## 개요

version --- app.py : runner

--- api :

--- ai

--- ai\_utils

--- ai\_model files

--- sql

--- sql & rd handler

--- statistics

--- files related to calculate the statsitics

--- flask

--- flask\_utils.py

--- flask\_post.pys

--- templates : html storage

--- customers : user space

--- static : images

--- app.py

--- app\_funcs.py

--- app\_downloader.py

--- continue.py

--- forms.py

--- requirements.txt

core process info : [주요 Story 연결고리](https://dshare.atlassian.net/wiki/spaces/~821813289/pages/753336490)

# Flask

/train\_page

→ input : dataset\_name, model\_type, column\_list, src\_type(redis,view,csv)

/retrain\_page

→ input: dataset\_name, model\_type, src\_type(redis,view,csv), model\_filename

/predict\_page

→ input: dataset\_name, model\_type, src\_type(redis,view,csv), model\_filename

/write\_history

→ input : text log

– 주의

* train\_page의 column\_list는 foramt 반드시 유지해야 함 [c1,c2,c3..]
* hyper\_params 는 json format

# DB

DB\_tables

* Engine\_live - for engine information for engine\_blank
  + id
  + model
  + text
* model\_name - information for generated models
  + id
  + type(계열)
  + name

* log\_history - information for recording what user did
  + id
  + uid
  + log

* Result - information for results of model training/predict
  + no
  + user\_id
  + stat\_dt
  + end\_dt
  + hyper\_parameter
  + model\_filename
  + input
  + pred
  + eval
  + loss

* login - information for user id
  + name

* cols\_view
  + id
  + type - csv, redis, view\_csv
  + name
  + cols
  + uid

# redis

* format
  + hash type : key-field-contents
    - (table\_name, row\_id, {col1:val1, col2:val2, col3:val3… })

# html

#### api.html

1. data\_blank

* Dataset load : 3 types

: csv files in customers directory

– data-type : csv

– name : dataset\_name

– data-cols : 열1,열2,열3..

: data from redis

– data-type : redis

– name : dataset\_name

– data-cols : 열1,열2,열3..

: view derived from csv

– data-type : view\_csv

– name : dataset\_name

– data-cols : 열1,열2,열3..

* for resnet engine, at least 4 columns needed.

2. engine blank

* train engine
  + 1. get data from Engine\_live db tables
  + 2. train and retrain for ai engine
  + class : resnet\_train / decision\_train
  + name : train/retrain

3. model\_file blank

* model file name
  + get data from model\_name db tables
  + id : modeltype-user\_id-datetime-format
    - ex) [DT\_test0\_21-05-10 16-15.0.pkl](http://127.0.0.1:8080/model_downloads/DT_test0_21-05-10 16-15.0.pkl)
  + class : decision\_model / resnet\_model

4. preds with metrics

* prediction file name
  + save in customer directory and Result db tables at the same time
  + id : type\_dataset\_preds\_by\_modelname\_time\_foramt
    - ex) csv\_reserve.csv\_[preds\_by\_DT\_test0\_21-05-10 16-15.0.pkl\_time\_21-05-10 16-16](http://127.0.0.1:8080/pred_downloads/preds_by_DT_test0_21-05-10 16-15.0.pkl_time_21-05-10 16-16)

#### successive\_excel.html

deal with preprocessing datas

* cannot edit the original column name
* types : redis, csv
* add, delete the columns/rows
* edit the data
* if type is csv → activate make view button

#### view.html

only see the existing data

* calculate the statistic
* see the representative data for each column
* show the chart

#### new\_excel.html

make entirely new data

* save to redis
* save to csv(directory)

#### home.html

1. deal with login and register

#### progresbar.html

1. trace the epoch and batch iteration

# core python files

0. app.py

* main page
* handle api.html and trigger download or load

1. ai\_post.py

* handles train/predict process

2. excel\_post.py

* handles process in related to edit/generate/get data
  + redis
  + csv
  + view\_csv

3. live\_engine\_post.py

* handles the process of save/load the former used engine

4. log\_post.py

* record the what user did
* notice the success/failure of the process

5. parsers.py

* make different data type compatible
  + redis-html
  + csv-html
  + csv-redis

6. view\_post

* caluculate the statistics of each column
* plot chart
* extract view from original csv
* show the view

\*\* ai\_utils.py

* define\_argparser : for subprocess run
* Base\_insert : basic structure for insert into DB
* process\_cols : change the column\_list into appropriate format
* process\_dt\_data process\_torch\_data : normalize the data
* Base\_subprocess : basic structure for train/predict

\*\* how to construct model runner if u add model

subprocess(self, config, modelname, now, root\_path, mode='train', neural\_net=True, column\_list=None, src\_type='csv',num\_cols=None)