Kaggle 프로젝트 보고서

과목 : 머신러닝
담당 교수 : 김성은 교수님
팀 : ST_ML2025_59
이름(학과) : 기현빈(기계시스템디자인공학과)
인공지능반도체 연계융합 복수전공

머리말

본 보고서는 머신러닝 과목의 프로젝트인 Kaggle의 '2025 1학기 서울과학기술대학교 머신러닝1 프로젝트'의 일환으로 작성하였다. 이 프로젝트의 목표는 다양한 기상 관측 데이터를 바탕으로 다음 날의 평균 기온 편차를 예측하는 머신러닝 모델을 구축하는 것이다. 날씨의 데이터의 복잡하고 비선형적인 기후 패턴을 분석하고 예측하는데 있어서 날씨의 특성을 이용한 여러 feature를 만들고 모델을 적용시킨다. 특히 본 프로젝트에서는 XGBoost, CatBoost, Ridge Regression 모델을 stacking 방식으로 결합하여 기온 예측의 정확도를 극대화하고자 하였다. 이 글에서는 머신러닝의 전반적 과정을 목차로 나누어 다룬다.

목차

1. 데이터셋 설명

train data set은 서울, 동두천, 강화, 인천, 이천, 양평 관측소의 2019-2024년 데이터셋이며, test dataset은 파주, 수원 관측소 데이터셋 (random suffle)이다. 제공되는 입력변수는 다음과 같다. (n은 0부터 23까지의 정수로 시간을 의미합니다)

id: 순서
station: 지상관측소 번호
station_name: 지상관측소 이름
date: 날짜(월-일)
cloud_cover_n: 증하층운량(10분위)
dew_point_n: 이슬점 온도(℃)
humidity_n: 습도(%)
local_pressure_n: 현지기압(hPa)
min_cloud_height_n: 최저운고(100m)
precipitation_n: 강수량(mm)
sea_level_pressure_n: 해면기압(hPa)
snow_depth_n: 적설(cm)
sunshine_duration_n: 일조(hr)
surface_temp_n: 지면온도(℃)
vapor_pressure_n: 증기압(hPa)
visibility_n: 시정(10m)
wind_speed_n: 풍속(m/s)
wind_direction_n: 풍향(°)
climatology_temp: 해당 날짜의 평균 기온(℃) (7년 평균 - 1월 1일인 경우 2018년부터 2024년까지 1월 1일 평균)

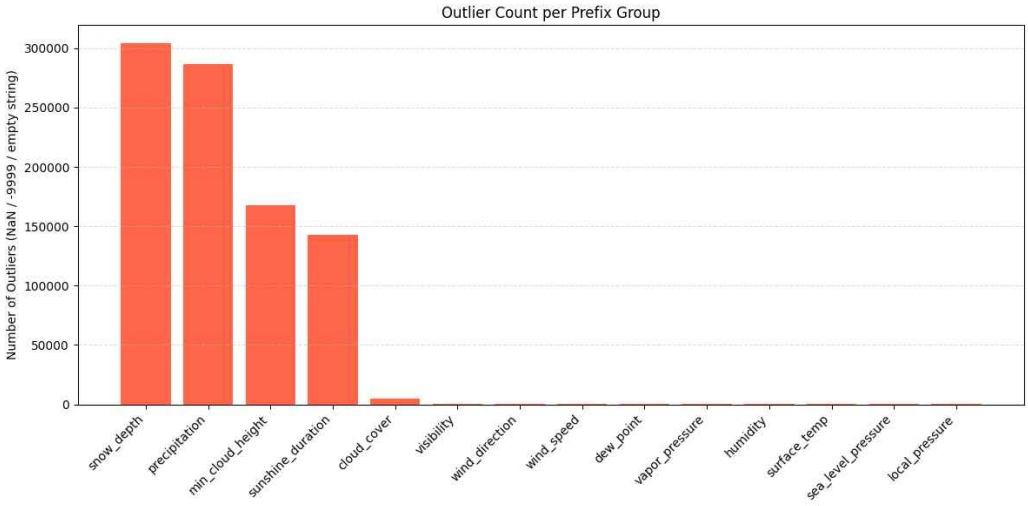
2. 데이터 전처리

2-1) 결측치 처리
결측치 처리는 kaggle의 결측치 처리 안내를 따른다. kaggle에는 다음과 같이 안내되어 있다.

* 결측치 처리:
-9999: 관측소 기계에서 감지한 결측치 또는 이상치
해당 칸이 비어있을 경우(NaN): 변수에 따라 의미가 달라집니다. snow_depth의 경우 눈이 오지 않았거나 sunshine_duration의 경우 해가 진 시간을 의미할 수 있습니다. 하지만 대부분 수치가 입력되어 있지만 일부분 비어있는 경우 결측치일 가능성이 있습니다.
변수의 의미를 기반으로 결측치의 형태를 파악하고 결측치를 보간 및 구간화하거나, 결측치가 너무 많은 경우 해당 변수를 제외하는 등 다양한 방법을 구상해보세요. (target에는 NaN값 또는 -9999가 없습니

다)

이에 따라서 -9999값은 NaN으로 반환하여 처리한다. 시간별 데이터 중 sunshine_duration_n: 일조(hr), snow_depth_n: 적설(cm), precipitation_n: 강수량(mm), min_cloud_height_n: 최저운고(100m)를 0으로 대체하고, 나머지 기상변수는 보간처리를 한다. 날씨의 변수는 연속적인 값임을 들어 앞뒤 두 값을 찾아 두 값을 직선위로 잇고, 그 선위에 값을 채워 넣는 선형 보간을 적용한다.

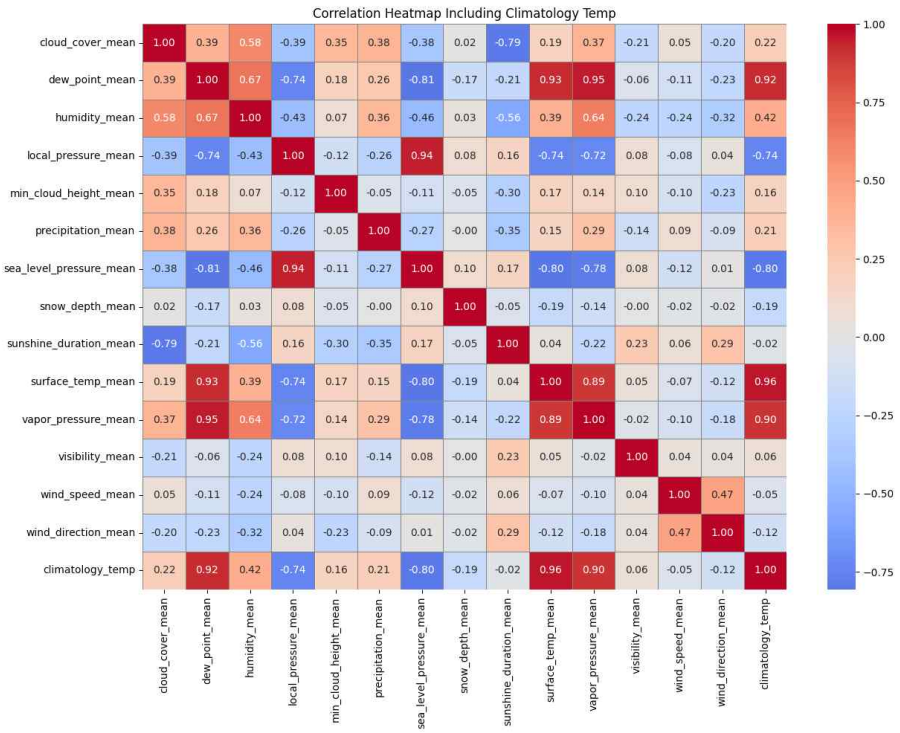


2-2) feature 설정

날씨의 특성들은 지구의 자전과 공전에 의해 하루 주기와 년 주기로 주기 안에서 같은 양상으로 반복되는 경향이 있다. 이를 표현하기 위해 다음과 같은 feature를 설정한다.

1. 데이터 분석

히트맵을 통해 (cloud_cover_mean, sunshine_duration_mean), (dew_point_mean, local_pressure_mean) 등의 관계가 상관성이 높은 것을 알 수 있다.



2. 계절의 주기성

day_sin, day_cos : 해당 날짜가 1년의 어디에 포함되어 있는지 알기 위해서 sin과 cos를 사용한다.

3. 날짜 처리

1년 단위의 날씨의 반복을 표현하기 위해 다음과 같은 feature를 만든다.

- dayofyear(1년 중 몇 번째 날), weekofyear(1년 중 몇 번째 주)

사람의 활동과 날씨의 관계를 표현하기 위해 다음의 feature를 만든다.

- weekday(요일을 one-hot coding을 한다.) ,is_weekend(토요일 또는 일요일은 1, 그 외는 0)

4. 날씨의 지리적 특성을 이용한 여러 파생 feature

vapor_pressure_features()

feature	의미
vapor_mean_0_23	하루(0~23시) 동안의 평균 수증기압
temp_18_23_avg	저녁(18~23시) 시간대의 평균 지면 온도
vapor_18_23_avg	저녁 시간대의 평균 수증기압
temp_vapor_night_interact	저녁 온도 × 수증기압 (상호작용 항, 야간 열기/수분 결합 특성)
prefix_18_23_avg *(8개)*	dew_point, humidity, local_pressure, sea_level_pressure, wind_speed, wind_direction, cloud_cover, visibility각 항목에 대해 18~23시 평균
prefix_vapor_night_interact *(8개)*	위 평균값 × vapor_18_23_avg(상호작용 항)
vapor_min	하루 중 수증기압의 최솟값
vapor_diff_23_12	정오(12시) 대비 자정(23시) 수증기압 차이
vapor_night_change	새벽(0시) 대비 밤(23시) 수증기압 변화
temp23_x_vapor23	23시 기온 × 23시 수증기압 (특정 시각 상호작용)

add_hourly_custom_features()

Feature Name	의미
temp_night_change	23시 - 0시 기온 변화 (야간 기온 변화)
dew_point_shift	18시 - 6시 이슬점 차이
dew_point_change_night	23시 - 18시 이슬점 변화
humidity_change_night	23시 - 18시 습도 변화

temp_evening_slope	19시 → 23시 기온 변화량
dew_temp_diff_23	23시 기온 - 23시 이슬점
humidity_mean_20_23	20~23시 습도 평균
surface_temp_std_19_23	19~23시 기온의 표준편차 (변동성)
humidity_min_20_23	20~23시 습도 최솟값
temp_x_humidity_19	19시 기온 × 23시 습도
dew_ratio_23	23시 이슬점 / (19시 기온 + ε)
humidity_log_23	23시 습도 로그값 (정규화 및 분포 안정화)
dew_x_humidity_23	23시 이슬점 × 23시 습도
dew_temp_gap_sq_19	(19시 기온 - 23시 이슬점)² (이슬점과 기온 차이의 제곱 → 응결 위험 등)
altitude_adj_temp_19	해발고도 보정 19시 기온 (지형 영향 반영)

add_advanced_station_features()

Feature Name	의미
surface_temp_adjust_ratio	지면 온도 조정 비율— 해발고도 / 기온계 높이
wind_exposure_index	풍속 노출 지수— 풍속계 높이 / 해발고도
sensor_height_std	센서 높이 표준편차— 기온/풍속/기압/강우계 높이의 변동성
night_temp_variation_adjusted	고도 보정 야간 기온 변화— 야간 기온 변화 × 고도
dew_x_temp_adjust_ratio	이슬점과 온도 보정 곱— 이슬점 × 온도 조정 비율
lat_x_lon	위도 × 경도— 위치 좌표의 곱
lat_div_lon	위도 ÷ 경도— 위도와 경도의 비율
lat_plus_lon	위도 + 경도— 위치 좌표의 합
lat_minus_lon	위도 - 경도— 위치 좌표의 차

add_lat_lon_combination()

Feature Name	설명
lat_x_lon	위도와 경도의 곱
lat_div_lon	위도를 경도로 나눈 값
lat_plus_lon	위도와 경도의 합
lat_minus_lon	위도에서 경도를 뺀 값

add_polynomial_features()

Feature Name	설명
lat_x_dew23	위도 × 이슬점
lat_x_humidity23	위도 × 습도
temp19_x_humidity23	기온 × 습도
ltitude_x_temp0	고도 × 야간기온변화
ew_div_humidity	이슬점 ÷ 습도 (건조지수)

2-3) 범주형 변수 인코딩

범주형 변수들은 모두 one-hot 인코딩해준다.

2-4) 이상치 제거

optuna를 이용해 hyperparmetor을 튜닝한다. optuna를 이용해 이상치 제거를 위한 클리핑에 사용된 변수를 튜닝해 1차로 -20 ~ 20사이의 값을 필터링하고 2차로 남아있는 값도 -10 ~ 10범위로 제한한다.

2-5) 훈련-검증 데이터 분할

전체 데이터의 80%는 모델 학습용으로 사용하고 20%는 성능 평가용으로 사용한다.

4. 데이터 모델

4-1) 모델 선정 이유

이 프로젝트에 사용된 학습 모델은 XGBoost, CatBoost, Ridgid Regression을 사용한다. 먼저 기초 모델은 XGBoost와 CatBoost이다. XGBoost는 tree로 tree를 여러 개 만들어가면서 잔여 오차를 줄여가는 boosting 방법을 사용하며 높은 정확도와 빠른 속도 eemddml 이유로 kaggle이나 ai 경진 대회에서 자주 쓰이는 모델이다. CatBoost도 XGBoost처럼 tree의 오차를 줄여가는 방식으로 작동한다. 하지만 방식에서 차이가 있다. 대표적으로 범주형 변수를 처리하는 방식에서 차이를 보인다. CatBoost에서는 자동으로 범주형 변수를 처리한다. 범주형 변수를 값으로 나타내기 위해 이전의 target의 값의 평균을 사용해 표현한다. 때문에 그 범주형 변수가 target과 어떠한 관계였는지의 수치를 보여줄 수 있게 된다. 그렇다면 왜 XGBoost와 CatBoost를 둘 다 사용하는 것일까? CatBoost는 범주형 변수에 강한데 비해 XGboost는 과적합 방지와 트리 성장 방식에서 다른 학습 방식을 사용해 Catboost는 일관성 있고 안정적인 구조를 만드는 반면 XGboost는 더 복잡하고 세밀한 경계를 잡을 수 있다.

항목	CatBoost	XGBoost
과적합 방지	Ordered Boosting (데이터 순서 사용)	Early Stopping, 정규화 등
트리 성장 방식	Oblivious Tree (대칭형)	일반적인 Decision Tree (비대칭)

Ridgid Regression 모델은 이 두 모델을 stacking 하는데 사용된다. stacking에서는 XGBoost와 CatBoost가 각각 만든 예측값을 새 feature로 만들고 이를 Ridgid Regression로 학습한다. stacking을 통해 XGBoos의 세밀함과 CatBoost의 안정성을 모두 살려 일반화된 예측을 하도록 해준다.

날씨 데이터를 예측하는데 있어서 이러한 모델의 특성은 큰 장점이 된다. 먼저 CatBoost는 범주형 데이터를 다루는데 능하다는 점에 있어서 station_name, weekday같은 문자 데이터의 target과의 상관성을 강조할 수 있다. 또한 XGboost는 변수간의 복잡한 관계를 잘 설명한다는데 있어서 비선형 데이터인 날씨의 패턴을 파악하는데 장점을 가진다. 이를 Ridgid Regression으로 조합하면서 두 모델의 장점을 이용해 예측 결과를 합침으로써 더 안정적인 최종 예측을 만들 수 있다.

4-2) 모델에 사용된 hyperparameter의 설정

1. XGBoost, CatBoost

XGBoost 모델과 CatBoost모델에서는 hyperparameter는 Optuna를 사용해 RMSE를 최소화하는 방향으로 튜닝하여 정하였다. optuna를 사용해 다음의 hyperparameter을 설정하였다.

XGBRegressor 모델 :

```
n_estimators=1000
learning_rate=0.0454
max_depth=6
subsample=0.9027
colsample_bytree=0.5553
reg_alpha=0.8207
reg_lambda=0.7416
random_state=42
```

Catboost 모델 :

iterations=1000
learning_rate=0.09982049649669206
depth=
l2_leaf_reg=3.0653347605901677
random_strength=1.980696940191291
bagging_temperature=0.4021996337809488
random_state=42
verbose=0

2.Ridgid Regression

stacking에서는 hyperparameter인 정규화 강도 alpha를 설정해야 한다. 클수록 과적합 방지 효과가 커진다.

	alpha = 0.1	alpha = 1
R^2 Score	0.83488	0.84214

5. 성능 평가

RMSE와 R^2 Score로 평가한다. RMSE는 작을수록, R^2 Score는 1에 가까울수록 잘 설명했음을 의미한다.

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}} \qquad R^2 = 1 - \frac{SS_{RES}}{SS_{TOT}} = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \overline{y})^2}$$

모델	R^2 Score (kaggle 기준)
XGBoost	0.80004
XGBoost와 CatBoost를 stacking	0.84217