

# Building Custom Machine Learning Algorithms with Apache SystemML

Fred Reiss

Chief Architect, IBM Spark Technology Center

Member, IBM Academy of Technology



SPARK SUMMIT 2016  
DATA SCIENCE AND ENGINEERING AT SCALE

# Roadmap

- What is Apache SystemML?
- Demo!
- How to get SystemML

# What is Apache SystemML?



SPARK SUMMIT 2016  
DATA SCIENCE AND ENGINEERING AT SCALE

# Origins of the SystemML Project

You are  
here.

2015

2016



**2011**

**2012**

**2013**

**2014**



**2007-2008:** Multiple projects at IBM Research – Almaden involving machine learning on Hadoop.

**2009:** We form a **dedicated team** for scalable ML

**2009-2010:** Through engagements with customers, we observe how data scientists create **ML solutions**.

2007

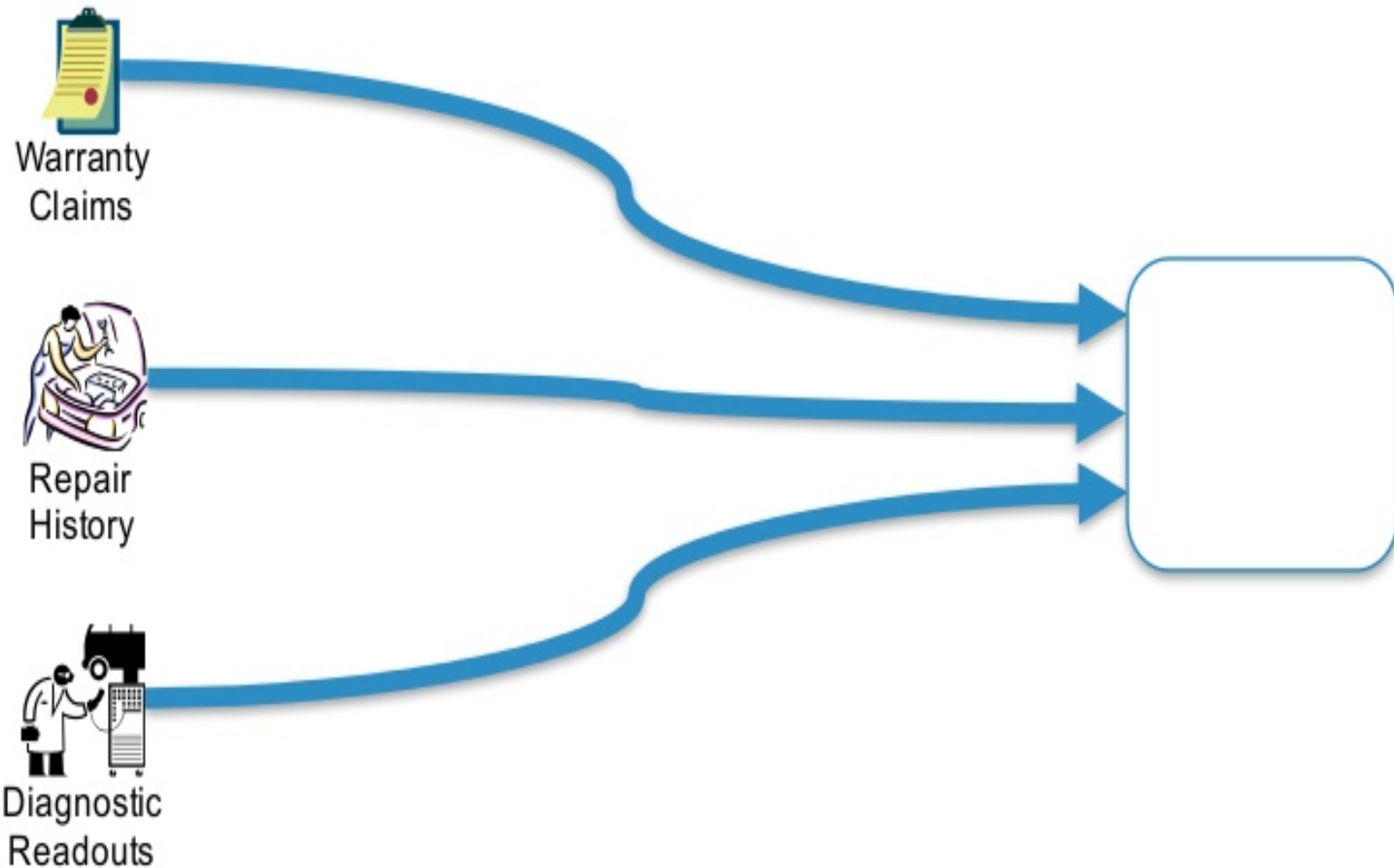
2008

2009

2010

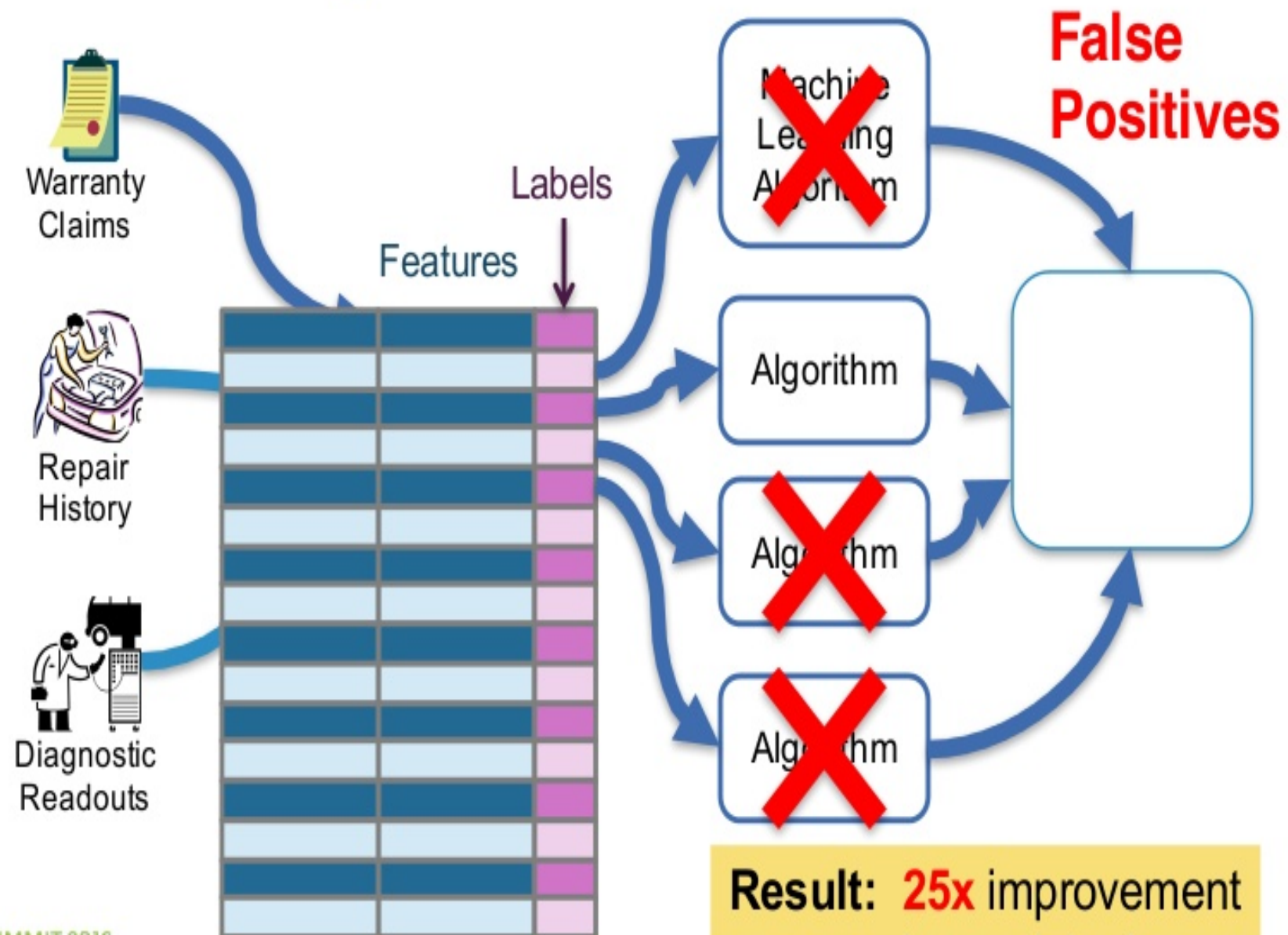


# Case Study: An Auto Manufacturer



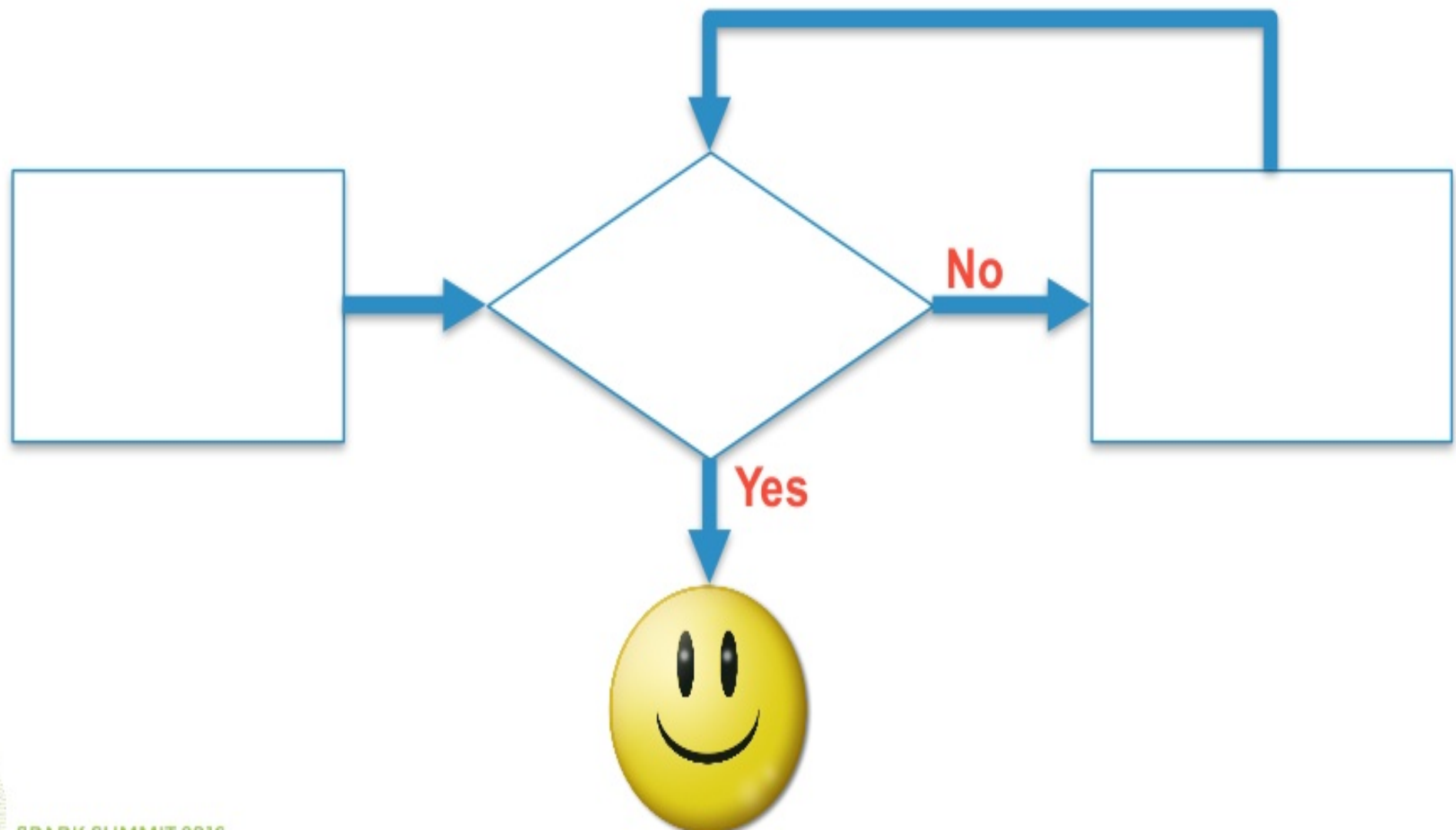


# Case Study: An Auto Manufacturer





# The Iterative Development Process



# State-of-the-Art: Small Data

Data  
Scientist



```
1 X = read.csv("X.csv") # explanatory variables
2 y = read.csv("Y.csv") # predicted variables
3
4 n = nrow(X)
5 m = ncol(X)
6
7 # Reshape the columns of X to fit the model
8 scale_lambda = matrix(1, nrow = 1, col = m)
9 lambda = t(scale_lambda) * $reg
10
11 # Compute the cross-validated error
12 A = t(X) %*% X + lambda
13 b = t(X) %*% y
14
15 beta = solve(A, b)
16
17 write.csv(beta, "beta.csv")
```

R or  
Python

Protein data set 200

Protein name	Seq. length (aa)	Mr	Mn	Mr/Mn	Inst. (kDa)	Mr/Mn
ADAMTS-1	1000	1000	1000	1.00	1000	1000
ADAMTS-2	1000	1000	1000	1.00	1000	1000
ADAMTS-3	1000	1000	1000	1.00	1000	1000
ADAMTS-4	1000	1000	1000	1.00	1000	1000
ADAMTS-5	1000	1000	1000	1.00	1000	1000
ADAMTS-6	1000	1000	1000	1.00	1000	1000
ADAMTS-7	1000	1000	1000	1.00	1000	1000
ADAMTS-8	1000	1000	1000	1.00	1000	1000
ADAMTS-9	1000	1000	1000	1.00	1000	1000
ADAMTS-10	1000	1000	1000	1.00	1000	1000
ADAMTS-11	1000	1000	1000	1.00	1000	1000
ADAMTS-12	1000	1000	1000	1.00	1000	1000
ADAMTS-13	1000	1000	1000	1.00	1000	1000
ADAMTS-14	1000	1000	1000	1.00	1000	1000
ADAMTS-15	1000	1000	1000	1.00	1000	1000
ADAMTS-16	1000	1000	1000	1.00	1000	1000
ADAMTS-17	1000	1000	1000	1.00	1000	1000
ADAMTS-18	1000	1000	1000	1.00	1000	1000
ADAMTS-19	1000	1000	1000	1.00	1000	1000
ADAMTS-20	1000	1000	1000	1.00	1000	1000

Data



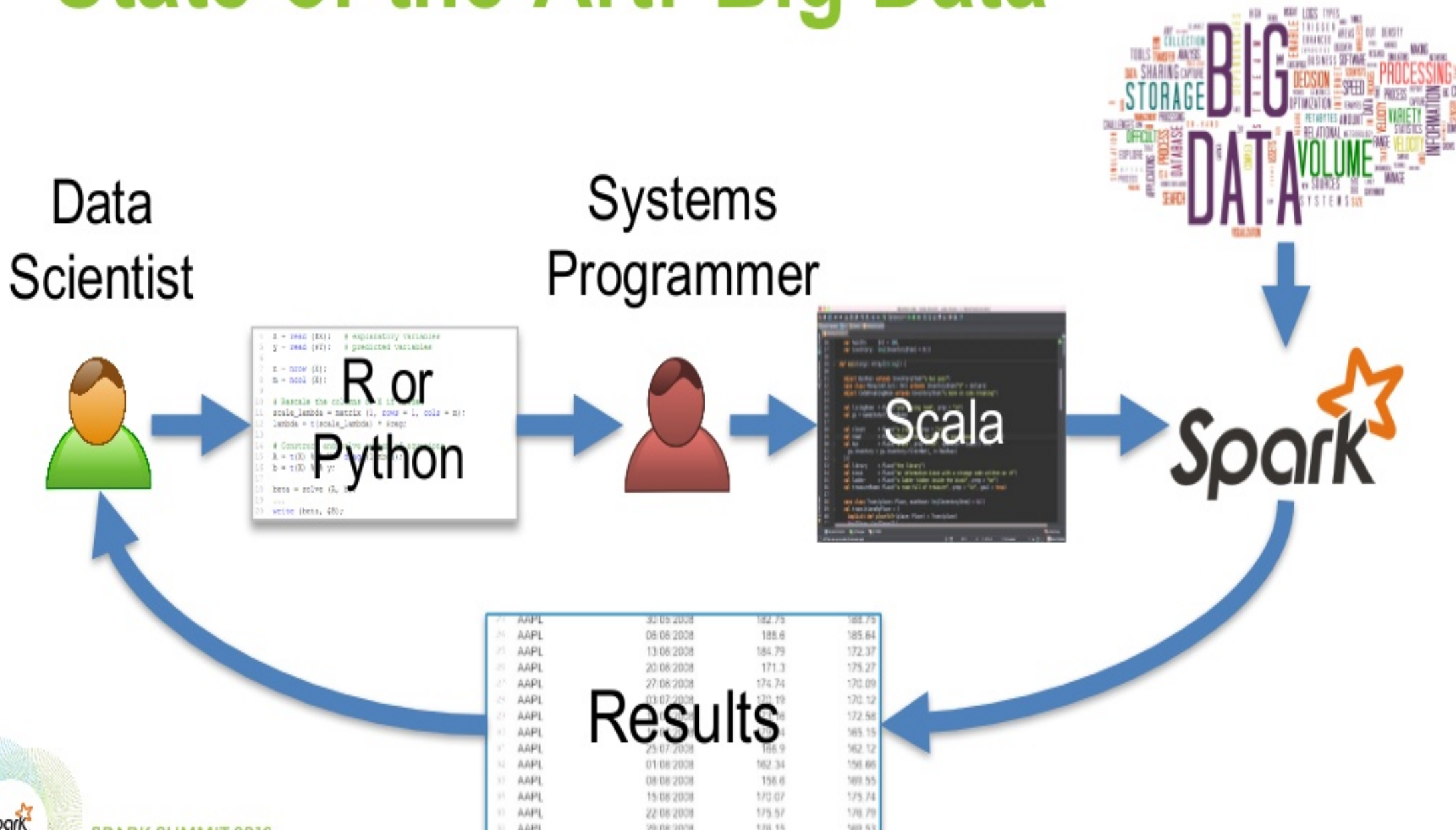
Personal  
Computer

Results

Symbol	Date	Open	High	Low
AAPI	2018-01-01	100.00	100.00	100.00
AAPI	2018-01-02	100.00	100.00	100.00
AAPI	2018-01-03	100.00	100.00	100.00
AAPI	2018-01-04	100.00	100.00	100.00
AAPI	2018-01-05	100.00	100.00	100.00
AAPI	2018-01-08	100.00	100.00	100.00
AAPI	2018-01-09	100.00	100.00	100.00
AAPI	2018-01-10	100.00	100.00	100.00
AAPI	2018-01-11	100.00	100.00	100.00
AAPI	2018-01-12	100.00	100.00	100.00
AAPI	2018-01-15	100.00	100.00	100.00
AAPI	2018-01-16	100.00	100.00	100.00
AAPI	2018-01-17	100.00	100.00	100.00
AAPI	2018-01-18	100.00	100.00	100.00
AAPI	2018-01-19	100.00	100.00	100.00
AAPI	2018-01-22	100.00	100.00	100.00
AAPI	2018-01-23	100.00	100.00	100.00
AAPI	2018-01-24	100.00	100.00	100.00
AAPI	2018-01-25	100.00	100.00	100.00
AAPI	2018-01-26	100.00	100.00	100.00
AAPI	2018-01-29	100.00	100.00	100.00
AAPI	2018-01-30	100.00	100.00	100.00
AAPI	2018-01-31	100.00	100.00	100.00



# State-of-the-Art: Big Data



# State-of-the-Art: Big Data



Data  
Scientist



Days or weeks per iteration  
Errors while translating  
algorithms

Spark

Results

21	AAPL	30.05.2008	182.75	184.75
25	AAPL	06.06.2008	188.6	185.64
25	AAPL	13.06.2008	184.79	172.37
26	AAPL	20.06.2008	171.3	175.27
27	AAPL	27.06.2008	174.74	170.09
28	AAPL	03.07.2008	170.19	170.12
29	AAPL	10.07.2008	170.18	172.58
30	AAPL	17.07.2008	169.0	165.15
31	AAPL	25.07.2008	166.9	162.12
32	AAPL	01.08.2008	162.34	158.66
33	AAPL	08.08.2008	158.6	169.55
34	AAPL	15.08.2008	170.07	175.74
35	AAPL	22.08.2008	175.57	176.79
36	AAPL	29.08.2008	178.15	169.53





# The SystemML Vision



Data  
Scientist



```
1 x = read.csv(x) # explanatory variables
2 y = read.csv(y) # predicted variables
3
4 x = as.matrix(x)
5 y = as.matrix(y)
6
7 # Rescale the columns to [0, 1]
8 scale_lambda = matrix(1, nrow = 1, ncol = ncol(x))
9 lambda = t(scale_lambda) * xreg
10
11 # Construct the design matrix
12 X = t(x)
13 b = t(x) * y
14
15 beta = solve(G, b)
16
17
18 value(beta, 400)
```

R or  
Python

SystemML

Spark

Results

21	AAPL	30/05/2008	182.75	184.75
25	AAPL	06/06/2008	188.6	185.64
26	AAPL	13/06/2008	184.79	172.37
28	AAPL	20/06/2008	171.3	175.27
29	AAPL	27/06/2008	174.74	170.09
29	AAPL	03/07/2008	170.19	170.12
29	AAPL	10/07/2008	170.18	172.58
30	AAPL	17/07/2008	169.9	165.15
31	AAPL	25/07/2008	166.9	162.12
32	AAPL	01/08/2008	162.34	158.66
33	AAPL	08/08/2008	158.6	169.55
34	AAPL	15/08/2008	170.07	175.74
35	AAPL	22/08/2008	175.57	176.79
36	AAPL	29/08/2008	178.15	169.53



# The SystemML Vision

Data  
Scientist



```
1 x = read(RX) # explanatory variables
2 y = read(Y) # predicted variables
3
4 z = read(X)
5 m = read(X)
6
7 # Rescale the columns to 0..1
8 scale_labels = labels[1, rows = 1, cols = 0]
9 labels = t(scale_labels) * freg
10
11 # Construct and fit the model
12 A = t(x) %>% as.matrix()
13 b = t(x) %>% as.matrix()
14
15 beta = solve(A, b)
16
17 write(beta, 20)
```

R or  
Python



Fast iteration  
Same answer

SystemML



Results

21	AAPL	30-05-2008	182.75	188.75
22	AAPL	06-06-2008	188.6	185.64
23	AAPL	13-06-2008	184.79	172.37
24	AAPL	20-06-2008	171.3	175.27
25	AAPL	27-06-2008	174.74	170.09
26	AAPL	03-07-2008	170.19	170.12
27	AAPL	10-07-2008	165.16	172.58
28	AAPL	17-07-2008	165.16	165.15
29	AAPL	24-07-2008	166.9	162.12
30	AAPL	01-08-2008	162.34	156.66
31	AAPL	08-08-2008	156.6	169.55
32	AAPL	15-08-2008	170.07	175.74
33	AAPL	22-08-2008	175.57	176.79
34	AAPL	29-08-2008	176.15	169.53





**2007-2008:** Multiple projects at IBM Research – Almaden involving machine learning on Hadoop.

**2009:** We form a dedicated team for scalable ML

**2009-2010:** Through engagements with customers, we observe how data scientists create machine learning algorithms.

2007

2008

2009

2010



# Research

2011

2012

2013

2014



# Apache SystemML

**June 2015:** IBM  
Announces open-  
source SystemML

**November 2015:**  
SystemML enters  
Apache incubation

**June 2016:**  
Second Apache  
release (0.10)

**September 2015:**  
Code available on  
Github

**February 2016:**  
First release (0.9) of  
Apache SystemML

2015

2016



# SystemML at IBM Watson Health

- Built algorithms for predicting treatment outcomes
  - Substantial improvement in accuracy
- Moved from Hadoop MapReduce to Spark
  - SystemML supports both frameworks
  - **Exact same code**
  - **300X faster** on 1/40<sup>th</sup> as many nodes



# SystemML at Cadent Technology



Cadent is a leading provider of TV advertising and data solutions, reaching over 140 million homes and trusted by the world's largest service providers.

*“SystemML allows Cadent to implement advanced numerical programming methods in Apache Spark, empowering us to leverage specialized algorithms in our predictive analytics software.”*

Michael Zargham  
Chief Scientist





# Demo!



**SPARK SUMMIT 2016**  
DATA SCIENCE AND ENGINEERING AT SCALE



# Demo Scenario

- **Application:** Targeted ads using demographic information tied to cookies
- **Problem:** The information is incomplete
- **Solution:** Estimate the missing values
  - Treat the problem as a **matrix completion** problem



# Data

- The U.S. Census Public Use Microdata Sample (PUMS) data set for 2010
- 10% sample of the U.S. population
  - We'll use just California today
- Use this full data set to generate synthetic incomplete data

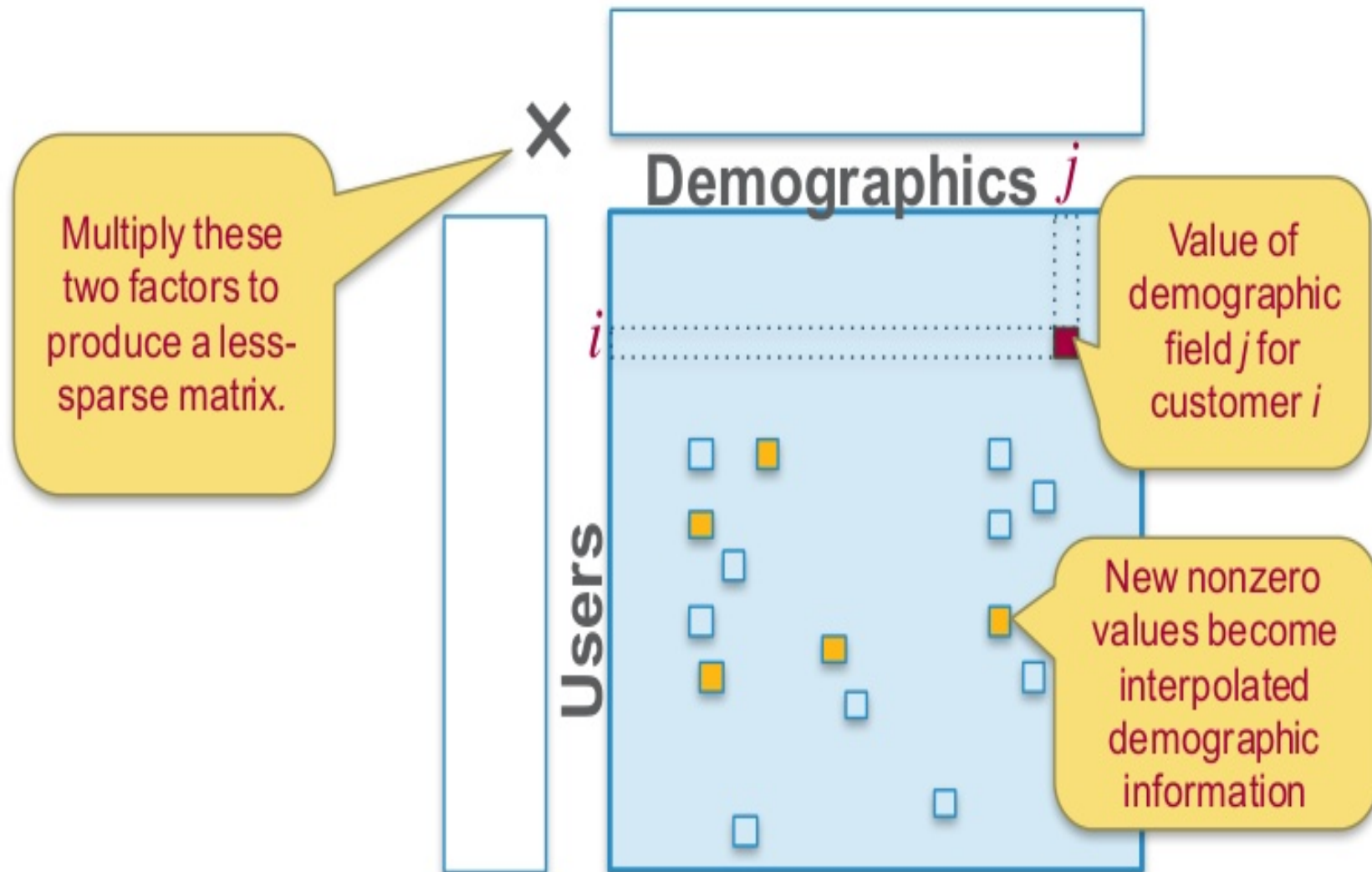


# Demo Scenario

- **Application:** Identify products that are complementary (often purchased together)
- **Problem:** Customers are not currently buying the best complements at the same time
- **Solution:** Suggest new product pairings
  - Treat the problem as a **matrix completion** problem



# Matrix Factorization



# Demo Part 1: Data wrangling

Zeppelin

Notebook

Interpreter

Configuration

Search in your notebooks

freiss

Logout

SystemML Census Demo

default

```
%spark

val baseDir = "/home/freiss/pums_2010"
val personPath = baseDir + "/person.csv"
val householdPath = baseDir + "/household.csv"

// Read in the raw data files and project out columns that are not demographic data
val personDF = sqlContext.read.format("com.databricks.spark.csv").option("header", "true").load(personPath)
  .select("SERIALNO", "STATE", "PWEIGHT", "RELATE", "OC", "RC", "SEX", "SSPA", "AGE", "QTRBIR", "HISPAN",
    "NUMRACE", "WHITE", "BLACK", "AIAN", "ASIAN", "NHAW", "OPI", "OTHER", "RACESHORT", "RACEDET", "RACECHKBX",
    "GQTP")

val householdDF = sqlContext.read.format("com.databricks.spark.csv").option("header", "true").load(householdPath)
  .select("SERIALNO", "STATE", "REGION", "DIVISION", "SUBSAMPL", "HWEIGHT",
    "PERSONS", "UNITTYPE", "HSEFLG", "VACS", "TENURE", "HHT", "P60", "P65", "P18", "NPF", "NCH", "NRCH", "PACC",
    "PARC", "UPART", "MULTG")

// Join the two tables, then project out the keys used to join
val extendedPersonDF =
  personDF.join(householdDF, personDF("SERIALNO") === householdDF("SERIALNO") and personDF("STATE") === householdDF("STATE"))
  .drop(personDF("SERIALNO")).drop(personDF("STATE")).drop(householdDF("SERIALNO")).drop(householdDF("STATE"))

val numExtendedPerson = extendedPersonDF.count()

z.put("demographics", extendedPersonDF)

baseDir: String = /home/freiss/pums_2010
personPath: String = /home/freiss/pums_2010/person.csv
householdPath: String = /home/freiss/pums_2010/household.csv
personDF: org.apache.spark.sql.DataFrame = [SERIALNO: string, STATE: string, PWEIGHT: string, RELATE: string, OC: string, RC: string, SEX: string, SSPA: string, AGE: string, QTRBIR: string, HISPAN: string, NUMRACE: string, WHITE: string, BLACK: string, AIAN: string, ASIAN: string, NHAW: string, OPI: string, OTHER: string, RACESHORT: string, RACEDET: string, RACECHKBX: string, GQTP: string]
```



# Demo Part 2: Custom algorithm

## Algorithm Customizability

ML algorithms are expressed in an R-like or Python-like syntax that includes linear algebra primitives, statistical functions, and ML-specific constructs. This high-level language significantly increases the productivity of data scientists as it provides (1) full flexibility in expressing custom analytics, and (2) data independence from the underlying input formats and physical data representations. Automatic optimization according to data and cluster characteristics ensures both efficiency and scalability.

### Poisson Nonnegative Matrix Factorization in SystemML's R-like Syntax

```
while (iter < max_iterations) {  
  iter = iter + 1;  
  H = (H * (t(W) %*% (V/(W%*%H)))) / t(colSums(W));  
  W = (W * ((V/(W%*%H)) %*% t(H))) / t(rowSums(H));  
  obj = as.scalar(colSums(W) %*% rowSums(H)) - sum(V * log(W%*%H));  
  print("iter=" + iter + " obj=" + obj);  
}
```





# Key Points

- SystemML, Spark, and Zeppelin work together
- Linear algebra is great for data science
- Customization is important



# How to get Apache SystemML



SPARK SUMMIT 2016  
DATA SCIENCE AND ENGINEERING AT SCALE

# The Apache SystemML Web Site

<http://systemml.apache.org>

The screenshot shows the Apache SystemML website. At the top left is the Apache SystemML logo, which consists of a red circle with 'ML' inside. To its right is the text 'Apache SystemML'. Below this is a large 'Apache SystemML' heading. Underneath the heading is the text 'Apache SystemML is a distributed and declarative machine learning platform.' and a blue button labeled 'Get SystemML'. On the right side, there is a navigation bar with links: 'Community', 'GitHub', 'Documentation', 'Download', and 'Apache'. The 'Community' link has a dropdown menu with 'Get Involved' and 'Who we are'. There are three callout boxes: one pointing to the 'Download' link with the text 'Download the binary release', one pointing to the 'Get Involved' link with the text 'Contribute to the project!', and one pointing to the 'Documentation' link with the text 'Browse the source!'. Another callout box points to the 'Try out some tutorials!' text on the right side of the page.

ML Apache SystemML

Download the binary release

Apache SystemML

Apache SystemML is a distributed and declarative machine learning platform.

Get SystemML

Community

Get Involved

Who we are

GitHub

Documentation

Download

Apache

Browse the source!

Try out some tutorials!

Contribute to the project!

# THANK YOU.

Please try out Apache SystemML!

<http://systemml.apache.org>

Special thanks to Nakul Jindal and Mike Dusenberry for helping with the demo!



SPARK SUMMIT 2016  
DATA SCIENCE AND ENGINEERING AT SCALE