Dacon 생육 환경 최적화 경진대회

1. 배경

안녕하세요 여러분! 😃 **생육 환경 최적화 경진대회**에 오신 것을 환영합니다.

4차산업혁명의 시대를 맞아 농업 분야에서도 스마트 팜 등 IT 기술을 널리 사용하여 더욱 효율적으로 작물을 재배하고 있습니다.

나아가, AI를 기반으로 작물의 효율적인 생육을 위한 최적의 환경을 도출한다면 식물 재배에 큰 도움이 되지 않을까요?

KIST 강릉분원은 인공지능(AI)를 활용하여 국내 고유 식물자원에서 유용한 천연물 소재를 탐색하고, 그 효능과 활성 등에 대해 연구합니다.

KIST와 함께 최적의 생육 환경 도출을 위한 AI 모델을 만들어 주세요.

2. 주제

- [Algorithm] 청경채 사진과 환경 데이터를 활용한 잎면적 예측 알고리즘 개발
- [Analytics] 적상추 데이터 활용 생육정도를 알 수 있는 정량 지표 발굴

3. 주최 / 주관

- 주최 / 주관: KIST 강릉분원
- 운영: 데이콘

Contents

01 경진대회 소개

1. 배경

안녕하세요 여러분! 😀 **생육 환경 최적화 경진대회**에 오신 것을 환영합니다.

4차산업혁명의 시대를 맞아 농업 분야에서도 스마트 팜 등 IT 기술을 널리 사용하여 더욱 효율적으로 작물을 재배하고 있습니다.

나아가, AI를 기반으로 작물의 효율적인 생육을 위한 최적의 환경을 도출한다면 식물 재배에 큰 도움이 되지 않을까요?

KIST 강릉분원은 인공지능(AI)를 활용하여 국내 고유 식물자원에서 유용한 천연물 소재를 탐색하고, 그 효능과 활성 등에 대해 연구합니다.

KIST와 함께 최적의 생육 환경 도출을 위한 AI 모델을 만들어 주세요.

2. 주제

- [Algorithm] 청경채 사진과 환경 데이터를 활용한 잎면적 예측 알고리즘 개발
- [Analytics] 적상추 데이터 활용 생육정도를 알 수 있는 정량 지표 발굴

3. 주최 / 주관

- 주최 / 주관: KIST 강릉분원
- 운영: 데이콘

01 경진대회 소개

1. 규칙

[Algorithm]

- 제출 횟수 및 최대 팀원
  a. 1일 최대 제출 횟수 : 5회
  b. 최종 제출 선택 개수 : 2개
  c. 팀 최대 인원 : 4명
- 리더보드

```
import numpy as np

def NMAE(true, pred):
    mae = np.mean(np.abs(true-pred))
    score = mae / np.mean(np.abs(true))
    return score
```

  a. 평가 산식 : NMAE (Normalized MAE)
  b. public score : 전체 테스트 데이터 중 25%
  c. private score : 전체 테스트 데이터 중 100%

✓ **Task**

Feature type : Image + Structured data
Target type : Regression

KIST 강릉분원에서는 AI를 활용하여 다양한 식물자원과 관련된 연구를 진행
이러한 연구의 일환으로 식물의 최적 생육 환경 조성을 위한 AI 모델 개발 경진대회를 주최
1일 전 청경채 이미지와, 1일 동안의 생육 환경 메타데이터를 통해 1일 후의 청경채 잎 중량을 예측

✓ **Metric : NMAE (Normalized MAE)**

예측치의 평균 절대 편차를 실측치의 평균으로 나눈 값

02 아키텍쳐 개요

✓ **Top 3 Leaderboard Score를 기록한 모델들의 ensemble로 최종 예측치를 산출**

LB Score : 0.22494

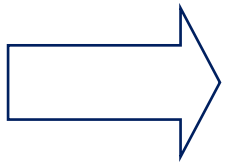
Swin Transformer(Image) + Dense Layer(Structured Data)

LB Score : 0.24807

EfficientNetV2(Image) + LSTM Layer(Structured Data)

LB Score : 0.25096

Swin Transformer(Image) + LSTM Layer(Structured Data)



LB Score : 0.21764

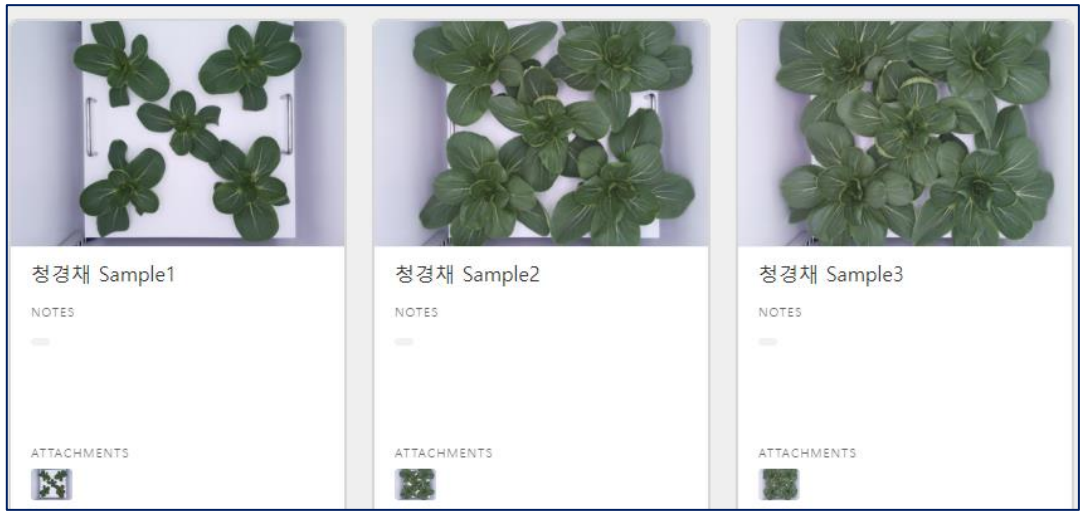
Final Score : 0.20446

Average Ensemble

(Top 47%)

02 아키텍쳐 개요 – Preprocessing & Feature Engineering

✓ Image Data



pretrained model을 통한 feature vector 추출

(resized image shape : 128 * 128 * 3)

✓ Structured data (timeseries)

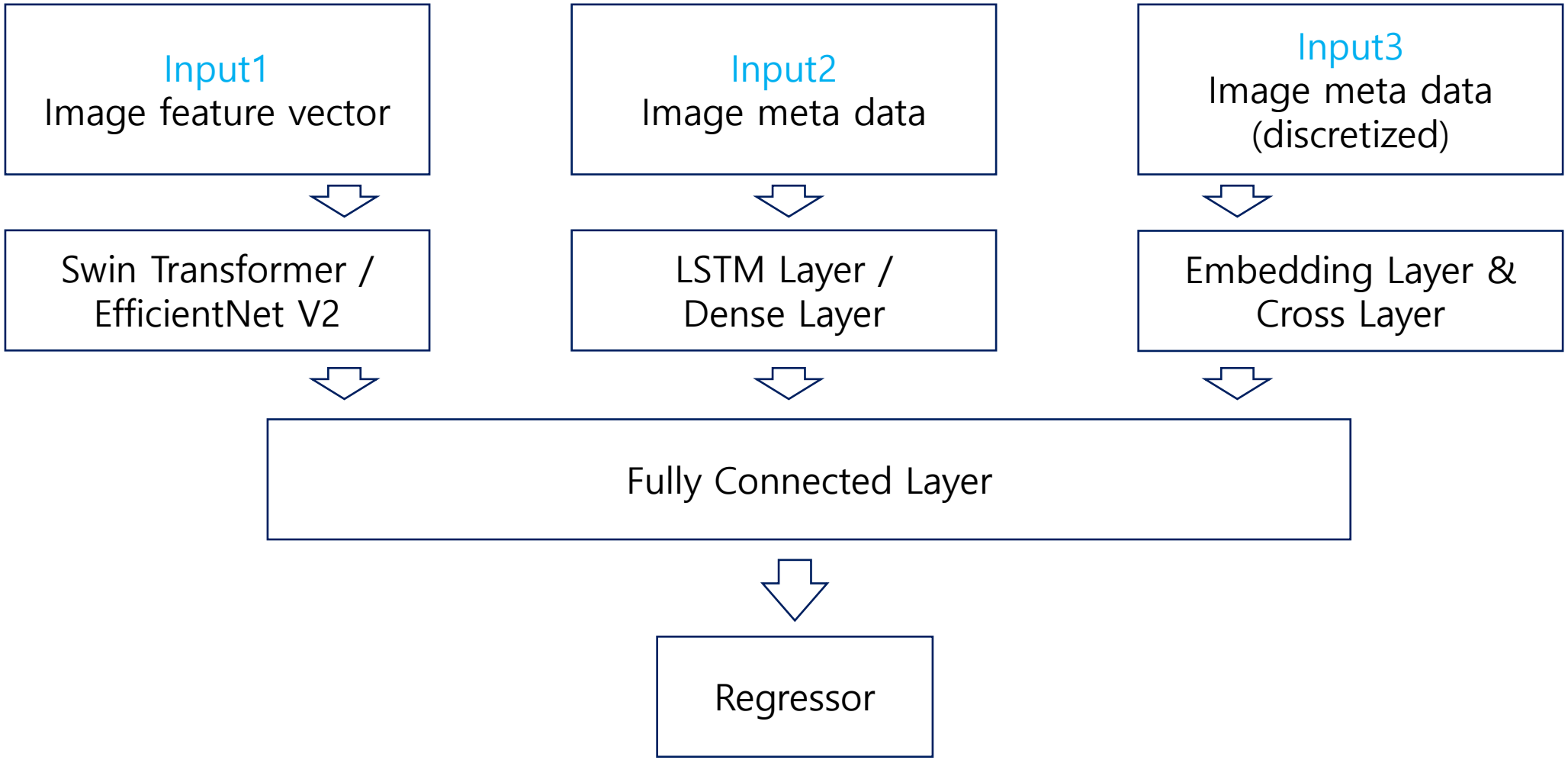
Numerical features (Standardization)

"내부온도관측치"
"외부온도관측치"
"내부습도관측치"
"외부습도관측치"
"CO2관측치"
"EC관측치"
"최근분무량"
"화이트 LED동작강도"
"레드 LED동작강도"
"블루 LED동작강도"
"냉방온도"
"냉방부하"
"난방온도"
"난방부하"
"총추정광량"
"백색광추정광량"
"적색광추정광량"
"청색광추정광량"

Categorical features

일부 Numerical feature에 대해
Discretized 된 feature 11개

02 아키텍쳐 개요 – Neural Network

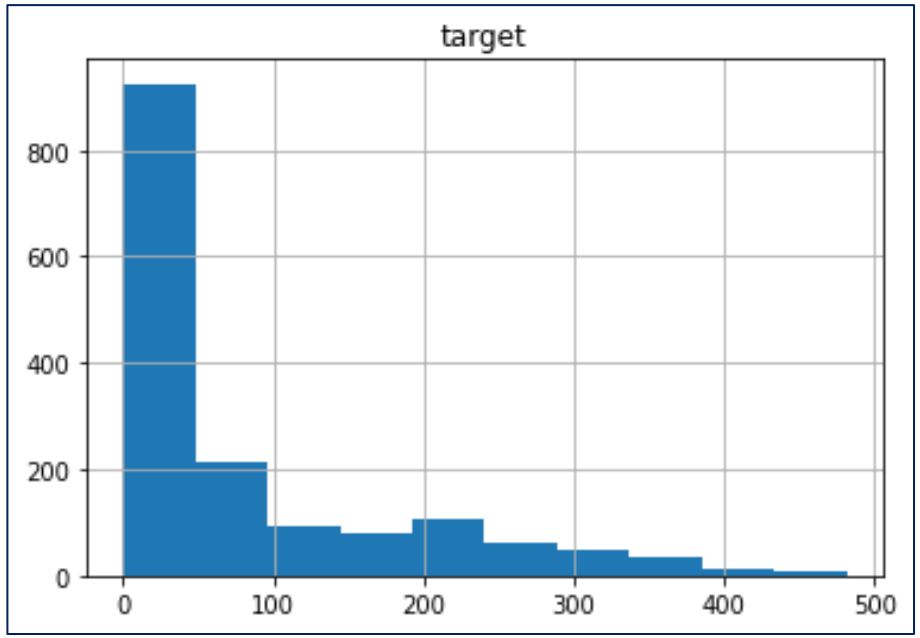


02 아키텍쳐 개요 – Target transformation

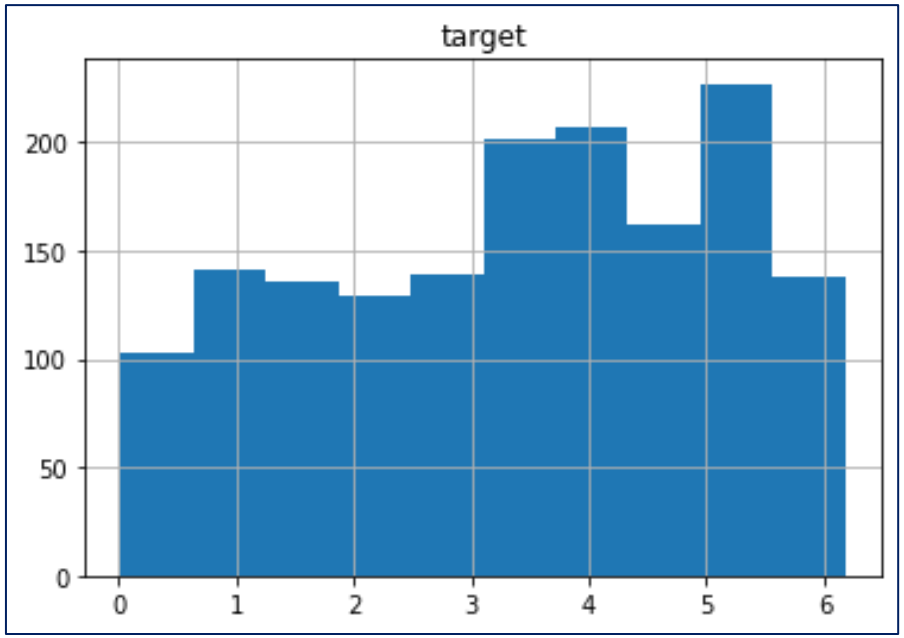
✓ **Regression 시 자주 사용되는 기법으로, target에 대해 log transformation하여 성능 향상을 시도**

target 값을 log(x+1) transformation 하여 예측 후,
exp(x)-1 으로 inverse transformation 하여 최종 예측치를 계산

Before log transformation

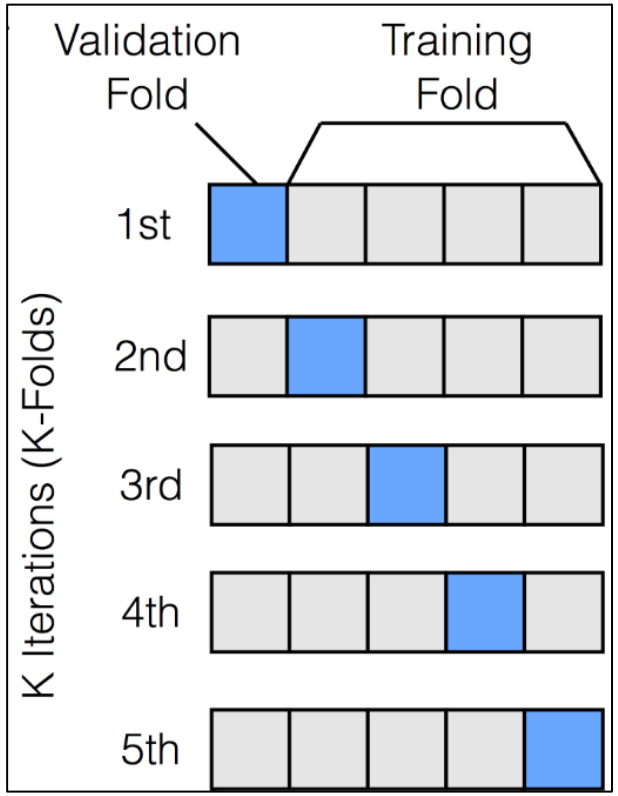


After log transformation



02 아키텍쳐 개요 – 학습 방법

✓ **5-Folds CV Model Ensemble**



03 결과 정리

- 이미지 데이터에 대해 여러 pretrained model을 활용해 보았고, ViT의 개량버전인 Swin Transformer의 성능이 확실히 좋다는 것을 확인
- Image와 meta data를 분리하여 학습 후 ensemble하는 방법을 활용해 보지 못한 부분이 아쉬운 점

674176	submission.csv edit	2022-05-20 12:44:37	0.2176416361 -	☑
673807	testResult_knnRandomnormalNA_pretrainedViT_Dense_v1_try1.csv edit	2022-05-19 10:30:19	0.2718325475 -	☐
673100	testResult_knnRandomnormalNA_pretrainedEffiV2_Dense_v1_try1.csv edit	2022-05-17 10:30:15	0.2480657804 -	☐
672720	testResult_knnRandomnormalNA_pretrainedSwin_LSTM_v1_try1.csv edit	2022-05-16 10:14:05	0.2509637342 -	☐
672603	testResult_knnRandomnormalNA_pretrainedSwin_Dense_v2_try1.csv edit	2022-05-15 17:07:36	0.2249387904 -	☐
671962	testResult_knnRandomnormalNA_swin_Dense_try1.csv edit	2022-05-13 09:21:26	0.3545310008 -	☐
670986	testResult_dropNA_Swin_Dense_try1.csv edit	2022-05-11 10:25:19	0.3480548106 -	☐

감사합니다

04 Appendix - 구현코드

```
img_size = 224
channels = 3

dropoutRate = 0.5

def create_model():
    input_list = []
    concat_list = []

    input_list.append(layers.Input(shape=(img_size, img_size, channels), dtype=tf.float32))

    # augmenation layer
    x = RandomColorDistortion(hue_flag=False)(input_list[-1])
    x = layers.RandomFlip(mode="horizontal_and_vertical")(x)
    x = tf.keras.layers.Rescaling(scale=1./255)(x)

    x = tf_hub.KerasLayer("https://tfhub.dev/sayakpaul/swin_base_patch4_window7_224_fe/1", name="pretrained", trainable=False)(x)

    x = layers.Dropout(dropoutRate)(x)
    x = layers.Dense(256, activation="relu")(x)
    x = layers.Dropout(dropoutRate)(x)
    concat_list.append(layers.Dense(64, activation="relu")(x))

    # create embedding layers
    embed_list = []
    for i in ct._transformers:
        embed_list.append(layers.Embedding(input_dim=i[1].n_bins, output_dim=16, mask_zero=False, embeddings_initializer="truncated_normal"))

    input_list.append(layers.Input(shape=(train_timeseries.shape[1], len(num_vars)), dtype=tf.float32))
    lstm_num = input_list[-1]

    input_list.append(layers.Input(shape=(train_timeseries.shape[1], len(bin_vars)), dtype=tf.float32))
    lstm_bin = input_list[-1]
```