**8팀 코드 사용 매뉴얼**

**먼저, csv format은 다음과 같습니다.**

**1)각 팀에게서 받은 raw csv**

Latitude – Longitude – Elevation – Timestamp

**2)처리용 csv**

Date – Startpoint – Endpoint – Difference – Place

**3)최종 csv**

Key – Date – Day – AvailableTime – Distance – AirScore – WeatherScore – Dorm - IOS

**코딩 과정은 다음과 같습니다.**

**0) 원본 csv파일 전처리를 해 제출용 코드를 얻습니다**.

* preprocessing.R: Mapmywalk 앱을 통해 얻은 원본 csv 파일을 이용해 전처리를 진행하는 코드입니다.

**1)1)raw csv에서 머무른 장소와, 머무른 시간을 얻기 위해 다음 두 코드를 이용합니다. 이 코드들을 이용해 2)처리용 csv를 얻습니다.**

* data\_place.R: 각 건물마다 좌표를 지정해 구역을 설정한 후, 특정 구역으로 이동할 때마다 start point와 end point를 잡고 place에 미리 지정해둔 건물의 번호를 작성하는 코드입니다.

-1 out

0 else(가용시간)

1 우정원 위도: 37.24548 ~ 37.24640 / 경도: 127.07588~ 127.07747

2 제2기숙사 여자동 위도: 37.24329~ 37.24400 / 경도: 127.07590 ~ 127.07781

3 제2기숙사 남자동 위도: 37.242199~ 37.24292 / 경도: 127.07680 ~ 127.077625

4 학생회관 위도: 37.24168~37.24227 / 경도: 127.07935~127.08044

5 멀티미디어관 위도: 37.24430~ 37.24488 / 경도: 127.07588~127.07697

6 공과대학 위도: 37.24532~37.24743 / 경도: 127.07993~127.08142

7 전자정보대학: 37.23878~37.23983 / 경도: 127.08279~127.08396

8 예술디자인대학 위도: 37.24131~37.24214 / 경도:127.08389~127.08514

9 국제대학 위도: 37.24184~37.24234 / 경도: 127.08086~127.08167

10 생명과학대학 위도: 37.24266~37.24306 / 경도: 127.08080~127.08177

11 체육대학: 37.24384~37.24488 / 경도: 127.07987~127.08083

12 외국어대학: 37.24476~37.24546 / 경도: 127.07693~127.07836

100~1200 : 해당 구역에서 갑작스럽게 20분 이상 나간 경우

10000: 어떤 구역인지 알 수 없으나 갑작스럽게 20분 이상 나간 경우

-1: 정문을 거쳐 나간 경우(out: 정문 거쳤는지 알아보는 함수)

* exception\_handling.R: data\_place 코드를 거친 후에 csv 파일을 보면 휴대폰 GPS 및 애플리케이션의 한계로 인해 위치가 불안정하게 움직이면서 예를 들어 실제 사용자는 1에 위치하지만 0 1 0 1 0 1 0 처럼 place가 반복되어 나타나는 경우가 발생합니다. 이런 경우 등의 다양한 예외를 처리해주기 위하여 해당 코드를 이용해 전처리를 진행해 보다 깔끔하게 시간에 따른 place 값을 얻을 수 있게 해주는 코드입니다. 세부적인 내용은 다음과 같습니다.

**1. 날짜별로 분류해 처리(1/2)**

(1) 100~1200, 10000 값 처리

- 해당값 이전 place와 이후 place가 모두 해당 값에 100을 나눈 것과 동일하면 그 값으로 변경

ex) 6 600 6 => 6 6 6

<우선순위: 이전place -> 이후 place>

- 이전 place하고만 일치하면 그 값으로 변경

ex) 6 600 0 => 6 6 0

- 이후 place하고만 일치하면 그 값으로 변경

ex) 0 600 6 => 0 6 6

- 아무것도 해당되지 않으면 그대로 둠

ex) 0 600 0 이나 10000

(2) 현재 place가 0인데 이전 place와 이후 place가 동일한 경우 동일하게 변경

ex) 6 0 6 0 6 => 6 6 6 6 6

(3) 바뀐 place값을 테이블에 갱신 후, 해당 파일 dataset\_replace.csv로 내보냄(현재는 주석처리 되어있음)

**2. 날짜별로 분류해 처리(2/2)**

(1) 연속적으로 동일한 place를 갖는 데이터를 합쳐줌

ex) 6 6 6 => 첫번째 6에서의 startpoint : startpoint/ 세번째 6에서의 endpoint: endpoint

(2) 다시 처리된 데이터를 datasetN\_placeclean.csv 파일로 내보냄(현재는 주석처리 되어있음)

**3. 시간차를 이용해 처리**

(1) 특정 건물에 머무른 시간이 3분 이하이면 그 건물을 지나가다가 측정된 것으로 가용시간으로 처리되어야 하기 때문에 이런 경우를 모두 0으로 처리한 후, 다시 데이터를 합침

(2) 다시 처리된 데이터를 datasetN\_placecleanfinal.csv 파일로 내보냄

**2)dataset number를 이용해 key를 추가합니다. 만약 A팀의 1번 dataset이면 키는 A1과 같이 설정했습니다. 그리고 2)처리용 csv에서 가용시간을 도출합니다. 이 가용시간을 key, 중복이 제거된 날짜, 요일과 함께 csv로 저장하여 임시 ‘Key – Date – Day – AvailableTime’ csv를 만듭니다.**

* get\_available\_time.R: key 를 생성하고, 가용시간의 요일별 평균을 구해 키, 날짜, 요일, 가용시간을 csv에 저장합니다.

**3)다음과 같은 코드를 이용해 2)처리용 csv에서 정문과의 거리, 기숙사 거주여부, 식사위치를 얻어냅니다. 그리고 미세먼지와 날씨 정보를 크롤링해 점수화합니다. 그리고 이 미세먼지 점수, 날씨 점수, 기숙사 거주여부, 식사위치를 Key, Date, Day에 맞게 위 2번의 임시 csv에 추가합니다.**

* get\_weather.py: 네이버 날씨에서 웹 크롤링을 파이썬으로 진행하는 코드입니다. 3,4,5월을 각각 진행하였으며, 최종 아웃풋은 점수로 변환되어 리스트 형식으로 반환되게 됩니다. 결과로 도출된 리스트를 adding\_weather.R로 복사하여 csv에 추가합니다. 이때 R에서는 리스트가 아닌 벡터 형태로 데이터를 사용하게 됩니다.
* get\_air.R: 경기도 대기환경 정보의 영통구 시간대별 csv파일을 Input으로 받습니다. 데이터 중, 11시부터 2시까지의 데이터만 추출하여 평균을 낸 후 날짜별로 정리하고, 평균을 점수로 변환하여 벡터 형식으로 반환합니다. 이때, 결과로 도출된 벡터를 adding\_weather.R로 복사하여 날씨 정보와 함께 csv에 추가합니다.
* get\_dorm.R(연주): 기숙사 여부 attribute를 도출하는 코드입니다. Input파일은 처리용 csv파일이며, output은 Bool형식으로 처리하여, 기숙사생 key의 dorm attribute에 1을, 기숙상이 아닌 key의 dorm attribute에는 0을 반환합니다.
* get\_distance.R(수민): place attribute에 따른 점심시간 정문과의 거리 attribute를 도출하는 코드입니다.
* adding\_weather.R: 크롤링한 날씨, 미세먼지 벡터의 값을 날짜에 맞게 csv에 추가하는 코드입니다.
* key\_plus.R: 사람마다 exception handling을 통해 나온 csv파일의 이름을 datasetN으로 변경한 후에, 해당 코드를 실행시킵니다. 이때 각 팀별로 dataset1~5(6)의 csv 파일이 한 폴더 내의 위치해야하며 폴더 별로 코드를 실행시키면서 직접 해당 조의 key값(R파일에서는 k)을 지정하여 key열이 추가된 datasetN\_A1.csv 파일을 얻습니다.
* IOS코드: Target attribute인 IOS를 제외한 나머지 모든 애트리뷰트를 가지고 있는 csv와 key\_plus 이후 생성된 csv 파일을 함께 이용하여 IOS값을 채워넣는 코드입니다. 이때 2개의 csv 파일에는 한 열에 고유 키 값이 작성되어 있어서 IOS값을 결정할 때 사용되는 기숙사 여부를 참고할 때 해당 키 값으로 두 csv파일을 매칭시켜 사용합니다.

전제)

1. 11-14 사이에 만드시 식사는 한 번 한다.

2. 학관/제2기숙사가 있으면 무조건 IS로 한다. (동아리 활동 등 기타 활동 인정하지 않음)

3. 단, 기숙사생의 경우-하교할 일이 없으므로 그냥 무조건 OS처리하면 된다 생각함.

input file

1. datasetA1.csv~datasetG5.csv (각 csv 파일 내에는 key, date, startpoint, endpoint, difference, place가 attribute로 존재)

2. temp\_final.csv (key, date, day, available time, distance, airscore, weatherscore, dorm)

과정1)

temp\_final으로부터 key 값에 따른 dorm값을 data frame으로 생성해 따로 저장 후, 아래 기준에서 기숙사생을 구분할 때 활용

과정2)

**[기준: 11시 ~ 14시]**

1. 나간 적 없는 경우 = 1

2. 나간 적 있는 경우

1) 제2긱 기숙사생 = 0

2) 제2긱 기숙사생이 아닌 경우

(1) 학관/제2긱에 20분 이상 머무른 경우 = 1

(2) 학관/제2긱에 20분 이상 머무르지 않은 경우

가. 11시~ 14시 사이에 나갔다 들어온 경우 = 0

나. 11시 ~ 14시 사이에 나갔으나 14시 내로 들어오지 않은 경우

ㄱ. 14시 이후에 학교 되돌아온 경우 = 0

ㄴ. 14시 이후에 학교 되돌아오지 않은 경우 = 1

다. 11시 이전부터 나간경우

ㄱ. 12시 30분 기준으로 이전에 들어온 경우 = 1

ㄴ. 12시 30분 기준으로 이전에 돌아오지 않은 경우 = 0

3. 해당 시간동안 데이터가 존재하지 않는 경우 = 0 \* 해당 예외처리는 이후 Modeling.R 파일에서 Distance 값을 이용해 바꾼다. 현재 코드에서는 3번의 경우는 1로 처리됩니다.

위 기준을 통해 IS, OS 를 판단합니다.

output file

1. 각 key값 별로 일자와 ios 분류에 대한 파일들

ex) “datasetA1\_ios.csv” ~ “datasetG5\_ios.csv”

1. Final.csv: temp\_final.csv에 IOS attribute 추가

**4)모든 attribute가 추가된 임시 csv에서 데이터 수집기간 외 데이터와 주말 데이터를 삭제합니다. 이로 3)최종 csv를 얻었습니다.**

* date\_cleaning.R: csv에서 데이터 수집기간 외 데이터와 주말 데이터를 삭제하는 코드입니다.

**5)최종으로 도출된 하나의 csv 파일을 이용해 RPART package를 이용해 모델을 도출합니다.**

* Modeling.R:최종 csv 파일을 input 값으로 넣은 후에 앞서 전처리 하지 않았던 3번의 경우를 Distance가 NA일 경우에 1로 처리해주며, Distance와 AvailableTime의 NA값 대신 평균 값을 집어넣어 모델링을 실시하며, 해당 모델링의 최종 Tree를 jpg파일로 자동으로 저장하며, Confusion Matrix는 콘솔에서 확인하는 과정을 거칩니다.
* 모델링할때는 test set과 train set을 각각 3:7의 비율로 나누어 진행합니다.

**3. distance.R**

**개요** | 정문과의 거리 attribute를 도출하기 위한 코드입니다.

기본적으로 점심시간의 거점(수업과 같이 오래 머무르는 원인을 제공하는 중심 지점)의 위치가 정문과 가까울수록 나가서 먹기 용이할 것이라는 전제에서 출발했습니다. 식사 전 뿐만 아니라 식사 후에 돌아오는 경우도 고려되야 했기에 점심시간 (11-14시) 내의 거점들의 평균 좌표를 산출한 후 이 좌표와 정문의 좌표간의 거리를 m 단위로 출력합니다.

**코드 사용** |

**입력**: 1차적으로 가공되어 Date – Startpoint – Endpoint – Difference – Place의 형태로 구성된 csv

**출력**: Key - Date - Distance로 구성된 최종 합산 전 중간 csv

**사용**:

1. 해당 팀의 데이터셋(5개 혹은 6개)를 워킹 디렉토리로 불러옵니다.
2. 아래 코드를 통해 불러 모든 데이터셋을 불러와 date에 넣습니다.

date <- c("dataset1","dataset2","dataset3","dataset4","dataset5")

1. 중복된 날짜를 제거한 고유 날짜 벡터를 만들고, 같은 일자끼리 filter 메쏘드를 통하여 분류합니다.
2. 해당 날짜에서 startpoint혹은 endpoint가 (둘 중 하나라도) 11시 이상 14시 미만인 경우를 filter를 통해 추출합니다. (점심시간 중에 걸치기만 하면 거점으로 인정하겠다)
3. 날짜, 시간 필터를 거친 벡터를 xy\_convert 함수에 입력.
4. 각 place 번호에 일치하는 위도 경도 좌표로 치환 후 x\_vector, y\_vector에 각각 저장함. 이 때, 중복 값은 넣지 않는다.
5. 각 벡터의 평균을 구하여 (x,y)형태의 벡터로 반환한다.
6. Imap 라이브러리를 통하여 반환된 벡터와 정문 좌표간의 거리를 m 단위로 계산하여 distance 벡터에 append한다.
7. 해당 벡터에 key값과 date 값을 매칭하여 csv로 출력한다.
8. 팀 데이터셋을 변경해가며 1-9의 과정을 반복. 코드를 통해 파일에 append한다

※파일 추출 코드 부분은 getAvailableTime.R 코드와 동일