ADS – Midterm Report

GROUP - 10

Pooja Goyal, Shashwat Mehra, Varsha Holennavar | Advances in Data Science | 11/17/2016

1. Part- I Data Wrangling

We have been provided with 2 data files, “Finland_Masked” and “Finland_Addresses_Area”

- Finland_Masked has the data of Power Consumption in kWh of Building-1 to Building -33
- We selected only Consumption of “Electricity” and “Distributed Heating”

1. LIBRARY’S USED:

```
1  library(sqldf)
2  library(dplyr)
3  library(tidyr)
4  library(ggmap)
5  library(RCurl)
6  library(XML)
7  library(rvest)
8  library(geosphere)
9  library(weatherData)
10 library(zoo)
11 library(Imap)
```

2. READING THE CSV:

```
13 #Reading the CSV's
14 addData = read.csv(file="Finland_addresses_area.csv",header = TRUE)
15 finland_Data = read.csv(file="Finland_masked.csv",header = TRUE)
```

3. FETCH THE GEO-LOCATIONS OF BUILDING’S:

To get the geo-locations (i.e. latitude and longitude) of each building, we used library “GGMAP” which has a function geocode() which gives the latitude and longitude of the location provided to the function.

```
19 #Getting the latitude and longitude of all the address's provided
20 vac = addData$vac
21 geocodes = geocode(as.character(addData$address))
22 addData = data.frame(addData[,2:3],geocodes)
23 addData = cbind(addData,vac)
```

Output:

address	area_floor._m.sqr	lon	lat	vac	nearestAirport
Metsahovintie 113	110	24.39346	60.21869	Building 1	EFHK
Otakaari 3	11697	24.82941	60.18835	Building 2	EFHF
Konemiehentie 1	469	24.82304	60.18715	Building 3	EFHF
Rakentajanaukio 4	8766	24.83034	60.18695	Building 4	EFHF
Vuorimiehentie 2	11068	24.82589	60.18294	Building 5	EFHF
Otakaari 1	41339	24.82843	60.18626	Building 6	EFHF
Puumiehenkuja 3	2661	24.82525	60.18740	Building 7	EFHF
Otakaari 4	8206	24.82740	60.18710	Building 8	EFHF
Vuorimiehentie 1	6455	24.82469	60.18179	Building 9	EFHF
Sahkomiehentie 4	7954	24.82580	60.18905	Building 10	EFHF
Puumiehenkuja 5	5100	24.82400	60.18776	Building 11	EFHF
Otakaari 5	12240	24.82996	60.18968	Building 12	EFHF
Otaniementie 9	5868	24.82846	60.18449	Building 13	EFHF
Tietotie 1	15467	24.82061	60.18540	Building 14	EFHF
Lampomiehenkuja 3	3358	24.83085	60.18155	Building 15	EFHF
Betonimiehenkuja 5	3268	24.83167	60.18104	Building 16	EFHF
Lampomiehenkuja 2	6967	24.83165	60.18215	Building 17	EFHF
Metallimiehenkuja 10	1019	24.83030	60.18251	Building 18	EFHF
Betonimiehenkuja 1	2005	24.83185	60.17991	Building 19	EFHF
Puumiehenkuja 2	7272	24.82599	60.18697	Building 20	EFHF
Tekniikantie 3	3640	24.82491	60.18059	Building 21	EFHF
Otakaari 26	1758	24.83080	60.18477	Building 22	EFHF
Betonimiehenkuja 3	3034	24.83128	60.18063	Building 23	EFHF
Kemistintie 1	14470	24.82512	60.18342	Building 24	EFHF
Sahkomiehentie 3	3770	24.82601	60.18920	Building 25	EFHF
Konemiehentie 2	12447	24.82269	60.18720	Building 26	EFHF
Konemiehentie 4	553	24.82242	60.18751	Building 27	EFHF
Runeberginkatu 14-16	7380	24.92314	60.17178	Building 28	EFHF
Runeberginkatu 22-24	1287	24.92221	60.17357	Building 29	EFHF
Arkadiankatu 24	1440	24.92337	60.17129	Building 30	EFHF
Otaniementie 17	17383	24.81879	60.18713	Building 31	EFHF
Otakaari 7	10029	24.83324	60.18921	Building 32	EFHF
Otaniementie 1	23473	24.83403	60.18148	Building 33	EFHF

4. CREATING A VARIABLE TO STORE THE URL'S WITH (LAT & LON):

```
#Getting the URL's in variable updUrl
x = 1
for (i in addData$vac) {
  url = "http://api.wunderground.com/auto/wui/geo/GeoLookupXML/index.xml?query="
  updUrl = paste(paste0(url,addData$lat),addData$lon,sep = ",")
}
```

5. GETTING THE NEAREST AIPORT STATION CODE:

We are getting the nearest airport station code and adding it to the "Finland_Address_Area" data-frame as a new column "nearestAirport"

```r
#Getting the nearest Aiport Station Code
source("NearestAirport.R")
for(i in 1:length(unique(updUrl))){
  stdCode = nearestAirportCal(updUrl[i],addData$lat[i],addData$lon[i])
  addData$nearestAiport[i] =stdCode
}
```

5.1 Nearest Airport Function:

We have created an R script, which has two functions:

- **nearestAirportCal(par-1, par-2,par-3):** This function goes to the URL, scrape the city, station code, latitude, longitude. Once the scraping and retrieval of data is done, the function then cleans the data and pass the cleansed values to another function getDistance().
- **getDistance(5-params):** This functions receives the station code along with the latitude & longitude of all the cities present in the web page as well as the latitude and longitude of the base city from where the distance is to be calculated. It computes on the basis of the radius of the earth and returns the station code with minimum distance in kms.

nearestAiportCal():

Step 1: Scraping city, icao(Station Codes), latitudes, longitudes

```r
nearestAirportCal = function(url,baseLat,baseLon){
  nearestAirport=""
  for(i in 1:length(url)){
    #Step1: Select airport under <airport> tags and getting the cities and station codes
    cities = url[i] %>%
      read_html() %>%
      html_nodes("airport") %>%
      html_nodes("station") %>%
      html_nodes("city") %>%
      html_text()
    stationCodes = url[i] %>%
      read_html() %>%
      html_nodes("airport") %>%
      html_nodes("station") %>%
      html_nodes("icao") %>%
      html_text()
    lat = url[i] %>%
      read_html() %>%
      html_nodes("airport") %>%
      html_nodes("station") %>%
      html_nodes("lat") %>%
      html_text()
    lon = url[i] %>%
      read_html() %>%
      html_nodes("airport") %>%
      html_nodes("station") %>%
      html_nodes("lon") %>%
      html_text()
```

Step 2: Cleaning of Scraped Data (Regular Expression Used)

```
#Step2: Clean the cities
toSend = ''
toSendStCode = ''
toSendLat = ''
toSendLon = ''
ctr = 1
lp = 1
for(x in cities){
  if(x != "" && stationCodes[lp]!=""){
    toSend[ctr] = gsub("[,]\\s*[-]","",x)
    toSendStCode[ctr] = stationCodes[lp]
    toSendLat[ctr] = lat[lp]
    toSendLon[ctr] = lon[lp]
    ctr = ctr+1
  }
  lp = lp+1
}
```

Step 3: Call getDistance() to get the nearest Airport and returning it

```
#Step3: Compute the distance between the city and the airports collected
toSendLat = as.numeric(toSendLat)
toSendLon = as.numeric(toSendLon)
nearestAirport[i] = getDistance(baseLat,baseLon,toSendStCode,toSendLat,toSendLon)
```

getDistance():

Step: Function which calculates distance between airport and base city and returning the station code of City which is nearest to the base city.

```
#Function for getting the nearest airport:
getDistance = function(bLat,bLon,stCodes,recLat,recLon){
  ctr = 1
  distance = ""
  rad <- pi/180
  R <- 6378.145
  for(i in stCodes){
    a1 <- recLat[ctr] * rad
    a2 <- recLon[ctr] * rad
    b1 <- bLat * rad
    b2 <- bLon * 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))

    distance[ctr] = R * c
    #print(distance[ctr])
    ctr = ctr + 1
  }

  return(stCodes[which.min(distance)])
}
```

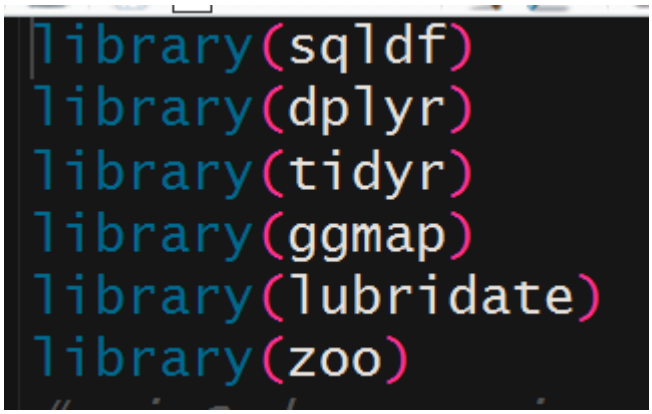
6. GET WEATHER DATA

We are using an R script for generating the weather data which computes the weather data of the nearest airports. The function works on unique airport values, so we basically pass the unique airport values to the function which provides us with csv's for individual data as well as the combined data of all the airports weather.

```
#Create Weather Data
source("WeatherScript.R")
uniqAirport = unique(addData$nearestAiport)
weatherData = createWeatherData(uniqAirport)
```

6.1 WeatherScript.R

Libraries Used:



Function createWeatherData consists of the following:

- ✓ Create a directory named "Weather Data", which will store the outputs
- ✓ Fetch weather for specific dates for a particular Airport
- ✓ Setting Wind_Speed = 0 for "Calm"
- ✓ Handling Outliers values
- ✓ Handling NA values
- ✓ Converting character to numeric
- ✓ Summarizing it for Consumption so as to get Hourly Data
- ✓ Return the clean data-frame

```
#uniqCodes = unique(addData$nearestAirport)
createWeatherData = function(uniqCodes){
  dir.create("Weather Data")
  setwd("Weather Data")
  for (i in uniqCodes) {
    dfw_wx = getWeatherForDate(i, start_date="2013/01/01", end_date="2013/12/08",opt_detailed=TRUE,
                               opt_custom_columns=TRUE, custom_columns=c(2:13))
    #write.csv(dfw_wx,"Weather Data.csv",row.names = FALSE)
    #New variables added =  Gust_SpeedMPH PrecipitationIn Events

    #---------------Setting the Wind_Speed = 0 for "Calm"           ---------------
    dfw_wx$Wind_SpeedMPH=ifelse(dfw_wx$Wind_SpeedMPH=='Calm',0,dfw_wx$Wind_SpeedMPH)

    #---------------Converting Wind_Speed and Humidity to numeric   ---------------
    dfw_wx$Wind_SpeedMPH=as.numeric(dfw_wx$Wind_SpeedMPH)
    dfw_wx$Humidity=as.numeric(dfw_wx$Humidity)

    #---------------Handling Outliers Values                        ---------------
    dfw_wx = dfw_wx %>% mutate(TemperatureF = ifelse(TemperatureF < -126 | TemperatureF > 136,NA,TemperatureF),
                               Dew_PointF = ifelse(Dew_PointF < -60 | Dew_PointF > 95,NA,Dew_PointF),
                               Humidity = ifelse(Humidity<0,NA,Humidity),
                               Sea_Level_PressureIn = ifelse(Sea_Level_PressureIn > 35 | Sea_Level_PressureIn <10,NA,Sea_Level_PressureIn),
                               VisibilityMPH = ifelse(VisibilityMPH > 10 | VisibilityMPH < -10,NA,VisibilityMPH),
                               Wind_SpeedMPH = ifelse(Wind_SpeedMPH > 1000 | Wind_SpeedMPH < 0,NA,Wind_SpeedMPH),
                               PrecipitationIn = ifelse(PrecipitationIn == 'NA',0,PrecipitationIn),
                               Gust_SpeedMPH = ifelse(Gust_SpeedMPH > 253 | Gust_SpeedMPH < 0,NA,Gust_SpeedMPH))
    gust = dfw_wx$Gust_SpeedMPH
    dfw_wx = dfw_wx[,-9]
    dfw_wx = dfw_wx %>% do(na.locf(.))
    dfw_wx$Gust_SpeedMPH = gust
    dfw_wx = dfw_wx %>% separate(Time,c("Date","Time_Stamp"),sep=" ")
    dfw_wx= dfw_wx %>% separate(Time_Stamp,c("Hours"),sep=":")

    #---------------Converting chr to numeric   ---------------
    dfw_wx$Wind_SpeedMPH=as.numeric(dfw_wx$Wind_SpeedMPH)
    dfw_wx$Humidity=as.numeric(dfw_wx$Humidity)
    dfw_wx$TemperatureF=as.numeric(dfw_wx$TemperatureF)
    dfw_wx$Dew_PointF=as.numeric(dfw_wx$Dew_PointF)
    dfw_wx$Sea_Level_PressureIn=as.numeric(dfw_wx$Sea_Level_PressureIn)
    dfw_wx$VisibilityMPH=as.numeric(dfw_wx$VisibilityMPH)
    dfw_wx$Hours=as.numeric(dfw_wx$Hours)
    dfw_wx$WindDirDegrees = as.numeric(dfw_wx$WindDirDegrees)
    dfw_wx$Gust_SpeedMPH = as.numeric(dfw_wx$Gust_SpeedMPH)
```

```
    dfw_wx1 = summarise(group_by(dfw_wx,Date,Hours),TemperatureF=mean(TemperatureF),Dew_PointF=mean(Dew_PointF),
                        Humidity=mean(Humidity),Sea_Level_PressureIn=mean(Sea_Level_PressureIn),
                        VisibilityMPH=mean(VisibilityMPH),Wind_Direction=max(Wind_Direction),
                        Wind_SpeedMPH=mean(Wind_SpeedMPH),Conditions=max(Conditions),
                        WindDirDegrees=mean(WindDirDegrees),Gust_SpeedMPH = mean(Gust_SpeedMPH),
                        PrecipitationIn = max(PrecipitationIn),Events = max(Events))
    dfw_wx1$Date = as.Date(dfw_wx1$Date)
    dfw_wx1$nearestAirport = i
    wName = paste("Weather Data-",i,".csv",sep="")
    write.csv(dfw_wx1,wName,row.names = FALSE)
  }
```

Handling Missing Time-Sequence Series:

- ✓ We are completing the time sequence as there are hour which are missing and will cause model to predict inaccurate results.
- ✓ We are creating a time-sequence series for the time duration we have fetched the weather data.
- ✓ Once we have the complete time series data frame, we inner join it with the original fetched data and get missing records with NA values filled
- ✓ We replace the NA values with the above records, thus completing the data sequence.
- ✓ Once the data sequence is completed we save it in respective csv for distinct airports.

```
#Creating a time sequence for missing hour calculations
timeData = data.frame(seq(as.POSIXct("2013-01-01 00:00:00"), as.POSIXct("2013-12-08 23:59:59"), by="hour"))
names(timeData)[1] = "DateTime"
timeData = timeData %>% separate(DateTime,c("Date","Time"),11)
timeData = timeData %>% separate(Time,"Hours",2)
timeData = timeData[-3]
timeData$Hours = as.numeric(timeData$Hours)
timeData$Date = as.Date(as.character(timeData$Date),format = "%Y-%m-%d")

for(i in uniqCodes){
  lnk = paste("Weather Data-",i,".csv",sep = "")
  flkn = paste("Final Weather Data-",i,".csv",sep="")
  newData = read.csv(lnk,header = TRUE)
  newData$Date = as.Date(as.character(newData$Date),format = "%Y-%m-%d")
  finalData = full_join(timeData,newData,by=c("Date","Hours"))
  finalData = finalData %>% do(na.locf(.))
  finalData$nearestAirport = i
  write.csv(finalData,flkn,row.names = FALSE)
}
```

Creating a Merge Weather Data for all Airports:

- ✓ We are merging all the weather data and putting into one File and saving the csv and also return the data frame to the main script for further functions.

```
#Create one final weather Data for all airports
countVar = 0
for(i in uniqCodes){
  finalLink = paste("Final Weather Data-",i,".csv",sep="")
  openFinal = read.csv(finalLink, header = TRUE)
  if (countVar == 0) {
    finalWeatherDF = openFinal
    countVar = countVar + 1
  }
  else{
    finalWeatherDF = rbind(finalWeatherDF,openFinal)
    countVar = countVar + 1
  }
}
return(finalWeatherDF)
write.csv(finalWeatherDF,"Final-Weather.csv",row.names = FALSE)
setwd("../")
}
```

7. FIRST MERGE:

We have performed an inner join between the address data and weather data(which we got from the function).

```
#First Merge = Weather Data + New Address Data on (nearestAirport)
inn = inner_join(addrsData,weatherData,by = "nearestAirport")
inn$Date = as.Date(as.character(inn$Date),format = "%Y-%m-%d")
setwd("../")
```

Output:

address	area_floor_m.sqr	lon	lat	vac	nearestAirport	Date	Hours	TemperatureF	Dew_PointF	Humidity	Sea_Level_PressureIn	VisibilityMPH	Wind_Direction	Wind_SpeedMPH	Conditions	WindDirDegrees	Gust_SpeedMPH	PrecipitationIn	Events
Metsahovintie 113	110	24.39346	60.21869	Building 1	EFHK	2013-01-01	0	35.6000000	33.8000000	93.00000	29.42000	2.8000000	SSE	13.850000	Light Rain	160.00000	NA	N/A	Rain
Metsahovintie 113	110	24.39346	60.21869	Building 1	EFHK	2013-01-01	1	33.8000000	32.9000000	96.50000	29.40500	2.2000000	SSE	15.550000	Rain	145.00000	NA	N/A	Rain
Metsahovintie 113	110	24.39346	60.21869	Building 1	EFHK	2013-01-01	2	33.8666667	33.5333333	98.33333	29.39000	3.3333333	SSE	15.966667	Light Snow	153.33333	NA	N/A	Snow
Metsahovintie 113	110	24.39346	60.21869	Building 1	EFHK	2013-01-01	3	33.8000000	33.8000000	100.00000	29.36000	4.6500000	SSE	14.400000	Light Rain	150.00000	NA	N/A	Rain-Snow
Metsahovintie 113	110	24.39346	60.21869	Building 1	EFHK	2013-01-01	4	35.6000000	33.8000000	93.00000	29.33000	5.6000000	SSE	13.250000	Light Rain	150.00000	NA	N/A	Rain
Metsahovintie 113	110	24.39346	60.21869	Building 1	EFHK	2013-01-01	5	36.5000000	35.6000000	96.50000	29.31500	6.2000000	SSE	10.950000	Light Rain	165.00000	NA	N/A	Rain
Metsahovintie 113	110	24.39346	60.21869	Building 1	EFHK	2013-01-01	6	37.4000000	36.5000000	96.50000	29.30000	6.2000000	South	13.800000	Light Rain	180.00000	NA	N/A	Rain
Metsahovintie 113	110	24.39346	60.21869	Building 1	EFHK	2013-01-01	7	37.4000000	35.6000000	93.00000	29.30000	6.2000000	South	13.850000	Light Rain	180.00000	NA	N/A	Rain
Metsahovintie 113	110	24.39346	60.21869	Building 1	EFHK	2013-01-01	8	37.2666667	36.0666667	94.66667	29.30667	5.1000000	South	14.833333	Light Rain	183.33333	NA	N/A	Rain
Metsahovintie 113	110	24.39346	60.21869	Building 1	EFHK	2013-01-01	9	37.4000000	35.6000000	93.00000	29.30000	3.7000000	South	13.250000	Rain	180.00000	NA	N/A	Rain
Metsahovintie 113	110	24.39346	60.21869	Building 1	EFHK	2013-01-01	10	37.4000000	36.5000000	96.50000	29.30000	4.6500000	South	16.700000	Light Rain	190.00000	NA	N/A	Rain
Metsahovintie 113	110	24.39346	60.21869	Building 1	EFHK	2013-01-01	11	37.4000000	36.5000000	96.50000	29.30000	4.3500000	South	14.950000	Rain	190.00000	NA	N/A	Rain
Metsahovintie 113	110	24.39346	60.21869	Building 1	EFHK	2013-01-01	12	37.4000000	35.6000000	93.00000	29.30000	4.9500000	South	16.150000	Rain	190.00000	NA	N/A	Rain
Metsahovintie 113	110	24.39346	60.21869	Building 1	EFHK	2013-01-01	13	37.4000000	35.6000000	93.00000	29.30000	3.4000000	South	17.250000	Light Rain	190.00000	NA	N/A	Rain
Metsahovintie 113	110	24.39346	60.21869	Building 1	EFHK	2013-01-01	14	37.2666667	36.3333333	96.66667	29.33000	2.7000000	SSW	15.966667	Light Rain	196.66667	NA	N/A	Rain
Metsahovintie 113	110	24.39346	60.21869	Building 1	EFHK	2013-01-01	15	37.4000000	36.5000000	96.50000	29.33000	1.8000000	South	12.700000	Drizzle	190.00000	NA	N/A	
Metsahovintie 113	110	24.39346	60.21869	Building 1	EFHK	2013-01-01	16	37.4000000	37.4000000	100.00000	29.33000	2.3500000	South	12.100000	Drizzle	190.00000	NA	N/A	
Metsahovintie 113	110	24.39346	60.21869	Building 1	EFHK	2013-01-01	17	37.4000000	37.4000000	100.00000	29.33000	2.8000000	SSW	12.700000	Light Drizzle	195.00000	NA	N/A	
Metsahovintie 113	110	24.39346	60.21869	Building 1	EFHK	2013-01-01	18	37.4000000	37.4000000	100.00000	29.34500	1.7500000	SSW	12.100000	Mostly Cloudy	200.00000	NA	N/A	
Metsahovintie 113	110	24.39346	60.21869	Building 1	EFHK	2013-01-01	19	37.4000000	37.4000000	100.00000	29.36000	4.6500000	SSW	9.200000	Mostly Cloudy	210.00000	NA	N/A	
Metsahovintie 113	110	24.39346	60.21869	Building 1	EFHK	2013-01-01	20	37.2666667	36.0666667	95.00000	29.36667	5.4000000	SW	9.866667	Mostly Cloudy	206.66667	NA	N/A	
Metsahovintie 113	110	24.39346	60.21869	Building 1	EFHK	2013-01-01	21	37.4000000	35.6000000	93.00000	29.39000	4.9500000	SSW	9.250000	Mostly Cloudy	205.00000	NA	N/A	
Metsahovintie 113	110	24.39346	60.21869	Building 1	EFHK	2013-01-01	22	37.4000000	35.6000000	93.00000	29.39000	4.3000000	SSW	9.800000	Mostly Cloudy	200.00000	NA	N/A	
Metsahovintie 113	110	24.39346	60.21869	Building 1	EFHK	2013-01-01	23	37.4000000	35.6000000	93.00000	29.42000	3.4000000	SSW	10.400000	Mostly Cloudy	195.00000	NA	N/A	
Metsahovintie 113	110	24.39346	60.21869	Building 1	EFHK	2013-01-02	0	36.5000000	35.6000000	96.50000	29.42000	1.2000000	South	9.250000	Mostly Cloudy	190.00000	NA	N/A	Fog
Metsahovintie 113	110	24.39346	60.21869	Building 1	EFHK	2013-01-02	1	35.6000000	35.6000000	100.00000	29.42000	1.8500000	SSW	8.650000	Mostly Cloudy	195.00000	NA	N/A	
Metsahovintie 113	110	24.39346	60.21869	Building 1	EFHK	2013-01-02	2	36.9333333	35.4000000	94.00000	29.44333	3.3333333	SSW	8.333333	Mostly Cloudy	193.33333	NA	N/A	
Metsahovintie 113	110	24.39346	60.21869	Building 1	EFHK	2013-01-02	3	35.6000000	35.6000000	100.00000	29.44000	1.2500000	South	6.900000	Mostly Cloudy	175.00000	NA	N/A	
Metsahovintie 113	110	24.39346	60.21869	Building 1	EFHK	2013-01-02	4	35.6000000	35.6000000	100.00000	29.44000	0.3500000	South	7.500000	Fog	180.00000	NA	N/A	Fog
Metsahovintie 113	110	24.39346	60.21869	Building 1	EFHK	2013-01-02	5	35.6000000	35.6000000	100.00000	29.44000	0.3000000	South	7.500000	Fog	190.00000	NA	N/A	Fog
Metsahovintie 113	110	24.39346	60.21869	Building 1	EFHK	2013-01-02	6	35.6000000	35.6000000	100.00000	29.44000	0.2000000	South	7.500000	Fog	180.00000	NA	N/A	Fog
Metsahovintie 113	110	24.39346	60.21869	Building 1	EFHK	2013-01-02	7	35.6000000	35.6000000	100.00000	29.47000	0.4000000	South	8.100000	Fog	185.00000	NA	N/A	Fog
Metsahovintie 113	110	24.39346	60.21869	Building 1	EFHK	2013-01-02	8	35.4000000	34.8000000	97.33333	29.49000	2.0666667	SSW	9.133333	Light Rain	210.00000	NA	N/A	Rain
Metsahovintie 113	110	24.39346	60.21869	Building 1	EFHK	2013-01-02	9	34.7000000	33.8000000	96.50000	29.50000	5.9000000	SW	5.800000	Light Rain	215.00000	NA	N/A	Rain
Metsahovintie 113	110	24.39346	60.21869	Building 1	EFHK	2013-01-02	10	34.7000000	33.8000000	96.50000	29.50000	6.2000000	SSW	6.350000	Mostly Cloudy	205.00000	NA	N/A	Rain
Metsahovintie 113	110	24.39346	60.21869	Building 1	EFHK	2013-01-02	11	35.6000000	32.9000000	90.00000	29.53000	6.2000000	WSW	9.800000	Mostly Cloudy	235.00000	NA	N/A	
Metsahovintie 113	110	24.39346	60.21869	Building 1	EFHK	2013-01-02	12	35.6000000	32.9000000	90.00000	29.53000	6.2000000	SW	6.900000	Light Rain	225.00000	NA	N/A	Rain
Metsahovintie 113	110	24.39346	60.21869	Building 1	EFHK	2013-01-02	13	35.6000000	32.0000000	87.00000	29.53000	6.2000000	SW	8.650000	Mostly Cloudy	220.00000	NA	N/A	
Metsahovintie 113	110	24.39346	60.21869	Building 1	EFHK	2013-01-02	14	34.8000000	32.3333333	90.66667	29.56333	6.2000000	SW	9.500000	Partly Cloudy	220.00000	NA	N/A	Rain
Metsahovintie 113	110	24.39346	60.21869	Building 1	EFHK	2013-01-02	15	33.8000000	32.0000000	93.00000	29.56000	6.2000000	SW	8.650000	Mostly Cloudy	215.00000	NA	N/A	
Metsahovintie 113	110	24.39346	60.21869	Building 1	EFHK	2013-01-02	16	33.8000000	32.0000000	93.00000	29.57500	6.2000000	SW	8.100000	Mostly Cloudy	215.00000	NA	N/A	
Metsahovintie 113	110	24.39346	60.21869	Building 1	EFHK	2013-01-02	17	33.8000000	32.0000000	93.00000	29.59000	6.2000000	SSW	8.650000	Mostly Cloudy	210.00000	NA	N/A	
Metsahovintie 113	110	24.39346	60.21869	Building 1	EFHK	2013-01-02	18	33.8000000	32.0000000	93.00000	29.59000	6.2000000	SW	8.050000	Mostly Cloudy	215.00000	NA	N/A	
Metsahovintie 113	110	24.39346	60.21869	Building 1	EFHK	2013-01-02	19	33.8000000	32.0000000	93.00000	29.59000	6.2000000	SSW	8.100000	Mostly Cloudy	205.00000	NA	N/A	

8. SECOND MERGE:

In the second merge we are creating a Final Data and 78 different model for prediction and classification. We are using a separate R script which has three functions:

- ✓ getFilteredRaw()
- ✓ baseHourCalculations()
- ✓ createSeperateData()

```
#Second Merge = First Merge + Weather Data
source("BuildingSeperation.R")
filterData = getFilteredRaw(finland_Data)
names(filterData)[5] = "Date"
names(filterData)[6] = "Hours"
#2nd Merge
finalMergeData = inner_join(filterData,inn,by = c('vac',"Date","Hours"))
finalMergeData = finalMergeData %>% mutate(Consumption = Consumption/area_floor._m.sqr)
#Calculate Base_Hour_Class
finMer = baseHourCalculation(finalMergeData)
query = sqldf("Select DISTINCT BuildingID,BuildingName,MeterID,type from pF")
#Seperating Buidlgin Data
createSeparateData(query,pF)
write.csv(pF,"Final_Finland_Data.csv",row.names = FALSE)
queryTest = sqldf("Select DISTINCT Conditions from pF")
```

Libraries used by the R script:

```
library(sqldf)
library(dplyr)
library(tidyr)
library(RCurl)
library(XML)
library(rvest)
library(data.table)
```

8.1 getFilteredRaw():

- Functions takes a rawData previously fetched, filter the data for electricity and distributed heating.
- Adds data for missing Building-9 and Building-27
- Creates all the columns for WeekDay, Date, Day, Months, Holiday

```
#Function which sends the filtered raw data
getFilteredRaw = function(rawData1){
  #rawData1 = read.csv(file="Finland_masked.csv",header = TRUE)
  #Selecting type = 'elect' and 'Dist_Heating'
  target = c("elect", "Dist_Heating")
  rawData1 = filter(rawData1,type %in% target)
  rawData1 = rawData1 %>% mutate(vac = ifelse(BuildingID == 81909,"Building 27",
                                              ifelse(BuildingID == 82254 |
                                                       BuildingID == 83427 |
                                                       BuildingID == 84681,
                                                     "Building 9",as.character(vac))))
  #Converting date to Date Format
  rawData1$date = as.Date(as.character(rawData1$date), format='%Y%m%d')
  #week of day
  rawData1$day <- weekdays(as.Date(rawData1$date))
  #Month of year
  rawData1$month <- months(as.Date(rawData1$date))
  #weekdays_weekend
  rawData1=rawData1 %>% mutate(Weekday = ifelse(day == 'Saturday'| day == 'Sunday',0,1))
  rawData1=rawData1 %>%mutate(Base_Hour_Flag= ifelse(hour == '0'|hour == '1' |
                                                       hour == '2'|hour == '3' |
                                                       hour == '4'|hour == '22'|
                                                       hour == '23',1,0))
  #holidayfinalscrape
  url<- 'http://www.timeanddate.com/holidays/finland/2013'
  webpage = read_html(url) %>%
    html_nodes(xpath='/html/body/div[1]/div[7]/article/section[1]/div[5]/table') %>%
    html_table()
  dataframe<-as.data.frame(webpage)
  col<- dataframe[-1,]
  date<-col$Date
  fullDate = paste("2013",date,sep=" ")
  col$Date = as.Date(as.character(fullDate), format='%Y %b %d')
  rawData1 = rawData1 %>% mutate(Holiday = ifelse(date %in% col$Date,1,0))

  return(rawData1)
}
```

8.2 baseHourCalculations():

- This function calculates the base_hour_class variable by aggregating the consumption based on (Building Name, type, weekday, month, holiday, date)
- Adding it to the final data

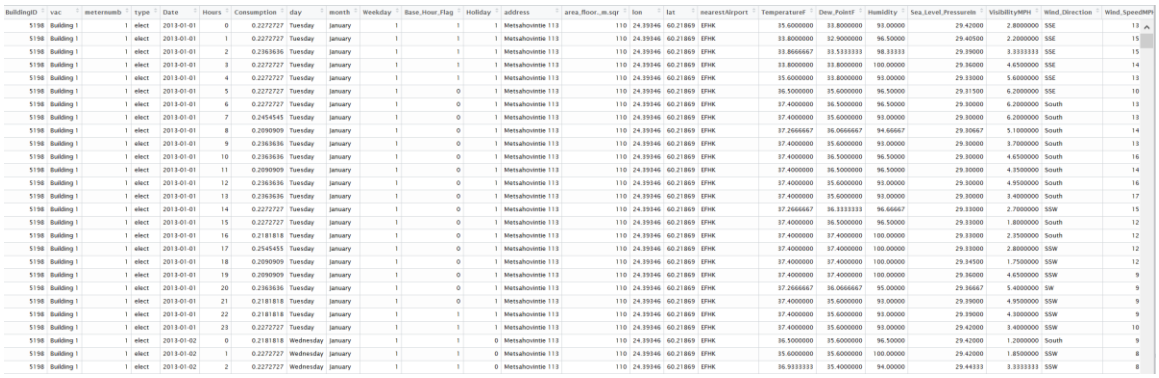
```
#Function for base hour calculations:
baseHourCalculation = function(finRawData){
  finRawData = finalMergeData
  df2 = aggregate(Consumption~vac+type+Weekday+month+Holiday+Date,
                  mean,
                  data=subset(finRawData,Hours<3 | Hours>22 ))
  names(df2)[7]="base_hr_usage"
  p1 = inner_join(finRawData,df2,by = c("vac", "type", "Weekday","Date","month", "Holiday"))
  View(p1)
  p1=p1 %>% mutate(Base_Hour_Class= ifelse(Consumption> base_hr_usage,"High","Low"))
  pF = subset(p1, select = -c(lat,lon,area_floor._m.sqr))
  names(p1)[2]="BuildingName"
  names(p1)[3]="MeterID"
}
```

8.3 createSeperateData():

- We pass the query which has unique 78 records, to this function and the final data returned from the previous function.
- We loop through the length of the query, and create a unique name for each file into a new directory created inside this function.

```
#Function for creating seperate 78 or any number of files as per the Raw Data
createSeparateData = function(que,finData){
  dir.create("Buildings Data")
  setwd("Buildings Data")
  lengQuery = length(que$BuildingID)
  for (varb in 1:lengQuery){
    name = paste(que$BuildingName[varb],que$BuildingID[varb],que$MeterID[varb],que$type[varb],sep = "_")
    bdng = paste(name,".csv",sep="")
    sepData = filter(finData,BuildingName %in% que$BuildingName[varb],
                     BuildingID %in% que$BuildingID[varb],
                     MeterID  %in% que$MeterID[varb],
                     type %in% que$type[varb])
    write.csv(sepData,bdng,row.names = FALSE)
  }
  setwd("../")
}
```

Final Data:



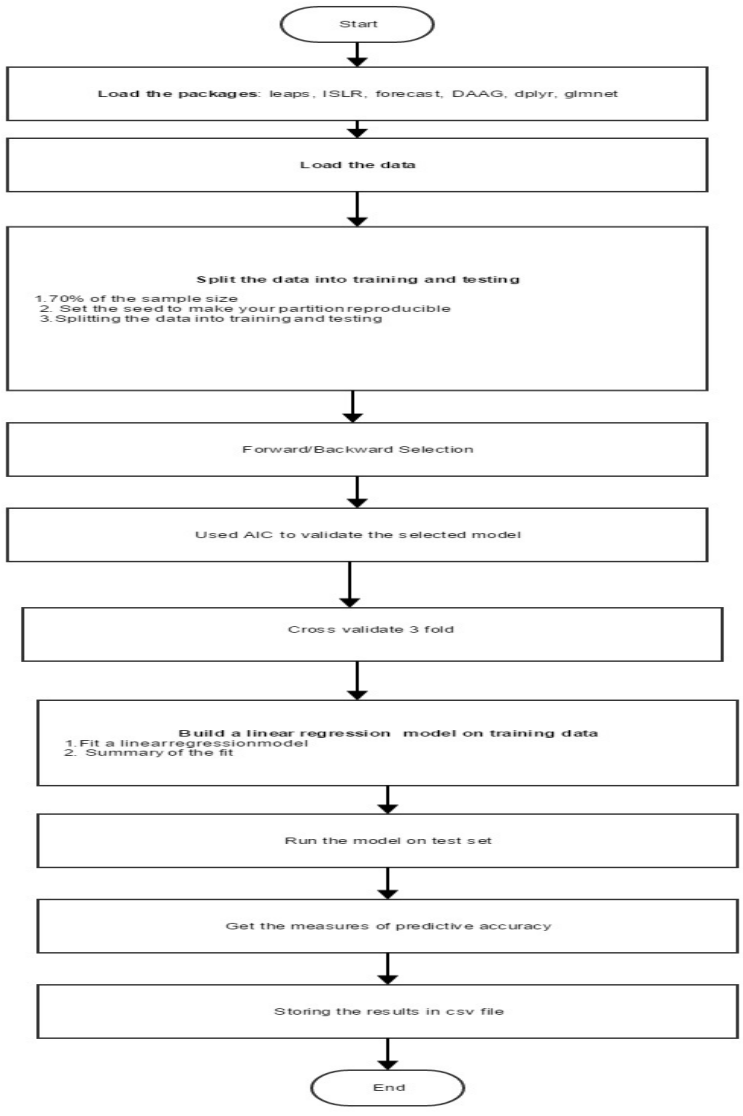
2. Part- II Data Modelling

A. Prediction
B. Classification

PREDICTION-REGRESSION (FOR ALL BUILDINGS DATA)

Build Regression, KNN, Random Forest and Neural Network models for all buildings as one dataset. Optimize your model through feature engineering and model selection.

Process Flow:



Regression model for all buildings as one dataset

1. Libraries used:

```
#Libraries used
library(MASS)
library(ISLR)
library(dplyr)
library(forecast)
library(leaps)
library(caret)
library(glmnet)
library(DAAG)
library(sqldf)
```

2. Read the file and remove unnecessary variables

```
#read file
RegModellingData <- read.csv(file="Final_Finland_Data.csv",header=TRUE,sep=",")
RegModellingData<- RegModellingData %>% select(-c(BuildingID,BuildingName,MeterID,
                                                  type,Date,nearestAirport,Wind_Direction,
                                                  Conditions,Gust_SpeedMPH,PrecipitationIn,
                                                  Events,VisibilityMPH,address,Base_Hour_Class,
                                                  day,month))
```

3. Perform Scaling

```
#Scaling
RegModellingData$TemperatureF = scale(as.numeric(RegModellingData$TemperatureF),
                                      center = TRUE,scale = TRUE)
RegModellingData$Dew_PointF = scale(as.numeric(RegModellingData$Dew_PointF),
                                    center = TRUE,scale = TRUE)
RegModellingData$Humidity = scale(as.numeric(RegModellingData$Humidity),
                                  center = TRUE,scale = TRUE)
RegModellingData$Sea_Level_PressureIn = scale(as.numeric(RegModellingData$Sea_Level_PressureIn),
                                              center = TRUE,scale = TRUE)
RegModellingData$Wind_SpeedMPH = scale(as.numeric(RegModellingData$Wind_SpeedMPH),
                                       center = TRUE,scale = TRUE)
RegModellingData$WindDirDegrees = scale(as.numeric(RegModellingData$WindDirDegrees),
                                        center = TRUE,scale = TRUE)
RegModellingData$DayNumber = scale(as.numeric(RegModellingData$DayNumber),
                                   center = TRUE,scale = TRUE)
RegModellingData$MonthNumber = scale(as.numeric(RegModellingData$MonthNumber),
                                     center = TRUE,scale = TRUE)
RegModellingData$base_hr_usage = scale(as.numeric(RegModellingData$base_hr_usage),
                                       center = TRUE,scale = TRUE)
RegModellingData$WeekDay = scale(as.numeric(RegModellingData$Weekday),
                                 center = TRUE,scale = TRUE)
RegModellingData$Base_Hour_Flag = scale(as.numeric(RegModellingData$Base_Hour_Flag),
                                        center = TRUE,scale = TRUE)
```

4. Split data into training and testing

First take 70% of the sample size of the data.
Then set the seed to make partition reproducible.
Then split this data into training and testing respectively

```
#70% of the sample size
smp_size <- floor(0.70 * nrow(RegModellingData))

#Set the seed to make your partition reproducible.
set.seed(123)
train_ind <- sample(seq_len(nrow(RegModellingData)), size = smp_size)

#Splitting the data into training and testing
train_data <- RegModellingData[train_ind, ]
names(train_data)
test_data <- RegModellingData[-train_ind, ]
```

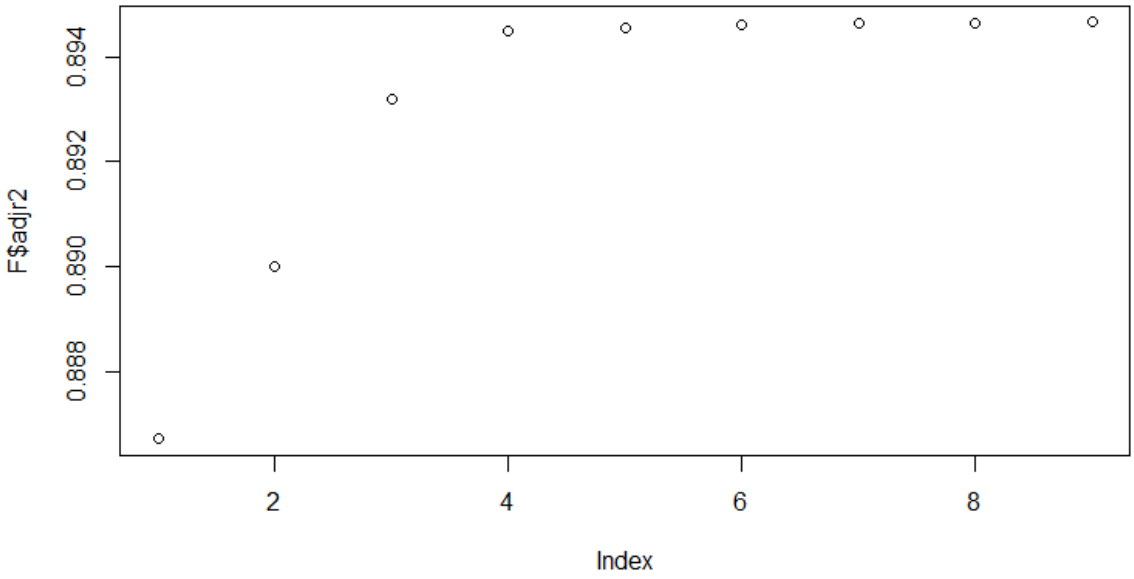
5. Forward Selection

Based on the adjr2 graph and checking the adjr2 values we chose the 9th model as below

```
#Forward Selection
regfit.fwd=regsubsets(Consumption ~ Weekday+Base_Hour_Flag+Holiday+TemperatureF+
                        Humidity+Sea_Level_PressureIn+Wind_SpeedMPH+
                        WindDirDegrees+base_hr_usage, data=train_data, nvmax=11, method="forward")
F=summary(regfit.fwd)
names(F)
F
plot(regfit.fwd,scale = "adjr2")
plot(F$rss)
F$rss
F$adjr2
plot(F$adjr2)
coef(regfit.fwd,9)
```

Output:

a. Graph for adjr2



b. Output values of Adrj2

```
> F$adjr2
[1] 0.8867072 0.8899982 0.8932019 0.8944821 0.8945492 0.8946021 0.8946446 0.8946532 0.8946589
> plot(F$adjr2)
> coef(regfit.fwd,9)
         (Intercept)             Weekday       Base_Hour_Flag             Holiday        TemperatureF
            Humidity
       1.450714e-02        3.877582e-03        -1.916691e-03       -1.197687e-03       -1.079736e-03
     3.294586e-04
Sea_Level_PressureIn       Wind_SpeedMPH       WindDirDegrees       base_hr_usage
       8.695648e-05        2.929689e-04        -9.896702e-05        2.910662e-02
```

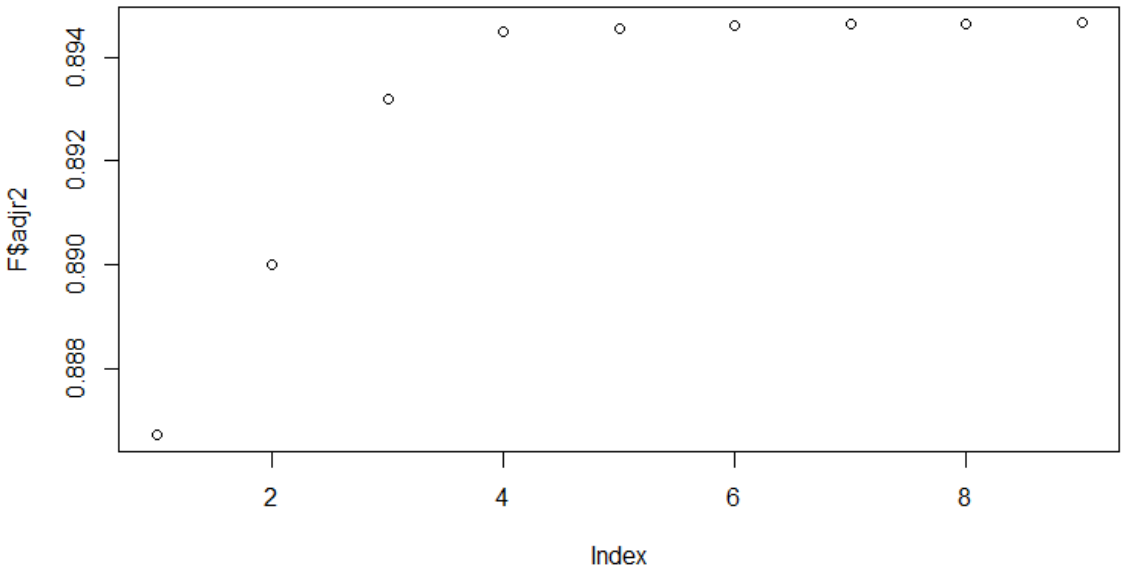
9. Backward Selection

Based on the adjr2 graph and checking the adjr2 values we chose the 9th model as below

```
#Backward Selection
regfit.bwd=regsubsets(Consumption ~ Weekday+Base_Hour_Flag+Holiday+TemperatureF+
                        Humidity+Sea_Level_PressureIn+
                        Wind_SpeedMPH+WindDirDegrees+base_hr_usage,
                      data=train_data, nvmax=11, method="backward")
F=summary(regfit.bwd)
names(F)
F
plot(F$rss)
plot(F$adjr2)
F$adjr2
coef(regfit.bwd,9)
```

Output:

a. Graph for adjr2



b. Output values of Adrj2

```
> plot(F$adjr2)
> F$adjr2
[1] 0.8867072 0.8899982 0.8932019 0.8944821 0.8945492 0.8946021 0.8946446 0.8946532 0.8946589
> coef(regfit.bwd,9)
         (Intercept)             Weekday       Base_Hour_Flag             Holiday        TemperatureF
            Humidity
       1.450714e-02        3.877582e-03       -1.916691e-03       -1.197687e-03       -1.079736e-03
      3.294586e-04
Sea_Level_PressureIn        Wind_SpeedMPH        WindDirDegrees        base_hr_usage
       8.695648e-05        2.929689e-04       -9.896702e-05        2.910662e-02
>
```

10. Fit a Linear Regression Model

```
#Fit a linear regression model
lm.fit=lm(Consumption ~ Weekday+Base_Hour_Flag+Holiday+TemperatureF+
            Humidity+Sea_Level_PressureIn+Wind_SpeedMPH+
           WindDirDegrees+base_hr_usage,data=train_data)
```

11. Step AIC to validate the selected model

```
#AIC
step <- stepAIC(lm.fit, direction="both")
step$anova
```

```
> step$anova
Stepwise Model Path
Analysis of Deviance Table

Initial Model:
Consumption ~ Weekday + Base_Hour_Flag + Holiday + TemperatureF +
    Humidity + Sea_Level_PressureIn + Wind_SpeedMPH + WindDirDegrees +
    base_hr_usage

Final Model:
Consumption ~ Weekday + Base_Hour_Flag + Holiday + TemperatureF +
    Humidity + Sea_Level_PressureIn + Wind_SpeedMPH + WindDirDegrees +
    base_hr_usage


  Step Df Deviance Resid. Df Resid. Dev      AIC
1                     435261   44.39755 -4000355
```

12. Perform 3 fold cross validation to confirm the model

```
#Cross validate 3 fold
CVlm(RegModellingData, lm.fit, m=3)
```

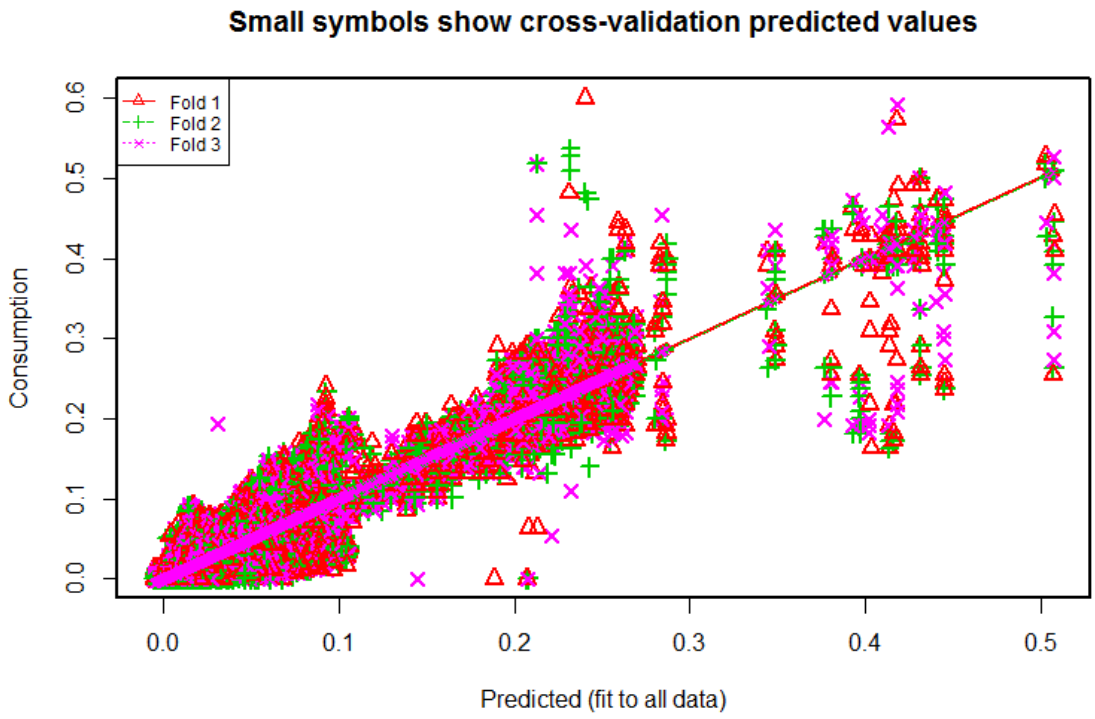
Output:

a. Variance table analysis

```
> CVlm(RegModellingData, lm.fit, m=3)
Analysis of Variance Table

Response: Consumption
                       Df Sum Sq Mean Sq  F value  Pr(>F)
Weekday                 1      2       2 2.22e+04 < 2e-16 ***
Base_Hour_Flag          1      2       2 1.87e+04 < 2e-16 ***
Holiday                 1      0       0 3.67e+02 < 2e-16 ***
TemperatureF            1     23      23 2.22e+05 < 2e-16 ***
Humidity                1      1       1 9.37e+03 < 2e-16 ***
Sea_Level_PressureIn    1      0       0 4.35e+01 4.3e-11 ***
Wind_SpeedMPH           1      2       2 2.41e+04 < 2e-16 ***
WindDirDegrees          1      0       0 1.27e+03 < 2e-16 ***
base_hr_usage           1    510     510 4.98e+06 < 2e-16 ***
Residuals          621807     64       0
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

b. Graph of 3 fold method:

As you can see all the 3 folds are almost overlapping we can assume the model is quite perfect for predictions



13. Performing Prediction and Checking Accuracy

```
#Prediction and check accuracy
reg_pred = predict(lm.fit, test_data)
reg_acc<-accuracy(reg_pred, test_data$Consumption)
rmse = sqrt(mean(mean(test_data$Consumption)-reg_acc)^2)
View(reg_acc)
summary(lm.fit)
```

Output:

a. Residuals

```
> summary(lm.fit)

Call:
lm(formula = Consumption ~ Weekday + Base_Hour_Flag + Holiday +
    TemperatureF + Humidity + Sea_Level_PressureIn + Wind_SpeedMPH +
    WindDirDegrees + base_hr_usage, data = train_data)

Residuals:
     Min       1Q   Median       3Q      Max
-0.25229 -0.00416 -0.00093  0.00286  0.30579
```

b. Co-efficients

```
                         Estimate Std. Error t value Pr(>|t|)
(Intercept)              1.45e-02   2.93e-05  494.50  < 2e-16 ***
Weekday                  3.88e-03   3.42e-05  113.35  < 2e-16 ***
Base_Hour_Flag          -1.92e-03   1.61e-05 -118.78  < 2e-16 ***
Holiday                 -1.20e-03   7.35e-05  -16.29  < 2e-16 ***
TemperatureF            -1.08e-03   1.71e-05  -63.02  < 2e-16 ***
Humidity                 3.29e-04   1.87e-05   17.57  < 2e-16 ***
Sea_Level_PressureIn     8.70e-05   1.75e-05    4.96  7.1e-07 ***
Wind_SpeedMPH            2.93e-04   1.91e-05   15.34  < 2e-16 ***
WindDirDegrees          -9.90e-05   1.66e-05   -5.95  2.6e-09 ***
base_hr_usage            2.91e-02   1.56e-05 1867.18  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.0101 on 435261 degrees of freedom
Multiple R-squared:  0.895,     Adjusted R-squared:  0.895
F-statistic: 4.11e+05 on 9 and 435261 DF,  p-value: <2e-16
```

c. ME, RMSE, MAE,MPE and MAPE

	ME	RMSE	MAE	MPE	MAPE
Test set	-3.19e-05	0.0101	0.00585	NaN	Inf

14. Write to file the required output

```
#Write to csv
write.csv(reg_acc, "Regression_FullData.csv", row.names = FALSE)
```

FINAL COMMENT: As you can see the RMSE value is 0.0101 and Adjusted R-squared value is 0.895 it is a very good model and is giving accurate predictions. Also, the model runs quite fast and isn't time consuming and the number of variables used are also optimal and a good fit.

PREDICTION-REGRESSION (FOR EACH BUILDING DATASET: 78 MODELS)

Build Regression, KNN, Random Forest and Neural Network models for each building dataset (78 different models). Optimize your model through feature engineering and model selection.

STEPS:

Step1: We have passed the query from the main script into this function which has the following data.

Step2: We are calculating the length of a record inside a query.

Step3: We are then running a for loop from 1 to the length of query.

Step4: We have saved the 78 files in a directory called Buildings Data. We are picking each file from that directory through a loop.

```
lengQuery = length(que$BuildingID)
setwd("Buildings Data")
for (varb in 1:lengQuery){

  name = paste(que$BuildingName[varb],que$BuildingID[varb],que$MeterID[varb],que$type[varb],
               sep = "_")
  bdng = paste(name,".csv",sep="")
  modelData = read.csv(bdng,header = TRUE)
  dimen = dim(modelData)
  if(sum(modelData$Consumption != 0)){
    modelData = modelData %>% select(-c(BuildingID,BuildingName,MeterID,
                                        type,Date,nearestAirport,Wind_Direction,
                                        Conditions,Gust_SpeedMPH,PrecipitationIn,
                                        Events,VisibilityMPH,address,Base_Hour_Class,
                                        day,month))
    modelData$TemperatureF = scale(as.numeric(modelData$TemperatureF),
                                   center = TRUE,scale = TRUE)
    modelData$Dew_PointF = scale(as.numeric(modelData$Dew_PointF),
                                 center = TRUE,scale = TRUE)
    modelData$Humidity = scale(as.numeric(modelData$Humidity),
                               center = TRUE,scale = TRUE)
    modelData$Sea_Level_PressureIn = scale(as.numeric(modelData$Sea_Level_PressureIn),
                                           center = TRUE,scale = TRUE)
    modelData$Wind_SpeedMPH = scale(as.numeric(modelData$Wind_SpeedMPH),
                                    center = TRUE,scale = TRUE)
    modelData$WindDirDegrees = scale(as.numeric(modelData$WindDirDegrees),
                                     center = TRUE,scale = TRUE)
    modelData$base_hr_usage = scale(as.numeric(modelData$base_hr_usage),
                                    center = TRUE,scale = TRUE)
    modelData$WeekDay = scale(as.numeric(modelData$Weekday),
                              center = TRUE,scale = TRUE)
    modelData$Base_Hour_Flag = scale(as.numeric(modelData$Base_Hour_Flag),
                                     center = TRUE,scale = TRUE)
```

Below is a snippet of the output of the query:

	BuildingID	BuildingName	MeterID	type
1	5198	Building 1	1	elect
2	5199	Building 2	1	elect
3	5286	Building 3	1	elect
4	5288	Building 4	1	elect
5	5290	Building 5	1	elect
6	5290	Building 5	3	elect
7	5290	Building 5	8	elect
8	5304	Building 6	1	elect
9	5304	Building 6	3	elect
10	5306	Building 7	1	elect
11	5308	Building 8	1	elect
12	5310	Building 5	1	elect
13	5311	Building 10	1	elect
14	5313	Building 11	1	elect
15	5314	Building 12	1	elect
16	5316	Building 13	1	elect
17	5317	Building 14	1	elect
18	5318	Building 14	1	elect
19	5322	Building 15	1	elect
20	5323	Building 15	1	elect
21	5325	Building 16	1	elect
22	5329	Building 17	1	elect
23	5332	Building 18	1	elect
24	5333	Building 19	1	elect
25	5335	Building 20	1	elect
26	5340	Building 21	1	elect
27	5351	Building 22	1	elect
28	5352	Building 23	1	elect
29	5355	Building 24	1	elect
30	5356	Building 25	1	elect

Rest of the code with steps explained as before:

```
     #70% of the sample size
    smp_size <- floor(0.70 * nrow(modelData))

    #Set the seed to make your partition reproducible.
    set.seed(123)
    train_ind <- sample(seq_len(nrow(modelData)), size = smp_size)

    #Splitting the data into training and testing
    train_data <- modelData[train_ind, ]
    test_data <- modelData[-train_ind, ]

    #Fit a linear regression model
    lm.fit=lm(Consumption ~ Weekday+Base_Hour_Flag+Holiday+TemperatureF+
                Humidity+Sea_Level_PressureIn+Wind_SpeedMPH+
                WindDirDegrees+base_hr_usage,
              data=train_data)
    #Cross validate 3 fold
    #CVlm(ModellingData, lm.fit, m=3)

    reg_pred = predict(lm.fit, test_data)
    reg_acc<-accuracy(reg_pred, test_data$Consumption)
    #rmse = sqrt(mean(mean(test_data$Consumption)-reg_acc)^2)

    if(varb == 1){
      rmseApp2 = reg_acc
      modelName = bdng
    }
    else{
      rmseApp2 = rbind(rmseApp2,reg_acc)
      modelName = rbind(modelName,bdng)
    }
  }
}
rmseApp2 = cbind(modelName,rmseApp2)
names(rmseApp2)[1] = "Model"
View(rmseApp2)
write.csv(rmseApp2, "RMSE_ALLBUILDINGS_REGRESSION.csv", row.names=FALSE)
setwd("../")
```

Output:

We are ignoring those Buildings which have Consumption = 0 for the entire time duration, as the prediction model will predict some garbage values for the same thus we are putting a check for each building which checks the Sum of Consumption is greater than 0 or not. Hence we have predictions for 75 buildings.

A snippet of the output:

V1	ME	RMSE	MAE	MPE	MAPE
Building 1_5198_1_elect.csv	0.000110497513391036	0.0319721523653926	0.0197674874366065	-Inf	Inf
Building 2_5199_1_elect.csv	-0.000106398284880906	0.00652820022383435	0.00522181101252042	NaN	Inf
Building 3_5286_1_elect.csv	0.000122643236746833	0.00347731395616386	0.00174424722276869	-Inf	Inf
Building 5_5290_1_elect.csv	-0.0000736873389080059	0.00321667239749738	0.00267815555239868	-Inf	Inf
Building 6_5304_1_elect.csv	-0.0000132376773910107	0.000433630436889738	0.000254715594456766	NaN	Inf
Building 6_5304_3_elect.csv	0.000124458027770123	0.00335808127356514	0.00272402022207911	-Inf	Inf
Building 7_5306_1_elect.csv	-0.000079571173429063	0.00519248182041109	0.00411337179827005	NaN	Inf
Building 8_5308_1_elect.csv	-0.0000332637356792442	0.0024356555035992	0.0019945529324789	NaN	Inf
Building 5_5310_1_elect.csv	-0.0000630034224297773	0.0023838964615769	0.00183965649312041	-Inf	Inf
Building 10_5311_1_elect.csv	-0.0000644966291536986	0.00337180173346923	0.00252762598619943	NaN	Inf
Building 11_5313_1_elect.csv	-0.0000210746627951719	0.00640170184540707	0.00466423012575518	NaN	Inf
Building 12_5314_1_elect.csv	-0.0000266305913258987	0.00757689823567873	0.00623586421005839	NaN	Inf
Building 13_5316_1_elect.csv	0.000230912265605354	0.0043158736737733	0.00290295516396073	NaN	Inf
Building 14_5317_1_elect.csv	-0.000142701589393968	0.00622076889649749	0.00418822334293887	Inf	Inf
Building 14_5318_1_elect.csv	-0.0000346186514099423	0.00205580182872274	0.00161863760051471	-Inf	Inf
Building 15_5322_1_elect.csv	-0.000012608521499393	0.00176204601681365	0.00144769482756685	-Inf	Inf
Building 15_5323_1_elect.csv	0.0000247206554181345	0.000943780831142913	0.000718547492509246	-Inf	Inf
Building 16_5325_1_elect.csv	0.0000792104328416334	0.00298234944781371	0.00239040140029321	NaN	Inf
Building 17_5329_1_elect.csv	-0.000103517272570499	0.00438065910880591	0.00358830508346582	NaN	Inf
Building 18_5332_1_elect.csv	0.0000316592553167302	0.00446661182796251	0.00363465910489541	NaN	Inf
Building 19_5333_1_elect.csv	0.0000187014841341917	0.00118002896621818	0.000905281089411407	-Inf	Inf
Building 20_5335_1_elect.csv	0.000021859614491675	0.00374542717841986	0.00289208103462536	NaN	Inf
Building 21_5340_1_elect.csv	-0.00000155209429764449	0.00414680409837503	0.00328537116505495	NaN	Inf
Building 22_5351_1_elect.csv	-0.0000665093297011488	0.00313476290258017	0.00210176422062405	NaN	Inf
Building 23_5352_1_elect.csv	-0.00000860350061415013	0.00351197319271215	0.00279645578889027	NaN	Inf
Building 24_5355_1_elect.csv	-0.000031145916180319	0.0032521072535466	0.00263444769747327	NaN	Inf
Building 25_5356_1_elect.csv	-0.0000115230671091401	0.00251831775402684	0.0019990838114434	NaN	Inf
Building 26_5357_1_elect.csv	-0.000018889504817993	0.00178366696595657	0.00144051207980578	-Inf	Inf
Building 26_5358_1_elect.csv	-0.0000265894312498702	0.00301789349826221	0.00242429288546395	NaN	Inf
Building 28_28096_1_elect.csv	-0.0000370836627873654	0.00732186955329949	0.00600777032045013	NaN	Inf
Building 28_28096_4_Dist_Heating.csv	-0.000250840733782358	0.0132258734434549	0.0101488436687158	NaN	Inf
Building 29_28108_3_elect.csv	0.0000180250755600782	0.0130114296591345	0.00978766783115272	NaN	Inf

PREDICTION-RANDOM FOREST (FOR ALL BUILDINGS DATA)

1. Random Forest model for all buildings as one dataset.

Random Forest is an ensemble learning based classification and regression technique. In a normal decision tree, one decision tree is built and in a random forest algorithm number of decision trees are built during the process. A vote from each of the decision trees is considered in deciding the final class of a case or an object, this is called ensemble process. This is a democratic process. Since , many decision trees are built and used in a process of Random Forest algorithm, it is called a forest.

1.Libraries used:

```
#libraries used
library(randomForest)
```

2.Read the file and remove unnecessary variables

```
#read file
RFModellingData <- read.csv(file="Final_Finland_Data.csv",header=TRUE,sep=",")
RFModellingData<- RFModellingData %>% select(-c(BuildingID,BuildingName,MeterID,type,Date,
                                                nearestAirport,Wind_Direction,Conditions,
                                                Gust_SpeedMPH,PrecipitationIn,Events,VisibilityMPH,
                                                address,Base_Hour_Class,day,month))
```

3.Perform Scaling

```
#Scaling
RFModellingData$TemperatureF = scale(as.numeric(RFModellingData$TemperatureF),
                                     center = TRUE,scale = TRUE)
RFModellingData$Dew_PointF = scale(as.numeric(RFModellingData$Dew_PointF),
                                   center = TRUE,scale = TRUE)
RFModellingData$Humidity = scale(as.numeric(RFModellingData$Humidity),
                                 center = TRUE,scale = TRUE)
RFModellingData$Sea_Level_PressureIn = scale(as.numeric(RFModellingData$Sea_Level_PressureIn),
                                             center = TRUE,scale = TRUE)
RFModellingData$Wind_SpeedMPH = scale(as.numeric(RFModellingData$Wind_SpeedMPH),
                                      center = TRUE,scale = TRUE)
RFModellingData$WindDirDegrees = scale(as.numeric(RFModellingData$WindDirDegrees),
                                       center = TRUE,scale = TRUE)
RFModellingData$DayNumber = scale(as.numeric(RFModellingData$DayNumber),
                                  center = TRUE,scale = TRUE)
RFModellingData$base_hr_usage = scale(as.numeric(RFModellingData$base_hr_usage),
                                      center = TRUE,scale = TRUE)
RFModellingData$WeekDay = scale(as.numeric(RFModellingData$Weekday),
                                center = TRUE,scale = TRUE)
RFModellingData$Base_Hour_Flag = scale(as.numeric(RFModellingData$Base_Hour_Flag),
                                       center = TRUE,scale = TRUE)
RFModellingData$MonthNumber = scale(as.numeric(RFModellingData$MonthNumber),
                                    center = TRUE,scale = TRUE)
```

4.Split data into training and testing

```
#Split the data into training and testing

#70% of the sample size
smp_size <- floor(0.70 * nrow(RFModellingData))

#Set the seed to make your partition reproducible.
set.seed(123)
train_ind <- sample(seq_len(nrow(RFModellingData)), size = smp_size)

#Splitting the data into training and testing
train_data <- RFModellingData[train_ind, ]
test_data <- RFModellingData[-train_ind, ]
```

5. Build the Random Forest

```
cross.sell.rf <- randomForest(rf.form,
                              train_data,
                              ntree=120,
                              importance=T)
```

15. PLOT A GRAPH.

Plot error rate across decision trees. Once there is not a significant reduction in error rate we choose that number for decision tree. Here we have chosen 120.

```
#plot the graph
plot(cross.sell.rf)
```

16. VARIABLE IMPORTANCE PLOT

It is also a useful tool and can be plotted using varImpPlot function. Top variables are selected and plotted based on Model Accuracy and Gini value. We can also get a table with decreasing order of importance based on a measure.

```
# Variable Importance Table
varImpPlot(cross.sell.rf,
           sort = T,
           main="Variable Importance",
           n.var=10)

var.imp <- data.frame(importance(cross.sell.rf,
                                 type=2))
```

17. PREDICTING RESPONSE VARIABLE USING RANDOM FOREST

```
# Predicting response variable
test_data$predicted.response <- predict(cross.sell.rf ,test_data)
reg_acc<-accuracy(test_data$predicted.response, test_data$Consumption)
rmse = sqrt(mean(mean(test_data$Consumption)-reg_acc)^2)
View(reg_acc)
```

18. WRITE THE REQUIRED OUTPUT TO CSV FILE

```
#Write to csv
write.csv(reg_acc, "RandomForest_FullData.csv", row.names = FALSE)
```

FINAL COMMENTS: It is computationally challenging and time consuming.

PREDICTION-RANDOM FOREST (FOR EACH BUILDING DATASET: 78 MODELS)

STEPS:

Step1: We have passed the query from the main script into this function which has the following data.

Step2: We are calculating the length of a record inside a query.

Step3: We are then running a for loop from 1 to the length of query.

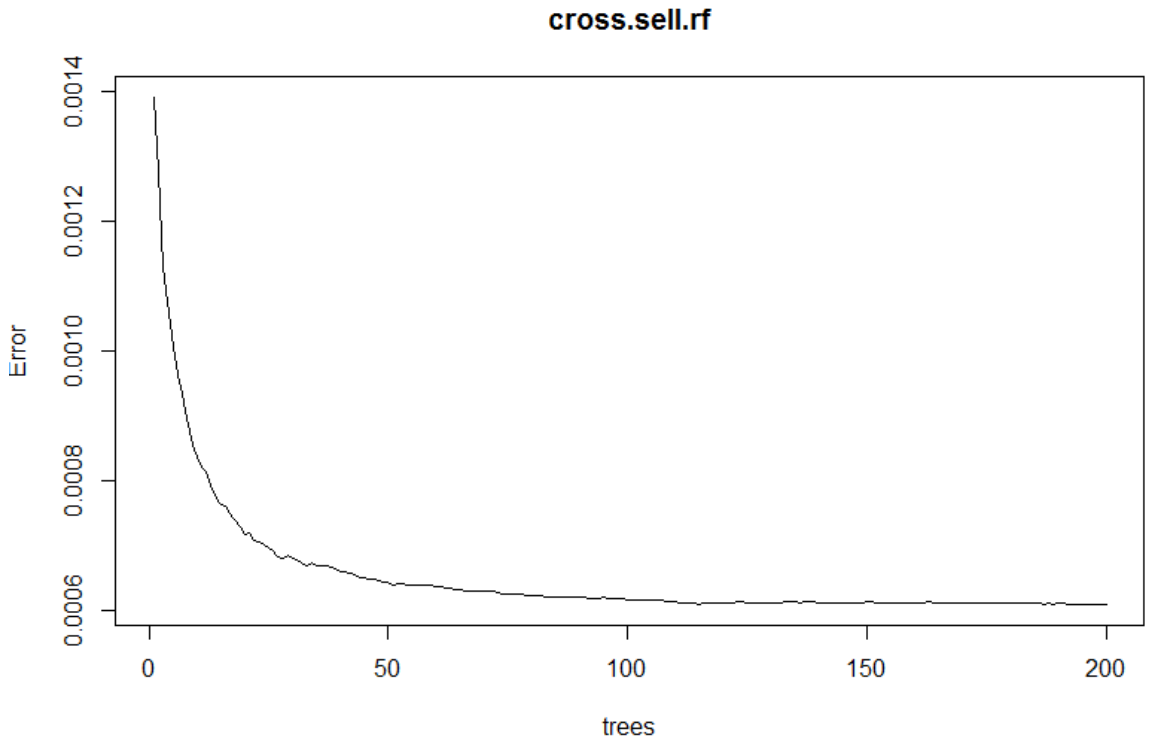
Step4: We have saved the 78 files in a directory called Buildings Data. We are picking each file from that directory through a loop.

Below is a snippet of the output of the query:

```
for (varb in 1:lengQuery){
  name = paste(que$BuildingName[varb],que$BuildingID[varb],que$MeterID[varb],que$type[varb],
               sep = "_")
  bdng = paste(name,".csv",sep="")
  randoModelData = read.csv(bdng,header = TRUE)
  if(sum(randoModelData$Consumption != 0)){
    randoModelData = randoModelData %>% select(-c(BuildingID,BuildingName,MeterID,
                                                  type,Date,nearestAirport,Wind_Direction,
                                                  Conditions,Gust_SpeedMPH,PrecipitationIn,
                                                  Events,VisibilityMPH,address,Base_Hour_Class,
                                                  day,month))
    modelData$TemperatureF = scale(as.numeric(modelData$TemperatureF),
                                   center = TRUE,scale = TRUE)
    modelData$Dew_PointF = scale(as.numeric(modelData$Dew_PointF),
                                 center = TRUE,scale = TRUE)
    modelData$Humidity = scale(as.numeric(modelData$Humidity),
                               center = TRUE,scale = TRUE)
    modelData$Sea_Level_PressureIn = scale(as.numeric(modelData$Sea_Level_PressureIn),
                                           center = TRUE,scale = TRUE)
    modelData$Wind_SpeedMPH = scale(as.numeric(modelData$Wind_SpeedMPH),
                                    center = TRUE,scale = TRUE)
    modelData$WindDirDegrees = scale(as.numeric(modelData$WindDirDegrees),
                                     center = TRUE,scale = TRUE)
    modelData$base_hr_usage = scale(as.numeric(modelData$base_hr_usage),
                                    center = TRUE,scale = TRUE)
    modelData$WeekDay = scale(as.numeric(modelData$Weekday),
                              center = TRUE,scale = TRUE)
    modelData$Base_Hour_Flag = scale(as.numeric(modelData$Base_Hour_Flag),
                                     center = TRUE,scale = TRUE)
```

BuildingID	BuildingName	MeterID	type
5198	Building 1	1	elect
5199	Building 2	1	elect
5286	Building 3	1	elect
5288	Building 4	1	elect
5290	Building 5	1	elect
5290	Building 5	3	elect
5290	Building 5	8	elect
5304	Building 6	1	elect
5304	Building 6	3	elect
5306	Building 7	1	elect
5308	Building 8	1	elect
5310	Building 5	1	elect
5311	Building 10	1	elect
5313	Building 11	1	elect
5314	Building 12	1	elect
5316	Building 13	1	elect
5317	Building 14	1	elect
5318	Building 14	1	elect
5322	Building 15	1	elect
5323	Building 15	1	elect
5325	Building 16	1	elect
5329	Building 17	1	elect
5332	Building 18	1	elect
5333	Building 19	1	elect
5335	Building 20	1	elect
5340	Building 21	1	elect
5351	Building 22	1	elect
5352	Building 23	1	elect
5355	Building 24	1	elect
5356	Building 25	1	elect
5357	Building 26	1	elect
5358	Building 26	1	elect

Plot error rate across decision trees. Once there is not a significant reduction in error rate we choose that number for decision tree. Here we have chosen 120.



REST OF THE CODE WITH STEPS EXPLAINED AS BEFORE:

```
    #70% of the sample size
    smp_size <- floor(0.70 * nrow(randoModelData))

    #Set the seed to make your partition reproducible.
    set.seed(123)
    train_ind <- sample(seq_len(nrow(randoModelData)), size = smp_size)

    #Splitting the data into training and testing
    train_data <- randoModelData[train_ind, ]
    test_data <- randoModelData[-train_ind, ]


    #building Randomforest
    rf.form<-Consumption ~ Weekday+Base_Hour_Flag+Holiday+TemperatureF+
      Humidity+Sea_Level_PressureIn+Wind_SpeedMPH+
      WindDirDegrees+base_hr_usage

    cross.sell.rf <- randomForest(rf.form,
                                  train_data,
                                  ntree=200,
                                  importance=T)

    # Predicting response variable
    test_data$predicted.response <- predict(cross.sell.rf ,test_data)
    reg_acc<-accuracy(test_data$predicted.response, test_data$Consumption)
    rmse = sqrt(mean(mean(test_data$Consumption)-reg_acc)^2)
    if(varb == 1){
      rmseAppRandom1 =  reg_acc
      modelName1 = bdng
    }
    else{
      rmseAppRandom1 = rbind(rmseAppRandom1,reg_acc)
      modelName1 = rbind(modelName1,bdng)
    }
  }
}
rmseAppRandom1 = cbind(modelName1,rmseAppRandom1)
```

OUTPUT:

We are ignoring those Buildings which have Consumption = 0 for the entire time duration, as the prediction model will predict some garbage values for the same thus we are putting a check for each building which checks the Sum of Consumption is greater than 0 or not. Hence we have predictions for 75 buildings.

A SNIPPET OF THE OUTPUT:

V1	ME	RMSE	MAE	MPE	MAPE
Building 1_5198_1_elect.csv	0.000501986006428794	0.0236734176392296	0.0150075049252574	-Inf	Inf
Building 2_5199_1_elect.csv	-0.0000669340460994753	0.00517232232111085	0.00338591612367233	-Inf	Inf
Building 3_5286_1_elect.csv	0.000102961537065395	0.00282090846917281	0.00167005977813411	-Inf	Inf
Building 5_5290_1_elect.csv	-0.0000491089233062909	0.0024550302985	0.00163389762189635	-Inf	Inf
Building 6_5304_1_elect.csv	0.000000372183523932644	0.000055202464944509	0.000012828024593077	NaN	Inf
Building 6_5304_3_elect.csv	0.0000230515733099778	0.00250951553736764	0.00166522333471518	-Inf	Inf
Building 7_5306_1_elect.csv	-0.00010740901353799	0.00410989269512113	0.0025851118518623	-Inf	Inf
Building 8_5308_1_elect.csv	-0.0000246914415802222	0.00182226625857233	0.00114152700311887	-Inf	Inf
Building 5_5310_1_elect.csv	-0.0000540686155529885	0.00174045175099757	0.00116817765647463	-Inf	Inf
Building 10_5311_1_elect.csv	-0.0000620200272503759	0.00258217143615728	0.00159105494607956	-Inf	Inf
Building 11_5313_1_elect.csv	-0.0000243820540234712	0.00514404347675359	0.00303483212882741	-Inf	Inf
Building 12_5314_1_elect.csv	-0.0000353783363714124	0.00577071536480928	0.00374405863591565	-Inf	Inf
Building 13_5316_1_elect.csv	0.00018176840364562 7	0.00214741663002519	0.000764194815620136	NaN	Inf
Building 14_5317_1_elect.csv	-0.000166058187290601	0.0046266593978079	0.00249480276264103	-Inf	Inf
Building 14_5318_1_elect.csv	-0.0000251846478501086	0.00154174389222471	0.0010336644858597	-Inf	Inf
Building 15_5322_1_elect.csv	0.0000220060594937834	0.00113776878426824	0.000795623584525487	-Inf	Inf
Building 15_5323_1_elect.csv	0.0000319720173401161	0.000615319356179641	0.000456075830408135	NaN	Inf
Building 16_5325_1_elect.csv	0.0000652780043700341	0.0022534172671076	0.00160022814330866	-Inf	Inf
Building 17_5329_1_elect.csv	-0.000072064249738524	0.00340710457541867	0.00218665408983423	-Inf	Inf
Building 18_5332_1_elect.csv	0.0000462457523747683	0.00343211800245522	0.00237795966587178	-Inf	Inf
Building 19_5333_1_elect.csv	0.00000202591676468288	0.00093772983682709	0.000605162656510289	-Inf	Inf
Building 20_5335_1_elect.csv	0.0000358161656926932	0.00276094013161959	0.00182874065625788	NaN	Inf
Building 21_5340_1_elect.csv	-0.0000262885644994252	0.00317544920558553	0.00196969863588348	-Inf	Inf
Building 22_5351_1_elect.csv	-0.0000789126462247364	0.00238422667119864	0.00141531796705348	-Inf	Inf
Building 23_5352_1_elect.csv	0.0000120860735506637	0.0027153610782125	0.00189036161184099	-Inf	Inf
Building 24_5355_1_elect.csv	-0.0000328305482924261	0.00250511807017941	0.00164658880652559	-Inf	Inf
Building 25_5356_1_elect.csv	0.0000288185259812899	0.00193666278286711	0.00137764561149841	NaN	Inf
Building 26_5357_1_elect.csv	-0.0000109869808569031	0.00117356749514007	0.00075424391953291	-Inf	Inf
Building 26_5358_1_elect.csv	-0.000000782371192186778	0.00228632818983679	0.00140615337581237	-Inf	Inf
Building 28_28096_1_elect.csv	0.0000115691941533005	0.00564458068758558	0.00373554722370423	-Inf	Inf
Building 28_28096_4_Dist_Heating.csv	-0.0000702751699805386	0.0100182200695791	0.00751309290750919	-Inf	Inf
Building 29_28108_3_elect.csv	0.0000656612226251986	0.00888513379939187	0.00544670644584648	-Inf	Inf

PREDICTION

2. KNN model for all buildings as one dataset.

1.Libraries used:

```
#libraries used
library(FNN)
```

2.Read the file and remove unnecessary variables

```
#read file
KnnModellingData <- read.csv(file="Final_Finland_Data.csv",header=TRUE,sep=",")
KnnModellingData <- KnnModellingData %>% select(-c(BuildingID,BuildingName,MeterID,
                                                   type,Date,nearestAirport,Wind_Direction,
                                                   Conditions,Gust_SpeedMPH,PrecipitationIn,
                                                   Events,VisibilityMPH,address,day,month,
                                                   Base_Hour_Class))
```

3.Perform Scaling

```
#scaling
KnnModellingData$TemperatureF = scale(as.numeric(KnnModellingData$TemperatureF),
                                      center = TRUE,scale = TRUE)
KnnModellingData$Dew_PointF = scale(as.numeric(KnnModellingData$Dew_PointF),
                                    center = TRUE,scale = TRUE)
KnnModellingData$Humidity = scale(as.numeric(KnnModellingData$Humidity),
                                  center = TRUE,scale = TRUE)
KnnModellingData$Sea_Level_PressureIn = scale(as.numeric(KnnModellingData$Sea_Level_PressureIn),
                                              center = TRUE,scale = TRUE)
KnnModellingData$Wind_SpeedMPH = scale(as.numeric(KnnModellingData$Wind_SpeedMPH),
                                       center = TRUE,scale = TRUE)
KnnModellingData$WindDirDegrees = scale(as.numeric(KnnModellingData$WindDirDegrees),
                                        center = TRUE,scale = TRUE)
KnnModellingData$DayNumber = scale(as.numeric(KnnModellingData$DayNumber),
                                   center = TRUE,scale = TRUE)
KnnModellingData$MonthNumber = scale(as.numeric(KnnModellingData$MonthNumber),
                                     center = TRUE,scale = TRUE)
KnnModellingData$base_hr_usage = scale(as.numeric(KnnModellingData$base_hr_usage),
                                       center = TRUE,scale = TRUE)
KnnModellingData$WeekDay = scale(as.numeric(KnnModellingData$Weekday),
                                 center = TRUE,scale = TRUE)
KnnModellingData$Base_Hour_Flag = scale(as.numeric(KnnModellingData$Base_Hour_Flag),
                                        center = TRUE,scale = TRUE)
```

4.Split data into training and testing

```
#70% of the sample size
smp_size <- floor(0.70 * nrow(KnnModellingData))

#Set the seed to make your partition reproducible.
set.seed(123)
train_ind <- sample(seq_len(nrow(KnnModellingData)), size = smp_size)

#Splitting the data into training and testing
train_data <- KnnModellingData[train_ind, ]
test_data <- KnnModellingData[-train_ind, ]
```

5. Build the model

We have used kdtree algorithm to build the model

```
#Build the model
knn.model <- knn.reg(train_data[,-2], test_data[,-2], y = train_data$Consumption,
                     k = 5, algorithm= "kd_tree")
```

6.Prediction and Accuracy

```
#Predict Accuracy
knnDataFrame = data.frame(actual= test_data$Consumption, predicted = knn.model$pred)
reg_acc<-accuracy(knnDataFrame$predicted, knnDataFrame$actual)
rmse = sqrt(mean(mean(test_data$Consumption)-reg_acc)^2)
View(reg_acc)
```

Output:

	ME	RMSE	MAE	MPE	MAPE
Test set	0.0007212722	0.01051543	0.005886718	-Inf	Inf

6. Write to csv file

```
#write to csv file
write.csv(reg_acc, "Knn_FullData.csv", row.names = FALSE)
```

PREDICTION-KNN (FOR EACH BUILDING DATASET: 78 MODELS)

STEPS:

Step1: We have passed the query from the main script into this function which has the following data.

Step2: We are calculating the length of a record inside a query.

Step3: We are then running a for loop from 1 to the length of query.

Step4: We have saved the 78 files in a directory called Buildings Data. We are picking each file from that directory through a loop.

Below is a snippet of the output of the query:

```
lengQuery = length(que$BuildingID)
setwd("Buildings Data")

for (varb in 1:lengQuery){
  name = paste(que$BuildingName[varb],que$BuildingID[varb],que$MeterID[varb],
               que$type[varb],sep = "_")
  bdng = paste(name,".csv",sep="")
  knnModelData = read.csv(bdng,header = TRUE)
  if(sum(knnModelData$Consumption != 0)){
    knnModelData = knnModelData %>% select(-c(BuildingID,BuildingName,MeterID,
                                              type,Date,nearestAirport,Wind_Direction,
                                              Conditions,Gust_SpeedMPH,PrecipitationIn,
                                              Events,VisibilityMPH,address,Base_Hour_Class,
                                              day,month))

    knnModelData$TemperatureF = scale(as.numeric(knnModelData$TemperatureF),
                                      center = TRUE,scale = TRUE)
    knnModelData$Dew_PointF = scale(as.numeric(knnModelData$Dew_PointF),
                                    center = TRUE,scale = TRUE)
    knnModelData$Humidity = scale(as.numeric(knnModelData$Humidity),
                                  center = TRUE,scale = TRUE)
    knnModelData$Sea_Level_PressureIn = scale(as.numeric(knnModelData$Sea_Level_PressureIn),
                                              center = TRUE,scale = TRUE)
    knnModelData$Wind_SpeedMPH = scale(as.numeric(knnModelData$Wind_SpeedMPH),
                                       center = TRUE,scale = TRUE)
    knnModelData$WindDirDegrees = scale(as.numeric(knnModelData$WindDirDegrees),
                                        center = TRUE,scale = TRUE)
    knnModelData$base_hr_usage = scale(as.numeric(knnModelData$base_hr_usage),
                                       center = TRUE,scale = TRUE)
    knnModelData$WeekDay = scale(as.numeric(knnModelData$Weekday),
                                 center = TRUE,scale = TRUE)
    knnModelData$Base_Hour_Flag = scale(as.numeric(knnModelData$Base_Hour_Flag),
                                        center = TRUE,scale = TRUE)
```

BuildingID	BuildingName	MeterID	type
5198	Building 1	1	elect
5199	Building 2	1	elect
5286	Building 3	1	elect
5288	Building 4	1	elect
5290	Building 5	1	elect
5290	Building 5	3	elect
5290	Building 5	8	elect
5304	Building 6	1	elect
5304	Building 6	3	elect
5306	Building 7	1	elect
5308	Building 8	1	elect
5310	Building 5	1	elect
5311	Building 10	1	elect
5313	Building 11	1	elect
5314	Building 12	1	elect
5316	Building 13	1	elect
5317	Building 14	1	elect
5318	Building 14	1	elect
5322	Building 15	1	elect
5323	Building 15	1	elect
5325	Building 16	1	elect
5329	Building 17	1	elect
5332	Building 18	1	elect
5333	Building 19	1	elect
5335	Building 20	1	elect
5340	Building 21	1	elect
5351	Building 22	1	elect
5352	Building 23	1	elect
5355	Building 24	1	elect
5356	Building 25	1	elect
5357	Building 26	1	elect

REST OF THE CODE WITH STEPS EXPLAINED AS BEFORE:

```
    #70% of the sample size
    smp_size <- floor(0.70 * nrow(knnModelData))

    #Set the seed to make your partition reproducible.
    set.seed(123)
    train_ind <- sample(seq_len(nrow(knnModelData)), size = smp_size)

    #Splitting the data into training and testing
    train_data <- knnModelData[train_ind, ]
    test_data <- knnModelData[-train_ind, ]

    knn.model <- knn.reg(train_data[,-2], test_data[,-2], y = train_data$Consumption,
                         k = 5, algorithm= "kd_tree")
    knnDataFrame = data.frame(actual= test_data$Consumption, predicted = knn.model$pred)
    reg_acc<-accuracy(knnDataFrame$predicted, knnDataFrame$actual)

    if(varb == 1){
      rmseAppKNN =  reg_acc
      bnnd = bdng
    }
    else{
      rmseAppKNN = rbind(rmseAppKNN,reg_acc)
      bnnd = rbind(bnnd,bdng)
    }
  }
}
rmseAppKNN = cbind(bnnd,rmseAppKNN)
write.csv(rmseAppKNN, "RMSE_ALLBUILDINGS_KNN.csv", row.names=FALSE)
```

OUTPUT:

We are ignoring those Buildings which have Consumption = 0 for the entire time duration, as the prediction model will predict some garbage values for the same thus we are putting a check for each building which checks the Sum of Consumption is greater than 0 or not. Hence we have predictions for 75 buildings.

A SNIPPET OF THE OUTPUT:

V1	ME	RMSE	MAE	MPE	MAPE
Building 1_5198_1_elect.csv	0.000155021592293223	0.0269149878668991	0.0175454914553574	-Inf	Inf
Building 2_5199_1_elect.csv	-9.11914735972181e-05	0.00220282634871089	0.00113468684222483	-Inf	Inf
Building 3_5286_1_elect.csv	0.000218500329395306	0.0031538496373872	0.00183561053268545	-Inf	Inf
Building 5_5290_1_elect.csv	-3.62722833534433e-05	0.000872745544699623	0.000571875392968078	-Inf	Inf
Building 6_5304_1_elect.csv	4.16410019572961e-06	0.000271905489023639	9.54278243231409e-05	-Inf	Inf
Building 6_5304_3_elect.csv	3.57056047396976e-05	0.00105239724761669	0.000747087561170278	-Inf	Inf
Building 7_5306_1_elect.csv	-8.2819108754704e-05	0.00224632987748324	0.00133108678109069	-Inf	Inf
Building 8_5308_1_elect.csv	-2.96070856722382e-05	0.000703879327108297	0.000458375475437648	-Inf	Inf
Building 5_5310_1_elect.csv	-5.6176552111107e-05	0.00106897806692379	0.000712540540366048	-Inf	Inf
Building 10_5311_1_elect.csv	-1.62015633742978e-05	0.00155222649553183	0.000850046108608053	-Inf	Inf
Building 11_5313_1_elect.csv	-4.00436260578135e-05	0.00357225195860527	0.00193548438457803	-Inf	Inf
Building 12_5314_1_elect.csv	-5.23167718840144e-05	0.00217653465946866	0.00138182353737272	-Inf	Inf
Building 13_5316_1_elect.csv	7.17050148811183e-05	0.00204925069798592	0.000621729040717273	-Inf	Inf
Building 14_5317_1_elect.csv	-0.000129785308241157	0.00491998708605624	0.00243395884223903	-Inf	Inf
Building 14_5318_1_elect.csv	-2.14725096357884e-05	0.000863989761189119	0.000560752751637798	-Inf	Inf
Building 15_5322_1_elect.csv	6.43230351186862e-06	0.000709129591944598	0.000513858833184979	-Inf	Inf
Building 15_5323_1_elect.csv	-2.73977439058156e-05	0.000613260017049374	0.000418752631259496	-Inf	Inf
Building 16_5325_1_elect.csv	-8.3239285364656e-06	0.00157309333617565	0.0011287992521882	-Inf	Inf
Building 17_5329_1_elect.csv	-8.28684802043115e-05	0.00145716473160161	0.000830736117446199	-Inf	Inf
Building 18_5332_1_elect.csv	-2.48625685663024e-05	0.0019596621026117	0.00137461316592537	-Inf	Inf
Building 19_5333_1_elect.csv	2.25178021248138e-05	0.000626102416590753	0.000384989616903741	-Inf	Inf
Building 20_5335_1_elect.csv	-4.40289664370411e-05	0.00211915782894036	0.0014092954159403	-Inf	Inf
Building 21_5340_1_elect.csv	4.7494121793756e-05	0.00165649225180274	0.0010342296761298	-Inf	Inf
Building 22_5351_1_elect.csv	3.42728814209108e-05	0.00255876012859086	0.0016797407085618	-Inf	Inf
Building 23_5352_1_elect.csv	-9.18003056976944e-06	0.00189737034780273	0.00138211649753196	-Inf	Inf
Building 24_5355_1_elect.csv	-2.51068965120546e-05	0.00103037856107606	0.000661039781299515	-Inf	Inf
Building 25_5356_1_elect.csv	2.55451773785183e-05	0.00127933431155885	0.000844153956002417	-Inf	Inf
Building 26_5357_1_elect.csv	-3.98734890911074e-05	0.00065949104057931	0.000401318317233075	-Inf	Inf
Building 26_5358_1_elect.csv	-9.2898966730593e-06	0.00104227787374226	0.000692084254535209	-Inf	Inf
Building 28_28096_1_elect.csv	-9.81243267568686e-05	0.00249160828973515	0.00164872635328058	-Inf	Inf
Building 28_28096_4_Dist_Heating.csv	-3.80812171905718e-05	0.00974177166733767	0.00740017846788293	-Inf	Inf

PREDICTION-Neural Network (FOR ALL BUILDINGS DATA)

3. NEURAL NETWORK MODEL FOR ALL BUILDINGS AS ONE DATASET.

Neural networks have always been one of the most fascinating machine learning model, not only because of the fancy backpropagation algorithm, but also because of their complexity (think of deep learning with many hidden layers) and structure inspired by the brain.
Neural networks have not always been popular, partly because they were, and still are in some cases, computationally expensive and partly because they did not seem to yield better results when compared with simpler methods.

1.Libraries used:

```
#Load Libraries
library(MASS)
library(grid)
library(neuralnet)
library(dplyr)
```

2.Read the file and remove unnecessary variables

```
#read file
NNModellingData <- read.csv(file="Final_Finland_Data.csv",header=TRUE,sep=",")
NNModellingData<- NNModellingData %>% select(-c(BuildingID,BuildingName,MeterID,
                                                type,Date,nearestAirport,Wind_Direction,
                                                Conditions,Gust_SpeedMPH,PrecipitationIn,Events,
                                                VisibilityMPH,address,Base_Hour_Class,day,month))
```

3.Perform Scaling

```
#View(ModellingData)
NNModellingData$TemperatureF = scale(as.numeric(NNModellingData$TemperatureF),
                                     center = TRUE,scale = TRUE)
NNModellingData$Dew_PointF = scale(as.numeric(NNModellingData$Dew_PointF),
                                   center = TRUE,scale = TRUE)
NNModellingData$Humidity = scale(as.numeric(NNModellingData$Humidity),
                                 center = TRUE,scale = TRUE)
NNModellingData$Sea_Level_PressureIn = scale(as.numeric(NNModellingData$Sea_Level_PressureIn),
                                             center = TRUE,scale = TRUE)
NNModellingData$Wind_SpeedMPH = scale(as.numeric(NNModellingData$Wind_SpeedMPH),
                                      center = TRUE,scale = TRUE)
NNModellingData$WindDirDegrees = scale(as.numeric(NNModellingData$WindDirDegrees),
                                       center = TRUE,scale = TRUE)
NNModellingData$DayNumber = scale(as.numeric(NNModellingData$DayNumber),
                                  center = TRUE,scale = TRUE)
NNModellingData$MonthNumber = scale(as.numeric(NNModellingData$MonthNumber),
                                    center = TRUE,scale = TRUE)
NNModellingData$base_hr_usage = scale(as.numeric(NNModellingData$base_hr_usage),
                                      center = TRUE,scale = TRUE)
NNModellingData$WeekDay = scale(as.numeric(NNModellingData$Weekday),
                                center = TRUE,scale = TRUE)
NNModellingData$Base_Hour_Flag = scale(as.numeric(NNModellingData$Base_Hour_Flag),
                                       center = TRUE,scale = TRUE)
```

4.Split data into training and testing

```
#Split the data into training and testing

#70% of the sample size
smp_size <- floor(0.70 * nrow(NNModellingData))

#Set the seed to make your partition reproducible.
set.seed(123)
train_ind <- sample(seq_len(nrow(NNModellingData)), size = smp_size)

#Splitting the data into training and testing
train_data <- NNModellingData[train_ind, ]
test_data <- NNModellingData[-train_ind, ]
```

5.Build the neural network

```
#Build the neural network (NN)
creditnet <- neuralnet(Consumption ~ Weekday+Base_Hour_Flag+Holiday+TemperatureF+
                         Humidity+Sea_Level_PressureIn+Wind_SpeedMPH+
                         WindDirDegrees+base_hr_usage, train_data, hidden =5, lifesign = "minimal",
                         linear.output = TRUE, threshold = 0.1)
```

7. Plot the neural network

Optimization of the Network Configuration

Pruning describes a set of techniques to trim network size (by nodes not layers) to improve computational performance and sometimes resolution performance.

```
# plot the NN
plot(creditnet, rep = "best")
```

Test the resulting output

```
## test the resulting output
temp_test <- subset(test_data, select = c("Weekday","Base_Hour_Flag","Holiday","TemperatureF",
                                          "Humidity","Sea_Level_PressureIn",
                                          "Wind_SpeedMPH","WindDirDegrees","base_hr_usage"))

creditnet.results <- compute(creditnet, temp_test)

results <- data.frame(actual = test_data$Consumption, prediction = creditnet.results$net.result)

summary(results)
```

Checking the accuracy

```
#Accuracy
reg_acc <- accuracy(results[,1],results[,2])
rmse = sqrt(mean(mean(test_data$Consumption)-reg_acc)^2)
View(rmse)
```

FINAL COMMENTS: It is computationally challenging and time consuming.

PREDICTION-Neural Network (FOR EACH BUILDING DATASET: 78 MODELS)

STEPS:

Step1: We have passed the query from the main script into this function which has the following data.

Step2: We are calculating the length of a record inside a query.

Step3: We are then running a for loop from 1 to the length of query.

Step4: We have saved the 78 files in a directory called Buildings Data. We are picking each file from that directory through a loop.

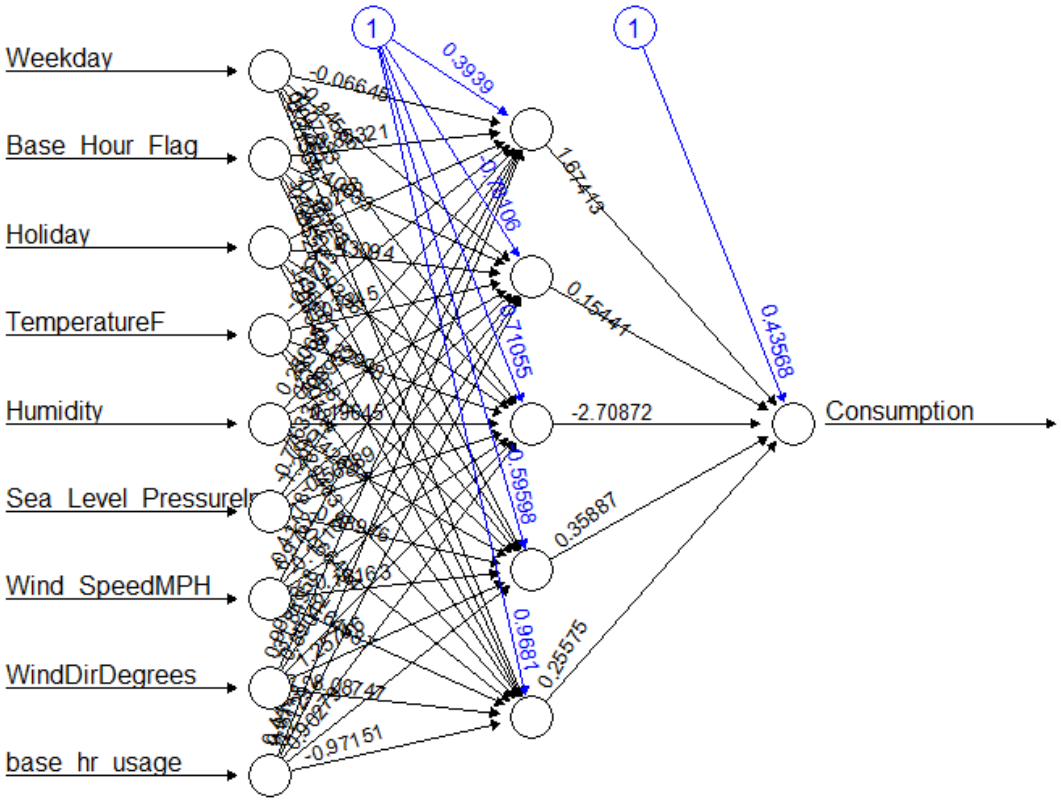
Below is a snippet of the output of the query:

```
getwd()
for (varb in 1:lengQuery){
  name = paste(que$BuildingName[varb],que$BuildingID[varb],que$MeterID[varb],
               que$type[varb],sep = "_")
  bdng = paste(name,".csv",sep="")
  neuroModelData = read.csv(bdng,header = TRUE)
  if(sum(neuroModelData$Consumption != 0)){
    neuroModelData = neuroModelData %>% select(-c(BuildingID,BuildingName,MeterID,
                                                  type,Date,nearestAirport,Wind_Direction,
                                                  Conditions,Gust_SpeedMPH,PrecipitationIn,
                                                  Events,VisibilityMPH,address,Base_Hour_Class,
                                                  day,month))

    neuroModelData$TemperatureF = scale(as.numeric(neuroModelData$TemperatureF),
                                        center = TRUE,scale = TRUE)
    neuroModelData$Dew_PointF = scale(as.numeric(neuroModelData$Dew_PointF),
                                      center = TRUE,scale = TRUE)
    neuroModelData$Humidity = scale(as.numeric(neuroModelData$Humidity),
                                    center = TRUE,scale = TRUE)
    neuroModelData$Sea_Level_PressureIn = scale(as.numeric(neuroModelData$Sea_Level_PressureIn),
                                                center = TRUE,scale = TRUE)
    neuroModelData$Wind_SpeedMPH = scale(as.numeric(neuroModelData$Wind_SpeedMPH),
                                         center = TRUE,scale = TRUE)
    neuroModelData$WindDirDegrees = scale(as.numeric(neuroModelData$WindDirDegrees),
                                          center = TRUE,scale = TRUE)
    neuroModelData$base_hr_usage = scale(as.numeric(neuroModelData$base_hr_usage),
                                         center = TRUE,scale = TRUE)
    neuroModelData$WeekDay = scale(as.numeric(neuroModelData$Weekday),
                                   center = TRUE,scale = TRUE)
    neuroModelData$Base_Hour_Flag = scale(as.numeric(neuroModelData$Base_Hour_Flag),
                                          center = TRUE,scale = TRUE)
```

REST OF THE CODE WITH STEPS EXPLAINED AS BEFORE:

```
    #70% of the sample size
    smp_size <- floor(0.70 * nrow(neuroModelData))

    #Set the seed to make your partition reproducible.
    set.seed(123)
    train_ind <- sample(seq_len(nrow(neuroModelData)), size = smp_size)

    #Splitting the data into training and testing
    train_data <- neuroModelData[train_ind, ]
    test_data <- neuroModelData[-train_ind, ]

    #Build the neural network (NN)
    creditnet <- neuralnet(Consumption ~ Weekday+Base_Hour_Flag+Holiday+TemperatureF+
                             Humidity+Sea_Level_PressureIn+Wind_SpeedMPH+
                             WindDirDegrees+base_hr_usage, train_data,
                           hidden =5, lifesign = "minimal",
                           linear.output = TRUE, threshold = 0.1)
    ## test the resulting output
    temp_test <- subset(test_data, select = c("Weekday","Base_Hour_Flag","Holiday",
                                              "TemperatureF","Humidity",
                                              "Sea_Level_PressureIn","Wind_SpeedMPH",
                                              "WindDirDegrees","base_hr_usage"))

    creditnet.results <- compute(creditnet, temp_test)
    results <- data.frame(actual = test_data$Consumption,
                          prediction = creditnet.results$net.result)
    #Checking the accuracy of the model
    reg_acc <- accuracy(results[,1],results[,2])
    if(varb == 1){
      rmseAppNeural =  reg_acc
      budnn = bdng
    }
    else{
      rmseAppNeural = rbind(rmseAppNeural,reg_acc)
      budnn = rbind(budnn,bdng)
    }
  }
}
rmseAppNeural = cbind(budnn,rmseAppNeural)
```

Graph: Below is the neural network for building 21



Output:

We are ignoring those Buildings which have Consumption = 0 for the entire time duration, as the prediction model will predict some garbage values for the same thus we are putting a check for each building which checks the Sum of Consumption is greater than 0 or not. Hence we have predictions for 75 buildings

A snippet of the output:

V1	ME	RMSE	MAE	MPE	MAPE
Building 1_5198_1_elect.csv	-0.000610296215121779	0.0310432044584986	0.0194164444771407	0.317408641665885	8.77226739775205
Building 2_5199_1_elect.csv	0.000128840408126823	0.00700270708358748	0.00544370887149452	-1.01518346132562	19.6517216399517
Building 3_5286_1_elect.csv	-0.000182935137992128	0.00438458701655914	0.0028214468497179	-48.0328185997185	210.362759465466
Building 5_5290_1_elect.csv	0.0000810328282522054	0.00455740236354958	0.00360781424225803	7.41320891944328	74.2168845075945
Building 6_5304_1_elect.csv	-0.0000400769088729117	0.00183404154128005	0.00123284168993584	93.2541822219561	149.304666695116
Building 6_5304_3_elect.csv	-0.0000787446739490279	0.00485327162235898	0.00350533680527166	-7.96279168792251	28.8128698119756
Building 7_5306_1_elect.csv	0.000121111568607007	0.00595884340730798	0.0045403641742478	-8.82498221579887	34.5717178475591
Building 8_5308_1_elect.csv	0.0000189140272562047	0.00347753614520751	0.00265425311247375	-4.96619481847587	35.4863330836208
Building 5_5310_1_elect.csv	0.0000927696252760462	0.00349472247158635	0.00275079367167454	-88.8884555599261	135.748397627036
Building 10_5311_1_elect.csv	0.0000558501111245183	0.00444059327701233	0.00339578170028908	-6.19570356163166	85.3460311361489
Building 11_5313_1_elect.csv	0.00000900660824989007	0.00691459752657328	0.00510777704095463	-12.4974084964636	53.9416355910822
Building 12_5314_1_elect.csv	-0.0000382976596629082	0.00762030268048183	0.00600027545924866	-0.306903284740489	14.7942928184302
Building 13_5316_1_elect.csv	-0.000136678722225245	0.0080403587148401	0.00581470186136111	296.945457551635	317.323743804329
Building 14_5317_1_elect.csv	0.000219131448765838	0.0067062617203264	0.00489570820709615	-3.8824397888652	167.328528007629
Building 14_5318_1_elect.csv	0.0000890428389060755	0.00314407664420751	0.00250383783598409	10.6623675048997	75.0762368709403
Building 15_5322_1_elect.csv	-0.0000368713536377142	0.00390299521744311	0.00295906475875498	26.3790836305402	154.202577751627
Building 15_5323_1_elect.csv	-0.000101601453030643	0.00341511588804531	0.00249460634794207	55.6684354952258	210.794408068976
Building 16_5325_1_elect.csv	-0.000121602602236058	0.00349137893206911	0.00267740314409984	-3.34010991916276	21.0778065381994
Building 17_5329_1_elect.csv	-0.000046127083325381	0.00519406232275903	0.00406746371670801	0.74846858385129	69.6531210333686
Building 18_5332_1_elect.csv	0.0000102004751375224	0.00480721607139352	0.00384663809886682	-2.68536540248263	30.2657758007038
Building 19_5333_1_elect.csv	-0.000130112905980001	0.00301026423365266	0.00228057164492925	-45.4023141930808	336.759552684477
Building 20_5335_1_elect.csv	-0.0000724823297721627	0.00420044204912587	0.00322120586093607	1.54775794175505	10.4705951451514
Building 21_5340_1_elect.csv	0.000000495318410180764	0.00465933718446218	0.0036158067598962	-8.17441366485344	33.6864237242624
Building 22_5351_1_elect.csv	0.0000862991608833558	0.00374528273673082	0.00279265745099484	-0.333226345323491	14.1383858237399
Building 23_5352_1_elect.csv	0.000019543495009105	0.0040888487991356	0.00321516360119541	-3.85143485976853	48.3509732651682
Building 24_5355_1_elect.csv	0.0000152120479639239	0.00405198341912385	0.00329031145836388	-0.846738523605968	14.4525523909528
Building 25_5356_1_elect.csv	-0.0000722601128072232	0.00349944671757649	0.00269759622225896	-2.56980902050048	16.9947804633669
Building 26_5357_1_elect.csv	0.0000273188500145014	0.00331822826684997	0.00242782021140331	-476.483949677479	522.086952381283
Building 26_5358_1_elect.csv	0.000025390164381421	0.00377906963362937	0.00292939362618817	3.22469210868309	81.26572768529
Building 28_28096_1_elect.csv	-0.0000271198047871402	0.00766181868521154	0.00611623688295342	42.4591197850301	112.442240881405
Building 28_28096_4_Dist_Heating.csv	0.0000564526874164669	0.0117585817172757	0.00887924703969057	23.0059644541149	102.125021669103

ANALYSIS OF PREDICTION:

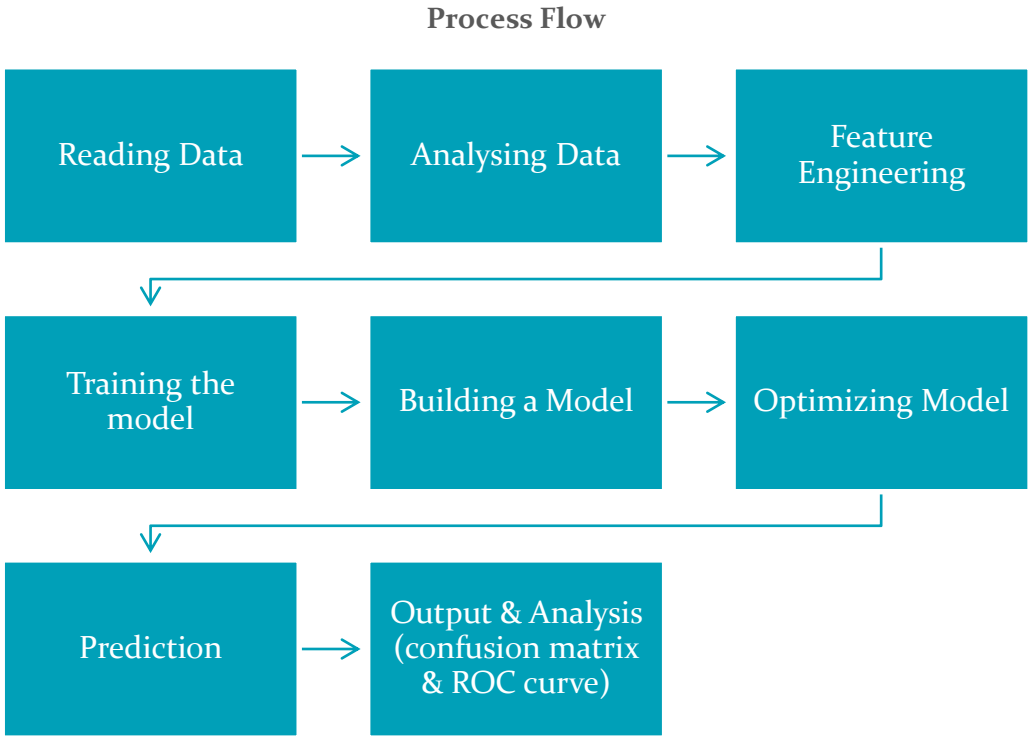
78 Buildings	RMSE Mean	One Dataset	RMSE Mean
KNN	0.003034378	KNN	0.010515429
Neural Network	0.005708638	Neural Network	0.615972085
Random Forest	0.003405507	Random Forest	0.223687187
Regression	0.004858729	Regression	0.010137215

Comparing the above RMSE values for different buildings, execution time, computational complexity and number of variables we have concluded the results as below:

- ✓ Best model for 78 Buildings Prediction - KNN
- ✓ Best model for one dataset - Regression

Classification

Classification Build a model to predict the Base_Hour_Class flag for this dataset.



Assumptions: Initially building a model on one building dataset which later would be used to generate model for other 77 datasets

Logistic Model:

Steps:

1) Libraries Required

```
library(ISLR)
library(leaps)
library(dplyr)
library(DAAG)
library(ROCR)
library(caret)
library(mlbench)
```

2)Code to read data from csv

```
data = read.csv(file="Building 1_5198_1_elect.csv",header = TRUE)
```

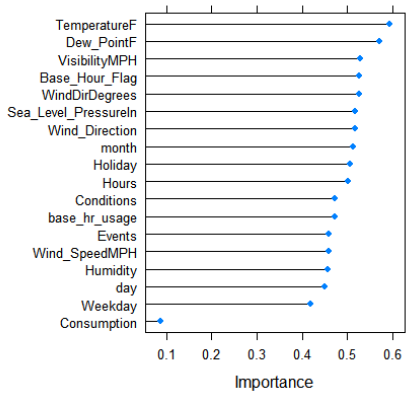
3) By analyzing data doing feature selection by removing variables which wouldn't be required

```
data <- Filter(function(x)!all(is.na(x)), data)
data <- Filter(function(x)!all(x=="N/A"), data)
drops <- c("Date","PrecipitationIn","BuildingID","BuildingName","MeterID","type","nearestAirport","address")
data <- data[ , !(names(data) %in% drops)]
View(data)
df<-data.frame(data)
```

4) For feature selection identifying highly correlated features by ranking feature by importance. Used LVQ model(Learning vector quantization model)

```
set.seed(7)
# prepare training scheme
control <- trainControl(method="repeatedcv", number=10, repeats=3)
# train the model
model <- train(Base_Hour_Class~., data, method="lvq", preProcess="scale", trControl=control)
# estimate variable importance
importance <- varImp(model, scale=FALSE)
# plot importance
plot(importance)
```

Output: Below shows that Temperature and Dew_PointF are important feature with a threshold above 0.5



5) Below code for spitting the code into train and test dataset

```
df <- data
#Take 70% of the data as the sample data
smp_size<-floor(0.70*nrow(df))
#Divide the data into train and test data. Sample data is basically train data
train_ind<-sample(seq_len(nrow(df)),size=smp_size)
train <- df[train_ind,]

# Rest 30% is test data
test <- df[-train_ind,]
```

6) Building a logistic model using below code

```
#logistic regression
lrmodel <- glm(Base_Hour_Class~.,family = binomial(link = "logit"), data = train)
summary(lrmodel)
```

7) Using step AIC on the above build model to validate the model

```
#using step AIC
step <- stepAIC(lrmodel, direction="forward")
step$anova
```

Output:

```
Base_Hour_Class ~ Hours + Consumption + day + month + Weekday +
    Base_Hour_Flag + Holiday + TemperatureF + Dew_PointF + Humidity +
    Sea_Level_PressureIn + VisibilityMPH + Wind_Direction + Wind_SpeedMPH +
    Conditions + WindDirDegrees + Events + base_hr_usage

Final Model:
Base_Hour_Class ~ Hours + Consumption + day + month + Weekday +
    Base_Hour_Flag + Holiday + TemperatureF + Dew_PointF + Humidity +
    Sea_Level_PressureIn + VisibilityMPH + Wind_Direction + Wind_SpeedMPH +
    Conditions + WindDirDegrees + Events + base_hr_usage


  Step Df Deviance Resid. Df   Resid. Dev AIC
1                       5663 1.163302e-05 164
```

8) Below code to predict the outcome by predict function

```
test.probs <- predict(lrmodel1, type = 'response',newdata = test)
table(test$Base_Hour_Class, test.probs > 0.5)
```

9) Below code to create confusion matrix

```
.
       tab=confusionMatrix(test$Base_Hour_Class,pred)
```

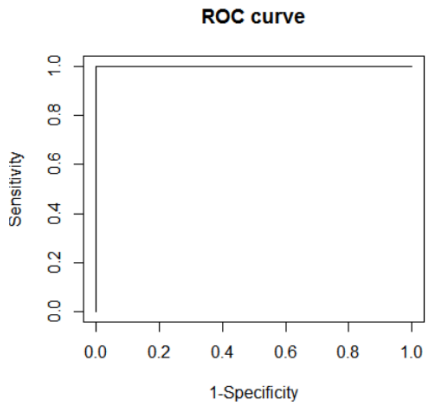
Output:

```
> tab
Confusion Matrix and Statistics

          Reference
Prediction High  Low
      High 1244    0
      Low     0 1219

               Accuracy : 1
                 95% CI : (0.9985, 1)
    No Information Rate : 0.5051
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 1
 Mcnemar's Test P-Value : NA

            Sensitivity : 1.0000
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 1.0000
             Prevalence : 0.5051
         Detection Rate : 0.5051
   Detection Prevalence : 0.5051
      Balanced Accuracy : 1.0000

       'Positive' Class : High
```

10) Below code to create ROC curve

```
prediction <- prediction(test.probs, test$Base_Hour_Class)
performance <- performance(prediction, measure = "tpr", x.measure = "fpr")
plot(performance, main="ROC curve", xlab="1-Specificity", ylab="Sensitivity")
```

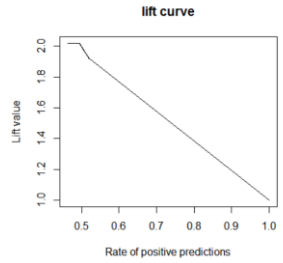
Output:



11) Additionally have also created lift curve

```
perf <- performance(prediction,"lift","rpp")
plot(perf, main="lift curve")
```

Output:



KNN Model:

Steps:

1) Libraries Required

```
library(ISLR)
library(leaps)
library(dplyr)
library(DAAG)
library(ROCR)
library(MASS)
library(caret)
```

2) Read the data from csv

```
data = read.csv(file="Building 1_5198_1_elect.csv",header = TRUE)
```

3) By analyzing data -doing feature selection by removing variables which wouldn't be required

```
drops <- c("Date","PrecipitationIn","BuildingID","BuildingName","MeterID","type",
           "nearestAirport","address","Gust_SpeedMPH","VisibilityMPH")
data <- data[ , !(names(data) %in% drops)]
```

4) Normalizing the numerical data

```
#normalising data
d1<-d[,c(1,2,8:11,13,15,17,3:7,12,14,16,18)]
View(d1)

normalize <- function(x) {
  num <- x - min(x)
  denom <- max(x) - min(x)
  return (num/denom)
}

t1<- as.data.frame(lapply(d1[,c(1:9)], normalize))
View(t1)
d2<-cbind(t1,d1[10:18])
View(d2)
summary(t1)
str(d)
```

5)Splitting the data into train and test data for modelling

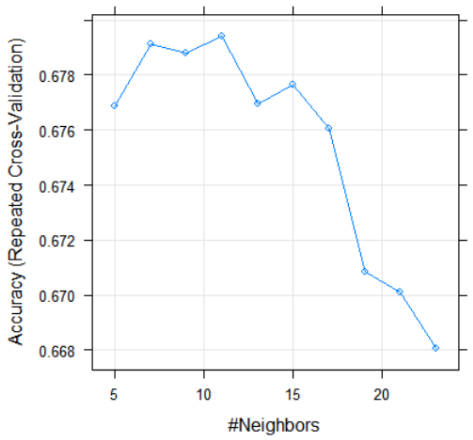
```
#Take 60% of the data as the sample data
smp_size<-floor(0.60*nrow(d2))

#Divide the data into train and test data. Sample data is basically train data
train_ind<-sample(seq_len(nrow(d2)),size=smp_size)
train <- d2[train_ind,]
View(train)

# Rest 40% is test data
test <- d2[-train_ind,]
View(test)
```

6)Creating a knn model using Caret package by validating the model using cross validation for better accuracy. Used sampling method "repeatedcv" performing 10 k fold resampling iterations.

```
ctrl <- trainControl(method="repeatedcv", number=10, repeats=3)
set.seed(12345)
knnFit1 <- train(Base_Hour_Class~., data=train, method="knn",
                 trControl=ctrl, tuneLength=10)
```

7) Plotting the model:

```
plot(knnFit1)
```



8) Below code for prediction

```
knnPredict1 <- predict(knnFit1, test)
```

9) Creating confusion Matrix

```
cmat1 <- confusionMatrix(knnPredict1, test$Base_Hour_Class, positive="High")
```

Output:

```
Confusion Matrix and Statistics

          Reference
Prediction High  Low
      High 1170  543
      Low   537 1034

               Accuracy : 0.6711
                 95% CI : (0.6548, 0.6872)
    No Information Rate : 0.5198
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.3411
 Mcnemar's Test P-Value : 0.8791

            Sensitivity : 0.6854
            Specificity : 0.6557
         Pos Pred Value : 0.6830
         Neg Pred Value : 0.6582
             Prevalence : 0.5198
         Detection Rate : 0.3563
   Detection Prevalence : 0.5216
      Balanced Accuracy : 0.6705

       'Positive' Class : High
```

10) Creating ROC curve

```
prediction <- prediction(knnPredict1,test$Base_Hour_Class )
performance <- performance(prediction, measure = "tpr", x.measure = "fpr")
plot(performance, main="ROC curve", xlab="1-Specificity", ylab="Sensitivity")
```

11) Writing confusion matrix in csv

```
tocsv <- data.frame(cbind(t(cmat1$overall),t(cmat1$byClass)))
View(tocsv)
write.csv(tocsv,file="KNN Matrix.csv",append = TRUE)
```

Neural Network Model:

Steps:

1)Library Used:

```
library(ISLR)
library(leaps)
library(dplyr)
library(DAAG)
library(ROCR)
library(MASS)
library(nnet)
library(caret)
```

2) Read the data from csv

```
data = read.csv(file="Building 1_5198_1_elect.csv",header = TRUE)
```

3) By analyzing data -doing feature selection by removing variables which wouldn't be required

```
drops <- c("Date","PrecipitationIn","BuildingID","BuildingName","MeterID","type",
           "nearestAirport","address","Gust_SpeedMPH","VisibilityMPH")
data <- data[ , !(names(data) %in% drops)]
```

4)Splitting the data into train and test data for modelling

```
#Take 60% of the data as the sample data
smp_size<-floor(0.60*nrow(d2))

#Divide the data into train and test data. Sample data is basically train data
train_ind<-sample(seq_len(nrow(d2)),size=smp_size)
train <- d2[train_ind,]
View(train)

# Rest 40% is test data
test <- d2[-train_ind,]
View(test)
```

5)Creating Model using nnet package

```
nnetmodel <- nnet(Base_Hour_Class~ Consumption + Base_Hour_Flag + Holiday +
                    TemperatureF + Dew_PointF + Humidity + Sea_Level_PressureIn + VisibilityMPH +
                    Wind_SpeedMPH + base_hr_usage,train, size=2,rang = 0.1,
                  decay = 5e-4, maxit = 350,MaxNWts = 5000)
```

6) Predicting model

```
#Use predict() function to predict the probabilities of the outcome
pr<-predict(nnetmodel, test,type = "class")
```

7) Creating Confusion Matrix

```
lc=confusionMatrix(test$Base_Hour_Class,pr)
```

Output:

```
#createmodel
nnetmodel <- nnet(Base_Hour_Class~ Consumption + Base_Hour_Flag + Holiday +
                    TemperatureF + Dew_PointF + Humidity + Sea_Level_PressureIn + VisibilityMPH +
                    Wind_SpeedMPH + base_hr_usage,train, size=2,rang = 0.1,
                  decay = 5e-4, maxit = 350,MaxNWts = 5000)
```

```
Confusion Matrix and Statistics

          Reference
Prediction High  Low
      High 1696    1
      Low     0 1587

               Accuracy : 0.9997
                 95% CI : (0.9983, 1)
    No Information Rate : 0.5164
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9994
 Mcnemar's Test P-Value : 1

            Sensitivity : 1.0000
            Specificity : 0.9994
         Pos Pred Value : 0.9994
         Neg Pred Value : 1.0000
             Prevalence : 0.5164
         Detection Rate : 0.5164
   Detection Prevalence : 0.5167
      Balanced Accuracy : 0.9997

       'Positive' Class : High
```

8) creating roc curve

```
#create ROC curve
require(ROCR)
prediction <- prediction(pr, test$Base_Hour_Class)
performance <- performance(prediction, measure = "tpr", x.measure = "fpr")
plot(performance, main="ROC curve", xlab="1-Specificity", ylab="Sensitivity")
```

Random Forest Model:

Steps:

1)Library Used:

```
library(randomForest)
library(caret)
library(ROCR)
```

2) Read the data from csv

```
data = read.csv(file="Building 1_5198_1_elect.csv",header = TRUE)
```

3) By analyzing data -doing feature selection by removing variables which wouldn't be required

```
drops <- c("Date","PrecipitationIn","BuildingID","BuildingName","MeterID","type",
           "nearestAirport","address","Gust_SpeedMPH","VisibilityMPH")
data <- data[ , !(names(data) %in% drops)]
```

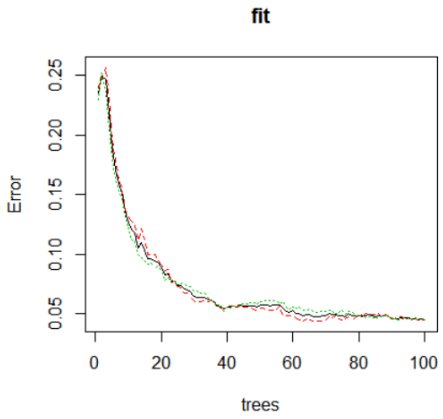
4)Splitting the data into train and test data for modelling

```
#Take 60% of the data as the sample data
smp_size<-floor(0.60*nrow(d2))

#Divide the data into train and test data. Sample data is basically train data
train_ind<-sample(seq_len(nrow(d2)),size=smp_size)
train <- d2[train_ind,]
View(train)

# Rest 40% is test data
test <- d2[-train_ind,]
View(test)
```

5)Creating Model using Random Forest

```
# Fitting model
fit <- randomForest(Base_Hour_Class ~.,train,ntree=100,importance=T,na.action = na.omit)
summary(fit)
plot(fit)
```

Output: Plotting the fit



6)Plotting the important variables

```
varImpPlot(fit,
           sort = T,
           main="Variable Importance",
           n.var=10)
```

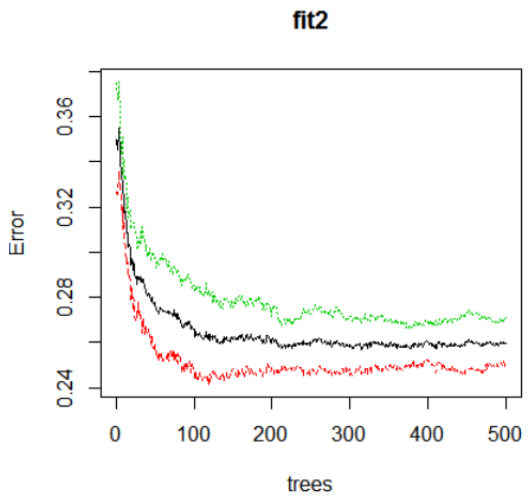
Output:

	MeanDecreaseAccuracy	MeanDecreaseGini
Hours	15.247247	144.335022
Consumption	44.712264	967.006597
day	16.635682	126.462881
month	24.268287	166.145626
Weekday	8.167492	46.641646
Base_Hour_Flag	8.586120	31.604206
Holiday	4.748867	7.119658
TemperatureF	16.692245	126.494525
Dew_PointF	13.417905	119.018928
Humidity	11.783517	103.161672
Sea_Level_PressureIn	16.094531	106.982329
VisibilityMPH	5.221358	22.364233
Wind_Direction	17.345453	166.913815
Wind_SpeedMPH	13.157327	94.245225
Conditions	14.924809	88.774813
WindDirDegrees	10.924127	84.435168
Events	5.459317	16.584914
base_hr_usage	27.798177	444.504585

7) Based on the important features plotted creating a new model

```
fit2 <- randomForest(Base_Hour_Class ~Hours + Base_Hour_Flag + Holiday +
                       TemperatureF + Dew_PointF + Humidity + Sea_Level_PressureIn +
                       VisibilityMPH + Wind_SpeedMPH + base_hr_usage,train,na.action = na.omit)
```

Output: plotting new model



8) Prediction function

```
#predict
test.probs <- predict(fit2, test)
str(test.probs)
```

9) Creating Confusion Matrix and writing in csv

```
tab=confusionMatrix(test$Base_Hour_Class,test.probs)
str(tab)
tocsv <- data.frame(cbind(t(tab$overall),t(tab$byClass)))
View(tocsv)
write.csv(tocsv,file="RandomForest Matrix.csv",append = TRUE)
```

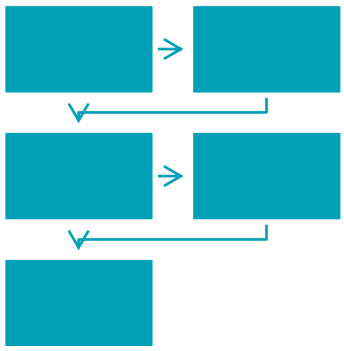
```
Confusion Matrix and Statistics

          Reference
Prediction High Low
      High  963 317
      Low   321 862

               Accuracy : 0.741
                 95% CI : (0.7232, 0.7582)
    No Information Rate : 0.5213
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.4811
 Mcnemar's Test P-Value : 0.9055

            Sensitivity : 0.7500
            Specificity : 0.7311
         Pos Pred Value : 0.7523
         Neg Pred Value : 0.7287
             Prevalence : 0.5213
         Detection Rate : 0.3910
   Detection Prevalence : 0.5197
      Balanced Accuracy : 0.7406

       'Positive' Class : High
```

1. Build Logistic Regression, KNN, Random Forest and Neural Network models for each building dataset (78 different models). Optimize your model through feature engineering and model selection.

Solution:

Building 78 models for each model using above approach and writing the output (Confusion Matrix) in one csv file

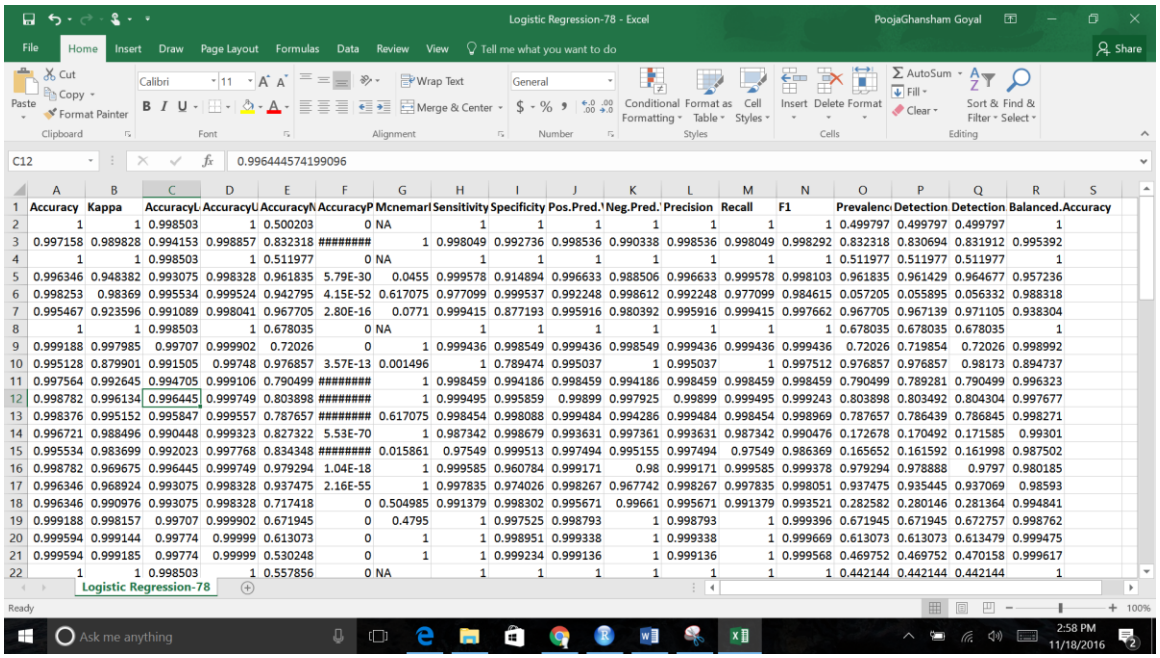
Process Flow diagram:



1) Logistic Regression:

```
library(ISLR)
library(leaps)
library(dplyr)
library(DAAG)
library(ROCR)
library(MASS)
library(caret)
library(mlbench)#
library(sqldf)

data = read.csv(file="Final_Finland_Data.csv",header = TRUE)
que = sqldf("Select DISTINCT BuildingID,BuildingName,MeterID,type from data")

lengQuery = length(que$BuildingID)
#setwd("Buildings Data")
for (varb in 1:lengQuery){
  name = paste(que$BuildingName[varb],que$BuildingID[varb],que$MeterID[varb],que$type[varb],sep = "_")
  bdng = paste(name,".csv",sep="")
  #Reading csv
  data = read.csv(file = bdng,header = TRUE)
  data <- Filter(function(x)!all(is.na(x)), data)
  data <- Filter(function(x)!all(x=="N/A"), data)
  drops <- c("Date","PrecipitationIn","BuildingID","BuildingName","MeterID","type","nearestAirport","address")
  data <- data[ , !(names(data) %in% drops)]
  na.omit(data)

  data$Weekday=as.factor(data$Weekday)
  data$Base_Hour_Flag=as.factor(data$Base_Hour_Flag)
  data$Holiday=as.factor(data$Holiday)
  newVar = data$Base_Hour_Class
  if(length(unique(newVar)) == 2){
    #training data set

    df <-as.data.frame(data)
    #Take 70% of the data as the sample data
    smp_size<-floor(0.70*nrow(df))
    #Divide the data into train and test data. Sample data is basically train data
    train_ind<-sample(seq_len(nrow(df)),size=smp_size)
    train <- df[train_ind,]



    # Rest 30% is test data
    test <- df[-train_ind,]

    #final model
    lrmodel1<- glm(Base_Hour_Class ~ Hours+Consumption + day + month +
                     Weekday + Base_Hour_Flag + Holiday + TemperatureF + Dew_PointF +
                     Humidity + Sea_Level_PressureIn + VisibilityMPH + Wind_Direction +
                     Wind_SpeedMPH  + WindDirDegrees + base_hr_usage, family = binomial(link = "logit"), data = train)

    #Predict the outcome using predict() function
    test.probs <- predict(lrmodel1, type = 'response',newdata = test)
    table(test$Base_Hour_Class, test.probs > 0.5)

    #Divide the lowpredicted values based on probability values
    pred<- rep("Low",length(test.probs))
    pred[test.probs<=0.5]<-"High"

    #confusion matrix
    lc=confusionMatrix(test$Base_Hour_Class,pred)
    tocsv <- data.frame(cbind(t(lc$overall),t(lc$byClass)))
    if(varb==1){
      logResClass = tocsv
    }
    else{
      logResClass = rbind(logResClass,tocsv)
    }
  }
}
write.csv(logResClass,"Logistic Regression-78.csv",row.names = FALSE)
```

Output: Confusion matrix for all 78 models



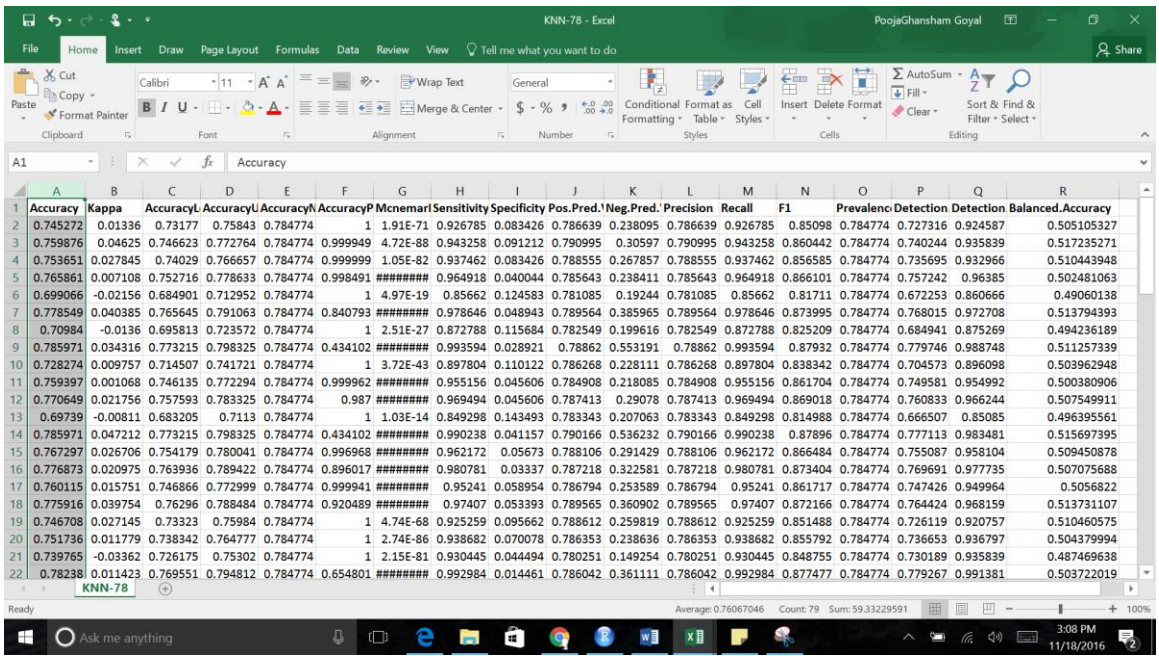
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	Accuracy	Kappa	AccuracyL	AccuracyU	AccuracyN	AccuracyP	Mcnemarl	Sensitivity	Specificity	Pos.Pred.\	Neg.Pred.'	Precision	Recall	F1	Prevalenc	Detection	Detection	Balanced.Accuracy	
2	1	1	0.998503	1	0.500203	0	NA	1	1	1	1	1	1	1	0.499797	0.499797	0.499797	1	
3	0.997158	0.989828	0.994153	0.998857	0.832318	########	1	0.998049	0.992736	0.998536	0.990338	0.998536	0.998049	0.998292	0.832318	0.830694	0.831912	0.995392	
4	1	1	0.998503	1	0.511977	0	NA	1	1	1	1	1	1	1	0.511977	0.511977	0.511977	1	
5	0.996346	0.948382	0.993075	0.998328	0.961835	5.79E-30	0.0455	0.999578	0.914894	0.996633	0.988506	0.996633	0.999578	0.998103	0.961835	0.961429	0.964677	0.957236	
6	0.998253	0.98369	0.995534	0.999524	0.942795	4.15E-52	0.617075	0.977099	0.999537	0.992248	0.998612	0.992248	0.977099	0.984615	0.057205	0.055895	0.056332	0.988318	
7	0.995467	0.923596	0.991089	0.998041	0.967705	2.80E-16	0.0771	0.999415	0.877193	0.995916	0.980392	0.995916	0.999415	0.997662	0.967705	0.967139	0.971105	0.938304	
8	1	1	0.998503	1	0.678035	0	NA	1	1	1	1	1	1	1	0.678035	0.678035	0.678035	1	
9	0.999188	0.997985	0.99707	0.999902	0.72026	0	1	0.999436	0.998549	0.999436	0.998549	0.999436	0.999436	0.999436	0.72026	0.719854	0.72026	0.998992	
10	0.995128	0.879901	0.991505	0.99748	0.976857	3.57E-13	0.001496	1	0.789474	0.995037	1	0.995037	1	0.997512	0.976857	0.976857	0.98173	0.894737	
11	0.997564	0.992645	0.994705	0.999106	0.790499	########	1	0.998459	0.994186	0.998459	0.994186	0.998459	0.998459	0.998459	0.790499	0.789281	0.790499	0.996323	
12	0.998782	0.996134	0.996445	0.999749	0.803898	########	1	0.999495	0.995859	0.99899	0.997925	0.99899	0.999495	0.999243	0.803898	0.803492	0.804304	0.997677	
13	0.998376	0.995152	0.995847	0.999557	0.787657	########	0.617075	0.998454	0.998088	0.999484	0.994286	0.999484	0.998454	0.998969	0.787657	0.786439	0.786845	0.998271	
14	0.996721	0.988496	0.990448	0.999323	0.827322	5.53E-70	1	0.987342	0.998679	0.993631	0.997361	0.993631	0.987342	0.990476	0.172678	0.170492	0.171585	0.99301	
15	0.995534	0.983699	0.992023	0.997768	0.834348	########	0.015861	0.97549	0.999513	0.997494	0.995155	0.997494	0.97549	0.986369	0.165652	0.161592	0.161998	0.987502	
16	0.998782	0.969675	0.996445	0.999749	0.979294	1.04E-18	1	0.999585	0.960784	0.999171	0.98	0.999171	0.999585	0.999378	0.979294	0.978888	0.9797	0.980185	
17	0.996346	0.968924	0.993075	0.998328	0.937475	2.16E-55	1	0.997835	0.974026	0.998267	0.967742	0.998267	0.997835	0.998051	0.937475	0.935445	0.937069	0.98593	
18	0.996346	0.990976	0.993075	0.998328	0.717418	0	0.504985	0.991379	0.998302	0.995671	0.99661	0.995671	0.991379	0.993521	0.282582	0.280146	0.281364	0.994841	
19	0.999188	0.998157	0.99707	0.999902	0.671945	0	0.4795	1	0.997525	0.998793	1	0.998793	1	0.999396	0.671945	0.671945	0.672757	0.998762	
20	0.999594	0.999144	0.99774	0.99999	0.613073	0	1	1	0.998951	0.999338	1	0.999338	1	0.999669	0.613073	0.613073	0.613479	0.999475	
21	0.999594	0.999185	0.99774	0.99999	0.530248	0	1	1	0.999234	0.999136	1	0.999136	1	0.999568	0.469752	0.469752	0.470158	0.999617	
22	1	1	0.998503	1	0.557856	0	NA	1	1	1	1	1	1	1	0.442144	0.442144	0.442144	1	

2) KNN Model

```
library(ISLR)
library(leaps)
library(dplyr)
library(DAAG)
library(ROCR)
library(MASS)
library(caret)

#reading data
data = read.csv(file="Final_Finland_Data.csv",header = TRUE)
names(data)
summary(data)
normalize <- function(x) {
  num <- x - min(x)
  denom <- max(x) - min(x)
  return (num/denom)
}
for (varb in 1:lengQuery){
  #removing na's
  data <- Filter(function(x)!all(is.na(x)), data)
  data <- Filter(function(x)!all(x=="N/A"), data)
  na.omit(data)
  drops <- c("Date","PrecipitationIn","BuildingID","BuildingName","MeterID","type"
             ,"nearestAirport","address","Gust_SpeedMPH","VisibilityMPH")
  data <- data[ , !(names(data) %in% drops)]

  #changing data types
  d<- as.data.frame(data)
  d$Weekday=as.factor(d$Weekday)
  d$Base_Hour_Flag=as.factor(d$Base_Hour_Flag)
  d$Holiday=as.factor(d$Holiday)
  newVar = d$Base_Hour_Class
  if(length(unique(newVar)) == 2){
  #normalising data
  d1<-d[,c(1,2,8:11,13,15,17,3:7,12,14,16,18)]

  t1<- as.data.frame(lapply(d1[,c(1:9)], normalize))
  d2<-cbind(t1,d1[10:18])
```

```
#Take 60% of the data as the sample data
smp_size<-floor(0.60*nrow(d2))

#Divide the data into train and test data. Sample data is basically train data
train_ind<-sample(seq_len(nrow(d2)),size=smp_size)
train <- d2[train_ind,]

# Rest 40% is test data
test <- d2[-train_ind,]

#creating model
ctrl <- trainControl(method="repeatedcv", number=10, repeats=3)
set.seed(12345)
knnFit1 <- train(Base_Hour_Class~., data=train, method="knn",
                 trControl=ctrl, tuneLength=10)
#prediction
knnPredict1 <- predict(knnFit1, newdata=d)

#ConfusionMatrix
cmat1 <- confusionMatrix(knnPredict1, data$Base_Hour_Class, positive="High")
tocsv <- data.frame(cbind(t(cmat1$overall),t(cmat1$byClass)))
if(varb==1){
  knnClass = tocsv
}
else{
  knnClass = rbind(knnClass,tocsv)
  }
 }
}
View(knnClass)
write.csv(knnClass,"KNN-78.csv",row.names = FALSE)
```

Output:

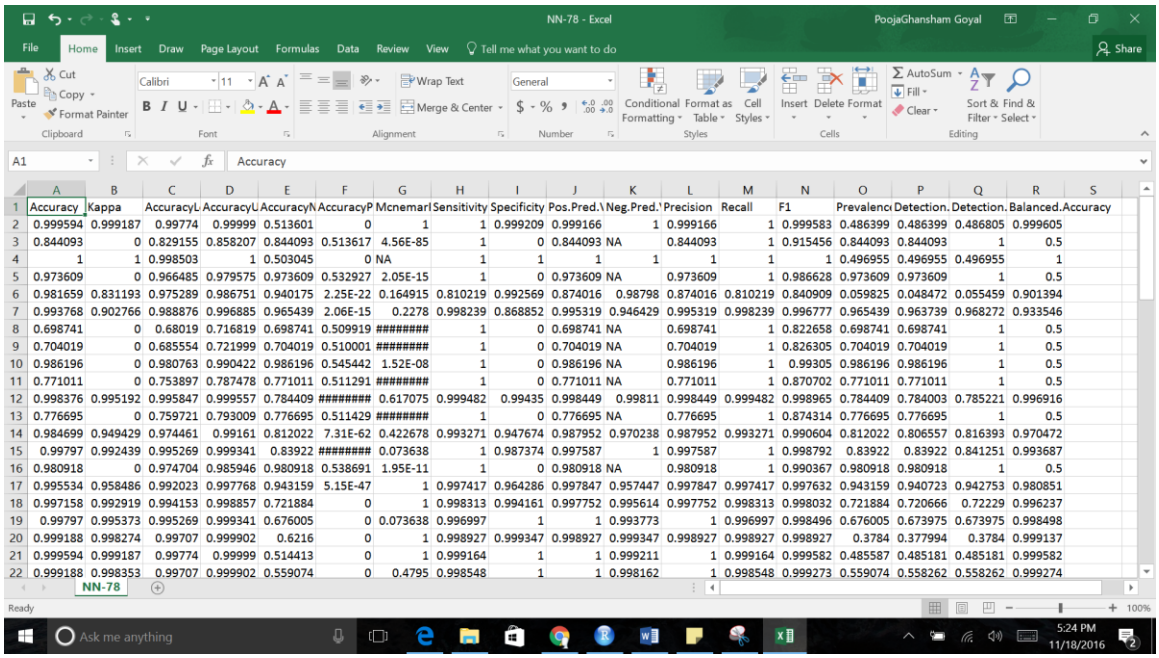


Accuracy	Kappa	AccuracyL	AccuracyU	AccuracyN	AccuracyP	Mcnemar	Sensitivity	Specificity	Pos.Pred.	Neg.Pred.	Precision	Recall	F1	Prevalenc	Detection	Detection	Balanced.Accuracy
0.745272	0.01336	0.73177	0.75843	0.784774	1	1.91E-71	0.926785	0.083426	0.786639	0.238095	0.786639	0.926785	0.85098	0.784774	0.727316	0.924587	0.505105327
0.759876	0.04625	0.746623	0.772764	0.784774	0.999949	4.72E-88	0.943258	0.091212	0.790995	0.30597	0.790995	0.943258	0.860442	0.784774	0.740244	0.935839	0.517235271
0.753651	0.027845	0.74029	0.766657	0.784774	0.999999	1.05E-82	0.937462	0.083426	0.788555	0.267857	0.788555	0.937462	0.856585	0.784774	0.735695	0.932966	0.510443948
0.765861	0.007108	0.752716	0.778633	0.784774	0.998491	########	0.964918	0.040044	0.785643	0.238411	0.785643	0.964918	0.866101	0.784774	0.757242	0.96385	0.502481063
0.699066	-0.02156	0.684901	0.712952	0.784774	1	4.97E-19	0.85662	0.124583	0.781085	0.19244	0.781085	0.85662	0.81711	0.784774	0.672253	0.860666	0.49060138
0.778549	0.040385	0.765645	0.791063	0.784774	0.840793	########	0.978646	0.048943	0.789564	0.385965	0.789564	0.978646	0.873995	0.784774	0.768015	0.972708	0.513794393
0.70984	-0.0136	0.695813	0.723572	0.784774	1	2.51E-27	0.872788	0.115684	0.782549	0.199616	0.782549	0.872788	0.825209	0.784774	0.684941	0.875269	0.494236189
0.785971	0.034316	0.773215	0.798325	0.784774	0.434102	########	0.993594	0.028921	0.78862	0.553191	0.78862	0.993594	0.87932	0.784774	0.779746	0.988748	0.511257339
0.728274	0.009757	0.714507	0.741721	0.784774	1	3.72E-43	0.897804	0.110122	0.786268	0.228111	0.786268	0.897804	0.838342	0.784774	0.704573	0.896098	0.503962948
0.759397	0.001068	0.746135	0.772294	0.784774	0.999962	########	0.955156	0.045606	0.784908	0.218085	0.784908	0.955156	0.861704	0.784774	0.749581	0.954992	0.500380906
0.770649	0.021756	0.757593	0.783325	0.784774	0.987	########	0.969494	0.045606	0.787413	0.29078	0.787413	0.969494	0.869018	0.784774	0.760833	0.966244	0.507549911
0.69739	-0.00811	0.683205	0.7113	0.784774	1	1.03E-14	0.849298	0.143493	0.783343	0.207063	0.783343	0.849298	0.814988	0.784774	0.666507	0.85085	0.496395561
0.785971	0.047212	0.773215	0.798325	0.784774	0.434102	########	0.990238	0.041157	0.790166	0.536232	0.790166	0.990238	0.87896	0.784774	0.777113	0.983481	0.515697395
0.767297	0.026706	0.754179	0.780041	0.784774	0.996968	########	0.962172	0.05673	0.788106	0.291429	0.788106	0.962172	0.866484	0.784774	0.755087	0.958104	0.509450878
0.776873	0.020975	0.763936	0.789422	0.784774	0.896017	########	0.980781	0.03337	0.787218	0.322581	0.787218	0.980781	0.873404	0.784774	0.769691	0.977735	0.507075688
0.760115	0.015751	0.746866	0.772999	0.784774	0.999941	########	0.95241	0.058954	0.786794	0.253589	0.786794	0.95241	0.861717	0.784774	0.747426	0.949964	0.5056822
0.775916	0.039754	0.76296	0.788484	0.784774	0.920489	########	0.97407	0.053393	0.789565	0.360902	0.789565	0.97407	0.872166	0.784774	0.764424	0.968159	0.513731107
0.746708	0.027145	0.73323	0.75984	0.784774	1	4.74E-68	0.925259	0.095662	0.788612	0.259819	0.788612	0.925259	0.851488	0.784774	0.726119	0.920757	0.510460575
0.751736	0.011779	0.738342	0.764777	0.784774	1	2.74E-86	0.938682	0.070078	0.786353	0.238636	0.786353	0.938682	0.855792	0.784774	0.736653	0.936797	0.504379994
0.739765	-0.03362	0.726175	0.75302	0.784774	1	2.15E-81	0.930445	0.044494	0.780251	0.149254	0.780251	0.930445	0.848755	0.784774	0.730189	0.935839	0.487469638
0.78238	0.011423	0.769551	0.794812	0.784774	0.654801	########	0.992984	0.014461	0.786042	0.361111	0.786042	0.992984	0.877477	0.784774	0.779267	0.991381	0.503722019

3)Neural Network:

```
setwd("C:/Users/goyal/Desktop/ADS/ads-midterm/Part2-Classification/Model2")

getwd()
library(ISLR)
library(leaps)
library(dplyr)
library(DAAG)
library(ROCR)
library(MASS)
library(nnet)
library(caret)
library(sqldf)
data = read.csv(file="Final_Finland_Data.csv",header = TRUE)
names(data)

que = sqldf("Select DISTINCT BuildingID,BuildingName,MeterID,type from data")
lengQuery = length(que$BuildingID)
#setwd("Buildings Data")
varb = 1
for (varb in 1:lengQuery){
  name = paste(que$BuildingName[varb],que$BuildingID[varb],que$MeterID[varb],que$type[varb],sep = "_")
  bdng = paste(name,".csv",sep="")
  data = read.csv(bdng,header = TRUE)
  #data
  data <- Filter(function(x)!all(is.na(x)), data)
  data <- Filter(function(x)!all(x=="N/A"), data)
  drops <- c("Date","PrecipitationIn","BuildingID","BuildingName","MeterID","type","nearestAirport","address","VisibilityMPH")
  data <- data[ , !(names(data) %in% drops)]


  data$Weekday=as.factor(data$Weekday)
  data$Base_Hour_Flag=as.factor(data$Base_Hour_Flag)
  data$Holiday=as.factor(data$Holiday)
  newVar = data$Base_Hour_Class
  length(unique(newVar))
  df<- data
  if(length(unique(newVar)) == 2){
    #splitting data
    #Take 70% of the data as the sample data
    smp_size<-floor(0.70*nrow(df))
    #Divide the data into train and test data. Sample data is basically train data
    train_ind<-sample(seq_len(nrow(df)),size=smp_size)
    train <- df[train_ind,]

    # Rest 30% is test data
    test <- df[-train_ind,]

    #createmodel
    nnetmodel <- nnet(Base_Hour_Class~ Consumption + Base_Hour_Flag + Holiday +
                        TemperatureF + Dew_PointF + Humidity + Sea_Level_PressureIn +
                        Wind_SpeedMPH + base_hr_usage,train, size=2,rang = 0.1,
                      decay = 5e-4, maxit = 350,MaxNWts = 5000)

    #Use predict() function to predict the probabilities of the outcome
    predicted<-predict(nnetmodel, test,type = "class")
    reference = test$Base_Hour_Class
    u = union(predicted, reference)
    t = table(factor(predicted, u), factor(reference, u))
    lc = confusionMatrix(t)


    tocsv <- data.frame(cbind(t(lc$overall),t(lc$byClass)))
    print(varb)
    if(varb==1){
      logResClass = tocsv
    }
    else{
      logResClass = rbind(logResClass,tocsv)
    }


  }
}
logResClass
```

Output:



3) Random Forest: Shashwat

2. Build Logistic Regression, KNN, Random Forest and Neural Network models for all buildings as one dataset. Optimize your model through feature engineering and model selection.

Solution:

Building each model using above approach and writing the output (Confusion Matrix) in one csv file

1) KNN Model

```
library(ISLR)
library(leaps)
library(dplyr)
library(DAAG)
library(ROCR)
library(MASS)
library(caret)

#reading data
data = read.csv(file="Final_Finland_Data.csv",header = TRUE)
names(data)
summary(data)
#removing na's
data <- Filter(function(x)!all(is.na(x)), data)
data <- Filter(function(x)!all(x=="N/A"), data)
na.omit(data)
drops <- c("Date","PrecipitationIn","BuildingID","BuildingName","MeterID","type","nearestAirport",
           "address","Gust_SpeedMPH","VisibilityMPH")
data <- data[ , !(names(data) %in% drops)]
View(data)
num_observation=nrow(data)

#changing data types
d<- as.data.frame(data)
d$Weekday=as.factor(d$Weekday)
d$Base_Hour_Flag=as.factor(d$Base_Hour_Flag)
d$Holiday=as.factor(d$Holiday)
View(d)
class(d$Wind_SpeedMPH)
View(d)
#--------------knn-------------------------#

kn=round(sqrt(num_observation))


#normalising data
d1<-d[,c(1,2,8:11,13,15,17,3:7,12,14,16,18)]
View(d1)
```

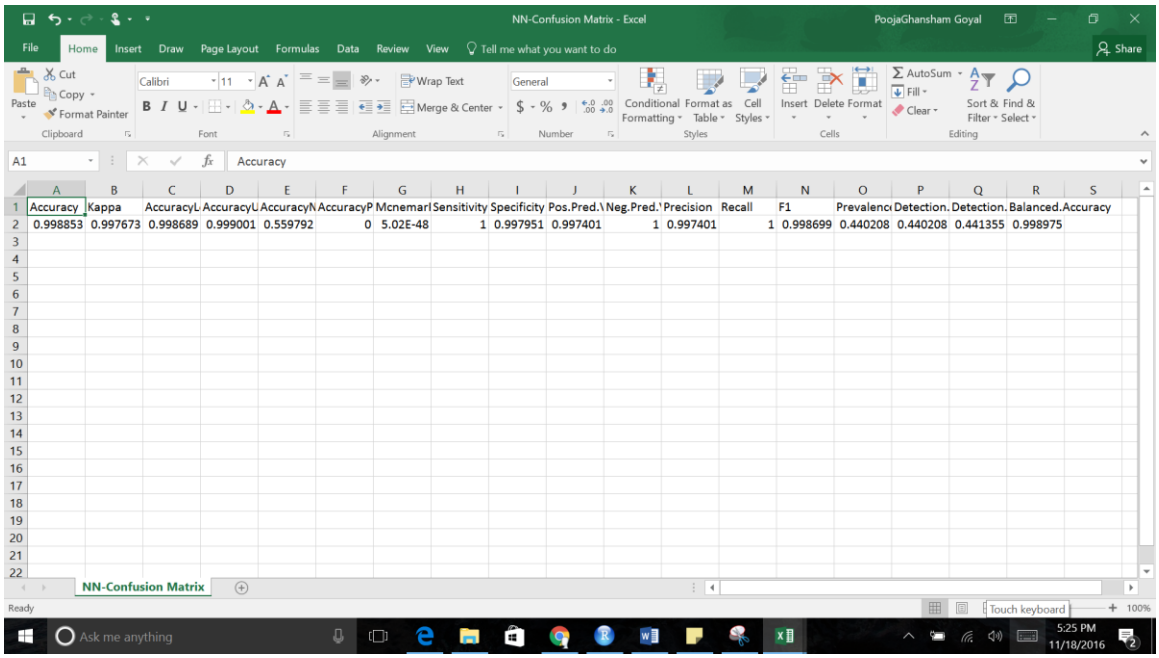
```
normalize <- function(x) {
  num <- x - min(x)
  denom <- max(x) - min(x)
  return (num/denom)
}

t1<- as.data.frame(lapply(d1[,c(1:9)], normalize))
View(t1)
d2<-cbind(t1,d1[10:18])
View(d2)
summary(t1)
str(d)
# split data

#Take 60% of the data as the sample data
smp_size<-floor(0.60*nrow(d2))

#Divide the data into train and test data. Sample data is basically train data
train_ind<-sample(seq_len(nrow(d2)),size=smp_size)
train <- d2[train_ind,]
View(train)

# Rest 40% is test data
test <- d2[-train_ind,]
View(test)

#creating model
ctrl <- trainControl(method="repeatedcv", number=10, repeats=3)
set.seed(12345)
knnFit1 <- train(Base_Hour_Class~., data=train, method="knn",
                 trControl=ctrl, tuneLength=10)
plot(knnFit1)

#prediction
knnPredict1 <- predict(knnFit1, newdata=d)


#ConfusionMatrix
cmat1 <- confusionMatrix(knnPredict1, data$Base_Hour_Class, positive="High")
cmat1
tocsv <- data.frame(cbind(t(cmat1$overall),t(cmat1$byClass)))
View(tocsv)
write.csv(tocsv,file="KNN Matrix.csv",append = TRUE)
```

Output: To be made

2)Neural Network

```
library(ISLR)
library(leaps)
library(dplyr)
library(DAAG)
library(ROCR)
library(MASS)
library(nnet)
library(caret)

data = read.csv(file="Final_Finland_Data.csv",header = TRUE)

names(data)

data <- Filter(function(x)!all(is.na(x)), data)
data <- Filter(function(x)!all(x=="N/A"), data)
drops <- c("Date","PrecipitationIn","BuildingID","BuildingName","MeterID","type","nearestAirport","address")
data <- data[ , !(names(data) %in% drops)]
View(data)
df<-data.frame(data)


#create train and test data

smp_size<-floor(0.60*nrow(df))
set.seed(123)
train_ind<-sample(seq_len(nrow(df)),size=smp_size)
train <- df[train_ind,]
test <- df[-train_ind,]

#createmodel
nnetmodel <- nnet(Base_Hour_Class~ Consumption + Base_Hour_Flag + Holiday +
                    TemperatureF + Dew_PointF + Humidity + Sea_Level_PressureIn + VisibilityMPH +
                    Wind_SpeedMPH + base_hr_usage,train, size=2,rang = 0.1,
                  decay = 5e-4, maxit = 350,MaxNWts = 5000)

#Use predict() function to predict the probabilities of the outcome
pr<-predict(nnetmodel, test,type = "class")

table(pr,test$Base_Hour_Class)


lc=confusionMatrix(test$Base_Hour_Class,pr)

#create ROC curve
require(ROCR)
prediction <- prediction(pr, test$Base_Hour_Class)
performance <- performance(prediction, measure = "tpr", x.measure = "fpr")
plot(performance, main="ROC curve", xlab="1-Specificity", ylab="Sensitivity")

tocsv <- data.frame(cbind(t(lc$overall),t(lc$byClass)))
View(tocsv)
write.csv(tocsv,file="NN-Confusion Matrix.csv",append = TRUE)
```

Output:

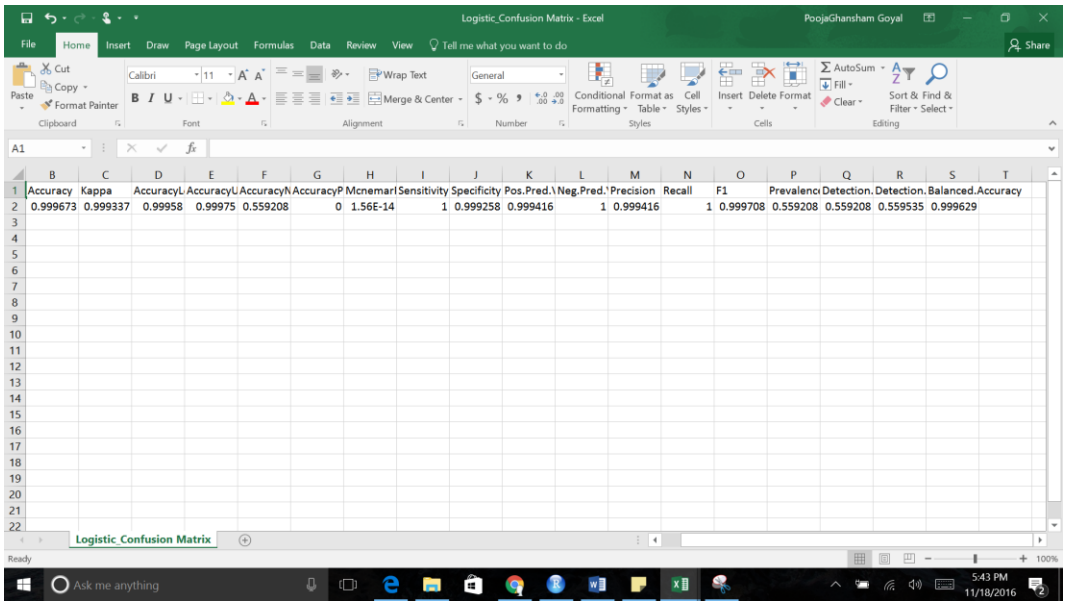


3)Logistic Regression

```
setwd("C:/Users/goyal/Desktop/ADS/ads-midterm/Part2-Classification/Model2")
install.packages("caret")
install.packages("ISLR")
install.packages("leaps")
install.packages("dplyr")
install.packages("DAAG")
install.packages("ROCR")
library(ISLR)
library(leaps)
library(dplyr)
library(DAAG)
library(ROCR)
library(MASS)
library(nnet)
library(caret)
library(mlbench)#feature selection

data = read.csv(file="Final_Finland_Data.csv",header = TRUE)
names(data)

View(data)
data <- Filter(function(x)!all(is.na(x)), data)
data <- Filter(function(x)!all(x=="N/A"), data)
drops <- c("Date","PrecipitationIn","BuildingID","BuildingName","MeterID","type","nearestAirport","address")
data <- data[ , !(names(data) %in% drops)]
View(data)

data$Weekday=as.factor(data$Weekday)
data$Base_Hour_Flag=as.factor(data$Base_Hour_Flag)
data$Holiday=as.factor(data$Holiday)
class(data$Holiday)
#feature selection (Identify highly correlated features in caret r packageR)

# ensure the results are repeatable
set.seed(7)
# prepare training scheme
control <- trainControl(method="repeatedcv", number=10, repeats=3)
# train the model
model <- train(Base_Hour_Class~., data, method="lvq", preProcess="scale", trControl=control)
# estimate variable importance
importance <- varImp(model, scale=FALSE)
```

```r
#training data set

df <-as.data.frame(data)
class(df$Holiday)
#Take 70% of the data as the sample data
smp_size<-floor(0.70*nrow(df))
#Divide the data into train and test data. Sample data is basically train data
train_ind<-sample(seq_len(nrow(df)),size=smp_size)
train <- df[train_ind,]

# Rest 30% is test data
test <- df[-train_ind,]

#logistic regression
lrmodel <- glm(Base_Hour_Class~.,family = binomial(link = "logit"), data = train)
summary(lrmodel)

#using step AIC
step <- stepAIC(lrmodel, direction="forward")
step$anova

#final model
lrmodel1<- glm(Base_Hour_Class ~ Hours+Consumption + day + month +
                 Weekday + Base_Hour_Flag + Holiday + TemperatureF + Dew_PointF +
                 Humidity + Sea_Level_PressureIn + VisibilityMPH + Wind_Direction +
                 Wind_SpeedMPH  + WindDirDegrees + base_hr_usage, family = binomial(link = "logit"), data = train)
summary(lrmodel1)

#Predict the outcome using predict() function
test.probs <- predict(lrmodel1, type = 'response',newdata = test)
table(test$Base_Hour_Class, test.probs > 0.5)

#Divide the lowpredicted values based on probability values
pred<- rep("Low",length(test.probs))
pred[test.probs<=0.5]<-"High"


#error table
logisticregression_table<-table(pred,test$Base_Hour_Class)

#confusion matrix
lc=confusionMatrix(test$Base_Hour_Class,pred)
tocsv <- data.frame(cbind(t(lc$overall),t(lc$byClass)))
View(tocsv)
write.csv(tocsv,file="Logistic_Confusion Matrix.csv",append = TRUE)

#create ROC curve
prediction <- prediction(test.probs, test$Base_Hour_Class)
performance <- performance(prediction, measure = "tpr", x.measure = "fpr")
plot(performance, main="ROC curve", xlab="1-Specificity", ylab="Sensitivity")

#Lift curve
perf <- performance(prediction,"lift","rpp")
plot(perf, main="lift curve")

write.csv(Prediction,file="Logistic_Confusion Matrix.csv",append = T, row.names = FALSE)
```

Output:



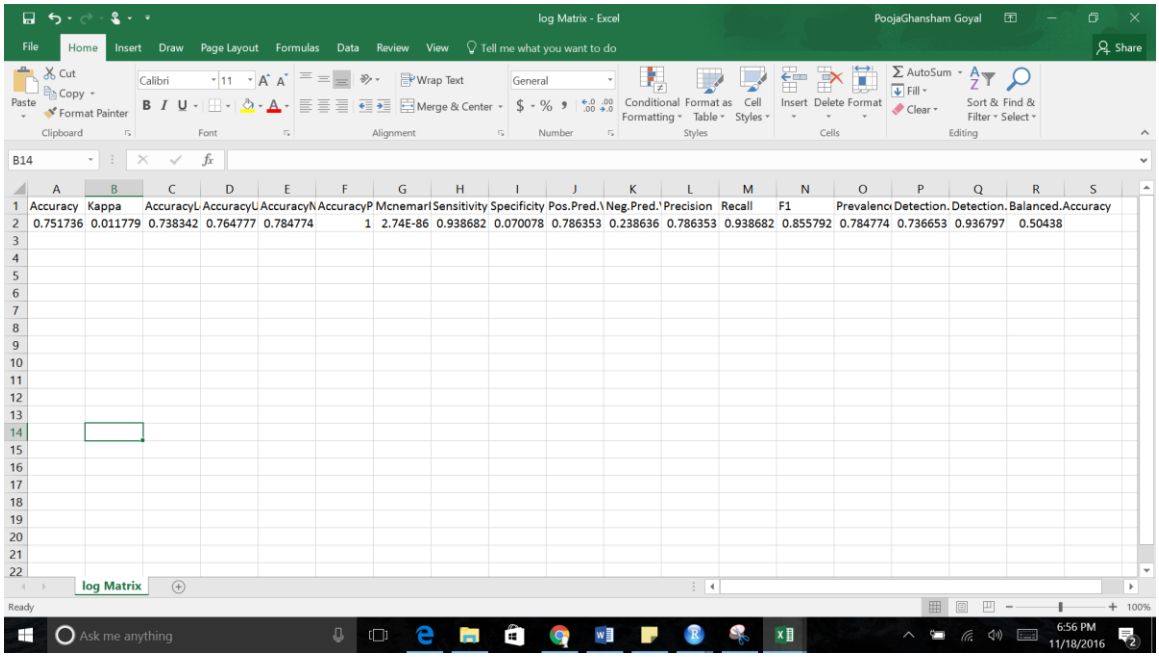
Accuracy	Kappa	AccuracyL	AccuracyU	AccuracyN	AccuracyP	Mcnemarl	Sensitivity	Specificity	Pos.Pred.\	Neg.Pred.'	Precision	Recall	F1	Prevalenc	Detection.	Detection.	Balanced.Accuracy
0.999673	0.999337	0.99958	0.99975	0.559208	0	1.56E-14	1	0.999258	0.999416	1	0.999416	1	0.999708	0.559208	0.559208	0.559535	0.999629

4)Random Forest

```
install.packages("randomForest")
install.packages("pROC")
library(randomForest)
library(caret)
library(ROCR)
# reading data
data = read.csv(file="Final_Finland_Data.csv",header = TRUE)
names(data)
View(data)

#filtering out na
data <- Filter(function(x)!all(is.na(x)), data)
data <- Filter(function(x)!all(x=="N/A"), data)
drops <- c("Date","PrecipitationIn","BuildingID","BuildingName","MeterID","type","nearestAirport","address")
data <- data[ , !(names(data) %in% drops)]
View(data)
data$Weekday=as.factor(data$Weekday)
data$Base_Hour_Flag=as.factor(data$Base_Hour_Flag)
data$Holiday=as.factor(data$Holiday)

#splitting data
df <- data
#Take 70% of the data as the sample data
smp_size<-floor(0.70*nrow(df))
#Divide the data into train and test data. Sample data is basically train data
train_ind<-sample(seq_len(nrow(df)),size=smp_size)
train <- df[train_ind,]
nrow(train)

# Rest 30% is test data
test <- df[-train_ind,]
nrow(test)

# Fitting model
fit <- randomForest(Base_Hour_Class ~.,train,ntree=100,importance=T,na.action = na.omit)
summary(fit)
plot(fit)
```

```
# Variable Importance Plot
varImpPlot(fit,
           sort = T,
           main="Variable Importance",
           n.var=10)

fit2 <- randomForest(Base_Hour_Class ~Hours + Base_Hour_Flag + Holiday +
                       TemperatureF + Dew_PointF + Humidity + Sea_Level_PressureIn +
                       VisibilityMPH + Wind_SpeedMPH + base_hr_usage,train,na.action = na.omit)
summary(fit2)
plot(fit2)

#predict
test.probs <- predict(fit2, test)
str(test.probs)

# confusion matrix
tab=confusionMatrix(test$Base_Hour_Class,test.probs)
str(tab)
tocsv <- data.frame(cbind(t(tab$overall),t(tab$byClass)))
View(tocsv)
write.csv(tocsv,file="Log Matrix.csv",append = TRUE)

#calculate error percentage
Error_Random <- (((tab [1,2]) + (tab [2,1])) /(( tab [2,1]) + (tab [1,2]) + ( tab[1,1])+( tab [2,2])))
Error_Random *100

#create ROC curve
prediction <- prediction(test.probs, test$Base_Hour_Class)
performance <- performance(prediction, measure = "tpr", x.measure = "fpr")
plot(performance, main="ROC curve", xlab="1-Specificity", ylab="Sensitivity")
```

Output:



Accuracy	Kappa	AccuracyL	AccuracyU	AccuracyN	AccuracyP	Mcnemarl	Sensitivity	Specificity	Pos.Pred.\	Neg.Pred.\	Precision	Recall	F1	Prevalenc	Detection.	Detection.	Balanced.Accuracy
0.751736	0.011779	0.738342	0.764777	0.784774	1	2.74E-86	0.938682	0.070078	0.786353	0.238636	0.786353	0.938682	0.855792	0.784774	0.736653	0.936797	0.50438

Analysis of Models in Part 1

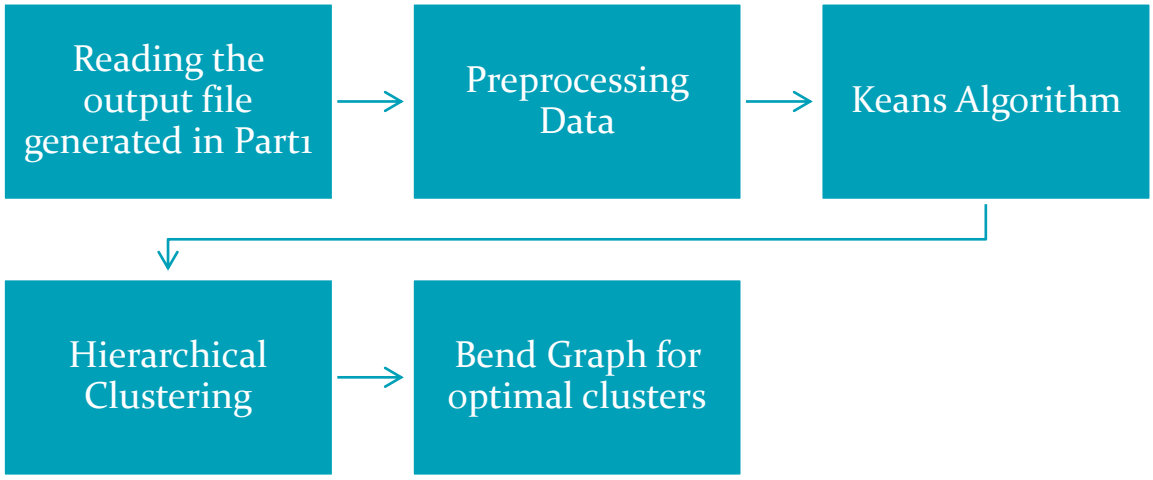
Models	Accuracy	Findings
Logistic Regression	0.9985 ~1	
KNN	0.6711	
Random Forest		
Neural Network	0.9997	

Analysis of Models in Part 2

Models	Accuracy	Findings
Logistic Regression	0.998853	
KNN	0.751736	
Random Forest	0.751736	
Neural Network	0.999673	

Clustering:

Using the building features, cluster the buildings using K-means and Hierarchical clustering. Using Bend graphs, choose the optimal number of clusters. Discuss your cluster features in a report



Steps:

1)Code to read the input file

```
data = read.csv(file="Final_Finland_Data.csv",header = TRUE)
```

2) Processing data to perform clustering on quantitative features

```
#processing data
data <- Filter(function(x)!all(is.na(x)), data)
data <- Filter(function(x)!all(x=="N/A"), data)
drops <- c("Date","PrecipitationIn","BuildingID","BuildingName","MeterID","type","nearestAirport",
           "address","VisibilityMPH","Gust_speedMPH")
data <- data[ , !(names(data) %in% "Gust_SpeedMPH")]
data <- data[ , !(names(data) %in% drops)]

data$Weekday=as.factor(data$Weekday)
data$Base_Hour_Flag=as.factor(data$Base_Hour_Flag)
data$Holiday=as.factor(data$Holiday)
```

3) Storing quantitative features a data-frame

```
nums<- sapply(data, is.numeric)
df<- data[,nums]
```

4) Performing k-means clustering with 5 clusters

```
#K-means with 5 clusters
k<-kmeans(df,5)
```

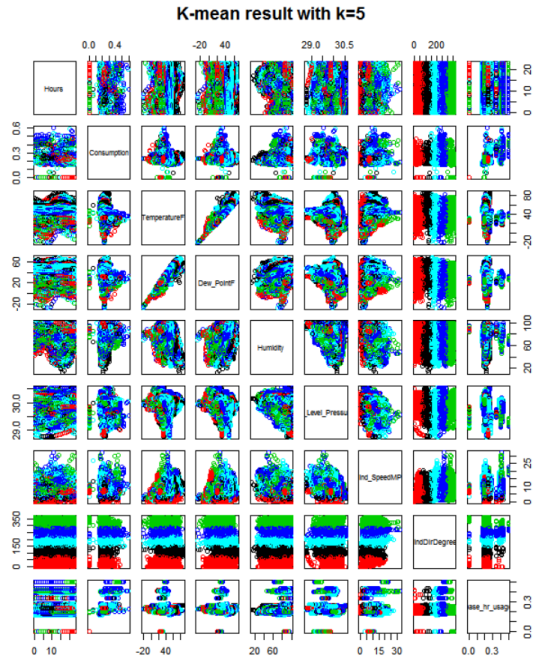
5) Result of k means

```
# kmeans results
k$cluster
```

```
K-means clustering with 5 clusters of sizes 92942, 85448, 157
919, 221557, 63951

Within cluster sum of squares by cluster:
[1] 123825633  96174505 288521225 303597639  79792301
 (between_SS / total_SS =  88.3 %)

Available components:

[1] "cluster"      "centers"      "totss"
[4] "withinss"     "tot.withinss" "betweenss"
```

Plotting cluster:

```
# kmeans results
plot(df, col=(k$cluster), main="K-mean result with k=5") #Scatterplot matrix
```



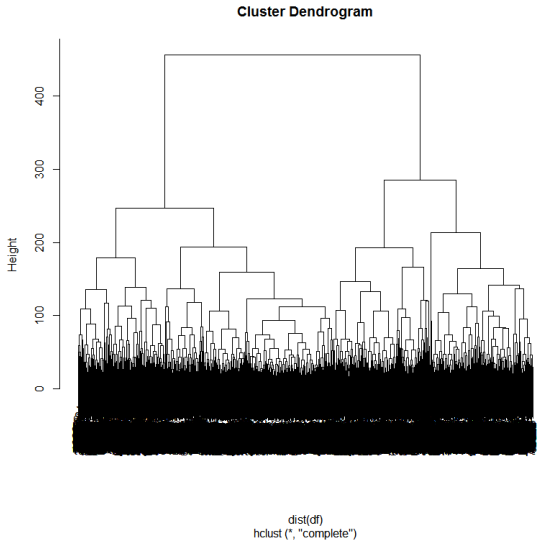
6) Creating hierarchical clustering

```
#hierarichal clustering
clusters<-hclust(dist(df))
plot(clusters)
```

Output: Unable to plot graph as the data is huge and throws a allocation memory size full error

```
> clusters<-hclust(dist(df))
Error: cannot allocate vector of size 1440.4 Gb
In addition: Warning messages:
1: In dist(df) :
  Reached total allocation of 16237Mb: see help(memory.size)
2: In dist(df) :
```

Alternative- used a subset of output generated in partı to create dendograms. By using subset data and following steps below is the dendogram formed



7) Using bend graph to find the optimal number of cluster

```
mydata <- df
wss <- (nrow(mydata)-1)*sum(apply(mydata,2,var))
for (i in 2:15) wss[i] <- sum(kmeans(mydata,
                                     centers=i)$withinss)

plot(1:15, wss, type="b", xlab="Number of Clusters",
     ylab="Within groups sum of squares")
```