

BentoML 튜토리얼

일시 2022년 04월 23일
장소 Google Meet
발표 이정훈



PROFILE

이정훈 Jung Hoon

인터파크 특집사 개발팀
가짜연구소 아카데미&커뮤니티 빌더

01

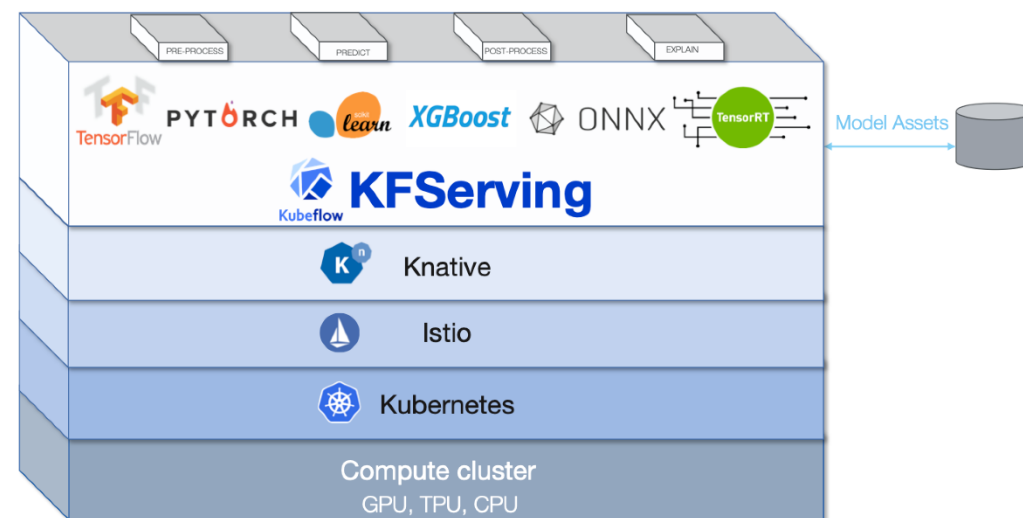
Serving

Online serving svcs Offline serving

01. Serving

Concept

- 모델 서빙이란 사용자에게 모델 예측 결과를 효율적으로 전달하는 방식이다. 서비스 기획과 운영에 따라 모델 서빙의 시스템 구성은 달라질 수 있다.
- 모델 서빙이 실시간에 가까워질수록, 운영 중인 모델의 수가 많아질수록 해결해야 할 문제는 눈덩이처럼 늘어난다. 이러한 문제는 크게 모델 버전 관리, 운영환경 구축, 고가용성과 확장성 등이 있다. 이 문제를 해결하기 위해 모델 컨테이너, 카나리 테스트, 롤백, 모델 저장소와 같은 개념이 필요하다.
- 단순히 데이터를 사용해 모델을 만들고 분석하는 것을 넘어, 실제 사업적 성과를 만들기 위해선 효율적인 모델 서빙 시스템이 필요하다.



01. Serving

Batch serving

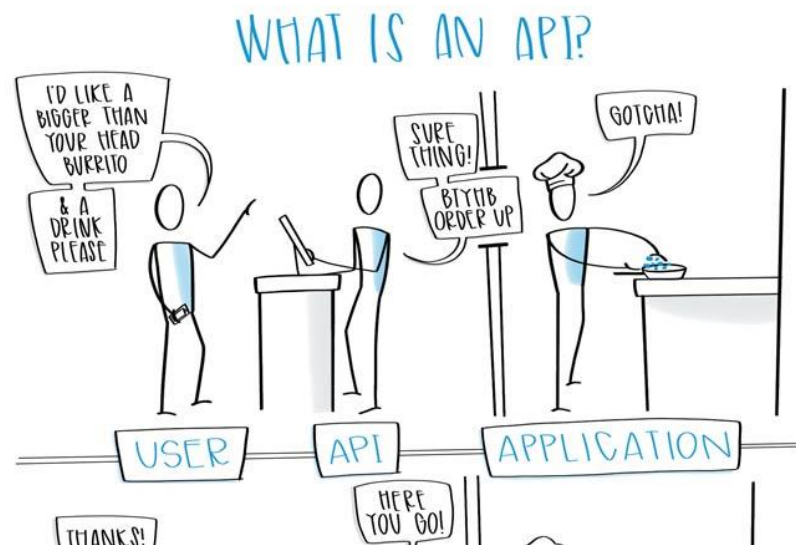
- Batch serving은 특정 주기로 서빙을 하고 Batch 단위로 많은 양을 한번에 처리한다. Airflow, Crontab 등을 사용해 특정 주기마다 값을 예측한다.
- 모델이 미리 테스트 데이터에 대한 추론 작업을 진행하고 해당 결과를 고성능 데이터베이스 (redis, memcached 등)에 캐싱한다. 그리고 DB에 저장된 결과를 사용자에게 서빙한다.
- 유사상품 및 영화추천 같이 결과값을 미리 저장하고 사용자에게 제공하면 되는 경우에 사용될 수 있다.
- Batch serving의 장점은 구현이 상대적으로 간단하다는 것이다. 사용자에게 직접적으로 모델이 노출되지 않고, 실시간으로 재학습되거나 변화 하는 부분이 없기 때문이다.
- Batch Serving의 단점은 새로운 데이터가 입력되었을 때이다. 새로운 모델을 적용하면 전체 데이터를 업데이트해야 한다. 업데이트가 완료되면 이전 시스템을 중지하고 새로운 시스템으로 교체한다.



01. Serving

Online serving

- Online serving는 API 서버를 만들어 실시간 요청에 따른 반응을 하는 서빙 방법이다.
따라서 동시다발적으로 들어오는 요청에 대한 확장 정책이 요구된다.
- Online serving은 애플의 시리나 챗봇 모델 같은 서비스를 만드는 경우에 필요하다.
- Online 방식의 장점은 Batch serving과는 다르게 입력값으로 들어온 데이터에 대해서만 모델을 적용한다는 것이다.
- Online 방식의 단점은 API Server가 안정적으로 운영되어야하기 때문에 서버 운영을 신경써야 한다는 점이다.



01. Serving

Edge deployment

- 앞에서 언급한 Batch 및 Online 방식은 모두 Client-Server 아키텍처이다. Edge deployment는 Client 단에 배포하는 방식으로 대표적인 케이스로 모바일 기기에 배포가 있다.
- Edge deployment의 장점은 서버의 부담이 줄어들고 서버 운영비용이 감소한다는 것이다.
- Edge deployment의 단점은 클라이언트로 사용할 수 있는 하드웨어(컴퓨터, 모바일 기기)나 웹 브라우저에 배포를 해야 하기 때문에 Edge 배포가 까다롭다는 것이다. 그래도 최근 하드웨어 및 소프트웨어 측면에서 빠르게 발전하고 있기 때문에 이전보다는 가능성인 보인다는 면이 있다. 소프트웨어 적으로는 Apple의 Core ML 및 Google의 TensorFlow JS와 같은 기술은 클라이언트 플랫폼에서 기계 학습 모델 추론을 실행하기 위한 SDK를 제공하고 있다. 이 뿐만 아니라 하드웨어 측면에서도 GPU, CPU 및 모바일 SoC에 텐서 코어와 같은 머신 러닝 추론 기능이 칩에 바로 내장된 상태로 제공되고 있다.



01. Serving

Serving 도구



mlflow



02

BentoML

Exercise

02. BentoML

Concept

- BentoML은 'Model Serving Made Easy' 라는 슬로건을 사용하는 것처럼, 간단하게 다양한 ML 프레임워크를 서빙할 수 있는 편의성에 기능이 집중되어 있다.
- 대표적인 프레임워크인 Scikit-Learn, PyTorch, Tensorflow2뿐만 아니라 널리 사용되는 딥러닝 라이브러리들(e.g. Transformers, Pytorch Lightning, Detectron 등) 역시 지원을 하고 있다.



02. BentoML

Titanic

- 경로 : /bentoml_tutorial/titanic/train.py
- 59~61번째 줄에 있는 부분이 Titanic 모델을 저장하는 과정이다.
- 60번째 줄에 pack 부분에서 사용이 되는 Model, Tokenizer 등을 셋팅한다.
- 61번째 줄의 save를 통해 저장한다.

```
42 if __name__ == '__main__':
43     # Directory
44     train_dir = "../data/train_titanic.csv"
45     test_dir = "../data/test_titanic.csv"
46     model_dir = "model_nb.pickle"
47
48     # Flow
49     train, test = load_data(train_dir, test_dir)
50     train_x, train_y, test_x, test_y = pre_processing(train, test)
51     model = build_model(train_x, train_y)
52     score = evaluation(model, test_x, test_y)
53     print("점수 :", score)
54
55     # file save
56     pickle.dump(model, open(model_dir, 'wb'))
57
58     # bentoml
59     titanic = Titanic()
60     titanic.pack('model', model)
61     titanic.save()
```

02. BentoML

Titanic

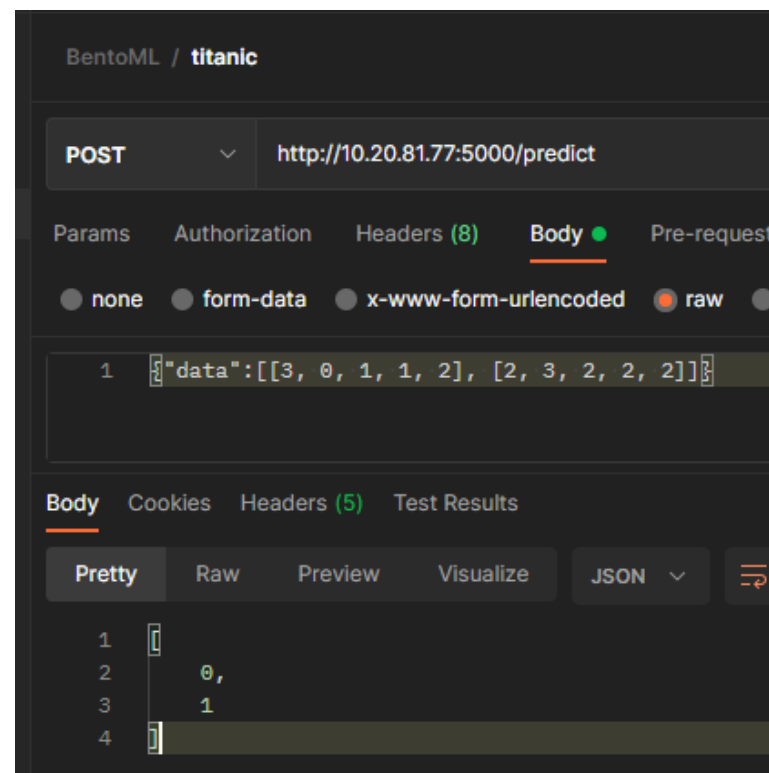
- 경로 : /bentoml_tutorial/titanic/service.py
- 앞에서 사용한 Titanic의 클래스 정의 부분이다.
- 10번째 env에서 필요한 라이브러리 환경등도 미리 정의할 수 있다.
- 11번째 artifacts 부분에서 pack할 부분을 정의해야 한다.
- 15번째 parsed_jsons는 json으로 입력된 데이터를 가져오는 부분이다.
- 16번째 self.artifacts.model 부분이 앞에서 pack한 model 파일을 가져오는 부분이다.

```
1  import pandas as pd
2
3  from typing import List
4  from bentoml import env, artifacts, api, BentoService
5  from bentoml.adapters import DataframeInput
6  from bentoml.frameworks.sklearn import SklearnModelArtifact
7  from bentoml.adapters import JsonInput
8  from bentoml.types import JsonSerializable
9
10 @env(infer_pip_packages=True)
11 @artifacts([SklearnModelArtifact('model')])
12 class Titanic(BentoService):
13     @api(input=JsonInput(), batch=True)
14     def predict(self, parsed_jsons: List[JsonSerializable]):
15         input_data = parsed_jsons[0]['data']
16         pred_y = self.artifacts.model.predict(input_data)
17         return [pred_y]
```

02. BentoML

Titanic

- API 서버 띄우기 : `bentoml serve Titanic:latest --port 1000`
- 요청 보내 결과 확인



02. BentoML

기존 방법

- 경로 : /bentoml_tutorial/titanic/inference.py
- 모델을 불러오는 부분과 flask와 연결해 API 서버를 구축한다.
- 전처리 도구가 별로 필요없는 정형 데이터의 경우 크게 복잡하지는 않지만.. 자연어처리나 이미지 같은 경우에는 복잡하다. 뒤의 실습에서 확인해보겠다.

7 lines (5 sloc) | 166 Bytes

```
1  import pickle
2
3  model_dir = "model_nb.pickle"
4
5  model = pickle.load(open(model_dir, 'rb'))
6  pred_y = model.predict([[3, 1, 17.4, 1, 0], [0, 0, 1.2, 1, 0]])
7  print(pred_y)
```

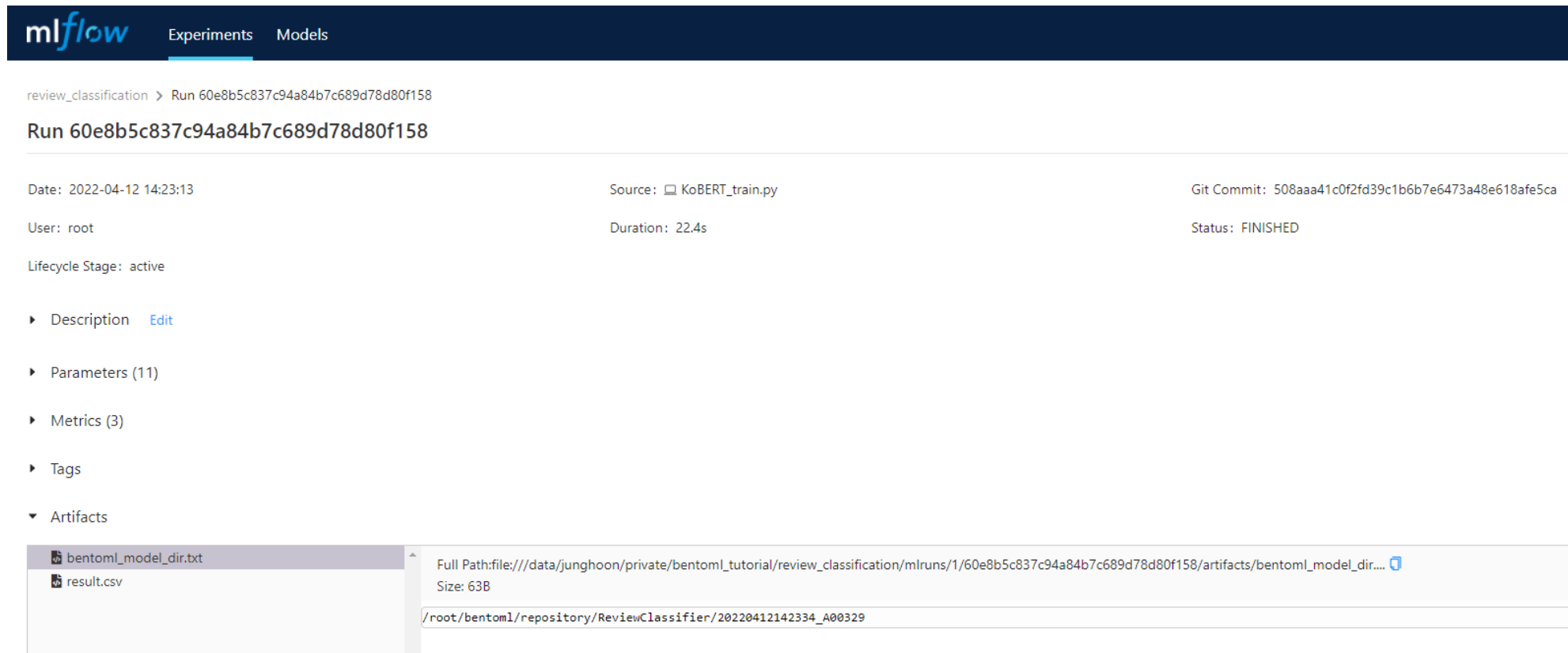
03

MLflow & BemtoML

Tracking & Serving

03. MLflow & BentoML


MLflow



The screenshot displays the MLflow web interface. At the top, there's a dark blue header with the 'mlflow' logo and navigation tabs for 'Experiments' and 'Models'. Below the header, the breadcrumb 'review_classification > Run 60e8b5c837c94a84b7c689d78d80f158' is visible. The main title of the run is 'Run 60e8b5c837c94a84b7c689d78d80f158'. Below this, a grid of metadata is shown: Date (2022-04-12 14:23:13), Source (KoBERT_train.py), Git Commit (508aaa41c0f2fd39c1b6b7e6473a48e618afe5ca), User (root), Duration (22.4s), and Status (FINISHED). The Lifecycle Stage is 'active'. A sidebar on the left contains expandable sections: Description (with an 'Edit' link), Parameters (11), Metrics (3), Tags, and Artifacts. The Artifacts section is expanded, showing a list of files: 'bentoml_model_dir.txt' and 'result.csv'. The details for 'bentoml_model_dir.txt' are shown on the right, including its full path, size (63B), and a download icon. The full path is '/root/bentoml/repository/ReviewClassifier/20220412142334_A00329'.

review_classification > Run 60e8b5c837c94a84b7c689d78d80f158



Run 60e8b5c837c94a84b7c689d78d80f158


Date: 2022-04-12 14:23:13 Source:  KoBERT_train.py Git Commit: 508aaa41c0f2fd39c1b6b7e6473a48e618afe5ca

User: root Duration: 22.4s Status: FINISHED

Lifecycle Stage: active

- ▶ Description [Edit](#)
- ▶ Parameters (11)
- ▶ Metrics (3)
- ▶ Tags
- ▼ Artifacts

 bentoml_model_dir.txt  Full Path:file:///data/junghoon/private/bentoml_tutorial/review_classification/mlruns/1/60e8b5c837c94a84b7c689d78d80f158/artifacts/bentoml_model_dir....

 result.csv Size: 63B

/root/bentoml/repository/ReviewClassifier/20220412142334_A00329

- MLflow는 tracking 툴로만 사용.
- BentoML의 모델 파일이 저장된 위치와 버전을 artifacts로 저장.

03. MLflow & BentoML

review_classification

- 경로 : /bentoml_tutorial/review_classification/NB_train.py
- 100번째 줄의 ReviewClassifier에서 BentoML을 세팅한다.
- 101~103번째 줄의 pack 부분에서 사용이 되는 모델과 관련 도구들을 세팅한다.
- 104번째 줄의 save를 통해 저장한다.
- 106번째 줄에서 bentoml의 모델이 저장되는 경로를 mlflow에 등록해준다.

```
99         print("6. packing")
100        review_classifier = ReviewClassifier()
101        review_classifier.pack('model', model)
102        review_classifier.pack('tokenizer', vectorizer)
103        review_classifier.pack('pos_tagging', pos_tagging)
104        saved_path = review_classifier.save()
105
106        with open("bentoml_model_dir.txt", "w") as f:
107            f.write(saved_path)
108        mlflow.log_artifact("result.csv")
109        mlflow.log_artifact("bentoml_model_dir.txt")
```

03. MLflow & BentoML

review_classification

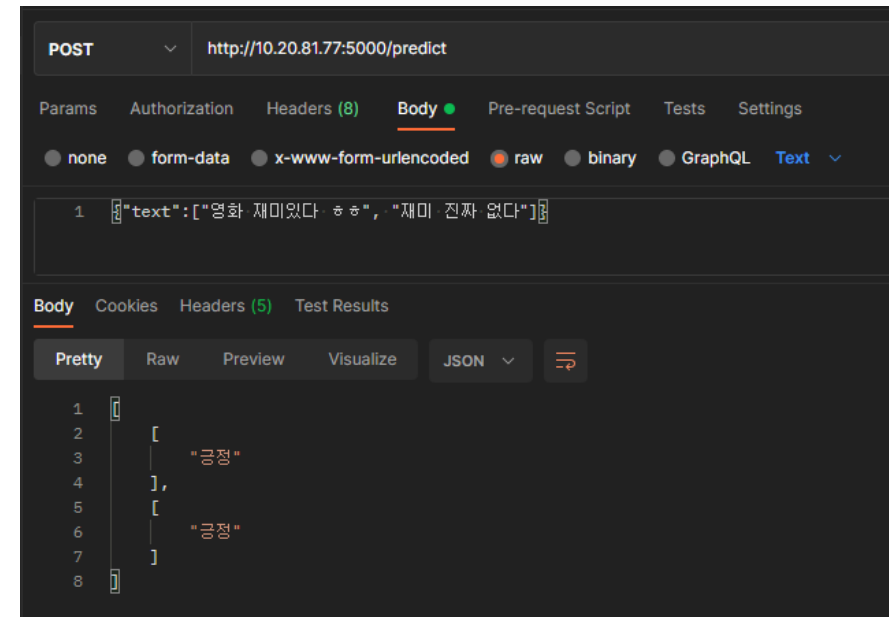
- 경로 : /bentoml_tutorial/review_classification/NB_service.py
- 10번째 줄에서 모델과 토큰라이저 형태소 분석 함수를 정의한다. 이 부분이 사전에 미리 정의되어야 앞에서 pack 할 수 있다.
- 14번째 줄에서 입력 받은 텍스트를 변수에 넣습니다.
- 16~18번째 줄에서 텍스트를 미리 패키징한 모듈을 불러와 차례대로 입력해 예측값을 추론하는 과정입니다.

```
1 from typing import List
2
3 from bentoml import artifacts, api, BentoService
4 from bentoml.adapters import JsonInput
5 from bentoml.frameworks.sklearn import SklearnModelArtifact
6 from bentoml.service.artifacts.common import PickleArtifact
7 from bentoml.types import JsonSerializable
8
9
10 @artifacts([SklearnModelArtifact('model'), PickleArtifact('tokenizer'), PickleArtifact('pos_tagging')])
11 class ReviewClassifier(BentoService):
12     @api(input=JsonInput(), batch=True)
13     def predict(self, parsed_jsons: List[JsonSerializable]):
14         input_texts = parsed_jsons[0]['text']
15
16         text = self.artifacts.pos_tagging(input_texts)
17         text = self.artifacts.tokenizer.transform(text)
18         pred_y = self.artifacts.model.predict(text)
19
20         return [pred_y]
```

03. MLflow & BentoML

review_classification

- API 서버 띄우기 : `bentoml serve ReviewClassifier:latest --port 1000`
- 요청 보내 결과 확인

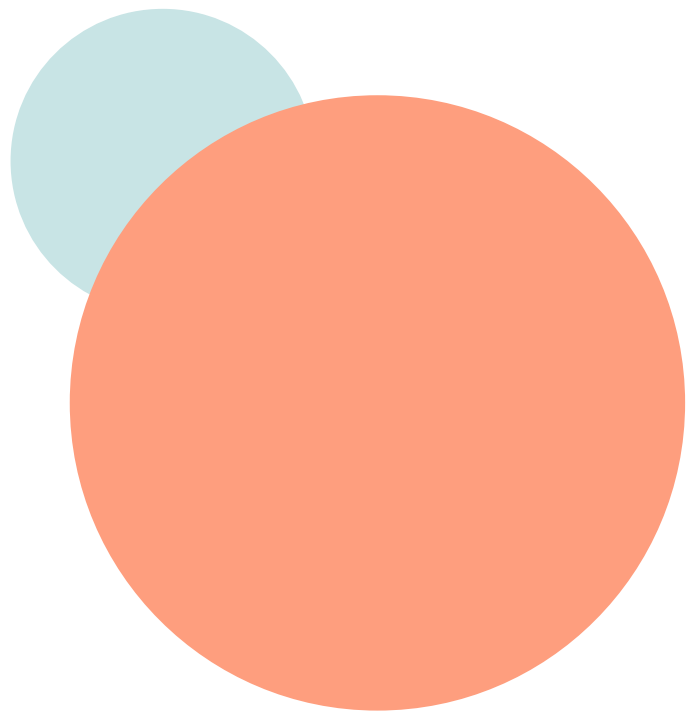


03. MLflow & BentoML

review_classification

- KoBERT 실습
- 경로 : /bentoml_tutorial/review_classification/KoBERT_train.py
- 파일로 저장하는 방법과 비교하면 훨씬 간단한 것을 알 수 있다.

```
9
10 if __name__ == "__main__":
11     # Setting Directory
12     model_dir = "model_kobert.pt"
13     tokenizer_dir = "tokenizer_kobert.pickle"
14     label_enc_dir = "label_enc_kobert.pickle"
15
16     # Setting Parameter
17     device = torch.device("cuda")
18
19     # Hyper Parameter
20     max_len = 10
21     dr_rate = 0.5
22
23     # Flow
24     input_text = ["시간버렸다", "와 감동스러운 영화 그 자체~!", "영화 대박 엄청 재미있어요"]
25
26     tokenizer = pickle.load(open(tokenizer_dir, 'rb'))
27     label_enc = pickle.load(open(label_enc_dir, 'rb'))
28
29     test_set = BERTDataset(input_text, [0]*len(input_text), tokenizer, max_len, True, False)
30     test_loader = torch.utils.data.DataLoader(test_set, batch_size=1, shuffle=False)
31
32     kobert_model, vocab = get_pytorch_kobert_model()
33     model = BERTClassifier(kobert_model, num_classes=len(label_enc.classes_), dr_rate=dr_rate).to(device)
34     model.load_state_dict(torch.load(model_dir))
35
36     for batch_id, (token_ids, valid_length, segment_ids, label) in enumerate(test_loader):
37         output = model(token_ids.long().to(device), valid_length, segment_ids.long().to(device))
38         _, output = torch.max(output, 1)
39         output = output.tolist()
40         output = label_enc.inverse_transform(output)
41         print(output)
```



일시 2022년 04월 23일
장소 Google Meet
발표 이정훈

THANK
YOU