

# 「산업 빅데이터 분석 실제」

(데이터 분석\_회귀, 분류, 군집 결과)

2020254008

최 원 희

2021. 12. 13.

# [데이터 분석 목적]

## 분석 데이터


자율주행 셔틀 운행 탑승객 및 주행거리

## 목적

자율주행 셔틀의 실제 운행 데이터 취득을 통한 누적거리 및 탑승객 현황의 데이터 분석을 통한 차량 운행 시간 조정 및 API 통신 연결에 부과 되는 통신료 절감, 기간별 증가율 수치 현황 파악 등


# [데이터셋 관리 툴]

FolderName: Left




17L.jpg

FolderName: Center



17C.bmp

FolderName: Right



17R.jpg

17 / 354

Play

Stop

Option

Folder List

Num	Path
7	F:\WDATASET\WObstacle\WGRP20_A007_20131106_LS_normal_1280_960_sunr
8	F:\WDATASET\WObstacle\WGRP20_A008_20131106_LS_normal_1280_960_sunr
9	F:\WDATASET\WObstacle\WGRP20_A009_20131106_LS_normal_1280_960_sunr
10	F:\WDATASET\WObstacle\WGRP20_A010_20131106_LS_normal_1280_960_sunr
11	F:\WDATASET\WObstacle\WGRP20_A011_20131106_LS_normal_1280_960_sunr
12	F:\WDATASET\WObstacle\WGRP20_A012_20131106_LS_normal_1280_960_sunr
13	F:\WDATASET\WObstacle\WGRP20_A013_20131106_LS_normal_1280_960_sunr
14	F:\WDATASET\WObstacle\WGRP20_A014_20131106_LS_normal_1280_960_sunr
15	F:\WDATASET\WObstacle\WGRP20_A015_20131106_LS_normal_1280_960_sunr
16	F:\WDATASET\WObstacle\WGRP20_A016_20131106_LS_normal_1280_960_sunr
17	F:\WDATASET\WObstacle\WGRP20_A017_20131106_LS_normal_1280_960_sunr
18	F:\WDATASET\WObstacle\WGRP20_A018_20131106_LS_normal_1280_960_sunr
19	F:\WDATASET\WObstacle\WGRP20_A019_20131106_LS_normal_1280_960_sunr
20	F:\WDATASET\WObstacle\WGRP20_A020_20131106_LS_normal_1280_960_sunr

Browse

Add

Edit

Delete

open Folder

Filepath List

Num	File Name
00001	0L.jpg
00002	1L.jpg
00003	2L.jpg
00004	3L.jpg
00005	4L.jpg
00006	5L.jpg
00007	6L.jpg
00008	7L.jpg
00009	8L.jpg
00010	9L.jpg
00011	10L.jpg
00012	11L.jpg
00013	12L.jpg
00014	13L.jpg
00015	14L.jpg
00016	15L.jpg
00017	16L.jpg

# [데이터 분석 종류]

운영 현황	서틀	SCN001		증감량	SCN002		증감량	SCN003		증감량	SCN004		증감량	SCN005		증감량	Total		증감량
	운영 지역	대구 수성알파시티			세종			서울 상암 DMC			고군산군도			고군산군도			10월 1주 10월 2주		
	일자	10월 1주	10월 2주		10월 1주	10월 2주		10월 1주	10월 2주		10월 1주	10월 2주							
주간 현황	주행거리(km)	43	62	19	23	63	40	53	57	4	21	46	25	49	41	-8	189	269	80
	탑승객(명)	9	29	20	3	0	-3	0	0	0	0	0	0	0	0	0	12	29	17
	DTG 데이터 용량(MB)	0.38	0.92	0.54	0.26	1.71	1.45	1.42	2.92	1.50	0.88	1.86	0.98	1.82	1.49	-0.33	4.76	8.9	4.14
	DVR 데이터 용량(GB)	32.48	55	22.52	13.77	90.35	76.58	65.84	39.22	-26.62	9.36	25.44	16.08	19.88	29.94	10.06	141.33	239.95	98.62
전체 현황	총 주행거리(km)	2976	3038	62	1119	1182	63	978	1035	57	298	344	46	303	344	41	5674	5943	269
	총 탑승객(명)	598	627	29	4697	4697	0	757	757	0	58	58	0	45	45	0	6155	6184	29
	총 DTG 데이터 용량(MB)	27.19	28.11	0.92	35.94	37.65	1.71	28.6	31.52	2.92	8.84	10.7	1.86	6.9	8.39	1.49	107.47	116.37	8.9
	총 DVR 데이터 용량(GB)	1503.91	1558.91	55	719.58	809.93	90.35	462.51	501.73	39.22	487.06	512.5	25.44	361.03	390.97	29.94	3534.09	3774.04	239.95

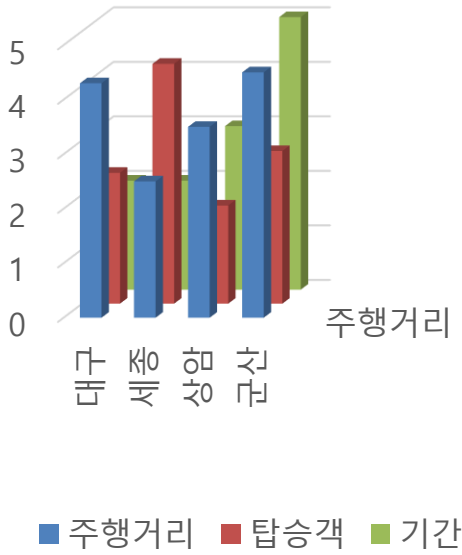
👉 각 차량에 장착된 NAS PC에서 검출된 데이터를 정리하여 표기 완료  
(서틀 이름, 운영 지역, 운행 일자, 해당 주간 현황, 누적 현황 정리)

[데이터 시각화 방법]



# [데이터 분석 예상 결과]

지역	차량번호	구분		기간
대구	SCN001	총 주행거리(km)	2,801	2020.01~2021.07
		총 탑승객(명)	574	
세종	SCN002	총 주행거리(km)	948	
		총 탑승객(명)	4,598	
상암	SCN003	총 주행거리(km)	766	
		총 탑승객(명)	747	
군산	SCN004	총 주행거리(km)	183	
		총 탑승객(명)	58	
	SCN005	총 주행거리(km)	157	
		총 탑승객(명)	40	



☞ 각 지역별 주행거리와 탑승객 꾸준히 상승 / 운행에 따른 차량 정비 모니터링 가능



# [데이터 분석\_회귀 결과]

- 매개변수 모델을 이용하여 통계적으로 변수들 사이의 관계를 추정하는 분석방법
- 주로 독립변수가 종속변수에 미치는 영향을 확인하고자 사용하는 분석 방법
- 하나의 종속변수와 하나의 독립변수 사이의 관계를 분석할 때 단순회귀분석, 하나의 종속변수와 여러 독립 변수 사이 관계를 규명할 때는 다중회귀분석이라고 함

```
X_test = sm.add_constant(X_test)

y_test_pred = model_trained.predict(X_test.drop(['INDUS', 'AGE'], axis=1))
y_test_pred.head()
```

```
210    22.787949
24     15.482380
36     22.346630
439    13.433993
161    36.890921
dtype: float64
```

```
print('Training MSE: {:.3f}'.format(mean_squared_error(y_train, y_train_pred)))
print('Training RMSE: {:.3f}'.format(np.sqrt(mean_squared_error(y_train, y_train_pred))))
print('Training MAE: {:.3f}'.format(mean_absolute_error(y_train, y_train_pred)))
print('Training MAPE: {:.3f}'.format(mean_absolute_percentage_error(y_train, y_train_pred)))
print('Training R2: {:.3f}'.format(r2_score(y_train, y_train_pred)))
```

```
Training MSE: 21.881
Training RMSE: 4.678
Training MAE: 3.315
Training MAPE: 51.174
Training R2: 0.756
```

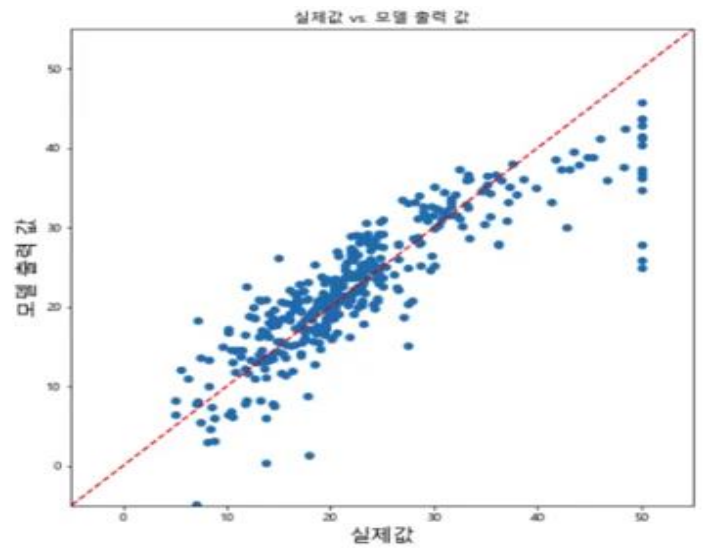
```
print('Testing MSE: {:.3f}'.format(mean_squared_error(y_test, y_test_pred)))
print('Testing RMSE: {:.3f}'.format(np.sqrt(mean_squared_error(y_test, y_test_pred))))
print('Testing MAE: {:.3f}'.format(mean_absolute_error(y_test, y_test_pred)))
print('Testing MAPE: {:.3f}'.format(mean_absolute_percentage_error(y_test, y_test_pred)))
print('Testing R2: {:.3f}'.format(r2_score(y_test, y_test_pred)))
```

```
Testing MSE: 23.063
Testing RMSE: 4.802
Testing MAE: 3.512
Testing MAPE: 43.946
Testing R2: 0.639
```

```
y_train_pred = model_trained.fittedvalues

plt.figure(figsize=(8, 8))
plt.title('실제값 vs. 모델 출력 값')
plt.scatter(y_train, y_train_pred)
plt.plot([-5, 55], [-5, 55], ls='--', c='red')
plt.xlabel('실제값', size=16)
plt.ylabel('모델 출력 값', size=16)
plt.xlim(-5, 55)
plt.ylim(-5, 55)
plt.show()
```

OLS Regression Results						
Dep. Variable:	MEDV	R-squared:	0.756			
Model:	OLS	Adj. R-squared:	0.748			
Method:	Least Squares	F-statistic:	93.01			
Date:	Fri, 12 Mar 2021	Prob (F-statistic):	1.02e-110			
Time:	03:07:02	Log-Likelihood:	-1186.3			
No. Observations:	404	AIC:	2421			
Df Residuals:	390	BIC:	2477			
Df Model:	13					
Covariance Type:	nonrobust					
	coef	std err	t	P> t	[0.025	0.975]
const	35.0744	5.844	6.002	0.000	23.586	46.564
CRIM	-0.1146	0.034	-3.419	0.001	-0.180	-0.049
ZN	0.0532	0.015	3.488	0.001	0.023	0.083
INDUS	0.0033	0.068	0.048	0.962	-0.131	0.139
CHAS	3.5085	1.013	3.464	0.001	1.517	5.500
NOX	-18.1357	4.211	-4.307	0.000	-26.414	-9.857
RM	3.8252	0.478	8.011	0.000	2.886	4.764
AGE	0.0111	0.015	0.714	0.476	-0.019	0.042
DIS	-1.5300	0.227	-6.731	0.000	-1.977	-1.083
RAD	0.3392	0.071	4.747	0.000	0.199	0.480
TAX	-0.0119	0.004	-2.918	0.004	-0.020	-0.004
BPTRATIO	-0.8842	0.149	-5.920	0.000	-1.178	-0.591
B	0.0095	0.003	3.227	0.001	0.004	0.015
LSTAT	-0.5782	0.059	-9.850	0.000	-0.694	-0.463
Omnibus:	137.227	Durbin-Watson:	1.873			
Prob(Omnibus):	0.000	Jarque-Bera (JB):	530.658			
Skew:	1.475	Prob(JB):	5.89e-116			
Kurtosis:	7.777	Cond. No.	1.55e+04			



# [데이터 분석\_분류 결과]

- 다수의 속성 또는 변수를 갖는 객체를 사전에 정해진 그룹 또는 범주의 하나로 분류하여 분석
- 데이터 수집을 위해 수집주기 및 방법에 관한 분석 필요
- 원천 데이터와 수집한 데이터 동일 시스템에 저장되어 있으므로 원천 데이터가 외부에 있는 경우와 비교했을 때 상대적으로 기술적 제약도 적은 편

```
import pandas as pd
import numpy as np
import time
from sklearn.model_selection import StratifiedKFold as SKF
from sklearn.metrics import roc_auc_score, f1_score, precision_score, accuracy_score

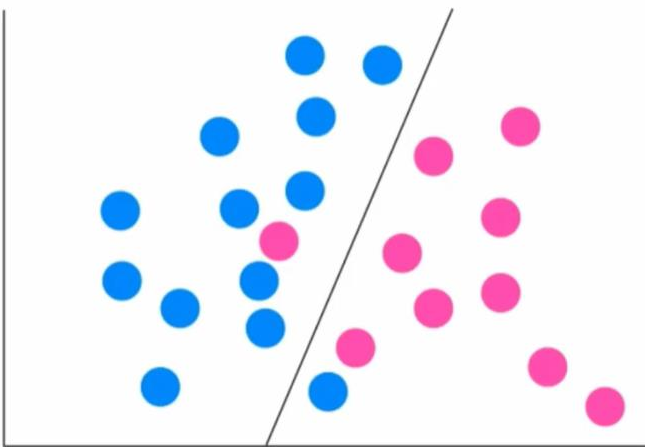
X_train = pd.read_csv('X_train.csv', encoding='cp949')
y_train = pd.read_csv('y_train.csv', encoding='cp949')
X_train = X_train.drop('cust_id', 1).fillna(0)

import xgboost
xgb_clf = xgboost.XGBClassifier()

from sklearn.linear_model import LogisticRegression
log_clf = LogisticRegression()

from sklearn.ensemble import RandomForestClassifier
rf_clf = RandomForestClassifier()

from sklearn.svm import LinearSVC as SVC
svc = SVC()
```



형태별 분류	위치별 분류	주체별 분류
<ul style="list-style-type: none"><li>정형 데이터</li><li>반정형 데이터</li><li>비정형 데이터</li></ul>	<ul style="list-style-type: none"><li>내부 데이터</li><li>외부 데이터</li></ul>	<ul style="list-style-type: none"><li>차량별 생성</li><li>주행거리 생성</li><li>탑승객수 생성</li></ul>

	model	time(sec)	acc	roc_auc	f1	prec
0	GaussianNB	0.060	0.478571	0.561810	0.564258	0.411565
1	QuadraticDiscriminantAnalysis	0.163	0.561429	0.549891	0.417406	0.468918
2	LogisticRegression	0.166	0.624000	0.500000	0.000000	0.000000
3	SGDClassifier	0.267	0.553143	0.523718	0.269117	0.206620
4	KNeighborsClassifier	0.315	0.598286	0.551149	0.402342	0.455406
5	LinearDiscriminantAnalysis	0.328	0.645714	0.570359	0.361091	0.562725
6	DecisionTreeClassifier	0.347	0.568000	0.541448	0.430379	0.426772
7	LinearSVC	1.037	0.473714	0.475191	0.311261	0.239456
8	XGBClassifier	1.516	0.619429	0.573021	0.432145	0.492654
9	RandomForestClassifier	4.330	0.642857	0.583466	0.417956	0.537872
10	GradientBoostingClassifier	5.958	0.656571	0.596425	0.435642	0.570028

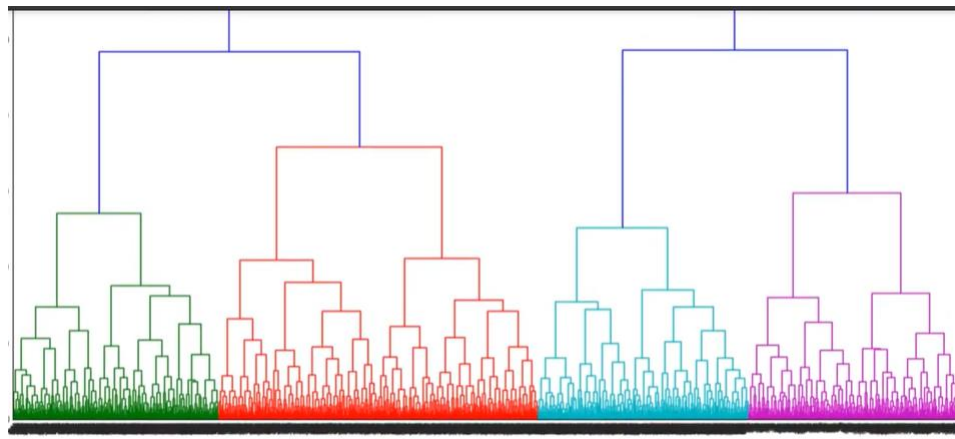
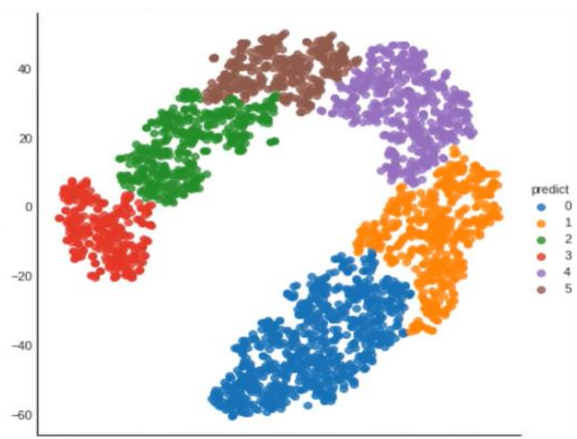


## [데이터 분석\_군집 결과]

- 데이터의 특성에 따라 유사한 것끼리 묶음
- 유사성을 기반으로 군집을 분류하고, 군집에 따라 유형별 특징을 분석하는 기법
- 군집으로 묶여진 텍스트끼리는 최대한 유사하고, 다른 군집으로 묶여진 텍스트들과 최대한 유사하지 않도록 분류

```
from scipy.spatial.distance import pdist, squareform
from scipy.cluster.hierarchy import linkage, dendrogram

distmatrix = pdist(df, metric='euclidean')
row_dist = pd.DataFrame(squareform(distmatrix))
row_dist
```



**감사합니다**