parameters=3!7-0.2-1:0.1.0;
product=486;
grid_definition=5;
nlat=42;
slat=36;
wlon=-110;
elon=-102
```

Preparatory program

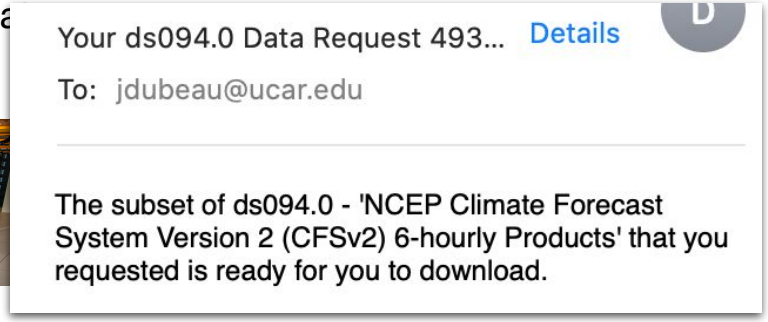
```
if dsnum == 'abc.d':
    req_mem = 2000
    req_time = 12

elif dsnum == 'wxy.z':
    req_mem = mem_function(rinfo)
    req_time = time_function(rinfo)

else:
    ...
```

Data delivered

Workload manager (Slurm or PBS)



Supercomputer grabs da



The problem

Our estimates for how much **memory** and **time** these subset requests will take are often **way off.**

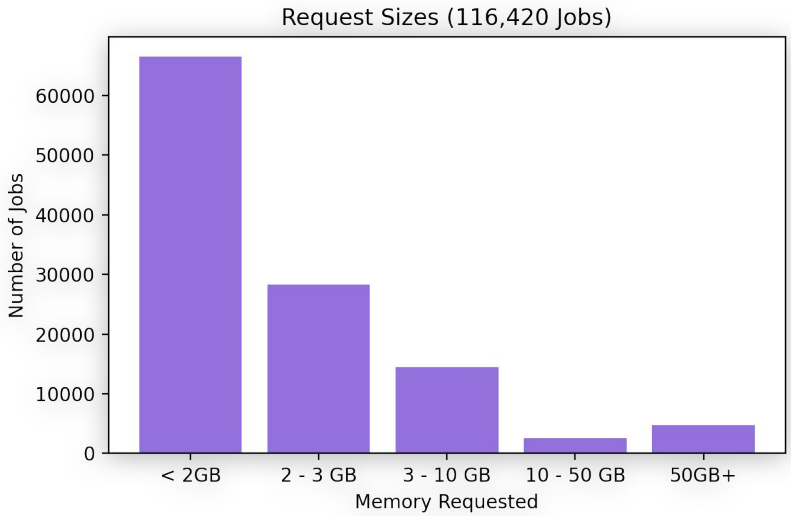
In other words, the **preparatory program** is too **simple**.
Our goal is to improve it.

We will use **machine learning** to do this.

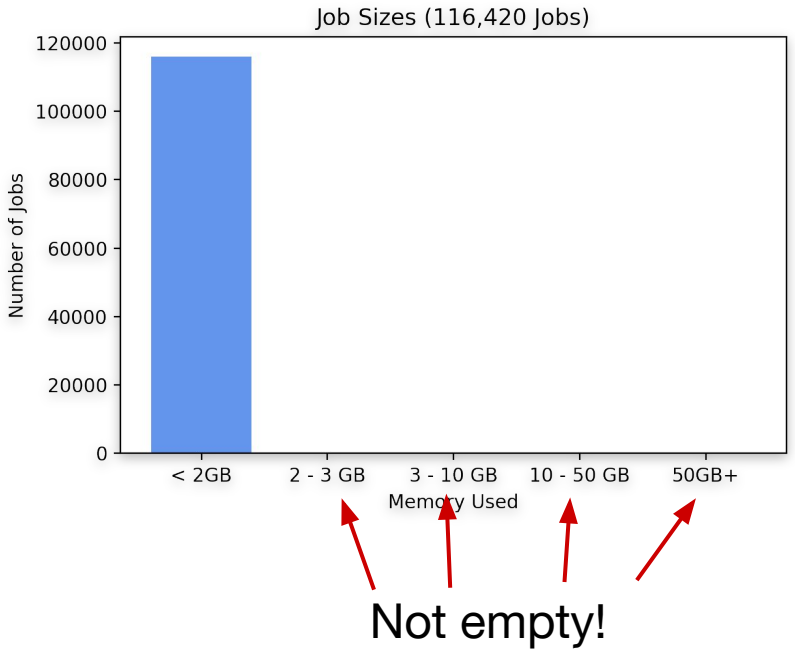
How far off are our estimates?

We collected data on 116,420 jobs ranging from Sep. 2020 - Apr. 2021.

Here's how often we **requested** certain amounts of memory...

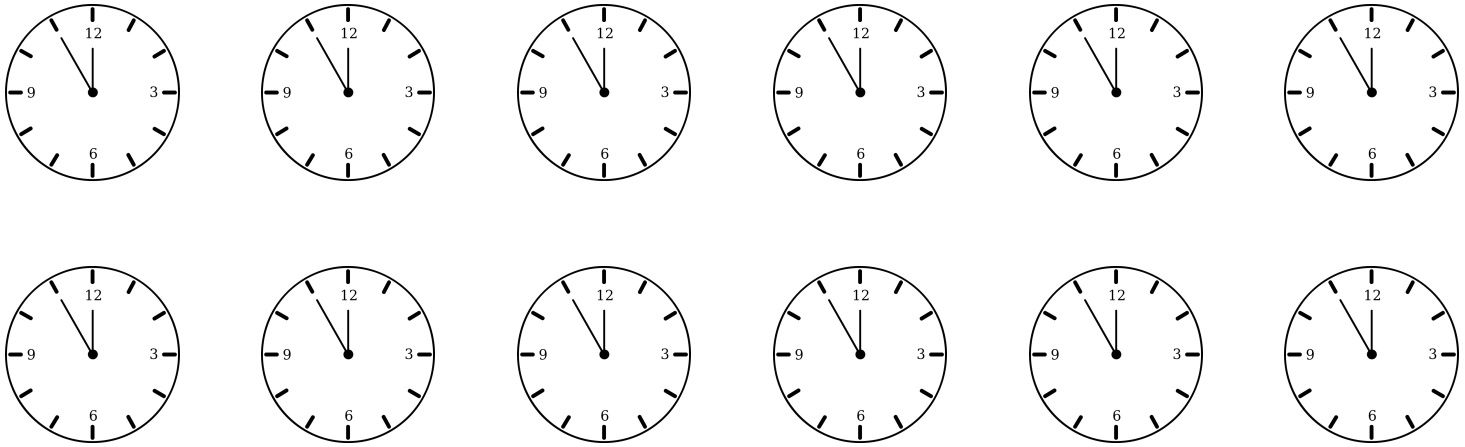


And how often we **used** those amounts of memory.

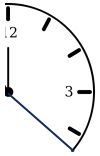


Our time estimates

Requested time for every job: **12 hours**



Actual time per job on average: **22 minutes**



(Not as important as predicting memory usage)

Why machine learning?

- It would probably be possible to **precisely calculate** how much memory these requests will need.
- But this would require **significantly more** specialist involvement.
- Plus, it would be highly dependent on the characteristics of the **dataset...**
- So when new datasets are **added** to the RDA, we would have to change the program every time.

Applying machine learning to our data

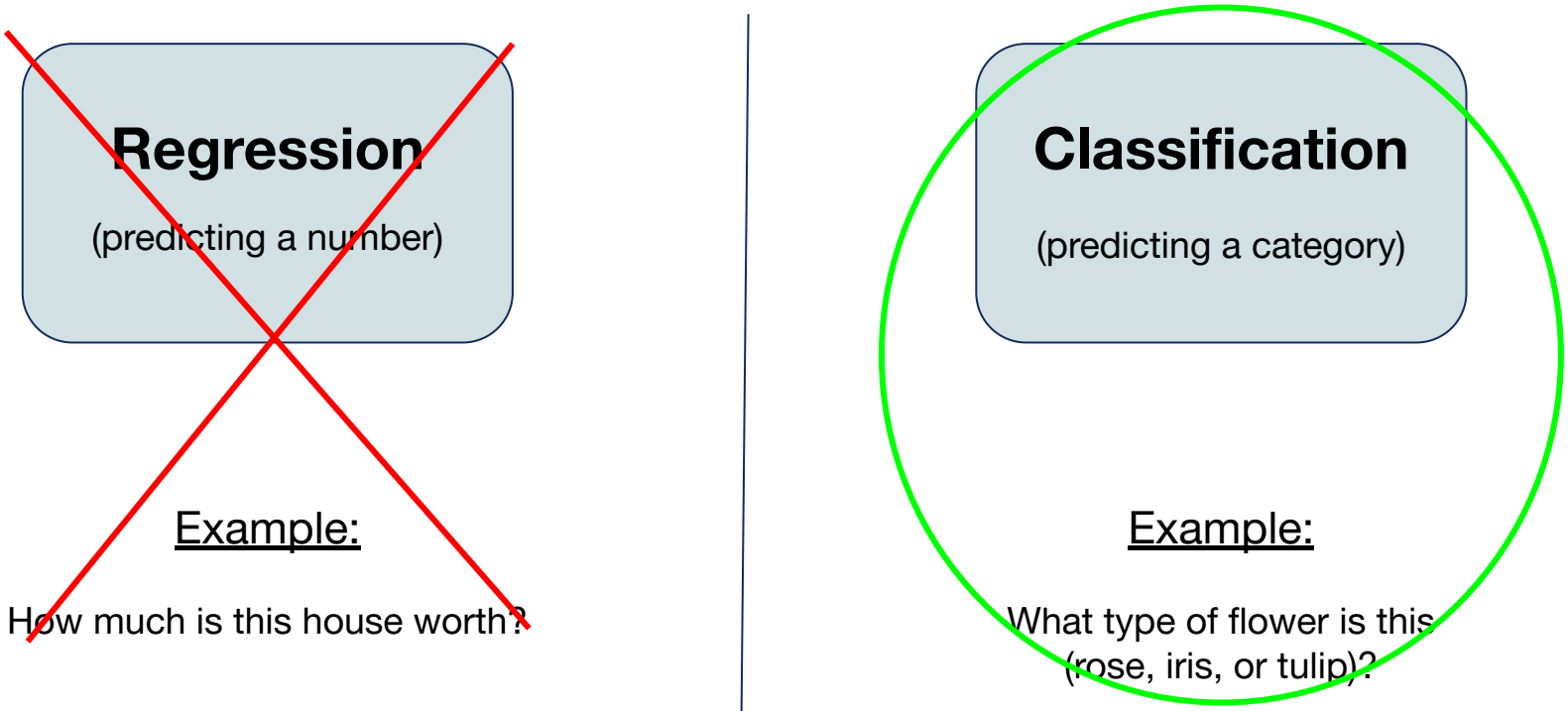
Training Data

Request Timespan	Request Area	# of Params	Converted?	Mem	Time
0y 0d 22h 59m	2.24	8	False	34.1 MB	1m 12s
17y 4d 21h 0m	344.0	29	False	44.8 MB	1m 7s
0y 30d 0h 0m	68.0	2	True	1452 MB	1m 8s
5y 335d 22h 59m	25.0	16	False	338.8 MB	1h 6m 9s

New Data

Request Timespan	Request Area	# of Params	Converted?	ML program	Mem	Time
1y 20d 5h 0m	38.0	12	True	⇒	?	?

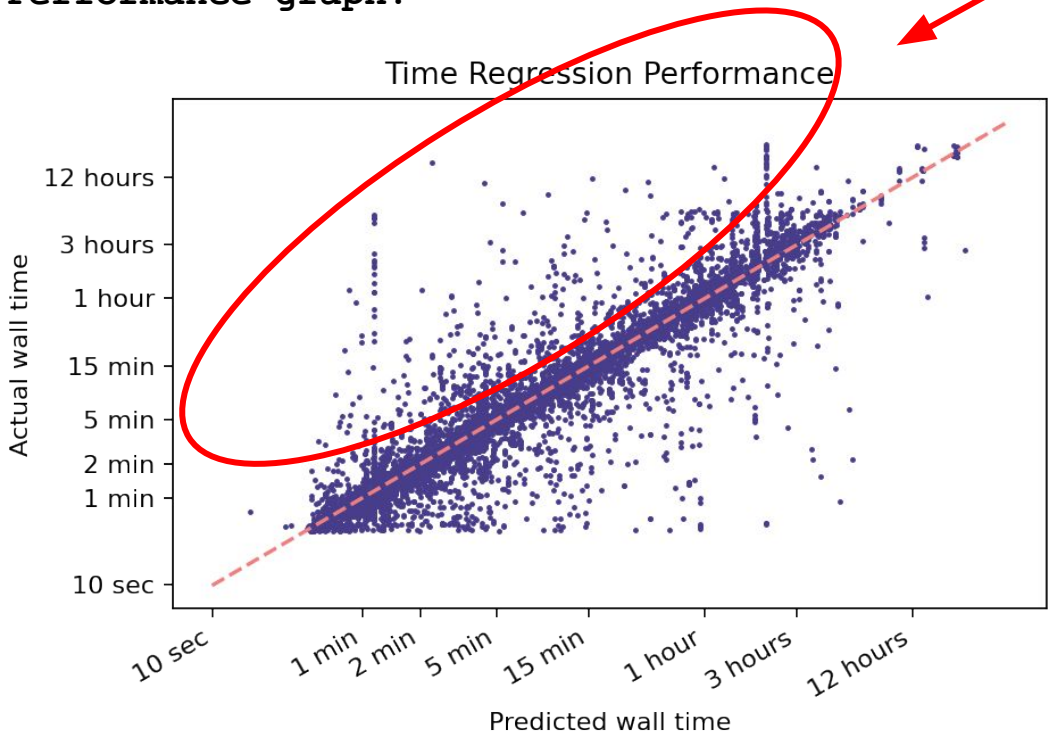
Two major types of algorithm



The problem with regression

Here is our best attempt at regression for predicting wall time:

```
Training score: 0.9447
Testing score:  0.8997
Performance graph:
```



These points are a **big problem!**

How can classification help?

The entries we care **most** about, unfortunately, are also the **rarest.**

Regression algorithms will naturally tend to **ignore** these.

But if we split the entries into **categories**...

Memory

Category 0	0MB	- 50MB	(22965 entries)
Category 1	50MB	- 100MB	(11126 entries)
	.		
	.		
	.		
Category 7	10GB	- 20GB	(33 entries)
Category 8	20GB	- 50GB	(22 entries)

We can tell the algorithm exactly how much **attention** to pay to each **category**.

Another benefit of classification

- The models don't just give us the predicted category -- they give us a **list** showing the **probability** of being in each category.

Input

Request Timespan	Request Area	# of Params	Converted?
1y 20d 5h 0m	38.0	12	True

Output

0	1	2	3	4	5	7	8
25%	55%	10%	1%	5%	1%	1%	2%

Thank you to D.J. Gagne for suggesting this strategy!

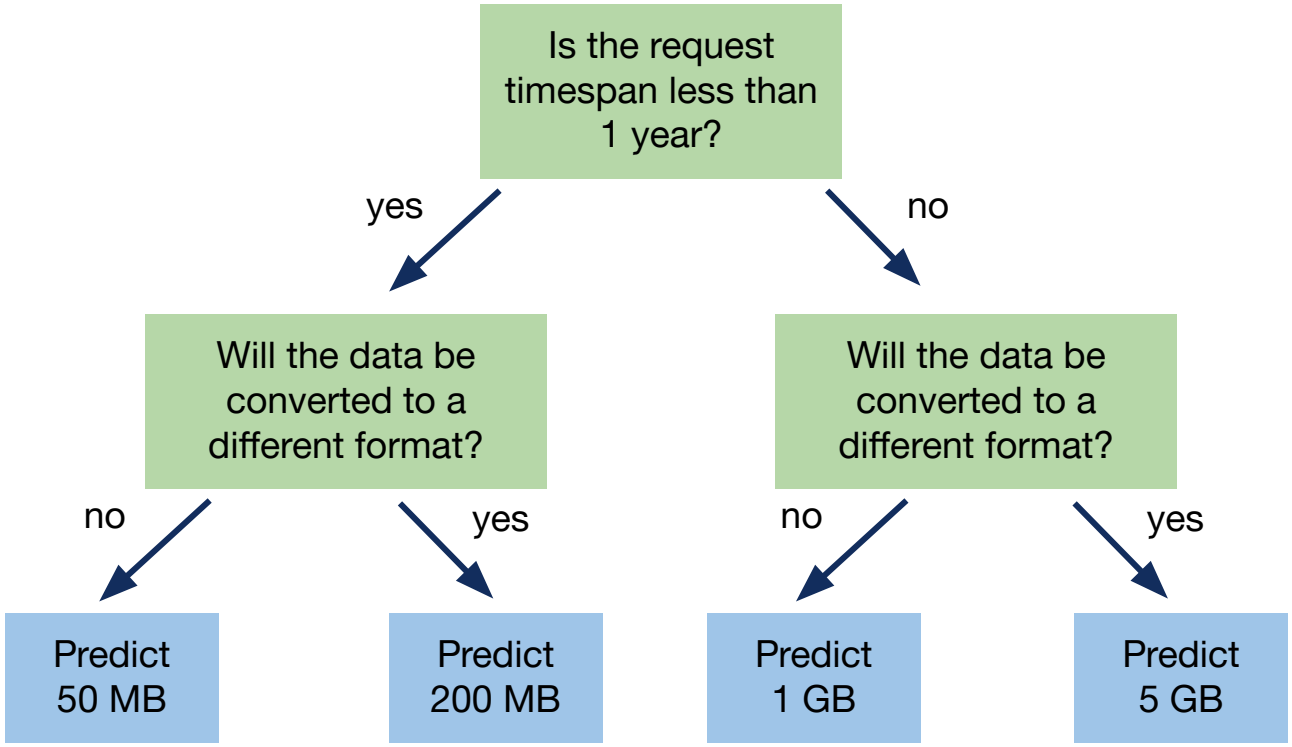
What model to use?

Some options:

- Linear Model (Logistic Regression, etc.) ← Not good enough performance
- Neural Network ← Good performance probably possible, but very difficult
- Decision Tree ← Building block for the two models below
- **Random Forest** ← Best performance!
- **Gradient Boosted Decision Trees** ← Best performance!

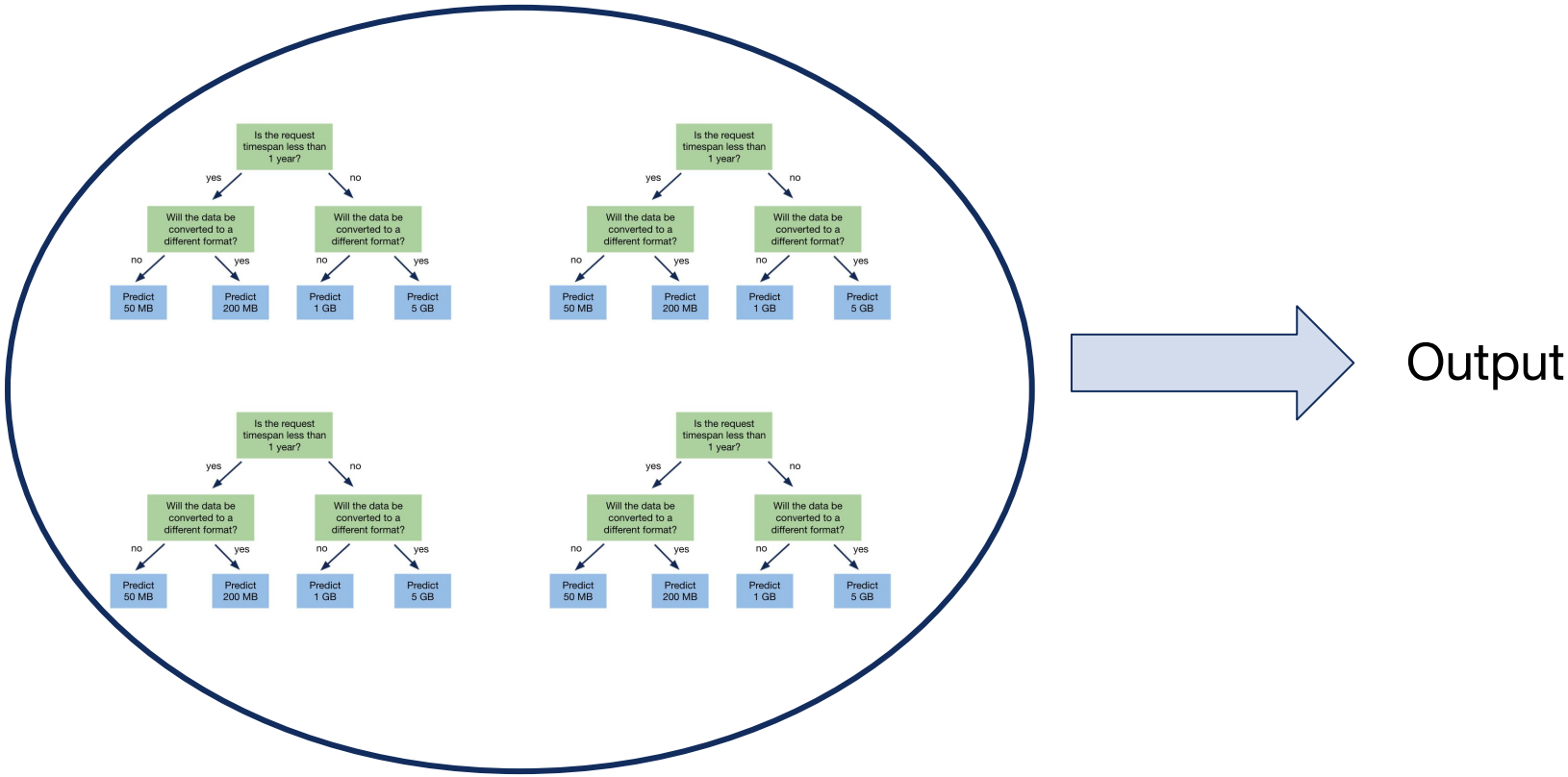
Decision tree / random forest

A decision tree might look like:



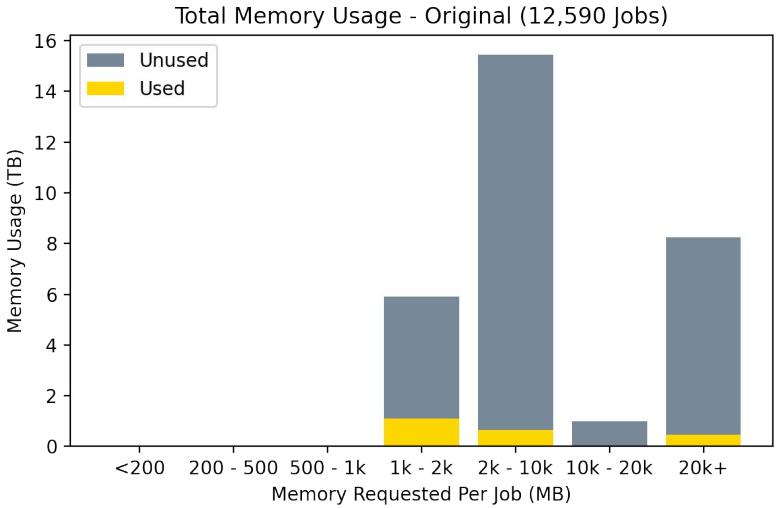
Decision tree / random forest

A random forest is made from a **collection** of decision trees.



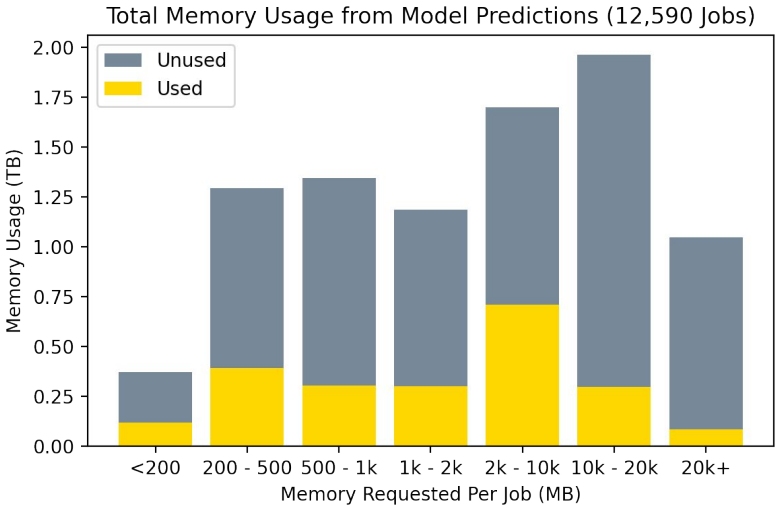
How did we do? (Memory)

On our reserved testing data...



Total requested: 68.48 TB
Total used: 2.51 TB

Total wasted: 65.98 TB



Total requested: 9.96 TB
Total used: 2.51 TB

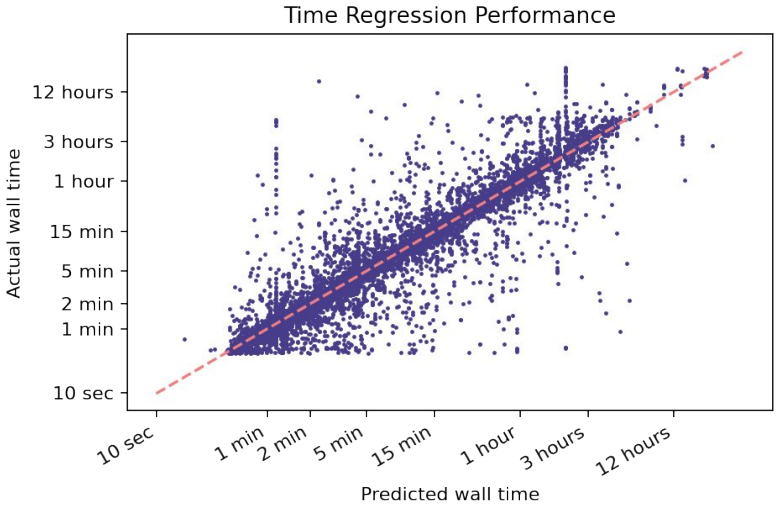
Total wasted: 7.46 TB

88.7% decrease!

We also achieved a **59.7% decrease** in unused time.

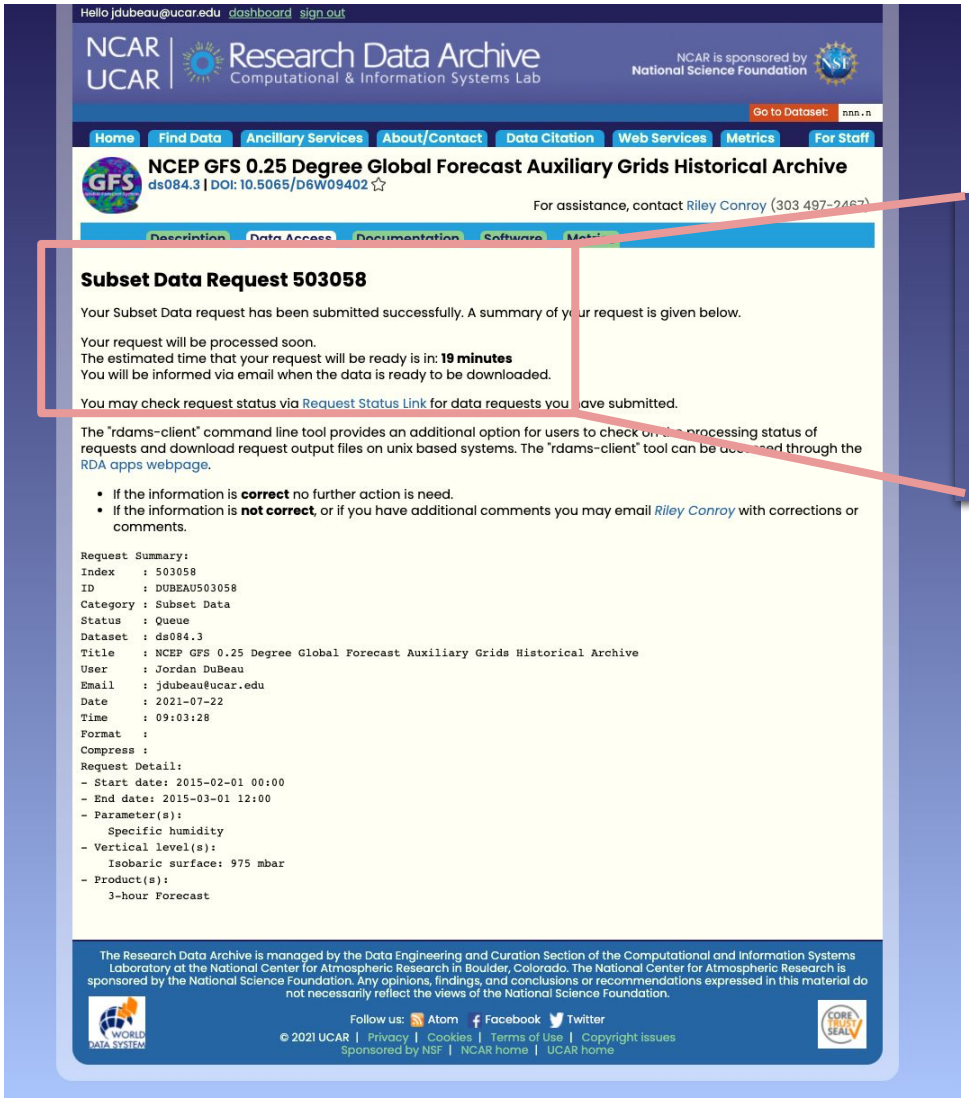
One more thing!

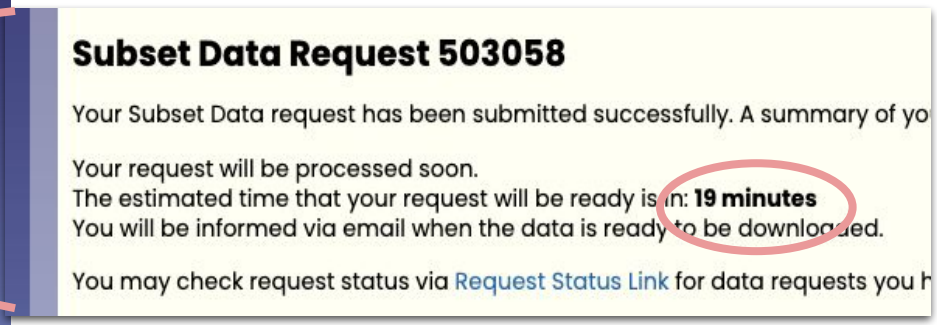
We had already trained that **regression** model for wall time...



We realized that model could still be **useful!**

Regression model in use





Users now have a time estimate for their request!

Thanks for listening!