```
title: "Dacon 13회 퇴근시간 버스승차인원 예측 모델링 경진대회"
author: "이준수"
date: "2019년 12월 6일" # 제출날짜

## 1. 라이브러리 및 데이터
setwd("c:/deep1")
test <-read.csv("c:/deep1/test.csv",  stringsAsFactors=FALSE,fileEncoding
= "UTF-8")
train <-read.csv("c:/deep1/train.csv",
stringsAsFactors=FALSE,fileEncoding = "UTF-8")
submission_sample <-read.csv("c:/deep1/submission_sample.csv",
stringsAsFactors=FALSE
                             ,fileEncoding = "UTF-8")

## 2. 데이터 전처리
install.packages("fastDummies")
library(fastDummies)

train$date<-as.Date(train$date)
train["weekday"]<-weekdays(train$date) #요일 생성
train<-dummy_cols(train, select_columns="weekday")#더미변수 생성

test$date<-as.Date(test$date)
test["weekday"]<-weekdays(test$date) #요일 생성
test<-dummy_cols(test, select_columns="weekday")#더미변수 생성

table(train$in_out)
train$in_out<-ifelse(train$in_out=="시내",0,1)
test$in_out<-ifelse(test$in_out=="시내",0,1)

train["X6.8ride"]<-train$X6.7_ride + train$X7.8_ride # 6 ~ 8시 승차인원
train["X8.10ride"]<-train$X8.9_ride + train$X9.10_ride # 8 ~ 10시 승차인원
train["X10.12ride"]=train$X10.11_ride+train$X11.12_ride

train["X6.8off"]<-train$X6.7_takeoff + train$X7.8_takeoff # 6 ~ 8시
하차인원
train["X8.10off"]<-train$X8.9_takeoff + train$X9.10_takeoff  # 8 ~ 10시
하차인원
train["X10.12off"]<-train$X10.11_takeoff + train$X11.12_takeoff

## 3. 탐색적 자료분석
#상관계수
install.packages("corrplot")
library(corrplot)
```

```
train21<-train[,9:21]
cortrain21<-cor(train21)
corrplot(cortrain21, method="number")
train22<-cbind(train[,30:35],train$X18.20_ride)
cortrain22<-cor(train22)
corrplot(cortrain22, method="number")

test["X6.8ride"]<-test$X6.7_ride + test$X7.8_ride
test["X8.10ride"]<-test$X8.9_ride + test$X9.10_ride
test["X10.12ride"]=test$X10.11_ride+test$X11.12_ride

test["X6.8off"]<-test$X6.7_takeoff + test$X7.8_takeoff
test["X8.10off"]<-test$X8.9_takeoff + test$X9.10_takeoff
test["X10.12off"]<-test$X10.11_takeoff + test$X11.12_takeoff

## 4. 외부데이터 사요
# 해당 주요 장소의 임의 지역 경도, 위도
jeju<-c(126.52969,33.51411) # 제주 측정소 근처
gosan<-c(126.16283, 33.29382 ) #고산 측정소 근처
seongsan<-c(126.8802,33.38677) #성산 측정소 근처
po<-c(126.5653,33.24616) #서귀포 측정소 근처

#측정소와 정류장 사이 거리 계산 적용
install.packages("geosphere")
library(geosphere)
#각 정류장과 제주도에 존재하는 4 군데의 기상 측정소와의 거리를 계산하여 t1~t4 에 할당
t1<-distm (jeju, train[,c('longitude','latitude')], fun=distHaversine)
t2<-distm (gosan, train[,c('longitude','latitude')], fun=distHaversine)
t3<-distm (seongsan, train[,c('longitude','latitude')],
fun=distHaversine)
t4<-distm (po, train[,c('longitude','latitude')], fun=distHaversine)

total<-rbind(t1,t2,t3,t4)
tab_t<-t(total) # 열과 행 회전
colnames(tab_t)<-c("jeju","gosan","seongsan","po") # 열이름
tab_f<-data.frame(tab_t)
train['dis_jeju']<-tab_f$jeju
train['dis_gosan']<-tab_f$gosan
train['dis_seongsan']<-tab_f$seongsan
train['dis_po']<-tab_f$po
#각 측정소와 정류소의 거리를 계산한 다음, 해당 정류소에서 가장 가까운 곳에 있는
측정소를 dist_name 변수의 값으로 넣는다
train['dist_name']<- colnames(tab_f)[max.col(-tab_f,ties.method="first")]
table(train$dist_name)

#test_data 에도 적용
te1<-distm (jeju, test[,c('longitude','latitude')], fun=distHaversine)
```

```
te2<-distm (gosan, test[,c('longitude','latitude')], fun=distHaversine)
te3<-distm (seongsan, test[,c('longitude','latitude')],
fun=distHaversine)
te4<-distm (po, test[,c('longitude','latitude')], fun=distHaversine)
total2<-rbind(te1,te2,te3,te4)
tab_te<-t(total2)
colnames(tab_te)<-c("jeju","gosan","seongsan","po")
tab_f2<-data.frame(tab_te)
test['dis_jeju']<-tab_f2$jeju
test['dis_gosan']<-tab_f2$gosan
test['dis_seongsan']<-tab_f2$seongsan
test['dis_po']<-tab_f2$po
test['dist_name']<- colnames(tab_f2)[max.col(-
tab_f2,ties.method="first")]
```

#기상청 데이터

활용(https://data.kma.go.kr/data/grnd/selectAsosRltmList.do?pgmNo=36 기상청 제주도 날씨 데이터)

```
weather<-read.csv("c:/deep1/weather.csv",stringsAsFactors=FALSE) #
```

외부데이터

#외부데이터에서 나오는 지점명들을 변경

```
weather$지점<-ifelse(weather$지점==184,"jeju",
                  ifelse(weather$지점==185,"gosan",
                         ifelse(weather$지점==188,"seongsan",
                                ifelse(weather$지점==189,"po",1))))
weather$일시<-as.Date(weather$일시)

# train, test 의 변수명과 통일시키고, Na 의 값은 0.0000 으로 변경
install.packages("reshape")
library(reshape)
weather<-rename(weather,c("일시"="date","지점"="dist_name"))
weather[is.na(weather)]<-0
colSums(is.na(train2))
```

#최종적으로 완성이 된 외부데이터와 현재 만들어진 train, test data 에 각각 merge 한다.

```
train2<-merge(train,weather, by=c("date","dist_name"), ,all.x=TRUE)
test2<-merge(test,weather, by=c("date","dist_name"), all.x=TRUE)
```

#미세먼지, 초미세먼지지 일평균 데이터(https://www.airkorea.or.kr/web 출처 에어코리아)

```
PM10<-read.csv("c:/deep1/PM10.csv",stringsAsFactors=FALSE) # 외부데이터
PM2.5<-read.csv("c:/deep1/PM2.5.csv",stringsAsFactors=FALSE) # 외부데이터
```

```
PM10$날짜<-as.Date(PM10$날짜)
PM2.5$date<-as.Date(PM2.5$date)
colnames(PM10)<-c("date","PM10")
PM10[is.na(PM10)]<-0 # Na의 값은 0.0000으로 변경
PM2.5[is.na(PM2.5)]<-0
#train, test data에 각각 merge 한다.
dust<-merge(PM10,PM2.5, by="date", all.x=TRUE)
train2<-merge(train2,dust, by="date", all.x=TRUE)
test2<-merge(test2,dust, by="date", all.x=TRUE)
#더미변수 생성
train2<-dummy_cols(train2, select_columns="dist_name")
test2<-dummy_cols(test2, select_columns="dist_name")


#주말은 1 평일은 0 더미변수 생성
train2['weekend']<-ifelse(train2$weekday=="토요일",1,
                          ifelse(train2$weekday=="일요일" ,1,0))
test2['weekend']<-ifelse(test2$weekday=="토요일",1,
                         ifelse(test2$weekday=="일요일" ,1,0))


#5. randomforest을 활용한 모델링
input_var <- X18.20_ride ~ in_out + latitude + longitude + X6.8ride +
X8.10ride + X10.12ride + X6.8off + X8.10off + X10.12off + weekday_일요일 +
weekday_월요일 + weekday_화요일 + weekday_수요일 + weekday_목요일 +
weekday_금요일 + weekday_토요일  +
  dis_jeju + dis_gosan + dis_seongsan + dis_po + 평균기온..C. +
최저기온..C. + 최고기온..C. +   일강수량.mm.   + 평균.풍속.m.s.  +
평균.이슬점온도..C. +평균.상대습도...+  평균.증기압.hPa. +
  평균.현지기압.hPa.+평균.해면기압.hPa.+ 합계.일조시간.hr.+ 합계.일사량.MJ.m2.+
평균.전운량.1.10.+  평균.지면온도..C. + PM10 + PM2.5+
  dist_name_gosan + dist_name_jeju + dist_name_po +dist_name_seongsan +
weekend

#무작위로 추출
install.packages(c("dplyr","randomForest","caret","ROCR"))
library(randomForest)
library(caret)
library(ROCR)
library(dplyr)
set.seed(1217)
sample_train2<-sample_frac(train2,0.01)# 1%만큼 무작위 추출
#최적의 파라미터 찾기
```

```
ntree<-c(400,500,600)
mtry<-c(2:4)
param<-data.frame(n=ntree,m=mtry)
param
for(i in param$n){
  cat('ntree=',i,'\n')
  for(j in param$m){
    cat('mtry')
    model_train<- randomForest(input_var, data=sample_train2,
ntree=i,mtry=j,
                               na.action=na.omit)
    print(model_train)
  }
}
```

# ntree=400, mtry=4 가 가장 낮은 MSE 가장 높은 설명분산을 가진다.

```
# 데이터분할
sn <- sample(1:nrow(sample_train2),size=nrow(sample_train2)*0.7)
out.tr<-sample_train2[sn,]
out.val <-sample_train2[-sn,]

#  모델학습
random <- randomForest(input_var, data=out.tr,ntree=400,
mtry=4,importance = T)
random
# 변수중요도
importance(random)
varImpPlot(random)
# 5. 모델평가(예측 반올림)
prd_v<-round(predict(random, newdata=out.val,type='class'))
rf.resid<-prd_v-out.val$X18.20_ride
sqrt(mean(rf.resid^2))#MSE

# 6. 예측결과
random2 <- randomForest(input_var, data=train2,ntree=400,
mtry=4,importance = T)
# 예측값 입력
test2['X18.20_ride']<-round(predict(random2, newdata=test2))
submission_sample$X18.20_ride<-test2$X18.20_ride
write.csv(submission_sample, "C:/deep1/submission.csv")
```