Finland Commercial Building Energy (Part One)

Team 12

Yalin Li
Zhi Li

Midterm Report: Finland Commercial Building Energy

Contents (part three)

Yalin, Zhi

Project Introduction:

The dataset contains power consumption for 78 buildings in the country of Finland. The whole data include 24 hours power information and weather data for each day of the year 2013.
Our goal is using the data science skills to understand the recent data we already have, and get the weather data to do feature engineering and build models for prediction, classification and clustering.
The building is identified by its building name (vac).
The unit for power is kWh of electricity usage and heating energy consumption data.
The data of date which has distinguish by weekday, weekend, holiday, month, and the day of week.

Problem Statement:

We get 78 buildings which located in different places in Finland. Each building has two different power type which are electricity usage and heating energy. The Vokia Inc want to monitor and reduce energy consumption in 78 of their buildings. Vokia Inc. owns these buildings and wants to understand and reduce energy usage and wants to make the buildings more energy efficient.
Our steps to help them are in the below:

1. Integrated data, which get the data only with electricity and heating energy
2. Use missing value filling techniques to fill any data that isn't complete and ensure that the time series is complete as well.
3. Extract features about weekday, weekend, holiday, month of year, day of week, and Base Hour Flag.
4. Using GOOGLE API to get latitude and longitude from address for each building. And retrieve the nearest airport location to get Station ID.
5. Make a log or write a try catch to given the warning about wrong address input.
6. After getting station ID of each nearest airport, retrieve the hourly temperature for each building for the duration of the dataset.
7. Cleanup the weather data and join the processed Raw Data and Address Data into one output file.
8. Calculate how much energy each building used in every square meter.

Data Preparation:

Tools Used: R language and Python script was used for preprocessing data. (Only step f using both R and Python)

a. Require “sqldf” in R, and get the elect and heating data from “FinlandMasked”, and then “rbind” two data frames, then we can get all the data only with elect and heating type.

```
require(sqldf)
#choosing the data in type 'elect' and 'Dist_Heating'
elc <- sqldf("select * from FinlandMasked where type = 'elect'")
dist_heat <- sqldf("select * from FinlandMasked where type = 'Dist_Heating'")

#combine two new table 'elc' and 'dist_heat' into one new table
FinlandMasked <- rbind(elc,dist_heat)
```

b. Fill the missing value in few rows in type elect and heating.

```
FinlandMasked$vac <- as.character(FinlandMasked$vac)
FinlandMasked$vac[FinlandMasked$BuildingID == 81909]<-"Building 27"
FinlandMasked$vac[FinlandMasked$BuildingID == 82254]<-"Building 9"
FinlandMasked$vac[FinlandMasked$BuildingID == 83427]<-"Building 9"
FinlandMasked$vac[FinlandMasked$BuildingID == 84681]<-"Building 9"
```

c. Use desc to sort a variable in descending order and do the extract features.

```
FinlandMasked <- FinlandMasked %>% mutate(DayOfWeek = as.numeric(format(FinlandMasked$Date, format = "%w")))
FinlandMasked <- FinlandMasked %>% mutate(MonthOfYear = as.numeric(format(FinlandMasked$Date, format = "%m")))
FinlandMasked <- FinlandMasked %>% mutate("Weekday/Weekend" = ifelse(DayOfWeek %in% 1:5 , 1, 0))
FinlandMasked <- FinlandMasked %>% mutate(Base_Hour_Flag = ifelse(hour %in% 0:4, 1, ifelse(hour %in% 22:23, 1, 0)))
FinlandMasked <- FinlandMasked %>% mutate(Day =  as.numeric(format(FinlandMasked$Date, format = "%d")))

FinlandMasked <- FinlandMasked %>% mutate(Holiday = ifelse(date == "20130101", 1, ifelse(date == "20130106", 1,
                                                    ifelse(date == "20130329", 1, ifelse(date == "20130331", 1,
                                                    ifelse(date == "20130401", 1, ifelse(date == "20130501", 1,
                                                    ifelse(date == "20130509", 1, ifelse(date == "20130519", 1,
                                                    ifelse(date == "20130622", 1, ifelse(date == "20131102", 1,
                                                    ifelse(date == "20131206", 1, ifelse(date == "20131225", 1,
                                                    ifelse(date == "20131226", 1, 0))))))))))))))
```

d. Calculate how much energy each building used in every square meter

```
#calculate kwh/square
full <- full %>% mutate("kwh/square" = full$Consumption/full$area_floor._m.sqr)
```

e. Extract site location information from FinlandAddress file. Location name and location type has been extracted, which will be used to attach to the merged dataset and create location dimension table by using GOOGLE API, and give warning if input the wrong address.

```
#library(RJSONIO)
#install.packages("jsonlite")
library(jsonlite)
i <- 1
while(i <= nrow(FinlandAddress)){
  c <- strsplit(as.character(FinlandAddress$address[i]),split = " ")
  url <- paste("https://maps.googleapis.com/maps/api/geocode/json?address=",c[[1]][1],"+",c[[1]][2],"+","&key=AIzaSyC2hIqaBPSkobx1vyyKUwvWhwqJNSn1LHY",sep = "")
  json <- fromJSON(url)
  if(json$status != "OK"){
    cat("The provided API key is invalid.")
  }else{
    FinlandAddress$lat[i] <- json$results$geometry$location$lat
    FinlandAddress$lng[i] <- json$results$geometry$location$lng
  }
  i <- i+1
}
```

f. Download all weather Station ID from http://weather.rap.ucar.edu/surface/stations.txt“website by using R. Using python to

find out all Finland Airport Station ID with latitude and longitude, generate “station_info.csv”.

```
download.file("http://weather.rap.ucar.edu/surface/stations.txt", "station.txt", method = "curl")
```

```
f=open('stations.txt','r')
content = f.readlines()
f.close()

start =False
interested_content =[]
for line in content:

    if line.__contains__('FINLAND'):
        start=True
        continue

    if start == True :
        line=line.strip()
        if line == '':
            start = False
            continue
        interested_content.append(line.strip())



f=open('station_info.csv','w')
#write header
f.write('station_id,latitude,longtitude\n')
for line in interested_content:
    station_name= line[0:17].strip()
    station_id=line[17:21].strip()
    if station_id == '': continue
    coord = line[36:51].split()
    latitu = float(coord[0]+'.'+coord[1].strip("N")) #skip "N"
    longtitu = float(coord[2]+'.'+coord[3].strip("E")) #skip "E"
    f.write(station_id+','+str(latitu)+','+str(longtitu)+'\n')
f.close()
```

g. Using R to read “station_info.csv” and calculate the shortest distance between each buildings and airport location by using function:

$$a = \sin^2(\Delta\varphi/2) + \cos\varphi_1 \cdot \cos\varphi_2 \cdot \sin^2(\Delta\lambda/2)$$

$$c = 2 \cdot \operatorname{atan2}(\sqrt{a}, \sqrt{(1-a)})$$

$$d = R \cdot c$$

```
stations <- read.csv("station_info.csv")
#add empty cloumns
FinlandAddress <- mutate(FinlandAddress, station_id="")
stations <- mutate(stations, distance="")
#Calculate distance in kilometers between two points
earth.dist <- function (long1, lat1, long2, lat2)
{
  rad <- pi/180
  a1 <- lat1 * rad
  a2 <- long1 * rad
  b1 <- lat2 * rad
  b2 <- long2 * rad
  dlon <- b2 - a2
  dlat <- b1 - a1
  a <- (sin(dlat/2))^2 + cos(a1) * cos(b1) * (sin(dlon/2))^2
  c <- 2 * atan2(sqrt(a), sqrt(1 - a))
  R <- 6378.145
  d <- R * c
  return(d)
}
x <- 1
while(x <= nrow(FinlandAddress)){
  lat <- as.numeric(FinlandAddress$lat[x])
  lng <- as.numeric(FinlandAddress$lng[x])
  y <- 1
  while(y <= nrow(stations)){
    lat_ <- as.numeric(stations$latitude[y])
    lng_ <- as.numeric(stations$longtitude[y])
    stations$distance[y] <- earth.dist(lng,lat,lng_,lat_)
    y <- y+1
  }
  stations <- arrange(stations, distance)
  FinlandAddress$station_id[x] <- as.character(stations$station_id[1])
  x <- x+1
}
```

h. Download weather data by using three different station ids respectively and write in three different csv file. (After we using function to get the shortest distance between each building and airport station, we got three different station ids.)

```
W_EFUT_all<-NULL
i<-1
while(i<=nrow(date))  {
  W_tempEFUT<- getDetailedWeather(station_id = "EFUT", date =date$Date[i], opt_custom_columns = T, custom_columns = c(2:14))
  W_EFUT_all<- rbind(W_EFUT_all, W_tempEFUT)
  i<-i+1
}

write.csv(W_EFUT_all,"W_EFUT_all.csv")

W_EFHK_all<-NULL
i<-1
while(i<=nrow(date))  {
  W_tempEFHK<- getDetailedWeather(station_id = "EFHK", date =date$Date[i], opt_custom_columns = T, custom_columns = c(2:14))
  W_EFHK_all<- rbind(W_EFHK_all, W_tempEFHK)
  i<-i+1
}
write.csv(W_EFHK_all,"W_EFHK_all.csv")

W_EFTP_all<-NULL
i<-1
while(i<=nrow(date))  {
  W_tempEFTP<- getDetailedWeather(station_id = "EFTP", date =date$Date[i], opt_custom_columns = T, custom_columns = c(2:14))
  W_EFTP_all<- rbind(W_EFTP_all, W_tempEFTP)
  i<-i+1
}
write.csv(W_EFTP_all,"W_EFTP_all.csv")
```

Note:

- Pick elect and heating data from FinlandMasked and rbind it back.
- Put Latitude and Longitude from JSON online to FinlandAddress
- After calculate the nearest distance put the station ID in FinlandAddress of each building.
- Combine FinlandMasked and FinlandAddress into fullData by using left join.
- Download weather data in three station ids and put into three csv for clean data.

Data Cleansing:

Tools Used: R script was used for cleaning the data. (Because we download three weather data by three different station ID, thus, we will clean each weather data separately and combine three together after we cleaned everything)

a. Deal with missing value

```
W_EFHK_all <- read.csv("W_EFHK_all.csv")
W_EFHK_all <- W_EFHK_all[-1]
W_EFHK_all$Time <- as.POSIXct(W_EFHK_all$Time,format="%m/%d/%y %H:%M")
sum(is.na(W_EFHK_all$Time))             #3
sum(is.na(W_EFHK_all$TemperatureF))
sum(is.na(W_EFHK_all$Dew_PointF))
sum(is.na(W_EFHK_all$Humidity))
sum(is.na(W_EFHK_all$Sea_Level_PressureIn))     #missing 1
W_EFHK_all$Sea_Level_PressureIn[which(is.na(W_EFHK_all$Sea_Level_PressureIn))] <- mean(W_EFHK_all$Sea_Level_PressureIn, na.rm = TRUE)
sum(is.na(W_EFHK_all$VisibilityMPH))            #missing 1
```

b. Deal with noisy data

```
summary(W_EFHK_all$VisibilityMPH)               #min -9999
#deal with noisy data
i <- 1
while(i <= nrow(W_EFHK_all)){
  if(W_EFHK_all$VisibilityMPH[i] == -9999){
    W_EFHK_all$VisibilityMPH[i] = "NA"
  }
  i <- i+1
}
```

c. Format “Time” of weather data and fullData (we put all information into fullData file except weather data), which help to combine two files.

```
summary(fullData$Time)
fullData$Time <- as.character(fullData$Time)
summary(weather_all$Time)
weather_all$Time <- as.character(weather_all$Time)
```

d. Removing duplicates (Using inner join to delete repetitive rows, time and other information)

```
full <- dplyr::inner_join(fullData, weather_all, by=c("station_id"="station_id","Time"="Time"))
```

e. Delete useless columns

```
#delete unused cloumns
full <- full[-16]
full <- full[-16]
full <- full[-25]
full <- full[-25]
```

Note:

Data Visualization (Part Three)

Tools Used: Tableau (Will Present on Sunday)