

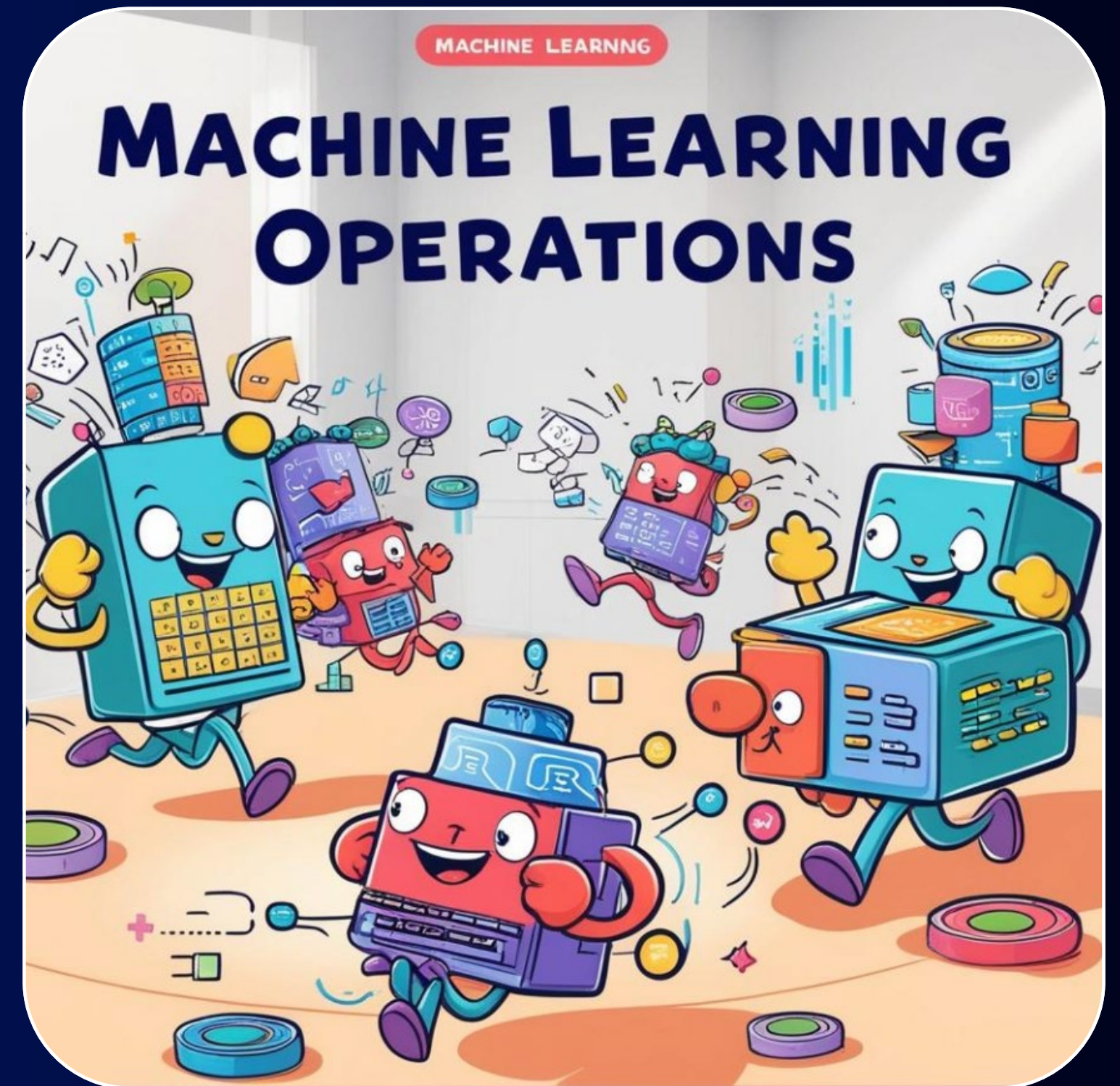
MACHINE LEARNING OPERATIONS



Dataset_version

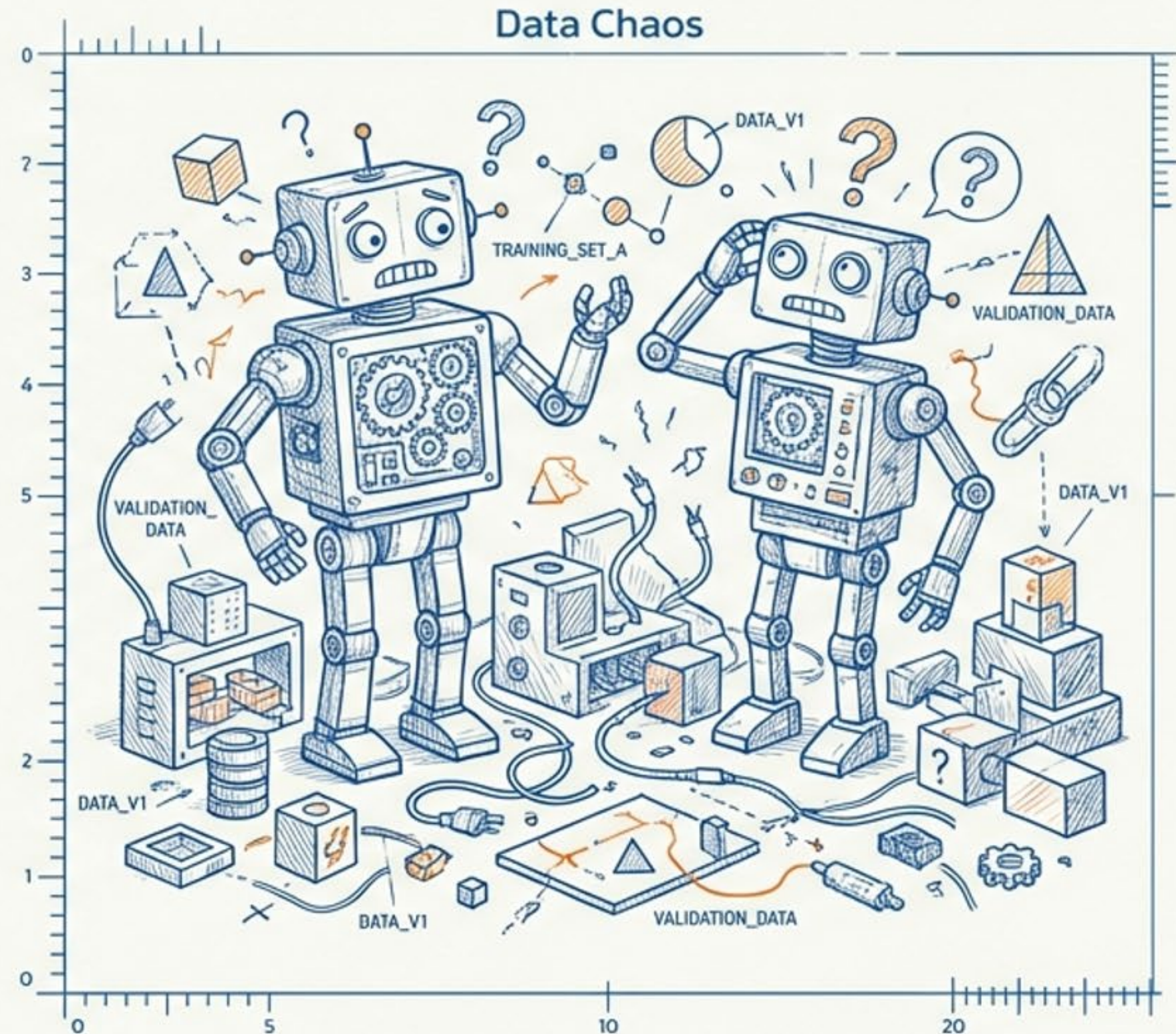


Presented by Asst. Prof. Dr. Tuchsana Ploysuwan



ปัญหาที่พบบ่อย: เมื่อข้อมูลไร้การควบคุม

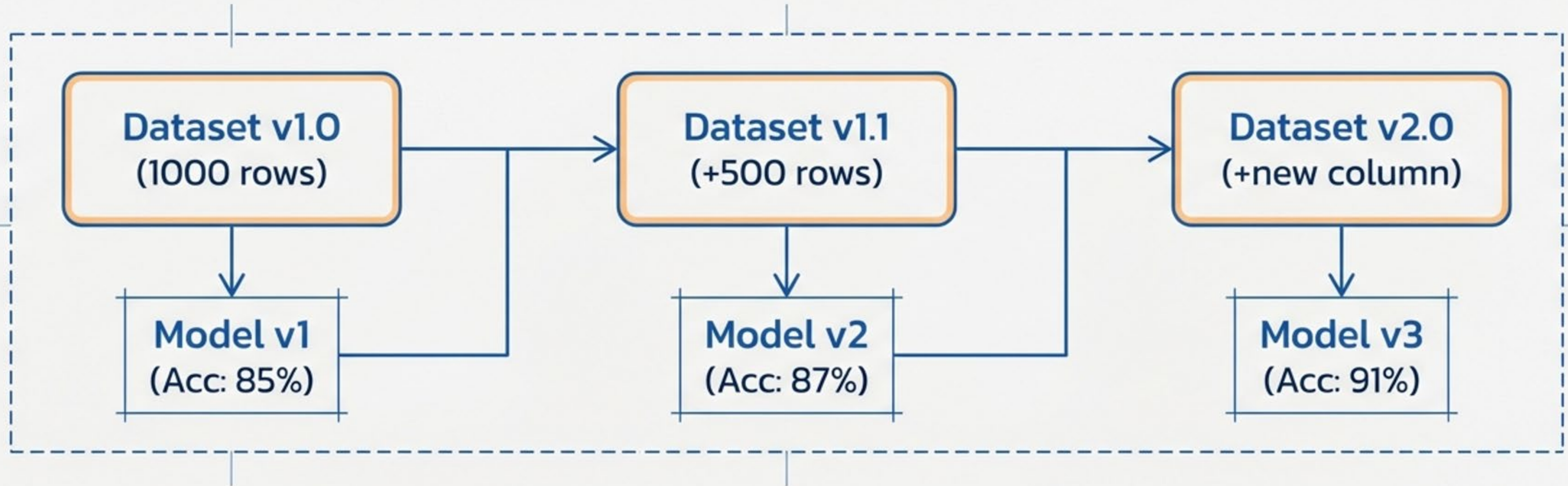
- ❌ Model ทำงานได้ดีเมื่อวานนี้ แต่วันนี้ Accuracy ตก ไม่รู้ว่าเปลี่ยนอะไรไป
- ❌ ใช้ Data ชุดไหนในการ Train Model version ที่ deploy อยู่?
- ❌ ใครแก้ไข Dataset? เมื่อไหร่? แก้อะไร?
- ❌ ต้องการกลับไปใช้ Data version เดิม แต่ไม่มี backup



Without Versioning, Machine Learning is just guesswork.

Dataset Versioning คืออะไร?

กระบวนการติดตามและจัดการการเปลี่ยนแปลงของชุดข้อมูลอย่างเป็นระบบ คล้ายกับที่ Git ทำกับ Source Code



Traceability:

ติดตามได้ว่า Model
ใช้ Data version ไหน



Reproducibility:

สามารถ reproduce
ผลลัพธ์ได้



Rollback:

กลับไป version
เดิมได้เมื่อเกิดปัญหา

การเลือก Versioning Strategy

1. Semantic Versioning

MAJOR.MINOR.PATCH

เหมาะกับ Production
และการสื่อสารในทีม

v1.0.0 → v1.1.0 → v2.0.0

v1.1.0 = Added rows,
v2.0.0 = Schema change

2. Date-based Versioning

YYYY-MM-DD

เหมาะกับ Daily Snapshots
หรือ Time-series

2024-01-01, 2024-02-15

Best for scheduled updates

3. Hash-based Versioning

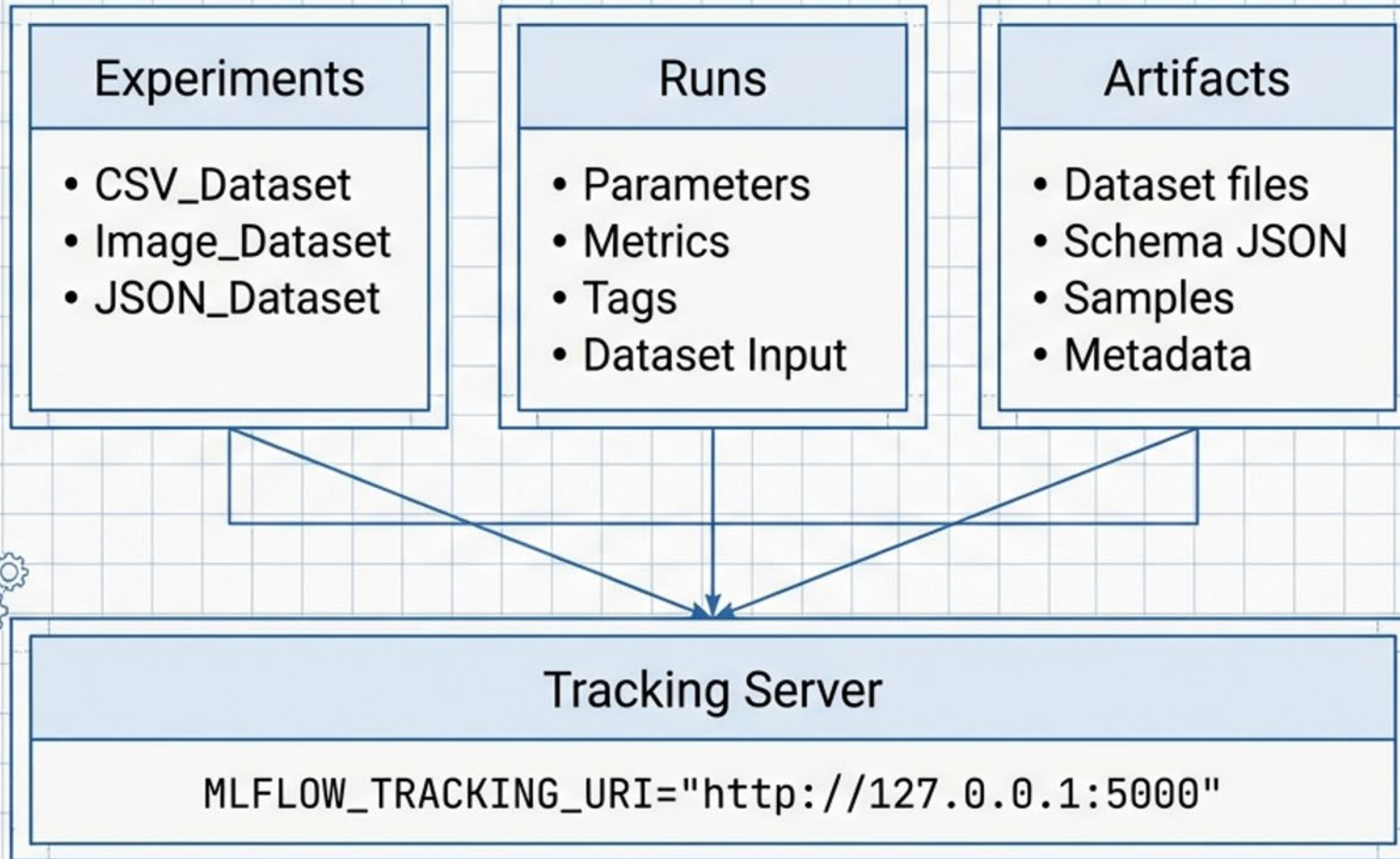
MD5 / Content Hash

The Gold Standard.
อัตโนมัติและแม่นยำที่สุด

a1b2c3d4...

Integrity guaranteed

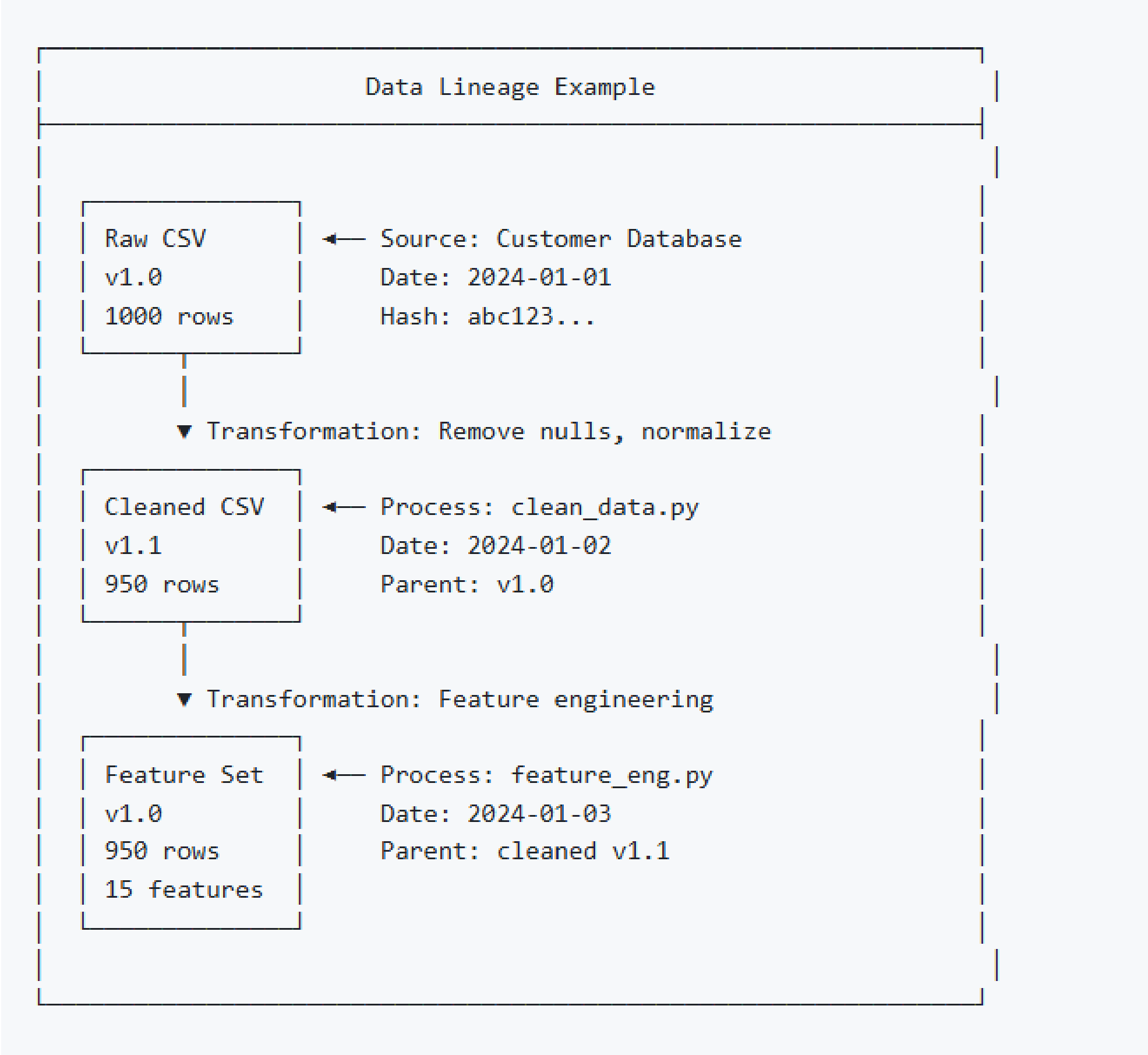
MLflow Architecture สำหรับ Dataset Tracking



- **Centralized Tracking:**
เก็บข้อมูลทุกอย่างไว้ที่เดียว
- **Metadata Logging:**
บันทึก schema และ statistics
- **UI Dashboard:**
ดูและเปรียบเทียบ versions ได้ง่าย

Dataset Versioning for Csv files

Data Lineage คือการติดตามที่มาและการเปลี่ยนแปลงของข้อมูลตลอด Pipeline



2.3 Dataset Versioning Strategies

การเลือก Versioning Strategy ที่เหมาะสมเป็นสิ่งสำคัญในการจัดการ Dataset ให้มีประสิทธิภาพ แต่ละ Strategy มีข้อดีและข้อเสียที่แตกต่างกัน ขึ้นอยู่กับลักษณะของโปรเจกต์และทีม

Strategy 1: Semantic Versioning (MAJOR.MINOR.PATCH)

Semantic Versioning ใช้หลักการเดียวกับ Software Versioning ที่เป็นที่ยอมรับ โดยแบ่งเลข version เป็น 3 ส่วน

```
# Version Number Meaning:
# MAJOR: Breaking changes (schema change, column removal)
# MINOR: Backward compatible additions (new rows, new columns)
# PATCH: Bug fixes (data corrections)

version_examples = {
    "1.0.0": "Initial dataset",
    "1.1.0": "Added 500 new samples",           # Minor: more data
    "1.1.1": "Fixed typos in labels",          # Patch: corrections
    "2.0.0": "Added 'region' column",          # Major: schema change
}
```

เมื่อไหร่ควรใช้:

- เมื่อต้องการสื่อสารลักษณะการเปลี่ยนแปลงให้ทีมเข้าใจ
- เมื่อมี downstream dependencies ที่ต้องรู้ว่า data เปลี่ยนแปลงอย่างไร
- เหมาะกับ production datasets ที่มี consumers หลายคน

Strategy 2: Date-based Versioning

Date-based Versioning เหมาะกับ datasets ที่มีการ update เป็นประจำ เช่น daily snapshots หรือ monthly reports

```
# Format: YYYY-MM-DD or YYYYMMDD

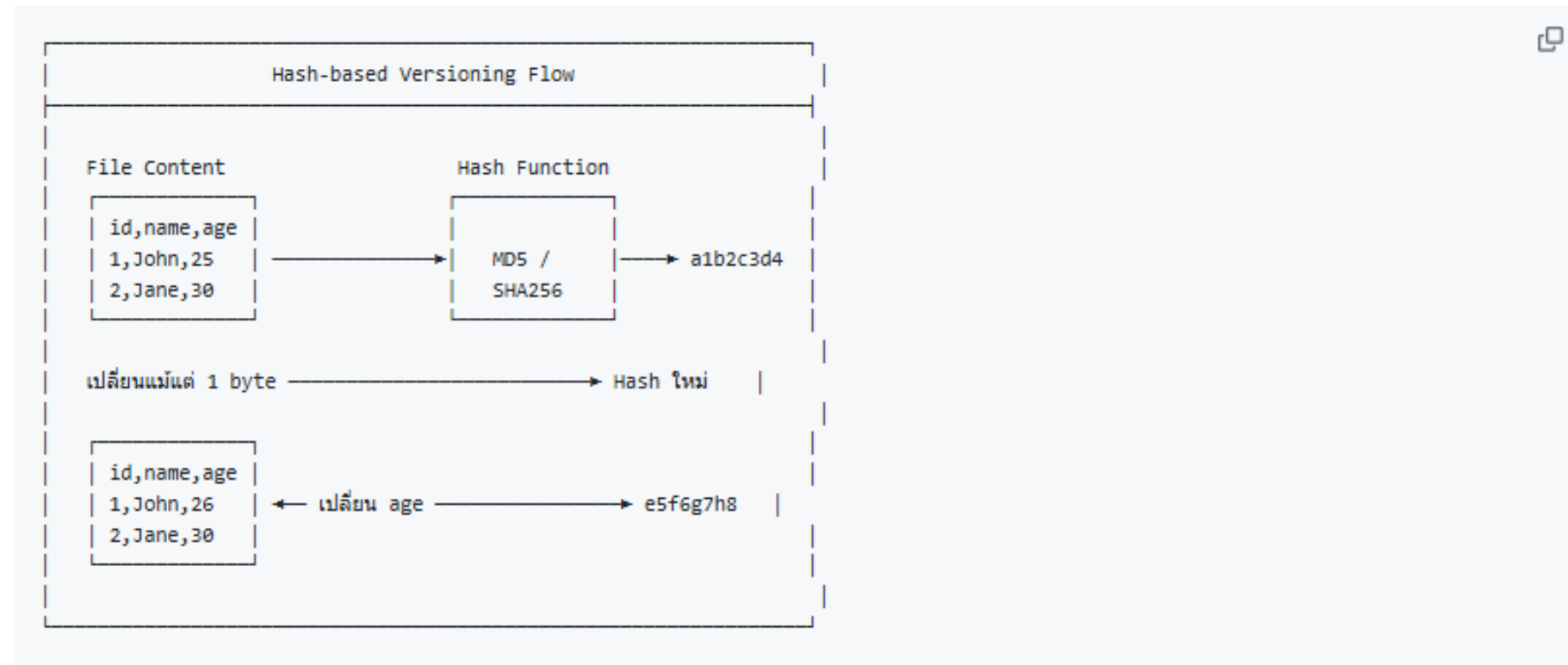
version_examples = {
    "2024-01-01": "January snapshot",
    "2024-02-01": "February snapshot",
    "2024-02-15": "Mid-month update",
}
```

เมื่อไหร่ควรใช้:

- Datasets ที่ update เป็น schedule (daily, weekly, monthly)
- Time-series data หรือ snapshot data
- เมื่อ "เวลา" เป็นข้อมูลที่สำคัญในการ track

Strategy 3: Hash-based Versioning

Hash-based Versioning ใช้ Content Hash เป็น Version Identifier ซึ่งเป็นวิธีที่ **อัตโนมัติและแม่นยำที่สุด** เนื่องจาก version จะเปลี่ยนก็ต่อเมื่อเนื้อหาของไฟล์เปลี่ยนจริงๆ เท่านั้น



ข้อดีของ Hash-based Versioning:

ข้อดี	คำอธิบาย
Automatic	ไม่ต้อง manual update version number
Unique	แต่ละ content มี hash เฉพาะตัว
Integrity Check	ตรวจสอบความถูกต้องของไฟล์ได้
Deduplication	ตรวจจับ duplicate datasets ได้
Reproducibility	มั่นใจได้ว่าใช้ data ชุดเดียวกัน

ข้อดี	คำอธิบาย
Automatic	ไม่ต้อง manual update version number
Unique	แต่ละ content มี hash เฉพาะตัว
Integrity Check	ตรวจสอบความถูกต้องของไฟล์ได้
Deduplication	ตรวจจับ duplicate datasets ได้
Reproducibility	มั่นใจได้ว่าใช้ data ชุดเดียวกัน

```
import hashlib
import pandas as pd
import os

def get_dataset_version(filepath):
    """
    สร้าง version จาก file content hash

    Args:
        filepath: path ไปยังไฟล์ dataset

    Returns:
        str: 8-character hash string เป็น version identifier
    """
    hash_md5 = hashlib.md5()
    with open(filepath, "rb") as f:
        for chunk in iter(lambda: f.read(4096), b''):
            hash_md5.update(chunk)
    return hash_md5.hexdigest()[:8] # โช้ 8 characters แรก

# ตัวอย่างการใช้งาน
version = get_dataset_version("data/customers.csv")
print(f"Dataset Version: {version}") # ผลลัพธ์: "a1b2c3d4"
```

การจัดการข้อมูลตาราง (CSV Versioning)

ID	Name	Age	City	Churn	Date
2001	Mike	25	Massena	False	2024-07-30
2002	John	29	City	True	2024-03-28
2003	John	33	Massena	True	2024-03-10
2004	John	33	Massena	False	2024-05-21

Version 1

Update

ID	Name	Age	City	Region	Churn	Date
2000	Mike	29	Massena	North	True	2024-03-29
2001	John	29	City	North	True	2024-03-28
2002	John	33	City	South	False	2024-03-30
2003	John	30	Massena	West	True	2024-05-20
2004	John	36	Massena	West	False	2024-05-20
2005	John	35	Massena	North	True	2024-05-24

Version 2

```
mlflow.data.from_pandas(df, targets="churn")
```

Strategy 1: การ Track ข้อมูล CSV (Tabular Data)

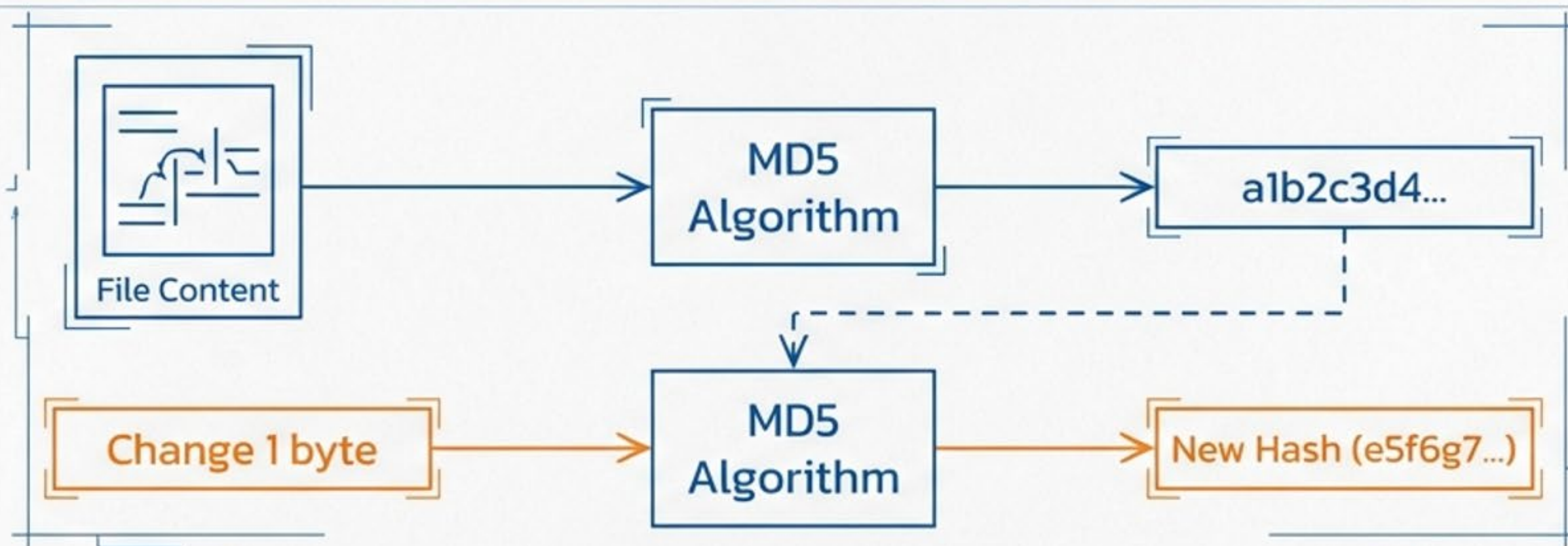
Key Metrics to Log

- num_rows (จำนวนแถว)
- num_columns (จำนวนคอลัมน์)
- memory_usage_mb
- missing_values
- churn_rate (Specific Metric)

```
1 # Create MLflow dataset
2 dataset = mlflow.data.from_pandas(
3     df,
4     source=csv_path_v1,
5     name="customer_churn_dataset",
6     targets="churn"
7 )
8
9 # Log input and metrics
10 mlflow.log_input(dataset, context="training")
11 mlflow.log_metric("num_rows", len(df))
12 mlflow.log_metric("churn_rate", df['churn'].mean())
```

หัวใจสำคัญ:
Convert Pandas
to MLflow Dataset

The Fingerprint: การยืนยันความถูกต้องด้วย Hash



ป้องกันการแอบแก้ไขข้อมูล (Prevents silent corruption)

MLFlow Integrity Check Implementation

```
1 def calculate_file_hash(filepath):
2     hash_md5 = hashlib.md5()
3     with open(filepath, "rb") as f:
4         # Read in chunks
5         for chunk in iter(lambda: f.read(4096), b''):
6             hash_md5.update(chunk)
7     return hash_md5.hexdigest()
8
9 # Log the fingerprint
10 mlflow.log_param("file_hash_md5", file_hash)
```

การใช้งานร่วมกับ Standard Libraries (Scikit-Learn)

MLflow รองรับ sklearn.datasets โดยตรง



Iris dataset



Wine dataset



California Housing

```
from sklearn.datasets import load_iris

# Direct integration
dataset = mlflow.data.from_pandas(
    df,
    source="sklearn.datasets.load_iris",
    name="iris_dataset"
)
```

The Dashboard: การเปรียบเทียบ Dataset Versions

MLflow Comparison UI table				
Run Name	Version	Rows	Churn Rate	Memory (MB)
customer_dataset_v2	2.0.0	1500	25.67%	0.15
customer_dataset_v1	1.0.0	1000	19.60%	0.05

ตารางเปรียบเทียบทำให้
เห็นความเปลี่ยนแปลง
ของข้อมูลได้ทันที

Data Drift
detected

4.2 Track CSV Dataset with MLflow

Now let's create an MLflow experiment and log our CSV datasets with proper versioning and metadata.

```
[5]: def calculate_file_hash(filepath):  
    """Calculate MD5 hash of a file for versioning."""  
    hash_md5 = hashlib.md5()  
    with open(filepath, "rb") as f:  
        for chunk in iter(lambda: f.read(4096), b''):  
            hash_md5.update(chunk)  
    return hash_md5.hexdigest()  
  
def get_dataset_stats(df):  
    """Generate statistics for a DataFrame."""  
    stats = {  
        "num_rows": len(df),  
        "num_columns": len(df.columns),  
        "columns": list(df.columns),  
        "dtypes": {col: str(dtype) for col, dtype in df.dtypes.items()},  
        "missing_values": df.isnull().sum().to_dict(),  
        "memory_usage_mb": df.memory_usage(deep=True).sum() / (1024 * 1024)  
    }  
  
    # Add numeric column statistics  
    numeric_cols = df.select_dtypes(include=[np.number]).columns  
    if len(numeric_cols) > 0:  
        stats["numeric_stats"] = df[numeric_cols].describe().to_dict()  
  
    return stats  
  
print("✅ Helper functions created!")
```

✅ Helper functions created!

```
[6]: # Create experiment for CSV datasets  
EXPERIMENT_NAME_CSV = "CSV_Dataset_Versioning_Lab"  
  
# Set or create experiment  
mlflow.set_experiment(EXPERIMENT_NAME_CSV)  
experiment = mlflow.get_experiment_by_name(EXPERIMENT_NAME_CSV)  
print(f"📁 Experiment: {EXPERIMENT_NAME_CSV}")  
print(f"    Experiment ID: {experiment.experiment_id}")
```

2026/02/02 08:46:47 INFO mlflow.tracking.fluent: Experiment with name 'CSV_Dataset_Versioning_Lab' does not exist. Creating a new experiment.

📁 Experiment: CSV_Dataset_Versioning_Lab
Experiment ID: 1

```
# Log CSV Dataset Version 1
print("📄 Logging CSV Dataset Version 1...")

with mlflow.start_run(run_name="customer_dataset_v1") as run:
    # Load the dataset for MLflow
    df = pd.read_csv(csv_path_v1)

    # Create MLflow dataset
    dataset = mlflow.data.from_pandas(
        df,
        source=csv_path_v1,
        name="customer_churn_dataset",
        targets="churn"
    )

    # Log the dataset
    mlflow.log_input(dataset, context="training")

    # Log dataset metadata as parameters
    mlflow.log_param("dataset_version", "1.0.0")
    mlflow.log_param("dataset_name", "customer_churn")
    mlflow.log_param("data_source", "synthetic")
    mlflow.log_param("creation_date", datetime.now().isoformat())

    # Log dataset statistics as metrics
    stats = get_dataset_stats(df)
    mlflow.log_metric("num_rows", stats["num_rows"])
    mlflow.log_metric("num_columns", stats["num_columns"])
    mlflow.log_metric("memory_mb", stats["memory_usage_mb"])
    mlflow.log_metric("missing_total", sum(stats["missing_values"].values()))
    mlflow.log_metric("churn_rate", df['churn'].mean())

    # Log the actual CSV file as artifact
    mlflow.log_artifact(csv_path_v1, artifact_path="datasets")

    # Log schema information
    schema_info = {
        "columns": stats["columns"],
        "dtypes": stats["dtypes"],
        "version": "1.0.0"
    }
    schema_path = os.path.join(CSV_DIR, "schema_v1.json")
    with open(schema_path, 'w') as f:
        json.dump(schema_info, f, indent=2)
    mlflow.log_artifact(schema_path, artifact_path="schema")

    # Log file hash for integrity
    file_hash = calculate_file_hash(csv_path_v1)
    mlflow.log_param("file_hash_md5", file_hash)

run_id_v1 = run.info.run_id
print(f"✅ Dataset v1 logged successfully!")
print(f"Run ID: {run_id_v1}")
print(f"File Hash: {file_hash}")
```

MLflow 3.8.1

CSV_Dataset_Versioning_Lab Machine learning

Runs Models Traces

metrics.rmse < 1 and params.model = "tree" Time created State: Active Datasets

Sort: Created Columns Group by

Run Name	Created	Dataset	Duration	Source	Models
customer_dataset_v1	24 seconds ago	customer_churn_dataset (5a57bafe)	207ms	ipykerne...	-

CSV_Dataset_Versioning_Lab > Runs >

customer_dataset_v1

Overview

Model metrics System metrics Traces Artifacts

Description

No description

Metrics (5)

Search metrics

Metric	Value
num_rows	1000
num_columns	6
memory_mb	0.045902252197265625
missing_total	0
churn_rate	0.196

Parameters (5)

Search parameters

Parameter	Value
dataset_version	1.0.0
dataset_name	customer_churn
data_source	synthetic
creation_date	2026-02-02T08:49:17.216849
file_hash_md5	318f2f9e4cdf686ac61acb8f1cb3020b

```
# Log CSV Dataset Version 1
print("📄 Logging CSV Dataset Version 1...")

with mlflow.start_run(run_name="customer_dataset_v1") as run:
    # Load the dataset for MLflow
    df = pd.read_csv(csv_path_v1)

    # Create MLflow dataset
    dataset = mlflow.data.from_pandas(
        df,
        source=csv_path_v1,
        name="customer_churn_dataset",
        targets="churn"
    )

    # Log the dataset
    mlflow.log_input(dataset, context="training")

    # Log dataset metadata as parameters
    mlflow.log_param("dataset_version", "1.0.0")
    mlflow.log_param("dataset_name", "customer_churn")
    mlflow.log_param("data_source", "synthetic")
    mlflow.log_param("creation_date", datetime.now().isoformat())

    # Log dataset statistics as metrics
    stats = get_dataset_stats(df)
    mlflow.log_metric("num_rows", stats["num_rows"])
    mlflow.log_metric("num_columns", stats["num_columns"])
    mlflow.log_metric("memory_mb", stats["memory_usage_mb"])
    mlflow.log_metric("missing_total", sum(stats["missing_values"].values()))
    mlflow.log_metric("churn_rate", df['churn'].mean())

    # Log the actual CSV file as artifact
    mlflow.log_artifact(csv_path_v1, artifact_path="datasets")

    # Log schema information
    schema_info = {
        "columns": stats["columns"],
        "dtypes": stats["dtypes"],
        "version": "1.0.0"
    }
    schema_path = os.path.join(CSV_DIR, "schema_v1.json")
    with open(schema_path, 'w') as f:
        json.dump(schema_info, f, indent=2)
    mlflow.log_artifact(schema_path, artifact_path="schema")

    # Log file hash for integrity
    file_hash = calculate_file_hash(csv_path_v1)
    mlflow.log_param("file_hash_md5", file_hash)

run_id_v1 = run.info.run_id
print(f"✅ Dataset v1 logged successfully!")
print(f"Run ID: {run_id_v1}")
print(f"File Hash: {file_hash}")
```

The screenshot shows the MLflow web interface for a run named 'customer_dataset_v1'. The 'Artifacts' tab is selected, showing a tree view of artifacts. The 'datasets' folder is expanded, revealing a file named 'customers_v1.csv'. The 'schema' folder is also visible. The path for the datasets artifact is displayed as 'mlflow-artifacts:/1/616082d841eb4236b869bfb913b744d3/artifacts/datasets'.

mlflow 3.8.1

CSV_Dataset_Versioning_Lab > Runs >

customer_dataset_v1

Overview Model metrics System metrics Traces **Artifacts**

▼ datasets

customers_v1.csv

▶ schema

datasets

Path: mlflow-artifacts:/1/616082d841eb4236b869bfb913b744d3/artifacts/datasets

```
# Log CSV Dataset Version 2
print("📁 Logging CSV Dataset Version 2...")

with mlflow.start_run(run_name="customer_dataset_v2") as run:
    # Load the dataset
    df = pd.read_csv(csv_path_v2)

    # Create MLflow dataset
    dataset = mlflow.data.from_pandas(
        df,
        source=csv_path_v2,
        name="customer_churn_dataset",
        targets="churn"
    )

    # Log the dataset
    mlflow.log_input(dataset, context="training")

    # Log dataset metadata
    mlflow.log_param("dataset_version", "2.0.0")
    mlflow.log_param("dataset_name", "customer_churn")
    mlflow.log_param("data_source", "synthetic")
    mlflow.log_param("creation_date", datetime.now().isoformat())
    mlflow.log_param("changes_from_v1", "Added region column, more samples")

    # Log statistics
    stats = get_dataset_stats(df)
    mlflow.log_metric("num_rows", stats["num_rows"])
    mlflow.log_metric("num_columns", stats["num_columns"])
    mlflow.log_metric("memory_mb", stats["memory_usage_mb"])
    mlflow.log_metric("missing_total", sum(stats["missing_values"].values()))
    mlflow.log_metric("churn_rate", df['churn'].mean())

    # Log artifact
    mlflow.log_artifact(csv_path_v2, artifact_path="datasets")

    # Log schema
    schema_info = {
        "columns": stats["columns"],
        "dtypes": stats["dtypes"],
        "version": "2.0.0"
    }
    schema_path = os.path.join(CSV_DIR, "schema_v2.json")
    with open(schema_path, 'w') as f:
        json.dump(schema_info, f, indent=2)
    mlflow.log_artifact(schema_path, artifact_path="schema")

    file_hash = calculate_file_hash(csv_path_v2)
    mlflow.log_param("file_hash_md5", file_hash)

run_id_v2 = run.info.run_id
print(f"✅ Dataset v2 logged successfully!")
print(f"Run ID: {run_id_v2}")
print(f"File Hash: {file_hash}")
```

MLflow 3.8.1 interface showing the 'CSV_Dataset_Versioning_Lab' experiment. The 'Runs' tab is selected, displaying a list of runs. The run 'customer_dataset_v2' is highlighted. The interface includes search filters, sorting options, and a table of runs.

Run Name	Created	Dataset	Duration	Source	Models
customer_dataset_v2	13 seconds ago	customer_churn_dataset (5305052f)	201ms	ipykerne...	-
customer_dataset_v1	54 minutes ago	customer_churn_dataset (5a57bafe)	207ms	ipykerne...	-

MLflow 3.8.1 interface showing the 'customer_dataset_v2' run details. The 'Artifacts' tab is selected, displaying a tree view of artifacts. The 'customers_v2.csv' file is highlighted. The interface includes tabs for Overview, Model metrics, System metrics, Traces, and Artifacts.

Overview | Model metrics | System metrics | Traces | **Artifacts**

datasets

- customers_v2.csv
- schema

Path: mlflow-artifacts:/1/1450ce3ab1cf4e4180eece864c631406/artifacts/datasets

Compare CSV Dataset Versions

```
: # Compare dataset versions
print("📊 Comparing Dataset Versions:")
print("=" * 60)

client = mlflow.tracking.MlflowClient()

# Get runs from the experiment
runs = client.search_runs(
    experiment_ids=[experiment.experiment_id],
    order_by=["start_time DESC"]
)

comparison_data = []
for run in runs:
    if "customer dataset" in run.info.run_name:
        version_info = {
            "Run Name": run.info.run_name,
            "Version": run.data.params.get("dataset_version", "N/A"),
            "Rows": run.data.metrics.get("num_rows", "N/A"),
            "Columns": run.data.metrics.get("num_columns", "N/A"),
            "Churn Rate": f"{run.data.metrics.get('churn_rate', 0):.2%}",
            "Memory (MB)": f"{run.data.metrics.get('memory_mb', 0):.2f}"
        }
        comparison_data.append(version_info)

comparison_df = pd.DataFrame(comparison_data)
print(comparison_df.to_string(index=False))
```

📊 Comparing Dataset Versions:

```
=====
      Run Name Version  Rows  Columns Churn Rate Memory (MB)
customer_dataset_v2  2.0.0 1500.0     7.0    25.67%      0.15
customer_dataset_v1  1.0.0 1000.0     6.0    19.60%      0.05
```

Dataset Versioning for Images files

The Challenge: Tabular vs. Image Data

ความท้าทายของข้อมูลรูปภาพ



โครงสร้างชัดเจน เปรียบเทียบความต่างได้ง่าย

Structure is clear. Changes are row-based. Easy to Diff.

ID	Name	Value
001	Product A	5400
002	Product B	5400
003	Product C	5400
004	Product D	5400
005	Product E	5400



ไม่มีโครงสร้าง ไฟล์ขนาดใหญ่ เปรียบเทียบยาก

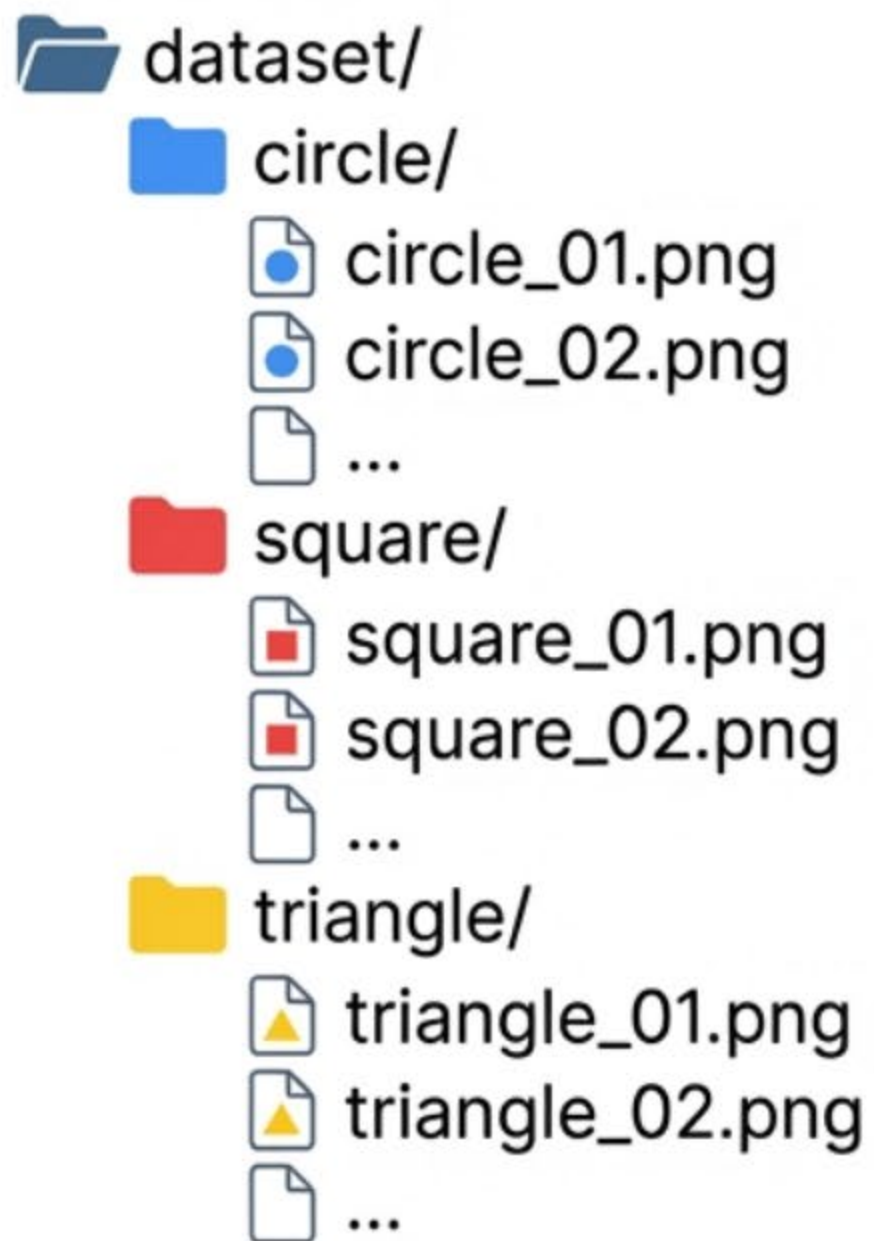
Unstructured. Heavy binary files. No easy 'Diff'.



For images, we cannot just track the file content.
We must track **Metadata** (parameters) and **Visual Samples** (artifacts).
สำหรับรูปภาพ เราต้องติดตามทั้ง Metadata และตัวอย่างรูปภาพ เพื่อยืนยันความถูกต้อง



The Scenario: Geometric Shapes Dataset



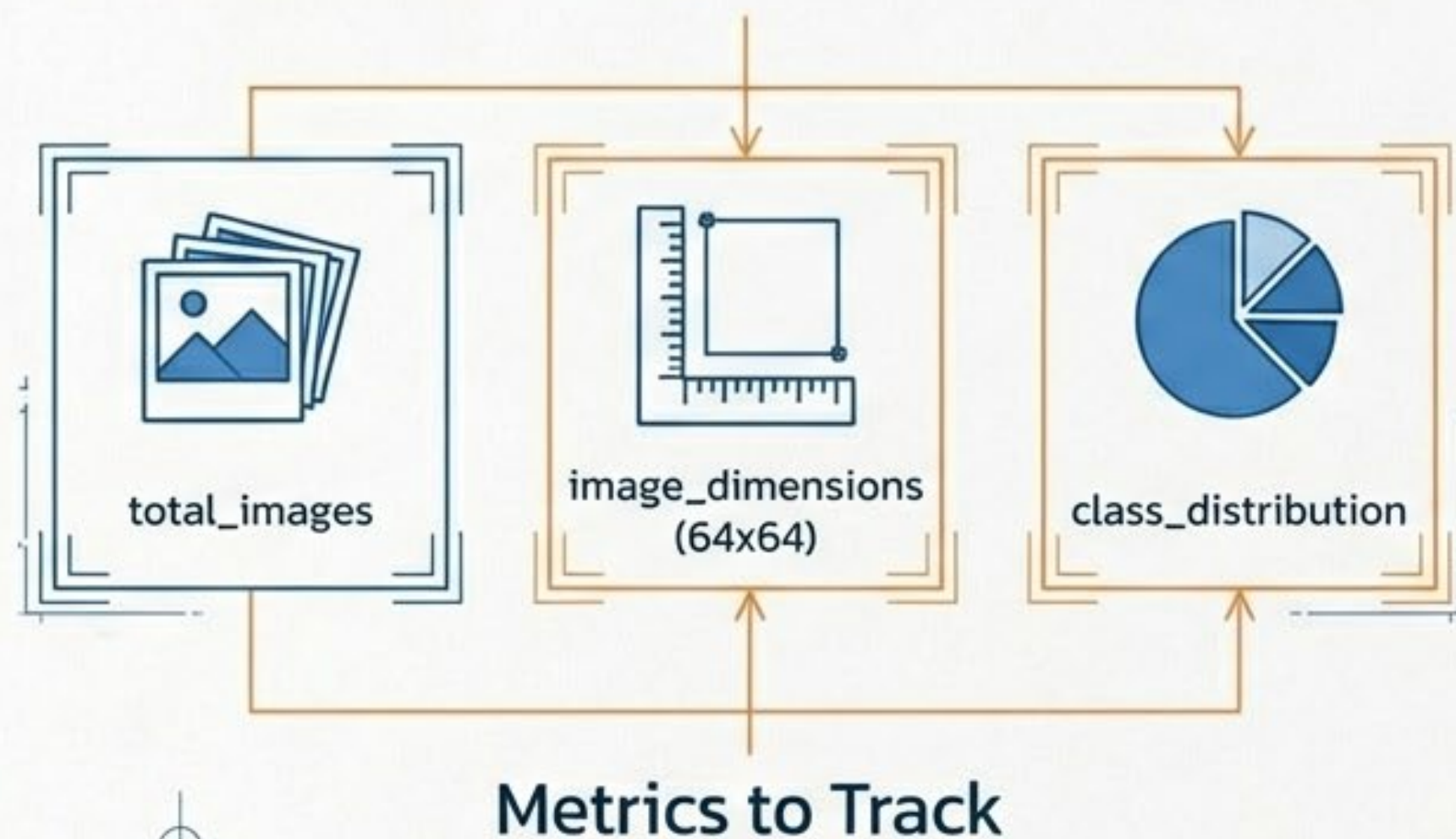
Goal: จำแนกประเภทรูปภาพทรงเรขาคณิต
(Image Classification)

Structure: จัดเก็บแบบ Folder-based

Classes: 3 Classes (Circle, Square, Triangle)

Strategy 2: การ Track ข้อมูลรูปภาพ (Unstructured Data)

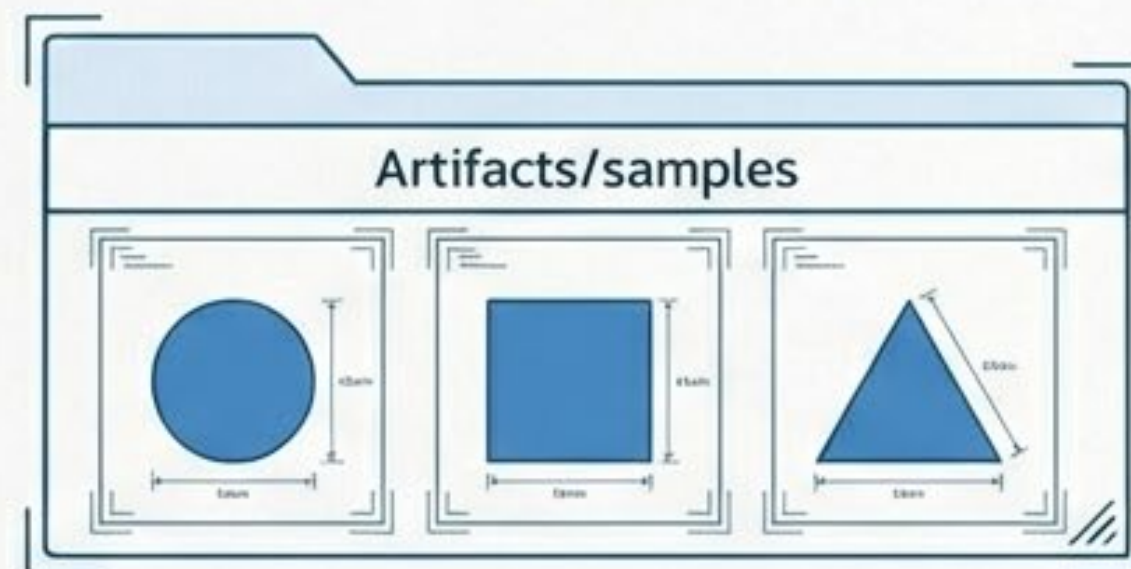
Challenge: ไม่สามารถเก็บ Pixel ทั้งหมดได้
ต้องเน้นที่ Metadata




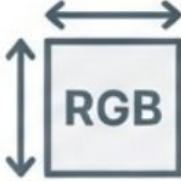


```
MLFlow Logging Code

# Log metadata
mlflow.log_metric("total_images", stats['total_images'])

# Log sample artifacts for preview
mlflow.log_artifact(img_path, artifact_path="samples/circle")
```



Baseline: Version 1.0.0 Specifications

Technical Spec Sheet			
	Total Images: 90 images (30 per class)		Image Size: 64x64 pixels (RGB)
	Format: PNG		Source: Synthetic Generation (create_shape_image)

นี่คือจุดเริ่มต้นของ Dataset เราจะทำการ Log สถานะของ Version 1 เพื่อใช้เป็น Baseline สำหรับการเปรียบเทียบในอนาคต

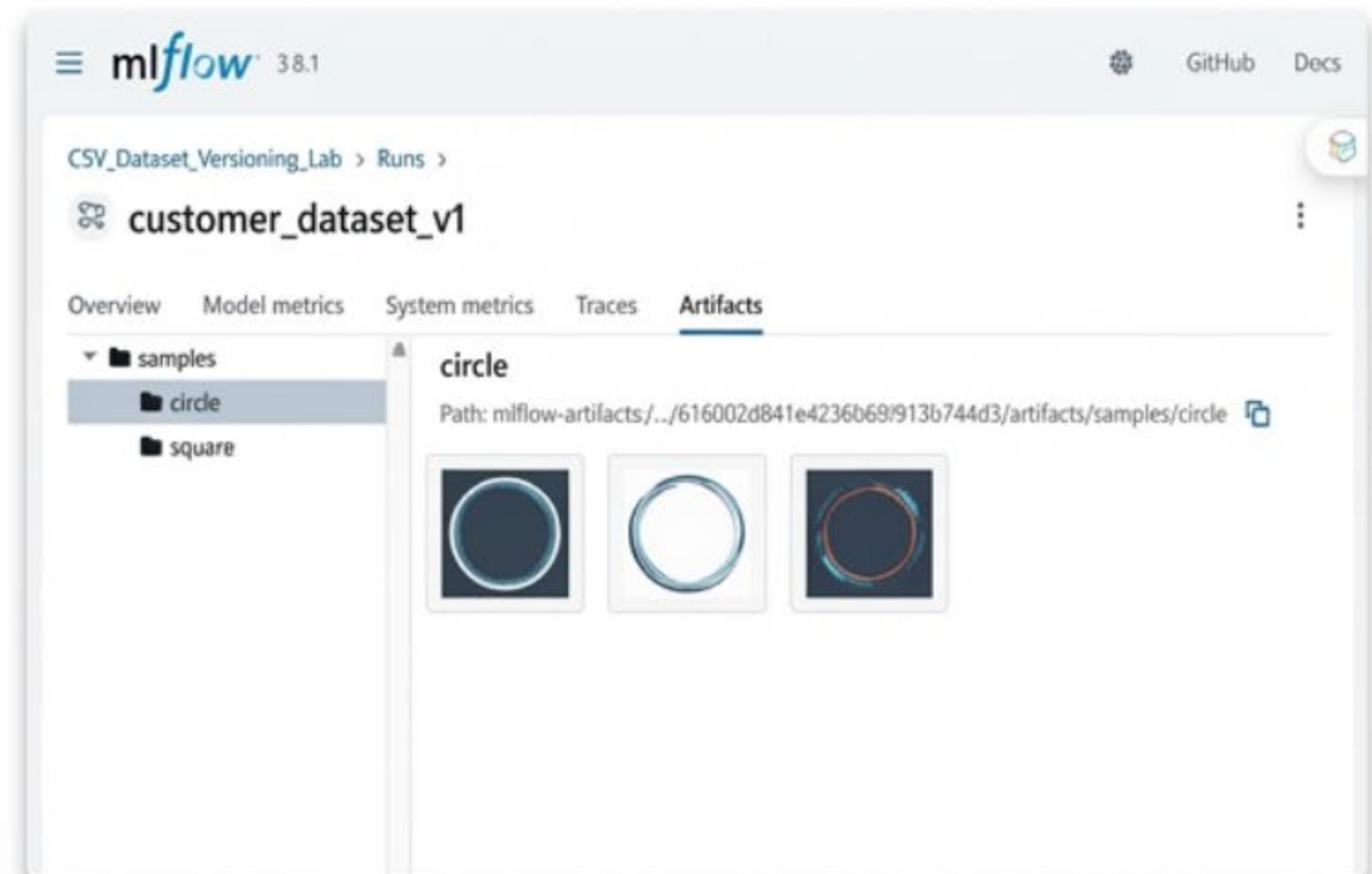
```
dataset_logging.py x
dataset_logging.py > ...
10
12 # Log parameters to define the dataset properties
13 mlflow.log_param("dataset_version", "1.0.0")
14 mlflow.log_param("dataset_name", "geometric_shapes")
15 mlflow.log_param("image_format", "PNG")
16 mlflow.log_param("image_size", "64x64")
17 mlflow.log_param("num_classes", 3)
18 mlflow.log_param("num_classes", 3)
19 mlflow.log_param("class_names", ["circle", "square", "triangle"])
```

การใช้ `mlflow.log_param`
ช่วยให้เราค้นหาและ filter dataset
ตามคุณสมบัติได้ในภายหลัง เช่น ค้นหา
dataset ทั้งหมดที่เป็นขนาด 64x64



```
# Log sample images (Only 3 samples per class)
for class_name in stats['classes'].keys():
    class_path = os.path.join(image_dir_v1, class_name)
    # Select first 3 images
    sample_images = os.listdir(class_path)[:3]

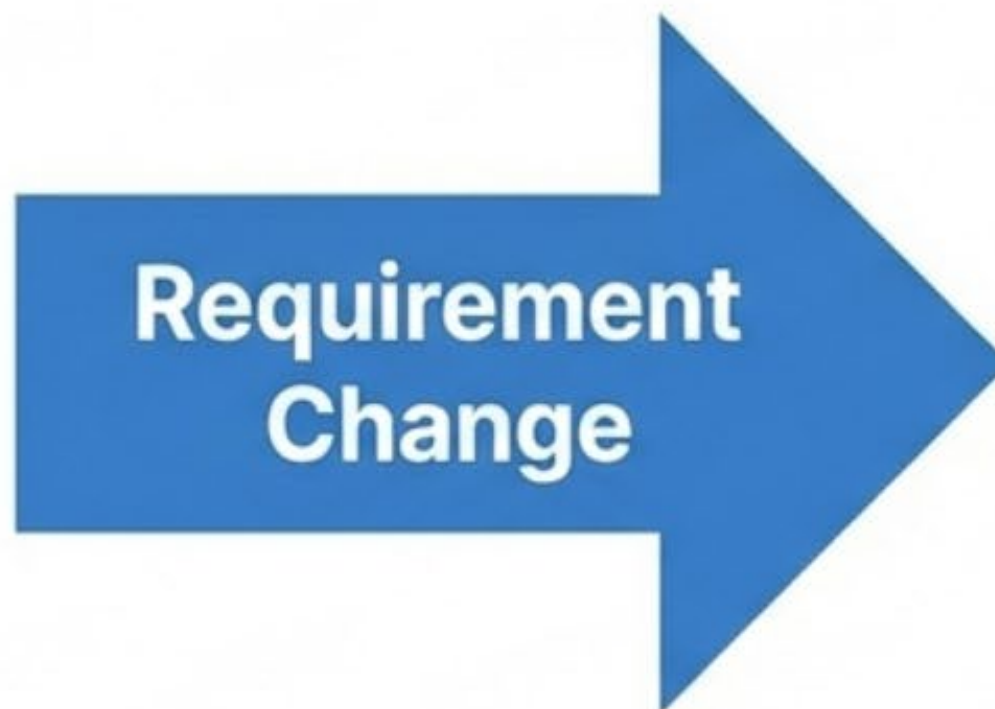
    for img_name in sample_images:
        img_path = os.path.join(class_path, img_name)
        mlflow.log_artifact(img_path, artifact_path=
            f"samples/{class_name}")
```



เก็บ Sample ไว้ใน Artifacts เพื่อให้ Data Scientist คนอื่น Preview ข้อมูลได้ทันที



90 images



150 images

Requirement: เพิ่มจำนวนรูปภาพ (Increase image count)
Change: จาก 30 รูป/class → 50 รูป/class
Total: 90 images → 150 images

```
: # Create Image Dataset Version 1 (smaller dataset)
print("🖼️ Creating Image Dataset Version 1...")
image_info_v1, image_dir_v1 = create_image_dataset(version=1, num_per_class=30)
print(f"    Total images: {image_info_v1['total_images']}")
print(f"    Classes: {image_info_v1['classes']}")
print(f"    Location: {image_dir_v1}")

# Create Image Dataset Version 2 (larger dataset)
print("\n🖼️ Creating Image Dataset Version 2...")
image_info_v2, image_dir_v2 = create_image_dataset(version=2, num_per_class=50)
print(f"    Total images: {image_info_v2['total_images']}")
print(f"    Classes: {image_info_v2['classes']}")
print(f"    Location: {image_dir_v2}")
```

```
🖼️ Creating Image Dataset Version 1...
Total images: 90
Classes: ['circle', 'square', 'triangle']
Location: ./mlflow_datasets_lab/images/v1
```

```
🖼️ Creating Image Dataset Version 2...
Total images: 150
Classes: ['circle', 'square', 'triangle']
Location: ./mlflow_datasets_lab/images/v2
```

```

def create_shape_image(shape, size=64, color=None):
    """Create an image with a specific shape."""
    if color is None:
        color = tuple(np.random.randint(100, 255, 3))

    # Create a white background
    img = Image.new('RGB', (size, size), color=(255, 255, 255))
    pixels = img.load()

    center = size // 2
    radius = size // 3

    if shape == 'circle':
        for x in range(size):
            for y in range(size):
                if (x - center) ** 2 + (y - center) ** 2 <= radius ** 2:
                    pixels[x, y] = color

    elif shape == 'square':
        for x in range(center - radius, center + radius):
            for y in range(center - radius, center + radius):
                if 0 <= x < size and 0 <= y < size:
                    pixels[x, y] = color

    elif shape == 'triangle':
        for y in range(center - radius, center + radius):
            width = int((y - (center - radius)) * radius / radius)
            for x in range(center - width, center + width):
                if 0 <= x < size and 0 <= y < size:
                    pixels[x, y] = color

    return img

```

```

def create_image_dataset(version, num_per_class=50):
    """Create a complete image dataset with multiple classes."""
    shapes = ['circle', 'square', 'triangle']
    dataset_info = {
        'version': version,
        'classes': shapes,
        'images_per_class': num_per_class,
        'total_images': num_per_class * len(shapes),
        'image_size': 64,
        'files': []
    }

    version_dir = os.path.join(IMAGE_DIR, f"v{version}")
    os.makedirs(version_dir, exist_ok=True)

    for shape in shapes:
        class_dir = os.path.join(version_dir, shape)
        os.makedirs(class_dir, exist_ok=True)

        for i in range(num_per_class):
            img = create_shape_image(shape)
            filename = f"{shape}_{i:04d}.png"
            filepath = os.path.join(class_dir, filename)
            img.save(filepath)
            dataset_info['files'].append({
                'path': filepath,
                'class': shape,
                'filename': filename
            })

    return dataset_info, version_dir

```

Image_Dataset_Versioning_Lab > Runs >

 shapes_dataset_v1[Overview](#) [Model metrics](#) [System metrics](#) [Traces](#) [Artifacts](#)Description 

No description

Metrics (5)

 Search metrics





Metric	Value
total_images	90
total_size_mb	0.0041522979736328125
class_circle_count	30
class_square_count	30
class_triangle_count	30

Parameters (7)

 Search parameters

Parameter	Value
dataset_version	1.0.0
dataset_name	geometric_shapes
image_format	PNG
image_size	64x64
num_classes	3
class_names	['circle', 'square', 'triangle']
creation_date	2026-02-03T13:14:49.573461

About this run

Created at	02/03/2026, 08:14:49 PM
Created by	root
Experiment ID	1 
Status	 Finished
Run ID	0556a10c6bc3423da39d0c0634a0999d 
Duration	345ms
Source	 ipykernel_launcher.py
Logged models	
Registered prompts	—

Datasets

None

Tags

[Add tags](#)

Registered models

None

shapes_dataset_v1

Overview Model metrics System metrics Traces **Artifacts**

▼ metadata

dataset_info.json

▼ samples

▼ circle

circle_0000.png

circle_0001.png

circle_0002.png

▶ square

▶ triangle

metadata/dataset_info.json 13.39KB

Path: mlflow-artifacts:/1/0556a10c6bc3423da39d0c0634a0999d/artifacts/metadata/dataset_info.json

```
{
  "version": 1,
  "classes": [
    "circle",
    "square",
    "triangle"
  ],
  "images_per_class": 30,
  "total_images": 90,
  "image_size": 64,
  "files": [
    {
      "path": "./mlflow_datasets_lab/images/v1/circle/circle_0000.png",
      "class": "circle",
      "filename": "circle_0000.png"
    },
    {
      "path": "./mlflow_datasets_lab/images/v1/circle/circle_0001.png",
      "class": "circle",
      "filename": "circle_0001.png"
    },
    {
      "path": "./mlflow_datasets_lab/images/v1/circle/circle_0002.png",
      "class": "circle",
      "filename": "circle_0002.png"
    },
    {
      "path": "./mlflow_datasets_lab/images/v1/circle/circle_0003.png",
      "class": "circle",
      "filename": "circle_0003.png"
    },
  ],
}
```

```
# Log Image Dataset Version 2
print("📁 Logging Image Dataset Version 2...")

with mlflow.start_run(run_name="shapes_dataset_v2") as run:
    # Calculate statistics
    stats = get_image_dataset_stats(image_dir_v2)

    # Log parameters
    mlflow.log_param("dataset_version", "2.0.0")
    mlflow.log_param("dataset_name", "geometric_shapes")
    mlflow.log_param("image_format", "PNG")
    mlflow.log_param("image_size", "64x64")
    mlflow.log_param("num_classes", len(stats['classes']))
    mlflow.log_param("class_names", list(stats['classes'].keys()))
    mlflow.log_param("creation_date", datetime.now().isoformat())
    mlflow.log_param("changes_from_v1", "Increased samples per class")

    # Log metrics
    mlflow.log_metric("total_images", stats['total_images'])
    mlflow.log_metric("total_size_mb", stats['total_size_mb'])

    for class_name, count in stats['classes'].items():
        mlflow.log_metric(f"class_{class_name}_count", count)

    # Log sample images
    for class_name in stats['classes'].keys():
        class_path = os.path.join(image_dir_v2, class_name)
        sample_images = os.listdir(class_path)[:3]

        for img_name in sample_images:
            img_path = os.path.join(class_path, img_name)
            mlflow.log_artifact(img_path, artifact_path=f"samples/{class_name}")

    # Log dataset info
    info_path = os.path.join(image_dir_v2, "dataset_info.json")
    with open(info_path, 'w') as f:
        json.dump({
            **image_info_v2,
            'stats': {k: v for k, v in stats.items() if k != 'image_dimensions'}
        }, f, indent=2, default=str)
    mlflow.log_artifact(info_path, artifact_path="metadata")

run_id_img_v2 = run.info.run_id
print(f"✅ Image Dataset v2 logged!")
print(f"    Run ID: {run_id_img_v2}")
print(f"    Total Images: {stats['total_images']}")

📁 Logging Image Dataset Version 2...
✅ Image Dataset v2 logged!
Run ID: 7f142e9224e549ac962d56812b251560
Total Images: 150
🔗 View run shapes_dataset_v2 at: http://localhost:5000/#/experiments/1/runs/7f142e9224e549ac962d56812b251560
🌿 View experiment at: http://localhost:5000/#/experiments/1
```

mlflow3.8.1

Image_Dataset_Versioning_Lab > Runs > shapes_dataset_v2

OverviewModel metricsSystem metricsTracesArtifacts

Description✎

No description

Metrics (5)

🔍 Search metrics

Metric	Value
total_images	150
total_size_mb	0.0041446685791015625
class_circle_count	50
class_square_count	50
class_triangle_count	50

Parameters (8)

🔍 Search parameters

Parameter	Value
dataset_version	2.0.0
dataset_name	geometric_shapes
image_format	PNG
image_size	64x64
num_classes	3
class_names	['circle', 'square', 'triangle']
creation_date	2026-02-03T15:14:12.206196
changes_from_v1	Increased samples per class

Dataset Versioning for Json

```
categories = ['sports', 'technology', 'politics', 'entertainment']

sample_texts = {
  'sports': [
    "The team won the championship game last night",
    "Athletes prepare for the upcoming Olympics",
    "Football season starts next month",
    "The basketball player scored 30 points",
    "Tennis tournament reaches final stages"
  ],
  'technology': [
    "New smartphone features advanced AI capabilities",
    "Tech company releases latest software update",
    "Scientists develop revolutionary battery technology",
    "Cloud computing transforms business operations",
    "Artificial intelligence continues to evolve"
  ],
  'politics': [
    "Government announces new economic policies",
    "Elections scheduled for next year",
    "International summit addresses climate change",
    "New legislation passes in parliament",
    "Diplomatic relations improve between nations"
  ],
  'entertainment': [
    "Movie breaks box office records",
    "Popular singer announces world tour",
    "Streaming service adds new content",
    "Award show celebrates best performances",
    "New video game receives critical acclaim"
  ]
}
```

กลยุทธ์การวัดผล JSON



เนื่องจากข้อมูลเป็น Semi-structured เราต้องนิยามตัวชี้วัด (Metrics) ที่สำคัญก่อนทำการ Log

```

categories = ['sports', 'technology', 'politics', 'entertainment']

sample_texts = {
    'sports': [
        "The team won the championship game last night",
        "Athletes prepare for the upcoming Olympics",
        "Football season starts next month",
        "The basketball player scored 30 points",
        "Tennis tournament reaches final stages"
    ],
    'technology': [
        "New smartphone features advanced AI capabilities",
        "Tech company releases latest software update",
        "Scientists develop revolutionary battery technology",
        "Cloud computing transforms business operations",
        "Artificial intelligence continues to evolve"
    ],
    'politics': [
        "Government announces new economic policies",
        "Elections scheduled for next year",
        "International summit addresses climate change",
        "New legislation passes in parliament",
        "Diplomatic relations improve between nations"
    ],
    'entertainment': [
        "Movie breaks box office records",
        "Popular singer announces world tour",
        "Streaming service adds new content",
        "Award show celebrates best performances",
        "New video game receives critical acclaim"
    ]
}

```

```

# Create experiment for JSON datasets
EXPERIMENT_NAME_JSON = "JSON_Dataset_Versioning_Lab"
mlflow.set_experiment(EXPERIMENT_NAME_JSON)

```

```

print(f"📁 Experiment: {EXPERIMENT_NAME_JSON}")

```

```

# Log JSON Dataset Version 1
print("📄 Logging JSON Dataset Version 1...")

```

```

with mlflow.start_run(run_name="text_classification_v1") as run:
    # Load and analyze
    with open(json_path_v1, 'r') as f:
        data = json.load(f)

    stats = get_json_dataset_stats(data)

    # Log parameters
    mlflow.log_param("dataset_version", "1.0.0")
    mlflow.log_param("dataset_name", "text_classification")
    mlflow.log_param("data_format", "JSON")
    mlflow.log_param("task_type", "multi-class classification")
    mlflow.log_param("num_classes", stats['unique_labels'])
    mlflow.log_param("creation_date", datetime.now().isoformat())

    # Log metrics
    mlflow.log_metric("total_samples", stats['total_samples'])
    mlflow.log_metric("avg_text_length", stats['avg_text_length'])
    mlflow.log_metric("num_classes", stats['unique_labels'])

    for label, count in stats['label_distribution'].items():
        mlflow.log_metric(f"label_{label}_count", count)

    # Log artifact
    mlflow.log_artifact(json_path_v1, artifact_path="datasets")

    # Log file hash
    file_hash = calculate_file_hash(json_path_v1)
    mlflow.log_param("file_hash_md5", file_hash)

    print(f"✅ JSON Dataset v1 logged!")
    print(f"Run ID: {run.info.run_id}")

```

บทสรุป (Conclusion)

- ➔ - **Dataset Versioning** ไม่ใช่ทางเลือก แต่เป็น 'สิ่งจำเป็น' สำหรับ Production
- ➔ - **MLflow** เปลี่ยน Data Chaos ให้เป็น Data Science



เริ่มต้น Track ข้อมูลของคุณวันนี้
เพื่อป้องกันความล้มเหลวในวันหน้า