다변량통계분석

제주도 **도로 교통량**과 **유의미한 변수**

삼각관계아님

20182813 이경욱
20192792 이예진
2020 이현지

목차

주제

제주도 도로 교통량과 유의미한 변수

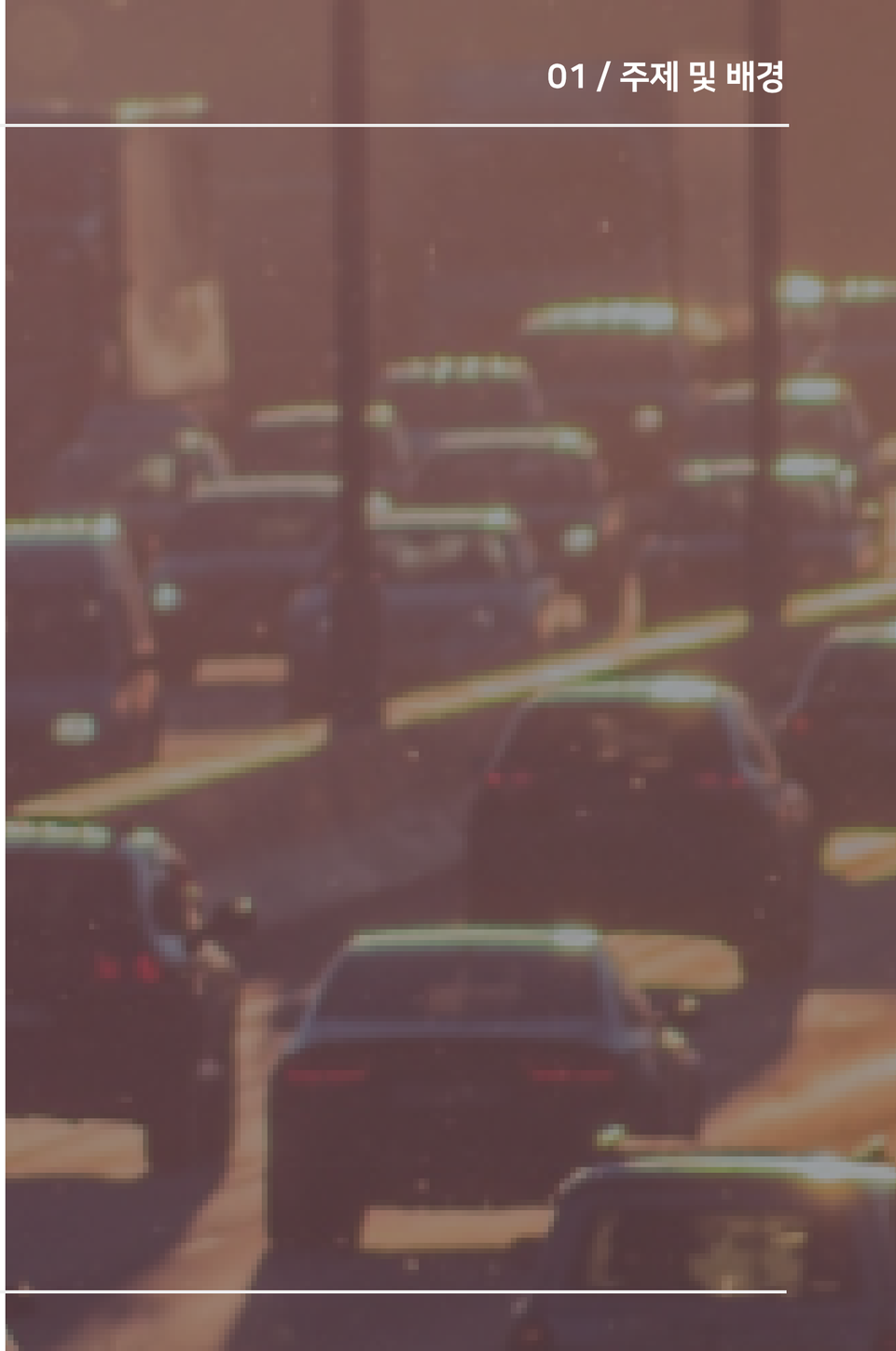
배경

인구
약 68만명
2022년 기준

증가율
연평균 1.3%

추정되는 상주인구
90만명 이상

제주도민 증가와 외국인의 증가
현재 제주도의 **교통체증**이 심각한 문제로 떠오름



문제 정의

제주도 도로 교통량에 유의미한 변수를 가려내고
이들을 통해 도로 교통량을 예측하는 모델 개발

문제 정의

데이터 소개

01 데이터 소개

해당 프로젝트는 Dacon에서 개최되고 있는
제주도 도로 교통량 예측 AI 경진대회를 기반으로 함.

제주도의 여러 교통 정보로부터
도로 교통량을 회귀 예측하는 task

02 변수 정의

독립 변수 : 요일, 시간, 해당 도로번호, 제한 속도, 제한 차종,
제한 중량, 도로 시작 지점, 도로 시작 위/경도,
도로 종료 지점, 도로 종료 위/경도

종속 변수 : 도로의 차량 평균 속도(km/h)

분석 과정

데이터 탐색

2022년 8월 이전 데이터만 존재
→ 단, 날짜가 모두 연속적이지 않음

약 470만여건의 데이터 (4,701,217개)

Target변수: 도로의 차량 평균 속도(km)
→ 교통량이 많다면 차량의 평균 속다가 떨어질 것으로 예상

변수
- id: 샘플 별 고유 id
- 날짜, 시간, 교통 및 도로구간 등 22개 변수

결측치는 없는 것으로 확인

한 가지 값만 존재하는 column 존재 -> 컬럼 제거
- 'vehicle_restricted', 'height_restricted'

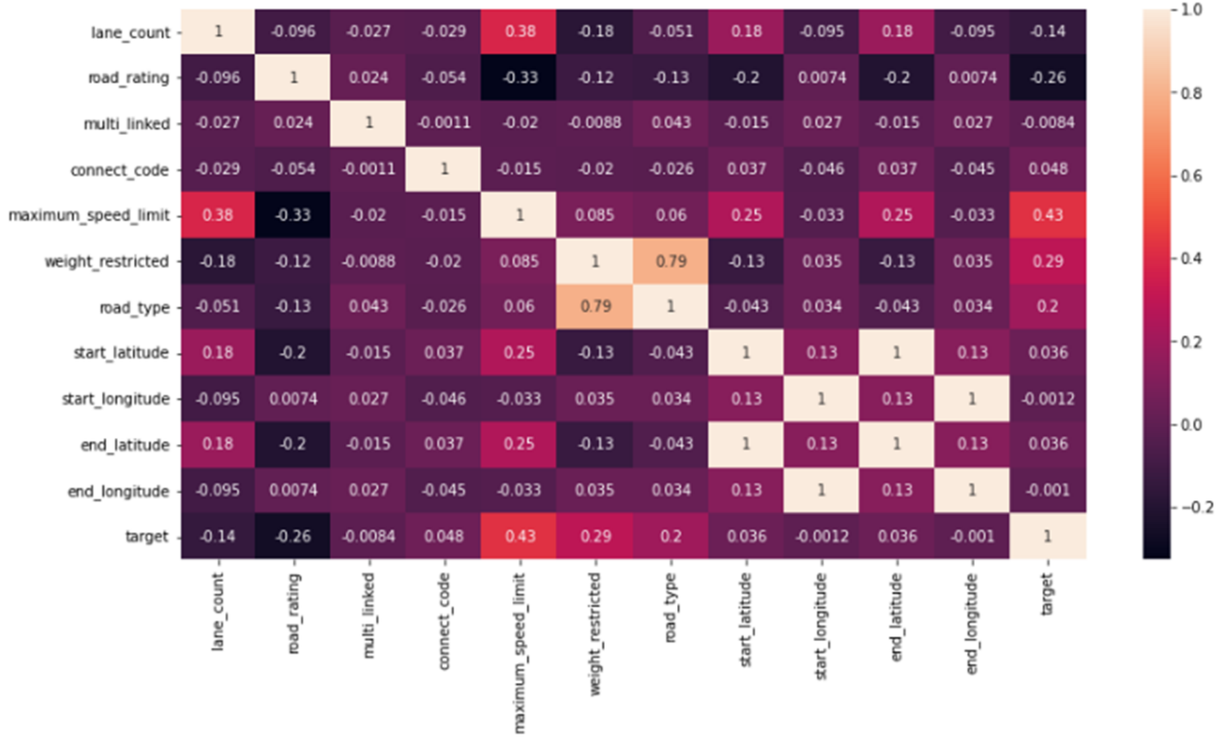
날짜 범위 확인
- 2021.09.01~2022.07.31의 데이터가 존재
- 예측을 해야 하는 항목은 2022.08의 데이터

분석 과정

상관관계

```
# "base_date"와 "base_hour"는 상관관계를 분석하지 않을 것이기에 임시로 object로 변형하겠습니다
train[["base_date", "base_hour"]] = train[["base_date", "base_hour"]].astype(object)
plt.figure(figsize = (14,7))
sns.heatmap(train.corr(), annot = True)

# object로 변형하였던 "base_hour", "base_date"를 다시 int형으로 변형하겠습니다
train[["base_hour", "base_date"]] = train[["base_hour","base_date"]].astype("int32")
```



예측해야하는 target값과의 상관관계가 높은 것중에 가장 눈에 띄는 컬럼은 **maximum_speed_limit**
→ 최고속도 제한이 높을 수록 양의 상관관계가 있다는 것

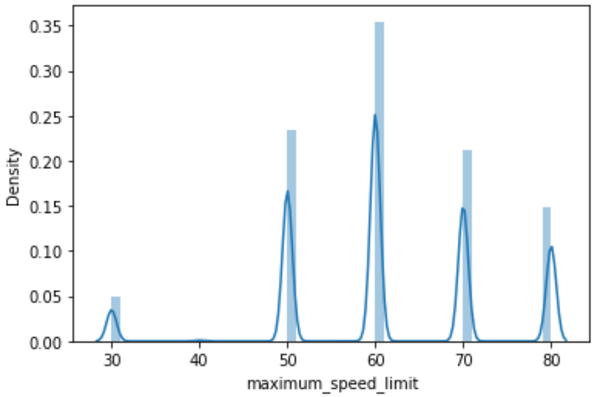
lane_count(차로 수)는 많을수록 속도가 더 높을 것 같지만
→ target과의 음의 상관관계

weight_restricted(통과 제한 하중)
→ target과의 양의 상관관계

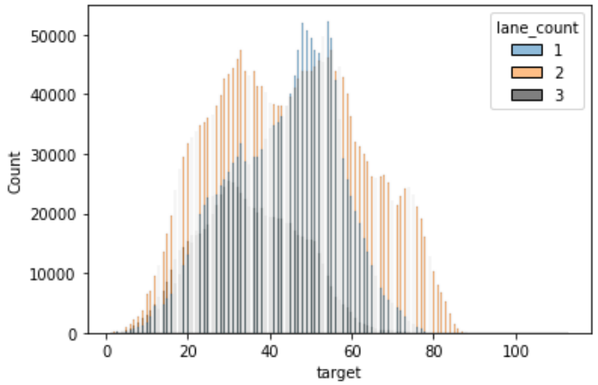
분석 과정

시각화

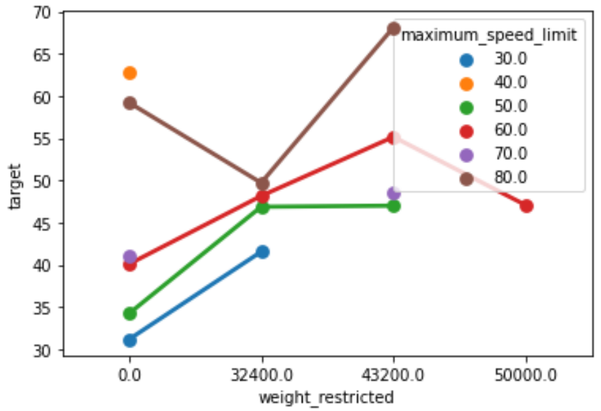
최고 제한 속도(maximum_speed_limit)



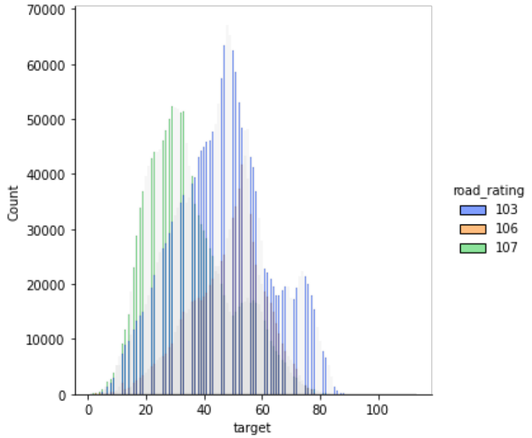
차로 수(lane_count)



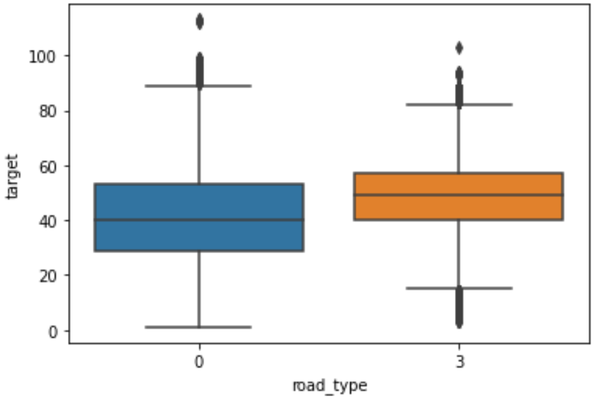
통과 제한 하중(weight_restricted)



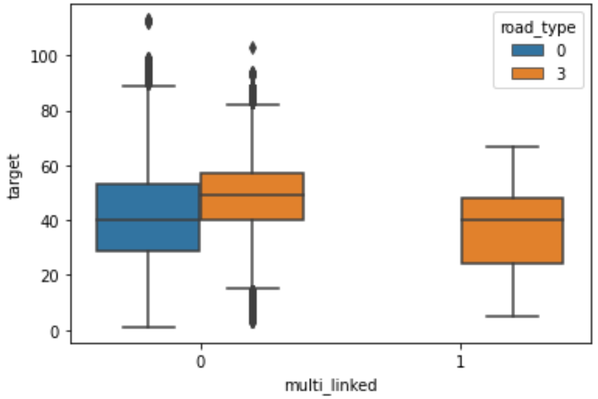
도로 등급(road_rating)



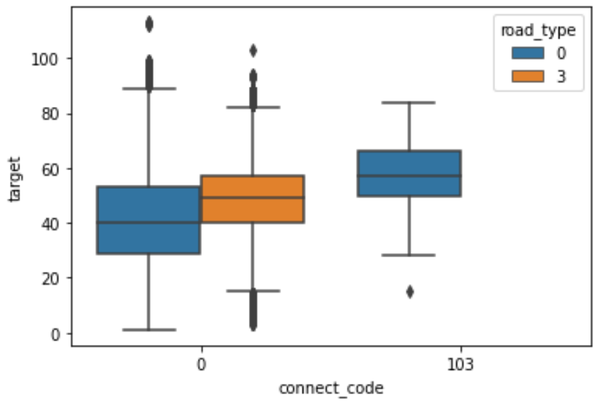
도로 유형(road_type)



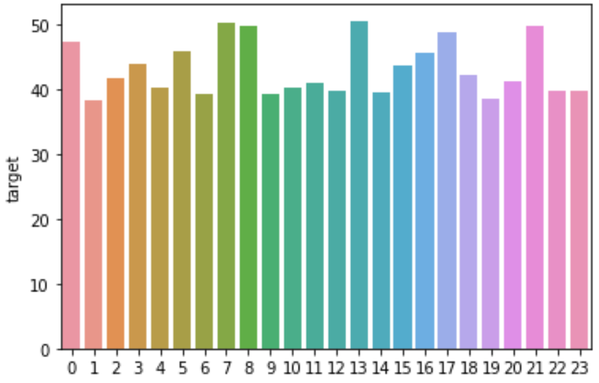
중용구간 여부(multi_linked)



연결로 코드(connect_code)



시간(base_hour)



분석 과정

데이터 분석

01

```
# road_name = 지방도 추출

road = data.road_name.unique()
road = [i for i in road if "지방" in i]

data = data.query("road_name in @road")

# 2022년 4월 1일 이전, 이후로 분할
test = data[data.base_date >= 20220401]
train = data[data.base_date < 20220401]
```

```
rt = test.shape[0]/train.shape[0]
print(f"train : test = {1-rt} : {rt}")
```

```
train : test = 0.661040270128357 : 0.338959729871643
```

데이터셋 분할

road_name 중 지방도만 추출
2022.04.01 이전과 이후로 분할

02

```
train["month"] = train['base_date'].astype("str").apply(
                                lambda x : x[4:6]).apply(
                                lambda x : int(x) if x[0]!=0 else int(x[-1]))

train["day"] = train['base_date'].astype("str").apply(
                                lambda x : x[-2:]).apply(
                                lambda x : int(x) if x[0]!=0 else int(x[-1]))

test["month"] = test['base_date'].astype("str").apply(
                                lambda x : x[4:6]).apply(
                                lambda x : int(x) if x[0]!=0 else int(x[-1]))

test["day"] = test['base_date'].astype("str").apply(
                                lambda x : x[-2:]).apply(
                                lambda x : int(x) if x[0]!=0 else int(x[-1]))
```

```
train["start_turn_restricted"] = train["start_turn_restricted"].map({"없음":0, "있음":1})
test["start_turn_restricted"] = test["start_turn_restricted"].map({"없음":0, "있음":1})

train["end_turn_restricted"] = train["end_turn_restricted"].map({"없음":0, "있음":1})
test["end_turn_restricted"] = test["end_turn_restricted"].map({"없음":0, "있음":1})
```

```
day_mean = train.groupby("day_of_week")["target"].mean().reset_index().values
day_mean = {i : j for i, j in day_mean}

train["day_of_week"] = train["day_of_week"].map(day_mean)
test["day_of_week"] = test["day_of_week"].map(day_mean)
```

```
road_info = ["lane_count", "maximum_speed_limit", "weight_restricted",
             "road_type", "start_turn_restricted","end_turn_restricted"]
day = ["day_of_week", "base_hour", "month", "day"]
col = road_info + day
```

```
y_tr = train["target"]
y_te = test["target"]

train = train[col]
test = test[col]
```

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()

scaler.fit(train)
train.loc[:, :] = scaler.transform(train)
test.loc[:, :] = scaler.transform(test)
```

전처리

데이터 전처리 진행

분석 과정_데이터 분석

도로관련 변수 요인분석(Factor Analysis)

```
# 탐색적요인분석
fa = FactorAnalyzer(n_factors=2, rotation='varimax').fit(train[road_info])
```

요인적재량 :

	0	1
lane_count	-0.095888	1.006479
maximum_speed_limit	-0.084552	0.510503
weight_restricted	0.795582	-0.219084
road_type	0.979173	0.230136
start_turn_restricted	0.029149	0.367815
end_turn_restricted	0.033676	0.258533

공통성 :

	0
lane_count	1.022194
maximum_speed_limit	0.267762
weight_restricted	0.680949
road_type	1.011742
start_turn_restricted	0.136138
end_turn_restricted	0.067973

요인점수 :

```
[[-0.52681027 -0.72142485]
 [-0.52681027 -0.72142485]
 [-0.99757481  1.66614112]
 ...
 [ 1.35971042  2.20839385]
 [-0.73345151  0.43622822]
 [ 1.35971042  2.20839385]]
```

Factor0

- 통과제한하중, 도로유형에 대한 factor loading 값이 큼
- "속도에 대한 간접적 요인"라고 명명

Factor1

- 차로수, 속도제한, 시작/도착지점 회전제한 유무 대한 factor loading 값이 큼
- "속도에 대한 간접적 요인"라고 명명

분석 과정_데이터 분석

요인점수 적용, 미적용 회귀분석 결과 비교

요인점수 적용

	MAE_factor	time_factor
LinearRegression	9.158527	0.045107
Lasso	9.361843	0.029187
Ridge	9.158527	0.011617
ElasticNet	9.392786	0.027852

요인점수 미적용

	MAE_nofactor	time_nofactor
LinearRegression	7.535561	0.099526
Lasso	9.549749	0.041844
Ridge	7.535572	0.030627
ElasticNet	9.594355	0.057791

속도 비교

final_benchmark.**time_nofactor**.mean() / final_benchmark.**time_factor**.mean() = **2.0198908111619915**

MAE 비교

final_benchmark.**MAE_factor**.mean() / final_benchmark.**MAE_nofactor**.mean() = **1.0834846010429615**

분석 결과 도출

MAE

요인변수를 활용해
feature를 줄인 데이터셋을
활용한 경우가 MAE가 평균적으로
1.08배 증가

하지만
Ridge회귀나 Elastic회귀
사용시 오히려 감소

모델 구현 속도

요인변수를 활용한 경우
속도가 1.7배 향상

Thank You