**TrainDB-ml 설치 방법**

**1. 환경 설정**

1. conda create -n TrainDB-ml python=3.7
2. conda activate TrainDB-ml
3. git clone [https://github.com/traindb-project/TrainDB-ml.git](https://github.com/traindb-project/traindb-ml.git)
4. cd TrainDB-ml
5. pip install -e .
6. pip install mlflow
7. pip install flask
8. pip install psycopg2
9. pip uninstall gensim
10. pip install gensim==3.8.3
11. Refer to [https://github.com/traindb-project/TrainDB-ml/blob/main/install-script/install\_TrainDB-ml.sh](https://github.com/traindb-project/traindb-ml/blob/main/install-script/install_traindb-ml.sh)

**2. Restful API service testing**

**2-1. TrainDB-ml 구동**

1. **cd server**

**TrainDB-ml/server 폴더 구성**

|  |
| --- |
| **vm:~/TrainDB-ml/server$ ls -l**  **합계 24**  **-rw-rw-r-- 1 jstcto jstcto 4513 11월 30 09:33 callee.py**  **-rw-rw-r-- 1 jstcto jstcto 5472 12월 1 15:43 caller.py**  **drwxrwxr-x 2 jstcto jstcto 4096 12월 1 15:43 etrimlwarehouse**  **drwxrwxr-x 4 jstcto jstcto 4096 11월 11 12:31 mlruns** |
| * **callee.py : 사용자 명령을 수신하여 TrainDB-ml engine에 전달하고, 명령 결과를 사용자로 전달해주는 웹서버** * **caller.py : 사용자 명령 발생기 예제 프로그램** * **etrimlwarehouse : TrainDB-ml engine에서 생성한 모델을 저장하는 폴더** * **mlruns : TrainDB-ml engine에서 생성한 모델을 mlflow framework용 모델을 저장하는 폴더** |

1. **python callee.py**

**callee.py server 사용법**

|  |
| --- |
| **vm:~/TrainDB-ml/server$ python callee.py -h**  **usage: callee.py [-h] [-p PORT] [-t TARGET]**  **optional arguments:**  **-h, --help show this help message and exit**  **-p PORT, -port PORT {port number (defalut=1234}**  **-t TARGET, -target TARGET {target model (dw=datawarehouse | mf=mlflow) | default=mf}** |
| * **-p, -port : 웹서버용 수신 포트 번호, 명령인자가 전달되지 않으면 기본으로 포트번호는 1234으로 동작** * **-t, -traget : 웹서버가 datawarehouse(“dw”입력)용으로 동작할지, mlflow(“mf”입력)용으로 동작할지를 결정, 명령인자가 전달되지 않으면 기본으로 mlflow용으로 동작** |

**callee.py server 실행**

|  |
| --- |
| **vm:~/TrainDB-ml/server$ python callee.py**  **Local mode is on, as no slaves are provided.**  **start loading pre-existing models.**  **Loaded 5 models. time cost 0.024575 s**  **Running on http://0.0.0.0:1234 -> mlflow**  **\* Serving Flask app 'callee' (lazy loading)**  **\* Environment: production**  **WARNING: This is a development server. Do not use it in a production deployment.**  **Use a production WSGI server instead.**  **\* Debug mode: off**  **\* Running on all addresses.**  **WARNING: This is a development server. Do not use it in a production deployment.**  **\* Running on http://10.0.2.15:1234/ (Press CTRL+C to quit)** |
|  |

**2-2. 모델 생성(create model): caller.py 소스를 아래와 같이 변경**

**Caller.py client 소스 수정**

|  |
| --- |
| **sql\_string = "create model instacart\_order\_product\_600k (add\_to\_cart\_order real, reordered real) from instacart\_order\_product\_600k.csv method uniform size 1000"** |
| * **Caller.py의 소스중 위의 명령을 주석 해제한다.** * **SQL 문을 문자열로 전달하려면, '-sql' 옵션 후, SQL 문자열을 포함**   **[예제] python caller.py -sql "create model instacart\_order\_product\_600k (add\_to\_cart\_order real, reordered real) from instacart\_order\_product\_600k.csv method uniform size 1000"** |

**2-3. caller.py 실행**

**caller client 사용법**

|  |
| --- |
| **vm:~/TrainDB-ml/server$ python caller.py -h**  **usage: caller.py [-h] [-a HOST] [-p PORT]**  **optional arguments:**  **-h, --help show this help message and exit**  **-a HOST, -host HOST {host ip (defalut=127.0.0.1)}**  **-p PORT, -port PORT {port number (defalut=1234)}** |
| * **-a, -host : 접속할 TrainDB-ml 웹서버 IP 주소, 명령인자가 전달되지 않으면 기본으로 127.0.0.1 접속** * **-p, -port : 접속할 TrainDB-ml 웹서버 포트 번호, 명령인자가 전달되지 않으면 기본으로 포트번호는 1234** |

caller client 실행

|  |
| --- |
| **vm:~/TrainDB-ml/server$ python caller.py**  **======================================================================**  **TO etriml http://127.0.0.1:1234 -> success**  **REQUEST(SQL) >>> create model instacart\_order\_products\_1g\_3000000 (add\_to\_cart\_order real, reordered real) from instacart\_order\_products\_1g.csv method uniform size 3000000**  **RESPONSE(JSON) <<< {'cause': 'success', 'cost': 4707, 'result': 'success'}**  **----------------------------------------------------------------------**  **time cost : 4707.0000 seconds**  **----------------------------------------------------------------------**  **Total elapsed time : 4707.0972 seconds**  **======================================================================** |

**Callee server 실행 결과**

|  |
| --- |
| **======================================================================**  **REQUEST(SQL) >>> create table instacart\_order\_products\_1g\_3000000 (add\_to\_cart\_order real, reordered real) from instacart\_order\_products\_1g.csv method uniform size 3000000**  **----------------------------------------------------------------------**  **Start creating model instacart\_order\_products\_1g\_3000000**  **Reading data file...**  **# usecols: {'y': ['add\_to\_cart\_order', 'real', None], 'x\_continous': ['reordered'], 'x\_categorical': [], 'gb': None}**  **XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX create begin mlflow**  **get frequency info from data....**  **Starting training kde models for model instacart\_order\_products\_1g\_3000000**  **training regression...**  **\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***  **finish training embedding.**  **embedding inference...**  **start normalizing data...**  **transform data from MDN training...**  **finish transforming data from MDN training...**  **< Epoch 0**  **< Epoch 1**  **< Epoch 2**  **< Epoch 3**  **< Epoch 4**  **< Epoch 5**  **< Epoch 6**  **< Epoch 7**  **< Epoch 8**  **< Epoch 9**  **< Epoch 10**  **< Epoch 11**  **< Epoch 12**  **< Epoch 13**  **< Epoch 14**  **< Epoch 15**  **< Epoch 16**  **< Epoch 17**  **< Epoch 18**  **< Epoch 19**  **Finish regression training.**  **training density...**  **\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***  **finish training embedding.**  **< Epoch 0**  **< Epoch 1**  **< Epoch 2**  **< Epoch 3**  **< Epoch 4**  **< Epoch 5**  **< Epoch 6**  **< Epoch 7**  **< Epoch 8**  **< Epoch 9**  **< Epoch 10**  **< Epoch 11**  **< Epoch 12**  **< Epoch 13**  **< Epoch 14**  **< Epoch 15**  **< Epoch 16**  **< Epoch 17**  **< Epoch 18**  **< Epoch 19**  **finish mdn training...**  **XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX create end mlflow**  **time cost: 4707s.**  **\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***  **\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***  **>>> MLFLOW RUN\_ID : 6f57adee769040c38795a52c380b941f <<<**  **\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***  **\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***  **======================================================================**  **RESPONSE(SQL) >>> {"cause": "success", "cost": 4707, "result": "success"}**  **======================================================================**  **127.0.0.1 - - [01/Dec/2021 10:58:34] "POST / HTTP/1.1" 200 -** |
| * **model 생성 완료되면, mlruns/0/ 디렉토리 아래 mlflow용 모델이 생성됨** * **예: ./mlruns/0/6f57adee769040c38795a52c380b941f/** * **여기서, 6f57adee769040c38795a52c380b941f --> runid** |

**2-4 모델 서빙 (model serving)**

**model serving 명령 및 출력**

|  |
| --- |
| **vm:~/TrainDB-ml/server$ mlflow models serve -m ./mlruns/0/6f57adee769040c38795a52c380b941f/artifacts/model -h 0.0.0.0 -p 8003 --no-conda**  **2021/12/01 11:46:26 INFO mlflow.models.cli: Selected backend for flavor 'python\_function'**  **2021/12/01 11:46:26 INFO mlflow.pyfunc.backend: === Running command 'gunicorn --timeout=60 -b 0.0.0.0:8003 -w 1 ${GUNICORN\_CMD\_ARGS} -- mlflow.pyfunc.scoring\_server.wsgi:app'**  **[2021-12-01 11:46:26 +0900] [2245999] [INFO] Starting gunicorn 20.1.0**  **[2021-12-01 11:46:26 +0900] [2245999] [INFO] Listening at: http://0.0.0.0:8003 (2245999)**  **[2021-12-01 11:46:26 +0900] [2245999] [INFO] Using worker: sync**  **[2021-12-01 11:46:26 +0900] [2246051] [INFO] Booting worker with pid: 2246051** |
| * **mlflow models serve -m ~/mlruns/0/{runid}/artifacts/model -h 0.0.0.0 -p 8003 –no-conda** * **[예제] runid가 '6f57adee769040c38795a52c380b941f'인 경우**   **$ mlflow models serve -m ./mlruns/0/6f57adee769040c38795a52c380b941f/artifacts/model -h 0.0.0.0 -p 8003 --no-conda** |

**2-5. select model: 소스를 아래와 같이 변경 (아래 2-4단계. mlflow model serving 이후 실행)**

**caller client 소스 수정**

|  |
| --- |
| **sql\_string = "select model avg (add\_to\_cart\_order real) from instacart\_order\_product\_600k where 0.9 <= reordered <= 1.0"** |
| * **Caller.py의 소스중 위의 명령을 주석 해제한다.** * **SQL 문을 문자열로 전달하려면, '-sql' 옵션 후, SQL 문자열을 포함**   **[예제] python caller.py -sql "select model avg (add\_to\_cart\_order real) from instacart\_order\_product\_600k where 0.9 <= reordered <= 1.0"** |

**Caller client 출력**

|  |
| --- |
| **vm:~/TrainDB-ml/server$ python caller.py**  **======================================================================**  **TO etriml http://127.0.0.1:1234 -> success**  **REQUEST(SQL) >>> select model avg (add\_to\_cart\_order real) from instacart\_order\_products\_1g\_3000000 where 0.9 <= reordered <= 1.0**  **RESPONSE(JSON) <<< {'cause': 'success', 'cost': 0.054445, 'result': 'success', 'value': 9.12553891779578}**  **----------------------------------------------------------------------**  **predict : 9.1255**  **time cost : 0.0544 seconds**  **----------------------------------------------------------------------**  **Total elapsed time : 0.0651 seconds**  **======================================================================** |
|  |

**Callee server 출력**

|  |
| --- |
| **======================================================================**  **REQUEST(SQL) >>> select avg (add\_to\_cart\_order real) from instacart\_order\_products\_1g\_3000000 where 0.9 <= reordered <= 1.0**  **----------------------------------------------------------------------**  **# model name: instacart\_order\_products\_1g\_3000000**  **# values: avg(add\_to\_cart\_order real)**  **# function return: (None, ['avg', 'add\_to\_cart\_order', 'real', None])**  **# values: avg(add\_to\_cart\_order real)**  **======================================================================**  **PARAMETERS**  **----------------------------------------------------------------------**  **func : avg**  **x\_lb : 0.9**  **x\_ub : 1.0**  **where : [[], [], {'reordered': [0.9, 1.0, True, True]}]**  **filter : [0.9, 1.0]**  **======================================================================**  **RESULT**  **----------------------------------------------------------------------**  **0 1**  **dummy 9.125539**  **----------------------------------------------------------------------**  **time cost: 0.0544s.**  **======================================================================**  **RESPONSE(SQL) >>> {"cause": "success", "cost": 0.054445, "result": "success", "value": 9.12553891779578}**  **======================================================================**  **127.0.0.1 - - [01/Dec/2021 09:38:08] "POST / HTTP/1.1" 200 -** |

**Model serving 출력**

|  |
| --- |
| **======================================================================**  **PARAMESTERS : <class 'pandas.core.frame.DataFrame'>**  **----------------------------------------------------------------------**  **func : avg**  **x\_lb : 0.0**  **x\_ub : 0.1**  **where : [[], [], {'reordered': [0.9, 0.1, True, True]}]**  **config : {'device': 'cpu', 'n\_jobs': 1, 'v': True, 'b\_show\_latency': True, 'b\_print\_to\_screen': True, 'result2file': None, 'b\_use\_integral': False, 'n\_division': 20, 'epsabs': 10.0, 'epsrel': 0.1, 'limit': 30, 'model\_suffix': '.dill', 'slaves': {'container': {}}, 'sampling\_only': False, 'plot': False}**  **groups : None**  **filter : [0.9, 0.1]**  **======================================================================**  **PREDICT : <class 'pandas.core.frame.DataFrame'>**  **----------------------------------------------------------------------**  **0 1**  **0 dummy 10.903682**  **======================================================================** |

**MLflow API deployging & serving**

1. mlflow models serve -m ./mlruns/0/{runid}/artifacts/model -h 0.0.0.0 -p 8003 --no-conda

**Java client example**

* [https://github.com/traindb-project/TrainDB-ml/blob/main/restapi-client/JavaRestClientTest.java](https://github.com/traindb-project/traindb-ml/blob/main/restapi-client/JavaRestClientTest.java)

**Training data path**

* put .csv files for training to TrainDB-ml/server/etrimlwarehouse/ folder
* csv file example: [https://github.com/traindb-project/TrainDB-ml/blob/main/server/etrimlwarehouse/instacart\_order\_product\_600k.csv](https://github.com/traindb-project/traindb-ml/blob/main/server/etrimlwarehouse/instacart_order_product_600k.csv)