1108 수상작 리뷰

주제

생육 환경 생성 AI 모델 결과를 바탕으로 상추의 일별 최대 잎 중량을 도출할 수 있는 최적의 생육 환경 조성

상추의 생육 환경 생성 AI 경진대회 - DACON

데이터

- **train_input - 총 28개 상추 케이스**
  - DAT : 생육일 (0~27일차)
  - obs_time : 측정 시간
  - 상추 케이스 별 환경 데이터 (1시간 간격)
- **train_target - 총 28개 상추 케이스**
  - DAT : 생육일 (1~28일차)
  - predicted_weight_g : 일별 잎 중량
- **test_input - 총 5개 상추 케이스**
  - DAT : 생육일 (0~27일차)
  - obs_time : 측정 시간
  - 상추 케이스 별 환경 데이터 (1시간 간격)
- **test_target - 총 5개 상추 케이스**
  - DAT : 생육일 (1~28일차)
  - predicted_weight_g : 일별 예측한 잎 중량

코드정리

(1) 전처리

- 데이터 불러오기

```
for idx, train_data in enumerate(train_data_list):
    globals()['train_data'+str(idx+1)] = pd.read_csv(f'{train_da

for idx, target in enumerate(train_target_list):
    globals()['target'+str(idx+1)] = pd.read_csv(f'{target}')
```

```
for i in range(1,29):
globals()['train_data'+str(i)].drop(columns=['일간누적분무량','일간
```

- 이상치 확인

```
train_data4.loc[train_data4['내부온도관측치'] < 10]
train_data6.loc[train_data6['시간당총광량'] < 0]
train_data13.loc[train_data13['시간당분무량'] < 0]
```

```
#내부온도 14, 15시 평균기온으로 값 변경
train_data4.groupby('obs_time')['내부온도관측치'].mean()

#1일차 15시와 동일한 값 넣기
train_data6.loc[15, '시간당백색광량'] = 18255.1900
train_data6.loc[15, '시간당적색광량'] = 1815.3156
train_data6.loc[15, '시간당청색광량'] = 3566.9205
train_data6.loc[15, '시간당총광량'] = 23637.4261

#마이너스 값 0으로 변경 (601, 649)
train_data13.loc[601, '시간당분무량'] = 0
train_data13.loc[649, '시간당분무량'] = 0
```

---

(2) 데이터 취합

- 관련도있는 컬럼으로 새로운 컬럼 생성
  - 배양액농도 : ec관측치 * 시간당분무량
  - 온습도 : 내부온도관측치 * 내부습도관측치
  - 온습도co2 : 내부온도관측치 * 내부습도관측치 * co2
  - 적색청색광 : 시간당적색광량 * 시간당청색광량
  - 광합성속도_적청광 : co2관측치/(시간당적색광+시간당청색광) * (내부온도+생육시기)
    - 0으로 나눠지는경우 inf 발생하므로 해당경우엔 0으로 변환

```
for i in range(1,29):
    globals()['train_data'+str(i)]['배양액농도'] = globals()['trai
    globals()['train_data'+str(i)]['광합성_적청광'] = globals()['t
    globals()['train_data'+str(i)]['광합성_적청광'] = globals()['t
```

```
min_temp = [] # 최저~최고기온
max_temp = []
min_white = [] # 최저~최고 백색광량
max_white = []
min_red = [] # 최저~최고 적색광량
max_red = []
min_blue = [] # 최저~최고 청색광량
max_blue = []

for i in range(1, 29):
    for j in range(0, 28):
        min_temp.append(globals()['train_data'+str(i)][globals()
        max_temp.append(globals()['train_data'+str(i)][globals()
        min_white.append(globals()['train_data'+str(i)][globals
        max_white.append(globals()['train_data'+str(i)][globals
        min_red.append(globals()['train_data'+str(i)][globals()
        max_red.append(globals()['train_data'+str(i)][globals()
```

```
        min_blue.append(globals()['train_data'+str(i)][globals()
        max_blue.append(globals()['train_data'+str(i)][globals()
```

(3) 데이터 취합 및 시간별 컬럼으로 확장

```
train_data_all =  pd.concat([train_data1,train_data2,train_data3
                         train_data11,train_data12,train_data13,
                         train_data21,train_data22,train_data23,
```

```
train_data_all['obs_time'] = pd.to_datetime(train_data_all['obs_
```

```
edit_df = pd.DataFrame()

# 일별- 시간별~컬럼 펼치기
for i in range(24):
    tmp = train_data_all[train_data_all['obs_time'] == i].reset_
    tmp = tmp[['내부온도관측치', '내부습도관측치', 'co2관측치', 'ec관측치
       '시간당백색광량', '시간당적색광량', '시간당청색광량',
       '배양액농도','시간당총광량','광합성_적청광' ]]

    tmp.columns = [f'{i}내부온도관측치', f'{i}내부습도관측치', f'{i}co
                   f'{i}시간당백색광량', f'{i}시간당적색광량', f'{i}시
                   f'{i}배양액농도',f'{i}시간당총광량',f'{i}광합성_적청
    edit_df = pd.concat([edit_df, tmp], axis=1)


# 일자별 신규생성 컬럼 추가
edit_df['최고기온'] = max_temp
edit_df['최저기온'] = min_temp
edit_df['최고백색광량'] = max_white
edit_df['최저백색광량'] = min_white
edit_df['최고적색광량'] = max_red
edit_df['최저적색광량'] = min_red
edit_df['최고청색광량'] = max_blue
```

```
edit_df['최저청색광량'] = min_blue


# DAT 컬럼 붙이기
tmp = train_data_all[train_data_all['obs_time'] == 23].reset_ind
tmp = tmp[['DAT']]
edit_dat_df = pd.concat([edit_df, tmp], axis=1)
edit_dat_df['DAT'] = edit_dat_df['DAT']+1
```

```
skew_df = edit_dat_df.copy()

# x데이터 왜곡도확인 및 로그화 진행

features_index = skew_df.dtypes[skew_df.dtypes != 'object'].inde
skew_features = skew_df[features_index].apply(lambda x : skew(x
skew_features_top = skew_features[skew_features > 2]
print(skew_features_top.sort_values(ascending=False))
```

- 정규화 진

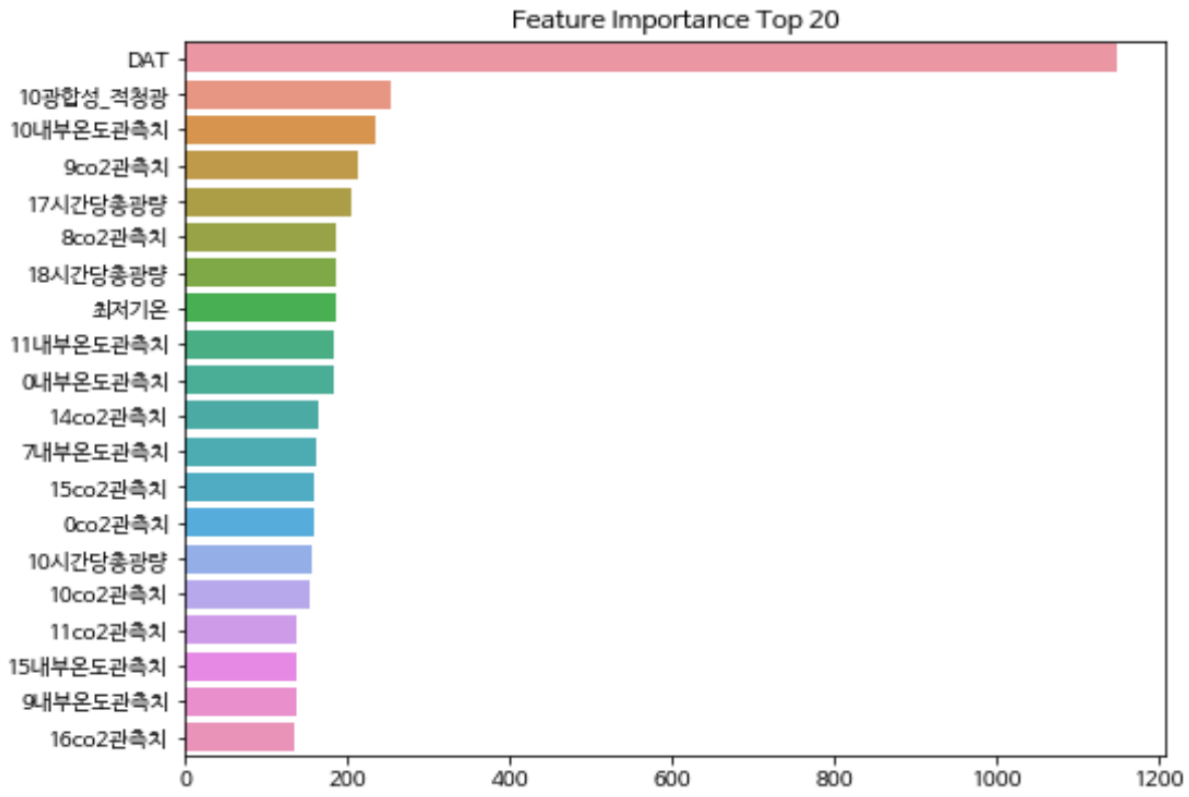
---

(4) 모델학습

```
x_train, x_test, y_train, y_test = train_test_split(skew_df, y_
```

```
# light_gbm
lg = LGBMRegressor(learning_rate = 0.1, min_child_samples = 15,

lg.fit(x_train, y_train)
pred_lg = lg.predict(x_test)
mse = mean_squared_error(y_test, pred_lg)
rmse = np.sqrt(mse)
print(f'lightgbm - MSE: {mse}, RMSE: {rmse}')

# lightgbm - MSE: 0.052311381527768895, RMSE: 0.228716815139964
```

```
# 피처 중요도
ft_importance_values = lg.feature_importances_

# 정렬과 시각화를 쉽게 하기 위해 series 전환
ft_series = pd.Series(ft_importance_values, index = x_train.colu
ft_top20 = ft_series.sort_values(ascending=False)[:20]

# 시각화
plt.figure(figsize=(8,6))
plt.title('Feature Importance Top 20')
sns.barplot(x=ft_top20, y=ft_top20.index)
plt.show()
```



(5) 예측

```
# 실제 예측값 뽑아야 하는 데이터
test_data_list = sorted(glob.glob('/content/drive/MyDrive/lettuc

# test_data 파일 생성
for idx, test_data in enumerate(test_data_list):
    globals()['test_data'+str(idx+1)] = pd.read_csv(f'{test_data
```

```
# CASE5

test_data5['배양액농도'] = test_data5['ec관측치'] * test_data5['시간
test_data5['광합성_적청광'] = test_data5['co2관측치']*(test_data5['
test_data5['광합성_적청광'] = test_data5['광합성_적청광'].replace([np


min_temp = [] # 최저~최고기온
max_temp = []
min_white = [] # 최저~최고 백색광량
max_white = []
min_red = [] # 최저~최고 적색광량
max_red = []
min_blue = [] # 최저~최고 청색광량
max_blue = []


for j in range(0, 28):
    min_temp.append(test_data5[test_data5['DAT'] == j]['내부온도관
    max_temp.append(test_data5[test_data5['DAT'] == j]['내부온도관
    min_white.append(test_data5[test_data5['DAT'] == j]['시간당백
    max_white.append(test_data5[test_data5['DAT'] == j]['시간당백
    min_red.append(test_data5[test_data5['DAT'] == j]['시간당적색광
    max_red.append(test_data5[test_data5['DAT'] == j]['시간당적색광
    min_blue.append(test_data5[test_data5['DAT'] == j]['시간당청색
    max_blue.append(test_data5[test_data5['DAT'] == j]['시간당청색
```

```
test_data5['obs_time'] = pd.to_datetime(test_data5['obs_time'])


ts5 = pd.DataFrame()

for i in range(24):
    tmp = test_data5[test_data5['obs_time'] == i].reset_index(dr
    tmp = tmp[['내부온도관측치', '내부습도관측치', 'co2관측치', 'ec관측치
       '시간당백색광량', '시간당적색광량', '시간당청색광량',
       '배양액농도','시간당총광량','광합성_적청광']]

    tmp.columns = [f'{i}내부온도관측치', f'{i}내부습도관측치', f'{i}co
                   f'{i}시간당백색광량', f'{i}시간당적색광량', f'{i}시
                   f'{i}배양액농도',f'{i}시간당총광량',f'{i}광합성_적청

    ts5 = pd.concat([ts5, tmp], axis=1)


# 일자별 신규생성 컬럼 추가
ts5['최고기온'] = max_temp
ts5['최저기온'] = min_temp
ts5['최고백색광량'] = max_white
ts5['최저백색광량'] = min_white
ts5['최고적색광량'] = max_red
ts5['최저적색광량'] = min_red
ts5['최고청색광량'] = max_blue
ts5['최저청색광량'] = min_blue


# DAT 컬럼 붙이기
tmp = test_data5[test_data5['obs_time'] == 23].reset_index(drop=
tmp = tmp[['DAT']]
ts5 = pd.concat([ts5, tmp], axis=1)
```

```
ts5['DAT'] = ts5['DAT']+1
```

차별점 및 배울점

- 데이터 특성과 모델 학습 안정성을 고려해 모델이 효과적으로 학습할 수 있도록 **정규화를 진행한 부분이 인상깊었다**.

→ 정규화로 인해 특정 변수의 단위 차이에 영향을 받지 않고 학습할 수 있었다.

- **일별~시간별~칼럼 펼치기**를 통해 더 세밀하게 분석하는 것이 인상깊었다.

→ 칼럼을 분리하여 시간대별 또는 특정 요일별 패턴을 더 명확하게 파악할 수 있었고, 시간적 흐름에 따라 모델에 반영하기 유용했다.