르완다(Rwanda)

이산화탄소 배출량 예측 모델

목차

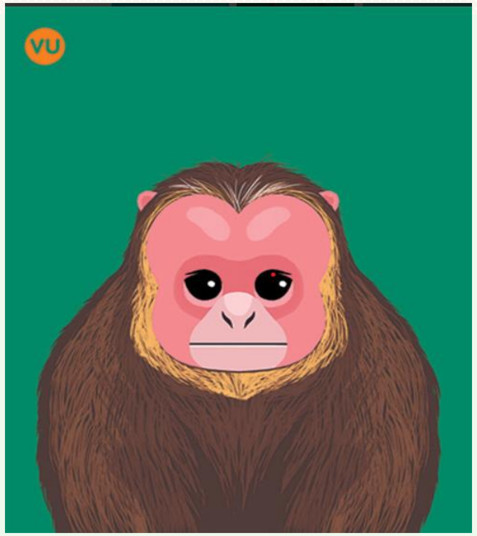
Part 1

프로젝트 서론

팀원소개

전정우

- PPT 제작
- 자료조사



김준성

- 모델링
  - CatBoost
  - Gradient Boost
  - HistGradient Boost
- 상관성 분석

정희지

- 모델링
  - Xgboost
  - ExtraTree
- 데이터 시각화
- 자료조사

강유정

- 모델링
  - RandomForest
  - LightGBM
- 데이터 시각화
- 자료조사

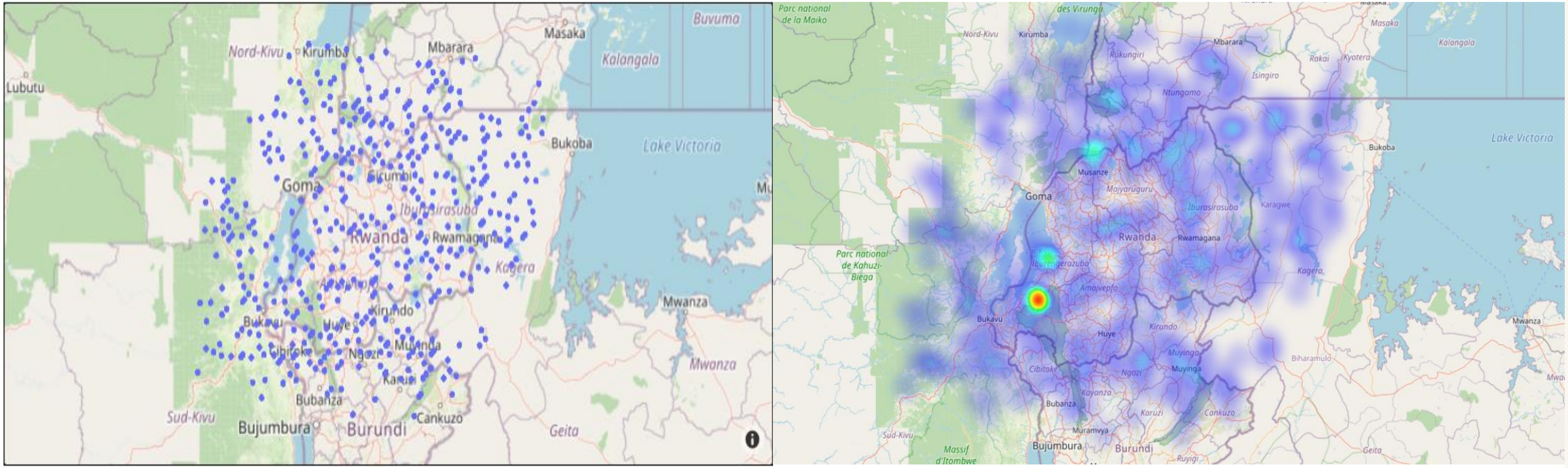
주제 배경>르완다(Rwanda)

1. 르완다는 2019년까지 10년간 연평균 경제성장률 7.2%, 2020년은 코로나19로 인하여 전년에 비해 대략 6%감소했다.
2. 해외직접투자(FDI),공적개발원조(ODA)를 적극적으로 받기 위해 정부 내 투명성을 확보하고, 효율성을 극대화하였다.
3. 다른 사하라이남 국가들에 비해 개발협력사업 진행상의 투명성이 확보되어있고, 효율적으로 진척이 이루어지는 것으로 알려져있다.
4. 다른 사하라 이남 국가들이 마이너스 성장을 하는 동안 르완다는 꾸준한 경제성장을 이루었다.
5. 동아프리카안에서도 작은 나라인 르완다에서 아프리카의 많은 주요 국제회의가 개최됨

주제 배경 > 르완다(Rwanda)

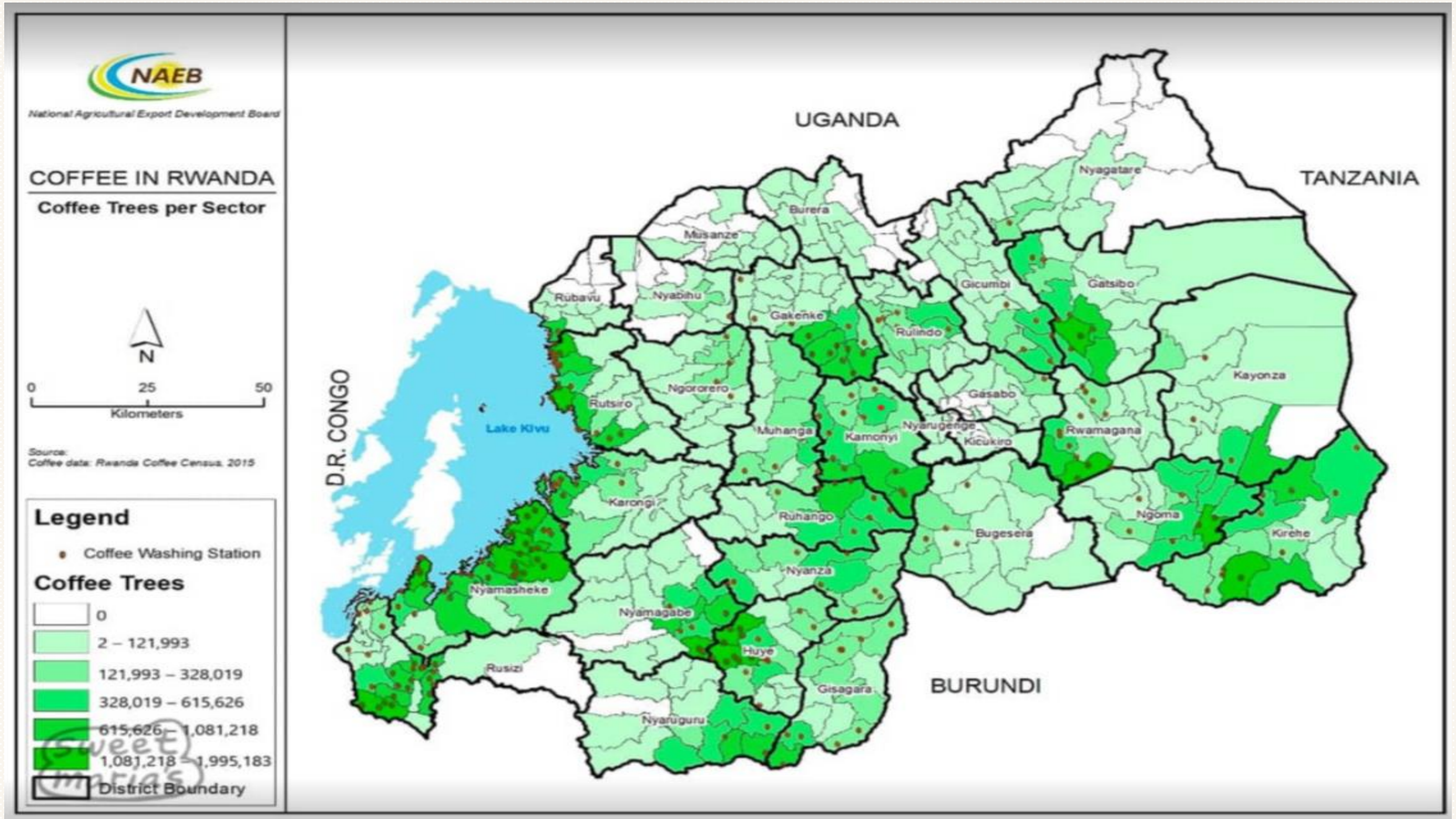
- 고산 지대 (평균 고도 1598m)
- 연중 20℃의 온화한 기후, 풍부한 강수량
- 아프리카 안에서 가장 조밀한 인구분포
- 인구의 90%가 농업에 종사하지만
  토지가 극단적으로 부족

주제 배경



르완다의 emission(이산화탄소) 지도 시각화

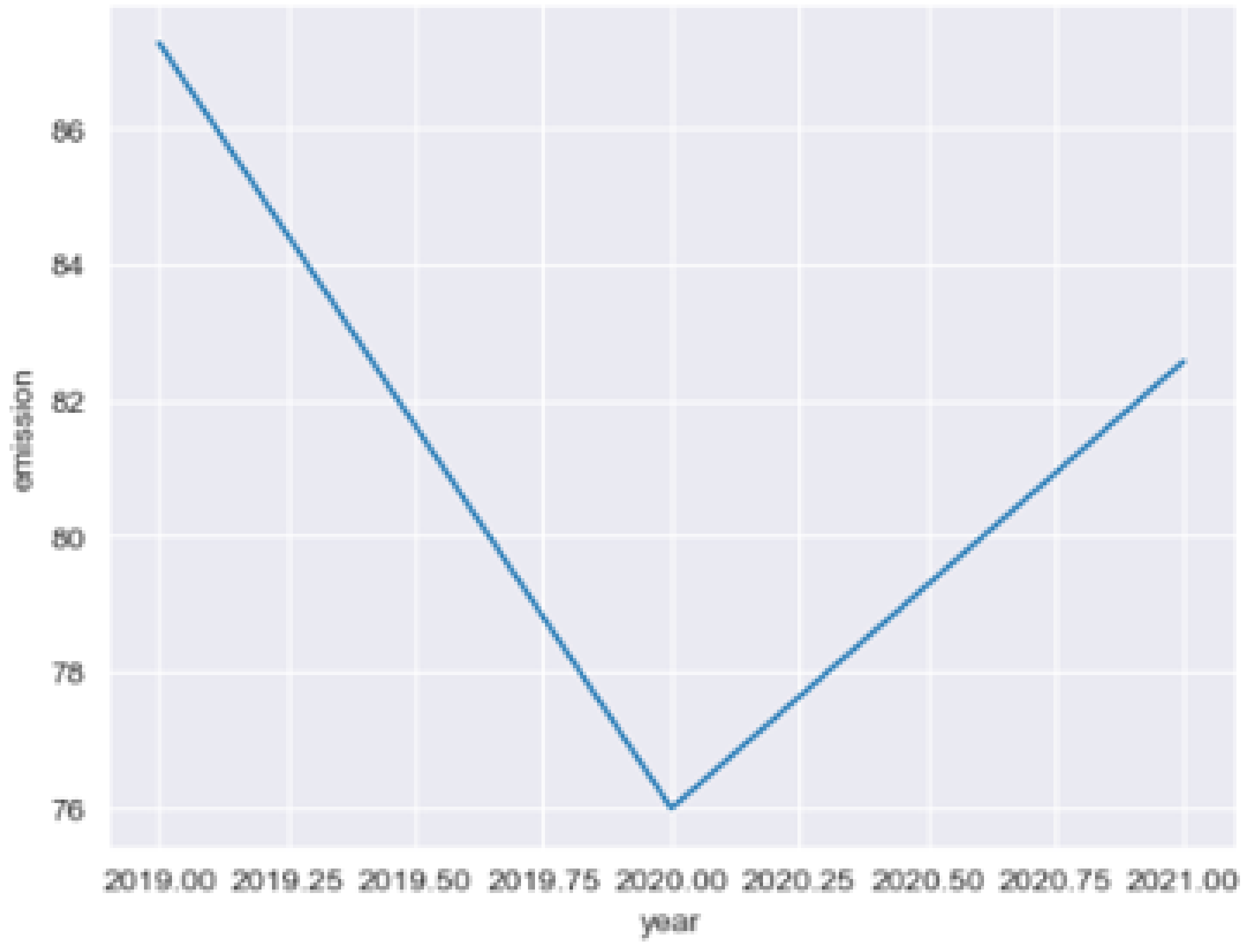
주제 배경



커피 공장이 있는 곳에서 가장 많은 양 배출

주제 배경

2019 – 2021년 르완다 이산화탄소 배출량



주제 배경

2014 – 2022년 르완다GDP 성장률

평가기준

Root Mean Squared Error (RMSE)

Submissions are scored on the root mean squared error. RMSE is defined as:

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$$

where $\hat{y}_i$ is the predicted value and y_i is the original value for each instance i.

RMSE(Root Mean Square Error)

가장 값이 작은 것이 좋은 모델

사용 라이브러리

LightGBM

scikits learn

pandas

Folium

NumPy

seaborn

CatBoost

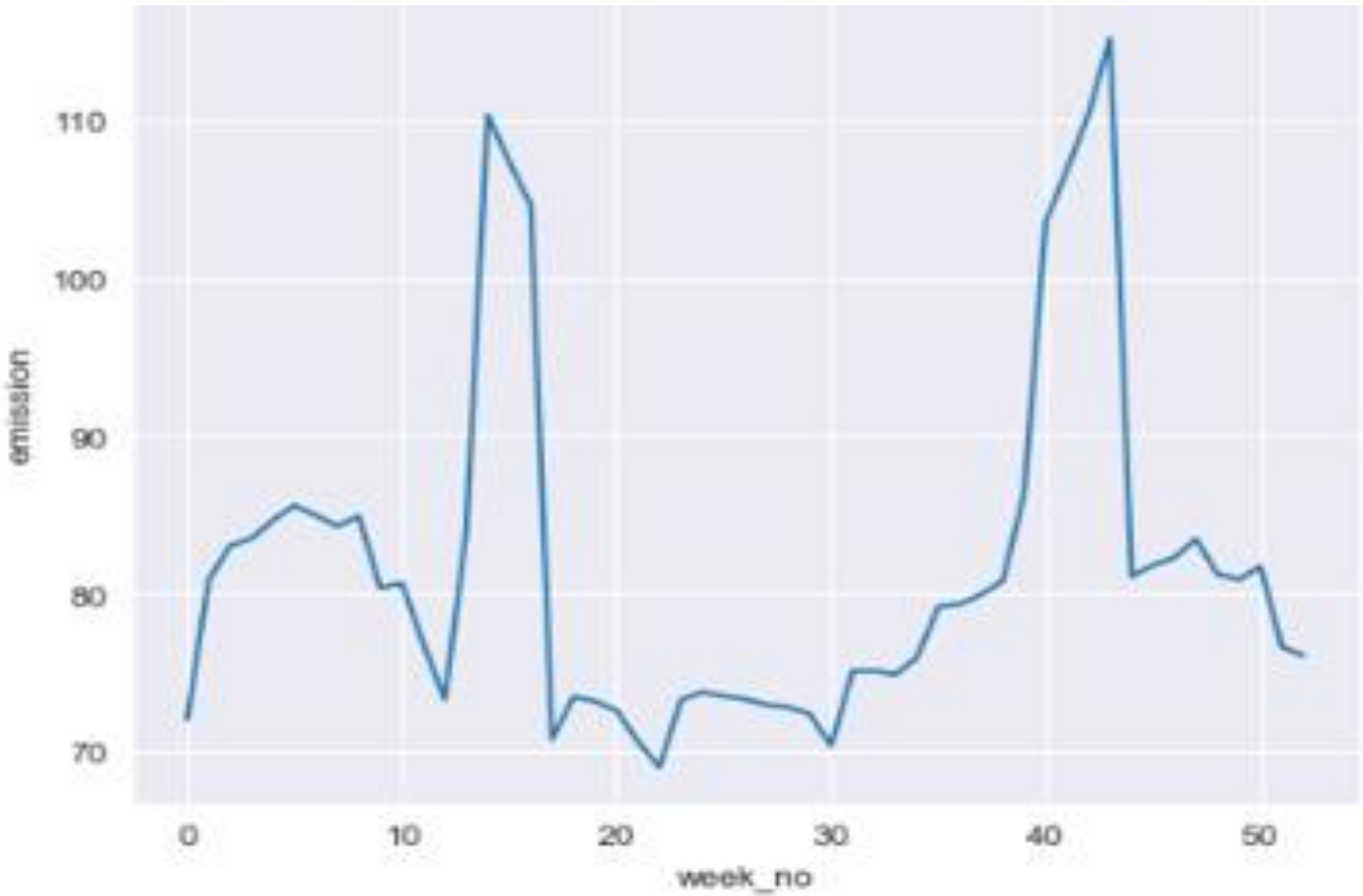
matplotlib

본론

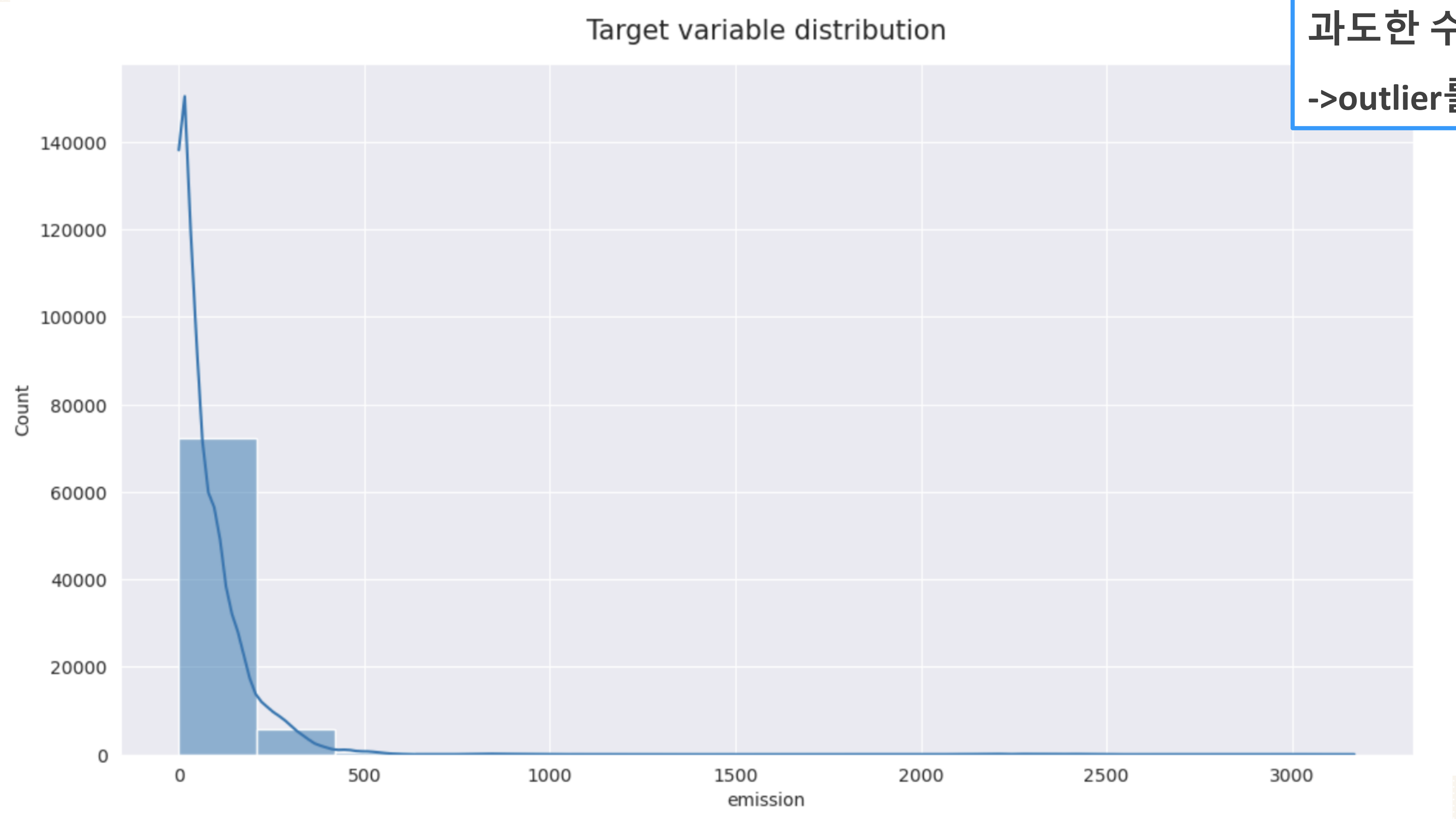
Part 2

데이터 분석

주차별 이산화탄소 배출량 변화



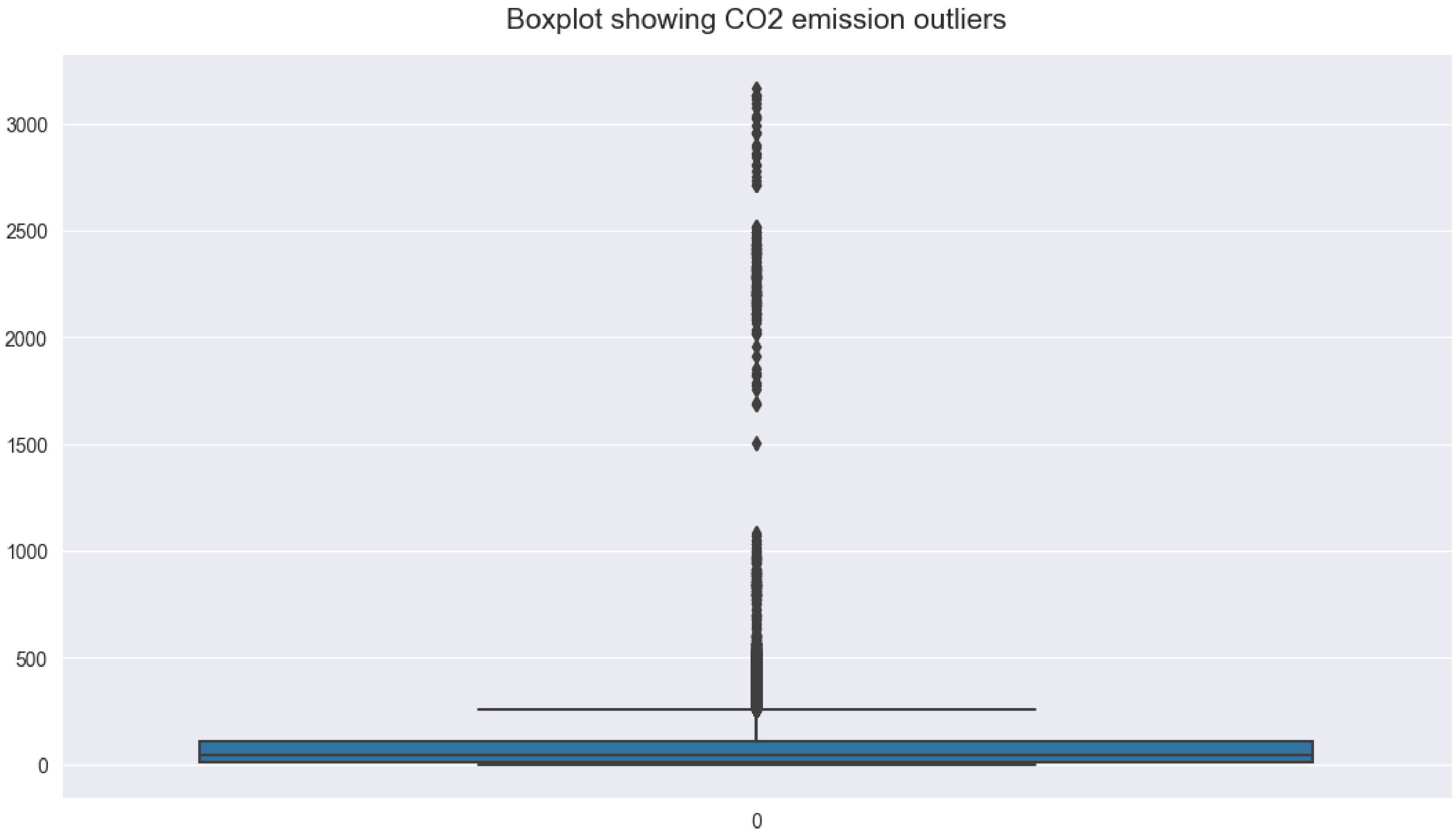
이산화탄소 배출량 분포



과도한 수치의 emission이 발생

->outlier를 자세히 살펴보기로 함

이산화탄소 배출량 분포



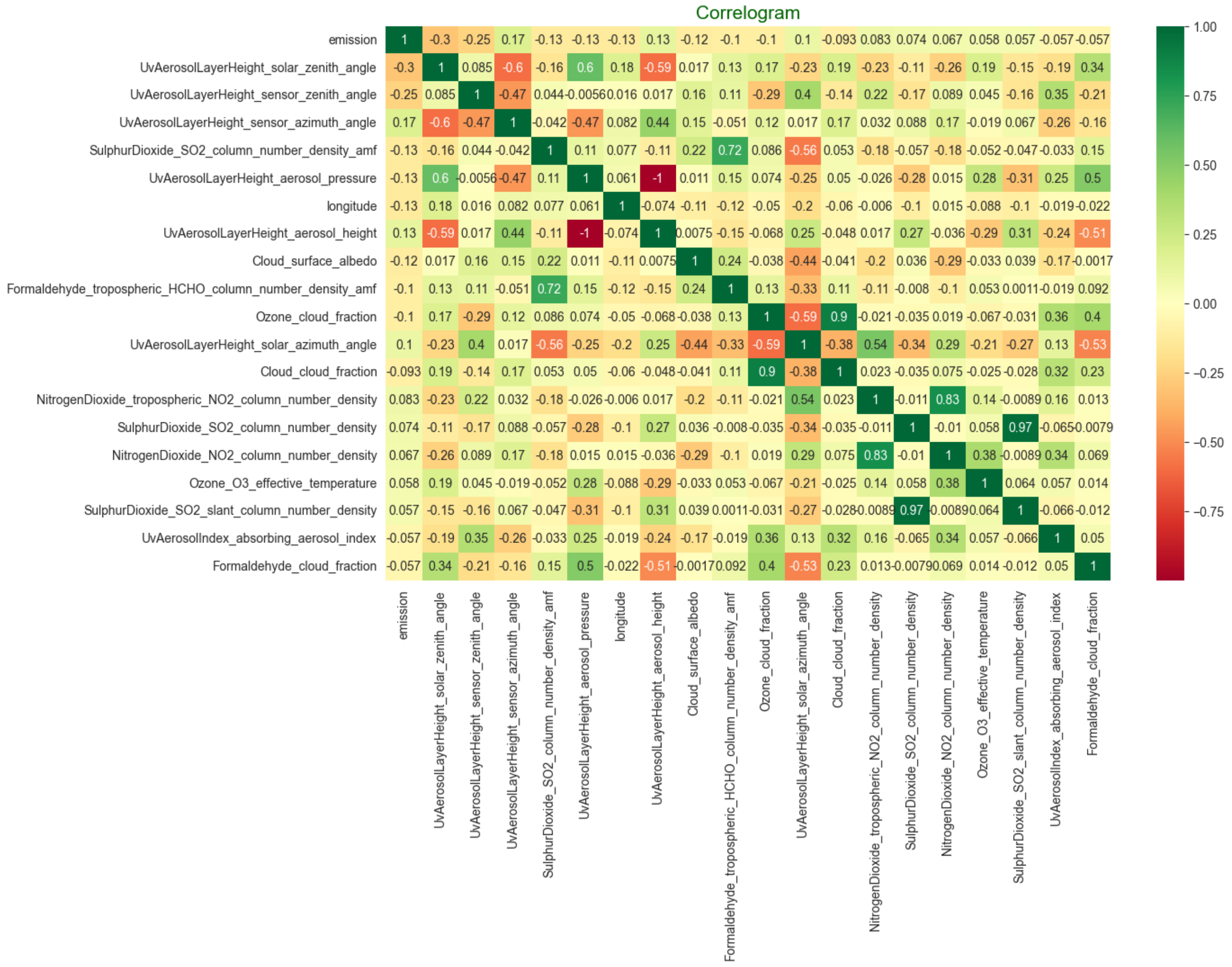
outlier가 3그룹으로

나눠지는 것으로 보임.

상관관계 분석

250-1500, 1500-2500, over 2500

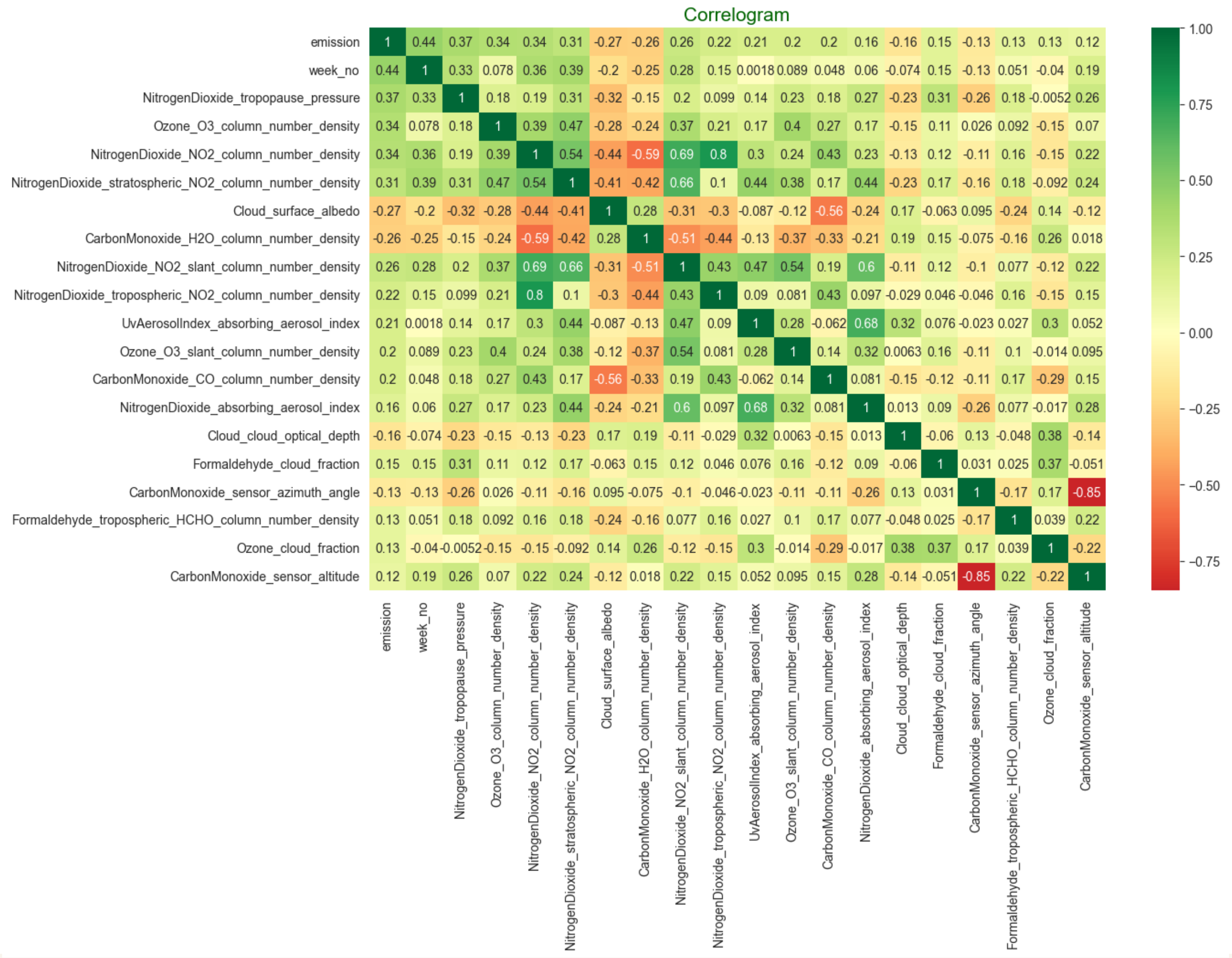
상관관계 분석



emission 250 ~ 1500 에서의 correlation모음

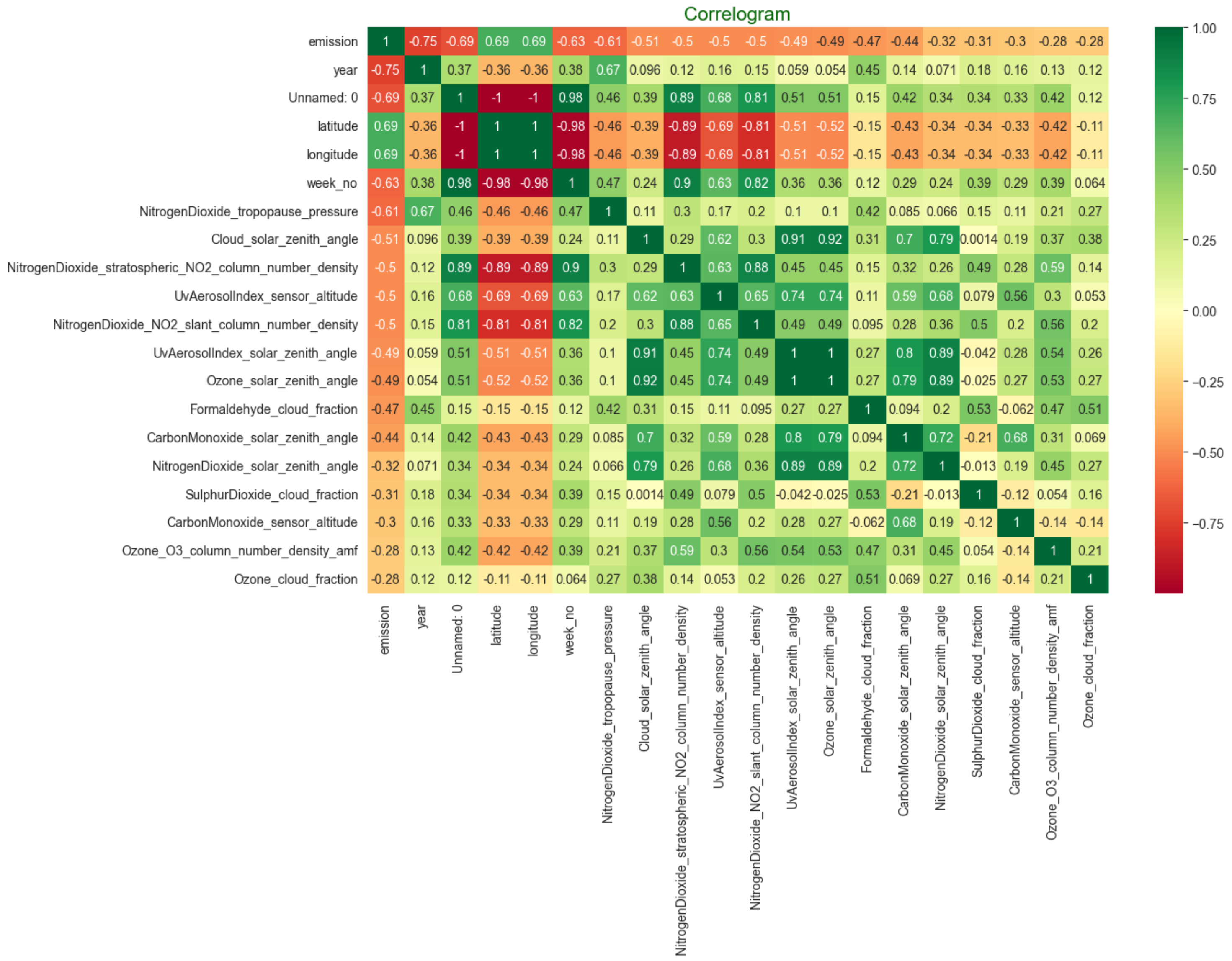
Part 2

상관관계 분석



emission 1500 ~ 2500 에서의 correlation모음

상관관계 분석



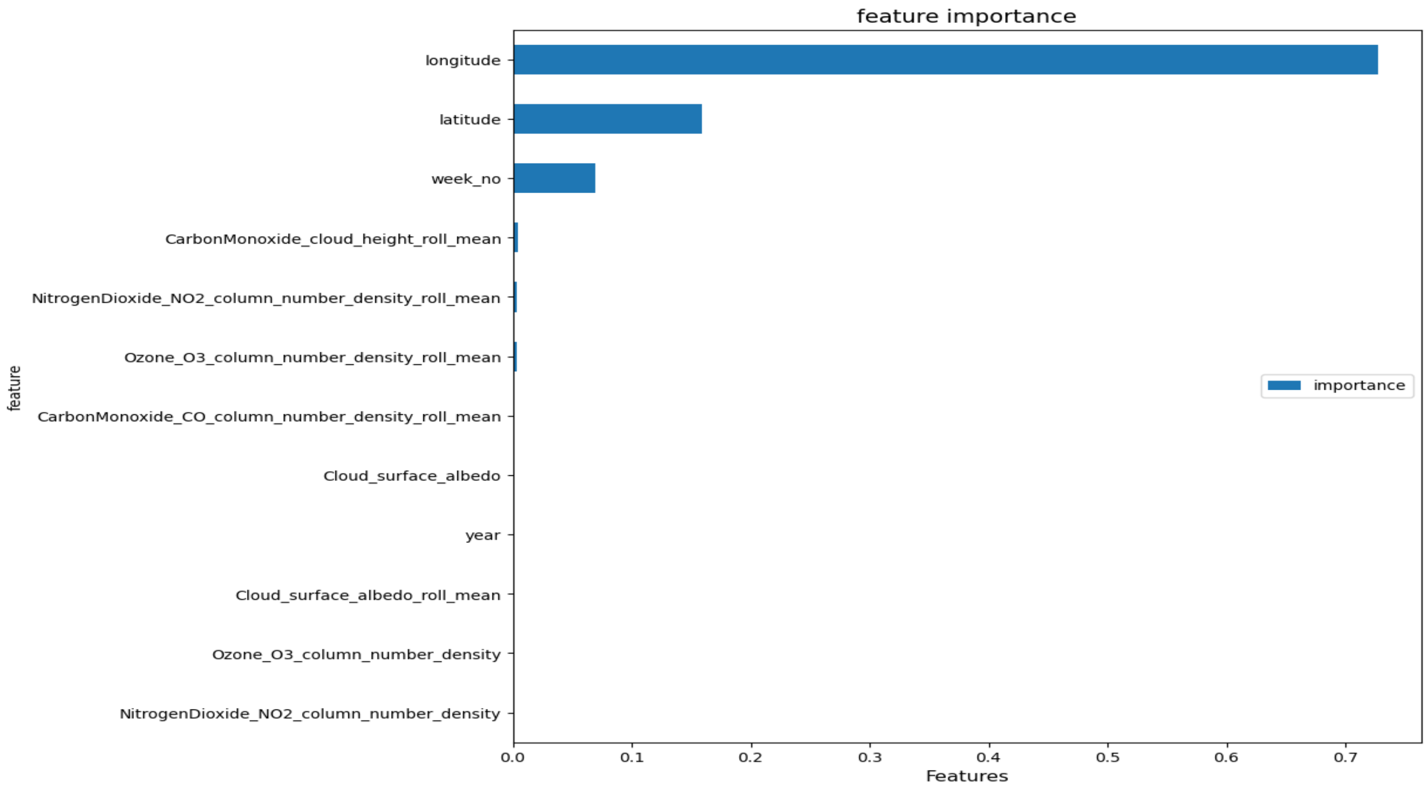
emission 2500 이상에서의 correlation모음

* 상관성 수치 차이 심함
-> feature importance를
분석 해보기로 함

특성 중요도

```
# 특성 중요도
impo_df = pd.DataFrame({'feature': X.columns, 'importance': clf.feature_importances_}).set_index('feature').sort_values(by = 'importance', ascending = False)
impo_df = impo_df[:12].sort_values(by = 'importance', ascending = True)
impo_df.plot(kind = 'barh', figsize = (10, 10))
plt.legend(loc = 'center right')
plt.title('feature importance', fontsize = 14)
plt.xlabel('Features', fontsize = 12)
plt.show()
```

✓ 1.2s



* random forest regression

feature importance

본론

Part 3

모델링

사용 모델

Random Forest	lightGBM	ExtraTree Regressor
XGBRegressor	CatBoost Regressor	Hist Gradient Boosting Regressor

Random Forest Regressor

Random Forest Regressor

RMSE Score

27.662948073269565

```
X = train_eng.drop(['ID_LAT_LON_YEAR_WEEK', 'location', 'emission'], axis = 1).fillna(train_eng.mean())
y = train_eng.emission

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = SEED)

clf = RandomForestRegressor( n_jobs=-1)
clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)

print(f'RMSE Score: {mean_squared_error(y_test, y_pred, squared=False)}')
```

✓ 8m 17.7s

```
RMSE Score: 27.662948073269565
```

Light GBM Regressor

lightGBM

RMSE Score

17.593425572791297

```
from lightgbm import LGBMRegressor
import numpy as np
from sklearn.model_selection import KFold

lgbm_model = LGBMRegressor(n_estimators=150,
                           learning_rate=0.2,
                           min_child_samples=40,
                           num_leaves=60
                           )

n_splits = 5
kf = KFold(n_splits=n_splits, shuffle=True)

lgbm_predictions = np.zeros(len(X))
lgbm_true_labels = np.zeros(len(X))
lgbm_test_predictions = np.zeros(len(df_test))

for fold, (train_idx, val_idx) in enumerate(kf.split(X, y)):
    X_train, X_val = X.iloc[train_idx], X.iloc[val_idx]
    y_train, y_val = y.iloc[train_idx], y.iloc[val_idx]

    lgbm_model.fit(X_train, y_train, eval_set=[(X_val, y_val)])
    lgbm_fold_preds = lgbm_model.predict(X_val)

    lgbm_fold_test_preds = lgbm_model.predict(df_test)

    lgbm_predictions[val_idx] = lgbm_fold_preds
    lgbm_true_labels[val_idx] = y_val
    lgbm_test_predictions += lgbm_fold_test_preds / n_splits

overall_metric_lgbm = np.sqrt(np.mean((lgbm_true_labels - lgbm_predictions) ** 2))
print("전체적인 RMSE (LGBMRegressor):", overall_metric_lgbm)
```

[41] ✓ 4.3s

Extra Tree Regressor, XGB Regressor

ExtraTree Regressor

RMSE Score

27.17834657953462

```
model = ExtraTreeRegressor(random_state=400,splitter='best')
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

```
print(f'RMSE Score: {mean_squared_error(y_test, y_pred, squared=False)}')
```

RMSE Score: 27.17834657953462

XGBRegressor

RMSE Score

24.717975327931423

```
xgb = XGBRegressor(n_estimators=400,learning_rate=0.2)
xgb.fit(X_train,y_train)
y_pred = xgb.predict(X_test)
```

```
print(f'RMSE Score: {mean_squared_error(y_test, y_pred, squared=False)}')
```

RMSE Score: 24.717975327931423

CatBoost Regressor

CatBoostRegressor

RMSE Score

8.7664498949419

```
cb = CatBoostRegressor(n_estimators=400,learning_rate=0.2)
cb.fit(X_train,y_train)
y_pred = cb.predict(X_test)
```

20s 9ms 2023.08.08 16:14:14에 실행되었습니다

```
389:	learn: 1.4016916	total: 18.9s	remaining: 485ms
390:	learn: 1.3983647	total: 18.9s	remaining: 436ms
391:	learn: 1.3938529	total: 19s	remaining: 387ms
392:	learn: 1.3923537	total: 19s	remaining: 339ms
393:	learn: 1.3880468	total: 19.1s	remaining: 291ms
394:	learn: 1.3865617	total: 19.3s	remaining: 244ms
395:	learn: 1.3794399	total: 19.3s	remaining: 195ms
396:	learn: 1.3775355	total: 19.4s	remaining: 147ms
397:	learn: 1.3743470	total: 19.4s	remaining: 97.6ms
398:	learn: 1.3716283	total: 19.5s	remaining: 48.8ms
399:	learn: 1.3683206	total: 19.5s	remaining: 0us
```

```
print(f'RMSE Score: {mean_squared_error(y_test, y_pred, squared=False)}')
```

```
RMSE Score: 8.76644989494919
```

Hist Gradient Boosting Regressor

Hist Gradient

Boosting Regressor

RMSE Score

24.720655581502935

```
# Split the data into features (X) and the target variable (y)
X = train_df.drop('emission', axis=1)
y = train_df['emission']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize the HistGradientBoostingRegressor model
hb = HistGradientBoostingRegressor(learning_rate=0.2)

# Fit the model to the training data
hb.fit(X_train, y_train)

# Make predictions on the test data
y_pred = hb.predict(X_test)

# Calculate the Root Mean Squared Error (RMSE) to evaluate the performance
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
print("Root Mean Squared Error:", rmse)
```

4s 883ms 2023.08.08 16:39:59에 실행되었습니다

```
Root Mean Squared Error: 24.720655581502935
```

Part 4

프로젝트 결론

X

프로젝트 결론

CatBoost Regressor의 점수가 가장 좋게 나타남

그외에 좋게 나왔던, xgboost, rf를 사용하여 앙상블 모델을 만듬

Ensemble 기법을 사용하여 해당 모델들을 기획

```
score_list, oof_list = pd.DataFrame(), pd.DataFrame()

models = [
    ('rf', RandomForestRegressor(random_state = seed)),
    ('et', ExtraTreesRegressor(random_state = seed)),
    ('xgb', XGBRegressor(random_state = seed)),
    ('lgb', LGBMRegressor(random_state = seed)),
    ('cb', CatBoostRegressor(random_state = seed, verbose = 0)),
    ('hgb', HistGradientBoostingRegressor(random_state = seed))
]
```

프로젝트 결론

```
Val Score: 0.50247 ± 0.01991 | Train Score: 0.43838 ± 0.00570 | rf
Val Score: 0.50185 ± 0.01989 | Train Score: 0.43638 ± 0.00641 | et
Val Score: 0.55309 ± 0.02013 | Train Score: 0.50559 ± 0.00124 | xgb
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000822 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 310
[LightGBM] [Info] Number of data points in the train set: 52682, number of used features: 3
[LightGBM] [Info] Start training from score 3.199796
[LightGBM] [Warning] Auto-choosing col-wise multi-threading, the overhead of testing was 0.001075 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 308
[LightGBM] [Info] Number of data points in the train set: 52682, number of used features: 2
[LightGBM] [Info] Start training from score 3.301991
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000559 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 310
[LightGBM] [Info] Number of data points in the train set: 52682, number of used features: 3
[LightGBM] [Info] Start training from score 3.252410
Val Score: 1.25611 ± 0.01445 | Train Score: 1.24069 ± 0.00729 | lgb
Val Score: 1.23242 ± 0.02062 | Train Score: 1.21744 ± 0.00760 | cb
Val Score: 1.20318 ± 0.01194 | Train Score: 1.18645 ± 0.00484 | hgb
```

Validation과 train set의 오차가 작은 것을 확인 할 수 있었음

31.1998

프로젝트 결론

데이터를 어떻게 편집 하느냐에 따라서도 점수가 많이 변환됨

결측치를 어떻게 조정하는가에 따라 머신러닝 점수가 매우 많이 변함

실제 모델예측과 현실과 다른 부분이 존재

THANK YOU