

Chapter 2: Data Cleansing

ทำความสะอาดข้อมูลให้พร้อมใช้งานด้วยเครื่องมือ
Distributed Processing: Apache Spark

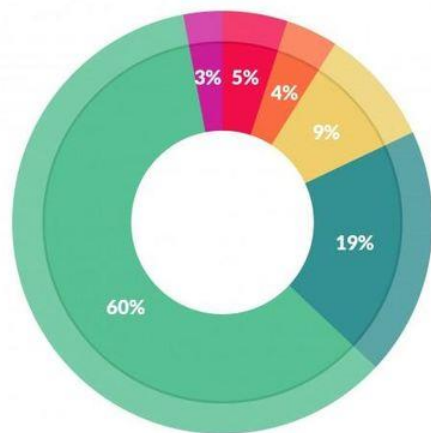




What is Data Cleansing & Data Quality



Data Scientist ใช้เวลาส่วนใหญ่ไปกับ การทำความสะอาดข้อมูล



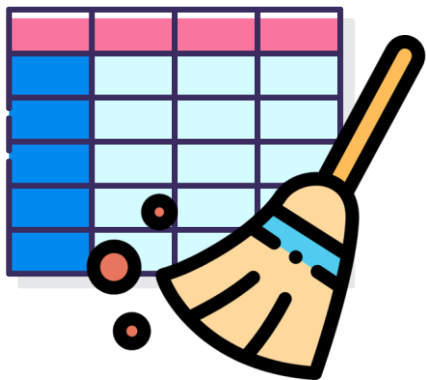
What data scientists spend the most time doing

- Building training sets: 3%
- Cleaning and organizing data: 60%
- Collecting data sets: 19%
- Mining data for patterns: 9%
- Refining algorithms: 4%
- Other: 5%

ผลสำรวจจาก Forbes บอกว่า Data Scientist ใช้เวลากว่า 60% เพื่อเตรียมและทำความสะอาดข้อมูล

<https://www.forbes.com/sites/gilpress/2016/03/23/data-preparation-most-time-consuming-least-enjoyable-data-science-task-survey-says/>

Data Cleansing คืออะไร



Data Cleansing การทำความสะอาดข้อมูล

เป็นการพัฒนาคุณภาพของข้อมูล โดยการค้นหาและแก้ไขความผิดพลาดของข้อมูล เช่น

- ข้อมูลไม่ถูกต้องตามโครงสร้าง (Format Error) เช่น อายุติดลบ
- ข้อมูลสูญหาย (Missing Data)
- ข้อมูลสูงกว่าค่าปกติ (Outlier) เช่น อายุ 670 ปี

Tip: ควรทำ Data Cleansing คู่กับผู้เชี่ยวชาญข้อมูลด้านนั้นด้วย เช่น ผู้เชี่ยวชาญทางการแพทย์ จะบอกเราได้ว่าเลือดมีกี่ประเภท หรือค่าความดันแบบไหนที่ผิดปกติ

Data ทำไม่ถึงผิด

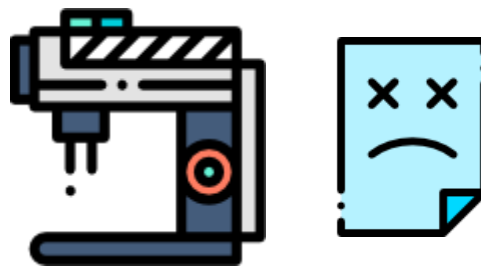
ถ้ายื่น Survey กระดาษ > คนพิมพ์เข้าระบบ

- คนกรอก ก็อาจจะกรอกผิดบ้าง หรือไม่อยากบอกข้อมูลบ้าง
- คนพิมพ์ใส่ระบบ ก็อาจจะพิมพ์ผิด



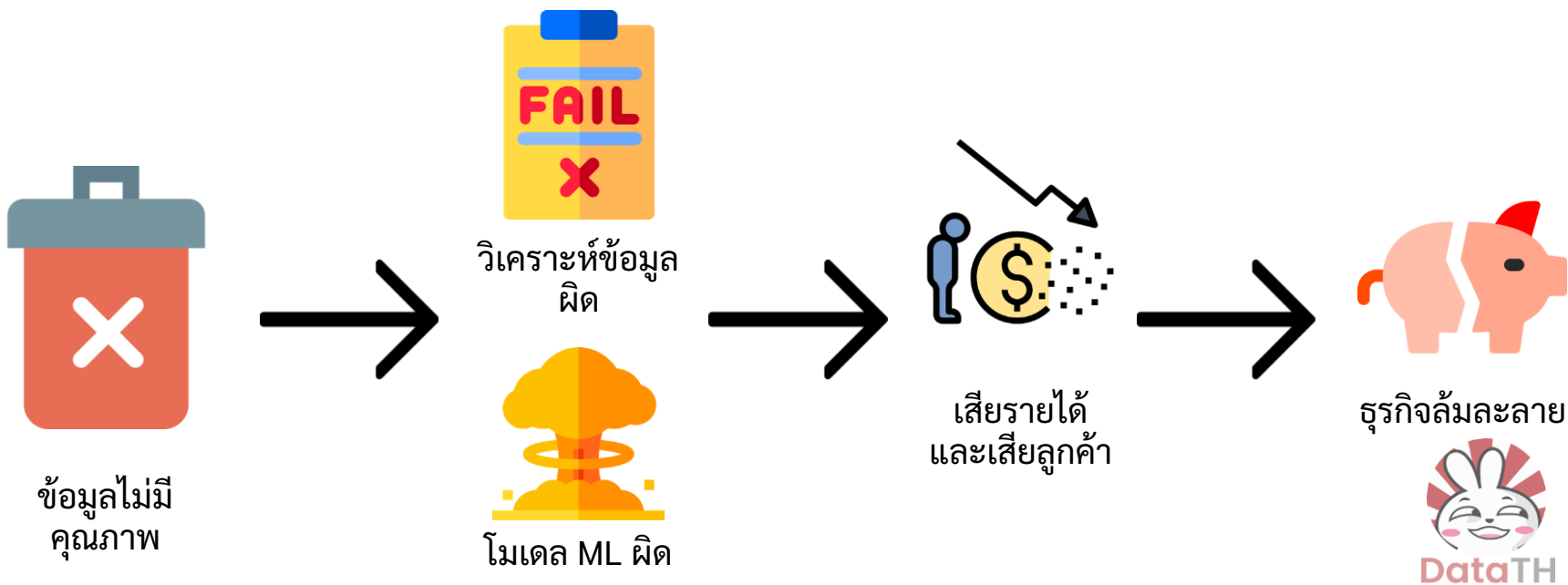
ถ้ามาจากเซนเซอร์เครื่องจักร > เข้าระบบอัตโนมัติ

- เซนเซอร์อาจจะมีปัญหาบ้าง หรือ วัดผลผิดบ้าง
- เกิดข้อผิดพลาดระหว่างการส่งข้อมูล



ทำไมต้องทำความสะอาดข้อมูล

Garbage in, Garbage out



Data ที่ไม่สะอาด มีราคาแพง

เศรษฐกิจอเมริกาเสียเงิน **\$3.1 พันล้าน** จากข้อมูลที่ไม่สะอาด ในปี 2016



Case Study: NASA เสียเงิน \$125 ล้าน จากข้อมูลไม่สะอาดในโปรเจก Mars Climate Orbiter

“The peer review preliminary findings indicate that **one team used English units (e.g., inches, feet and pounds)** while the **other used metric units (e.g. centimeter, kilometer, and kilogram)** for a key spacecraft operation.”

<https://www.ibmbigdatahub.com/infographic/extracting-business-value-4-vs-big-data>

<https://mars.jpl.nasa.gov/msp98/news/mco990930.html>



ทำไม Data Cleaning ถึงยาก



กระบวนการที่
ไม่มีวันจบสิ้น

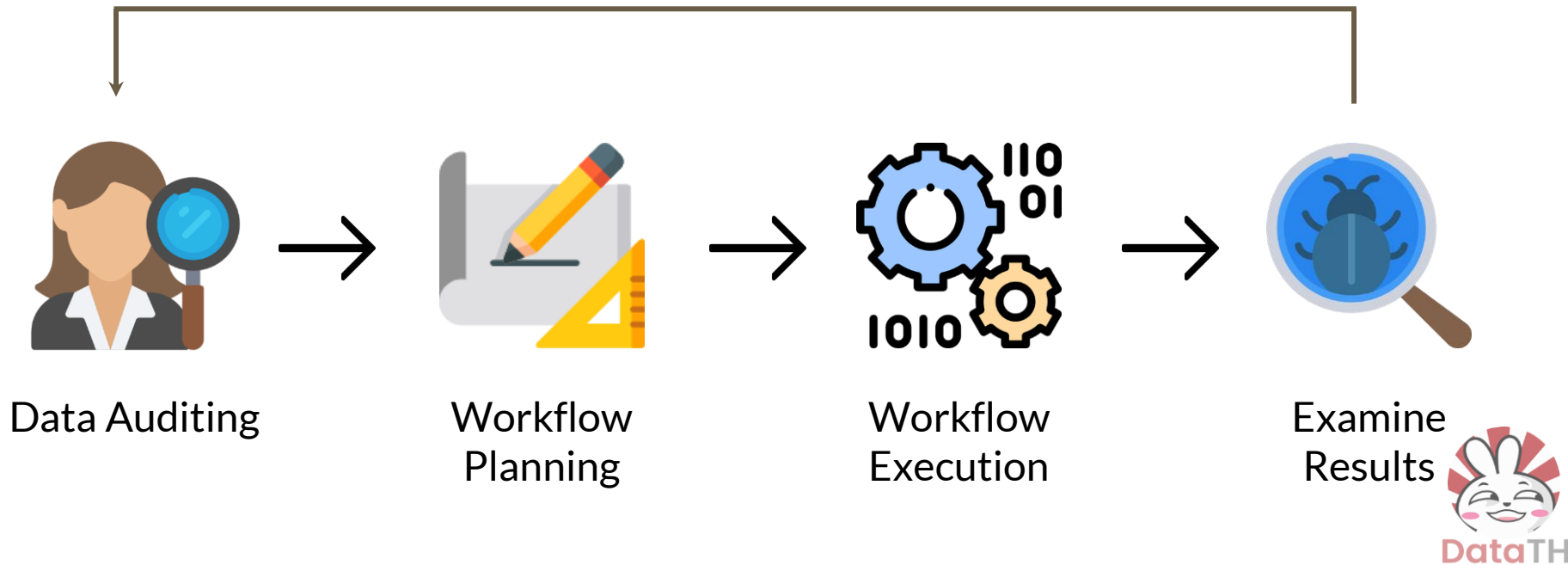


ยากที่จะรู้ว่าเกิด
จากอะไร

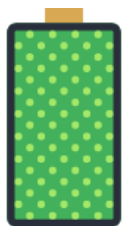


แต่ละ Source มักมี
ข้อมูลโครงสร้าง
แตกต่างกัน

Data cleansing เป็นกระบวนการที่ไม่มีวันจบสิ้น



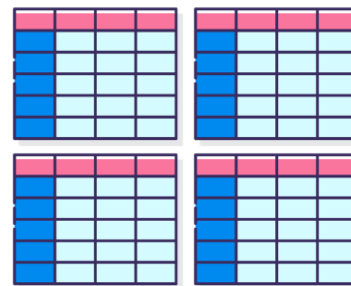
Data Quality - คุณภาพข้อมูลที่ดี คืออะไร



Completeness
ข้อมูลครบ ไม่มี
Missing Values



Validity
ข้อมูลไม่ผิดข้อจำกัด
เช่น อายุไม่ควรติดลบ

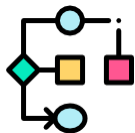


Consistency
ข้อมูลจากหลาย Data Source
ควรจะใช้โครงสร้างคล้ายกัน

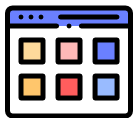
3 เครื่องมือสำหรับ Data Quality



Data Dictionary



Data Lineage



Data Catalogue

Data Dictionary



ไฟล์ที่รวบรวมรายละเอียดของทุกคอลัมน์
ในตารางข้อมูล เพื่อให้ทุกคนในองค์กร
เข้าใจตรงกัน และสำคัญมากในการทำควา
สะอาดข้อมูล

คอลัมน์สำคัญใน Data Dictionary:

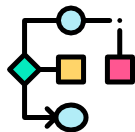
- ชื่อคอลัมน์
- ประเภทข้อมูลในคอลัมน์
- ตัวอย่างข้อมูลในคอลัมน์
- คำอธิบายของคอลัมน์

employees

Column	Data type	Nullable	Description
emp_no	integer	not null	Unique employee number
first_name	varchar(100)	not null	Employee first name
last_name	varchar(100)	not null	Employee last name
dob	date	null	Date of birth if known
card_no	char(6)	null	Employee access control card number
edu	integer	not null	Education level 9 - unknown 4 - elementary 3 - Middle school 2 - High school 1 - University
dept_id	integer	not null	Employee department ID. Ref: departments
eval	date	null	Date of last performance review
current	bit	not null	1 - current employee, 0 - past employee



Data Lineage

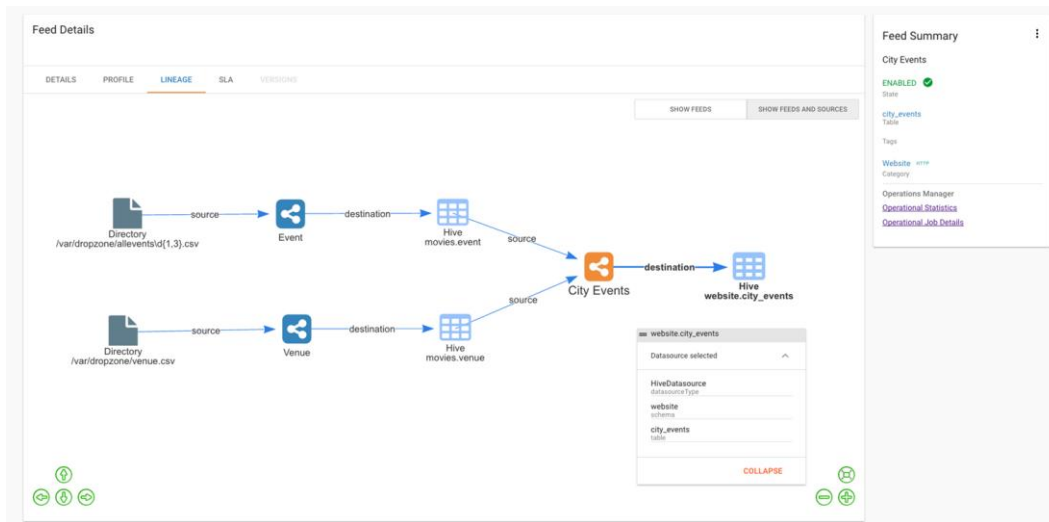


การเดินทางของข้อมูลตั้งแต่ต้นจนจบ

มีประโยชน์มากในการตรวจสอบ
Error ของข้อมูล ว่ามาจากจุดไหนใน
ระบบ



Apache Atlas



Customer Orders.csv

Description

There is no description available for this asset.

Tags

subPersonalInfo

Added:

Mar 7, 2018 2:25 PM

Format:

CSV

Connection

Source:

BM_COS_EDITOR_Transform C...

Source type:

Cloud Object Storage

Classification

None

Schema: 15 Columns 5000+ Rows

4 Columns anonymized

Preview (100 rows)

	CITY	FIRST_NAME	LAST_NAME	GENDER	AGE
	Type: String	Type: String	Type: String	Type: String	Type: String
61f1ff4b	Plymouth	5c480c05ed67	Column anonymized		
61f1ff4b	Leipzig	6507c45a9cef	The values and format of data in this column are substituted by this policy: Substitute Personal Information only		
61f1ff4b	Leipzig	1504975c21606	Learn more		
15a3aa	Manaus	e476c7288948d2	1e893c09c9e6	60783e873d19a	56
1d2b0a	College Station	773b30943d396e	3161b2a939f6b	32a56f2a19f7dC	45
1d2b0a	College Station	86426a39ac6163	bdb6ae87eb0e0a	32a56f2a19f7dC	45
61f1ff4b	Jaipur	7ef7fae2c73fc8a5	21c4def4075f47	60783e873d19a	39
61f1ff4b	Musashino	e39a94d25059e6	0e604501a660c	60783e873d19a	49
61f1ff4b	Musashino	f4713ec721f59516	80e46a43ce079	60783e873d19a	49



Exploratory Data Analysis (EDA)

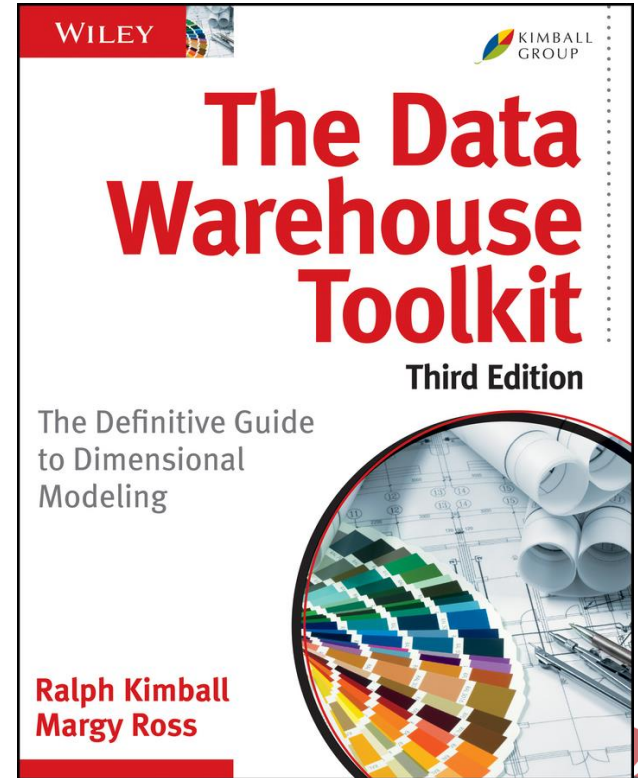


Data Profiling

“Data Profiling ใช้การดึงข้อมูลจากแหล่งข้อมูล เพื่อค้นหาและความสัมพันธ์ แทนที่จะดูจากข้อมูลที่ไม่อาจไม่อัปเดตจากเอกสารการใช้งาน”

- Ralph Kimball, Margy Ross

วิธีการดูภาพรวมของข้อมูล เพื่อหาว่ามีปัญหาอะไรกับข้อมูล



ตัวอย่างของการทำ Data Profiling

ใช้ข้อมูล Hotel Booking
Demand บน Kaggle

<https://www.kaggle.com/jessemostipak/hotel-booking-demand>

	is_canceled	lead_time	arrival_date_year	arrival_date_week_number	arrival_date_day_of_month
count	119390.000000	119390.000000	119390.000000	119390.000000	119390.000000
mean	0.370416	104.011416	2016.156554	27.165173	15.798241
std	0.482918	106.863097	0.707476	13.605138	8.780829
min	0.000000	0.000000	2015.000000	1.000000	1.000000
25%	0.000000	18.000000	2016.000000	16.000000	8.000000
50%	0.000000	69.000000	2016.000000	28.000000	16.000000
75%	1.000000	160.000000	2017.000000	38.000000	23.000000
max	1.000000	737.000000	2017.000000	53.000000	31.000000



ประเภทของ EDA

(Exploratory Data Analysis)

วิธีการเจาะลึกเพื่อดูรายละเอียดของข้อมูล และความสัมพันธ์ของแต่ละคอลัมน์ / แถว

ใช้ตัวเลขหรือกราฟฟิก

1. **แบบใช้ตัวเลข** - ใช้ตัวเลขทางสถิติ เช่น หาค่า Min, Max, Mean
2. **แบบใช้กราฟฟิก** - ใช้การพลอตกราฟ (Data Visualisation) เช่น Boxplot, Histogram

ดูรายละเอียดตัวแปรที่ตัว

1. **ดูทีละตัว (Univariate)** - เช่น ดูค่า Mean ของคอลัมน์เดียว
2. **ดูทีละหลายตัว (Multivariate)** - เช่น การคำนวณ Covariance หรือ การวาด Scatterplot เปรียบเทียบระหว่าง 2 ตัวแปร



EDA แบบใช้ตัวเลข

(Exploratory Data Analysis)

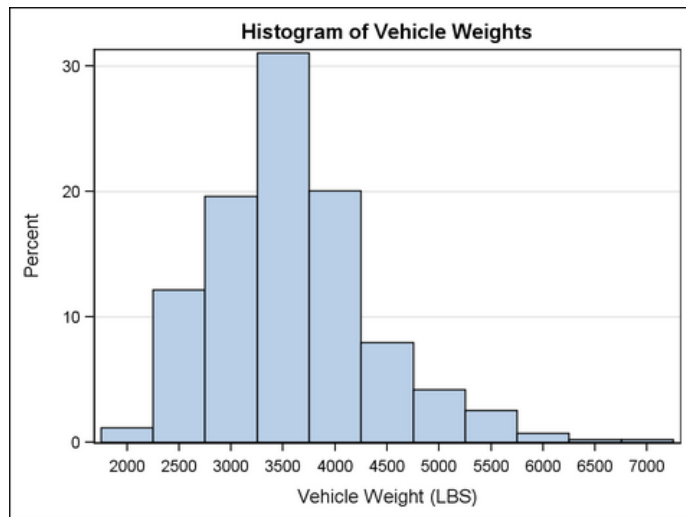
สถิติเชิงพรรณนา (Descriptive Statistics)

- การกระจายตัวของข้อมูล (Distribution):
 - จำนวนข้อมูล (count)
 - ค่าเฉลี่ย (mean)
 - ข้อมูล 25%, 50%, 75%
 - ค่าสูงสุด - ต่ำสุด
- ความแปรปรวนของข้อมูล (Variance):
 - ค่าความเบี่ยงเบนมาตรฐาน (Standard Deviation)

	is_canceled	lead_time
count	119390.000000	119390.000000
mean	0.370416	104.011416
std	0.482918	106.863097
min	0.000000	0.000000
25%	0.000000	18.000000
50%	0.000000	69.000000
75%	1.000000	160.000000
max	1.000000	737.000000



EDA แบบใช้กราฟฟีก

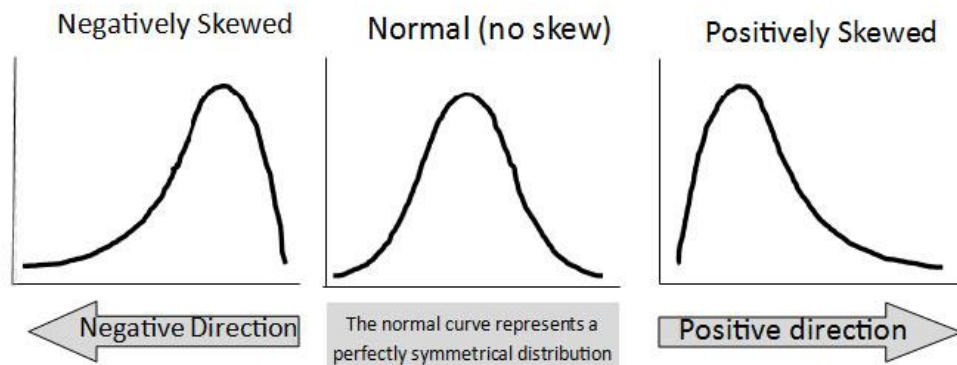


1 LBS = 0.45 kg

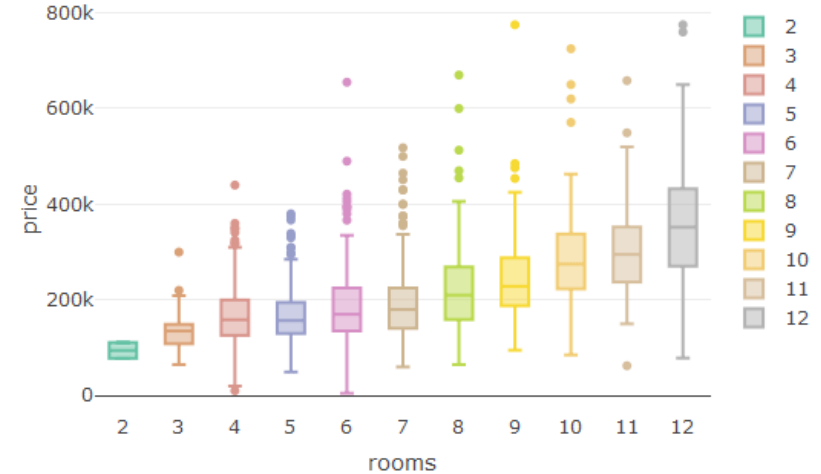
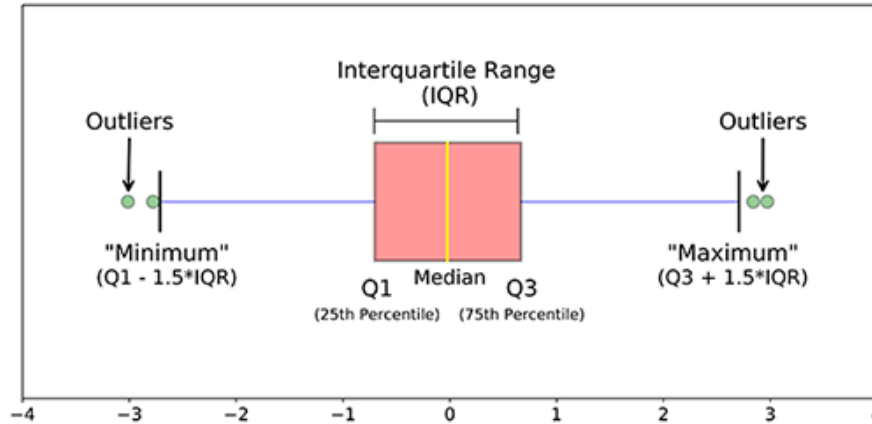
Histogram

แสดงการกระจายตัวของข้อมูล (เป็นลักษณะยอดเขาเดียว หรือหลายยอด หรือไม่มียอดเลย) และความเบี่ยงเบนของข้อมูล (Skewness)

Skewness - ความเบี่ยงเบนของข้อมูล



EDA แบบใช้กราฟฟีก (2)



Boxplot

แสดงการกระจายตัวของข้อมูล และค่าที่เกินจากปกติมาก (Outliers)



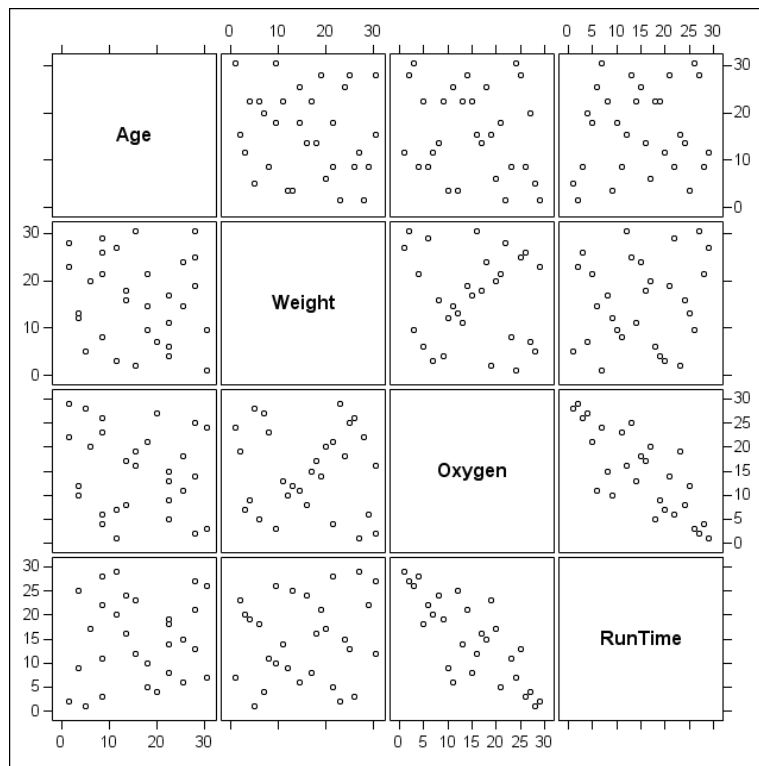
EDA แบบใช้กราฟพีค (3)

Scatterplot

ใช้หาความสัมพันธ์ระหว่าง 2 ตัวแปร

Scatterplot Matrix

ใช้หาความสัมพันธ์ระหว่างทุกคู่ตัวแปร ->





Data Anomaly

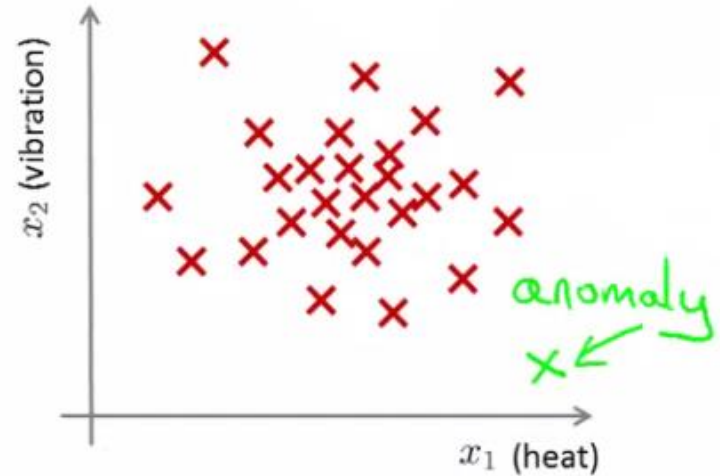


What is Data Anomaly

ความผิดปกติของข้อมูลที่เกิดขึ้นได้จากตอนเก็บข้อมูล ซึ่งทำให้ข้อมูลไม่สมบูรณ์

ตัวอย่าง Data Anomaly:

- พิมพ์ผิด (Lexical Error)
- ข้อมูลซ้ำ (Duplication)
- ข้อมูลไม่สม่ำเสมอ (Inconsistency)
- ข้อมูลหายบางส่วน (Missing Values)
- ข้อมูลเกินค่าปกติ (Outliers)
- ฯลฯ



ประเภทของ Data Anomaly

1. **Syntactical Anomalies** - เกิดจากข้อผิดพลาดในการกรอกข้อมูล เช่น พิมพ์ผิด (spelling mistake), ประเภทข้อมูลผิด (domain format error), มีตัวอักษรผิดปกติ (syntactical error), มีค่าแปลก (irregularity)
2. **Semantic Anomalies**: เกิดจากข้อผิดพลาดในการเก็บข้อมูล เช่น เก็บข้อมูลซ้ำ (duplication), เก็บข้อมูลไม่ตรงตามเงื่อนไข (integrity constraint violation)
3. **Coverage Anomalies**: เกิดจากข้อผิดพลาดในความสมบูรณ์ของข้อมูล เช่น ค่าหาย (missing value)
4. **Outliers**: ข้อมูลที่สูงหรือต่ำกว่าปกติ

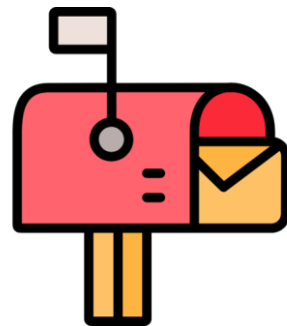


1. Syntactical Anomalies คืออะไร และวิธีแก้ไข

Syntactical Anomalies เช่น พิมพ์ผิด (Spelling Mistake) หรือคำแปลก (Irregularities) เช่น รหัสไปรษณีย์ 6 หลัก, ชื่อจังหวัดที่ไม่มีจริง

วิธีแก้ไข:

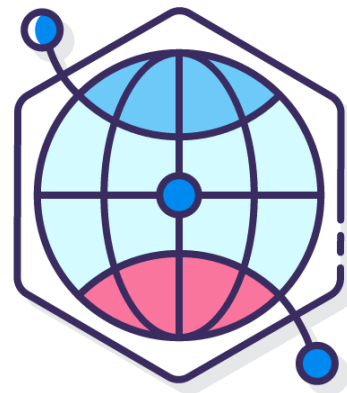
1. แก้ไขที่การเก็บข้อมูล: เช่น ถ้าเป็นฟอร์มออนไลน์ สามารถเขียนโค้ดตรวจสอบข้อมูล (Data Validation) หรือทำเป็นช้อยส์ให้เลือก
2. แก้หลังจากเก็บข้อมูลมาแล้ว: เปรียบเทียบกับแหล่งอ้างอิงข้อมูลที่น่าเชื่อถือ เช่น ตรวจสอบที่อยู่จาก Google Map API



2. Semantic Anomalies คืออะไร และวิธีแก้ไข

ประเภทของ Semantic Anomalies

1. ค่าไม่ตรงตามเงื่อนไข (Integrity Constraints)
เช่น อายุติดลบ หาค่าที่ผิด และแก้ไขให้ถูก
2. ค่าซ้ำ (Duplication)
เช็คก่อนว่าคอลัมน์นั้นซ้ำได้มั้ย แล้วลบหรือแก้ไขตามความเหมาะสม
3. ค่าขัดกัน (Contradictions)
เช่น วันจบมาก่อนวันเริ่ม แก้ไขเหมือนค่าซ้ำ



วิธีค้นหา Syntactical และ Semantics Anomalies

Regular Expression (regex) = pattern สำหรับค้นหาตัวหนังสือ

ทำไม regex ถึงมีประโยชน์

- ฟังก์ชันเยอะ และยืดหยุ่น
- เขียนได้สั้น ๆ
- ง่ายสำหรับการค้นหาและแทนที่
- เรียนครั้งเดียว นำไปใช้ได้หลายภาษาโปรแกรมมิ่ง

ตัวอย่าง

[A-Z] = ค้นหา 1 ตัวอักษร จาก A ถึง Z

[A-Z]+ = ค้นหา 1+ ตัวอักษร จาก A ถึง Z

[a-zA-Z0-9]* = ค้นหา 0+ ตัวอักษร จาก a ถึง z, A ถึง Z, และ 0 ถึง 9

Tip: ใช้ <https://regex101.com/> สำหรับเขียน Regular Expression



3. Coverage Anomalies หรือ Missing Values

ข้อมูลที่หายไปจากระบบ สามารถเกิดโดยบังเอิญ (MAR - Missing At Random)

หรือไม่บังเอิญ (MNAR - Missing Not At Random)

สำหรับ Data Analyst,
Data Scientist



- มองหาข้อมูลหาย
- ตรวจสอบว่าข้อมูลหายเพราะอะไร
- หาวิธีแก้ไขปัญหาคือข้อมูลหายที่เหมาะสมที่สุด เช่น ตัดทิ้งจากการวิเคราะห์ หรือ แทนค่าที่หาย (impute) ด้วยค่าเฉลี่ย, Regression Model

สำหรับ Data Engineer



- มองหาข้อมูลหาย
- ตรวจสอบข้อมูลดิบ (Raw Data) ที่มาจากแหล่งข้อมูล
- แก้ไขปัญหากับทีมที่ดูแลแหล่งข้อมูล

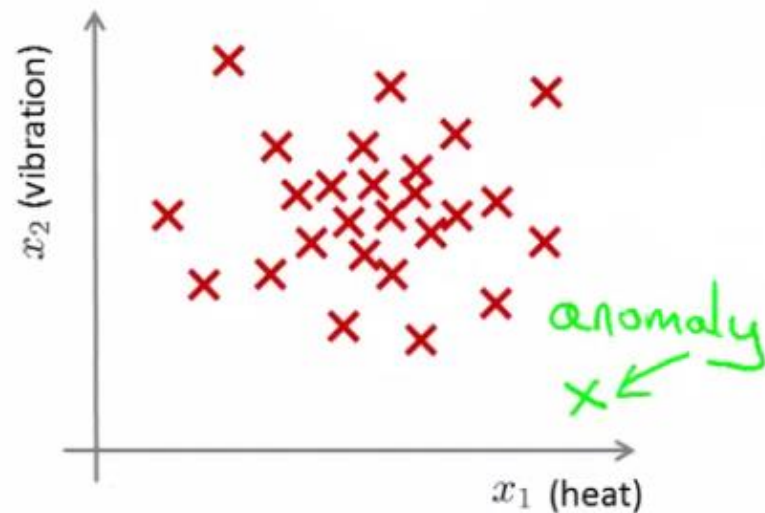
Tip: ปัญหาถ้าแก้ได้ที่ต้นเหตุ (แหล่งข้อมูล) จะดีกับงานมากกว่าในระยะยาว



Outliers คืออะไร

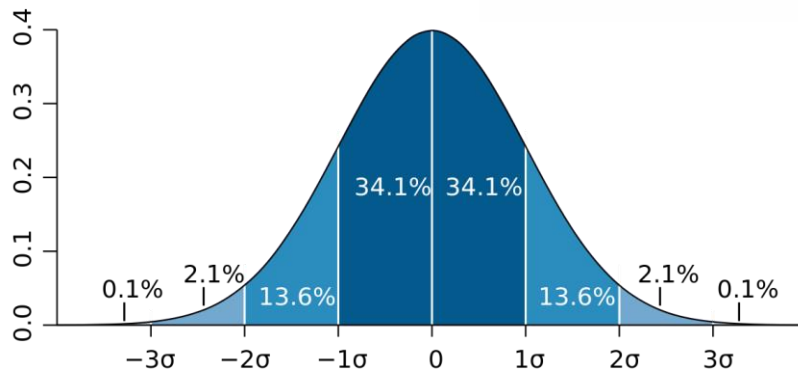
Outliers คือ ค่าที่ห่างจากค่าส่วนใหญ่มาก ซึ่งแปลว่าค่าอาจจะผิดปกติ

มีประโยชน์มากสำหรับการตรวจสอบการโกง ในบัตรเครดิต (Fraud Detection)



วิธีการเช็ค Outliers

1. Boxplot
2. กฎ 3 sigma





Distributed Data Processing

(การประมวลผลข้อมูล แบบกระจายศูนย์)



Distributed Data Processing คืออะไร

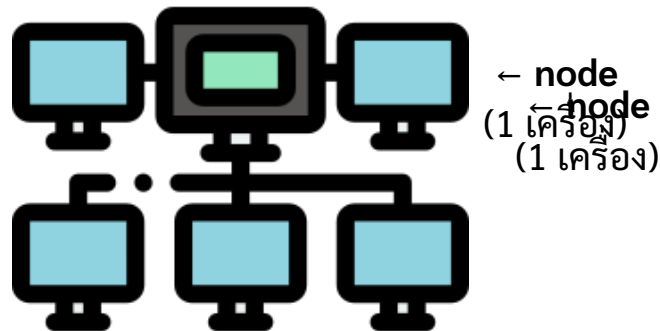
Standalone (1 Node)



ประมวลผลด้วยคอมพิวเตอร์ 1 เครื่อง

ทำงานได้แค่ทีละอย่าง

Cluster (2+ Nodes)



ประมวลผลด้วยคอมพิวเตอร์หลายเครื่อง

ทำได้พร้อมกันหลายอย่าง หรือทำอย่างเดียวกันก็เร็วขึ้น เพราะกระจายงานกันทำได้

ตัวอย่าง Distributed Data Processing : Word Count การนับจำนวนของแต่ละคำ จากหนังสือ 1 เล่ม

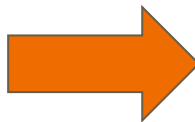
Standalone (1 คน)

Apple: 10 คำ
Bee: 5 คำ
Cat: 9 คำ
Dog: 13 คำ
...

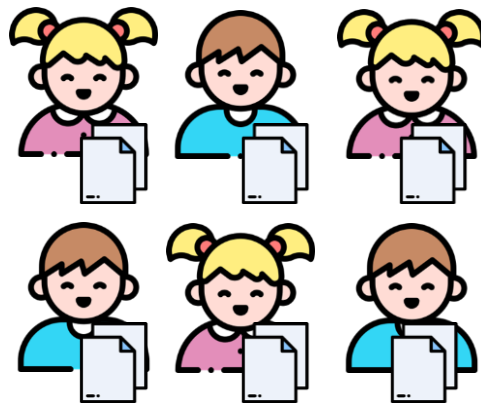


ใช้ผู้เชี่ยวชาญ 1 คน นับทั้งเล่ม

อ่านได้แค่ทีละหน้า



Cluster (2+ คน)



← node
(1 คน)

ใช้เด็กหลายคน นับคนละบท

อ่านได้พร้อมกันหลายหน้า

Hadoop MapReduce



ส่วนหนึ่งของ Hadoop: Hadoop MapReduce เป็นวิธีหนึ่งในการทำ Distributed Data Processing สำหรับข้อมูลขนาดใหญ่ ที่ได้รับความนิยมมากในช่วงก่อนหน้านี้
สร้างขึ้นโดย Yahoo! ในปี 2006 จากงานวิจัยของ Google ที่ตีพิมพ์ในปี 2003



ใช้คอมพิวเตอร์ทั่วไป
(Commodity
Hardware) มาทำงาน
ร่วมกัน

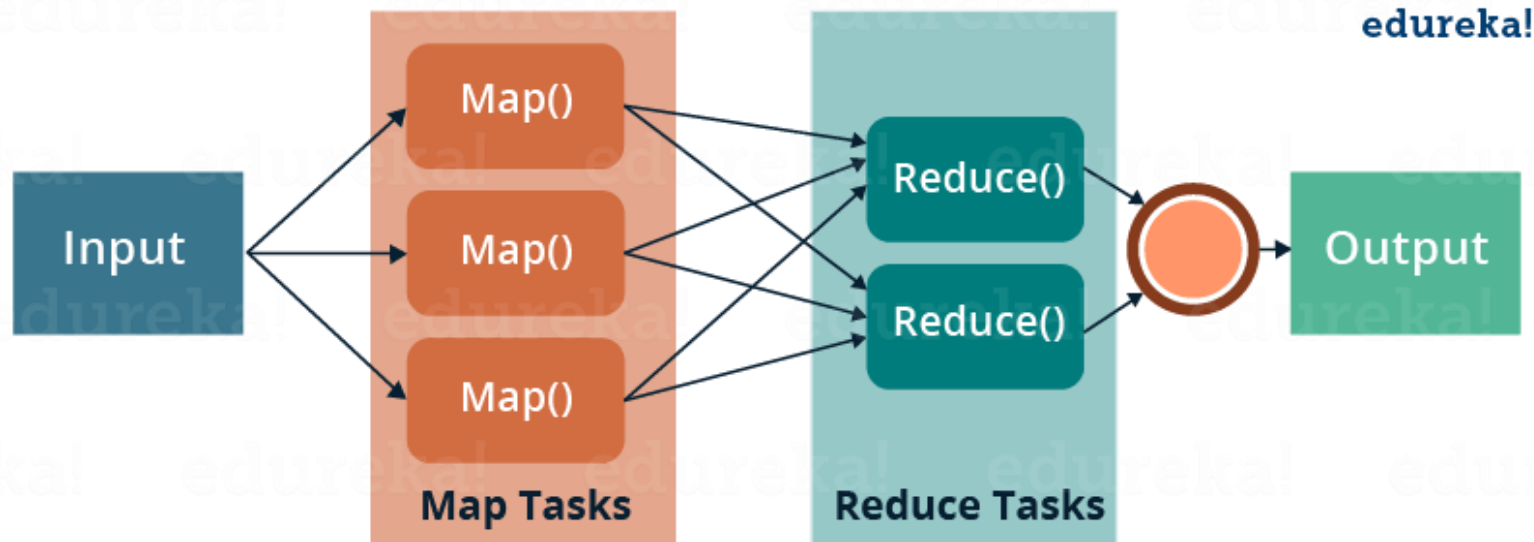


เขียนยาก ทำให้ความนิยม
ตกลงไปในปัจจุบัน



ทำงานช้า เพราะต้องเขียน
ไฟล์เก็บข้อมูลทุกขั้นตอน

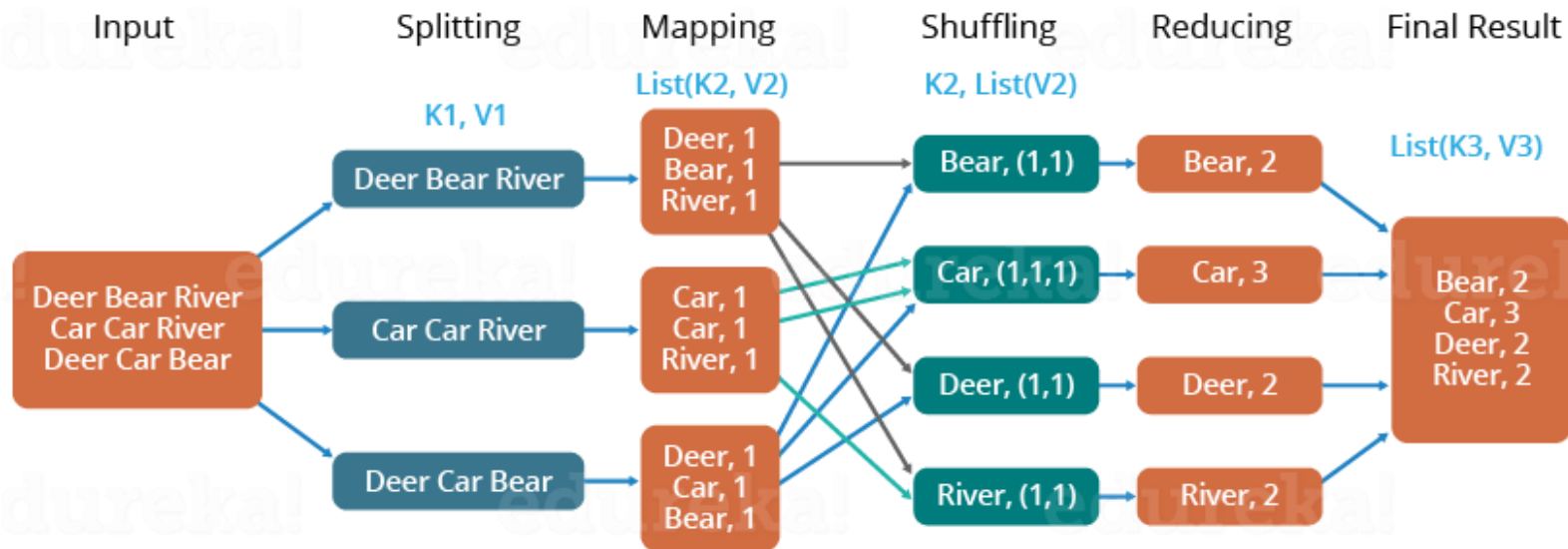
การทำงานของ Hadoop MapReduce



ตัวอย่าง Hadoop MapReduce

The Overall MapReduce Word Count Process

edureka!





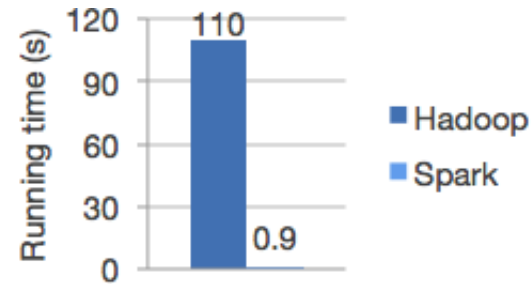
มารู้จักกับ Apache Spark



Apache Spark

เทคโนโลยีในการทำ Distributed Data Processing สำหรับข้อมูลขนาดใหญ่ ที่คิดค้นโดย University of California ในปี 2014

- ✓ เก็บข้อมูลบนหน่วยความจำ (Memory) แทนการเขียนไฟล์เก็บข้อมูลทุกขั้นตอน
- ✓ เก็บการเปลี่ยนแปลงของข้อมูล แทนการเก็บข้อมูลที่เปลี่ยนแปลงแล้ว
- ✓ ทนต่อการล่ม (Fault-tolerant) ด้วย RDD (Resilient Distributed Dataset)
- ✓ สามารถใช้งานได้จากหลายภาษา เช่น Python, R, Java, Scala
- ✓ มี Component ให้ใช้งานเยอะ เช่น MLlib, Spark SQL,

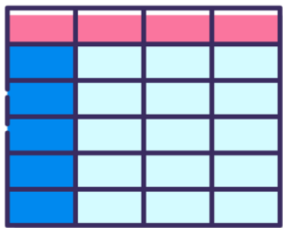


เปรียบเทียบความเร็วในการรัน logistic regression บน Hadoop MapReduce และ Spark

พบว่า Spark เร็วกว่าถึง 100 เท่า



ส่วนเสริม (Module) ที่มากับ Spark



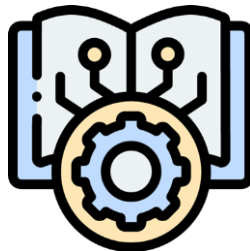
Spark SQL

สามารถเขียน SQL
เพื่อทำงานกับข้อมูลบน
Spark ได้



Spark Streaming

เขียนโปรแกรมรองรับ
ข้อมูลที่เข้ามาอย่างรวดเร็ว



MLlib

สร้างโมเดล Machine
Learning สำหรับข้อมูล
ขนาดใหญ่



GraphX

การประมวลผลข้อมูลใน
รูปแบบของกราฟ

รันบน

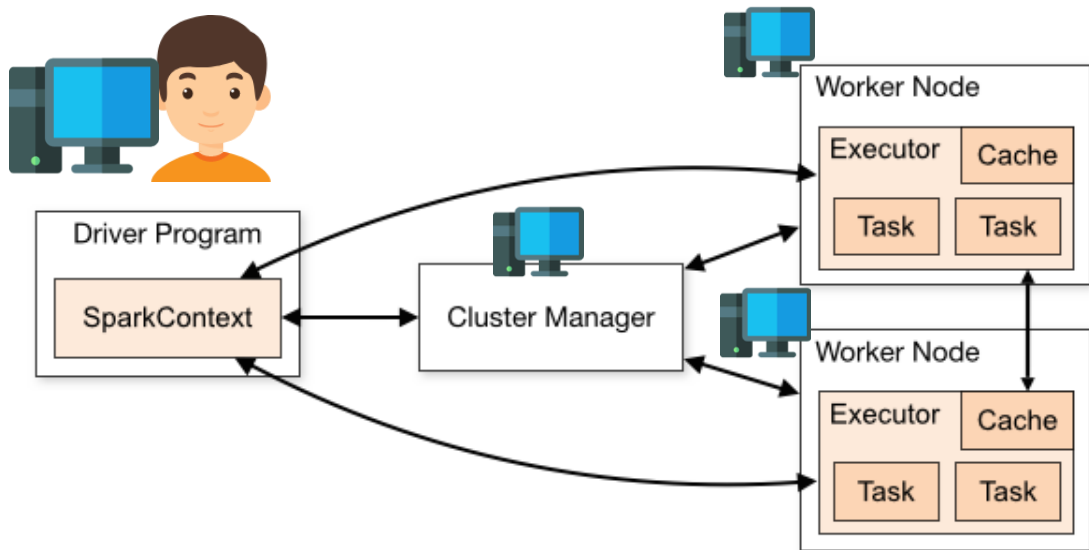


3 วิธี ในการรับคำสั่ง Spark

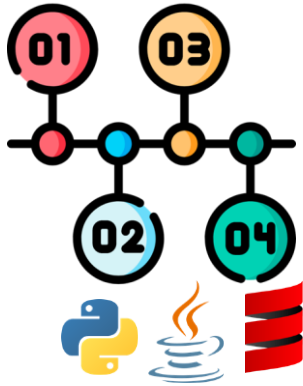


1. ส่งงาน Spark ผ่านการเขียนโปรแกรม เช่น PySpark, SparkR
2. ส่งงาน Spark ที่ละคำสั่ง ด้วย Spark Shell
3. ส่งงาน Spark ด้วยการส่งงาน (Job) ไปรันบน cluster ด้วย Spark Submit

ตัวอย่างการใช้ Spark Submit

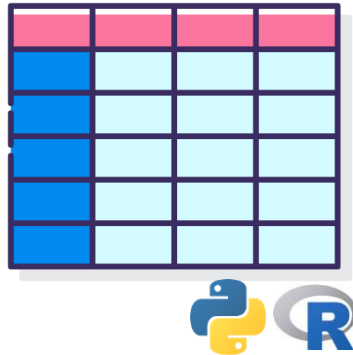


ประเภทข้อมูลใน Spark



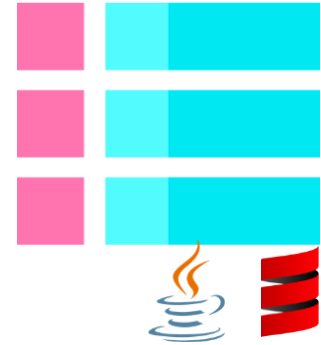
**Resilient Distributed
Datasets (RDD)**
(Java / Scala / Python)

วิธีทำงานกับข้อมูลแบบ
พื้นฐาน มีคำสั่งซับซ้อน



**Spark DataFrame &
Spark SQL (Python / R)**

วิธีทำงานกับข้อมูลแบบ
ตาราง Relational Database



DataSet (Java / Scala)

ข้อมูลแบบ DataFrame ที่มี
การเช็คโค้ดตอน Compile
และบังคับกำหนด Type ข้อมูล

RDD (Resilient Distributed Datasets)

วิธีการเก็บข้อมูลแบบพื้นฐานของ Spark

สามารถรันคำสั่งบน RDD ได้ โดยแบ่งเป็นคำสั่ง 2 แบบ

Transformation

- ไม่ทำงานทันที (Lazy)
เก็บเป็นประวัติไว้ก่อน
- ตอนที่ทำงาน จะสร้าง RDD ใหม่ทุกครั้ง (ข้อมูล RDD ไม่สามารถเขียนทับได้ - Immutable)
- ตัวอย่างคำสั่ง: map(), filter()

Action

- ทำงานทันที และบังคับ Transformation ก่อนหน้านี้ให้ทำงานด้วย
- ผลลัพธ์ไม่ได้เป็น RDD เช่น เป็นตัวเลข หรือ อาจจะไม่มียผลลัพธ์
- ตัวอย่างคำสั่ง: count(), collect(), take(5)



RDD: Transformation vs Action

Transformation:

5.33 ms

ยังไม่ทำงานทันที

```
▶ %%timeit

from pyspark.sql.functions import when

dt.withColumn("CountryNoEIRE", when(dt['Country'] == 'EIRE', 'Ireland').otherwise(dt['Country']))

↳ 100 loops, best of 3: 5.33 ms per loop
```

Action:

63.1 ms

ทำงานทันที และสั่ง
ให้ Transformation
ก่อนหน้านี้ทำงานด้วย

```
▶ %%timeit

from pyspark.sql.functions import when

dt.withColumn("CountryNoEIRE", when(dt['Country'] == 'EIRE', 'Ireland').otherwise(dt['Country'])).take(1)

↳ 10 loops, best of 3: 63.1 ms per loop
```



Spark DataFrame

แปลงข้อมูลเป็นตาราง ทำให้ทำงานกับข้อมูลได้ง่ายขึ้น
คล้าย DataFrame ใน Pandas และ R

- คำสั่ง **select** สำหรับเลือกข้อมูลด้วยเงื่อนไขต่าง ๆ
- คำสั่ง **withColumn** สำหรับเพิ่มคอลัมน์ใหม่
- คำสั่ง **na.fill** สำหรับเติมข้อมูล Missing Values
- แปลงเป็น Pandas DataFrame เพื่อทำ Data Visualisation ได้
- พร้อมใช้งานกับ SQL ด้วย Spark SQL

na ย่อมาจาก Not Applicable
ซึ่งหมายถึง null value หรือ
ข้อมูลที่หายไป

Tip: บางอย่างทำใน DataFrame ง่ายกว่า แต่เราอยากได้ RDD
เราก็สามารถแปลง RDD → DataFrame → RDD ได้



Spark SQL



ใช้ **SQL** ในการดึงข้อมูลจาก Spark DataFrame

มีประโยชน์มากสำหรับการเขียน ETL ด้วย Spark

แปลง Spark DataFrame เป็น **TempView** หรือ **GlobalTempView**



ใช้คำสั่ง SQL ในการดึงหรือแปลงข้อมูลบน TempView

เกร็ดเสริม: Spark 3

Spark 3 ออกมาช่วง June 2020

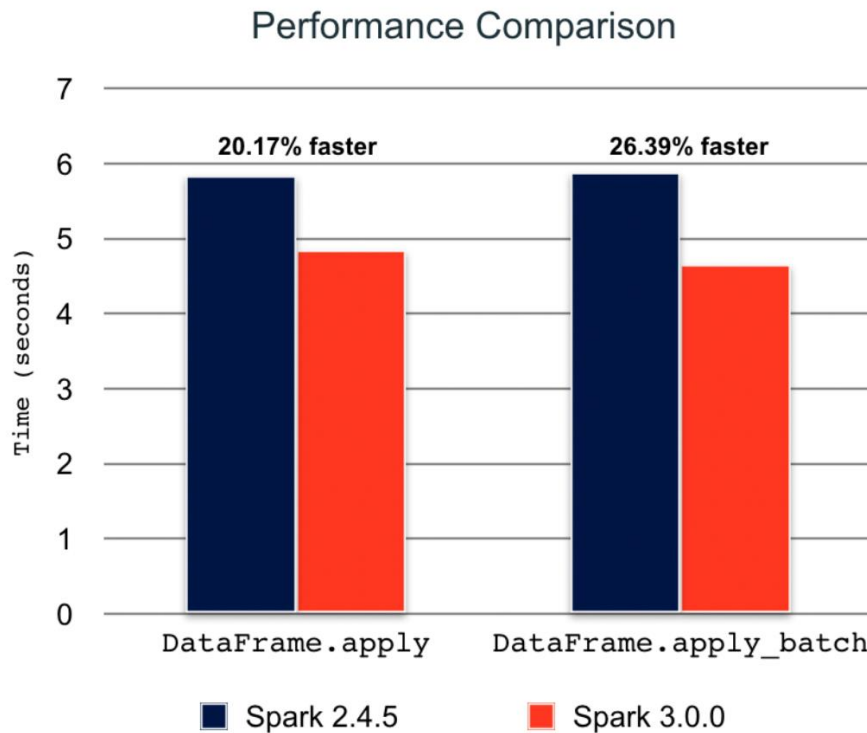
1. เร็วกว่าเดิม

- **Adaptive query execution (AQE)** ปรับการอ่านข้อมูลในระหว่างที่รันงานให้ใช้เวลาให้น้อยที่สุด
- **Dynamic partition pruning** ปรับปรุงการ join ให้เร็วขึ้นโดยการใช้ partition
- **GPU Aware** ใช้งานร่วมกับการ์ดจอ โดยอัตโนมัติ

2. รองรับ SQL ตามมาตรฐาน ANSI

(American National Standards Institute)

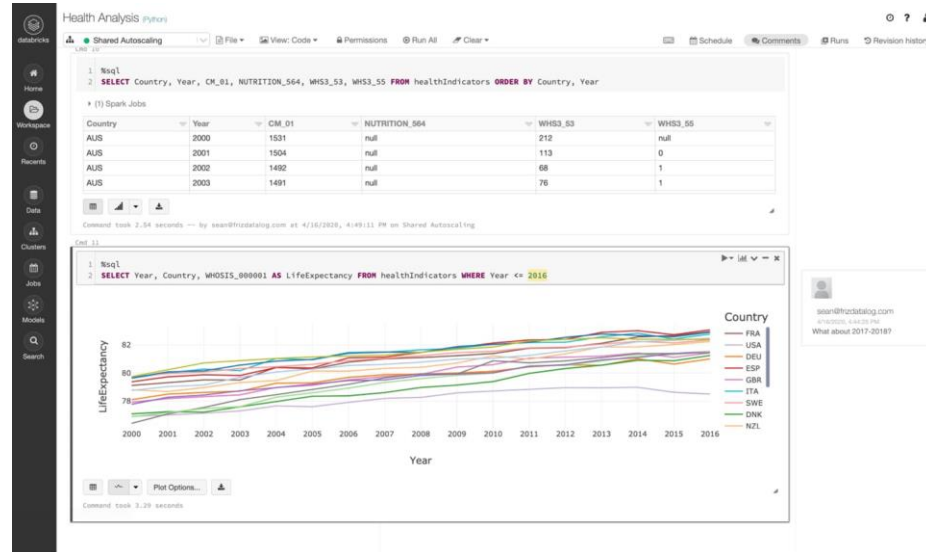
3. PySpark <3 Pandas



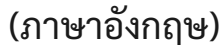
เกร็ดเสริม: Databricks

Data Platform ออนไลน์ โดยทีมพัฒนา Spark

- มี UI แบบ Notebook ให้ใช้งานง่าย รองรับ Python, R, SQL
- รองรับการรัน Spark job ด้วย Spark-submit หรือ Pyspark
- สามารถใช้ได้บน AWS, GCP, และ Microsoft Azure



By DataCamp



RDD:

<https://www.datacamp.com/community/blog/pyspark-cheat-sheet-python>

DataFrame:

<https://www.datacamp.com/community/blog/yspark-sql-cheat-sheet>

By DataTH



<https://blog.datath.com/cheatsheet-pyspark/>

Python for Data Science Cheats Sheet

PySpark - SQL Basics

Learn Python for data science Interactively at www.datacamp.com

PySpark SQL & SparkSQL

PySpark SQL is Apache Spark's module for working with structured data.

Initializing SparkSession

A SparkSession can be used to create DataFrames, register DataFrames as tables, execute SQL over tables, cache tables, and read parquet files.

```
>>> from pyspark.sql import SparkSession
>>> spark = SparkSession \
>>> .builder \
>>> .appName("PySpark SQL basic example") \
>>> .getOrCreate()
>>> spark.conf.get("option", "none-value") \
>>> .getOrCreate()
```

Creating DataFrames

From RDDs

```
>>> from pyspark.sql.types import *
>>> infer Schema
>>> sc = spark.sparkContext
>>> lines = sc.textFile("file:///usr/local/airflow/airflow.log")
>>> parent = lines.map(lambda l: l.split("\t"))
>>> parent = lines.map(lambda p: Row(name=p[0], age=int(p[1]))
>>> .createFromRDD(parent, ["name", "age"])
>>> schema = StructType([
>>>     StructField("name", StringType, True),
>>>     StructField("age", IntegerType, True)
>>> ])
>>> spark.createDataFrame(parent, schema).show()

+-----+-----+
|name   |age   |
+-----+-----+
|John   |21    |
|Diana  |18    |
|Alex   |25    |
|Mike   |19    |
|Ashley |22    |
|Ethan  |20    |
|Olivia |17    |
|Noah   |23    |
|Sophia |16    |
|Liam   |24    |
|Emma   |15    |
|Ava    |14    |
|Lucas  |13    |
|Isabella|12    |
|Evelyn |11    |
|Carter |10    |
|Mia    |9     |
|Ethan  |8     |
|Olivia |7     |
|Noah   |6     |
|Liam   |5     |
|Emma   |4     |
|Ava    |3     |
|Lucas  |2     |
|Isabella|1     |
+-----+-----+
```

From Parquet Data Sources

JSON

```
>>> df = spark.read.json("customer.json")
>>> df.show()

+-----+-----+-----+-----+
|address|email  |last_name|phone_number|
+-----+-----+-----+-----+
|New York|jane.smith@company.com|Smith|212-456-7890|
|New York|john.doe@company.com|Doe|212-555-1234|
|New York|sarah.jones@company.com|Jones|212-555-5678|
|New York|michael.brown@company.com|Brown|212-555-9012|
|New York|emma.wilson@company.com|Wilson|212-555-3456|
|New York|david.miller@company.com|Miller|212-555-7890|
|New York|olivia.garcia@company.com|Garcia|212-555-2345|
|New York|lucas.martinez@company.com|Martinez|212-555-6789|
|New York|sophia.lopez@company.com|Lopez|212-555-0123|
|New York|liam.gonzalez@company.com|Gonzalez|212-555-4567|
|New York|emma.harris@company.com|Harris|212-555-8901|
|New York|noah.clark@company.com|Clark|212-555-2345|
|New York|olivia.lewis@company.com|Lewis|212-555-6789|
|New York|noah.walker@company.com|Walker|212-555-0123|
|New York|olivia.young@company.com|Young|212-555-4567|
|New York|noah.king@company.com|King|212-555-8901|
|New York|olivia.burns@company.com|Burns|212-555-2345|
|New York|noah.hill@company.com|Hill|212-555-6789|
|New York|olivia.scott@company.com|Scott|212-555-0123|
|New York|noah.green@company.com|Green|212-555-4567|
|New York|olivia.adams@company.com|Adams|212-555-8901|
|New York|noah.baker@company.com|Baker|212-555-2345|
|New York|olivia.nelson@company.com|Nelson|212-555-6789|
|New York|noah.carter@company.com|Carter|212-555-0123|
|New York|olivia.mitchell@company.com|Mitchell|212-555-4567|
|New York|noah.perez@company.com|Perez|212-555-8901|
|New York|olivia.roberts@company.com|Roberts|212-555-2345|
|New York|noah.taylor@company.com|Taylor|212-555-6789|
|New York|olivia.anderson@company.com|Anderson|212-555-0123|
|New York|noah.phillips@company.com|Phillips|212-555-4567|
|New York|olivia.cook@company.com|Cook|212-555-8901|
|New York|noah.morgan@company.com|Morgan|212-555-2345|
|New York|olivia.bell@company.com|Bell|212-555-6789|
|New York|noah.foster@company.com|Foster|212-555-0123|
|New York|olivia.perry@company.com|Perry|212-555-4567|
|New York|noah.henderson@company.com|Henderson|212-555-8901|
|New York|olivia.coleman@company.com|Coleman|212-555-2345|
|New York|noah.jenkins@company.com|Jenkins|212-555-6789|
|New York|olivia.peters@company.com|Peters|212-555-0123|
|New York|noah.hart@company.com|Hart|212-555-4567|
|New York|olivia.murray@company.com|Murray|212-555-8901|
|New York|noah.sullivan@company.com|Sullivan|212-555-2345|
|New York|olivia.watson@company.com|Watson|212-555-6789|
|New York|noah.flynn@company.com|Flynn|212-555-0123|
|New York|olivia.hughes@company.com|Hughes|212-555-4567|
|New York|noah.burns@company.com|Burns|212-555-8901|
|New York|olivia.hill@company.com|Hill|212-555-2345|
|New York|noah.scott@company.com|Scott|212-555-6789|
|New York|olivia.green@company.com|Green|212-555-0123|
|New York|noah.adams@company.com|Adams|212-555-4567|
|New York|olivia.baker@company.com|Baker|212-555-8901|
|New York|olivia.nelson@company.com|Nelson|212-555-2345|
|New York|noah.carter@company.com|Carter|212-555-6789|
|New York|olivia.mitchell@company.com|Mitchell|212-555-0123|
|New York|noah.perez@company.com|Perez|212-555-4567|
|New York|olivia.roberts@company.com|Roberts|212-555-8901|
|New York|noah.taylor@company.com|Taylor|212-555-2345|
|New York|olivia.anderson@company.com|Anderson|212-555-6789|
|New York|noah.phillips@company.com|Phillips|212-555-0123|
|New York|olivia.cook@company.com|Cook|212-555-4567|
|New York|noah.morgan@company.com|Morgan|212-555-8901|
|New York|olivia.bell@company.com|Bell|212-555-2345|
|New York|noah.foster@company.com|Foster|212-555-6789|
|New York|olivia.perry@company.com|Perry|212-555-0123|
|New York|noah.henderson@company.com|Henderson|212-555-4567|
|New York|olivia.coleman@company.com|Coleman|212-555-8901|
|New York|noah.jenkins@company.com|Jenkins|212-555-2345|
|New York|olivia.peters@company.com|Peters|212-555-6789|
|New York|noah.hart@company.com|Hart|212-555-0123|
|New York|olivia.murray@company.com|Murray|212-555-4567|
|New York|noah.sullivan@company.com|Sullivan|212-555-8901|
|New York|olivia.watson@company.com|Watson|212-555-2345|
|New York|noah.flynn@company.com|Flynn|212-555-6789|
|New York|olivia.hughes@company.com|Hughes|212-555-0123|
|New York|noah.burns@company.com|Burns|212-555-4567|
|New York|olivia.hill@company.com|Hill|212-555-8901|
|New York|noah.scott@company.com|Scott|212-555-2345|
|New York|olivia.green@company.com|Green|212-555-6789|
|New York|noah.adams@company.com|Adams|212-555-0123|
|New York|olivia.baker@company.com|Baker|212-555-4567|
|New York|olivia.nelson@company.com|Nelson|212-555-8901|
|New York|noah.carter@company.com|Carter|212-555-2345|
|New York|olivia.mitchell@company.com|Mitchell|212-555-6789|
|New York|noah.perez@company.com|Perez|212-555-0123|
|New York|olivia.roberts@company.com|Roberts|212-555-4567|
|New York|noah.taylor@company.com|Taylor|212-555-8901|
|New York|olivia.anderson@company.com|Anderson|212-555-2345|
|New York|noah.phillips@company.com|Phillips|212-555-6789|
|New York|olivia.cook@company.com|Cook|212-555-0123|
|New York|noah.morgan@company.com|Morgan|212-555-4567|
|New York|olivia.bell@company.com|Bell|212-555-8901|
|New York|noah.foster@company.com|Foster|212-555-2345|
|New York|olivia.perry@company.com|Perry|212-555-6789|
|New York|noah.henderson@company.com|Henderson|212-555-0123|
|New York|olivia.coleman@company.com|Coleman|212-555-4567|
|New York|noah.jenkins@company.com|Jenkins|212-555-8901|
|New York|olivia.peters@
```

The image is a promotional graphic for a PySpark cheatsheet. At the top left is the 'DataTH BLOG' logo. A navigation bar contains links: 'Data Science', 'Data Engineer', 'Interviews', and 'Free Courses & Books'. The central part features the Apache Spark logo with Thai text: 'คู่มือ Clean Data ด้วย' (Clean Data Guide with), 'APACHE Spark', and 'เครื่องมือ Big Data ที่บริษัททั่วโลกใช้' (Big Data tool used by companies worldwide). Below this is a section header 'Cheatsheet วิธีใช้ และเทคนิคใน Pyspark ฉบับสมบูรณ์' (Complete PySpark Usage and Techniques Cheatsheet). Underneath is a table of contents with 27 items, starting with '1 PySpark คืออะไร' (What is PySpark) and ending with '2.9 วิธีนับจำนวนแถว' (How to count rows).

Workshop 2:

Data Cleansing with Spark



Workshop 2 – Data Cleansing with Spark

มาลองใช้ Apache Spark เพื่อให้ข้อมูลของเรามีคุณภาพกันเถอะ

Input:

- ข้อมูลดิบ (CSV)



Output:

- ข้อมูลที่ทำความสะอาดเรียบร้อยแล้ว (CSV)

