Power of data. Simplicity of design. Speed of innovation.

Hands on Introduction to Apache Spark for Data Engineers, Data Scientists, and Developers

Agenda

- Bio
- IBM
- Decision Trees
- Decision Trees in Spark ML
- Jupyter Notebook in Data Science Experience

- **Do I look like an IBMer?**
- **Do I look like a sales guy?**

IBM is All-in on Spark

"It's like Spark just got blessed by the enterprise rabbi."

Ben Horowitz,
Cofounder of Venture Capital firm Andreessen Horowitz

Contribute to the Core

Launch Spark Technology Cluster (STC), 300 engineers

Open source SystemML

Partner with Databricks

Infuse the Portfolio

Integrate Spark throughout portfolio

3,500 employees working on Spark-related topics

Spark however customers want it – standalone, platform or products

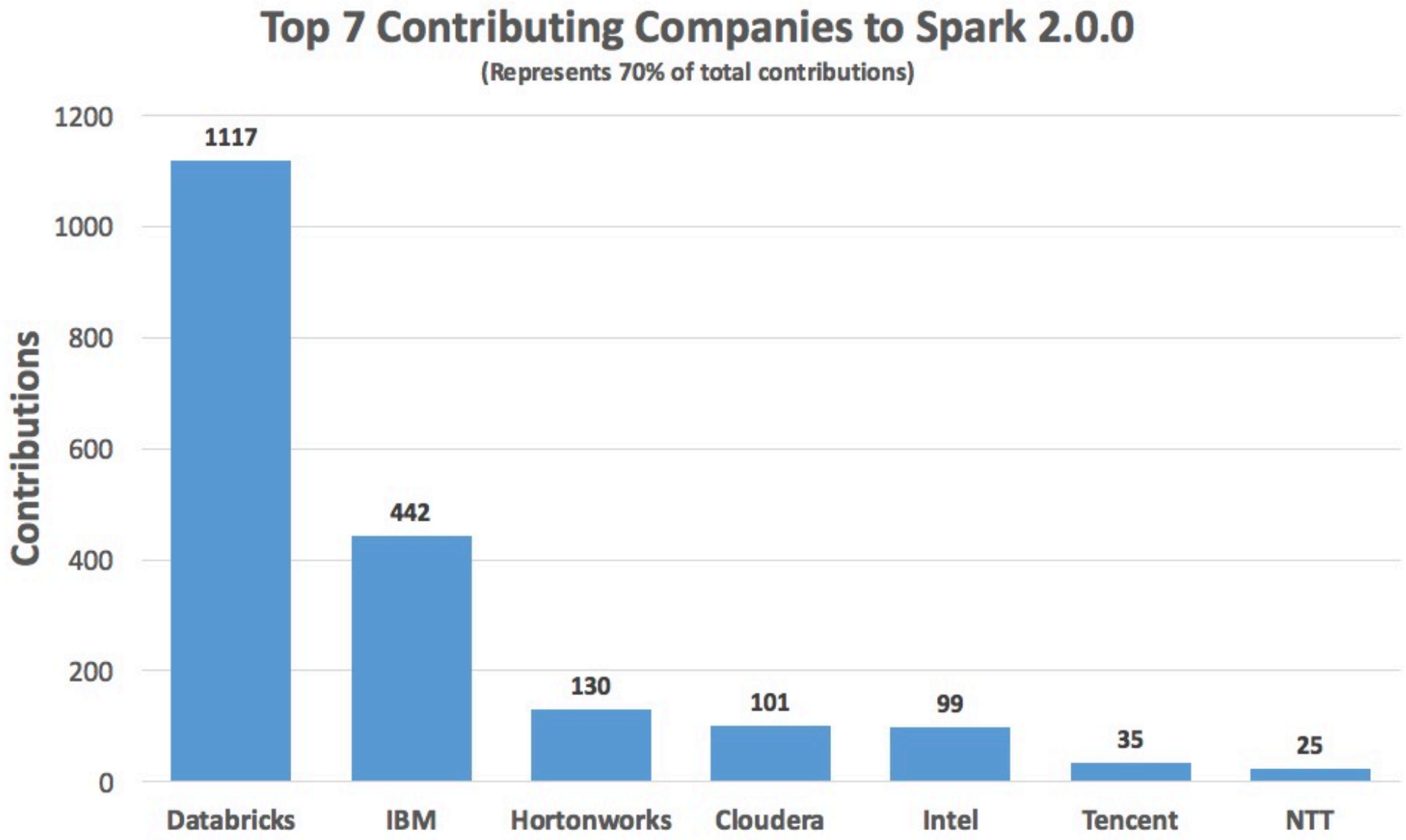
Foster Community

Educate 1M+ data scientists and engineers via online courses

Sponsor AMPLab, creators and evangelists of Spark

Contribute to the Core: Spark Technology Center

Contribute to the Core: Actual Code!

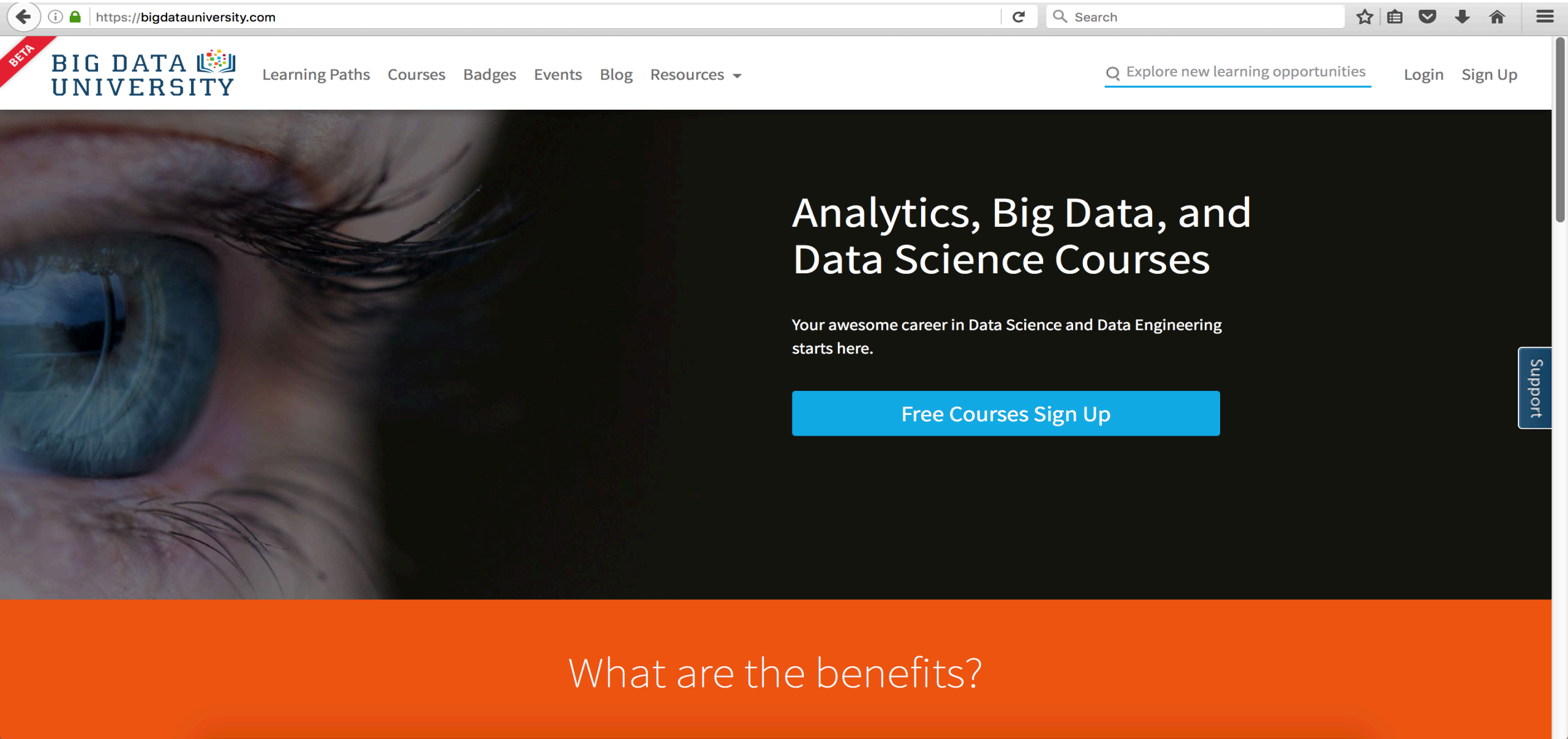


Infuse the Portfolio

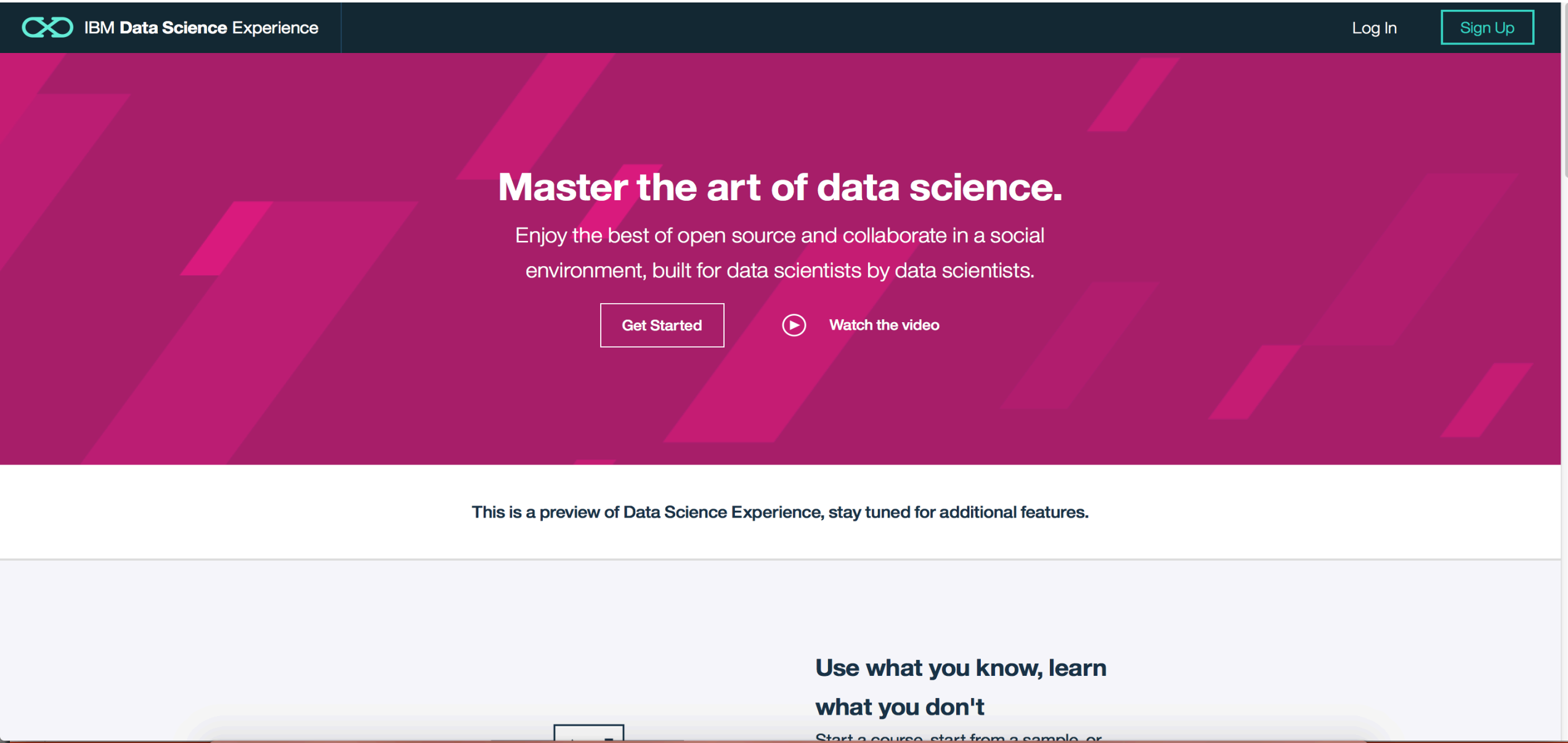
IBM Bluemix

IBM Watson Health

Foster the Community: Big Data University



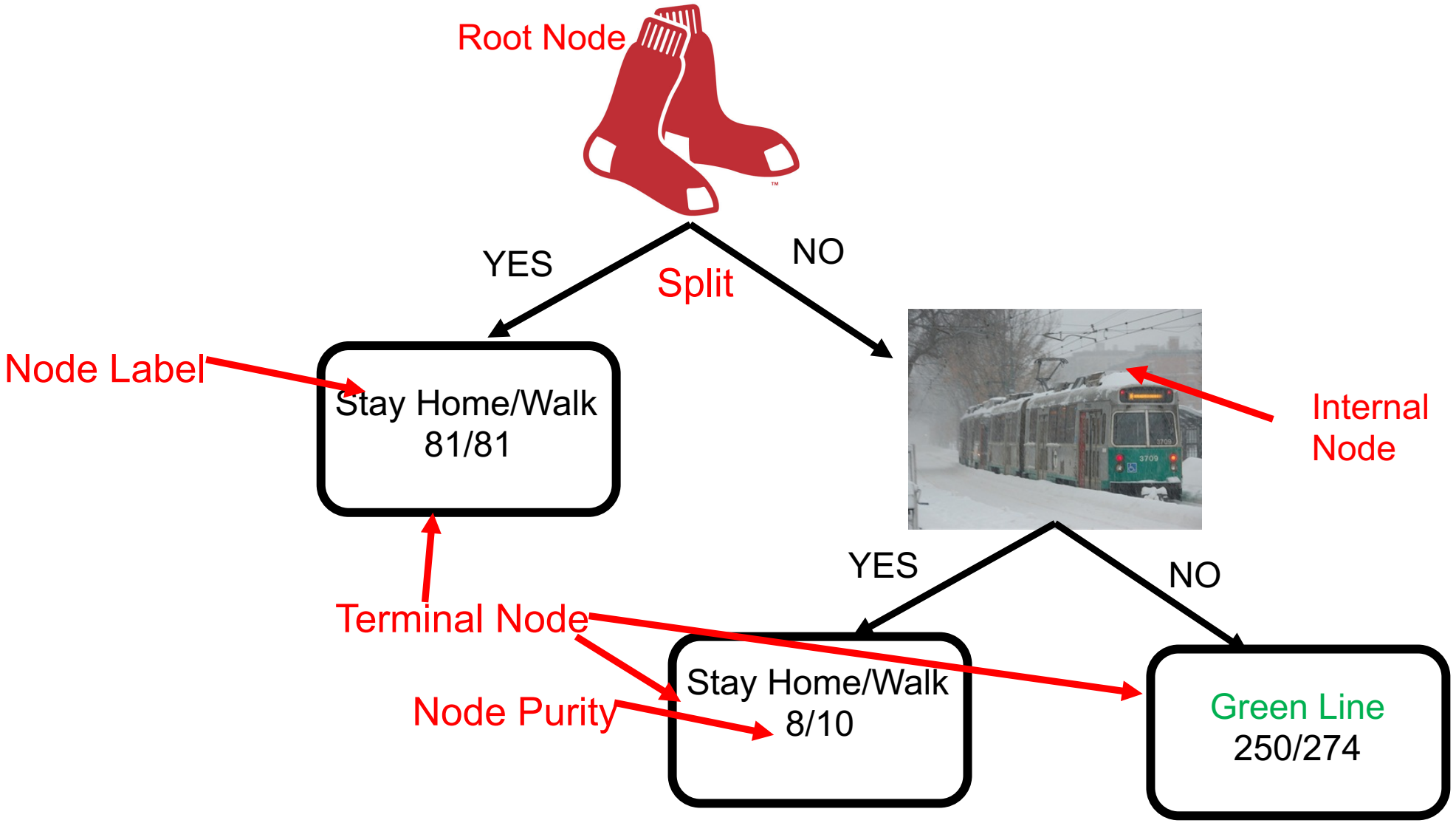
Foster the Community: Data Science Experience



DECISION TREES

Decision Tree –Visual Structure

"Should I take the Green line?"



Decision Tree Classifier - What is it?

- **Two Main Types**
  - Classification
    - Used to assign labels or classes
  - Regression
    - Used for continuous target variable

- **Input variables can be continuous or discrete**
- **Output:**
  - A tree that describes the decision flow.
    - These trees can be turned into a set of rules
      - If the Red Sox are playing then stay home
  - Leaf nodes return either a probability score or classification.

General Algorithm Classification

- **To construct tree T from training set S**
  - The decision tree is a greedy algorithm that performs a recursive binary partitioning of the feature space. The tree predicts the same label for each bottommost (leaf) partition. Each partition is chosen greedily by selecting the *best split* from a set of possible splits, in order to maximize the information gain at a tree node. In other words, the split chosen at each tree node is chosen from the set argmaxsIG(D,s) argmaxs IG(D,s) where IG(D,s) is the information gain when a split s is applied to a dataset DD.

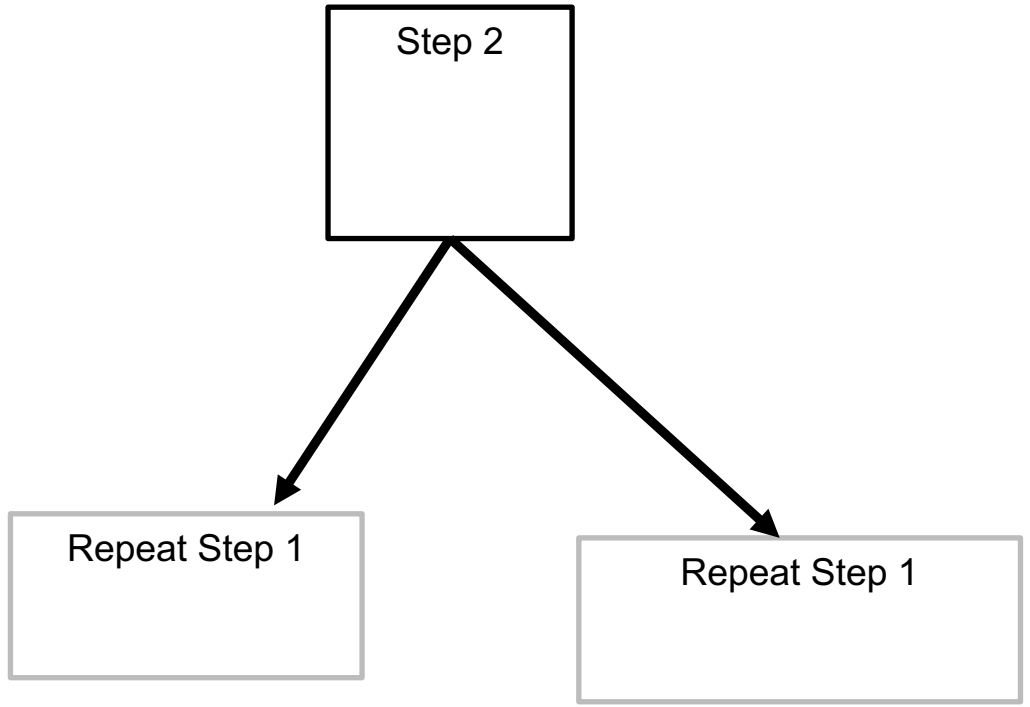
- **What does this mean in plain English?**
  - The algorithm tries to find the best splits. The best splits give us the best separation of the data.

Step 1: Pick the Most “Informative" Attribute

Impurity	Task	Formula	Description
Gini impurity	Classification	$\sum_{i=1}^{C} f_i(1-f_i)$	f_i is the frequency of label i at a node and C is the number of unique labels.
Entropy	Classification	$\sum_{i=1}^{C} -f_i log(f_i)$	f_i is the frequency of label i at a node and C is the number of unique labels.
Variance	Regression	$\frac{1}{N}\sum_{i=1}^{N}(y_i-\mu)^2$	y_i is label for an instance, N is the number of instances and μ is the mean given by $\frac{1}{N}\sum_{i=1}^{N} y_i$.

Step 2 & 3: Partition on Most "Informative" Attribute

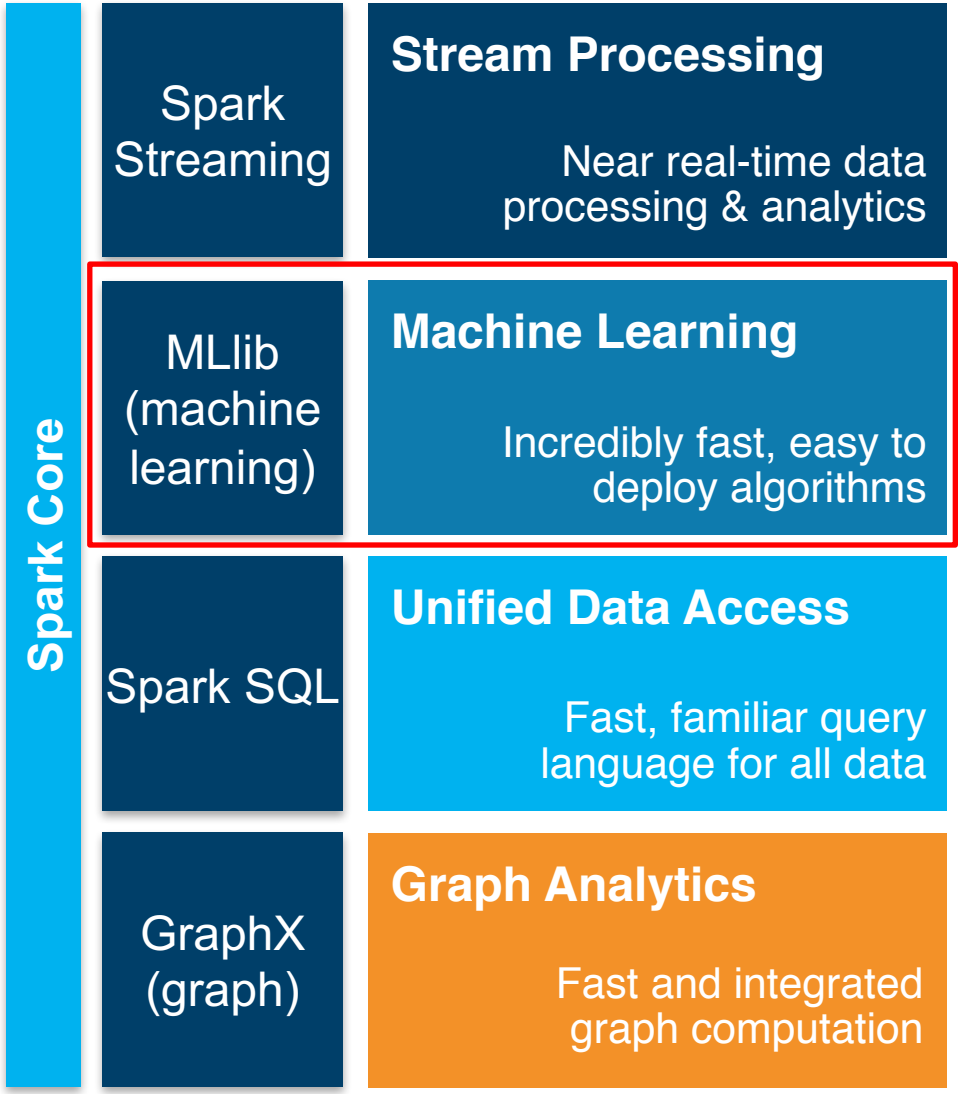
- **Step 2: Split on the most informative attribute**
  - In our example "are the Red Sox playing"
- **Step 3: At each resulting node, repeat Steps 1 and 2**
  - until node is "pure" or stopping condition is met



Model Evaluation

- **Test data**
- **Receiver Operating Characteristics (ROC)**
- **Confusion Matrix**
- **Do the splits make sense?**
  - What does the domain expert say?
- **How deep is the tree?**
  - Too many layers are prone to over-fit

Spark ML



- **Micro-batch event processing** for near real-time analytics
- Process live streams of data (IoT, Twitter, Kafka)
- No multi-threading or parallel processing required

- **Predictive and prescriptive analytics**, and smart application design, from statistical and algorithmic models
- Algorithms are pre-built

- **Query your structured data sets** with SQL or other dataframe APIs
- Data mining, BI, and insight discovery
- Get results faster due to performance

- **Represent data in a graph**
- Represent/analyze systems represented by nodes and interconnections between them
- Transportation, person to person relationships, etc.

DECISION TREES IN SPARK ML

Function in Spark 1.6 MLlib

- **pyspark.mllib.tree.DecisionTree.trainClassifier(**
  - data – Training data: RDD of LabeledPoint. Labels should take values {0, 1, ..., numClasses-1}.
  - numClasses – Number of classes for classification.
  - categoricalFeaturesInfo – Map storing arity of categorical features. An entry (n -> k) indicates that feature n is categorical with k categories indexed from 0: {0, 1, ..., k-1}.
  - impurity – Criterion used for information gain calculation. Supported values: “gini” or “entropy”. (default: “gini”)
  - maxDepth – Maximum depth of tree (e.g. depth 0 means 1 leaf node, depth 1 means 1 internal node + 2 leaf nodes). (default: 5)
  - maxBins – Number of bins used for finding splits at each node. (default: 32)
  - minInstancesPerNode – Minimum number of instances required at child nodes to create the parent split. (default: 1)
  - minInfoGain – Minimum info gain required to create a split. (default: 0.0))

Argument: data

- data – Training data: RDD of **LabeledPoint**. Labels should take values {0, 1, ..., numClasses-1}.
  - Remember Spark MLlib, so Resilient Distributed Data
  - LabeledPoint? It's a specific type of RDD with the format
    - Used for supervised learning
    - (Dependent variable: float, [independent variable, ...])
    - (1.0, [3, 4.567])
    - pyspark.mllib.regression.LabeledPoint(*label*, *features*)

Argument: numClasses

- numClasses
  - Number of classes for classification
  - NOT USED FOR REGRESSION

Argument: categoricalFeaturesInfo

- categoricalFeaturesInfo
  - Specifies which features are categorical and how many categorical values each of those features can take. This is given as a map from feature indices to feature arity (number of categories). Any features not in this map are treated as continuous.
  - Map storing arity of categorical features. An entry (n -> k) indicates that feature n is categorical with k categories indexed from 0: {0, 1, ..., k-1}.

Argument: impurity

- impurity
  - Splitting measure
  - Criterion used for information gain calculation. Supported values: “gini” or “entropy”. (default: “gini”)
    - Gini measure of statistical dispersion that measures inequality
    - Entropy or randomness

Argument: maxDepth

- maxDepth
  - Maximum depth of tree (e.g. depth 0 means 1 leaf node, depth 1 means 1 internal node + 2 leaf nodes). (default: 5)
  - Deeper trees are more expressive (potentially allowing higher accuracy), but they are also more costly to train and are more likely to overfit.

Arguments: maxBins

- **maxBins**
  - Number of bins used for finding splits at each node. (default: 32)
  - Number of bins used when discretizing continuous features. Increasing maxBins allows the algorithm to consider more split candidates and make fine-grained split decisions. However, it also increases computation and communication.
  - Note that the maxBins parameter must be at least the maximum number of categories *M* for any categorical feature

Argument: minInstancesPerNode

- minInstancesPerNode
  - Minimum number of instances required at child nodes to create the parent split.
  - default: 1
  - Helps prevent overfitting

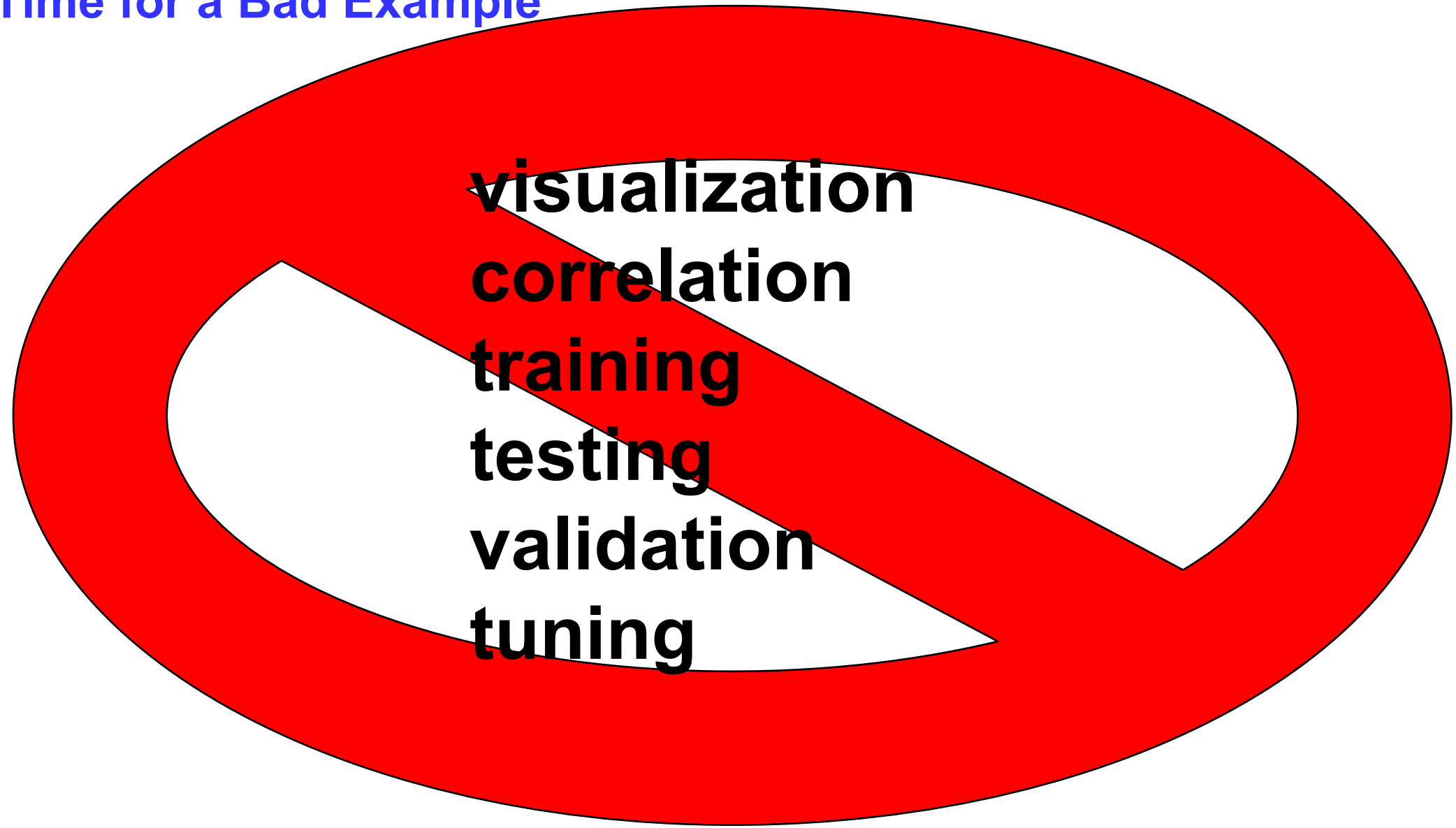
Arguments: minInfoGain

- minInfoGain
  - Minimum info gain required to create a split.
  - The higher this is, the less likely you are to overfit
  - default: 0.0

Arguments: Spark 2.0

- algo
  - Classification or Regression
- subsampling
  - More used for advanced decision trees
  - Determines data sampling rate for GB or Random
- maxMemoryInMB:
  - Amount of memory to be used for collecting sufficient statistics.

Time for a Bad Example



Where to go from here?

- **Great foundation for more complex algorithms**
  - Random Forest
  - Gradient Boosted Trees
- **Try it with your own dataset**

Contact

Joseph Kambourakis
Open Source Analytics Technical Evangelist
joseph.kambourakis@ibm.com
@mouthorjoe