Azure MLOps 실습자료

Azure ML 학습자료

기계 학습 모델 만들기

Azure Machine Learning을 사용하여 기계 학습 솔루션 빌드 및 운영

MLflow 및 Azure Machine Learning을 사용하여 ML 모델 추적

Azure ML 파이프라인 기초 실습

Microsoft Azure Machine Learning Studio

- 기계 학습 리소스 생성
  - 기계 학습 작업 영역 만들기
    - 리소스 그룹 : azure-mlops
    - 이름 : mlops-pipeline
    - 지역 : 미국 동부
    - 컨테이너 레지스트리 : mlopscd (고유한 이름, 도메인)
    - 검토+만들기
- Studio 시작하기
  - 실제 작업을 진행할 인스턴스 생성
  - Compute - Compute instances - [+New]
    - Standard_DS3_v2 선택
- Notebooks 시작
- 데이터 업로드
  - Notebook에서 data 폴더 생성 → winequality-red.csv 업로드
    https://www.kaggle.com/datasets/uciml/red-wine-quality-cortez-et-al-2009?resource=download
- 모델 학습

  ▼ train.ipynb 생성 (파일 경로 수정 필요)
  와인 퀄리티 데이터를 트레이닝
  Compute, Python 실행중(초록색)인지 오른쪽 상단 확인

```
import logging
from pathlib import Path

import pandas as pd
from joblib import dump
from sklearn import preprocessing
from sklearn.experimental import enable_hist_gradient_boosting  # noqa
from sklearn.ensemble import HistGradientBoostingRegressor, RandomForestRegressor
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split

logger = logging.getLogger(__name__)

def prepare_dataset(test_size=0.2, random_seed=1):
    dataset = pd.read_csv(
        "./data/winequality-red.csv",
        delimiter=",",
    )
    dataset = dataset.rename(columns=lambda x: x.lower().replace(" ", "_"))
    train_df, test_df = train_test_split(dataset, test_size=test_size, random_state=random_seed)
    return {"train": train_df, "test": test_df}

def train():
    logger.info("Preparing dataset...")
    dataset = prepare_dataset()
    train_df = dataset["train"]
    test_df = dataset["test"]

    # separate features from target
    y_train = train_df["quality"]
    X_train = train_df.drop("quality", axis=1)
    y_test = test_df["quality"]
    X_test = test_df.drop("quality", axis=1)

    logger.info("Training model...")
    scaler = preprocessing.StandardScaler().fit(X_train)
    X_train = scaler.transform(X_train)
    X_test = scaler.transform(X_test)
    model = RandomForestRegressor(max_depth=30).fit(X_train, y_train)
    # model = HistGradientBoostingRegressor(max_iter=50).fit(X_train, y_train)

    y_pred = model.predict(X_test)
    error = mean_squared_error(y_test, y_pred)
    logger.info(f"Test MSE: {error}")

    logger.info("Saving artifacts...")
    Path("artifacts").mkdir(exist_ok=True)
    dump(model, "artifacts/wine_model.joblib")
    dump(scaler, "artifacts/wine_scaler.joblib")

if __name__ == "__main__":
    logging.basicConfig(level=logging.INFO)
    train()
```

- 모델 등록
  왼쪽 메뉴에서 Models 확인, 다양한 모델 프레임워크로 모델을 등록할 수 있음.
  이번 예제에서는 코드를 통해 모델을 업로드

  ▼ register_model.ipynb 생성
  리소스그룹에서 subscription(구독) ID 복사
  생성 후 Models에서 wine_scaler, wine_model 확인 가능

```
from azureml.core import Workspace
from azureml.core.model import Model
```

```
# 작업중인 ML Studio 관련 정보
ws = Workspace(subscription_id="<your subscription id>",
               resource_group="azure-mlops",
               workspace_name="mlops-pipeline")

dname = "wine"
model = Model.register(
    ws,
    model_name=f"{dname}_model",
    model_path=f"./artifacts/{dname}_model.joblib",
    tags={'area': f"{dname}", 'type': "regression"},
    description=f"RandomForest regression model to predict {dname} quality"
    )

# scaler도 저장
scaler = Model.register(
    ws,
    model_name=f"{dname}_scaler",
    model_path=f"./artifacts/{dname}_scaler.joblib")

print(f"wine_model - name: {model.name}, id: {model.id}, ver: {model.version}")
print(f"wine_scaler - name: {scaler.name}, id: {scaler.id}, ver: {scaler.version}")
```

- 저장된 모델 로드 테스트

  ▼ load_model.ipynb 생성

  리소스그룹에서 구독ID 복사
  모델 로드 후 notebook에서 wine_scaler, wine_scaler 실행해서 결과 확인

```
from azureml.core import Workspace
from azureml.core.model import Model

ws = Workspace(subscription_id="<your subscription id>",
               resource_group="azure-mlops",
               workspace_name="mlops-pipeline")
wine_model = Model(
    ws,
    'wine_model',
    version=1)
wine_scaler = Model(
    ws,
    'wine_scaler',
    version=1
)
```

Azure ML 파이프라인 예제 실습

- azure_ml_example.ipynb 만들기
- 작업공간(Workspace) 설정 가져오기

```
from azureml.core import Workspace
ws = Workspace.from_config()
print('Workspace name: ' + ws.name,
      'Azure region: ' + ws.location,
      'Subscription id: ' + ws.subscription_id,
      'Resource group: ' + ws.resource_group, sep='\n')
```

- 실험공간 생성
  Jobs(Experiment) 메뉴에서 확인

```
from azureml.core import Experiment
experiment = Experiment(workspace=ws, name="diabetes-experiment")
```

- 데이터 준비

```
# Azure ML에 opendatasets를 내장. Diabetes 데이터 사용
from azureml.opendatasets import Diabetes
from sklearn.model_selection import train_test_split

# pandas_dataframe을 가져와서 결측값(NA,Not Available) 제거
x_df = Diabetes.get_tabular_dataset().to_pandas_dataframe().dropna()
y_df = x_df.pop("Y")

X_train, X_test, y_train, y_test = train_test_split(x_df, y_df, test_size=0.2, random_state=66)

print(X_train)
```

- 모델 훈련하면서 로그 남기고 모델 파일 업로드

```
from sklearn.linear_model import Ridge
from sklearn.metrics import mean_squared_error
from sklearn.externals import joblib
import math

# alphas에 따라 모델링이 달라지며 미리 값을 선언
alphas = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]

# alpha값 하나씩 증가
for alpha in alphas:
    run = experiment.start_logging()
    run.log("alpha_value", alpha)

    # 모델링이후에 rmse 성능지표
    model = Ridge(alpha=alpha)
    model.fit(X=X_train, y=y_train)
    y_pred = model.predict(X=X_test)
    rmse = math.sqrt(mean_squared_error(y_true=y_test, y_pred=y_pred))
    run.log("rmse", rmse)

    # outputs에 pkl파일로 저장
    model_name = "model_alpha_" + str(alpha) + ".pkl"
    filename = "outputs/" + model_name

    joblib.dump(value=model, filename=filename)
    run.upload_file(name=model_name, path_or_stream=filename)
    run.complete()
    print(f"{alpha} exp completed")
```

- Studio 에서 실험 결과 확인 및 모델 다운로드
  레포트 페이지로 가는 링크 생성
  - output 폴더 확인
  - Jobs 확인
    Display name. 선택, 모두 선택
    확대. 자체를 레포트 페이지로 활용 가능

알파값이 커지는 시간 순서에 따라 알파 벨류가 높아짐
rmse도 계속 높아짐. 베스트 모델이 첫번째 모델이 될듯..

- 노트북에서 확인

```
experiment
```

- Best model 탐색 후 다운로드

```
minimum_rmse_runid = None
minimum_rmse = None

for run in experiment.get_runs():
    run_metrics = run.get_metrics()
    run_details = run.get_details()
    # each logged metric becomes a key in this returned dict
    run_rmse = run_metrics["rmse"]
    run_id = run_details["runId"]

    if minimum_rmse is None:
        minimum_rmse = run_rmse
        minimum_rmse_runid = run_id
    else:
        if run_rmse < minimum_rmse:
            minimum_rmse = run_rmse
            minimum_rmse_runid = run_id

# id 출력. 차트에서 구분 안됨
print("Best run_id: " + minimum_rmse_runid)
print("Best run_id rmse: " + str(minimum_rmse))
```

```
# 파일명 출력
from azureml.core import Run
best_run = Run(experiment=experiment, run_id=minimum_rmse_runid)
print(best_run.get_file_names())
```

```
# 다운로드(notebook files)
best_run.download_file(name=str(best_run.get_file_names()[0]))
```

- DataStore 에 Input/Output 데이터셋 등록
  나중에 활용하도록 x와 y를 numpy로 변환해서 csv파일로 데이터 스토어에 저장
  Datastores에 workspaceblobstore (Default)로 Blob Storage에 저장됨
  Browse에서 확인 가능

```
import numpy as np
from azureml.core import Dataset

np.savetxt('features.csv', X_train, delimiter=',')
np.savetxt('labels.csv', y_train, delimiter=',')

datastore = ws.get_default_datastore() # Workspace의 datastore get
datastore.upload_files(files=['./features.csv', './labels.csv'],
                       target_path='diabetes-experiment/',
                       overwrite=True)
```

```
input_dataset = Dataset.Tabular.from_delimited_files(path=[(datastore, 'diabetes-experiment/features.csv')])
output_dataset = Dataset.Tabular.from_delimited_files(path=[(datastore, 'diabetes-experiment/labels.csv')])
```

- Best model 등록
  models에 생성.
  Version, Artifacts 확인 가능

```
import sklearn

from azureml.core import Model
from azureml.core.resource_configuration import ResourceConfiguration

model = Model.register(workspace=ws,
                       model_name='diabetes-experiment-model',
                       model_path=f"./{str(best_run.get_file_names()[0])}", # 첫번째 파일로 선택
                       model_framework=Model.Framework.SCIKITLEARN,  # SCIKITLEARN 사용
                       model_framework_version=sklearn.__version__,
                       sample_input_dataset=input_dataset, # 데이터셋 지정
                       sample_output_dataset=output_dataset,
                       resource_configuration=ResourceConfiguration(cpu=1, memory_in_gb=0.5), # 리소스 환경 설정
                       description='Ridge regression model to predict diabetes progression.',
                       tags={'area': 'diabetes', 'type': 'regression'})

print('Name:', model.name)
print('Version:', model.version)
```

- 모델 배포
  배포하는데 시간 좀 걸릴 수 있음(약 8분)
  Models에 배포확인. Endpoint
  Deployment logs 확인

```
service_name = 'diabetes-service'

service = Model.deploy(ws, service_name, [model], overwrite=True)
service.wait_for_deployment(show_output=True) # output 나올때까지 대기
```

- 배포 서비스 테스트
  테스트 2가지 방법
  - 노트북

    ```
    import json

    # 이미 로딩되어 있는 값을 json형식으로 변경
    input_payload = json.dumps({
        'data': X_train[0:2].values.tolist(),
        'method': 'predict'
    })

    # service.run에 endpoint에 담아서 output
    output = service.run(input_payload)

    print(output)
    ```

  - [Models]-[Endpoints]-[서비스명]-[Test]

- 서비스 삭제(현재 서비스)
  서비스는 추가 비용 발생

```
service.delete()
```