

# 제주도 도로 교통량 예측 AI 경진대회

2022 인공지능 텀프로젝트 - 나만의 미니 챌린지 최종 발표

21011928 전주혁

# 목차

contents

- I . INTRO
- II. 데이터 가공
- III. 리더보드 제작 / 베이스라인 구축
- IV. outro

# I INTRO

# I INTRO(프로젝트 설명)

## [배경] - 사회적 문제

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.

## [주제]

제주도 도로 교통량 예측 AI 알고리즘 개발 -> 사회적 문제 해결

## [설명]

제주도의 교통 정보로부터 도로 교통량 회귀 예측

## II 데이터 가공

# 데이터 가공

## 첫 번째 데이터셋

base_date	day_of_w	base_hour	road_in_us	lane_count	road_ratio	road_name	multi_link	connect_c	maximum_vehicle	weight_res	height_res	road_type	start_node	start_latitude	start_longitude	start_turn	end_node	end_latitude	end_longitude	end_turn	target					
TRAIN_000	20220623	목	17	0	1	106	지방도111	0	0	60	0	32400	0	3	제3교래교	33.42775	126.6626	없음	제3교래교	33.42775	126.6623	없음	52			
TRAIN_000	20220728	목	21	0	2	103	일반국도1	0	0	60	0	0	0	0	광양사거리	33.50073	126.5291	있음	KAL사거리	33.50481	126.5262	없음	30			
TRAIN_000	20211010	일	7	0	2	103	일반국도1	0	0	80	0	0	0	0	창고천교	33.27915	126.3686	없음	상창육교	33.28007	126.3621	없음	61			
TRAIN_000	20220311	금	13	0	2	107	태평로	0	0	50	0	0	0	0	0	남양리조트	33.24608	126.5672	없음	서현주택	33.24557	126.5662	없음	20		
TRAIN_000	20211005	화	8	0	2	103	일반국도1	0	0	80	0	0	0	0	0	0	애월샛시	33.46221	126.3266	없음	애월입구	33.46268	126.3302	없음	38	
TRAIN_000	20210913	월	7	0	2	107	경찰로	0	0	60	0	0	0	0	0	시청입구2	33.24995	126.5057	없음	서호2차현	33.25218	126.5061	없음	28		
TRAIN_000	20220106	목	0	0	2	107	-	0	0	60	0	0	0	0	가동	33.41841	126.268	없음	나동	33.41418	126.2694	없음	39			
TRAIN_000	20211213	월	16	0	2	107	외도천교	0	0	60	0	0	0	3	외도천교	33.48239	126.4416	없음	외도천교	33.48233	126.4423	없음	28			
TRAIN_000	20211004	월	15	0	2	107	경찰로	0	0	60	0	0	0	0	0	신성교회	33.25307	126.5064	없음	서호2차현	33.25218	126.5061	없음	14		
TRAIN_000	20211208	수	2	0	1	103	일반국도1	0	0	50	0	0	0	0	0	양수장	33.36172	126.767	없음	제2가시교	33.36434	126.7694	없음	52		
TRAIN_000	20220623	목	11	0	1	103	일반국도9	0	0	60	0	0	0	0	0	0	노루생이	33.41942	126.4915	없음	노루생이	33.42267	126.4929	없음	47	
TRAIN_000	20220724	일	2	0	1	107	중정로	0	0	50	0	0	0	0	0	0	선경오피스	33.24851	126.5698	없음	정방수퍼	33.24863	126.5678	없음	40	
TRAIN_000	20211229	수	10	0	3	106	번영로	0	0	70	0	0	0	0	0	0	명도암교차	33.48571	126.6042	있음	버드내교차	33.48005	126.6255	있음	60	
TRAIN_000	20220507	토	7	0	2	103	일반국도1	0	0	60	0	0	0	0	0	0	서흥교	33.26411	126.554	없음	서흥동사거리	33.26368	126.551	없음	28	
TRAIN_000	20220203	목	16	0	1	107	-	0	0	60	0	43200	0	3	송죽교	33.31691	126.6246	없음	송죽교	33.31707	126.6239	없음	58			
TRAIN_000	20220501	일	16	0	1	103	일반국도1	0	0	30	0	0	0	0	0	0	아라조동북	33.478	126.5438	없음	제2아라교	33.47745	126.5427	없음	32	
TRAIN_000	20220701	금	22	0	2	107	연동로	0	0	50	0	0	0	0	0	0	그랜드호텔	33.48588	126.49	없음	불천5교	33.48597	126.4864	없음	35	
TRAIN_000	20211001	금	17	0	2	106	중산간서로	0	0	70	0	0	0	0	0	0	중산간서로	33.45242	126.3826	없음	장전1교차	33.45273	126.3853	있음	50	
TRAIN_000	20220319	토	16	0	2	106	중산간서로	0	0	70	0	0	0	0	0	0	광령3교차	33.46223	126.4236	있음	고성교차로	33.45797	126.4101	없음	56	
TRAIN_000	20220210	목	12	0	2	103	일반국도1	0	0	80	0	0	0	0	0	0	오조한도교	33.46532	126.9086	없음	송내교차로	33.47037	126.9028	없음	46	
TRAIN_000	20220701	금	21	0	1	103	일반국도1	0	0	70	0	0	0	0	0	0	0	중합운동장	33.5001	126.5129	있음	동산교	33.50013	126.512	없음	21
TRAIN_000	20220119	수	0	0	2	103	일반국도1	0	0	80	0	0	0	0	0	0	0	동부장익음	33.27817	126.6677	없음	농업용관천	33.27988	126.686	없음	70
TRAIN_000	20210907	화	23	0	1	106	지방도111	0	0	60	0	0	0	0	0	0	0	양천동	33.51847	126.6457	없음	병덕와교차	33.52766	126.6448	없음	37
TRAIN_000	20220205	토	7	0	2	107	새서귀로	0	0	60	0	0	0	0	0	0	0	한솔고기국	33.25195	126.5109	없음	삼주연립1	33.25105	126.5106	없음	33
TRAIN_000	20210927	월	13	0	1	106	지방도111	0	0	60	0	0	0	0	0	0	0	제5산록교	33.29173	126.5044	없음	제4산록교	33.29012	126.489	없음	54
TRAIN_000	20220107	금	8	0	2	103	일반국도1	0	0	60	0	0	0	0	0	0	0	서흥교	33.26411	126.554	없음	서흥교	33.26412	126.5543	없음	31
TRAIN_000	20210903	금	17	0	3	106	지방도113	0	0	70	0	0	0	0	0	0	0	강정교	33.2506	126.4893	없음	도순교	33.25044	126.4798	없음	50

데이터콘 대회에서 제공하는 train 데이터를 이용

# II 데이터 가공

## 두 번째 데이터셋

일자	시도명	읍면동명	거주인구	근무인구	방문인구	총 유동인구	교통량	평균 속도	평균 소요	평균 기온	일강수량	평균 풍속	데이터기준일자
2018-01-01	서귀포시	남원읍	291408.9	18744.13	219588.9	529741.9	76.2	53	40.571	2.35	0	3.325	2020-12-15
2018-01-02	서귀포시	남원읍	290944.7	27781.68	166779.6	485505.9	72.8	53	40.771	4.2	0	2.8	2020-12-15
2018-01-03	서귀포시	남원읍	287121.5	27767.08	181151.4	496040	73.971	52.371	41.514	3.575	0	3.925	2020-12-15
2018-01-04	서귀포시	남원읍	279036.1	27045.99	188198.6	494280.7	70.286	52.143	41.514	3.2	1.625	2.45	2020-12-15
2018-01-05	서귀포시	남원읍	287206	24824.67	171332.5	483363.2	66.314	52.257	41.486	1.875	2.75	3.875	2020-12-15
2018-01-06	서귀포시	남원읍	287820.4	20225.76	203507.1	511553.2	74.829	53.171	40.743	2.85	0	3.225	2020-12-15
2018-01-07	서귀포시	남원읍	295788.9	17764.31	186689.5	500242.8	62.6	53.171	41.314	5.875	8.625	1.85	2020-12-15
2018-01-10	서귀포시	남원읍	291347.6	23248.08	165175.9	479771.6	87.943	48.314	49.8	-2.25	9.25	2.667	2020-12-15
2018-01-12	서귀포시	남원읍	297122.3	20657.86	139731.4	457511.6	47.343	47.686	47.286	-1.967	5.25	2.9	2020-12-15
2018-01-13	서귀포시	남원읍	288884.1	17467.83	184490.6	490842.5	72.086	51.286	42.8	2.167	0	2.133	2020-12-15
2018-01-14	서귀포시	남원읍	291240.7	17695.9	224387.6	533324.2	67.114	53.114	41.2	5.733	0.375	1.667	2020-12-15
2018-01-15	서귀포시	남원읍	281654.2	26377.3	208010.1	516041.6	71.743	52.657	41.114	7.95	0	2.267	2020-12-15
2018-01-16	서귀포시	남원읍	282357.4	25995.98	190381.8	498735.1	68.371	51.743	42.286	10.275	57.875	2.7	2020-12-15
2018-01-17	서귀포시	남원읍	280542.9	25847.14	205060	511450.1	69.371	51.829	41.8	7.175	0.75	2.775	2020-12-15
2018-01-18	서귀포시	남원읍	282849	26147.11	226565.4	535561.5	73.257	52.8	41.057	6.6	0	2.2	2020-12-15
2018-01-19	서귀포시	남원읍	279346.8	26372.83	231199.3	536918.9	72	51.886	41.743	5.2	0	2.075	2020-12-15
2018-01-20	서귀포시	남원읍	283495.5	20840.95	241941.2	546277.7	72.943	53.286	40.914	4.625	0	2.675	2020-12-15
2018-01-21	서귀포시	남원읍	287369	19680.53	241988.6	549038.1	73.343	53.543	40.829	4.825	0	2.55	2020-12-15
2018-01-22	서귀포시	남원읍	277951.8	26731.67	203651.9	508335.3	69.8	52.8	41.057	4.825	2.125	2.525	2020-12-15
2018-01-23	서귀포시	남원읍	281898.8	25017.66	197426.5	504343	73.829	52.343	41.257	-1.8	1	4.25	2020-12-15
2018-01-24	서귀포시	남원읍	285863.9	23810.6	176802.6	486477.1	71.4	50.686	43.486	-5.15	2.125	4.525	2020-12-15

공공 데이터 포털의 제주특별자치도\_효율적인 교통량 측정을 위한 날씨유동인구 활용

# II 데이터 가공

## 데이터 가공 방법 1

```
import pandas as pd

jeju = pd.read_csv('./제주특별자치도_주제7_효율적인 교통량 측정을 위한 날씨유동인')

a=jeju.groupby(['읍면동명']).agg({'평균 속도':'mean'}).reset_index()
a.rename(columns = {'평균 속도' : '평균속도_mean'}, inplace = True)

b=jeju.groupby(['읍면동명']).agg({'평균 속도':'median'}).reset_index()
b.rename(columns = {'평균 속도' : '평균속도_median'}, inplace = True)
a = pd.merge(a,b, on = '읍면동명')

b=jeju.groupby(['읍면동명']).agg({'교통량':'mean'}).reset_index()
b.rename(columns = {'교통량' : '교통량_mean'}, inplace = True)
a = pd.merge(a,b, on = '읍면동명')

b=jeju.groupby(['읍면동명']).agg({'교통량':'median'}).reset_index()
b.rename(columns = {'교통량' : '교통량_median'}, inplace = True)
a = pd.merge(a,b, on = '읍면동명')

b=jeju.groupby(['읍면동명']).agg({'평균 소요 시간':'mean'}).reset_index()
b.rename(columns = {'평균 소요 시간' : '평균소요시간_mean'}, inplace = True)
a = pd.merge(a,b, on = '읍면동명')

b=jeju.groupby(['읍면동명']).agg({'평균 소요 시간':'median'}).reset_index()
b.rename(columns = {'평균 소요 시간' : '평균소요시간_median'}, inplace = True)
a = pd.merge(a,b, on = '읍면동명')

b=jeju.groupby(['읍면동명']).agg({'거주인구':'mean'}).reset_index()
b.reset_index().rename(columns = {'거주인구' : '거주인구_mean'}, inplace = True)
a = pd.merge(a,b, on = '읍면동명')
```

```
b=jeju.groupby(['읍면동명']).agg({'거주인구':'median'}).reset_index()
b.rename(columns = {'거주인구' : '거주인구_median'}, inplace = True)
a = pd.merge(a,b, on = '읍면동명')
```

```
b=jeju.groupby(['읍면동명']).agg({'군무인구':'mean'}).reset_index()
b.rename(columns = {'군무인구' : '군무인구_mean'}, inplace = True)
a = pd.merge(a,b, on = '읍면동명')
```

```
b=jeju.groupby(['읍면동명']).agg({'군무인구':'median'}).reset_index()
b.rename(columns = {'군무인구' : '군무인구_median'}, inplace = True)
a = pd.merge(a,b, on = '읍면동명')
```

```
b=jeju.groupby(['읍면동명']).agg({'방문인구':'mean'}).reset_index()
b.rename(columns = {'방문인구' : '방문인구_mean'}, inplace = True)
a = pd.merge(a,b, on = '읍면동명')
```

```
b=jeju.groupby(['읍면동명']).agg({'방문인구':'median'}).reset_index()
b.rename(columns = {'방문인구' : '방문인구_median'}, inplace = True)
a = pd.merge(a,b, on = '읍면동명')
```

```
b=jeju.groupby(['읍면동명']).agg({'총 유동인구':'mean'}).reset_index()
b.rename(columns = {'총 유동인구' : '총유동인구_mean'}, inplace = True)
a = pd.merge(a,b, on = '읍면동명')
```

```
b=jeju.groupby(['읍면동명']).agg({'총 유동인구':'median'}).reset_index()
b.rename(columns = {'총 유동인구' : '총유동인구_median'}, inplace = True)
a = pd.merge(a,b, on = '읍면동명')
```

```
a.to_csv('add_data.csv', index = False, encoding = 'ANSI')
```

읍면동명을 기준으로 groupby 후 컬럼 별 mean값과 median 값으로 새 피쳐 대체



## 데이터 가공

## 데이터 가공 방법 2

```
import pandas as pd
import numpy as np

train_x = pd.read_csv('./train.csv')

cluster_centers = np.array([[33.2799141, 126.7207056], [33.2482109, 126.5113278]])

def make_cluster_41(train):
    from sklearn.cluster import KMeans
    train_c = train[['start_latitude', 'start_longitude']]
    k_mean = KMeans(n_clusters=41, init=cluster_centers, random_state=np.random)
    train['location_cluster_41'] = k_mean.fit_predict(train_c)
    return train

train_x = make_cluster_41(train_x)
addlist=[('강남구', '대치동', '대치동'), ('강남구', '반포동', '반포동'), ('강남구', '삼성동', '삼성동'), ('강남구', '신사동', '신사동'), ('강남구', '논현동', '논현동'), ('강남구', '역삼동', '역삼동'), ('강남구', '개포동', '개포동'), ('강남구', '도곡동', '도곡동'), ('강남구', '우정동', '우정동'), ('강남구', '서초동', '서초동'), ('강남구', '천지동', '천지동'), ('강남구', '표선동', '표선동'), ('강남구', '오래동', '오래동'), ('강남구', '외곡동', '외곡동'), ('강남구', '한성동', '한성동'), ('강남구', '한림동', '한림동'), ('강남구', '화곡동', '화곡동'), ('강남구', '이도1동', '이도1동'), ('강남구', '이도2동', '이도2동'), ('강남구', '조천동', '조천동'), ('강남구', '한경면', '한경면')]

def adr(df):
    df['읍면동명']='알수없음'
    for i in range(0,len(addlist)):
        df['읍면동명'][(df['location_cluster_41'] == i)] = addlist[i]

adr(train_x)

print(train_x.info())

jeju = pd.read_csv('./외부데이터.csv', encoding = 'cp949')

print(jeju.info())
train_x = pd.merge(train_x, jeju, on = '읍면동명')

train_x.drop(['읍면동명'], axis = 1 , inplace = True)
jeju.drop(['읍면동명'], axis = 1 , inplace = True)
```

데이콘 제주도 도로 교통량 데이터를  
전처리 해주었던 읍면동명과 매칭되도록  
위도와 경도 데이터로 총 41개  
clustering을 해준 뒤

두 데이터를 읍면동명 기준으로  
merge하여 한 데이터셋으로 병합

## II 데이터 가공

### 데이터 가공 방법 3

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split

df = pd.read_csv('train_merge.csv')

X = df.drop(['target'], axis = 1)
y = df['target']
train_x, test_x, train_y, test_y = train_test_split(X, y, test_size = 0.1, random_state = 42)

train_x.to_csv('Final_train_x.csv', index = False)
test_x.to_csv('Final_test_x.csv', index = False)
train_y.to_csv('Final_train_y.csv', index = False)
test_y.to_csv('Final_test_y.csv', index = False)
```

병합해준 데이터를 split을 활용하여  
90%는 train 데이터로 10%는 테스트 데이터로 분리

## 리더보드 제작 / 베이스라인 구축

# 리더보드 제작 / 베이스라인 구축

## 1. 리더보드 제작

### 제주도 교통량 예측

2022학년도 2학기 인공지능 템프로젝트

3 teams · 2 days ago

[Overview](#) [Data](#) [Code](#) [Discussion](#) [Leaderboard](#) [Rules](#) [Team](#) [Host](#)

[Submissions](#)

[Late Submission](#)

...

Overview

Edit

#### Description

Eval

+ Add Page

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.

제주도의 교통 정보로부터 도로 교통량 회귀 예측

# 리더보드 제작 / 베이스라인 구축

## 1. 리더보드 제작 - 1차

<a href="#">submit_dnn (3).csv</a>			16 days ago	10.56281	10.56281	<input type="checkbox"/>
<a href="#">submit_dnn (1).csv</a>			16 days ago	Failed	Failed	
<a href="#">submit_dnn.csv</a>			16 days ago	10.41468	10.41468	<input type="checkbox"/>
<a href="#">submit_dnn.csv</a>			16 days ago	10.41787	10.41787	<input type="checkbox"/>
<a href="#">submit_dnn.csv</a>			16 days ago	13.32379	13.32379	<input type="checkbox"/>
<a href="#">submit_dnn.csv</a>			16 days ago	10.41776	10.41776	<input type="checkbox"/>
<a href="#">submit_dnn.csv</a>			16 days ago	10.16374	10.16374	<input type="checkbox"/>
<a href="#">submit_dnn.csv</a>			16 days ago	10.36675	10.36675	<input type="checkbox"/>
<a href="#">submit_dnn.csv</a>			16 days ago	13.31924	13.31924	<input type="checkbox"/>
<a href="#">submit_dnn.csv</a>			16 days ago	13.47213	13.47213	<input type="checkbox"/>
<a href="#">submit_dnn.csv</a>			16 days ago	13.58588	13.58588	<input type="checkbox"/>
<a href="#">submit_dnn.csv</a>			16 days ago	13.64689	13.64689	<input type="checkbox"/>
<a href="#">submit_dnn.csv</a>			16 days ago	12.68250	12.68250	<input type="checkbox"/>
<a href="#">submit_dnn.csv</a>			16 days ago	12.97071	12.97071	<input type="checkbox"/>
<a href="#">submit_dnn.csv</a>			16 days ago	13.25675	13.25675	<input type="checkbox"/>
<a href="#">submit_dnn.csv</a>			16 days ago	13.60167	13.60167	<input type="checkbox"/>
<a href="#">submit_dnn.csv</a>			16 days ago	28.73257	28.73257	<input type="checkbox"/>

여러 번의 시도 끝에  
10.56281의 MAE 값을  
Baseline 1로 설정

# 리더보드 제작 / 베이스라인 구축

## Baseline 1

```
str_col = ['day_of_week', 'start_turn_restricted', 'end_turn_restricted']
for i in str_col:
    le = LabelEncoder()
    le=le.fit(train_x[i])
    train_x[i]=le.transform(train_x[i])

    for label in np.unique(test_x[i]):
        if label not in le.classes_:
            le.classes_ = np.append(le.classes_, label)
        test_x[i]=le.transform(test_x[i])





















train_x = train_x.drop(['id', 'base_date', 'road_name', 'start_node_name', 'end_node_name', 'vehicle_restricted', '평균속도_mean', '평균속도_median', '교통량_mean', '교통량_median', '평균 소요 시간_mean', '평균 소요 시간_median', '거주인구', '거주인구 median', '근무인구_mean', '근무인구 median', '방문인구_mean', '방문인구 median', '총 유동인구_mean', '총 유동인구 median'], axis=1)

test_x = test_x.drop(['id', 'base_date', 'road_name', 'start_node_name', 'end_node_name', 'vehicle_restricted', '평균속도_mean', '평균속도_median', '교통량_mean', '교통량_median', '평균 소요 시간_mean', '평균 소요 시간_median', '거주인구', '거주인구 median', '근무인구_mean', '근무인구 median', '방문인구_mean', '방문인구 median', '총 유동인구_mean', '총 유동인구 median'], axis=1)
```

Baseline 1이기 때문에 별다른 FE없이  
Scale이 큰 피쳐들만 drop 시키고 학습한 결과를 baseline 1로 설정하였음

# 리더보드 제작 / 베이스라인 구축

## Baseline 2

Name		Date	Public	Private	Benchmark
<a href="#">base.csv</a>	 	10 days ago	4.83437	4.83437	<input checked="" type="checkbox"/>
<a href="#">submission_round(2).csv</a>	 	10 days ago	4.99622	4.99622	<input type="checkbox"/>
<a href="#">submission_round(1).csv</a>	 	10 days ago	5.61569	5.61569	<input type="checkbox"/>
<a href="#">submission_round.csv</a>	 	10 days ago	4.83437	4.83437	<input type="checkbox"/>
<a href="#">submit_dnn(5).csv</a>	 	10 days ago	9.70845	9.70845	<input type="checkbox"/>
<a href="#">submit_dnn(4).csv</a>	 	10 days ago	10.56281	10.56281	<input type="checkbox"/>
<a href="#">submission_round(2).csv</a>	 	13 days ago	4.83437	4.83437	<input type="checkbox"/>
<a href="#">submission_round(1).csv</a>	 	13 days ago	5.71416	5.71416	<input type="checkbox"/>
<a href="#">submission_round.csv</a>	 	13 days ago	5.91510	5.91510	<input type="checkbox"/>
<a href="#">submission(18).csv</a>	 	13 days ago	5.91853	5.91853	<input type="checkbox"/>

여러 번의 시도 끝에  
4.83437의 MAE 값을  
Baseline 2로 설정

# 리더보드 제작 / 베이스라인 구축

## Baseline 2

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train=sc.fit_transform(x_train)
test=sc.transform(test)
```

Baseline 1에서 drop 시켰던 피쳐들을 다 살려서  
Standard Scale을 해줌

+  
Epoch 수를 늘린 결과를

Baseline2로 설정하였음

```
for stop in range(1000):

    # 그래디언트 초기화
    optimizer.zero_grad()
    # Forward 계산
    hypothesis = model(x_train).cuda()
    # Error 계산
    cost = loss(hypothesis, y_train).cuda()
    # Backward 계산
    cost.backward()
    # 가중치 갱신
    optimizer.step()

    if stop % 100 == 0:
        print(stop, cost.item())
```





## 느낀점

1. 문제를 푸는 것만 진행해보다 직접 문제를 만들어보니 흥미로움
2. 데이터 가공할 때 꽤 어려움이 많음

# Thank you