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
JUNE 6-8, 2016 SAN FRANCISCO

Roadmap

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

What is Apache SystemML?

Origins of the SystemML Project

You are here.

2015

2016

2011 2012 2013 2014

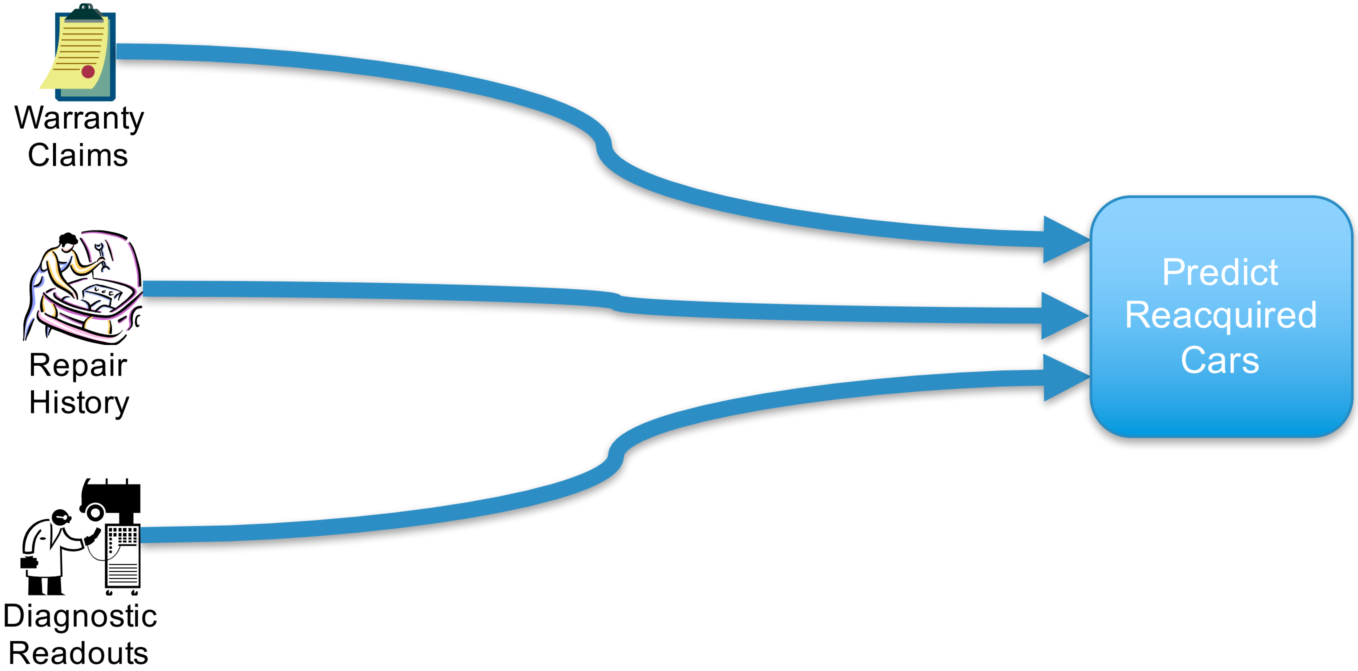
2007-2008: Multiple projects at IBM Research – Almaden involving machine learning on Hadoop.

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

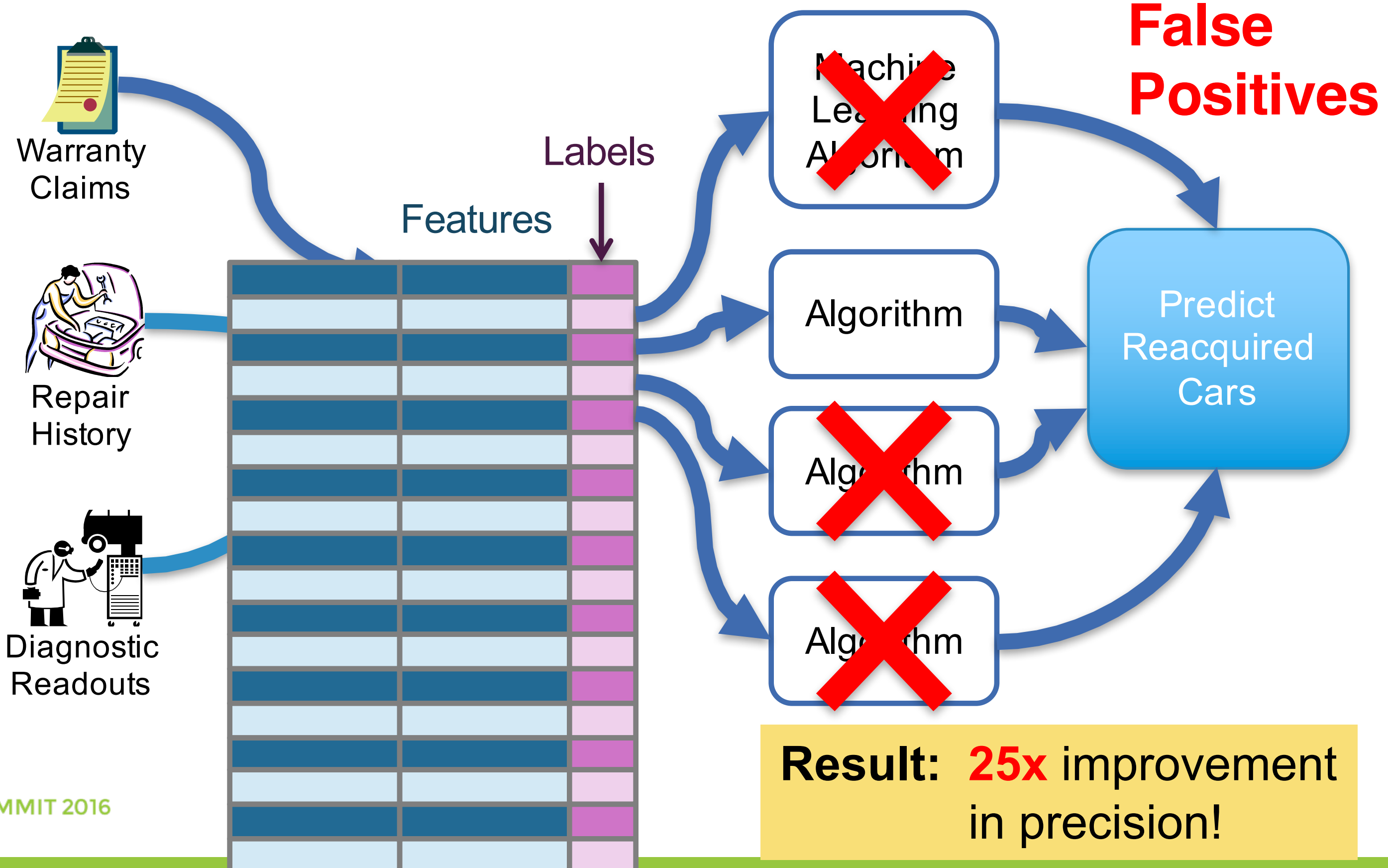
2009-2010: Through engagements with customers, we observe how data scientists create **ML solutions**.

2007 2008 2009 2010

Case Study: An Auto Manufacturer

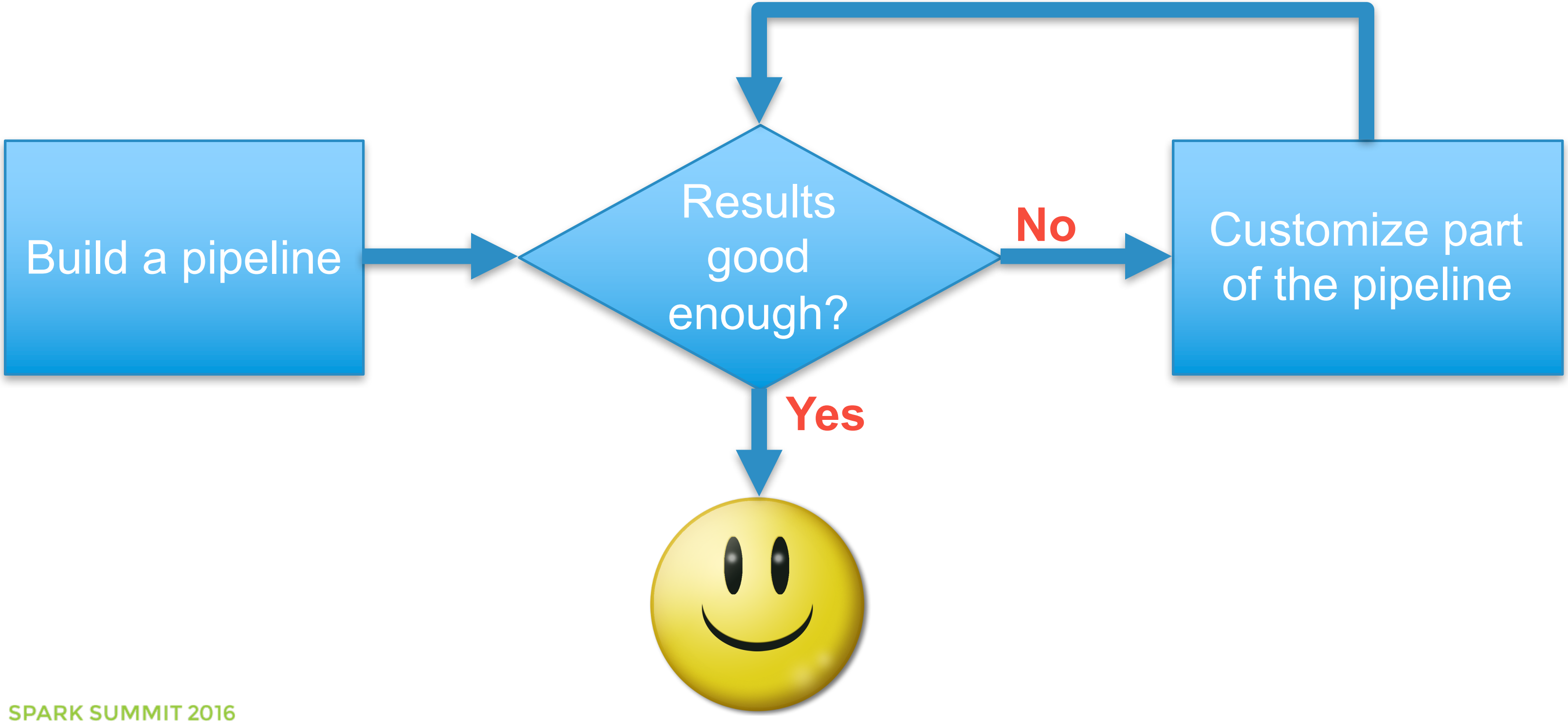


Case Study: An Auto Manufacturer

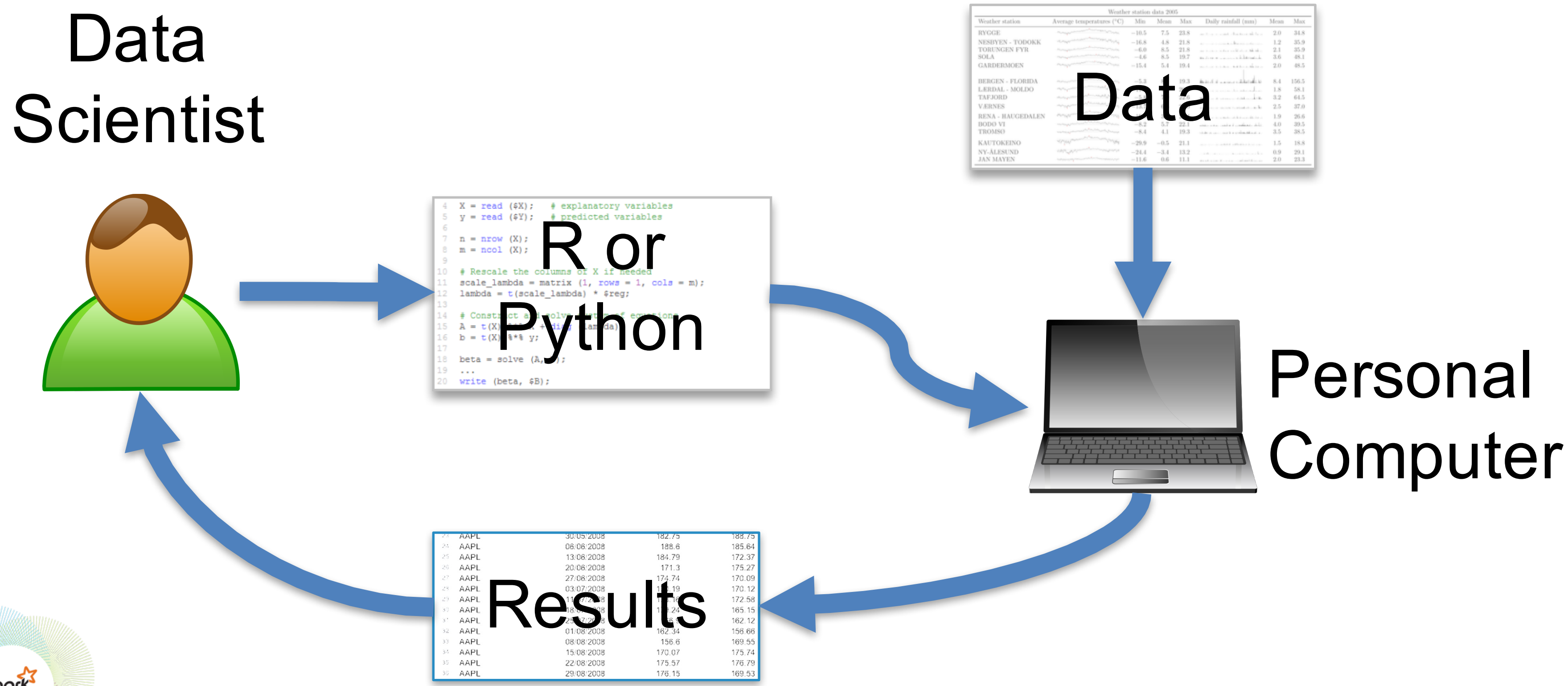


Result: **25x** improvement in precision!

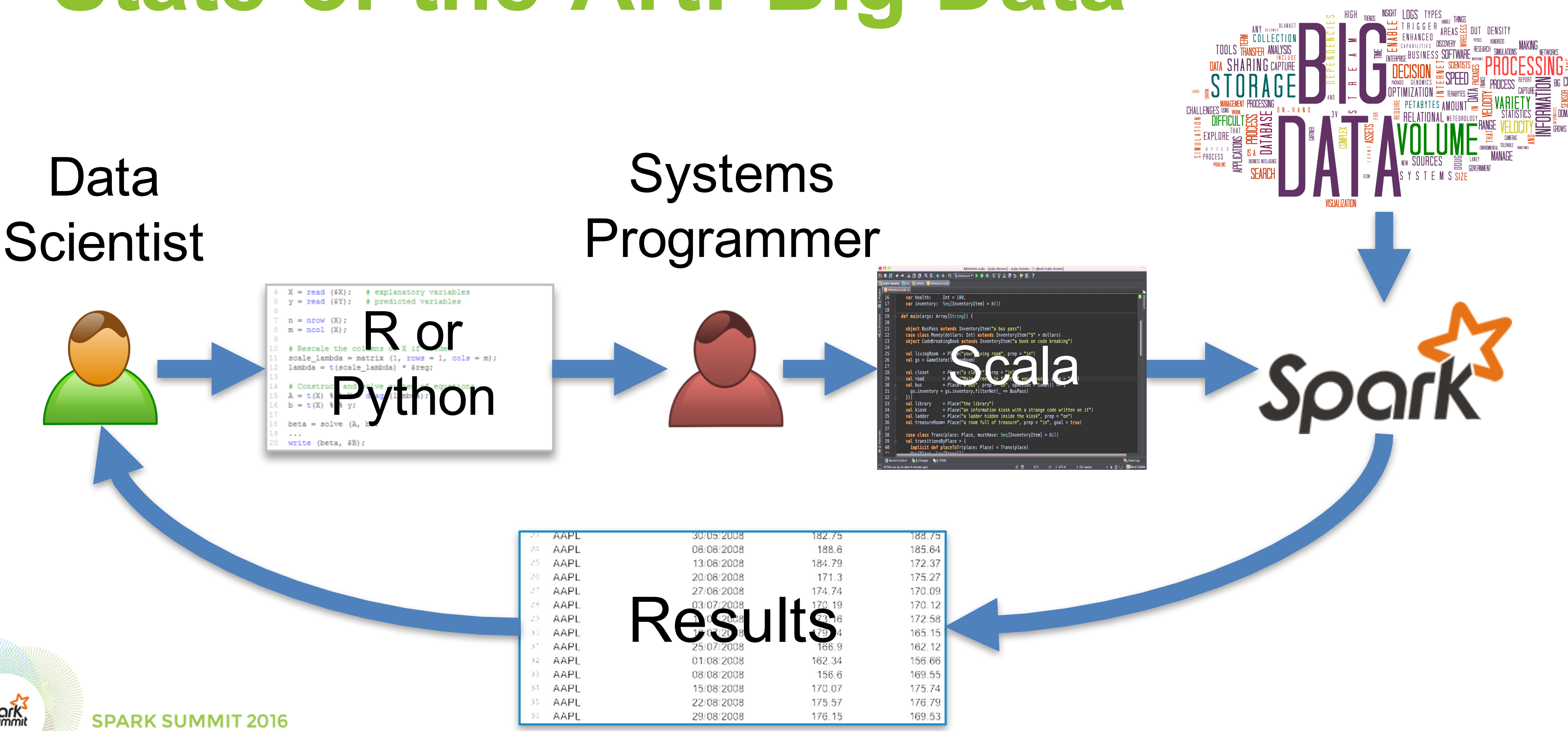
The Iterative Development Process



State-of-the-Art: Small Data

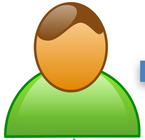


State-of-the-Art: Big Data



State-of-the-Art: Big Data

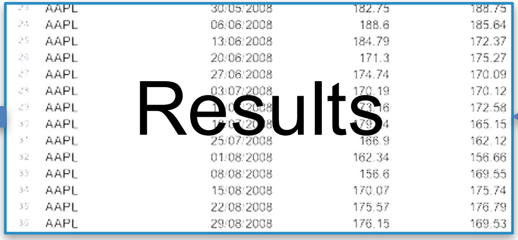
Data Scientist



😞 Days or weeks per iteration

😞 Errors while translating algorithms

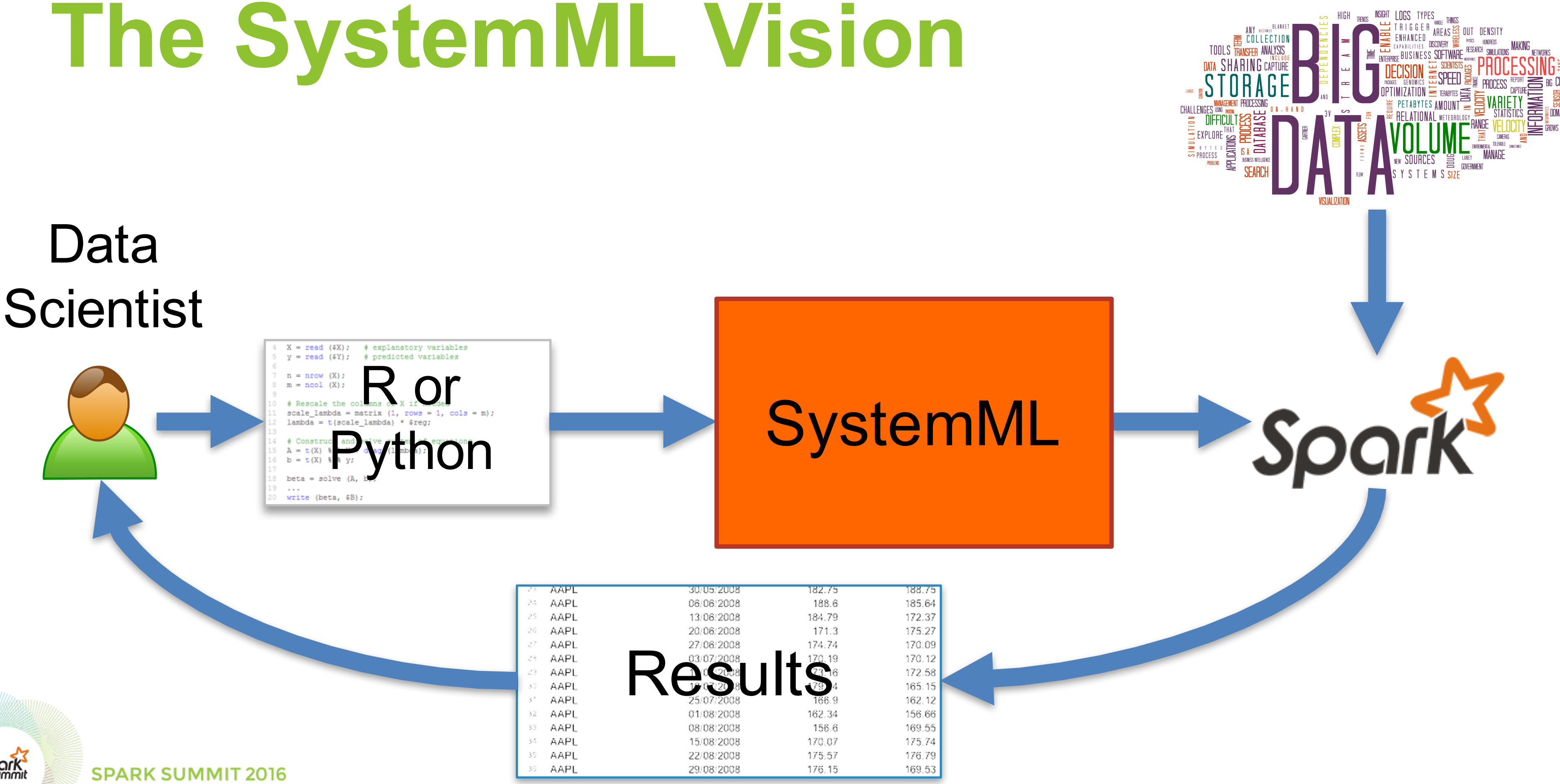
Spark



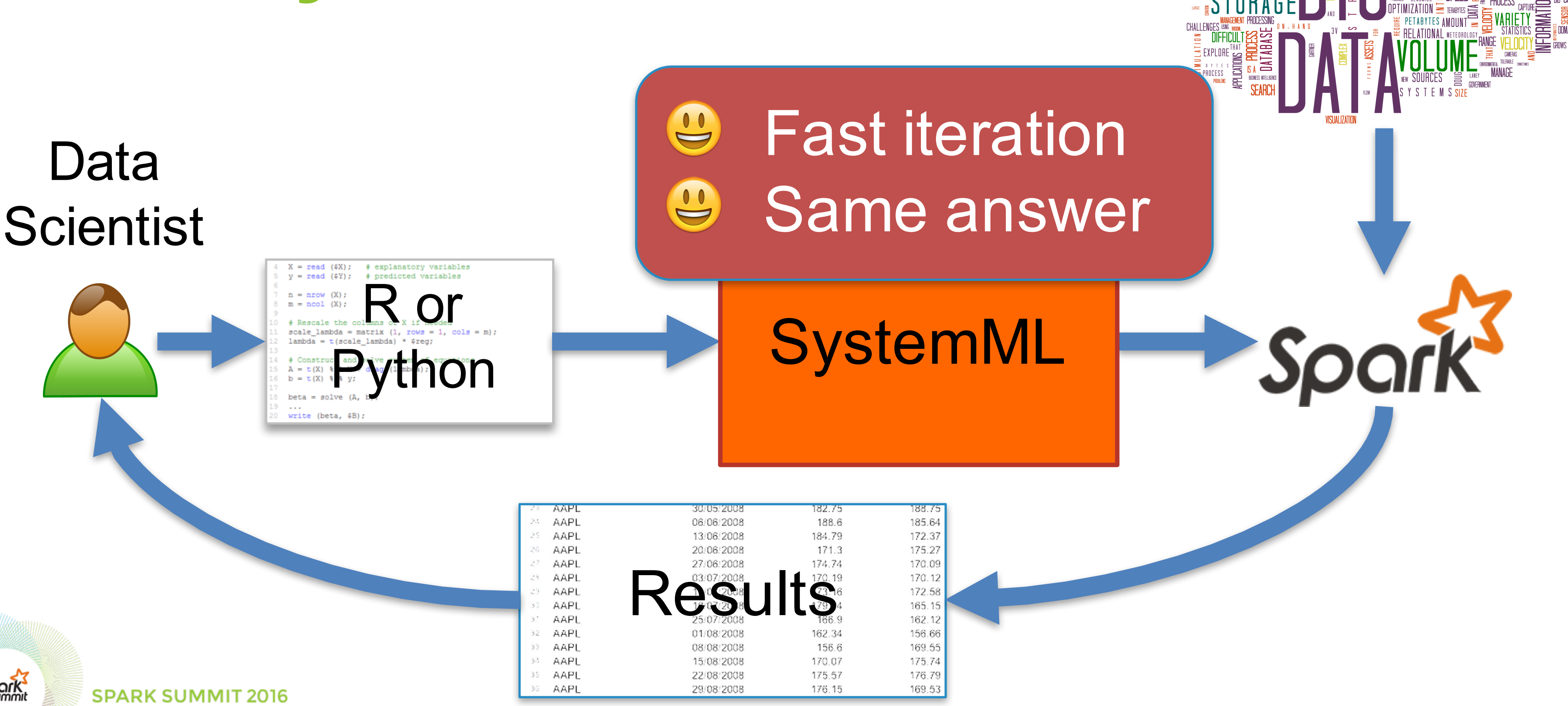
AAPL	30/05/2008	182.75	188.75
AAPL	06/06/2008	188.6	185.64
AAPL	13/06/2008	184.79	172.37
AAPL	20/06/2008	171.3	175.27
AAPL	27/06/2008	174.74	170.09
AAPL	03/07/2008	170.19	170.12
AAPL	[illegible]	[illegible]	172.58
AAPL	[illegible]	[illegible]	165.15
AAPL	25/07/2008	166.9	162.12
AAPL	01/08/2008	162.34	156.66
AAPL	08/08/2008	156.6	169.55
AAPL	15/08/2008	170.07	175.74
AAPL	22/08/2008	175.57	176.79
AAPL	29/08/2008	176.15	169.53

Results

The SystemML Vision



The SystemML Vision



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

September 2015: Code available on Github

November 2015: SystemML enters Apache incubation

February 2016: First release (0.9) of Apache SystemML

June 2016: Second Apache release (0.10)

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 1/40th 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!

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

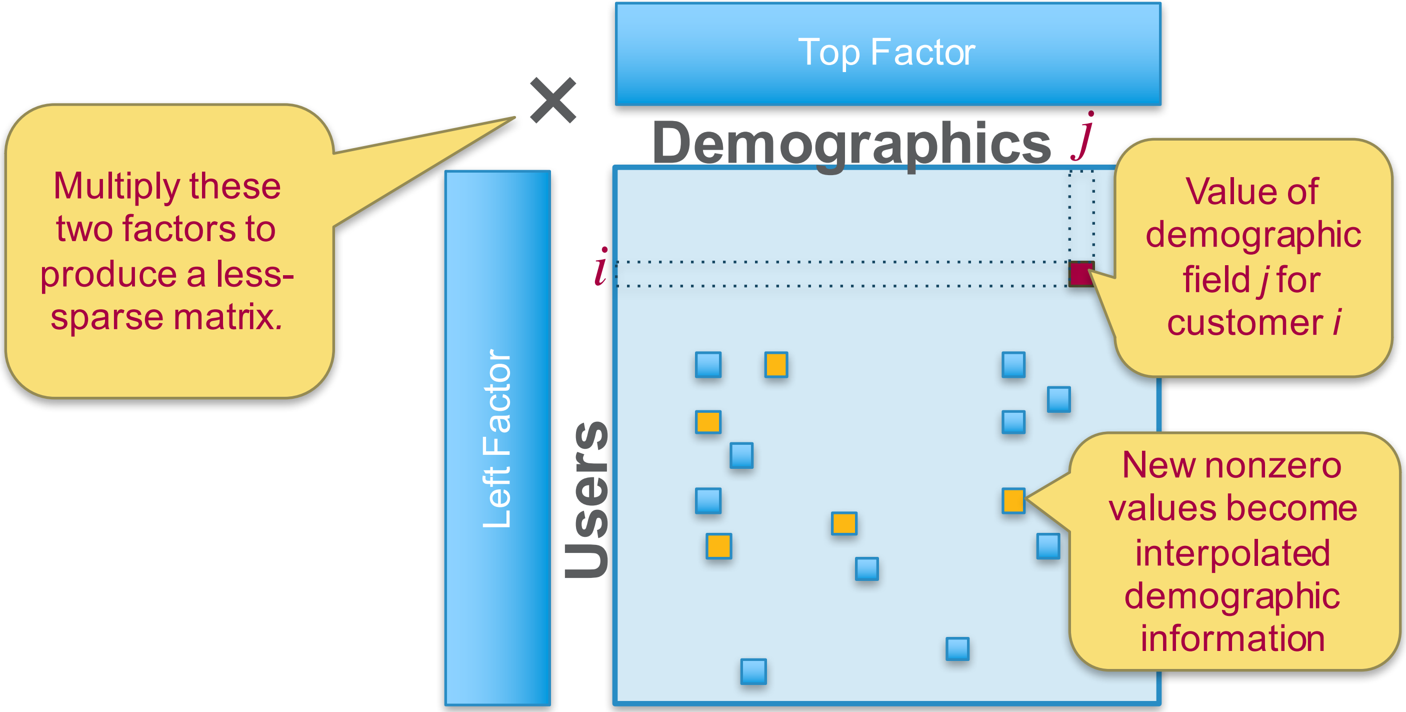
Data

- 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 frreiss Logout

SystemML Census Demo

default

FINISHED

```
%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",
    "GQTYP")

val householdDF = sqlContext.read.format("com.databricks.spark.csv").option("header","true").load(householdPath)
    .select("SERIALNO","STATE","REGION","DIVISION","SUBSAMPL","HWEIGHT",
    "PERSONS","UNITTYPE","HSUBFLG","VACS","TENURE","HHT","P60","P65","P18","NPF","NOCH","NRCH","PAOC",
    "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: strin
```

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

The Apache SystemML Web Site

http://systemml.apache.org

THANK YOU.

Please try out Apache SystemML!

http://systemml.apache.org

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