# AI 資料競賽平台 SA/SD

# Data Repository

## 流程

1. 手動上傳資料集 ( dataset ) 與 資料處理設定檔 ( metadata.yaml ) 到 Server。
    a. 例: cp dataset_A /tmp/dataset_A

2. 執行程式 ( add.py ) 並附加資料集目錄名稱。
    a. 例: python3 add.py “/tmp/rawData/dataset_A”

3. 程式 ( add.py ) 檢查資料處理設定檔 ( metadata.yaml ) 是否正確。
    a. metadata.yaml 是否存在。
    b. metadata.yaml 是否可解析。
    c. metadata.yaml 是否有所需欄位。
    d. metadata.yaml 的 dataset name 是否在 MySQL 設定資料表裡已經存在。

4. 若 程式 ( add.py ) 檢查 dataset name 在 MySQL 尚未存在，則：
    a. 程式 ( add.py ) 根據 設定檔 ( metadata.yaml ) 在 資料庫 MySQL 新增 設定資料。
    b. 程式 ( add.py ) 新增目錄，名稱為 ${pwd}/history/rawData/${dataset name}
    c. 程式 ( add.py ) 複製資料集到 ${pwd}/history/rawData/${dataset name}
    例: cp -rf /tmp/dataset_A ${pwd}/history/rawData/dataset_A
    d. 程式 ( add.py ) 根據資料集在 MySQL 產生資料表。

## 設定檔 metadata.yaml 格式

```
dataset:
    description: ${ String }
    name: ${ String }
    creater: ${ String [name] }
    resource:
      - type: ${ Enum: train, test, validation, result }
        name: ${ String
        files:
          - ${ String  [full path] }
        folders:
          - ${ String  [full path] }
        Features:
          ${ String: String [f_101: double] }

** PS
**** Table Name = dataset name + “_” + resource name
**** Add full path to the features.
```

# Database Schema

dataset

| Name | Type | |
|---|---|---|
| ID | Int | PK |
| name | varchar(255) | |
| description | text | |
| state | Enum (creating, deleting, stable) | |
| metadata | JSON | |

resource

| Name | Type | |
|---|---|---|
| ID | Int | PK |
| dataset_id | Int | FK |
| type | Enum ( training, testing, validation, result ) | |
| creater | varchar(125) | |
| table_name | varchar(255) | |
| format | varchar(125) | |

merge_info

| Name | Type | |
|---|---|---|
| ID | Int | PK |
| resource_id | Int | FK |
| block_head | int | FK |
| block_size | int | |

datasetName_resourceName

| Name | Type | |
|---|---|---|
| ID | Int | PK |
| feature_1 | float | |

| | | |
|---|---|---|
| feature_2 | float | |
| full_path | varchart(512) | |

# Data Challenge

## 流程

1. 手動上傳 split.yaml
2. 執行程式 ( commit.py ) 並傳入設定檔路徑名稱。
   ex: python3 commit.py split.yaml
3. 程式 ( commit.py ) 檢查 split.yaml
   a. 格式是否正確 ?
   b. Dataset name 與 reasource name 是否在資料庫存在 ?
   c. 此次要對 resource 做 split 的 tag 是否已經存在 ?
4. 程式 ( commit.py ) 根據設定檔 ( split.yaml ) 對資料表做 Shuffle 與 Split, 產生新表。

## 設定檔 split.yaml 格式

```
dataset:
    name: ${ String }
    creater: ${ String }
    resource:
        name: ${ String }
        tag: ${ String }
        blockSplit:
            shuffle: ${ Boolean }
            split:
                training: ${ int }
                testing: ${ int }

** PS
**** Table Name = dataset name + "_" + resource name + "_" + tag
```

## Tables Schema

split

| Name | Type | |
|---|---|---|
| ID | Int | PK |
| resource_id | Int | FK |
| tag | varchar(255) | |
| type | Enum ( training, testing, validation, result ) | |
| creater | varchar(125) | |

| | | |
|---|---|---|
| table_name | varchar(255) | |
| format | varchar(125) | |

Unique Key: resource_id + tag

datasetName_resourceName_tag_type

| Name | Type | |
|---|---|---|
| ID | Int | PK |
| feature_1 | float | |
| feature_2 | float | |
| full_path | varchart(512) | |