



목 차

I. 개요

1. 소개

II. 프로그래밍 가이드 문서

0_local_iris_classification_requirement_sklearn

0_local_iris_classification_sklearn.ipynb

1_local_platform_iris_classification_requirement_sklearn

1_1_local_platform_iris_classification_preprocess_train_sklearn.ipynb

1_2_local_platform_iris_classification_inference_sklearn.ipynb

2_1_1_platform_iris_classification_preprocess_sklearn.py

2_1_2_platform_iris_classification_preprocess_sklearn_sub.py

2_2_1_platform_iris_classification_train_sklearn.py

2_2_2_platform_iris_classification_train_sklearn_sub.py

III. 수행 절차

01) T3Q.cep_데이터수집 파이프라인_iris_classification_sklearn

02) T3Q.cep_데이터변환 파이프라인_iris_classification_sklearn

03) T3Q.dl_프로젝트 설정_실행환경 관리_iris_classification_sklearn

04) T3Q.dl_프로젝트 설정_전처리모듈 관리_iris_classification_sklearn

05) T3Q.dl_프로젝트 설정_학습 알고리즘 관리

_iris_classification_sklearn

06) T3Q.dl_학습플랫폼_데이터셋 관리_iris_classification_sklearn

07) T3Q.dl_학습플랫폼_전처리모델 설계_iris_classification_sklearn

08) T3Q.dl_학습플랫폼_전처리모델 관리_iris_classification_sklearn

09) T3Q.dl_학습플랫폼_학습모델 설계_iris_classification_sklearn

10) T3Q.dl_학습플랫폼_학습모델 관리_iris_classification_sklearn

11) T3Q.dl_추론플랫폼_추론모델 관리_iris_classification_sklearn

12) T3Q.dl_추론플랫폼_추론API관리_iris_classification_sklearn

13) T3Q.cep_실시간 추론 파이프라인_iris_classification_sklearn

1. 소개

- 아이리스 분류(IRIS CLASSIFICATION)
- 사이킷런을 이용한 아이리스 품종 분류 워크플로우(workflow)
 - 아이리스 데이터 파일(iris.csv) 열기
 - 학습을 위한 데이터 전처리(훈련 데이터, 테스트 데이터)
 - SVM을 이용한 모델 생성 및 훈련
 - 아이리스 품종 추론
- 아이리스 데이터 세트
 - 아이리스 데이터 세트에는 150개의 데이터 세트가 저장되어 있다.
 - 아이리스 데이터 세트는 꽃받침 길이, 꽃받침 폭, 꽃잎 길이, 꽃잎 폭 총 5개의 컬럼으로 구성되어 있다.

0_local_iris_classification_requirement_sklearn

- 로컬에 설치 되어야 할 패키지 버전
 - ✓ numpy 1.22.3
 - ✓ pandas 1.4.2
 - ✓ scikit-learn 1.1.1

0_local_iris_classification_sklearn.ipynb

- 로컬 개발 코드
 - ✓ 로컬에서 주피터 노트북(Jupyter Notebook), 주피터 랩(JupyterLab) 또는 파이썬(Python)을 이용한다.
 - ✓ 사이킷 런(scikit-learn), 텐서플로우(tensorflow), 파이토치(pytorch)를 사용하여 딥러닝 프로그램을 개발한다.
 - ✓ 파일명: 0_local_iris_classification_sklearn.ipynb
- 로컬 개발 워크플로우(workflow)
 - ✓ 로컬 개발 워크플로우를 다음의 4단계로 분리한다.
 - 데이터 세트 준비(Data Setup)
 - 로컬 저장소에서 전처리 및 학습에 필요한 학습 데이터 세트를 준비한다.
 - 데이터 전처리(Data Preprocessing)
 - 데이터 세트의 분석 및 정규화(Normalization)등의 전처리를 수행한다.
 - 데이터를 모델 학습에 사용할 수 있도록 가공한다.
 - 추론과정에서 필요한 경우, 데이터 전처리에 사용된 객체를 meta_data 폴더 아래에 저장한다.
 - 학습 모델 훈련(Train Model)
 - 데이터를 훈련에 사용할 수 있도록 가공한 뒤에 학습 모델을 구성한다.
 - 학습 모델을 준비된 데이터 세트로 훈련시킨다.
 - 정확도(Accuracy)나 손실(Loss)등 학습 모델의 성능을 검증한다.
 - 학습 모델의 성능 검증 후, 학습 모델을 배포한다.
 - 배포할 학습 모델을 meta_data 폴더 아래에 저장한다.
 - 추론(Inference)
 - 저장된 전처리 객체나 학습 모델 객체를 준비한다.
 - 추론에 필요한 테스트 데이터 세트를 준비한다.
 - 배포된 학습 모델을 통해 테스트 데이터에 대한 추론을 진행한다.

T3Q

5

0_local_iris_classification_sklearn.ipynb

#Imports

```
import pandas as pd
import sklearn.svm as svm
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

1. 데이터 세트 준비(Data Setup)

```
# local에 저장된 아이리스 데이터 csv 파일을 읽어온다.
df = pd.read_csv('iris.csv')
```

2. 데이터 전처리(Data Preprocessing)

iris.csv 는 첫 번째부터 네 번째 열이 아이리스 특징을 나타내는 레이블이고 나머지 1개 열이 아이리스 품종을 나타내는 csv 파일이다.

학습:80%, 테스트:20%

```
x_train, x_test, y_train, y_test = train_test_split(df.iloc[:, :-1], df.iloc[:, -1], test_size=0.2, random_state=777)
```

3. 학습 모델 훈련(Train Model)

svm 형태로 모델을 만들고 학습한다.

```
svm_clf = svm.SVC(kernel = 'linear').fit(x_train, y_train)
```

정확도를 평가한다.

```
print(accuracy_score(y_test, svm_clf.predict(x_test)))
```

4. 추론(Inference)

결과를 예측한다.

```
svm_clf.predict(x_test)
```


1_local_platform_iris_classification_requirement_sklearn

- 설치 되어야 할 패키지 버전
 - ✓ numpy 1.22.3
 - ✓ pandas 1.4.2
 - ✓ scikit-learn 1.1.1
 - ✓ pickleshare 0.7.5

1_1_local_platform_iris_classification_preprocess_train_sklearn.ipynb

◆ 플랫폼 업로드를 쉽게하기 위한 로컬 개발 코드

- ✓ T3Q.ai(T3Q.cep + T3Q.dl): 빅데이터/인공지능 통합 플랫폼
 - ✓ 플랫폼 업로드를 쉽게하기 위하여 로컬에서 2개의 파일로 나누어 코드를 개발한다.
 - ✓ 파일 1_1
 - ✓ 전처리와 학습모델을 개발하는 코드, 전처리 객체나 성능이 검증된 학습모델 객체를 저장한다.
 - ✓ 파일명 : 1_1_local_platform_iris_classification_preprocess_train_sklearn.ipynb
 - ✓ 파일 1_2
 - ✓ 학습모델을 통해 추론(예측)을 개발하는 코드, 전처리 객체나 학습모델 객체를 불러와서 추론을 수행한다.
 - ✓ 파일명 : 1_2_local_platform_iris_classification_inference_sklearn.ipynb
- 전처리 객체 또는 학습모델 객체
 - ✓ 전처리 객체나 학습모델 객체는 meta_data 폴더 아래에 저장한다.
 - 데이터 세트(학습 데이터/테스트 데이터)
 - ✓ 학습과 테스트에 사용되는 데이터를 나누어 관리한다.
 - ✓ 학습 데이터: dataset 폴더 아래에 저장하거나 dataset.zip 파일 형태로 저장한다.
 - ✓ 테스트 데이터: test_dataset 폴더 아래에 저장하거나 test_dataset.zip 파일 형태로 저장한다.

◆ 로컬 개발 워크플로우(workflow)

- ✓ 로컬 개발 워크플로우를 다음의 4단계로 분리한다.

1.데이터 세트 준비(Data Setup)

- ✓ 로컬 저장소에서 전처리 및 학습에 필요한 학습 데이터 세트를 준비한다.

2.데이터 전처리(Data Preprocessing)

- ✓ 데이터 세트의 분석 및 정규화(Normalization)등의 전처리를 수행한다.
- ✓ 데이터를 모델 학습에 사용할 수 있도록 가공한다.
- ✓ 추론과정에서 필요한 경우, 데이터 전처리에 사용된 객체를 meta_data 폴더 아래에 저장한다.

3.학습 모델 훈련(Train Model)

- ✓ 데이터를 훈련에 사용할 수 있도록 가공한 뒤에 학습 모델을 구성한다.
- ✓ 학습 모델을 준비된 데이터 세트로 훈련시킨다.
- ✓ 정확도(Accuracy)나 손실(Loss)등 학습 모델의 성능을 검증한다.
- ✓ 학습 모델의 성능 검증 후, 학습 모델을 배포한다.
- ✓ 배포할 학습 모델을 meta_data 폴더 아래에 저장한다.

4.추론(Inference)

- ✓ 저장된 전처리 객체나 학습 모델 객체를 준비한다.
- ✓ 추론에 필요한 테스트 데이터 세트를 준비한다.
- ✓ 배포된 학습 모델을 통해 테스트 데이터에 대한 추론을 진행한다.

#Imports

import pandas as pd

import sklearn.svm as svm

from sklearn.model_selection import train_test_split

from sklearn.metrics import accuracy_score

import pickle

1. 데이터 세트 준비(Data Setup)

local에 저장된 아이리스 데이터 csv 파일을 읽어온다.

df = pd.read_csv('iris.csv')

2. 데이터 전처리(Data Preprocessing)

iris.csv 는 첫 번째부터 네 번째 열이 아이리스 특징을 나타내는 레이블이고 나머지 1개 열이 아이리스 품종을 나타내는 csv 파일이다.

학습:80%, 테스트:20%

```
x_train, x_test, y_train, y_test = train_test_split(df.iloc[:, :-1], df.iloc[:, -1],  
test_size=0.2, random_state=777)
```

3. 학습 모델 훈련(Train Model)

svm 형태로 모델을 만들고 학습한다.

```
svm_clf = svm.SVC(kernel = 'linear').fit(x_train, y_train)
```

정확도를 평가한다.

```
print(accuracy_score(y_test, svm_clf.predict(x_test)))
```

학습 모델 저장

학습 모델을 저장한다.

```
with open('meta_data/iris_model.p', 'wb') as f:  
    pickle.dump(svm_clf, f)
```

1_2_local_platform_iris_classification_inference_sklearn.ipynb

- ◆ 플랫폼 업로드를 쉽게하기 위한 로컬 개발 코드
 - T3Q.ai(T3Q.cep + T3Q.di): 빅데이터/인공지능 통합 플랫폼
 - 플랫폼 업로드를 쉽게하기 위하여 로컬에서 2개의 파일로 나누어 코드를 개발한다.
 - 파일 1_1
 - 전처리와 학습모델을 개발하는 코드, 전처리 객체나 성능이 검증된 학습모델 객체를 저장한다.
 - 파일명 : 1_1_local_platform_iris_classification_preprocess_train_sklearn.ipynb
 - 파일 1_2
 - 학습모델을 통해 추론(예측)을 개발하는 코드, 전처리 객체나 학습모델 객체를 불러와서 추론을 수행한다.
 - 파일명 : 1_2_local_platform_iris_classification_inference_sklearn.ipynb
- 전처리 객체 또는 학습모델 객체 : 전처리 객체나 학습모델 객체는 meta_data 폴더 아래에 저장한다.
- 데이터 세트(학습 데이터/테스트 데이터)
 - 학습과 테스트에 사용되는 데이터를 나누어 관리한다.
 - 학습 데이터: dataset 폴더 아래에 저장하거나 dataset.zip 파일 형태로 저장한다.
 - 테스트 데이터: test_dataset 폴더 아래에 저장하거나 test_dataset.zip 파일 형태로 저장한다.
- 로컬 개발 워크플로우(workflow)
 - 로컬 개발 워크플로우를 다음의 4단계로 분리한다.
 - 1.데이터 세트 준비(Data Setup)
 - 로컬 저장소에서 전처리 및 학습에 필요한 학습 데이터 세트를 준비한다.
 - 2.데이터 전처리(Data Preprocessing)
 - 데이터 세트의 분석 및 정규화(Normalization)등의 전처리를 수행한다.
 - 데이터를 모델 학습에 사용할 수 있도록 가공한다.
 - 추론과정에서 필요한 경우, 데이터 전처리에 사용된 객체를 meta_data 폴더 아래에 저장한다.
 - 3.학습 모델 훈련(Train Model)
 - 데이터를 훈련에 사용할 수 있도록 가공한 뒤에 학습 모델을 구성한다.
 - 학습 모델을 준비된 데이터 세트로 훈련시킨다.
 - 정확도(Accuracy)나 손실(Loss) 등 학습 모델의 성능을 검증한다.
 - 학습 모델의 성능 검증 후, 학습 모델을 배포한다.
 - 배포할 학습 모델을 meta_data 폴더 아래에 저장한다.
 - 4.추론(Inference)
 - 저장된 전처리 객체나 학습 모델 객체를 준비한다.
 - 추론에 필요한 테스트 데이터 세트를 준비한다.
 - 배포된 학습 모델을 통해 테스트 데이터에 대한 추론을 진행한다.

```
#Imports
```

```
import pandas as pd
```

```
import pickle
```

4. 추론(Inference)

```
# 추론을 위해 아이리스 데이터 csv 파일을 읽어온다.
```

```
df = pd.read_csv('iris.csv')
```

```
# 학습 모델을 불러온다.
```

```
with open('meta_data/iris_model.p', 'rb') as f:
```

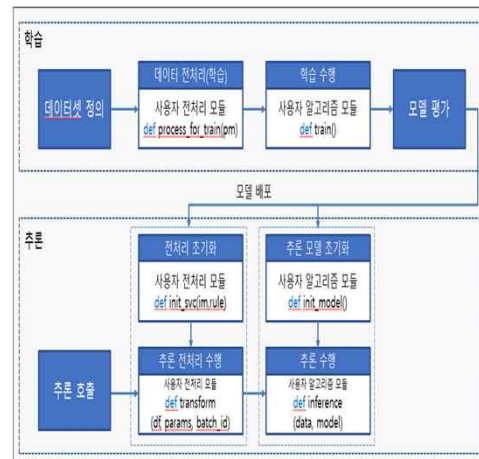
```
    model = pickle.load(f)
```

```
# 결과를 예측한다.
```

```
model.predict(df.iloc[:, :-1])
```

2_platform_process

- 파일명 : iris_classification_preprocess_sklearn.py
 - ✓ from iris_classification_preprocess_sub import exec_process
 - ✓ def process_for_train(pm):
 - exec_process(pm)
 - ✓ def init_svc(im, rule):
 - ✓ def transform(df, params, batch_id):
- 파일명: iris_classification_preprocess_sklearn_sub.py
 - ✓ def exec_process(pm):
- 파일명: iris_classification_train_sk.py
 - ✓ from iris_classification_train_sk_sub import exec_train, exec_inference
 - ✓ def train(tm):
 - exec_train(tm)
 - ✓ def init_svc(im):
 - ✓ def inference(df, params, batch_id):
 - exec_inference(df, params, batch_id)
- 파일명: iris_classification_train_sk_sub.py
 - ✓ def exec_train(tm):
 - ✓ def exec_inference(df, params, batch_id):
 - ✓ def display_result(tm, model, x_test, y_test):
 1. 데이터 세트 준비(Data Setup)
 2. 데이터 전처리(Data Preprocessing)
 3. 학습 모델 훈련(Train Model)
 4. 추론(Inference)



2_1_1_platform_iris_classification_preprocess_sklearn.py

```
# 파일명: iris_classification_preprocess_sk.py
from iris_classification_preprocess_sk_sub import exec_process
import pickle
import logging
def process_for_train(pm):
    logging.info('[hunmin log] the start line of the function [process_for_train]')
    exec_process(pm)
    logging.info('[hunmin log] the end line of the function [process_for_train]')
def init_svc(im, rule):
    meta_path = im.meta_path
    return {"meta_path": meta_path, "rule": rule}
def transform(df, params, batch_id):
    logging.info('[hunmin log] df : {}'.format(df))
    logging.info('[hunmin log] df.shape : {}'.format(df.shape))
    logging.info('[hunmin log] type(df) : {}'.format(type(df)))
    logging.info('[hunmin log] the end line of the function [transform]')
    return df
```

T3Q

16

"""

학습용 데이터 전처리

pm.source_path
는 경로

데이터셋 관리 메뉴에서 저장한 데이터를 가져오

pm.target_path

전처리 완료된 데이터를 저장하는 경로

exec_process(pm) 함수 내부에

pm.source_path에 저장된 데이터를 가져와서

pm.target_path에 데이터를 저장하는 코드를 작성한다. """

"""

"""

추론 서비스 초기화

im.meta_path

전처리 모델 및 메타

데이터를 저장한 경로

"""

"""

추론을 위한 데이터 변환

df

pandas dataframe 형태로 변환되어 받게 된다.

params

init_svc 함수의 return 값을 params 변수로 전달받는다.

params['meta_path'] params['model_path']

"""

2_1_2_platform_iris_classification_preprocess_sklearn_sub.py

```
# 파일명: iris_classification_preprocess_sub.py

import os
import numpy as np
import pandas as pd
import pickle
import logging

def exec_process(pm):
    #####
    # 1_1_local_platform_iris_classification_preprocess_train_sklearn.ipynb 파일의
    # 1.데이터 세트 준비(Data Setup) 부분을 여기에 작성한다.
    #####

    logging.info('[hunmin log] the start line of the function [exec_process]')

    # 데이터 파일이 저장된 경로.
    logging.info('[hunmin log] source_path : {}'.format(pm.source_path))
    # 데이터 파일을 저장할 경로.
    logging.info('[hunmin log] target_path : {}'.format(pm.target_path))
```

T3Q

17

"""

학습용 데이터 전처리

pm.source_path
는 경로

pm.target_path

데이터셋 관리 메뉴에서 저장한 데이터를 가져오

전처리 완료된 데이터를 저장하는 경로

exec_process(pm) 함수 내부에

pm.source_path에 저장된 데이터를 가져와서

pm.target_path에 데이터를 저장하는 코드를 작성한다.

exec_process(pm) 함수 내부에

로컬에서 개발한 파일(1_1_local_platform_iris_classification_preprocess_train_sklearn.ipynb)의

1.데이터 세트 준비(Data Setup) 부분을 작성한다.

"""

```

# 데이터 파일을 가져옴.
df = pd.read_csv(os.path.join(pm.source_path, os.listdir(pm.source_path)[-1]))
# 가져온 데이터 파일을 저장.
df.to_csv(pm.target_path+"/iris.csv", index = False)
list_files_directories(pm.source_path)

# 저장 파일 확인
def list_files_directories(path):
    # Get the list of all files and directories in current working directory
    dir_list = os.listdir(path)
    logging.info('[hunmin log] Files and directories in {} : {}'.format(path, dir_list))
    logging.info('[hunmin log] dir_list : {}'.format(dir_list))
    logging.info('[hunmin log] finish exec_process')

```

"""

1) stats 항목 파악

- 다음과 같이 총 5개(4개의 특징 + 1개의 종)의 항목이 있다는 것을 확인할 수 있다.
- 데이터는 총 150개가 있음을 확인할 수 있다.
- 우리의 목적은 각 스탯의 항목들을 보고, 해당 특징을 가진 꽃이 무슨 꽃(종)인지 예측하는 모델을 구현하는 것이다.
- 이를 위해서 **각 stat 항목이 각 포지션마다 어떤 분포를 가지는지 시각화**하여 전체 데이터 분포를 파악한다.
- 결과를 보고 데이터를 어떻게 전처리하여 활용할지 결정한다.

"""

2) 데이터 시각화

- 데이터를 잘 분류할 수 있는 좋은 모델을 만들기 위해서는 좋은 데이터를 학습시키는 것이 중요하다.
- 좋은 데이터를 학습시키기 위해서는, 데이터가 어떤 특징을 가지고 있는지 파악하고, 잘못된 데이터를 정리하거나, 학습에 도움이 되는 데이터(=Features)를 잘 추출하여 모델에 학습시키는 것이 중요하다.
- 데이터를 시각화하면, 그 특징을 파악하는 것이 훨씬 더 쉽다.

"""

2_2_1_platform_iris_classification_train_sklearn.py

```
# 파일명: iris_classification_train_sk.py
from iris_classification_train_sk_sub import exec_train, exec_inference, exec_display_result
from sklearn import metrics
from sklearn.metrics import accuracy_score
import logging
import pickle
import os

def train(tm):
    logging.info('[hunmin log] the start line of the function [train]')
    exec_train(tm)
    logging.info('[hunmin log] the finish line of the function [train]')
def init_svc(im):
    model = pickle.load(open(os.path.join(im.model_path, 'model.p'), 'rb'))
    return {'model':model}
def inference(df, params, batch_id=1):
    logging.info('[hunmin log] the start line of the function [inference]')
    result = exec_inference(df, params, batch_id)
    logging.info('[hunmin log] the end line of the function [inference]')
    return result
```

파일명: iris_classification_train_sklearn.py

```
from iris_classification_train_sk_sub import exec_train, exec_inference,
exec_display_result
from sklearn import metrics
from sklearn.metrics import accuracy_score
import logging
import pickle
import os
```

```
def train(tm):
```

```
'''
```

train(tm) : 학습에 필요한 필수 함수

labels

tm.features.get('y_value')

data load

tm.load_data_for_cnn() # cnn 데이터

tm.load_data_for_all() # 기타 데이터

ex) (train_id, train_x, train_y), (test_id, test_x, test_y) = tm.load_data_for_cnn()

param

tm.param_info[common_params]

tm.param_info[algo_params]

common_params

nn_type NN 유형

init_method 초기화 방법

opt_method 최적화 방법

learning_rate Learning Rate

dropout_ratio Dropout Ratio

random_seed 랜덤 seed
autosave_p 자동저장 주기
epoch 학습수행횟수
batch_size 배치 사이즈
algo_params
ui 에서 정의

```
save
    tm.save_result_metrics(eval_results)
```

```
eval_results
    step
    predict_y
    actual_y
    test_id
    confusion_matrix
```

```
'''
```

2_2_1_platform_iris_classification_train_sklearn.py

```
# 파일명: iris_classification_train_sk.py
from iris_classification_train_sk_sub import exec_train, exec_inference, exec_display_result
from sklearn import metrics
from sklearn.metrics import accuracy_score
import logging
import pickle
import os

def train(tm):
    logging.info('[hunmin log] the start line of the function [train]')
    exec_train(tm)
    logging.info('[hunmin log] the finish line of the function [train]')
def init_svc(im):
    model = pickle.load(open(os.path.join(im.model_path, 'model.p'), 'rb'))
    return {'model':model}
def inference(df, params, batch_id=1):
    logging.info('[hunmin log] the start line of the function [inference]')
    result = exec_inference(df, params, batch_id)
    logging.info('[hunmin log] the end line of the function [inference]')
    return result
```

```
def display_result(tm, history, model, x_test, y_test):
```

```
'''
```

```
display_result(tm, history, model, x_test, y_test) : 학습 결과 그래프 출력
```

```
'''
```

```
logging.info('[hunmin log] the start line of the function [display_result]')
```

```
exec_display_result(tm, history, model, x_test, y_test)
```

```
logging.info('[hunmin log] the start line of the function [display_result]')
```

```
def init_svc(im):
```

```
'''
```

```
init_svc(im) : 추론 서비스 초기화
```

```
labels
```

```
im.features.get('y_value')
```

```
params
```

```
im.param_info
```

```
nn_info
```

```
im.nn_info
```

```
model_path
```

```
im.get_model_path()
```

```
return 값을 inference 함수에서 params 으로 접근
```

```
ex) return {"svc_config":svc_config, "estimator":estimator}
```

```

'''

model = pickle.load(open(os.path.join(im.model_path, 'model.p'), 'rb'))
return {'model':model}

def inference(df, params, batch_id):
'''
inference : init_svc 의 return 값으로 부터 추론
ex)
svc_config = params['svc_config']
estimator = params['estimator']
test_data = df.iloc[:, 0].values.tolist()

results 를 json 형태로 리턴한다
'''

logging.info('[hunmin log] the start line of the function [inference]')
result = exec_inference(df, params, batch_id)
logging.info('[hunmin log] the end line of the function [inference]')
return result

```

2_2_2_platform_iris_classification_train_sklearn_sub.py

```
# 파일명: iris_classification_train_sk_sub.py
#Imports
import tensorflow as tf
import os
import pandas as pd
import sklearn.svm as svm
import pickle
import logging
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

def exec_train(tm):
    #####
    ## 1. 데이터 세트 준비(Data Setup)
    #####
    logging.info("[hunmin log] train_data_path = {}".format(tm.train_data_path))
    logging.info("[hunmin log] model_path = {}".format(tm.model_path))
    df = pd.read_csv(os.path.join(tm.train_data_path, os.listdir(tm.train_data_path)[-1]))

    #####
    ## 2. 데이터 전처리(Data Preprocessing)
    #####
    # iris.csv 는 첫 번째부터 네 번째 열이 아이리스 특징을 나타내는 레이블이고 나머지 1개 열이 아이리스 품종
    # 을 나타내는 csv 파일이다.
    # 학습:80%, 테스트:20%
    x_train, x_test, y_train, y_test = train_test_split(df.iloc[:, :-1], df.iloc[:, -1], test_size=0.2, random_state=777)
```

```
#####
## 1. 데이터 세트 준비(Data Setup)
#####
"""
#malimg= './dataset/malimg.npz'
#classes_data_path = './dataset/malimg_classes.csv'
malimg = os.path.join(tm.train_data_path, 'dataset/malimg.npz')
classes_data_path = os.path.join(tm.train_data_path, 'dataset/malimg_classes.csv')
"""

# tm.train_data_path      전처리 완료된 데이터 저장 경로
# tm.model_path           학습 수행 후 학습모델을 저장하는 경로

#####
#####
## 2. 데이터 전처리(Data Preprocessing)
#####
#####
# iris.csv 는 첫 번째부터 네 번째 열이 아이리스 특징을 나타내는 레이블이고 나
# 머지 1개 열이 아이리스 품종을 나타내는 csv 파일이다.
# 학습:80%, 테스트:20%
```

2_2_2_platform_iris_classification_train_sklearn_sub.py

```
#####
## 3. 학습 모델 훈련(Train Model)
#####
model = svm.SVC(kernel='linear').fit(x_train, y_train)
accuracy = accuracy_score(y_test, model.predict(x_test))
logging.info("[hunmin log] model = {}".format(model))
logging.info("[hunmin log] accuracy = {}".format(accuracy))
logging.info("[hunmin log] model_build")

#####
## 학습 모델 저장
#####
pickle.dump(model, open(os.path.join(tm.model_path, 'model.p'), 'wb'))
display_result(tm, model, x_test, y_test)
```

```
#####
##
## 3. 학습 모델 훈련(Train Model)
#####
#####
# svm 모델을 활용하여 모델 생성 후 훈련한다.

#####
####
## 학습 모델 저장
#####
####
```


2_2_2_platform_iris_classification_train_sklearn_sub.py

```
#####
## 4. 추론(Inference)
#####
def exec_inference(df, params, batch_id):
    logging.info('[hunmin log] the start line of the function [exec_inference]')

    # 학습 모델 준비
    model = params['model']
    logging.info('[hunmin log] model : {}'.format(model))
    results = {}
    # 결과 예측
    scores = model.predict(df)
    logging.info('[hunmin log] scores : {}'.format(scores))
    results['inference'] = scores.tolist()
    return results
```

```
#####
#####
## 4. 추론(Inference)
#####
#####
# 학습 모델을 준비한다.
# 결과를 예측한다.
```

2_2_2_platform_iris_classification_train_sklearn_sub.py

```

def display_result(tm, model, x_test, y_test):
    #####
    ## 플랫폼 시각화
    #####
    val_accuracy = accuracy_score(y_test, model.predict(x_test))
    val_loss = 0

    eval_results={}
    predict_y = model.predict(x_test).tolist()
    actual_y = y_test.tolist()
    logging.info('[hunmin log] predict_y : {}'.format(predict_y))
    logging.info('[hunmin log] actual_y : {}'.format(actual_y))

    eval_results['predict_y'] = predict_y
    eval_results['actual_y'] = actual_y
    eval_results['loss']= val_loss
    eval_results['accuracy'] = val_accuracy

    # confusion_matrix 계산(eval_results)
    eval_results['confusion_matrix'] = metrics.confusion_matrix(actual_y, predict_y).tolist()
    tm.save_result_metrics(eval_results)

```

```

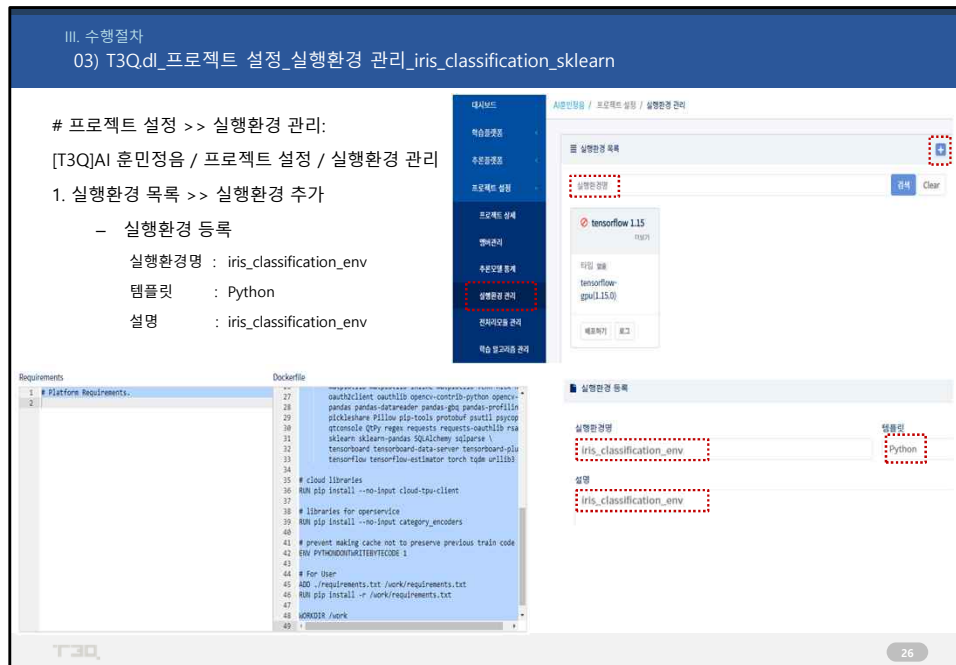
#####
#####
## 플랫폼 시각화
#####
#####
# Accuracy, Loss, Confusion Matrix, Precision/Recall/F1-score

```

III. 수행절차

수행절차 소개

- 01) T3Q.cep_데이터수집 파이프라인_iris_classification_sklearn
- 02) T3Q.cep_데이터변환 파이프라인_iris_classification_sklearn
- 03) T3Q.dl_프로젝트 설정_실행환경 관리_iris_classification_sklearn
- 04) T3Q.dl_프로젝트 설정_전처리모델 관리_iris_classification_sklearn
- 05) T3Q.dl_프로젝트 설정_학습 알고리즘 관리_iris_classification_sklearn
- 06) T3Q.dl_학습플랫폼_데이터셋 관리_iris_classification_sklearn
- 07) T3Q.dl_학습플랫폼_전처리모델 설계_iris_classification_sklearn
- 08) T3Q.dl_학습플랫폼_전처리모델 관리_iris_classification_sklearn
- 09) T3Q.dl_학습플랫폼_학습모델 설계_iris_classification_sklearn
- 10) T3Q.dl_학습플랫폼_학습모델 관리_iris_classification_sklearn
- 11) T3Q.dl_추론플랫폼_추론모델 관리_iris_classification_sklearn
- 12) T3Q.dl_추론플랫폼_추론API관리_iris_classification_sklearn
- 13) T3Q.cep_실시간 추론 파이프라인_iris_classification_sklearn



실행환경 추가 내용 및 절차

1) Requirements

User Requirements.

최소한의 패키지만 등록합니다 !!!

2) Dockerfile

FROM ubuntu:20.04

ARG DEBIAN_FRONTEND=noninteractive

RUN apt-get update && apt-get install -y wget

python3.8

python3-pip

python3-dev

python3.8-dev

postgresql

libpq-dev

RUN update-alternatives --install /usr/bin/python3 python3

/usr/bin/python3.8 1

RUN pip3 install --upgrade pip

libraries for operservice

RUN pip install --no-input kubernetes pygresql pyjwt pyarrow pandas

flask flask-sqlalchemy flask-cors flask-bcrypt flask-migrate flask-restful flask-

rest-jsonapi

tensorflow

2) Dockerfile-계속

```
# generic libraries
RUN pip install --no-input beautifulsoup4 bs4 community contextlib2 gensim
holidays html5lib ₩
ipykernel ipython ipython-genutils ipython-sql ₩
jupyter jupyter-client jupyter-console jupyter-core jupyterlab-pygments
jupyterlab-widgets ₩
kaggle keras Keras-Preprocessing keras-vis korean-lunar-calendar
LunarCalendar ₩
matplotlib matplotlib-inline matplotlib-venn nltk notebook numpy nvidia-ml-
py3 ₩
oauth2client oauthlib opencv-contrib-python opencv-python openpyxl ₩
pandas pandas-datareader pandas-gbq pandas-profiling pathlib ₩
pickleshare Pillow pip-tools protobuf psutil pycopg2 PyYAML ₩
qtconsole QtPy regex requests requests-oauthlib rsa seaborn ₩
sklearn sklearn-pandas SQLAlchemy sqlparse ₩
tensorboard tensorboard-data-server tensorboard-plugin-wit ₩
tensorflow tensorflow-estimator torch tqdm urllib3 xgboost xlrd xlwt
# cloud libraries
RUN pip install --no-input cloud-tpu-client
# libraries for operservice
RUN pip install --no-input category_encoders
# prevent making cache not to preserve previous train code
ENV PYTHONDONTWRITEBYTECODE 1
# For User
ADD ./requirements.txt /work/requirements.txt
RUN pip install -r /work/requirements.txt
WORKDIR /work
# libraries for operservice
RUN pip install --no-input category_encoders
# prevent making cache not to preserve previous train code
ENV PYTHONDONTWRITEBYTECODE 1
# For User
```

≡ 실행환경 목록

실행환경명

binary_classification_env

더보기

타입 Python

binary_classification_env

배포하기

로그

txt

vorl

==

|장]

로그 확인

2022-06-22 05:07:22,405 [INFO] root: ["stream": "update-alternatives: using /usr/bin/g++ to provide /usr/bin/c++ (c++) in auto mode|update-alternatives: warning: skip creation of /usr/share/man/man1/c++.1.gz because associated file /usr/share/man/man1/g++.1.gz (of link group c++) doesn't exist"]

2022-06-22 05:07:22,413 [INFO] root: ["stream": "Setting up build-essential (12.8ubuntu1.1) ..."]

2022-06-22 05:07:22,413 [INFO] root: ["stream": "Setting up libalgorithm-diff-xs-perl (0.04-6) ..."]

2022-06-22 05:07:22,416 [INFO] root: ["stream": "Setting up libalgorithm-merge-perl (0.08-3) ..."]

2022-06-22 05:07:22,419 [INFO] root: ["stream": "Setting up libpython3-dev-amd64 (3.8.2-0ubuntu2) ..."]

2022-06-22 05:07:22,424 [INFO] root: ["stream": "Setting up liblapack-2.4-2amd64 (2.4.49+dfsg-2ubuntu1.9) ..."]

2022-06-22 05:07:22,427 [INFO] root: ["stream": "Setting up dirnogr (2.2.19-3ubuntu2.1) ..."]

2022-06-22 05:07:22,544 [INFO] root: ["stream": "Setting up python3-dev (3.8.2-0ubuntu2) ..."]

III. 수행절차
04) T3Qdl 프로젝트 설정_전처리 모듈 관리_iris_classification_sklearn

- 프로젝트 설정 / 전처리모듈관리
 - 전처리모듈 관리>> + [전처리 모듈 추가] 실행
 - 전처리모듈 등록
 - ① 기본정보
 - 전처리명: iris_classification_premodule
 - 실행환경 선택 : iris_classification_env
 - GPU지원 : GPU 지원(해제)
 - ② 입력 형태: object
 - ③ 출력 형태: default
 - ④ 파라미터
 - ⑤ 소스 코드 : T3Qai_platform_iris_classification_preprocess_sk.zip [파일업로드]
 - iris_classification_preprocess_sk.py (실행모듈 선택)
 - iris_classification_preprocess_sk_sub.py

T3Q

28

전처리모듈 등록

① 기본정보

전처리명: iris_classification_premodule

실행환경 선택 : iris_classification_env

GPU지원 : GPU 지원(해제)

② 입력 형태: object

③ 출력 형태: default

전처리모듈 등록

기본 정보

binary_classification_premodul

binary_classification_env

ON GPU 지원

입력 형태

file

출력 형태

default

파라미터

이름을 입력해주세요

값을 입력해주세요