제주도 도로 교통량 예측 AI 경진대회

2022.10.24~2022.11.14

팀명 : 치타델레

주최: 제주 테크노파크, 제주특별자치도

주관: 데이콘

CONTENTS

CONTENTS

Introduction

1. **배경**

   제주도내 주민등록인구는 2022년 기준 약 68만명으로, 연평균 1.3%정도 매년 증가하고 있습니다.
   또한 외국인과 관광객까지 고려하면 전체 상주인구는 90만명을 넘을 것으로 추정되며,
   제주도민 증가와 외국인의 증가로 현재 제주도의 교통체증이 심각한 문제로 떠오르고 있습니다.

2. **주제**

   제주도 도로 교통량 예측 AI 알고리즘 개발

3. **설명**

   제주도의 교통 정보로부터 도로 교통량 회귀 예측

Introduction

4. 제공 데이터

- train.csv
  - 2021.09.01 ~ 2022.07.31
  - 4701217개의 데이터

- test.csv
  - 2022.08.01 ~ 2022.08.31
  - 291241개의 데이터

5. 외부 데이터

- 휴일 여부 데이터 사용
  - 주식 휴장일 데이터 응용
  - exchange_calendars library 사용

Introduction

6. 데이터 설명

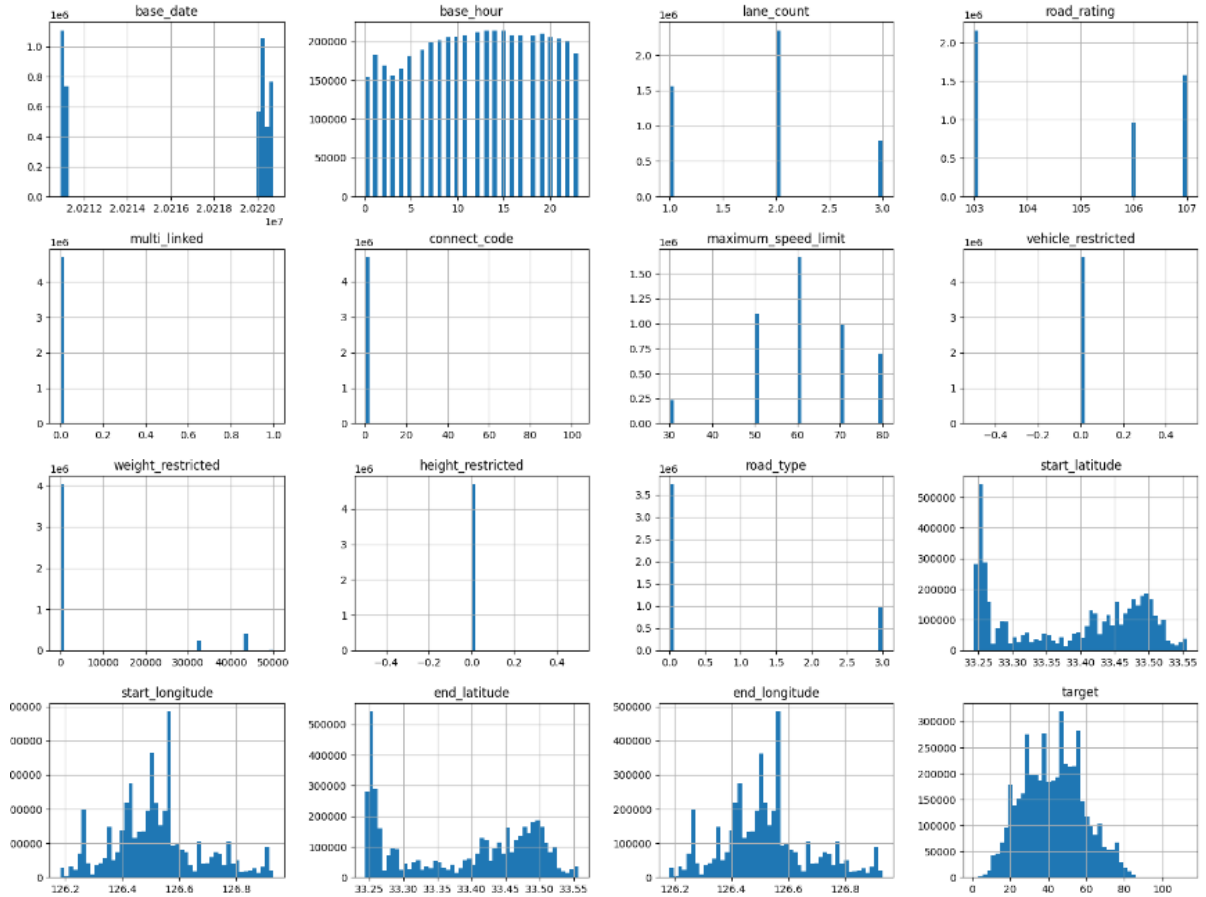
- id : 샘플 별 아이디(ex: TRAIN_0000001)
- base_date : 날짜(ex: 20220623)
- day_of_week : 요일
- base_hour : 시간대(00시 ~ 23시)
- lane_count : 차로수(차선 개수)
- road_rating : 도로등급(국토교통부 교통체계 링크 참조. 101부터 107까지 총 7개 분류)
- multi_linked : 중용구간(2개 이상의 노선이 도로의 일정 구간을 공동으로 사용) 여부
- connect_code : 연결로 코드(국토교통부 교통체계 링크 참조. 000 ~ 108 총 9개 분류)
- maximum_speed_limit : 최고속도제한(단위: km/h)
- weight_restricted : 통과제한하중
- height_restricted : 통과제한높이
- road_type : 도로유형(국토교통부 교통체계 링크 참조. 000 ~ 004)
- start_latitude, start_longitude : 시작지점 위도, 경도
- end_latitude, end_longitude : 도착지점 위도, 경도
- start_turn_restricted, end_turn_restricted : 시작지점, 도착지점 회전제한 유무(ex: 있음, 없음)
- road_name : 도로명
- start_node_name, end_node_name : 시작지점명, 도착지점명
- vehicle_restricted : 통과제한차량
- target : 도로 위 차량의 평균속도(단위 : km/h)

CONTENTS

EDA

EDA

1. **데이터 분포 분석**
   - 위・경도를 제외한 feature 중 categorical 다수 존재
   - base_hour : 대체로 균등한 분포를 보임
   - 위・경도 : numerical 데이터
   - 1개의 값을 가지는 feature 존재
     : multi_linked, connect_code,
     vehicle_restricted, height_retricted

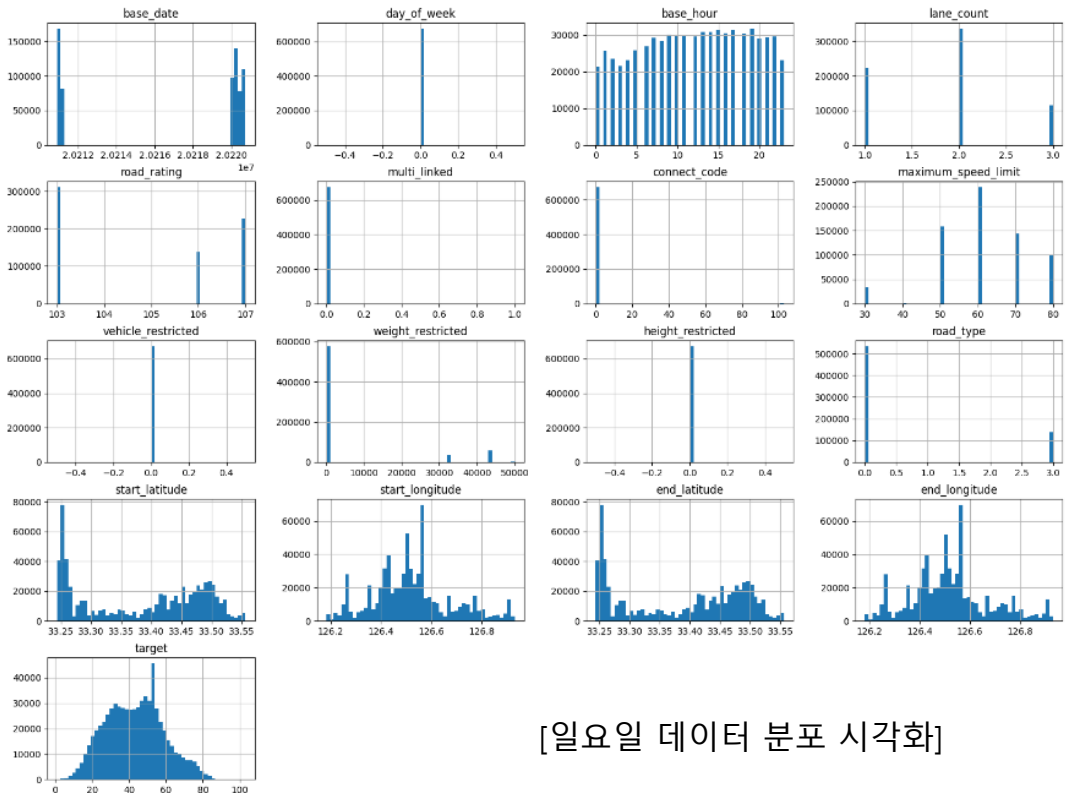


[데이터 분포 시각화]

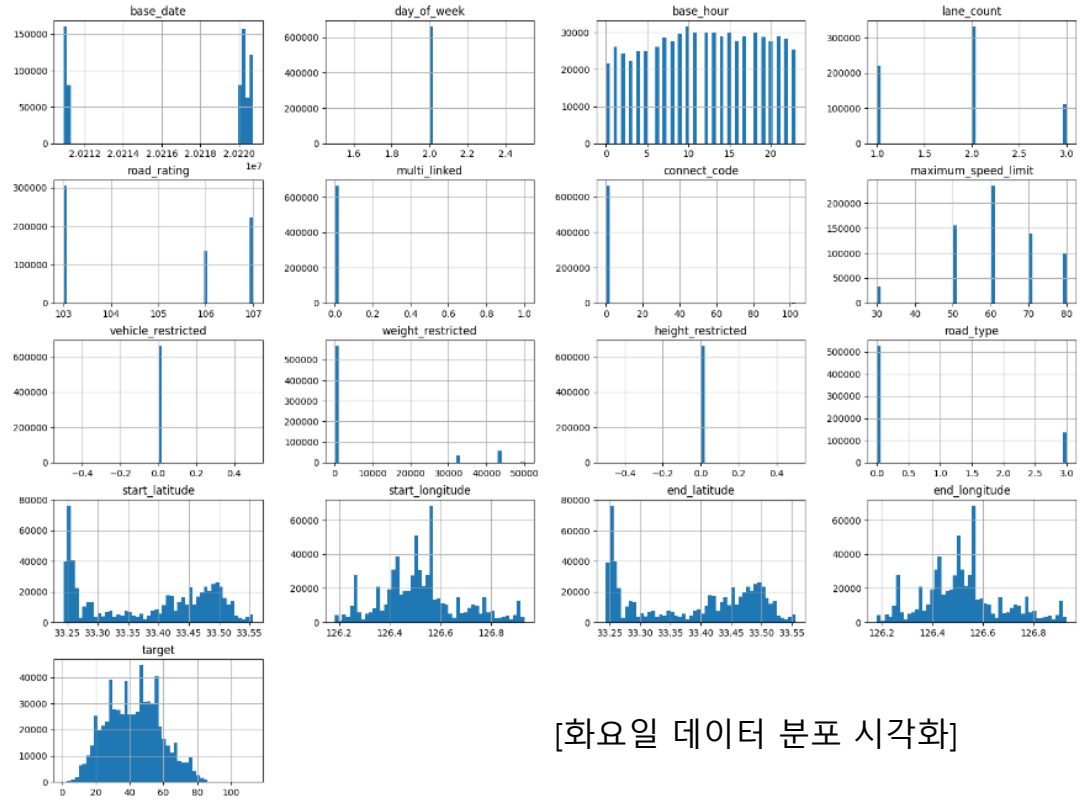
EDA

2. 요일별 데이터 분석

- 각 요일을 수로 mapping
- 요일별 분포의 차이 작음



[일요일 데이터 분포 시각화]

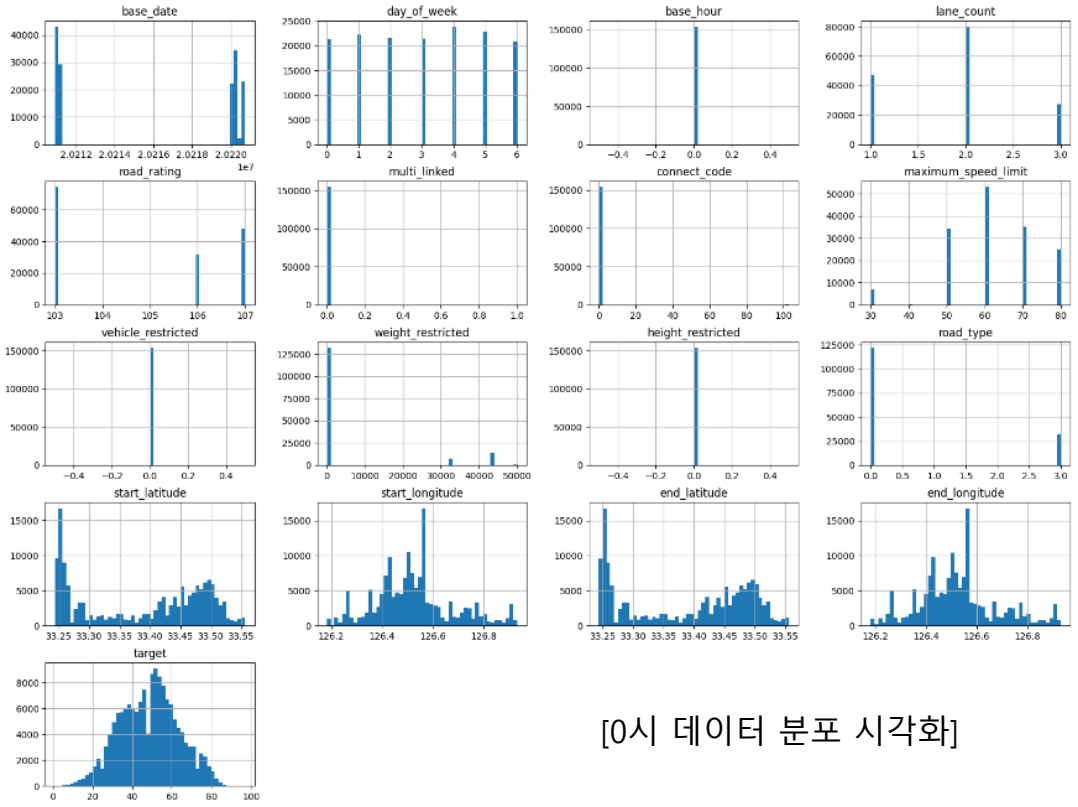


[화요일 데이터 분포 시각화]

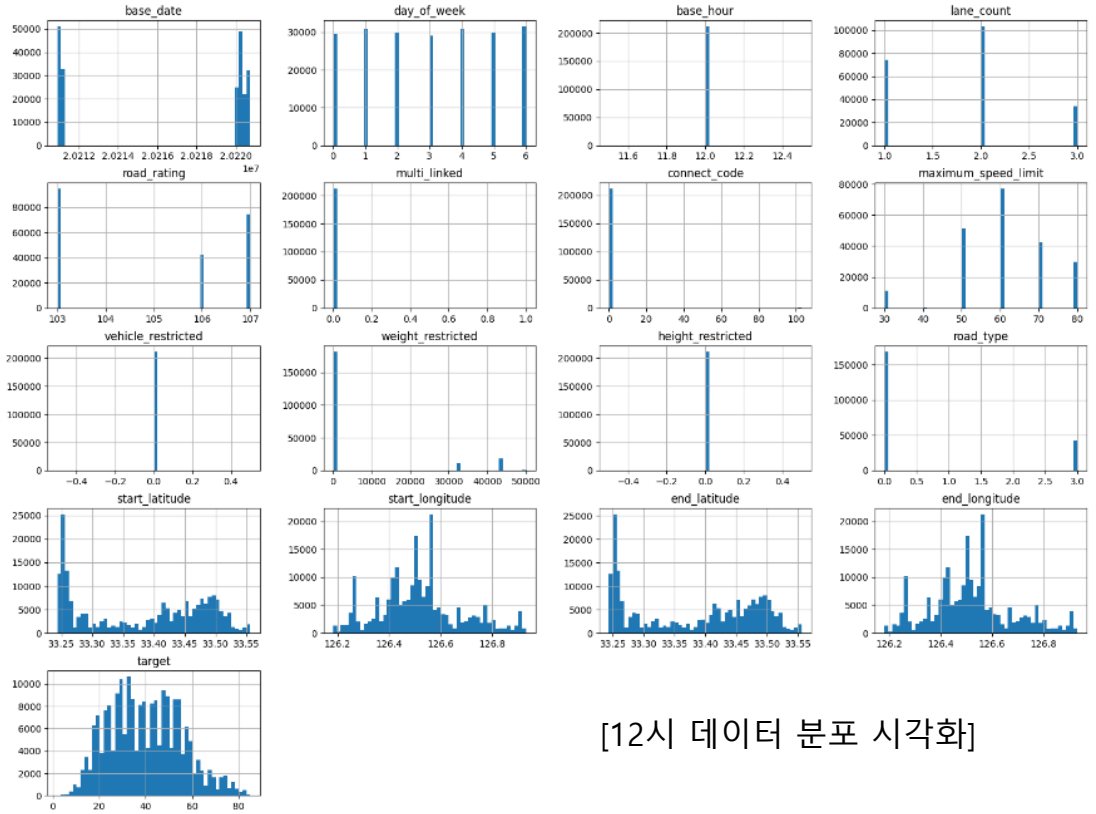
EDA

3. 시간별 데이터 분석

- 시간대별 분포 차이 존재(y축 변화)



[0시 데이터 분포 시각화]

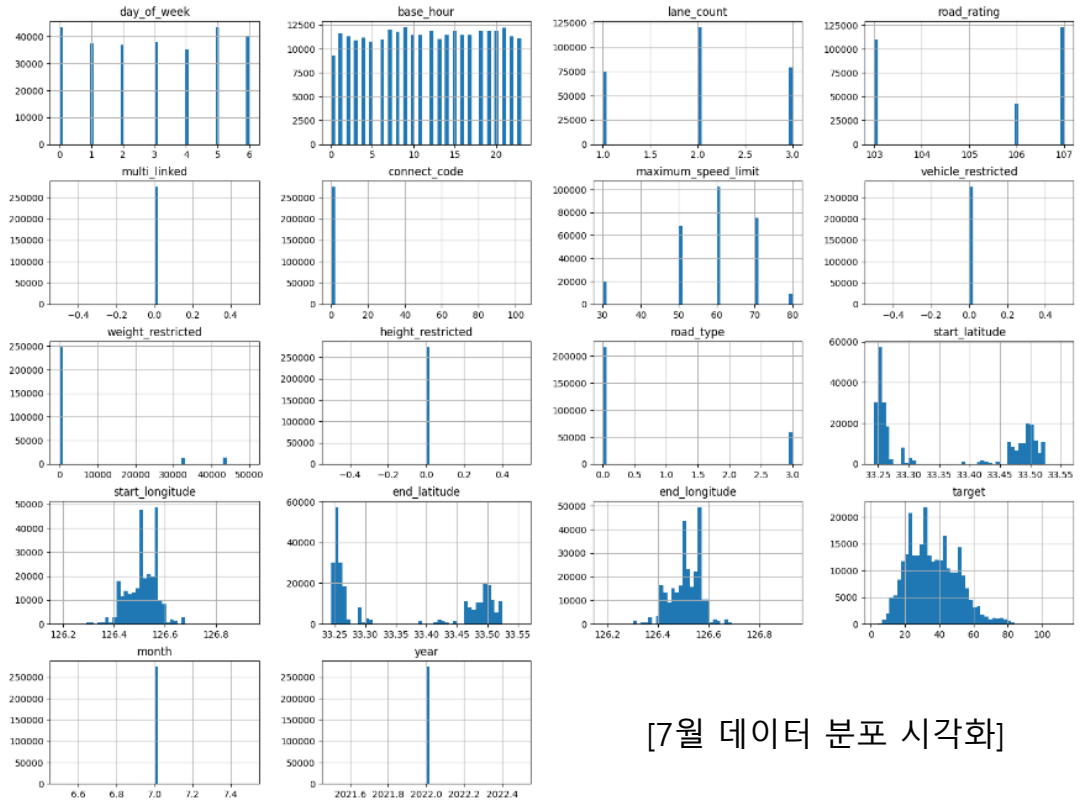


[12시 데이터 분포 시각화]

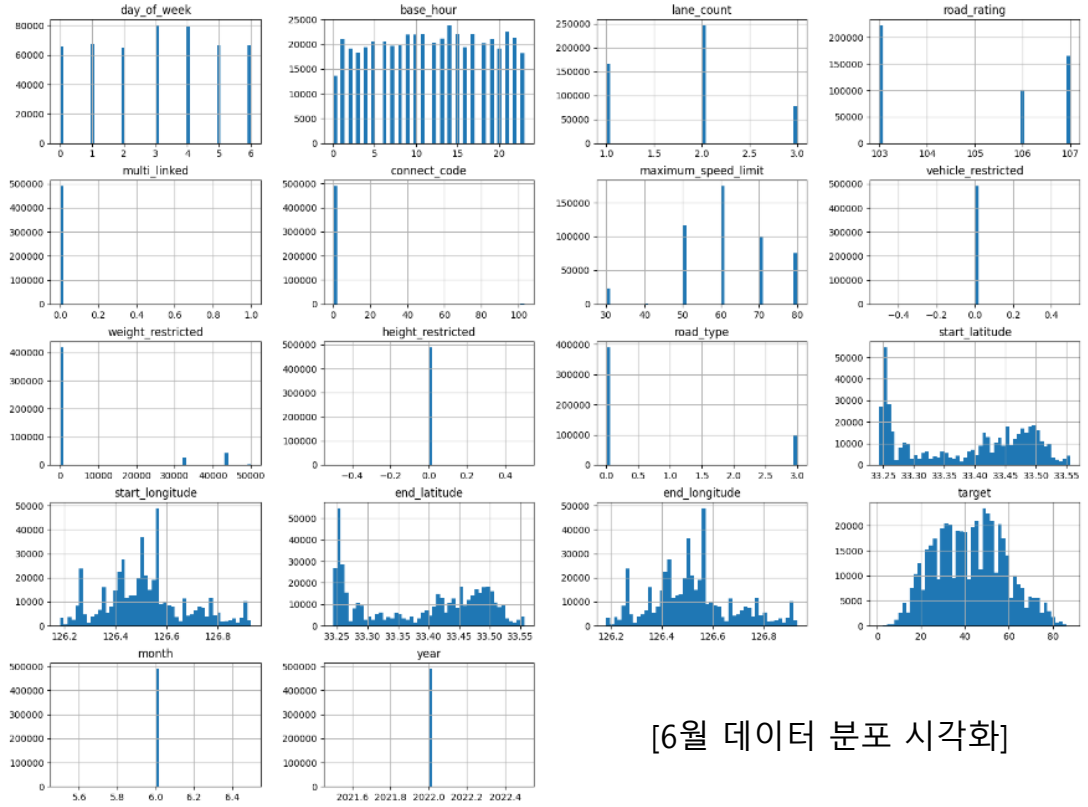
EDA

4. 월별 데이터 분석

- 4월, 8월 데이터 없음
- 7월, 11월 데이터 상대적으로 적음



[7월 데이터 분포 시각화]

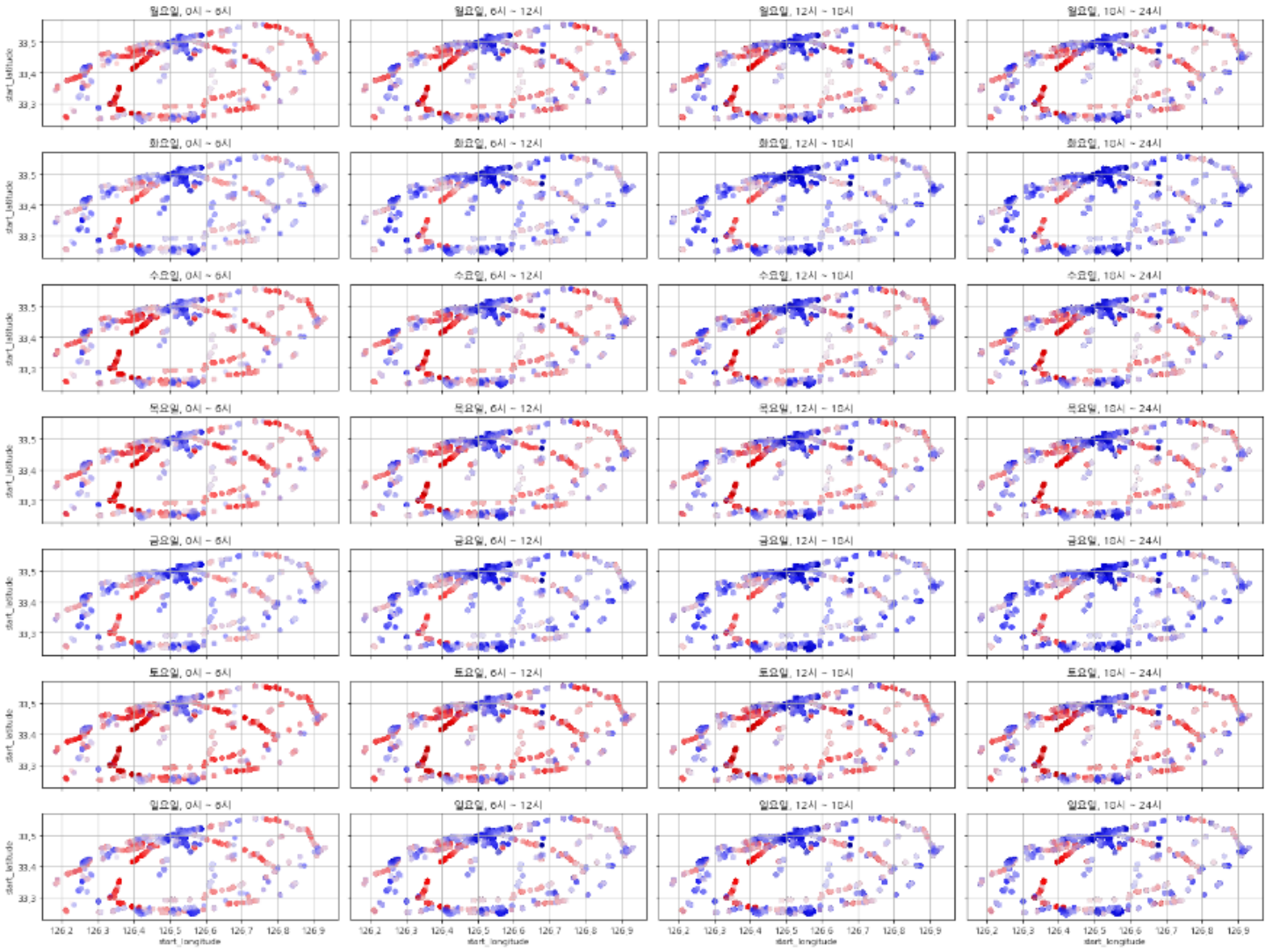


[6월 데이터 분포 시각화]

EDA

5. 요일과 시간에 따른 도로 사용 빈도 분석

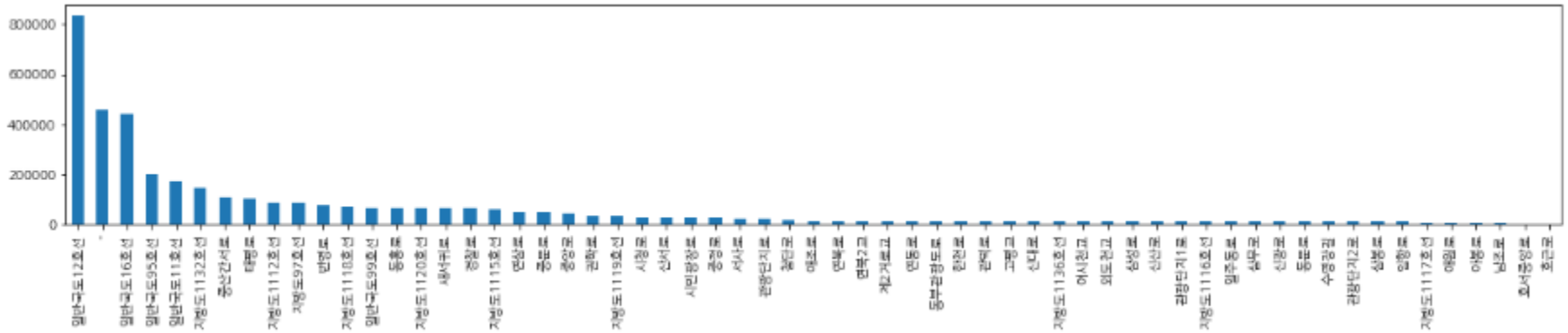
- 토요일 빈도 높음
- 0~6시 빈도 높음



[요일과 시간에 따른 도로 사용 빈도 시각화]

6. 도로에 따른 사용 빈도 분석

- 일반국도 12호선 다수 사용
- 대체로 일반국도 사용 빈도 많음

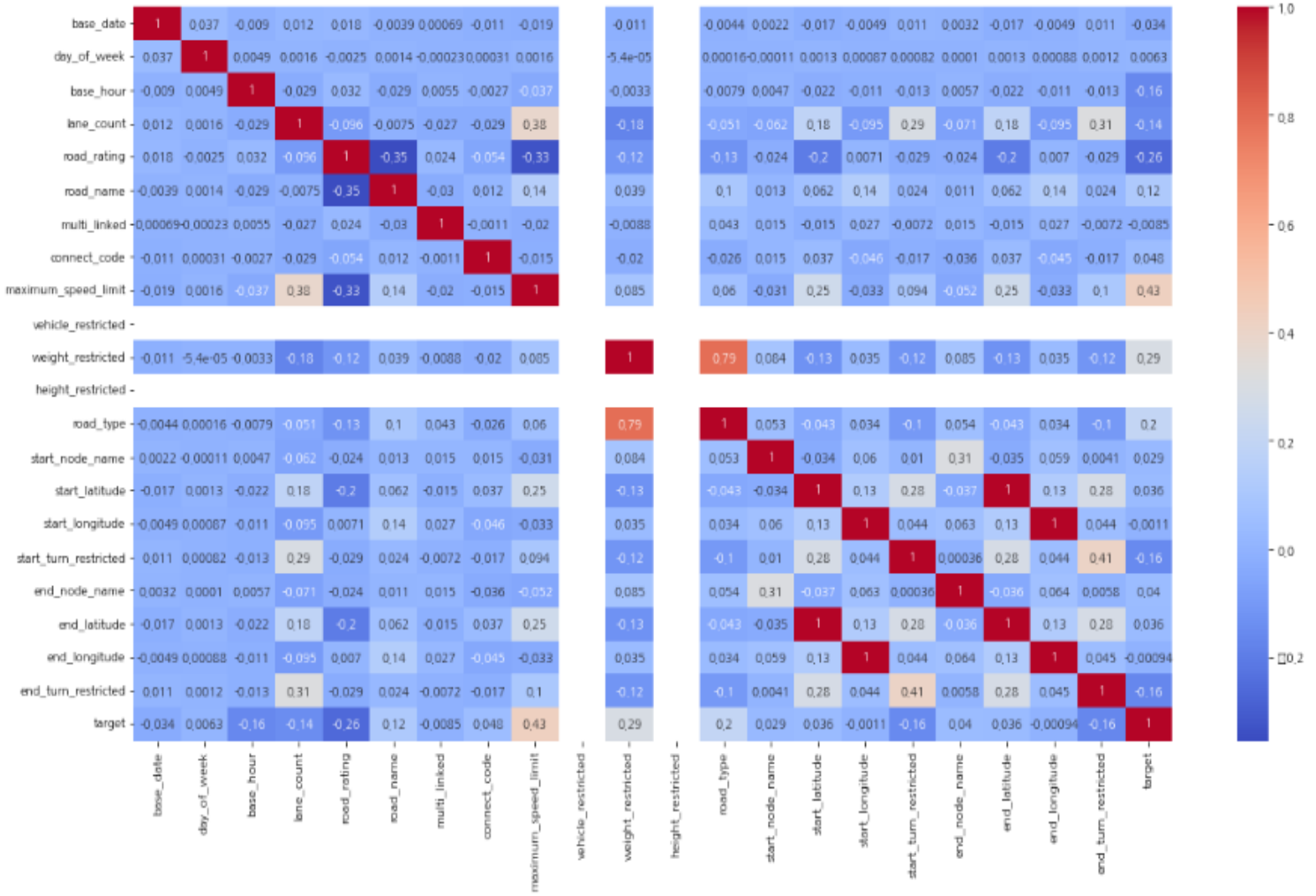


[도로에 따른 사용 빈도 시각화]

EDA

7. 상관관계분석

- Pearson Correlation Coefficient(PCC)
- 대체로 feature간 상관관계 낮음
- road_type - weight_restricted 높은 상관관계



[feature 상관관계 분석]

CONTENTS

Feature Engineering

Feature Engineering

1. 위 · 경도 차이 추가

- 끝 지점과 시작 지점의 위 · 경도 차이 사용
  - 위도 차이 = (end_latitude − start_latitude)
  - 경도 차이 = (end_longitude − start_longitude)

2. 휴일 여부 추가

- 해당 일자가 휴일인지 여부 사용
  - 휴일 : 1, 휴일이 아닐 경우 : 0

- 주식 휴장일 데이터 응용
  - exchange_calendars 라이브러리 사용

3. 연도, 월 변경

- 날짜가 속하는 연도와 월 데이터 사용

```python
def diff_long_lat_add(data):
    data['diff_longtitude'] = data['end_longitude'] - data['start_longitude']
    data['diff_latitude'] = data['end_latitude'] - data['start_latitude']

    return data

def holiday_add(data):
    krx = ecals.get_calendar("XKRX")
    krx_holiday = pd.DataFrame(krx.schedule.loc["2021-01-01":"2022-12-31"])

    open_date = pd.to_datetime(krx_holiday['open'])
    open_date = open_date.dt.strftime('%Y%m%d')
    open_date = open_date.astype(int)
    open_date = list(open_date)

    data['holiday'] = data['base_date'].isin(open_date)
    holiday_exist = {True: 0, False: 1}
    data['holiday'] = data['holiday'].map(holiday_exist)

    return data

def month_year_add(data):
    data['year'] = data['base_date'].apply(lambda e: str(e)[0:4])
    data['year'] = data['year'].astype(int)
    data['month'] = data['base_date'].apply(lambda e: str(e)[4:6])
    data['month'] = data['month'].astype(int)

    return data
```

Feature Engineering

4. **time quarter 추가**
   - 각 시간을 4개 구간으로 나누어 사용
   - 시간에 따른 도로 사용 빈도 기반

5. **최고 속도 제한 단위 변경 후 추가**
   - 기존 속도 단위(km/h)를 변경(m/s)

6. **위 · 경도 거리 및 방위각 추가**
   - WSG84 좌표 기반 거리 및 방위각
   - 전방 및 후방 방위각 사용
   - pyproj 라이브러리 사용

```python
def time_quarter(data):
    quarter = 3

    if 6 > data >= 0:
        quarter = 0

    elif 12 > data >= 6:
        quarter = 1

    elif 18 > data >= 12:
        quarter = 2

    return quarter

def time_quarter_add(data):
    data['time_quarter'] = data['base_hour'].apply(lambda e: time_quarter(e))

    return data

def get_distance(data):
    g = Geod(ellps='WGS84')
    # 2D distance in meters with longitude, latitude of the points
    fwd_azimuth, back_azimuth, distance_2d = g.inv(data['start_longitude'], data['start_latitude'], data['end_longitude'],
                                                   data['end_latitude'])

    return fwd_azimuth,back_azimuth, distance_2d

def distance_azimuth_add(data):
    vector = list(data.apply(get_distance,axis=1).values)
    data[['fwd_azimuth','back_azimuth','distance']] = pd.DataFrame(vector, columns=['fwd_azimuth', 'back_azimuth', 'distance'])

    return data

def meter_sec_add(data):
    data['maximum_meter_second'] = data['maximum_speed_limit'] / 3.6

    return data
```

Feature Engineering

7. scaling

- min max scaling 적용
  - standard scaling을 적용했을 때보다 향상된 성능을 나타냄

8. encoding

- Label encoder 사용
  - categorical 데이터 수치화

9. One Hot encoding 및 PCA

- 위・경도 One Hot encoding 후 2차원 PCA 적용
  - 위・경도는 지구에서 위치를 나타냄으로 PCA 적용 시 효율적이라는 가설

Feature Engineering

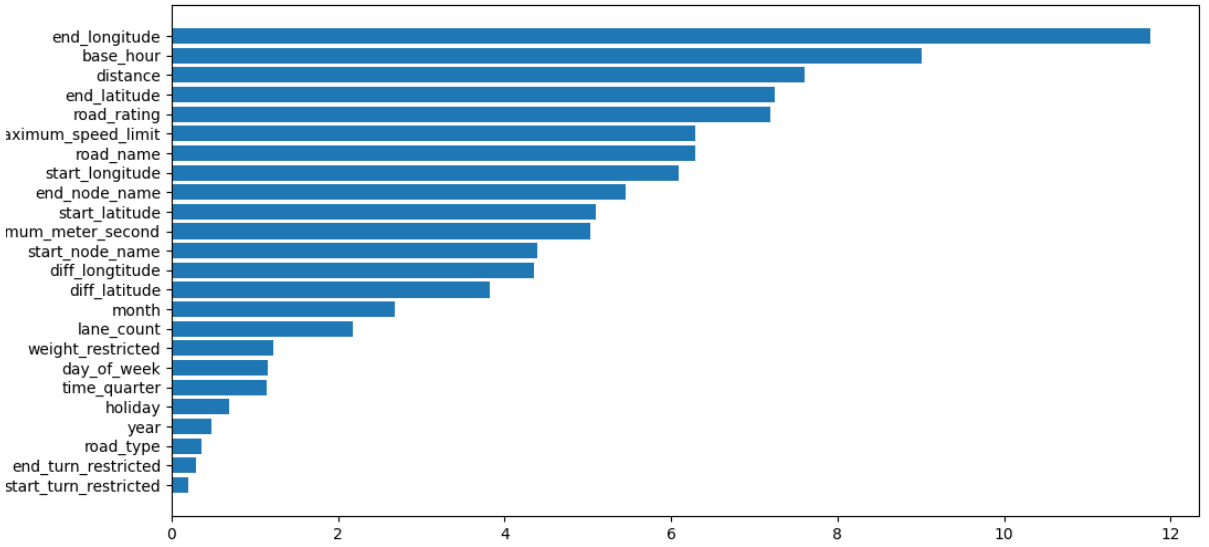
10. data cleansing

- 값이 1개인 feature 제거
  - multi_linked, connect_code, vehicle_restricted, height_retricted

11. feature importance

- Boosting 기반 importance
  - categorical data의 비율이 높음
  - catboost를 사용한 feature importance 추출

- Permutation importance
  - 다수의 feature를 여러 번 섞어서 importance 추출



[boosting 기반 feature importance 시각화]

```
0.3258 ± 0.0011  end_longitude
0.2604 ± 0.0002  maximum_speed_limit
0.2180 ± 0.0009  road_name
0.2079 ± 0.0004  road_rating
0.2053 ± 0.0005  start_latitude
0.1983 ± 0.0008  end_latitude
0.1799 ± 0.0009  base_hour
0.1522 ± 0.0006  road_visit_count
0.1059 ± 0.0002  end_node_name
0.1006 ± 0.0002  start_longitude
0.0678 ± 0.0002  start_node_name
0.0275 ± 0.0002  lane_count
0.0257 ± 0.0002  month
0.0138 ± 0.0001  road_type
0.0094 ± 0.0001  day_of_week
0.0094 ± 0.0000  holiday
0.0083 ± 0.0001  visitor
0.0031 ± 0.0000  start_turn_restricted
0.0025 ± 0.0000  weight_restricted
0.0018 ± 0.0000  end_turn_restricted
       0 ± 0.0000  vehicle_restricted
       0 ± 0.0000  height_restricted
```

[permutation importance 기반 시각화]

CONTENTS

Modeling

Modeling

1. Catboost 사용

- 데이터의 수가 많으므로 deep learning 모델은 비효율적
  - 최적화 기간이 boosting 모델에 비해 상대적으로 많이 필요

- categorical 데이터의 비율이 높음
  - boosting 모델 중 Catboost 사용 적합

- 빠른 학습 시간

2. hyperparameter tuning

- Optuna를 활용한 tuning

- 데이터의 20%를 validation set으로 활용

- MAE 사용
  - 대회에서 제시하는 평가지표

```python
def main():
    tuning = False

    if tuning:
        sampler = TPESampler(seed=RANDOM_STATE)
        optuna_cbrm = optuna.create_study(direction='minimize', sampler=sampler)
        optuna_cbrm.optimize(cat_boost_tuning, n_trials=50)

        cbrm_trial = optuna_cbrm.best_trial
        cbrm_trial_params = cbrm_trial.params
        print('Best Trial: score {},\nparams {}'.format(cbrm_trial.value, cbrm_trial_params))

    else:
        model, label_encoders, scalers, one_hot_encoder_dict, one_hot_pca_dict = cat_boost_fit()
        test_data = test_preprocessing(label_encoders, scalers, one_hot_encoder_dict, one_hot_pca_dict)
        predict_to_csv(model, test_data)
```

```python
def cat_boost_tuning(trial):
    data, encoders, scalers, one_hot_encoder_dict, pca_dict = train_preprocessing()

    cbrm_param = {
        'iterations': 200000,
        'early_stopping_rounds': 100,
        'eval_metric': 'MAE',
        'learning_rate': 0.03810245233316924,
        'reg_lambda': trial.suggest_float('reg_lambda', 1e-5, 100),
        'depth': 11,
        'min_data_in_leaf': trial.suggest_int('min_data_in_leaf', 1, 30),
        'leaf_estimation_iterations': trial.suggest_int('leaf_estimation_iterations', 1, 15),
        'bagging_temperature': trial.suggest_loguniform('bagging_temperature', 0.01, 100.00),
        'devices':'0:3',
        'task_type':'GPU',
        'random_state' : RANDOM_STATE,
        'random_strength' : trial.suggest_int('random_strength', 0, 100),
    }

    evals = [(data['X_val'], data["y_val"])]
    model = CatBoostRegressor(**cbrm_param)
    model.fit(data["X_train"], data["y_train"], eval_set=evals, verbose=False)

    return model.evals_result_['validation']['MAE'][-1]
```

3. 사용된 hyperparameter

iterations	200000
early_stopping_rounds	100
learning_rate	0.03810245233316924
min_data_in_leaf	19
max_depth	11
eval_metric	'MAE'
task_type	"GPU"
devices	'1:2'
reg_lambda	9.485166914376475
leaf_estimation_iterations	13
bagging_temperature	0.029396564330935907
random_strength	85

CONTENTS

1. **평가지표**
    - MAE

2. **예측 결과**
    - loss : 3.1231
    - 등수 : 30 / 712(상위 5%)

THANK YOU

감사합니다