

Pycon 2021

MLflow를 이용한 머신러닝 실험부터 배포까지

sean.park@mathpresso.com

박선준 (Sean)

QANDA, Everyone's Personal Tutor.

Image Search “Strong Entry Point”



Study Tools “Higher Engagement”



Personalized Content “Effective Learning”



FREE

Strategy for Integration
view 33M | 1d ago

Google

[기초개념] 삼각함수의 그래프
10:10

An Introduction to Complex Numbers
view 120M | 2d ago

Google

Was this answer helpful?

누적
앱 다운로드

36M

월간 활성
사용자 (MAU)

9.8M

일 평균
질문수

6.4M

초당
업로드수

74

누적
문제해결수

2.5BN



교육차트 1위
20개국



누적 투자액
\$105M+



지원언어
7 KR,JP,VN,ID,TH,EN,ES

Agenda.

1. 머신러닝 실험부터 배포까지

- 1.1 머신러닝 프로젝트의 어려움
- 1.2 MLflow Tracking
- 1.3 MLflow Model
- 1.4 Serving with MLflow
- 1.5 회사의 활용 사례 소개

2. MLflow 을 이용한 실습

- 2.1 tracking server 를 설정하고 S3 와 연동하기
- 2.2 MLflow 에 모델 기록하기
- 2.3 mlflow에 기록된 모델을 Docker Image 화 하고, 실행시켜보기

1.1 머신러닝 프로젝트의 어려움

각 실험을 체계적으로 파악하고 관리하기가 어렵습니다.

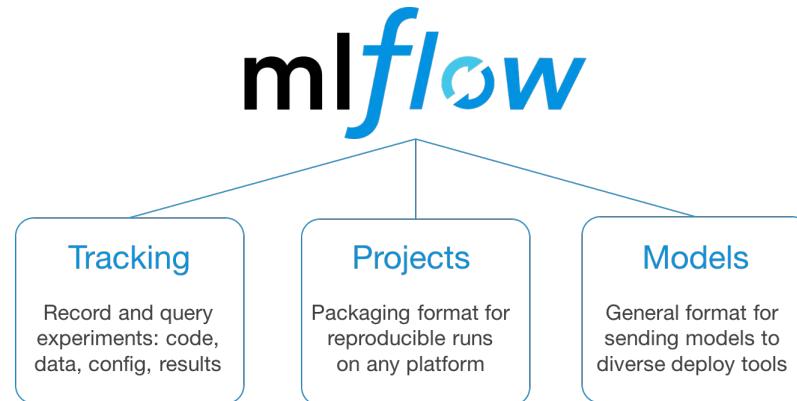
- 어떤 데이터를 이용해서 학습을 진행했었지 ?
- 이 모델에서 사용된 Feature 는 어떤 것이었지 ?
- 어떤 Parameter 를 이용해서 학습을 진행했었지 ?
- 실험 결과가 어디에 저장되어 있지 ?
- 학습된 모델의 버전은 어떻게 관리하지 ?
- 학습에 사용된 코드는 어떤 버전이지 ?

모델을 배포하는 것에 난이도가 있습니다.

- 사용하는 프레임워크가 연구자마다 상이한데, 배포는 어떤 방식으로 진행하지?
- 배포된 모델의 버전관리는 어떻게 하지?

1.2 MLflow Tracking

1. **Tracking** - 실험에서 사용된 코드, parameter, metric 등 실험에서 생산되는 모든 결과를 기록하고 조회할 수 있습니다. Python, REST, R, JAVA APIs 를 제공합니다.
2. **Project** - 어떤 플랫폼에서도 실험을 재현 가능하도록 패키징하는 방법을 제공합니다.
3. **Models** - 머신러닝 모델을 패키징하고, 서빙 할 수 있는 표준화된 방법을 제안합니다.



1.2 MLflow Tracking

Parameters : key-value inputs to your code

Metrics : numeric values that can update over time

Artifacts : files, including data and models

Tags and Notes : any additional information

Pycon2021

Track machine learning training runs in an experiment. [Learn more](#)

Experiment ID: 1 Artifact Location: file:///Users/sean/project/pycon2021/mlflow/mlruns/

Notes

None

Search Runs: [Filter](#) [Search](#) [Clear](#)

Showing 2 matching runs [Compare](#) [Delete](#) [Download CSV](#)

	Start Time	Run Name	User	Source	Version	Models	Metrics	Tags
<input type="checkbox"/>	2021-08-03 10:44:19	-	sean	track.py	-	-	2.5	1.2.0-GA
<input type="checkbox"/>	2021-08-03 10:44:19	-	sean	track.py	-	-	1.55	1.1.0-RC

[Load more](#)

1.2 MLflow Tracking

```

import mlflow

if __name__ == "__main__":
    mlflow.set_tracking_uri("http://localhost:5000")

    # Create an experiment and log two runs under it
    experiment_id = mlflow.create_experiment("Pycon2021")

    with mlflow.start_run(experiment_id=experiment_id):
        mlflow.log_metric("m", 1.55)
        mlflow.set_tag("s.release", "1.1.0-RC")

    with mlflow.start_run(experiment_id=experiment_id):
        mlflow.log_metric("m", 2.50)
        mlflow.set_tag("s.release", "1.2.0-GA")

    # Search all runs in experiment_id
    df = mlflow.search_runs([experiment_id], order_by=[ "metrics.m"])
    print(df[["metrics.m", "tags.s.release", "run_id"]])
    print("--")

    # Search the experiment_id using a filter_string with tag
    # that has a case insensitive pattern
    filter_string = "tags.s.release ILIKE '%rc%'"
    df = mlflow.search_runs([experiment_id], filter_string=filter_string)
    print(df[["metrics.m", "tags.s.release", "run_id"]])

```

Pycon2021

Track machine learning training runs in an experiment. Learn more

Experiment ID: 1 Artifact Location: file:///Users/sean/project/pycon2021/mlflow/mlruns/1

Notes: None

Search Runs: metrics.rmse < 1 and params.model = "tree" and tags.mlflow.source.type = "LOCAL"

	Start Time	Run Name	User	Source	Version	Models	Metrics	Tags
<input type="checkbox"/>	2021-08-03 10:44:19	-	sean	track.py	-	-	2.5	1.2.0-GA
<input type="checkbox"/>	2021-08-03 10:44:19	-	sean	track.py	-	-	1.55	1.1.0-RC

1.2 MLflow Tracking

exp_ltr_xgboost_sean2 > Run c144923f77e54b96836d18559f3589a1 ▾

Date: 2021-06-11 10:20:36	Source: main.py
User: sean	Duration: 9.0s
Status: FINISHED	
Parent Run: 3a53022flee4442b8ca18269d86515f1	

Notes 

None

Parameters

Name	Value
base_score	0.5
booster	gbtree
colsample_bylevel	1
colsample_bynode	0.8
colsample_bytree	0.4
gamma	0
gpu_id	-1
importance_type	gain
interaction_constraints	
learning_rate	0.1
max_delta_step	0
max_depth	9
min_child_weight	1
missing	nan
monotone_constraints	0

Metrics

Name	Value
ER	
ER_at_1	
ER_at_6	

Tags

Name	Value	Actions
No tags found.		

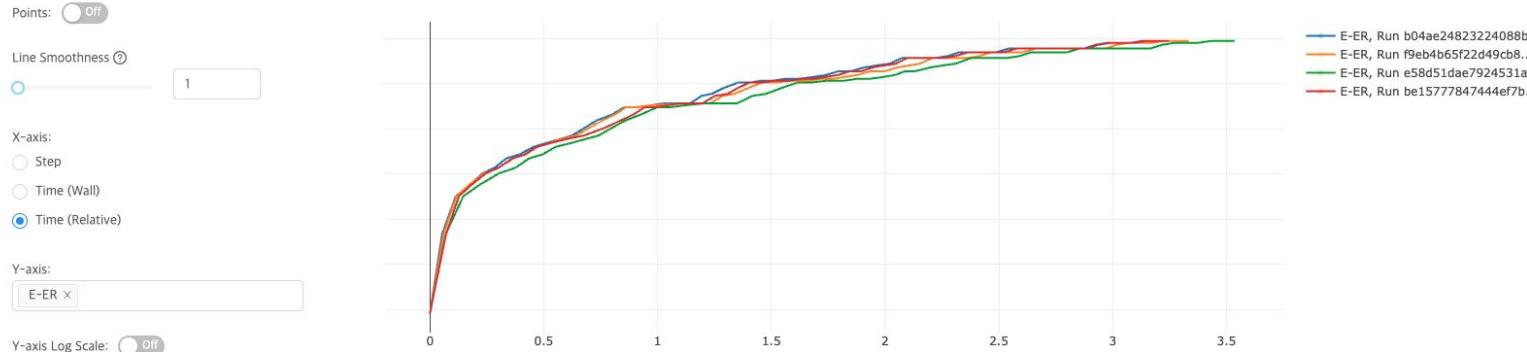
Add Tag

Artifacts

1.2 MLflow Tracking - Metric

- Accuray / Loss 등 임의의 Metric 을 실험마다 기록해 둘 수 있다.
- 이는 진행된 실험 간 비교를 할 때 유용하게 사용할 수 있다.

▼ Metrics	
Name	Value
ER ↗	
ER_at_1 ↗	
ER_at_6 ↗	



1.2 MLflow Tracking - Tags

- 각 실험에 임의의 Tag 를 달아 둘 수 있다.
- Search Runs 기능을 통하여 tag 를 이용하여 실험에 필터를 걸어서 원하는 실험을 찾아볼 수 있다.
- 완료된 실험들의 결과를 해석할 때 유용하게 사용할 수 있다.

MLflow Tracking UI Screenshot showing the 'Tags' section for an experiment named 'exp_ltr_xgboost_sean2' (Experiment ID: 40).

The 'Tags' section shows a table with columns: Name, Value, and Actions. It displays the message "No tags found.".

The 'Add Tag' form includes fields for Name and Value, and a 'Add' button.

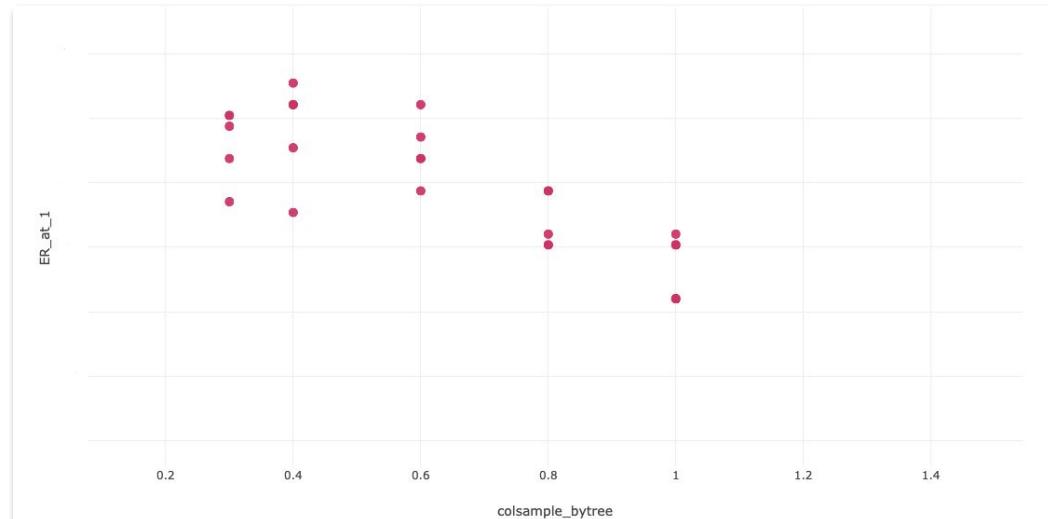
The 'Search Runs' bar contains the query "tags classifier = 'logistic'".

The results table shows 50 matching runs, each with a checkbox, Start Time, User, and Tags. The columns are: Start Time, User, Tags.

	Start Time	User	Tags
<input type="checkbox"/>	2021-06-11 10:27:13	sean	experiment logistic
<input type="checkbox"/>	2021-06-11 10:27:05	sean	LCSSeq logistic
<input type="checkbox"/>	2021-06-11 10:26:57	sean	LCSSeq logistic
<input type="checkbox"/>	2021-06-11 10:26:49	sean	LCSSeq logistic
<input type="checkbox"/>	2021-06-11 10:26:49	sean	LCSSeq logistic
<input type="checkbox"/>	2021-06-11 10:25:15	sean	Overlap logistic
<input type="checkbox"/>	2021-06-11 10:25:07	sean	Overlap logistic
<input type="checkbox"/>	2021-06-11 10:24:59	sean	Overlap logistic
<input type="checkbox"/>	2021-06-11 10:24:51	sean	Overlap logistic
<input type="checkbox"/>	2021-06-11 10:24:50	sean	Overlap logistic
<input type="checkbox"/>	2021-06-11 10:23:12	sean	Levenshtein logistic
<input type="checkbox"/>	2021-06-11 10:23:02	sean	Levenshtein logistic
<input type="checkbox"/>	2021-06-11 10:22:54	sean	Levenshtein logistic
<input type="checkbox"/>	2021-06-11 10:22:46	sean	Levenshtein logistic
<input type="checkbox"/>	2021-06-11 10:22:45	sean	Levenshtein logistic
<input type="checkbox"/>	2021-06-11 10:21:10	sean	Jaccard logistic
<input type="checkbox"/>	2021-06-11 10:21:01	sean	Jaccard logistic
<input type="checkbox"/>	2021-06-11 10:20:54	sean	Jaccard logistic
<input type="checkbox"/>	2021-06-11 10:20:46	sean	Jaccard logistic
<input type="checkbox"/>	2021-06-11 10:20:45	sean	Jaccard logistic
<input type="checkbox"/>	2021-06-11 10:19:04	sean	Cosine logistic
<input type="checkbox"/>	2021-06-11 10:18:56	sean	Cosine logistic
<input type="checkbox"/>	2021-06-11 10:18:47	sean	Cosine Intrinsic

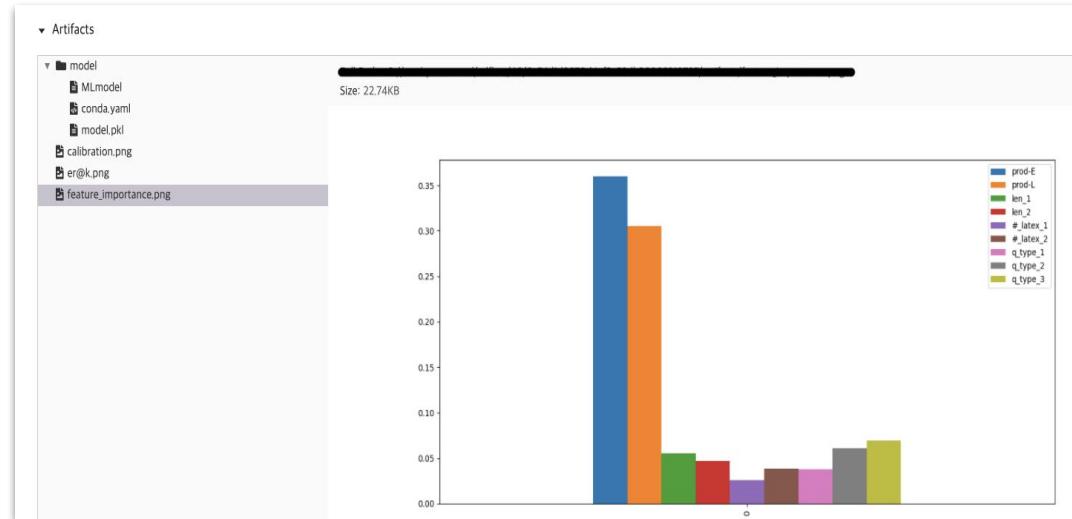
1.2 MLflow Tracking - Metric & Parameter

- Tag 를 통하여 실험을 필터링하여,
Parameter 와 Metric 의 상관관계를
분석할 수 있다.
- 왼쪽 예시는 Accuracy 와 xgboost 의
parameter 인 colsample_bytree 간의
상관관계를 보여준다.
- colsample_by_tree 가 낮을 수록 더
높은 정확도를 보임을 확인할 수 있다.



1.2 MLflow Tracking - Artifacts

- 실험에 사용된 임의의 파일을 업로드하여 관리할 수 있다.
- 아래 예시는 xgboost 의 Feature importance 를 matplotlib을 통하여 시각화한 자료



1.3 MLflow Models

```
import numpy as np
from sklearn.linear_model import LogisticRegression

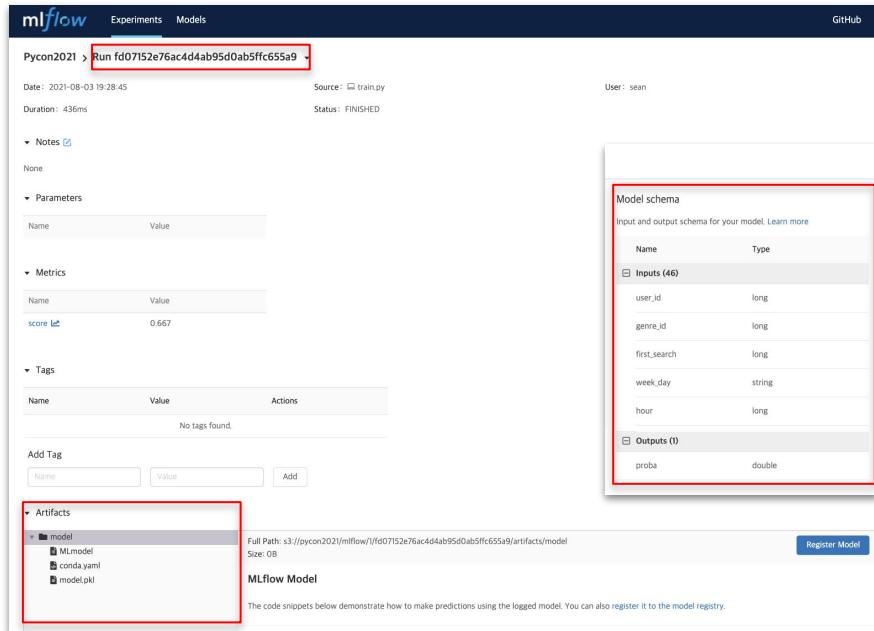
import mlflow
import mlflow.sklearn

if __name__ == "__main__":

    X = np.array([-2, -1, 0, 1, 2, 1]).reshape(-1, 1)
    y = np.array([0, 0, 1, 1, 1, 0])
    lr = LogisticRegression()
    lr.fit(X, y)

    experiment = mlflow.get_experiment_by_name("Pycon2021")
    with mlflow.start_run(experiment_id=experiment.experiment_id):
        score = lr.score(X, y)
        print("Score: %s" % score)
        mlflow.log_metric("score", score)
        mlflow.sklearn.log_model(lr, "model")
        print("Model saved in run %s" % mlflow.active_run().info.run_uuid)
```

1.3 MLflow Models



The screenshot shows the MLflow UI for a run named "Run fd07152e76ac4d4ab95d0ab5ffcc655a9". The page includes the following sections:

- Run Information:** Date: 2021-08-03 19:28:45, Source: train.py, Duration: 436ms, Status: FINISHED, User: sean.
- Notes:** None.
- Parameters:** A table with columns Name and Value.
- Metrics:** A table with columns Name and Value, showing a single entry: score with value 0.667.
- Tags:** A table with columns Name, Value, and Actions, showing "No tags found."
- Add Tag:** A form with fields for Name and Value, and an Add button.
- Artifacts:** A section listing artifacts:
 - model** (selected): Contains files MLmodel, conda.yaml, and model.pkl.
- MLflow Model:** A summary card with Full Path: s3://pycon2021/mlflow/fd07152e76ac4d4ab95d0ab5ffcc655a9/artifacts/model, Size: 0B, and a Register Model button.
- Model schema:** A table showing input and output schema for the model. The inputs schema is highlighted with a red border.

Name	Type
Inputs (46)	
user_id	long
genre_id	long
first_search	long
week_day	string
hour	long
Outputs (1)	
proba	double

1.1 머신러닝 프로젝트의 어려움



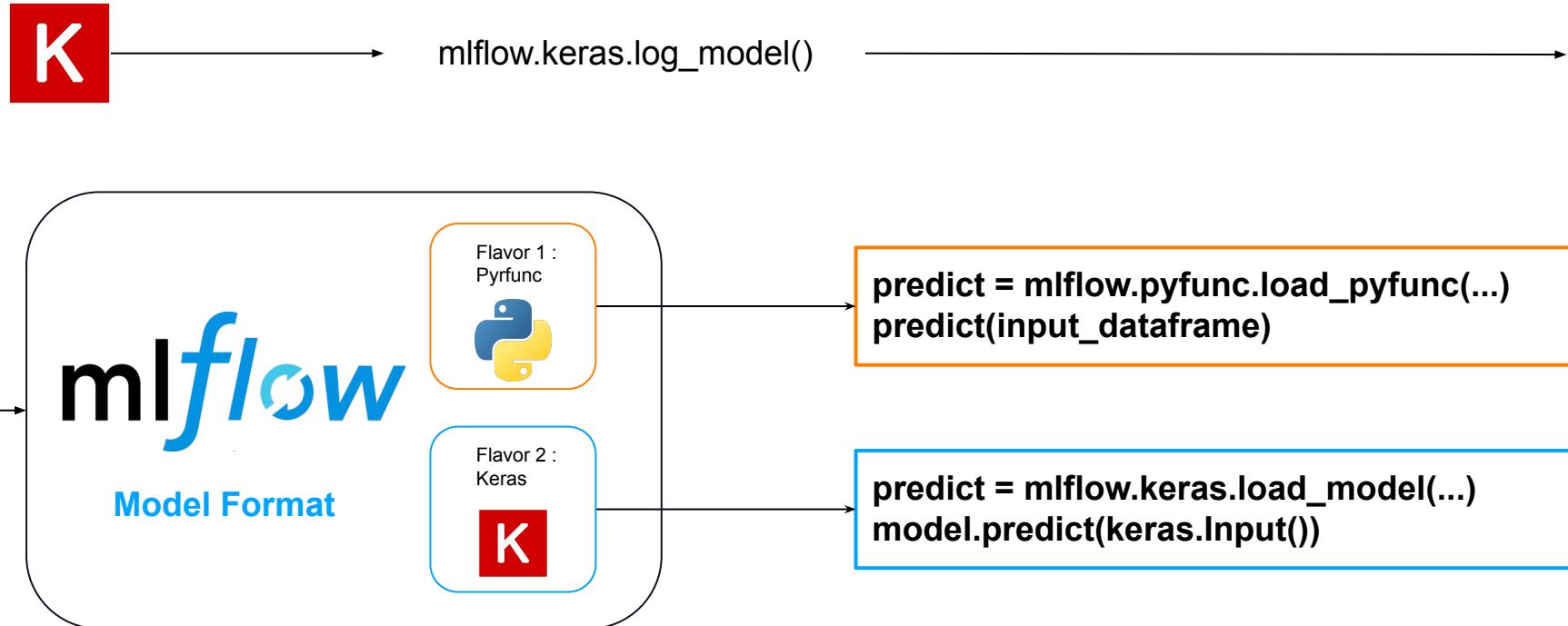
각 실험을 체계적으로 파악하고 관리하기가 어렵습니다.

- 어떤 데이터를 이용해서 학습을 진행했었지 ? - **mlflow artifact / ETL script with git**
- 이 모델에서 사용된 Feature 는 어떤 것이었지 ? - **mlflow logging**
- 어떤 Parameter 를 이용해서 학습을 진행했었지 ? - **mlflow logging**
- 실험 결과가 어디에 저장되어 있지 ? - **mlflow (tracking server / s3)**
- 학습된 모델의 버전은 어떻게 관리하지 ? - **mlflow model run id**
- 학습에 사용된 코드는 어떤 버전이지 ? - **mlflow with git**

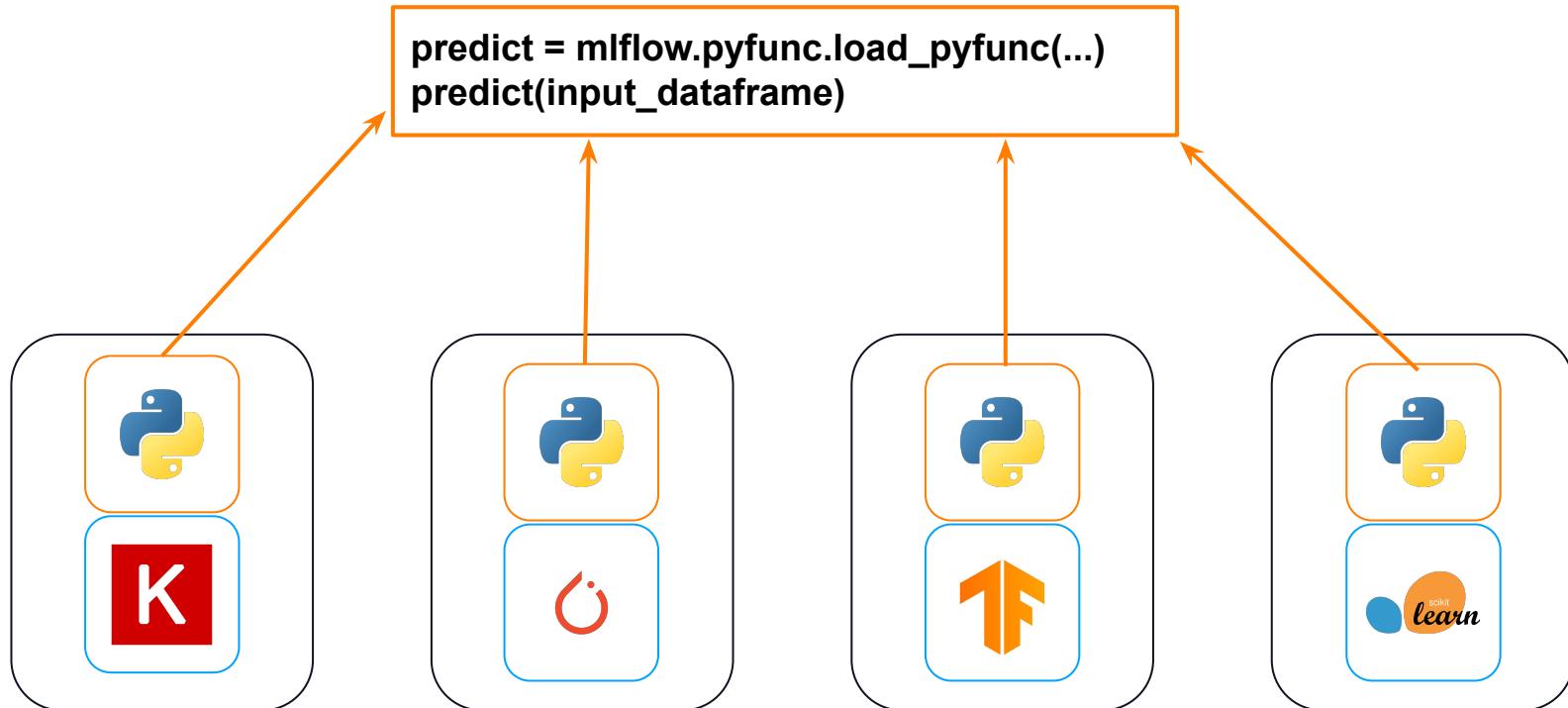
모델을 배포하는 것에 난이도가 있습니다.

- 사용하는 프레임워크가 연구자마다 상이한데, 배포는 어떤 방식으로 진행하지?
- 배포된 모델의 버전관리는 어떻게 하지?

1.3 MLflow Models - Flavor



1.3 MLflow Models - Flavor



1.3 MLflow Models

Model schema

Input and output schema for your model. [Learn more](#)

Name	Type
Inputs (36)	
user_id	long
genre_id	long
first_search	long
week_day	string
hour	long
Outputs (1)	
proba	double

Make Predictions

Predict on a Spark DataFrame:

```
import mlflow
logged_model = 's3://mathpresso-rnd/mlflow/8/c891cf59cff3422598c7511e05d01590/artifacts/stepner-dcn'

# Load model as a Spark UDF.
loaded_model = mlflow.pyfunc.spark_udf(logged_model)

# Predict on a Spark DataFrame.
df.withColumn(loaded_model, 'my_predictions')
```

Predict on a Pandas DataFrame:

```
import mlflow
logged_model = 's3://mathpresso-rnd/mlflow/8/c891cf59cff3422598c7511e05d01590/artifacts/stepner-dcn'

# Load model as a PyFuncModel.
loaded_model = mlflow.pyfunc.load_model(logged_model)

# Predict on a Pandas DataFrame.
import pandas as pd
loaded_model.predict(pd.DataFrame(data))
```

1.4 Serving with MLflow

- mlflow models 는 실험적 기능으로 build-docker 기능을 제공합니다.
- build-docker 명령어는 Flask 기반의 웹 서버를 이미지화합니다.
- /ping 과 /invocations 두 종류의 endpoint를 제공하여 /invocations 을 통하여 mlflow에 저장된 모델의 추론 결과를 반환받을 수 있습니다.

```
def init(model: PyFuncModel):  
    """  
    Initialize the server. Loads pyfunc model from the path.  
    """  
    app = flask.Flask(__name__)  
    input_schema = model.metadata.get_input_schema()  
  
    @app.route("/ping", methods=["GET"])  
    def ping(): # pylint: disable=unused-variable  
        """  
        Determine if the container is working and healthy.  
        We declare it healthy if we can load the model successfully.  
        """  
        health = model is not None  
        status = 200 if health else 404  
        return flask.Response(response="\n", status=status, mimetype="application/json")  
  
    @app.route("/invocations", methods=["POST"])  
    @catch_mlflow_exception  
    def transformation(): # pylint: disable=unused-variable  
        """  
        Do an inference on a single batch of data. In this sample server,  
        we take data as CSV or json, convert it to a Pandas DataFrame or Numpy,  
        generate predictions and convert them back to json.  
        """
```

1.4 Serving with MLflow

- Flask Application 은 gunicorn 과 nginx 를 앞단에 감싸줌으로써 효과적으로 cpu 사용합니다.
- 즉, build-docker 명령어는 Flask + gunicorn + nginx 기반의 웹 서버를 이미지화하며, 추가적인 gunicorn 설정을 넣어 줄 수 있습니다.

```

def _serve_pyfunc(model):
    conf = model.flavors[pyfunc.FLAVOR_NAME]
    bash_cmds = []
    if pyfunc.ENV in conf:
        if not os.environ.get(DISABLE_ENV_CREATION) == "true":
            _install_pyfunc_deps(MODEL_PATH, install_mlflow=True)
            bash_cmds += ["source /miniconda/bin/activate custom_env"]
    nginx_conf = resource_filename(mlflow.models._name_, "container/scoring_server/nginx.conf")

    # option to disable manually nginx. The default behavior is to enable nginx.
    start_nginx = False if os.getenv(DISABLE_NGINX, "false").lower() == "true" else True
    nginx = Popen(["nginx", "-c", nginx_conf]) if start_nginx else None

    # link the log streams to stdout/err so they will be logged to the container logs.
    # Default behavior is to do the redirection unless explicitly specified by environment variable.

    if start_nginx:
        check_call(["ln", "-sf", "/dev/stdout", "/var/log/nginx/access.log"])
        check_call(["ln", "-sf", "/dev/stderr", "/var/log/nginx/error.log"])

    cpu_count = multiprocessing.cpu_count()
    os.system("pip -V")
    os.system("python -V")
    os.system('python -c"from mlflow.version import VERSION as V; print(V)"')
    cmd = (
        "gunicorn -w {cpu_count} ".format(cpu_count=cpu_count)
        + "${GUNICORN_CMD_ARGS} mlflow.models.container.scoring_server.wsgi:app"
    )
    bash_cmds.append(cmd)
    gunicorn = Popen(['/bin/bash', "-c", " && ".join(bash_cmds)])

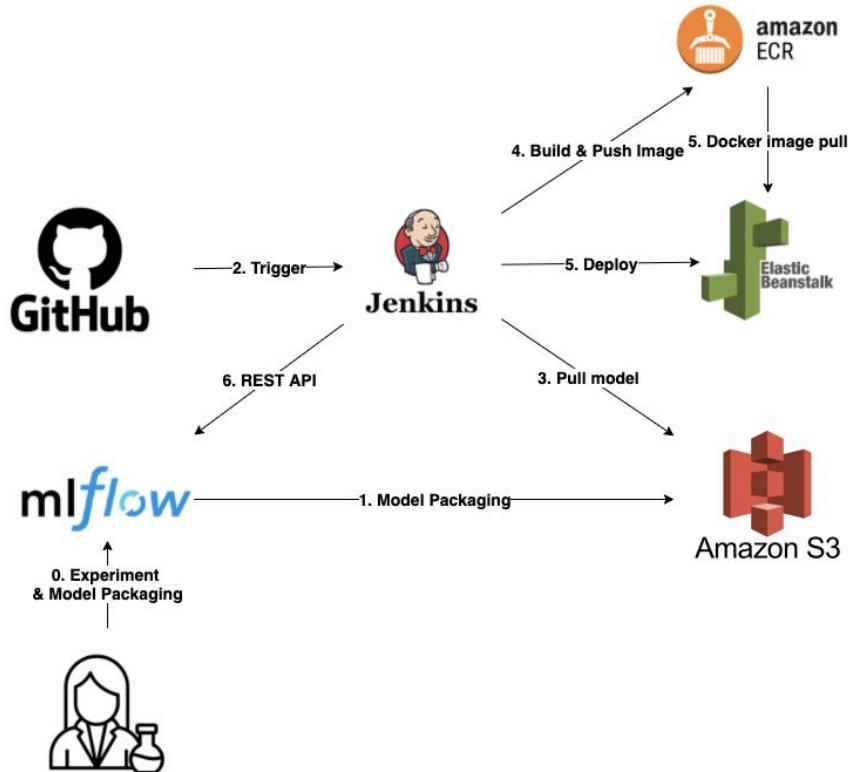
    procs = [p for p in [nginx, gunicorn] if p]

    signal.signal(signal.SIGTERM, lambda a, b: _sigterm_handler(pids=[p.pid for p in procs]))
    # If either subprocess exits, so do we.
    awaited_pids = _await_subprocess_exit_any(procs=procs)
    _sigterm_handler(awaited_pids)

```

1.4 회사에서 활용 사례 소개

- 많은 ML 엔지니어의 백그라운드가 SW에 있지 않기 때문에, 앞선 Iteration에 있어 병목이 발생합니다.
- ML 엔지니어의 가장 큰 병목인 배포 과정을 MLflow를 통하여 자동화하여 배포의 난이도를 낮춰 ML 프로덕트 개발 속도를 증진시키고자 하는 목적이 있었습니다.
- MLflow 에서 제공하는 Model Packaging 기능을 활용하여 배포의 난이도를 낮추고자 함.



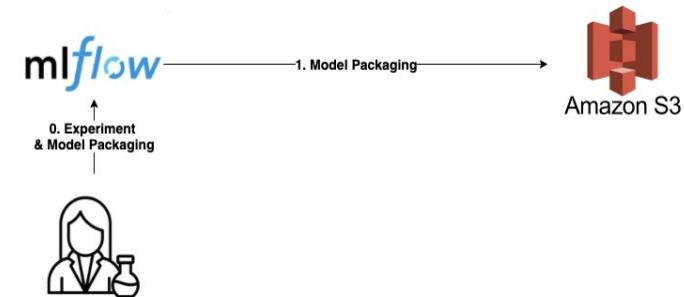
1.4 회사에서 활용 사례 소개 - 실험 및 모델 패키징

[\[REDACTED\] / ai-team-notebook](#) Private

< Code Issues Pull requests Actions Projects Wiki Security Insights Settings

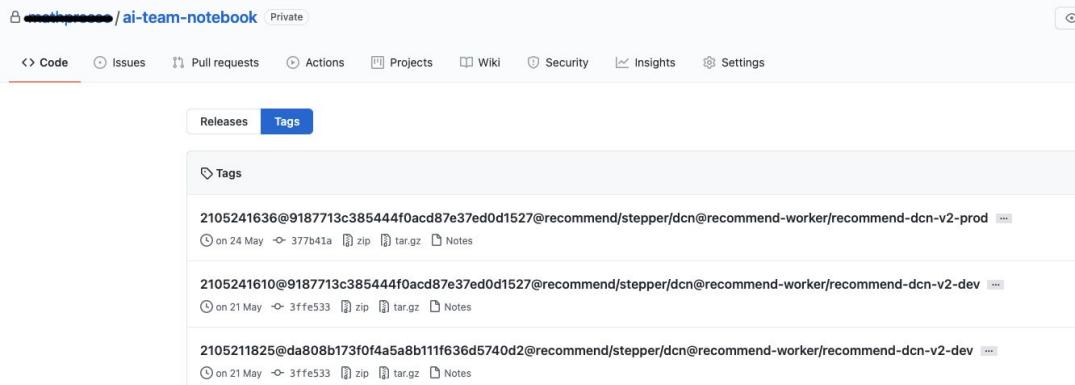
master 3 branches 14 tags Go to file Add file Code

	sean Park	Add GraphValidator.ipynb
		✓ f3934b4 5 hours ago 174 commits
	analysis	Add GraphValidator.ipynb
	data	Merge branch 'master' of github.com:mathpresso/ai-team-notebook
	external-request/sean	Add community feed related materials
	nlp	Merge pull request #18 from mathpresso/sean/search-ltr-wo-lm
	ocr	Add ocr directory and move projects
	recommend	add test_local.py
	video-solution	Add sq_postprocess.py for uq clustering.
	.gitignore	Add gitignore MacOS cache files
	Dockerfile	update deploy scripts
	Dockerrun.aws.json	update deploy scripts
	Jenkinsfile	Add mlflow log deploy history to Jenkinsfile
	README.md	Update project structure to Readme.md
	build-docker.sh	update help
	download_artifacts.sh	Update Jenkinsfile 1
	unicorn_conf.py	update deploy scripts



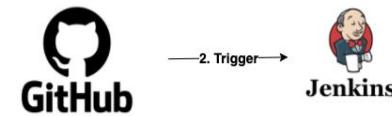
1.4 회사에서 활용 사례 소개 - Jenkins

- {datetime}@{run_id}@{project_path}@{eb_app}/{eb_env}
- github에 Tag를 달면 Tag의 내용을 기반하여 배포가 수행된다.



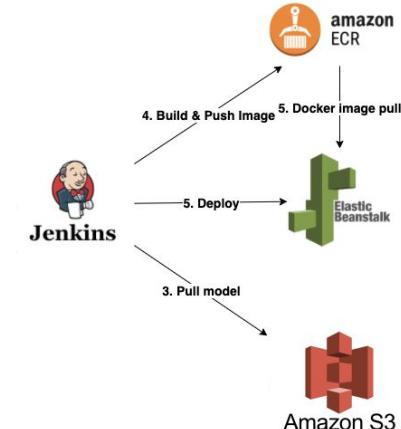
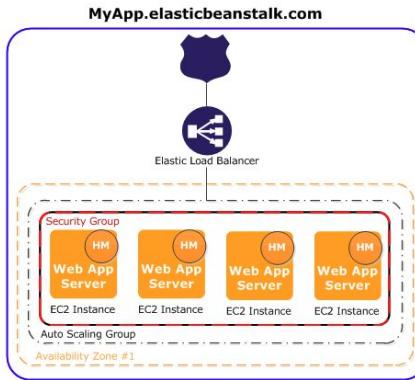
A screenshot of a GitHub repository page for 'ai-team-notebook'. The repository is private. The navigation bar includes Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. The main content shows the 'Tags' tab selected, displaying three tags:

- 2105241636@9187713c385444f0acd87e37ed0d1527@recommend/stepper/dcn@recommend-worker/recommend-dcn-v2-prod (created on 24 May, 377b41a commit, zip, tar.gz, Notes)
- 2105241610@9187713c385444f0acd87e37ed0d1527@recommend/stepper/dcn@recommend-worker/recommend-dcn-v2-dev (created on 21 May, 3ffe533 commit, zip, tar.gz, Notes)
- 2105211825@da808b173f0f4a5a8b111f636d5740d2@recommend/stepper/dcn@recommend-worker/recommend-dcn-v2-dev (created on 21 May, 3ffe533 commit, zip, tar.gz, Notes)



1.4 회사에서 활용 사례 소개 - ElasticBeanstalk

- 로드 밸런싱, Auto Scaling부터 시작하여 애플리케이션 상태 모니터링에 이르기까지 배포를 자동으로 처리합니다.
- ElasticBeanstalk → k8s



Application versions					
	Version label	Description	Date created	Source	Deployed to
<input type="checkbox"/>	recommend-dcn-v2-prod:9187713c385444f0acd87e37ed0d1527-2105241636	2105241636@9187713c385444f0acd87e37ed0d1527@recommend/stepper/dcn@recommend-worker/recommend-dcn-v2-prod: 377b41a: add test_local.py	2021-05-24T16:39:07+09:00	recommend-worker/recommend-dcn-v2-prod:9187713c385444f0acd87e37ed0d1527-2105241636.zip	recommend-dcn-v2-prod
<input type="checkbox"/>	recommend-dcn-v2-dev:9187713c385444f0acd87e37ed0d1527-2105241610	2105241610@9187713c385444f0acd87e37ed0d1527@recommend/stepper/dcn@recommend-worker/recommend-dcn-v2-dev: 3ffe533: Merge branch 'master' of https://github.com/mathpresso/ai-team-notebook	2021-05-24T16:19:41+09:00	recommend-worker/recommend-dcn-v2-dev:9187713c385444f0acd87e37ed0d1527-2105241610.zip	recommend-dcn-v2-dev
<input type="checkbox"/>	recommend-dcn-v2-dev:da808b173f0f4a5a8b111f636d5740d2-2105211825	2105211825@da808b173f0f4a5a8b111f636d5740d2@recommend/stepper/dcn@recommend-worker/recommend-dcn-v2-dev: 3ffe533: Merge branch 'master' of https://github.com/mathpresso/ai-team-notebook	2021-05-21T18:25+09:00	recommend-worker/recommend-dcn-v2-dev:da808b173f0f4a5a8b111f636d5740d2-2105211825.zip	-
<input type="checkbox"/>	recommend-dcn-v2-prod:da808b173f0f4a5a8b111f636d5740d2-2105211802	2105211802@da808b173f0f4a5a8b111f636d5740d2@recommend/stepper/dcn@recommend-worker/recommend-dcn-v2-prod: 3ffe533: Merge branch 'master' of https://github.com/mathpresso/ai-team-notebook	2021-05-21T18:04:48+09:00	recommend-worker/recommend-dcn-v2-prod:da808b173f0f4a5a8b111f636d5740d2-2105211802.zip	-

1.4 회사에서 활용 사례 소개 - Model Registry




mlflow Experiments Models GitHub Docs

Registered Models > recommend-worker/recommend-dcn-v2-prod ▾

Created Time: 2021-04-08 18:51:37 Last Modified: 2021-06-16 18:47:55

▼ Description 

None

▼ Tags

Name	Value	Actions
No tags found.		

Add Tag

Name	Value	Add
------	-------	-----

▼ Versions

All Active(1)

Compare

	Version	Registered at ▾	Created by	Stage	Description
<input type="checkbox"/>	Version 7	2021-05-24 16:47:00		Production	
<input type="checkbox"/>	Version 6	2021-05-21 18:09:42		None	

6. REST API



Run 9187713c385444f0acd87e37ed0d1527 ▾

▼ Tags

Name	Value	Actions
ECR_IMAGE	.dkr.ecr.ap-northeast-2.amazonaws.com/mathpresso/ai-team/ml-worker/recommend/stepper/dcnn:9187713c385444f0acd87e37ed0d1527-2105241636	 

1.1 머신러닝 프로젝트의 어려움



각 실험을 체계적으로 파악하고 관리하기가 어렵습니다.

- 어떤 데이터를 이용해서 학습을 진행했었지 ? - **mlflow artifact / ETL script with git**
- 이 모델에서 사용된 Feature 는 어떤 것이었지 ? - **mlflow logging**
- 어떤 Parameter 를 이용해서 학습을 진행했었지 ? - **mlflow logging**
- 실험 결과가 어디에 저장되어 있지 ? - **mlflow (tracking server / s3)**
- 학습된 모델의 버전은 어떻게 관리하지 ? - **mlflow model run id**
- 학습에 사용된 코드는 어떤 버전이지 ? - **mlflow with git**



모델을 배포하는 것에 난이도가 있습니다.

- 사용하는 프레임워크가 연구자마다 상이한데, 배포는 어떤 방식으로 진행하지?
- 배포된 모델의 버전관리는 어떻게 하지?

2. MLflow 를 이용한 실습

1. Tracking server 를 설정하고 S3 와 연동하기
2. 임의의 Framework 에 대해 Customized 된 모델 mlflow 에 기록하기
3. mlflow에 기록된 모델을 Docker Image 화 하고, 실행시켜보기

2.1 Tracking server 를 설정하고 S3 와 연동하기

```
mlflow server \  
    --backend-store-uri sqlite:///mlflow.db \  
    --default-artifact-root s3://pycon2021/mlflow/ \  
    --host 0.0.0.0 \  
    --port 5000
```

```
(pycon2021) sean@ip-192-168-0-5 scripts % bash 01.start_tracking_server.sh  
[2021-08-04 08:13:43 +0900] [86510] [INFO] Starting gunicorn 20.1.0  
[2021-08-04 08:13:43 +0900] [86510] [INFO] Listening at: http://0.0.0.0:5000 (86510)  
[2021-08-04 08:13:43 +0900] [86510] [INFO] Using worker: sync  
[2021-08-04 08:13:43 +0900] [86513] [INFO] Booting worker with pid: 86513  
[2021-08-04 08:13:43 +0900] [86514] [INFO] Booting worker with pid: 86514  
[2021-08-04 08:13:43 +0900] [86515] [INFO] Booting worker with pid: 86515  
[2021-08-04 08:13:43 +0900] [86516] [INFO] Booting worker with pid: 86516
```

-

2.2 MLflow 에 Model 기록하기

- MLflow 는 프레임워크에 관계없이 모델을 패키징할 수 있는 방법을 제공합니다.
- artifacts 를 활용하여 모델의 추론에 필요한 임의의 파일을 Load 할 수 있습니다.

```
import mlflow.pyfunc

# Define the model class
class AddN(mlflow.pyfunc.PythonModel):

    def __init__(self, n):
        self.n = n

    def predict(self, context, model_input):
        # some process using necessary_file
        with open(context.artifacts["necessary_file"], 'r') as f:
            print(f.read())

        return model_input.apply(lambda column: column + self.n)

if __name__ == "__main__":
    mlflow.set_tracking_uri("http://localhost:5000")
    experiment = mlflow.get_experiment_by_name("Pycon2021")

    with mlflow.start_run(experiment_id=experiment.experiment_id):

        artifacts = {
            "necessary_file" : "./necessary_file",
        }

        # Construct and save the model
        model_path = "add_n_model"
        add5_model = AddN(n=5)

        mlflow.pyfunc.log_model(
            artifact_path="models",
            python_model = add5_model,
            artifacts=artifacts)
```

2.3 MLflow에 기록된 모델로 Docker Image 만들기

```
# Build an image that can serve mlflow models.
ARG BASE_IMAGE=ubuntu:18.04

FROM ${BASE_IMAGE}

ARG MLFLOW_VER="1.19.0"

ENV MLFLOW_DISABLE_ENV_CREATION="true"
ENV DISABLE_NGINX="true"

RUN apt-get -y update \
    && apt-get install -y --no-install-recommends \
        wget \
        curl \
        ca-certificates \
        bzip2 \
        build-essential \
        cmake \
        git-core \
    && rm -rf /var/lib/apt/lists/*

# Download and setup miniconda
RUN curl -L https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh >> miniconda.sh
RUN bash ./miniconda.sh -b -p /miniconda; rm ./miniconda.sh;
ENV PATH="/miniconda/bin:$PATH"

# Set up the program in the image
WORKDIR /opt/mlflow
RUN pip install "mlflow==${MLFLOW_VER}"
ADD model /opt/ml/model
RUN python -c "from mlflow.models.container import _install_pyfunc_deps; _install_pyfunc_deps('/opt/ml/model', install_mlflow=False)"

CMD /bin/bash -c "source /miniconda/bin/activate custom_env && gunicorn --bind=0.0.0.0:8080 mlflow.models.container.scoring_server.wsgi:app"

EXPOSE 8080
```

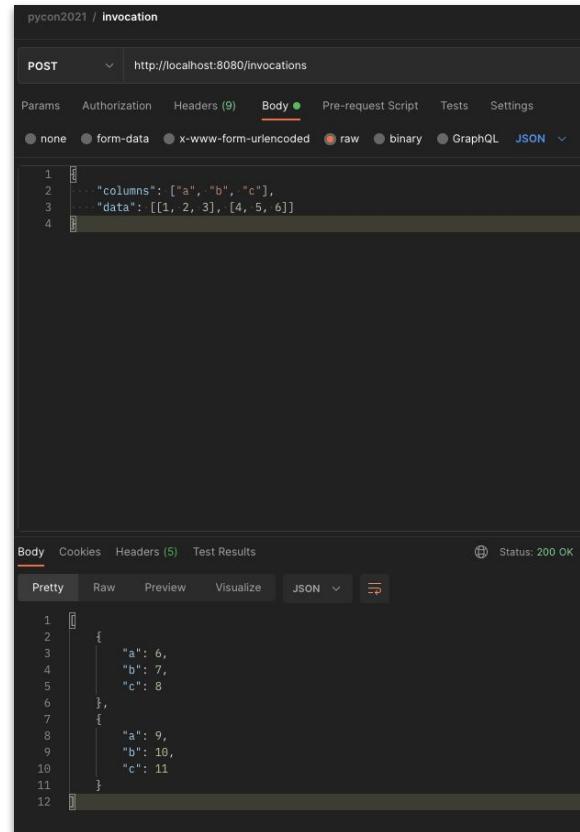


custom_img

1.88GB

2.3 mlflow에 기록된 모델에 요청 보내고 확인하기

```
sean@Seanui-MacBookPro ~ % docker run -p 8080:8080 ffbb27d2c2ef
[2021-08-03 23:28:30 +0000] [8] [INFO] Starting gunicorn 20.1.0
[2021-08-03 23:28:30 +0000] [8] [INFO] Listening at: http://0.0.0.0:8080 (8)
[2021-08-03 23:28:30 +0000] [8] [INFO] Using worker: sync
[2021-08-03 23:28:30 +0000] [21] [INFO] Booting worker with pid: 21
```



The screenshot shows a Postman interface for a POST request to `http://localhost:8080/invocations`. The `Body` tab is selected, showing the following JSON payload:

```
1 ... "columns": ["a", "b", "c"],
2 ...
3 ...
4 ... "data": [[1, 2, 3], [4, 5, 6]]
```

The response tab shows the following JSON data:

```
1 []
2 {
3   "a": 6,
4   "b": 7,
5   "c": 8
6 },
7 [
8   {
9     "a": 9,
10    "b": 10,
11    "c": 11
12 }
```

The status bar indicates `Status: 200 OK`.