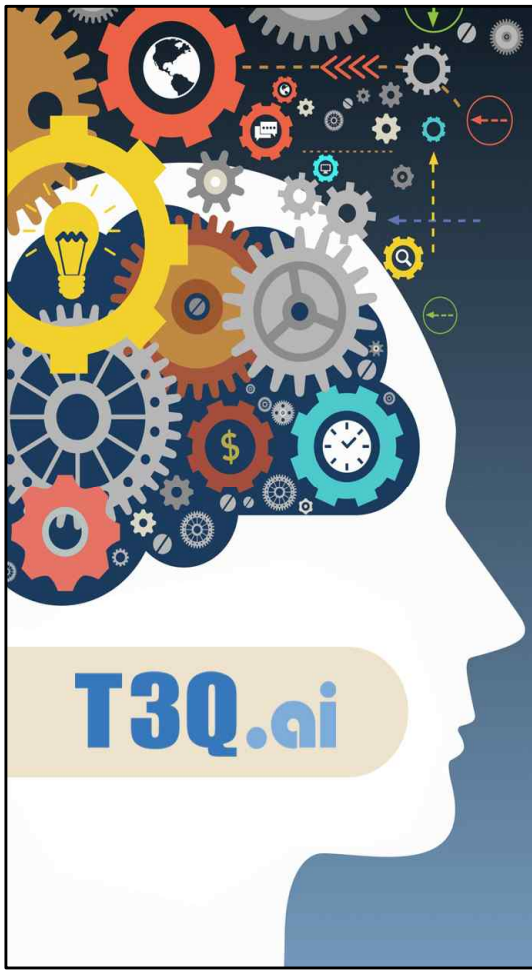


III장. AI훈민정음_3
Satellite_Classification
(다중레이블된 지표면 분류)

목 차

1. 소개 : 다중레이블된 지표면 분류

미국 지질 조사국(USGS)에서 제공하는 인공위성에서 찍은 미국 전역의 다양한 토지 이미지 중 차량과 관련된 토지 이미지를 분류하는 예제

1. 데이터셋

UC Merced Land Use Dataset:
21가지 목록, 각 100장으로 이루어진
PNG 형식의 미국 토지 이미지 데이터셋

이 중 차량과 관련된 5가지 목록 사용

2. 전처리 및 학습

전처리:
이미지 크기를 (56, 56, 3)으로 설정,
레이블 원-핫 인코딩

학습:
EfficientNetB5를 활용한 전이학습

3. 추론 결과

고속도로, 교차로, 고가도로,
주차장, 일반도로
총 5가지로 위성 이미지 분류

4. 기대 효과

교통 상황 파악
도로 상황 개선 자료로 활용 가능

I. 개요

데이터셋

- 고속도로, 교차로, 고가도로, 주차장, 일반도로 총 5개의 목록으로 이루어져 있음

고속도로

교차로

고가도로

주차장

일반도로

원-핫 인코딩

- 표현하고 싶은 단어의 인덱스에 1의 값을 부여하고, 다른 인덱스에는 0을 부여하는 벡터 표현 방식
- 범주형 데이터를 계산할 수 있는 형태로 만들어 주는 작업

고속도로 데이터 원-핫 인코딩

고속도로	교차로	고가도로	주차장	일반도로
1	0	0	0	0

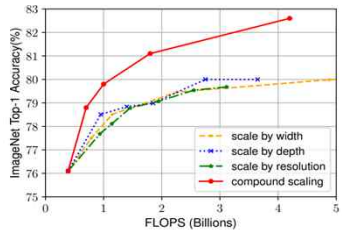
5개의 레이블 중 고속도로는 첫 번째에 해당
원-핫 인코딩을 통해 고속도로를 [1, 0, 0, 0, 0]으로 표현

I. 개요

EfficientNet

- 2019년 CNN 모델의 크기 조정 방안으로 개발된 모델
- 기존의 모델들은 성능 향상을 위해
  1. 필터의 수를 늘림(너비)
  2. 층을 많이 쌓음(깊이)
  3. 입력 이미지 크기를 키움(해상도)
- EfficientNet은 compound scaling을 통해 위 3가지 간 최적의 조합을 찾음

compound scaling



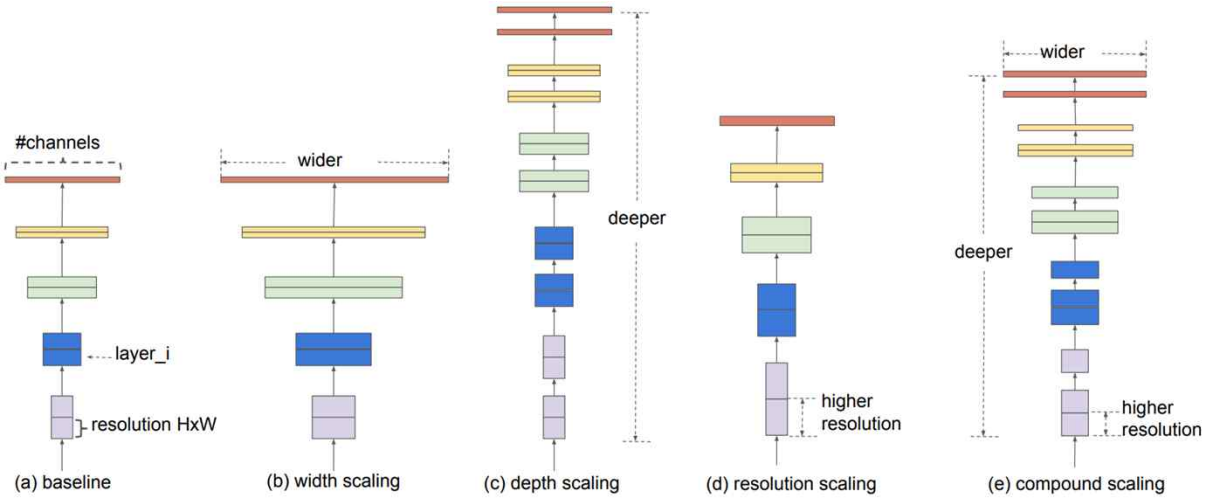
Model	FLOPS	Top-1 Acc.
Baseline model (EfficientNet-B0)	0.4B	77.3%
Scale model by depth (d=4)	1.8B	79.0%
Scale model by width (w=2)	1.8B	78.9%
Scale model by resolution (r=2)	1.9B	79.1%
Compound Scale (d=1.4, w=1.2, r=1.3)	**1.8B**	**81.1%**

너비, 깊이, 해상도 모두를 조절하는 것이 가장 성능이 좋음

6

모델 성능향상을 위한 방법

a) 기본 모델 b) 너비 조절 c) 깊이 조절 d) 해상도 조절 e) compound scaling



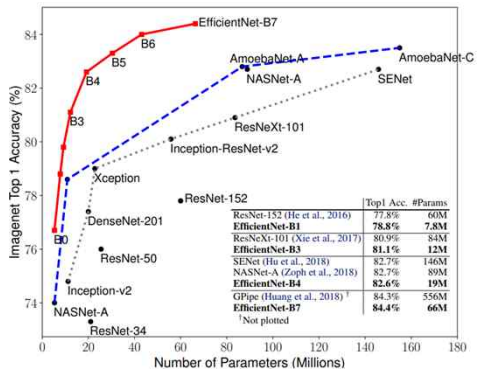
내용 출처: Tan, M. & Le, Q.. (2019). EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. Available from https://proceedings.mlr.press/v97/tan19a.html

I. 개요

EfficientNet

- EfficientNet은 B0부터 B7까지 총 8종류 존재
- 모든 EfficientNet 모델은 B0 모델에서 복합 계수 ϕ만큼 확장됨

EfficientNet 모델들의 정확도와 param 변화



	Top1 Acc.	#Params
ResNet-152 (He et al., 2016)	77.8%	60M
EfficientNet-B1	**78.8%**	**7.8M**
ResNeXt-101 (Xie et al., 2017)	80.9%	84M
EfficientNet-B3	**81.1%**	**12M**
SENet (Hu et al., 2018)	82.7%	146M
NASNet-A (Zoph et al., 2018)	82.7%	89M
EfficientNet-B4	**82.6%**	**19M**
GPipe (Huang et al., 2018) †	84.3%	556M
EfficientNet-B7	**84.4%**	**66M**

†Not plotted

모델이 확장될수록
정확도와
param의 수가 증가

복합 계수 ϕ 계산식

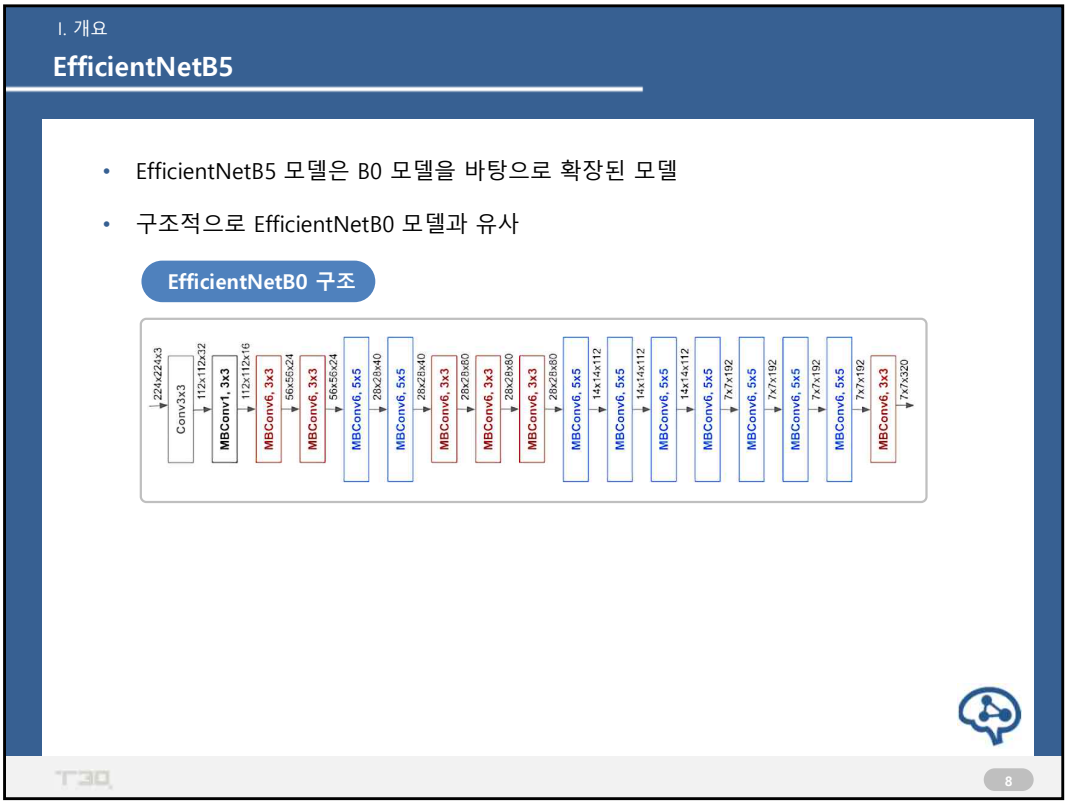
$$\text{depth: } d = \alpha^\phi$$
$$\text{width: } w = \beta^\phi$$
$$\text{resolution: } r = \gamma^\phi$$

$$\text{s.t. } \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$$
$$\alpha \geq 1, \beta \geq 1, \gamma \geq 1$$

ϕ는 사용 가능한 리소스 수를 제어하는 사용자 지정 계수로
네트워크 너비, 깊이 및 해상도에 추가 리소스를 각각 추가해 계산됨

ϕ가 1일때, 가장 최적화된 B0 모델의 α는 1.2, β는 1.1, γ는 1.15

내용 출처: Tan, M. & Le, Q.. (2019). EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. Available from https://proceedings.mlr.press/v97/tan19a.html



EfficientNetB0의 구조

Stage i	Operator $\hat{\mathcal{F}}_i$	Resolution $\hat{H}_i \times \hat{W}_i$	#Channels $\hat{C}_i$	#Layers $\hat{L}_i$
1	Conv3x3	224×224	32	1
2	MBConv1, k3x3	112×112	16	1
3	MBConv6, k3x3	112×112	24	2
4	MBConv6, k5x5	56×56	40	2
5	MBConv6, k3x3	28×28	80	3
6	MBConv6, k5x5	14×14	112	3
7	MBConv6, k5x5	14×14	192	4
8	MBConv6, k3x3	7×7	320	1
9	Conv1x1 & Pooling & FC	7×7	1280	1

1) conv3x3: 3x3 커널을 2간격으로 사용해 32장의 224 x 224 특성맵들이 생성됨

2) MBConv1: MBConv는 MnasNet에서 사용하는 Conv 구조. 3x3 커널을 1간격으로 사용해 16장의 112 x 112 특성맵들이 생성됨

3) MBConv6: 3x3 커널을 2간격으로 사용해 24장의 112 x 112 특성맵들이 생성됨, 2개의 층을 쌓음

4) MBConv6: 5x5 커널을 2간격으로 사용해 40장의 56 x 56 특성맵들이 생성됨, 2개의 층을 쌓음

5) MBConv6: 3x3 커널을 2간격으로 사용해 80장의 28 x 28 특성맵들이 생성됨, 3개의 층을 쌓음

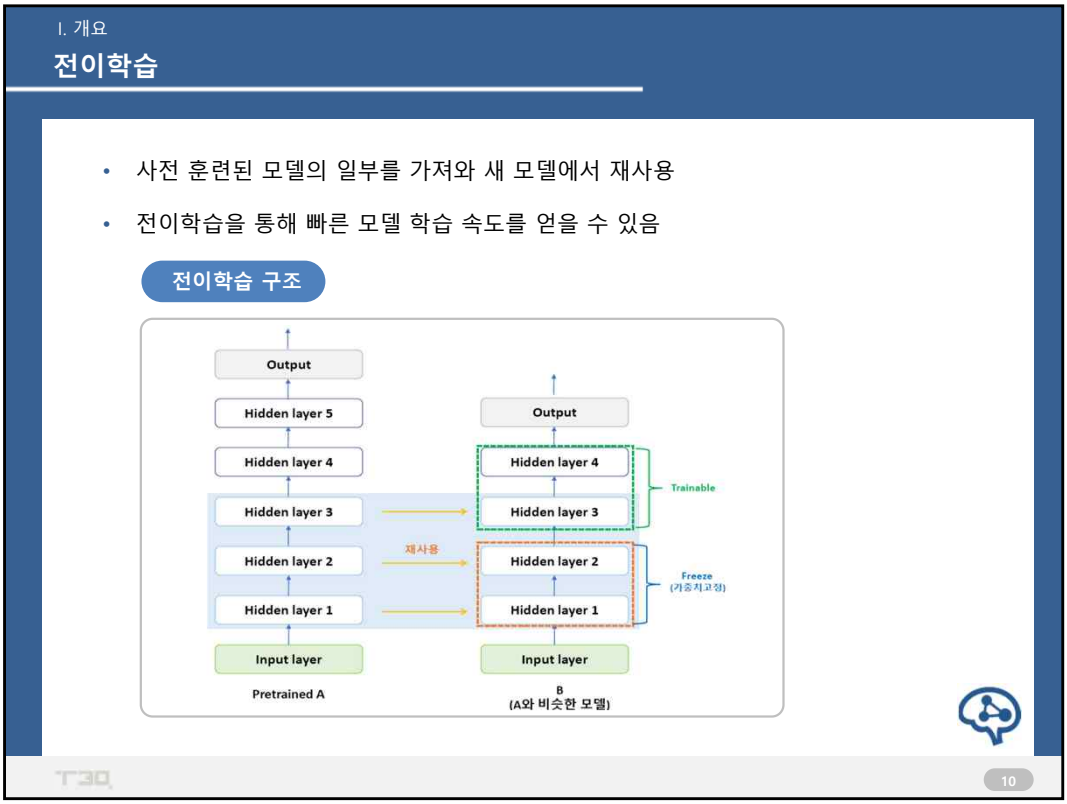
6) MBConv6: 5x5 커널을 1간격으로 사용해 112장의 14 x 14 특성맵들이 생성됨, 3개의 층을 쌓음

7) MBConv6: 5x5 커널을 2간격으로 사용해 192장의 14 x 14 특성맵들이 생성됨, 4개의 층을 쌓음

8) MBConv6: 3x3 커널을 1간격으로 사용해 320장의 7 x 7 특성맵들이 생성됨

9) Conv1x1&Pooling&FullyConnected: 1x1 커널을 1간격으로 사용해 1280장의 7 x 7 특성맵들이 생성됨, AdaptiveAvgPool2d(1)를 진행하고 1280개의 뉴런 생성

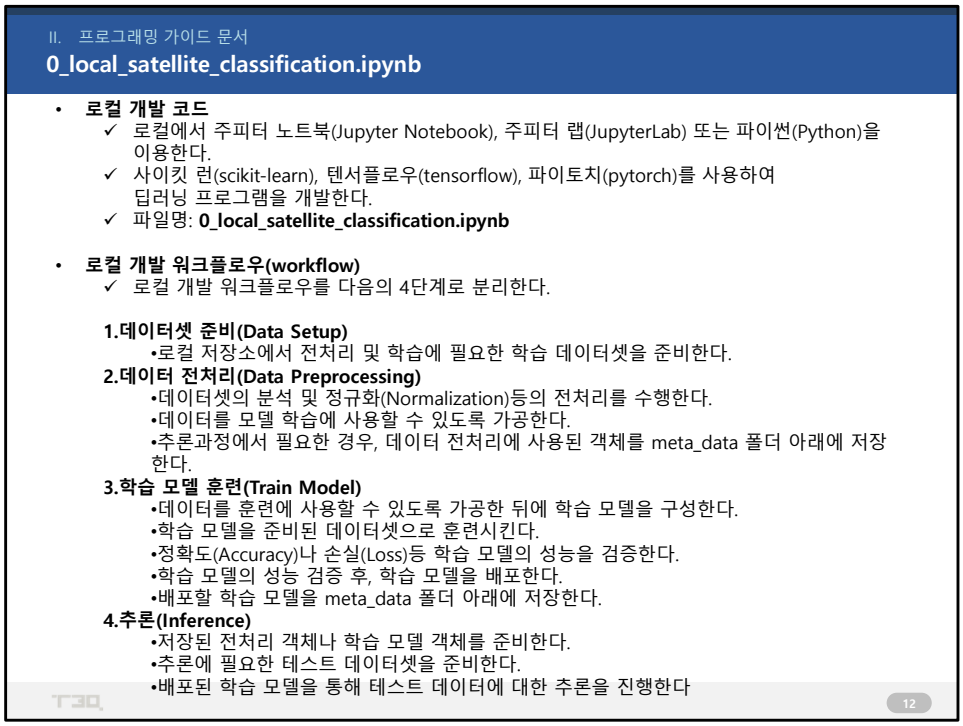
내용 출처: Tan, M. & Le, Q.. (2019). EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. Available from https://proceedings.mlr.press/v97/tan19a.html



이미지 출처: 강수철. (2020). 인공지능 전이학습과 응용 분야 동향. 정보통신기획평가원(IITP). https://t1.daumcdn.net/cfile/tistory/9972414A5E527AC226

0_local_satellite_classification_requirement

- 로컬에 설치 되어야 할 패키지 버전
  - ✓ numpy 1.21.5
  - ✓ opencv 4.5.1
  - ✓ pillow 9.0.1
  - ✓ matplotlib 3.5.2
  - ✓ tensorflow 2.6.0
  - ✓ tensorflow-gpu 2.6.0



0_local_satellite_classification.ipynb

==========

```
#imports
import zipfile
import tensorflow as tf
from tensorflow.keras import layers
import matplotlib.pyplot as plt
from tensorflow.keras.callbacks import EarlyStopping
from glob import glob
import numpy as np
from PIL import Image
import cv2
```

1. 데이터셋 준비(Data Setup)

```
# dataset.zip 파일 압축을 meta_data 폴더 안에 풀어준다.
zip_source_path = './dataset.zip'
zip_target_path = './meta_data'

extract_zip_file = zipfile.ZipFile(zip_source_path)
extract_zip_file.extractall(zip_target_path)

extract_zip_file.close()

# dataset 파일 안에 있는 데이터 경로를 가져온다.
train_dir = zip_target_path + "/dataset/train"
test_dir = zip_target_path + "/dataset/test"
val_dir = zip_target_path + "/dataset/validation"
```

2. 데이터 전처리 (Data Preprocessing)

```
IMGSIZE = 56
BATSIZE = 32

# efficientnet에 맞춰 데이터를 전처리한다.
train_eff = tf.keras.preprocessing.image_dataset_from_directory(train_dir,
                                                label_mode = "categorical",
                                                image_size = (IMGSIZE,IMGSIZE),
                                                batch_size = BATSIZE
                                                )
test_eff = tf.keras.preprocessing.image_dataset_from_directory(test_dir,
                                                label_mode = "categorical",
                                                image_size = (IMGSIZE,IMGSIZE),
                                                batch_size = BATSIZE
                                                )
val_eff = tf.keras.preprocessing.image_dataset_from_directory(val_dir,
                                                label_mode = "categorical",
                                                image_size = (IMGSIZE,IMGSIZE),
                                                batch_size = BATSIZE
                                                )

labels_name = test_eff.class_names
print(labels_name)
```

3. 학습 모델 훈련(Train Model)

EfficientNet

- efficientnet은 keras.Model의 인스턴스입니다.
- 이 함수는 ImageNet에서 사전 훈련된 가중치로 선택적으로 로드된 Keras 이미지 분류 모델을 반환합니다.
- B0부터 B7까지 총 8개의 종류로 이루어져있습니다.

```
# efficientnet50을 불러온다.
eff_base = tf.keras.applications.EfficientNetB5(include_top= False)
# trainable 속성을 True로 설정한다.
eff_base.trainable = True

# 모델을 생성한다.
inputs = tf.keras.Input(shape=(IMGSIZE,IMGSIZE,3))
x = eff_base(inputs)
x = layers.GlobalAveragePooling2D()(x)
outputs = layers.Dense(5, activation="softmax")(x) # label 수
eff_model = tf.keras.Model(inputs, outputs)

# compile
eff_model.compile(
    loss = tf.keras.losses.categorical_crossentropy,
    optimizer = tf.keras.optimizers.Adam(learning_rate= 0.001),
    metrics = ["accuracy"]
)

eff_model.summary()
```

```
early_stopping = EarlyStopping(monitor='val_accuracy', mode='max', verbose=1, patience=1)

# 모델을 학습한다.
history = eff_model.fit(train_eff,
            epochs = 10,
            steps_per_epoch = len(train_eff),
            validation_data = val_eff,
            validation_steps = len(val_eff),
            callbacks = [early_stopping]
            )

# 모델 성능을 확인하기 위해 시각화한다.

plt.style.use("default")
fig, ax = plt.subplots(1,2,figsize=(20,5))

ax[0].set_title("Loss")
ax[1].set_title("Accuracy")

ax[0].plot(history.history["loss"], label="Train")
ax[0].plot(history.history["val_loss"], label="Validation")
ax[1].plot(history.history["accuracy"], label="Train")
ax[1].plot(history.history["val_accuracy"], label="Validation")

ax[0].legend(loc="upper right")
ax[1].legend(loc="lower right")

plt.show()
```

4. 추론 (Inference)

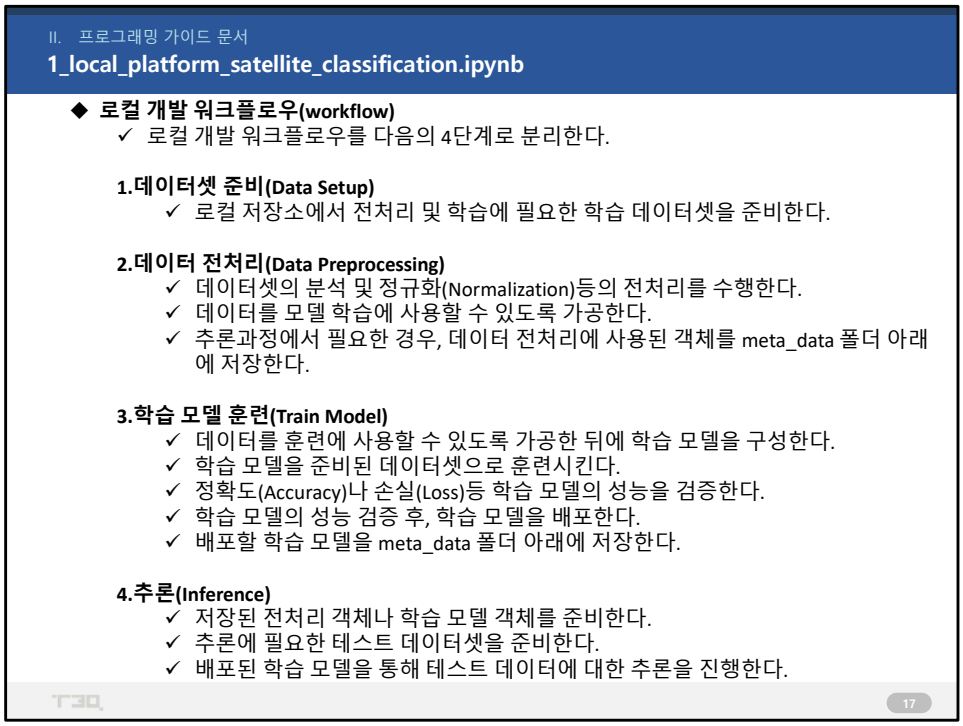
```
zip_test_target_path = './meta_data/test_dataset'

# test_dataset.zip 파일을 test_dataset 폴더에 압축을 풀어준다.
zip_test_source_path = './test_dataset.zip'

extract_zip_file = zipfile.ZipFile(zip_test_source_path)
extract_zip_file.extractall(zip_test_target_path)

extract_zip_file.close()

test_files = glob(zip_test_target_path + '/*.png')
test_files

# 추론한다.
result = []

for test_file in test_files:
    image = Image.open(test_file)
    image = image.resize((IMGSIZE, IMGSIZE))
    numpy_image = np.array(image)
    opencv_image = cv2.cvtColor(numpy_image, cv2.COLOR_RGB2BGR)
    image = cv2.resize(opencv_image, (IMGSIZE, IMGSIZE))
    image = image.reshape(1, IMGSIZE, IMGSIZE, 3)
    result.append(labels_name[np.argmax(eff_model.predict(image))])
result

# 데이터를 시각화하고 추론 결과를 출력한다.
for i in range(len(test_files)):
    test_image = Image.open(test_files[i])
    test_image = np.array(test_image)
    plt.title(result[i])
    plt.imshow(test_image)
    plt.show()
```

1_local_platform_satellite_classification.ipynb

◆ **플랫폼 업로드를 쉽게하기 위한 로컬 개발 코드**
- ✓ T3Q.ai(T3Q.cep + T3Q.dl): 빅데이터/인공지능 통합 플랫폼
- ✓ 플랫폼 업로드를 쉽게하기 위하여 로컬에서 아래의 코드(파일1)를 개발한다.
- ✓ 파일 1(파일명): 1_local_platform_satellite_classification.ipynb

▪ **전처리 객체 또는 학습모델 객체**
- ✓ 전처리 객체나 학습모델 객체는 meta_data 폴더 아래에 저장한다.

▪ **데이터셋(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.




```
# 파일명: satellite_classification_preprocess.py

'''
from satellite_classification_preprocess_sub import exec_process
'''
import logging

logging.basicConfig(level=logging.INFO)


def process_for_train(pm):
    exec_process(pm)
    logging.info('[hunmin log] the end line of the function [process_for_train]')


def init_svc(im, rule):
    return {}


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
```

```
# 파일명: satellite_classification_preprocess_sub.py

import os
import zipfile
import logging


def exec_process(pm):

    logging.info('[hunmin log]  the start line of the function [exec_process]')
    logging.info('[hunmin log] pm.source_path : {}'.format(pm.source_path))

    # 저장된 파일을 확인한다.
    list_files_directories(pm.source_path)

    # pm.source_path의 dataset.zip 파일을
    # pm.target_path의 dataset 폴더에 압축을 풀어준다.
    my_zip_path = os.path.join(pm.source_path,'dataset.zip')
    extract_zip_file = zipfile.ZipFile(my_zip_path)
    extract_zip_file.extractall(pm.target_path)
    extract_zip_file.close()

    # 압축을 푼 파일을 확인한다.
    list_files_directories(pm.target_path)

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

```
# 저장 파일 확인함수
def list_files_directories(path):
    # 현재 작업 디렉토리의 모든 파일 및 디렉토리 목록을 가져온다.
    dir_list = os.listdir(path)

    logging.info('[hunmin log] Files and directories in {} :'.format(path))
    logging.info('[hunmin log] dir_list : {}'.format(dir_list))
```

```
# 파일명: satellite_classification_train.py

'''
from satellite_classification_train_sub import exec_train, exec_init_svc, exec_inference
'''
import logging

logging.basicConfig(level=logging.INFO)


def train(tm):
    exec_train(tm)
    logging.info('[hunmin log] the end line of the function [train]')


def init_svc(im):
    params = exec_init_svc(im)
    logging.info('[hunmin log] the end line of the function [init_svc]')
    return { **params }


def inference(df, params, batch_id):
    result = exec_inference(df, params, batch_id)
    logging.info('[hunmin log] the end line of the function [inference]')
    return { **result }
```

```
# 파일명: satellite_classification_train_sub.py

# Imports
import os
import numpy as np
import pandas as pd
import tensorflow as tf
from tensorflow.keras import layers
from tensorflow.keras.models import load_model
from tensorflow.keras.callbacks import EarlyStopping
import logging
import base64
import io
from PIL import Image
import cv2
import pickle

logging.info(f'[hunmin log] tensorflow ver : {tf.__version__}')

# 사용할 gpu 번호를 적는다.
# os.environ["CUDA_VISIBLE_DEVICES"]='0,1'

gpus = tf.config.experimental.list_physical_devices('GPU')
if gpus:
    try:
        tf.config.experimental.set_visible_devices(gpus, 'GPU')
        logging.info('[hunmin log] gpu set complete')
        logging.info('[hunmin log] num of gpu: {}'.format(len(gpus)))

    except RuntimeError as e:
        logging.info('[hunmin log] gpu set failed')
        logging.info(e)

########### 플랫폼 train(tm) 부분의 코드 입니다. ###########

IMGSIZE = 56
BATSIZE = 32

def exec_train(tm):

    logging.info('[hunmin log] the start line of the function [exec_train]')
    logging.info('[hunmin log] tm.train_data_path : {}'.format(tm.train_data_path))

    # 저장된 파일을 확인한다.
    list_files_directories(tm.train_data_path)

    ######################################################################
    ## 1. 데이터셋 준비(Data Setup)
    ######################################################################

    my_path = os.path.join(tm.train_data_path, 'dataset') + '/'

    train_dir = my_path + 'train'
    test_dir = my_path + 'test'
    val_dir = my_path + 'validation'
```

```
    ######################################################################
    ## 2. 데이터 전처리(Data Preprocessing)
    ######################################################################

    # efficientnet에 맞춰 데이터를 전처리한다.
    train_eff = tf.keras.preprocessing.image_dataset_from_directory(train_dir,
                                                    label_mode = "categorical",
                                                    image_size = (IMGSIZE,IMGSIZE),
                                                    batch_size =
BATSIZE
                                                    )
    test_eff = tf.keras.preprocessing.image_dataset_from_directory(test_dir,
                                                    label_mode = "categorical",
                                                    image_size = (IMGSIZE,IMGSIZE),
                                                    batch_size =
BATSIZE
                                                    )
    val_eff = tf.keras.preprocessing.image_dataset_from_directory(val_dir,
                                                    label_mode = "categorical",
                                                    image_size = (IMGSIZE,IMGSIZE),
                                                    batch_size = BATSIZE
                                                    )
    labels_name = test_eff.class_names
    logging.info('[hunmin log] labels_name : {}'.format(labels_name))

    ######################################################################
```

```
## 3. 학습 모델 훈련(Train Model)
 ######################################################################

 # 단일 gpu 혹은 cpu학습 기능이다.
 if len(gpus) < 2:
     eff_model = model_build_and_compile(IMGSIZE)
 #multi-gpu
 else:
     strategy = tf.distribute.MirroredStrategy()
     logging.info('[hunmin log] gpu devices num {}'.format(strategy.num_replicas_in_sync))
     with strategy.scope():
         eff_model = model_build_and_compile(IMGSIZE)

 # 사용자가 입력한 파라미터이다.
 batch_size = int(tm.param_info['batch_size'])
 epochs = int(tm.param_info['epoch'])

 # gpu에 따른 batch_size 설정 기능이다.
 batch_size = batch_size * len(gpus) if len(gpus) > 0 else batch_size
```

```
early_stopping = EarlyStopping(monitor='val_accuracy', mode='max', verbose=1,
patience=1)

    # 모델을 학습한다.
    history = eff_model.fit(train_eff,
                epochs = epochs,
                steps_per_epoch = len(train_eff),
                validation_data = val_eff,
                validation_steps = len(val_eff),
                callbacks = [early_stopping]
                )

    logging.info('[hunmin log] model train complete')
    logging.info('[hunmin log] accuracy : {}'.format(eff_model.evaluate(val_eff)[1]*100))

    ######################################################################
    ## 플랫폼 시각화
    ######################################################################
    '''
    plot_metrics(tm, history, eff_model, test_eff)
    '''

    ######################################################################
    ## 학습 모델 저장
    ######################################################################
```

```
    logging.info('[hunmin log] tm.model_path : {}'.format(tm.model_path))
    eff_model.save(os.path.join(tm.model_path, 'model.h5'))

    # 저장한 파일을 확인한다.
    list_files_directories(tm.model_path)
    logging.info('[hunmin log]  the finish line of the function [exec_train]')

def exec_init_svc(im):

    logging.info('[hunmin log] im.model_path : {}'.format(im.model_path))

    # 저장된 파일을 확인한다.
    list_files_directories(im.model_path)

    ######################################################################
    ## 학습 모델 준비
    ######################################################################

    # 모델을 불러온다.
    model = load_model(os.path.join(im.model_path, 'model.h5'))

    return {"model": model}
```

```
def exec_inference(df, params, batch_id):

    ################################################################
    ## 4. 추론(Inference)
    ################################################################

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

    # 학습 모델을 준비한다.
    model = params['model']
    columns = ['freeway', 'intersection', 'overpass', 'parkinglot', 'runway']
    new_df = pd.DataFrame(columns = columns)
    data = df.iloc[0, 0]
    decoded_data = io.BytesIO(base64.b64decode(data))
    image = Image.open(decoded_data)
    numpy_image = np.array(image)
    opencv_image = cv2.cvtColor(numpy_image, cv2.COLOR_RGB2BGR)
    image = cv2.resize(opencv_image, (IMGSIZE, IMGSIZE))
    image = image.reshape(1, IMGSIZE, IMGSIZE, 3)

    pred_idx = model.predict(image)
    logging.info('[hunmin log] pred_idx : {}'.format(pred_idx))

    predict = columns[np.argmax(pred_idx)]

    # json형식으로 변환할 수 있는 dictionary로 반환한다.
```

```
    result = { 'inference' : predict }
    logging.info('[hunmin log] response : {}'.format(result))

    return result


# 저장 파일 확인함수
def list_files_directories(path):
    # 현재 작업 디렉토리의 모든 파일 및 디렉토리 목록 가져온다.
    dir_list = os.listdir(path)
    logging.info('[hunmin log] Files and directories in {} :'.format(path))
    logging.info('[hunmin log] dir_list : {}'.format(dir_list))


    ##################################################################
    ## exec_train(tm) 호출 함수
    ##################################################################
```

```
# 모델 생성 함수
def model_build_and_compile(IMGSIZE):
    # efficientnet50을 불러온다.
    eff_base = tf.keras.applications.EfficientNetB5(include_top= False)
    # trainable 속성을 True로 설정한다.
    eff_base.trainable = True
    # 모델을 생성한다.
    inputs = tf.keras.Input(shape=(IMGSIZE,IMGSIZE,3))
    x = eff_base(inputs)
    x = layers.GlobalAveragePooling2D()(x)
    x = layers.Flatten()(x)
    outputs = layers.Dense(5, activation="softmax")(x) # label 수
    model = tf.keras.Model(inputs, outputs)
    logging.info('[hunmin log] model.summary() : {}'.format(model.summary()))

    # compile
    model.compile(
        loss = tf.keras.losses.categorical_crossentropy,
        optimizer = tf.keras.optimizers.Adam(learning_rate= 0.001),
        metrics = ["accuracy"]
    )

    return model

# 플랫폼에서 실행되는 시각화 함수
def plot_metrics(tm, history, model, x_test):
```

```
from sklearn.metrics import accuracy_score, confusion_matrix, log_loss

accuracy_list = history.history['accuracy']
loss_list = history.history['loss']

for step, (acc, loss) in enumerate(zip(accuracy_list, loss_list)):
    metric={}
    metric['accuracy'] = acc
    metric['loss'] = loss
    metric['step'] = step
    tm.save_stat_metrics(metric)

predict_y = []
actual_y = []

for image, label in x_test.take(1):
    model_prediction = model.predict(image)
    predict_y = np.argmax(model_prediction, axis = 1).tolist()
    actual_y = np.argmax(label, axis = 1).tolist()

eval_results={}
eval_results['predict_y'] = predict_y
eval_results['actual_y'] = actual_y
eval_results['loss'] = history.history['val_loss'][-1]
eval_results['accuracy'] = history.history['val_accuracy'][-1]

# confusion_matrix(eval_results)를 계산한다.
eval_results['confusion_matrix'] = confusion_matrix(actual_y, predict_y).tolist()
tm.save_result_metrics(eval_results)
logging.info('[hunmin log] accuracy and loss curve plot for platform')
```

```
# PM 클래스: pm 객체
class PM:
    def __init__(self):
        self.source_path = './'
        self.target_path = './meta_data'

# TM 클래스: tm 객체
class TM:
    param_info = {}
    def __init__(self):
        self.train_data_path = './meta_data'
        self.model_path = './meta_data'
        self.param_info['batch_size'] = 32
        self.param_info['epoch'] = 10

# IM 클래스: im 객체
class IM:
    def __init__(self):
        self.model_path = './meta_data'

# pm 객체
pm = PM()
print('pm.source_path:', pm.source_path)
print('pm.target_path: ', pm.target_path)

# tm 객체
```

```
tm = TM()
print('tm.train_data_path: ', tm.train_data_path)
print('tm.model_path: ', tm.model_path)
print('tm.param_info[₩'batch_size₩']: ', tm.param_info['batch_size'])
print('tm.param_info[₩'epoch₩']: ', tm.param_info['epoch'])

# im 객체
im = IM()
print('im.model_path: ', im.model_path)

# inferecne(df, params, batch_id) 함수 입력
params = {}
batch_id = 0

import io
import pandas as pd

# base64 encoding: freeway.png
data = # '(11_1) Request_satellite_classification' 파일 안에 있는 base64 형식의 이미지 이용
data = io.StringIO(data)
df = pd.DataFrame(data)
print('df: ', df)
print('df.dtypes:', df.dtypes)
df.columns

process_for_train(pm)
train(tm)
transform(df, params, batch_id)
params = init_svc(im)
inference(df, params, batch_id)
```

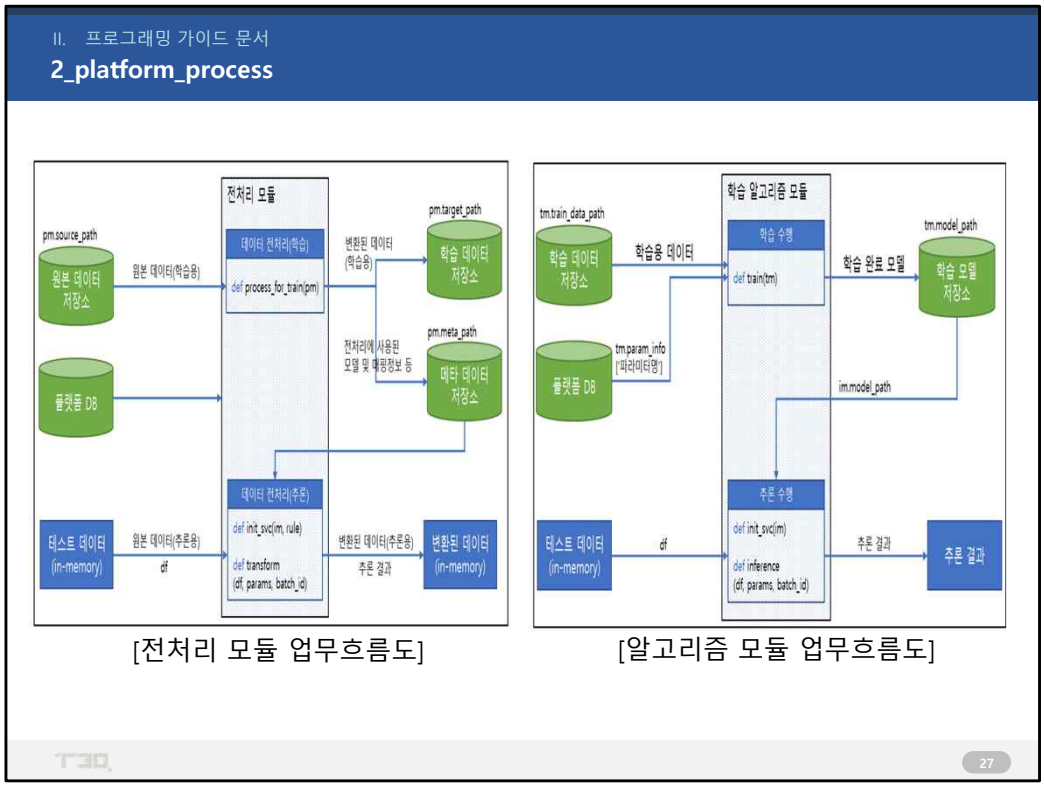
2_platform_process

- 파일명 : satellite_classification_preprocess.py
  - ✓ from satellite_classification_preprocess_sub import exec_process
  - ✓ def process_for_train(pm):
  - ✓ def init_svc(im, rule):
  - ✓ def transform(df, params, batch_id):

- 파일명: satellite_classification_preprocess_sub.py
  - ✓ def exec_process(pm):

- 파일명: satellite_classification_train.py
  - ✓ from satellite_classification_train_sub import exec_train, exec_init_svc, exec_inference
  - ✓ def train(tm):
  - ✓ def init_svc(im):
  - ✓ def inference(df, params, batch_id):

- 파일명: satellite_classification_train_sub.py
  - ✓ def exec_train(tm):
  - ✓ def exec_init_svc(im):
  - ✓ def exec_inference(df, params, batch_id):
    1. 데이터셋 준비(Data Setup)
    2. 데이터 전처리(Data Preprocessing)
    3. 학습 모델 훈련(Train Model)
    4. 추론(Inference)



0. 빅데이터/인공지능 통합 플랫폼 [T3Q.ai]
   - 빅데이터 플랫폼 [T3Q.cep]
   - 인공지능 플랫폼 [T3Q.dl]
   - 빅데이터/인공지능 통합 플랫폼 [T3Q.ai (T3Q.cep + T3Q.dl)]

1. 머신러닝(Machine Learning)과 딥러닝(Deep Learning) 프로그래밍 패턴
   (1) 데이터셋 불러오기(Dataset Loading)
   (2) 데이터 전처리(Data Preprocessing)
      - 데이터 정규화(Normalization)
      - 학습과 테스트 데이터 분할(Train/Test Data Split) 등
   (3) 학습 모델 구성(Train Model Build)
   (4) 학습(Model Training)
   (5) 학습 모델 성능 검증(Model Performance Validation)
   (6) 학습 모델 저장(배포) 하기(Model Save)
   (7) 추론 데이터 전처리((Data Preprocessing)
   (8) 추론(Inference) 또는 예측(Prediction)
   (9) 추론 결과 데이터 후처리(Data Postprocessing)

2. 빅데이터/인공지능 통합 플랫폼[T3Q.ai]에서 딥러닝 프로그래밍 하고 인공지능 서비스 실시간 운용하기
   - 6개의 함수로 딥러닝 프로그래밍 하고 인공지능 서비스 실시간 운용하기

   (1) process_for_train(pm) 함수
      - 데이터셋 준비(Dataset Setup)에 필요한 코드 작성
   (2) init_svc(im, rule) 함수
      - 전처리 객체 불러오기 에 필요한 코드 작성(생략 가능)
   (3) transform(df, params, batch_id) 함수
      - 추론 데이터 전처리((Data Preprocessing) 에 필요한 코드 작성(생략 가능)

(4) train(tm) 함수
- 데이터셋 불러오기(Dataset Loading)
- 데이터 전처리(Data Preprocessing)
- 학습 모델 구성(Train Model Build)
- 학습(Model Training)
- 학습 모델 성능 검증(Model Performance Validation)
- 전처리 객체 저장
- 학습 모델 저장(배포) 하기에 필요한 코드 작성

(5) init_svc(im) 함수
- 전처리 객체 불러오기
- 학습모델 객체 불러오기에 필요한 코드 작성

(6) inference(df, params, batch_id) 함수
- 추론 데이터 전처리((Data Preprocessing)
- 추론(Inference) 또는 예측(Prediction)
- 추론 결과 데이터 후처리(Data Postprocessing)에 필요한 코드 작성

==========

3. 전처리 모듈 관리, 학습 알고리즘 관리 함수 설명

1) 프로젝트 설정/전처리모듈 관리 함수

```
def process_for_train(pm):
    """
    (1) 입력: pm
      # pm.source_path: 학습플랫폼/데이터셋 관리 메뉴에서 저장한 데이터를 불러오는 경로
      # pm.target_path: 처리 완료된 데이터를 저장하는 경로
    (2) 출력: None
    (3) 설명:
      # 데이터셋 관리 메뉴에서 저장한 데이터를 불러와서 필요한 처리를 수행
      # 처리 완료된 데이터를 저장하는 기능, pm.target_path에 저장
      # train(tm) 함수의 tm.train_data_path를 통해 데이터를 불러와서 전처리와 학습을 수행
    """

def init_svc(im, rule):
    """
    (1) 입력: im, rule
    (2) 출력: None
    (3) 설명:
      # process_for_train(pm) 함수에서 저장한 전처리 객체와 데이터에 적용된 룰(rule)을
        불러오는 기능
      # 전처리 객체, 룰(rule) 불러오기 기능 없이 처리
    """
    return {}

def transform(df, params, batch_id):
    """
    (1) 입력: df, params, batch_id
      # df: 추론모델관리와 추론API관리, 실시간 추론을 통해 전달되는 추론 입력 데이터
        (dataframe 형태)
      # params: init_svc(im, rule) 함수의 리턴(return) 값을 params 변수로 전달
    (2) 출력: df
    (3) 설명:
      # df(추론 입력 데이터)에 대한 전처리를 수행한 후 전처리 된 데이터를
        inference(df, ...) 함수의 입력 df에 전달하는 기능
      # df(추론 입력 데이터)를 전처리 없이 inference(df, params, batch_id) 함수의 입력 df
        에 리턴(return)
    """
    return df
```

2) 프로젝트 설정/학습 알고리즘 관리 함수

```
def train(tm):
    """
    (1) 입력: tm
      # tm.train_data_path: pm.target_path에 저장한 데이터를 불러오는 경로
      # tm.model_path: 전처리 객체와 학습 모델 객체를 저장하는 경로
    (2) 출력: None
    (3) 설명:
      # pm.target_path에 저장한 데이터를 tm.train_data_path를 통해 데이터를 불러오는 기능
      # 데이터 전처리와 학습 모델을 구성하고 모델 학습을 수행
      # 학습 모델의 성능을 검증하고 배포할 학습 모델을 저장
      # 전처리 객체와 학습 모델 객체를 저장, tm.model_path에 저장
      # init_svc(im) 함수의 im.model_path를 통해 전처리 객체와 학습 모델 객체를 준비
    """

def init_svc(im):
    """
    (1) 입력: im
      # im.model_path: tm.model_path에 저장한 전처리 객체와 학습 모델 객체 등을
        불러오는 경로
    (2) 출력: 전처리 객체와 학습 모델 객체 등을 딕셔너리(dictionary) 형태로 리턴(return)
    (3) 설명:
      # tm.model_path에 저장한 전처리 객체와 학습 모델 객체 등을 불러오는 기능
      # 전처리 객체, 룰(rule) 불러오기 기능 없이 처리
      # 전처리 객체와 학습 모델 객체 등을 딕셔너리(dictionary) 형태로 리턴(return)
      # 리턴(return) 값을 inference(df, params, batch_id) 함수의 입력 params 변수로 전달
    """

    return { "model": model, "param": param }

def inference(df, params, batch_id):
    """
    (1) 입력: df, params, batch_id
      # df: transform(df, params, batch_id)함수의 리턴(return) 값으로 전달된 df, 추론 입력
        데이터 (dataframe 형태)
      # params  init_svc(im) 함수의 return 값을 params 변수로 전달
        ## 학습 모델 객체 사용 예시        model=params['model']
        ## 전처리(pca) 객체 사용 예시      pca=params['pca']
    (2) 출력: 추론 결과를 딕셔너리(dictionary) 형태로 리턴(return)
    (3) 설명:
      # 전처리 객체를 사용하여 df(추론 입력 데이터)에 대한 전처리 수행
      # 배포된 학습 모델(model)을 사용하여 df(추론 입력 데이터)에 추론(예측)을 수행
      # 추론 결과를 딕셔너리(dictionary) 형태로 리턴(return)
    """
    return {"inference": result}
```

==========

4. 전처리 모듈 관리, 학습 알고리즘 관리 함수 설명(AI 훈민정음 프로젝트)

1) 프로젝트 설정/전처리모듈 관리 함수(AI 훈민정음 프로젝트)

```
import logging

def process_for_train(pm):
    """
    (1) 입력: pm
        # pm.source_path: 학습플랫폼/데이터셋 관리 메뉴에서 저장한 데이터를 불러오는 경로
        # pm.target_path: 처리 완료된 데이터를 저장하는 경로
    (2) 출력: None
    (3) 설명:
        # 데이터셋 관리 메뉴에서 저장한 데이터를 불러와서 필요한 처리를 수행
        # 처리 완료된 데이터를 저장하는 기능, pm.target_path에 저장
        # train(tm) 함수의 tm.train_data_path를 통해 데이터를 불러와서 전처리와 학습을 수행
    (4) 추가 설명:
        # 함수 구조는 원형대로 유지
        # 실질적인 기능을 하는 함수를 서브모듈 함수(exec_process)로 정의하여 사용함
        # 함수명                          서브함수명
        # process_for_train(pm)         exec_process(pm)
        # 함수의 정상적인 동작 체크를 위해 마지막 라인(the end line)에 로그 출력 수행
    """

    exec_process(pm)

    logging.info('[hunmin log] the end line of the function [process_for_train]')

def init_svc(im, rule):
    """
    (1) 입력: im, rule
    (2) 출력: None
    (3) 설명:
      # process_for_train(pm) 함수에서 저장한 전처리 객체와 데이터에 적용된 룰(rule)을
        불러오는 기능
      # 전처리 객체, 룰(rule) 불러오기 기능 없이 처리
    """
    return {}

def transform(df, params, batch_id):
    """
    (1) 입력: df, params, batch_id
      # df: 추론모델관리와 추론API관리, 실시간 추론을 통해 전달되는 추론 입력 데이터
        (dataframe 형태)
      # params: init_svc(im, rule) 함수의 리턴(return) 값을 params 변수로 전달
    (2) 출력: df
    (3) 설명:
      # df(추론 입력 데이터)에 대한 전처리를 수행한 후 전처리 된 데이터를 inference(df, ...)
        함수의 입력 df에 전달하는 기능
      # df(추론 입력 데이터)를 전처리 없이 inference(df, params, batch_id) 함수의 입력 df에
        리턴(return)
    (4) 추가 설명:
      # 함수 구조는 원형대로 유지
      # 함수의 정상적인 동작 체크를 위해 마지막 라인(the end line)에 로그 출력 수행
    """
    logging.info('[hunmin log] the end line of the function [transform]')
    return df
```

2) 프로젝트 설정/ 학습 알고리즘 관리 함수(AI 훈민정음 프로젝트)

```
import logging

def train(tm):
    """
    (1) 입력: tm
        # tm.train_data_path: pm.target_path에 저장한 데이터를 불러오는 경로
        # tm.model_path: 전처리 객체와 학습 모델 객체를 저장하는 경로
    (2) 출력: None
    (3) 설명:
        # pm.target_path에 저장한 데이터를 tm.train_data_path를 통해 데이터를 불러오는 기능
        # 데이터 전처리와 학습 모델을 구성하고 모델 학습을 수행
        # 학습 모델의 성능을 검증하고 배포할 학습 모델을 저장
        # 전처리 객체와 학습 모델 객체를 저장, tm.model_path에 저장
        # init_svc(im) 함수의 im.model_path를 통해 전처리 객체와 학습 모델 객체를 준비
    (4) 추가 설명:
        # 함수 구조는 원형대로 유지
        # 실질적인 기능을 하는 함수를 서브모듈 함수(exec_train)로 정의하여 사용함
        # 함수명                          서브함수명
        # train(tm)                       exec_train(tm)
        # 함수의 정상적인 동작 체크를 위해 마지막 라인(the end line)에 로그 출력 수행
    """

    exec_train(pm)

    logging.info('[hunmin log] the end line of the function [train]')

def init_svc(im):
    """
    (1) 입력: im
        # im.model_path: tm.model_path에 저장한 전처리 객체와 학습 모델 객체 등을 불러오는
          경로
    (2) 출력: 전처리 객체와 학습 모델 객체 등을 딕셔너리(dictionary) 형태로 리턴(return)
    (3) 설명:
        # tm.model_path에 저장한 전처리 객체와 학습 모델 객체 등을 불러오는 기능
        # 전처리 객체, 룰(rule) 불러오기 기능 없이 처리
        # 전처리 객체와 학습 모델 객체 등을 딕셔너리(dictionary) 형태로 리턴(return)
        # 리턴(return) 값을 inference(df, params, batch_id) 함수의 입력 params 변수로 전달
    (4) 추가 설명:
        # 함수 구조는 원형대로 유지
        # 실질적인 기능을 하는 함수를 서브모듈 함수(exec_init_svc)로 정의하여 사용함
        # 함수명                          서브함수명
        # init_svc(im)                    exec_init_svc(im)
        # 함수의 정상적인 동작 체크를 위해 마지막 라인(the end line)에 로그 출력 수행

    """
    params = exec_init_svc(im)
    logging.info('[hunmin log] the end line of the function [init_svc]')
    return {**params}
```

```
def inference(df, params, batch_id):
        """
        (1) 입력: df, params, batch_id
          # df: transform(df, params, batch_id)함수의 리턴(return) 값으로 전달된 df, 추론 입력
            데이터(dataframe 형태)
          # params: init_svc(im) 함수의 리턴(return) 값을 params 변수로 전달
            ## 학습 모델 객체 사용 예시        model=params['model']
            ## 전처리(pca) 객체 사용 예시      pca=params['pca']
        (2) 출력: 추론 결과를 딕셔너리(dictionary) 형태로 리턴(return)
        (3) 설명:
          # 전처리 객체를 사용하여 df(추론 입력 데이터)에 대한 전처리 수행
          # 배포된 학습 모델(model)을 사용하여 df(추론 입력 데이터)에 추론(예측)을 수행
          # 추론 결과를 딕셔너리(dictionary) 형태로 리턴(return)
        (4) 추가 설명:
          # 함수 구조는 원형대로 유지
          # 실질적인 기능을 하는 함수를 서브모듈 함수(exec_inference)로 정의하여 사용함
          # 함수명                                    서브함수명
          # inference(df, params, batch_id)         exec_inference(df, params, batch_id)
          # 함수의 정상적인 동작 체크를 위해 마지막 라인(the end line)에 로그 출력 수행
        """

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

수행절차 소개

01) T3Q.cep_데이터수집 파이프라인_satellite_classification : 해당 예제에서는 수행 절차 없음

02) T3Q.cep_데이터변환 파이프라인_satellite_classification : 해당 예제에서는 수행 절차 없음

03) T3Q.dl_프로젝트 설정_실행환경 관리_satellite_classification

04) T3Q.dl_프로젝트 설정_전처리 모듈 관리_satellite_classification

05) T3Q.dl_프로젝트 설정_학습 알고리즘 관리_satellite_classification

06) T3Q.dl_학습플랫폼_데이터셋 관리_satellite_classification

07) T3Q.dl_학습플랫폼_전처리 모델 설계_satellite_classification

08) T3Q.dl_학습플랫폼_전처리 모델 관리_satellite_classification

09) T3Q.dl_학습플랫폼_학습모델 설계_satellite_classification

10) T3Q.dl_학습플랫폼_학습모델 관리_satellite_classification

11) T3Q.dl_추론플랫폼_추론모델 관리_satellite_classification

12) T3Q.dl_추론플랫폼_추론API관리_satellite_classification

13) T3Q.cep_실시간 추론 파이프라인_satellite_classification

실행환경 추가 내용 및 절차

1) Requirements

```
==================================================
# User Requirements.
==================================================
```

2) Dockerfile

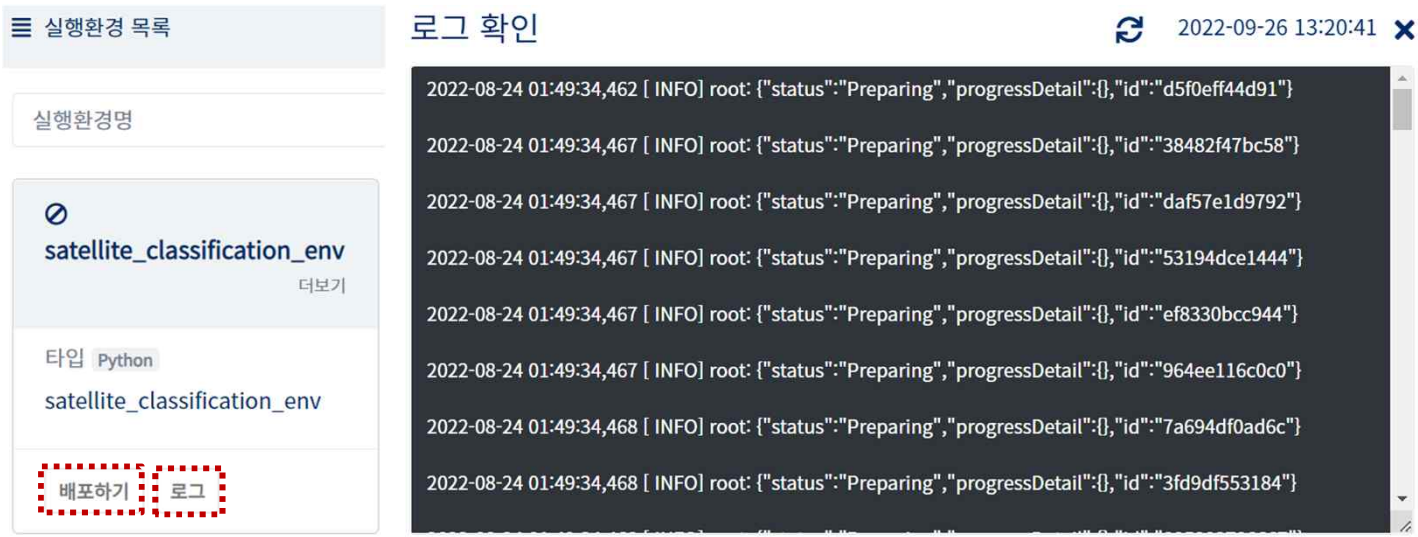
```
==================================================
FROM tensorflow/tensorflow:2.4.1-gpu

ARG DEBIAN_FRONTEND=noninteractive

RUN apt-key adv --keyserver keyserver.ubuntu.com --recv-keys A4B469963BF863CC

RUN apt-get update && apt-get install -y wget ₩
    python3.8 ₩
    python3-pip ₩
    python3-dev ₩
    python3.8-dev ₩
    postgresql ₩
    libpq-dev

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

# opencv
RUN apt-get -y install libgl1-mesa-glx
```

==========
2) Dockerfile-계속
==========

```
# generic libraries
RUN pip install --no-input numpy==1.19.5 ₩
        torch scikit-learn imbalanced-learn xgboost ₩
        fastai keras keras-preprocessing keras-vis ₩
        matplotlib pillow nltk ₩
        opencv-contrib-python opencv-python openpyxl imageio pretty_midi ₩
        pickleshare pip-tools protobuf psutil psycopg2 PyYAML ₩
        pathlib requests requests-oauthlib rsa ₩
        tensorboard tensorboard-data-server tensorboard-plugin-wit ₩
        sqlalchemy grpcio tqdm urllib3 xlrd xlwt lightgbm

# 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

#COPY ./work /work
USER root

RUN mkdir /mldatas
RUN mkdir /data
RUN mkdir /data/aip
RUN mkdir /data/aip/logs

WORKDIR/work
```

==========

추가된 실행 환경 [저장] 하고, [배포하기] 시작 , 완료 후 [로그] 에서 확인



04) T3Q.dl_프로젝트 설정_전처리 모듈 관리_satellite_classification

- 프로젝트 설정 >>전처리모듈관리
  - 전처리모듈 관리>> [전처리 모듈 추가] 실행
  - 전처리모듈 등록 시 순서와 입력 정보
  ① 기본정보
    - 전처리명: satellite_classification_premodule
    - 실행환경 선택 : satellite_classification_env
    - GPU지원 : GPU 지원(체크)
  ② 입력 형태: file
  ③ 출력 형태: default
  ④ 파라미터
  ⑤ 소스 코드 : T3Q.ai_platform_satellite_classification_preprocess.zip [파일업로드]
    - LICENSE.txt
    - README.txt
    - platform_process.txt
    - satellite_classification_preprocess.py (실행모듈 선택)
    - satellite_classification_preprocess_sub.py

- 전처리모듈 등록 시 순서와 입력 정보

① 기본정보

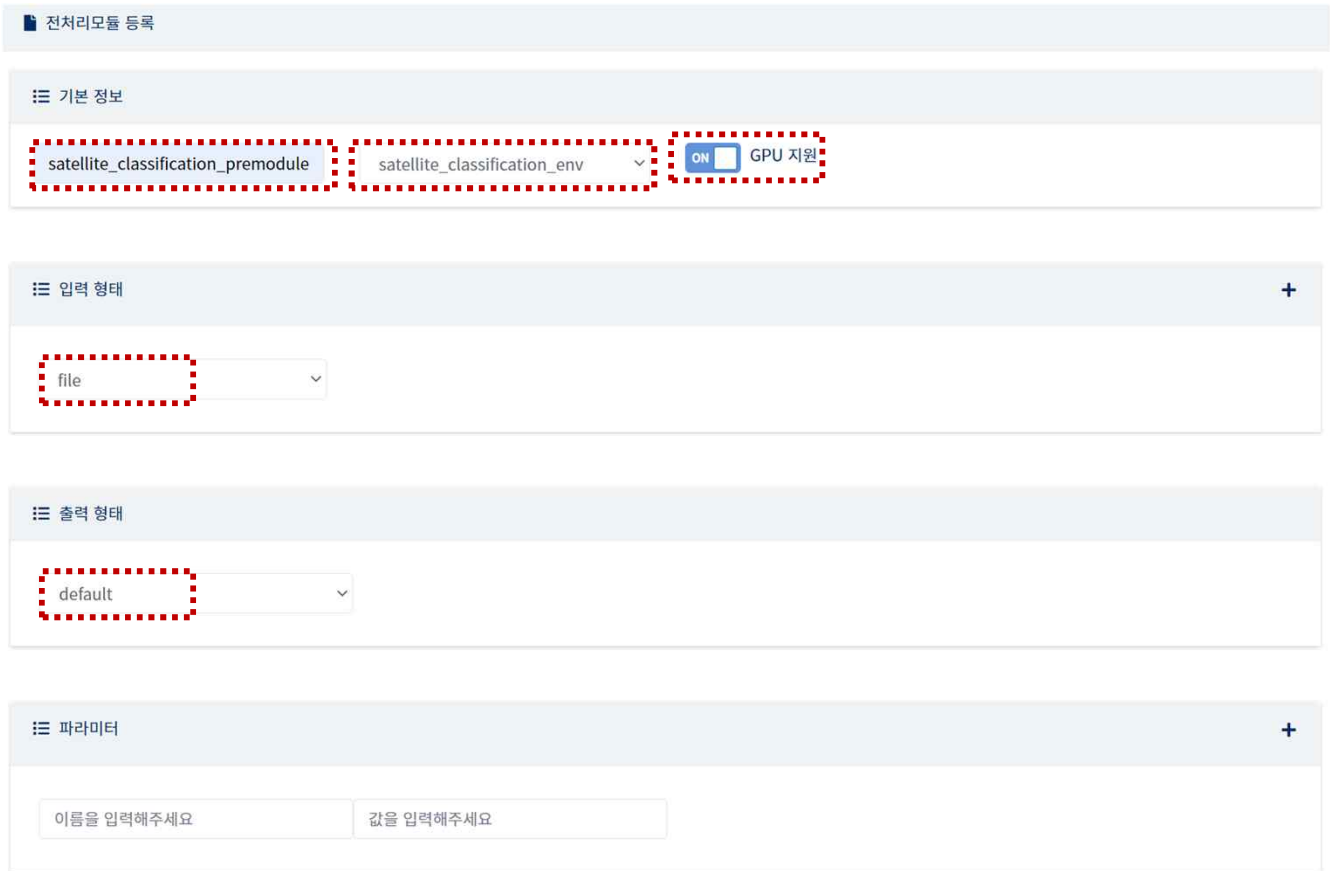
전처리명: satellite_classification_premodule

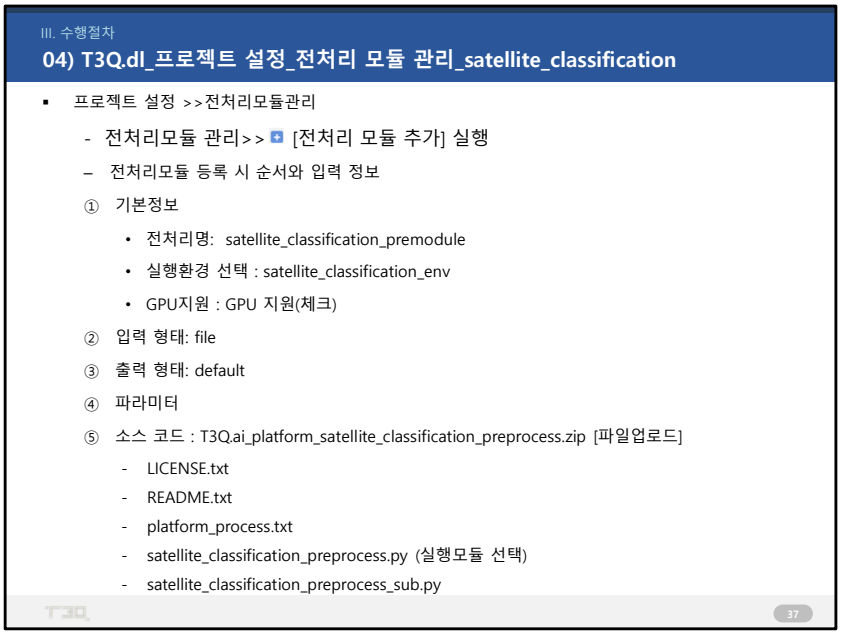
실행환경 선택 : satellite_classification_env

GPU지원 : GPU 지원(체크)

② 입력 형태: file

③ 출력 형태: default

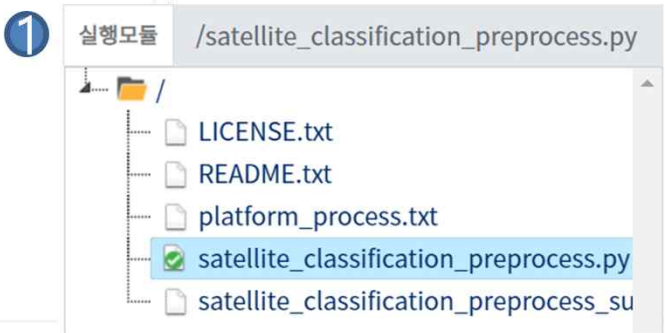




전처리모듈 등록 시 순서와 입력 정보

⑤ 소스 코드 : T3Q.ai_platform_satellite_classification_preprocess.zip [파일업로드] 누름

- LICENSE.txt
- README.txt
- platform_process.txt
- satellite_classification_preprocess.py (실행모듈 선택)
- satellite_classification_preprocess_sub.py



2 소스 코드

실행모듈 /satellite_classification_preprocess.py

/
LICENSE.txt
README.txt
platform_process.txt
satellite_classification_preprocess.py
satellite_classification_preprocess_su

선택파일 /satellite_classification_preprocess.py

```
1  # 파일명 : satellite_classification_preprocess.py
2
3
4  from satellite_classification_preprocess_sub import exec_process
5
6  import logging
7
8
9
10 def process_for_train(pm):
11     exec_process(pm)
12     logging.info('[hunmin log] the end line of the function [process_for_train]')
13
14
15 def init_svc(im, rule):
16     return {}
17
18
19 def transform(df, params, batch_id):
20     logging.info('[hunmin log] df : {}'.format(df))
21     logging.info('[hunmin log] df.shape : {}'.format(df.shape))
22     logging.info('[hunmin log] type(df) : {}'.format(type(df)))
23     logging.info('[hunmin log] the end line of the function [transform]')
24     return df
```

파일업로드

05) T3Q.dl_프로젝트 설정_학습 알고리즘 관리_satellite_classification

- 프로젝트 설정 >> 학습 알고리즘 관리
  - 학습 알고리즘 관리>>
    - [알고리즘 추가] 실행

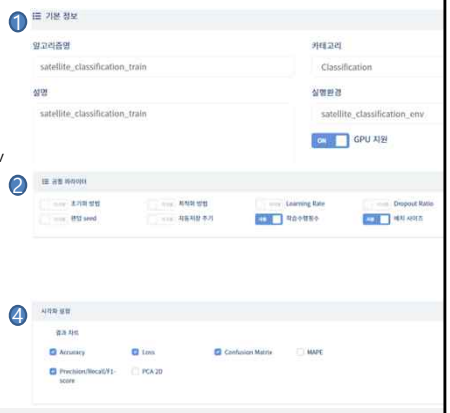
- 학습 알고리즘 등록 시 순서와 입력 정보

① 기본정보: 다음과 같이 입력
-알고리즘명: satellite_classification_train
-설명 : satellite_classification_train
-카테고리 : Classification
-실행환경 : satellite_classification_env
-GPU 지원 : 체크

② 공통 파라미터: 아래 가지 항목 사용
- 학습수행횟수: 사용
- 배치 사이즈 : 사용

③ 모델 파라미터

④ 시각화 설정: 아래 4개 항목만 체크
- Accuracy : 체크
- Loss : 체크
- Confusion Matrix : 체크
- Precision/Recall/F1-score : 체크



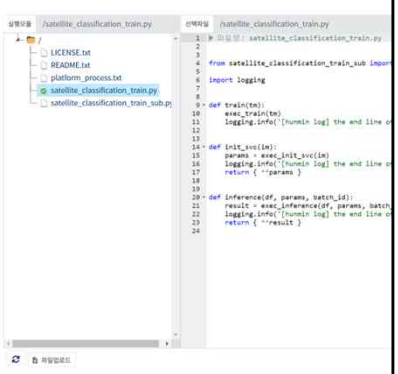
III. 수행절차

05) T3Q.dl_프로젝트 설정_학습 알고리즘 관리_satellite_classification

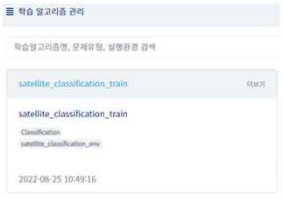
- 프로젝트 설정 >> 학습 알고리즘 관리
  - 학습 알고리즘 관리>> [알고리즘 추가] 실행
  - 학습 알고리즘 등록 시 순서와 입력 정보

⑤ 소스코드 업로드 : [파일업로드]

: T3Q.ai_platform_satellite_classification_train.zip
- LICENSE.txt
- README.txt
- platform_process.txt
- satellite_classification_train.py
- satellite_classification_train_sub.py



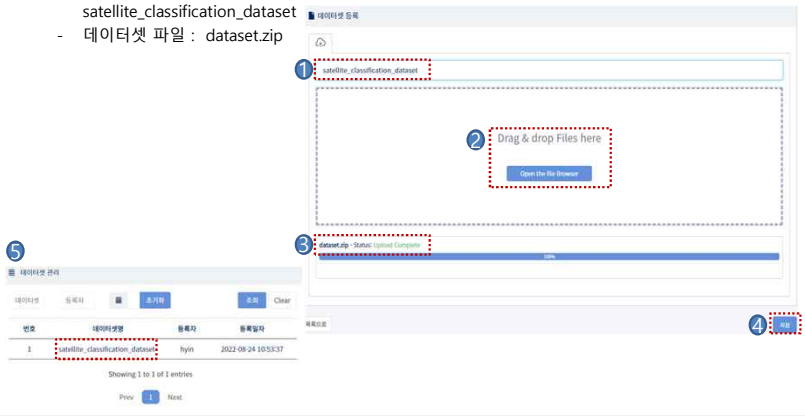
: 실행 모듈 설정 >> [저장]
- /satellite_classification_train.py 실행 모듈 지정
- [저장]을 누른다.



III. 수행절차

06) T3Q.dl_학습플랫폼_데이터셋 관리_satellite_classification

- 학습플랫폼 >> 데이터셋 관리
- 데이터셋 등록 절차
  - 데이터셋 관리>> 등록 실행
  - 데이터셋 명 : satellite_classification_dataset
  - 데이터셋 파일 : dataset.zip



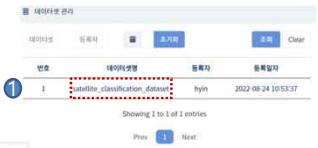
III. 수행절차

07) T3Q.dl_학습플랫폼_전처리 모델 설계_satellite_classification

- 학습플랫폼 >> 전처리 모델 설계
- 전처리 모델 설계 절차

① 학습플랫폼 >> 데이터셋 관리
  - 데이터셋 명 : satellite_classification_dataset 선택

② satellite_classification_dataset 아래의 [전처리 모델 설계] 선택



07) T3Q.dl_학습플랫폼_전처리 모델 설계_satellite_classification

- 학습플랫폼 >> 전처리 모델 설계

② 전처리 모델 설계 절차

Step1. 기본 정보

- 모델명: satellite_classification_premodel

Step2. 데이터셋 등록:

- 참조데이터셋 : satellite_classification_dataset

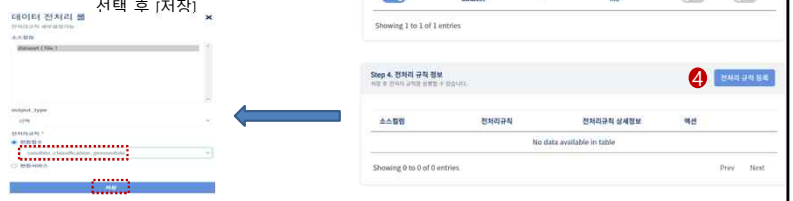
Step3. ID/LABEL 지정

Step4. 전처리 규칙 정보

③ - [전처리 규칙 등록] 선택

전처리 규칙 등록 시 절차

- 소스컬럼 : dataset(file) 선택
- 변환 함수 : satellite_classification_premodule 선택 후 [저장]



▪ 학습플랫폼 >> 전처리 모델 관리
▪ 전처리 모델 설계 상세
Step1. 기본 정보
- 모델명: satellite_classification_premodel
- 참조데이터셋 : satellite_classification_dataset
Step2. 데이터셋 컬럼정보
Step3. 전처리 룰 정보
Step4. 전처리 실행정보
전처리 상세
- 전처리 상세 버튼 누름
- 진행상황 확인
- 0_satellite_classification_premodule – [로그] 아래 [보기] 누름
[학습 모델 설계] 선택하여 다음 단계 진행

로그 확인

마지막 로딩된 시간 : 2022-09-26 13:40:43

```
2022-08-24 01:56:12,834 [ INFO] root: ### preprocessing start ###
2022-08-24 01:56:12,834 [ INFO] root: params={'pre_dataset_id': 997, 'rule': {'source_column':
['dataset'], 'rule': 'preModel', 'rule_type': 'satellite_classification_premodule', 'mod': 'U',
'param': {}, 'rule_no': '0', 'source_type': ['file'], 'module_info': '{"deploy_dt": "2022-08-24
10:49:22", "template": "Python", "version": "1.0", "status": "deployed", "image_name": 363,
"module_name": "satellite_classification_preprocess"}', 'output_type': ['default']}, 'do_fit':
True, 'test_no': None, 'test_dataset_path': None, 'log_path': '/data/aip/logs'}
2022-08-24 01:56:12,872 [ WARN] root: datasource_repo_id : 159, datasource_repo_obj :
<DataSourceRepo 159>, repo_type : path
2022-08-24 01:56:12,893 [ INFO] root:
module_path=/data/aip/logs/t3qai/premodule/premodule_561/1
2022-08-24 01:56:12,956 [ INFO] root: dp_module=<module
'satellite_classification_preprocess' from
'/data/aip/logs/t3qai/premodule/premodule_561/1/satellite_classification_preprocess.py'>
2022-08-24 01:56:12,956 [ INFO] root: [hunmin log]  the start line of the function
[exec_process]
```

학습플랫폼 >> 학습모델 설계

1. AI 훈민정음 >> 학습플랫폼 >> 학습모델 설계 상세 과정
   1) Step 1. 기본 정보
      학습모델명 satellite_classification_trainmodel
      전처리모델
      [사용] 체크
      satellite_classification_premodel satellite_classification_premodel

   2) Step 2. 모델 설계
      문제유형 Classification
      알고리즘 satellite_classification_train
      평가방법
      None

   3) Step 3. 상세 설계
      알고리즘 선택 시 상세 설계가 가능합니다. (step 2. 먼저 진행)

      (1) 공통 파라미터
          학습수행횟수 (epoch) : *10
          배치 사이즈 (개) : *32

   4) [저장] 누름

학습플랫폼 >> 학습모델 관리

1. 학습플랫폼 >> 학습모델 관리

2 학습모델 상세

1) Step 1. 학습모델 기본정보
모델명 satellite_classification_trainmodel
전처리 모델명 satellite_classification_premodel
satellite_classification_premodel

2) Step 2. 학습모델 알고리즘 정보
문제유형 Classification
알고리즘 satellite_classification_train
평가방법 / 평가값 none 없음

3) Step 3. 학습수행 관리정보
(1) 공통 파라미터

랜덤seed	777
학습수행횟수 (epoch) *	10
배치 사이즈 (개) *	32
CPU	32 코어
Memory	64 Gi
GPU	1 코어

(2) 학습상태
학습상태 시작전 [- ~ -]

Accuracy Loss Confusion Matrix Precision/Recall/F1-score

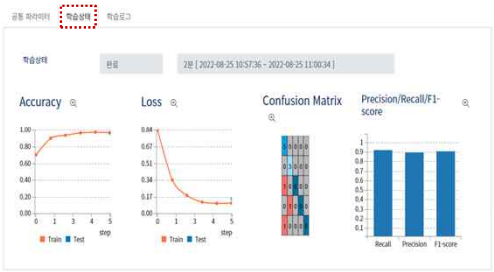
[실행] 버튼 누름

III. 수행절차

10) T3Q.dl_프로젝트 설정_학습모델 관리_satellite_classification

- 학습플랫폼 >> 학습모델 관리
- 학습모델 상세
- [실행] 완료 후

- [학습상태]

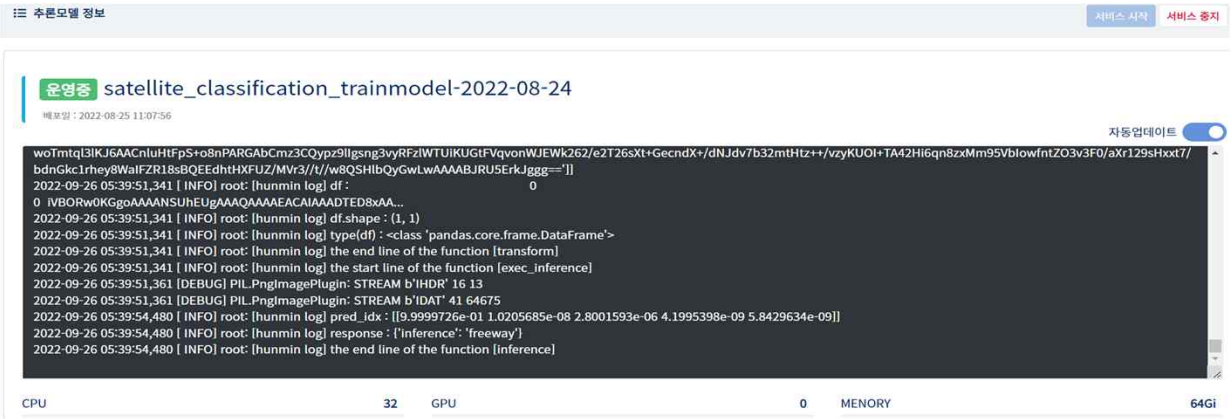


- [학습로그]

- 모델 배포 : [모델 배포] 버튼 누르고 [배포] 누름

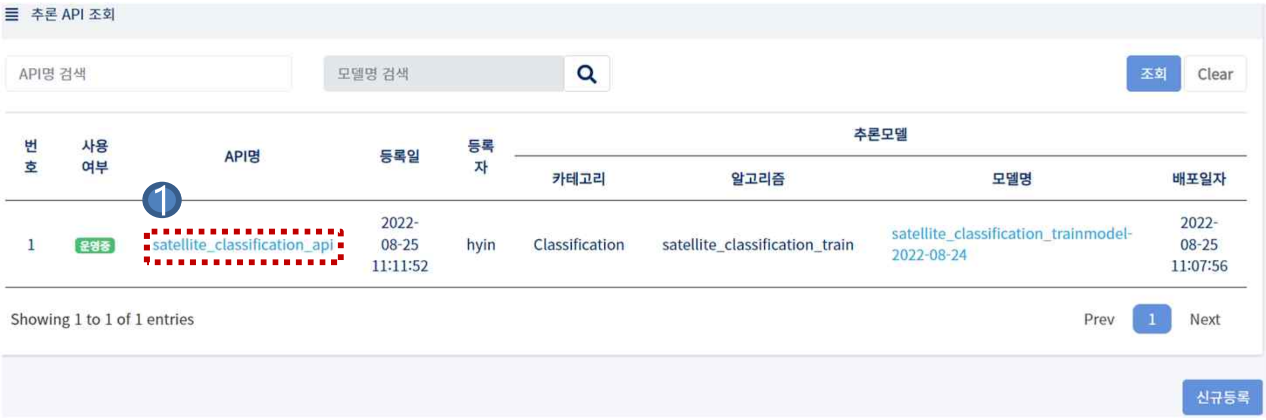
추론플랫폼 >> 추론모델관리

1. 추론플랫폼 >> 추론모델관리 - 서비스 시작
2. 추론플랫폼 >> 추론모델관리 – 추론모델 정보 확인

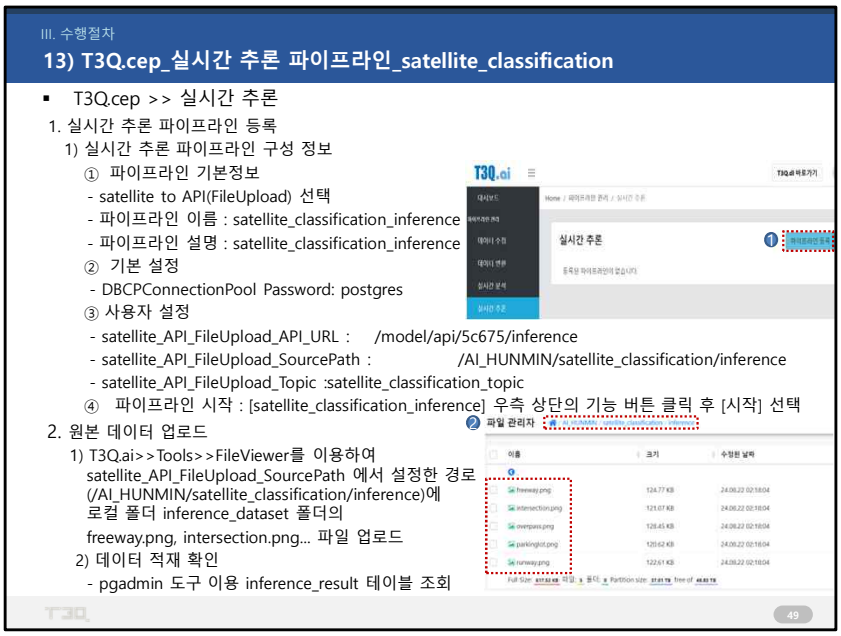


3. 추론플랫폼 >> 추론모델관리 - 추론모델 검증
[추론모델테스트]-[요청] 에 아래의 값을 넣고, [테스트] 눌러 수행
요청 : 입력 예시 : [['/data/aip/file_group/pm/pm_334/ds_441/satellite/1/1230.png']]
요청 : -> (11_1) Request_satellite_classification.txt파일 안에 있는 base64 형식의
이미지의 내용을 복사 붙여넣기 한다.
응답 : "{₩"inference₩":₩"freeway₩"}₩n"

▪추론플랫폼 >> 추론API관리
1.추론플랫폼 >> 추론API관리 – 신규 등록
2.추론플랫폼 >> 추론API관리 – 추론 API 상세



=> 요청 :{“data”:” [[‘테스트 데이터 값 입력=’]]”}=>[API 호출] 클릭
=> 응답 : {“data”:[“결과”]}



(2) 데이터 적재 확인
T3Q.ai >> Tools>> PgAdmin

#inference_origin
inference_result

테이블 조회

#select * from inference_origin;

select * from inference_result;

데이터 저장 확인

==========
SELECT * FROM public.inference_result
where url like '%/model/api/5c675/inference%'
order by start_time desc