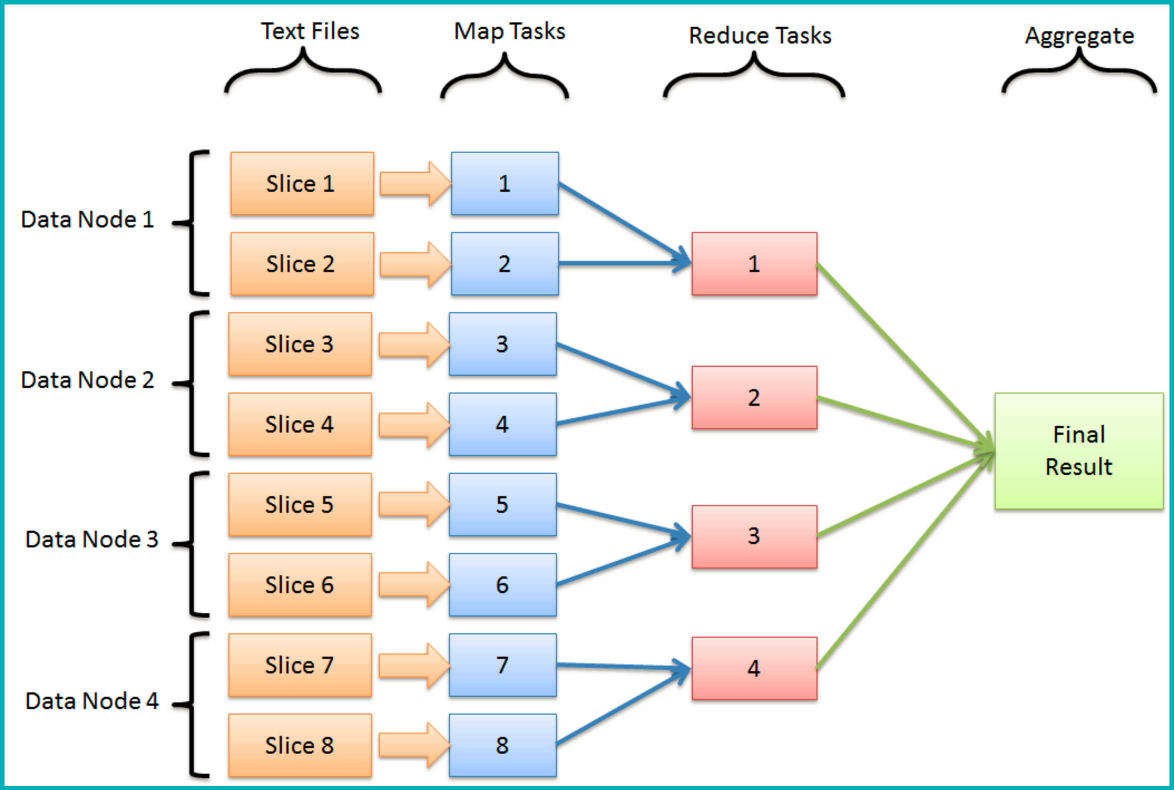
A Big Data Solution: Cluster Computing

Cluster computing

- **Instead of storing the data on one machine, use 100 machines and store 1/100th of the data on each machine**
  - Read data from each machine in parallel
- **For our 1TB of data, the access time is now under 2 minutes**

How do we aggregate the data on each machine to perform an analysis on the entire data set?



This is known as the Map-Reduce Paradigm

A Connected Car example:

```
Username, trip date, miles, vehicle
User1, 1/1/2017, 8, truck
User2, 1/1/2017, 23, car
User3, 1/2/2017, 2, van
User2, 1/2/2017, 34, car
User3, 1/2/2017, 7, van
User1, 1/4/2017, 16, truck
User1, 1/5/2017, 4, truck
```


How many miles has each user driven?

The Map Step

Key, value pairs

User1, 1/1/2017, 8, truck
User2, 1/1/2017, 23, car

User1, 8
User2, 23

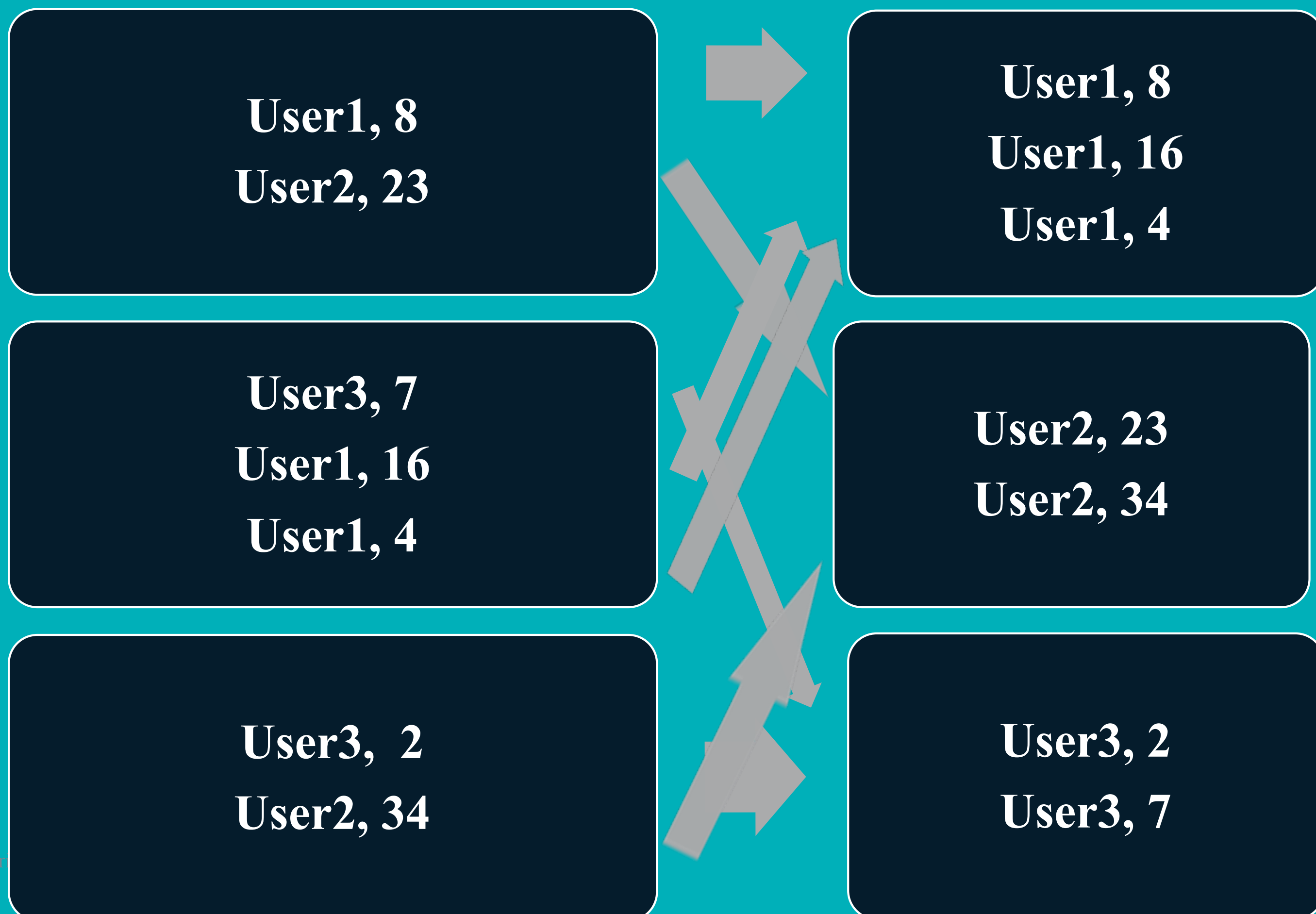
User3, 1/1/2017, 2, van
User2, 1/2/2017, 34, car

User3, 2
User2, 34

User3, 1/2/2017, 7, van
User1, 1/4/2017, 16, truck
User1, 1/5/2017, 4, truck

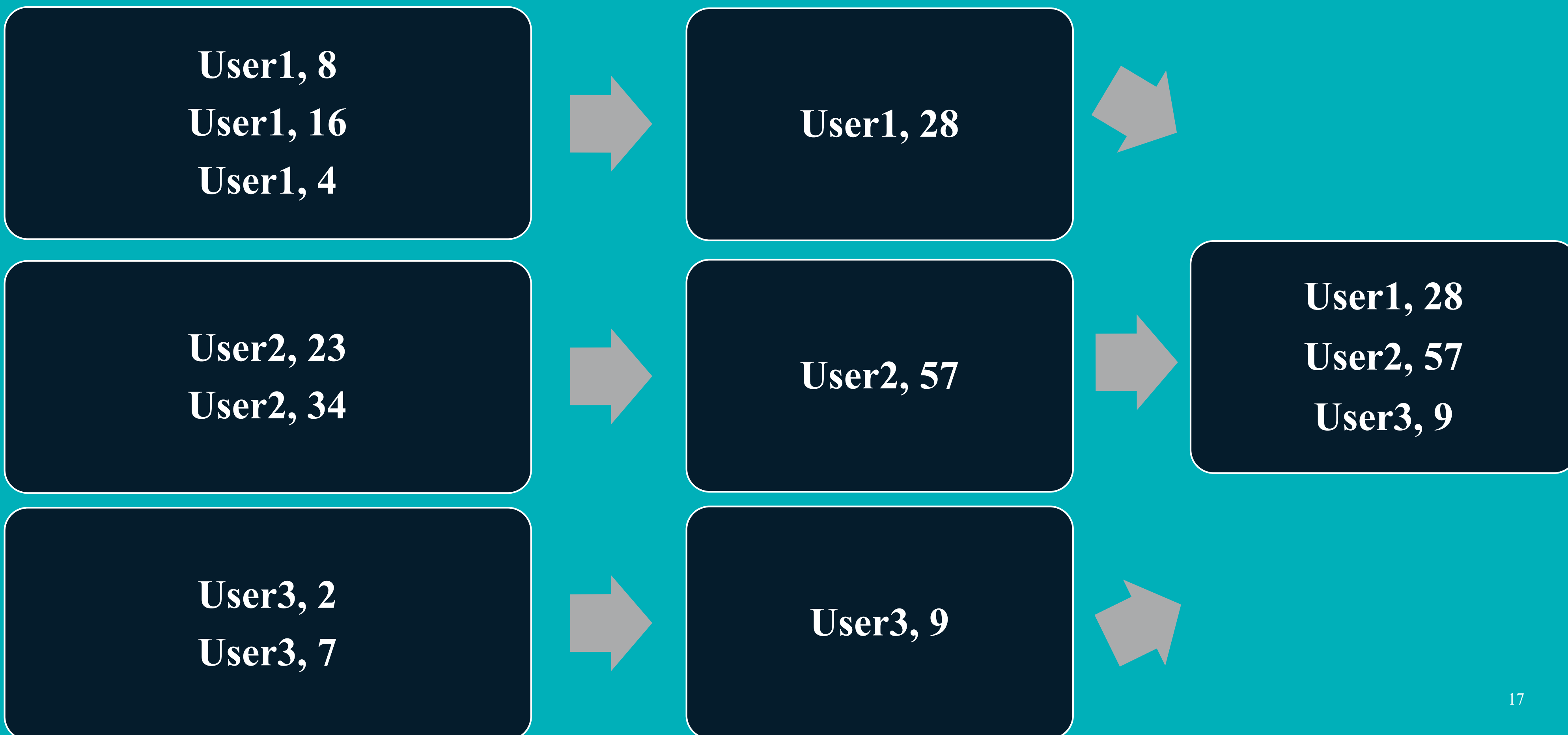
User3, 7
User1, 16
User1, 4

The Shuffle Step



With larger data, some reducing may happen before the shuffle.

The Reduce Step



Tools we use

Machine Learning

Applications of machine learning to connected cars

- Loss models based on driving data
- Crash detection
- Traffic predictions
- Autonomous cars
- Predictive maintenance
- More effective ride sharing

Models we use

- Generalized Linear Models
- Gradient Boosted Machines
- Neural Networks
- Clustering
- Random Forests

Tools we use

python

R

H_2O.ai

DOMINO

TensorFlow

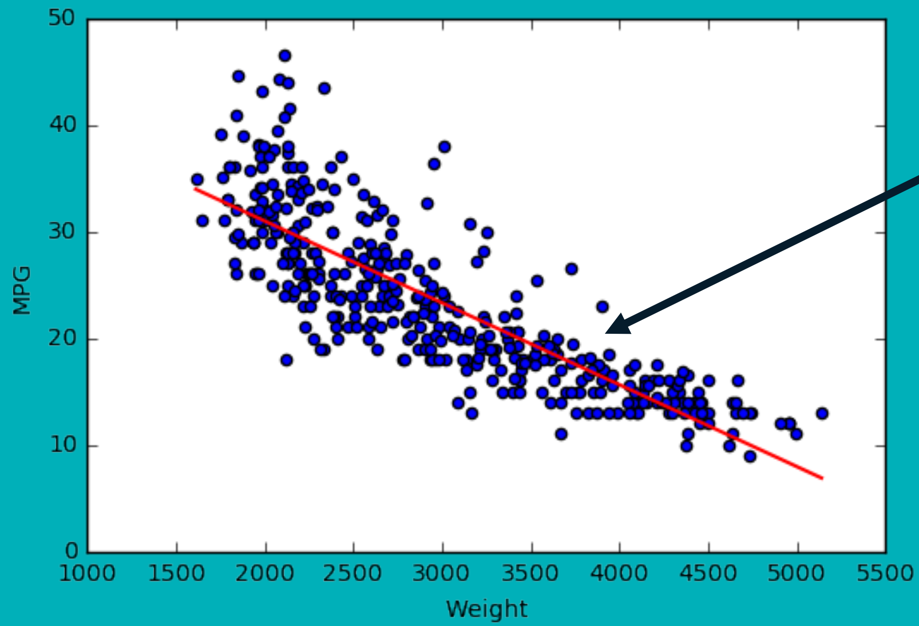
Gradient Boosted Machines

Gradient Boosted Machines

- **Classification/Regression tree based approach**
- **Compared to generalized linear models:**
  - \+ More accurate(generally)
  - \+ Better at finding interactions between variables
  - \+ Able to handle missing values
  - \- Less interpretable
  - \- Slower to train

A "Connected Car" Example

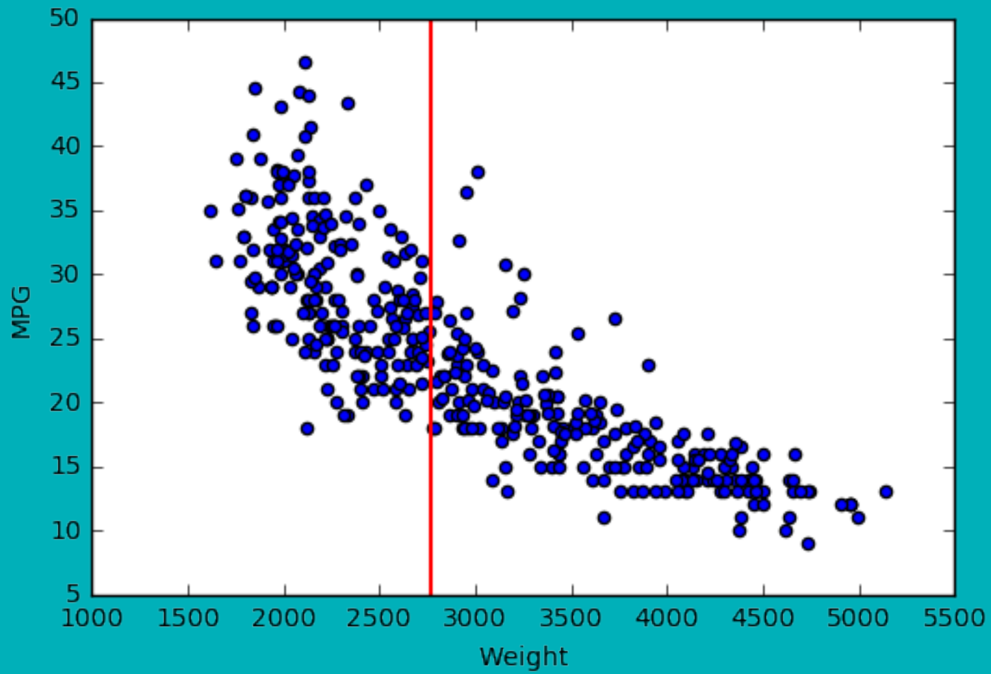
Suppose we want to estimate a car's MPG with its weight

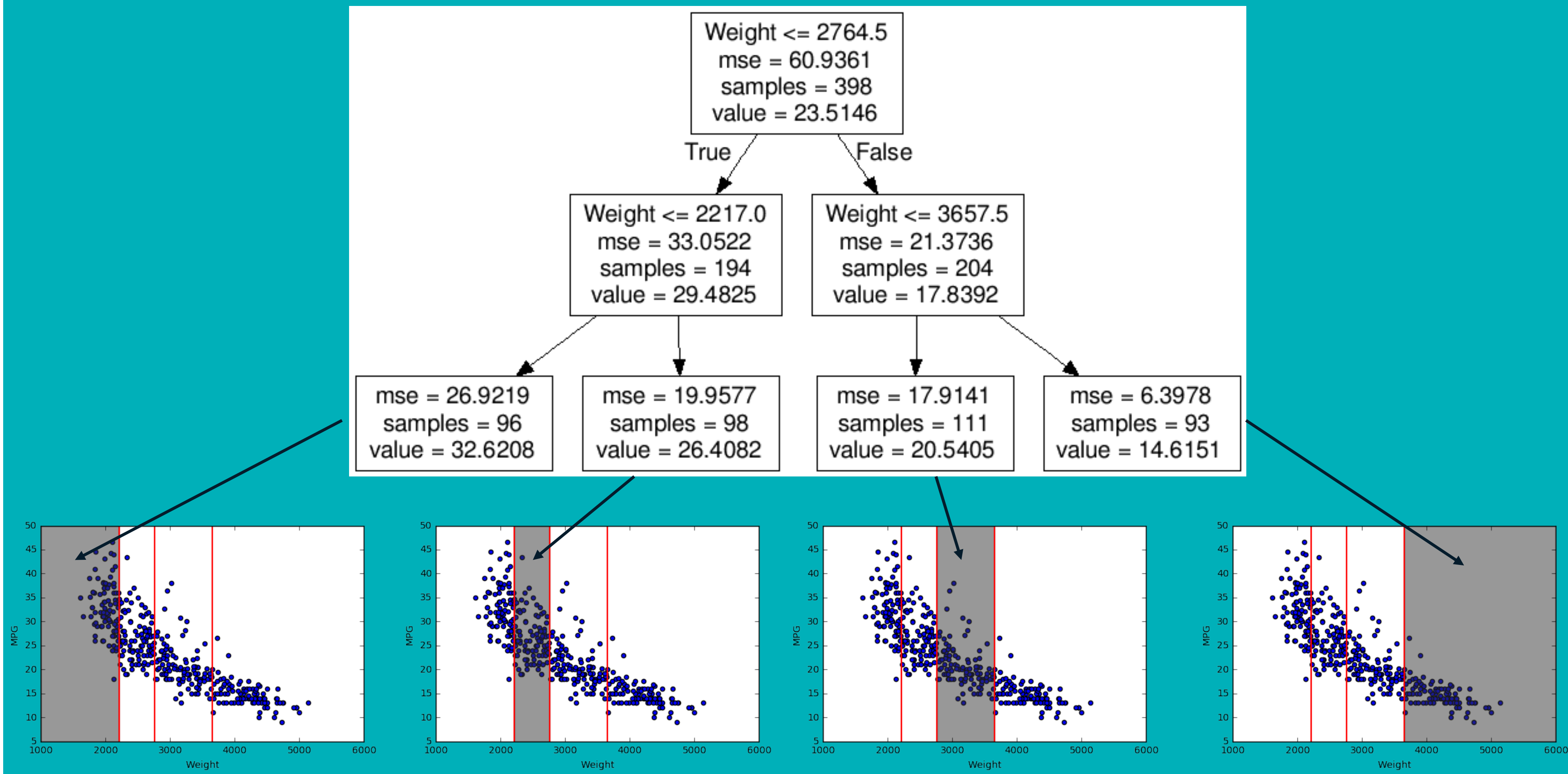


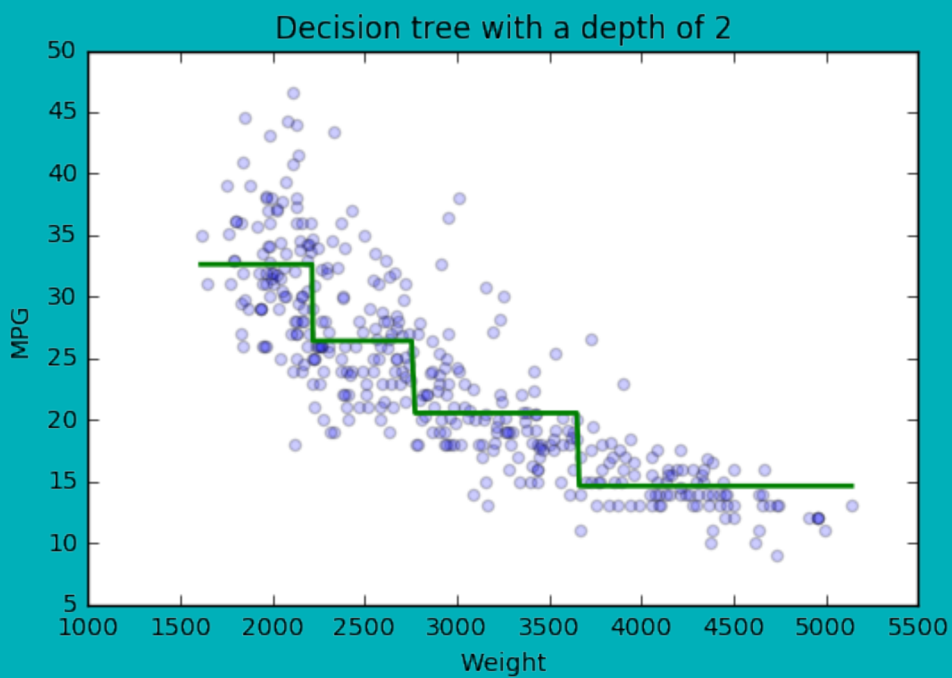
Red line is how we would fit with linear regression

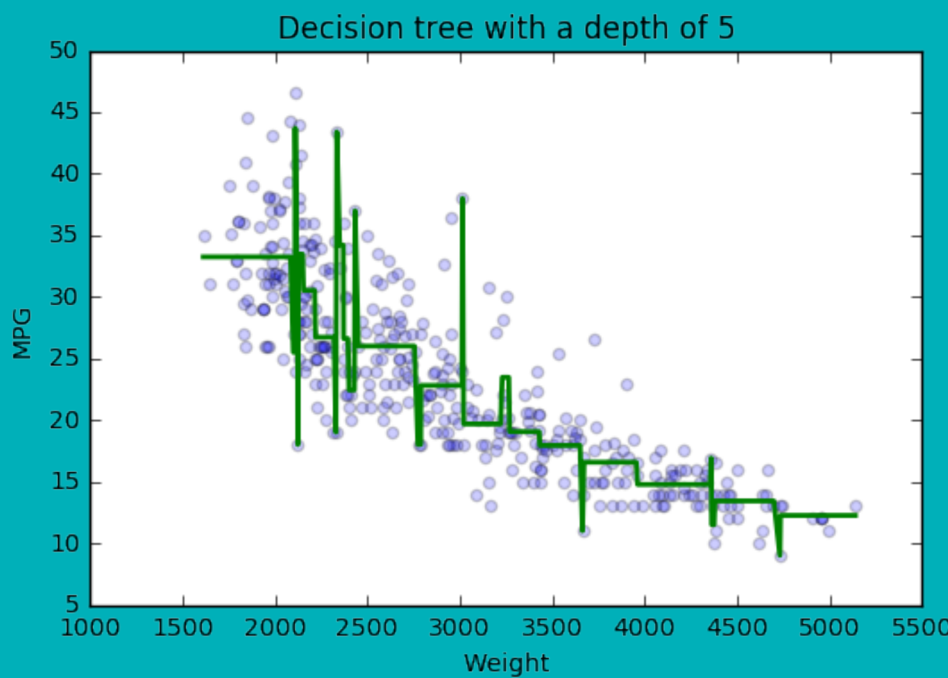
Decision Trees

Choose a split point that minimizes the sum of the errors in both regions




Weight <= 2764.5
mse = 60.9361
samples = 398
value = 23.5146
True
False
Weight <= 2217.0
mse = 33.0522
samples = 194
value = 29.4825
Weight <= 3657.5
mse = 21.3736
samples = 204
value = 17.8392
mse = 26.9219
samples = 96
value = 32.6208
mse = 19.9577
samples = 98
value = 26.4082
mse = 17.9141
samples = 111
value = 20.5405
mse = 6.3978
samples = 93
value = 14.6151
MPG
Weight


Decision tree with a depth of 2
MPG
Weight
50
45
40
35
30
25
20
15
10
5
1000
1500
2000
2500
3000
3500
4000
4500
5000
5500

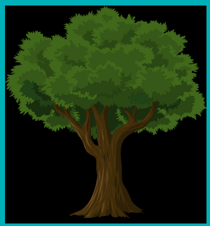

Decision tree with a depth of 5
MPG
Weight
50
45
40
35
30
25
20
15
10
5
1000
1500
2000
2500
3000
3500
4000
4500
5000
5500

Some issues with plain decision trees

- **High variance – prone to overfitting**
- **If you keep adding trees indefinitely you will fit training set perfectly, but will not generalize**
    - Can pick simpler trees – fixed depth, fixed number of leaves, etc.

Gradient Boosting

- **Instead of just using one tree, we build trees in succession, allowing each tree to “learn” from the one before it.**

Model =  **+ weight1 ***  **weight2 *** 

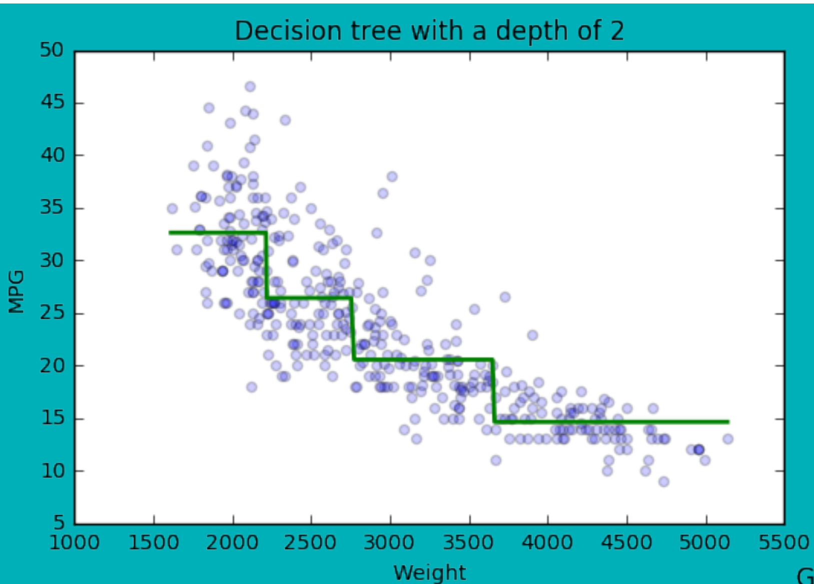
Compensates for errors made
by previous tree
Fit to residual of previous

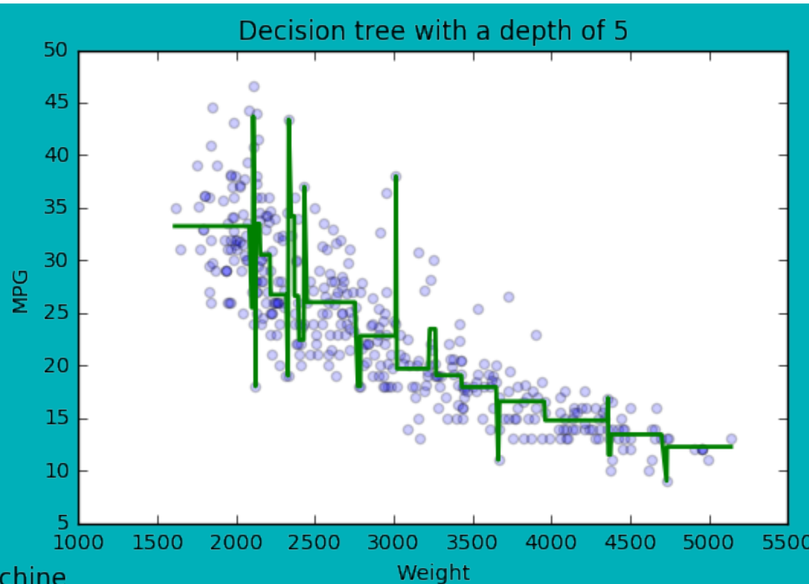
Compensates for errors made
by previous tree
Fit to residual of previous

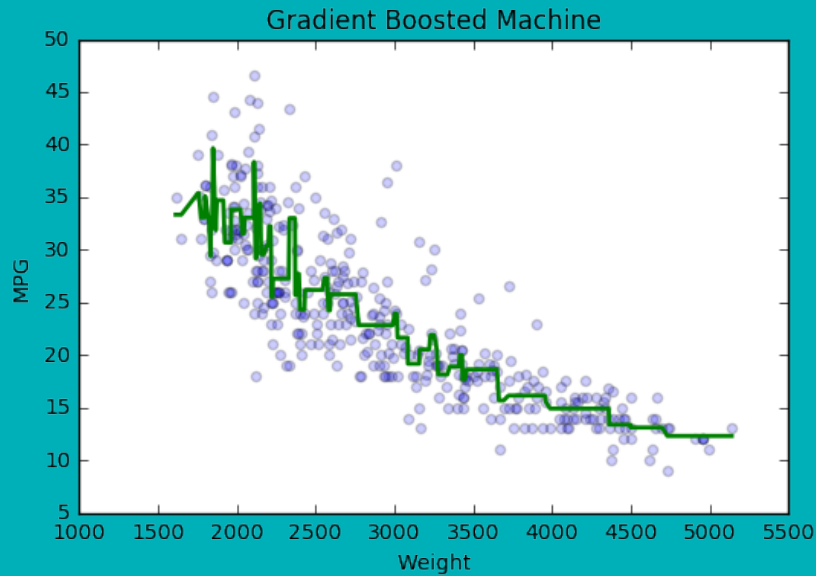
Gradient? Boosting?

- **Gradient boosting = gradient descent + boosting**
- **Boosting = Combining many weak learners into a strong learner**
- **Gradient descent**
  - Algorithm that minimizes functions by iteratively stepping in the direction of the negative gradient
  - We actually fit each subsequent tree to the negative gradient of the previous tree (which corresponds to the residual in linear regression)

- **Advantages**
  - Much less prone to overfitting than decision trees
  - Can use any loss function
  - Low bias compared to linear models
  - Handles interactions between variables well
  - Can be used for regression and classification
- **Disadvantages**
  - Parameter tuning (number of trees, tree depth, learning parameter)
  - Can be slow to train
  - Harder to interpret







Fit with 20 trees, each with a maximum depth of 2. These are parameters that we would normally tune.

The Future

- **Autonomous vehicles**
- **Communication between cars**

- **These will present more data and more challenges, including privacy and security.**

Thank you

If you want to learn more…

- **… about Arity**
  - www.arity.com
- **… about Hadoop/MapReduce**
  - http://hadoop.apache.org/
  - https://research.google.com/archive/mapreduce.html
- **…about gradient boosting / other machine learning algorithms**
  - Elements of Statistical Learning by Hastie, Tibshirani, Friedman
  - (great book, FREE pdf available LEGALLY online)