



A Taste of Random Decision Forests on Apache Spark

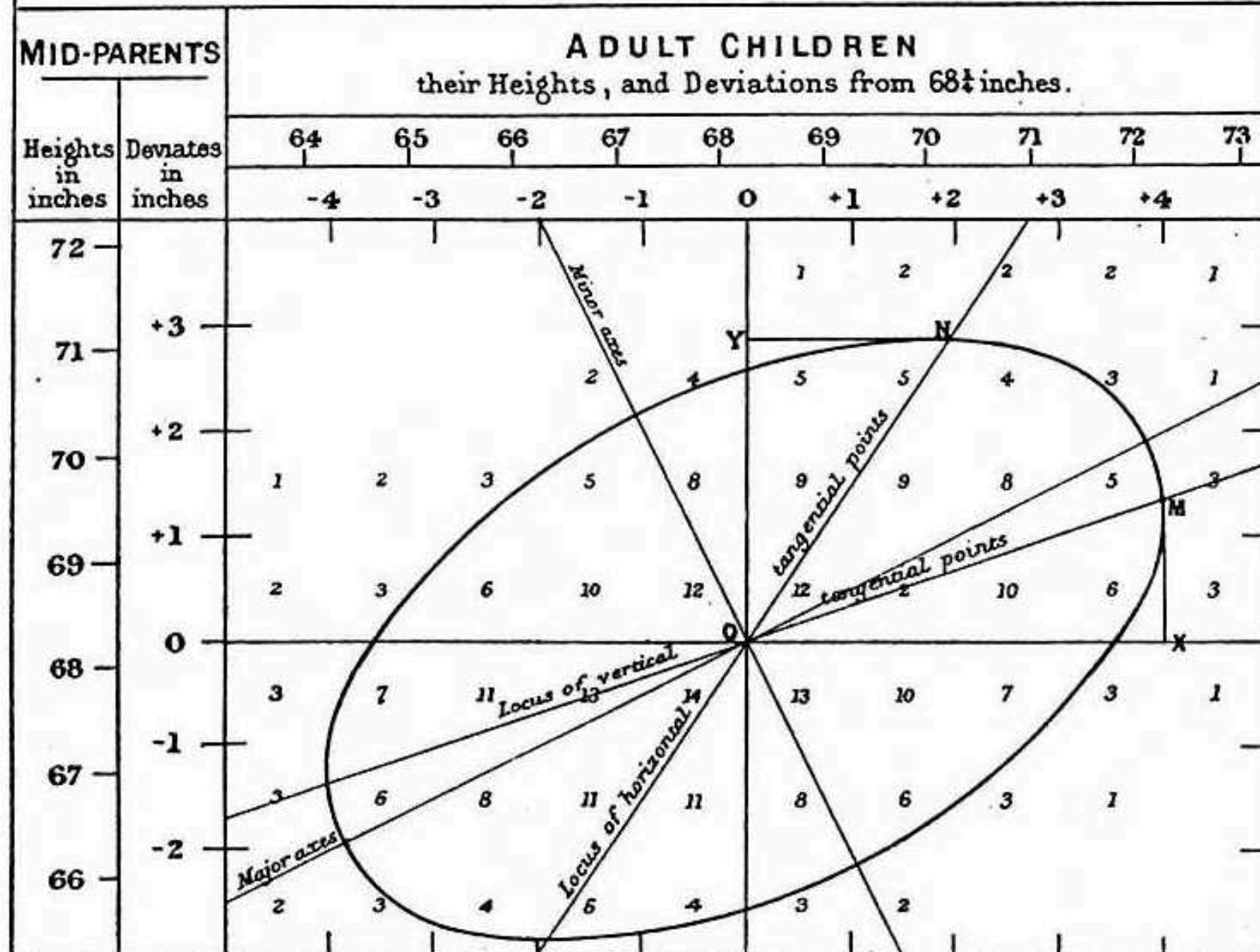
Sean Owen | Director, Data Science @ Cloudera



The Roots of Prediction

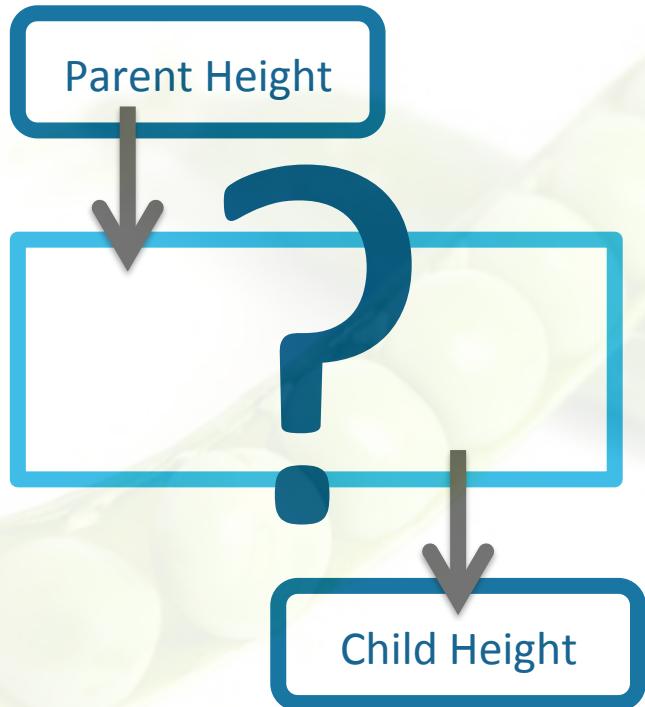
DIAGRAM BASED ON TABLE I.

(all female heights are multiplied by 1'08)

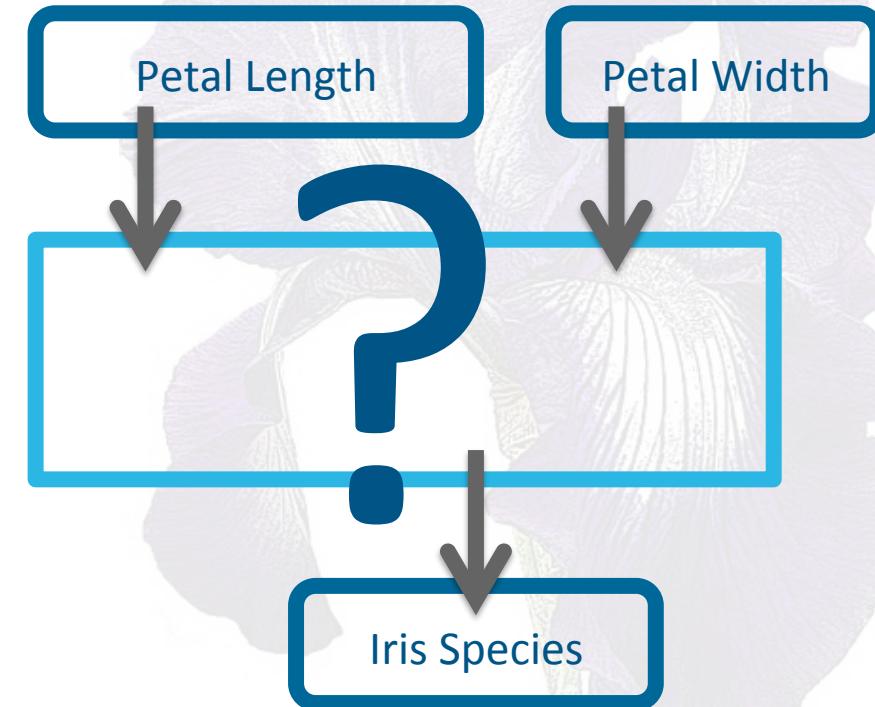


Input + Model = Output

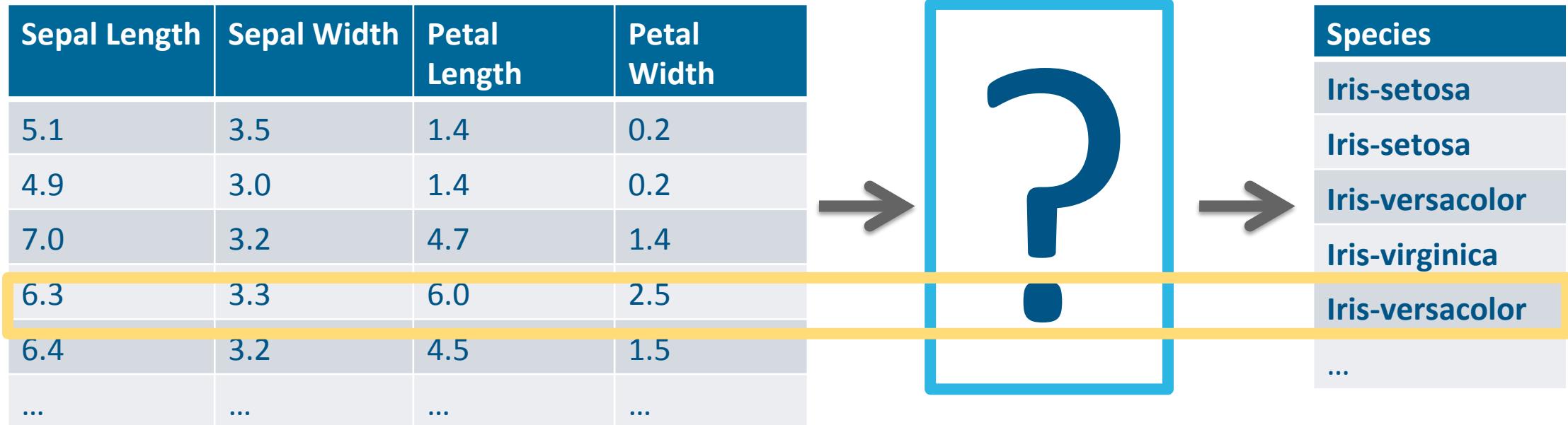
Galton's Regression



Classifying Iris Species

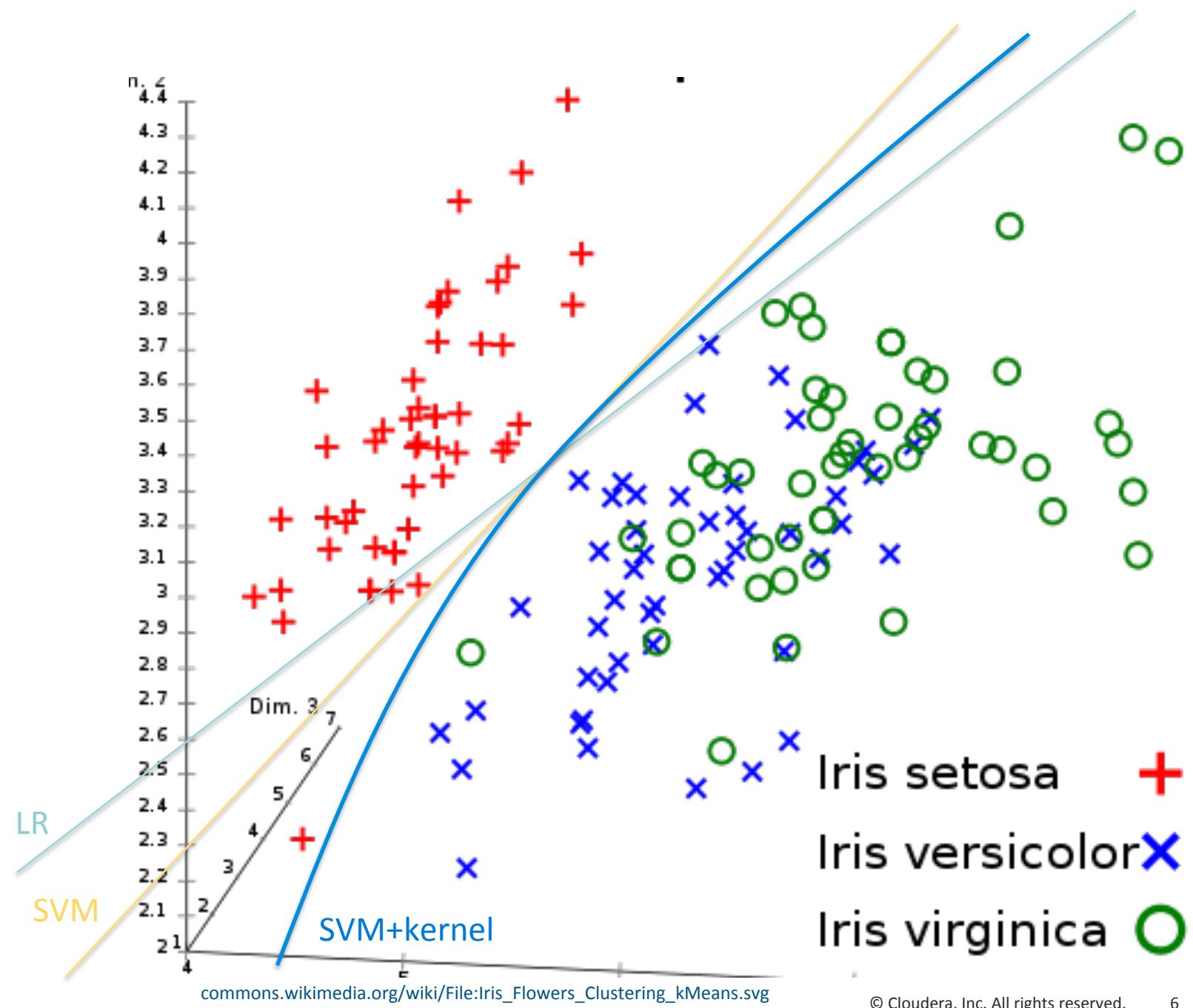


Feature Vectors in the Iris Data Set



$6.4, 3.2, 4.5, 1.5, \text{Iris-versicolor}$

Classically,
classification has been
linear models,
drawing ideal
category dividing lines

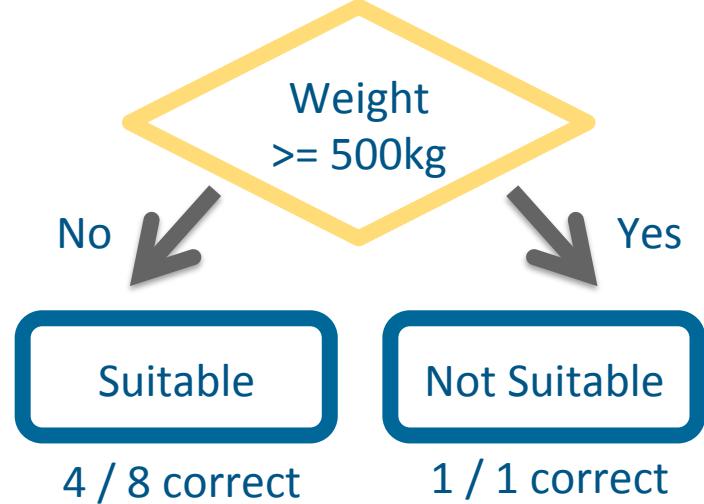


Introducing Decision Trees

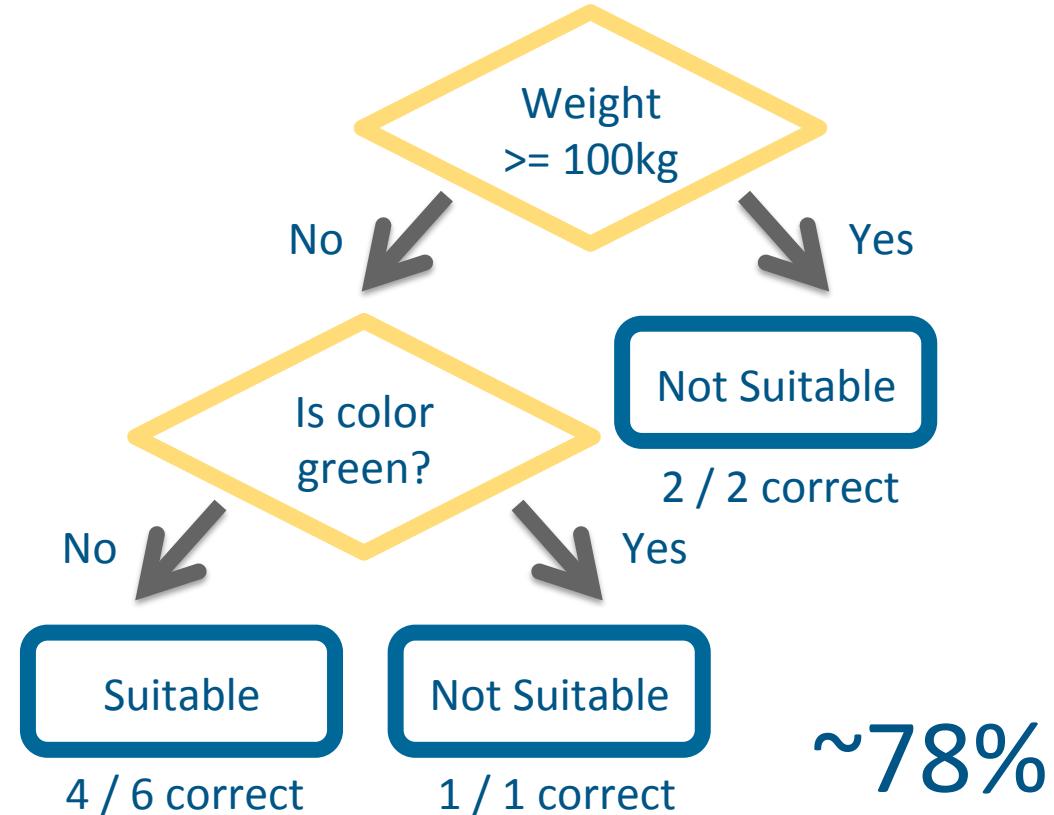
Good Pet Data Set

| Name | Weight (kg) | Legs | Color | Suitable Pet? |
|-------------|-------------|------|-------|---------------|
| Fido | 20.5 | 4 | Brown | Yes |
| Mr. Slither | 3.1 | 0 | Green | No |
| Nemo | 0.2 | 0 | Tan | Yes |
| Dumbo | 1390.8 | 4 | Grey | No |
| Kitty | 12.1 | 4 | Grey | Yes |
| Jim | 150.9 | 2 | Tan | No |
| Millie | 0.1 | 100 | Brown | No |
| McPigeon | 1.0 | 2 | Grey | No |
| Spot | 10.0 | 4 | Brown | Yes |

Possible Decision Trees

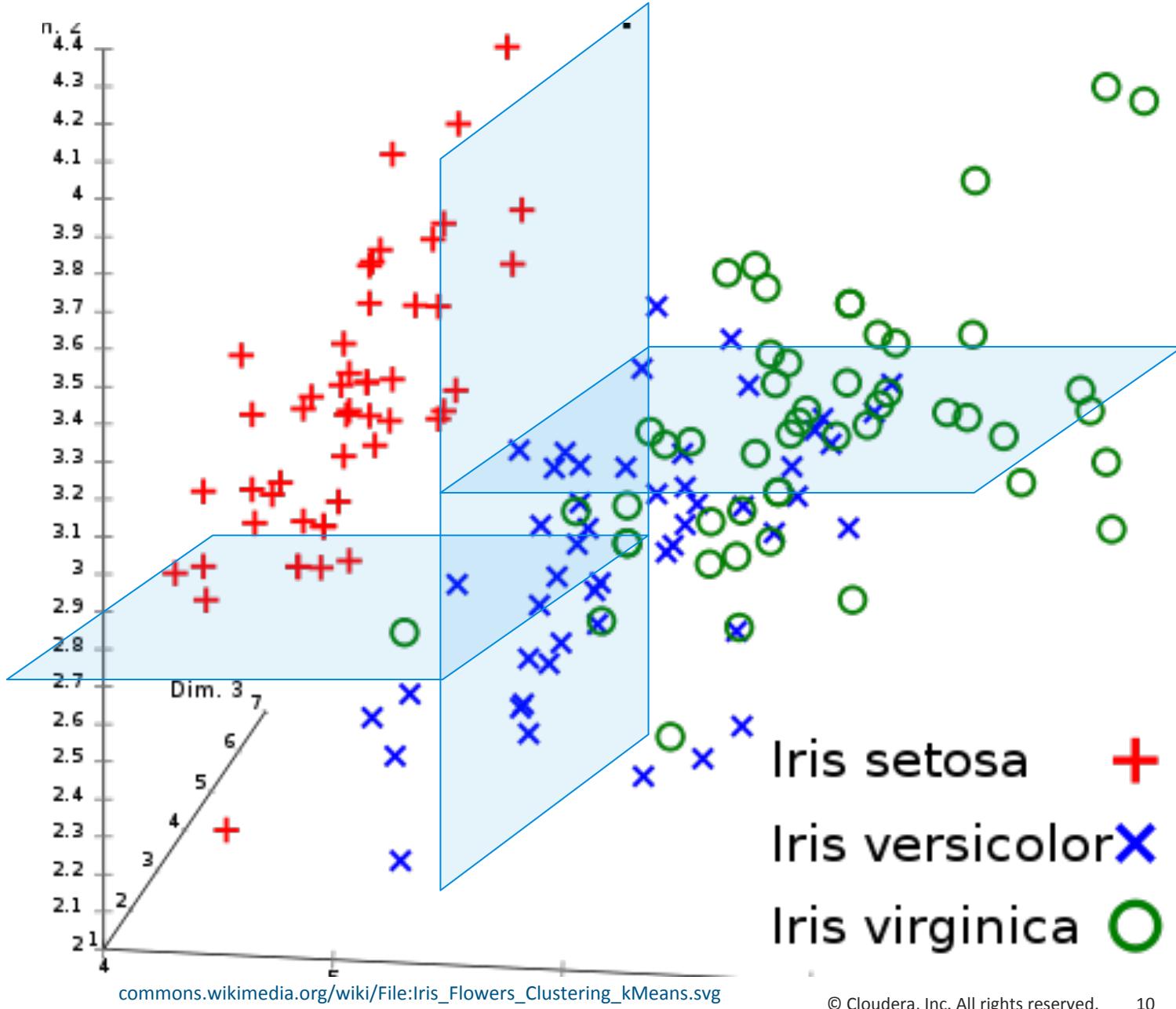


$\sim 56\%$



$\sim 78\%$

Simple decision trees
create axis-aligned,
stair-step decision
boundaries



Interpreting Models

Linear Models give Coefficients

```
> library("e1071")
> model <- svm(Species ~ ., data = iris,
                 method = "C-classification")
> model$sv
```

```
Sepal.Length Sepal.Width Petal.Length Petal.Width
9      -1.74301699 -0.36096697 -1.33575163 -1.3110521
14     -1.86378030 -0.13153881 -1.50569459 -1.4422448
16     -0.17309407  3.08045544 -1.27910398 -1.0486668
...
...
```

Decision Trees give Decisions

```
> library("party")
> cf <- cforest(Species ~ ., data = iris)
> party:::prettytree(cf@ensemble[[1]],
                      names(cf@data@get("input")))

1) Petal.Length <= 1.9; ...
   2)* weights = 0
1) Petal.Length > 1.9
  3) Petal.Width <= 1.7; ...
    4) Petal.Length <= 4.8; ...
      5) Sepal.Length <= 5.5; ...
        6)* weights = 0
...
...
```

Decision Trees in MLlib

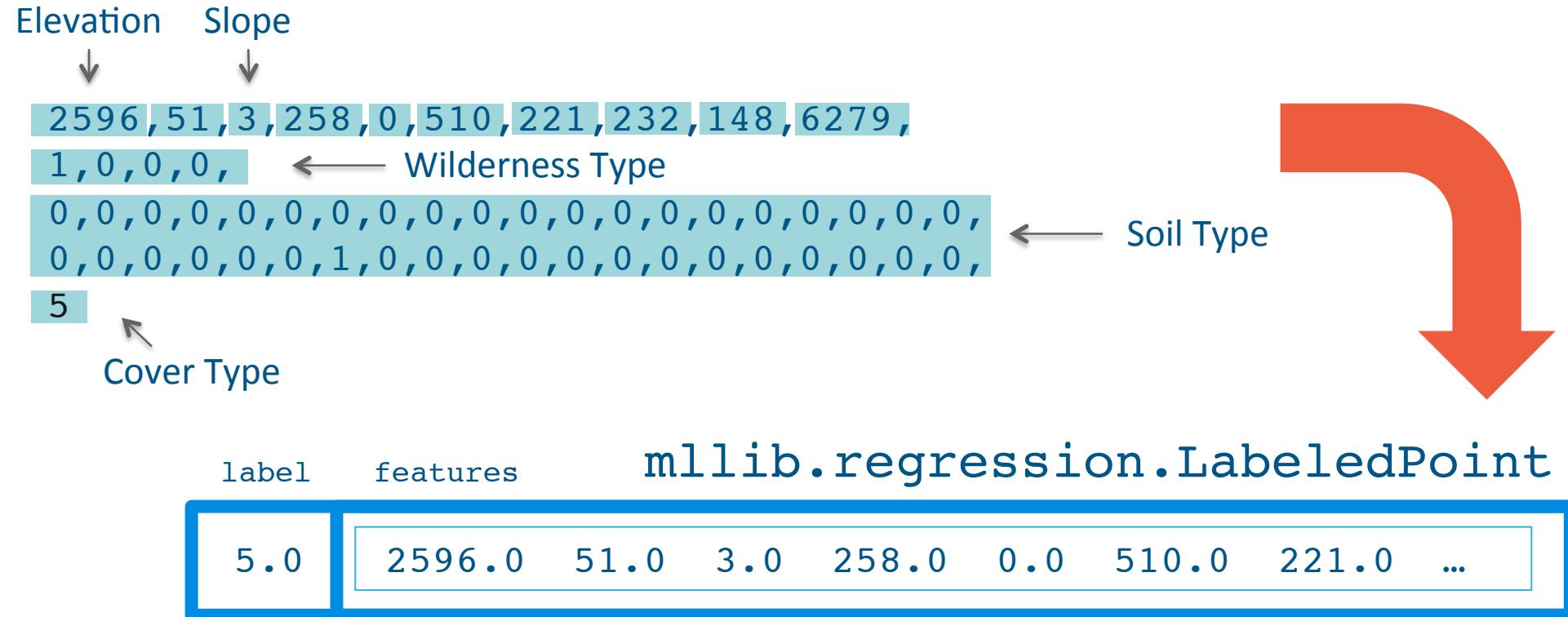
Covertype Data Set

581,012 examples

54 features
([elevation](#), [slope](#), [soil type](#), etc.) predicting forest cover type
([spruce](#), [aspen](#), etc.)



Encoding as LabeledPoint



Building a Decision Tree in MLlib

```
import org.apache.spark.mllib.linalg._  
import org.apache.spark.mllib.regression._  
import org.apache.spark.mllib.tree._  
  
val rawData = sc.textFile("/user/ds/covtype.data")  
  
val data = rawData.map { line =>  
    val values = line.split(',').map(_.toDouble)  
    val featureVector = Vectors.dense(values.init)  
    val label = values.last - 1  
    LabeledPoint(label, featureVector)  
}  
  
val Array(trainData, testData) =  
    data.randomSplit(Array(0.9, 0.1))  
trainData.cache()  
  
val model = DecisionTree.trainClassifier(  
    trainData, 7, Map[Int, Int](), "gini", 4, 100)
```

```
DecisionTreeModel classifier  
  of depth 4 with 31 nodes  
  If (feature 0 <= 3042.0)  
    If (feature 0 <= 2509.0)  
      If (feature 3 <= 0.0)  
        If (feature 12 <= 0.0)  
          Predict: 3.0  
        Else (feature 12 > 0.0)  
          Predict: 5.0  
      Else (feature 3 > 0.0)  
        If (feature 0 <= 2465.0)  
          Predict: 2.0  
        Else (feature 0 > 2465.0)  
          Predict: 2.0  
    Else (feature 0 > 2509.0)  
  ...
```

Evaluating a Decision Tree

```
import org.apache.spark.mllib.evaluation._  
import org.apache.spark.mllib.tree.model._  
import org.apache.spark.rdd._  
  
def getMetrics(model: DecisionTreeModel,  
               data: RDD[LabeledPoint]) = {  
    val predictionsAndLabels = data.map(example =>  
        (model.predict(example.features), example.label))  
    new MulticlassMetrics(predictionsAndLabels)  
}  
  
val metrics = getMetrics(model, testData)  
  
metrics.confusionMatrix  
metrics.precision
```

| | | | | | | | |
|---------|---------|--------|------|-----|------|------|-------|
| 14406.0 | 6459.0 | 6.0 | 0.0 | 0.0 | 0.0 | 0.0 | 416.0 |
| 5706.0 | 22086.0 | 415.0 | 13.0 | 0.0 | 11.0 | 40.0 | |
| 0.0 | 439.0 | 3056.0 | 65.0 | 0.0 | 13.0 | 0.0 | |
| 0.0 | 0.0 | 139.0 | 98.0 | 0.0 | 0.0 | 0.0 | |
| 0.0 | 926.0 | 28.0 | 2.0 | 0.0 | 0.0 | 0.0 | |
| 0.0 | 461.0 | 1186.0 | 37.0 | 0.0 | 61.0 | 0.0 | |
| 1120.0 | 27.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 925.0 |

0.6988527889097195

~ 70% accuracy

metrics.precision
metrics.confusionMatrix

Better Than Random Guessing?

```
def classProbabilities(data: RDD[LabeledPoint]) = {  
    val countsByCategory =  
        data.map(_.label).countByValue()  
    val counts =  
        countsByCategory.toArray.sortBy(_._1).map(_._2)  
    counts.map(_.toDouble / counts.sum)  
}  
  
val trainProbs = classProbabilities(trainData)  
val testProbs = classProbabilities(testData)  
  
trainProbs.zip(testProbs).map {  
    case (trainProb, testProb) => trainProb * testProb  
}.sum  
  
}  
ms.  
case (trainProb, testProb) => trainProb * testProb  
trainProbs.zip(testProbs).sum.
```

| | | | | | | |
|-------|-------|------|-----|-----|-----|------|
| 7758 | 10303 | 1302 | 86 | 348 | 636 | 755 |
| 10383 | 13789 | 1743 | 116 | 466 | 851 | 1011 |
| 1310 | 1740 | 220 | 15 | 59 | 107 | 128 |
| 102 | 136 | 17 | 1 | 5 | 8 | 10 |
| 348 | 462 | 58 | 4 | 16 | 28 | 34 |
| 636 | 845 | 107 | 7 | 29 | 52 | 62 |
| 751 | 997 | 126 | 8 | 34 | 62 | 73 |

0.37682125742281786

~ 38% accuracy

Hyperparameters

```
trainClassifier(trainData,  
    7, Map[Int, Int](),  
    "gini", 4, 100)
```

Impurity: Gini

Measure how much decision decreases impurity using Gini impurity (vs. entropy)

Maximum Depth: 4

Build trees to depth of 4 decisions before terminating in a prediction

Maximum Bins: 100

Consider 100 different decision rules for a feature when choosing a next best decision

Decisions Should Make Lower Impurity Subsets

Gini Impurity

- 1 minus probability that random guess i (probability p_i) is correct
- Lower is better

$$1 - \sum p_i^2$$

Entropy

- Information theory concept
- Measures mixed-ness, unpredictability of population
- Lower is better

$$-\sum p_i \log p_i$$

Tuning Hyperparameters

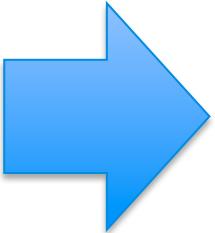
```
val evaluations =  
  for (impurity <- Array("gini", "entropy");  
       depth    <- Array(1, 20);  
       bins     <- Array(10, 300)) yield {  
  
    val model = DecisionTree.trainClassifier(  
      trainData, 7, Map[Int, Int](),  
      impurity, depth, bins)  
  
    val accuracy =  
      getMetrics(model, testData).precision  
  
    ((impurity, depth, bins), accuracy)  
  }  
  
evaluations.sortBy(_._2).reverse.foreach(println)
```

```
((entropy,20,300),0.9125545571245186)  
((gini,20,300),0.9042533162173727)  
((gini,20,10),0.8854428754813863)  
((entropy,20,10),0.8848951647411211)  
((gini,1,300),0.6358065896448438)  
((gini,1,10),0.6355669661959777)  
...
```

~ 91% accuracy

One-Hot / 1-of-n Encoding

..., cat, ...
..., dog, ...
..., fish, ...



..., 1, 0, 0, ...
..., 0, 1, 0, ...
..., 0, 0, 1, ...

Categoricals Revisited

```
2596,51,3,258,0,510,221,232,148,6279,  
1,0,0,0, ← Wilderness Type (4 features)  
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
5
```



```
2596,51,3,258,0,510,221,232,148,6279,  
0, ← Wilderness Type (0 – 3)  
26, ← Soil Type (0 – 39)  
5
```

Categoricals Revisited

```
val data = rawData.map { line =>
  val values = line.split(',').map(_.toDouble)

  val wilderness = values.slice(10, 14).
    indexOf(1.0).toDouble
  val soil = values.slice(14, 54).
    indexOf(1.0).toDouble

  val vector = Vectors.dense(
    values.slice(0, 10) :+ wilderness :+ soil)
  val label = values.last - 1

  LabeledPoint(label, vector)
}

...
DecisionTree.trainClassifier(trainData,
  7, Map(10 -> 4, 11 -> 40),
  impurity, depth, bins)
...
```

```
((entropy,30,300),0.9438383977425239)
((entropy,30,40),0.938934581368939)
((gini,30,300),0.937127912178671)
((gini,30,40),0.9329467634811934)
((entropy,20,40),0.9280773598540899)
((gini,20,300),0.9249630062975326)
...
```

~ 94% accuracy

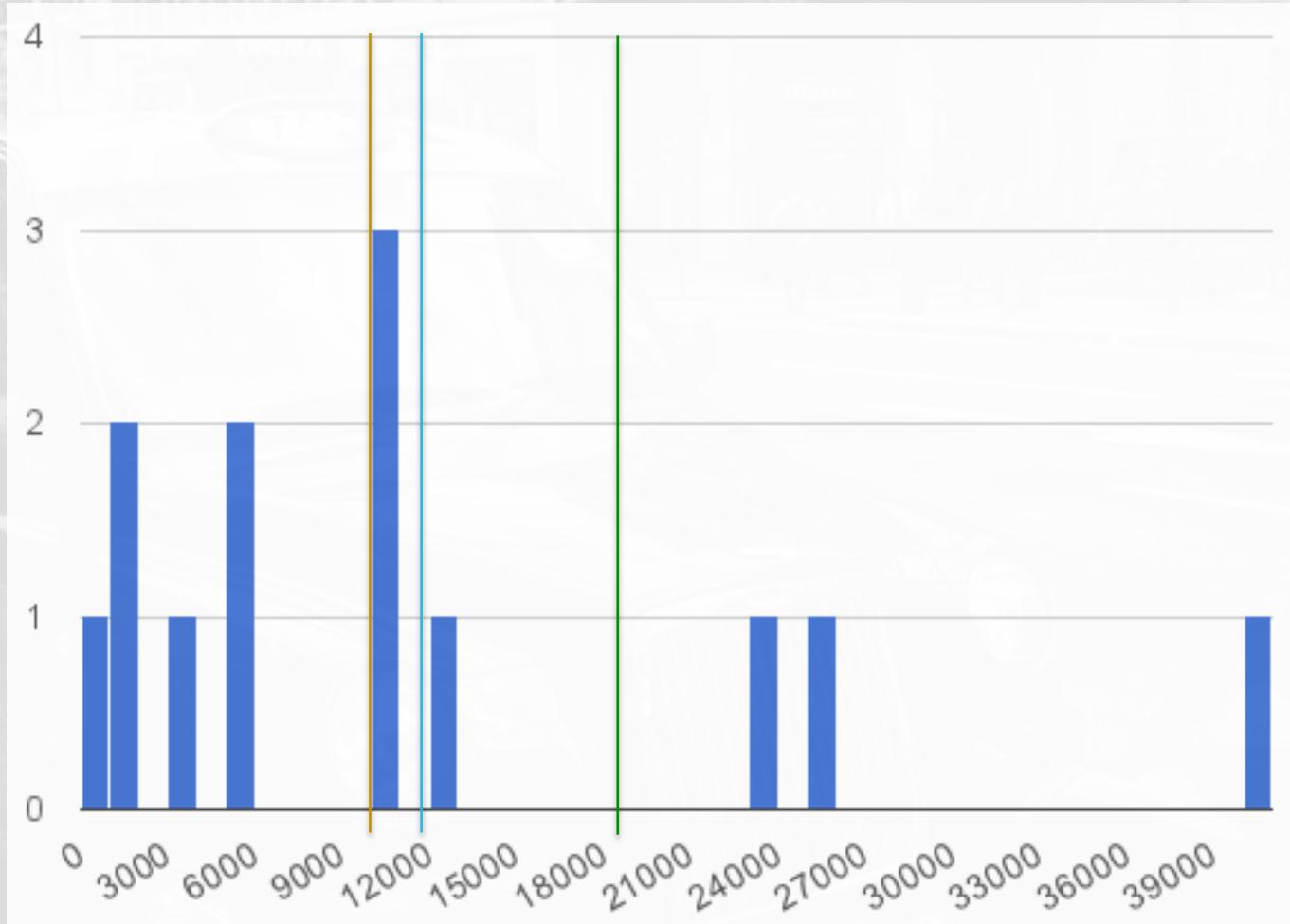
Decision Trees to Random Decision Forests

**Wisdom of the
Crowds:** How many
black cabs operate in
London?

cloudera



My Guess 10,000
Mean Guess 11,170
Correct 19,000



**Random Decision
Forests** leverage the
wisdom of a crowd of
Decision Trees

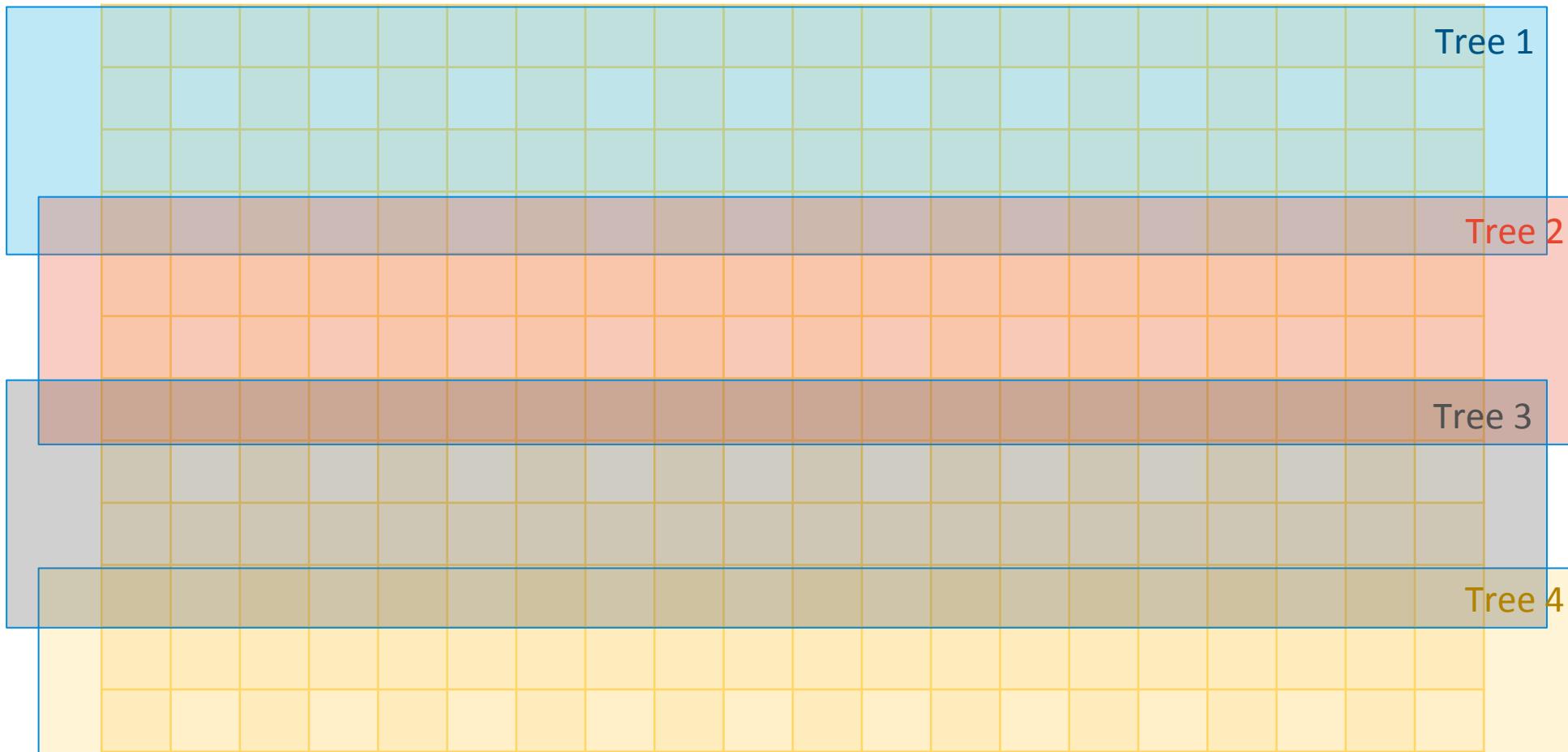
cloudera



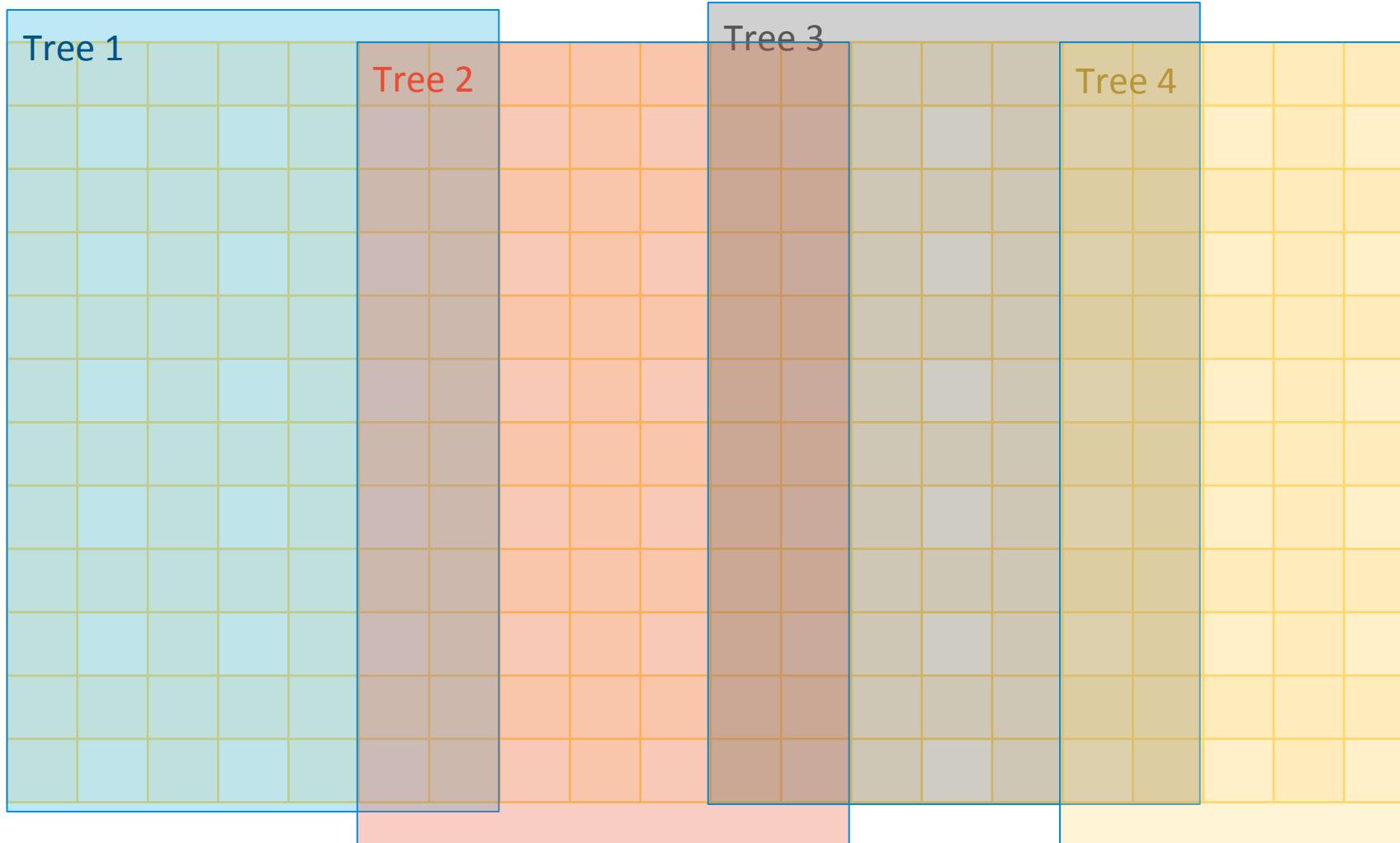
How to Create a Crowd?



Trees See Subsets of Examples



Or Subsets of Features



Diversity of Opinion

```
If (feature 0 <= 4199.0)      If (feature 11 <= 0,5)  
  If (feature 1 <= 14.0)        Predict: 1.0  
    Predict: 3.0  
  ...  
...
```

```
  If (feature 0 <= 3042.0)  
    If (feature 0 <= 2509.0)  
      If (feature 3 <= 0.0)  
        If (feature 12 <= 0.0)  
          Predict: 3.0  
        ...
```

Random Decision Forests

```
RandomForest.trainClassifier(trainData,  
    7, Map(10 -> 4, 11 -> 40),  
    20, "auto", "entropy", 30, 300)
```

Number of Trees: 20

How many different decision
trees to build

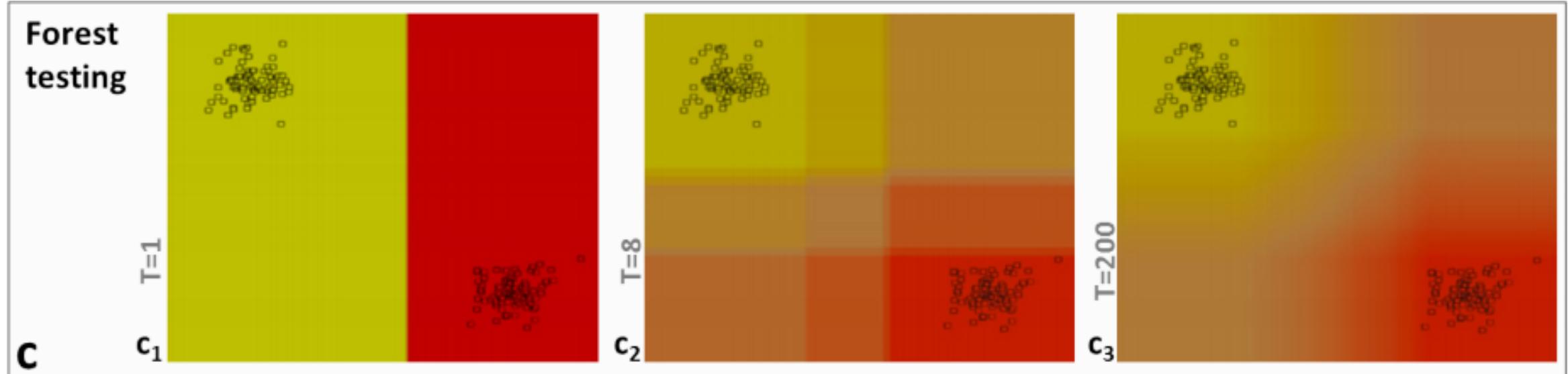
Feature Subset Strategy:

auto

Intelligently pick a number of
different features to try at each
node

~ 96% accuracy

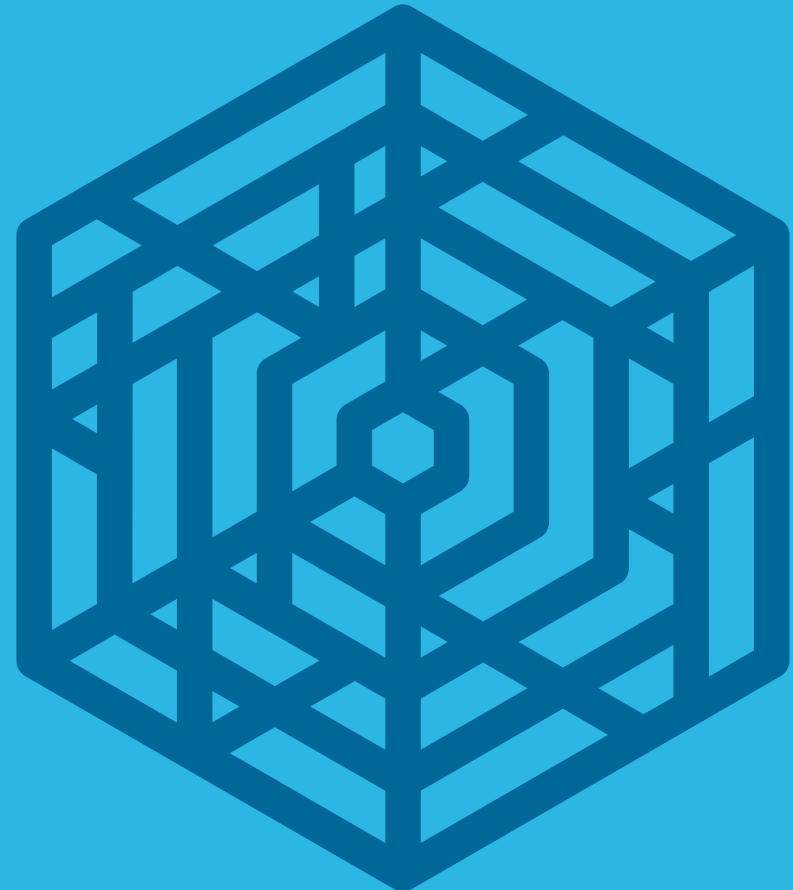
Forests Learn More Complex Boundaries



A. Criminisi, J. Shotton and E. Konukoglu. "Decision Forests for Classification, Regression, Density Estimation, Manifold Learning and Semi-Supervised Learning." p. 26

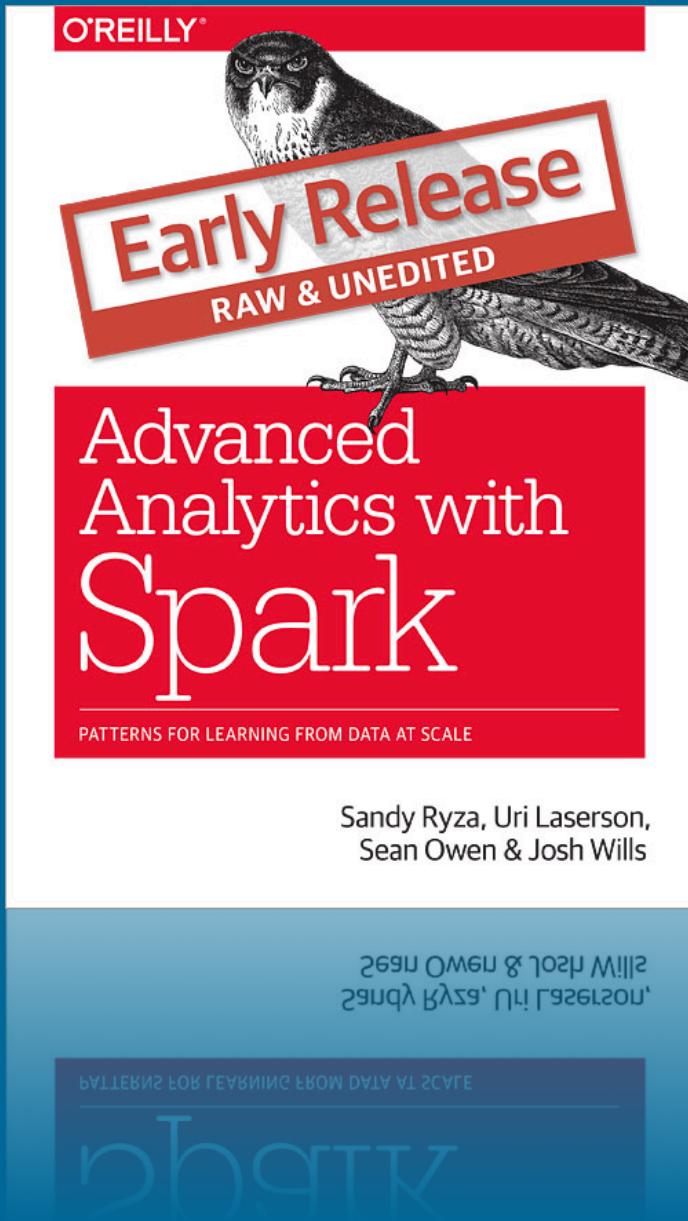
Try **Support Vector Machines**,
Logistic Regression in MLLib
Real-time scoring with
Spark Streaming
Use random decision
forests for **regression**





cloudera
Thank You

@sean_r_owen
sowen@cloudera.com



Chapter 4