The Key to Machine Learning Is Prepping the Right Data

Building your in-memory data lake,
Applying Data Quality, and
Getting your data in shape for Machine Learning

Jean Georges Perrin / @jgperrin / Zaloni

JGP

@jgperrin

Jean Georges Perrin

Chapel Hill, NC

I🏗SW

Big Data, Analytics, Web, Software, Architecture, Small Data, DevTools, Java, NLP, Knowledge Sharer, Informix, Spark, CxO…

IBM CHAMPION

ZALONI

THE DATA LAKE COMPANY

Software, Services, Solutions

USA (Raleigh, NC), India, Dubai…

http://zaloni.com

@zaloni

And What About You?

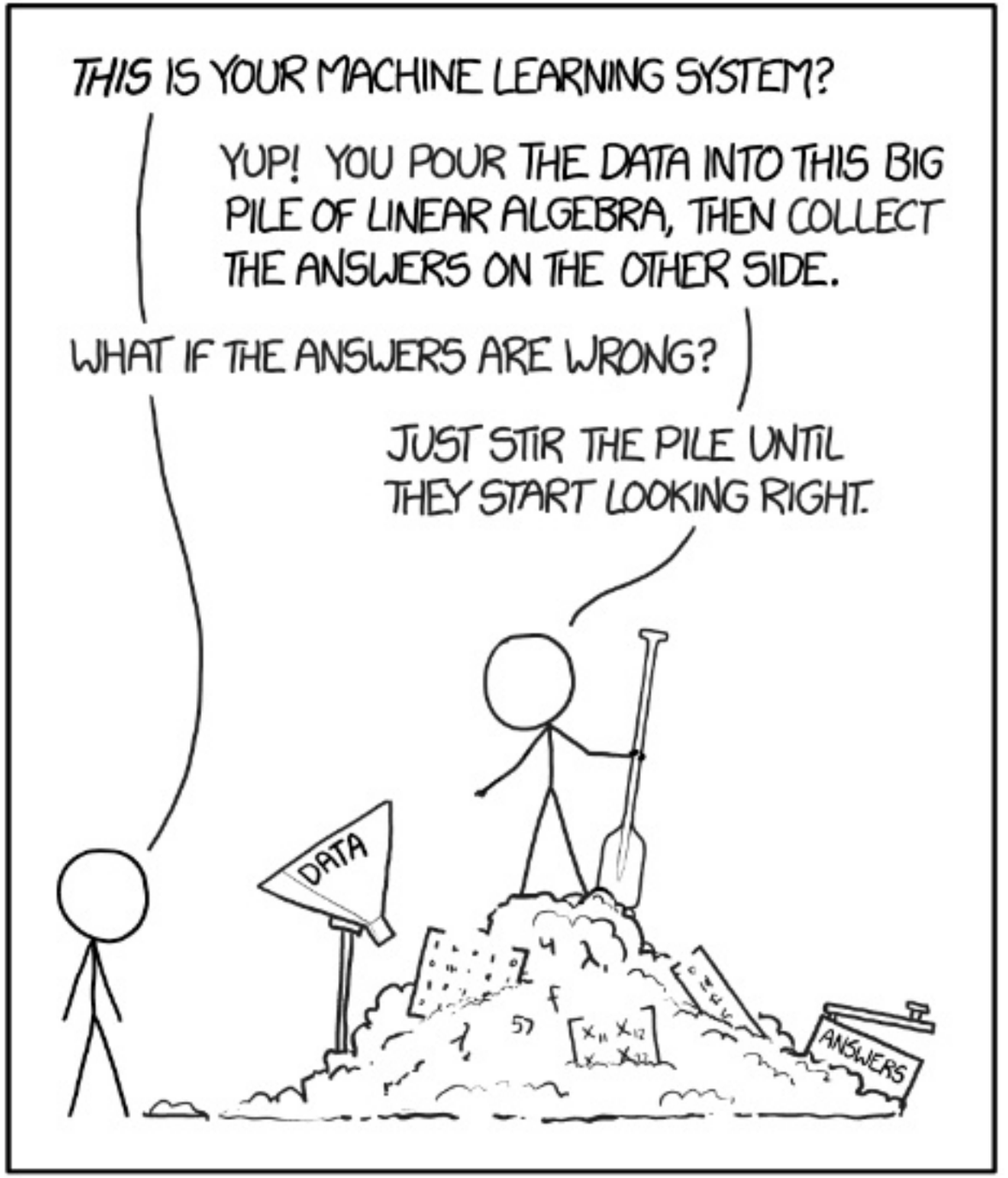
- Who is familiar with Data Quality?
- Who has already implemented Data Quality in Spark?
- Who is expecting to be a ML guru after this session?
- Who is expecting to provide better insights faster after this session?

All Over Again

As goes the old adage:

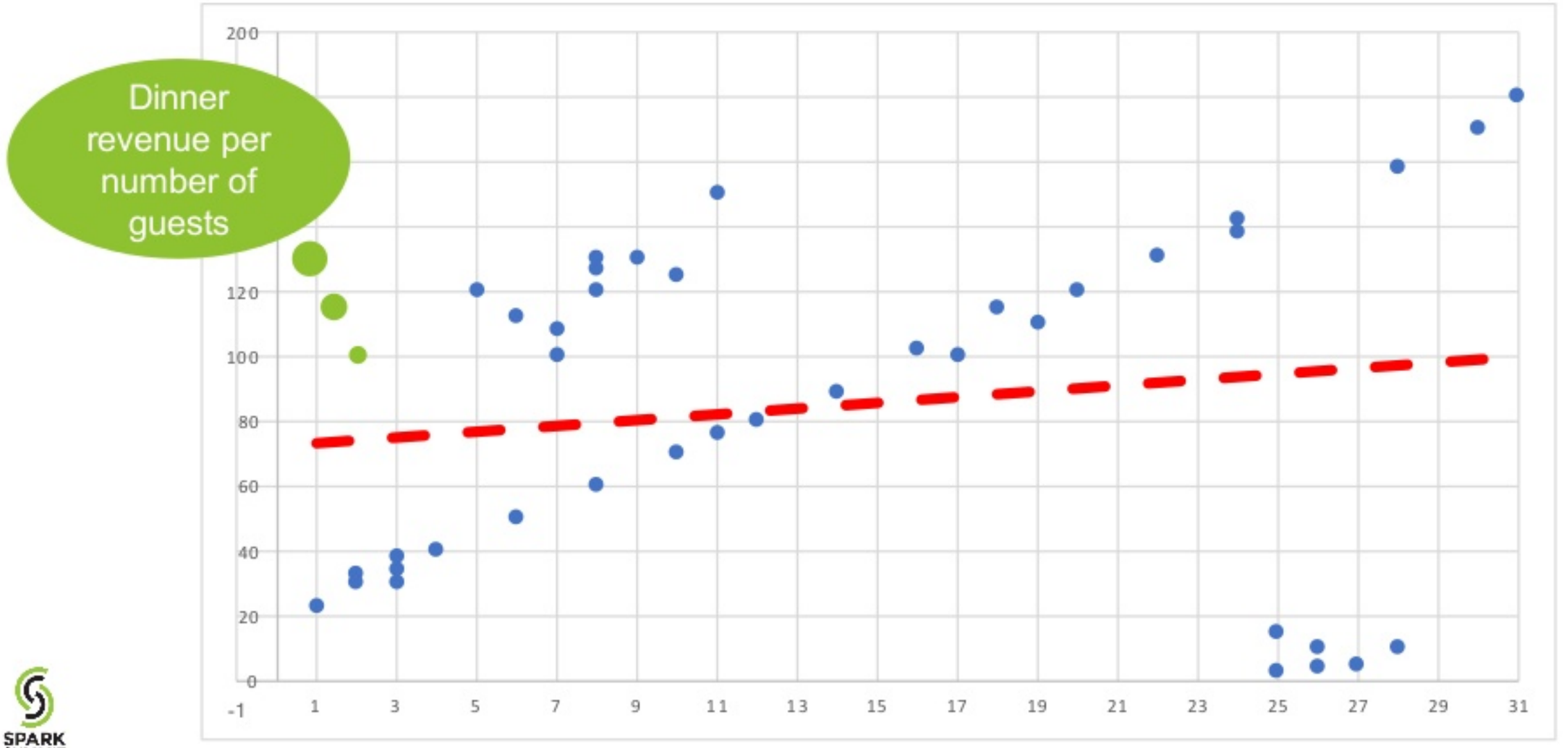
Garbage In,

Garbage Out

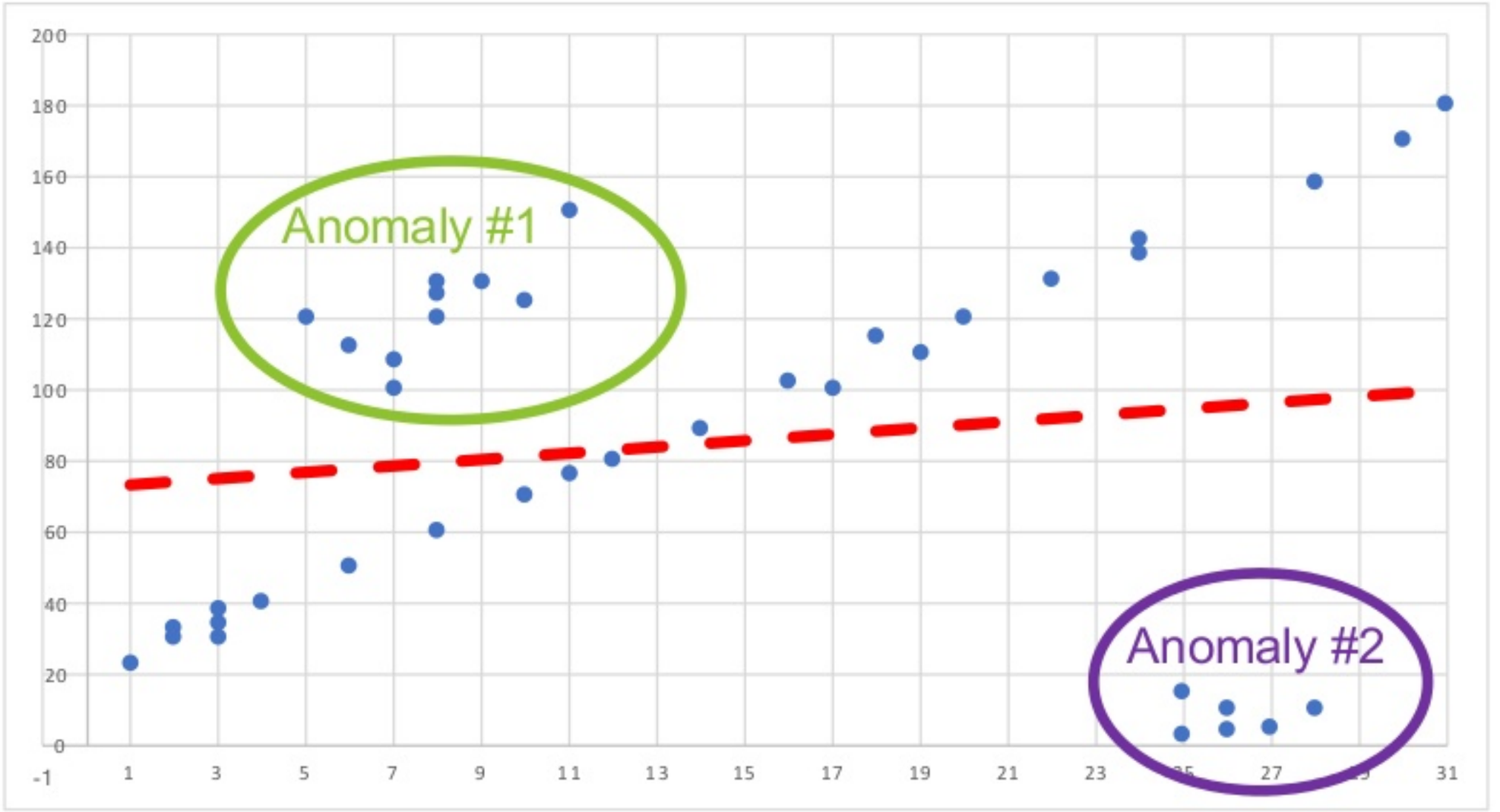


xkcd

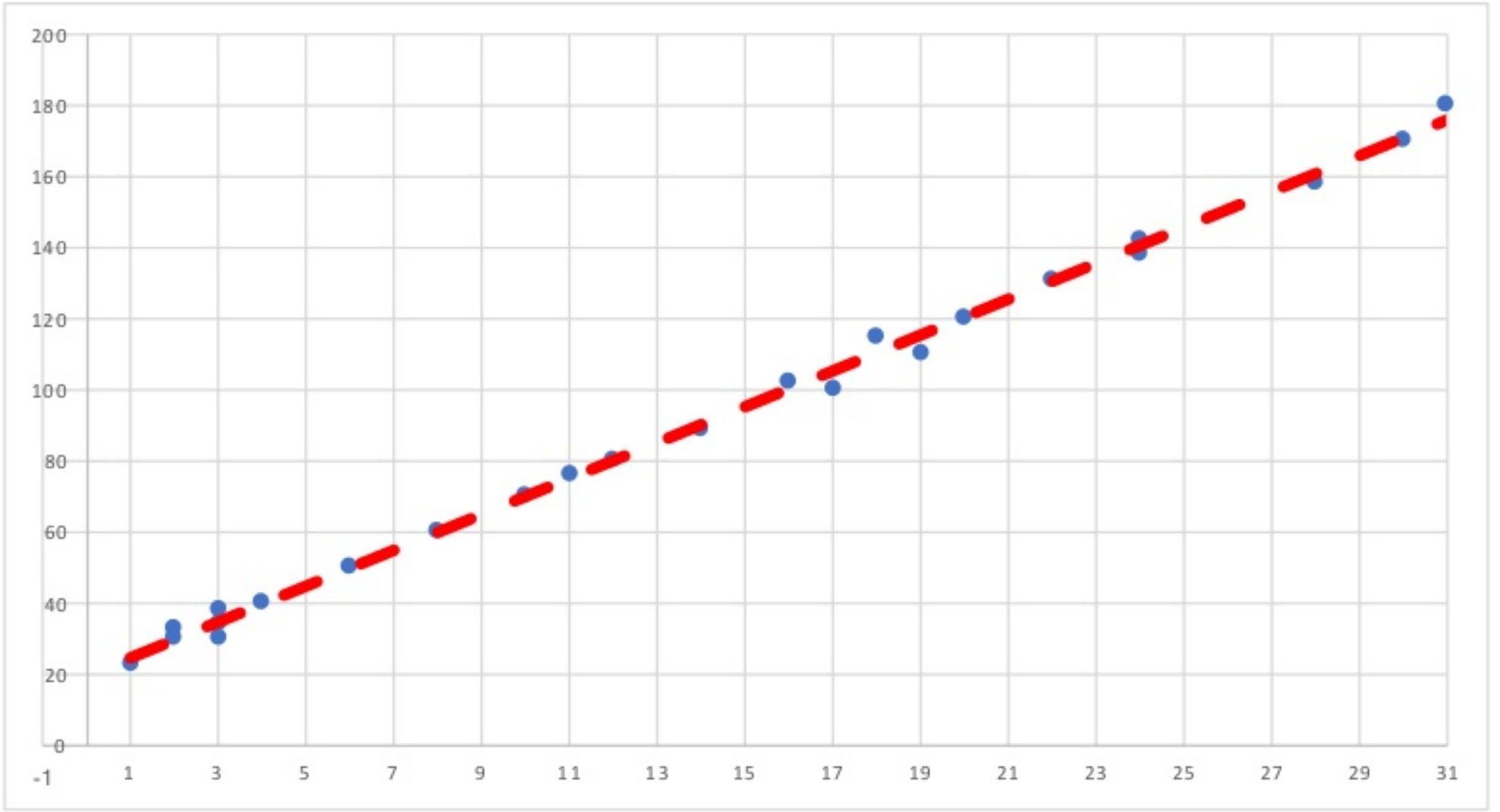
If Everything Was As Simple…



…as a Visual Representation



I Love It When a Plan Comes Together



Data is like a box of chocolates, you never know what you're gonna get

Jean "Gump" Perrin

June 2017

Data From Everywhere

Databases

RDBMS, NoSQL

Files

CSV, XML, Json, Excel, Photos, Video

Machines

Services, REST, Streaming, IoT…

What Is Data Quality?

Skills (CACTAR):

- Consistency,
- Accuracy,
- Completeness,
- Timeliness,
- Accessibility,
- Reliability.

To allow:

- Operations,
- Decision making,
- Planning,
- **Machine Learning**.

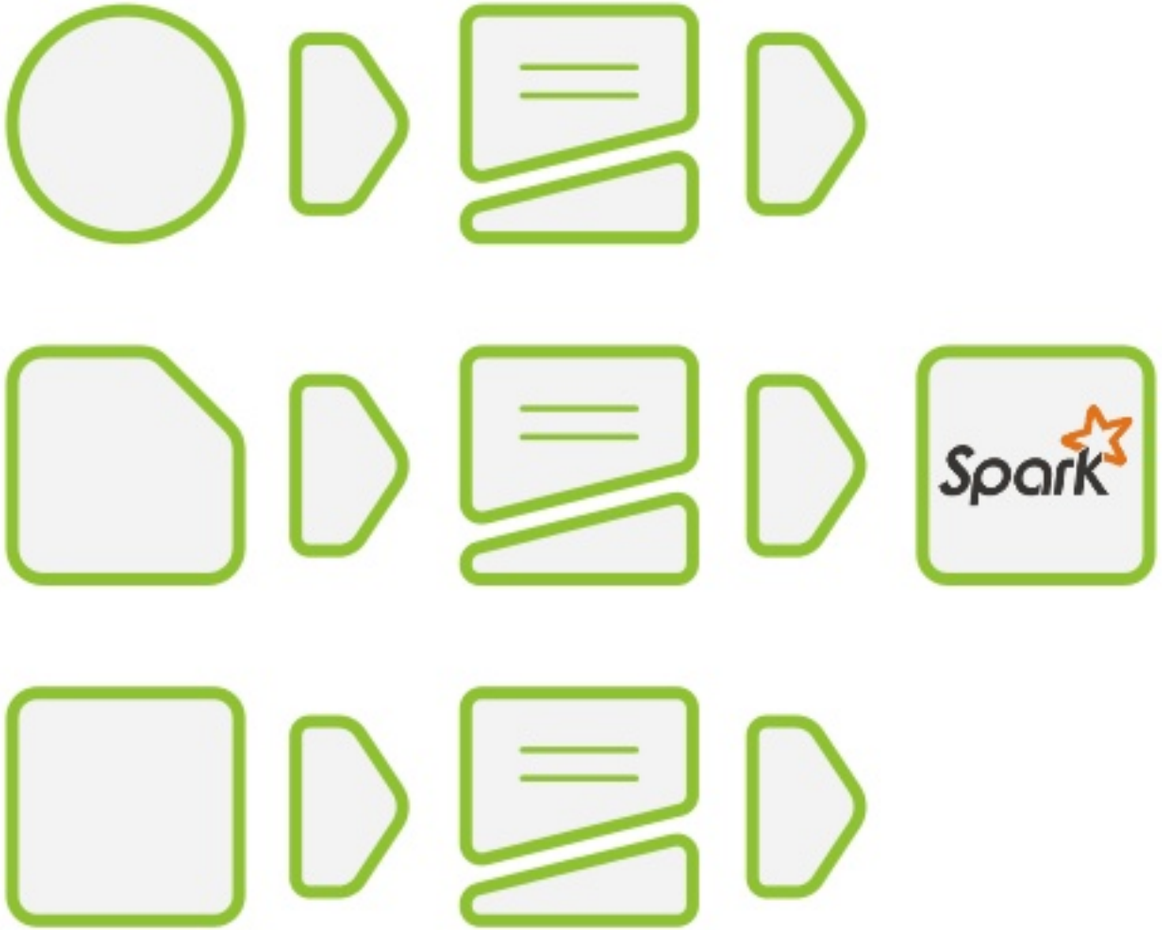
Ouch, Bad Data Story.

- Hubble was blind because of a 2.2nm error
- Legend says it's a metric to imperial/US conversion

Now it Hurts

- Challenger exploded in 1986 because of a defective O-ring
- Root causes were:
  - Invalid data for the O-ring parts
  - Lack of data-lineage & reporting

#1 Scripts and the Likes



- Source data are cleaned by scripts (shell, Java app…)
- I/O intensive
- Storage space intensive
- No parallelization

#2 Use Spark SQL

- All in memory!
- But limited to SQL and built-in function

#3 Use UDFs

- User Defined Function
- SQL can be extended with UDFs
- UDFs benefit from the **cluster architecture** and **distributed processing**
- A tool like **Zaloni's Mica** *(shameless plug)* helps evaluate the result

SPRU Your UDF

1. **Service**: build your code, it might be already existing!
2. **Plumbing**: connect your business logic to Spark via an UDF
3. **Register**: the UDF in Spark
4. **Use**: the UDF is available in Spark SQL and via callUDF()

Code Sample #1.2 - Plumbing

```
package net.jgp.labs.sparkdq4ml.dq.udf;

import org.apache.spark.sql.api.java.UDF1;
import net.jgp.labs.sparkdq4ml.dq.service.*;

public class MinimumPriceDataQualityUdf
  implements UDF1< Double, Double > {

  public Double call(Double price) throws Exception {
    return MinimumPriceDataQualityService.checkMinimumPrice(price);
  }
}
```

/jgperrin/net.jgp.labs.sparkdq4ml

Code Sample #1.3 - Register

```
SparkSession spark = SparkSession
  .builder().appName("DQ4ML").master("local").getOrCreate();

spark.udf().register(
  "minimumPriceRule",
  new MinimumPriceDataQualityUdf(),
  DataTypes.DoubleType);
```

/jgperrin/net.jgp.labs.sparkdq4ml

Code Sample #1.4 - Use

Using CSV, but could be Hive, JDBC, name it…

```
String filename = "data/dataset.csv";
Dataset<Row> df = spark.read().format("csv")
  .option("inferSchema", "true").option("header", "false")
  .load(filename);
df = df.withColumn("guest", df.col("_c0")).drop("_c0");
df = df.withColumn("price", df.col("_c1")).drop("_c1");
df = df.withColumn(
  "price_no_min",
  callUDF("minimumPriceRule", df.col("price")));
df.createOrReplaceTempView("price");
df = spark.sql("SELECT guest, price_no_min AS price FROM price WHERE
price_no_min > 0");
```

/jgperrin/net.jgp.labs.sparkdq4ml

```
+-----+-----+
|guest|price|
+-----+-----+
|    1|23.24|
|    2|30.89|
|    2|33.74|
|    3|34.89|
|    3|29.91|
|    3| 38.0|
|    4| 40.0|
|    5|120.0|
|    6| 50.0|
|    6|112.0|
|    8| 60.0|
|    8|127.0|
|    8|120.0|
|    9|130.0|
+-----+-----+
```

Code Sample #1.4 - Use

```
String filename = "data/dataset.csv";
Dataset<Row> df = spark.read().format("csv")
   .option("inferSchema", "true").option("header", "false")
   .load(filename);
df = df.withColumn("guest", df.col("_c0")).drop("_c0");
df = df.withColumn("price", df.col("_c1")).drop("_c1");
df = df.withColumn(
   "price_no_min",
   callUDF("minimumPriceRule", df.col("price")));
df.createOrReplaceTempView("price");
df = spark.sql("SELECT guest, price_no_min AS price FROM price WHERE
price_no_min > 0");
```

/jgperrin/net.jgp.labs.sparkdq4ml

```
+-----+-----+------------+
|guest|price|price_no_min|
+-----+-----+------------+
|    1| 23.1|        23.1|
|    2| 30.0|        30.0|
|    2| 33.0|        33.0|
|    3| 34.0|        34.0|
|   24|142.0|       142.0|
|   24|138.0|       138.0|
|   25|  3.0|        -1.0|
|   26| 10.0|        -1.0|
|   25| 15.0|        -1.0|
|   26|  4.0|        -1.0|
|   28| 10.0|        -1.0|
|   28|158.0|       158.0|
|   30|170.0|       170.0|
|   31|180.0|       180.0|
+-----+-----+------------+
```

Code Sample #1.4 - Use

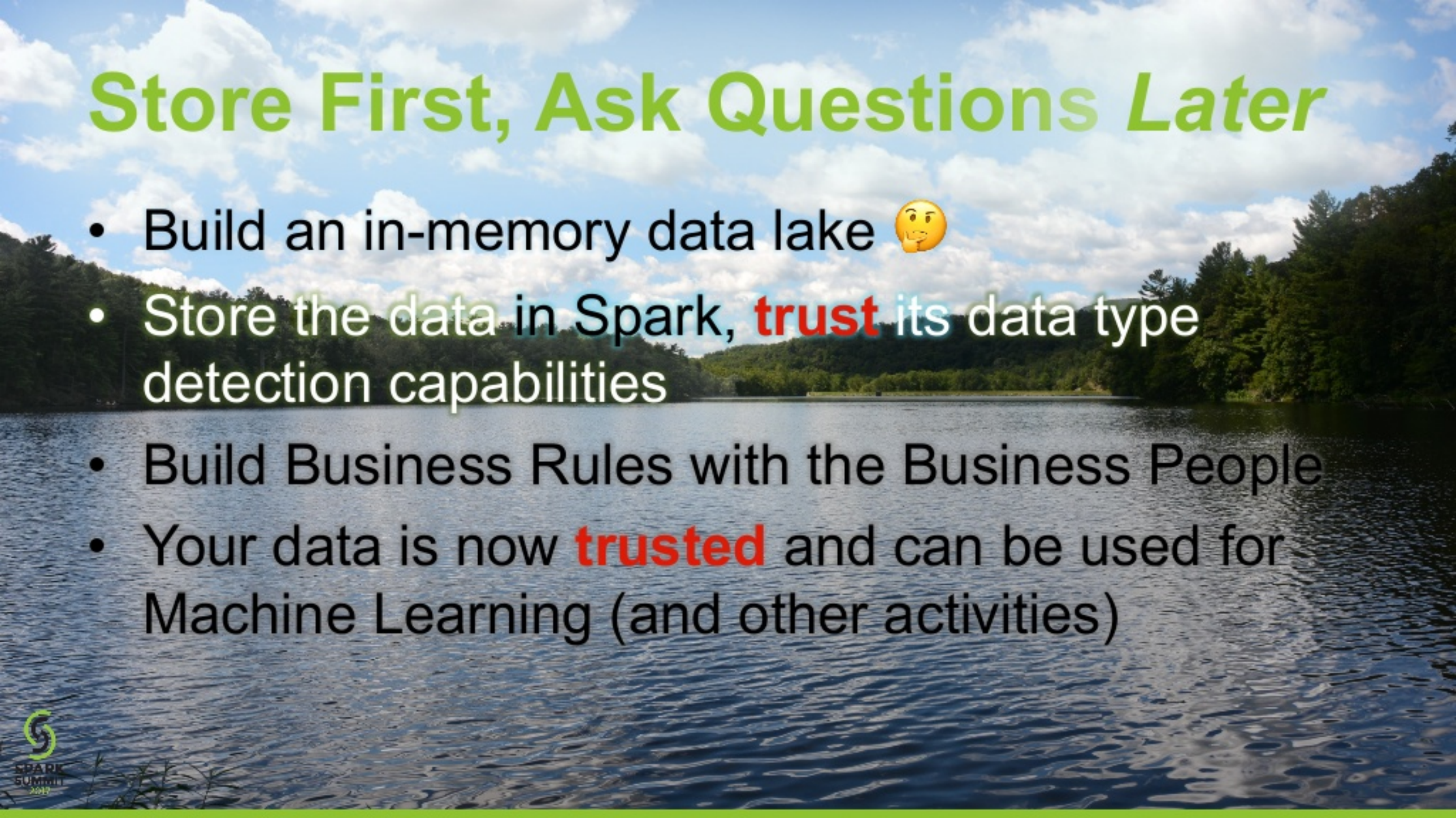
```
String filename = "data/dataset.csv";
Dataset<Row> df = spark.read().format("csv")
   .option("inferSchema", "true").option("header", "false")
   .load(filename);
df = df.withColumn("guest", df.col("_c0")).drop("_c0");
df = df.withColumn("price", df.col("_c1")).drop("_c1");
df = df.withColumn(
   "price_no_min",
   callUDF("minimumPriceRule", df.col("price")));
df.createOrReplaceTempView("price");
df = spark.sql("SELECT guest, price_no_min AS price FROM price WHERE
price_no_min > 0");
```

/jgperrin/net.jgp.labs.sparkdq4ml

```
+-----+-----+
|guest|price|
+-----+-----+
|    1| 23.1|
|    2| 30.0|
|    2| 33.0|
|    3| 34.0|
|    3| 30.0|
|    4| 40.0|
|   19|110.0|
|   20|120.0|
|   22|131.0|
|   24|142.0|
|   24|138.0|
|   28|158.0|
|   30|170.0|
|   31|180.0|
+-----+-----+
```

Store First, Ask Questions *Later*

- Build an in-memory data lake 🤔
- Store the data in Spark, **trust** its data type detection capabilities
- Build Business Rules with the Business People
- Your data is now **trusted** and can be used for Machine Learning (and other activities)



Data Can Now Be Used for ML

- Convert/Adapt dataset to **Features** and **Label**

- Required for **Linear Regression** in MLlib
  - Needs a column called **label** of type double
  - Needs a column called **features** of type VectorUDT

Code Sample #2 - Register & Use

```
spark.udf().register(
  "vectorBuilder",
  new VectorBuilder(),
  new VectorUDT());

df = df.withColumn("label", df.col("price"));
df = df.withColumn("features", callUDF("vectorBuilder", df.col("guest")));

// ... Lots of complex ML code goes here ...

double p = model.predict(features);
System.out.println("Prediction for " + feature + " guests is " + p);
```

/jgperrin/net.jgp.labs.sparkdq4ml

```
+-----+-----+-----+--------+------------------+
|guest|price|label|features|        prediction|
+-----+-----+-----+--------+------------------+
|    1| 23.1| 23.1|   [1.0]|24.563807596513133|
|    2| 30.0| 30.0|   [2.0]|29.595283312577884|
|    2| 33.0| 33.0|   [2.0]|29.595283312577884|
|    3| 34.0| 34.0|   [3.0]| 34.626759028642 64|
|    3| 30.0| 30.0|   [3.0]| 34.626759028642 64|
|    3| 38.0| 38.0|   [3.0]| 34.626759028642 64|
|    4| 40.0| 40.0|   [4.0]| 39.65823474470739|
|   14| 89.0| 89.0|  [14.0]| 89.97299190535493|
|   16|102.0|102.0|  [16.0]|100.03594333748444|
|   20|120.0|120.0|  [20.0]|120.16184620174346|
|   22|131.0|131.0|  [22.0]|130.22479763387295|
|   24|142.0|142.0|  [24.0]|140.28774906600245|
+-----+-----+-----+--------+------------------+

Prediction for 40.0 guests is 220.79136052303852
```

Key Takeaways

- Build your **data lake in memory**.
- **Store first, ask questions later**.
- **Sample your data** (to spot anomalies and more).
- Rely on **Spark SQL and UDFs** to build a consistent & coherent dataset.
- Rely on **UDFs for prepping ML formats**.
- **Use Java**.

Don’t forget
to rate

SPARK
SUMMIT
2017

Thank You.

Jean Georges “JGP” Perrin
jgperrin@zaloni.com
@jgperrin

IBM CHAMPION

Backup Slides & Other Resources

More Reading & Resources

- My blog: http://jgp.net
- Zaloni's blog (BigData, Hadoop, Spark): http://blog.zaloni.com
- This code on GitHub: https://github.com/jgperrin/net.jgp.labs.sparkdq4ml
- Java Code on GitHub: https://github.com/jgperrin/net.jgp.labs.spark

/jgperrin/net.jgp.labs.sparkdq4ml

Data Quality

- **Consistency**: across multiple data sources, servers, platform…
- **Accuracy**: the value must mean something, I was born, in France, on 05/10/1971 but I am a Libra (October).
- **Completeness**: missing fields, values, rows…
- **Timeliness**: how recent and relevant is the data, e.g. 45m Americans change address every year.
- **Accessibility**: how easily can data be accessed an manipulated (Mica…).
- **Reliability**: reliability of sources, of input, production…

ZALONI
THE DATA LAKE COMPANY

Simplifying big data to enable the data-powered enterprise

Industry-leading enterprise data lake management, governance and self-service platform

Expert data lake professional services (Design, Implementation, Workshops, Training)

Solutions to simplify implementation and reduce business risk

ZALONI
THE DATA LAKE COMPANY

Simplifying big data to enable the data-powered enterprise

- Bedrock Data Lake Management and Governance Platform
- Mica Self-service Data and Catalog

Software

- Design & Implementation
- Discovery Workshops
- Training

Services

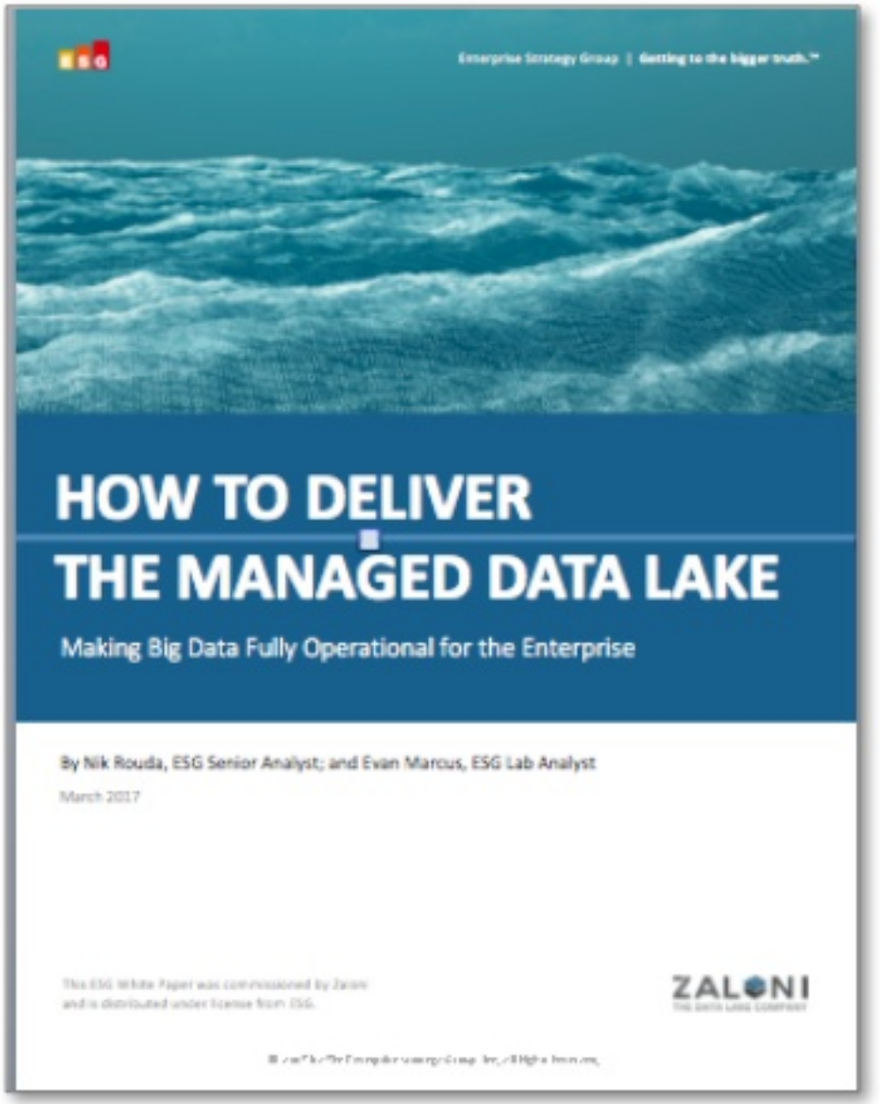
- Data Lake in a Box
- Ingestion Factory
- Custom Solutions

Solutions

New Buyer's Guide on Data Lake Management and Governance

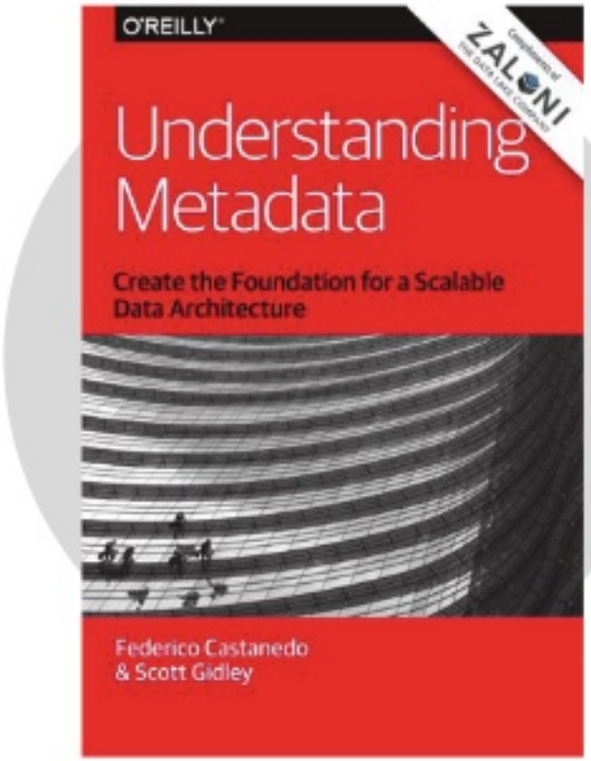
- Zaloni and Industry analyst firm, Enterprise Strategy Group, collaborated on a guide to help you:

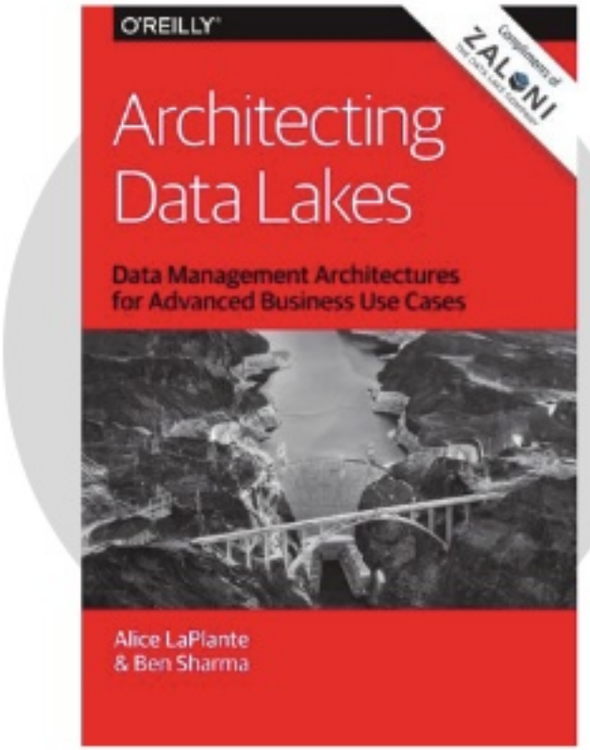
1. Define evaluation criteria and compare common options
2. Set up a successful proof of concept (PoC)
3. Develop an implementation that is future-proofed



Free eBooks to help you future-proof your data lake initiative

Download now at: resources.zaloni.com





ZALONI

THE DATA LAKE COMPANY

The Next Generation

DATA LAKE

Data Lake 360°

BEDROCK

DATA LAKE MANAGEMENT AND GOVERNANCE PLATFORM

MICA

SELF-SERVICE DATA PLATFORM

Photo & Other Credits

- JGP @ Zaloni: Pexels
- Machine Learning Comic, xkcd, Attribution-NonCommercial 2.5 License
- Box of Chocolates: Pexels
- Data From Everywhere: multiple authors, Pexels
- The Hubble Space Telescope in orbit, Public Domain, Nasa
- Challender Explosion, Public Domain, Nasa
- Criticality of data quality as exemplied in two disasters, Craig W. Fishera, Bruce R. Kingmab, https://pdfs.semanticscholar.org/b9b7/447e0e2d0f255a5a9910fc26f2663dc738f4.pdf.
- Store First, Ask Questions Later: Jean Georges Perrin, Attribution-ShareAlike 4.0 International
- Data Quality for Dummies, https://www.linkedin.com/pulse/data-quality-dummies-lee-cullom

Code on GitHub

Fork it and like it @
https://github.com/jgperrin/net.jgp.labs.sparkdq4ml

Abstract

The key to machine learning is prepping the right data

Everyone wants to apply Machine Learning and its data. Sure. It sounds easy. However, when it comes to implementation, you need to make sure that the data matches the format expecting by the libraries.

Machine learning has its challenges and understanding the algorithms is not always easy, but in this session, we'll discuss methods to make these challenges less daunting. After a quick introduction to data quality and a level-set on common vocabulary, we will study the formats that are required by Spark ML to run its algorithms. We will then see how we can automate the build through UDF (User Defined Functions) and other techniques. Automation will bring easy reproducibility, minimize errors, and increase the efficiency of data scientists.

Software engineers who need to understand the requirements and constraints of data scientists.

Data scientists who need to implement/help implement production systems.

Fundamentals about data quality and how to make sure the data is usable – 10%.

Formats required by Spark ML, new vocabulary – 25%.

Build the required tool set in Java – 65%.