11/25

청경채 성장 예측 AI 경진대회

→ 환경 데이터로부터 청경채의 일별 잎 면적 증감률 예측하는 AI 알고리즘 개발

[DATA INFO]

- train_input (폴더)
  총 58개의 청경채 케이스 데이터 포함
  각 케이스별 환경 데이터 제공 (1분 간격)
- train_target (폴더)
  총 58개의 청경채 케이스 데이터 포함
  rate: 각 청경채 케이스의 잎 면적 증감률 (1일 간격)
- test_input (폴더)
  총 6개의 청경채 케이스 데이터 포함
  각 케이스별 환경 데이터 제공 (1분 간격)
- test_target (폴더)
  총 6개의 청경채 케이스 데이터 포함
  rate: 각 케이스의 잎 면적 증감률 (1일 간격, 제출 양식에서는 값이 0으로 가려져 있음)
- sample_submission.zip (제출 양식)
  테스트 케이스(6개)에 대한 일별 예측 결과를 포함
  제출 시 rate 값에 모델의 예측값을 채워야 함

---

1. 데이터 전처리

```
import pandas as pd
from sklearn.preprocessing import StandardScaler

# 데이터 불러오기
train_data = pd.read_csv('data/train_input.csv')
train_target = pd.read_csv('data/train_target.csv')

# 결측치 처리 및 필요 컬럼 선택
train_data = train_data.fillna(train_data.mean())
X = train_data[['내부온도관측치', '내부습도관측치', 'CO2관측치', 'EC관측치']]

# 데이터 스케일링
scaler = StandardScaler()
scaled_X = scaler.fit_transform(X)
```

설명: 데이터를 불러와 결측치를 평균값으로 대체하고, 주요 변수만 선택 후 스케일링한다.

2. 타겟 데이터 준비 및 데이터 분리

```
# 타겟 데이터 준비
y = train_target['rate']

# 학습용/테스트용 데이터 분리
from sklearn.model_selection import train_test_split
scaled_X_train, scaled_X_test, y_train, y_test = train_test_split(scaled_X,
y, test_size=0.2, random_state=42)
```

설명: 타겟 변수(`rate`)를 준비하고, 학습 데이터와 테스트 데이터를 8:2 비율로 분리한다

3. 데이터 증강 (Feature Engineering)

```
# 새로운 피처 생성
train_data['온습도_곱'] = train_data['내부온도관측치'] * train_data['내부습도관측
치']
train_data['CO2_EC_비율'] = train_data['CO2관측치'] / (train_data['EC관측치']
+ 1e-5)

# 스케일링된 데이터에 새로운 피처 추가
scaled_features = pd.DataFrame(scaled_X, columns=['내부온도관측치', '내부습도관
측치', 'CO2관측치', 'EC관측치'])
scaled_features['온습도_곱'] = train_data['온습도_곱']
scaled_features['CO2_EC_비율'] = train_data['CO2_EC_비율']
```

설명: 온도와 습도의 곱, CO2와 EC의 비율과 같은 새로운 피처를 생성하고 기존 데이터에 추가한다.

4. 피처 중요도 분석 (Feature Selection)

```
from sklearn.ensemble import RandomForestRegressor

# 임시 모델로 피처 중요도 분석
rf = RandomForestRegressor(random_state=42)
rf.fit(scaled_features, y)
feature_importances = pd.DataFrame({'Feature': scaled_features.columns, 'Im
portance': rf.feature_importances_}).sort_values(by='Importance', ascending
=False)

# 상위 중요 피처만 선택
selected_features = feature_importances['Feature'][:5].values
scaled_X_selected = scaled_features[selected_features]
```

설명: 랜덤 포레스트를 활용해 피처 중요도를 분석하고, 중요도가 높은 상위 5개의 피처를 선택한다.

---

5. 데이터 재분리

```
scaled_X_train, scaled_X_test, y_train, y_test = train_test_split(scaled_X_
selected, y, test_size=0.2, random_state=42)
```

설명: 중요 피처만 포함된 데이터를 학습 데이터와 테스트 데이터로 다시 분리한다.

---

6. 모델 생성

```
xg_reg = xgb.XGBRegressor(objective='reg:squarederror', max_depth=5)
xg_reg.fit(scaled_X_train, y_train)
pred = xg_reg.predict(scaled_X_test)
for k in pred:
    result = np.hstack([result, k])
result
```

설명: XGBoost Regressor를 사용해 학습 데이터를 기반으로 모델을 생성하고 테스트 데이터를 예측한다.

---

7. 결과값 로그 변환 및 압축

```
expm1_result = np.expm1(result)
expm1_result = pd.DataFrame(columns=['label'], data=expm1_result)
total_result = []
for case in tqdm(range(1, 7)):
    if case < 10:
        case = '0' + str(case)
    particular_label = pd.read_csv(f'data/test_target/TEST_{case}.csv')
    particular_label['rate'] = list(expm1_result['label'][:len(particular_l
abel)].values)
    expm1_result = expm1_result.drop([i for i in range(len(particular_labe
l))])
    expm1_result = expm1_result.reset_index(drop=True)
    total_result.append(particular_label)
for i in range(len(total_result)):
    total_result[i].to_csv(f'TEST_0{i+1}.csv', index=False)
file_ls = ['TEST_01.csv', 'TEST_02.csv', 'TEST_03.csv', 'TEST_04.csv', 'TES
T_05.csv', 'TEST_06.csv']
with zipfile.ZipFile("submission.zip", 'w') as my_zip:
    for i in file_ls:
        my_zip.write(i)
    my_zip.close()
```

설명: 모델의 예측 결과를 로그 변환 후, 각 테스트 케이스별로 `rate` 값을 추가해 CSV 파일로 저장하고 이를 압축하여 제출 파일을 생성한다.

배울 점

- **피처 중요도 기반 선택**: 랜덤 포레스트를 활용해 중요한 피처만 선택함으로써 데이터 차원을 줄이고 모델 효율성을 높인 점이 인상깊었음.
- **결과 처리 자동화**: 예측 결과를 CSV 파일로 저장하고 압축 파일로 제출 형식을 자동화한 과정은 반복 작업을 줄이는 데 유용하다는 걸 알게 됨.