



Understanding Parallelization of Machine Learning Algorithms in Apache Spark™

About Richard



Richard Garris has spent 15 years advising customers in data management and analytics. As a Director of Solutions Architecture at Databricks the past 3 years, he works closely with data scientists to build machine learning pipelines and advise companies on best practices in advanced analytics. He has advanced degrees from Carnegie Mellon and The Ohio State University.

Agenda

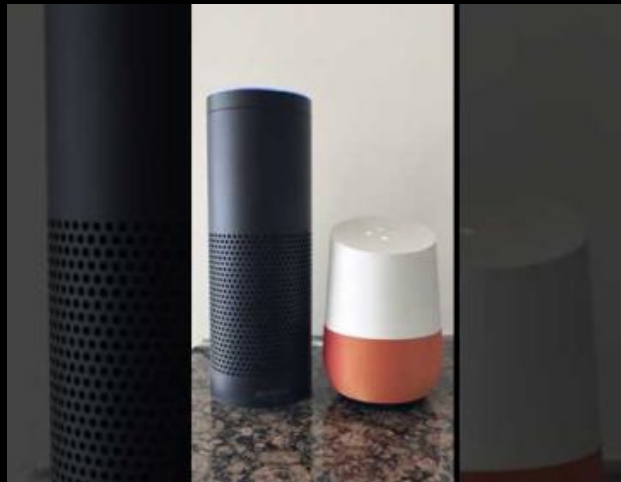
- Machine Learning Overview
- Machine Learning as an Optimization Problem
- Logistic Regression
- Single Machine vs Distributed Machine Learning
- Other Algorithms e.g. Random Forest
- Other ML Parallelization Techniques
- Demonstration
- Q&A

AI is Changing the World

Self-driving cars



Alexa/Google Home



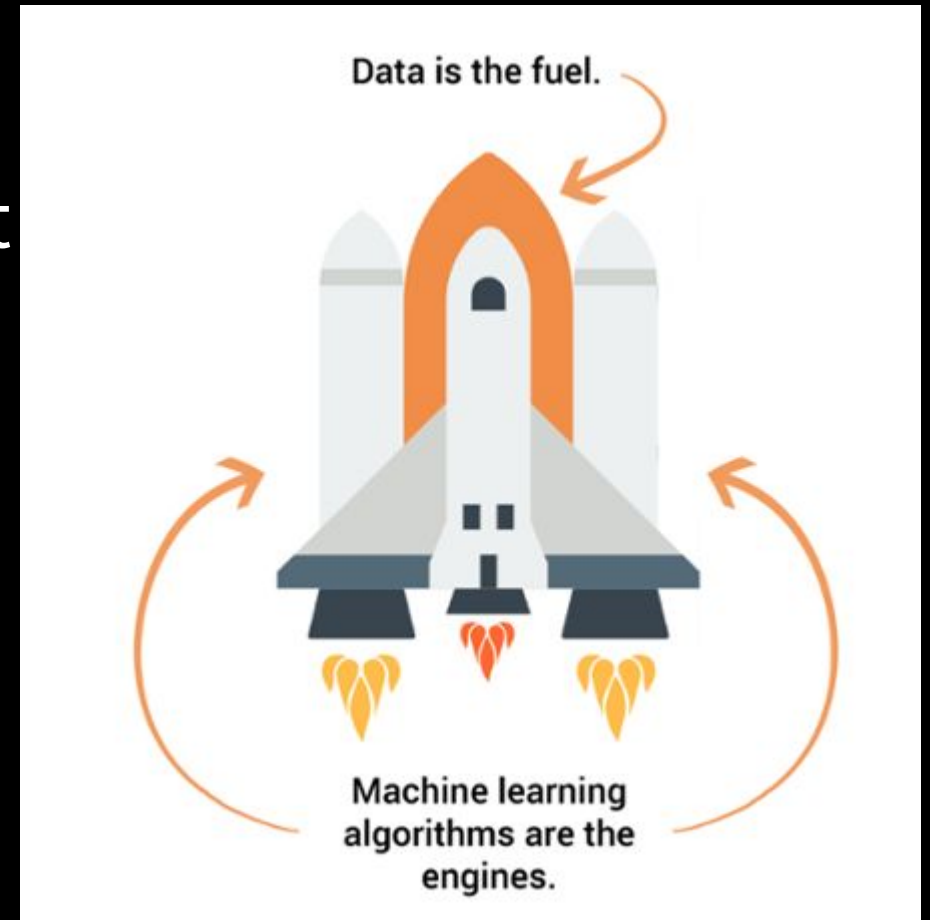
AlphaGo



Data is the Fuel that Drives ML

Andrew Ng calls the algorithms the “rocket ship” and the data “the fuel that you feed machine learning” to build deep learning applications

*Source: Buckham & Duffy



What is Machine Learning?

Supervised Machine Learning

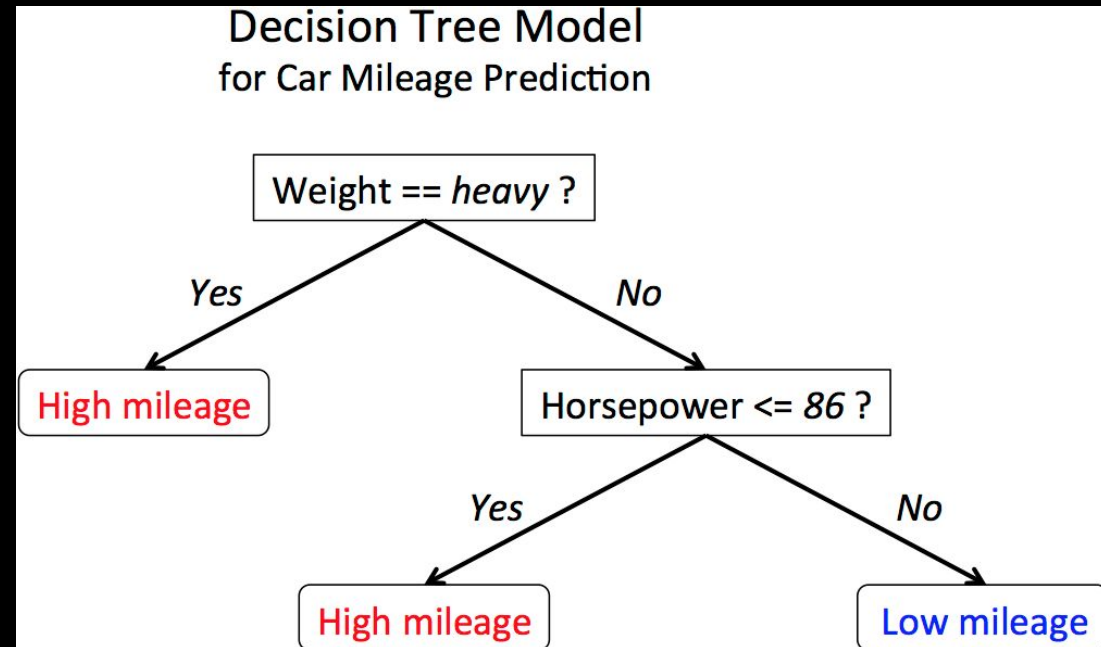
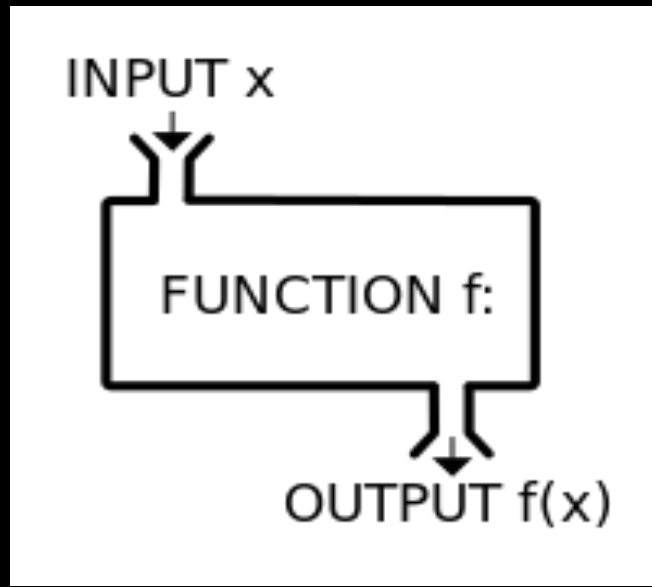
A set of techniques that, given a set of examples, attempts to predict outcomes for future values.

The set of techniques are the algorithms which includes both transformational algorithms (featurizers) and machine learning methods (estimators)

Examples are used to “train” the algorithm to produce the **Model**

A Model is a Mathematical Function

- A model is a function: $f(x)$
- Linear regression $y = b_0 + b_1x_1 + b_2x_2$



Example Bank Marketing

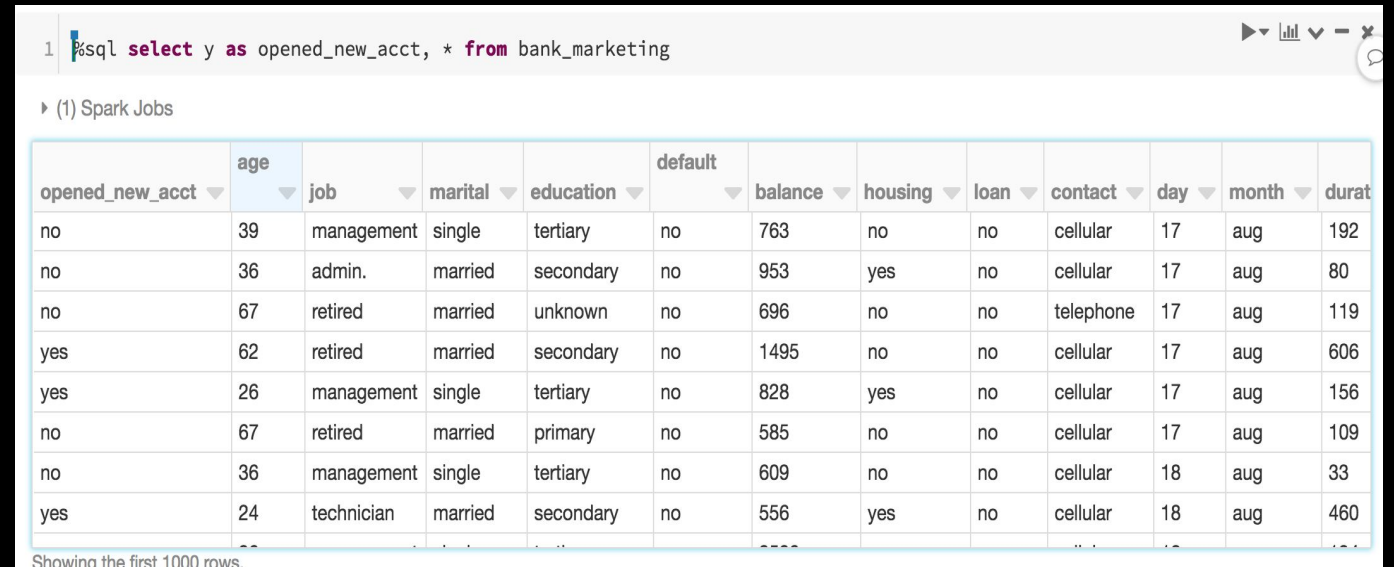
Who has heard of Signet Bank?

“data on the specific transactions of a bank’s customers can improve models for deciding what product offers to make.”¹

(1) *Data Science for Business*

Provost, Foster

O’Reilly 2013



The screenshot shows a Databricks SQL interface. At the top, a SQL query is entered: `select y as opened_new_acct, * from bank_marketing`. Below the query, it indicates "(1) Spark Jobs". The main part of the interface is a table displaying the first 1000 rows of data. The table has 13 columns: `opened_new_acct`, `age`, `job`, `marital`, `education`, `default`, `balance`, `housing`, `loan`, `contact`, `day`, `month`, and `duration`. The data rows show various customer profiles, such as a 39-year-old in management, a 36-year-old in administration, and a 24-year-old technician.

opened_new_acct	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration
no	39	management	single	tertiary	no	763	no	no	cellular	17	aug	192
no	36	admin.	married	secondary	no	953	yes	no	cellular	17	aug	80
no	67	retired	married	unknown	no	696	no	no	telephone	17	aug	119
yes	62	retired	married	secondary	no	1495	no	no	cellular	17	aug	606
yes	26	management	single	tertiary	no	828	yes	no	cellular	17	aug	156
no	67	retired	married	primary	no	585	no	no	cellular	17	aug	109
no	36	management	single	tertiary	no	609	no	no	cellular	18	aug	33
yes	24	technician	married	secondary	no	556	yes	no	cellular	18	aug	460

Showing the first 1000 rows.

But ...

given a set of examples how do I get training data to the function (i.e. model)?

```
1 %sql select y as opened_new_acct, * from bank_marketing
```

► (1) Spark Jobs

opened_new_acct	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration
no	39	management	single	tertiary	no	763	no	no	cellular	17	aug	192
no	36	admin.	married	secondary	no	953	yes	no	cellular	17	aug	80
no	67	retired	married	unknown	no	696	no	no	telephone	17	aug	119
yes	62	retired	married	secondary	no	1495	no	no	cellular	17	aug	606
yes	26	management	single	tertiary	no	828	yes	no	cellular	17	aug	156
no	67	retired	married	primary	no	585	no	no	cellular	17	aug	109
no	36	management	single	tertiary	no	609	no	no	cellular	18		
yes	24	technician	married	secondary	no	556	yes	no	cellular	18		

Showing the first 1000 rows.

Given this data ...

I want to predict if my customer will do this

I want to create this Logistic regression function

$$p(x) = \frac{\exp(b_0 + b_1 x)}{1 + \exp(b_0 + b_1 x)}$$

Machine Learning is an Optimization Problem

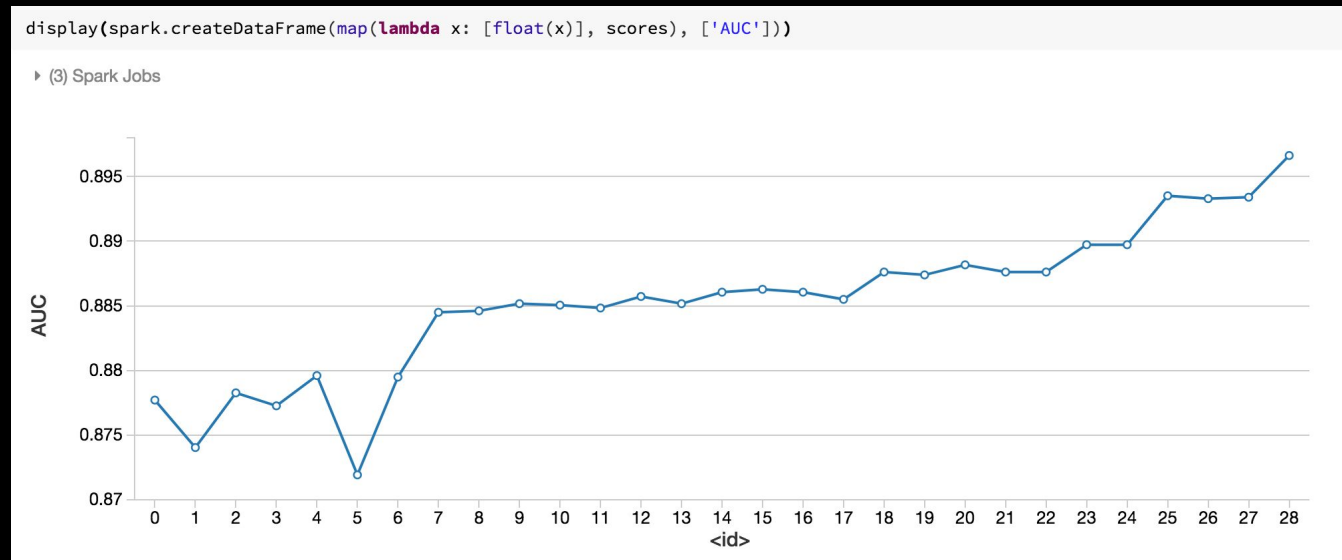
- Given the training data how do I *as quickly as possible* come up with the *most optimal equation* given *training data X* that solves for Y
- In generalized linear regression, the goal is to **learn** the coefficients (i.e. weights) based on the examples and fit a line based on the points
- In decision tree learning, we are learning where to split on features
- Time can be saved by *minimizing the amount of work* and optimization is achieved *when the result produced yields the least error on the test set*
- Getting to the optimal model is called **convergence**

Error

The goal of the optimization is to either maximize a positive metric like the area under the curve (AUC) or minimize a negative metric (like error.)

This goal is called the objective function (or the loss / cost function.)

Over each iteration the AUC goes up.



Train - Tune - Test

Data is split into samples

Training is used to fit the model(s)

The tuning (or validation) set is used to find the optimal parameters

The test set is used to evaluate the final model



Source: <https://towardsdatascience.com/train-validation-and-test-sets-72cb40cba9e7>

Amount of Work

- Big O Notation - provides a worst case scenario for how long an algorithm will take relative to the amount of data (or number of iterations)
- In Spark MLlib we always want scalable machine learning so that the runtime doesn't explode when we get massive datasets
- Example: finding a single element in a set of 1M values (say it takes 0.001 seconds per comparison):
 - Item by Item Search - **$O(N)$** :
 - Worst Case - 16 minutes
 - Binary Search - **$O(\log N)$**
 - Worst Case - 6 milliseconds
 - Hash Search - **$O(1)$**
 - Worst Case - 1 millisecond

Data Representation

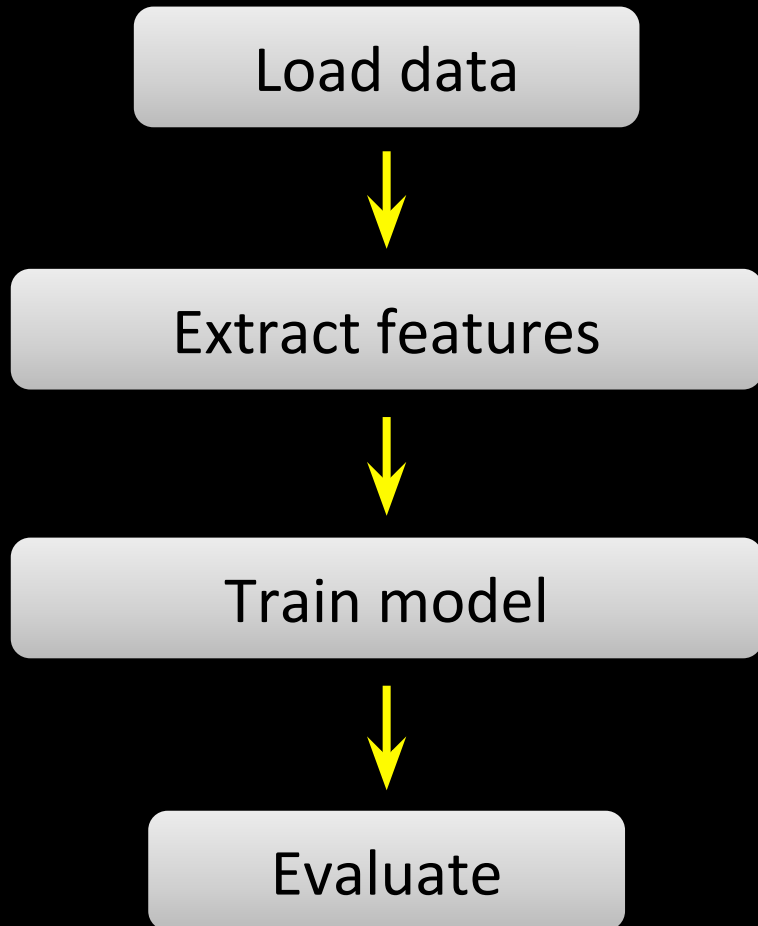
Pandas - DataFrames represented on a single machine as Python data structures

RDDs - Spark's foundational structure Resilient Distributed Dataset is represented as a reference to partitioned data without types

DataFrames - Spark's strongly typed optimized distributed collection of rows

All ML Algorithms need to represent data as vectors (array of Double values) in order to train machine learning algorithms

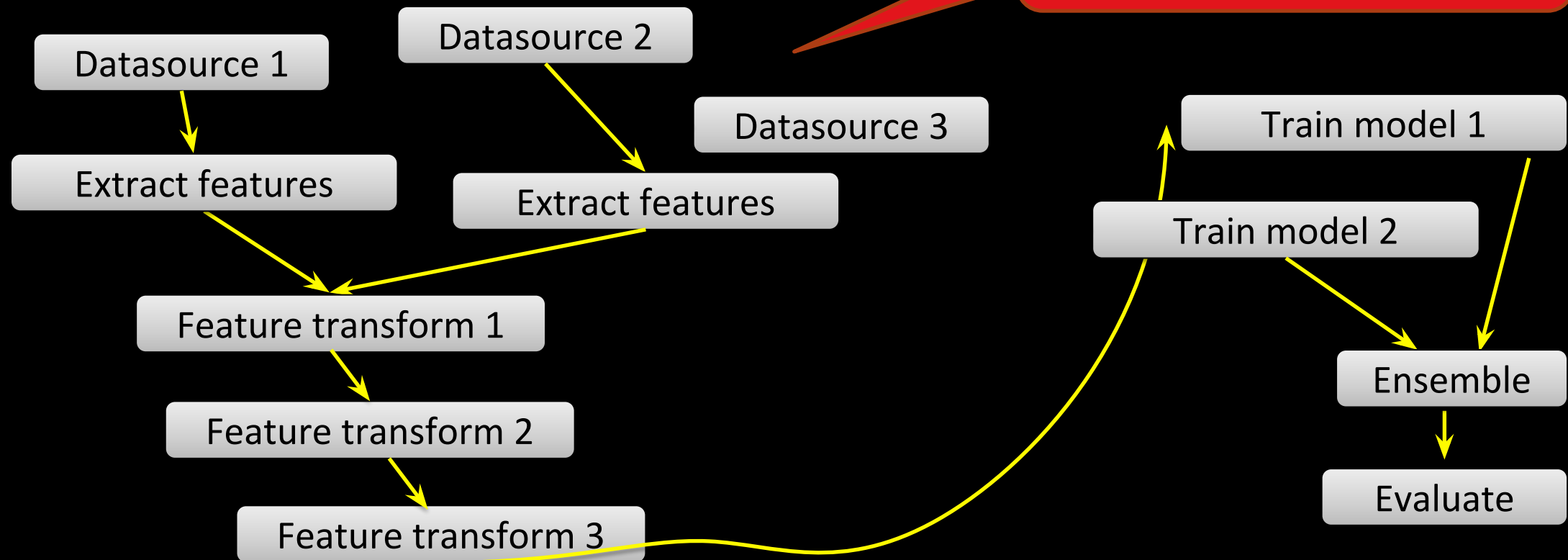
ML Pipelines



A very simple pipeline

ML Pipelines

A real pipeline!

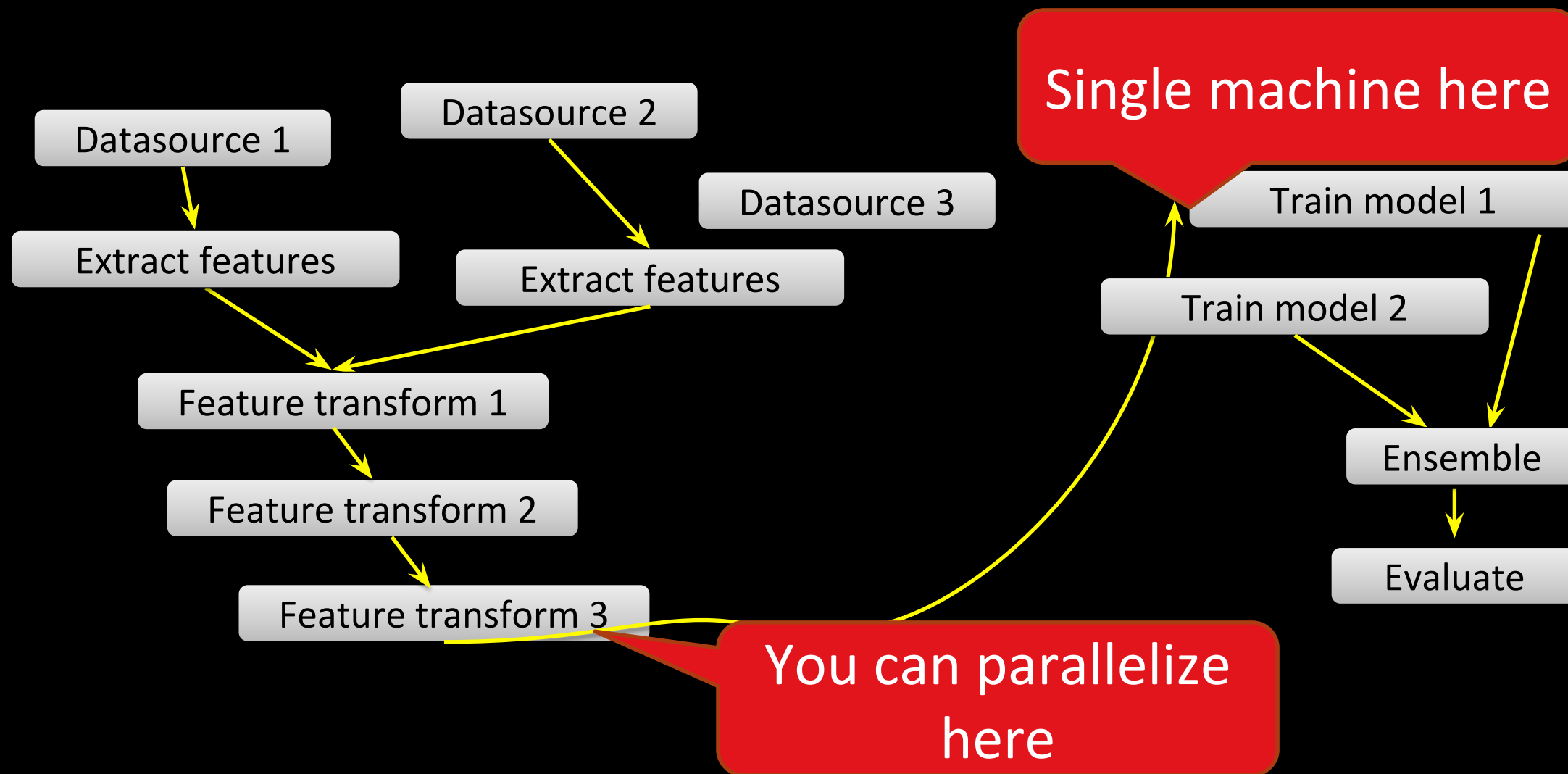


Parallelization Technique # 1

Feature engineer in Spark but train the model on a single driver machine

- Gather source data, perform joins, do complex ETL and feature engineer
- Works well when you have big data for your transactions but smaller aggregated or subsamples once you have features to train on
- May need to use a larger size driver node that can fit all the training features into memory (may need to adjust `spark.driver.maxResultSize`)
- Use Scikit-learn, R, TensorFlow (optionally with GPU) or other single machine learners

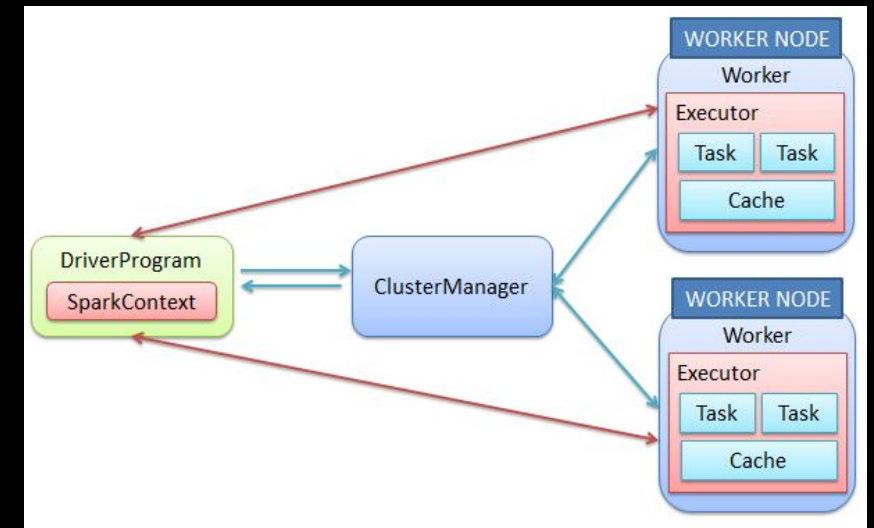
Parallelization in ML Pipelines



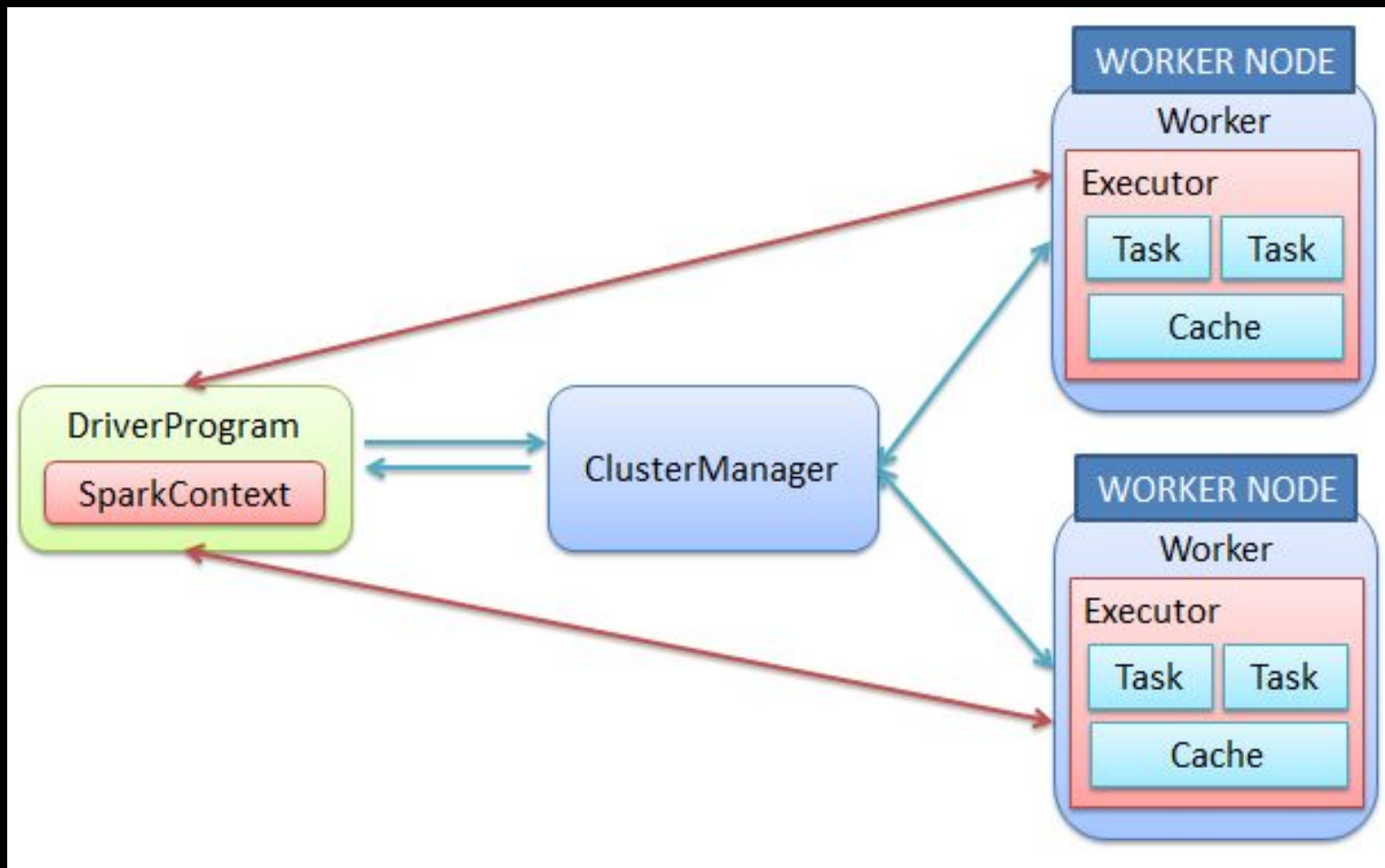
Parallelization Technique # 2

Train the entire model with distributed machine learning

- MLlib built for large scale model creation (millions of instances)
- You will create a single model with all this data (you use model selection techniques like Train Tune Validation splits or a cross validator)
- Spark Deep Learning, Horvod (for TensorFlow), XGBoost (the JVM version) and H2O also support distributed ML



Spark Architecture



Parallelization Technique # 3

Create many models one on each worker

- Works for creating one model per customer, one model per set of features for feature selection, one model per set of hyperparameters for tuning...
- Spark-sklearn helps facilitate creating one model per worker
- You can as of Spark 2.3 use the cross validation parallelism parameter to run do model selection / hyperparameter tuning in parallel in MLlib as well
- Broadcast your data but use Spark to train models on each worker with different sets of hyperparameters to find the optimal model

Single Machine Learning

In a single machine algorithm such as scikit-learn, the data is loaded into the memory of a single machine (usually as Pandas or NumPy vectors)

The single machine algorithm iterates (loops) over the data in memory and using a single thread (sometimes multicore) takes a “guess” at the equation

The error is calculated and if the error from the previous iteration is less than the tolerance or the iterations are \geq max iterations then STOP

Distributed Machine Learning

In Distributed Machine Learning like Spark MLlib, the data is represented as an RDD or a DataFrame typically in distributed file storage such as S3, Blob Store or HDFS

As a best practice, you should cache the data in memory across the nodes so that multiple iterations don't have to query the disk each time

The calculation of the gradients is distributed across all the nodes using Spark's distributed compute engine (similar to MapReduce)

After each iteration, the results return to the driver and iterations continue until either `max_iter` or `tol` is reached



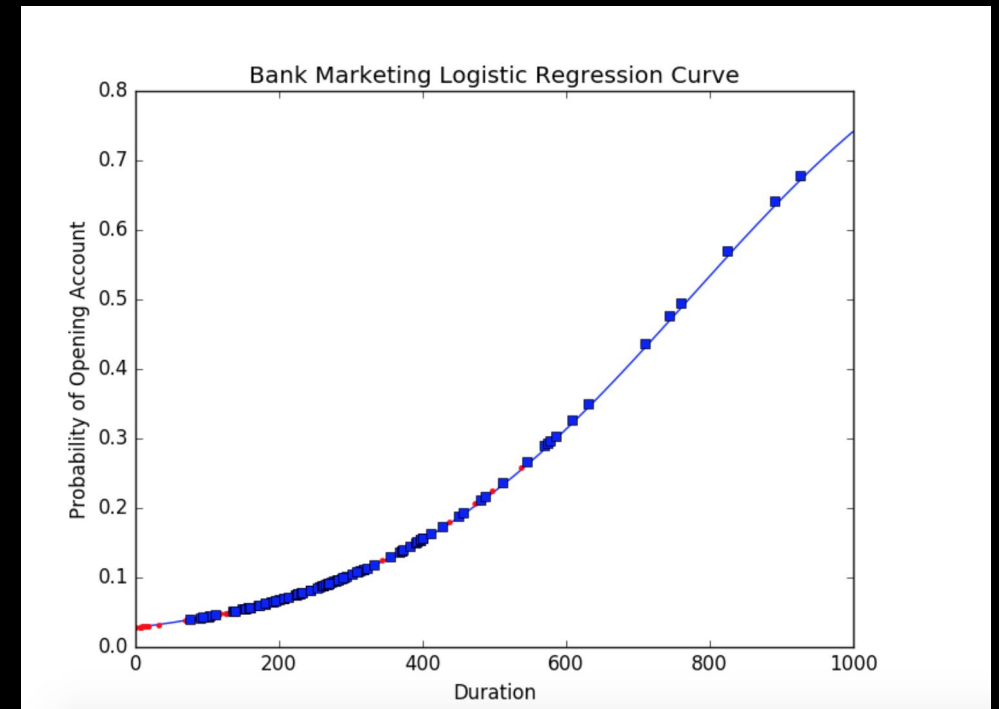
Algorithms

Logistic Regression Basics

The goal of logistic regression is to fit a sigmoid curve over the dataset.

Why not use a standard straight line?
Probabilities have to be between 0 and 1

$$p(x) = \frac{\exp(b_0 + b_1 x)}{1 + \exp(b_0 + b_1 x)}$$

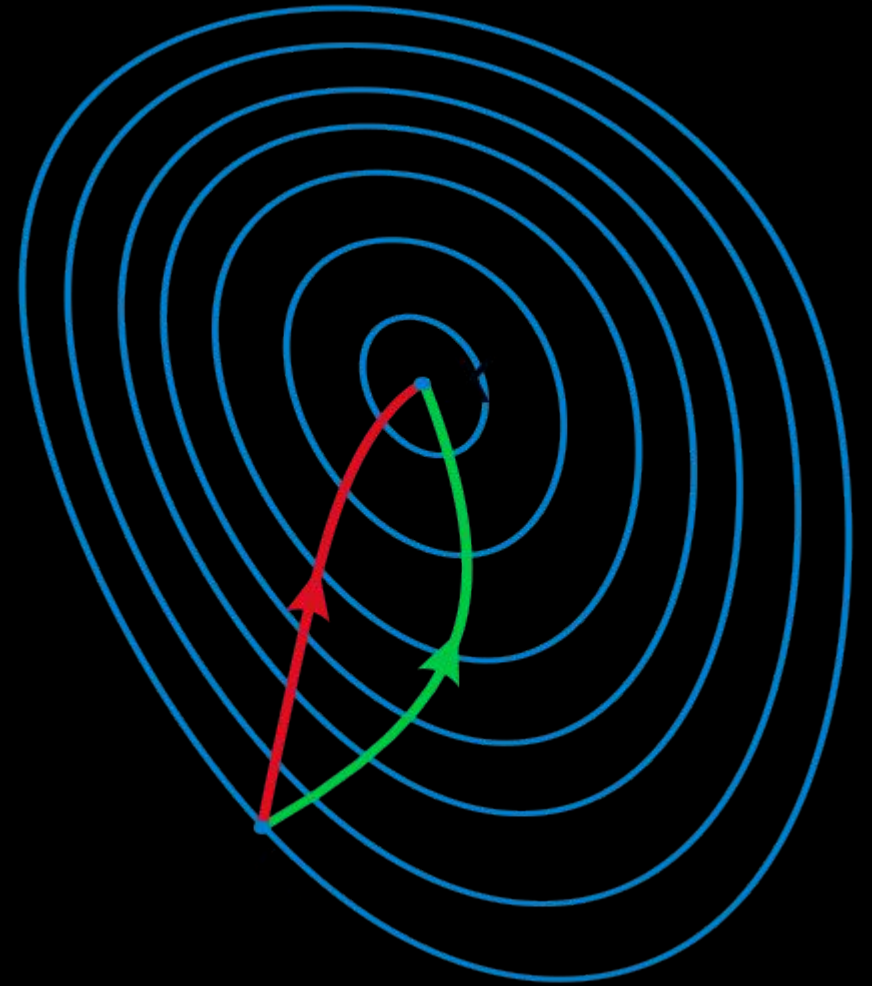


Logistic Regression Solvers

There are many methods to finding an optimal equation in Logistic regression

- Stochastic Gradient Descent
- LBFGS
- Newtonian

The basic premise is at each step you “guess” a new model by taking a step toward reducing the error or increasing the AUC



Random Forest

Random Forest uses a collection of weighted decision trees trained on random subsample of data to “vote” on the label

Random Forest is a general purpose algorithm that works well for larger datasets due to its ability to find non-linearities in the data

Because Random Forest uses a number of trees it “averages out” outliers and noise and therefore is resistant to overfitting

Each decision tree has depth and a number of decision trees in the forest

Single vs Distributed Random Forest

In Spark because each tree is trained on a subsample of data (and features) and do not depend on the other trees many trees can be trained in parallel.

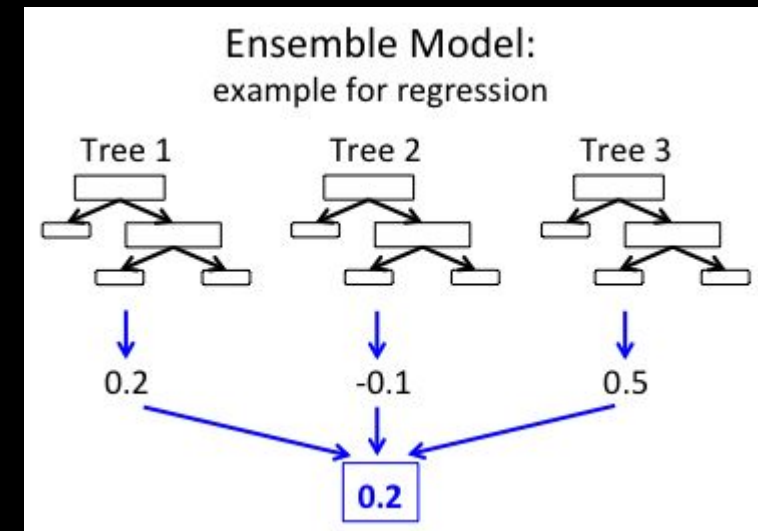
The selection of features to split on is also distributed

Splits are chosen based on information gain

Tree creation is stopped when maxDepth is

hit, information gain is $< \text{minInfoGain}$ or

no split candidates have minInstancesPerNode



Demonstration

<https://community.cloud.databricks.com/?o=1526931011080774#notebook/2040911831196221/command/1744427503559073>

Databricks Community Edition or Free Trial

<https://databricks.com/try-databricks>

Additional Questions?

Contact us at <http://go.databricks.com/contact-databricks>