

[빅데이터전문가]
빅데이터분석 머신러닝 활용

ZenSla

A팀

강신우
이준혁
윤성준
장홍석
정준형

2022년 7월

목 차

1. 팀 소개
2. 프로젝트 개요
3. 데이터 처리 기획
4. 데이터 분석 과정
5. 시각화 (화면 구현)

Part 1,
팀 소개



팀 소개



강신우

팀장

- 프로젝트 관리 및 회의록 작성
- 충전소 데이터 수집
- 최종 입지 시각화용 웹 개발
- 발표 자료 작성



이준혁

부 팀장

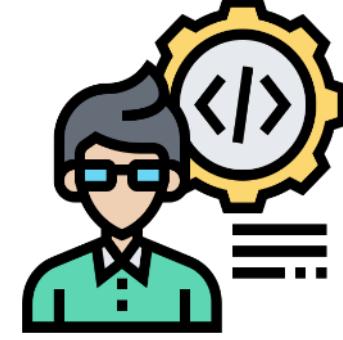
- 카카오 Open API 데이터 수집 및 저장
- 입지분석 데이터 처리 및 분석
- 입지분석 결과 시각화
- 결과 보고서 작성
- 통계기법 자문



윤성준

팀원

- 차량, 인구수 데이터 수집 및 저장
- 뉴스기사 크롤링
- 발표 자료 ppt 작성
- 2차 군집 알고리즘 탐색



장홍석

팀원

- 형태소 분석 및 워드클라우드 시각화
- 전기차 데이터 전처리 및 분석
- 지역별 전기차 등록수 시각화
- DB설계
- 최종 입지 시각화용 웹 개발
- 전주시 인구수 데이터 전처리
- 전국 연료별 차량 데이터 수집 및 저장
- 웹 이벤트 작성 및 처리



정준형

팀원

Part 2,

프로젝트 개요

1. 개발 일정 관리

2. 개발환경

3. 프로젝트 배경



프로젝트 개요

- 개발 일정 관리

프로젝트 일정

SUN	MON	TUE	WED	THU	FRI	SAT
6.5	6 현충일	7 프로젝트 시작 컨텐츠 회의	8 요구사항 명세서 작성 및 자료 수집 시작 보고서 작성 시작	9 기능적 요구사항 정의 및 유스케이스 작성 자료 수집	10 요구 사항 명세서 및 자료수집 완료 카카오 API 활용법 공부	11 개인 별 부족한 점 보안
12 2주차 목표 설정 수집한 데이터 전처리, 전기차 예측 분석, HTML 작업 완료	13 수집 데이터 전처리	14 HTML 레이아웃 작성 및 데이터 예측 분석 모형 작업	15 HTML 작업 및 발표 자료 작성	16 HTML 작업 완료 및 발표 자료 마무리 1차 군집화 시작 Mcip, lscp 알고리즘 공부	17 학원 임시 휴일	18 개인 별 부족한 점 보안
19 3주차 목표 설정 1차 군집 분석 완성 2차 군집 분석을 위한 준비	20 중간 발표 HTML 작업, 2차 군집 공부	21 다중 선형 회귀 분석 완료	22 웹 페이지 작업 및 2차 군집	23 2차 군집을 위한 상관 분석 JSP 지도, 막대 그래프 이벤트 연동	24 1차 군집 완료 웹 페이지 구현 지도, 막대그래프 이벤트 연동	25 개인 별 부족한 점 보안
26 4주차 목표 설정 2차 군집 완성, 웹 페이지 구현 완성 발표 자료 준비 완성	27 웹 페이지 지도와 막대 그래프 연동 완성 2차 군집 시작	28 웹 페이지 및 맵 카테고리 구현 2차 군집	29 웹 페이지 구현 2차 군집 완료	30 웹 페이지 구현 최종 발표 자료 작성	7.1 웹 페이지 구현 마무리 최종 발표 자료 및 보고서 마무리	2 웹 페이지 테스트 및 발표 준비
3 웹 페이지 테스트 및 발표 준비	4 최종 발표					

프로젝트 개요

- 개발환경

개발환경

Technic

Python 3.8.8
HTML 5
CSS 3
JAVA 1.8.0_212
Kakaomap API
Java script
jQuery 1.8.1
Tomcat 8.5
D3.js 3.1.7
BootStrap 5.1.3
Chart.js 2.6.0
Highchart.js 10.1.0

Tool

Eclipse 2019-12
Spyder 4.2.5
Jupyter Notebook 6.3.0
QGIS 3.24.3

Data Base

MySQL 8.0.19

프로젝트 관리

SVN(형상관리)
Notion(일정관리)
한글(회의록 등)
Excel(데이터 관리)
PowerPoint(발표자료)

프로젝트 개요

- 프로젝트 배경

ESG



Environment (환경)

Social (사회적)

Governance (지배구조)

과거 기업은 1차원 목적(이윤추구 등)을 가지고 기업을 운영

현재 기업은 비재무적 요소 친환경, 지배구조까지 고려해 기준을 만들

프로젝트 개요

- 프로젝트 배경

프로젝트 선정 배경



"넓고 강해지는 유럽 ESG 환경 규제, 시장 친출 기회로 삼아야"

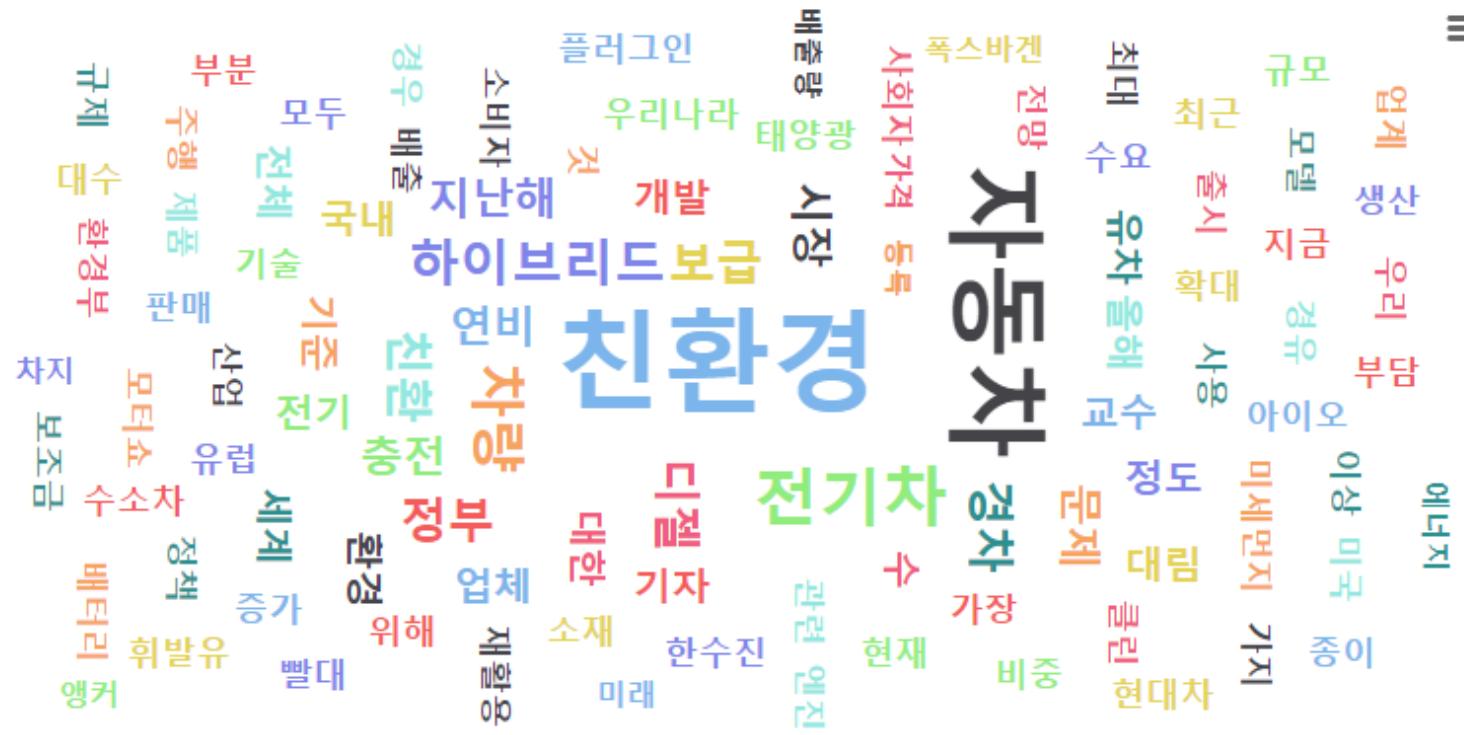
이날 행사는 최근 유럽연합(EU)이 친환경 디자인친환경 배터리 규정, 플라스틱 규제, 산업배출에 대한 지침 개정 등 다양한 규제를 연이어 도입함에 따라 우리 기업이 유럽으로 수출 시 의무해야 하는 제도들에 대한 정보와 주요국 사례를 공유하기 위해 마련했다.

Google에 의해 종료된 광고입니다

이날 행사는 최근 유럽연합(EU)이 친환경 디자인·친환경 배터리 규정, 플라스틱 규제, 산업배출에 대한 지침 개정 등 다양한 규제를 연이어 도입함에 따라 우리 기업이 유럽으로 수출 시 유의해야 할 제들을 대한 정보와 주요국 사례를 공유하기 위해 마련했다.

이날 발표에서 주별기유럽연합대사관 권순모 환경관은 “환경이 현 EU 정책당국의 최우선 정책으로 제”라면서 “유럽사회 전반에서 기후변화 대응과 생태계 복원에 대한 관심이 높아지면서 환경 규제 점점 강력해지고 넓어지는 추세”라고 분석했다.

한국과학기술연구원 유럽연구소 서정호 대외협력실장은 “최근 EU의 환경규제는 유럽의 경제체계를 순환경제로 전환시킬 미래 경쟁력을 확보하는 방향으로 추진되고 있다”며 “우리 기업이 유럽시장 진출에 성공하기 위해서는 제품을 판매하는데서 그치는 것이 아니라 추후 제품이 어떻게 재활용될 있는지까지도 고려해야 한다”고 강조했다.

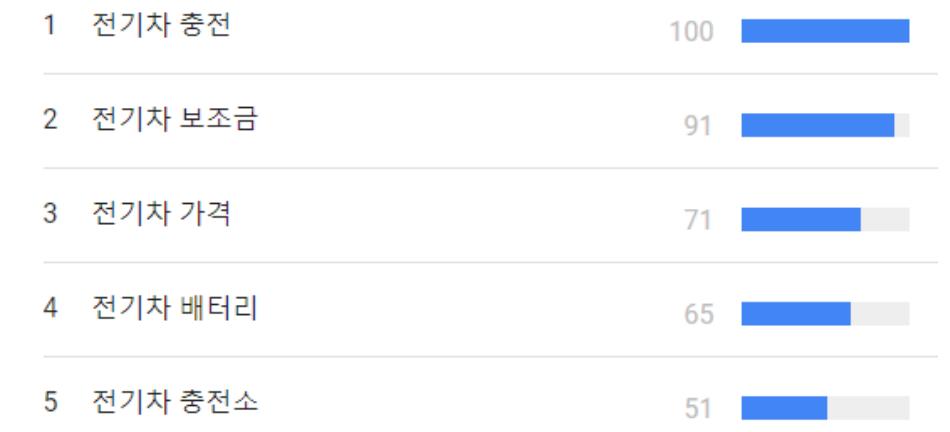
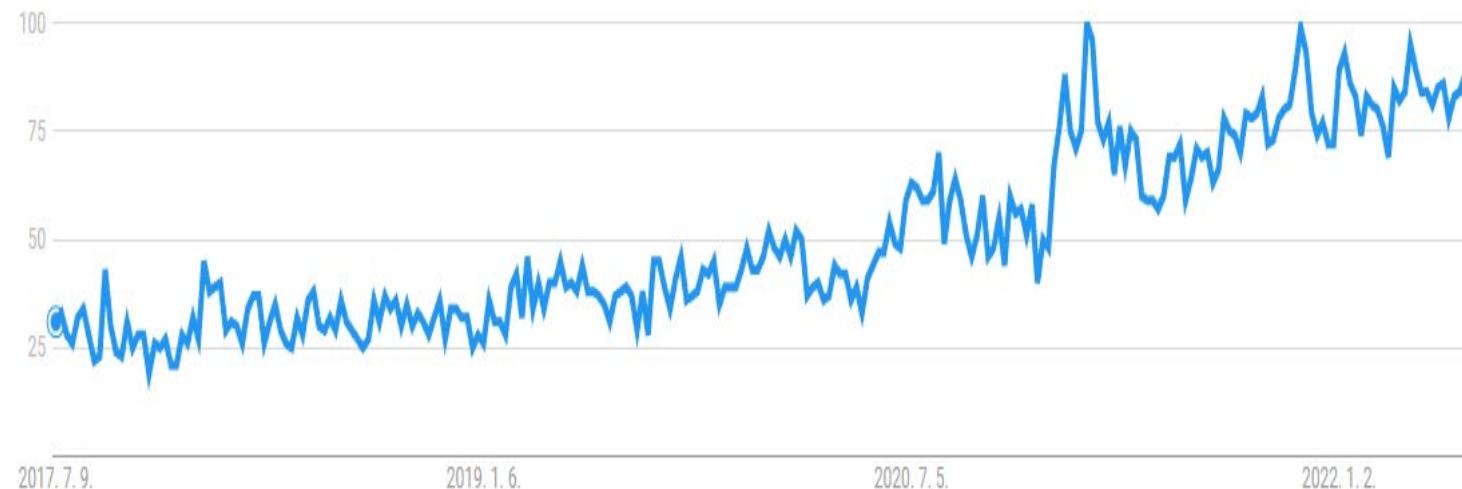


프로젝트 개요

- 프로젝트 배경

프로젝트 선정 배경

구글 전기차 관심도 변화 (2017.7 ~ 2022.6)

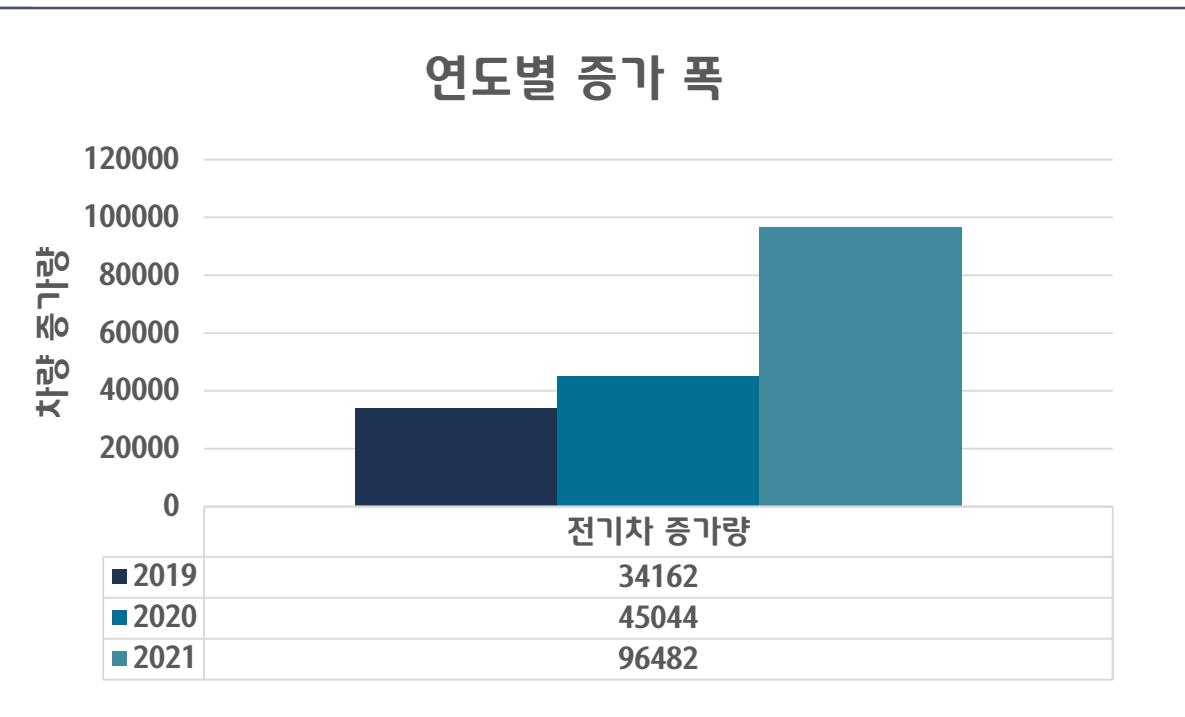
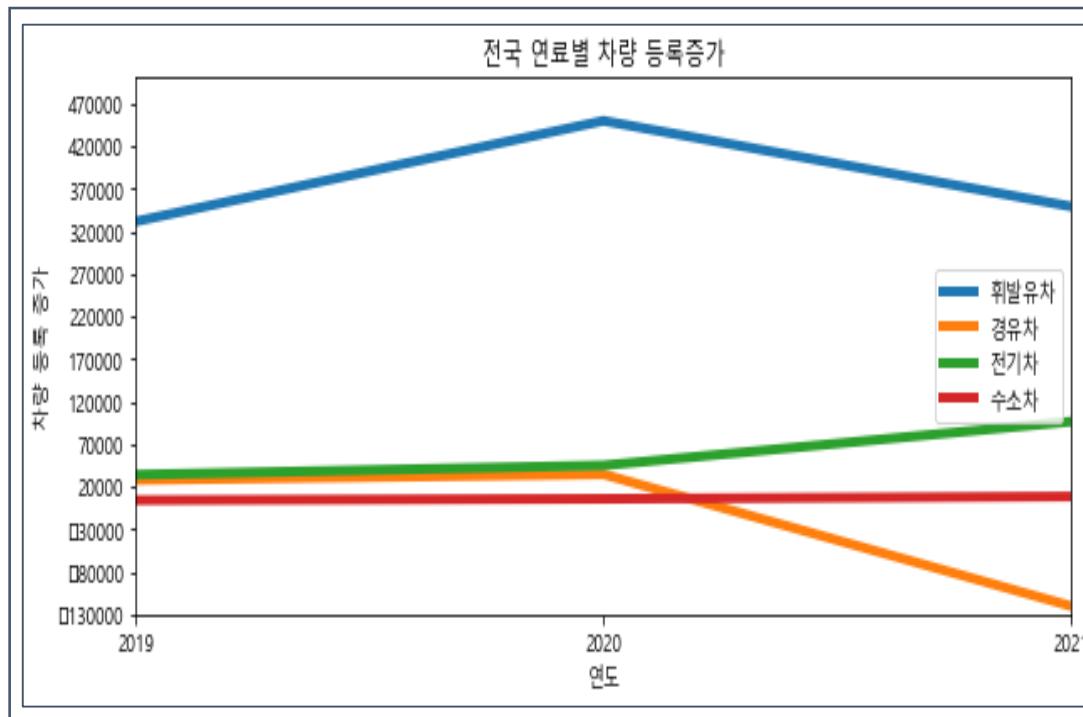


- 실제로 해당 정책으로 인해 전기차에 대한 관심이 많아졌다.

프로젝트 개요

- 프로젝트 배경

프로젝트 선정 배경



2019년 기준 휘발유차 약 1.05배 증가

2019년 기준 전기차 약 3배 증가

자료 : 국토 교통부

프로젝트 개요

- 프로젝트 배경

프로젝트 선정 배경

정부 전기차 충전 정책 만족도

단위: %



자료: 소프트베리

정부 전기차 충전 정책 만족도. 그래픽 김은교 기자

The JoongAng

시급한 개선이 필요한 전기차 충전 정책

단위: %

전기차 충전시설 보급 확대 40

미흡한 전기차 충전시설 관리 32

급속충전기 보급 확대 21

전기차 충전시설 정보의 실시간 확인 7



The JoongAng

자료: 소프트베리

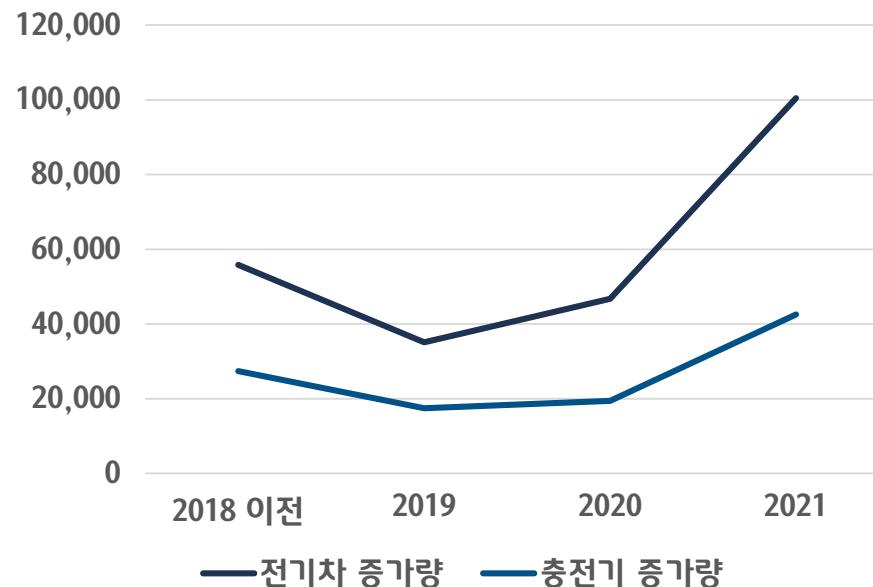
- 정책으로 전기차 수요는 크게 늘었지만, 충전 시설의 확충 정책이 부족하여 문제가 생긴 걸 확인할 수 있음.

프로젝트 개요

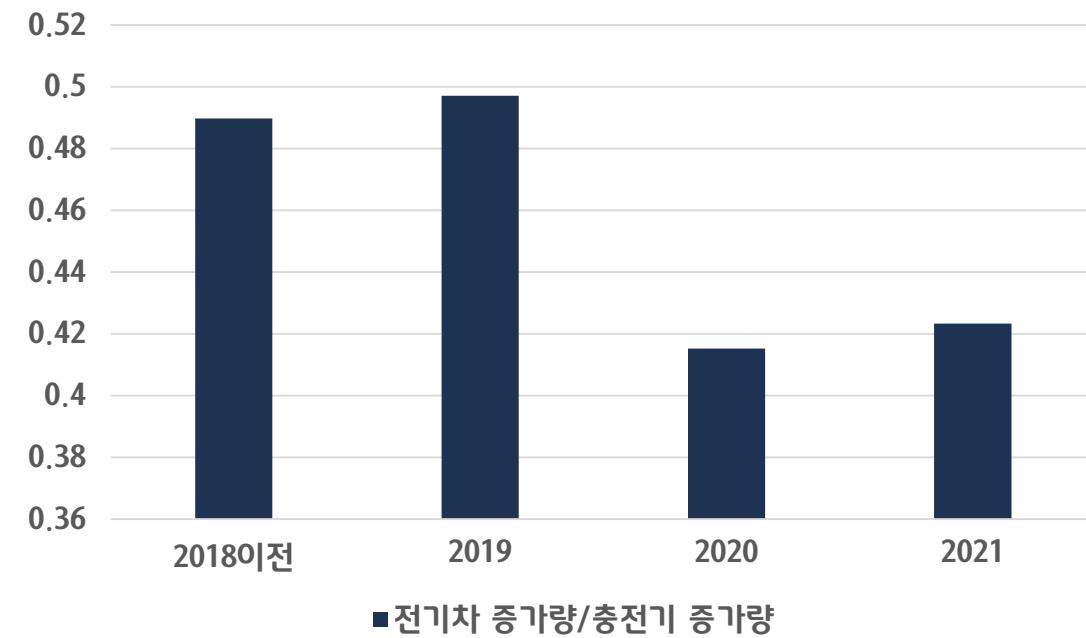
- 프로젝트 배경

프로젝트 선정 배경

연도별 전기차, 충전기 증가 현황



전기차 증가량 대비 충전기 증가량

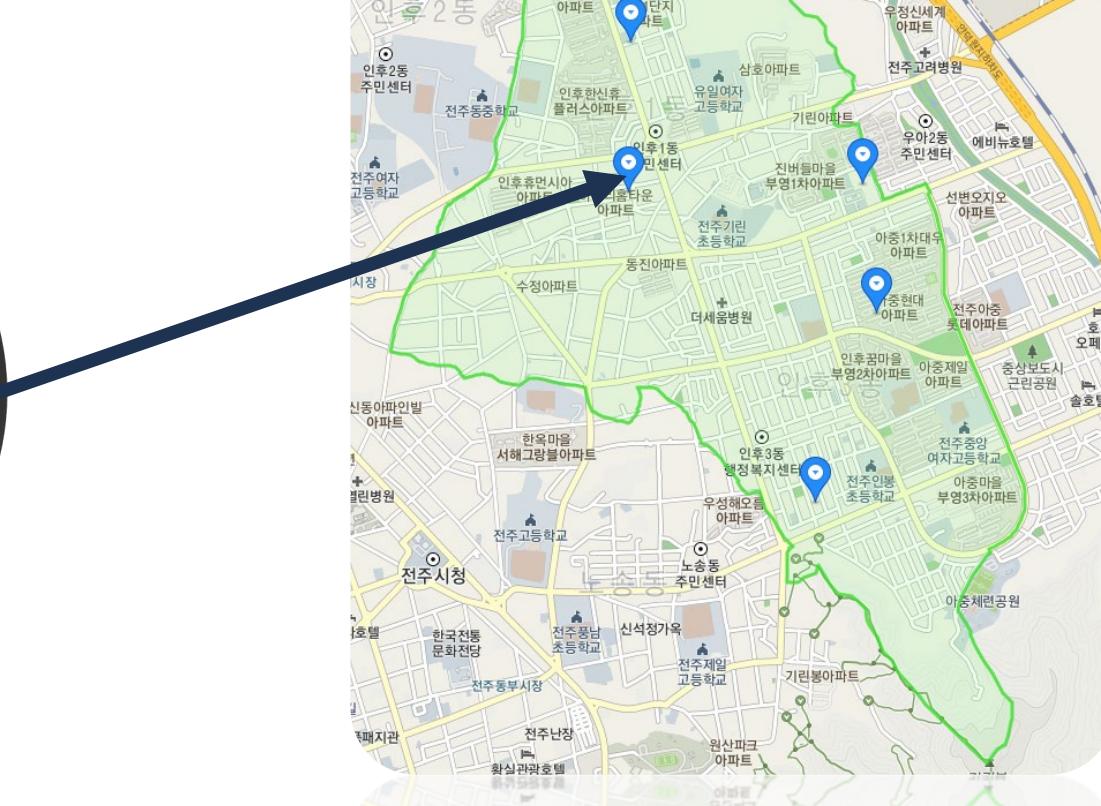


자료 : 국토 교통부

프로젝트 개요

- 프로젝트 배경

프로젝트 목적



- 전주시 전기차 충전소 보급확대를 위한 입지 분석
- 입지 선정 알고리즘 구축, 표준화 모델 제시

Part 3, **데이터 처리 기획**

1. 데이터 수집 계획

2. 데이터 처리 과정

3. 데이터 베이스 설계



데이터 처리 기획

- 데이터 수집 계획

데이터 수집

수집 대상

국토 교통부, 공공 데이터 포털, 국가공간정보포털, 행정표준코드관리시스템, 카카오 맵 (법정동 별 건물 데이터) 등

수집 기간

2022.06.08 ~ 2022.06.15

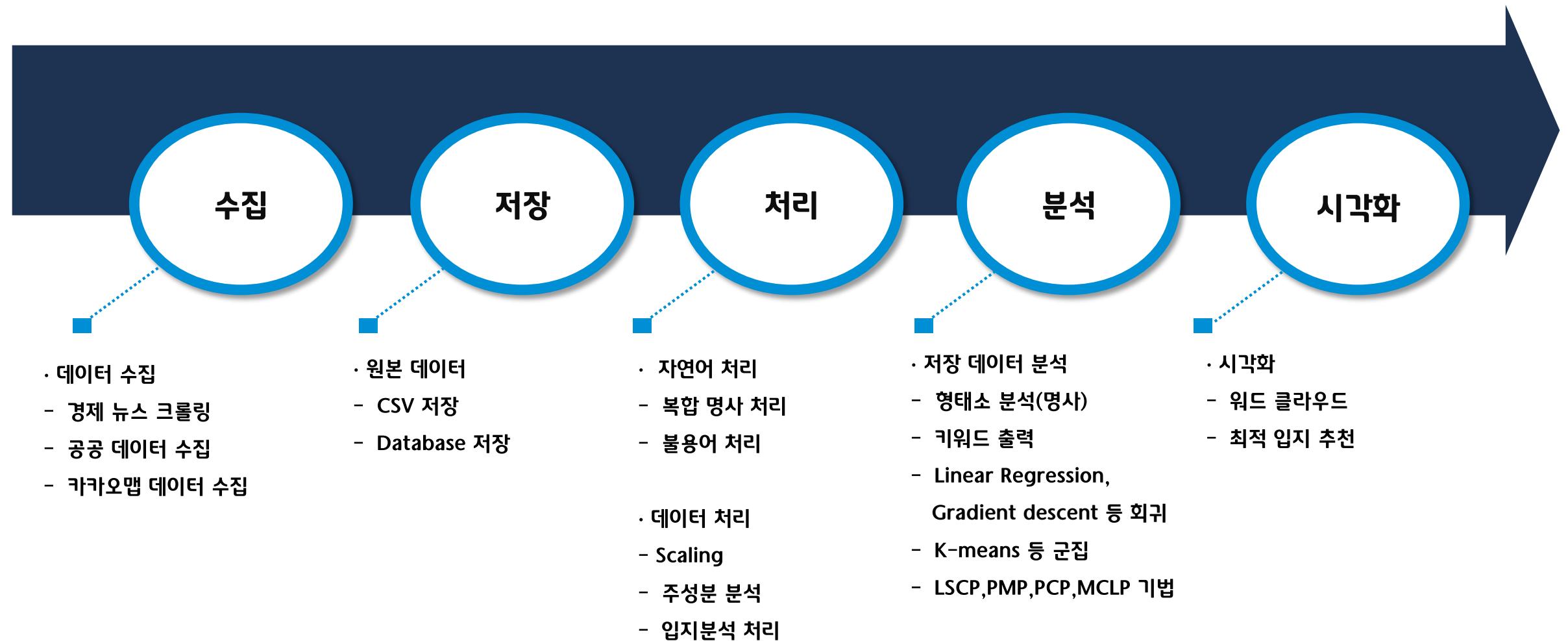
수집 방법

크롤링, Open API, 공공 데이터 활용 및 기관 문의

수집 내용

뉴스기사 키워드 추출, 전기차 및 충전소 현황 등

데이터 처리 과정



DB설계도

1

장소				
	눌리 이름*	클리 이름*	도메인 데이터 타입	널 허용
장소 번호	pno	N/A	INTEGER	N·N
장소 이름	pname	N/A	VARCHAR(60)	NULL
장소 주소	paddr	N/A	VARCHAR(60)	NULL
도로명 주소	pnewaddr	N/A	VARCHAR(60)	NULL
동 이름	pdong	N/A	VARCHAR(30)	NULL
img url 주소	pimg	N/A	VARCHAR(256)	NULL
카테고리 대분류	pcate	N/A	VARCHAR(30)	NULL
카테고리 소분류	plcate	N/A	VARCHAR(30)	NULL
카테고리1	pcated1	N/A	VARCHAR(30)	NULL
카테고리2	pcated2	N/A	VARCHAR(30)	NULL
카테고리3	pcated3	N/A	VARCHAR(30)	NULL
카테고리4	pcated4	N/A	VARCHAR(30)	NULL
카테고리5	pcated5	N/A	VARCHAR(30)	NULL
전화번호	ptel	N/A	VARCHAR(30)	NULL
위도	plat	N/A	VARCHAR(30)	NULL
경도	plon	N/A	VARCHAR(30)	NULL
utmk_x	utmk_x	N/A	VARCHAR(30)	NULL
utmk_y	utmk_y	N/A	VARCHAR(30)	NULL

2

크롤링				
	눌리 이름*	클리 이름*	도메인 데이터 타입	널 허용
기사번호	cr_no	N/A	INTEGER	N·N
기사제목	cr_word	N/A	VARCHAR(256)	NULL
기사주소	wordcount	N/A	INTEGER	NULL

3

차트				
	눌리 이름*	클리 이름*	도메인 데이터 타입	널 허용
지역번호	ano	N/A	BIGINT	N·N
시군구명	goo	N/A	VARCHAR(256)	NULL
읍면동명	dong	N/A	VARCHAR(256)	NULL
인구수	people	N/A	INTEGER	NULL
출전소 개수	charger	N/A	INTEGER	NULL
전기차 대수	car	N/A	INTEGER	NULL
주소	addr	N/A	VARCHAR(256)	NULL
전기차 예측값	pcar	N/A	INTEGER	NULL
급속 출전기	qcharger	N/A	INTEGER	NULL

4

최종결과				
	눌리 이름*	클리 이름*	도메인 데이터 타입	널 허용
장소 번호	bno	N/A	INTEGER	N·N
장소 이름	bname	N/A	VARCHAR(60)	NULL
장소 주소	baddr	N/A	VARCHAR(60)	NULL
동 이름	bdong	N/A	VARCHAR(30)	NULL
카테고리 대분류	bfcate	N/A	VARCHAR(30)	NULL
위도	blat	N/A	VARCHAR(30)	NULL
경도	blon	N/A	VARCHAR(30)	NULL

1. Kakao Map API DB

2. 워드 클라우드를 위한 DB

3. 지역분석을 위한 DB

4. 입지 추천 장소에 관한 DB

Part 4, **데이터 분석 과정**

1. 데이터 수집 및 저장

2. 데이터 처리

3. 데이터 분석



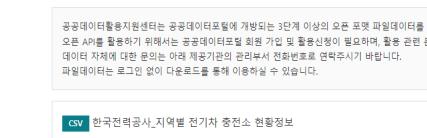
데이터 분석 과정

- 데이터 수집 및 저장

데이터 수집 및 저장



데이터 상세



```

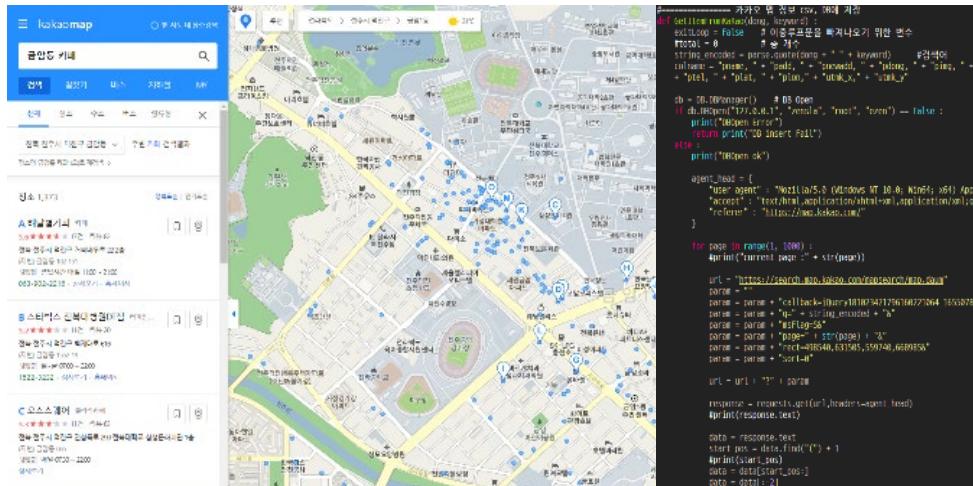
Spyder (Python 3.8)
File Edit Insert Cell Help Debug Console Projects Job View Help
D:\MyPy\Python\News\scraping\news.py D:\MyPy\Python\News\scraping\news.py K-means.py names22.py
125 # 환경은 임의로 설정해 놓았던 대로 잊으 경우
126 # 1) 풀어서하고
127 # 2) 디렉션에서 풀어놓고는 편리
128 if press_m in press_elt_bt_dict.keys():
129     print("[" + str(press_m) + "] 버튼을 누르면 " + K_MEANS + ".format(press_elt_bt.text, press_m))
130     press_elt_bt_dict[press_m].click()
131     time.sleep(sleep_sec)
132
133 break
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
587
588
589
589
590
591
592
593
594
595
596
597
597
598
599
599
600
601
602
603
604
605
606
607
607
608
609
609
610
611
612
613
613
614
615
615
616
616
617
617
618
618
619
619
620
620
621
621
622
622
623
623
624
624
625
625
626
626
627
627
628
628
629
629
630
630
631
631
632
632
633
633
634
634
635
635
636
636
637
637
638
638
639
639
640
640
641
641
642
642
643
643
644
644
645
645
646
646
647
647
648
648
649
649
650
650
651
651
652
652
653
653
654
654
655
655
656
656
657
657
658
658
659
659
660
660
661
661
662
662
663
663
664
664
665
665
666
666
667
667
668
668
669
669
670
670
671
671
672
672
673
673
674
674
675
675
676
676
677
677
678
678
679
679
680
680
681
681
682
682
683
683
684
684
685
685
686
686
687
687
688
688
689
689
690
690
691
691
692
692
693
693
694
694
695
695
696
696
697
697
698
698
699
699
700
700
701
701
702
702
703
703
704
704
705
705
706
706
707
707
708
708
709
709
710
710
711
711
712
712
713
713
714
714
715
715
716
716
717
717
718
718
719
719
720
720
721
721
722
722
723
723
724
724
725
725
726
726
727
727
728
728
729
729
730
730
731
731
732
732
733
733
734
734
735
735
736
736
737
737
738
738
739
739
740
740
741
741
742
742
743
743
744
744
745
745
746
746
747
747
748
748
749
749
750
750
751
751
752
752
753
753
754
754
755
755
756
756
757
757
758
758
759
759
760
760
761
761
762
762
763
763
764
764
765
765
766
766
767
767
768
768
769
769
770
770
771
771
772
772
773
773
774
774
775
775
776
776
777
777
778
778
779
779
780
780
781
781
782
782
783
783
784
784
785
785
786
786
787
787
788
788
789
789
790
790
791
791
792
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
800
801
801
802
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
810
811
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1470
1471
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1480
1481
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1490
1491
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1500
1501
1501
1502
1502
1503
1503
1504
1504
1505
1505
1506
1506
1507
1507
1508
1508
1509
1509
1510
```

데이터 분석 과정

- 데이터 수집 및 저장

데이터 수집 및 저장

카카오 맵 정보 CSV를 DB에 저장



```
try:
    json_data = json.loads(data)
except :
    print("error load page....")
    break
list = json_data["place"]
if list == None : break
count = 0
for item in list :
    #print(item["address"])
    temp_list = item["address"].split(" ")
    if dong not in temp_list :
        exitLoop = True
        continue
    if keyword=="주차장" :
        if item["last_cate_name"] == "전기자동차 충전소":
            continue
    if keyword=="식당" :
        if item["cate_name_depth2"] == "카페":
            continue
```

```

sql = "insert into place(" + colname + ")"
sql += "values"
sql += "(" + str(item["name"]) + ", '"
sql += str(item["address"]) + ", '"
sql += str(item["new_address"]) + ", '"
sql += doneg + ", '"
sql += str(item["img"]) + ", '"
sql += keyword + ", '"
sql += str(item["last_cate_name"]) + ", '"
sql += str(item["cate_name_depth1"]) + ", '"
sql += str(item["cate_name_depth2"]) + ", '"
sql += str(item["cate_name_depth3"]) + ", '"
sql += str(item["cate_name_depth4"]) + ", '"
sql += str(item["cate_name_depth5"]) + ", '"
sql += str(item["tel1"]) + ", '"
sql += str(item["lat1"]) + ", '"
sql += str(item["lon1"]) + ", '"
# 좌표변환 실행
utmk_x, utmk_y = w8_84_to_utm(item['lon'], item['lat'])
sql += str(utmk_x) + ", "
sql += str(utmk_y) + ")"
#print(sql)

```

```

# csv 파일에 행 단위 데이터 삽입
f = open("D:/LJH/project/data/" + fileName, "a", newline="", encoding='utf-8')
wr = csv.writer(f)
wr.writerow([str(item["name"]), str(item["address"]), str(item["new_address"]), dong, str(item["img1"]), keyword, str(item["last_cate_name"]),
            str(item["cate_name_depth1"]), str(item["cate_name_depth2"]), str(item["cate_name_depth3"]), str(item["cate_name_depth4"]),
            str(item["cate_name_depth5"]), str(item["tel1"]), str(item["lat1"]), str(item["lon1"]), str(utmk_x), str(utmk_y)])
f.close()

try :
    # insert 구문 실행
    db.RunSQL(sql)
except :
    print("DB insert Error")
    db.DBClose()
    break;
if (count == 0 or exitLoop == True) : break
db.CloseQuery()
db.DBClose()
return print("DB Close Success")

```

지도 좌표계 변환

```
#----- 좌표계 변환기
proj_UTMK = Proj(init='epsg:5179') # UTMK
proj_WGS84 = Proj(init='epsg:4326') # WGS84

def ts_w84_to_utmkk(x,y):
    x1,y1 = transform(proj_WGS84, proj_UTMK
                      return x1, y1
```

데이터 분석 과정

- 데이터 수집 및 저장

데이터 수집 및 저장 결과

가공

202206291120_place_group.csv	2022-06-29 오전 11:20	한컴오피스 2018 ...	4KB
202206291128_mM_pca_2.csv	2022-06-29 오전 11:28	한컴오피스 2018 ...	5KB
202206291137_final_data.csv	2022-06-29 오전 11:37	한컴오피스 2018 ...	3KB
군집_donglist.txt	2022-06-29 오후 12:41	텍스트 문서	1KB
군집_donglist_dic.txt	2022-06-29 오후 12:43	텍스트 문서	1KB

미가공

LSMD_ADMIN_SECT_UMD_전북	2022-07-04 오전 10:09	파일 풀더	
202206271626_place.csv	2022-06-27 오후 4:35	한컴오피스 2018 ...	3,997KB
전라북도_전기차 현황_20190620..csv	2022-06-29 오전 10:22	한컴오피스 2018 ...	8KB
전주시_덕진구 법정동코드 조회자료.csv	2022-07-04 오전 10:53	한컴오피스 2018 ...	2KB
전주시_덕진구 법정동코드 조회자료.xls	2022-07-04 오전 10:53	Microsoft Excel 9...	31KB
전주시_완산구 법정동코드 조회자료.csv	2022-07-04 오전 10:54	한컴오피스 2018 ...	3KB
전주시_완산구 법정동코드 조회자료.xls	2022-07-04 오전 10:53	Microsoft Excel 9...	8KB
한국전력공사_지역별 전기차 현황정보_2...	2022-07-01 오후 3:28	한컴오피스 2018 ...	5KB
한국전력공사_지역별 전기차 현황정보1....	2022-06-10 오후 3:15	한컴오피스 2018 ...	5KB
행정안전부_지역별(법정동) 성별 연령별 ...	2022-06-29 오전 9:32	한컴오피스 2018 ...	9,890KB

bno	bname	badl	bdong	bfcate	blat	blon
<code>mysql> select * from building;</code>						
1	노원공영주차장	전북	전주시_완산구	호자동2가	1237-8	호자동2가
2	전주대성교육원 주차장	전북	전주시_완산구	호자동2가	1311-1	호자동2가
3	무지개마을 주차장	전북	전주시_완산구	호자동2가	1158-20	호자동2가
4	한우빌딩 주차장	전북	전주시_완산구	호자동2가	1352	호자동2가
5	전주비전대학교 주차장	전북	전주시_완산구	호자동2가	1070	호자동2가
6	평나무4길 공영주차장	전북	전주시_덕진구	인후동1가	781-7	인후동1가
7	전주아중현대아파트 주차장	전북	전주시_덕진구	인후동1가	858-2	인후동1가
8	아중지구 신립청묘 공영주차장	전북	전주시_덕진구	인후동1가	907-2	인후동1가
9	인후동진배들 주차장	전북	전주시_덕진구	인후동1가	807-6	인후동1가
10	산림조합중앙회 전북지역본부 주차장	전북	전주시_덕진구	인후동1가	791	인후동1가
11	근영여고 앞 공영주차장	전북	전주시_완산구	중화산동2가	651-6	중화산동2가
12	에코리움 주차장	전북	전주시_완산구	중화산동2가	570-1	중화산동2가
13	전주병연 주차장	전북	전주시_완산구	중화산동2가	168	중화산동2가
14	공용주차장	전북	전주시_완산구	중화산동2가	644-3	중화산동2가
15	24시풍천민물장어 주차장	전북	전주시_완산구	중화산동2가	581-5	중화산동2가

lano	goo	dong	people	charger	car	addr	pcar	qcharger
4511110100	전주시_완산구	동성동2가	142	71	0	원산_국제	0	0
4511110300	전주시_완산구	동성동3가	98	28	0	원산_국제	0	0
4511110400	전주시_완산구	동성동4가	80	33	0	원산_국제	0	0
4511110500	전주시_완산구	동성동12가	273	96	1	원산_국제	0	0
4511110600	전주시_완산구	동성동22가	185	74	1	원산_국제	4	0
4511110700	전주시_완산구	동성동32가	133	34	0	원산_국제	0	0
4511110800	전주시_완산구	동성동33가	367	114	0	원산_국제	0	0
4511110900	전주시_완산구	동성동34가	164	48	1	원산_국제	4	0
4511111000	전주시_완산구	동성동35가	103	31	0	원산_국제	0	0
4511111100	전주시_완산구	동성동36가	295	76	0	원산_국제	0	0
4511111200	전주시_완산구	동성동37가	306	244	3	원산_국제	11	0
4511111300	전주시_완산구	동성동38가	248	63	0	원산_국제	0	0
4511111400	전주시_완산구	동성동39가	290	25	0	원산_국제	0	0
4511111500	전주시_완산구	동성동40가	207	7	0	원산_국제	0	0
4511111600	전주시_완산구	동성동41가	171	111	0	원산_국제	0	0
4511111700	전주시_완산구	동성동42가	216	106	1	원산_국제	4	0
4511111800	전주시_완산구	동성동43가	467	340	1	원산_국제	7	1
4511111900	전주시_완산구	동성동44가	912	163	1	원산_국제	33	1
4511112000	전주시_완산구	동성동45가	3206	114	9	원산_국제	7	1
4511112100	전주시_완산구	동성동46가	5237	95	2	원산_국제	15	4

데이터 분석 과정

- 데이터 처리

법정동 추출

```

path = "D://LJH//project//data"           # 경로는 사용자에 맞게 설정
os.chdir(path)                          # 경로를 path로 설정
df1 = pd.read_csv("전주시 완산구 법정동코드 조회자료.csv", encoding="euc-kr")
df2 = pd.read_csv("전주시 덕진구 법정동코드 조회자료.csv", encoding="euc-kr")

def appendDong(df_list, df):
    for i in range(0, len(df)):
        df_list.append(df['법정동명'][i].split(" ")[-1])

dong_list = []
appendDong(dong_list, df1)
appendDong(dong_list, df2)
print(dong_list)

```

**전주시 완산구, 덕진구 법정동을 불러와서
데이터를 활용 할 수 있도록 변경**

결과 :

In [6]: print(dong_list)

```

['완산구', '중앙동1가', '중앙동2가', '중앙동3가', '중앙동4가', '경원동1가', '경원동2가', '경원동3가', '풍남동1가', '풍남동2가', '풍남동3가', '전동', '전동3가', '다가동1가', '다가동2가', '다가동3가', '다가동4가', '고사동', '교동', '태평동', '중노송동', '남노송동', '동완산동', '서완산동1가', '서완산동2가', '동서학동', '서서학동', '중화산동1가', '중화산동2가', '서신동', '석구동', '원당동', '평화동1가', '평화동2가', '평화동3가', '중인동', '용복동', '삼천동1가', '삼천동2가', '삼천동3가', '효자동1가', '효자동2가', '효자동3가', '대성동', '색장동', '상림동', '서노송동', '덕진구', '진북동', '인후동1가', '인후동2가', '덕진동1가', '덕진동2가', '금암동', '팔복동1가', '팔복동2가', '팔복동3가', '산정동', '금상동', '우아동1가', '우아동2가', '우아동3가', '호성동1가', '호성동2가', '호성동3가', '전미동1가', '전미동2가', '송천동1가', '송천동2가', '반월동', '화전동', '용정동', '성덕동', '원동', '고랑동', '여의동', '만성동', '장동', '팔복동4가', '도도동', '강릉동', '도덕동', '남정동', '중동', '여의동2가']

```

주성분 분석 준비

DF 만든 후 DB에 저장

추출한 데이터 합침

```
# 인구수와 전기차 데이터를 left조인으로 합치고 nan은 0으로 설정
df_temp = pd.merge(df, df_jeonju_car, left_index=True, right_index=True, how='left')
df_temp = pd.merge(df_temp, df_jeonju_car, left_index=True, right_index=True, how='left')
df_temp = df_temp.fillna(0)

# 전기차 충전소와 마지막 데이터 위치 전환
col_list = df_temp.columns.to_list()
col_list[-1] = col_list[-1], col_list[col_list.index('전기차 충전소')]

# 최종 Dataframe = df2_temp
df2_temp = df_temp[col_list]

# 결측치가 있는지 확인
df2_temp.isna().sum()

# 최종 Dataframe 출력 확인
df2_temp.head()
```

필요 데이터 전처리

```

#+++++ 전주시 인구수
filename = "행정안전부_지역별(법정동) 성별 연령별 주민등록 인구수_20220531.csv"
df_people = pd.read_csv(filename, encoding="euc-kr")

# 법정동코드를 이용하여 전주시 법정동만 추출
df_jeonju_people_all = df_people[(df_people['법정동코드'] >= 4511110100) & (df_people['법정동코드'] <= 4511313800)]
df_col_list = df_jeonju_people_all.columns # columns 함수를 사용하여 col_list 생성
a = np.where(np.array(df_col_list) == '만19세남자')[0][0] # 만 19세 남자 인덱스 위치
b = np.where(np.array(df_col_list) == '만69세남자')[0][0] # 만 69세 남자 인덱스 위치
c = np.where(np.array(df_col_list) == '만19세여자')[0][0] # 만 19세 여자 인덱스 위치
d = np.where(np.array(df_col_list) == '만69세여자')[0][0] # 만 69세 여자 인덱스 위치
df_dong = df_jeonju_people_all[['읍면동명']] # 읍면동명만 추출
df_male = df_jeonju_people_all.iloc[ :, a:b+1] # 만19~69세 남자만 추출
df_female = df_jeonju_people_all.iloc[ :, c:d+1] # 만19~69세 여자만 추출

# 추출한 데이터들을 열 기준으로 merge //
df_jeonju_people = pd.merge(df_dong, df_male, how="left", left_index=True, right_index=True)
df_jeonju_people = pd.merge(df_jeonju_people, df_female, how="left", left_index=True, right_index=True)
# '읍면동명'을 인덱스로 설정
df_jeonju_people = df_jeonju_people.set_index('읍면동명')
# 행별 합계를 '계'로 설정
df_jeonju_people['계'] = df_jeonju_people.sum(axis=1)
# '계'만 추출하고 '계' -> 'people'로 변경
df_jeonju_people = df_jeonju_people['계'].to_frame()
df_jeonju_people.rename(columns = {'계' : 'people'}, inplace = True)

#+++++ 전주시 전기차
filename = "전라북도_전기차_현황_20190620..csv"
df_car = pd.read_csv(filename, encoding="euc-kr")

# '시.군'에서 '전주시'만 추출
df_jeonju_car = df_car[df_car['시.군'] == '전주시']
# '이하 주소'를 인덱스로 설정
df_jeonju_car = df_jeonju_car.set_index('이하 주소')
# '대수'만 추출하고 '대수' -> 'car'로 변경
df_jeonju_car = df_jeonju_car['대수'].to_frame()
df_jeonju_car.rename(columns = {'대수' : 'car'}, inplace = True)

```

주성분 분석 준비

결과 값

```
...: df2_temp.head()
Out[12]:
   카페 주차장 식당 편의점 대형마트 car 공공기관 영화관 people 전기차 충전소
강흥동      0     0     0     0     0  0.0    2     0   61.0      0
경원동1가   20    8    34     1     0  1.0   10     0  185.0      1
경원동2가   9    2    20     2     0  0.0    1     0  133.0      0
경원동3가   19   5    59     4     0  0.0   26     0  367.0      1
고랑동      1    1     9     4     0  1.0    1     0 1277.0      2
```



csv 생성 후 저장

```
#===== 저장할 csv 생성
current_date = datetime.today().strftime("%Y%m%d%H%M")

fileName = current_date + '_place_group.csv' # 파일 이름

df2_temp.to_csv("D:/LJH/project/data/" + fileName, sep=',', encoding="euc-kr")

# 원본 DataFrame df, 복사본 df_t
df_t = df2_temp.copy()
```



	A	B	C	D	E	F	G	H
1	법정동	people	식당	카페	편의점	공공기관	대형마트	영화관
2	강흥동	61	0	0	0	2	0	0
3	경원동1가	185	34	20	1	10	0	0
4	경원동2가	133	20	9	2	1	0	0
5	경원동3가	367	58	19	4	27	0	0
6	고랑동	1277	9	1	4	1	0	0
7	고사동	467	202	88	8	13	0	6
8	교동	912	94	50	3	3	0	0
9	금상동	582	3	2	0	2	0	0
10	금암동	13386	10	14	45	13	0	0
11	남노송동	1616	21	8	1	24	0	0
12	남정동	173	0	1	0	1	0	0
13	다가동1가	230	11	9	1	3	0	0
14	다가동2가	207	2	2	0	1	0	0
15	다가동3가	177	82	23	2	3	0	0
16	다가동4가	212	64	31	1	8	0	0
17	대성동	1039	15	6	2	7	0	0
18	덕진동1가	5086	293	89	22	44	0	1
19	덕진동2가	10335	186	34	16	32	0	0
20	동서학동	2943	4	11	7	9	0	0
21	동완산동	1177	17	1	0	4	0	0

주성분 분석(데이터 준비 & 처리)

Min - Max Scaler

```
#===== 데이터 스케일링
from sklearn.preprocessing import MinMaxScaler      # MinMaxScaler 사용 (정규화)
from sklearn.preprocessing import StandardScaler     # StandardScaler 사용 (표준화)

# MinMaxScaler
scaler_mm = MinMaxScaler()
df_scaled_mm = scaler_mm.fit_transform(df_t.to_numpy())
df_scaled_mm = pd.DataFrame(df_scaled_mm, columns=df_t.columns, index=df_t.index)
# 데이터 분류 확인을 위해 임의로 총전소 10개 미만 : t1 / 10이상 20개 미만 : t2 / 20개 이상 : t3로 설정
df_scaled_mm.loc[df_t['전기차 충전소'] < 10, 'label'] = 't1'
df_scaled_mm.loc[(df_t['전기차 충전소'] >= 10) & (df_t['전기차 충전소'] < 20), 'label'] = 't2'
df_scaled_mm.loc[df_t['전기차 충전소'] >= 20, 'label'] = 't3'
# 결과 확인
print(df_scaled_mm.head())
In [15]: print(df_scaled_mm.head())
          카페 주차장 식당 편의점 ... 영화관 people 전기차 충전소 label
강릉동  0.000000  0.000000  0.000000 ... 0.0  0.001664  0.000000    t1
경원동1가 0.121212  0.058824  0.078704  0.020833 ... 0.0  0.005047  0.015873    t1
경원동2가  0.054545  0.014706  0.046296  0.041667 ... 0.0  0.003629  0.000000    t1
경원동3가  0.115152  0.036765  0.136574  0.083333 ... 0.0  0.010013  0.015873    t1
고랑동   0.006061  0.007353  0.020833  0.083333 ... 0.0  0.034841  0.031746    t1
[5 rows x 11 columns]
```

Standard Scaler

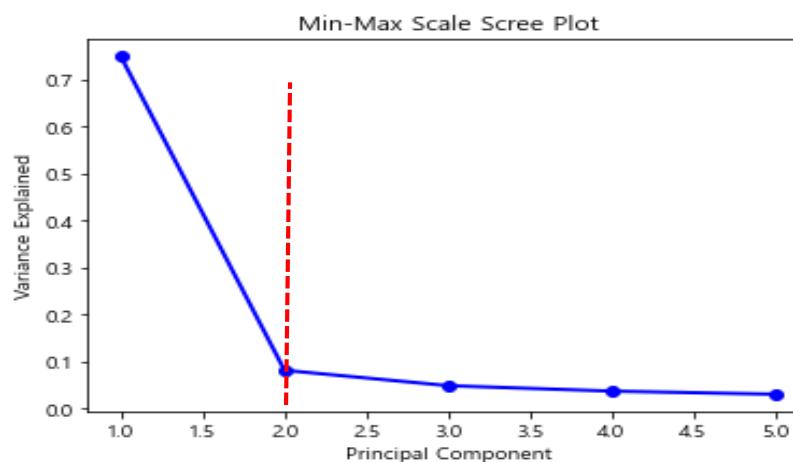
```
# StandardScaler
scaler_st = StandardScaler()
df_scaled_st = scaler_st.fit_transform(df_t.to_numpy())
df_scaled_st = pd.DataFrame(df_scaled_st, columns=df_t.columns, index=df_t.index)
# 데이터 분류 확인을 위해 임의로 총전소 10개 미만 : t1 / 10이상 20개 미만 : t2 / 20개 이상 : t3로 설정
df_scaled_st.loc[df_t['전기차 충전소'] < 10, 'label'] = 't1'
df_scaled_st.loc[(df_t['전기차 충전소'] >= 10) & (df_t['전기차 충전소'] < 20), 'label'] = 't2'
df_scaled_st.loc[df_t['전기차 충전소'] >= 20, 'label'] = 't3'
# 결과 확인
print(df_scaled_st.head())
In [16]: print(df_scaled_st.head())
          카페 주차장 식당 편의점 ... people 전기차 충전소 label
강릉동  -0.774328 -0.525189 -0.752178 ... -0.599942 -0.613785    t1
경원동1가 -0.211559 -0.079741 -0.497975 ... -0.586916 -0.525163    t1
경원동2가 -0.521082 -0.413827 -0.602647 ... -0.592379 -0.613785    t1
경원동3가 -0.239698 -0.246784 -0.311061 ... -0.567798 -0.525163    t1
고랑동   -0.746190 -0.469508 -0.684889 ... -0.472209 -0.436542    t1
[5 rows x 11 columns]
```

데이터 분석 과정

- 데이터 처리

주성분 분석

Scree Plot을 통해 분석에 포함시킬 주성분 개수 확인



```
....: x = df_scaled_mm.iloc[:, :-2].values
....: n_comp = 2 # 주성분을 2개로 설정
....: pca = PCA(n_components=n_comp)
....: pca_fit = pca.fit_transform(x)
....: print(sum(pca.explained_variance_ratio_))
0.8326530017279976
```

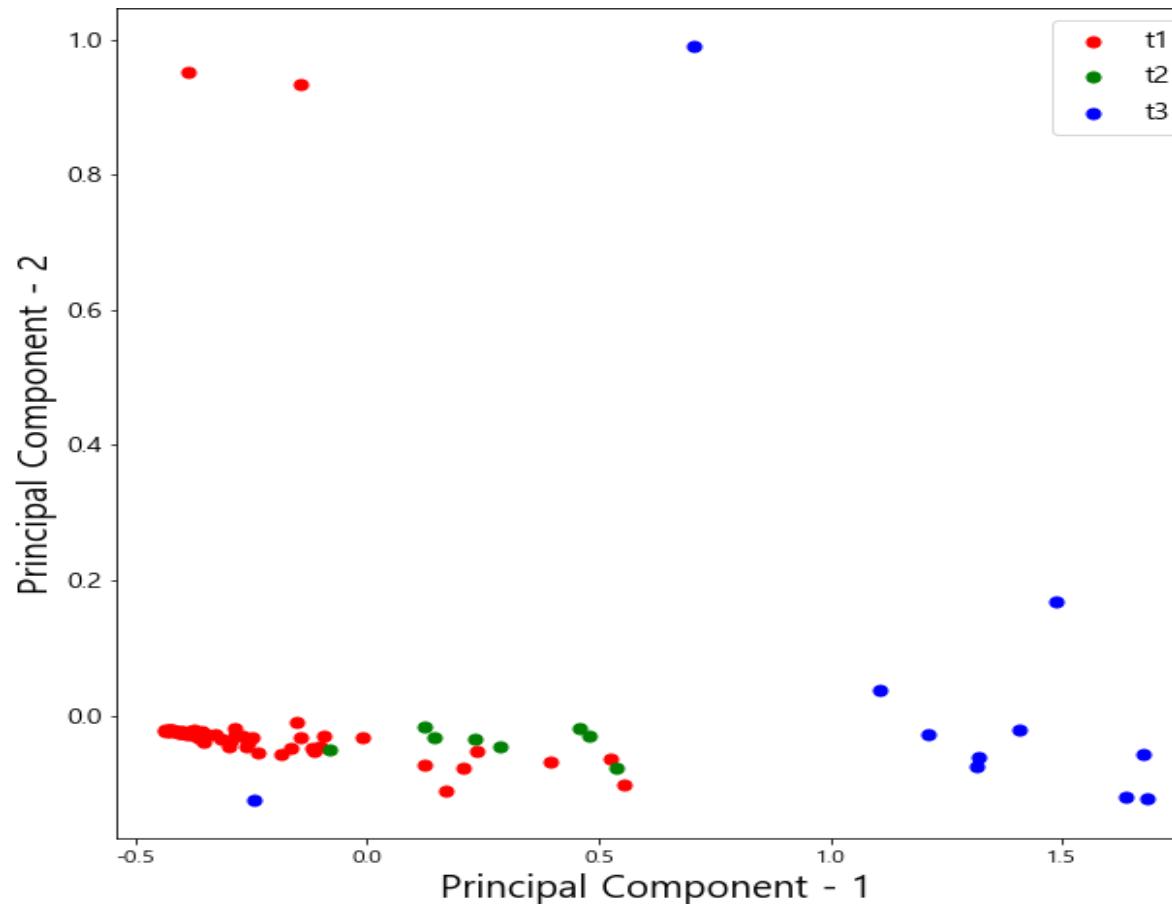
```
=====
PC1 구성
카페          0.339392
주차장        0.092730
식당          0.512461
편의점        0.391139
대형마트      0.026280
car           0.311160
공공기관      0.424589
영화관         0.058634
people         0.423537
dtype: float64
=====

PC2 구성
카페          -0.067713
주차장        -0.105108
식당          0.027989
편의점        -0.010707
대형마트      0.974847
car           -0.052527
공공기관      -0.104788
영화관         0.051925
people         0.129257
dtype: float64
```

제 2 주성분 이후부터 분산의 기울기가 줄어듦
주성분 2개가 적당하다고 판단

주성분 분석 (데이터 표현력 확인)

산점도를 통해 결과로 도출된 주성분이 데이터를 얼마나 잘 표현하는지 확인



T1 : 충전소 개수가 10개 미만

T2 : 충전소 개수가 10개 이상 20개 미만

T3 : 충전소 개수가 20개 이상

주성분 분석 (최종 데이터)

```
#===== 최종 데이터셋 : df_final
# PCA분석 결과 확인하고 영향력이 높은 변수 채택
df_final = df_t.loc[:,('people','식당','카페','편의점','공공기관','대형마트','영화관')]
# 결과 확인
print(df_final.head())
```

```
In [19]: print(df_final.head())
   people  식당  카페  편의점  공공기관  대형마트  영화관
강릉동      61.0     0     0       0       2       0       0
경원동1가    185.0    34    20       1      10       0       0
경원동2가    133.0    20     9       2       1       0       0
경원동3가    367.0    59    19       4      26       0       0
고랑동     1277.0     9     1       4       1       0       0
```

1차 군집에 활용 할 주성분 결과 및 데이터 저장

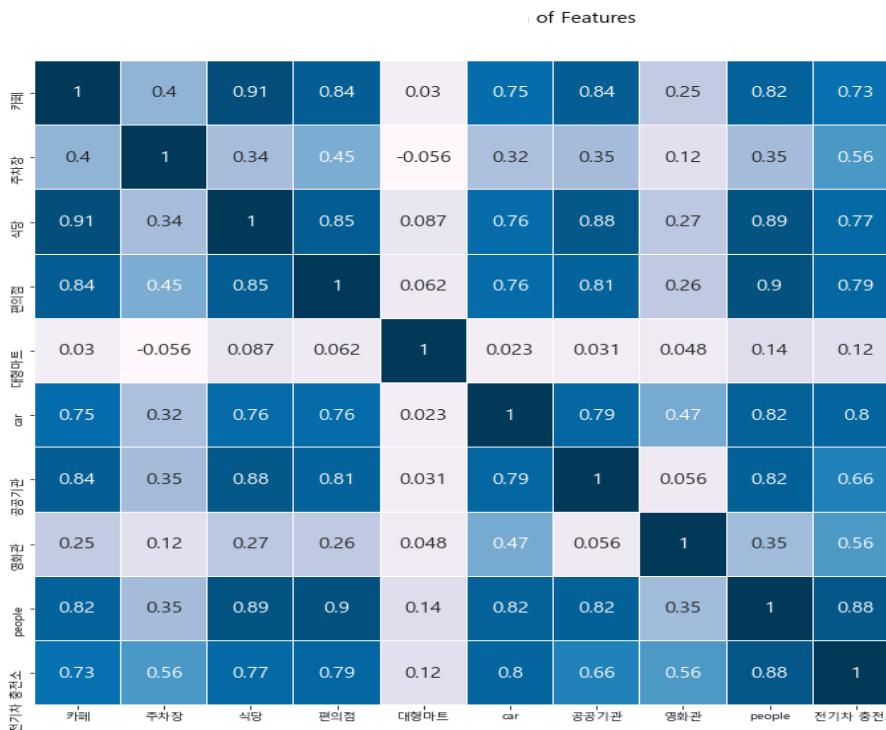
```
#===== 최종 데이터 저장할 csv 생성
current_date = datetime.today().strftime("%Y%m%d%H%M")

fileName = current_date + '_final_data.csv' # 파일 이름

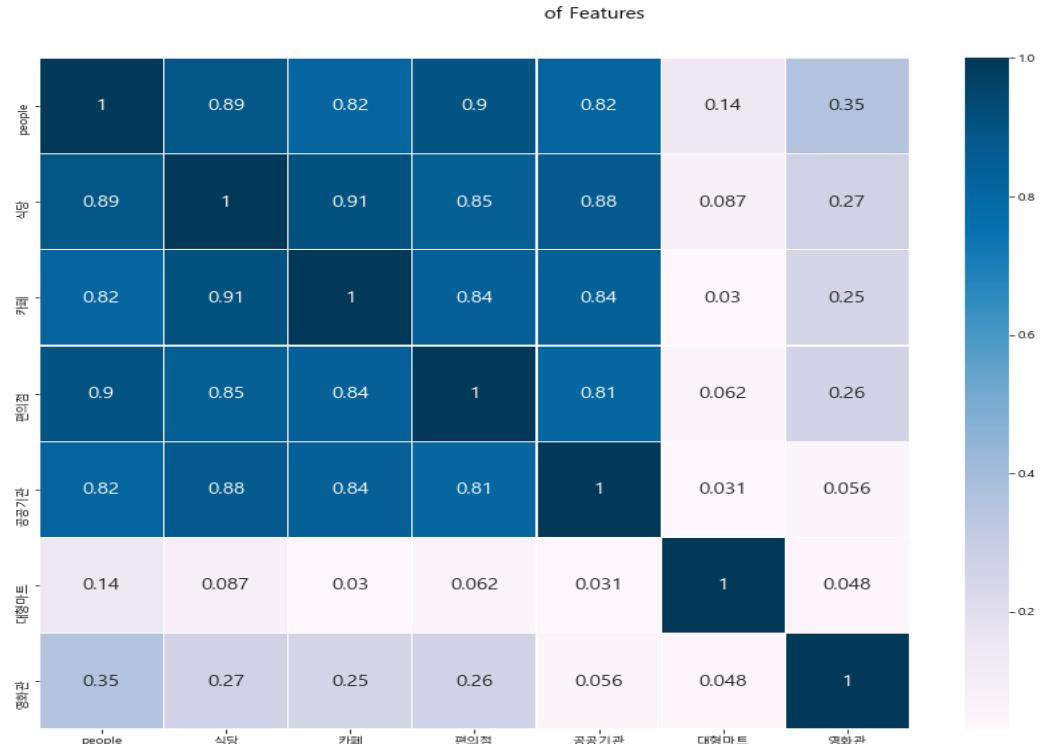
df_final.to_csv("D:/LJH/project/data/"+fileName, sep=',', encoding="euc-kr")
```

상관 분석

모든 변수와의 상관계수



주성분 분석 후 무의미한 변수 제거



- 다중 공선성을 보이는 각 변수들과의 관계를 주성분 분석을 통해 해결 하였음.

입지분석 처리

위치

```
# 파일이 있는 경로로 path 설정
path = "D:/瓢/JH/seoul_data/data"
os.chdir(path)

# 읽어들 길게 가져오기
emd = geopandas.read_file('LSMD ADM_SECT_LMD_45.shp', encoding = 'ANSI')
# 전주 데이터만 사용
emd = emd[emd['COL ADM SE'] == '45110']
emd.head()
```

EMD_CD	EMD_NM	SGG_OID	COL ADM SE	GID	geometry
293	다가동3가	10241	45110	929	POLYGON ((967577.773 1757942.305, 967598.481 1...
294	중인동	10239	45110	930	POLYGON ((964716.960 1754453.411, 964735.574 1...
295	경원동2가	10238	45110	931	POLYGON ((968422.323 1757914.224, 968421.869 1...
326	상립동	15988	45110	875	POLYGON ((960920.693 1756733.714, 960927.885 1...
327	삼천동3가	15987	45110	876	POLYGON ((962437.386 1757056.410, 962446.083 1...

Client

```
jj_restaurant = pd.read_csv('place.csv', encoding='utf-8')
jj_restaurant = jj_restaurant[(jj_restaurant['pfcate'] == '식당') |
                               (jj_restaurant['pfcate'] == '편의점') | (jj_restaurant['pfcate'] == '카페')]

jj_restaurant = jj_restaurant[['pname', 'pdong', 'pfcate', 'plcate', 'utmk_x', 'utmk_y']]
jj_restaurant.columns = ['pname', 'pdong', 'pfcate', 'plcate', 'utmk_x', 'utmk_y']

jj_restaurant = geopandas.GeoDataFrame(jj_restaurant, geometry=geopandas.points_from_xy(jj_restaurant['utmk_x'], jj_restaurant['utmk_y']))
jj_restaurant.set_crs(epsg = 5179, inplace = True)
jj_restaurant.head(2)
```

	pname	pdong	pfcate	plcate	utmk_x	utmk_y	geometry
0	필드오	진북동	카페	디저트카페	966678.137038	1.759592e+06	POINT (966678.137 1759591.504)
1	페더럴	진북동	카페	카페	967309.888649	1.759511e+06	POINT (967309.889 1759511.105)

Facility

```
jj_facility = pd.read_csv('place.csv', encoding='utf-8')
jj_facility = jj_facility[(jj_facility['pfcate'] == '주차장')]

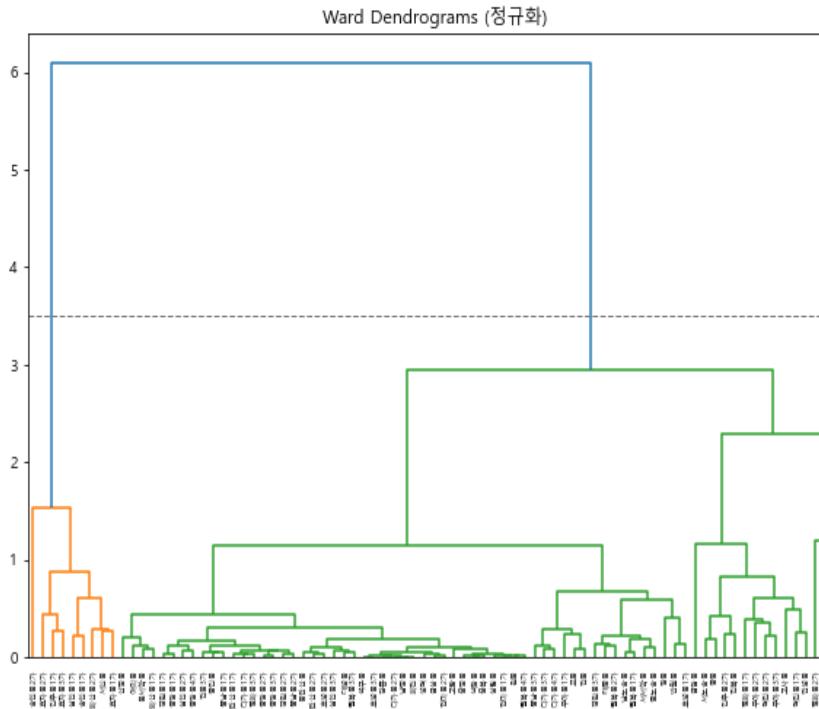
jj_facility = jj_facility[['pname', 'padd', 'pdong', 'pfcate', 'plat', 'plon', 'utmk_x', 'utmk_y']]
jj_facility.columns = ['pname', 'padd', 'pdong', 'pfcate', 'plat', 'plon', 'utmk_x', 'utmk_y']

jj_facility = geopandas.GeoDataFrame(jj_facility, geometry=geopandas.points_from_xy(jj_facility['utmk_x'], jj_facility['utmk_y']))
jj_facility.set_crs(epsg = 5179, inplace = True)
jj_facility.head(2)
```

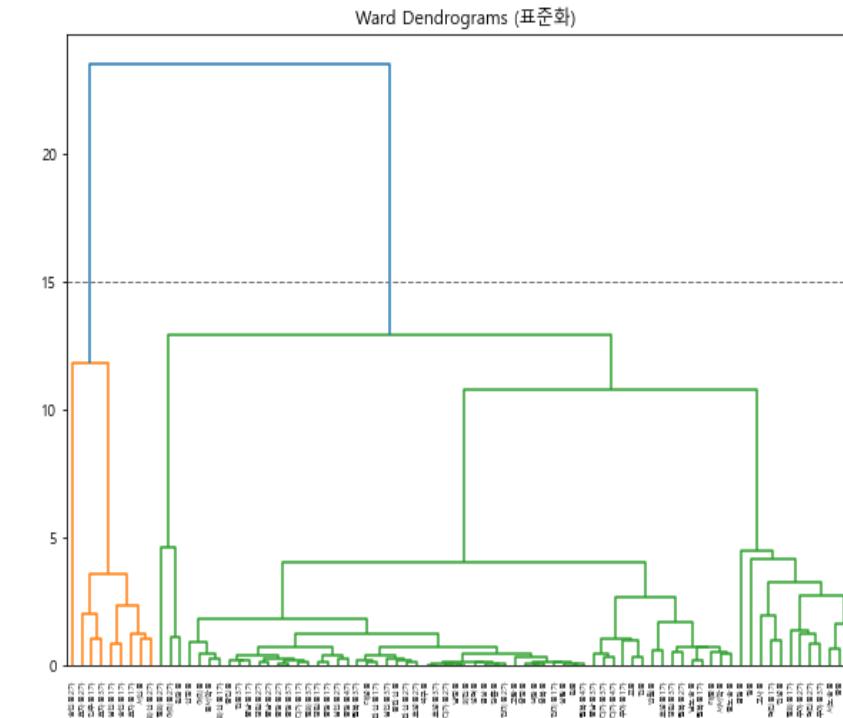
	pname	padd	pdong	pfcate	plat	plon	utmk_x	utmk_y	geometry
48	이정형외과의원주차장	전북	전주시	덕진구 진북동 321-7	진북동	주차장	35.829476	127.143258	967778.525143 1.759279e+06
49	덕진구청 주차장1	전북	전주시	덕진구 진북동 416-12	진북동	주차장	35.829118	127.134262	966965.843738 1.759242e+06

Hierarchical Clustering

Min - Max Scaler



Standard Scaler

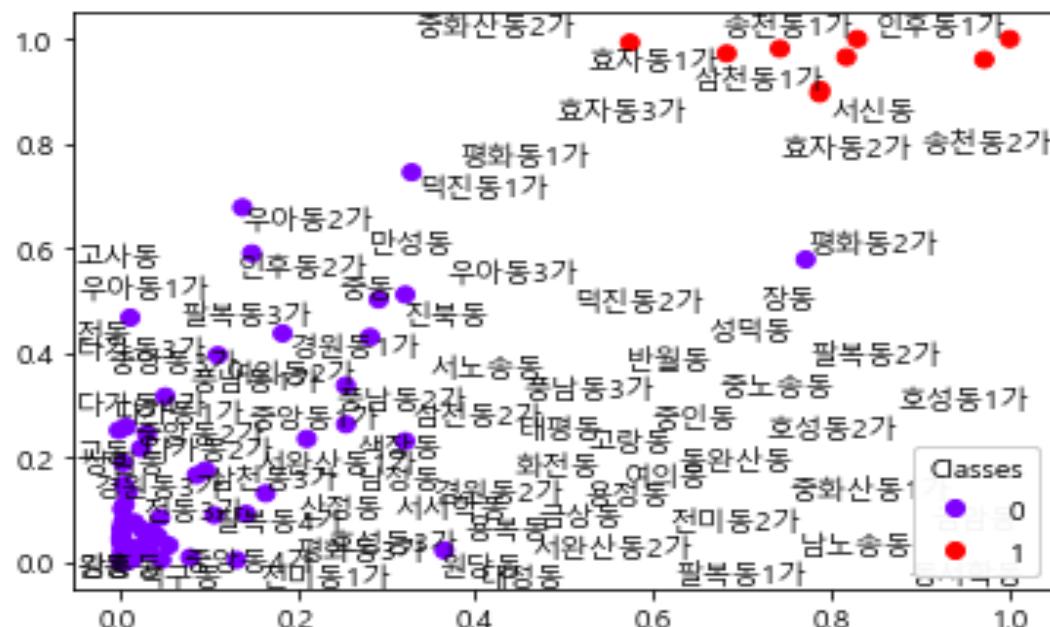


- 군집간의 거리가 먼 Ward 연결법을 선택
- 군집 개수 2개

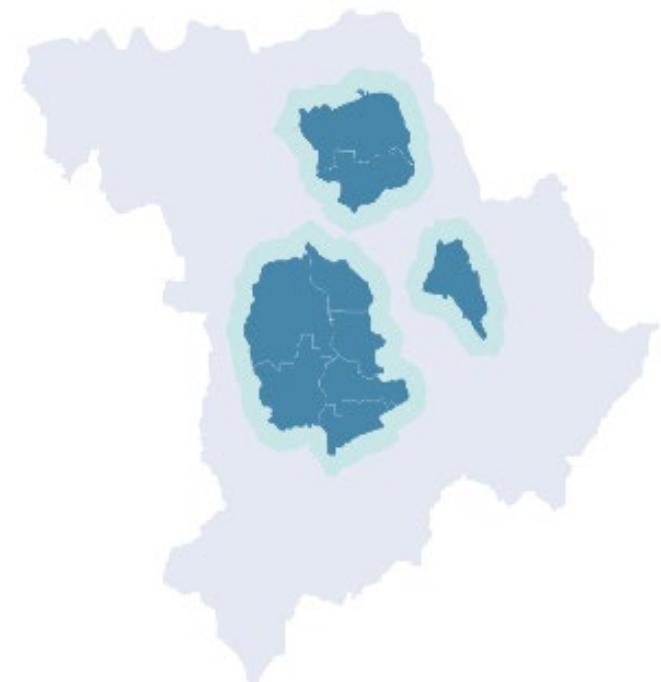
데이터 분석 과정

- 분석 (1차 군집)

Hierarchical Clustering

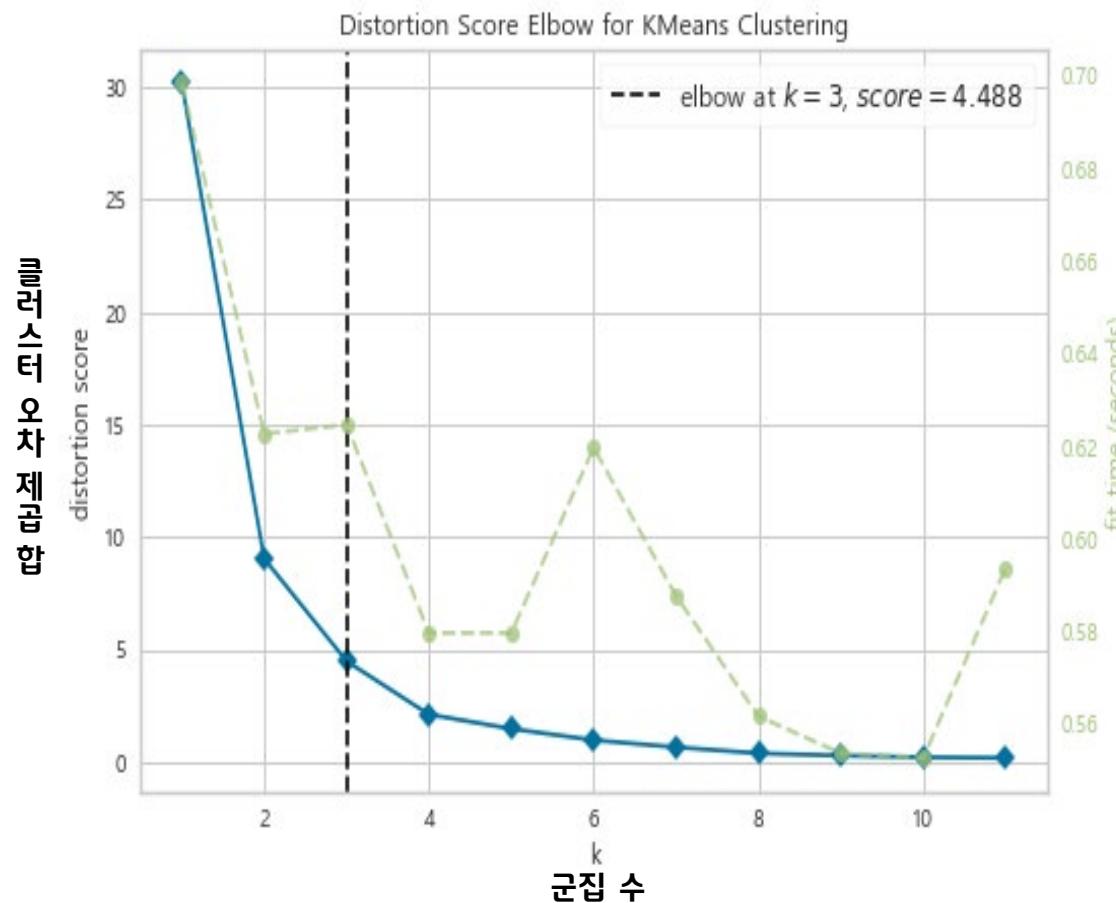


결과



```
.... df_simple = df.loc[cluster.labels_ == 1]
.... labels_s = df_simple.index
.... print(labels_s)
Index(['삼천동1가', '서신동', '송천동1가', '송천동2가', '인후동1가', '중화산동2가', '효자동1가', '효자동2가',
       '효자동3가'],
      dtype='object')
```

Elbow, Silhouette method

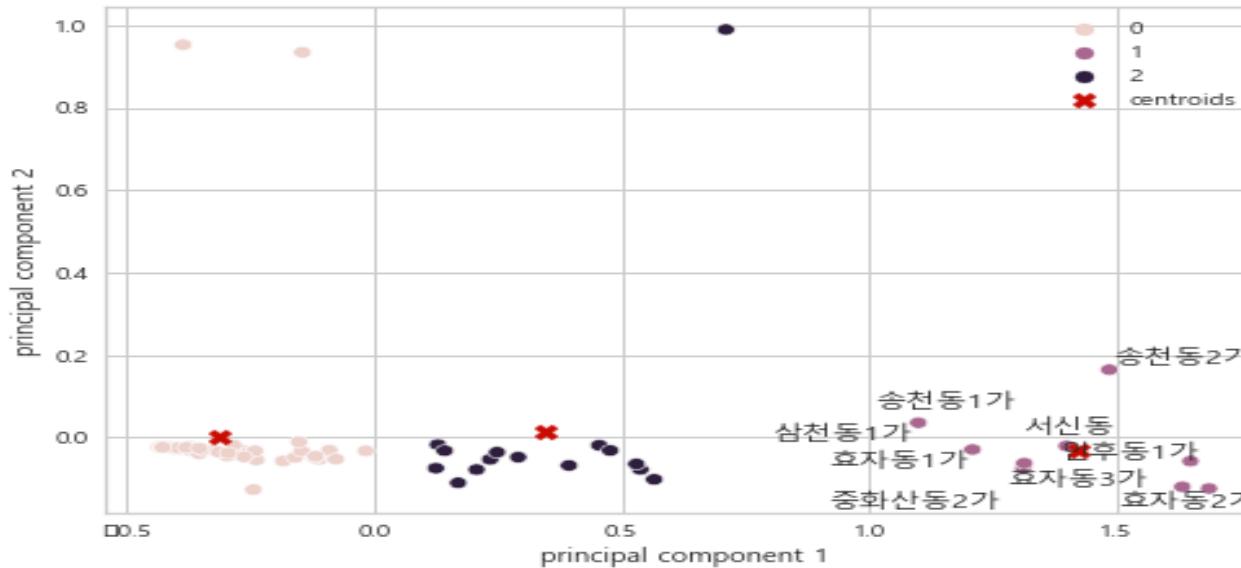


**Elbow, Silhouette method 결과 값에 근거하여
K - means, K-medoids, GMM 분석을 하기 위해
최적의 군집 개수를 분석
결과 : 군집 수는 3개가 적당하다고 판단**

데이터 분석 과정

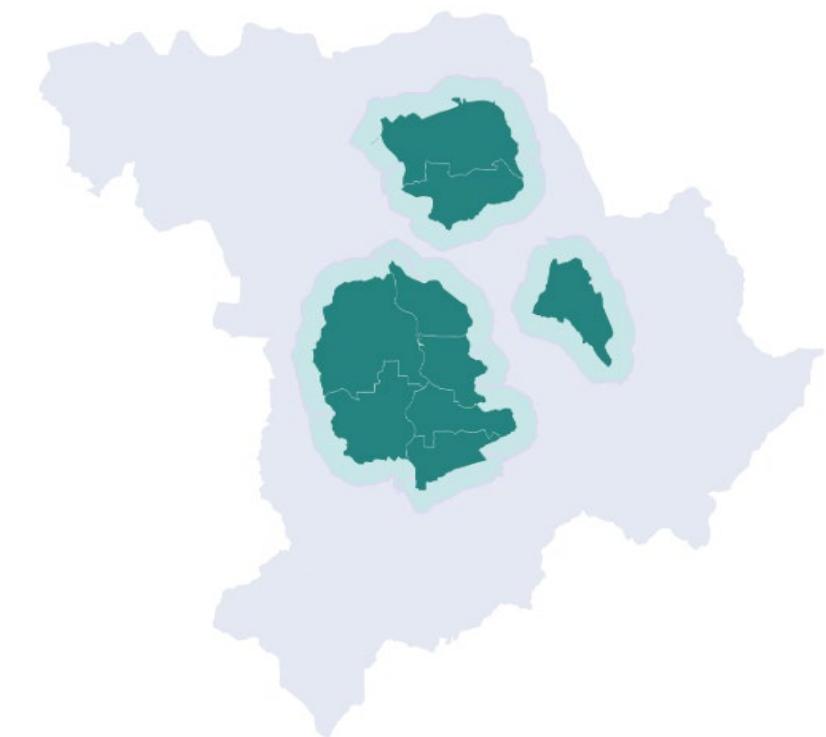
- 분석 (1차 군집)

K-Means



```
...: df_simple = df.loc[kmeans.labels_ == 1]
...: labels_s = df_simple.index
...: print(labels_s)
Index(['삼천동1가', '서신동', '송천동1가', '송천동2가', '인후동1가', '중화산동2가', '효자동1가', '효자동2가',
       '효자동3가'],
      dtype='object')
```

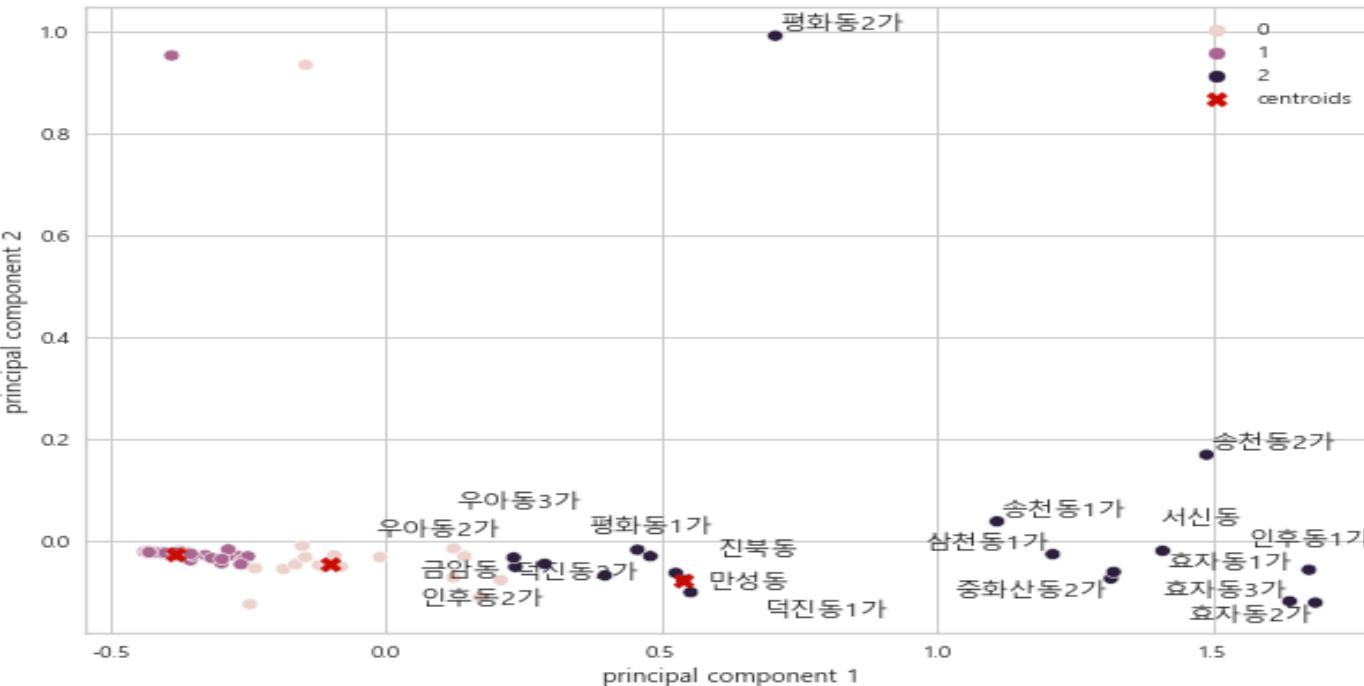
결과



데이터 분석 과정

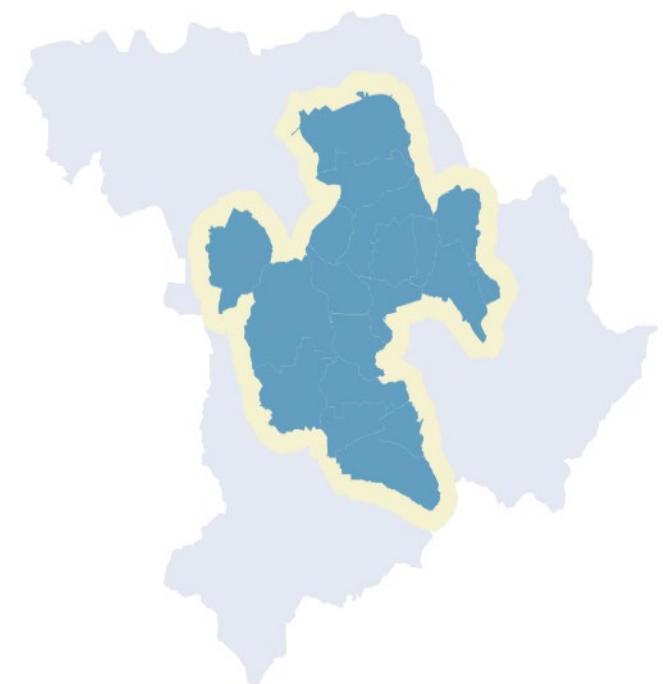
- 분석 (1차 군집)

K-Medoids

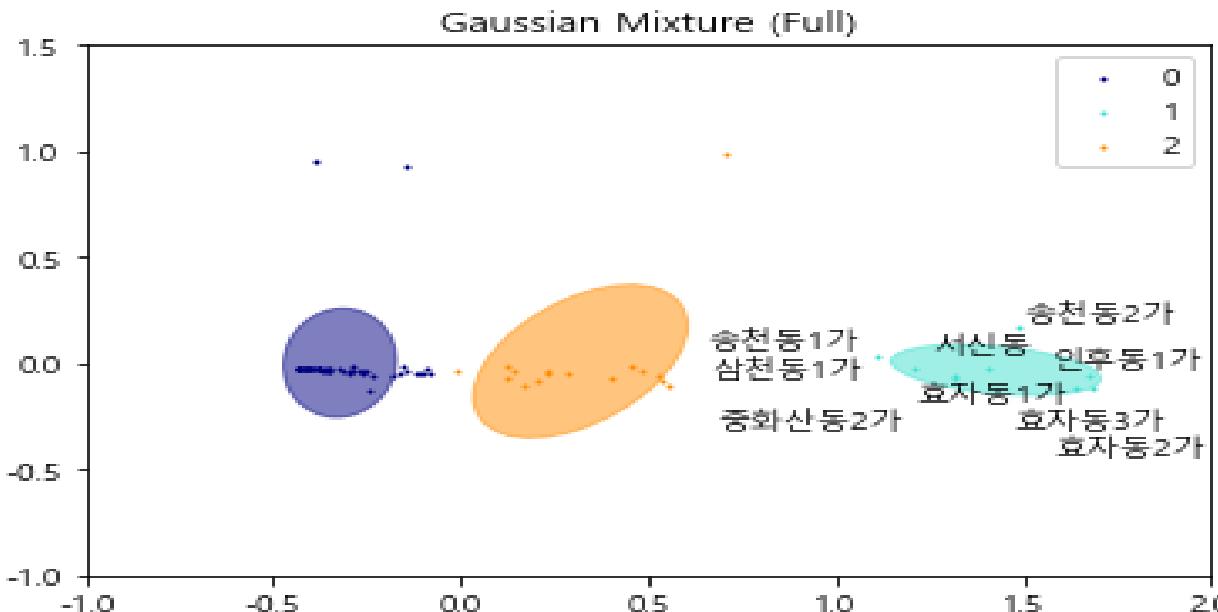


```
...: df_simple = df.loc[kmedoid.labels_ == 2]
...: labels_s = df_simple.index
...: print(labels_s)
Index(['금암동', '덕진동1가', '덕진동2가', '만성동', '삼천동1가', '서신동', '송천동1가', '송천동2가',
       '우아동2가', '우아동3가', '인후동1가', '인후동2가', '중화산동2가', '진북동', '평화동1가', '평화동2가',
       '효자동1가', '효자동2가', '효자동3가'],
      dtype='object')
```

결과



Gaussian Mixture Model: GMM

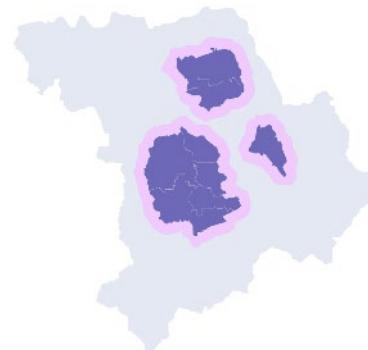


```
...: df_simple = df.loc[gmm_full.predict(data) == 1]
...: labels_s = df_simple.index
...: print(labels_s)
Index(['삼천동1가', '서신동', '송천동1가', '송천동2가', '인후동1가', '중화산동2가', '효자동1가', '효자동2가',
       '효자동3가'],
      dtype='object')
```

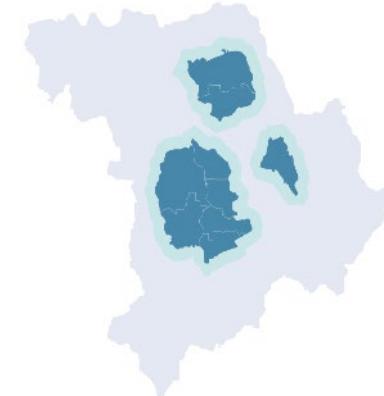
데이터 분석 과정

- 분석 (1차 군집)

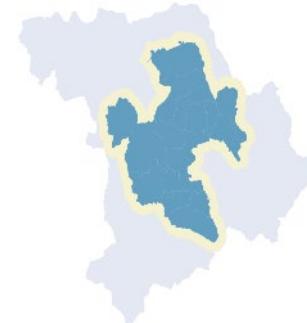
법정동 군집 분석 결과



Gaussian Mixture Model



Hierarchical Clustering



K-means Clustering



K-medoids Clustering

데이터 분석 과정

- 분석 (1차 군집)

법정동 군집 분석 결과

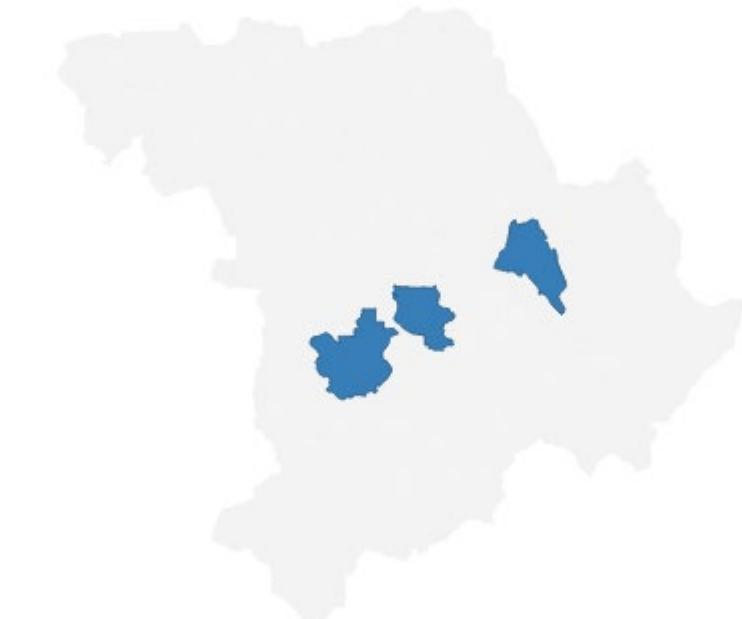
군집 결과가 많이 나온 법정동 대상

군집 분석 결과

9개의 지역(삼천동1가, 서신동, 송천동1,2가 인후동 1가,
증화산동 2가 효자동1,2,3가)

PCA 가중치가 높았던
식당, 카페, 공공기관 3개를 기준

인후동 1가, 증화산동 2가, 효자동 2가
우선 입지 대상 법정동 선정



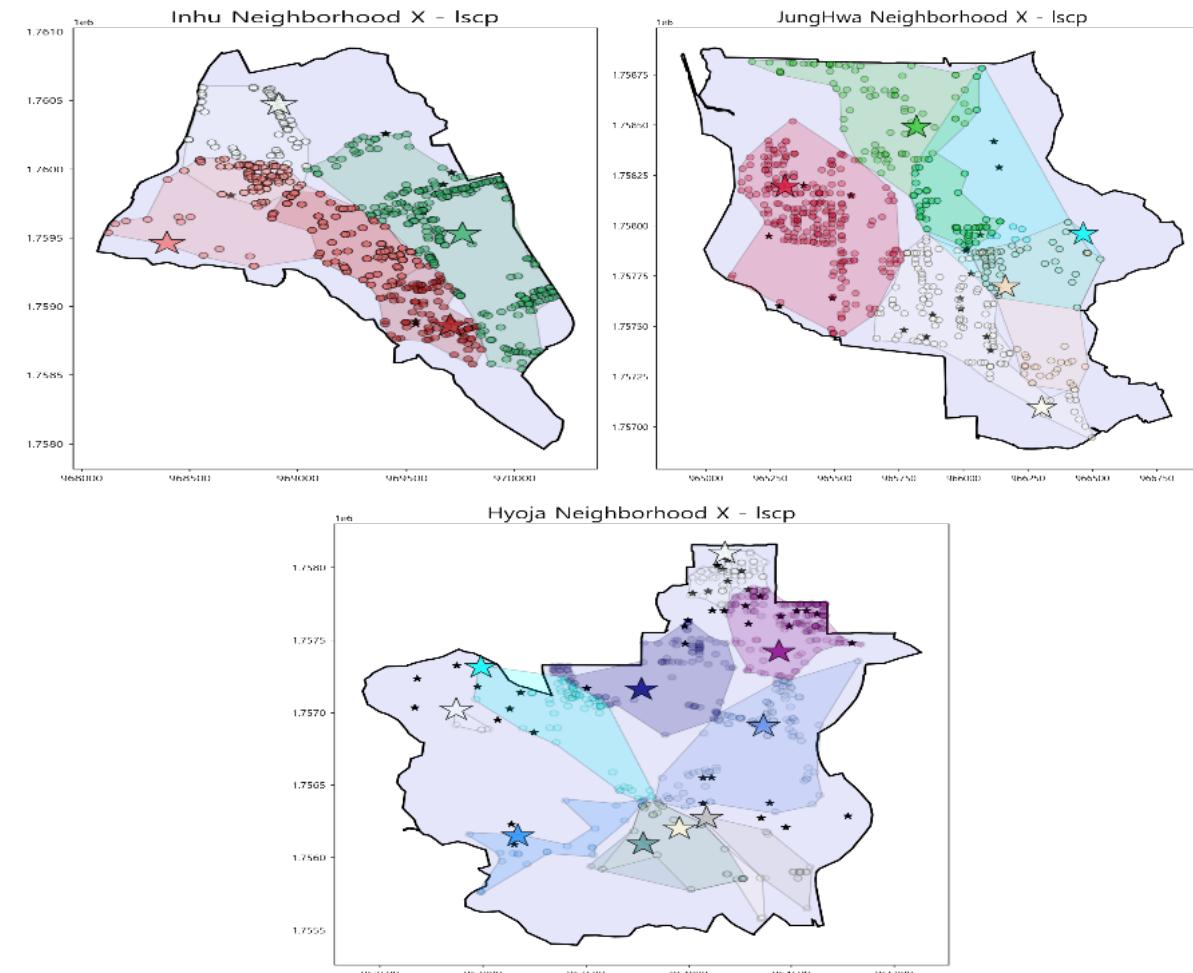
데이터 분석 과정

- 분석 (2차 군집)

LSCP (Location Set Covering Problem)

$$\begin{aligned}
 & \text{Minimize} \sum_{j \in J} x_j \\
 & \text{s.t.} \sum_{j \in N_i} x_j \geq 1 \quad \text{for all } i \in I \\
 & \quad x_j \in \{0, 1\} \quad \text{for all } j \in J
 \end{aligned}$$

문자	의미
i	수요 포인트 index
j	설비지역 포인트 index
I	수요 포인트 집합
J	설비지역 포인트 집합
x	설비 후보 지역 중 위치 j에 설치되면 1, 아니면 0
y	적어도 하나의 설비로 그 포인트가 커버되면 1, 아니면 0



- 지역 수요를 최대한 만족시킬 수 있는 최적 입지 선정
- 법정동 내 식당, 편의점, 카페 등을 수요 포인트로 주차장을 설비 후보 지역으로 선정하여 분석

데이터 분석 과정

- 분석 (2차 군집)

PMP (P-Median Problem)

Inputs:

h_i = 수요지 i 의 수요량

d_{ij} = 수요지 i 와 시설물의 입지점 j 의 거리

p = 시설물의 수

Decision variables:

x_j = 1, 만약 노드 j 에 시설물이 설치되면,
0, 그렇지 않으면.

y_{ij} = 1, 만약 노드 j 에 시설물이 노드 i 의
총수요를 충족시키면,
0, 그렇지 않으면.

$$\text{Subject to } \min \sum_i \sum_j h_i d_{ij} y_{ij} \quad (1-1)$$

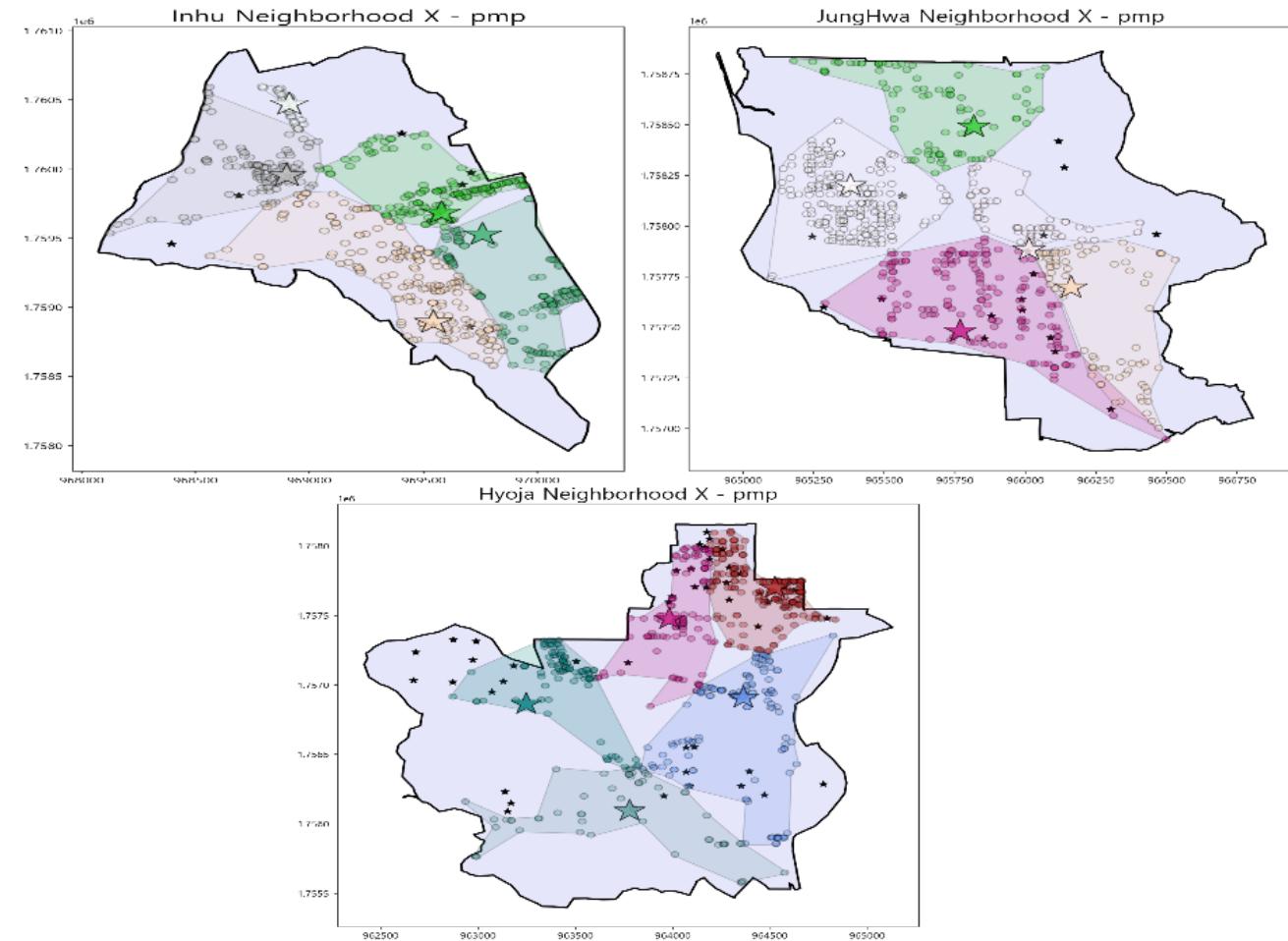
$$\sum_j y_{ij} = 1 \quad (\text{for all } i) \quad (1-2)$$

$$\sum_j x_j = p \quad (1-3)$$

$$y_{ij} \leq x_j \quad (\text{for all } i, j) \quad (1-4)$$

$$y_{ij} \in \{0, 1\} \quad (\text{for all } i, j) \quad (1-5)$$

$$x_j \in \{0, 1\} \quad (\text{for all } j) \quad (1-6)$$



- 수요지로부터 5개의 서비스 후보 지역간의 총 거리가 가장 짧게 있는 입지 선정
- 법정동 내 식당, 편의점, 카페 등을 수요 포인트로 주차장을 서비스 후보 지역으로 선정하여 분석

데이터 분석 과정

- 분석 (2차 군집)

PCP (P-Center Problem)

Inputs:

h_i = 수요지 i 의 수요량

d_{ij} = 수요지 i 와 시설물의 입지점 j 의 거리

p = 시설물의 수

Decision variables:

$x_j = \begin{cases} 1, & \text{만약 노드 } j \text{에 시설물이 설치되면,} \\ 0, & \text{그렇지 않으면.} \end{cases}$

$$\min W$$

s.t.

$$\sum_{j \in J} y_{ij} = 1, \quad i \in I$$

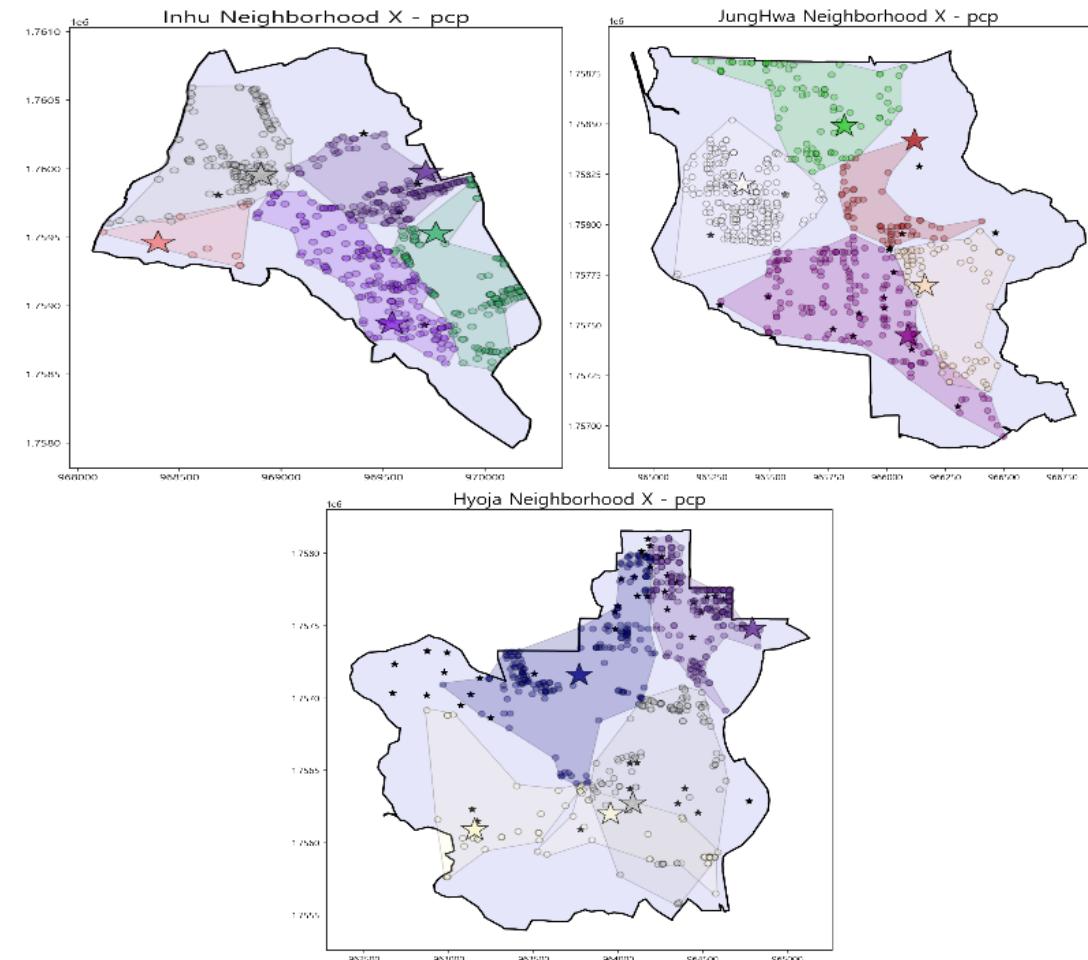
$$\sum_{j \in J} x_j = p,$$

$$y_{ij} \leq x_j, \quad i \in I, j \in J$$

$$\sum_{j \in J} d_{ij} y_{ij} \leq W, \quad i \in I$$

$$x_j \in \{0, 1\}, \quad j \in J$$

$$y_{ij} \in \{0, 1\}, \quad i \in I, j \in J$$



- 수요지로부터 5개의 서비스 후보 지역에 가장 먼 거리의 최소값을 찾는 입지 선정
- 법정동 내 식당, 편의점, 카페 등을 수요 포인트로 주차장을 서비스 후보 지역으로 선정하여 분석

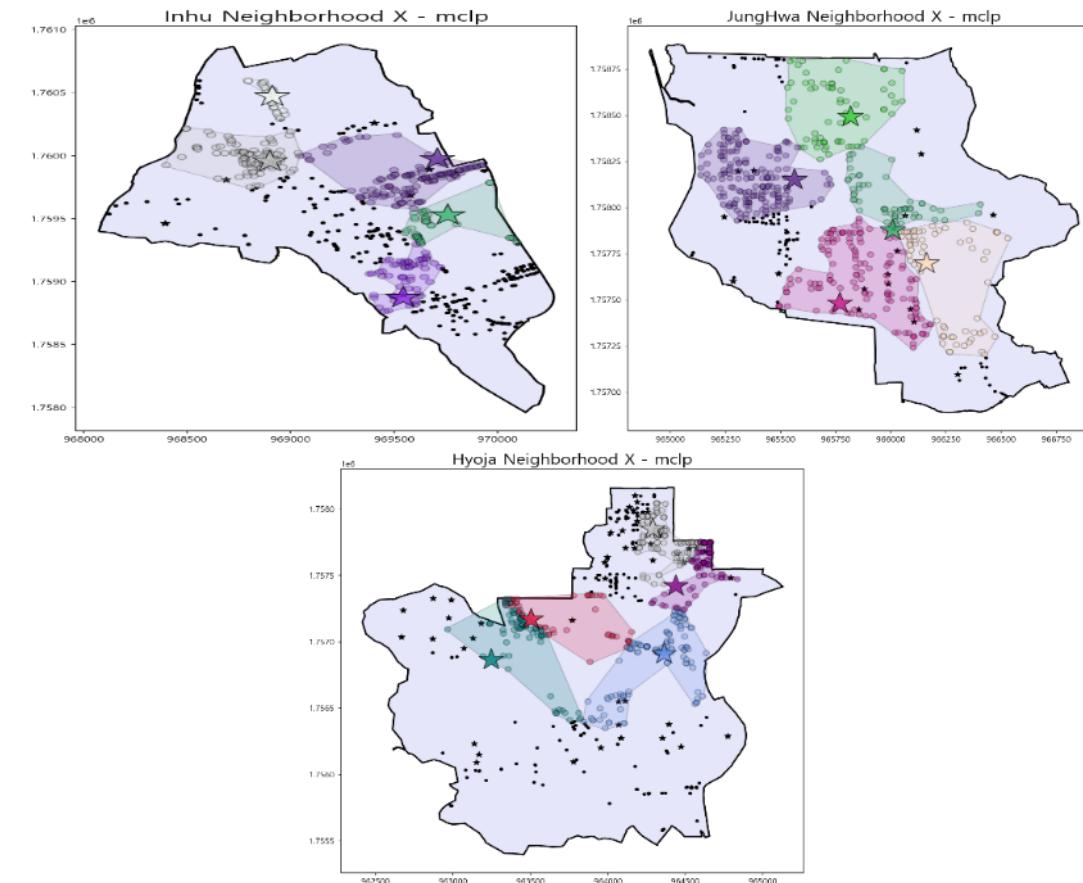
데이터 분석 과정

- 분석 (2차 군집)

MCLP (Maximal Covering Location Problem)

$$\begin{aligned}
 & \text{Maximize} \sum_{i \in I} w_i y_i \\
 \text{s.t. } & y_i \leq \sum_{j \in N_i} x_j \quad \text{for all } i \in I \\
 & \sum_{j \in J} x_j = K \\
 & x_j, y_i \in 0, 1 \quad \text{for all } i \in I, j \in J
 \end{aligned}$$

문자	의미
i	수요 포인트 index
j	설비지역 포인트 index
I	수요 포인트 집합
J	설비지역 포인트 집합
K	설치해야하는 설비 개수
x	설비 후보 지역 중 위치 j에 설치되면 1, 아니면 0
y	적어도 하나의 설비로 그 포인트가 커버되면 1, 아니면 0
w	입지 선정 지수=가중치



- 반경 300m 기준을 잡고 5개의 설비 후보 지역에서 최대 수요를 만족시킬 수 있는 최적 입지 선정
- 법정동 내 식당, 편의점, 카페 등을 수요 포인트로 주차장을 설비 후보 지역으로 선정하여 분석

데이터 분석 과정

- 분석 (2차 군집)

모델 선택

수요지-입지 거리

Min : 최소치

Max : 최대치

Means : 평균

Stds : 분산

1. 인후동1가

2. 중화산동2가

3. 효자동2가

	stats	lscp	pmp	pcp	mclp
0	abs_min	0.000000	0.000000	0.000000	0.000000
1	abs_max	1246.686038	1299.523600	1219.347691	297.573683
2	mean_means	747.969143	223.690933	344.327813	69.334661
3	mean_stds	330.760099	192.637412	258.391669	56.873628

	stats	lscp	pmp	pcp	mclp
0	abs_min	0.000000	0.000000	0.000000	0.000000
1	abs_max	1197.146898	1225.255738	1151.016663	299.935282
2	mean_means	500.117026	181.158290	180.539265	115.479364
3	mean_stds	323.568070	184.301892	171.397666	71.210361

	stats	lscp	pmp	pcp	mclp
0	abs_min	0.000000	0.000000	0.000000	0.000000
1	abs_max	588.813099	1688.345740	1244.739469	295.641173
2	mean_means	200.411755	363.694810	642.582413	105.692156
3	mean_stds	126.617273	356.620215	325.503118	87.335686

- 분석 후 각 모델을 평가한 결과 MCLP모델이 다른 모델들에 비해 최적의 값이 나왔음

- 또한 제한된 조건에서 입지를 선정하는 MCLP모델이 충전소 입지 선정에 더 적합하다고 판단함

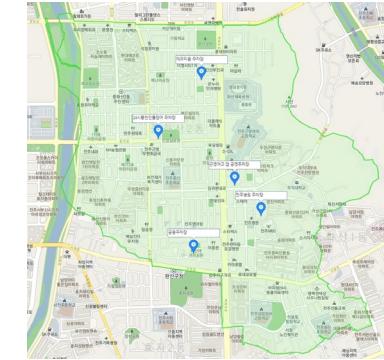
데이터 분석 과정

- 분석 (2차 군집)

2차 군집 분석 결과

<인후동1가>

	pname	padd	pdong	pfcate	plat	plon
402	팽나무4길 공영주차장	전북 전주시	덕진구	인후동1가 781-7	인후동1가	주차장 35.835600 127.155625
403	전주아중현대아파트 주차장	전북 전주시	덕진구	인후동1가 858-2	인후동1가	주차장 35.831795 127.165153
404	아중지구 산림청옆 공영주차장	전북 전주시	덕진구	인후동1가 907-2	인후동1가	주차장 35.825882 127.162799
405	인후3동진버들 주차장	전북 전주시	덕진구	인후동1가 807-6	인후동1가	주차장 35.835826 127.164604
406	주차장	전북 전주시	덕진구	인후동1가 791	인후동1가	주차장 35.840265 127.155736



<중화산동2가>

	pname	padd	pdong	pfcate	plat	plon
8651	근영여고 앞 공영주차장	전북 전주시	완산구	중화산동2가 651-6	중화산동2가	주차장 35.816772 127.123729
8658	주차장	전북 전주시	완산구	중화산동2가 570-1	중화산동2가	주차장 35.822315 127.121573
8659	전주병원 주차장	전북 전주시	완산구	중화산동2가 166	중화산동2가	주차장 35.815159 127.125430
8661	공용주차장	전북 전주시	완산구	중화산동2가 644-3	중화산동2가	주차장 35.813209 127.121072
8663	주차장	전북 전주시	완산구	중화산동2가 591-5	중화산동2가	주차장 35.819228 127.118780

<인후동1가>



<중화산동2가>

	pname	padd	pdong	pfcate	plat	plon
12663	노외공영주차장	전북 전주시	완산구	효자동2가 1237-8	효자동2가	주차장 35.816469 127.104692
12677	전주대평생교육원 주차장	전북 전주시	완산구	효자동2가 1311-1	효자동2가	주차장 35.810304 127.096009
12682	주차장	전북 전주시	완산구	효자동2가 1158-20	효자동2가	주차장 35.812614 127.106333
12688	현우빌딩 주차장	전북 전주시	완산구	효자동2가 1352	효자동2가	주차장 35.808004 127.105539
12709	전주비전대학교 주차장6	전북 전주시	완산구	효자동2가 1070	효자동2가	주차장 35.807542 127.093195

<효자동2가>

- MCLP 분석결과 전기차 충전소 최적 입지 데이터 선정
- 선정된 데이터를 DB에 저장하여 웹을 통해 시각화

Part 5,

시각화(화면 구현)

1. 프론트 페이지

2. 메인 페이지

3. 젠슬라 맵



워드 클라우드

크롤링 crawling				
	논리 이름*	클리 미ドル*	도메인 데이터 타입	널 허용
☞ 기사번호	cr_no	N/A	INTEGER	N-N
● 기사제목	cr_word	N/A	VARCHAR(256)	NULL
● 기사주소	wordcount	N/A	INTEGER	NULL

```
89     lines = text.split(/[\.,\!]+/g),
90     data = lines.reduce((arr, word) => {
91         let obj = Highcharts.find(arr, obj => obj.name === word);
92         if (obj)
93         {
94             obj.weight += 3000 ;
95         } else
96         {
97             obj =
98             {
99                 name: word,
100                weight: 150000
101            };
102            arr.push(obj);
103        }
104        return arr;
105    }, []);
106
107 Highcharts.chart('container2', {
108     accessibility: {
109         screenReaderSection: {
110             beforeChartFormat: '<h5>{chartTitle}</h5>' +
111                 '<div>{chartSubtitle}</div>' +
112                 '<div>{chartLongdesc}</div>' +
113                 '<div>{viewTableButton}</div>'
114         },
115     },
116     series: [
117         {
118             type: 'wordcloud',
119             data,
120             name: 'Occurrences'
121         },
122         title: {
```

친환경 자동차 뉴스기사 트렌드



시각화(화면 구현)

- 메인 페이지

입지 추천

최종결과 building					
필수	필수	필수	선택	선택	선택
장소 번호	bno	N/A	INTEGER	N-N	
장소 이름	bname	N/A	VARCHAR(60)	NULL	
장소 주소	baddr	N/A	VARCHAR(60)	NULL	
동 이름	bdong	N/A	VARCHAR(30)	NULL	
카테고리 대분류	bfcate	N/A	VARCHAR(30)	NULL	
위도	blat	N/A	VARCHAR(30)	NULL	
경도	blon	N/A	VARCHAR(30)	NULL	



```

main.jsp zenslamap.jsp PlaceDTO.java
16
17  public boolean GetBuilding(String dong)
18  {
19      String sql = "select bno,bname,baddr,bfcate,blat,blon from Building where bdong = '"+dong+"'";
20      if(this.DBOpen() == false) return false;
21      this.OpenQuery(sql);
22      while(this.GetNext() == true)
23      {
24          String bno   = this.GetValue("bno");
25          String bname = this.GetValue("bname");
26          String baddr = this.GetValue("baddr");
27          String bfcate = this.GetValue("bfcate");
28          String blat   = this.GetValue("blat");
29          String blon   = this.GetValue("blon");
30
31          if( this.buildinglist == null )
32          {
33              this.buildinglist = new ArrayList<PlaceVO>();
34          }
35          PlaceVO vo = new PlaceVO();
36          vo.setBno(bno);
37          vo.setBname(bname);
38          vo.setBaddr(baddr);
39          vo.setBfcate(bfcate);
40          vo.setBlat(blat);
41          vo.setBlon(blon);
42          buildinglist.add(vo);
43      }
44      this.CloseQuery();
45      this.DBClose();
46      return true;
47  }
48
49 // 리스트의 크기 구하기
..
```

최적 입지 선정

1지역		2지역		3지역	
번호	이름	번호	이름	번호	이름
1	팡나무4길 공영주차장				전북 전주시 덕진구 인후동1가 781-7
2	전주아중현대아파트 주차장				전북 전주시 덕진구 인후동1가 858-2
3	아중지구 산림청옆 공영주차장				전북 전주시 덕진구 인후동1가 907-2
4	인후3동진버들 주차장				전북 전주시 덕진구 인후동1가 807-6
5	산림조합중앙회 전북지역본부 주차장				전북 전주시 덕진구 인후동1가 791

[맵으로가기]

- 지도에서 해당 입지 정보 보기

ZenslaMAP

충전소입지추천



인후동1가

인구수 (19세~70세) 36652 명 | 전기차 등록수 (예측치) 89 대

시각화(화면 구현)

- 메인 페이지

히트맵, 차트

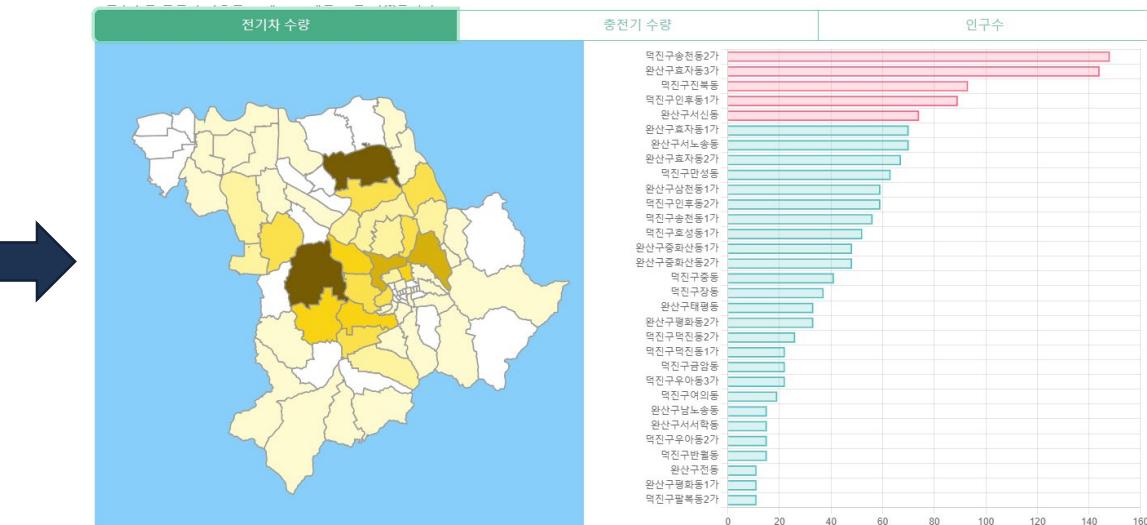
차트			
논리 이름*	물리 이름*	도메인 데이터 타입	설명
지역번호	ano	N/A	BIGINT N-N
시군구명	goo	N/A	VARCHAR(256) NULL
읍면동명	dong	N/A	VARCHAR(256) NULL
인구수	people	N/A	INTEGER NULL
출전소 개수	charger	N/A	INTEGER NULL
전기차 대수	car	N/A	INTEGER NULL
주소	addr	N/A	VARCHAR(256) NULL
전기차 예측값	pchar	N/A	INTEGER NULL
급속 충전기	qcharger	N/A	INTEGER NULL

```

28@ public boolean GetChart()
29{
30    String sql = "select ano,goo,dong,addr,people,charger,pchar from chart";
31    if(this.DBOpen() == false) return false;
32    this.OpenQuery(sql);
33    ArrayList<Integer> carlist = new ArrayList<Integer>();
34    ArrayList<Integer> chargerlist = new ArrayList<Integer>();
35    ArrayList<Integer> peoplelist = new ArrayList<Integer>();
36    while(this.GetNext() == true)
37    {
38        String ano = this.GetValue("ano");
39        String goo = this.GetValue("goo");
40        String dong = this.GetValue("dong");
41        String addr = this.GetValue("addr");
42        int people = this.GetValue("people");
43        int charger = this.GetInt("charger");
44        int car = this.GetInt("pchar");
45
46        if( this.list == null )
47        {
48            this.list = new ArrayList<ChartVO>();
49            ChartVO vo = new ChartVO();
50            vo.setAno(ano);
51            vo.setGoo(goo);
52            vo.setDong(dong);
53            vo.setAddr(addr);
54            vo.setPeople(people);
55            vo.setCharger(charger);
56            vo.setCar(car);
57            list.add(vo);
58            carlist.add(car);
59            chargerlist.add(charger);
60            peoplelist.add(people);
61        }
62        int maxcar = carlist.get(0);
63        int mincar = carlist.get(0);
64        int maxcharger = chargerlist.get(0);
65        int mincharger = chargerlist.get(0);
66        int maxpeople = peoplelist.get(0);
67        int minpeople = peoplelist.get(0);
68        for(int i=0 ; i < carlist.size(); i++)
69    }
70}

```

● 선택된 카테고리에
따라서 새로 정렬



```

137@ public void ThisSort(String code)
138{
139    switch(code)
140    {
141        case "a1" :
142            list2 = this.list.stream().sorted(Comparator.comparingInt(ChartVO::getCar).reversed()).collect(Collectors.toList());
143            break;
144        case "a2" :
145            list2 = this.list.stream().sorted(Comparator.comparingInt(ChartVO::getCharger).reversed()).collect(Collectors.toList());
146            break;
147        case "a3" :
148            list2 = this.list.stream().sorted(Comparator.comparingInt(ChartVO::getPeople).reversed()).collect(Collectors.toList());
149            break;
150    }
151}

```

시각화(화면 구현)

- 젠슬라 맵

추천 지역, 입지 보기



- d3-geo json
지역정보로부터
폴리곤 생성

```
1 [ { "korea.json": [ { "type": "FeatureCollection", "name": "JeonjuBound", "crs": { "type": "name", "properties": { "name": "urn:ogc:def:crs:OGC:1.3:CRS84" } }, "features": [ { "type": "Feature", "properties": { "fid": 1, "EMD_CD": "45111115", "EMD_NM": "\u573a\u5316", "SGG_OID": 10241, "COL_ADM_SE": "45110", "GID": 929 }, "geometry": { "type": "Feature", "properties": { "fid": 2, "EMD_CD": "45111135", "EMD_NM": "\u5316\u5316", "SGG_OID": 10239, "COL_ADM_SE": "45110", "GID": 930 }, "geometry": { "type": "Feature", "properties": { "fid": 3, "EMD_CD": "45111165", "EMD_NM": "\u5316\u5316", "SGG_OID": 10238, "COL_ADM_SE": "45110", "GID": 931 }, "geometry": { "type": "Feature", "properties": { "fid": 4, "EMD_CD": "45111145", "EMD_NM": "\u5316\u5316", "SGG_OID": 10598, "COL_ADM_SE": "45110", "GID": 875 }, "geometry": { "type": "Feature", "properties": { "fid": 5, "EMD_CD": "45111139", "EMD_NM": "\u5316\u5316", "SGG_OID": 10597, "COL_ADM_SE": "45110", "GID": 876 }, "geometry": { "type": "Feature", "properties": { "fid": 6, "EMD_CD": "45111205", "EMD_NM": "\u5316\u5316", "SGG_OID": 10546, "COL_ADM_SE": "45110", "GID": 877 }, "geometry": { "type": "Feature", "properties": { "fid": 7, "EMD_CD": "45111235", "EMD_NM": "\u5316\u5316", "SGG_OID": 12145, "COL_ADM_SE": "45110", "GID": 878 }, "geometry": { "type": "Feature", "properties": { "fid": 8, "EMD_CD": "45111316", "EMD_NM": "\u5316\u5316", "SGG_OID": 12466, "COL_ADM_SE": "45110", "GID": 879 }, "geometry": { "type": "Feature", "properties": { "fid": 9, "EMD_CD": "45111315", "EMD_NM": "\u5316\u5316", "SGG_OID": 12465, "COL_ADM_SE": "45110", "GID": 880 }, "geometry": { "type": "Feature", "properties": { "fid": 10, "EMD_CD": "45111320", "EMD_NM": "\u5316\u5316", "SGG_OID": 13425, "COL_ADM_SE": "45110", "GID": 881 }, "geometry": { "type": "Feature", "properties": { "fid": 11, "EMD_CD": "45111333", "EMD_NM": "\u5316\u5316", "SGG_OID": 13108, "COL_ADM_SE": "45110", "GID": 882 }, "geometry": { "type": "Feature", "properties": { "fid": 12, "EMD_CD": "45111331", "EMD_NM": "\u5316\u5316", "SGG_OID": 11513, "COL_ADM_SE": "45110", "GID": 883 }, "geometry": { "type": "Feature", "properties": { "fid": 13, "EMD_CD": "45111345", "EMD_NM": "\u5316\u5316", "SGG_OID": 13745, "COL_ADM_SE": "45110", "GID": 884 }, "geometry": { "type": "Feature", "properties": { "fid": 14, "EMD_CD": "45111323", "EMD_NM": "\u5316\u5316", "SGG_OID": 11827, "COL_ADM_SE": "45110", "GID": 885 }, "geometry": { "type": "Feature", "properties": { "fid": 15, "EMD_CD": "45111128", "EMD_NM": "\u5316\u5316", "SGG_OID": 15668, "COL_ADM_SE": "45110", "GID": 886 }, "geometry": { "type": "Feature", "properties": { "fid": 16, "EMD_CD": "45111124", "EMD_NM": "\u5316\u5316", "SGG_OID": 10295, "COL_ADM_SE": "45110", "GID": 888 }, "geometry": { "type": "Feature", "properties": { "fid": 17, "EMD_CD": "45111129", "EMD_NM": "\u5316\u5316", "SGG_OID": 10293, "COL_ADM_SE": "45110", "GID": 889 }, "geometry": { "type": "Feature", "properties": { "fid": 18, "EMD_CD": "45111107", "EMD_NM": "\u5316\u5316", "SGG_OID": 10289, "COL_ADM_SE": "45110", "GID": 890 }, "geometry": { "type": "Feature", "properties": { "fid": 19, "EMD_CD": "45111118", "EMD_NM": "\u5316\u5316", "SGG_OID": 10288, "COL_ADM_SE": "45110", "GID": 891 }, "geometry": { "type": "Feature", "properties": { "fid": 20, "EMD_CD": "45111133", "EMD_NM": "\u5316\u5316", "SGG_OID": 10287, "COL_ADM_SE": "45110", "GID": 892 }, "geometry": { "type": "Feature", "properties": { "fid": 21, "EMD_CD": "45111132", "EMD_NM": "\u5316\u5316", "SGG_OID": 10285, "COL_ADM_SE": "45110", "GID": 893 }, "geometry": { "type": "Feature", "properties": { "fid": 22, "EMD_CD": "45111130", "EMD_NM": "\u5316\u5316", "SGG_OID": 10277, "COL_ADM_SE": "45110", "GID": 896 }, "geometry": { "type": "Feature", "properties": { "fid": 23, "EMD_CD": "45111121", "EMD_NM": "\u5316\u5316", "SGG_OID": 10276, "COL_ADM_SE": "45110", "GID": 897 }, "geometry": { "type": "Feature", "properties": { "fid": 24, "EMD_CD": "45111114", "EMD_NM": "\u5316\u5316", "SGG_OID": 10275, "COL_ADM_SE": "45110", "GID": 898 }, "geometry": { "type": "Feature", "properties": { "fid": 25, "EMD_CD": "45111102", "EMD_NM": "\u5316\u5316", "SGG_OID": 10267, "COL_ADM_SE": "45110", "GID": 899 }, "geometry": { "type": "Feature", "properties": { "fid": 26, "EMD_CD": "45111136", "EMD_NM": "\u5316\u5316", "SGG_OID": 10271, "COL_ADM_SE": "45110", "GID": 900 }, "geometry": { "type": "Feature", "properties": { "fid": 27, "EMD_CD": "45111126", "EMD_NM": "\u5316\u5316", "SGG_OID": 10270, "COL_ADM_SE": "45110", "GID": 901 }, "geometry": { "type": "Feature", "properties": { "fid": 28, "EMD_CD": "45111123", "EMD_NM": "\u5316\u5316", "SGG_OID": 10265, "COL_ADM_SE": "45110", "GID": 902 }, "geometry": { "type": "Feature", "properties": { "fid": 29, "EMD_CD": "45111125", "EMD_NM": "\u5316\u5316", "SGG_OID": 10264, "COL_ADM_SE": "45110", "GID": 903 }, "geometry": \u2022 } ] } ] }
```

```
public boolean GetBuilding(String dong)
{
    String sql = "select bno,bname,badd,bfcate,blat,blon from Building where bdong = '" + dong + "' ;";
    if(this.DBOpen() == false) return false;
    this.OpenQuery(sql);
    while(this.GetNext() == true)
    {
        String bno   = this.GetValue("bno");
        String bname = this.GetValue("bname");
        String badd  = this.GetValue("badd");
        String bfcate = this.GetValue("bfcate");
        String blat   = this.GetValue("blat");
        String blon   = this.GetValue("blon");

        if( this.buildinglist == null )
        {
            this.buildinglist = new ArrayList<PlaceVO>();
        }
        PlaceVO vo = new PlaceVO();
        vo.setBno(bno);
        vo.setBname(bname);
        vo.setBadd(badd);
        vo.setBfcate(bfcate);
        vo.setBlat(blat);
        vo.setBlon(blon);
        buildinglist.add(vo);
    }
    this.CloseQuery();
    this.DBClose();
    return true;
}
```



최종결과		building			
논리 이름*	클리 이름*	도메인	데이터 타입	널 허용	
장소 번호	bno	N/A	INTEGER	N·N	
장소 이름	bname	N/A	VARCHAR(60)	NULL	
장소 주소	baddr	N/A	VARCHAR(60)	NULL	
동 이름	bdong	N/A	VARCHAR(30)	NULL	
카테고리 대분류	bfcate	N/A	VARCHAR(30)	NULL	
위도	blat	N/A	VARCHAR(30)	NULL	
경도	blon	N/A	VARCHAR(30)	NULL	

시각화(화면 구현)

- 젠슬라 맵

주변시설 조회

장소	place
邻里 이름*	풀리 이름* 도메인 데이터 타입
장소 번호	pno N/A INTEGER
장소 이름	pname N/A VARCHAR(60) NULL
장소 주소	paddr N/A VARCHAR(60) NULL
도로명 주소	pnewaddr N/A VARCHAR(60) NULL
동 이름	pdong N/A VARCHAR(30) NULL
img url 주소	pimg N/A VARCHAR(256) NULL
카테고리 대분류	pcate N/A VARCHAR(30) NULL
카테고리 소분류	pcatec N/A VARCHAR(30) NULL
카테고리1	pcated1 N/A VARCHAR(30) NULL
카테고리2	pcated2 N/A VARCHAR(30) NULL
카테고리3	pcated3 N/A VARCHAR(30) NULL
카테고리4	pcated4 N/A VARCHAR(30) NULL
카테고리5	pcated5 N/A VARCHAR(30) NULL
전화번호	ptel N/A VARCHAR(30) NULL
위도	plat N/A VARCHAR(30) NULL
경도	plon N/A VARCHAR(30) NULL
utmk_x	utmk_x N/A VARCHAR(30) NULL
utmk_y	utmk_y N/A VARCHAR(30) NULL

ZenslaMAP

충전소입지추천
인후동1가
지역입지순위 1 순위 | 지역입지점수 21 점

주변 편의

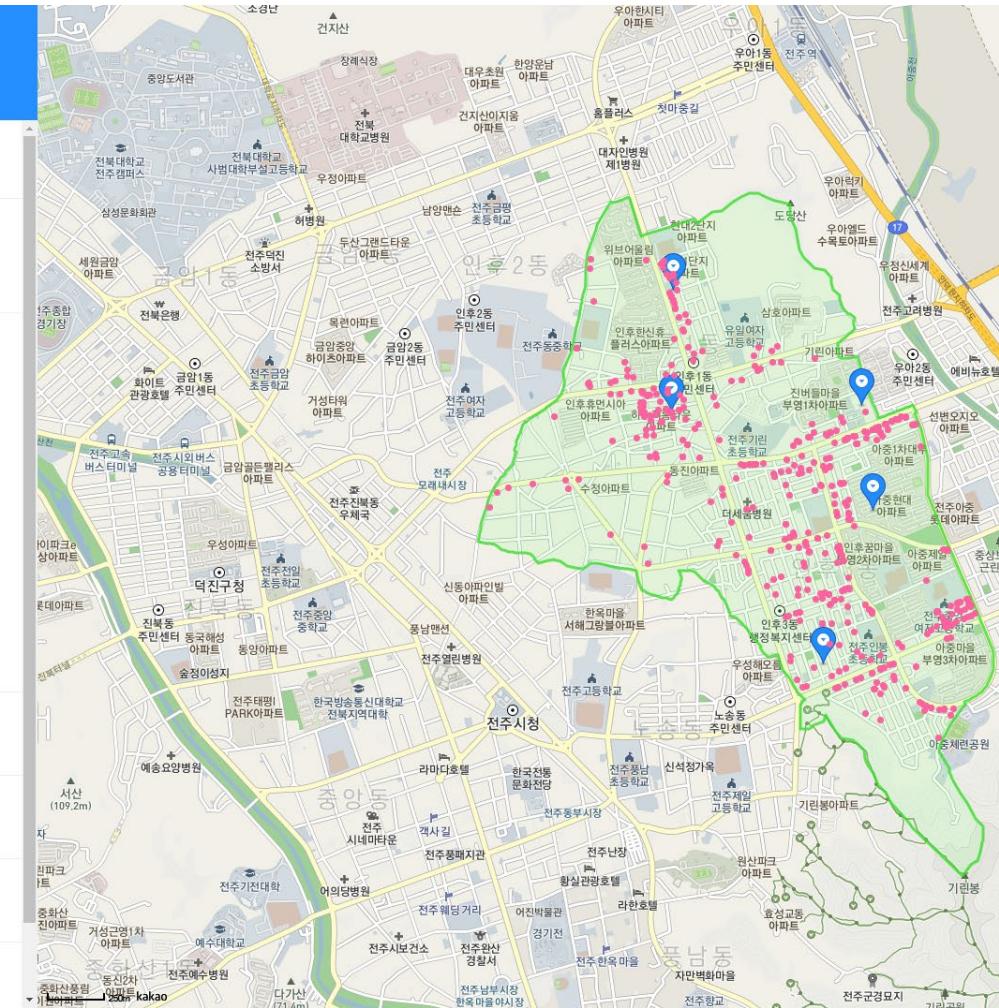


다른지역 >

- 인후동1가
- 중화산동2가
- 효자동2가

추천입지장소

- 1 평나무4길 공영주차장
전북 전주시 완진구 인후동1가 781-7
- 2 전주아중현대아파트 주차장
전북 전주시 완진구 인후동1가 858-2
- 3 아중지구 산림청옆 공영주차장
전북 전주시 완진구 인후동1가 907-2
- 4 인후3동진버들 주차장
전북 전주시 완진구 인후동1가 807-6
- 5 산림조합중앙회 전복지역본부 주차장
전북 전주시 완진구 인후동1가 791



```

public boolean GetCate(String cate, String area)
{
    String sql = "select pno, pname, paddr, plat, plon from place where pcate = '"+cate+"' and pdong = '"+area+"'";
    if(this.DBOpen() == false) return false;
    this.OpenQuery(sql);
    while(this.GetText() == true)
    {
        String pno = this.GetValue("pno");
        String name = this.GetValue("pname");
        String add = this.GetValue("paddr");
        String pcate = this.GetValue("pcate");
        String plat = this.GetValue("plat");
        String plon = this.GetValue("plon");
        if( this.cateList == null )
        {
            this.cateList = new ArrayList<PlaceVO>();
        }
        PlaceVO vo = new PlaceVO();
        vo.setPno(pno);
        vo.setPname(name);
        vo.setPaddr(add);
        vo.setPcate(pcate);
        vo.setPlat(plat);
        vo.setPlon(plon);
        cateList.add(vo);
    }
    this.CloseQuery();
    this.DBClose();
    return true;
}
  
```